HARDWARE IN THE LOOP FOR HYDRAULIC CONTROL OF MINING MACHINES

Diploma Thesis

TAN WEN

University of Leoben
Institute for Automation
Leoben, Austria

Supervisors:

O.Univ.-Prof. Dipl.-Ing. Dr.techn. Paul O'Leary
University of Leoben, Austria
Ass.Prof. Dipl.-Ing. Dr.mont. Gerhard Rath
University of Leoben, Austria

May 2007

TAN WEN

University of Leoben
Peter-Tunner-Straße 27

A-8700 Leoben, Austria

Internet: http://automation.unileoben.ac.at/
E-Mail: automation@unileoben.ac.at
TAN.WEN@unileoben.ac.at
Tel.: ++43 (0)3842 402 1234
Fax.: ++43 (0)3842 402 1234

This thesis was typeset using LaTeX, Times, 12 pt, one side, 600 dpi.
TeX is a trademark of the American Mathematical Society.

Abstract

The development of many modern products and processes is characterized by integration of digital control systems. Computer-aided methods for modeling, simulation and design are increasingly required; the reason are the increasing complexity and inter-relationship between the design of the processes and control systems.
The aim of this diploma thesis was a Real- Time Simulation, in this case a Hardware in the Loop Simulation for hydraulic control of mining machines and visualisation of these simulation models.
The developed system can be used in the implementation phase to emulate the behavior of the machines as a vehicle to test the control soft- and hardware.
The simulation was designed with two different function-block oriented, graphical programming environments. After testing, a system with Matlab/Simulink and Real-Time Workshop implemented on a programmable logic control (PLC) was chosen.
A complete real-time model was developed that is capable of simulating the essential properties of the real system (kinematics, dynamic behavior of the hydraulic system and the mechanical construction, further the cutting load force). The visualisation of the model is performed by displaying selected images from a rendered simulation video.
The final system will be applied to test the automation software with the aim of improving the cutting accuracy, before the machine prototype is available.

Kurzfassung

Die Entwicklung vieler moderner Produkte und die dafür notwendigen Prozesse erfolgt unter Einbindung von digitalen Automatisierungssystemen. Computergestütze Methoden zur Modellbildung und Simulation werden immer häufiger eingesetzt auf Grund des hohen Grades an Komplexität und Wechselwirkungen der Systeme miteinander.
Das Ziel der vorliegenden Arbeit war eine Real-Time Simulation, in diesen Fall Hardware in the loop Simulation, der hydraulischen Schneidarmsteuerung einer Bergbaumaschine und die Visualisierung eines durch die Simulation erhaltenen Modells.
Das fertige System kann dazu verwendet werden, in der Inbetriebnahmephase zum Testen der Steuerungssoftware und -hardware die Funktionen der Maschine zu ersetzen.
Die Simulation wurde mit zwei verschiedenen funktionsblockorientierten, graphischen Programmiersystemen erstellt. Nach Auswertung der Tests wurde entschieden, ein System mit Matlab/Simulink und Real-Time Workshop auf einer speicherprogammierbaren Steuerung (SPS) zu implementieren.
Es wurde ein komplettes, echtzeitfähiges Modell entwickelt, das die wesentlichen Eigenschaften des realen Systems abbildet (Kinematik, Dynamik des hydraulischen und mechanischen Systems, sowie der Schneidlast). Die Visualisierung des Modells erfolgt durch dynamische Auswahl eines Bildes aus einer gerenderten Video-Simulation.
Das fertige System wird eingesetzt werden, um die Automatisierungssoftware für bessere Schneidgenauigkeit zu entwickeln und zu testen, bevor der Test am mechanischen Prototypen erfolgt.

Keywords

Real Time, Hardware in the loop, Simulation.

Acknowledgements

- I express my deep gratitude to my supervisor O.Univ.-Prof. Dipl.-Ing. Dr.techn. Paul O'Leary that give me the chance to join the Institute and write my diploma thesis there.
- I would like to thank my co-supervisor Prof. Dipl.-Ing. Dr.mont. Gerhard Rath for his continuous help and support. I have to admit that I learn very much from him before and during the diploma work.
- I would like to thank the Sandvik Mining and Construction that give me the chance to write the Diploma thesis for them.
- I am grateful for the support, help and advice from the colleagues at the institute for automation. And many thinks to Frau Doris Widek for her support and help.
- My deepest thanks to my parents Guangming Tan and Lizhen Xu for their patience, support and understanding. And thanks them to give me the chance to study in Austria.
- I would like also to think all my friends in Leoben for their help.
- Last but not the least, I would like to thank my friend Christoph Piribauer, his parents and his family for the understanding and support, and the wonderful life in Austria.

Declaration

I declare that I, myself, composed this thesis and that the work contained therein is my own, except where stated.

TAN WEN
Leoben, May, 2007

Contents

Chapter 1

Theoretical Background

1.1 Real-time

1.1.1 Real-Time System

1.1.1.1 Definition

The system is real-time when the system can guarantee that an event will occur precisely after a specified time period. It does not matter at all how big the time slice is. We use this to differ it from the Model time which is the self-governing simulation runtime (in German: Laufzeit) from the software. We speak of Real-time capable systems when the model time is synchronized to the real time. In computer science, Real-time computing, is used to study hardware and software systems which are subject to a 'real-time constraint'.
A real-time computer system is a system with which the correct operation of this system is not only depend on logical results of the computation, but also on the physical instant which these results are produced.
Real-time system are needed for applications that must determistically perform a critical task without interruption from the other noncritical tasks.

1.1.1.2 Classification of Real-Time System

In following we classify the real-time from five different perspectives. The first two classifications from the point of view of the characteristics of the application are: at first hard real-time vs. soft real-time, further fail-safe and fail-operational, i.e., on factor outside the computer system. The other three classifications, guaranteed-timeliness and best-effort, resource-adequate and resource-inadequate, event-triggered and time-triggered, are from the point of view of design and implementation, i.e., on factors inside the computer system. [15]

1. Hard Real-Time System versus Soft Real-Time System A hard real-time (immediate real-time) system it is important that it perform at the correct instant, the completion of an operation after its deadline is considered useless. A soft real-time (on-line) system will tolerate such lateness. In Table 1.1 shown some different between hare-time and soft real-time [15]

characteristic	hard real-time	soft real-time
response time	hard required	soft required
peak-load performance	predictable	degraded
control of pace	environment	computer
safety	often critical	non-critical
size of data files	small or medium	large
redundancy type	active	checkpoint-recovery
data integrity	short-term	long-term
error detection	automation	user assisted

Table 1.1: hard real-time versus soft real-time

2. Fail-Safe versus Fail-Operational

   If a safe state can be identified and reached upon the occurrence of a failure, then we call the system fail-safe. Fail-safe is a characteristic of the controlled object, not the computer system.
   If a computer system must provide a minimal level of service to avoid a catastrophe even in the case of a failure, then we call it Fail-operational.

3. Guaranteed-Response versus Best-Effort

   If we start out with a specified fault- and load-hypothesis and deliver a design that makes it possible to reason about the adequacy of the design without reference to probabilistic arguments, even in the case of a peak load and fault scenario, we say this system is a guaranteed response. And these systems required careful planning and extensive analysis during the design phase.
   We speak of a best-effort, when the analytic response guaranteed cannot given.

4. Resource-Adequate versus Resource-Inadequate

   Guaranteed response systems are based on the principle of resource adequacy, i.e., there are enough computing resources available to handle the specified peak load and the fault scenario.

5. Event-Triggered versus Time-Triggered

   In the event-triggered approach, all communication and processing activities are initiated whenever a significant change of state is noted.
   In the time-triggered approach, all communication and processing activities are initiated at the predetermined points in time.

1.1.1.3 Application

Today more and more processors are used to control the technical facilities or processes such as machines, process plan and transportation are real-time systems. The reaction of these systems does not depend on the technical processes –neither at normal nor at critical situation:

- In cars, the Airbag-control system is a real-time system. It muss work up the mess of sensors and then decide if it should open the airbag and how strong it should be in the shortest time. The reaction time is about 1ms;
- Anti lock brake system [ABS] is also a real-time system. The reaction time of it is under 1ms;
- The Patriot-rocket demand most for real-time system. The time limit should be hold in nanosecond range, because of the high velocity (more than 1000m/s) and small hit radius (should be less than 1 meter). So the reaction time is less than 1 ns;

1.1.2 Real-Time Simulation

1.1.2.1 Why Simulation

Simulation is an imitation of some real thing, state of affairs, or process.
Simulation is increasingly used in many fields, particular in the range of engineering, and it can be used to show the eventual real effects of alternative conditions and courses of action.
There are some reasons why we apply it:

- It is maybe too complex and too expensive to research a real system, and it's maybe also too dangerous to do it. Example:
  - Crash test
  - Flight simulator to train the pilots.
- The real system does not exist till now.
- It's not possible directly to observe the real system. For example, the real system work too fast or too slow.
- It's much more easy to modify a simulation model than the real system.
- The result is exactly repeatable.
- By way of comparison, it is safer and cheaper.

1.1.2.2 Real-time Simulation

The development of many modern products and processes is characterised by integration of digital control systems. The integration can be performed by the components (hardware) and/or by the functions (software). Now computer-aided methods for modeling, simulation and design are more and more required, because of the increasing complexity and inter-relationship between the design of the processes and the design of the control system.
With regard to the speed of the computation required, simulation methods can be divide into [14]

i. Simulations without time limitations
some application:

- Basic investigation of behavior
- Verification of theoretical models
- Process design
- Control-system design

ii. Real-time simulations
some application:

- Process simulation:
- hardware in the loop simulation
- training of operators
- Controller simulation:
- testing of control algorithms (controller prototyping)
- Process and controller simulation

iii. Simulation which are faster than real-time

some application:

- Model-based control systems
- Predictive control
- Adaptive control
- On-line optimization
- Development of strategies,planning,scheduling
- Components for real-time simulation

Here, real-time simulation means that the simulation of a component is performed such that the input and output signals show the same time - dependent values as the real, dynamically operating component. There are three different kinds of real-time simulation methods:

1. Real process but simulated control system
   (control design control prototyping)
2. Simulated process and simulated control system
   (control design software in the loop)
3. Simulated process but real control system
   (hardware in the loop) discussed in section 1.2 in detail.

1.2 Hardware in the Loop

1.2.1 Embedded System

Embedded system (called ES for short) is a special purpose digital system embedded in the environment. Due to the rapid development of digital-processor technology, ES is used in many things in our daily life, for example: big things such as engine controllers in car, washing machine, small things such as mp3, PDA. And it is also used in many control system in industries. Many of these systems operate in safety-critical situations and in most cases are very complex such as land vehicles, satellite control system, spacecrafts,aircrafts, nuclear reactor controller. This is one of the important reason why so much research effort is put into the development of methodologies for the design, development, implementation and testing of embedded control system. Another reason is the demand for minimizing the time to the market.
The Embedded system consists normally of four parts: embedded micro processor, peripheral hardware devices, embedded operating system, and application program.
Compared with the general-purpose computer, ES have the following character:
An embedded system performs one or a few pre-defined tasks, usually with special requirements. Because of this, the system is smaller, high integrity and have low power dissipation, high mobility. Hardware and software of Embedded system are design to be high efficient, in as little as possible silicon wafer realize as much as possible the performance. In order to advance performance speed and dependability of system,the software is normally fix in memory chip or single board computer, not like general computer is in the carrier such as disks. The ES is also not able to develop itself.
A lot of conventional mechanical or electronic control system within many products are replaced by an embedded real-time system. The embedded real-time system has the following characteristics [15]:

i. Mass Production

ii. Static Structure

iii. Verification of theoretical models

iv. Process design

v. Control-system design

History of Development

Looking back, there are four important phases during the development of embedded system. First phase, embedded system is implemented in a single board computer without an operating system the given function of the conventional control system. So it uses assembly language, after stopping the memory will be immediately ridded to minimize the resource requirements.
In the second phase MPU, I/O devices, and RAM, ROM that are part of embedded system are integrated in VLSI. And the products functionality is augmented adding software function to improve the the utility of the intelligent products.
The third phase requires a redesign of the software. So the software designer needs to introduce a software architecture and an operating systems.
Fourth phase, the embedded system is already part of a larger system that needs to communicate with its environment.

Design Tools for Embedded Control System

The software tools can be classified in [25]:

- Real-time operating system:
  For this kind of system the correct computation depends both on the results and on the time when these results are available.
- Programming languages:
  We need programming languages that are real-time friendly to ensure the use of high-level constructs like information hiding and strong typing, but which at the same time are efficient and resource-aware [6]. The following three are part of this kind of programming language: Procedural Languages, functional and Logical Languages, Synchronous languages:
- Middleware frameworks and design patterns:
  The framework methodology does not widely apply to real-time system. In contrast to this the design pattern approach is successfully applied.

- Automatic code generation:
  This kind of tools are embedded in control design software tools - like Matlab, which can generate source/object code for different target control platforms.
- Verification tools:
  There are two major verification methods, one is based on theorem provers, the other is based on model checkers.
- Software analysis
- Testing tools
- Real-time supervisors, redundant and backup system.

1.2.2 Hardware in the Loop

One of the best definition of an Hardware in the loop (HIL) is given by Isermann [15]:
The simulated process can be operated with the real hardware.
The simulated process replaces either fully or partially the controlled process consisting of actuator, physical process and sensors.
The purpose of HIL simulation is to develop and test the complex real-time embedded system.
Hardware in the loop simulation is a kind of real-time simulation in which the input and output signal show the same time values as the real process. And it is characterized by the operation of real components in connection with real-time simulated components.
The implementation strategy of Hardware in the loop is dependent on the project and resource. Before anything is decided, we need to know the categorization of HIL which is shown in table 1.2 [15]. And from the table we notice that not all the combination are possible. Figure 1.1 shows all variations of the configurations.

Case	Actuators		Process		Sensors	
	Real	Sim.	Real	Sim.	Real	Sim.
1	yes		yes			yes
2	yes		yes		yes	
3	yes			yes		yes
4	yes			yes	yes	
5		yes	yes			yes
6		yes	yes		yes	
7		yes		yes		yes
8		yes		yes	yes	

Table 1.2: HIL categories by [15]

Configurations like 3,7 and 8 are impossible, because a simulated actuator which delivers a physical output is a real actuator of the system, and similarly a simulated sensor which reads physical inputs is a real sensor. Configuration 4 is the real working system in which nothing is simulated and 5 is just the opposite to 4, it is a fully simulated system. This kind of

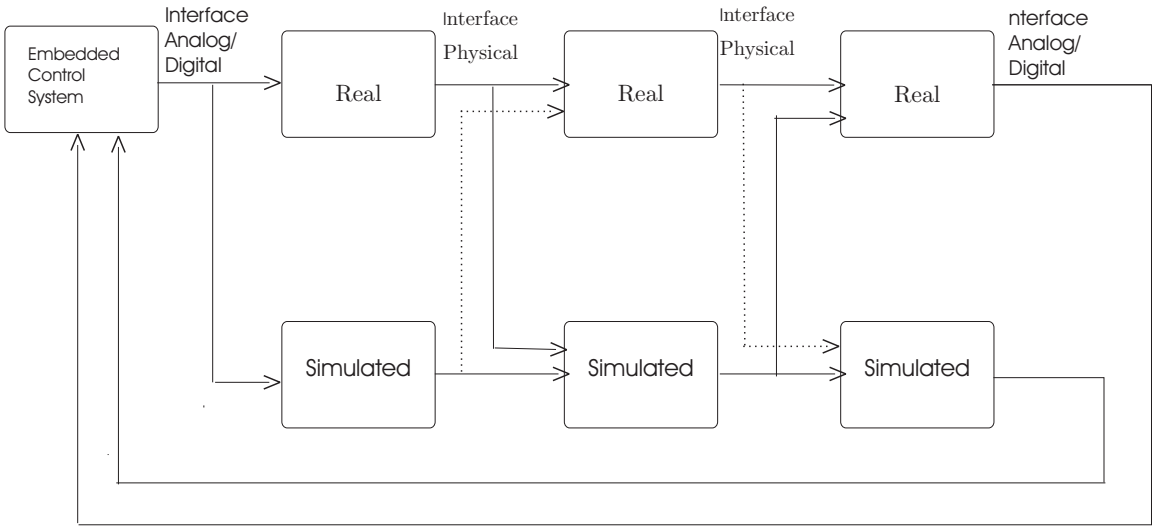

Figure 1.1: Architectures of HIL

configuration is called Full Simulation. It is a Partial Simulation when a real component is in the simulation loop.

The reason why we do not connect the embedded system under test (SUT) to the real plan are the follows:

- Duration: Tight development schedules are needed in most of development ranges, so it does not allow to SUT to wait for prototype.
- Cost: in most cases, the plan is more expensive than the real-time simulator. And it needs not to operate real process.
- Safety: can test SUT under extreme conditions.

In some cases it is not possible to build one prototype.

The SUT gives outputs to the process here which is replaced by simulator and the simulator give the sensor date back to the SUT.In Figure 1.2 [26] shown the simulation environment and the graphical presentation of the connection between SUT and HIL simulator. Actuator, process and sensor constitute the controlled process which is simulated fully or partially. Normally, actuator is real, process and the sensor are simulated. Because actuator and the control hardware are formed in one integrated subsystem.

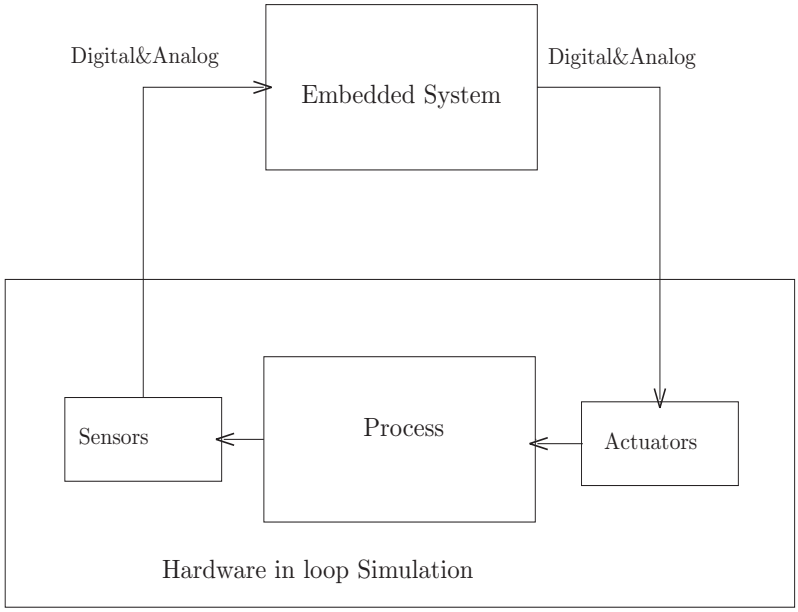

Figure 1.2: HIL simulation

1.2.3 HIL Simulation Hardware and Software

Computer Hardware

The hardware for HIL simulation needs [16]:

- A computer system that is able to perform the real-time requirements of the simulation
- Device on the computer which allows operator control of the simulation, simulation data collection, storage, analysis and display
- Set of I/O interface between computer and SUT

I/O device

There are many different categories of I/O used in embedded system. I/O interfaces are useable from several sources that support signal types like:

- Analog
- Discrete digital
- Serial
- Real-time data bus

- Network

In opposition to a SUT with low I/O rate, for high I/O rate SUT, we need a performance simulation computer system with following requirement [16]:

- Support for multiple high performance CPUs
- Support for real-time operations
- High data transfer rates
- Support for a variety of I/O devices

Simulation Software

The HIL simulation software has three basic parts [16]:

- Initialize the simulation software and hardware
- A dynamic loop including I/O, simulation model evaluation, date variable integration
- Shutdown of the simulation software and hardware

1.2.4 Implementation Tools of HIL

There are several combinations of software and hardware can realize the HIL simulation.In the following chapter, some of these combinations will be introduced.

1. First possible combination [13]:

   Hardware: real-time hardware of National Instruments
   Software: LabView real-time Module of National Instruments

   National Instruments offers integrated software and hardware products to create real-time applications. Application will be developed in Windows with LabView Graphical development and performed the applications on a real-time hardware target.

   There are many real-time hardware target of National Instruments:

   - Compact FieldPoint Real-Time Controllers (later will be described in detail): a rugged real-time platform for industrial control and automation consisting of a real-time processor with various I/O options and integrated signal conditioning.

- CompactRIO Real-Time Controllers: integrate a real-time processor, FPGA technology, and I/O modules in an extremely rugged package. It is ideal to fill requirements of high speed control, utilize custom digital protocols, and monitor system in harsh environment.

Figure 1.3: CompactRIO Real-Time Controllers[1]

- Desktop PCs: Here is Desktop or industry computer the real-time target and support PCI hardware integration with NI data acquisition devices.
- PXI Real-Time Controllers: is a high-performance platform for test and control applications, including compatibility for synchronization and triggering and a wide array of I/O, shown in Figure 1.3

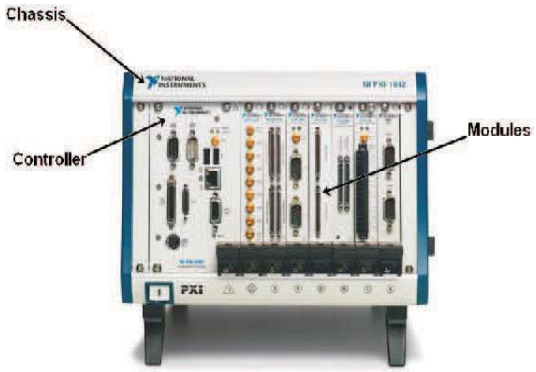

Figure 1.4: PXI Real-Time Controllers[2]

- Compact Vision Systems: is a platform for automated machine vision and inspection systems with IEEE 1394 compatibility for up to three cameras, shown in Figure 1.5

[1] www.ni.com
[2] www.ni.com
[3] www.ni.com

Figure 1.5: Compact Vision Systems[3]

Compact FieldPoint systems consist of a controller with an embedded processor running an RTOS and a variety of I/O modules. These systems feature rugged hardware, designed to operate in industrial environments. In addition the software structure of Compact FieldPoint systems implements a built-in publish/subscribe protocol, making these ideal for creating distributed applications, shown in Figure 1.6 The reason why this hardware is exactly described is that it will be applied in the following project.
Compact FieldPoint is an easy-to-use, highly expandable programmable automation controller composed of rugged I/O modules and intelligent communication interfaces. Download LabView application to the embedded controller for reliable, stand-alone operation and connect the sensors directly to high-accuracy analog and discrete I/O modules. The Compact FieldPoint I/O modules filter, calibrate, and scale raw sensor signals to engineering units and perform self-diagnostics to look for problems, such as an open thermocouple. Compact FieldPoint network communication interfaces automatically publish measurements with an Ethernet network. Access I/O points nearby or miles away on the network using the same simple read/write software framework. Connect virtually any sensor type with the wide variety of I/O modules. The most common sensor types include thermocouples, RTDs, strain gauges, 4 to 20 mA sensors, and a variety of digital signals from 5 to 30 VDC and 0 to 250 VAC.
With Compact FieldPoint and FieldPoint systems, powerful control and measurement systems using LabView Real-Time can be fast developed and easily embedded into an application on the intelligent controllers for reliable distributed or stand-alone deployment.

Figure 1.6: Compact FieldPoint[4]

[4]www.ni.com

Like what we referred to before, there are many products in the family of Compact FieldPoint like controller, I/O modules, some of them we will need later for our project. In following most of them will be introduced.

- Controller Interfaces:
  for building embedded process control, distributed I/O, and data-logging intelligent controls running LabView real-time.
- Network Interfaces: for adding Ethernet or Serial expansion I/O to any PAC or PC with Compact FieldPoint network interfaces.
- I/O Modules: for connecting to industrial I/O, sensors, actuators, and other industrial devices with analog, digital I/O, specialty, and combo modules.
- Backplanes: for setting Compact FieldPoint modules in rugged environments with solid metal backplanes.
- Connectivity accessories: for connecting signals with integrated or DIN-rail-mounted connector blocks, cables for serial ports, and covering for unused slots.
- Power Supplies and Power Distribution: for regulating, filtering, and distributing power with DC supplies.
- Mounting: for choosing from panel-mounting, DIN-rail-mounting, and 19 in. rack-mounting options.
- CompactFlash: for storing your downloaded application or performing embedded data logging with CompactFlash cartridges, which contain solid-state nonvolatile memory.
- FieldPoint: for creating Distributed I/O systems with FieldPoint Real-Time Controllers or Ethernet/Serial network interfaces.

We will not need them all, the combination we need for such products is controller interface, I/O modules and connectivity accessory. How they are set up is showed in Figure 1.7

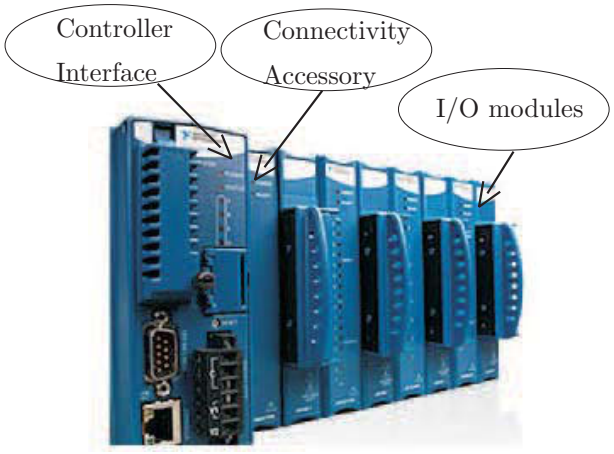

Figure 1.7: Compact FieldPoint[5]

[5] www.ni.com

Before we talk much about hardware, now let us talk about the software.
The LabView real-time module extends the LabView development environment to deliver deterministic, real-time performance. Use graphical programming or simulation tools to develop the applications and download the applications to run on an independent hardware target with real-time operating system (RTOS). The real-time modules can provide commercial off-the-shelf hardware for the real-time target, so it is not necessary to spend time verifying that the OS or code works on the hardware.
Except the real-time module, we will also need some other software function of LabView. The real-time module can let the system to be real-time capable. If we use simulation tools or Simulink software to write a program, the LabView Simulation Interface Toolkit is to be interface with the simulation environment. In addition, the toolkit adds patented user interface tools for monitoring and controlling data in the Simulink environment. And it can perform off line simulations on a desktop or download the dynamic model to a real-time system for HIL testing. The LabView Simulation Interface Toolkit brings the amount of user interface functionality of LabView to instrument simulation models.

2. The second way of possible combination [3] [19]:

   Hardware: The innovative industrial PC Automation PC 620 or Control System2003 from Bernecker & Rainer company (B & R for short)
   Software: Real-Time Workshop (from The MathWorks company)

   We will first talk about Control System 2003. The B & R System 2003 can be used as both a complete control system as well as a remote I/O system for the expansion of industrial PCs and controllers from all B & R system families. Distributed systems can also be created. Many different interfaces for field bus systems and networks guarantee trouble-free communication.
   CPUs of system 2003, shown in Figure 1.8. System 2003 central processing units cover a wide performance spectrum. The optimal price/performance ratio is achieved by fine-tuning processing power, memory capacity, integrated communication interfaces, and local slots for I/O screw-in modules. Clearly arranged diagnostic LEDs have been implemented to indicate the controller's status. Programming is achieved in a uniform manner using B & R Automation Studio.

   I/O Modules, shown in Figure 1.9 for the System 2003, B & R offers a large number of I/O modules in various designs. Analog values, digital signals, timers and counters allow many process variables to be handled and various actuators to be controlled.

   The new APC620, shown in Figure 1.10, relies on experience collected over many years of industrial PC development and many applications. The result: the APC620 - providing optimal adaptability and ergonomics. The mechanical design is based on the results of extensive shock and vibration tests that place the highest demands on the materials. The APC620's main advantages are its modular design, the flexibility of the slots and the well thought out arrangement of interfaces and drives.
   The display units have also been updated with new technology. Modular interfaces allow adjustments to be made to meet various requirements.

[6] ©B&R 2007, courtesy of B&R
[7] ©B&R 2007, courtesy of B & R company

Figure 1.8: CUP of System 2003[6]

Figure 1.9: Digital Mixed Modules of System2003[7]

The APC620 with Intel Pentium M and Celeron M processors is available for high performance applications that require a powerful processor. These processors, developed especially for mobile computing, offer many advantages for industrial applications as well. They combine a high computing capacity with low power consumption. The clock rates range from 600 MHz to 1.8 GHz. The Intel 855GME chip set contains two integrated graphic engines that provide optimal use of memory for the system and graphics.

Before we introduce Real-Time Workshop, let us first talk about Simulink. Because we need it to design the models and systems.

Simulink is a platform for multi domain simulation and Model-Based Design for dynamic systems. It offers an interactive graphical environment and a customizable set of block libraries that let you accurately design, simulate, implement, and test control, signal processing, communications, and other time-varying systems, and can be extended

[8] ©B&R 2007, courtesy of B&R company

Figure 1.10: Automation PC 620 [8]

for specialized applications.
Add-on products extend the Simulink environment with tools for specific modeling and design tasks and for code generation, algorithm implementation, test, and verification. Simulink is integrated with MATLAB, providing immediate access to an extensive range of tools for algorithm development, data visualization, data analysis and access, and numerical computation.
With Simulink, one can quickly create, model, and maintain a detailed block diagram of system using a comprehensive set of predefined block. It offers tools for hierarchical modeling, date management, and subsystem customization. And it makes easy to create concise, accurate representations, regardless of system's complexity.
after building your model in Simulink, you can simulate its dynamic behavior and view the results live. Simulink provides several features and tools to ensure the speed and accuracy of your simulation, including fixed-step and variable-step solvers and a graphical debugger.
Now we will introduce Real-Time Workshop.

Real-Time Workshop generates and executes stand-alone C code for developing and testing algorithms modeled in Simulink. The resulting code can be used for many real-time and non-real-time applications, including simulation acceleration, rapid prototyping, and hardware-in-the-loop testing. You can interactively tune and monitor the generated code using Simulink blocks and built-in analysis capabilities, or run and interact with the code outside the MATLAB and Simulink environment.
Real-Time Workshop is the foundation for Simulink code generation. It generates ANSI/ISO-C compliant code for an entire model or for an individual subsystem, enabling the code to run on any microprocessor or real-time operating system (RTOS). Add-on products (available separately) extend Real-Time Workshop with additional code

generation support and capabilities.
Real-Time Workshop is an integral part of the Simulink environment. You interact with Real-Time Workshop using the Simulink Model Explorer graphical user interface, giving you a single, consolidated place in Simulink to configure code generation settings. From the Model Explorer you can:

- Generate code for your Simulink models or subsystems
- Select a Real-Time Workshop target
- Configure the target for code generation
- Manage multiple configuration sets

The Model Adviser in Simulink checks your model configuration and offers advice on how to optimize or tune a configuration set based on your stated goals or style.
Real-Time Workshop intrinsically generates code and offers the most complete support available for Simulink features and components, including:

- Model referencing, enabling incremental code generation
- Embedded MATLAB function blocks in Simulink and embedded MATLAB functions in Stateflow
- Bus objects, enabling you to generate structures in your code
- Atomic subsystems, enabling code reuse via reentrant C functions
- Simulink S-functions, enabling you to simulate and interface with legacy code

Real-Time Workshop also supports a wide range of applications, from algorithm deployment with Stateflow, Simulink Fixed Point, and the Signal Processing Blockset to real-time simulation of systems modeled with the Aerospace Blockset, SimMechanics, SimPowerSystems, and other Simulink products.
And it can also generate code for Large-Scale applications. Real-Time Workshop provides incremental code generation, enabling you to generate code for specific blocks in your model without recoding the entire model. This component-based approach streamlines the development of very large models and reduces code generation build times.
For defining and controlling Data, Real-Time Workshop lets you control the way model data appears in the generated code. It also enables you to manage your data by:

- Declaring data types using built-in block data types (integer, floating-point, and fixed-point)
- Specifying storage to tune and calibrate parameters or constants
- Specifying storage to monitor and log signal data
- Reusing storage to minimize locally scoped data

Real-Time Workshop generates code from data stored in the diagram or in a data dictionary provided by the Simulink Model Explorer. This capability makes it easy to redeploy code from a single model to different targets by incorporating different data dictionary sets.
For executing code in a Real-Time environment, Real-Time Workshop provides a complete framework for executing the generated code in real-time and incorporating it into your execution environment. It generates single-rate or multi rate code based on the periodic sample times specified in the model. Code is deployed with or without an RTOS, and in single, multitasking, or asynchronous mode.
Simulink and Real-Time Workshop provide a complete set of target-independent capabilities for real-time deployment. These include:

- The ability to specify priorities for each rate in your model
- Production-quality counters and timers for computing absolute and elapsed time
- A Rate Transition Block to specify data transfer mechanisms between rates (for example, semaphore, mutex, and double buffering) to trade off data integrity, determinism, and performance
- Overrun detection for incorporating error-handling logic for each rate

3. The third possible way of combination [19]:

   Hardware: PC-compatible hardware
   Software: xPC Target 3.2 (from The MathWorks company)

   xPC Target provides a high-performance, host-target prototyping environment to enable you to connect your Simulink and Stateflow models to physical systems and execute them in real-time on PC-compatible hardware. xPC Target includes prove capabilities for hardware-in-the-loop simulation of control systems. xPC Target enables you to add I/O interface blocks to your models, automatically generate code with Real-Time Workshop and Stateflow Coder (both available separately), and download the code to a second PC running the xPC Target real-time kernel. You can use any PC with Intel or AMD 32-bit processors as your real-time target. The target PC can be a desktop computer, an industrial computer, PC/104, PC/104+, CompactPCI, all-in-one embedded PC, or any other PC-compatible form factor (like the hardware we described from B&R). Figure 1.11 showed prototyping setup using a laptop PC as the host computer and a single-board computer as a real-time target.

   After introducing some features of xPC, we will now show how to work with it. With a host computer running MATLAB, Simulink, Real-Time Workshop, xPC Target, and a C compiler as your development environment, you can create real-time applications and run them on a target PC using the xPC Target real-time kernel. You control execution on the target PC from MATLAB, using either a graphical or a command-line interface, supplied or custom host graphical user interfaces (GUIs), a standard Internet browser, or the target PC command-line interface. You can tune model parameters, acquire and view

[9] www.mathworks.com

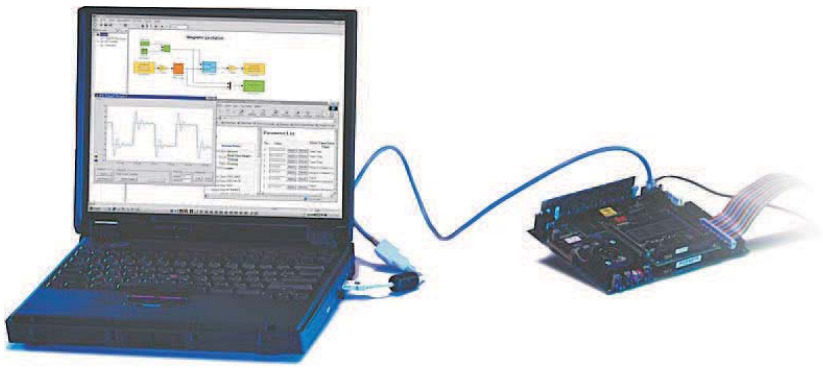

Figure 1.11: example an host and target computer[9]

signals, and obtain signal and target status information directly from your target PC.
Using standard PC hardware and commercial off-the-shelf I/O boards, xPC Target converts a standard PC into a real-time rapid prototyping or hardware-in-the-loop system. High performance is achieved by booting the real-time kernel rather than DOS or Windows.
Both interrupt-handling and polling modes are supported within the real-time kernel. Interrupt mode provides the highest application flexibility. Polling mode runs with less overhead, enabling you to achieve smaller sample times for applications. Using a high-performance Intel or AMD processor, you can achieve sample rates approaching 50 KHz.
The communicating between the host and target computers is s single communications link. Design your application on the host computer and download the real-time application to the target PC. The same communications interface is also used to pass commands and parameter changes to the target PC. You can choose either RS-232 or TCP/IP communications. RS-232 communication uses a null modem cable and a standard PC COM port on both the host and target PC, supporting rates of up to 115 kBaud. RS-232 communication provides the advantage of easier setup, without requiring an Ethernet card. TCP/IP communication is faster, providing data rates up to 1 Gbit/sec over any distance. xPC Target includes both an RS-232 cable and a PCI Ethernet card for the target PC.
Accessing the Target Application, the communicate with the target PC via an object-oriented, MATLAB command-line interface, which is used to pass commands to the target. You can also include the commands in M-files for detailed batch testing. The command-line interface consists of three function groups: target application control, parameter tuning, and signal acquisition (data acquisition). Graphical user interfaces (GUIs), built upon the command-line interface, can be used on the host or the target PC, or through a standard Internet browser. xPC Target Explorer, the host GUI, enables you to configure, control, and monitor operations of the target system, including access to multiple targets running concurrently.

Here is one example using xPC Target for Hardware in the loop simulation.
Automatic transmission system that includes an event-driven controller will be used in this example. First a non-real-time simulink model of the closed-loop system will be

written and the results of running a simulation of this model in Simulink which is shown in Figure 1.12.

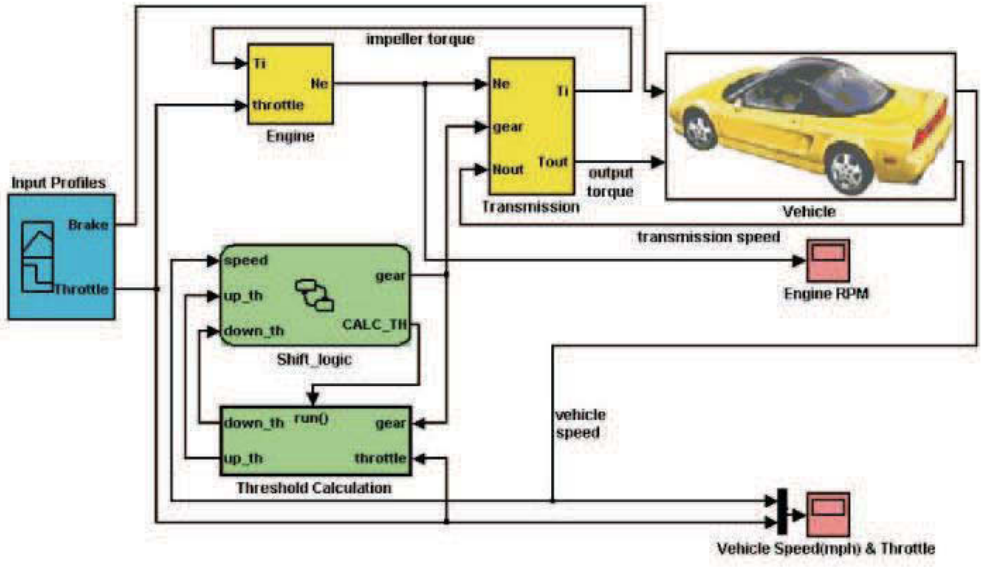

Figure 1.12: Yellow blocks represent the plant, an automotive powertrain system.
Blue blocks represent the throttle and brake inputs to the powertrain system.
Green blocks represent the controller portion of the system and include an event-driven controller and algorithmic calculations.
Red blocks represent the output oscilloscope displays. [11]

We can modify the Simulink model to interface with prototype hardware or actual plant hardware such as an engine or motor, or the controller hardware such as a prototype PC or embedded control unit. This approach allows us to test the controller and/or plant in real-time. The I/O device interface blocks used in the Simulink model are provided as part of xPC Target. The real-time operating system and the tools to monitor the operation of the application in real-time are also provided with xPC Target. Real-Time Workshop is used to generate code directly from the Simulink model. This generated code includes the needed driver interface code, derived from the xPC Target driver blocks used in the Simulink model. The complete application, including the I/O interfaces, is compiled, linked, and downloaded to xPC Target for real-time testing.
We can also modify the system so that the plant and controller are separate components that communicate with each other using a CAN (controller area network) bus, shown in Figure 1.13. This mimics the real system where a controller would be implemented on an embedded control unit which would communicate through a CAN bus with the actual hardware. This is done using CAN I/O blocks that are provided with xPC Target. Real-Time Workshop is used to generate the C code that is automatically downloaded and run on a dedicated target PC running the xPC Target real-time kernel.

[11] www.mathworks.com
[12] www.mathworks.com

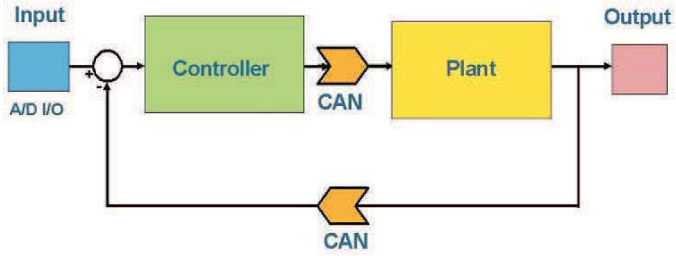

Figure 1.13: Control system architecture with CAN communication[12]

The Simulink diagram in Figure 1.14 represents the modified system including the CAN blocks, shown in orange. The inputs are provided by physical brake and throttle pedals connected to the target PC using an A/D (analog to digital) input board. The simulation scopes are replaced with xPC Target Scope blocks, shown in red. These blocks allow you to view signals in real-time on a monitor connected to the target PC. You can see that this Simulink model is very similar to the original control architecture outlined above.

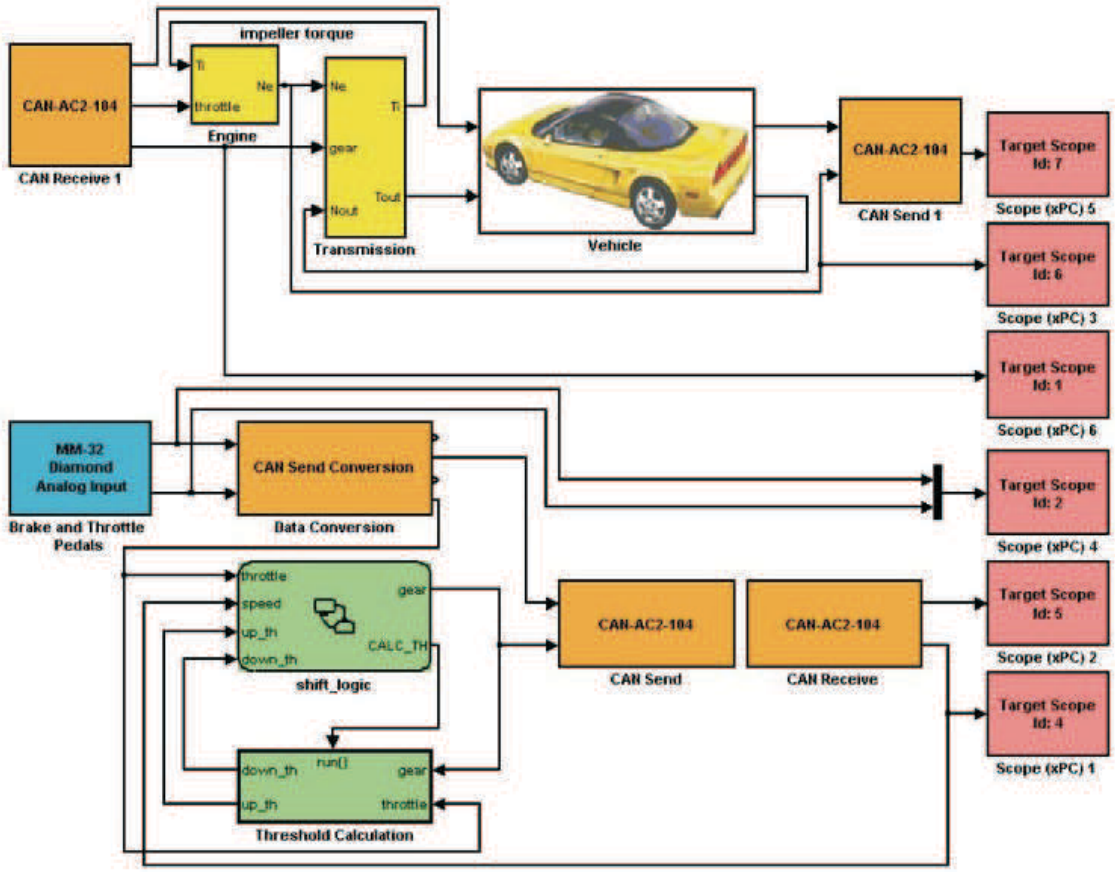

Figure 1.14: Real-time Simulink model with CAN communication[13]

[13] www.mathworks.com

Now we can run the real-time simulation, blow is the step to run the simulation:

- Connect a second target PC configured with the needed I/O cards to your main host PC using a standard Ethernet cable or RS-232 cable.
- Connect your pedals to the target PC through the A/D board input connection.
- Generate code from the real-time model and download it to the target PC using Real-Time Workshop.
- Run the simulation and view the target screen outputs as you change the throttle pedal input, all in real-time.
- Plot data logged from the simulation for additional analysis.

The hardware setup is shown in Figure 1.15. The blue box is the xPC TargetBox industrial PC, which is being used as the target PC. The laptop serves as the host PC.

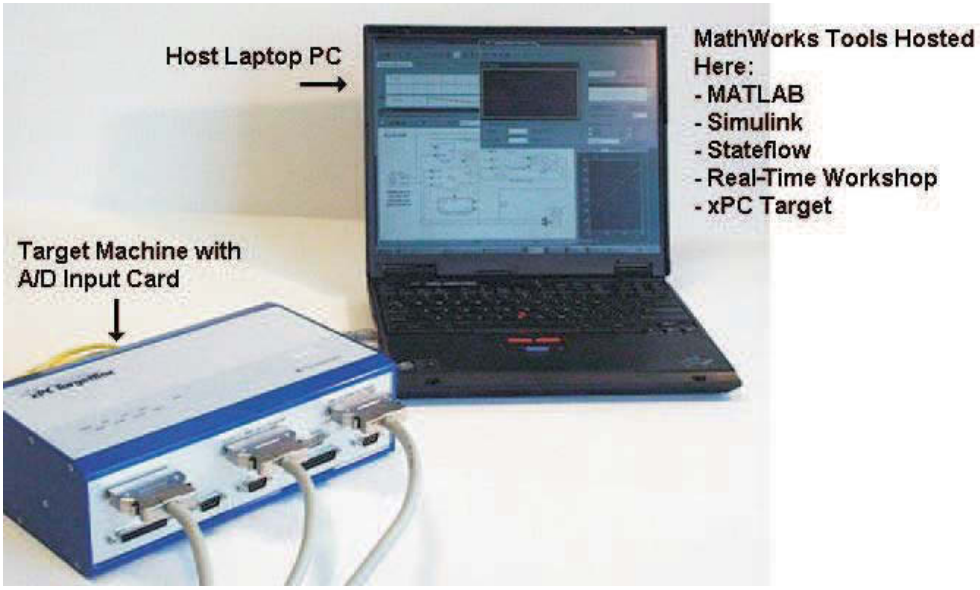

Figure 1.15: Host-target hardware setup[14]

4. The fourth possible way of combination [19]:

   Hardware: Desktop or Laptop PC or Other real-time hardware
   Software: Real-Time Windows Target 2.7 (from The MathWorks company)

   Real-Time Windows Target lets you run Simulink and Stateflow models in real-time on your desktop or laptop PC for rapid prototyping or hardware-in-the-loop simulation of control system and signal processing algorithms. You can create and control a real-time execution entirely through Simulink. Using Real-Time Workshop to generate C code, compile it, and start real-time execution on Microsoft Windows while interfacing to real

[14] www.mathworks.com

hardware using PC I/O boards. Other Windows applications can continue to run during operation and can use all CPU cycles not needed by the real-time task.
Now we will introduce shortly how to work with Real-Time Windows Target. In fig. 1.16 showed the exactly work flow.
The close integration with Simulink which we explained in detail before external mode makes it easy to use Real-Time Windows Target with Simulink. You can run your Simulink models in real-time and interface with physical devices. Using your desktop computer, you can implement real-time control, hardware-in-the-loop simulation, and other real-time applications from the Simulink environment.
And then Real-Time Workshop which was explained in detail already generates C code and creates a binary file using the supplied C compiler. The resulting binary file is ready to run in real-time on your Windows PC.
By selecting Simulink external mode, the Real-Time Windows Target kernel loads the application file into memory and establishes a connection with Simulink. You can then control model execution, data logging, parameter tuning, signal viewing, and starting or stopping real-time execution from the Simulink toolbar.

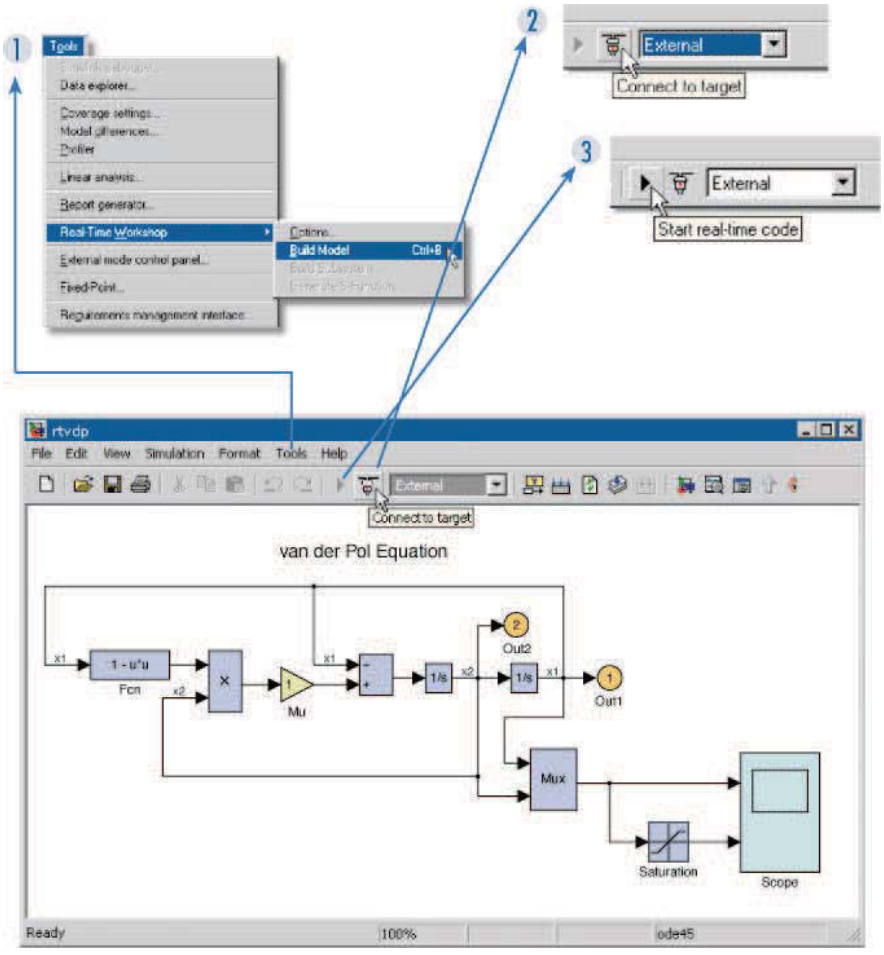

Figure 1.16: You control Real-Time Windows Target through Simulink dialog boxes and menus. You can generate and compile C code for your model (1), connect to the target to load it into memory (2), and start real-time code execution (3).[16]

5. The fifth possible way of combination [19] [13]

   Hardware: real-time hardware of National Instruments and Real-Time Workshop
   Software: LabView real-time Module of National Instruments

   First we can write the model or system with Simulink and Real-Time Workshop in Matlab. The real-time Module from LabView makes it possible to import the Matlab code to LabView code and then the code can be download and run in real-time hardware. One function called MATLAB Script (Windows, Not in Base Package) make this possible and easy.
   To execute external MATLAB scripts, you must have MATLAB installed on your computer to use MATLAB Script Nodes because the script nodes invoke the MATLAB script server to execute MATLAB scripts. Because LabView uses actives technology to implement MATLAB Script Nodes, they are available only on Windows.
   Use this node to execute external scripts. Figure 1.17 shown the command in LabView. Enter the script in the node or right-click the node border to import text into the node. Right-click the node border to add input and output terminals. Right-click a terminal to set its data type. When you create a MATLAB script, shown in 1.18 you must use an acceptable data type.

[16] www.mathworks.com
[17] www.ni.com
[18] www.ni.com

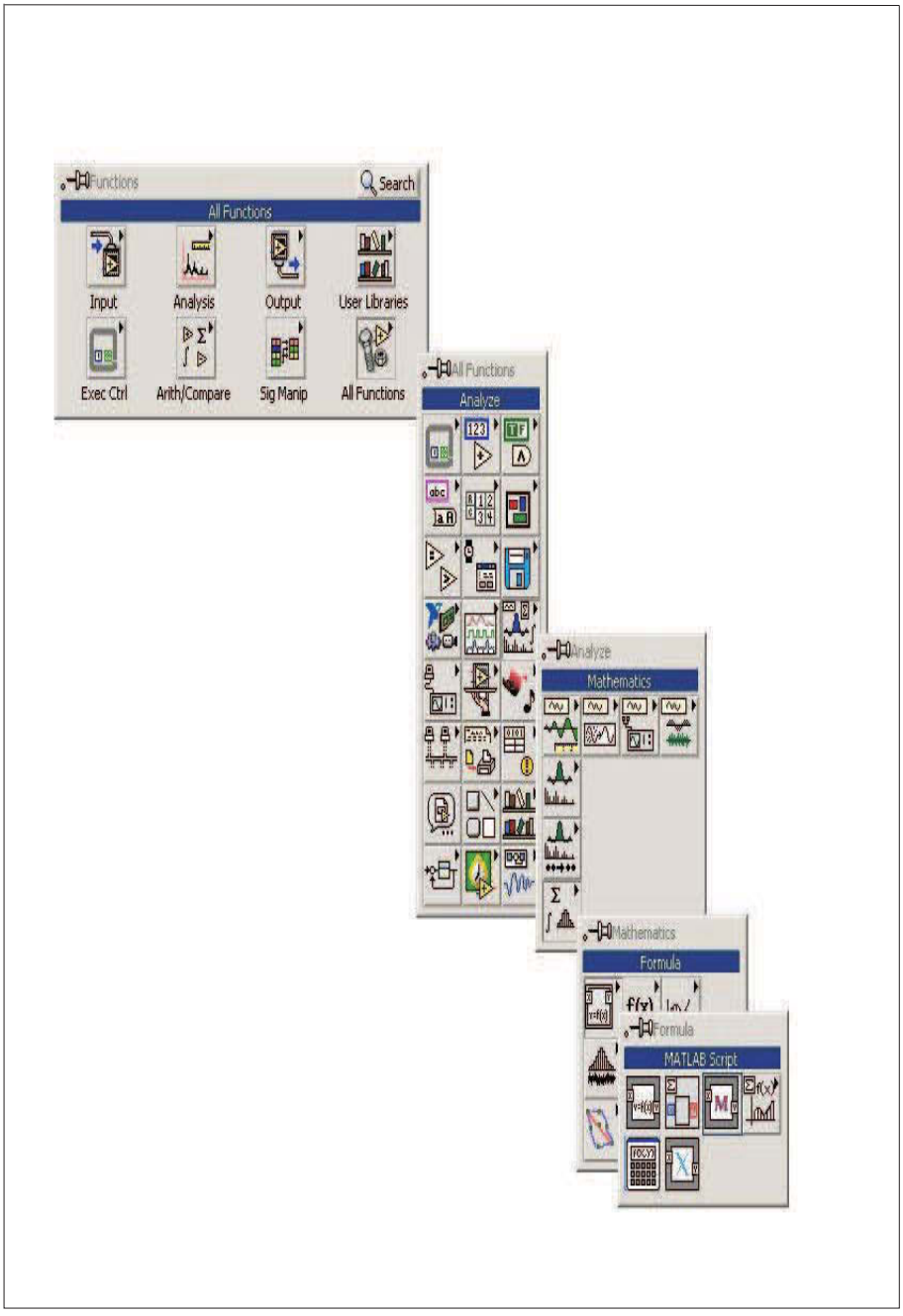

Figure 1.17: LabView command for Matlab script[17]

6. The last which to be introduced is the product from firm Dspace who is the producer of engineering tools for developing and testing mechatronic control systems [7].
   The software, for example, Automotive Simulation Models (ASM) are open to Simulink (Matlab/Simulink) models for the real-time simulation of standard automotive applications like diesel engine, gasoline engines and vehicle dynamics. The models will typically be used on a dSPACE Simulator for hardware-in-the-loop testing of electronic control units (ECUs) or during the design phase of controller algorithms for early validation by off line simulation. They are complete and independent models that support all

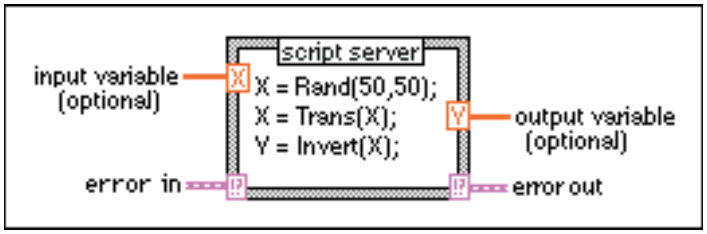

Figure 1.18: Matlab skript[18]

the relevant phases of the model-based development process. From hardware, they have Hardware-in-the-loop simulators, shown in 1.19.

Figure 1.19: Dspace simulator[19]

[19]www.dspace.de

Chapter 2

Mining Machine Simulation

In this chapter we will write some basic information and knowledge about the relevant areas of the mining machine what we will later simulate to understand and solve the problem. And then we will also look for the important and necessary information for the simulations.

2.1 Introduction of the Mining Machine

The machine in details can be seen in Figure 2.1. The excavation work is done by a cutting head on a telescopic boom which is independently movable in horizontal direction around the turret and the vertical direction around the linkage between the turret and the telescopic boom. The cutted materials fall down on the loading table which collect the materials and feed the chain conveyor, and it transports the material through the machine to the back. The material is reloaded onto the swiveling belt conveyer used to load. The rear stabilizer gives additional support and stability to stabilize the miner during the whole operation.

2.1.1 The Cutting Process

Here we will describe the general process to cut a tunnel.
First the miner positioning itself to the right place by moving on the crawler, and then the loading table and rear stabilizers are lowered so that the position of miner is fixed. The cutting head cuts into the wall to form an initial penetration and to give depth, in the 'sump in' process. This is done with the boom fixed in horizontal and vertical direction, it only by extracting the telescopic cylinder.
There are two phase of the cutting process. First is cutting, shown in Figure 2.2 and then profiling. After the depth is reached , the mining machine moves forward on the crawler tracks

[1] ©Sandvik 2007, courtesy of Sandvik

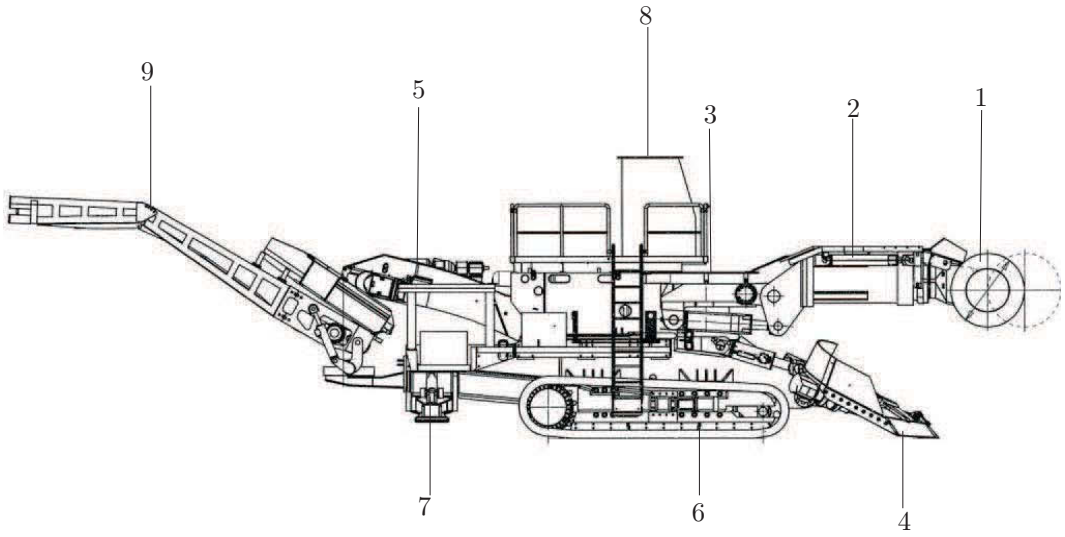

Figure 2.1: Main Parts and Function of the Mining Machinen[1]
1 Cutter head with gearbox 6 Crawler Track
2 Telescopic cutter boom 7 Rear stabilizer with frame
3 Turret 8 Operator cab
4 Loading table 9 Swivel belt conveyer
5 Chain conveyor

and pulls back the telescopic boom. The process is performed with the retracted telescopic boom in order to get a steadier cutting process with less wear of components. The cut cycle is performed with horizontal movement, (swiveling) from one side to the other with a certain cutting height. When the boom reaches the end point of each swiveling movement the boom is raised or lowered to cut another.
After a complete phase we need to perform a profiling cut, to get a smoother profile. The boom makes a movement around the contour of the phase to adjust the profile. This is usually done with a higher oil flow which allows higher profiling speed. When a complete face is cut, the miner sump-in again and repeats the procedure. There can also be additional profiling cuts before the miner makes another sump-in to cut another phase. It is desirable to cut the tunnel profile as accurate as possible when the cost of lining of the tunnel increases significantly if the inaccuracy is large. The miners are today run by an operator placed on the machine who, more or less, manually cuts the desired tunnel profile. The control system of the miner today is relatively advanced and a fully automated operation could be a reality in the future. To be able to fully automate the miner the performance of the miner has to be very accurate to get a satisfying result.

[2] ©Sandvik 2007, courtesy of Sandvik

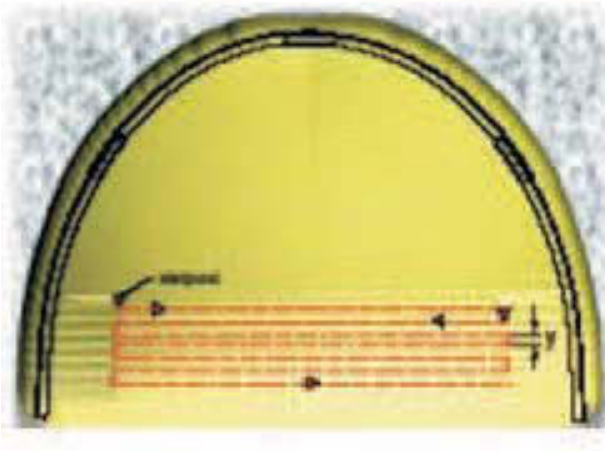

Figure 2.2: cutting process[2]

2.1.2 Profile Control System

The profile control system is designed to indicate when the boom has reached the pre-programmed limit of the tunnel wall. The head is modeled as two spheres in the profile control system and an indication is given when the edge of the outer sphere has reached the pre-programmed limit. This is optional equipment on the miners today but helps the operator to cut an accurate tunnel profile.

2.1.3 Cutting Technology

Having the right geometry and speed of the cutting head, and the right power behind it is necessary for the best utilization of the miner and to maintain the most economic durability.
The cutting head, shown in Figure 2.3 consists of water cooled picks mounted on the cutting head body in a pattern optimized to get the best cutting performance and homogeneous material size of the cut material. The picks are specially developed for optimal properties and cutting performance.
The geometry and design of the cutting head is adapted to certain conditions, such as cutting speed, slewing speed (horizontal movement) and the conditions of the rock. There are different heads to choose from, with different numbers of picks and positions of these picks, depending on the hardness of the rock.

[3] ©Sandvik 2007, courtesy of Sandvik

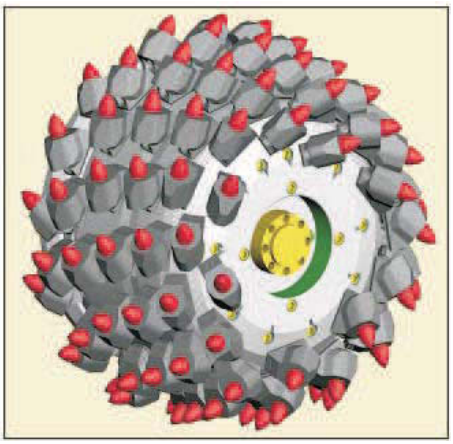

Figure 2.3: cutting head[3]

2.1.4 Kinematics

Definition to describe Kinematic

The simplest way to describe the series robot is two types of matrices [20] [8]:

- Transformation matrices and
- Coordinate mapping matrices

The transformation matrices m_i describe the rotation of robot arms around its respective rotations axis. So for one robot having p axis, it needs to have also exactly p transformation matrices. The general formula is:

$$m_i = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \cos(u_i) & -\sin(u_i) \\ 0 & 0 & \sin(u_i) & \cos(u_i) \end{bmatrix} \tag{2.1}$$

From this formula, we can notice that every axis rotation is describe as the rotation (with angle u_1) around the local x-axis.
The coordinate mapping matrices γ_i describe the coordinate transformation between two coordinate systems. For calculating the robot with p axis, we need p - 1 coordinate mapping matrices. But when the origin of the end-effector system is not the origin of the last axis of

robot - for example, Tool Center Point - then we need one more matrix. The general formula is:

$$\gamma_i = \begin{bmatrix} 1 & 0 & 0 & 0 \\ d_i & \cos(\alpha_i) & -\sin(\alpha_i) & 0 \\ 0 & \sin(\alpha_i) & \cos(\alpha_i) & 0 \\ a_i & 0 & 0 & 1 \end{bmatrix} \tag{2.2}$$

The parameters d_i, a_i, α_i are called Denavit-Hartenberg-Parameter.

Design of Mining Machine

For the design of the machine we will simulated we need the Denavit-Hartenberg-Parameter. In the following, we will make one Kinematic model of it.
We can see in the Figure 2.4, it shows the basic geometric proportion of the mining machine to describe the cutter boom and the coordinates systems. The z coordinates of every coordinates system shown in the same direction in order to assume an application of coordinate mapping matrices. The x coordinates show in the direction of rotation axis in order to enable the application of transformation matrices.

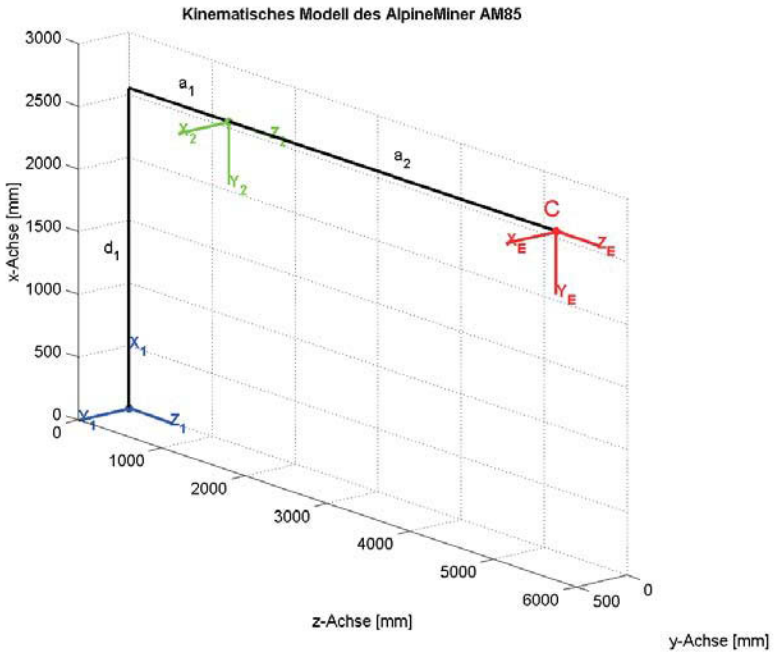

Figure 2.4: kinematic model

For the further consideration we will take point C (center of the middle point of two cutting heads) as the end-factor of model.
We will research the the geometry of the kinematic model with Denavit-Hartenberg-Paramete

- shown in Table 2.1. It is used for axis 2 to define the end - effector coordinate system. The result from it is the displacement of the a_2.

Axis i	Distance a_i [mm]	Offset d_i [mm]	Deflection-angle α_i [rad]
1	1200	2430	$\frac{\pi}{2}$
2	3894	0	0

Table 2.1: Denavit - Hartenberg - Parameter

Forward Kinematics

To describe the kinematics of the mining machine we need two coordinate mapping matrices and two transformation matrices. The complete transformation matrix G is the product of the matrices:

$$G = m_1\gamma_1 m_2\gamma_e \tag{2.3}$$

Here γ_e describe every coordinate mapping matrices that point c needs to move on the axis 2. The matrix G is then all the transformation matrices needed to describe the transformation one points from the end - effector coordination system to fix frame coordinate system which is axis 1 in Figure 2.4.
Now we can describe point c in the fix frame coordinate system :

$$C_{rast} = GC_{ee} \tag{2.4}$$

c_{ee} is the position of point c in the end - effector coordination system which can be describe in homogeneous coordinate:

$$C_{ee} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{2.5}$$

We can write c_{rast} after adoption of all the parameters:

$$G_{rast} = \begin{bmatrix} 1 \\ d_1 + \sin(u_1)a_2 \\ -\sin(u_1)a_1 - \sin(u_1)\cos(u_2)a_2 \\ \cos(u_1)a_1 + \cos(u_1)\cos(u_2)a_2 \end{bmatrix} \tag{2.6}$$

Inverse Kinematics

For the inverse kinematics we know one point with position and direction of end - effector coordinate, and need from this to ensure the articular angle u_i. So we have the Matrix of the end position in fixed frame coordinate:

$$G_0 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ x_p & 0 & -1 & 0 \\ y_p & 1 & 0 & 0 \\ z_p & 0 & 0 & 1 \end{bmatrix} \tag{2.7}$$

The machine has two degree of freedom (u_2 and u_2),so that we need two coordinates (x_p and y_p) to define it. But in the space we need the third coordinate z_p which is necessary with x_p and y_p to fix the position. And setting the matrix G_0 is equal to matrix G, we get the u_i as the function of the x_p and y_p :

$$u_1 = \arcsin(-\frac{y_p}{(a_1 + \cos(u_2)a_2}) \tag{2.8}$$

$$u_2 = \arcsin(-\frac{x_p + d_1}{a_2}) \tag{2.9}$$

2.1.5 Hydraulic System

The cutter boom angles are controlled with hydraulic proportional valves. A load compensation makes the oil flow independent from the pressures in the cylinder. A counterbalance valve closes the meter-out side, if the load is overrunning. During still stand, these valves close and provide safety against unexpected motion caused by tube rupture or other events.
When the system does not consume much oil, a load sensing pump reduces the supply pressure.

2.2 Simulation Requirement

For this project we need HIL simulation. We simulate the mining machine to test the control systems. This system should be real-time capable with similar behavior like the real machine. We need to simulate the essential properties of the real system: kinematics, dynamic behavior of the hydraulic system and the mechanical construction, further the cutting load force. For visualisation we will show the cutting load force and the video of the movement of the cutting arm - changing x and y coordinate.

2.3 Simulation Details

2.3.1 Hydraulic

The PID controller calculation

In an industrial process a PID controller attempts to correct the error between a measured process variable and a desired set point by calculating and then outputting a corrective action that can adjust the process accordingly.
The PID controller calculation (algorithm) involves three separate modes; the Proportional mode, the Integral mode and Derivative mode. The proportional mode determines the reaction to the current error, the integral mode determines the reaction based on recent errors and the derivative mode determines the reaction based on the rate by which the error has been changing. The weighted sum of the three modes is outputted as a corrective action to a control element such as a control valve or heating element.

Valve Spool

The relative amplitude of the control piston (-1..0..1) alleges the gradation of the oil flow and the opening of the piston chamber, shown in Figure 2.5. A normalised input signal is used so that the simulation of every kind of valves is independent on the absolute valve of piston stroke: the inertia of the spool is simulated with a rate limitation for the position change.
The normalised input is then expanded with a scale factor to the nominal value of the electric input current. A backlash element regards the influence of the moving direction, see in Figure 2.6.

Directional Control Valve

With the position of the control piston the oil that flows over the respective edge is controlled.
The opening of the four orifices is modeled with look-up tables with the spool position as input. The spool position is not represented in millimeters, but with the current needed for a certain elongation. So the look-up table can be expressed with the measured valve characteristics (oil flow over input current), shown in Figure 2.6.
Beside the opening k, the oil flow depends also on the pressure across the edge, see the equation 2.10.

$$q = k \frac{q_N}{\sqrt{P_N}} \sqrt{p} \tag{2.10}$$

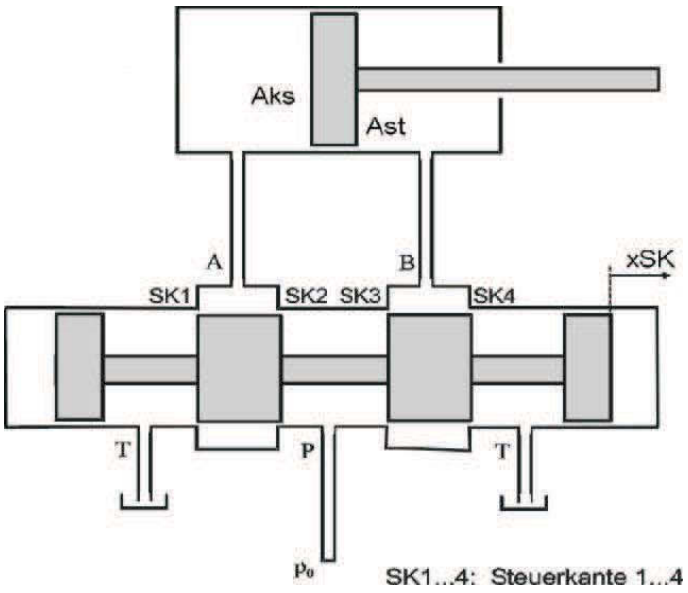

Figure 2.5: Control piston

q_N ... nominal oil flow
P_N ... pressure drop for nominal oil flow
k ... valve opening factor 0..1
Negative pressures may occur, which cause an oil flow into the opposite direction. To avoid a negative argument for the square root, a modification must be done,see equation 2.11:

$$q = k\frac{q_N}{\sqrt{P_N}} sign(p)\sqrt{|p|} \tag{2.11}$$

During operation it could happen, that the oil pressure tends to go below zero, which is not possible in reality. A vacuum will occur, this is called cavitation. A saturation block inhibits the pressure falling below zero.
In Figure 2.6 show the simulation model of hydraulic in horizontal and vertical direction.

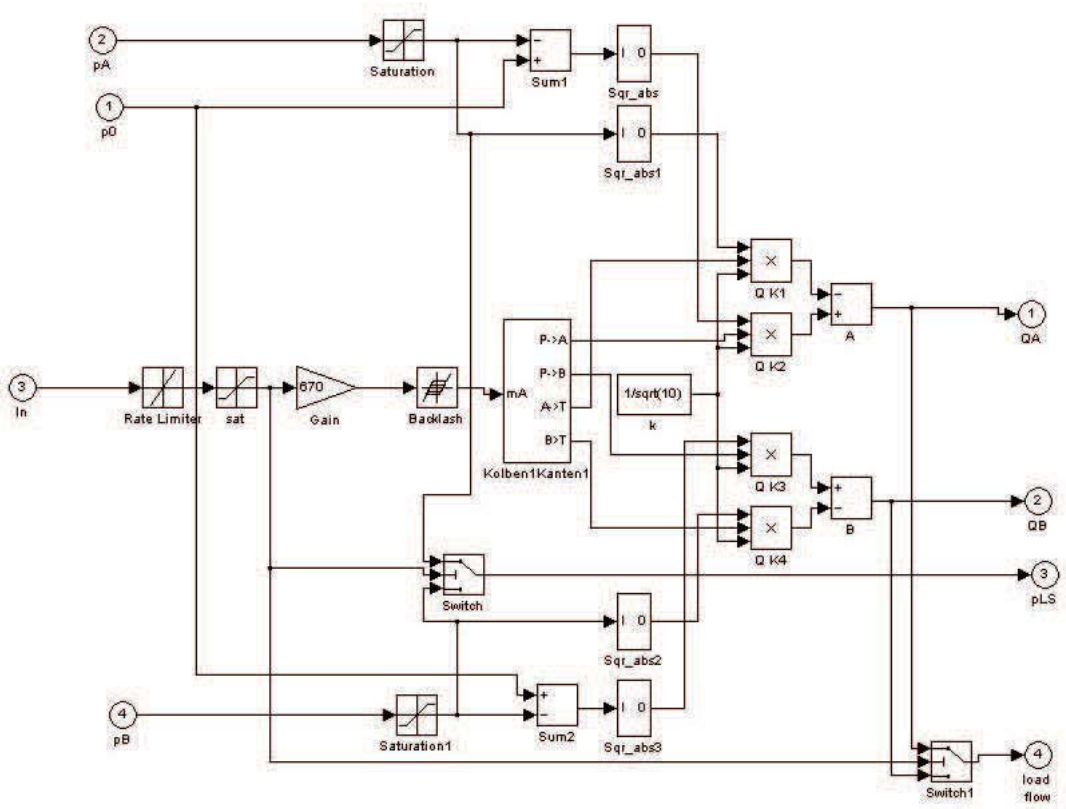

Figure 2.6: Hydraulic model in horizontal und vertical direction

2.3.2 Cylinder

The model for a hydraulic cylinder is shown in Figure 2.7.

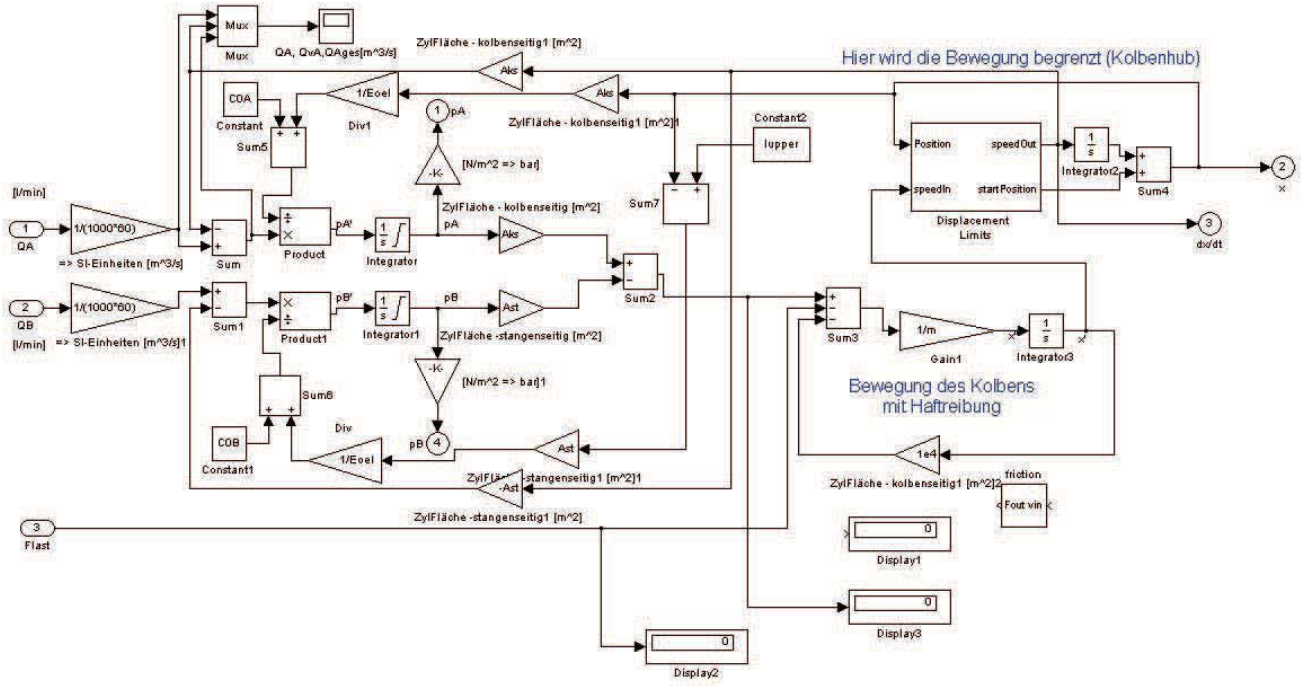

Figure 2.7: Simulation model of cylinder in horizontal and vertical direction

The oil flows coming from the valve are inputs, whereas the pressures are outputs. The pressures are calculated in the following way.
The Bulk modulus of a solid or liquid material describes the volume change caused by a pressure change and is defined in eqation 2.12:

$$\mathrm{E} = -\mathrm{V}\frac{\partial p}{\partial V} \tag{2.12}$$

E ... Bulk modulus of the oil (approx. 1.4GN/m2)
V ... Volume of the oil
p ... pressure of the oil

Expanding with ∂t gives:

$$\mathrm{E} = -\mathrm{V}\frac{\partial p}{\partial t}\frac{\partial t}{\partial V} \tag{2.13}$$

The oil volume inside the cylinder is decreased by the incoming oil and the motion of the piston. This gives equation 2.14:

$$\mathrm{E} = -\mathrm{V}\frac{\dot{p}}{q - \dot{x}A} \tag{2.14}$$

Solving for the pressure change gives the formula 2.15 for modeling the pressure in a cylinder chamber, see Figure 2.7:

$$\dot{p} = \frac{E}{V}(q - \dot{x}A) \tag{2.15}$$

The pressure is obtained by integrating over time.
Multiplication with the area gives the force applied to the piston, which is modeled with its mass m. An additional input allows to apply the external load force to the piston of the cylinder. After integrating the acceleration, the speed of the piston is obtained. When the piston reaches its motion limits, this speed must be overridden and set to zero. In spite, motion in the opposite direction must be allowed. These functions are performed inside the limitation block in the model in Figure 2.7.

2.3.3 Simulation of Mechanical System

Figure 2.8 shows the simplified geometric model of the cutting system of the mining machine. It is a two-degree-freedom serial robot. The geometric of cutting arm can be simplified in Figure 2.9. In most situations, the motion of the machine is not combined, each direction is driven separately. This allows a simplified simulation that neglects the Coriolis forces. Consequently two separate simulation models for the vertical and horizontal motions without any coupling are used.

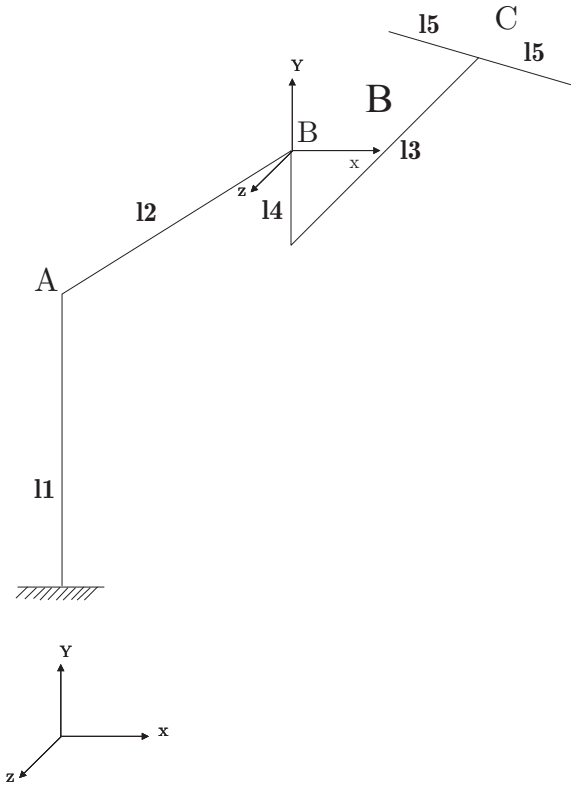

Figure 2.8: Simple geometric model

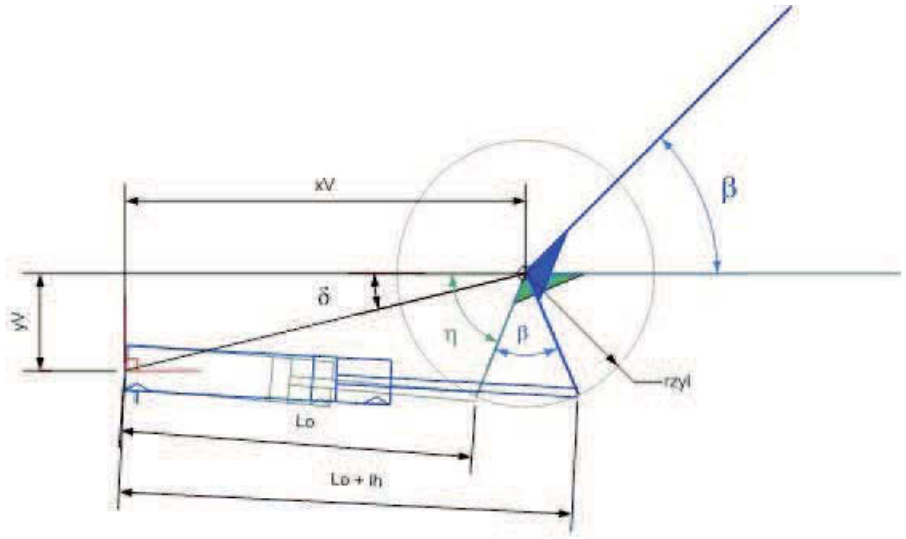

Figure 2.9: Simplified geometric model of cutting arm

Figure 2.10 shows the model for the horizontal system. The cutting arm is characterised with its moment of inertia Jhor. It is driven by the torque of a rotational spring, which represents the elasticity of the steel construction. The force output for the cylinder model is calculated from this torque. The momentum of an external cutting force is also added at this point. A model for viscous friction lets oscillations vanish after some time.

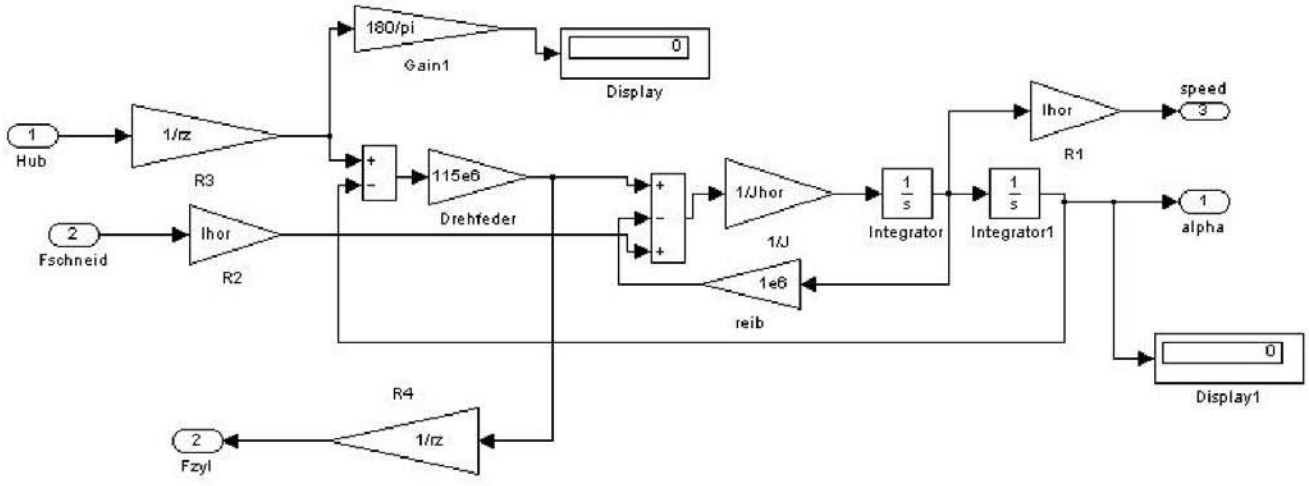

Figure 2.10: Horizontal system in simulation

The model for the vertical motion is shown in Figure 2.11. Additionally the gravitation force has to be regarded.

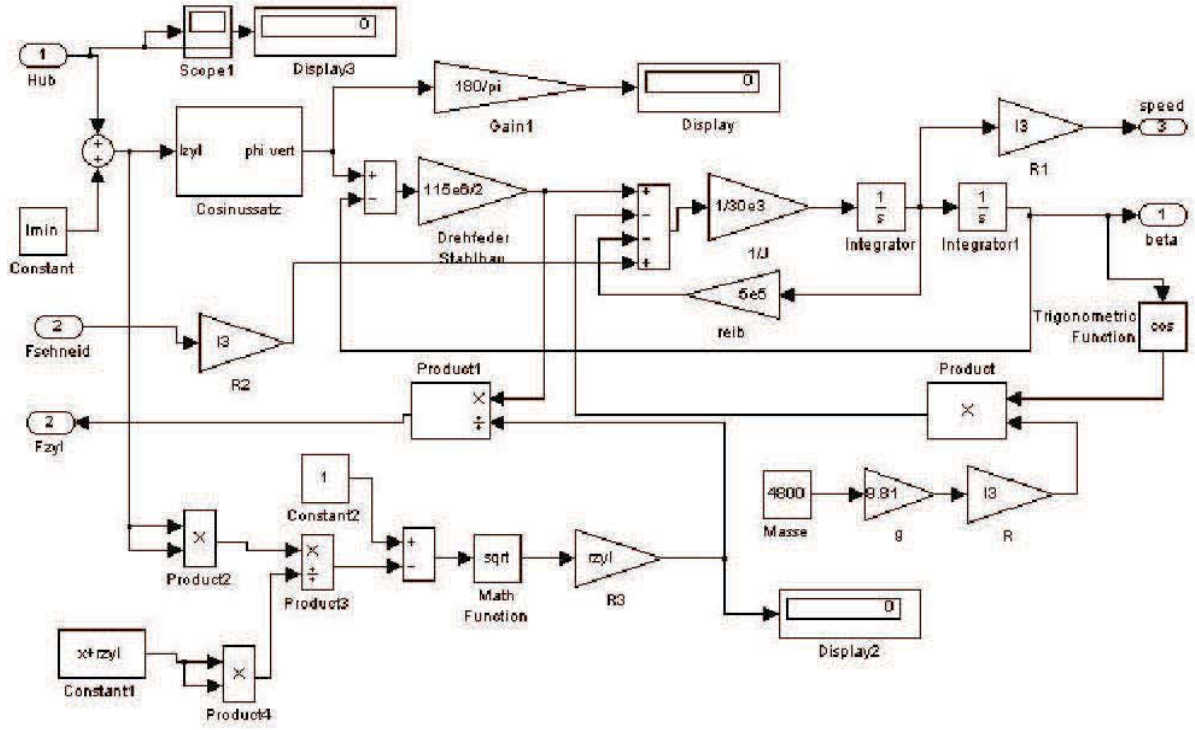

Figure 2.11: Vertical system in simulation

For this model we also use law of cosine, see formula 2.16, to get the β, the vertical angle of cutting head. We will explain it with the draft in Figure 2.12. With Xv and Yv the x and the angle η_1 can be calculated. x is a, r_{zyl} is b and l_min+ hub is c, Now unknows of the law of cosine which we need to calculate γ are completed. The maximal angle η which is in consequence of the vertical swing area is also given. From this all, angle β can be calculated. In Figure 2.13 give the simulation model of law of cosine.

$$\alpha^2 = b^2 + c^2 - 2bc\cos(\alpha) \tag{2.16}$$

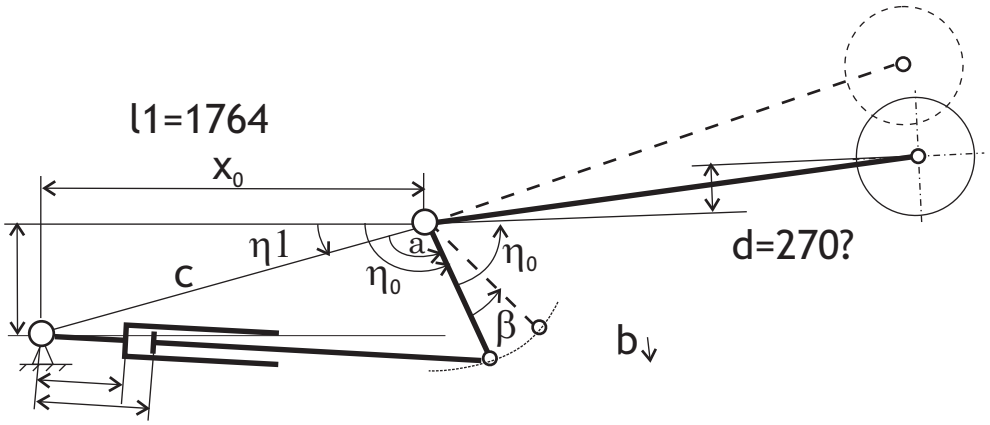

Figure 2.12: Draft of Law of Cosine

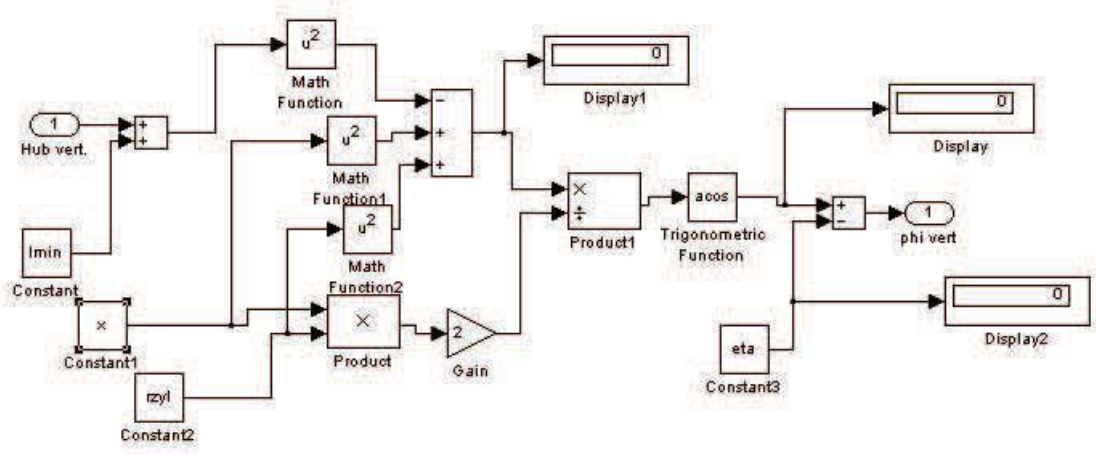

Figure 2.13: Simulation model of Law of Cosine

2.3.4 Kinematics

After solving the geometric problem, it means the horizontal and vertical angles can be calculated, we can give the equation of motion which gives the end position of the cutting head (forward kinematics).

$$x = -l_2 \sin\alpha - l_4 \sin\alpha \sin\beta - l_3 \sin\alpha \cos\beta \tag{2.17}$$

$$y = -l_4 \cos\beta + l_3 \sin\beta \tag{2.18}$$

In Figure 2.14 shown how to build and realize the kinematic model in the simulation.

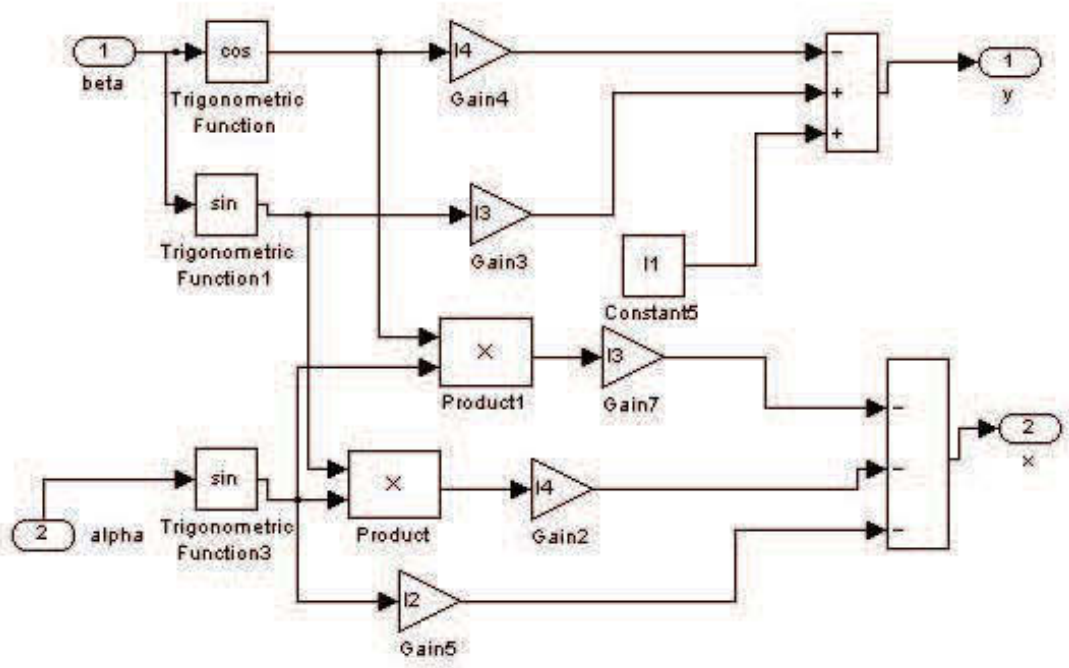

Figure 2.14: Kinematic model in simulation

Chapter 3

Hardware in the Loop Simulation with LabView

In this chapter we have done the simulation with LabView, the Real-Time Module provides a toolbox for simulation. It contains integrators and mathematical function that are required for solving ordinary differential equation (ODE). Following we will present every detail needed in simulation and all process of it.

3.1 Experiment Setup

We used the hardware and software from the National Instrument. Compact FieldPoint and Real-Time module which was discussed in the section 1.2.4. Figure 3.1 shows the whole setup for this simulation. The signal generator will create one signal as input or we can use the joy stick to change signal manually. They are connected to analog input of Compact Field-Point hardware with the cable. The output is connected to an oscilloscope. The connection between processor (Compact FieldPoint) and computer is through TCP/IP which realize the data exchange.

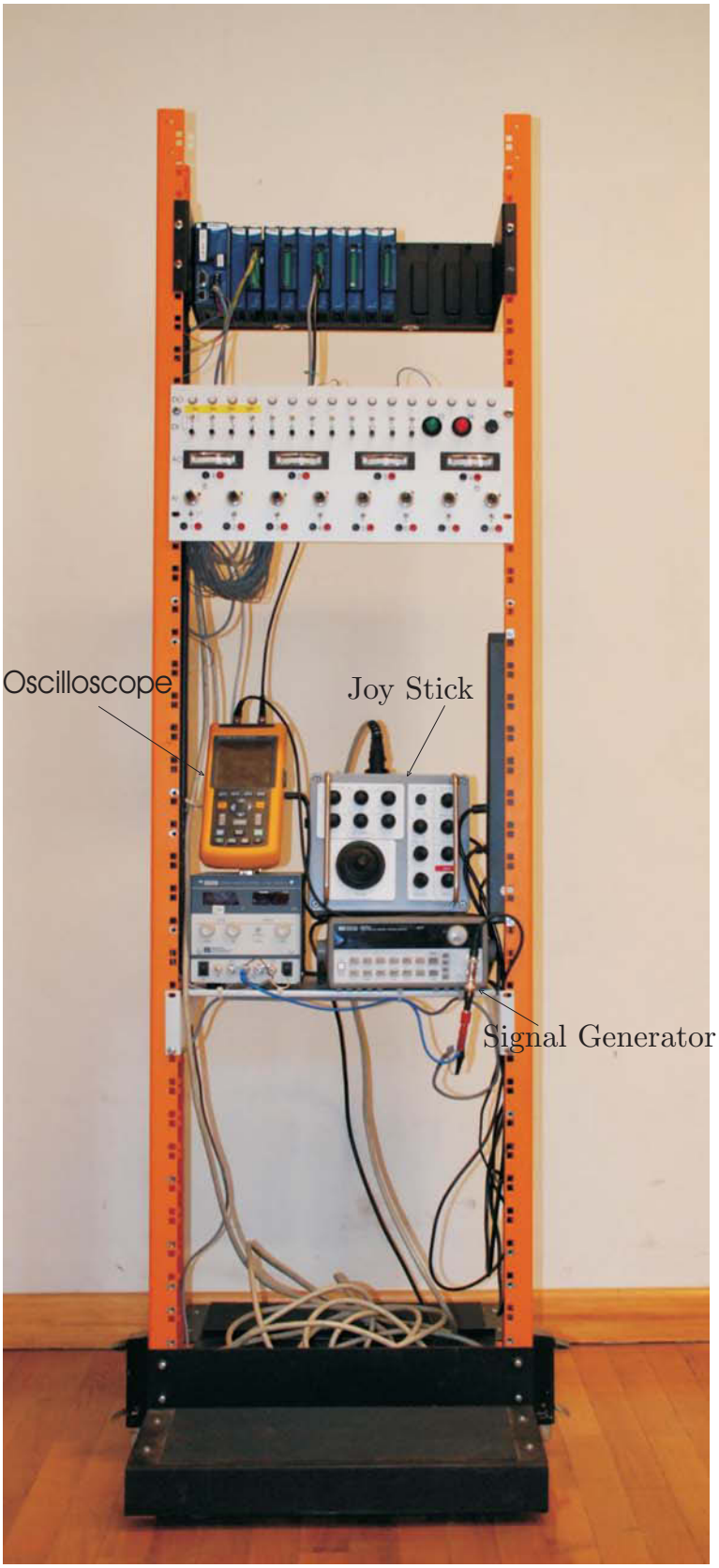

Figure 3.1: Setup for LabView simulation

3.2 Configuration

NI offers a software called NI Measurement & Automation Explorer, see Figure 3.2, to configurate the hardware (for us the Compact FieldPoint) and define the inputs and outputs.

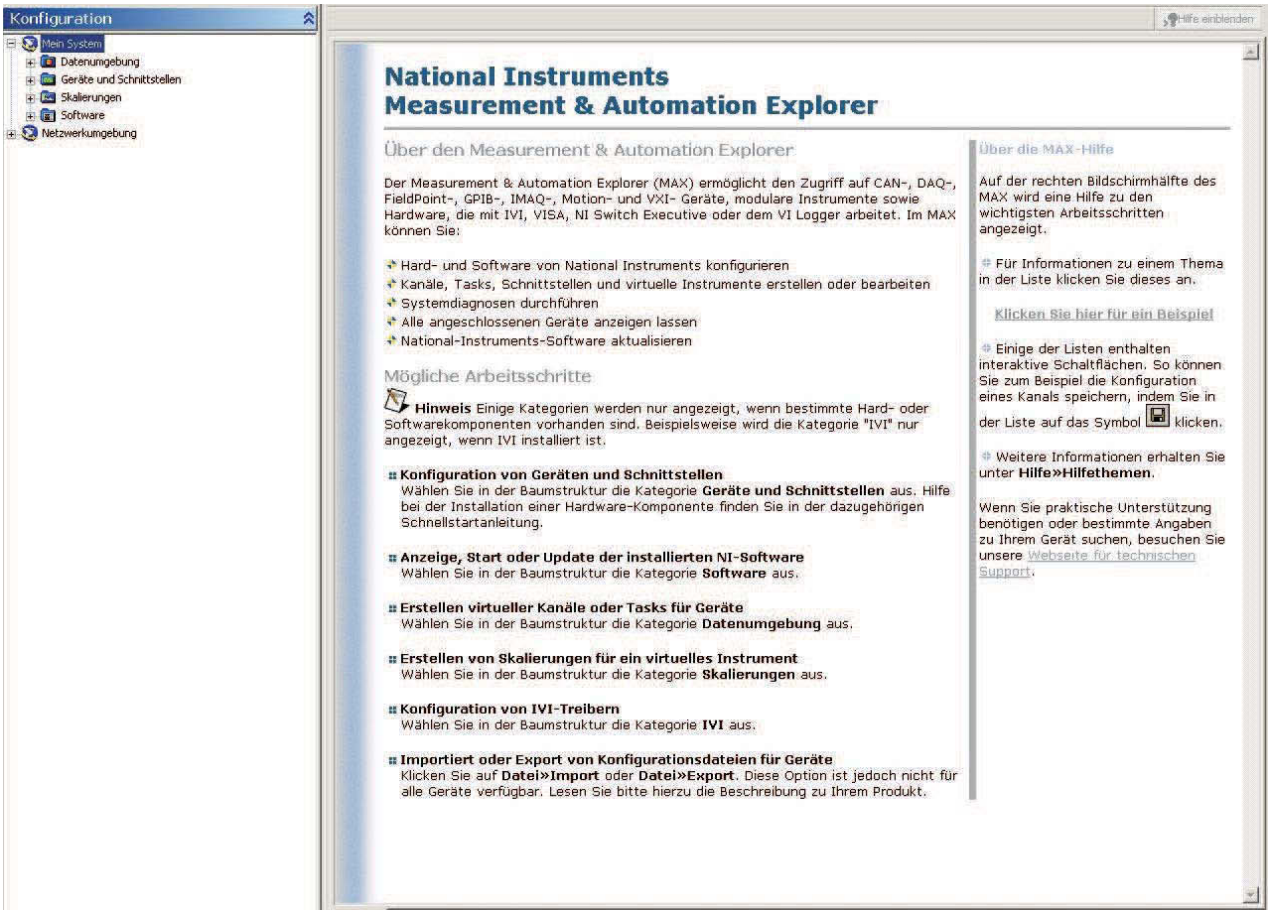

Figure 3.2: NI Measurement & Automation Explorer

At the ´konfiguration´ window, pressing down the right mouse button at ´Netzwerkumgebung´ will display the Internet connect (IP Address). From this displayed window right-click the IP Address you can find and choose ´Find Devices´, shown in Figure 3.3. In this you can define all the input and output channels, their ranges and so on, shown in Figure 3.4. If there are some old data still down in the processor, you can choose ´Dateitransfer´, see Figure 3.5, to rename and delete it, shown in Figure 3.6.

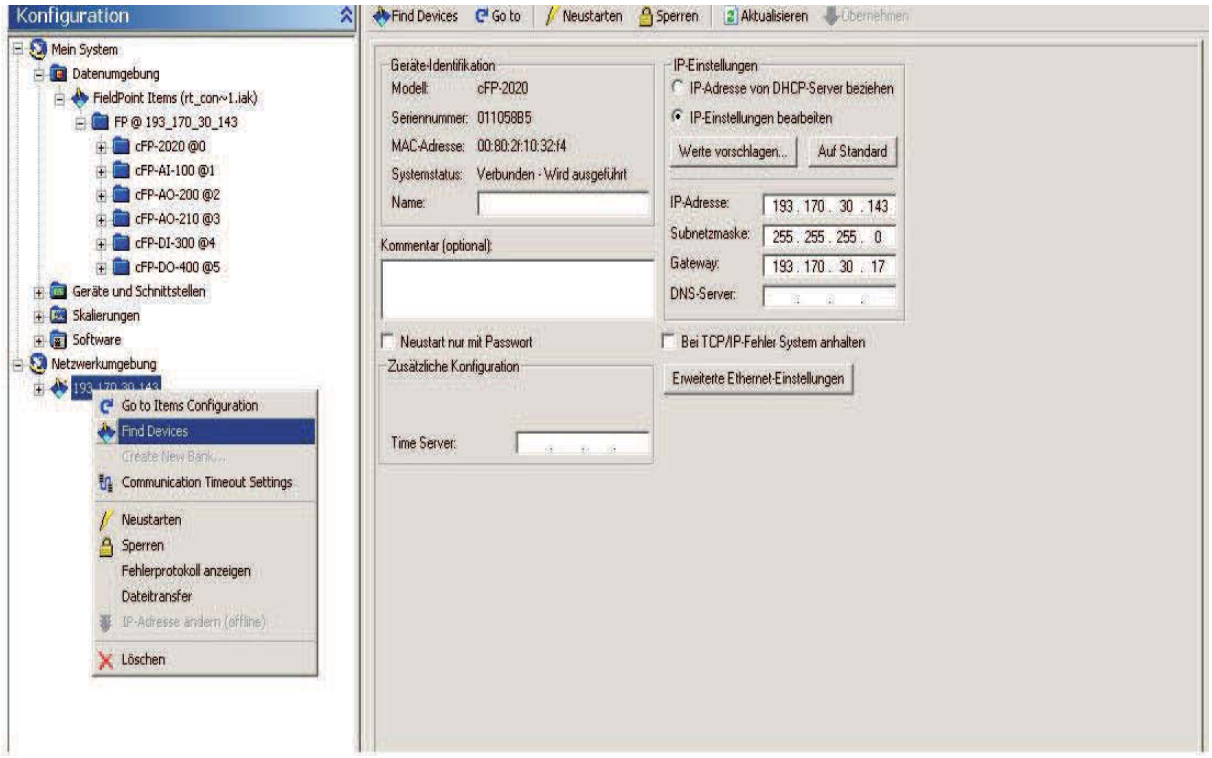

Figure 3.3: Find Devices in Measurement & Automation Explorer

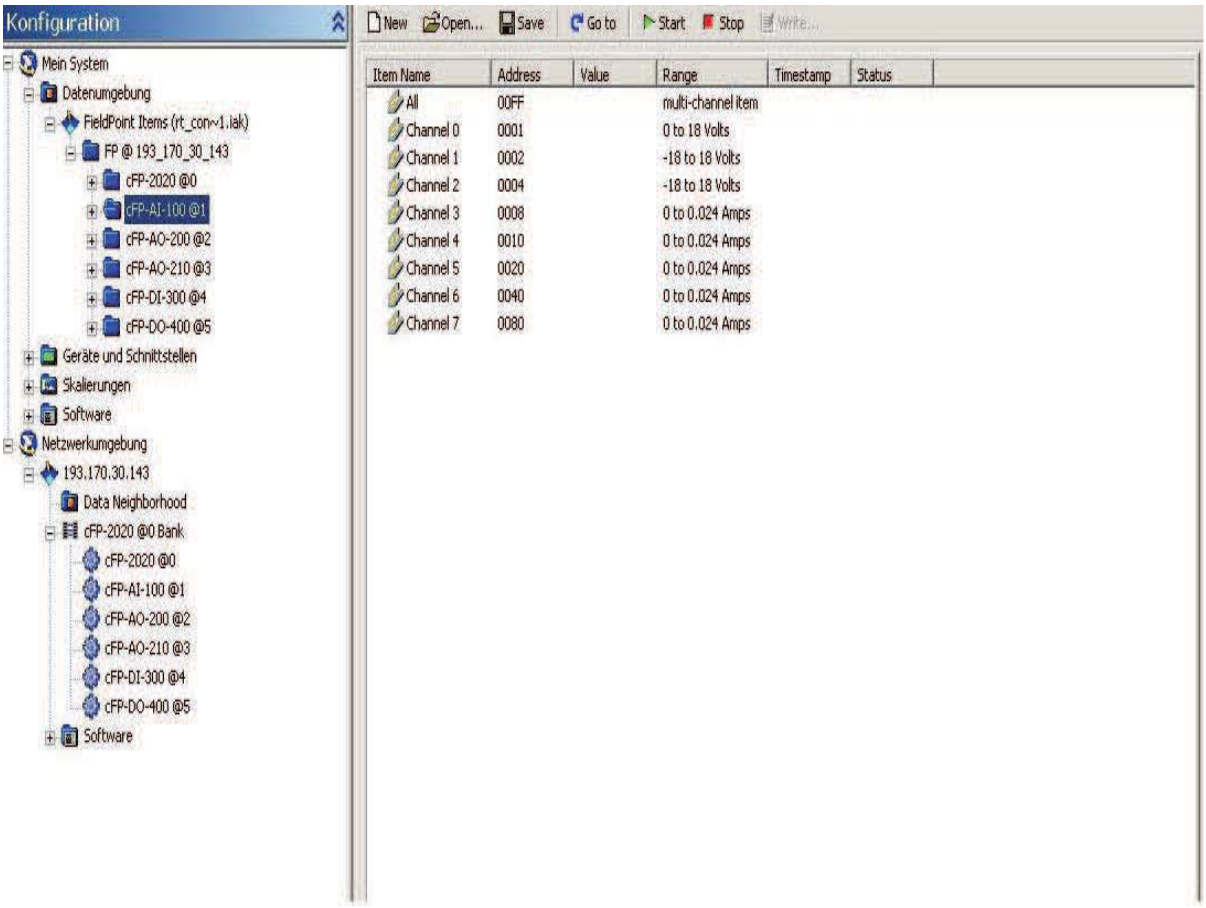

Figure 3.4: Configuration of input and output

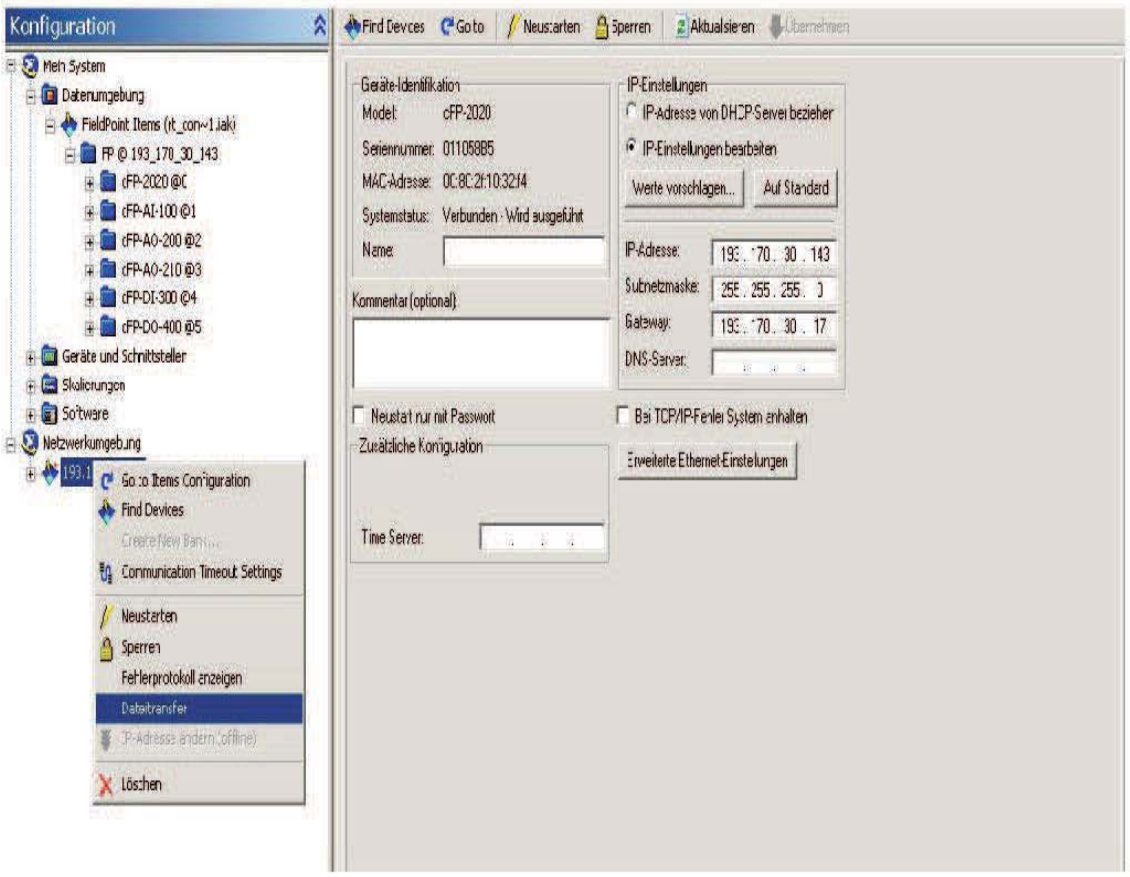

Figure 3.5: Dataitransfer in Measurement & Automation Explorer

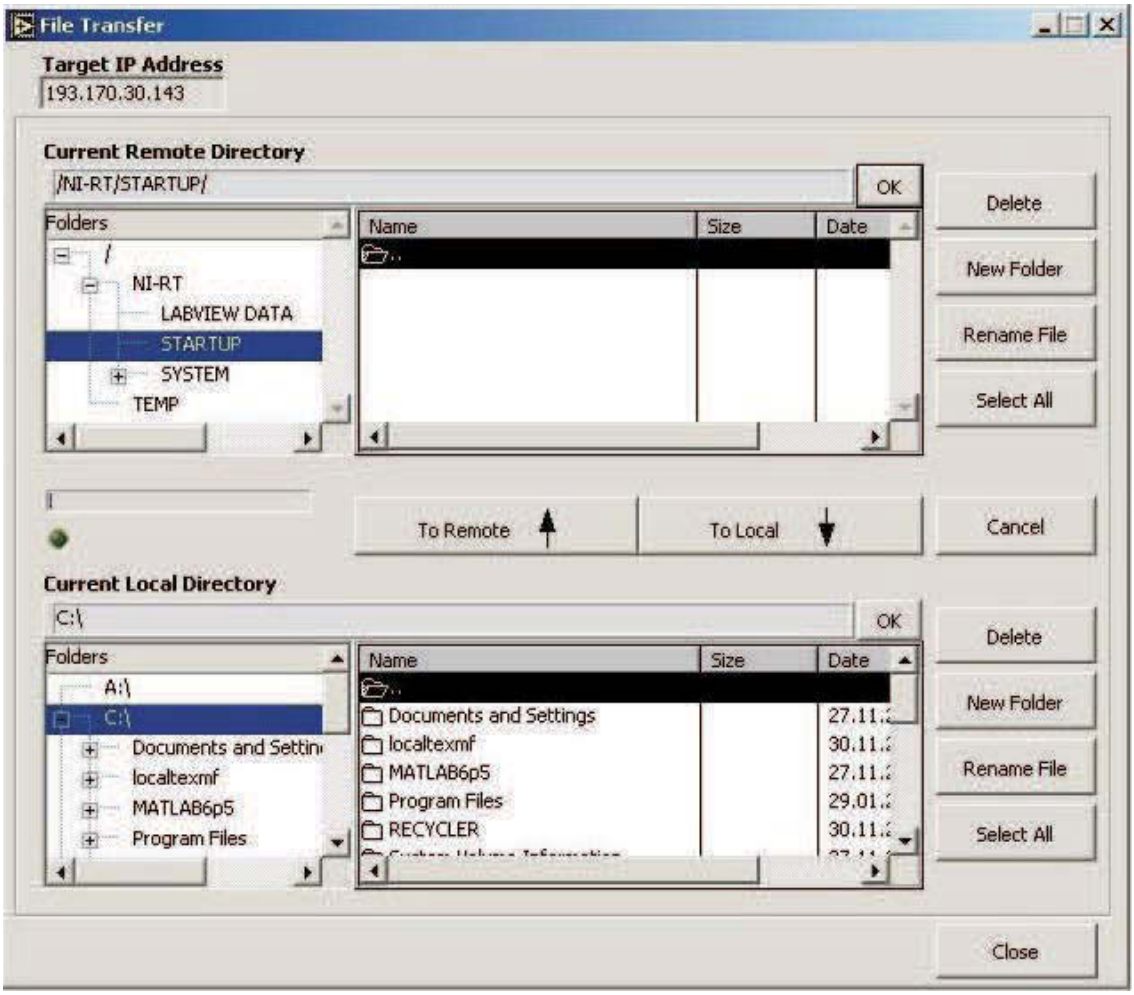

Figure 3.6: File Transfer and Startup

3.3 Simulation Model and Process

The simulation model is designed with the LabView programming environment, the program source code see Appendix A. But only the properties of mechanical construction and kinematics which are considered already in detail in section 2.3 are built. The reason why we have not made the simulation part of hydraulic and cylinder will be explained in the section 3.5.
Some important details for the LabView simulation will be introduced in the following.
After configuration of all the inputs and outputs, we can get the input signal through the LabView function called FieldPoint Read, with which the input signal channel can be chosen, shown in Figure 3.7 and 3.8.

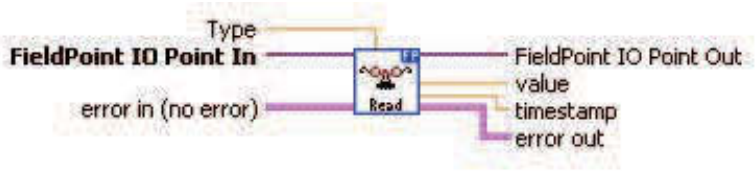

Figure 3.7: Function FieldPoint read

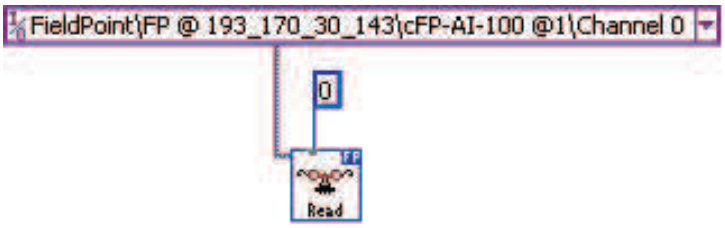

Figure 3.8: FieldPoint read in simulation model

For the output the LabView provide a similar function called FieldPoint Write, shown in Figure 3.9, with this we also can choose through which output channel we want write our output, shown in Figure 3.10.

Figure 3.9: Function FieldPoint write

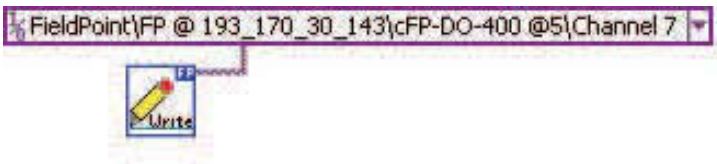

Figure 3.10: FieldPoint write in simulation model

For the simulation is also important how to exchange the data between simulation loops and while loops [12].
It can be put into one or more ordinary While loops running in parallel with the Simulation loop. Data can be exchanged between the loops using local variables. In the Figure 3.11 shown for example the value of the vertical pivoting angle is generated in the while loop, being used in the simulation loop via a local variable. Note how both loops are stopped by only one Stop button. A local variable is used, and a boolean constant drawn to another local variable causes the while loop to halt when the Stop button in the simulation diagram is pressed.

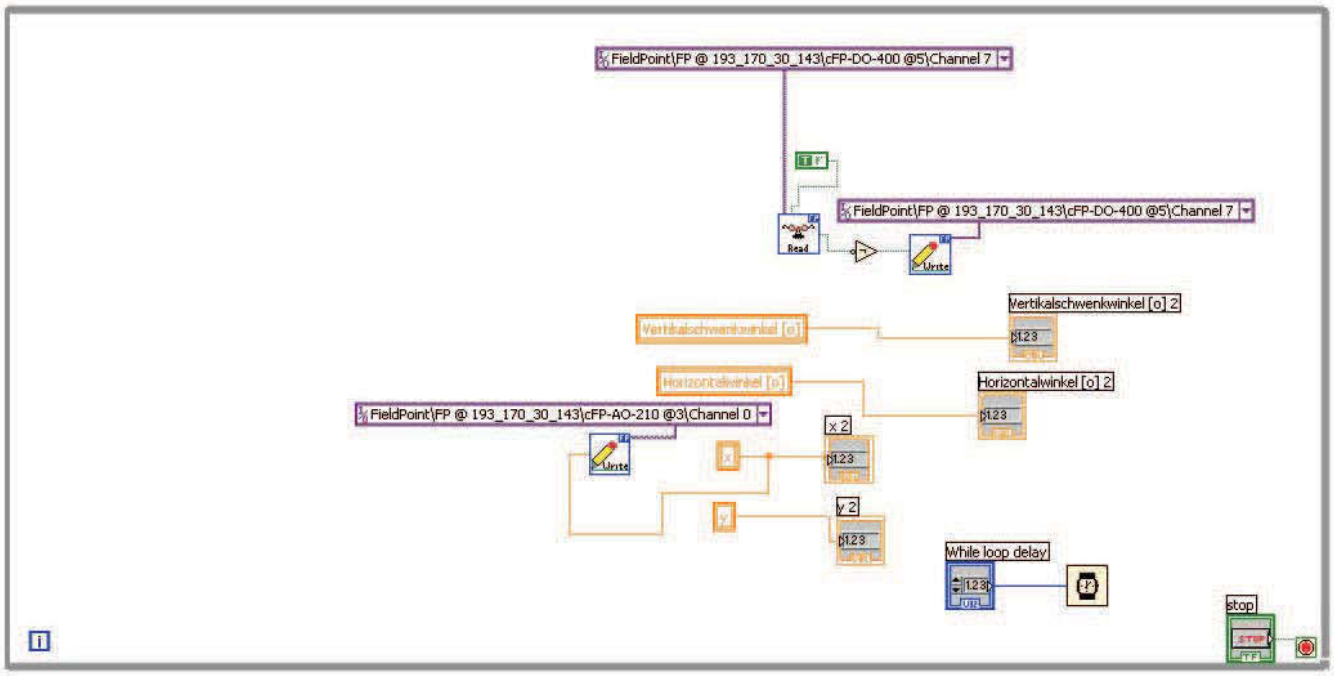

Figure 3.11: Data exchange between simulation loops and while loops

After building the simulation model, we need to run the ´RT communication wizard´, the window shown in Figure 3.12. The wizard removes controls in the top-level loop of VI and replace them with RT FIFOs. The Wizard generates a normal priority parent VI to access the RT FIFOs and share data with another TCP/IP, UDP, DataSocket, or Logos.

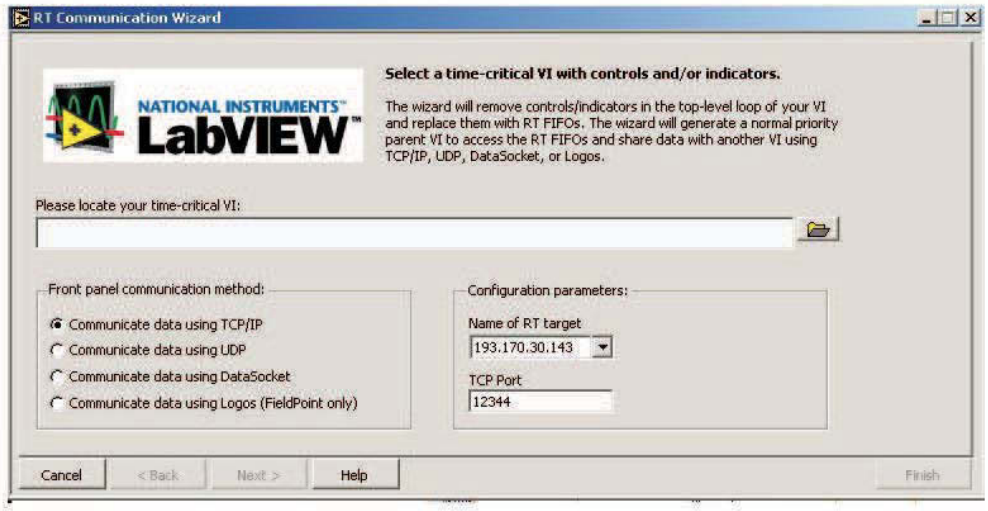

Figure 3.12: RT Communication Wizard

Following is the process of the wizard.
In the window ´RT communication wizard´, the communication paths, the RT targets and the time critical VI that will be simulated later should be selected, shown in Figure 3.12. At the window ´RTCW Loop Selector´, the while or the simulation loop is chosen to compile, shown in Figure 3.13. Like we introduced before, for the data exchange between this two loops, we will choose the while loop. At last we can choose the control and indicators (in/output), shown in Figure 3.14. After all this, there VIs (Virtual Instrument, a program part in LabView from National Instrument) will be generated automatically - Host VI, Time-Critical priority Loop (TCL) VI and Normal Priority Loop (NPL) VI. While the Host VI on the computer operates, TCL and NPL are operating on the processor.

Figure 3.13: Selection of loop

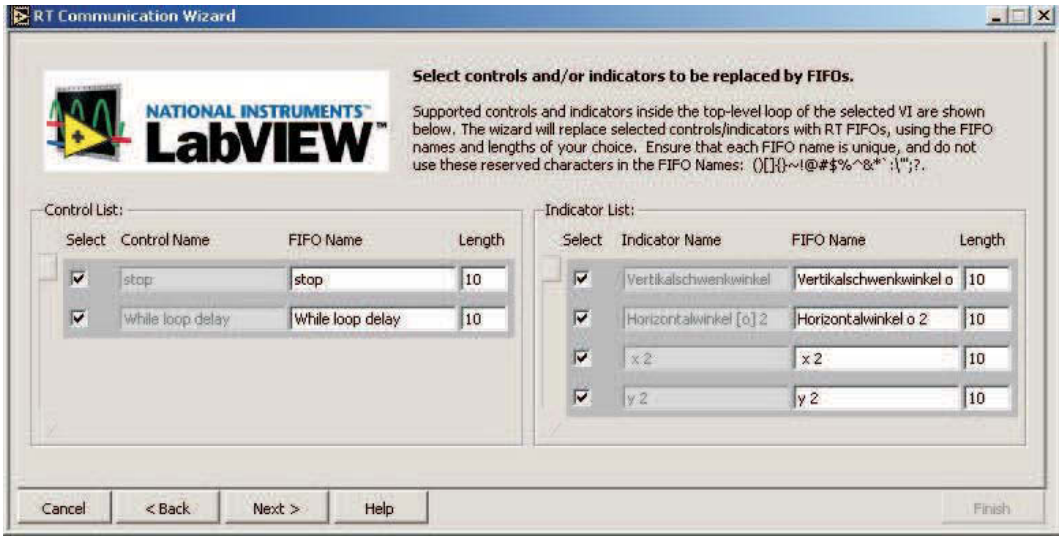

Figure 3.14: Select the control and indicator

There are three possibility to run a simulation:

1. The program is implemented on the PC and gets all outputs via the network from the FieldPoint hardware. With this the simulation is not hard real-time, since it is executed on the PC like a standard LabView application.

2. The program can be downloaded to the hardware and executed there. This program on the hardware contains also the routines for data exchange and communication with the PC, consequently there is no guarantee fro performance in real-time. And the result can be monitored on PC through Host VI. But after a power-down, the program on the hardware is lost and must be reloaded again.

3. The program can be implemented in setup.exe file that is started automatically after each power.up of the hardware. In this case ,the program is permanently installed and needs no further download from the PC. This is the way, how the FieldPoind hard- and software may be used as an autonomous real-time automation system.
   For the real-time simulation on the FieldPoint hardware, we need to download the program to it. In NPL VI, we need to do 'Build Application or Shared Library', shown in Figure 3.15. After all the application is built, the simulation program can run on the FieldPoint hardware, it is means the TCL and NPL VIs run on the processor. And if we want to monitor the program and result on computer, we can open the Host VI.
   The compiler generates three software objects from the project: A simulation loop running in real-time, further a communication process that exchanges data between the simulation and the PC via network. The simulation loop on the hardware is a separate task that does not directly communicate over the network, which maybe stalls the process execution (since TCP/IP is no real-time communication protocol). Finally, a LabView program running on the PC for visualisation is the third part of the project.

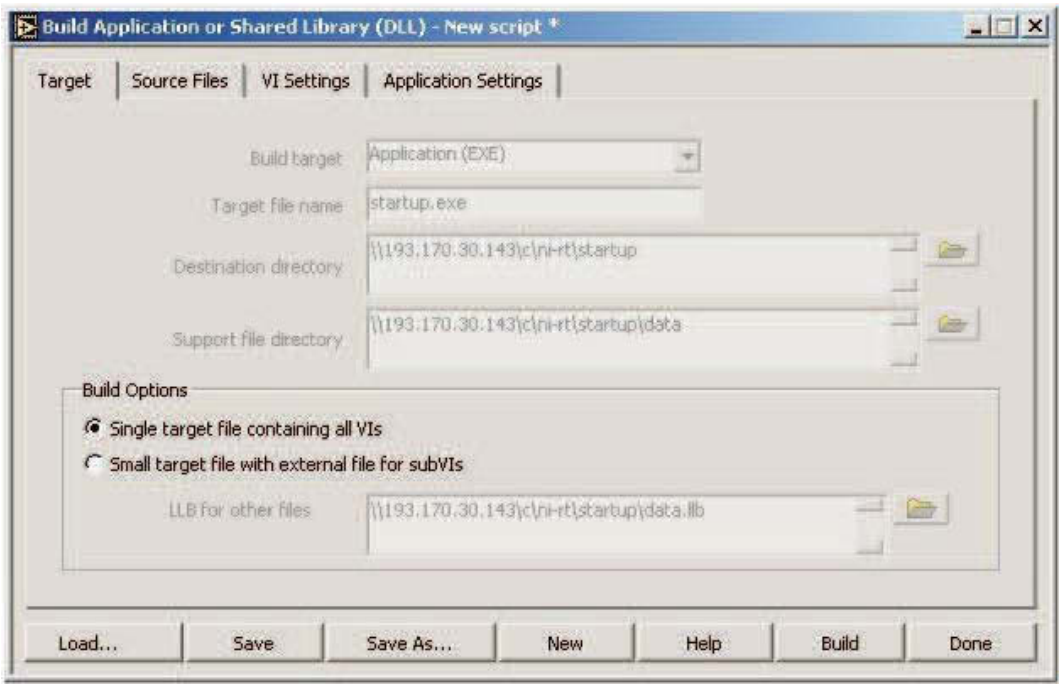

Figure 3.15: Build Application

3.4 Experiment Results

The simulation frame is a software loop dedicated for simulation blocks only. The properties of this frame determine the solver for the ODEs, further the simulation times, which has to be set to infinite for continuously running systems. With the experimental system in Figure 3.16, several experiments were done. An important test is the response time of the simulation. For this the waveform generator inputs a step signal to the analogue input. the reaction is monitor on an analogue output an oscilloscope, shown in Figure 3.17.
Increasing the complexity of the simulation loop leads to problems, for example an initial time delay of approximately four seconds, in Figure 3.17. It seems that the system has problems to update the inputs when the calculation needs to much time. Since there is not feedback possibility for the elapsed computing time in the simulation loop, this fact could not be proved. Further there were matching problems with the different releases of hard-, firm- and software. It was not possible to solve the problem until the end of this work.

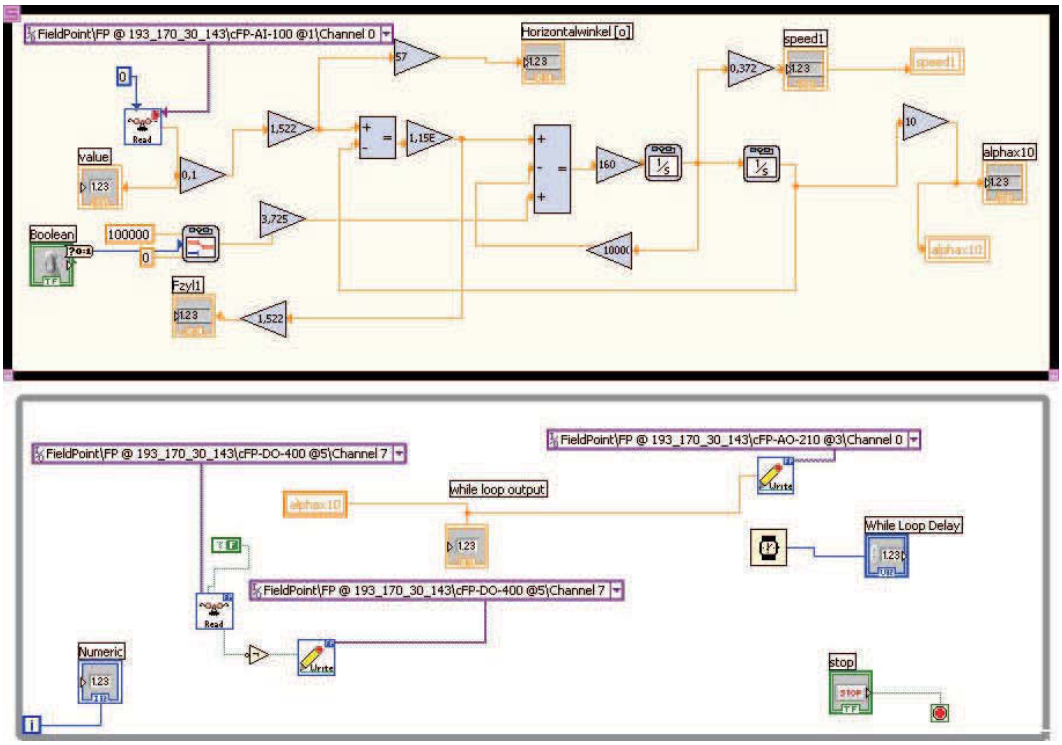

Figure 3.16: Test simulation model with LabView in horizontal direction

Figure 3.17: Delay in the simulation

3.5 Conclusion

The FieldPoint hardware and the Real-Time module of LabView should allow to run a real-time simulation on the hardware together with a comfortable visualisation and operation facility on the PC Unfortunately it was not possible to obtain a proper implementation due to version problems among the components until the end of this work.

Chapter 4

Hardware in the Loop Simulation with Matlab

In the previous chapter a simulation system was constructed based on the LabView environment. Alternatively a system was built and tested with Matlab/simulink.

4.1 Experiment Setup

We have used the hardware the Bernecker & Rainer (B&R) company and software of the Matlab/Simulink - Real-Time Workshop from the Mathworks company which were already presented in chapter 1.2.4. In Figure 4.1 the whole setup for this simulation. Like the simulation with LabView we also need a signal generator as input and a visualisatin was implemented.

4.2 Simulation Model design and Visualisation

4.2.1 Model Design

The simulation model is designed with the Matlab/Simulink programming environments. The code can be found in Appendix B. The data flow diagram in Figure 4.2 shown the process of the model building. The complete model is developed with the essential properties of the real system, kinematic, mechanical construction, and dynamic behavior of the hydraulic system which already were considered in detail in chapter 2.3.

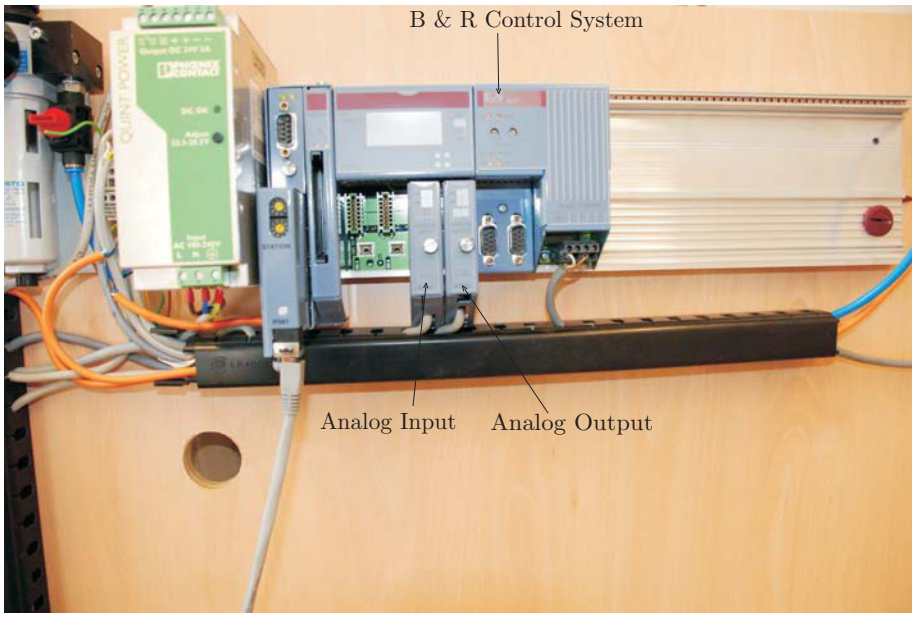

Figure 4.1: Experiment setup

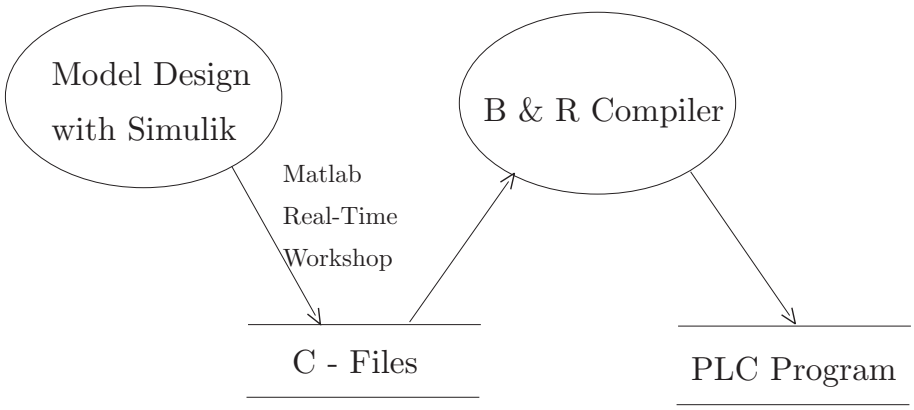

Figure 4.2: Data flow diagram of the model building procedure

After the model is written with Simulink, it should be run in real-time workshop, shown in Figure 4.3. As result we get some automatically generated files with C-language code. Then the C-lauguage code will be compiled with B & R compiler, so at last we will get the PLC program, shown in Figure 4.4.

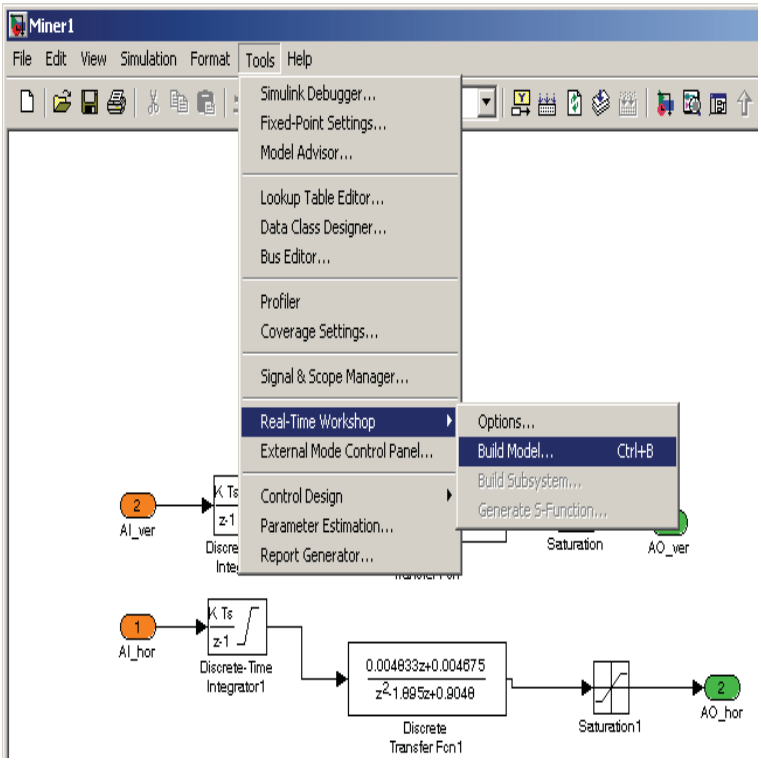

Figure 4.3: Real-Time Model build with Matlab

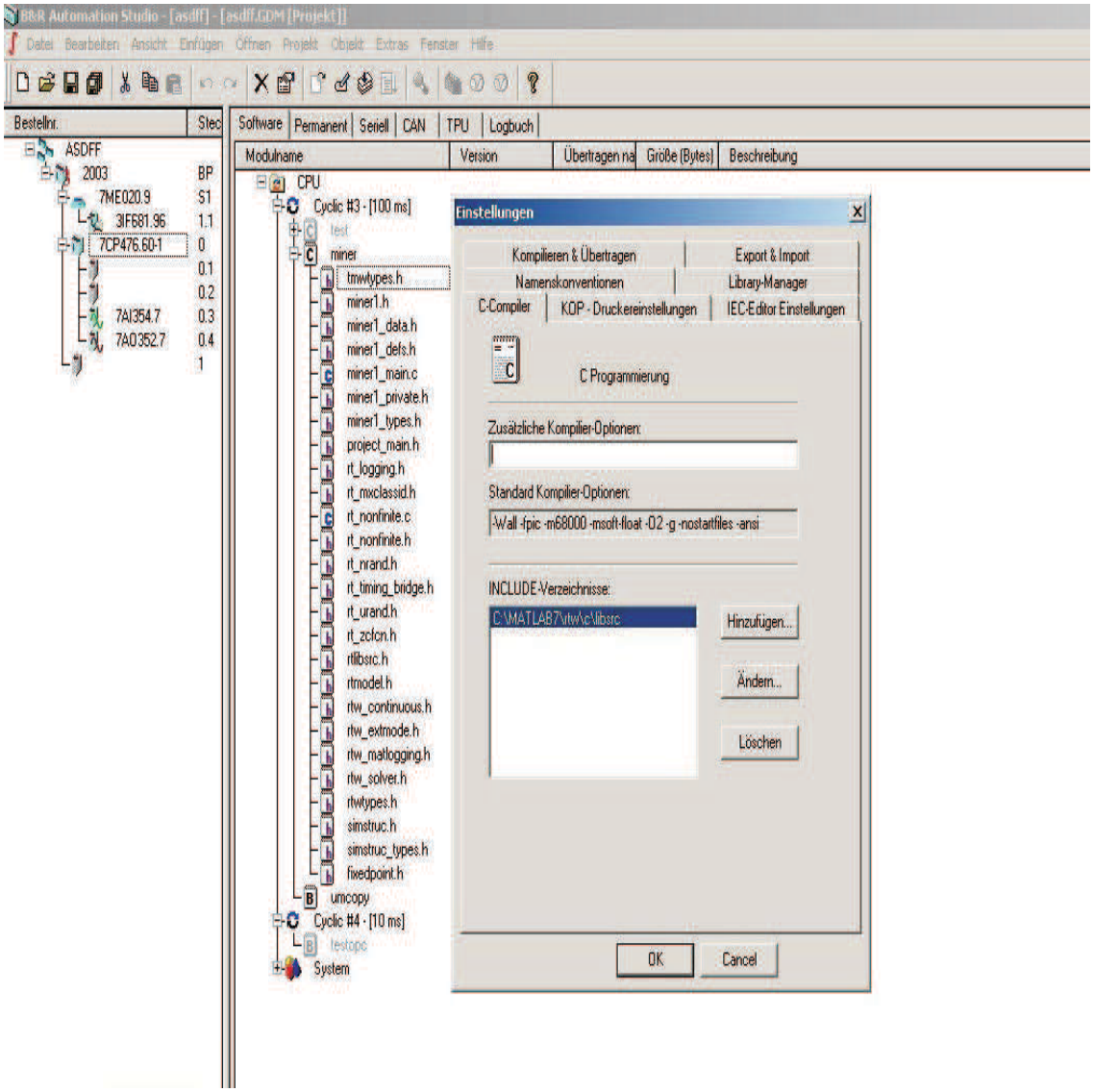

Figure 4.4: B& R Automation Studio and Adjustment

4.2.2 Visualisation

In the Figure 4.5 the data flow diagram of the process of visualisation is shown. For the visualisation we need first to draw one cutting arm model with the animation software Cinema - 4D, shown in Figure 4.6, and then create a video of it. This video contains the images for all possible positions: with the help of Macromedia Flash Professional a script program is added, shown in Figure 4.7, which will select the the image of the video to be displayed according to the machine arm coordinates. Then the publishing procedure generates a SWF file (shock wave) and some auxiliary html files for display in a Web-browser. The html source codes can be found in Appendix C.

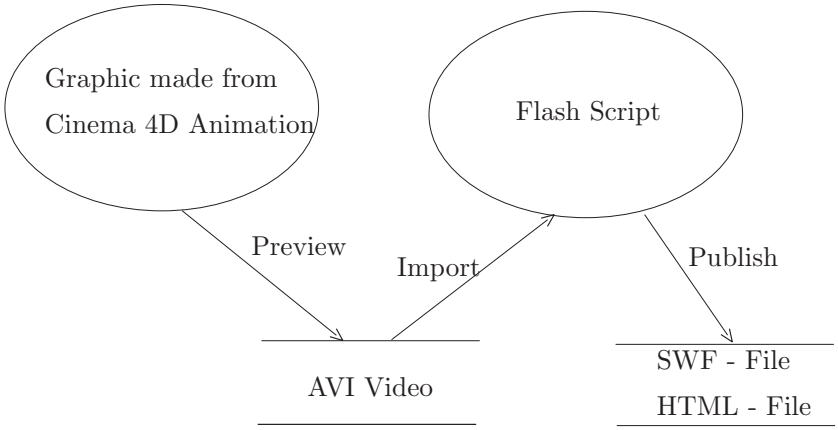

Figure 4.5: Data flow diagram of the Visualisation

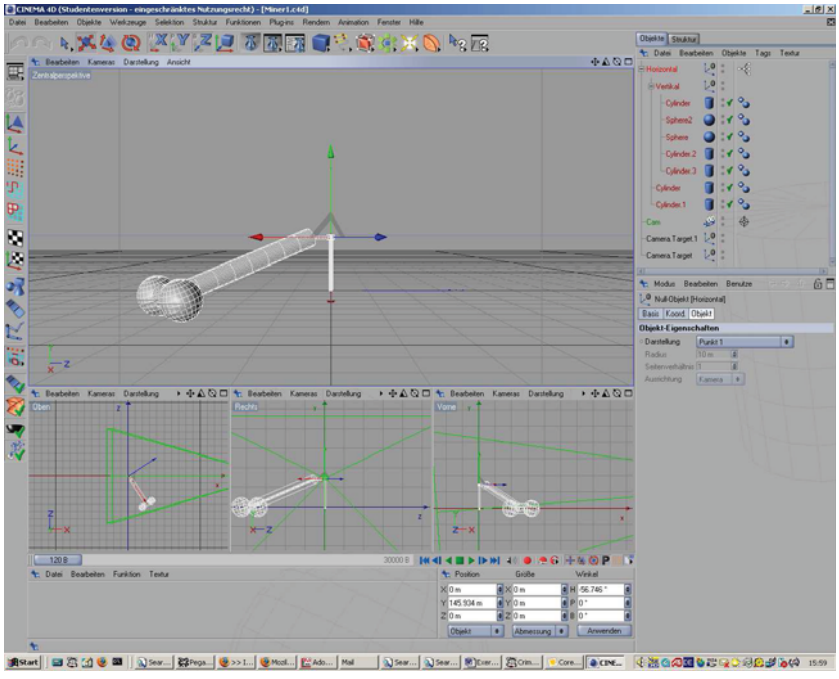

Figure 4.6: Cinema 4D model of cutting arm

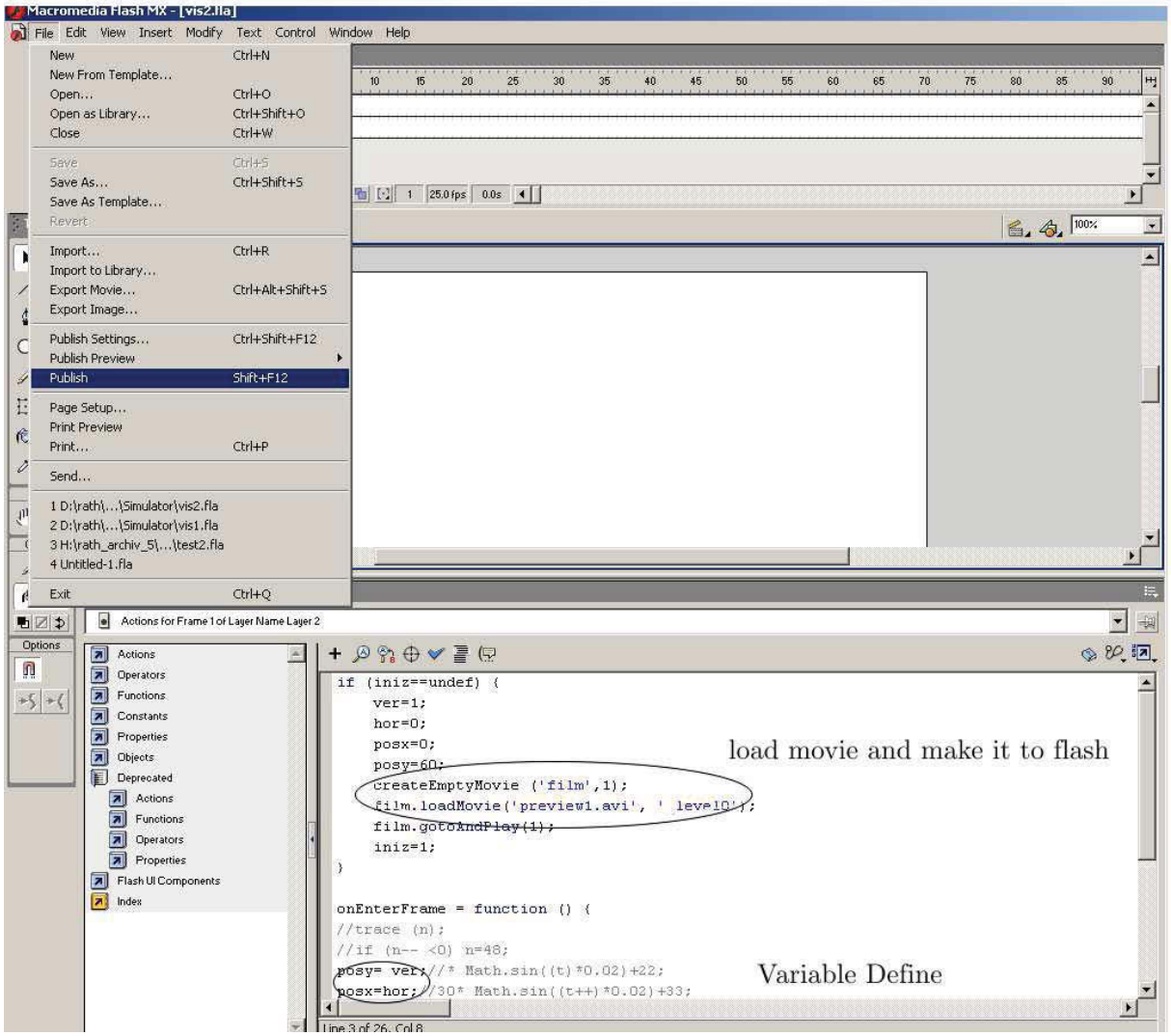

Figure 4.7: Flash Action Script for the cutting arm visualisation

4.3 Real-Time Process

The PLC program run on the B & R control system and perform the simulation in real-time. The results - x and y coordinates can be monitored on the computer. Through the B & R web server the data from PLc program - simulation results are sent to main html file to refresh the flash video with the new coordinates of the cutting arm. Through one browser we can monitor the flash video, which show the new position of the cutting arm every half second. In Figure 4.8 is the data flow diagram of the whole process. And in Figure 4.9 the web interface of the visualisation is shown.

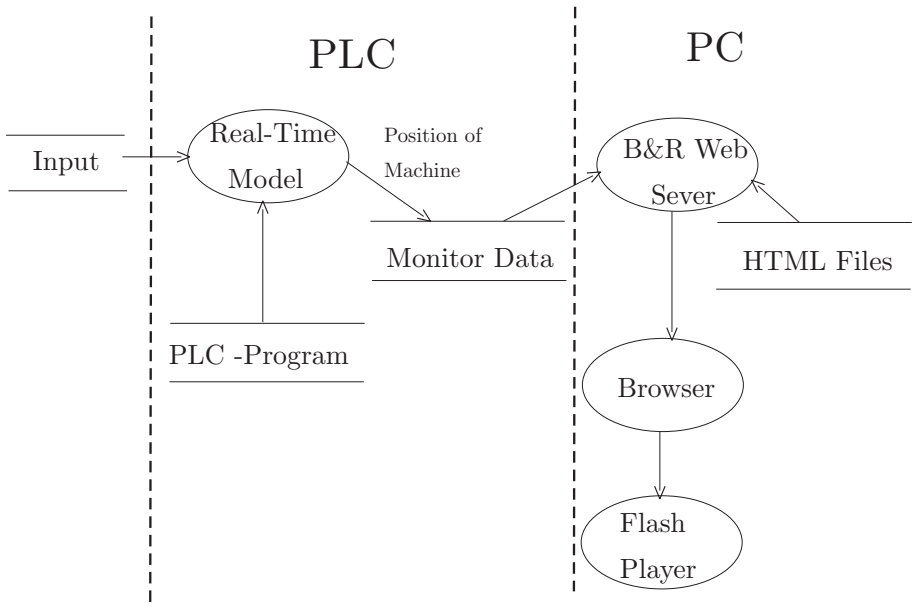

Figure 4.8: Data flow diagram of the teal-time process

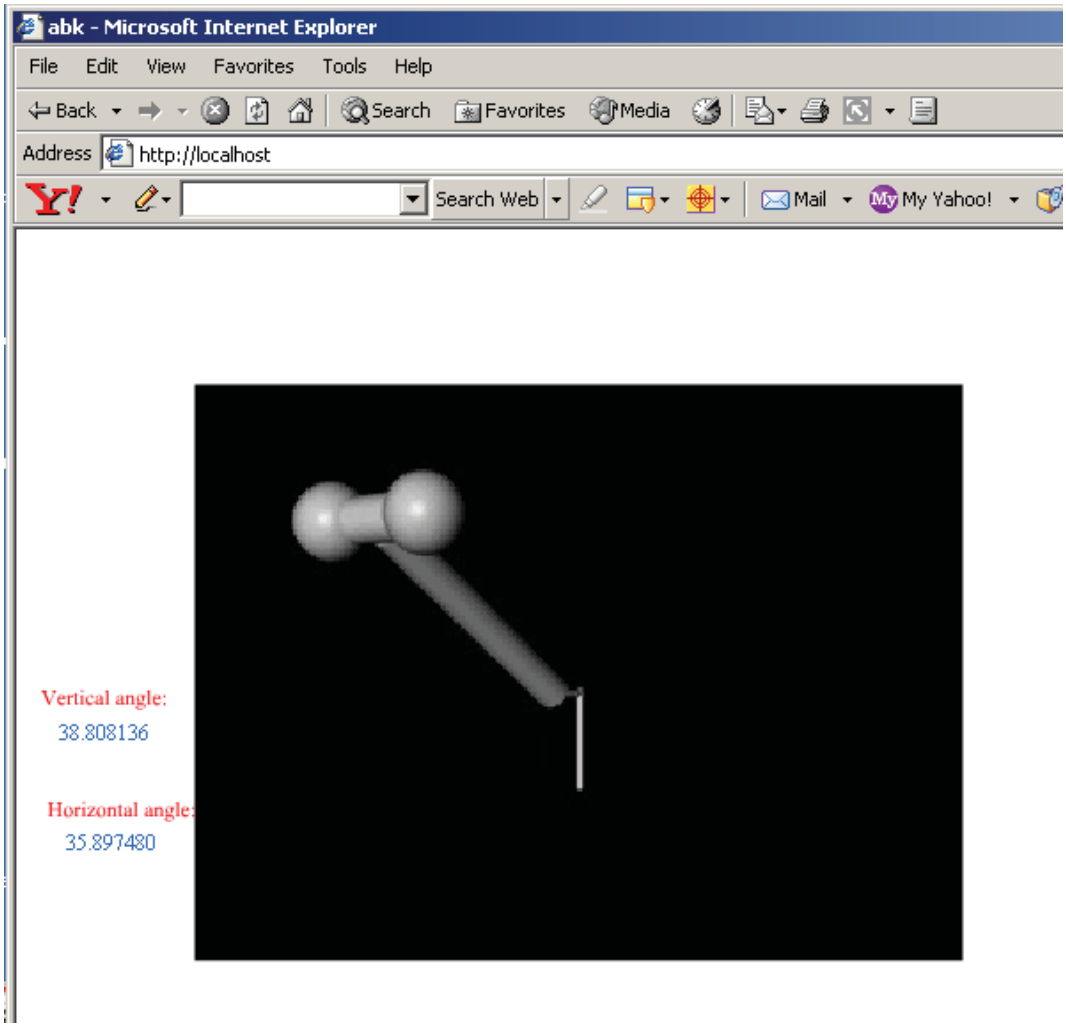

Figure 4.9: Visualisation screen at run-time

Chapter 5

Summary and Outlook

The aim of this work was a Real- Time Simulation, in this case a Hardware in the Loop Simulation for hydraulic control of mining machines and visualisation of these simulation models.
In the first chapter we introduced the Real - Time Simulation that the simulation of a component is performed such that the input and output signals show the same time-dependent values as the real, dynamically operating component. There are many reasons why we use Real - Time Simulation. It may be too complex and too expensive to research a real system, and it's may be too dangerous. In many cases the real system does not exist yet. It's much more easy to modify a simulation model than the real system and so on. Then we introduced the category of the Real - Time Simulation. There are three types of it – real process and simulated control system, simulated process and simulated control system,simulated process and real control system. The Hardware in the loop simulation is one of the Real - Time Simulation in which the process is simulated and the control system is real. In the HIL simulation you can use the real actuator, process and sensor or simulate them. For our model we simulate the process. Then we show some ways - the combination of the hard- and software to realize the HIL simulation. We used two ways of them for our simulation.
In the second chapter we introduced some details of the mining machine we need to simulate and the simulation details. The simulation model includes kinematics, dynamic behavior of the hydraulic system and the mechanical construction, further the cutting load force.
In the third and fourth chapter we program the simulation with two different function-block oriented, graphical programming environments – LabView and Matlab/simulink. After testing, we notice we failed to make a real-time capable model with LabView. And the model with Matlab running on an industrie control hardware from Bernecker & Rainer is ready for use. The final system will be applied to test the automation software with the aim of improving the cutting accuracy, before the machine prototype is available.

Acronyms and Abbreviations

ES ..... Embedded system
SUT ..... Embedded system under test
HIL ..... Hardware in the loop
NI ..... National Instruments
RTOS ..... Real-time operating system
GUIs ..... Graphical user interfaces
VI ..... Virtual Instrument
NPL ..... Normal Priority Loop
TCL ..... Time-Critical priority Loop
ODE ..... Ordinary differential equation
PFGA ..... field programmable gate array
PLC ..... programmable logic control
SPS ..... speicherprogammierbaren Steuerung

List of Figures

List of Tables

Bibliography

[1] G. K. A.M. van Tilborg. *Foundations of Real-Time Computing: Fromal Specifications and Methods*.

[2] D. G. B. Furht, D. Grostick and M. M. G. Rabbat, J. Parker. *Real-Time Unix Systems: Design and Application Guide*.

[3] B&R. Control system and industrie pcs. http://www.br-automation.com/cps/rde/xchg/br-productcatalogue/hs.xsl.

[4] A. A. R. Daniel J. Burns. *Hardware-in-the-Loop Control System Development using Matlab and xPC*. Department of Electrical Engineering, Center for System Science and Engineering, Arizona State University, Tempe, AZ 85287-7606, 2002.

[5] B. P. Douglass. *Real-Time Design Patterns: Robust Scalable Architecture fro Real-Time Systems*.

[6] D.Parnas. On the criteria to be Used in Decomposing systems into Modules . *Communications of the ACM*, 14:221–227, 1997.

[7] Dspace. Products: software and hardware. http://www.dspace.de/ww/de/gmb/home/products.cfm?nv=n2.

[8] A. M. Ewald Fauster, Florian Grün. Modellierung und Simulaiton der Postionskalibration einer Bergbaumaschine.

[9] R. Forsberg. *Cut Profile Accuracy of Roadheaders:Influence of Different Components and Measures for Improvements*. Master of science thesis, Stockholm, Sweden, 2005.

[10] D. L. S.-P. R. M. Goodall. Targeted Processing for Real-Time Embedded Mechatronic Systems. *Control Engineering Practice*, 15:363–375, 2007.

[11] N. Halbwachs. *Synchronous programming of Reactive Systems*.

[12] F. Haugen. Introduction to labview simulation module 1.0. http://techteach.no/publications/labview.

[13] N. Instruments. Real-Time hardware and Real-Time software. http://www.ni.com.

[14] S. S. Isermann R., Schaffnit J. Hardware-in-the-loop Simulation for the Design and Testing of Engine-Control Systems. *Control Engineering Practice*, 7:643–653, 1999.

[15] H. kopetz. *Real-Time System: Design Principles for Distributed Embedded Applications*. Kluwer Academic Publishers, boston/Dordredcht/London, 1997.

[16] P. Ledin Jim A. Hardware-in-the-loop Simulation . *Embedded Systems Conference*, Spring 2000.

[17] T. V. M. Linjama, J. L. J. Gustafsson, and M. K. V. Aaltonen. Hardware-in-the-Loop Environment for Servo System Controller Design, Tuning and Testing. *Microprocessors and Microsystems*, 24:13–21, 2000.

[18] S. P. M. Schiebe. *Real-Time Systems Engineering and Application*.

[19] MathWorks. Real-time workshop 6.6, 2007. http://www.mathworks.com/products/rtwt/index.html?ref=pfo.

[20] H. S. M.Husty, A.Karger and W. Steinhilper. Kinematik und Robotik.

[21] M. A. R. Mohamed E. Fayad, Louis J. Hawn and W.-T. T. Jay W. Schooley. *Hardware-in-the-Loop Simulation: An Application of Colbert's Object-Oriented Software Development Method*. McDonnell Douglas Missile System Company, P.O.Box 516, Saint Louis, MO 63166-0516, 1992.

[22] M. N. Nuksit Noomwongs, Hidehisa Yoshida and T. Y. Katsuhiro Kobayashi. Study on Handling and Stability Using tire Hardware-in-the-Loop Simulator. *JSAE Review*, 24:457–464, 2003.

[23] J. S. Pieter J. Mosterman, Gautam Biswas. A Hybrid Modeling and Verification Paradigm for Embedded Cotrol Systems. *Control Engineering Practice*, 6:511–521, 1998.

[24] D. K. R. Ewald, J. Hutter, A. S. F. Liedhegener, W. Schenkel, and M. Reik. *Proportional and Servo Valve Technology*.

[25] M. A. A. Sanvido. *Hardware-in-the-loop Simulation Framework*. doctoral thesis, 2002.

[26] S. W. Sanvido Marco A.A. Design of a Framework for Hardware in the loop Simulations and Its Application to a Model Helicopter.

[27] S. W. Sanvido Marco A.A, Cechticky Vaclav. Testing Embedded Control System Using Hardware-in-the-loop Simulationg and Temporal Logic. *Automatic Control Laboratory, Swiss Federal Institute of Technology*, 2000.

[28] B. Zupančič. Extension Software for Real-Time Control System Design and Implementatin with Matlab-Simulink. *Simulation Practice and Theory*, 6:703–719, 1998.

Appendix A

LabView Code

A.1 Whole Model

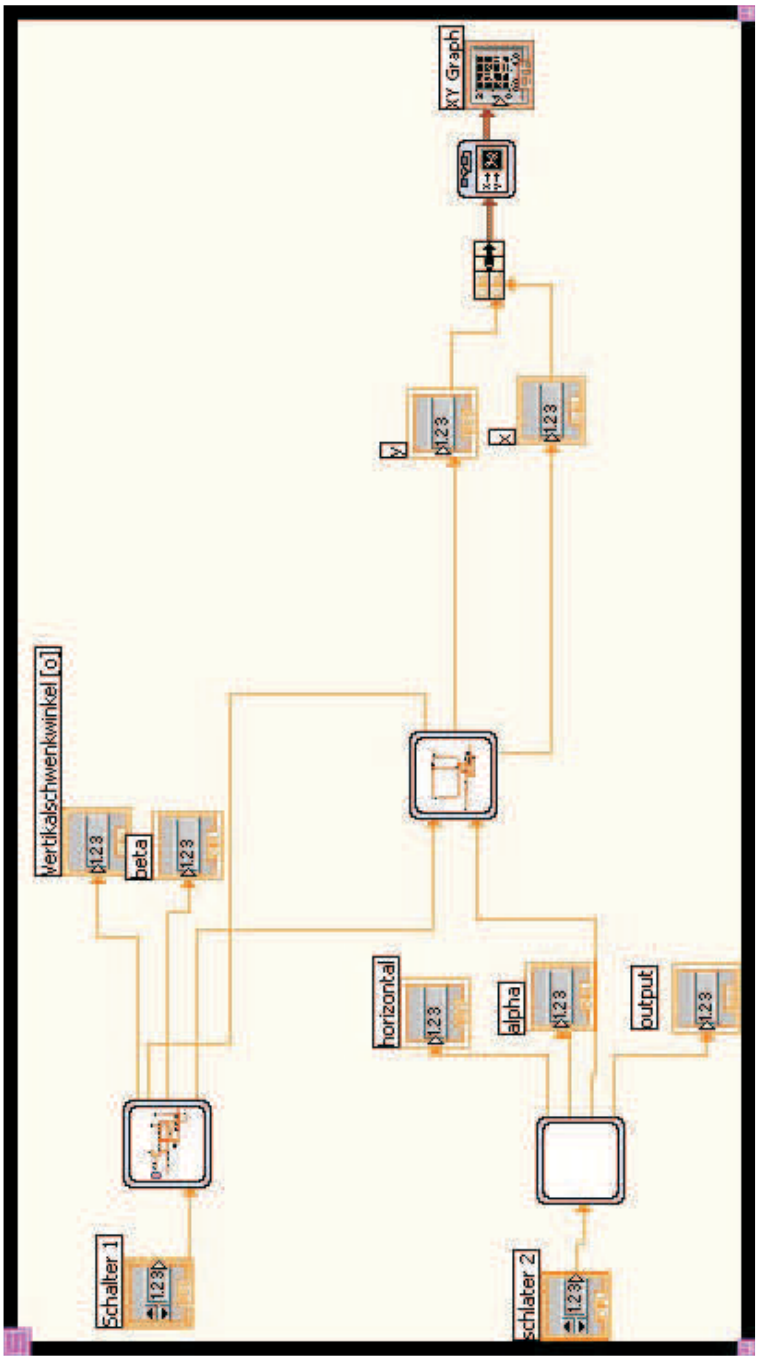

Figure A.1: Whole Model

A.2 Mechanical Construction in Horizontal Direction

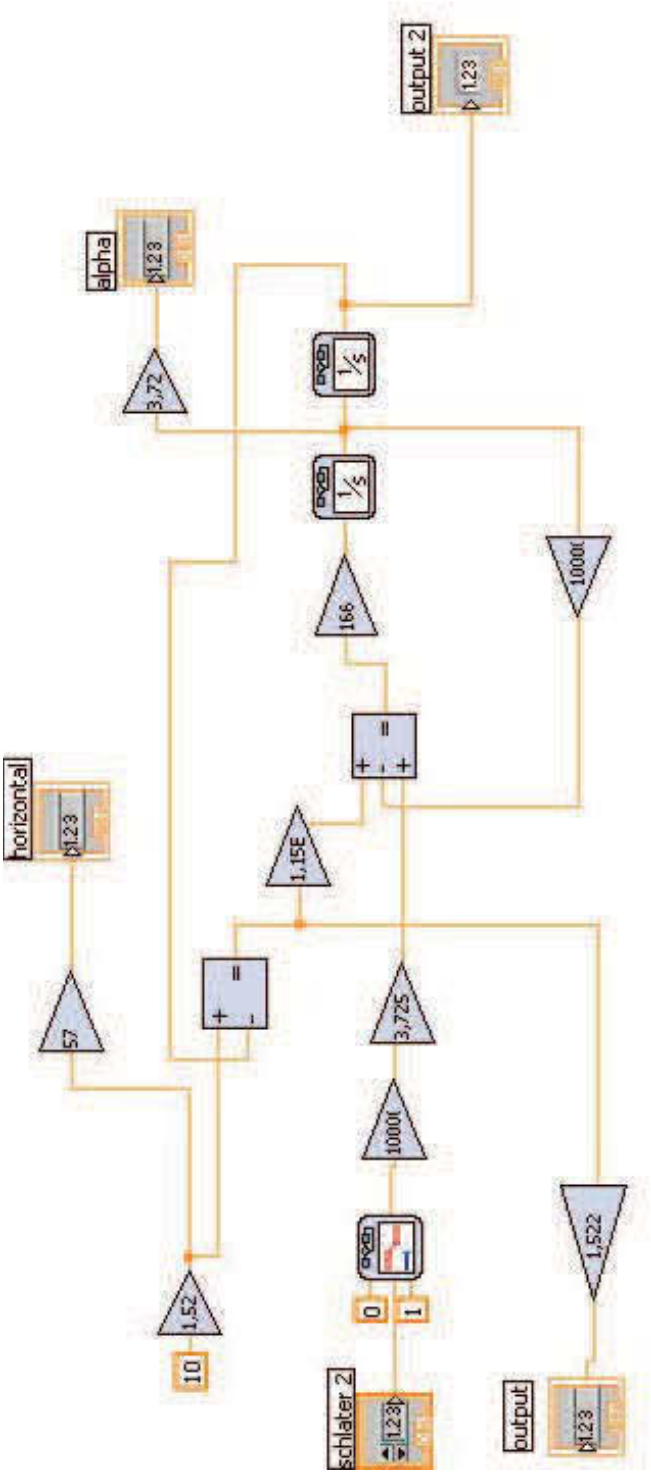

Figure A.2: Subsystem of Mechanical Construction in Horizontal Direction

A.3 Mechanical Construction in Vertical Direction

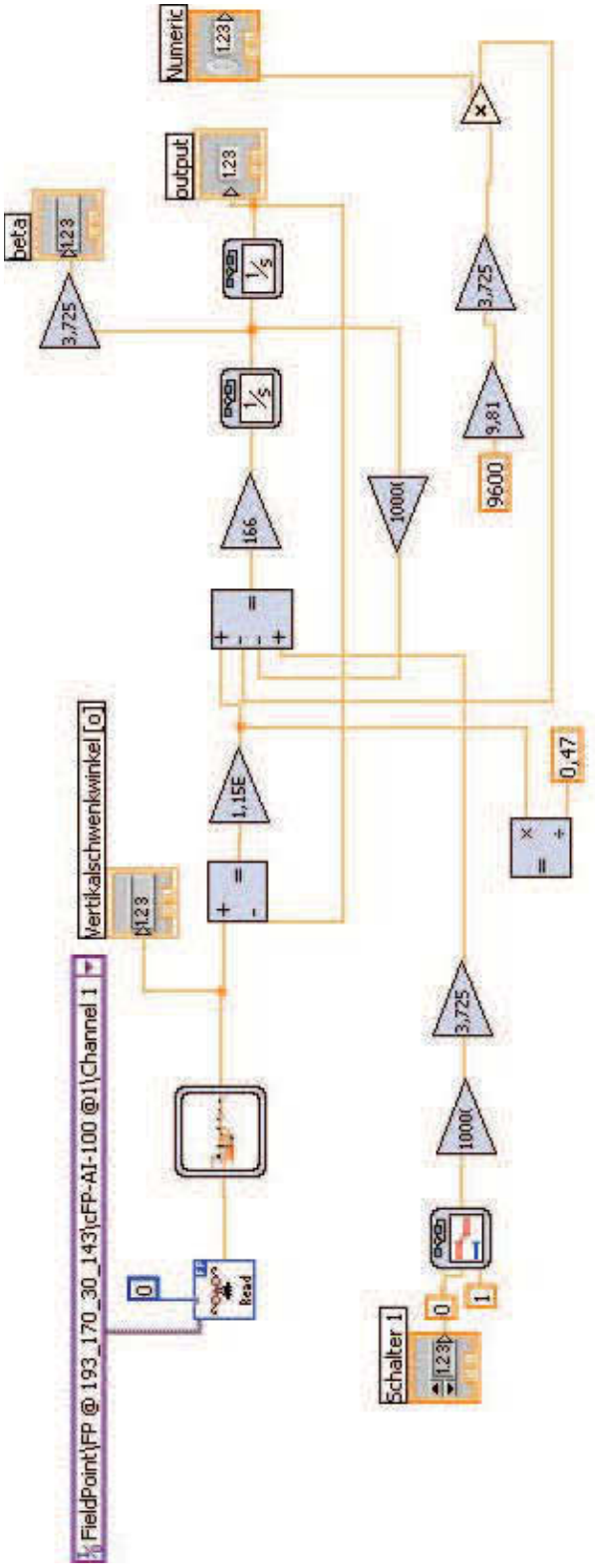

Figure A.3: Subsystem of Mechanical Construction in Vertical Direction

A.3.1 Law of Cosine

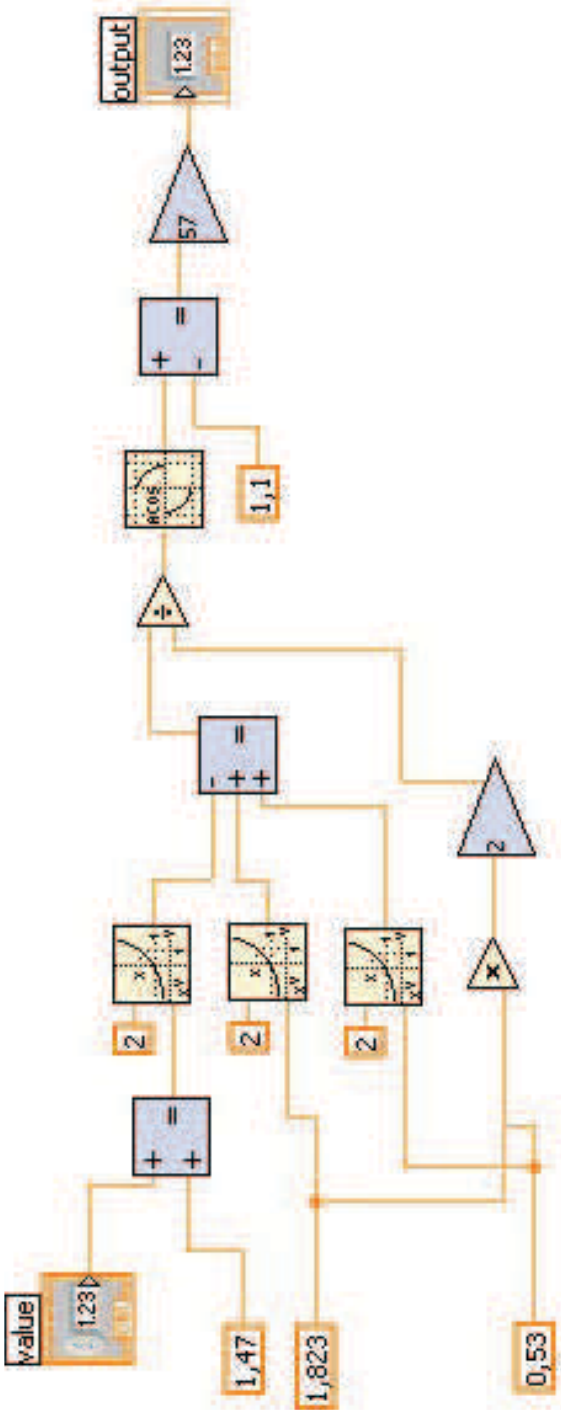

Figure A.4: Subsystem with Law of Cosine in the Vertical Direction

A.4 Kinematic Model

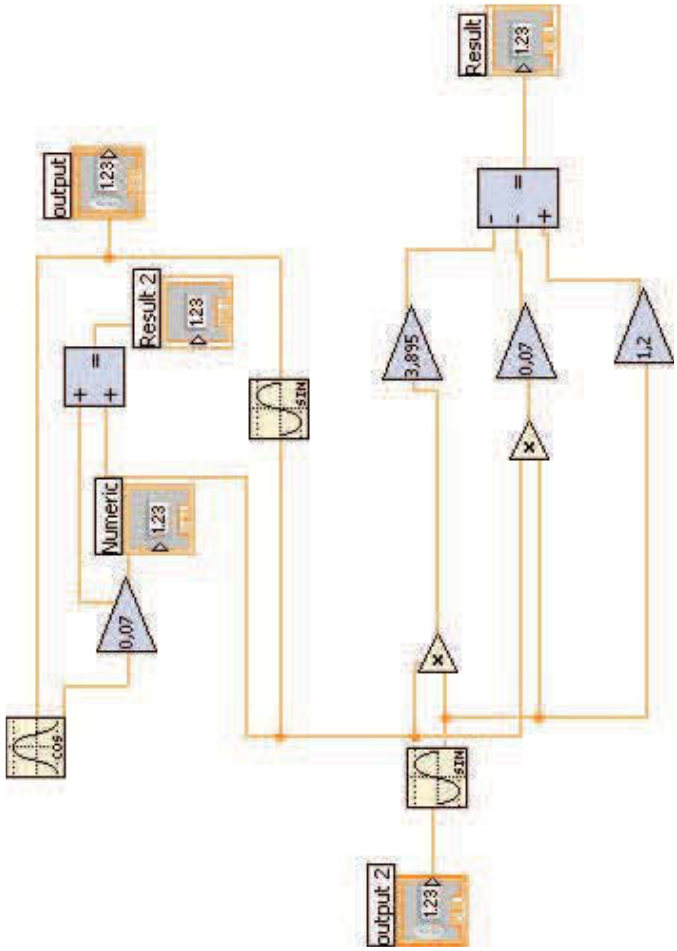

Figure A.5: Kinematic Model

Appendix B

Matlab Code

B.1 Whole Model

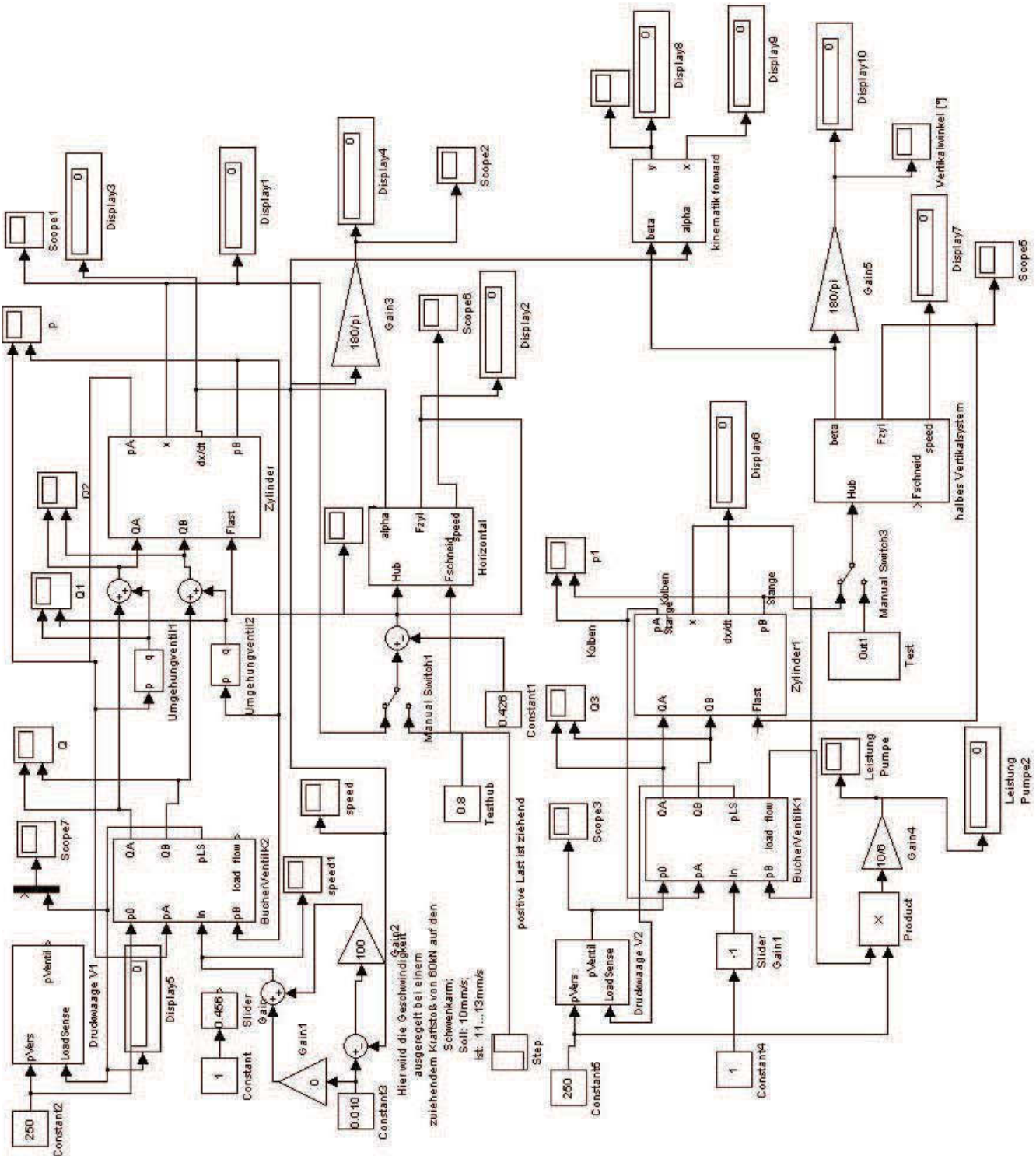

Figure B.1: Whole Model

B.2 Simulation Model in Vertical Direction

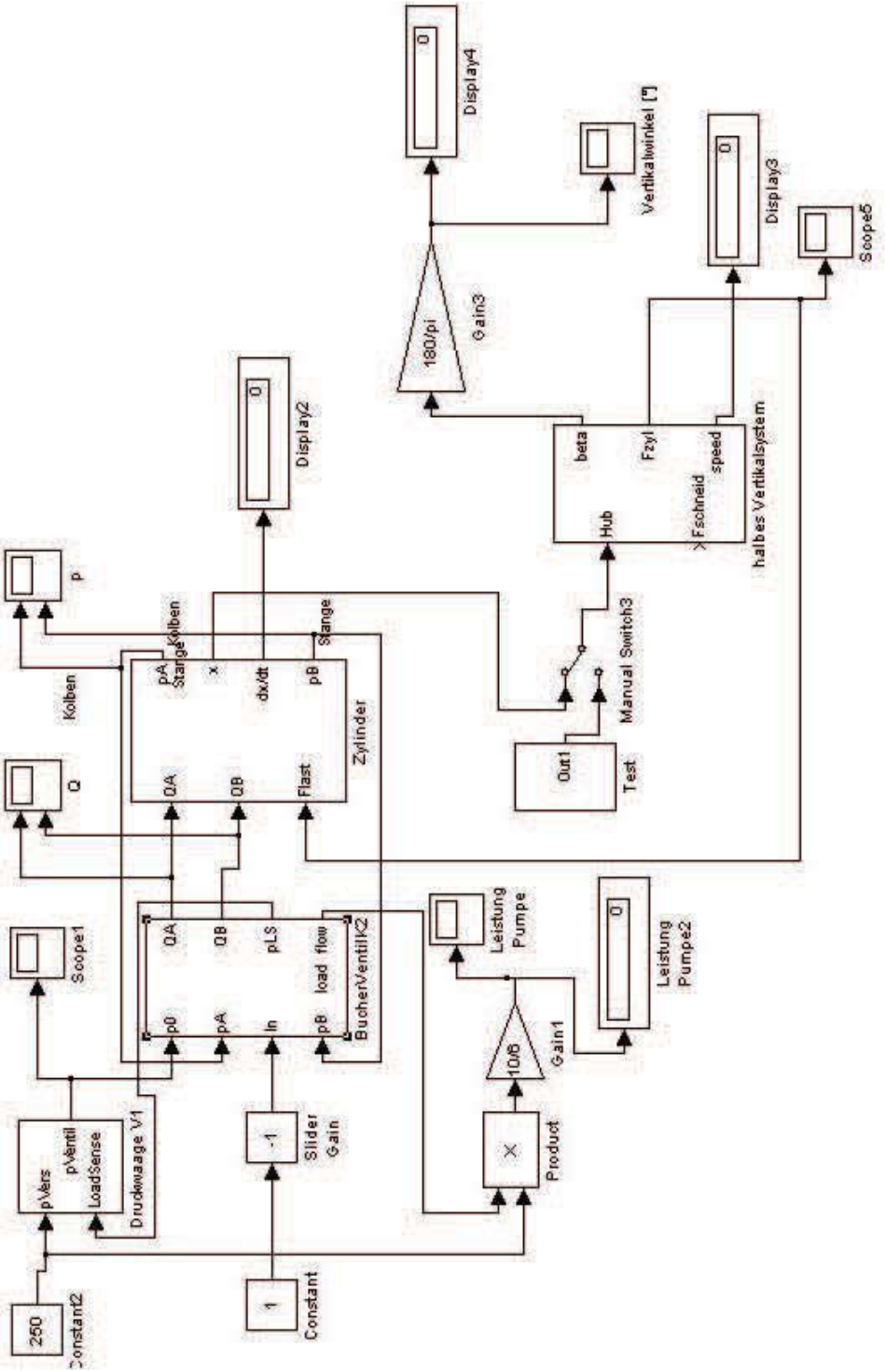

Figure B.2: Simulation Model in Vertical Direction

B.2.1 load Compensation in Vertical Direction

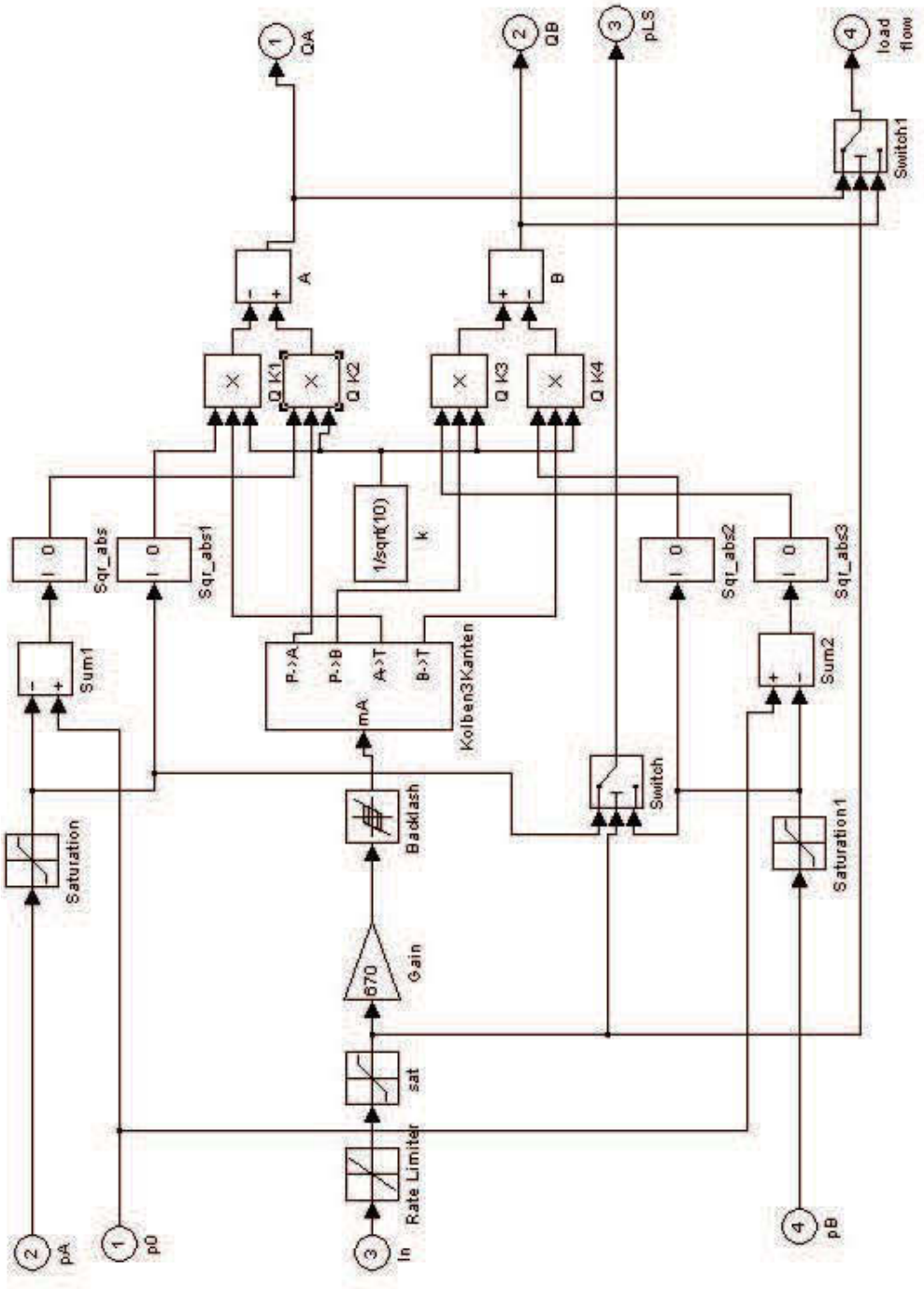

Figure B.3: load Compensation in Horizontal Direction

B.2.2 Valve in Vertical Direction

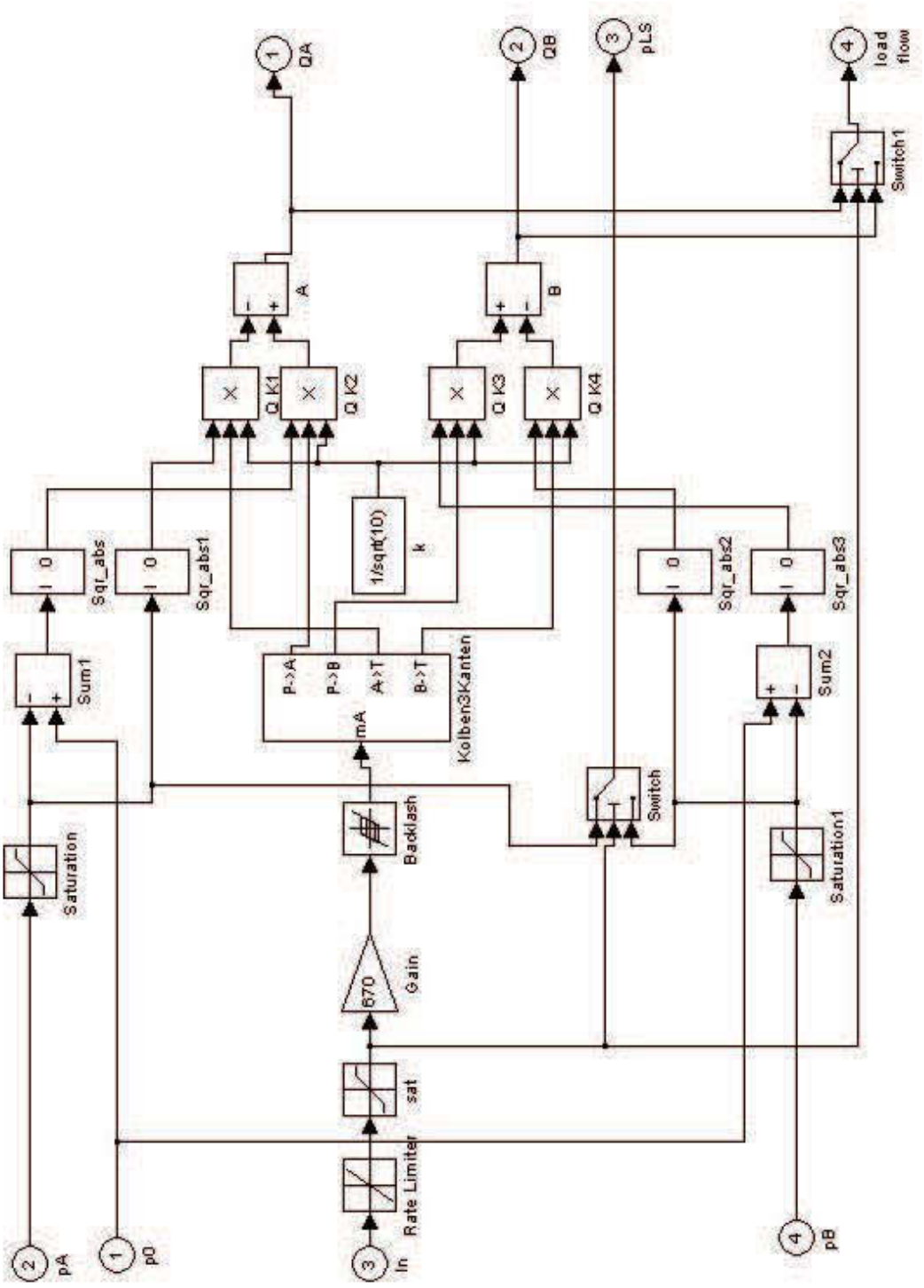

Figure B.4: Valve in Vertical Direction

B.2.3 Cylinder in Vertical Direction

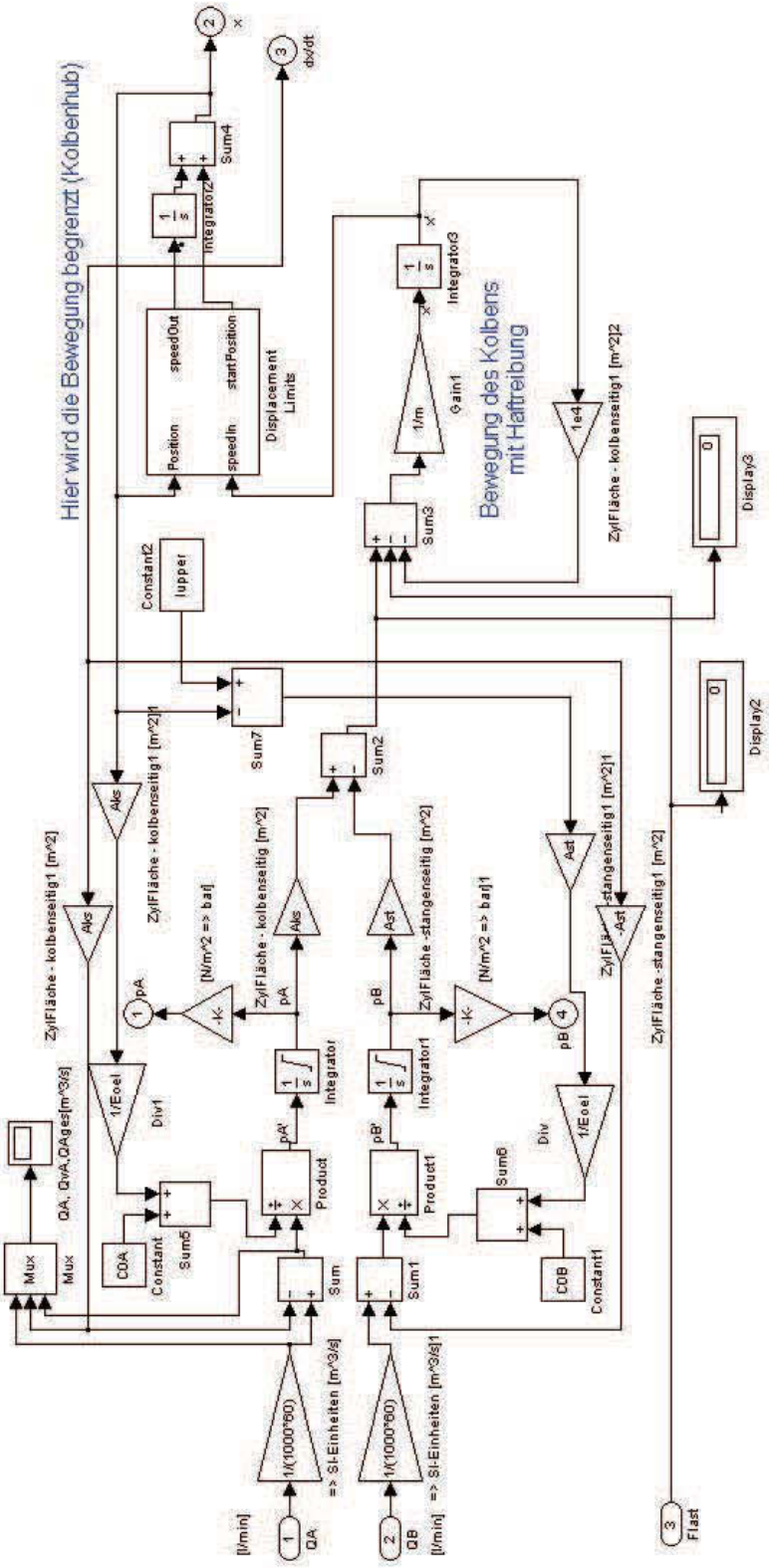

Figure B.5: Cylinder in Vertical Direction

B.2.4 Mechanic Construction in Vertical Direction

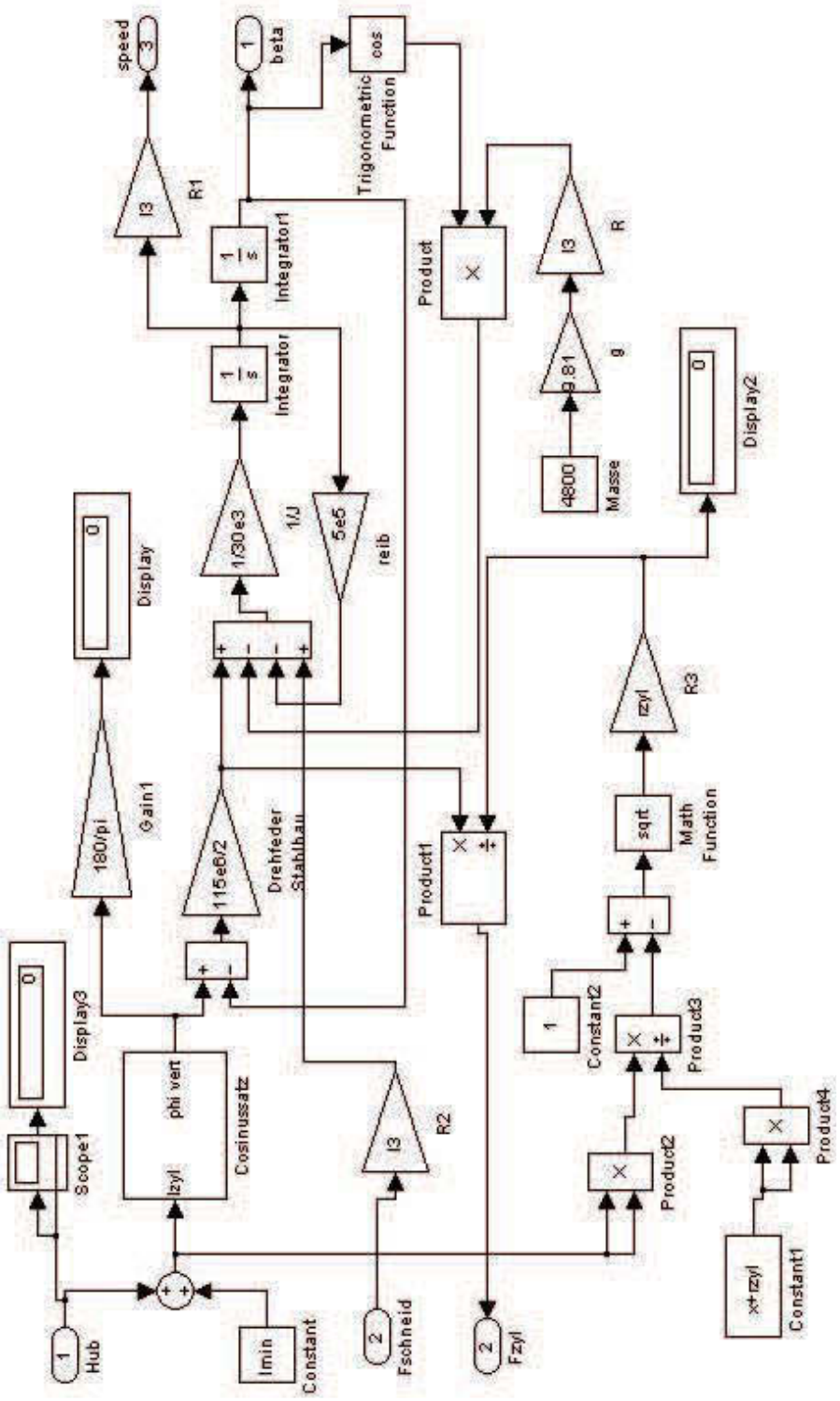

Figure B.6: Mechanic Construction in Vertical Direction

B.3 Simulation Model in Horizontal Direction

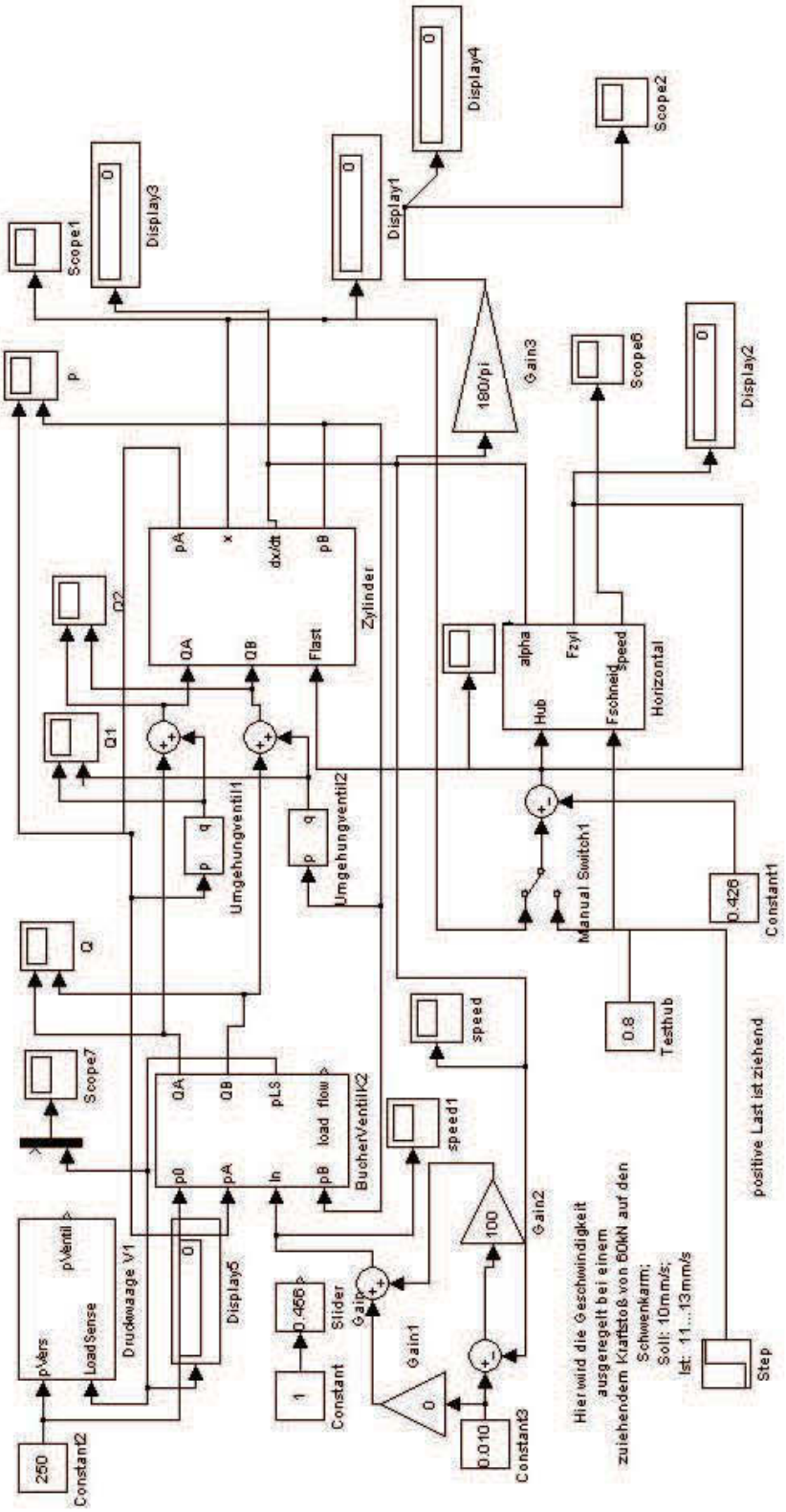

Figure B.7: Simulation Model in Horizontal Direction

B.3.1 load Compensation in Horizontal Direction

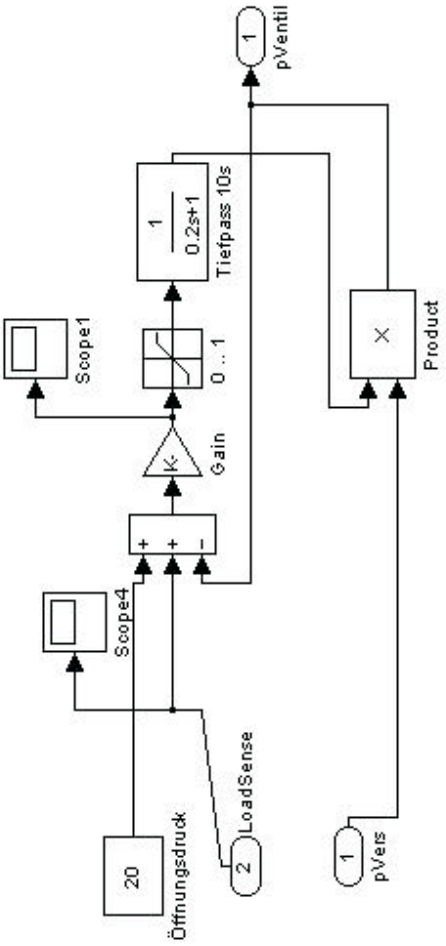

Figure B.8: load Compensation in Horizontal Direction

B.3.2 Valve in Horizontal Direction

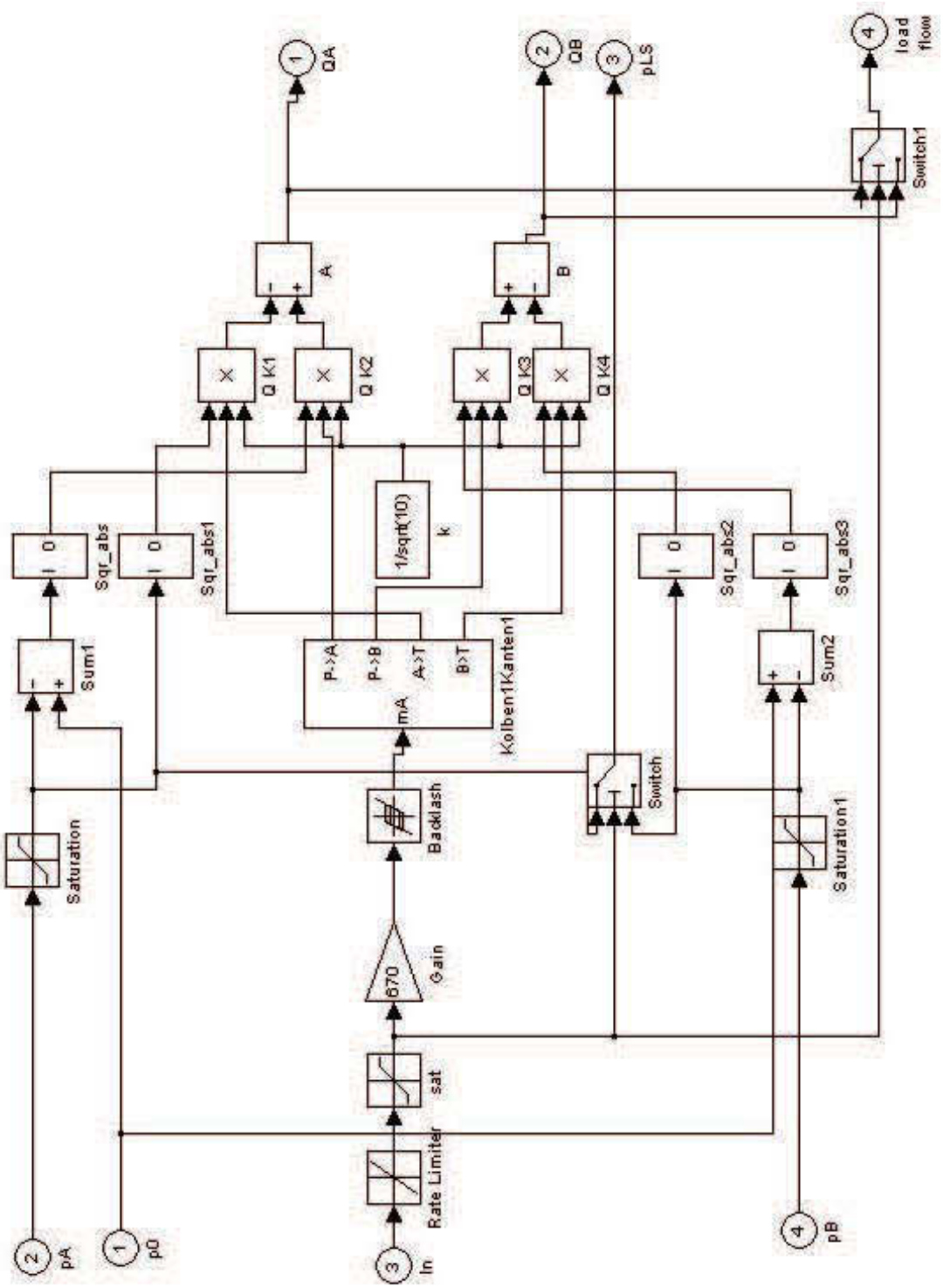

Figure B.9: Valve in Horizontal Direction

B.3.3 Cylinder in Horizontal Direction

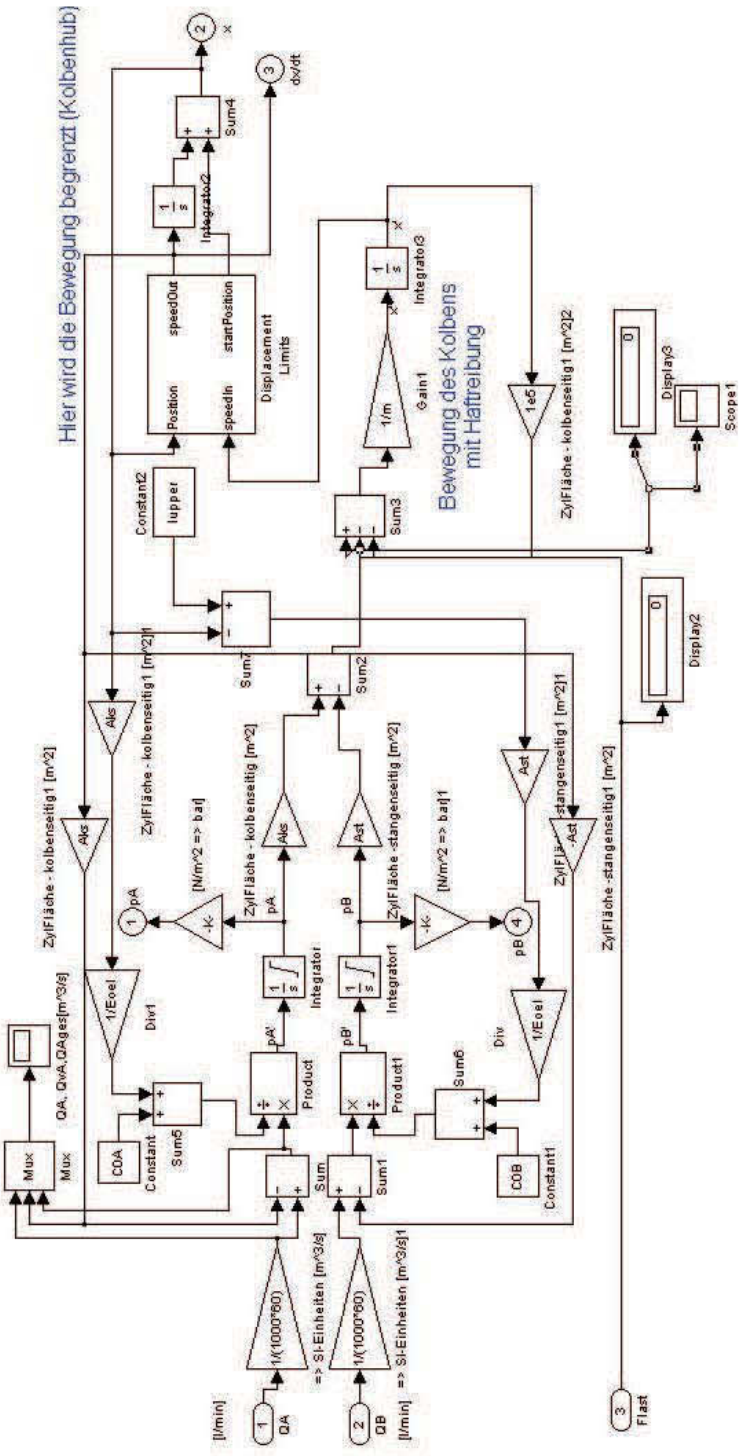

Figure B.10: Cylinder in Horizontal Direction

B.3.4 Mechanic Construction in Horizontal Direction

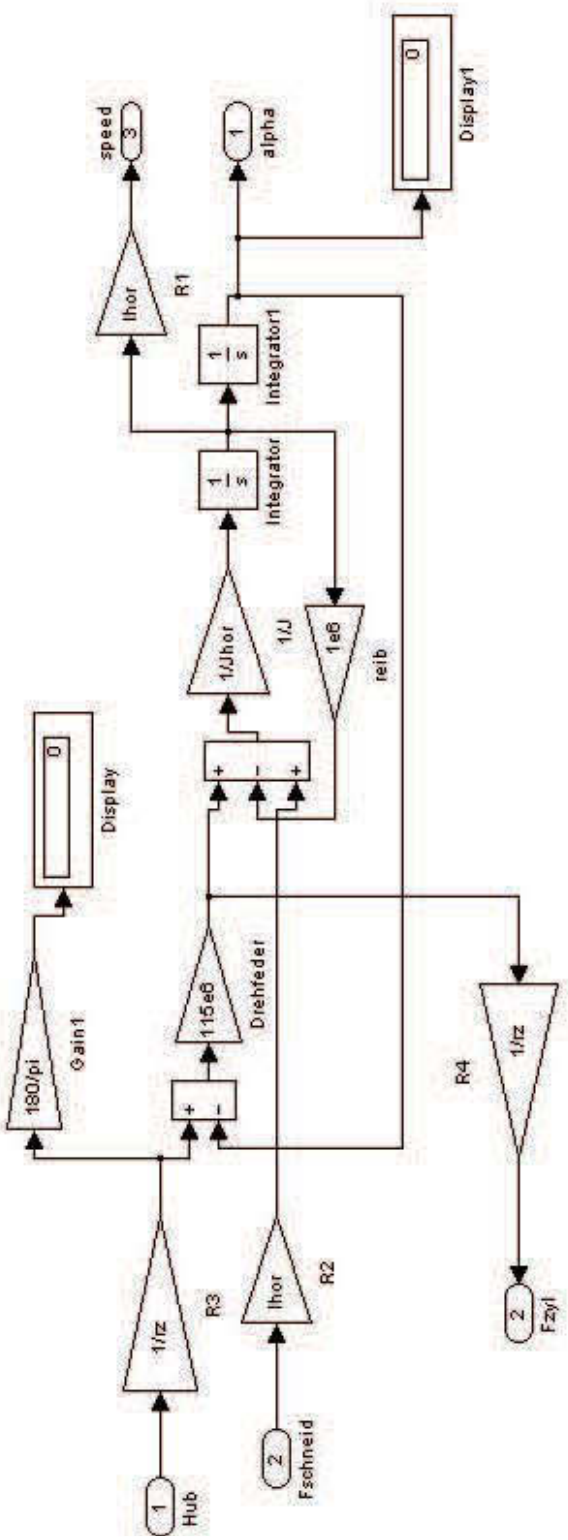

Figure B.11: Mechanic Construction in Horizontal Direction

B.4 Kinematic Model

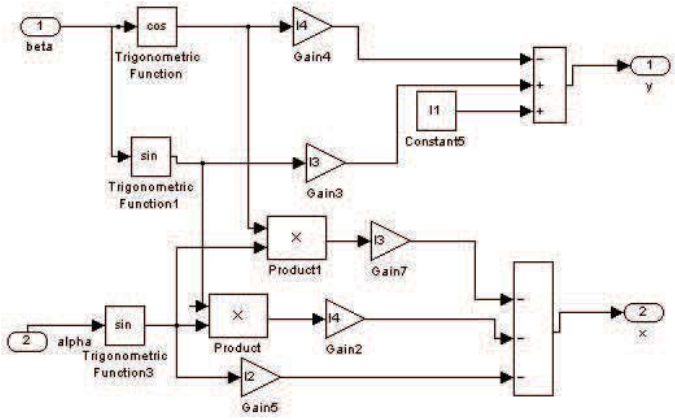

Figure B.12: Kinematic Model

Appendix C

HTML File

C.1 Main HTML File

```
<html>
<head>
<title>abk</title>

</head>
<frameset rows="500,*">

<frame name="banner3" scrolling="auto"  src="vis1.html">
 <frame name="banner" scrolling="auto"  src="test2AA.html">

 <noframes>
 <body>
  

<p>This page uses frames, but your browser doesn't support them.
</p>
</body>
</noframes>
</frameset>

</html>
```

C.2 HTML File of Flash

```
<HTML>
<HEAD>
<meta http-equiv=Content-Type content="text/html;
  charset=ISO-8859-1">

<TITLE>vis1</TITLE>
</HEAD>

<BODY bgcolor="#FFFFFF">
<!-- URL's used in the movie-->
<!-- text used in the movie-->
<OBJECT classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"
 codebase=
 "http://download.macromedia.com/
 pub/shockwave/cabs/flash/swflash.cab#version=6,0,0,0"
 WIDTH="550" HEIGHT="400" id="vis1" ALIGN="">
 <PARAM NAME=movie VALUE="vis1.swf">
 <PARAM NAME=quality VALUE=high>
 <PARAM NAME=bgcolor VALUE=#FFFFFF>
 <EMBED src="vis1.swf" quality=high bgcolor=#FFFFFF
  WIDTH="550" HEIGHT="400" NAME="vis1" ALIGN=""
 TYPE="application/x-shockwave-flash"
 PLUGINSPAGE="http://www.macromedia.com/go/getflashplayer">

</EMBED>
</OBJECT>
</BODY>
</HTML>
```

C.3 HTML File for catch X/Y Coordinates

```
<HTML>
<HEAD>
<meta http-equiv=Content-Type content="text/html;
 charset=ISO-8859-1">
<meta HTTP-EQUIV="REFRESH" CONTENT="2">
<TITLE>test22</TITLE>
<SCRIPT LANGUAGE='JAVASCRIPT'>
function mist(){
parent.frames[0].vis1.SetVariable
('ver','<PVI>
ReadVar @/Pvi/LNINA2/COM2/PVITEST/CPU/verti</PVI>');
parent.frames[0].vis1.SetVariable
('hor','<PVI>
ReadVar @/Pvi/LNINA2/COM2/PVITEST/CPU/hori</PVI>');
}
</SCRIPT>
</HEAD>
<BODY bgcolor="#FFFFFF" onload="mist()">

<p align="center"> </p>

<h2 align="center">Aktuelle Werte</h2>

<p align="center"> </p>
<div align="center"><center>

<table BORDER="1" CELLPADDING="3">
 <tr>
   <th>Variable</th>
   <th>Type</th>
   <th>Value</th>
 </tr>

 <tr>
   <td>Alpha</td>
   <td>[ ]</td>
   <td ID="uff" ALIGN="RIGHT">
   <PVI>
   ReadVar @/Pvi/LNINA2/COM2/PVITEST/CPU/verti
   </PVI></td>
  </tr>

  <tr>
   <td>Beta</td>
```

```
    <td>[mV]</td>
    <td ALIGN="RIGHT">
    <PVI>
    ReadVar @/Pvi/LNINA2/COM2/PVITEST/CPU/hori
    </PVI></td>
   </tr>

</table>
</center></div>
<p align="center"><br>

</BODY>
</HTML>
```