ISSN 0280-5316
ISRN LUTFD2/TFRT--5822--SE

Control of an Autonomous Wheel Loader

Jonatan Björkman

Department of Automatic Control
Lund University
September 2008

Lund University
Department of Automatic Control
Box 118
SE-221 00 Lund Sweden

Document name
MASTER THESIS

Date of issue
September 2008

Document Number
ISRN LUTFD2/TFRT--5822--SE

Author(s)
Jonatan Björkman

Supervisor
Torbjörn Martinsson Volvo, Eskilstuna
Karl-Erik Årzén Automatic Control, Lund (Examiner)

Sponsoring organization

Title and subtitle
Control of an Autonomous Wheel Loader (Reglering av en autonom hjullastare)

Abstract
This is the report of a thesis work about controlling an autonomous wheel loader. The objective of the work was to create a complete prototype of an autonomous wheel loader that can operate without any seated driver and without any kind of remote controlling. Another important objective was that this prototype needs to support future development of the autonomous wheel loader.
The work was completed within the time frame and a structured method of development was used to develop the prototype machine. The work began with an investigation about what is needed on an autonomous wheel loader and why. This helped improve the support of future development of the prototype wheel loader. After the investigation was done a design was made about how to control the wheel loader autonomously. This design included which computers to use and how to integrate the new computers with the existing system on the wheel loader. When the design was finished the implementation began, which included programming, Simulink model design, installing new systems on the prototype wheel loader etc.
Lastly everything that was designed and implemented was verified. The prototype of the autonomous wheel loader was demonstrated with a short cycle while using all its functions as a wheel loader. Since the autonomous functions had been implemented the demonstration was very easy to implement by simply programming it to use its different functions in order.

Keywords

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title
0280-5316

ISBN

Language
English

Number of pages
41

Recipient's notes

Security classification

Acknowledgements

First of all I would like to thank Torbjörn Martinsson at the Hauler Loader Division, Volvo Construction Equipment in Eskilstuna, and Professor Karl-Erik Årzén at the Department of Automatic Control, the Faculty of Engineering at Lund University, for giving me this opportunity to do this thesis work at Volvo Construction Equipment in Eskilstuna.

I also express my gratitude to all the employees that gave helpful advice in Volvo Construction Equipment in Eskilstuna, especially Johan Kernell for enduring many questions and my supervisor Torbjörn Martinsson for providing excellent guidance, experience and insight.

I would like to thank Wang Qi for always being there for me.

Also thanks to David Gunstad and Peter Alnefeldt who made my travelling time so much more comfortable.

And lastly I thank all my family and friends for supporting me and my work.

Table of Contents

1 Introduction

This is the report of the thesis work carried out by the author for the Faculty of Engineering at Lund University. The work was done at Volvo Construction Equipment in Eskilstuna from the middle of February 2008 until the end of June 2008. In short, the work is about making a wheel loader being able to work autonomously. In Section 1.1 the outline of the entire report can be found. More introduction and background information can be found further down in the first section of the report.

1.1 Outline of Thesis

Section one in the report is the introduction. There information can be found such as information about the company, about the machines and about the project. Section two of the report discusses the objective of the thesis. Section three describes the method with which the work was carried out. Section four, five, six and seven describes the different working phases that are introduced in Section three that describes the working procedure. These four sections from fourth to seventh are divided into the investigation phase, the design phase, the implementation phase and the verification phase. More information about the phases and the working procedure is found in their respective sections. Section eight is the last section and discusses conclusions and further work in the project.

1.2 Background

In this part of the report background information that does not directly influence the work is discussed. This includes information about the company where the work was done and information about the involved machines.

1.2.1 Volvo Construction Equipment

Volvo Construction Equipment is a part of the Volvo Group that develops construction equipment. They have more than 170 different models of wheel loaders, haulers, diggers and graders etc. In Eskilstuna Volvo CE has several thousand employees and are mostly focused on development of wheel loaders and haulers.

The Hauler Loader Division is the division within Volvo Construction Equipment that takes care of research and development of both the wheel loaders and the haulers. It is within this division in Eskilstuna that the thesis work presented in the report has been done. Brief explanations of the wheel loaders and the haulers are found in the sections below.

1.2.2 Wheel Loaders

A wheel loader is a vehicle mainly used for loading goods with the help of the tool on the front of the machine. For one who is not used to the branch, the wheel loader could be considered as “one of those big yellow vehicles used in construction sites”. But there are indeed differences between these vehicles, and the wheel loaders have specific features such as the arm with the bucket used for loading and an articulated steering. Figure 1 below shows a picture of the wheel loader model used in the thesis work.

Figure 1: Picture of the L120F wheel loader model.

The wheel loaders developed by Volvo CE are all using the steering mechanism referred to as “articulated steering”. With the articulated steering the loader is steering around a fixed pivot point exactly between the front and rear axis. Figure 2 below shows a picture of the pivot point. With the steering around the pivot point the wheel loader becomes more flexible, for example the driver can steer the load in the bucket along an arc. The articulated steering also improves stability and strength since the front axis can be solid. This is important since lifting a heavy load in the bucket will cause a heavy load on the front axis.

Figure 2: Picture of the steering pivot point between the front and rear axis.

The Volvo wheel loaders are produced in many different sizes and models, ranging from the smaller L50 with a mass between 8 and 9 tons to the giant L350 with a mass between 50 and 56 tons (Volvo Construction Equipment 2005a, Volvo Construction Equipment 2007). What they all have in common is the strength and flexibility which makes them popular. The same machine can be used for loading cargo on pallets, loading gravel from a pile or loading logs with a simple change of the front-end tool. Other common features of the Volvo wheel loaders are the "four-wheel drive" and the tilt able rear axis which allows for more flexible driving.

1.2.3 Haulers

Although the work described in this report does not directly involve a hauler, it is mentioned in the investigation section and is therefore worth mentioning. The haulers are also part of "one of those big yellow machines used in construction sites" and are used to haul cargo. A picture of a hauler can be found in Figure 3 below. Similarly to the wheel loaders, the haulers use the "articulated steering" mechanism with a fixed pivot point for steering. Together with a "six-wheel drive" they are designed to be able to traverse difficult terrain (Volvo Construction Equipment 2005b). It is also worth mentioning that the haulers are often used in conjunction with the wheel loaders, with the wheel loaders for loading the haulers and the haulers for hauling.

Figure 3: Picture of a hauler.

1.3 The Autonomous Machine Project

The thesis work reported here is part of a larger project in the Hauler Loader Division called the autonomous machine project. This project is based on the idea of making wheel loaders and haulers working with simpler tasks autonomously, and thus increasing the profitability. By working autonomously the vehicle will not need any human driver at all, not even remotely. For example it would be enough just to tell the machine to start loading gravel, and it would carry on loading by itself until it is told to stop.

The autonomous machine project is divided into several stages where the thesis work is within the first stage. The first stage in the project is to create a working prototype wheel loader that can be used for further development. The goal of the prototype is to have a complete hardware system and a hardware-near software layer, such that a computer can have full control of the machine.

In the first stage of the project there were three thesis workers doing different work and one project leader which was an employee of Volvo CE. One of the thesis workers was the author of this report, and I worked with the hardware-near software layer. It included designing computer systems for two different computers to be used in the prototype machine as well as designing feedback controller loops. The other two thesis workers, David Gunstad and Peter Alnefeldt, worked directly with the hardware for the prototype machine. They designed many circuits and interfaces for the wheel loader. Lastly the project leader was Torbjörn Martinsson, who is an employee of Volvo CE in Eskilstuna.

2 Thesis Objective

The objective of the thesis can be shortly summarized into three points;

- A working prototype wheel loader which has full control of all the wheel loader functions
- A complete hardware-near software layer, which ensures full control of the wheel loader functions and works as a foundation for future development
- This report, a presentation and a demonstrated autonomous wheel loader

The main focus of the thesis is the software platform as a foundation for the prototype machine, which means the computer should have full control of all the machine functions but advanced working tasks are not necessary to be implemented. For example it is considered outside the scope of the thesis to implement an autonomous task that combines all the wheel loader functions to perform a both cost and time effective bucket filling. But it is considered inside the scope to make sure the autonomous task is possible to implement in the future.

The software must be able to make the machine demonstrate its functions while there is no driver seated though. The demonstration is for future development and for presentation purposes, and the video can be used in the presentation of the thesis. If the machine functions all work after the thesis work is completed, future development can focus on how to combine the machine functions for advanced working tasks.

3 Method of Development

Every kind of work benefits from a plan in advance about how the work should be done, and thus this thesis work was also well planned. The total work was divided into four different phases which were called the investigation phase, the design phase, the implementation phase and the verification phase. These four phases are in the report presented in separate sections, from Section 4 to Section 7 with Section 4 being the next section after this one.

The investigation phase includes an investigation of what an autonomous wheel loader is supposed to be able to do in the future, and why. From these answers requirements on the autonomous wheel loader can be made to make sure future development is supported. Since the thesis work is supposed to develop a software platform for future development the requirements on the autonomous wheel loader must be met, such that in a year or two no problems will arise from non thoughtful development at the start. The investigation phase was done before any of the design and implementation work was done.

In the design phase the design of the software platform was done such that the requirements from the prior investigation phase were met. The design includes details such as which computers should be used, what systems on the computers to use, which will be the programming languages, what will be the programming structure and so on. If any new sensors are necessary to meet the requirements found at the investigation phase, these would also have to be designed for.

The implementation phase includes the implementation of the designed systems. The section presenting the implementation phase will deal with problems such as how the communication is done between the different subsystems, and other problems arising when dealing with the implementation.

In the verification phase the results from the prior phases are verified so that they are working and behaving according to the results of the investigation phase. This includes all the subsystems on the wheel loader and all the functions on the computers.

4 Investigation Phase

The investigation part of the work is perhaps the most important part of the work, and perhaps also the most tedious one. The goal of the investigation phase is to decide what to do before it is being done to insure the quality of the work.

Now a natural way to start the investigation phase is to start from the goal and work backwards. When the goal is known one can think about what is required to achieve it. The goal of the thesis is to develop a software system to be used on a prototype autonomous wheel loader which has been somewhat discussed already in Section 2 above. But the software system will also be a foundation for future development on the prototype machine, so the long term goals also have to be considered. An obvious long term goal is that the autonomous wheel loader should be able to carry out more advanced tasks than simply having control of the functions. These tasks often include haulers to haul the cargo the wheel loaders are loading. These kinds of tasks are now going to be discussed as a possible application for the autonomous wheel loader.

4.1 Autonomous Wheel Loader Applications

Knowledge of the possible future applications or tasks of an autonomous wheel loader helps when considering how to design a system that will hopefully work as a foundation for future development. With this knowledge, decisions made in the thesis work might still show to be a bad decision in the future although the probability is smaller.

The investigation focused on perhaps the most commonly used application of a wheel loader, the short loading cycle. An illustration of the short loading cycle can be seen in Figure 4 below. It is a simple application with a wheel loader loading cargo from a pile to a hauler. It can be divided into three main tasks, filling the bucket from a pile, dumping the cargo in the bucket on the hauler, and the task of moving between the bucket filling and the dumping.

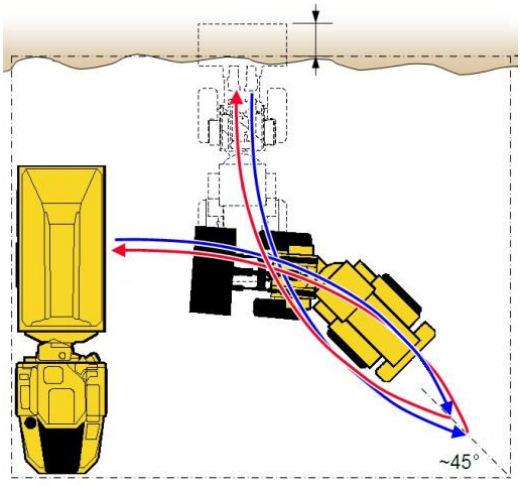

Figure 4: An illustration of the short loading cycle.

Figure 5: Picture of a wheel loader unloading on a hauler

The short loading cycle is a possible application for an autonomous wheel loader in the future and therefore the thesis work need to keep it in mind. The investigation should answer questions like "Which requirements are necessary to put on an autonomous wheel loader to make it able to perform the short loading cycle?" and the results should be the basis of the design work.

The outcomes from the investigation are divided into two parts. The first part is presented in Section 4.2 below and is dealing with the wheel loader's functions, which means what kind of requirements to put on the autonomous wheel loader so it can carry out for example the short loading cycle. The second part is presented in Section 4.3 below and is dealing with many questions that arise about safety on an autonomous wheel loader.

4.2 Function Requirements

This is the first part dealing with the requirements of an autonomous wheel loader's functions. The short loading cycle introduced above consists of three phases which requirements are discussed here, how to fill the bucket from a pile, how to empty the bucket in a hauler and how to move the machine from filling to emptying and back. It is yet again worth mentioning that the prototype machine developed in the thesis work is not supposed to be able to carry out all these tasks, but the prototype machine should be developed in such a way that future development with these tasks as goal is supported.

To fill a bucket from a pile is more difficult than it seems. For an autonomous wheel loader to be attractive in the future it needs to be able to fill a bucket effectively, i.e. with as little fuel consumption as possible and with as a full bucket as possible. When an experienced driver fills a bucket he combines all of the functions lifting, tilting, throttle and brake with light fingertips for a smooth and effective result. An autonomous wheel loader must be able to work in a similar way, with parallel movements and high precision to achieve an effective filling. The extreme case of the necessity of precision for the bucket is handling of pallets. To fetch a pallet, the wheel loader would need a precision of no less than 2 cm at the tip of the bucket.

To empty the bucket in a hauler, an experienced driver lifts the bucket just slightly over the edge of the hauler before tilting. The driver also uses the brake and throttle such that the cargo will keep unloading in the centre of the hauler even when the front tip of the bucket moves due to tilting. For an autonomous wheel loader to function the same way, it needs to know where the hauler is located with a good precision. The better precision, the less fuel consumption since lifting a full bucket consumes much.

Lastly, for an autonomous wheel loader to move effectively between a pile and a hauler it needs sensors to see where the pile and the hauler are located in advance so that the computer can calculate an efficient route. Also the pile keeps changing after each time the bucket is filled. Therefore these sensors need to update the data about the structure of the pile after each bucket filling to understand how to effectively fill the bucket next time.

From the discussions above, a few requirements for an autonomous wheel loader can be summarized;

- A precision of less than 2 cm for the tip of the bucket.
- Parallel and smooth movements
- Additional sensors for locating the hauler and the pile, as well as for analyzing the pile structure.

4.3 Safety Requirements

Of course, there must be extensive requirements on the safety of an autonomous wheel loader. The vehicle is indeed very heavy, and can inflict great damage on individuals or property if care is not taken. But it is also important to keep the objective of the thesis in mind, which is to construct a prototype machine and not a final product. Therefore the requirements on safety are first and foremost for a prototype machine to be used in the working environment.

To conclude, the prototype machine will need a remote emergency stop device that will always work. To ensure this, the stop device will need to stop the machine if it lost its remote contact. Also the emergency stop device must be installed in such a way that when activated, the machine always stops.

5 Design Phase

This section discusses how the software system on the prototype autonomous wheel loader is designed. The design of the software system includes the structure of the software, its intended purpose and other thoughts before the implementation. Section 6 below discusses about the implementation of the software.

5.1 Overview

In Figure 6 below a brief overview of the designed software system for the autonomous wheel loader can be found.

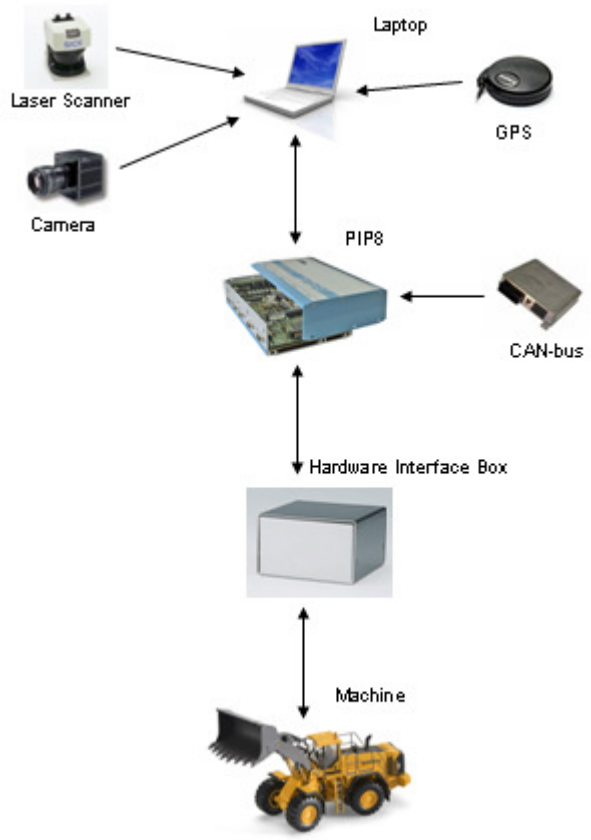

Figure 6: The Design Overview

To control the autonomous wheel loader, two computers are used. In this report, they are referred to as laptop and PIP8. The PIP8 PC is a hardware-near computer with I/O cards for AD, DA and CAN bus communication. The PIP8 PC also communicates with the laptop computer through an Ethernet cable. The laptop acts as the master of the two computers, it tells the PIP8 PC what do and how, and the PIP8 PC does what it has been told. The laptop does not have the ability to control the wheel loader directly, because that is what the PIP8 PC does with its analog and digital outputs. Neither does the laptop receive feedback from the

wheel loader directly; again this is what the PIP8 PC does with its analog and digital inputs. The PIP8 PC thus controls the wheel loader directly with its I/O ports, and these signals are connected to a hardware interface box that the thesis work does not have any influence on. With this design, the laptop does not have to worry about how a process is carried out. The laptop instead takes care of machine strategies, such as how to synchronize arm lifting, bucket tilting and throttle to achieve a good bucket filling. Since some of these machine strategies require extra information that is not available on the current wheel loaders, the laptop will also have access to data from additional sensors such as GPS, laser scanner and cameras that the PIP8 has no access to.

The wheel loader is considered in the paper to have several independent processes to be controlled, such as lift, tilt and steering. In reality, they are not independent and they all influence each other somewhat. The processes are in the thesis work still considered independent though, considering this is a first prototype machine and the limited extent of the work. There is only one laptop and only one PIP8, but the PIP8 runs several parallel and almost identical processes for the different wheel loader processes. More discussion about the wheel loader processes and the software structure on the PIP8 can be found in Section 5.3.

5.2 Laptop Software Design

The first and foremost duty of the laptop in the system is to take care of machine strategies. To achieve this, the laptop needs to be able to communicate with the PIP8 PC to give commands and receive feedback. The laptop also needs to communicate with the additional sensors.

Furthermore, the laptop is designed to be used as a development tool in the future. This is actually very useful in the thesis work as well, especially for testing and tuning purposes. This means that it is designed to have a user interface where full control of the wheel loader is easily accessed for testing the machine functions. For example, with the user interface the user could order the PIP8 PC to do a step response, a ramp or a simple command like turning on the lights. The user interface also has complete machine feedback, which means it asks the PIP8 for signal values and writes them to the screen. The user interface is also designed to be able to log signals of choice, such that a step response can be made and the data signals plotted after the test is done.

5.3 PIP8 Software Design

The PIP8 PC is the computer that has direct control over the wheel loader. The PIP8 PC is designed to control the machine by sending an analog voltage from its DA ports, as if there actually were a driver in the cab. In the same way, it can receive analog voltages as feedback from the machine with its AD ports. The main duty of the PIP8 computer is of course to control the machine, and therefore this PC must have feedback controller loops designed and implemented for the different wheel loader functions to control. This section will be discussing about how to design such a feedback controller and which choices were made.

5.3.1 Controller Strategy

In order to decide which strategy to use when designing a feedback controller loop, some information about the wheel loader processes and how the computer can control them are necessary. Figure 7 below shows an overview over the systems in a wheel loader.

Most of the wheel loader functions that the PIP8 PC controls are hydraulic, such as lifting the arm, tilting the bucket, steering the vehicle and also the braking system. In short, the hydraulic system works by building up a high oil pressure which is used by a hydraulic cylinder as an actuator. Inside the hydraulic cylinder there is a piston and a piston rod. The piston is pushed by the high oil pressure and in turn will either push the piston rod out of the cylinder, or back in. The piston rod is then connected to for example the arm of the wheel loader in such a way that it will lift the arm when pushed. The high oil pressure is in turn generated by a hydraulic pump which is driven by the wheel loader motor, and the pistons in the hydraulic cylinders are controlled by a pulse-width modulated (PWM) signal. Now, the PWM signal is sent by an internal electrical control unit (ECU) which uses the analog signals from the levers the driver uses in the cab to generate the PWM signal.

The driver thus sends signals to the ECU in the wheel loader by using the controls in the cab, and the ECU control the machine. For example when the driver pushes the throttle pedal, a signal will tell the ECU how much the pedal was pushed. The ECU then tells the motor how much it should increase or decrease the motor RPM. In the same way when the driver wants to lift the arm he uses the levers in the cab, the levers sends signals to the ECU and the ECU generate a PWM signal to the hydraulic cylinder that actuates the arm.

The design of the autonomous wheel loader is simply to replace the signal a driver would send to the ECU by using the controls in the cab with the signal from the PIP8 PC. With this design, it is also easy to switch the autonomous wheel loader between an autonomous mode and a manual mode, so the machine can still be driven by an ordinary driver in the manual mode. In Figure 7 below, the block labelled “Driver” would therefore in the autonomous wheel loader be the PIP8 PC instead, and the ECU will not notice the difference.

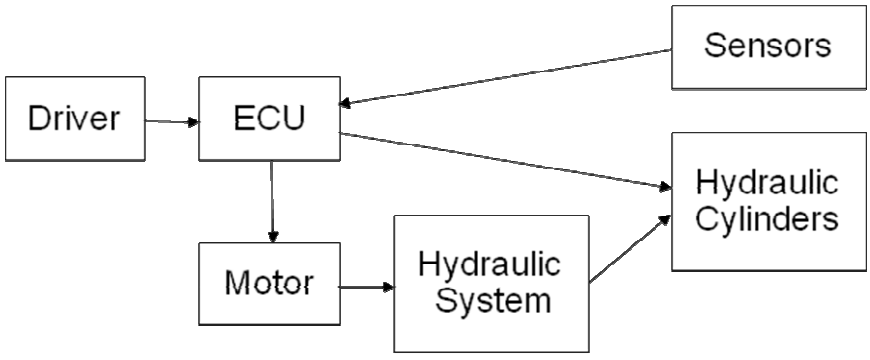

Figure 7: Block Diagram of the wheel loader systems

In Section 5.1 above, the text mentions that each wheel loader process will be considered independent. This is the first major choice made, which was done to make the controlling strategy easier to consider since then there can be independent controllers designed for each of the processes. Secondly, because of the highly complicated processes as described above the first step in the controller strategy will be to design simple PI controllers with anti-windup. With limited time, a nonlinear lifting framework and with no control over the system inside the ECU, the choice was made not to make a complete model of the process for a state feedback controller.

6 Implementation Phase

This section of the report includes discussions about how the systems described above were actually implemented to work on the autonomous wheel loader. In Section 5 the design of the software systems on the laptop and the PIP8 PC was introduced and this section will cover the implementation of these two software systems. In this section you can find block diagrams illustrating the structure of the software as well as other details affecting the results. Also the implementation of the feedback controller is discussed. As an example different varieties of anti-windup were tested and resulted in different behaviours.

6.1 Laptop Software Implementation

The implementation of the laptop software system was done according to Figure 8 below. The software consists of 5 different parts, 4 of them used for communicating with the 4 devices as discussed in Section 5 above, and the fifth is the user interface. The main duty of the laptop software is to take care of machine strategies, for example how to combine the different functions like lift and tilt to achieve an effective bucket filling. This has to be supported by the laptop software but not implemented in this thesis work. Therefore there is no part in the figure below for machine strategies, but the software system supports an implementation of such a part since the communication and user interface is already taken care of.

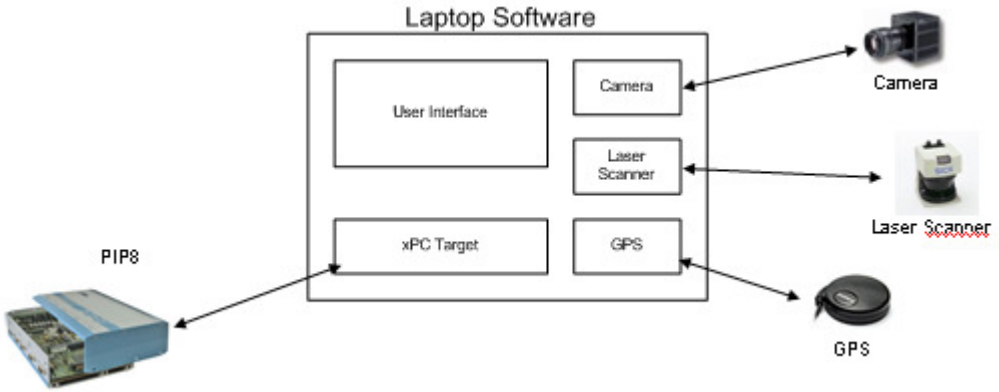

Figure 8: Laptop Software Structure

The user interface and the communication to both the PIP8 PC and the GPS will be discussed below in their own sections. During the time of the thesis work, the project team did not receive a camera or a laser scanner for the prototype machine. Therefore the two parts of the software for communication with the two systems are left undone and are thus not reported in the paper.

It is also worth mentioning that the Laptop Software was implemented with Microsoft Visual Studio C++ 6.0.

6.1.1 Communication with the PIP8 PC

This section describes how the communication between the laptop and the PIP8 PC works and therefore somewhat touches upon the implementation of the PIP8 PC.

The PIP8 PC uses xPC Target by The MathWorks, which is a software environment for prototyping, testing and deploying real-time systems (The MathWorks 2008a). With xPC Target the PIP8 PC can run a Simulink model in real-time while the laptop can monitor all the model signals, change model parameters etc, all while the model is still running. By using this software environment, the laptop can completely control the application on the PIP8 PC through a serial or Ethernet cable while it is running. In the thesis work, an Ethernet cable was used together with the TCP/IP network protocol.

Together with xPC Target there is also a dynamic link library (DLL) which can be used to create custom applications to control the PIP8 PC (The MathWorks 2008c). This API was therefore used for the communication to the PIP8 PC on the laptop software. In the work presented in the paper, functions were created in a separate C++ file to take care of the different communication functions such as reading signals or setting parameters at the same time as handling errors. These functions were created in such a way that they can be used for future development of the prototype wheel loader.

6.1.2 Communication with the GPS

Since the laptop software is supposed to take care of machine strategy, information about positioning is necessary. A simple task, such as moving from one part of a working area to another would be much simpler to carry out if there was some feedback about the position of the wheel loader. Therefore in the thesis work a GPS module was purchased and in the laptop software a thread for acquiring position data was written.

The GPS model used in the work is a Garmin GPS 18 USB, a small module which was connected to the laptop through a USB cable. The communication between the laptop and the GPS module is done by sending packages over the USB interface, such as information about position, or a command to start sending position packages once every second (Garmin International 2006). Each package has a well defined structure which specifies which byte represents what kind of data.

The software thread in the laptop acquiring GPS position data was written with the help of the Garmin Device Interface Specification. Simple functions were written that can be used for future development of the autonomous wheel loader.

6.1.3 User Interface

During the thesis work a user interface on the laptop was also implemented. It was programmed in Microsoft Visual Studio C++ 6.0 just like the communication parts above. The win32 library was used so that the user interface can be used on computers running any of the windows operation system. With the user interface the user has full control of all of the autonomous wheel loader functions as well as feedback of all essential machine states. The user can for example make a step response for a certain function and receive a file of data for analysis. Some of the functions in the user interface were implemented when the need arose, and some from the beginning. A view of the main window in the user interface is found in Figure 9 below.

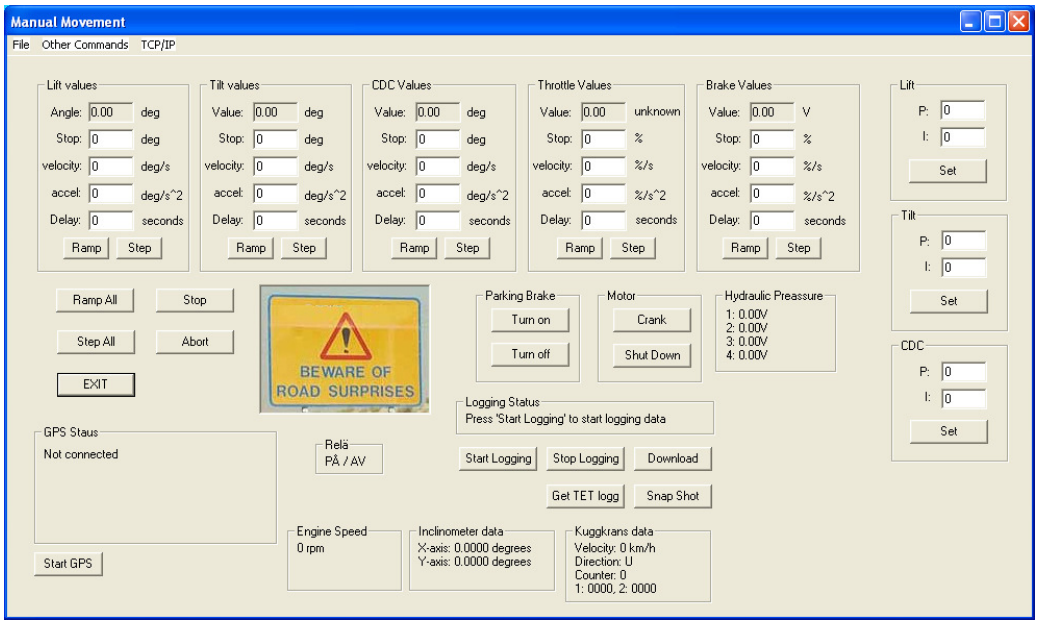

Figure 9: A view of the user interface on the laptop software.

The user interface seen in the picture above may look massive at first but the only thing it does is giving the user full control of the machine. In the top part of the window there are five areas for sending steps or ramps of reference values in five different wheel loader functions. There is feedback values spread around in the window from the GPS, the wheel loaders internal ECUs, hydraulic pressure sensors and velocity sensors and so on. All the other wheel loader functions that are not directly controllable from this window are instead controllable in the menu item labelled “Other Commands”.

6.2 PIP8 Software Implementation

In Section 6.1.2 above it is already noted that the PIP8 PC is running a software environment called xPC Target which is developed by The MathWorks. xPC Target is a tool for developing, testing and running real-time systems (The MathWorks 2008a). In principal, the tool has the ability to generate C-code from a Simulink model, compile it and run it on an embedded computer such as the PIP8 PC. xPC Target also provides tools for controlling, monitoring and logging signals from the model running on the PIP8 PC (The MathWorks 2008b). Therefore the software running on the PIP8 PC was developed as a model in Simulink on the laptop.

6.2.1 Simulink Model

In Figure 10 below there is the top view of the model developed to run on the PIP8 PC. The model is developed to execute fifty times every second. The execution starts in the block to the left, data is sent via the arrows and the model continues to execute the blocks in order of the arrows. The left-most block called Analog Input reads analog data from the wheel loader and passes the values on. Next in order there are six parallel blocks, i.e. they run independently from each others. These are subsystems for lifting the arm, tilting the bucket, throttling the motor, steering the vehicle, applying brakes and reading a speed sensor. These blocks are the ones actually taking care of the corresponding functions by using a controller as discussed in Section 5.3.1 above. The arrows leaving these blocks are the actual control signals, where the top four arrows goes to the Analog Output block and the fifth arrow becomes translated into a PWM signal. Lastly, there is a block called CAN bus. This block reads predefined messages on the CAN bus, such as the engine RPM. The laptop can then access the data with the help of the xPC Target API, as described above in Section 6.1.1.

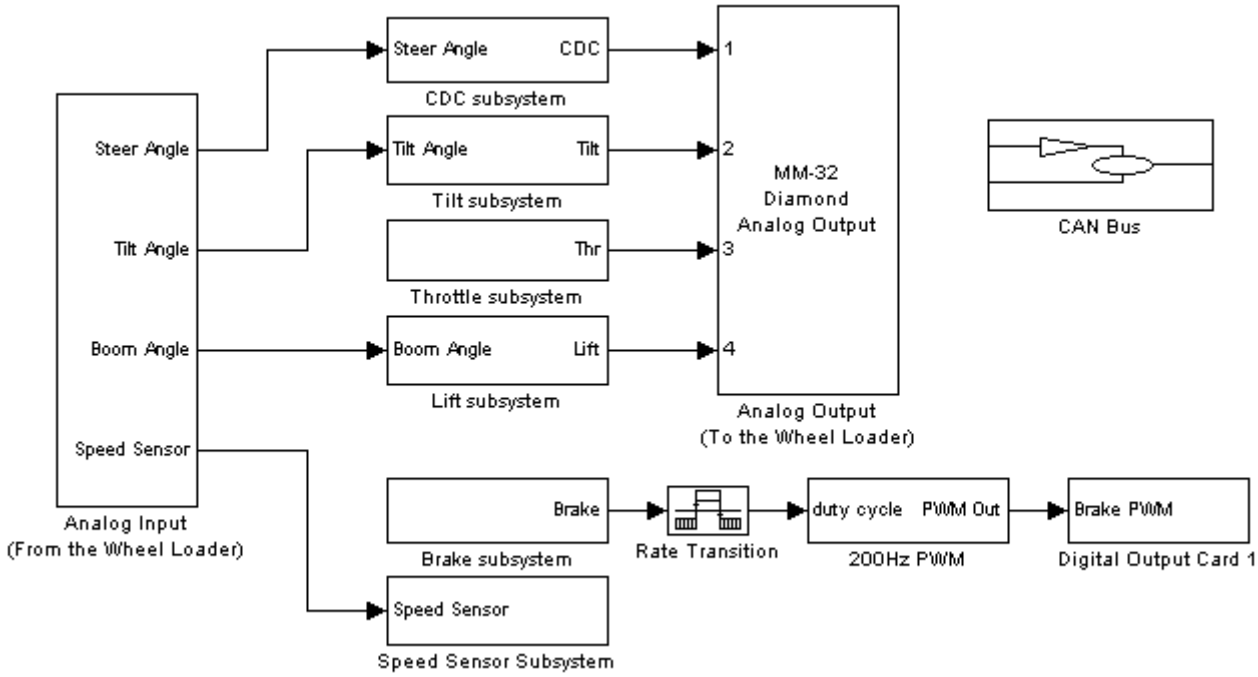

Figure 10: Top view of the Simulink model developed for the PIP8 PC

Each of these blocks described above have an interior Simulink structure as well. Inside these blocks a reference value for the controller is generated to be subtracted from the sensor input and then sent to the controller, which in turn produces a control signal value. The reference value generated is dependent on parameters which the laptop can change, so that the laptop can control how the reference values should behave. Many of these functions were implemented with the help of a Simulink block called Embedded Matlab Function, which is a block where the developer writes the functions as a Matlab function.

6.2.2 Controller Implementation

As mentioned in the discussion about the Simulink model above, the feedback controller loops can be found in the different blocks representing the different machine functions. These controllers were implemented in the model above and the gain parameters for the proportional part and the integral part were tuned on the actual prototype wheel loader. No specific method of determining gain parameters were used, the parameters were simply given what was though as a reasonable value and then fine tuned. The final controller used was a standard PI controller with tracking as anti-windup. This controller was satisfactory and therefore the work did not implement a more complicated controller, such as a PID controller or a state feedback controller. To better understand the need of anti-windup this section firstly shows the controller without anti-windup. The Simulink block of the controller without anti-windup can be found in Figure 11.

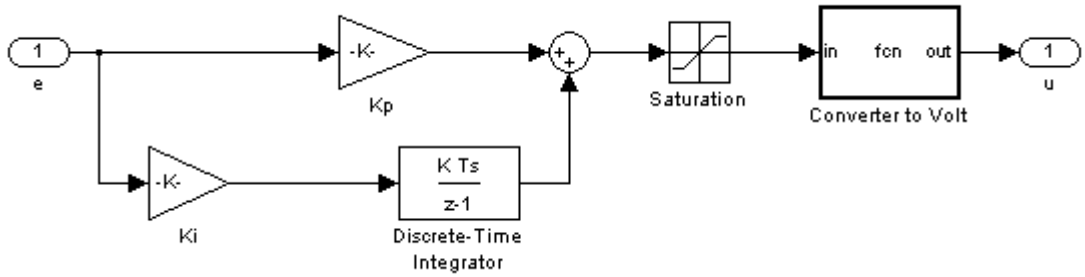

Figure 11: The feedback controller without anti-windup.

The input signal e is the difference between the reference value and the actual feedback value, i.e. the error. The error is sent to both the proportional part (P-part) and the integration part (I-part) respectively and then added together, saturated, converted to Volts and sent out as a control signal. The problem called integration windup arises when the wheel loader process is saturating, for example when the arm cannot be lifted faster than with a fixed max velocity. When this happens, the integration part continues to grow and there will be a large overshoot. This was tested on the actual wheel loader lifting framework with the controller setup seen in Figure 11 above, and the step response can be seen in Figure 12 below.

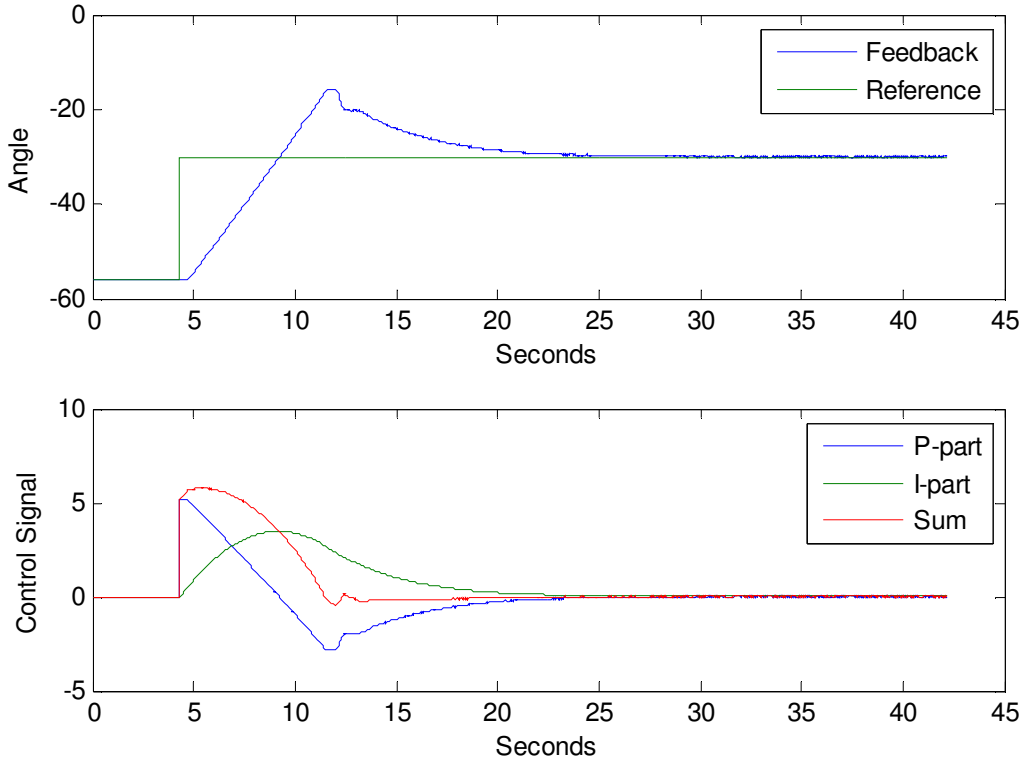

Figure 12: Step response of the controller without anti-windup. The top graph shows how the angle follows the reference value and the bottom graph shows the control signal.

In the figure above, it is easy to see the large overshoot when the arm keeps lifting because the integral part is so big. It is clear that anti-windup is necessary in one way or another. The final version of the controller implemented in the thesis work uses tracking as anti-windup. Tracking calculates how much the control signal was saturated and adds it to the term which is going to be integrated (Årzén 2006). Then the integration term is affected if the control signal is saturated, and if weighted with a constant will effectively counter the integration windup. Figure 13 below shows the Simulink block of the final controller with anti-windup.

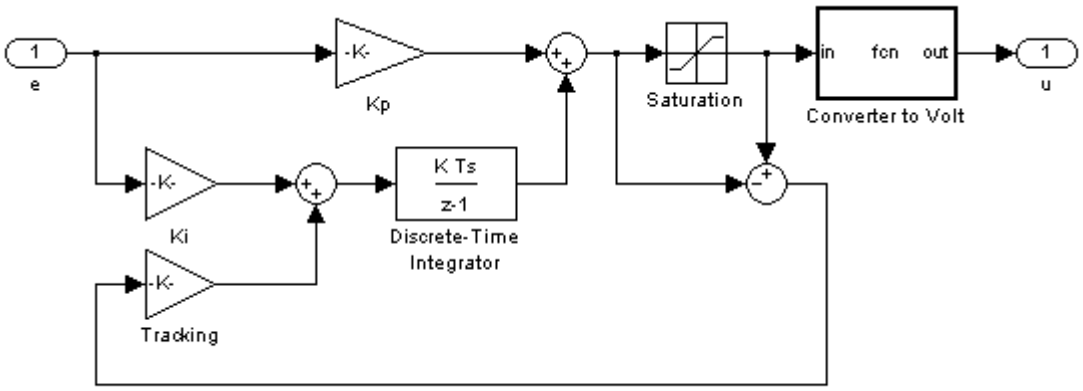

Figure 13: The final controller with tracking as anti-windup.

With this controller, the same test as above with same parameters and same step gives the step response shown in Figure 14 below.

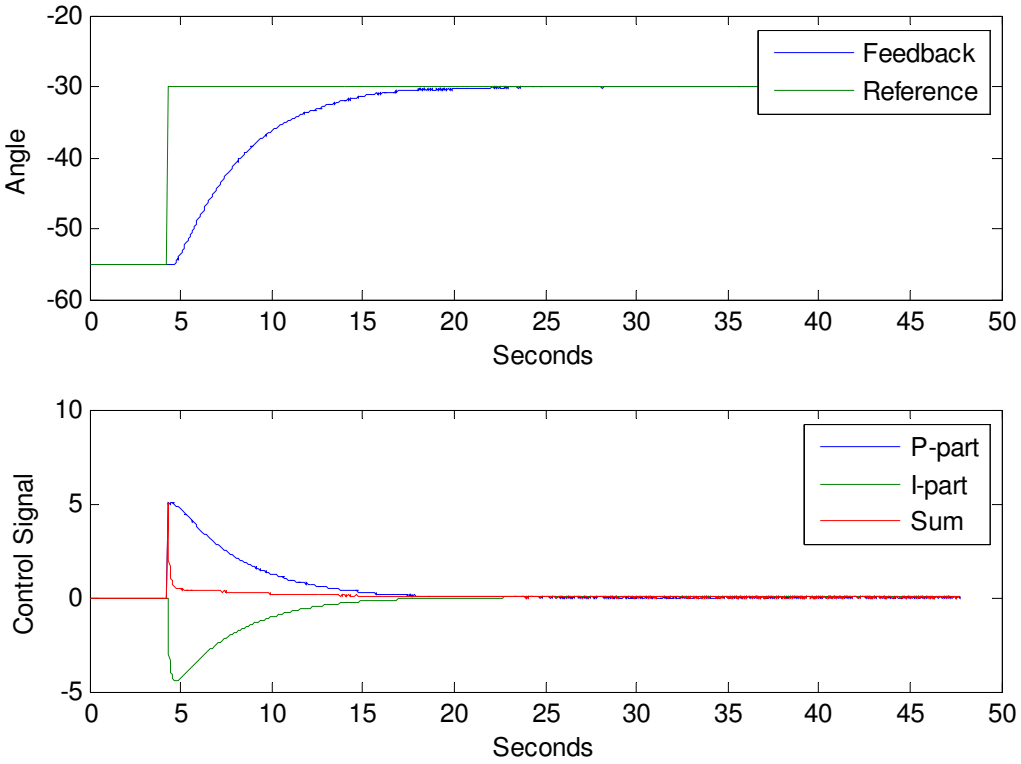

Figure 14: Step response of the final controller with tracking as anti-windup. The top graph shows how the angle follows the reference value and the bottom graph shows the control signal.

Comparing this step response with the previous one without anti-windup, the improvement is very clear. There s no overshoot and the arm moves smoothly.

There were also other types of anti-windup procedures tested. For example in Figure 15 below there is the Simulink block of the controller where the control signal from the integrator part is saturated to ±0.3. Figure 16 below shows the corresponding step response.

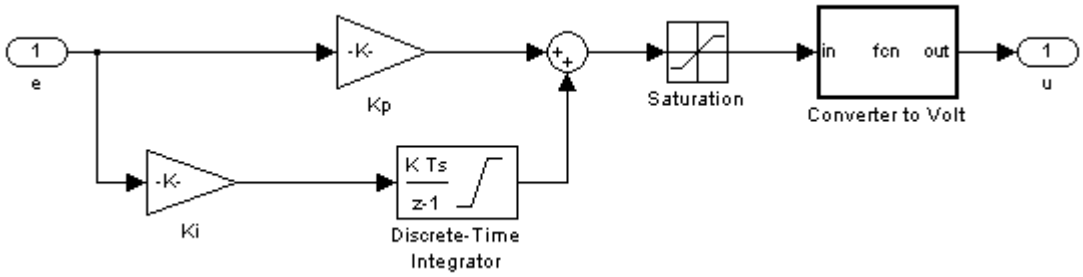

Figure 15: Simulink block of the controller with saturated integration part

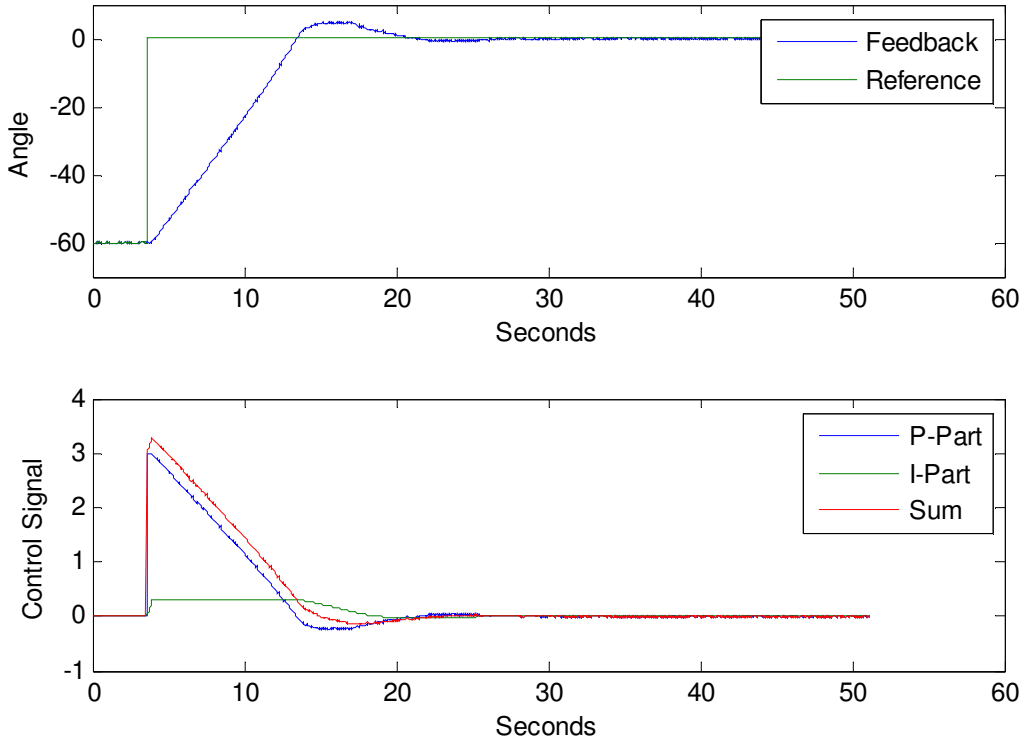

Figure 16: Step response of the controller with saturated integrator part. The top graph shows how the angle follows the reference value and the bottom graph shows the control signal.

We can see that even though the integration control signal is saturated it still causes an overshoot in the step response, but smaller. Therefore this version of anti-windup was not used.

Another attempt of anti-windup was made by limiting the integrator part such that it is disabled when the error is greater then a specified constant. Figure 17 below shows the Simulink block of the controller limited in such a way and Figure 18 shows the corresponding step response.

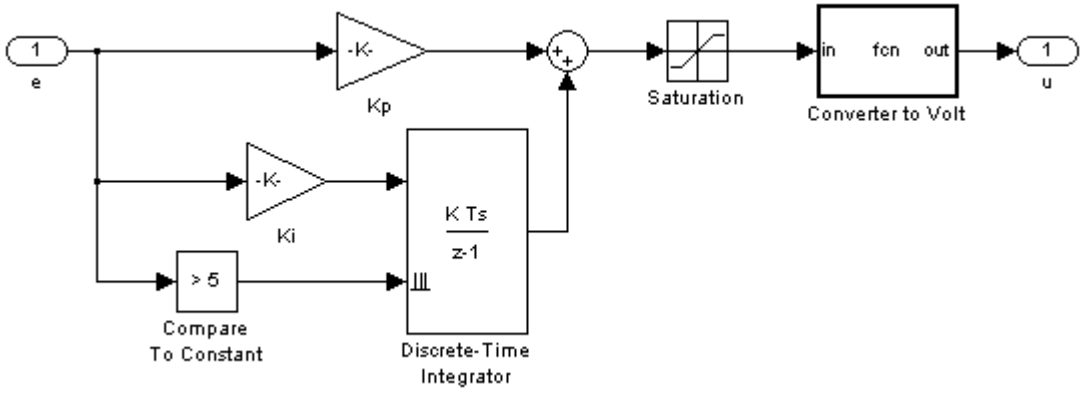

Figure 17: Simulink block of the controller with disabled integrator at large errors

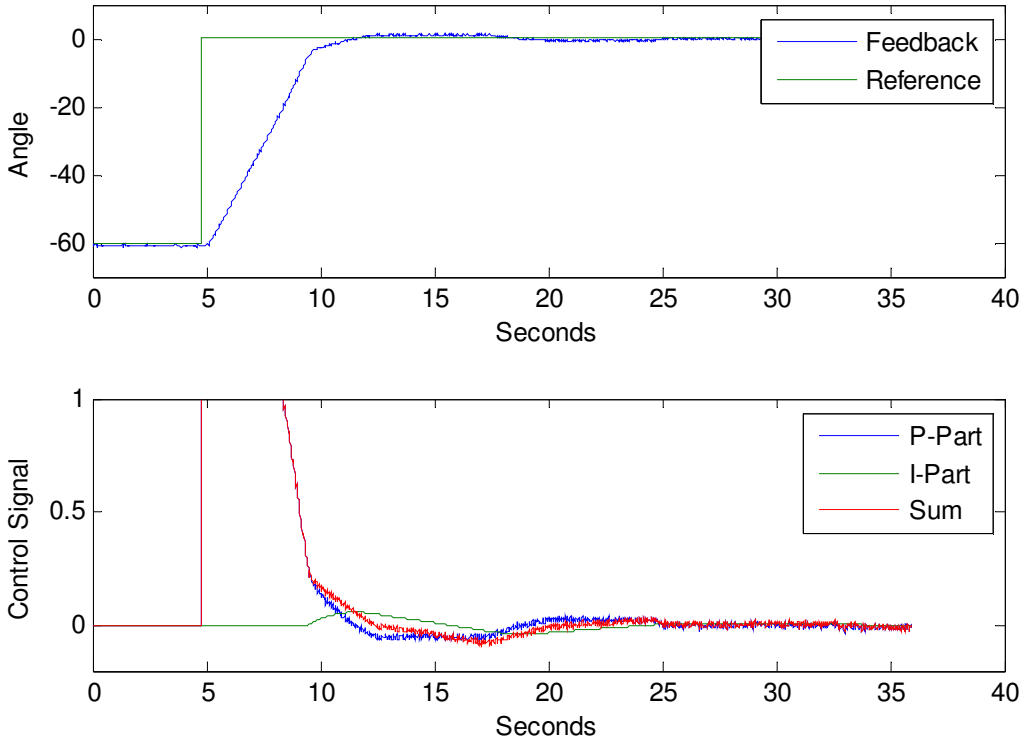

Figure 18: Step response of the controller with disabled integrator at large errors. The top graph shows how the angle follows the reference value and the bottom graph shows the control signal.

What we can see from this type of anti-windup is that when the integrator starts working due to the error becomes smaller than the specified constant the step response is affected. It is easy to see an "elbow" in the step response just at the same time as the integrator part starts working. Therefore, this anti-windup technique was not used in the final controller.

7 Verification Phase

The final phase during the thesis work was to verify the designed and implemented systems. This section covers the work that was done in the verification phase and presents issues that arose.

7.1 Verification of the Laptop Software

The laptop software was implemented to support future development of machine strategies, since the actual strategies were considered outside the scope of the work. Therefore the parts that were verified were what has been discussed above in both the design and the implementation phase; the user interface and the communications to the PIP8 PC and the GPS. The communication to the PIP8 was always working according to the xPC Target API discussed in Section 6.1.1 above, and the user interface never had any problems as well.

After a few solved programming troubles to make the GPS device send position data to the laptop accordingly, several tests were made to obtain graphs showing what kind of accuracy that can be expected from the position data. A small program was made which gathered data from the GPS device once every second and wrote the data in a file on the hard drive on the laptop. Figure 19 below shows a measurement done during one hour of time while the GPS device had a fixed position on the roof of a wheel loader that was not moving.

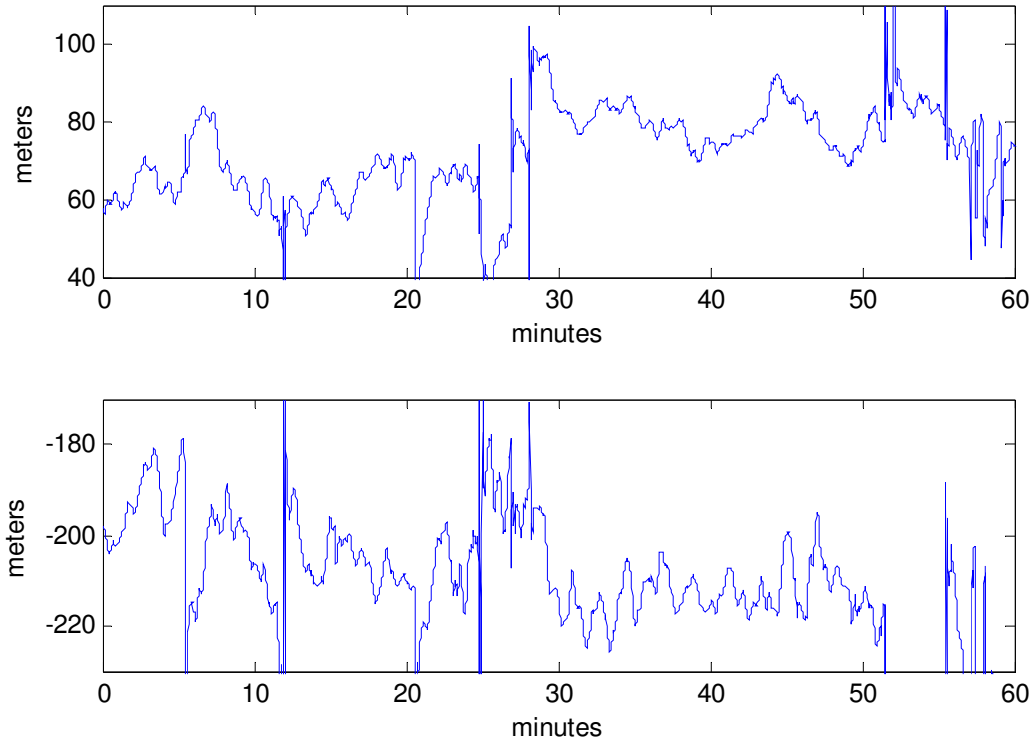

Figure 19: Graph of received position data from one GPS module. The top graph shows the latitude position and the bottom graph the longitude position.

This graph shows that at least the GPS seem to work properly, the signal is received and the functions programmed in the thesis work are working properly. But the actual position data seem to have large errors. We can also see slower variations which is a little strange. To investigate this further a second GPS device was bought to see if their two signals will follow each others. That test was done with two laptops, each having one GPS device which was not further away from each others than one meter. The two laptops started the measurement at the same time. Figure 20 below shows the result.

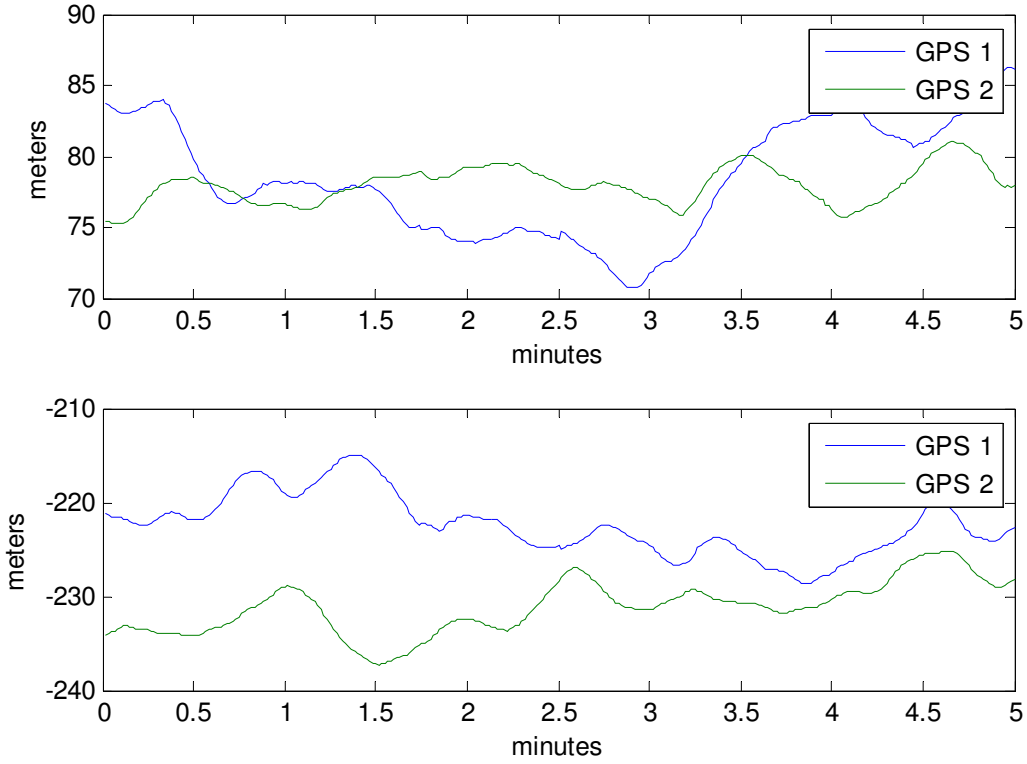

Figure 20: Graph of received position data from two GPS devices. The top graph shows the latitude position and the bottom graph the longitude position.

We can see a hint in the graphs above that there is a similarity between the two curves, i.e. they follow each other somewhat. But not at all as much as what was hoped for. If the two signals would follow each other more then perhaps it would be possible to design a simple differential GPS system where one GPS is stationary to receive a better position estimation. But since the result does not allow it the work was not done.

It is also worth mentioning that the GPS might not need high accuracy in the future if the autonomous wheel loader uses other sensors such as laser scanners and cameras for local positioning.

7.2 Verification of the PIP8 Software

The software on the PIP8 PC was implemented as described already by designing models in Simulink, and then letting xPC Target generate C-code, compile and download the model to the PIP8 PC. The procedure worked smoothly and erased many possible scenarios for errors. There was still one major problem arising from BIOS settings on the PIP8 PC to make it compatible with the xPC Target software. The problem was found as a warning message appearing on the PIP8 PC saying "CPU Overloaded". This message appears when the time required for executing an iteration in the downloaded model becomes larger than the model's specified sample time. To see if this was the cause of the warning message the task execution time (TET) of the application on the PIP8 PC was logged, see Figure 21 below. After discussions with both The MathWorks and the developer of the PIP8 PC, the BIOS settings could be identified as a reason and the problem solved.

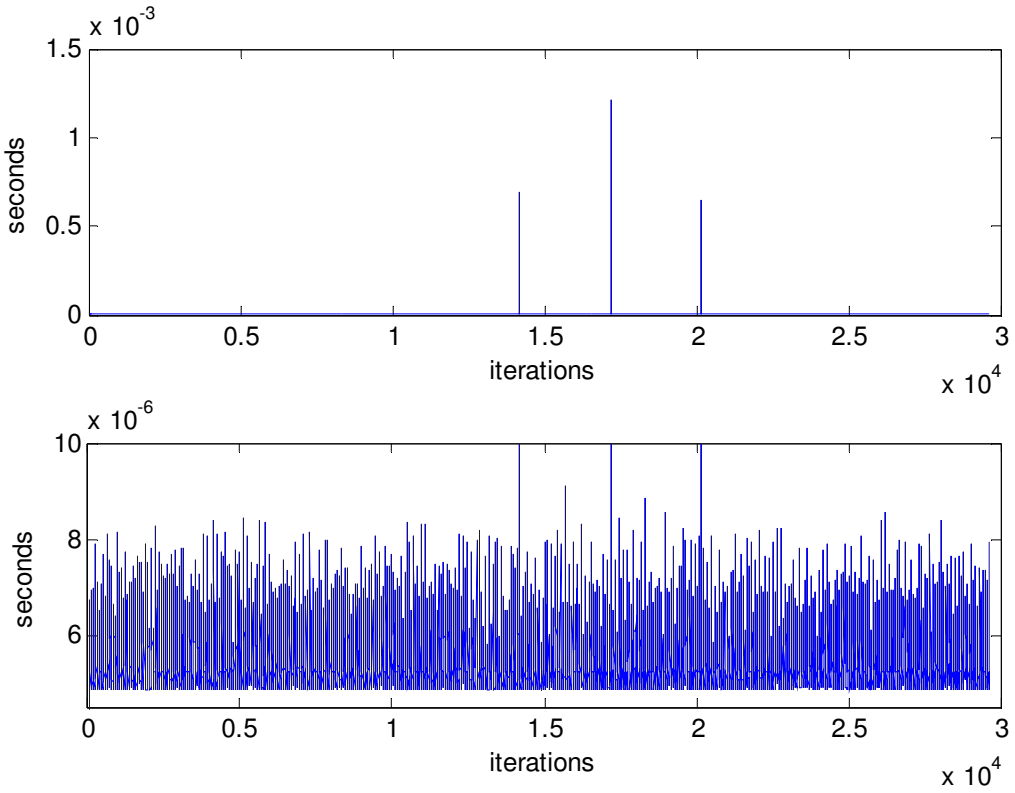

Figure 21: Task Execution Time problem

In the two graphs above the same data is plotted but with different y-axes to illustrate the size of the spikes.

7.3 Verification on the Autonomous Wheel Loader

In this part the result from testing the functions on the real machine will be presented. These results include step responses of lifting the arm, tilting the bucket, steering the vehicle, applying brakes and so on.

In Section 6.2.2 above some results were already presented, such as which type of controller is used and the respectively step response for lifting the arm. In this section more step responses for different functions are shown as well as logged values for an entire demonstration cycle.

One of the first things noticed while doing step responses in the thesis work was that the feedback signals from the angel sensors in the prototype wheel loader were very noisy. Therefore a digital filter was constructed in Simulink by sampling the signals faster than the feedback controllers and presenting the controllers with a calculated mean value from the faster sampling rate. With this filter the feedback signals from the angel sensors became much less noisy. See Figure 22 below for graphs of the angel sensor signals before and after the filtering.

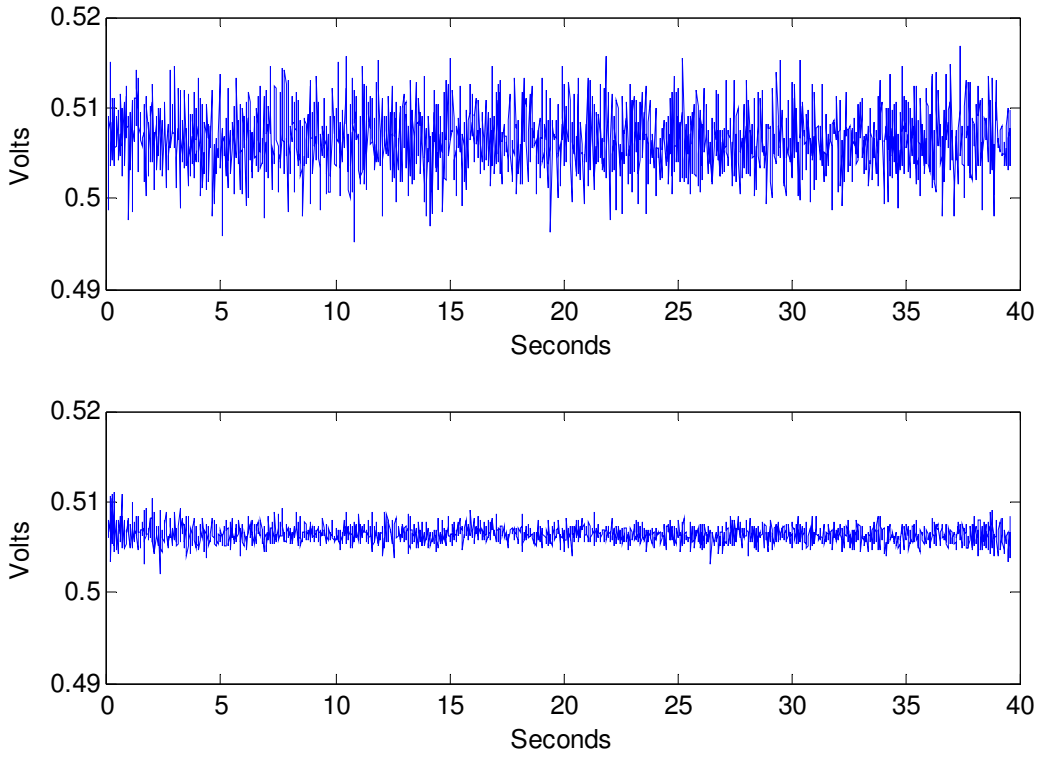

Figure 22: Graphs of the angle sensor signal. The top graph shows the signal before the filter and the bottom graph shows the signal after the filter.

In the graphs above, the same axes are used to better illustrate the effect of the filtering. Also the two graphs were measured at the same time, simply by monitoring the signal before and after the filter block in Simulink.

The first tested subsystem was lifting of the arm. This function was often used for verification; perhaps because it is so easy to see the results with the eyes. A step response of the lift subsystem is found in Figure 23 below. Note that this figure has already been shown in Section 6.2.2 above, but there for discussing the controller strategy and not for verification.

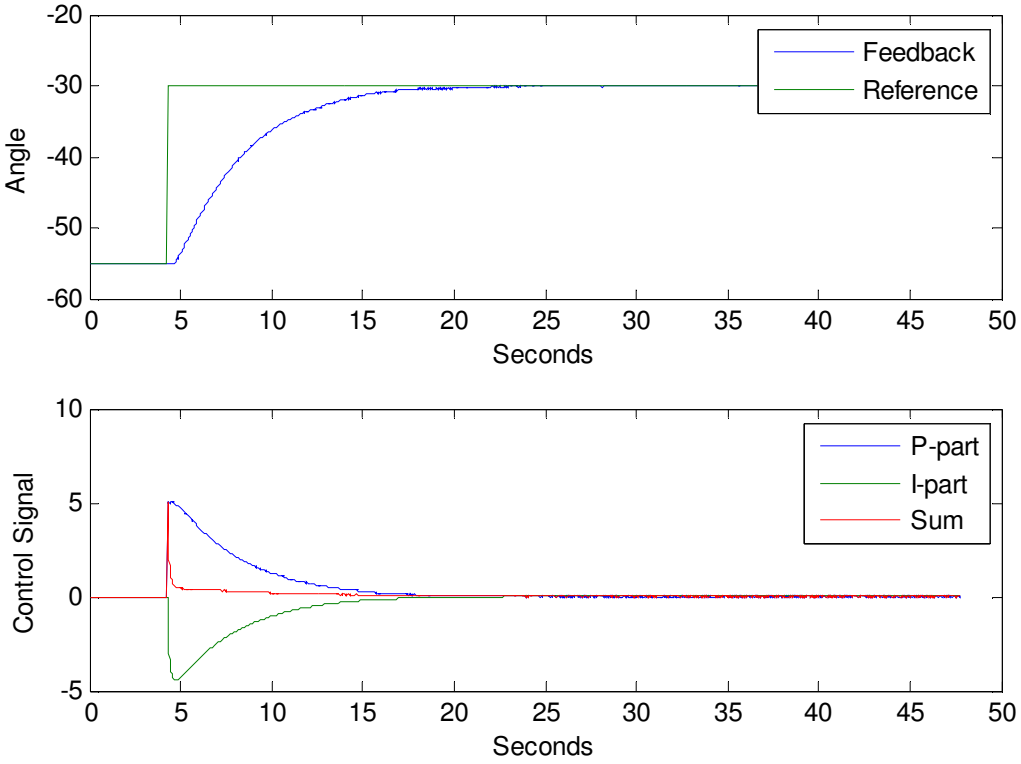

Figure 23: Step response for lifting the arm. The top graph shows how the angle follows the reference value and the bottom graph shows the control signal.

The figure shows that the lifting subsystem indeed works as intended. There were still a few interesting things noted about this process that cannot be seen in the figure above. Firstly it is highly dependent on the current RPM of the motor in the wheel loader; if the RPM is higher then the lifting will go faster. The step response above was done with an RPM about 780; the lowest one. Ideally the controller should be aware of the current RPM and use gainscheduling when controlling the lift, but even though the implemented controller does not use gainscheduling it works very well for higher RPM as well. Secondly, the process works very differently if the arm is being raised or lowered. Lowering the arm is really much faster than raising it, not only because the load is helping while lowering but also because the actual hydraulic system works differently for the two different functions. This fact has to be taken into consideration for future development.

The next function on the autonomous wheel loader that was tested is tilting the bucket. The same controller is used for this function, except that the gain for both the proportional part and the integral part were altered to suit the process. Tilting the bucket is much easier than lifting the arm, perhaps because there is very much load to counter while lifting but also because the hydraulic cylinders taking care of the two systems are different. The step response taken from the autonomous wheel loader while tilting the bucket can be found in Figure 24 below.

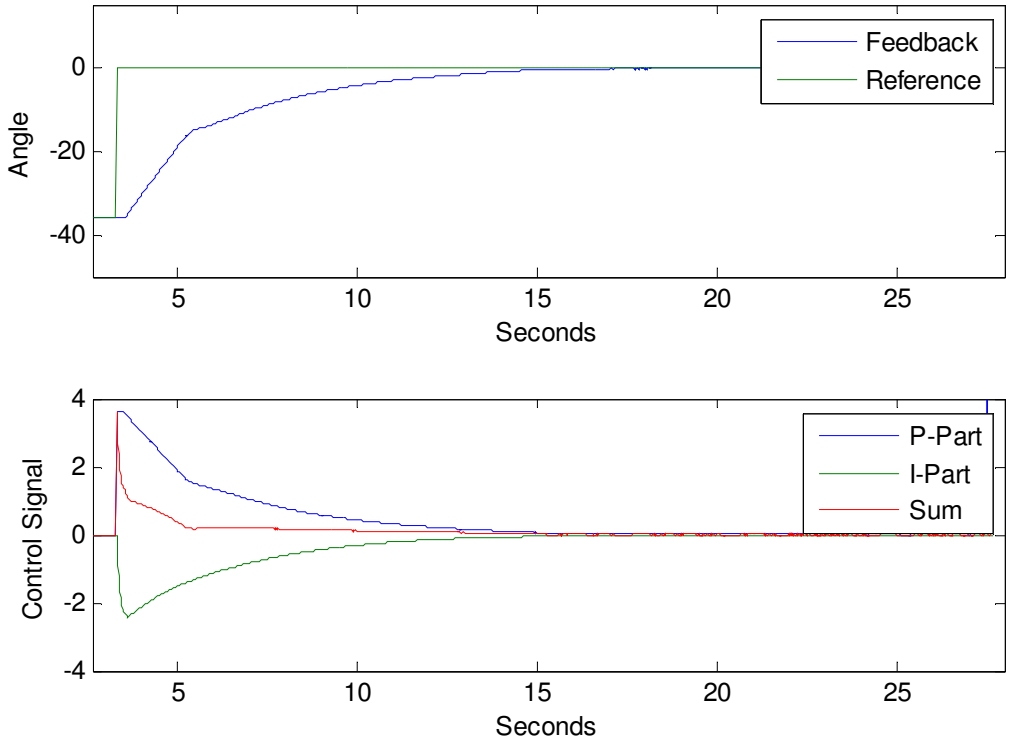

Figure 24: Step response for tilting the bucket. The top graph shows how the angle follows the reference value and the bottom graph shows the control signal.

This step response had the desired appearance, but we can see an "elbow" in the step response. We can see that this elbow appears just as the process stops being saturated. It probably appears because of nonlinearity in the process, for example the PWM signal sent by the ECU to the hydraulic cylinder might not be well calibrated with the arm framework and the controls in the cab. There could be a nonlinear point in the actuator speed when the process is about to saturate. The controller is the same as used for lifting the arm except for the gain parameters, so the elbow should not appear from the controller.

Some things discussed above about lifting the arm are also true in this case. For example the wheel loader can tilt the bucket much faster if the machine RPM is higher and there is a slight difference between tilting upwards and down, but this difference is not as noticeable compared to lifting and lowering the arm.

The third controller implemented was for steering the vehicle. In Section 1.2.3 amongst the introduction to wheel loaders there is written that the wheel loaders are using "articulated" steering. That means the wheel loaders are steering around a fixed pivot point in the middle between the two wheel axes. The actual actuator in this case is also a hydraulic cylinder just like for lifting and tilting. The same controller is used, except for the gain parameters which were tuned to better fit the steering process. The step response can be found below in Figure 25.

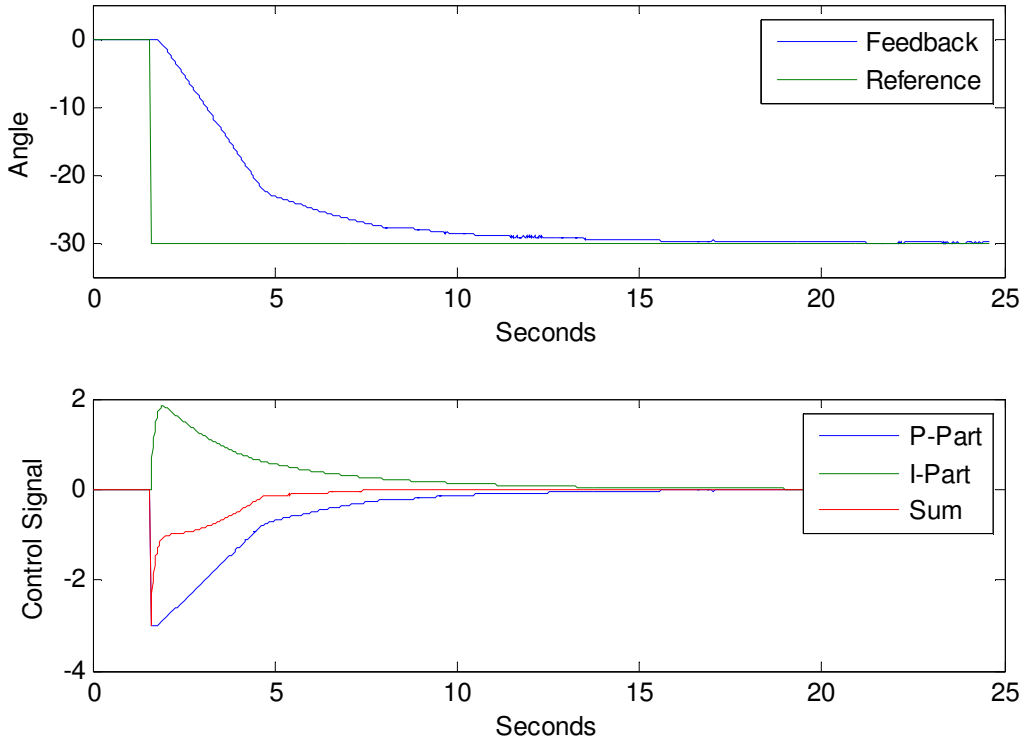

Figure 25: Step response for steering the machine. The top graph shows how the angle follows the reference value and the bottom graph shows the control signal.

In this case we can also see the "elbow" discussed above for tilting the bucket. Again the controller is the same except for the gain parameters so the elbow probably appears from a nonlinear point in the process when the actuator is about to saturate. This process does not share the other features of lifting and tilting except for using a hydraulic cylinder as an actuator. When steering it does not matter much if you steer to the right or to the left, the process is the same. What makes the process more complicated though is the friction to the ground, since friction is a nonlinear phenomenon an ideal controller would take that into consideration. Now the controller does not take this into consideration, and from lack of time a new one could not be done.

The thesis work did not implement a feedback controller for the brake and throttle systems, the throttle subsystem already has a controller implemented in one of the electrical control units (ECU) and the brake system consists of an electrical brake valve, which is similarly actuated by sending a PWM signal. Still it is of course possible to make a step response, by simply asking for an RPM and see what happens, or a brake pressure in the other case. The step response for the throttle subsystem is found in Figure 26 below and the step response for the brake subsystem in Figure 27.

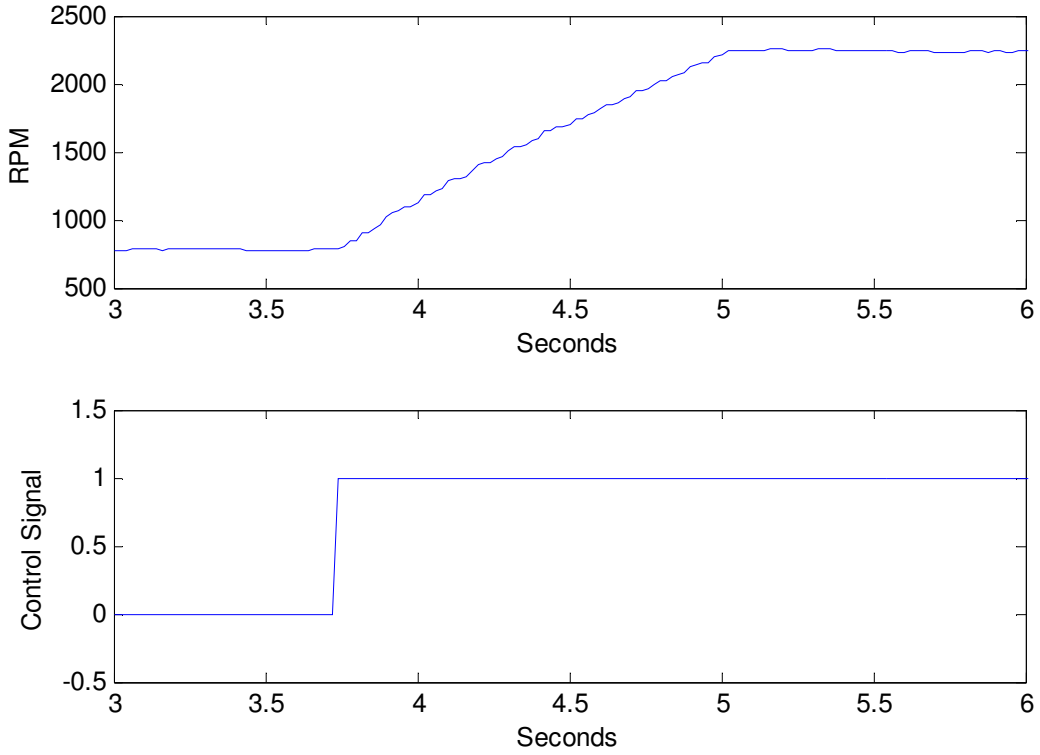

Figure 26: Step response of increasing throttle. The top graph shows the machine RPM and the bottom graph shows the control signal.

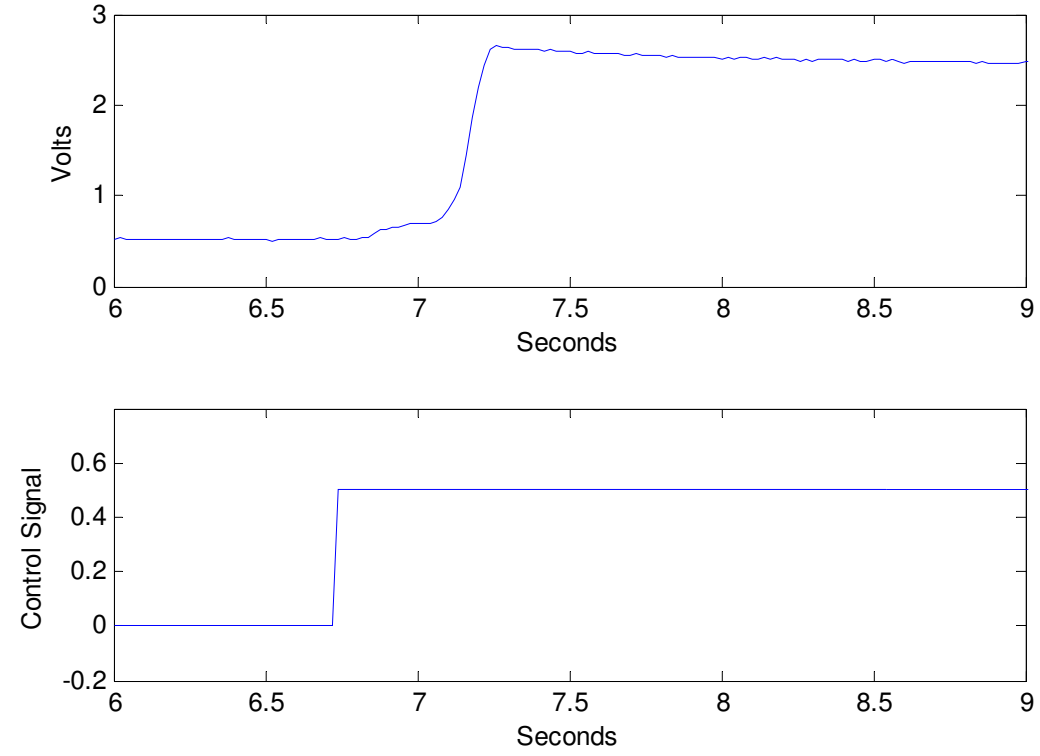

Figure 27: Step response of applying the electrical brake valve. The top graph shows the brake pressure and the bottom graph shows the control signal.

Many other systems were of course tested and verified, such as the horn, the blinkers, the working lights, the rotation light and so on. For these more simple systems there are no needs for any controllers and therefore there are no step responses to show.

Lastly in this section the results of a larger cycle where all subsystem are used are presented. This cycle is meant as a demonstration of what the autonomous machine is capable of. It is a total of five figures, from 28 to 32 displaying lift, tilt, steering, throttle and brake respectively.

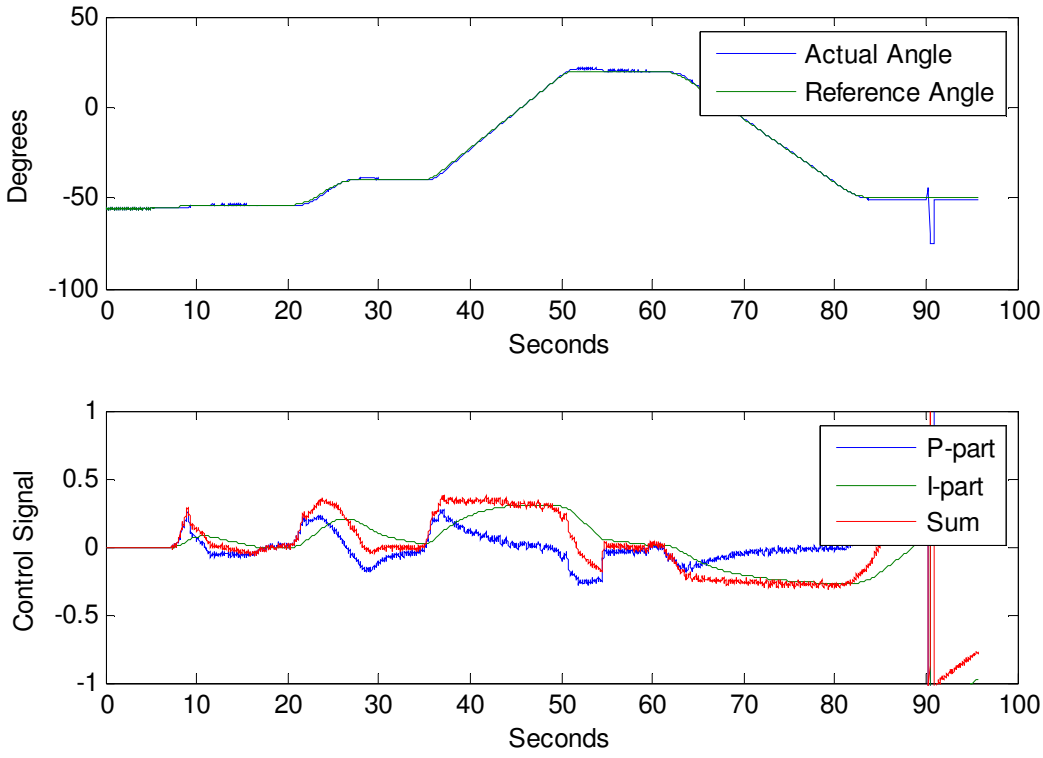

Figure 28: Lift graph for the demonstration cycle. The top graph shows how the angle follows the reference value and the bottom graph shows the control signal.

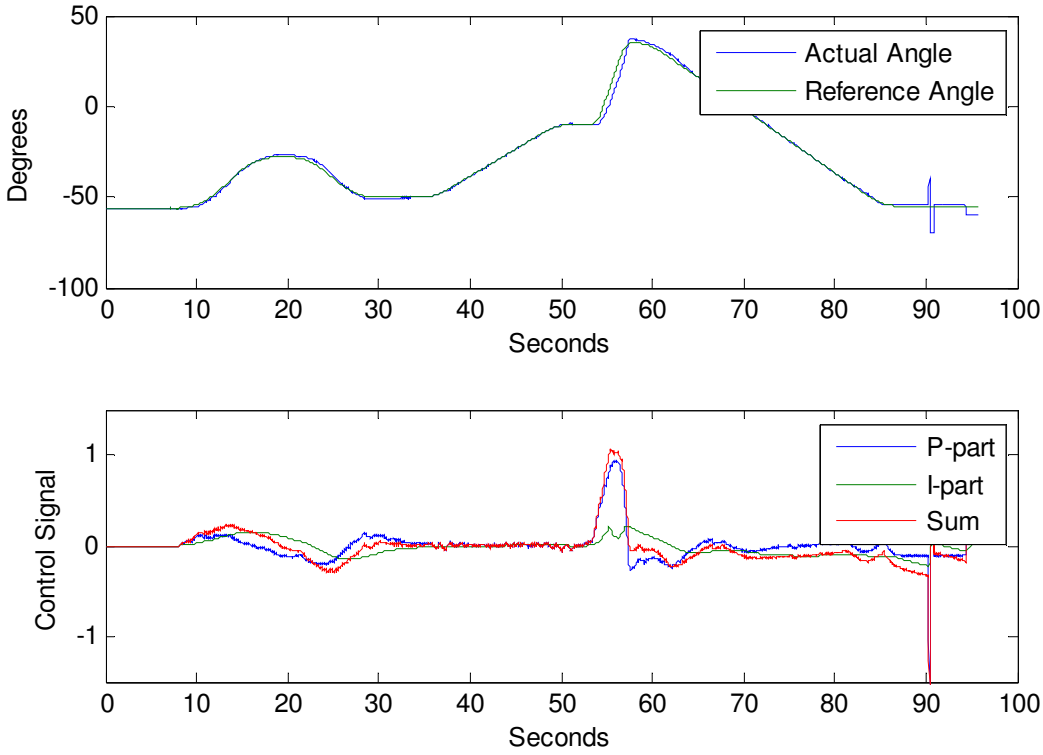

Figure 29: Tilt graph for the demonstration cycle. The top graph shows how the angle follows the reference value and the bottom graph shows the control signal.

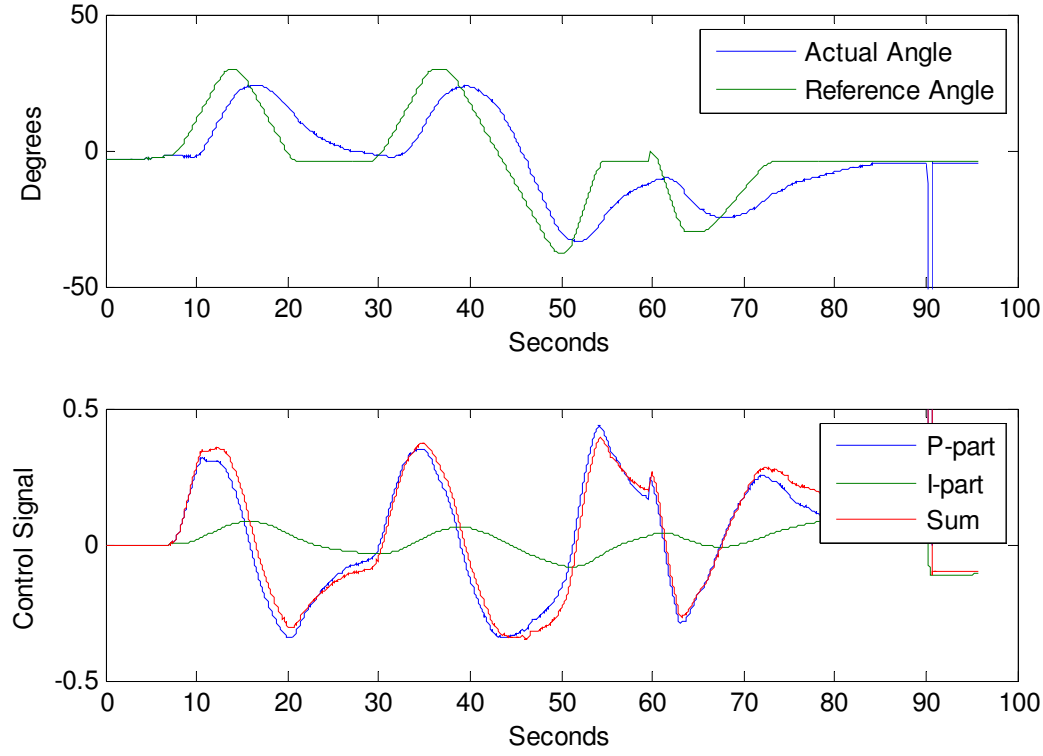

Figure 30: Steering graph for the demonstration cycle. The top graph shows how the angle follows the reference value and the bottom graph shows the control signal.

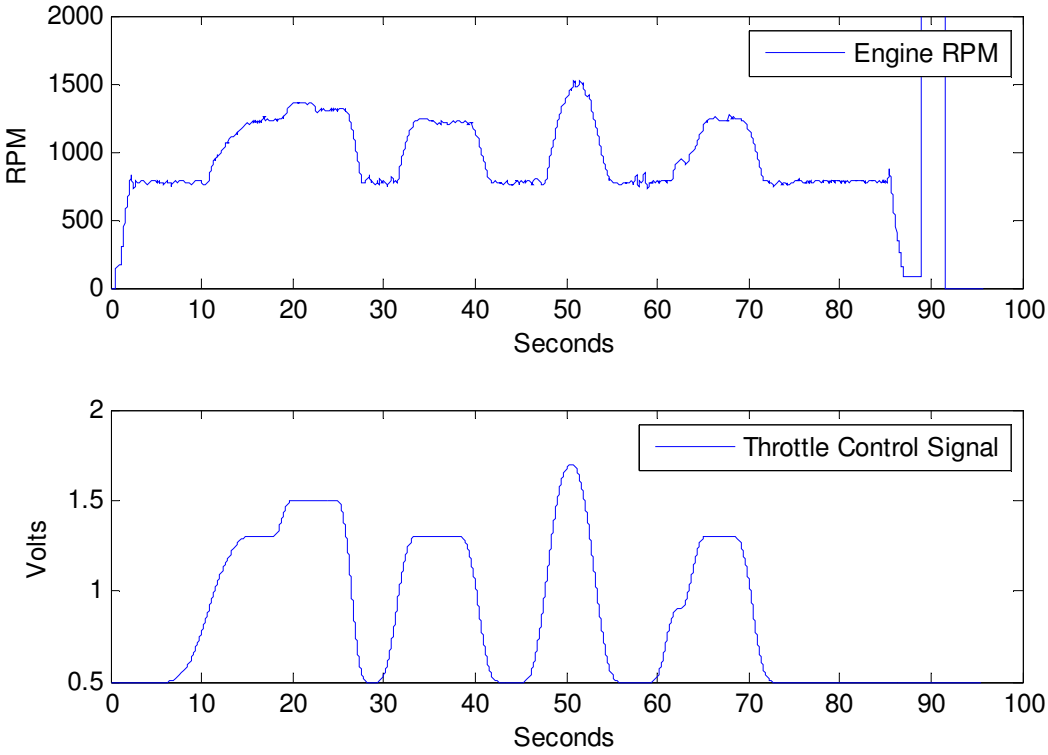

Figure 31: Throttle graph for the demonstration cycle. The top graph shows the engine RPM and the bottom graph shows the throttle control signal.

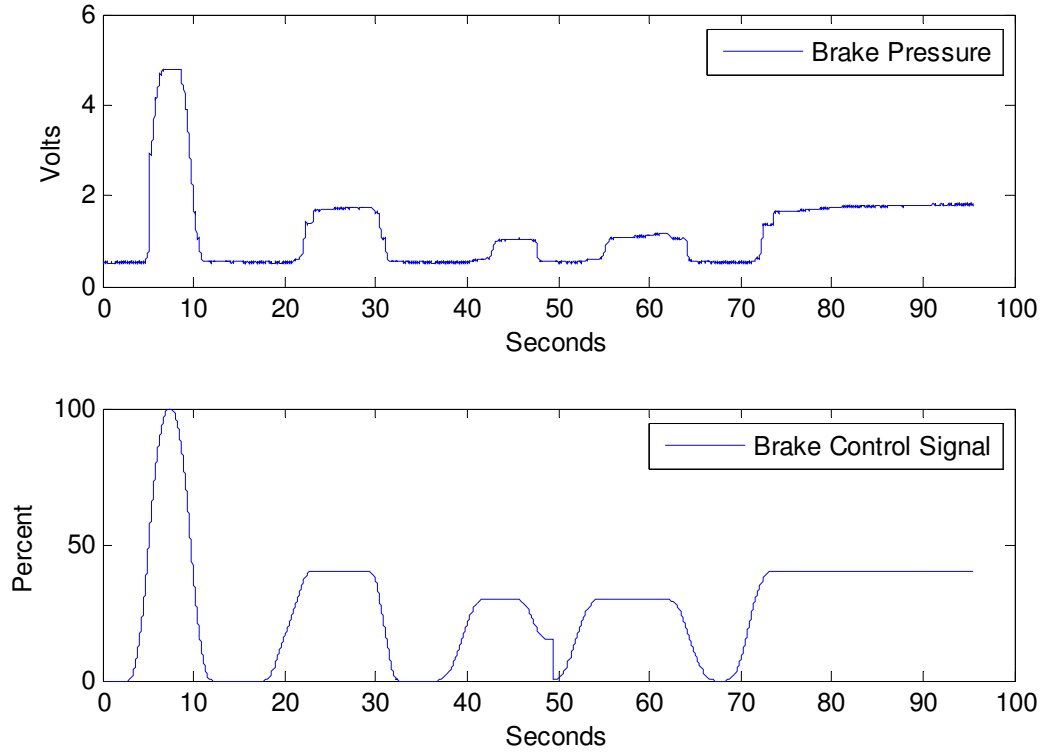

Figure 32: Brake graph for the demonstration cycle. The top graph shows the brake pressure and the bottom graph shows the control signal.

In these graphs you can notice in the end many values start behaving strangely. This is because the wheel loader was programmed to shut down the engine after the cycle is completed which also shuts down the internal ECUs. When the ECUs go offline, the signals that are received from them are no longer available. The ECUs are only offline for a short moment though before they are brought back online, which can be seen in the graphs.

8 Conclusion

The results of the work are very satisfactory. When watching the graphs in the Figures 28 – 32 showing the demonstration cycle you can see that the signals follows the reference values very well. Also, watching the real machine performing the same demonstration cycle is very satisfactory when the wheel loader moves so smoothly.

In Section 2 the thesis objective was discussed. The discussion shortly summarizes three important thesis objectives;

- A working prototype wheel loader which has full control of all the wheel loader functions
- A complete hardware-near software layer
- A foundation for future development

The first two important objectives are quite obvious that they are met. The prototype autonomous wheel loader works very well as well as the software platform. The third important objective is harder to realize. In principle it is impossible to know how the future will develop an autonomous wheel loader, but with the investigation phase that was done the work should be as a good a foundation as it can be.

8.1 Further Work

The most obvious remaining work is to make the prototype wheel loader being able to perform more advanced tasks. There are so many levels of difficulty in the work ahead of the autonomous machine project so defining where to start is possibly the most important thing to do.

There are some smaller details that are remaining work from the thesis work presented in the report. Since one of the I/O cards on the PIP8 PC did not work it must be fixed. There should be a different system for generating the PWM signal to the electrical brake valve as well as a different system for receiving the velocity sensor signals. This is because these two signals can be much faster than the PIP8 PC can manage to work with.

The feedback controllers implemented in this first stage of the project are standard PI controllers with anti-windup. I feel that for better performances new controllers should be implemented. For example in the cases of both lifting and tilting the process works differently upwards and downwards. Also higher machine RPM makes these two processes faster. An ideal controller should take this into consideration and is therefore left here in the section for further work. The feedback controller for steering the vehicle should also take the non-linear friction into consideration.

9 Reference List

The MathWorks (2008a): *xPC Target Getting Started*

The MathWorks (2008b): *xPC Target User Guide*

The MathWorks (2008c): *xPC Target API Guide*

Garmin International (2006): *Garmin Device Interface Specification*

Volvo Construction Equipment (2005a): *Volvo Hjullastare L50E*

Volvo Construction Equipment (2005b): *Volvo Dumprar*

Volvo Construction Equipment (2007): *Volvo Hjullastare L350F*

Karl-Erik Årzén (2006*)*: *Real Time Systems: Course Material L6*