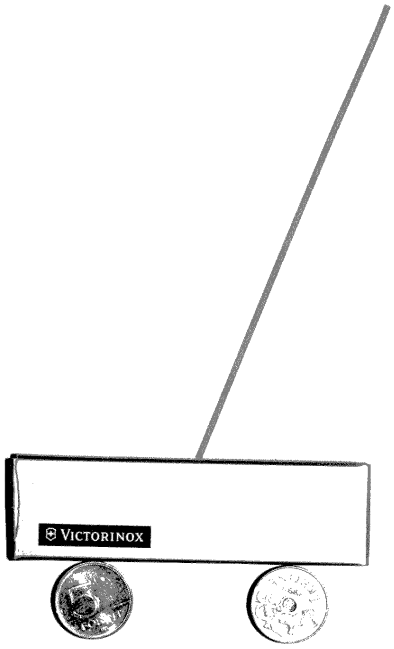
David² Inverted Pendulum

Project work



Members: **DÁVID BEDŐK**
and **DÁVID BRAUN**
Tutors: **TERJE AARSETH** Associate Professor
and **WEBJØRN REKDALSBAKKEN** Associate Professor
College: Aalesund University College (AaUC)
NORWAY
Mother college: Budapest Tech John von Neumann (BMF-NIK)
HUNGARY
Datum: ***02nd February 2005 – 26th April 2005***

PREFACE

This documentation is a project work of two Hungarian students made at Ålesund University College, Norway. It was an ERASMUS scholarship which was a three months long work in the spring semester of 2005.

So far we principally wrote software applications. Approximately one year ago both of us made an image processing application in Borland Delphi 7, after that we have begun to work in a complex project which we have recently worked with. One of us has dealt with music recognition[1] and the other one has written a navigation and map software for Pocket PC[2]. Beside these projects we became acquainted with 3D programming (manipulator arm), genetic algorithm, etc.

This kind of work (Inverted Pendulum) was alien from us because we had no experience in using these sorts of software and hardware. It was the first time we had to use data-flow programming (LabView) so seriously. We generally use Microsoft Visual Studio (.net, C++, C#) to write programs. Furthermore we could know the efficiency of the MATLAB and its accessory Simulink.

This project work was a great experience where we have come to know the rules, the application of control theory in practice, and we saw a problem from an engineer approach too.

We thank our tutors for their kind help in the project and thank the schools that gave us the opportunity to work here in Norway.

Dávid Bed□k Dávid Braun

[1] Hungarian webpage: http://roberta.obuda.kando.hu/braundavid/
[2] Hungarian webpage: http://roberta.obuda.kando.hu/qwaevisz/

PROJECT PLAN

This is a project plan for the students
David Bed□k
David Braun
from Budapest Polytechnic College.

The two students are staying at Aalesund University College (AaUC) for the period primo February to primo May 2005, altogether 12 weeks, in connection with an Erasmus scholarship. They have a background of two years of education in a Bachelor programme of Control Engineering. At AaUC they will be enrolled in the Cybernetics program. Most of the time they will be working with a project in control theory. This project will be evaluated in the same way as the final projects of our 3rd year bachelor students, and it has the size of 15 ECT student credits.

Project background

In the Cybernetics program at Aalesund University College the students have a considerable amount of laboratory exercises and practical projects. In the third year they are working with physical models of dynamic systems like motion platforms, a heave-compensated lift, inverted pendulums and different kinds of servo-systems. Much of the laboratory activity is in collaboration with the nautical studies, and the development and construction of ship simulators have become a speciality of AaUC. These simulators combine the competence from several fields and represent an integration of graphical visualisation systems, nautical instrumentation, ship manoeuvring models and control systems. At AaUC there have been built two full-scale simulators in cooperation with local maritime industry. Much of our laboratory activity is connected to this area. To study unstable systems we also have built physical models for controlling an inverted pendulum. These models are also included as part of our laboratory education.
One quite demanding task is to make the pendulum balance with the aid of a camera with a frame grabber card and the use of a servo motor. There are many challenges in managing the control of such a system considering both hardware and software, and there are many tools that can be used to provide a good solution. We had thought our students from Budapest should get the opportunity to work with this experiment, in particular to investigate the use of software and equipment from National Instruments to solve the problem.

Project description

In short the project comprises the construction of a feedback control system that will control the movements of the pendulum to make it remain in vertical

position. The pendulum is a steel rod placed on a cart. The cart is moved by a servo motor through a rubber belt.
To get an understanding of the problem, the students will first be working with the description of the dynamic response of the pendulum. They will work out a mathematical model for the physical system, which will be the base for analyzing the behaviour of the system. An important part of the project is to find a realistic and well functioning mathematical model, as this will lay the ground for deciding the right control strategy for the feedback loop. The input to the control system will come from a camera which is monitoring the movement of the pendulum. On basis of the camera input an image processing algorithm will calculate the angle of the pendulum and the position of the cart. The speed of the cart and the pendulum will be estimated as the derivatives of the position and the angle respectively. These variables will be the state variables of the system which are used in the control algorithm. The control algorithm calculates the necessary input to the servo motor to drive the cart in the right direction.

Project equipment and software

In addition to the physical model of the pendulum the following equipment and software tools will be used:

An industrial camera of type Jai CV-A50/A60.
A frame grabber card of type National Instruments PCI-1407 Single channel, Analog, Monochrome.
A National Instruments I/O-card and the type PCI-6024 Daq card.

National Instruments LabView 7.1
National Instruments IMAQ Vision software.
National Instruments software driver for the PCI-6024 Daq card.

Control strategy

The control strategy will be based on modal control with a pole placement algorithm.
The control loop will be as follows:

> The camera together with the image processing algorithm measures the states of the pendulum (x, $v = \dot{x}$, θ, $\omega = \dot{\theta}$). These four states are fed back to the reference position of the pendulum through a proper K-matrix. The error signal represents the new position of the cart. The necessary manipulating signal to the servo drive is generated to move the cart. The servo drive moves the cart (with the pendulum) to its new position.

The control system have two reference signals, one for the x-position of the cart and one for the angle ($\theta = 0$) of the pendulum. Both can be controlled by choosing an appropriate model and using integrators in the loop.

Control theory

The control strategy used will be based on state space design. Lectures will be given to help the students get the necessary background in mathematical modelling of dynamic systems and state space theory. The students will be introduced to the control law and how to derive the correct eigenvalues (poles) of the closed system in agreement with the requirements posed on it. The underlying system will be of an order from three to six depending on the model to be chosen. Requirements to the system's response will be discussed with regard to rise time, damping and settling time. The students will be introduced to Matlab/Simulink and learn how to use these tools to model and test control systems. Implementation of the control loop will be done with LabView 7.1.

Project schedule

The project will last for 12 weeks. This is a brief schedule of the work to be done over these weeks.

Week	*Work to be done*
1	Introduction to the project and working conditions.
2	Introduction to LabView and DAQ systems.
3-4	Studying National Instruments IMAQ Vision System and installing necessary NI hardware and device drivers.
5-6	Introduction to Matlab/Simulink. Learning about State Space models and State Space design. Introduction to pole placement methods.
7-8	Building a mathematical model of the pendulum. Analysing the system and designing the control loop.
9-10	Implementation of the control loop. Develop the image processing algorithm and the pole placement algorithm.
11-12	Testing of system. Writing project report.

The students will meet with professors twice a week to discuss the project status and to guide the students through new theory and methods. In agreement with the students the meetings will be on Mondays and Fridays at 09.00.

Webjørn Rekdalsbakken
Assoc. Prof.

Terje Aarseth
Assoc. Prof.

Aalesund University College
6025 Aalesund
Norway
Tel. +47 70 16 12 00
Fax. +47 70 16 13 00
Email: wr@hials.no ta@hials.no

TABLE OF CONTENTS

ABSTRACT

We have a physical model which consists of a cart on a rubber belt with a steel rod standing on it. If we leave it, naturally the rod will fall down. Our task is to affect the system in order to keep the rod in the upright.

We have a camera as input which is monitoring the movement of the pendulum. The manipulating signal will control the servo drive that drives the AC motor. When the rod is falling down on the right side, the car has to move quickly in the same direction.

First of all we constructed the model of the physical system writing differential equations. Using the calculated equations we made a mathematical model using State Space Analysis and State Space Design.

Our first mathematical model was 4th order, but after some simplifications we got a 3rd order system. We analysed these open-loop systems with Matlab Simulink.

In the end we used the 3rd order system to work with. To make a closed-loop system we applied modern control theory and Pole Placement Control. With the help of this technique we can change the original poles of the unstable system to make it stable.

Beside the Pole Placement Control we have to apply the control law. This can be written with an equation that is a linear function of the states. It produces the manipulating signal that will take effect on the cart. In the above mentioned equation we not only need the states but also some kind of coefficients for each term. These coefficients will give the gains of the feedback. To calculate these values we have to decide the wanted characteristic of the system. We used LQR design (optimal calculation) and Butterworth filter, as well. However we made the model of both systems we realized only the second mentioned. We did some simulation of these systems using Simulink.

Since we had a realistic and well functioning model, we tried to implement it in practice. We made our application in LabView and used its predefined components to take the hardware. With NI IMAQ Vision we constructed an image processing algorithms to measure all of the states and build the control law.

By the result of the experimentations the rod is able to stay in balance for a few minutes without external assistance.

1 – INTRODUCTION

In this report we try to demonstrate the procedure of designing a feedback control system.

After we review the similar projects all around the world in Chapter 3 and Chapter 4 we present the whole physical model and the steps of the State Space Design.

In Chapter 5 we discuss the manner of the Pole Placement Control in details. Here you can come to know the two methods we used to find the values of the feedback gains. Beside the calculations a lot of simulations can be found in this chapter.

Chapter 6 introduces the hardware applications we used. We mention the features of the camera the grabber card, the I/O car, the servo drive and the AC motor.

In Chapter 7 the main image processing algorithm is presented in details, using the components of LabView.

Chapter 8 and Chapter 9 introduce two important topic of the solution: the manipulation of the servo drive and the calibration of the camera.

In Chapter 10 you can find the whole realization of the system. You can find how we measure the states and how we calculate the manipulating signal.

There were two groups working with the same pendulum model. They both managed to realize the system but one of them worked with higher order system. However their model was more complicated they achieved more precise result in their work.

2 – LITERATURE REVIEW

This topic, named Inverted Pendulum, is a very popular project in the whole world. If we write these two magical words to any search engine on the web, we will find several of information about Inverted Pendulum.

2.1 Full description about Inverted Pendulums

On most of the sites we can find information about State Space Design, open- or closed-loop representation, transfer functions and/or LQR design (for mentioned some examples). On www.engin.umich.edu/ group/ ctm/ examples/ pend/ invpen.html we can read an extremely detailed documentation about the mentioned topics, and lots of other, as well.

The owner of the site is the University of Michigan.

Our project is almost the same that this one, but several other projects exist (based inverted pendulum) which are quite different.

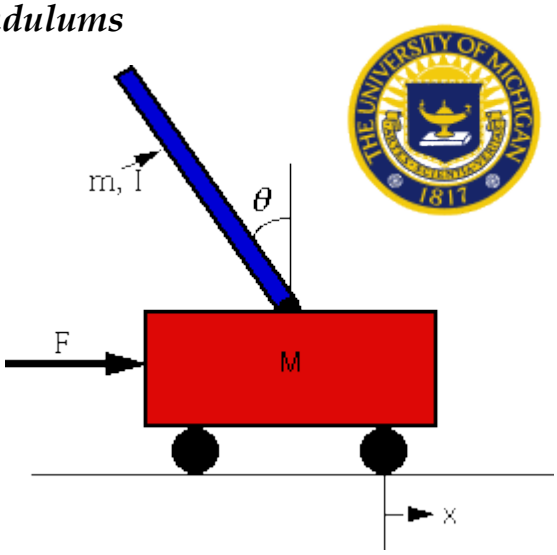
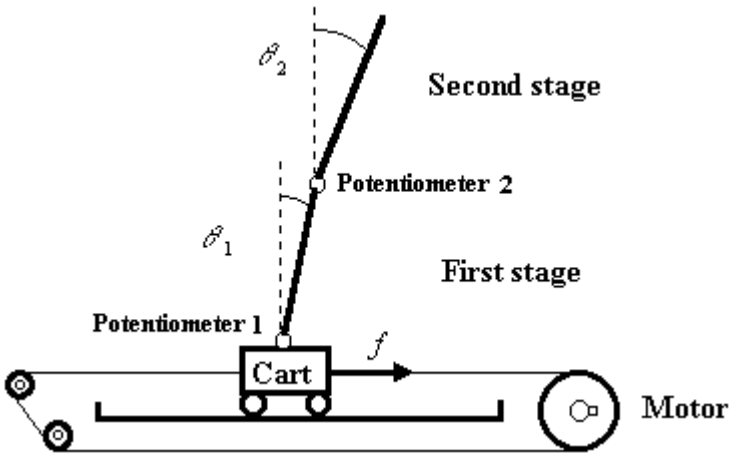


Figure 1: Inverted Pendulum Model

2.2 Two Stage Inverted Pendulum



Pendulum

We are working with a simple Inverted Pendulum, which rod consists of only one part. But other groups construct a more complicated rod, with two stages. For instance on www.aptronix.com/ fuzzynet/ applnote/ twostage.htm.

Figure 2: Two stage Inverted

2.3 Triple inverted pendulum

If we saw a pendulum with two stages, we will not be surprised hearing about the triple inverted pendulum on wwwa.mpi-magdeburg.mpg.de/ research/ pendel/ index_e.html.
Its control algorithm is implemented in C on a PC running under the operating system Realtime Linux with a Meilhaus ME2600 I/O board.
The owner of the pendulum is Max-Planck-Institut in Germany.

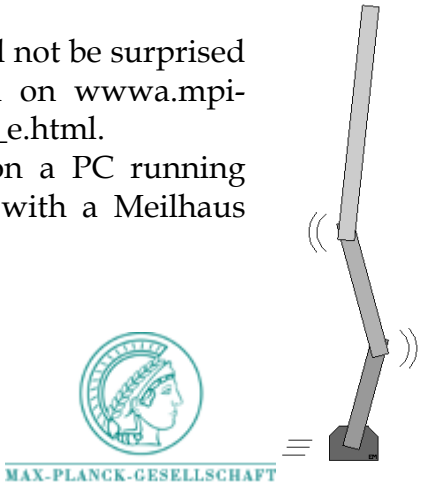
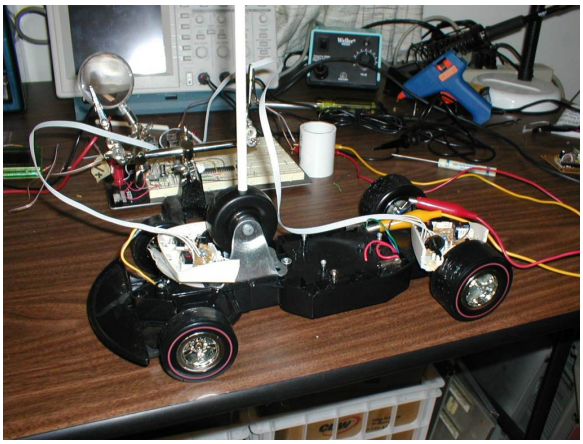


Figure 3: Triple Inverted Pendulum

2.4 Inverted Pendulum Car

An interesting way to design an inverted pendulum, when the cart not only can move in two directions on a rail, but also it is able to move in any other directions (of course in one plane). We add a bit more freedom to the inverted pendulum, if we equip the rod on a "real" car.



Some examples for this topic: www.obrador.com/ EE471Design/ EE471Design.htm and http:// 4north.no-ip.com:8080/ pics/ pendulum/. Thanks for James R. Weeks (Project Coordinator of EE471 Inverted Pendulum) and Josh Pieper (4north.no-ip.com).

Figure 4: Inverted Pendulum Car

2.5 Mathworks Inverted Pendulum

MathWorks is a famous group of several Matlab programs, and accessories. Naturally the Inverted Pendulum project is not an anon project for them: www.mathworks.com/ matlabcentral/ fileexchange/ loadFile.do ?objectId= 3790&objectType=file.

2.6 BaliBot, An Inverted Pendulum Robot

The final project which we demonstrate is a robot, named BaliBot. Its motion is based on an inverted pendulum system: http:// home.earthlink.net/ ~botronics/ index/ balibot.html.

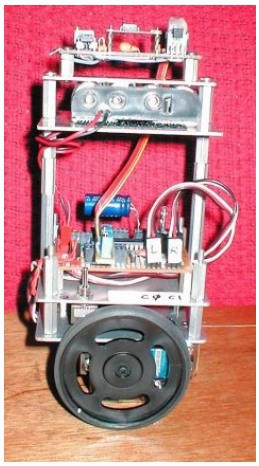


Figure 5: BaliBot

3 – MODELLING OF THE PHYSICAL SYSTEM

Designing a control system needs a well-thought-out mathematical and physical description of the system. However we can know well the control theory and be able to use the methods to design the correct system, if there is a wrong physical model it won't work properly.

3.1 The Model

First of all we represent the pure physical model in a co-ordinate system (**Figure 6**).

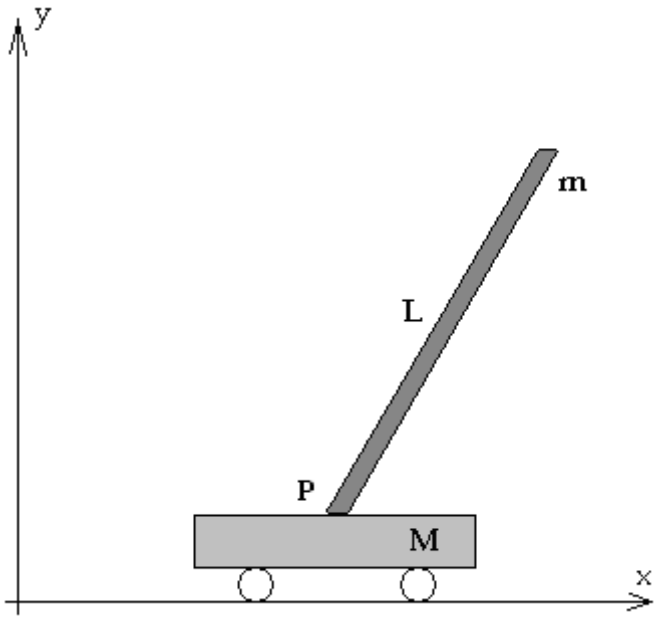


Figure 6: Pure physical model

Where *M* is the mass of the cart, *m* is the mass of the rod, and *L* is the length of it. The rod can rotate around the *P* point. The cart and the pendulum itself can only move in the *x* direction. The camera will show us similar frames about the cart and its movement.
After we know the schematic draw of the system we can examine the forces acting on the pendulum. **Figure 7** shows us the coordinates of the centre of gravity where the sum of the forces is acting.

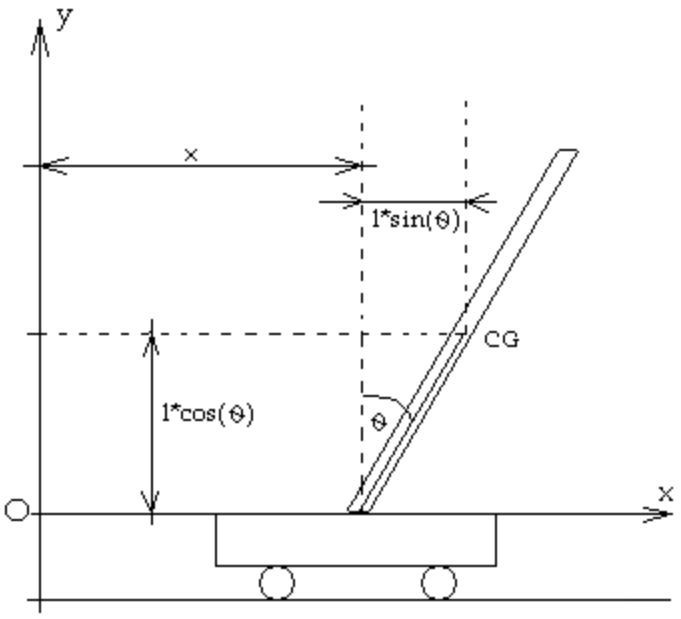


Figure 7: Co-ordinates of the centre of gravity

Since the steel rod we model in this section is homogeneous the forces act on the half of the length of it.

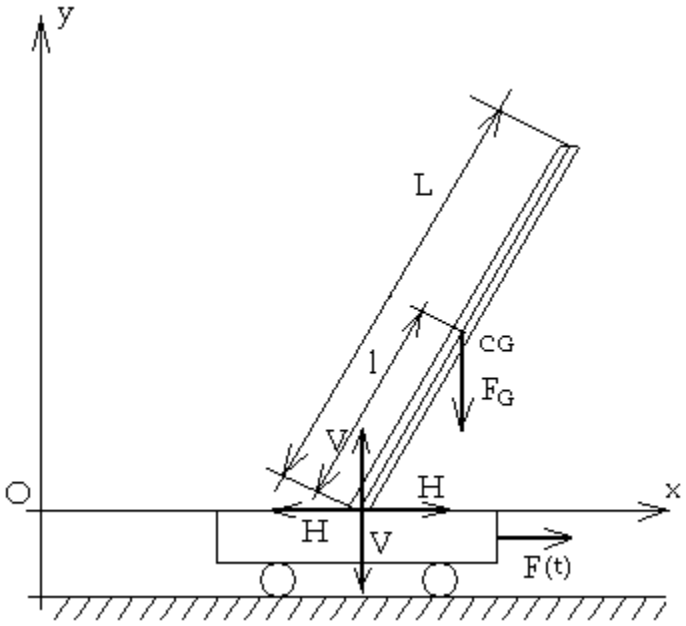


Figure 8: Physical forces

Since we neglect the friction of the cart and the pivot point we didn't note the forces of them. In the real model we can't dispense with the friction of course but we can solve the problem by changing the control system we made.

Notation of the figures (**Table 1**).

Name of datum	Description	Value
M	Mass of the cart	1 kg
m	Mass of the rod	0.1 kg
L	Length of the rod	0.74 m
CG	Center of gravity (rod)	
l	l=L/2	0.37 m
F_G	Force of gravity (rod)	
H	Force in Horizontal direction	
V	Force in Vertical direction	
□	Angle of the rod measured from vertical	
O	Orion (co-ordinate system)	
$F_{(t)}$	Puller force	

Table 1: Notation of the figures

3.2 Physical equations

The coordinates of the centre of the pendulum rod are (X_G,Y_G).

$$X_G = x + l\sin(\square) \quad \text{(M-1)}$$
$$Y_G = l\cos(\square) \quad \text{(M-2)}$$

The rotational motion of the pendulum rod about its centre of gravity can be described by (**M-3**) where J is the moment of inertia of the rod.

$$J\ddot{\Theta} = V l\sin(\square) - H l\cos(\square) \quad \text{(M-3)}$$

The horizontal motion of centre of gravity of pendulum rod is given by ((**M-4**)-(**M-7**)) where $\dot{X}_G$ is the velocity and $\ddot{X}_G$ is the acceleration of centre of gravity of pendulum rod in horizontal direction.

$$H = m\ddot{X}_G \quad \text{(M-4)}$$
$$\dot{X}_G = \dot{X} + l\cos(\square)\,\dot{\Theta} \quad \text{(M-5)}$$
$$\ddot{X}_G = \ddot{X} + l\cos(\square)\,\ddot{\Theta} - l\sin(\square)\,\dot{\Theta}^2 \quad \text{(M-6)}$$
$$H = m\ddot{X} + ml\cos(\square)\,\ddot{\Theta} - ml\sin(\square)\,\dot{\Theta}^2 \quad \text{(M-7)}$$

The vertical motion of centre of gravity of pendulum rod is ((**M-8**)-(**M-11**)) where $\dot{Y}_G$ is the velocity and $\ddot{Y}_G$ is the acceleration of centre of gravity of pendulum rod in vertical direction.

$$V - mg = m\ddot{Y}_G \quad \text{(M-8)}$$
$$\dot{Y}_G = -l\sin(\square)\,\dot{\Theta} \quad \text{(M-9)}$$
$$\ddot{Y}_G = -l\sin(\square)\,\ddot{\Theta} - l\cos(\square)\,\dot{\Theta}^2 \quad \text{(M-10)}$$
$$V = mg - ml\sin(\square)\,\ddot{\Theta} - ml\cos(\square)\,\dot{\Theta}^2 \quad \text{(M-11)}$$

The horizontal motion of the cart is described by (**M-12**).

$$M\ddot{X} = F_{(t)} - H \quad \text{(M-12)}$$

Now, we have four equations, which describe the physical model.

$$J\ddot{\Theta} = Vl\sin(\Theta) - Hl\cos(\Theta) \quad \text{(M-3)}$$
$$H = m\ddot{X} + ml\cos(\Theta)\ddot{\Theta} - ml\sin(\Theta)\dot{\Theta}^2 \quad \text{(M-7)}$$
$$V = mg - ml\sin(\Theta)\ddot{\Theta} - ml\cos(\Theta)\dot{\Theta}^2 \quad \text{(M-11)}$$
$$M\ddot{X} = F_{(t)} - H \quad \text{(M-12)}$$

If we substitute H from equation (**M-7**) for equation (**M-12**), we got a new equation about $F_{(t)}$ (**M-13**).

$$M\ddot{X} = F_{(t)} - m\ddot{X} + ml\cos(\Theta)\ddot{\Theta} - ml\sin(\Theta)\dot{\Theta}^2$$
$$\ddot{X}(M+m) + ml\cos(\Theta)\ddot{\Theta} - ml\sin(\Theta)\dot{\Theta}^2 = F_{(t)} \quad \text{(M-13)}$$

Consume the other two equations [(**M-3**) and (**M-11**)] and the above used (**M-7**) equation we got a new formula (**M-14**).

$$J\ddot{\Theta} = (mg - ml\sin(\Theta)\ddot{\Theta} - ml\cos(\Theta)\dot{\Theta}^2)l\sin(\Theta) - (m\ddot{X} + ml\cos(\Theta)\ddot{\Theta} - ml\sin(\Theta)\dot{\Theta}^2)l\cos(\Theta)$$
$$J\ddot{\Theta} = mgl\sin(\Theta) - m\ddot{X}l\cos(\Theta) - ml^2\ddot{\Theta}$$
$$(J + ml^2)\ddot{\Theta} - mgl\sin(\Theta) + m\ddot{X}l\cos(\Theta) = 0 \quad \text{(M-14)}$$

We got the two main equations of the system.

$$\ddot{X}(M+m) + ml\cos(\Theta)\ddot{\Theta} - ml\sin(\Theta)\dot{\Theta}^2 = F_{(t)} \quad \text{(M-13)}$$
$$(J + ml^2)\ddot{\Theta} - mgl\sin(\Theta) + m\ddot{X}l\cos(\Theta) = 0 \quad \text{(M-14)}$$

3.3 Simplification

These are the equations we will use in the further design, but there is a little problem about these correspondences. Since they involve sin(□) and cos(□) they are nonlinear equations. To get an easier approach we have to linearize them, so we neglect these elements. Of course we can't just leave but we should substitute them to get the linear form of the equations.

We suppose, that the angle of pendulum is a small angle, hence we use the following simplification ((**M-15**)-(**M-17**)).

$$\sin(\Theta) \cong \Theta \quad \text{(M-15)}$$
$$\cos(\Theta) \cong 1 \quad \text{(M-16)}$$
$$\dot{\Theta}^2 \cong 0 \quad \text{(M-17)}$$

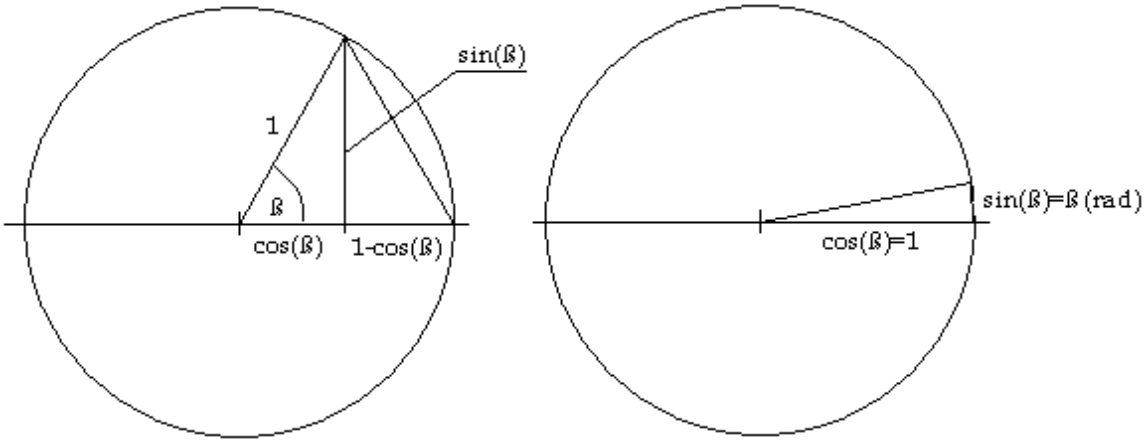


Figure 9: Explication about angles

Use the three simplifications mentioned above [(**M-15**), (**M-16**) and (**M-17**)], we got these new equations (**M-18** and **M-19**).

$$\ddot{X}\,(M+m) + ml\,\ddot{\Theta} = F_{(t)} \qquad \text{(M-18)}$$

$$(J + ml^2)\,\ddot{\Theta} - mgl\Box + m\ddot{X}\,l = 0 \qquad \text{(M-19)}$$

3.4 Completions

If we consider the friction, we have to add a new term to the (**M-18**) equation (**M-20**) where b is the friction coefficient acting on the cart. But we suppose that b ≈ 0.

$$\ddot{X}\,(M+m) + ml\,\ddot{\Theta} + b\dot{X} = F_{(t)} \qquad \text{(M-20)}$$

The moment of inertia of the rod is $(1/3)ml^2$. Use these data we get **M-21**.

$$\ddot{X}\,(M+m) + ml\,\ddot{\Theta} = F_{(t)} \qquad \text{(M-18)}$$

$$(\frac{1}{3}ml^2 + ml^2)\,\ddot{\Theta} - mgl\Box + m\ddot{X}\,l = 0$$

$$\frac{4l}{3}\,\ddot{\Theta} - g\Box + \ddot{X} = 0 \qquad \text{(M-21)}$$

Now, we denote $\ddot{\Theta}$ from equations (**M-18**) and (**M-21**).

$$\frac{4l}{3}\,\ddot{\Theta} - g\Box + \ddot{X} = 0 \qquad \text{(M-21)}$$

$$\ddot{X} = g\Box - \frac{4l}{3}\,\ddot{\Theta}$$

$$(g\Box - \frac{4l}{3}\,\ddot{\Theta})(M+m) + ml\ddot{\Theta} = F_{(t)}$$

$$g\Box(M+m) - \frac{l\ddot{\Theta}(4M+m)}{3} = F_{(t)}$$

$$g\Box(M+m) - F_{(t)} = \frac{1}{3}l\ddot{\Theta}\,(4M+m)$$

$$\ddot{\Theta} = \frac{3g(M+m)}{l(4M+m)}\Theta - \frac{3}{l(4M+m)}F_{(t)} \qquad \text{(M-22)}$$

After that, we denote $\ddot{X}$ from equations (**M-18**) and (**M-21**).

$$\frac{4l}{3}\ddot{\Theta} - g\square + \ddot{X} = 0 \qquad \text{(M-21)}$$

$$l = \frac{3g\Theta}{4\ddot{\Theta}} - \frac{3\ddot{X}}{4\ddot{\Theta}}$$

$$\ddot{X}\,(M+m) + \left(\frac{3g\Theta}{4\ddot{\Theta}} - \frac{3\ddot{X}}{4\ddot{\Theta}}\right)m\ddot{\Theta} = F_{(t)}$$

$$\ddot{X} = -\frac{3gm}{4M+m}\Theta + \frac{4}{4M+m}F_{(t)} \qquad \text{(M-23)}$$

4 – STATE SPACE ANALYSIS AND STATE SPACE DESIGN

The classical control theory is based on the relationship of the input and the output, working with the transfer function. In modern approach of control theory the main elements of the design are the system equations. They are written in terms of *n* first order differential equation which is combined to first order vector-matrix differential equations. The main advantage of this approach, that the increasing number of inputs and outputs won't increase the complexity of the equations.

4.1 Definition

The state of a system at any time in t_0 is the amount of information at t_0 that, together with all inputs for $t \geq t0$ uniquely determines the behaviour of the system for all $t \geq t0$.
The standard form of the state equation of a liner time-invariant system is given by:

$$\dot{X}(t) = AX(t) + Bu(t)$$

$$y(t) = CX(t) + Du(t)$$

where $\dot{X}(t)$ is the time derivative of the vector $X(t)$.

In the equations:

$X(t)$	vector of the states of an n-order system; size: n×1
A	system matrix; size: n×n
B	input matrix; size: n×r
u(t)	input vector; size: r×1
C	output matrix; size: p×1
D	matrix to represent direct coupling between input and output. (In most of the cases it is 0.)

If we describe the model by the system equations we can note the state equations of the system. The general form of the state equations allows more than one input and output. These systems are called: multivariable systems.

4.2 Beginning

From the physical model we have two equations which describe the system [(**M-22**) and (**M-23**)].

$$\ddot{\Theta} = \frac{3g(M+m)}{l(4M+m)}\Theta - \frac{3}{l(4M+m)}F_{(t)} \qquad \text{(M-22)}$$

$$\ddot{X} = -\frac{3gm}{4M+m}\Theta + \frac{4}{4M+m}F_{(t)} \qquad \text{(M-23)}$$

4.3 System variables of the 4th order open-loop system

The next step is to determine the system variables [(**S-1**)-(**S-4**)]! We chose the displacement, its derivate (the velocity), the angle of the rod and its derivate (the angular velocity), as well.

$$X_1 = X \tag{S-1}$$
$$X_2 = \dot{X} \tag{S-2}$$
$$X_3 = \Theta \tag{S-3}$$
$$X_4 = \dot{\Theta} \tag{S-4}$$

We affect on the system with $F_{(t)}$. Call its "input force" and denote it 'u' (**S-5**)!

$$F_{(t)} = u \tag{S-5}$$

4.4 System equations of the 4th order open-loop system

Then we prescribe the system equations (**S-6**)-(**S-9**).

$$\dot{X}_1 = X_2 \tag{S-6}$$
$$\dot{X}_2 = -\frac{3gm}{4M+m}\Theta + \frac{4}{4M+m}F_{(t)} \tag{S-7}$$
$$\dot{X}_3 = X_4 \tag{S-8}$$
$$\dot{X}_4 = \frac{3g(M+m)}{l(4M+m)}\Theta - \frac{3}{l(4M+m)}F_{(t)} \tag{S-9}$$

We can write it in vector-matrix format, as well:

$$\begin{bmatrix}\dot{X}_1\\ \dot{X}_2\\ \dot{X}_3\\ \dot{X}_4\end{bmatrix} = \begin{bmatrix}0 & 1 & 0 & 0\\ 0 & 0 & -\frac{3gm}{4M+m} & 0\\ 0 & 0 & 0 & 1\\ 0 & 0 & \frac{3g(M+m)}{l(4M+m)} & 0\end{bmatrix} * \begin{bmatrix}X_1\\ X_2\\ X_3\\ X_4\end{bmatrix} + \begin{bmatrix}0\\ \frac{4}{4M+m}\\ 0\\ -\frac{3}{l(4M+m)}\end{bmatrix} * \bar{u}$$

$$\bar{y} = \begin{bmatrix}1 & 0 & 1 & 0\end{bmatrix} * \begin{bmatrix}X_1\\ X_2\\ X_3\\ X_4\end{bmatrix}$$

So the state variables can be written in the next form [(**S-10**)-(**S-13**)].

$$\dot{X}_1 = X_2 \tag{S-10}$$
$$\dot{X}_2 = \frac{1}{4M+m}\left(-3gmX_3 + 4u\right) \tag{S-11}$$

$$\dot{X}_3 = X_4 \tag{S-12}$$

$$\dot{X}_4 = \frac{3}{l(4M+m)}\left(g(M+m)X_3 - u\right) \tag{S-13}$$

4.5 Implement the 4th order open-loop system in Matlab Simulink

The system given by the state equations can be also described with a simulation diagram. It demonstrates the connection between the states and we can also simulate the response of the system for one or more given inputs. We can edit the simulation diagram in Matlab Simulink (**Figure 10** and **Figure 11**).

Script file name: DIP01_StateSpaceAnalysisM.m

Model file name: DIP01_StateSpaceAnalysisMDL.mdl

```
%David2 Inverted pendulum

%Constants
l=0.5; %(Length of pendulum)/2 [m]
m2=0.1; %Mass of pendulum [kg]
M1=1.0; %Mass of cart [kg]
g=9.81; %Gravitation [m/s^2]

F_step=1; %[s]
F0=0;
F1=0.1; %[N]

tstep=0.01;
tstart=0;
tstop=6;
tspan=[tstart tstop];
options=simset('solver','ode5','fixedstep',tstep);
%Start sim
sim('DIP01_StateSpaceAnalysisMDL',tspan,options);

figure(1);
clf;

%Plot the position of the cart and the angle of the pendulum
plot(t,y,t,teta)
axis([0 tstop -1.5 1.5])
title('David2 Inverted Pendulum')
xlabel('Time [s]')
ylabel('Position [m], Angle [rad]')
legend('Cart position','Angle of pendelum')
```

Figure 10: Matlab 7 Simulink Model

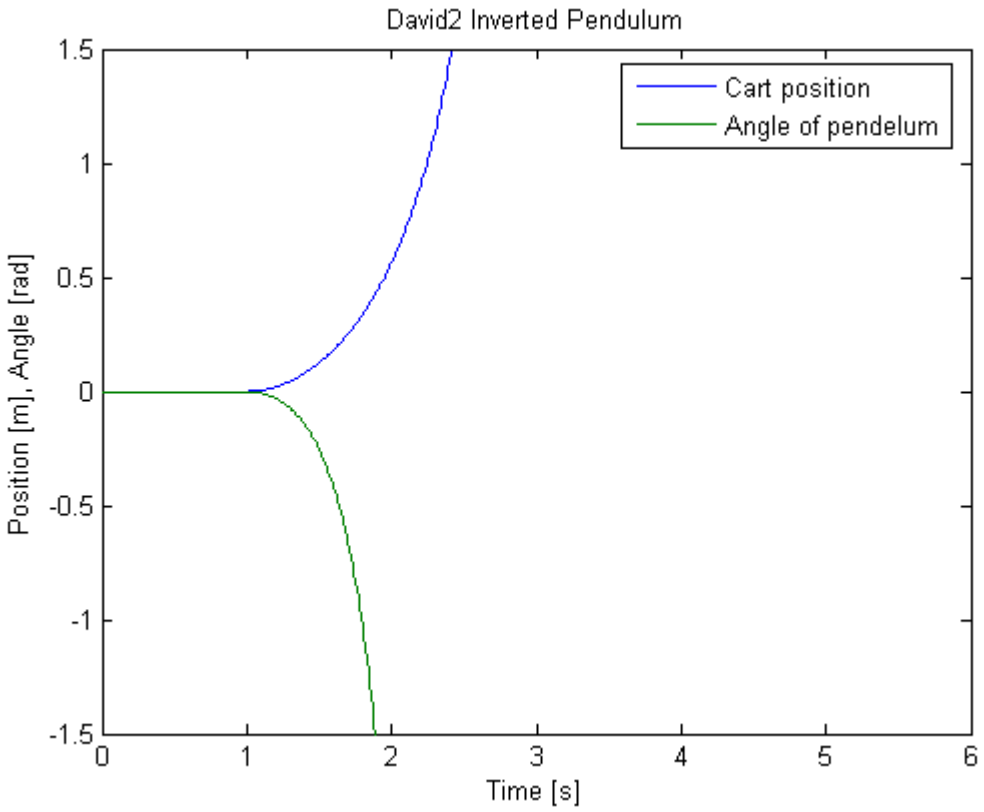


Figure 11: Response of the 4th order open-loop system

4.6 Simplification

In the design we endeavour to take the complication of the system as low as we can. So that we get a simplification to get 3rd order system instead of the 4th order system we have now. To achieve it we have to bring in some new data.
So we have two new equations [(**S-14**) and (**S-15**)] with the use of the new data.

$$\square = K_p u \quad \text{(S-14)}$$

$$\dot{X} = K_p R u \quad \text{(S-15)}$$

Name of data	Description	Value
K_p	Motor constant	31.3 (radV)/s
R	Radius of the shaft	0.015 m
□	Velocity of the angle	

Table 2:New data

We start the new calculation from an earlier equation (**M-21**).

$$\frac{4l}{3}\ddot{\Theta} - g\Theta + \ddot{X} = 0 \qquad \text{(M-21)}$$

$$\ddot{\Theta} = \frac{3}{4l}\left(g\Theta - \ddot{X}\right)$$

$$\dot{\Theta} = \int \ddot{\Theta} dt = \frac{3}{4l}\left(-\dot{X} + g\int \Theta dt\right)$$

$$\dot{\Theta} = -\frac{3}{4l}\dot{X} + \frac{3}{4l}\int g\Theta dt \qquad \text{(S-16)}$$

4.7 System variables of the 3rd order open-loop system

Our new state variables are (**S-17**), (**S-18**) and (**S-19**).

$$X_1 = X \qquad \text{(S-17)}$$

$$X_2 = \Theta \qquad \text{(S-18)}$$

$$X_3 = \int g\Theta dt \qquad \text{(S-19)}$$

4.8 System equations of the 3rd order open-loop system

Let us see the state equations [(**S-20**)-(**S-22**)].

$$\dot{X}_1 = K_p R u \qquad \text{(S-20)}$$

$$\dot{X}_2 = -\frac{3}{4l}\dot{X}_1 + \frac{3}{4l}X_3 = -\frac{3}{4l}K_p R u + \frac{3}{4l}X_3 \qquad \text{(S-21)}$$

$$\dot{X}_3 = gX_2 \qquad \text{(S-22)}$$

Now we can write vector-matrix format too.

$$\begin{bmatrix}\dot{X}_1\\ \dot{X}_2\\ \dot{X}_3\end{bmatrix} = \begin{bmatrix}0 & 0 & 0\\ 0 & 0 & \frac{3}{4l}\\ 0 & g & 0\end{bmatrix} * \begin{bmatrix}X_1\\ X_2\\ X_3\end{bmatrix} + \begin{bmatrix}K_p R\\ -\frac{3}{4l}K_p R\\ 0\end{bmatrix} * \bar{u}$$

$$\bar{y} = \begin{bmatrix}1 & 1 & 0\end{bmatrix} * \begin{bmatrix}X_1\\ X_2\\ X_3\end{bmatrix}$$

$$A = \begin{bmatrix}0 & 0 & 0\\ 0 & 0 & \frac{3}{4l}\\ 0 & g & 0\end{bmatrix} \quad B = \begin{bmatrix}K_p R\\ -\frac{3}{4l}K_p R\\ 0\end{bmatrix} \quad C = \begin{bmatrix}1 & 1 & 0\end{bmatrix}$$

Replacing the constants we get the following vector-matrix format.

$$\begin{bmatrix}\dot{X}_1\\ \dot{X}_2\\ \dot{X}_3\end{bmatrix} = \begin{bmatrix}0 & 0 & 0\\ 0 & 0 & 2.027\\ 0 & 9.81 & 0\end{bmatrix} * \begin{bmatrix}X_1\\ X_2\\ X_3\end{bmatrix} + \begin{bmatrix}0.4695\\ -0.951676\\ 0\end{bmatrix} * \bar{u}$$

4.9 Implement the 3rd order open-loop system in Matlab Simulink

By the result of the simplification we have only three state variables, so our system is now a 3rd order system. It makes our job easier. Because of the change of the states our simulation diagram will also change. The modified diagram is shown on **Figure 12**.

Script file name: DIP02_StateSpaceAnalysisM.m

Model file name: DIP02_StateSpaceAnalysisMDL.mdl

```
%David2 Inverted pendulum

%Constants
l=0.37; %(Length of pendulum)/2 [m]
g=9.81; %Gravitation [m/s^2]
R=0.015
Kp=31.3

F_step=1; %[s]
F0=0;
F1=0.1; %[N]

tstep=0.01;
tstart=0;
tstop=6;
tspan=[tstart tstop];
options=simset('solver','ode5','fixedstep',tstep);
%Start sim
sim('DIP02_StateSpaceAnalysisMDL',tspan,options);

figure(1);
clf;

%Plot the position of the cart and the angle of the pendulum
plot(t,x,t,theta)
axis([0 tstop -1.5 1.5])
title('David2 Inverted Pendulum')
xlabel('Time [s]')
ylabel('Position [m], Angle [rad]')
legend('Cart position','Angle of pendelum')
```

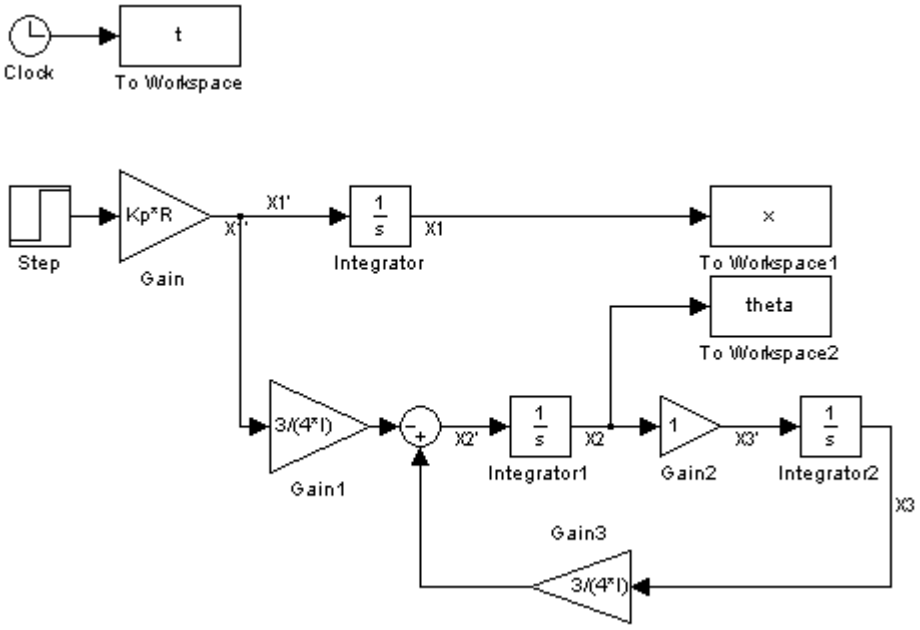


Figure 12: Simulation diagram of the 3rd order open-loop system

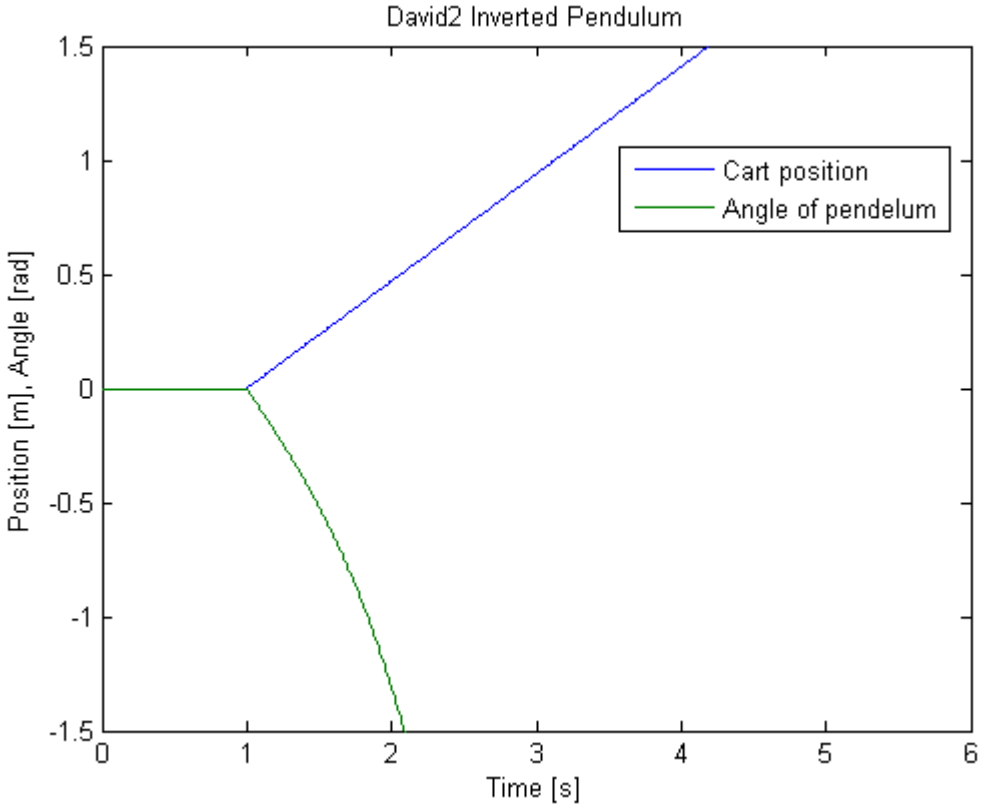


Figure 13: Response of the 3rd order open-loop system

5 – POLE-PLACEMENT CONTROL

Pole-placement design is part of the modern control theory. It is based on the state-space model and the state equations. To be able to use the pole placement we have to know the control law. In general, the plant input is u(t) is made of function of the states.

$$u(t)=f[x(t)]$$

We have to use linear time-invariant analogue systems for pole-placement. We have to write our model in the following form [(**P-1**) and (**P-2**)].

$$\dot{\bar{X}} = A\bar{X} + B\bar{u} \quad \text{(P-1)}$$

$$\bar{y} = C\bar{X} \quad \text{(P-2)}$$

A = System matrix
B = Input matrix
C = Output matrix

In pole-placement design, the control law is specified as a linear function of the states (**P-3**).

$$\bar{u} = -\begin{bmatrix} k_1 & k_2 & k_3 \end{bmatrix} * \bar{X} \quad \text{(P-3)}$$

To be able to change the poles of the system we should know the original poles. To calculate it we need the matrices describe the system. With the help of Matlab it is very easy to get the original poles.
From previous calculations we know that:

$$A = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & \frac{3}{4l} \\ 0 & g & 0 \end{bmatrix} \quad B = \begin{bmatrix} K_p R \\ -\frac{3}{4l} K_p R \\ 0 \end{bmatrix} \quad C = \begin{bmatrix} 1 & 1 & 0 \end{bmatrix} \quad D = 0$$

We shall use the "pole" command in Matlab which waits a State Space Object for input. To get this object first we have to use the "ss" command, after that we can use the above mentioned "pole" command.

Script file name: DIP03_PolePlacementM1.m

```
l=0.37;
m=0.1;
R=0.015;
g=9.81;
Kp=31.3;

A=[0, 0, 0;0, 0,3/(4*l);0, g, 0];
B=[Kp*R;-3*Kp*R/(4*l);0];
C=[1 1 0];
D=0;

psys=ss(A,B,C,D)
```

```
pole(psys)
```

Result:

```
a =
          x1      x2      x3
   x1      0       0       0
   x2      0       0   2.027
   x3      0    9.81       0

b =
            u1
   x1   0.4695
   x2  -0.9517
   x3        0

c =
       x1  x2  x3
   y1   1   1   0

d =
       u1
   y1   0

Continuous-time model.

ans =
    4.4593
   -4.4593
         0
```

We got the poles of the original system. We can see that the system is unstable in this condition because there is a positive item in it. To make the system stable we should place the poles to the left half of the s-plane.

From equation (**P-1**) and (**P-3**) we can get a new function about $\dot{\overline{X}}(t)$.

$$\dot{\overline{X}} = A\overline{X} + B\overline{u} \qquad \text{(P-1)}$$

$$\overline{u} = -K\overline{X} \qquad \text{(P-3)}$$

$$\dot{\overline{X}} = A\overline{X} + B\left(-K\overline{X}\right)$$

$$\dot{\overline{X}} = \overline{X}\left(A - BK\right) \qquad \text{(P-6)}$$

5.1 Calculate the new vector-matrix format of the 3rd order system

Adopting the equations (**P-6**).

$$\begin{bmatrix} K_pR \\ -\frac{3}{4l}K_pR \\ 0 \end{bmatrix} * \begin{bmatrix} k_1 & k_2 & k_3 \end{bmatrix} = \begin{bmatrix} k_1K_pR & k_2K_pR & k_3K_pR \\ -k_1\frac{3}{4l}K_pR & -k_2\frac{3}{4l}K_pR & -k_3\frac{3}{4l}K_pR \\ 0 & 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix}\dot{X}_1\\ \dot{X}_2\\ \dot{X}_3\end{bmatrix} = \begin{bmatrix}-k_1K_pR & -k_2K_pR & -k_3K_pR\\ k_1\dfrac{3}{4l}K_pR & k_2\dfrac{3}{4l}K_pR & \dfrac{3}{4l}+k_3\dfrac{3}{4l}K_pR\\ 0 & g & 0\end{bmatrix} * \begin{bmatrix}X_1\\ X_2\\ X_3\end{bmatrix} = \bar{X}(A-BK)$$

$$A' = \begin{bmatrix}-k_1K_pR & -k_2K_pR & -k_3K_pR\\ k_1\dfrac{3}{4l}K_pR & k_2\dfrac{3}{4l}K_pR & \dfrac{3}{4l}+k_3\dfrac{3}{4l}K_pR\\ 0 & g & 0\end{bmatrix}$$

5.2 Characteristic equation of the 3rd order closed-loop system

A' is the system matrix for the closed-loop system. The characteristic equation for the closed-loop system is (**P-7**).

$$|\lambda I - A'| = 0 \qquad \text{(P-7)}$$

Calculate this:

$$\lambda * \begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{bmatrix} = \begin{bmatrix}\lambda & 0 & 0\\ 0 & \lambda & 0\\ 0 & 0 & \lambda\end{bmatrix}$$

$$\begin{bmatrix}\lambda & 0 & 0\\ 0 & \lambda & 0\\ 0 & 0 & \lambda\end{bmatrix} - A' = \begin{bmatrix}\lambda + k_1K_pR & k_2K_pR & k_3K_pR\\ -k_1\dfrac{3}{4l}K_pR & \lambda - k_2\dfrac{3}{4l}K_pR & -\dfrac{3}{4l}-k_3\dfrac{3}{4l}K_pR\\ 0 & -g & \lambda\end{bmatrix}$$

Replacing the constants we get the following matrix:

$$\begin{bmatrix}\lambda + 0.4695k_1 & 0.4695k_2 & 0.4695k_3\\ -0.951689k_1 & \lambda - 0.951689k_2 & -2.027 - 0.951689k_3\\ 0 & -9.81 & \lambda\end{bmatrix} = \lambda I - A'$$

Let us calculate the equation (**P-7**).

$$|\lambda I - A'| = 0 \qquad \text{(P-7)}$$

$$\left|\begin{bmatrix}\lambda + 0.4695k_1 & 0.4695k_2 & 0.4695k_3\\ -0.951689k_1 & \lambda - 0.951689k_2 & -2.027 - 0.951689k_3\\ 0 & -9.81 & \lambda\end{bmatrix}\right| = 0$$

$$(\lambda + 0.4695k_1)(\lambda^2 - 0.951689k_2\lambda - 19.88487 - 9.33606k_3) + 0.4695k_2 * 0.951689k_1\lambda +$$
$$+ 0.4695k_3 * 9.33606k_1 = 0$$
$$\lambda^3 - 0.951689k_2\lambda^2 - 19.88487\lambda - 9.33606k_3\lambda + 0.4695k_1\lambda^2 - 0.446817k_1k_2\lambda - 9.335946k_1 -$$
$$- 4.38238k_1k_3 + 0.446817k_1k_2\lambda + 4.38238k_1k_3 = 0$$
$$\lambda^3 - (0.951689k_2 - 0.4695k_1)\lambda^2 + (-19.88487 - 9.33606k_3)\lambda - 9.335946k_1 = 0$$

We will get three more simple equations, if we adopt the following common fact (**P-8**).

$$(s - \lambda_1)(s - \lambda_2)(s - \lambda_3) = 0$$
$$(s^2 - \lambda_2 s - \lambda_1 s + \lambda_1\lambda_2)(s - \lambda_3) = 0$$
$$s^3 - \lambda_3 s^2 - \lambda_2 s^2 + \lambda_2\lambda_3 s - \lambda_1 s^2 + \lambda_1\lambda_3 s + \lambda_1\lambda_2 s - \lambda_1\lambda_2\lambda_3 = 0$$
$$s^3 - (\lambda_1 + \lambda_2 + \lambda_3)s^2 + (\lambda_1\lambda_2 + \lambda_1\lambda_3 + \lambda_2\lambda_3)s - \lambda_1\lambda_2\lambda_3 = 0 \qquad \text{(P-8)}$$

5.3 Equations with K and λ values

Now we can denote the λ values with k_1, k_2 and k_3 that will be the loop-back gains of the controlled system [(**P-9**)-(**P-11**)].

$$\lambda_1 + \lambda_2 + \lambda_3 = 0.951689k_2 - 0.4695k_1 \qquad \text{(P-9)}$$
$$\lambda_1\lambda_2 + \lambda_1\lambda_3 + \lambda_2\lambda_3 = -19.88487 - 9.33606k_3 \qquad \text{(P-10)}$$
$$\lambda_1\lambda_2\lambda_3 = 9.335946k_1 \qquad \text{(P-11)}$$

5.4 The closed loop system in Matlab SimuLink

Since we introduced the K values now we have a closed loop system. So we have to modify the simulation diagram to build in the feedback loop. It is shown by the Figure 14.

Model file name: DIP04_PolePlacementMDL.mdl

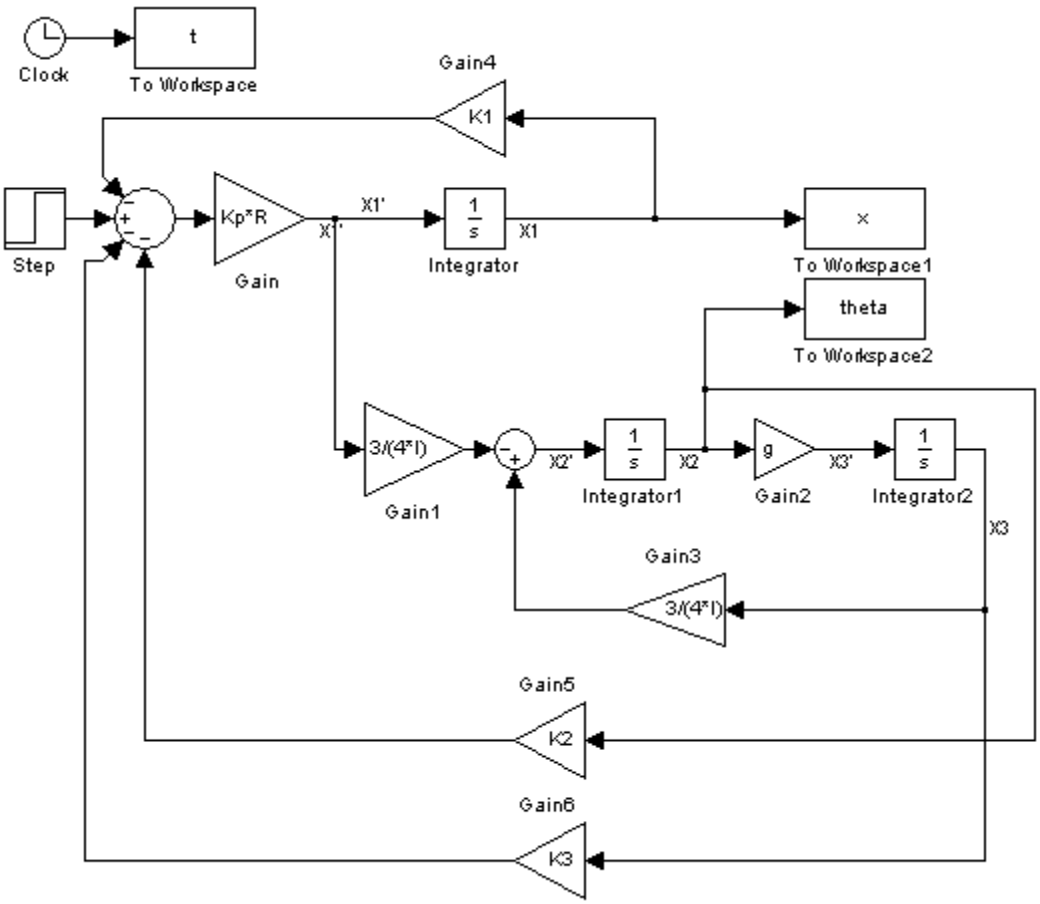


Figure 14: Closed loop of our 3rd order system

5.5 Determine K values with LQR Design

To get the correct *k* values we have to decide the characteristic of the response of the system. We can use predefined characteristics or methods. For example using Linear Quadratic Regulator (LQR) method we can calculate the optimal gain matrix K we saw in the *control law*.
The substance of this method to minimize the J(u) cost function denoted here:

$$J(u) = \int_0^\infty (x^T Qx + u^T Ru + 2x^T Nu)dt$$

for the continuous-time state-space model: $\dot{\overline{X}} = A * \overline{X} + B * \overline{u}$.
In most of the cases N=0. We can calculate K values with LQR method using Matlab. During the test we defined Q=I and R=1, where Q is a diagonal matrix (where its rank equals to the number of the state variables), which defines the weight of the state variables. I is the identity matrix. We got the under mentioned values:

```
>> A
A =
```

```
         0         0         0
         0         0    2.0270
         0    9.8100         0

>> B
B =
    0.4695
   -0.9517
         0

>> I = eye(size(A))
I =
     1     0     0
     0     1     0
     0     0     1

>> [K,S,e] = LQR(A,B,I,1)
K =
   -1.0000  -10.6107   -4.9048

S =
    2.7749    2.4197    1.0816
    2.4197   12.3433    5.6874
    1.0816    5.6874    2.7547

e =
  -4.6021 + 0.9061i
  -4.6021 - 0.9061i
  -0.4244
```

Result:

$k_1 = -1$

$k_2 = -10.6107$

$k_3 = -4.9048$

The LQR method gives us an optimum calculation of the K values. Since in the testing phase, we can change our values according to the system behaviour. If we use the K values calculated by LQR probably we can find better values changing the weights of the variables in matrix Q.

5.6 Implement the closed-loop system with LQR Design in Matlab

Since we have the simulation diagram of the closed-loop system we just use the K values calculated above. Running the modified version of the script we wrote in Chapter STATE SPACE ANALYSIS AND STATE SPACE DESIGN we can see the response of the relevant system. Using these values we got the response shown on **Figure 15**.

Script file name: DIP04_PolePlacementM2.m

Model file name: DIP04_PolePlacementMDL.mdl

```
%David2 Inverted pendulum
%Constants
l=0.37; %(Length of pendulum)/2 [m]
g=9.81; %Gravitation [m/s^2]
R=0.015
Kp=31.3
%Loop-back
K1=-1;
K2=-10.6107;
K3=-4.9048;

F_step=1; %[s]
F0=0;
F1=0.1; %[N]

tstep=0.01;
tstart=0;
tstop=15;
tspan=[tstart tstop];
options=simset('solver','ode5','fixedstep',tstep);
%Start sim
sim('DIP04_PolePlacementMDL',tspan,options);

figure(1);
clf;

%Plot the position of the cart and the angle of the pendulum
plot(t,x,t,theta)
axis([0 tstop -1.5 1.5])
title('David2 Inverted Pendulum')
xlabel('Time [s]')
ylabel('Position [m], Angle [rad]')
legend('Cart position','Angle of pendelum')
```

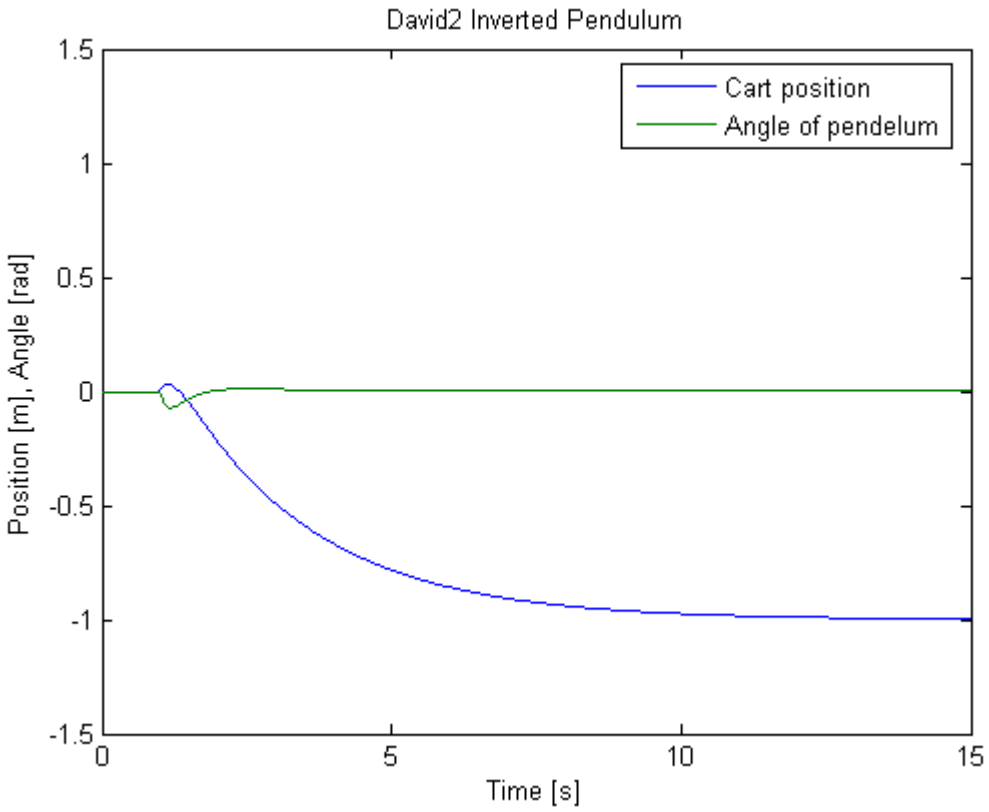


Figure 15: Response of the closed-loop system with the K values calculated by the LQR method

5.7 Determine K values with Butterworth filter

The other way we used to calculate the gain matrix K was the approximation of the characteristic with the Butterworth polynom. Since we use the step function to simulate the input we should examine the response of the Butterworth filter to be proven it will be a good characteristic for our system. It is shown by **Figure 16**.

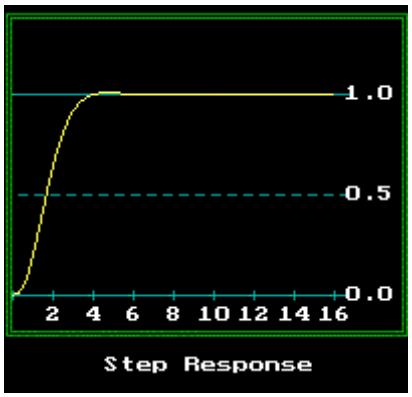


Figure 16: Response of a 3rd order normalized Butterworth polynom for a step input

So we can see the advantages of the filter: short response time and little overshoot.
The Butterworth polynom has a general form:

$$s^n + a_{n-1}s^{n-1} + \ldots + a_2 s^2 + a_1 s + 1 \ldots$$

Where the a_n coefficients are shown on the **Table 3**.

n	a_1	a_2	a_3	a_4	a_5	a_6	a_7
2	1.414214						
3	2.000000	2.000000					
4	2.613126	3.414214	2.613126				
5	3.236068	5.236068	5.236068	3.236068			
6	3.863703	7.464102	9.141620	7.464102	3.863703		
7	4.493959	10.097835	14.591794	14.591794	10.097835	4.493959	
8	5.125831	13.137071	21.846151	25.688356	21.846151	13.137071	5.125831
9	5.758770	16.581719	31.163437	41.986386	41.986386	31.163437	16.581719
10	6.392453	20.431729	42.802061	64.882396	74.233429	64.882396	42.802061

Table 3: Coefficients for Butterworth polynom

In our case the Butterworth polynom is like (**P-12**).

$$s^3 + 2s^2 + 2s + 1 \qquad \text{(P-12)}$$

We want our system to have fast response so we can write the following equation:

$$T_r = \frac{T_p}{2}$$

where T_r is the time of the response and T_p is the process time.

We can denote T_r from the equation:

$$T_r \cong \frac{k}{\omega_0}$$

where ω_0 is the cut-off frequency and k is a coefficient which is 1,5 rad for 2nd, 3rd, and 4th order systems (it is equal 1 rad for 1st order systems).
If we assume that the process time is 1.5 second ($T_p = 1.5$) we can get the value of ω_0.

$$\omega_0 = \frac{k}{T_r} = \frac{1.5}{\frac{T_p}{2}} = \frac{1.5}{0.75} = 2\left[\frac{rad}{\sec}\right] \qquad \text{(P-13)}$$

The Butterworth polynom given above is a normalized form. In that case $\omega_0 = 1$. In our case we have to use a substitution in the polynom:

$$s = \frac{s}{\omega_0}$$

Suit this expression to the polynom (**P-12**) will be (**P-14**).

$$\frac{s^3}{\omega_0^3} + \frac{2s^2}{\omega_0^2} + \frac{2s}{\omega_0} + 1 = 0 \qquad \text{(P-14)}$$

If we simplify it we get (**P-15**).

$$\frac{1}{\omega_0^3}(s^3 + 2\omega_0 s^2 + 2\omega_0^2 s + \omega_0^3) = 0 \qquad \text{(P-15)}$$

Substitute the values given above (**P-16**).

$$\frac{1}{8}(s^3 + 4s^2 + 8s + 8) = 0 \qquad \text{(P-16)}$$

Use the equation (**P-8**) we get a new equation (**P-17**).

$$s^3 + 4s^2 + 8s + 8 = s^3 - (\lambda_1 + \lambda_2 + \lambda_3)s^2 + (\lambda_1\lambda_2 + \lambda_1\lambda_3 + \lambda_2\lambda_3)s - \lambda_1\lambda_2\lambda_3 \qquad \text{(P-17)}$$

Use the equations (**P-9**)-(**P-11**) we get new equations:

$$\lambda_1 + \lambda_2 + \lambda_3 = 0.951689k_2 - 0.4695k_1 = -4 \qquad \text{(P-18)}$$
$$\lambda_1\lambda_2 + \lambda_1\lambda_3 + \lambda_2\lambda_3 = -19.88487 - 9.33606k_3 = 8 \qquad \text{(P-19)}$$
$$\lambda_1\lambda_2\lambda_3 = 9.335946k_1 = -8 \qquad \text{(P-20)}$$

Now we can calculate the K values:

$$k_1 = -0.8569$$
$$k_2 = -4.6258$$
$$k_3 = -2.9868$$

5.8 Implement the closed-loop system with Butterworth filters

If we use these values in the feedback loop we get a new response function of the system (**Figure 17**).

Script file name: DIP05_PolePlacementM3.m

Model file name: DIP04_PolePlacementMDL.mdl

```
%David2 Inverted pendulum
%Constants
l=0.37; %(Length of pendulum)/2 [m]
g=9.81; %Gravitation [m/s^2]
R=0.015
Kp=31.3
%Loop-back
K1=-0.8569;
K2=-4.6258;
K3=-2.9868;

F_step=1; %[s]
F0=0;
F1=0.1; %[N]

tstep=0.01;
tstart=0;
tstop=15;
tspan=[tstart tstop];
options=simset('solver','ode5','fixedstep',tstep);
%Start sim
sim('DIP04_PolePlacementMDL',tspan,options);

figure(1);
clf;

%Plot the position of the cart and the angle of the pendulum
plot(t,x,t,theta)
axis([0 tstop -2 2])
title('David2 Inverted Pendulum')
xlabel('Time [s]')
ylabel('Position [m], Angle [rad]')
legend('Cart position','Angle of pendelum')
```

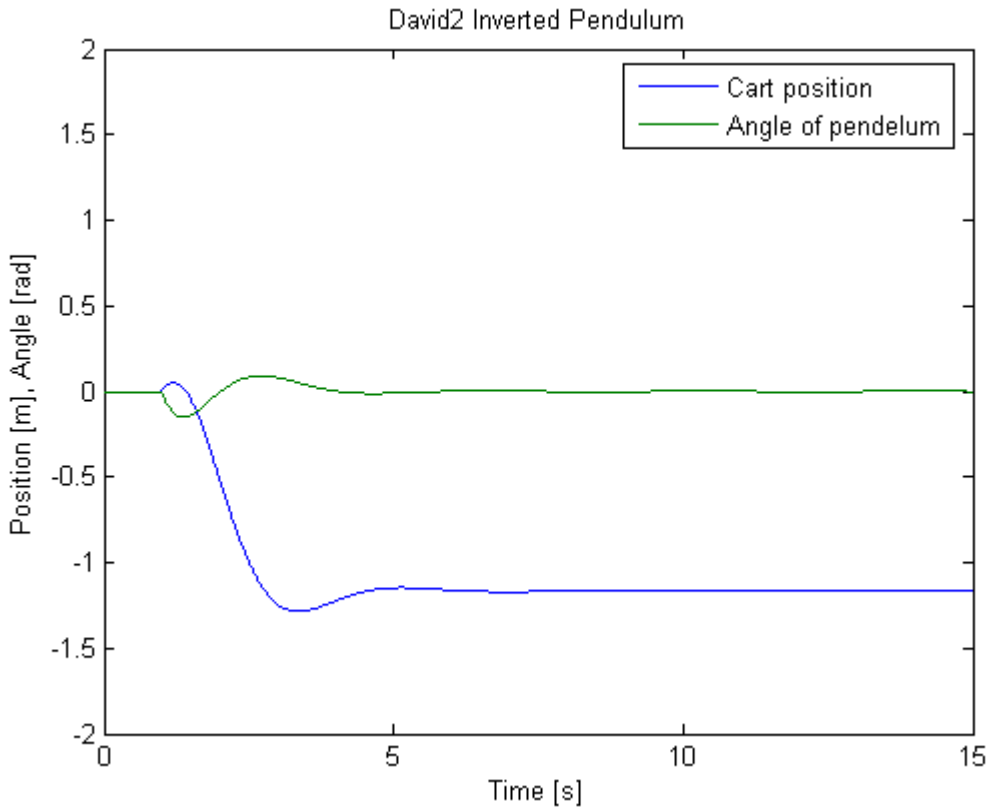


Figure 17: Response of the closed-loop system with the K values calculated by the Butterworth polynom

5.9 Mathematical calculation using Matlab

If we would like to compute the whole calculation with another A and B matrix (for instance our motor constant has changed), and we use only a calculating machine, we pay for this many time. There is an easier way using Matlab. Write the following M-file, and let us see the result. It will be the same we got.

Script file name: DIP06_PolePlacementM4.m

```
l=0.37;
m=0.1;
R=0.015;
g=9.81;
Kp=31.3;
A=[0, 0, 0;0, 0,3/(4*l);0, g, 0];
B=[Kp*R;-3*Kp*R/(4*l);0];
w0=2;
a=conv([1 w0 w0^2],[1 w0]);
regulatedpoles=roots(a);

K = place(A,B,regulatedpoles)
```

Result:

```
K =

   -0.8569   -4.6258   -2.9868
```

6 – HARDWARE

6.1 Camera

We will grab the pictures with a JAI Compact Industrial Monochrome CCD Camera CV-A50C (**Figure 18**). It has two different resolutions: CCIR [752x582] and EIA [768x494]. We will use the first one (CCIR). Further information can be read in **Appendix F**.

Figure 18: JAI CV-A50C

We use this camera with a Pentax C815B(C30811) lens. Its details can be found in (**Table 4**).

Type	**C815B(C30811)**
Format Size	2/3
Focal length	8.5mm
Max. Aperture Ratio	1:1.5
Horizontal 1/4	24.02
Angel of View 1/3	31.87
(Degrees) 1/2	42.09
Min. Object Distance	0.2m
Back Focal Length	10.9mm
Filter Size	40.5mm P=0.5mm
Mount (Flange back)	C (17.526mm)
Weight	120g
Remarks	Lock Screw Extra

Table 4: Pentax C815B(C30811)

6.2 Frame grabber card

We have to connect the camera to the PC with an adequate grabber card. We use for this purpose a National Instruments IMAQ PCI-1407 / PXI-1407 monochrome analog framegrabber (**Figure 19**).
The National Instruments 1407 devices are ideal for machine vision and scientific imaging end users and OEM developers. An NI 1407 has a single high-accuracy

monochrome video input, external triggering capabilities, and easy-to-use image acquisition driver software. NI 1407 advanced features include partial image scanning, programmable gain and offset, and onboard decimation and LUT processing. For easy configuration of both RS-170 and CCIR monochrome cameras, NI 1407 devices include NI-IMAQ image acquisition driver software and the NI Measurement & Automation Explorer configuration utility.
Details:

- Synchronization for multiple camera acquisition
- Partial image acquisition; 256-byte look-up (LUT) table
- Programmable gain and offset
- Pixel jitter less than 2 ns
- NI-IMAQ software for Windows 2000/NT/XP
- 1-channel monochrome image acquisition board for standard video (RS-170 and CCIR VGA)

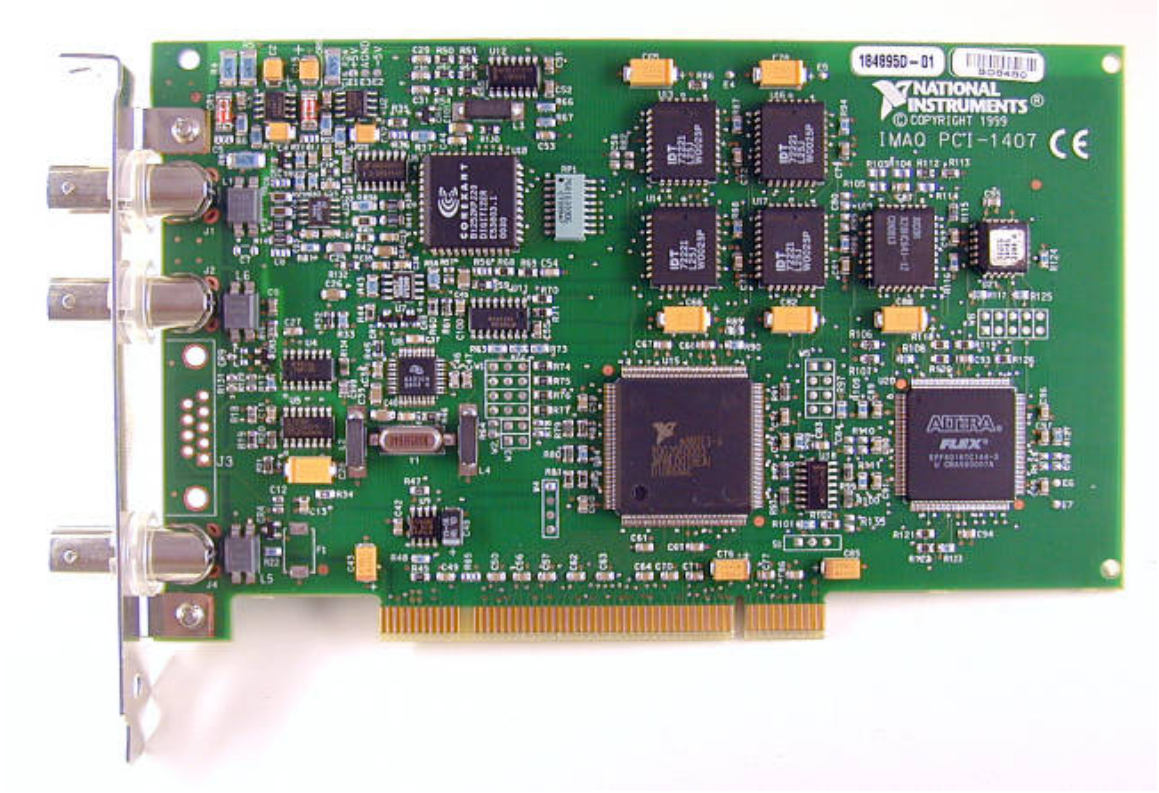


Figure 19: National Instruments IMAQ PCI-1407 / PXI-1407

6.3 I/O card

If we want to control our Servo Drive with a computer we have to use an I/O card, for instance: National Instruments PCI-6024E (**Figure 21**). It is a 200 kb/s, 12-Bit, 16 Analog Input Multifunction DAQ.
NI low-cost E Series multifunction data acquisition devices provide full functionality at a price to meet the needs of the budget-conscious user. They are ideal for applications ranging from continuous high-speed data logging to

control applications to high-voltage signal or sensor measurements when used with NI signal conditioning. Synchronize the operations of multiple devices using the RTSI bus or PXI trigger bus to easily integrate other hardware such as motion control and machine vision to create an entire measurement and control system.

Figure 20: National Instrument Logo

Details:

- Two 12-bit analog outputs; 8 digital I/O lines; two 24-bit counters
- FREE award-winning global services and support -- www.ni.com/support
- NI-DAQ Measurement Services to simplify configuration and measurements
- For new designs, NI recommends using the M Series PCI-6221
- NIST-traceable calibration certificate and more than 70 signal conditioning options
- Superior integration – LabVIEW, CVI, and Measurement Studio for Visual Basic and Visual Studio .NET

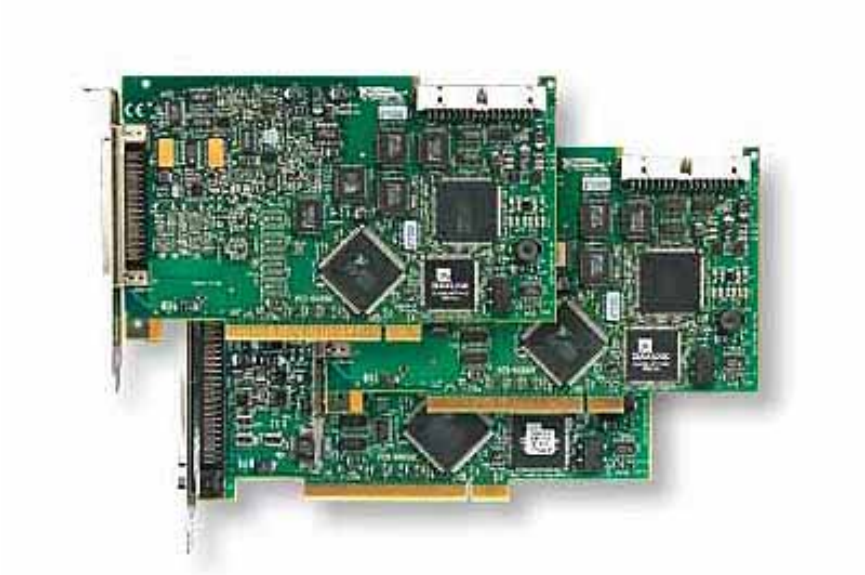
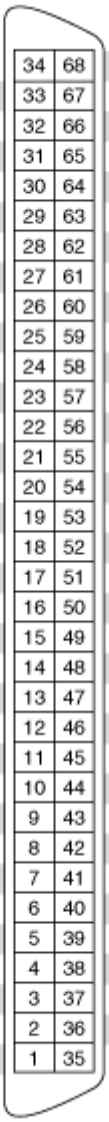


Figure 21: National Instruments PCI-6024E

National Instruments DAQ 6024E card has 2 analog output and 16 analog input channels. To control the servo drive we will use one analog output (21 [Analog Output 1] and 54 [Analog Output Ground])(**Table 5**).



AI 8	34	68	AI 0
AI 1	33	67	AI GND
AI GND	32	66	AI 9
AI 10	31	65	AI 2
AI 3	30	64	AI GND
AI GND	29	63	AI 11
AI 4	28	62	AI SENSE
AI GND	27	61	AI 12
AI 13	26	60	AI 5
AI 6	25	59	AI GND
AI GND	24	58	AI 14
AI 15	23	57	AI 7
AO 0[1]	22	56	AI GND
AO 1[1]	21	55	AO GND
AO EXT REF[1]	20	54	AO GND
P0.4	19	53	D GND
D GND	18	52	P0.0
P0.1	17	51	P0.5
P0.6	16	50	D GND
D GND	15	49	P0.2
+5 V	14	48	P0.7
D GND	13	47	P0.3
D GND	12	46	AI HOLD COMP
PFI 0/AI START TRIG	11	45	EXT STROBE
PFI 1/AI REF TRIG	10	44	D GND
D GND	9	43	PFI 2/AI CONV CLK
+5 V	8	42	PFI 3/CTR 1 SRC
D GND	7	41	PFI 4/CTR 1 GATE
PFI 5/AO SAMP CLK	6	40	CTR 1 OUT
PFI 6/AO START TRIG	5	39	D GND
D GND	4	38	PFI 7/AI SAMP CLK
PFI 9/CTR 0 GATE	3	37	PFI 8/CTR 0 SRC
CTR 0 OUT	2	36	D GND
FREQ OUT	1	35	D GND

Table 5: Pinout NI DAQ 6024E

6.4 Servo drive and AC Motor

To be able to move the cart we need a servomotor which is driven by a servo drive. We will use an OMRON AC servomotor type: R88M-WP40030H-S1-D. This is a flat style servomotor. Its maximum revolution is 3000 rpm and the maximum moment of it is 1.27 Nm.

The servo drive is an OMRON AC servo drive type: R88D-WT04H. With the help of it we can specify the mode of the control of the motor. There are two ways to control this drive: speed control and torque control.

To set the speed control we have to use the Pn300 parameter. Here we can specify the speed command reference voltage. (The interval the servo drive can get voltage.)

If we use the Pn400 parameter we can use the torque control that means we will control the torque of the motor when we put a voltage on it.

We use speed control, because in this case we can handle the friction easier. The features of the speed control can be found in the **Appendix E**.
There is one of the more important things to set properly the offset of the servo drive. If we add zero voltage through the I/O card to the servo drive, the cart must stay in its original position.

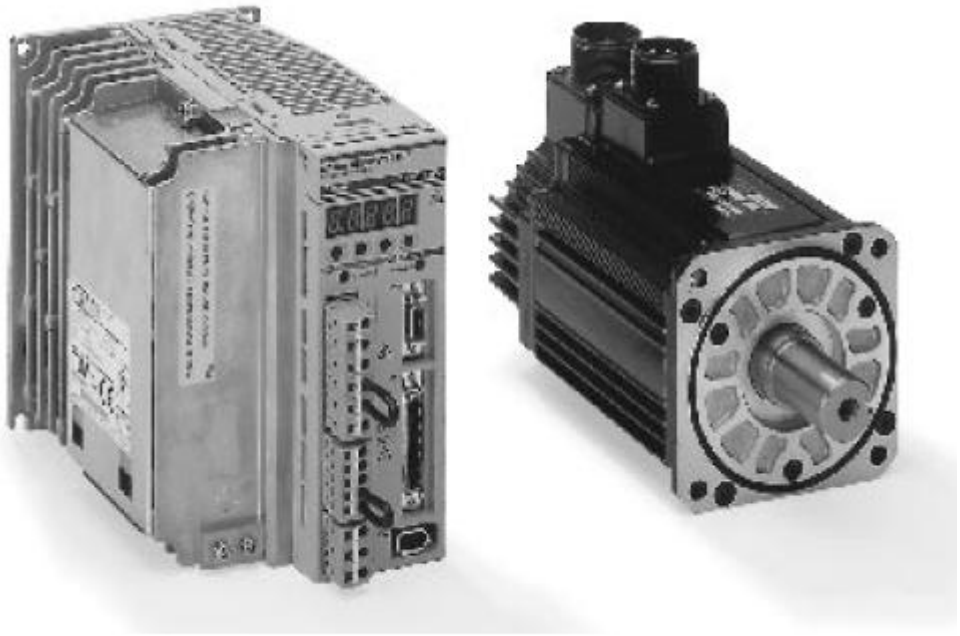


Figure 22: Servo drive and AC Motor

7 – IMAGE PROCESSING WITH IMAQ VISION

To implement the solution of the main problem first we have to realize the parts of the system. We use LabView to achieve our goal. LabView was made by National Instruments. It differs from the other "common" programming languages, because it is a codeless, graphical, block-diagram oriented developmental environment. It has a lot of useful pre-defined components called VI-s (Virtual Instrument). IMAQ is a component of LabView too. It is an abbreviation of Image Acquisition. With the help of it, it is easy to process a simple image from file or from the camera also.

7.1 Hardware

In real life the human eye can easily observe a process to give some useful information to the brain. The eye of a computer is a camera, which we connected properly.
We put the camera into a grabber card (**Figure 23**, National Instruments - IMAQ PCI-1407/PXI-1407), and we will use LabView to work its grabbed pictures.

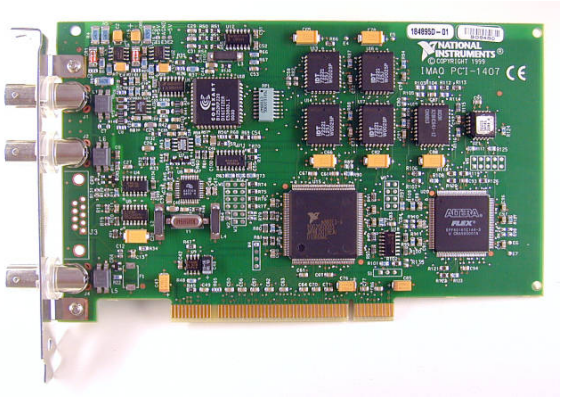


Figure 23: Grabber card

7.2 Software

We can examine which devices and software are available in LabView with National Instruments Measurement & Automation Explorer (MAX)(**Figure 24**).

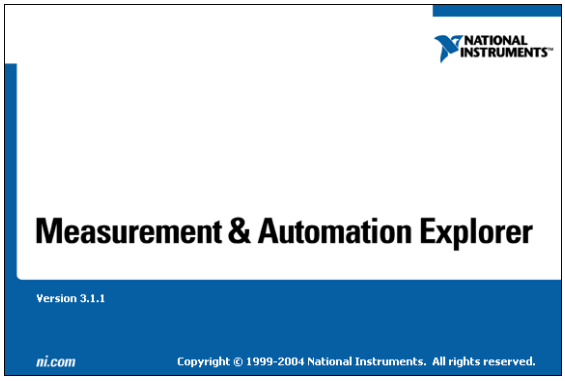


Figure 24: Measurement & Automation Explorer

As we see in (**Figure 25**), we connected our camera, and here we can also see its grabbed pictures in real time. We can read some information about picture size, etc.

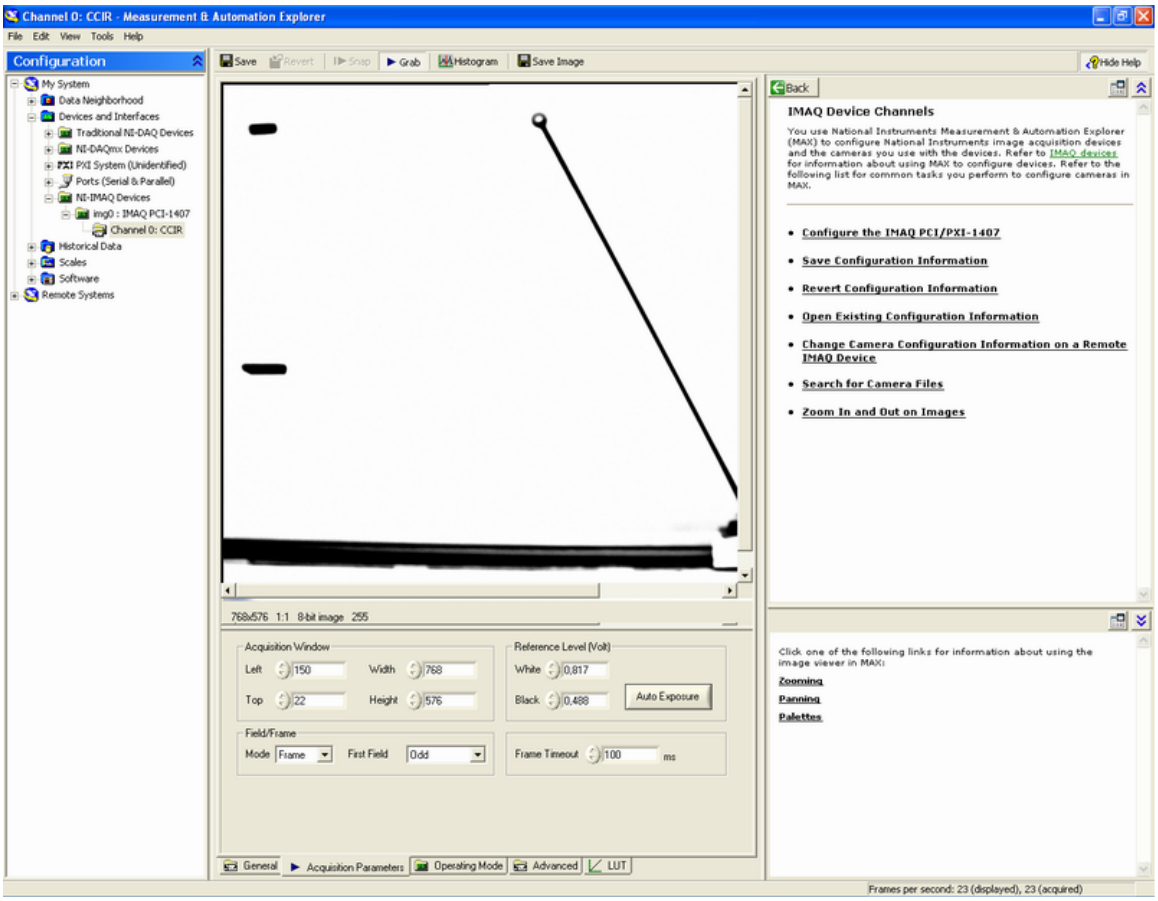


Figure 25: Camera in MAX

7.3 Pictures data in MAX

Camera Description
- Manufacturer: Standard
- Model: N/A
- Channel: 0

Acquisition Window
- Left: 150
- Top: 22
- **Width: 768**
- **Height: 576**

Reference Level (Volt)
- **White: 0.817**
- **Black: 0.488**

Field/Frame
- Mode: Frame
- First Field: Odd

Frame Timeout: 100 ms
Video Lock Mode: Standard
CSYNC Direction: Input
PCLK Source: Internal 1x
Aspect Ratio
    Pixels Per Line: 944
Lock Speed: Slow
Clamp
    Start: 90
    Stop: 130
Lookup Table: Normal

7.4 Write a VI

After that, we write a VI in LabView 7.1, where we initialize, set, visualize and close the grabbed pictures (**Figure 26**). This VI is presented as an attachment named IMAQPCI_1407.vi.

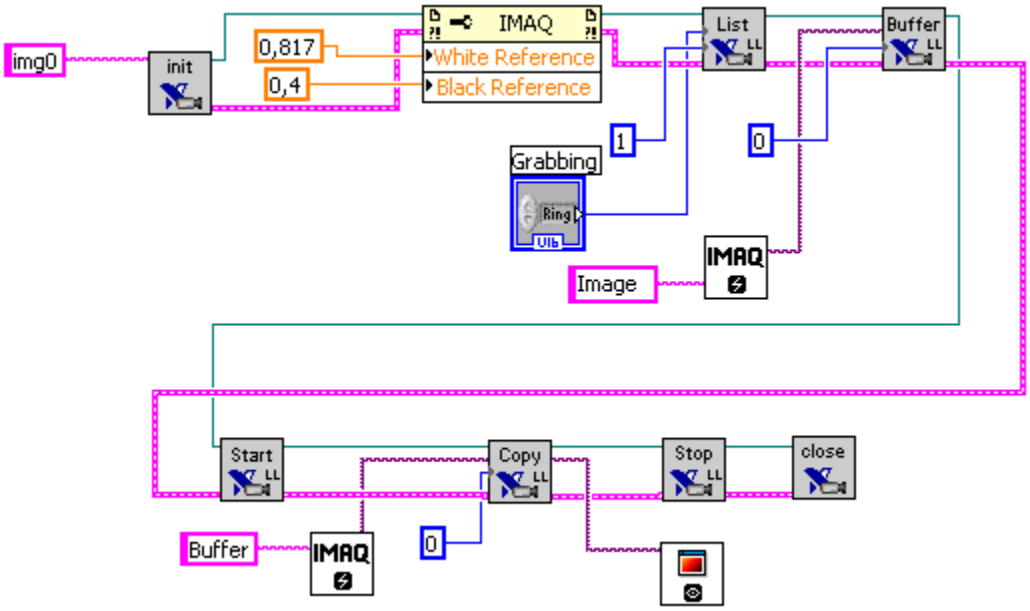
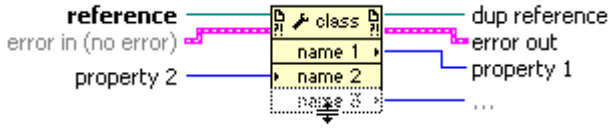


Figure 26: IMAQPCI_1407.vi

7.5 IMAQ Init element

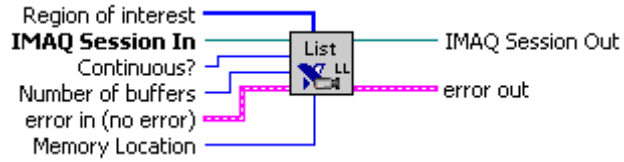
The Interface Name must match the configuration file name used in Measurement & Automation Explorer. The device name of our camera is "img0". If we set the Interface Name, we will get the IMAQ Session Out, which identifies the initialized device.
During the whole VI we guide an error signal.

7.6 Property Node element:



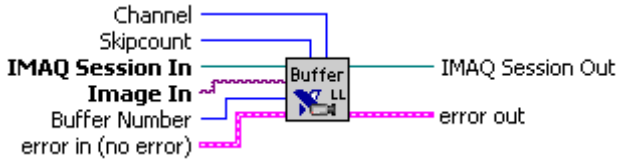
With this element we can set several properties, but we need to set the White and the Black Reference values. We took these values from National Instruments Measurement & Automation Explorer.

7.7 IMAQ Configuration List element:



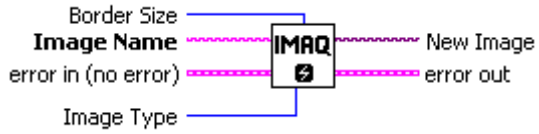
We adjusted the Continuous input to continuous with a Ring (its label is Grabbing), and the Number of buffers to one. The "continuous" specifies whether the acquisition is continuous or one-shot.

7.8 IMAQ Configure Buffer element:



We wire into this element the Buffer Number and the Image In, which is the reference to the image that receives the captured pixel data. We take this Image In from IMAQ Create.

7.9 IMAQ Create element:

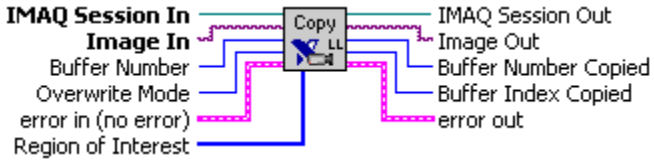


We add an Image Name (each image created must have a unique name), and get a New Image.

7.10 IMAQ Start element:

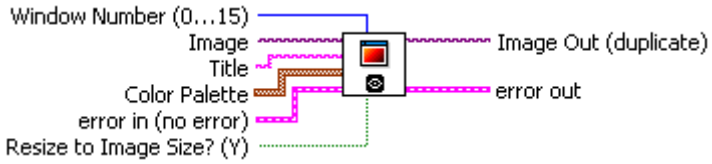
It starts an asynchronous image acquisition.

7.11 IMAQ Copy Acquired Buffer element:



It returns a copy of an acquired image. IMAQ Copy allows you to copy an image from onboard memory to system memory or from system memory to system memory. We add an image (Image In) and set the Buffer Number to zero.

7.12 IMAQ WindDraw element:



Now, we have all settings and preparation to visualize a grabbed image. With IMAQ WindDraw we display an image in an image window. The image window appears automatically when the VI is executed.

7.13 IMAQ Stop element:

It stops the currently executing acquisition on the IMAQ device.

7.14 IMAQ Close element:

Finally IMAQ Close stops the acquisition if one is in progress, releases resources associated with the acquisition, and closes the specified IMAQ session.

8 – MANIPULATION OF SERVO DRIVE IN LABVIEW

8.1 What is the next step?

We have already analysed how to grab picture from the camera, now we have to deal with the servo drive and the AC motor. The camera was our input, and the servo drive will be our output hardware.

8.2 Write a VI

We write a VI in LabView 7.1 where we will give a value for the Servo Drive, and the cart will be moving accordingly.

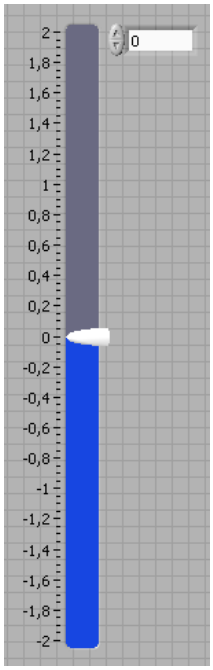


Figure 27: Servo Drive Front Panel

We use a Pointer Slide to set an optional value. The front panel is very simple, let us see the block diagram (**Figure 28**).

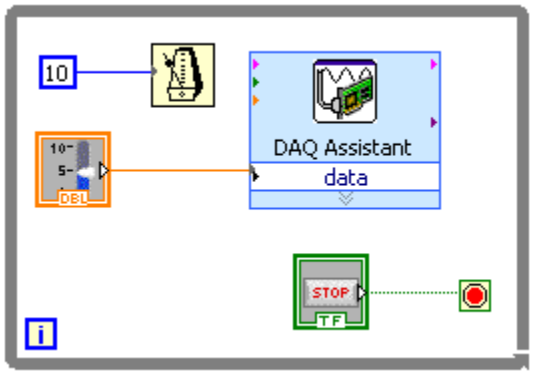


Figure 28: Servo Drive Block Diagram

We put our program in a while loop, so we can follow the events in real time, and we use a delay element.

With this we reach that the servo gets a new value only in every 10 milliseconds.

8.3 DAQ Assistant

The most important thing to set the DAQ Assistant properly (**Figure 29**). We have to set the Output Range: Max will be +10 Volts, and Min will be minus -10 Volts. The sign will determinate the direction.

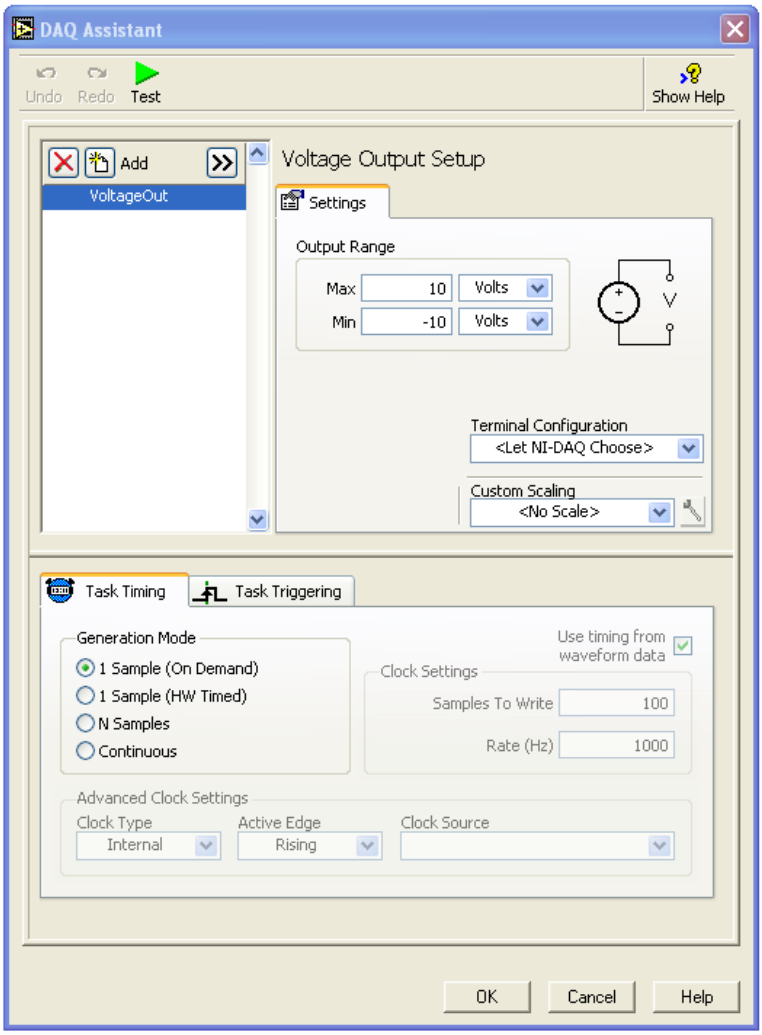


Figure 29: DAQ Assistant properties

The I/O card has several analog input and output channels. Now, we are using the Number 1 Analog OutPut physical channel (ao1). We must adjust this physical channel in the adequate window (**Figure 30**).

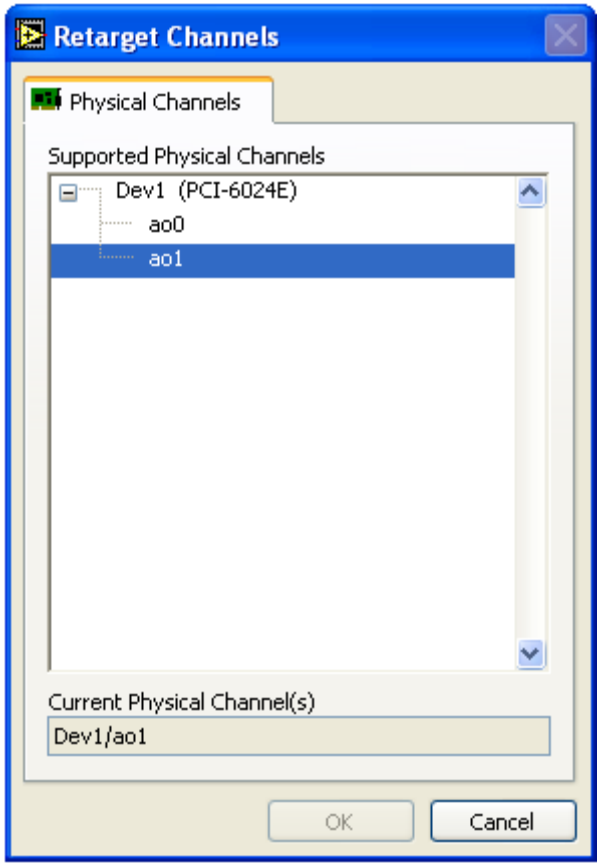


Figure 30: Physical channels

9 – CALIBRATION OF THE CAMERA

9.1 Why necessary to use calibration?

Before we begin to build our main VI, we have to put the camera in a right position. It is very important, because when we use up the grabbed picture we will suppose the following things:

- The analysed lines are clear (we will get back this event).
- If the angle of the pendulum is zero, the rod is parallel with the left and right side of the picture.
- The track of the car is perpendicular with the left and right side of the picture.

9.2 Theory

If these come true, we have to calculate a ration between pixels and meter. To specify this, we have to know how many meters is one pixel on the picture.

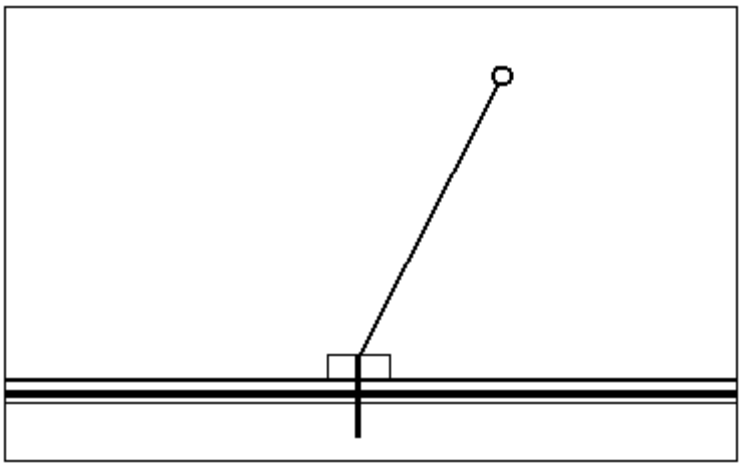


Figure 31: The model

In the future we will analyse approximately the labelled three lines (**Figure 32**). What means, that these line will be clear? We are working with a grey-scale picture (8 bit), and we will want to detect where are the black pixels in the labelled lines (we will see, why is it good for us). It is possibly only that case, if nothing disturbs the observed points in the mentioned lines.

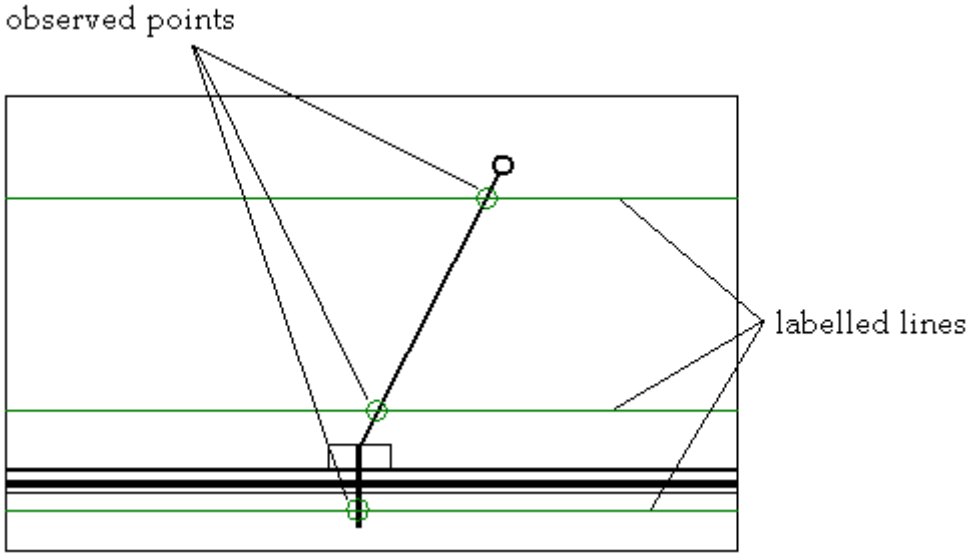


Figure 32: Labeled lines

To set the grabbed picture in an adequate position, we pasted up four black strips on the wall (**Figure 33**).

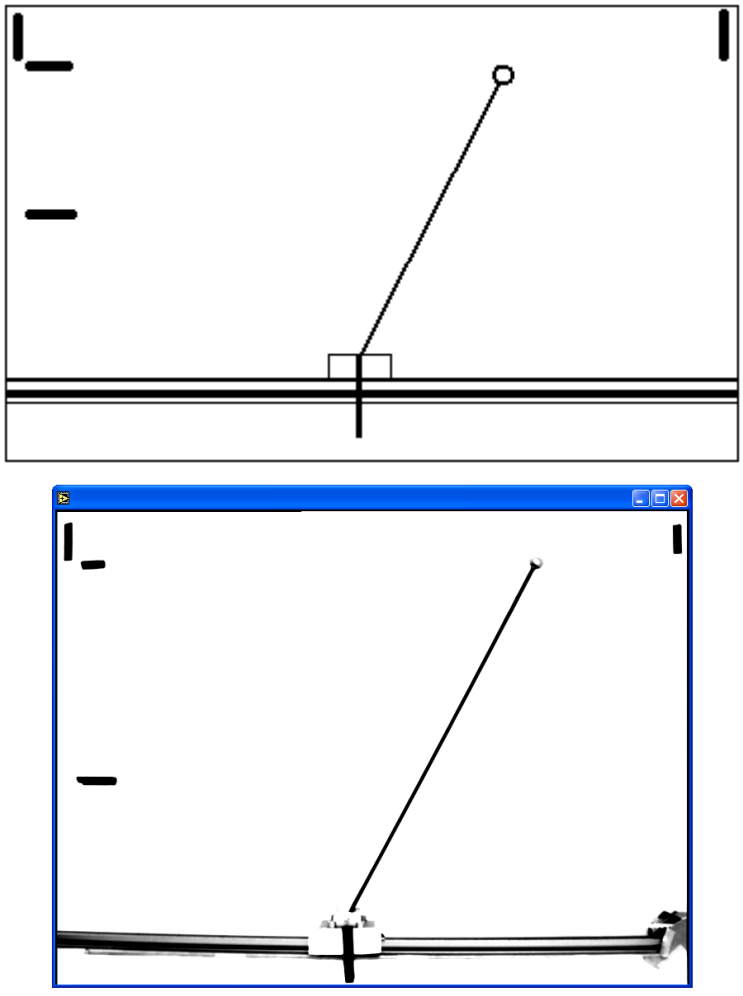


Figure 33: Four black strips

We measured the distance in meters between the two pairs of strips (**Figure 34**).

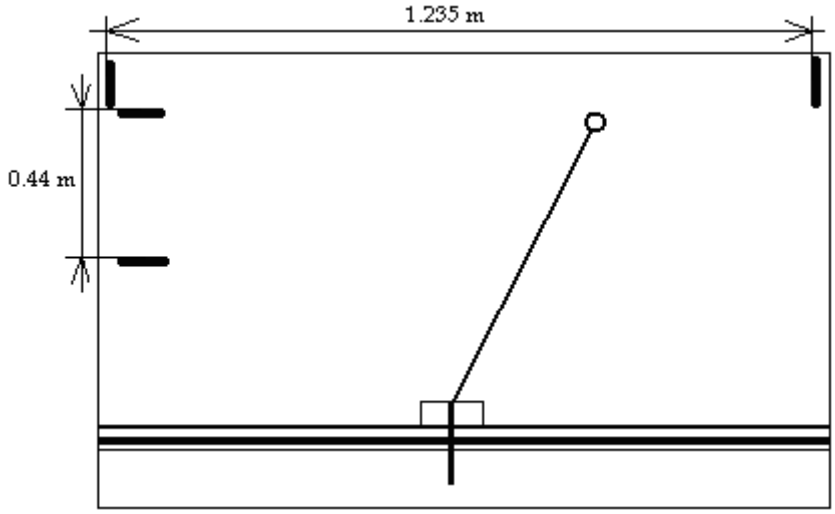


Figure 34: Distances

Now, we are working in this situation, and calculate several positions in pixels (**Figure 35**).

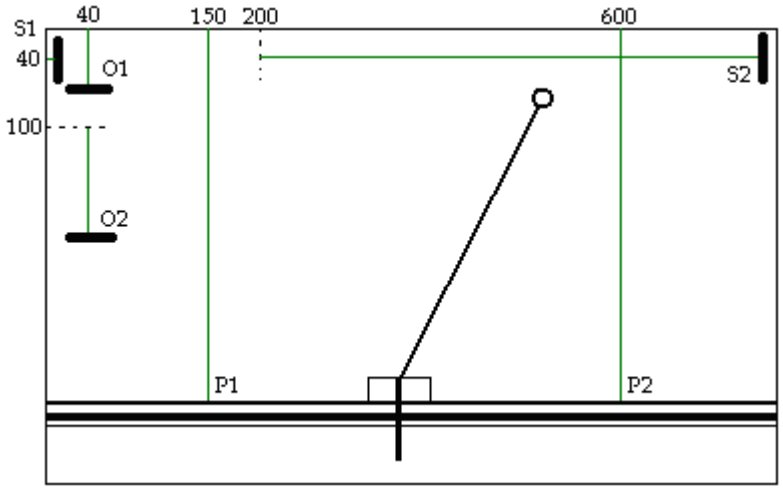


Figure 35: Calibration

The labelled lines nominate where we try to find black points (**Table 6**).

O1	We begin finding from the top side of the picture to the first black point in the 40th column.
O2	We begin finding from the 100th row to the first black point in the 40th column.
S1	We begin finding from the left side of the picture to the first black point in the 40th row.
S2	We begin finding from the 200th column to the first black point in the 40th row.
P1	We begin finding from the top side of the picture to the first black point in the 150th column.
P2	We begin finding from the top side of the picture to the first black point in the 600th column.

Table 6: Observed points

If P1 equals to P2, our track is perpendicular with the left and right side of the picture. After that, we are able to calculate the ration between pixels and meter (**C-1** and **C-2**).

$$Ratio1 = 0.44/(O2-O1) \quad \text{(C-1)}$$
$$Ratio2 = 1.235/(S2-S1) \quad \text{(C-2)}$$

Ration1 and Ration2 should be equal, but in real measurement it cannot be materialized. Therefore we use in our program the average of the above mentioned rations.

9.3 Write a VI

After all of that, let us see the Calibration.vi in the running phase (**Figure 36**).

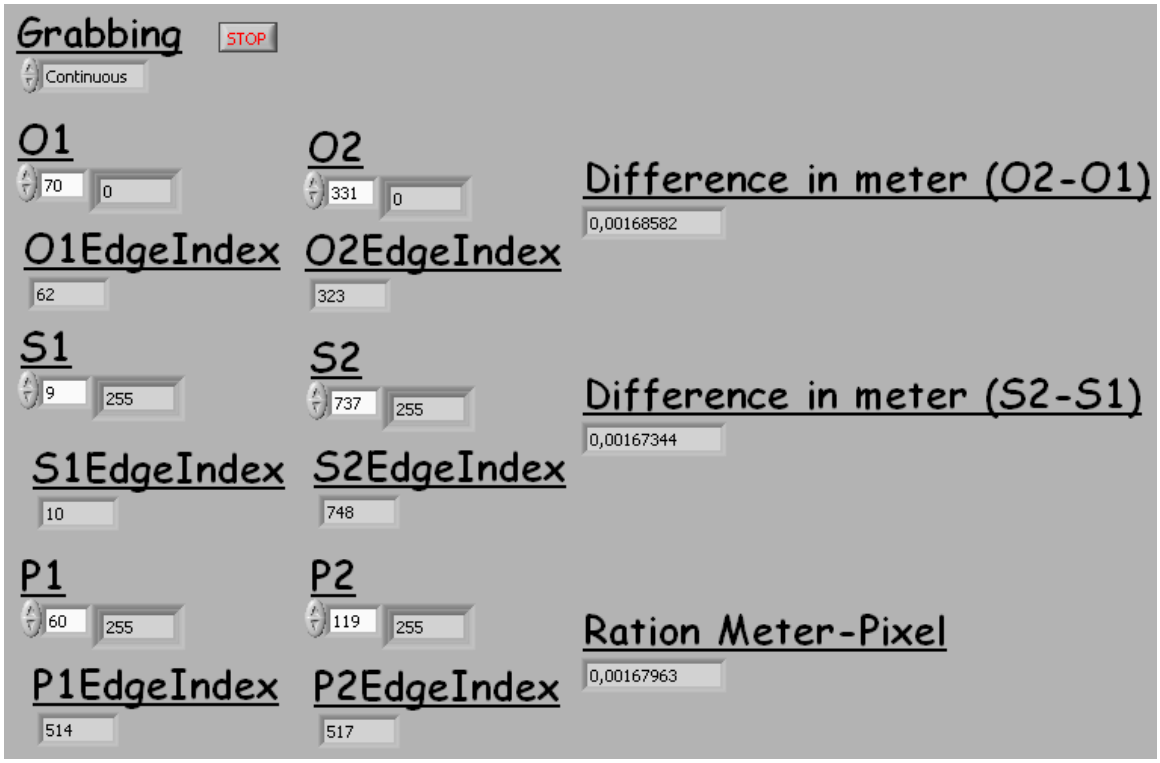


Figure 36: Calibration.VI Front Panel

If we put the grabbed picture and our result in one edited picture, we will get a graphical result (**Figure 37**).

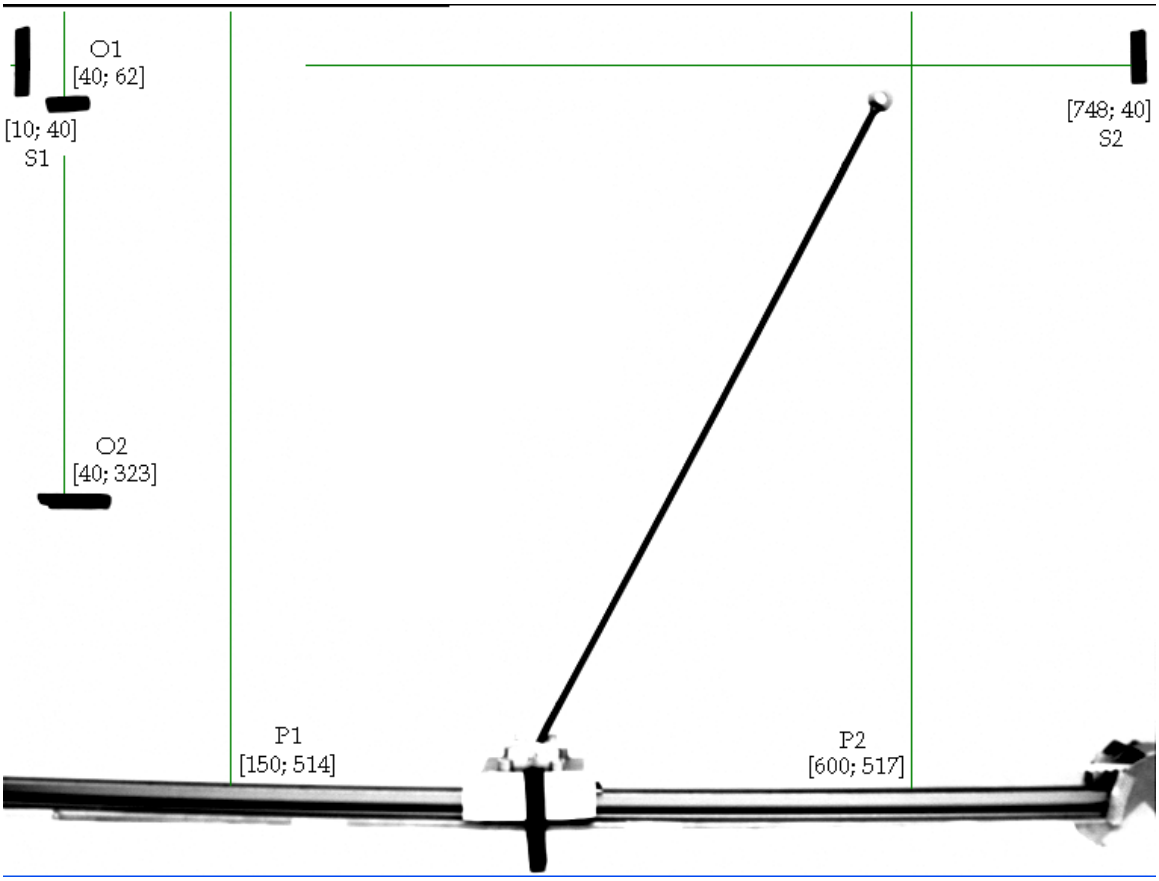


Figure 37: Graphical result

We will analyse the Calibartion.vi elements in a latter chapter, but the LabView Block Diagram is available in the **Figure 38** and **Figure 39**.

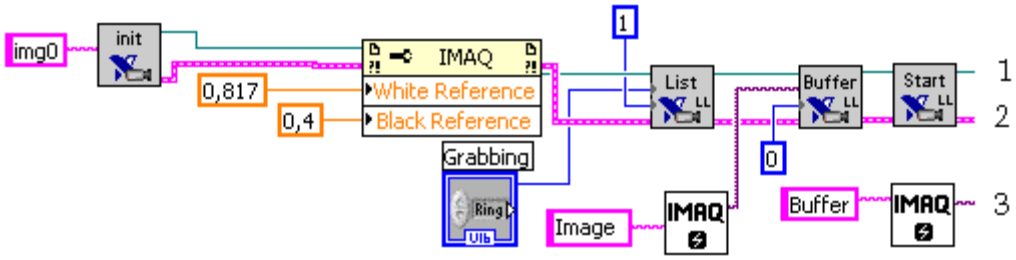


Figure 38: Calibration.VI - Part I

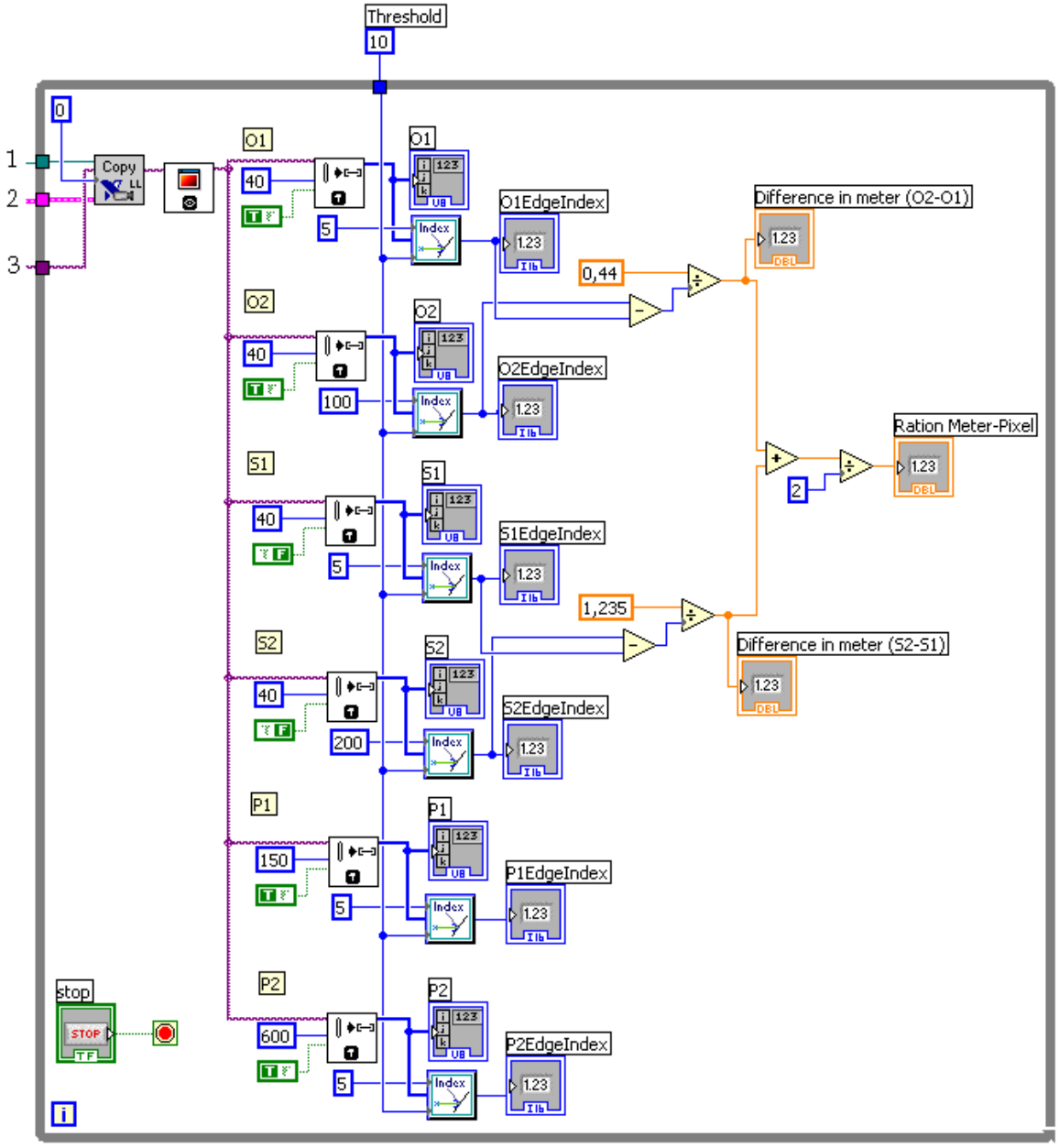


Figure 39: Calibration.VI - Part II

10 – REALIZATION OF THE INVERTED PENDULUM

10.1 Tasks

During Pole-Placement design we used the following linear function as control-law (**P-3**).

$$\bar{u} = -\begin{bmatrix} k_1 & k_2 & k_3 \end{bmatrix} * \bar{X} \qquad \text{(P-3)}$$

Then we have calculated the K values (k_1, k_2, k_3) in two different ways.

a) with LQR Design

$$k_1 = -1$$
$$k_2 = -10.6107$$
$$k_3 = -4.9048$$

b) with Butterworth polynom

$$k_1 = -0.8569$$
$$k_2 = -4.6258$$
$$k_3 = -2.9868$$

This K values include the whole physical model and the property of the servo drive.
Write the (**P-3**) equation in other format (**R-1**).

$$u = -k_1 x_1 - k_2 x_2 - k_3 x_3 \qquad \text{(R-1)}$$

The AC motor and the servo drive needs input signal, to move either direction. This signal is positive or negative value which represents positive or negative voltage. The result of the control law will be the adequate voltage.

After that, our task in LabView 7.1 is quite simple.

- Firstly we grab pictures from the camera.
- Then we measure the system state variables ($\bar{X}$).
- We know the K values, so we realize the control law.
- We add the result of the control law for the servo drive.

We work with our 3rd order system [(**S-17**)-(**S-19**)].

$$X_1 = X \qquad \text{(S-17)}$$
$$X_2 = \Theta \qquad \text{(S-18)}$$
$$X_3 = \int g\Theta dt \qquad \text{(S-19)}$$

So we have to measure the displacement of the cart and the angle of the rod, then we have to calculate the 3rd state variables.

Let us see our tasks graphically (**Figure 40**).

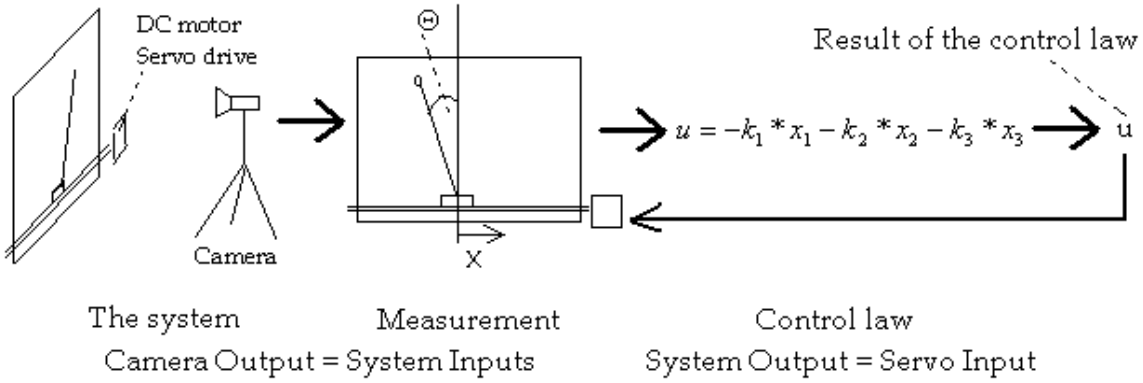


Figure 40: Tasks

10.2 Measure the states

We have to measure our state variables first. Let us start the angle of the pendulum, after that the position of the cart.

10.2.1 Angle of the pendulum

10.2.1.1 Mathematical background

The most important question is that, how we measure an angle in a picture? The method will be fast enough, because it runs in every cycle (and there are lots of cycles per second)!

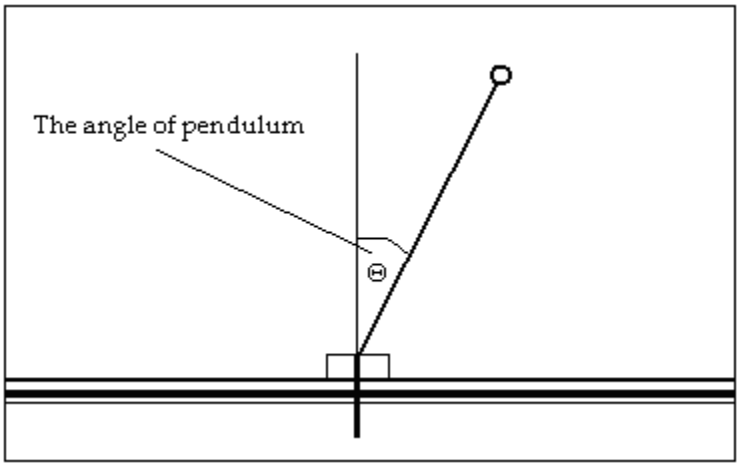


Figure 41: The angle of pendulum

Because of the requirement (agility), we cannot analyze the whole picture (**Figure 41**). One of the best methods, in that case we analyze only two rows, and find two special points. After that, we use mathematical rules to get the adequate angle.

If we write a SubVI which can calculate a special column index of any rows we are closer to the result. What does "special column index" mean? Our grabbed picture is a gray-scale 8-bits picture, and most of the pixels are white, or almost white (In 8-bits white is 255 (a computer stores any data as a number), or approximately between 200 and 255). If we find a black or an almost black pixel (In 8-bits black is 0, or approximately between 0 and 50), we have found a special one! Our SubVI will find these blacks, or almost blacks pixels in a row.

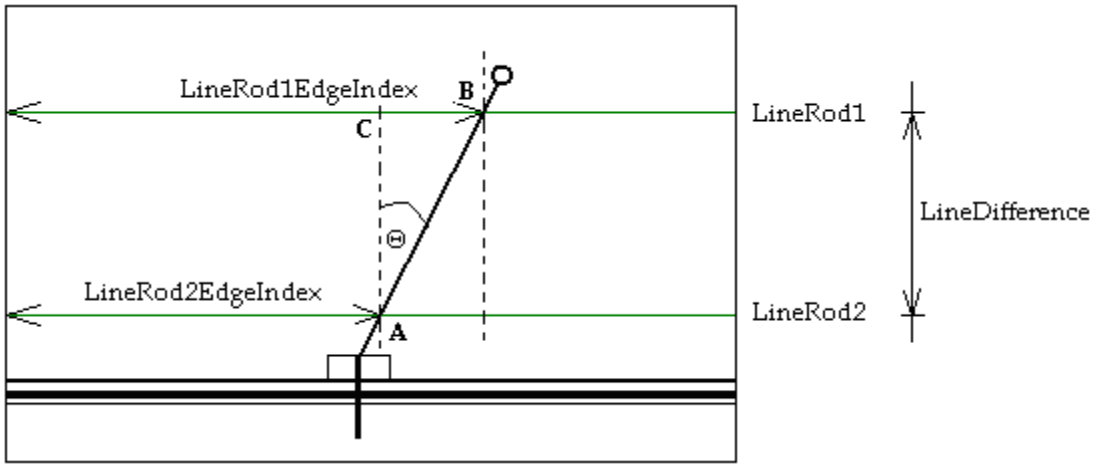


Figure 42: Mathematics background

After that, we have two "EdgeIndex": LineRod1EdgeIndex and LineRod2EdgeIndex, and we know the difference between LineRod1 and LindRod2 (**Figure 42**)! From these values we will know enough information of the ABC triangle (**Figure 43**) to calculate any data into this two dimension figure (but we want to know the CAB angle only).

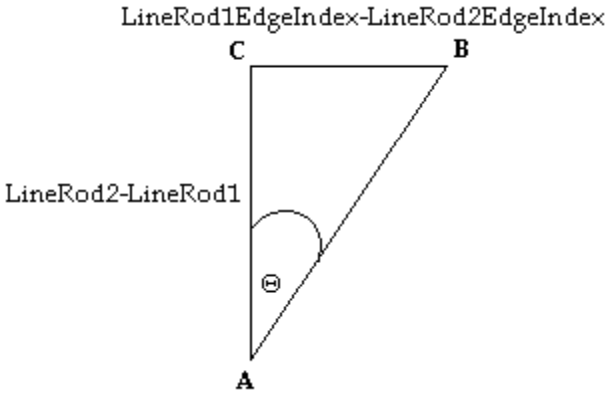


Figure 43: ABC Triangle

The correct mathematical formula is (**M-1**).

$$\Theta = ArcTan\left(\frac{LineRod1EdgeIndex - LineRod2EdgeIndex}{LineRod2 - LineRod1}\right) \qquad \text{(M-1)}$$

10.2.1.2 Chosen lines

Our chosen lines are the following:

LineRod1 = 150
LineRod2 = 450

We have to choose our lines as far as we can, because this distance determinates what is the minimal angle that we can measure. For the above mentioned data it is:

Minimal angle that we can measure: ~0.19°

To calculate the (**M-1**) equation, we have to write a SubVI, called EdgeIndex.vi, to get LineRod1EdgeIndex and LineRod2EdgeIndex.

10.2.1.3 EdgeIndex element

We add Line as a vector to this SubVI, because we have an other element which can read any rows or columns of the picture into a vector (one dimension array). We add a StartPos, as well, because we use this VI in the Calibartion.vi too, and there we not only find edge from the side of the picture, but also from any other rows or columns (this process, where we find black pixels, is named edge detection).

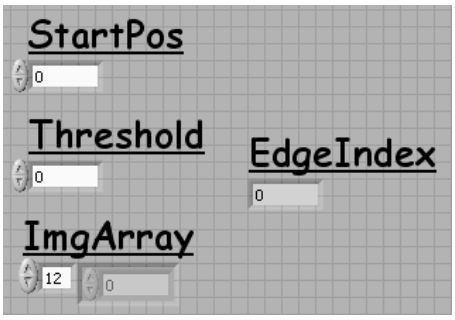


Figure 44: EdgeIndex.vi Front Panel

The lighting can be very changeable, so we have not got pure black (0) and pure white (255) value most of the time. We use a Threshold value to solve this problem. If we find a value between (0) and (0+Treshold), we analyze that as black pixel (so we found an edge).
We have only one output, the special index of our input vector. It means the edge distance from the border of the picture in pixels.

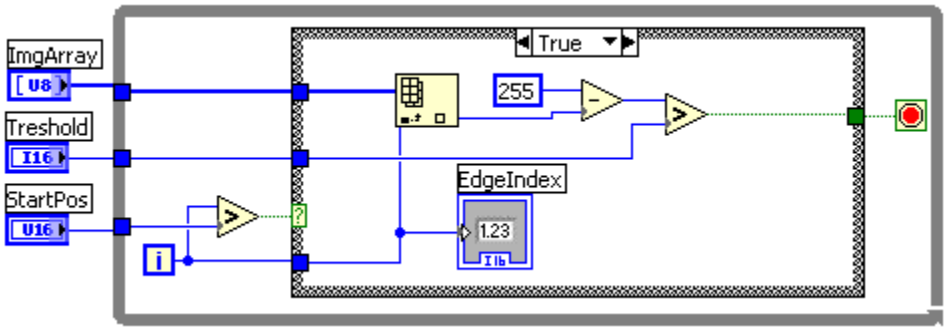


Figure 45: EdgeIndex.vi Block Diagram

10.2.1.4 Build this SubVI into our main VI

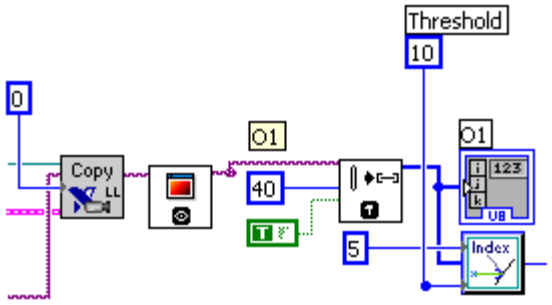
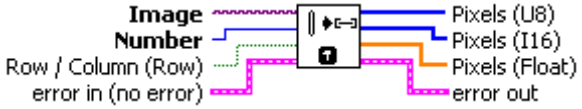


Figure 46: An example

The visible example (**Figure 46**) is looking for an edge in the 40th column. It starts finding from the 5th pixel and use 10 as Threshold.

10.2.1.5 IMAQ GetRowCol element



When we add an image as input into to this element, it gives us a vector which can be used in our EdgeIndex.vi.
Row / Column (Row) defines Number as a row number when FALSE and as a column number when TRUE.

10.2.2 Position of the cart

10.2.2.1 The same method

Now we are able to calculate the position of the cart in the same way. We choose an adequate line (CasPosLine), and our EdgeIndex.vi can determine CarPosEdgeIndex (**Figure 47**).

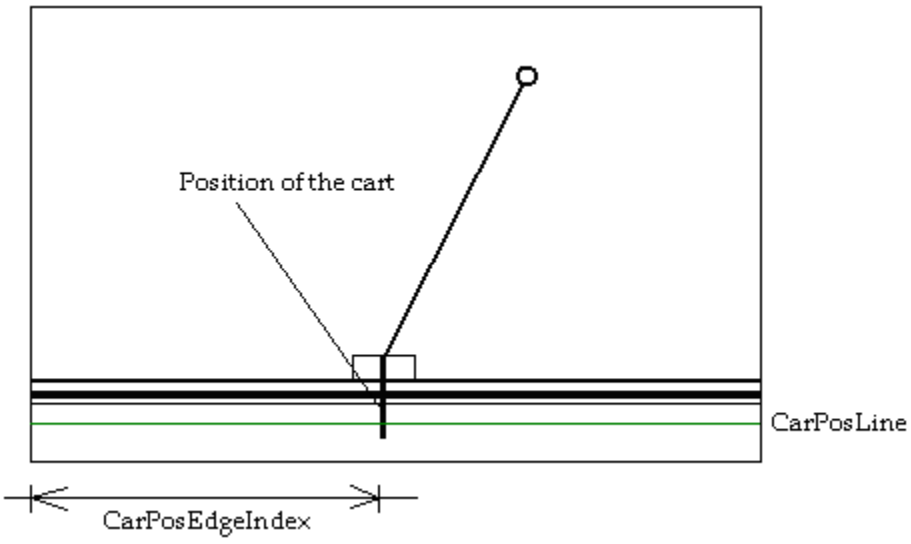


Figure 47: Position of the cart

10.2.2.2 Chosen lines

CarPosLine = 540

10.2.2.3 Conversion from pixels to meters

Our first state variable is the position of the cart, but not in pixels, because it depends on the resolution of the camera! We have to convert the CarPosEdgeIndex values from pixels to meters. We will apply our Calibration.vi, to determine the ration number (**M-2**).

CarPosMeter = CarPosLine * AVGMeter (M-2)

In Calibration.vi we called AVGMeter as Ration Meter-Pixel. For instance we measured, that one pixel in the picture is 0.00169764 meters (but it is different in every occasion).

10.2.2.4 We use a reference

Our purpose is that our cart should stay in the middle of the track. Hence we use a reference position, and we count it the zero position of the cart. When the cart moves, the position is changing in negative or positive direction (**Figure 48**).

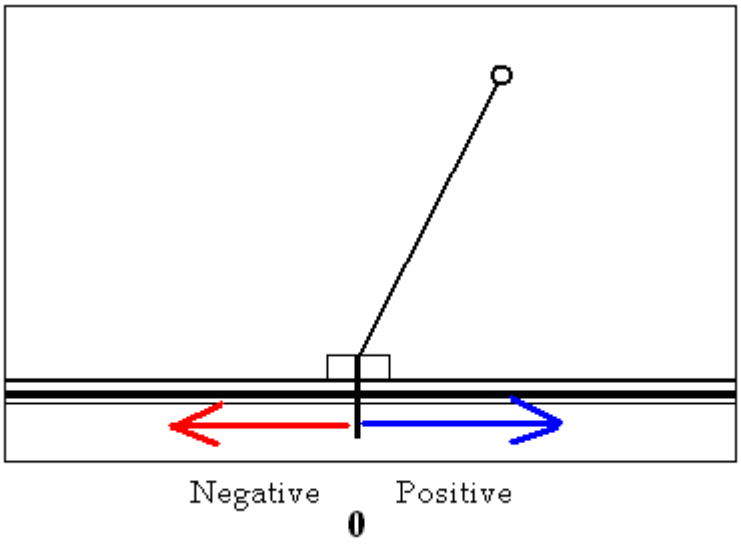


Figure 48: Reference and directions

Our reference position is not a concrete value, but we use the first grabbed picture in the main VI, and the current CarPosEdgeIndex will be our reference.

10.2.3 A part of our main VI

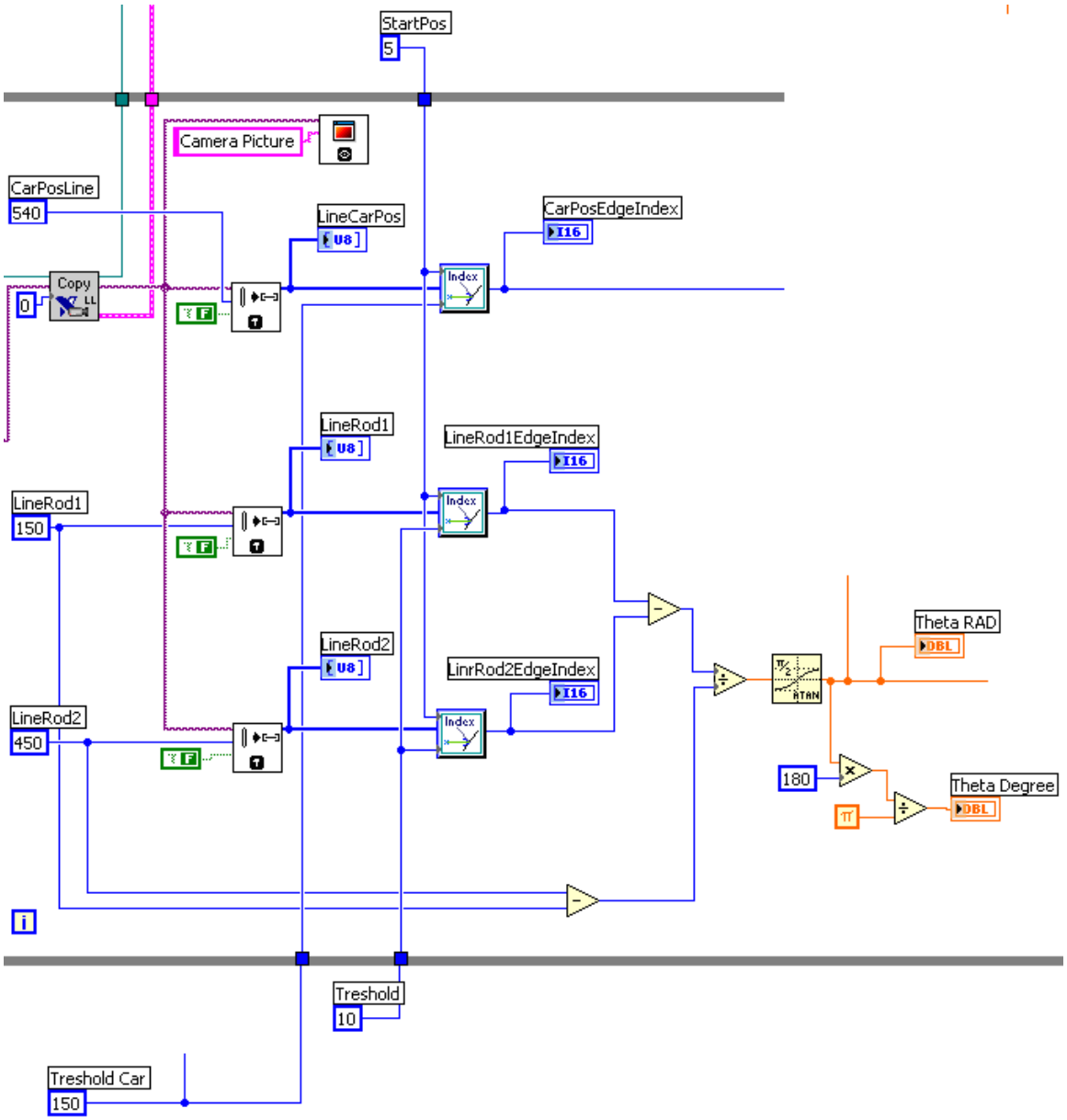


Figure 49: Measuring the states

10.3 The whole VI

We realized all parts of the system but now we have to connect them to get the full system in practice. The figure of the full program can be found in the **Appendix G**. We append some comment on it. The functions we made has to be repeated all the time to give the correct values move the cart. We solve this with a while loop which runs until the user push the Stop button placed on the screen. In every cycle of the loop we need the three state variables. (We also need the K values but they are always the same if the system works properly.) The only procedure we can place out of the loop is the calculation of the start position of the cart. It is not too complicated because we only have to examine the first frame

the camera gives us and find the cart on it. Then we use the edge detection algorithm and store the pixel index it gives. This will be the start position of the cart. We will need this value in every cycle but since it is always the same we will only calculate it at once. We can name this index as the reference point of the position. (In the case of the angle, the reference value is the 0 rad.)
On the figure of the program we can see that we use constant values for the edge detection. It means that we are always looking for the rod or the cart in the same line of the grabbed picture. That's why we need to calibrate the camera properly. If something change in the environment and we can not calibrate the camera we can also change the lines.
When the program has calculated all the states we need to use the control law we mentioned above. We multiply these values by the negative corresponding K values and then we have to sum them. This value will be the output of the program and it will be added to the I/O card as input. Actually this is a voltage value which will pass through the I/O card and "drive" the servo motor through the servo drive. It is important to save the card from the invalid voltage values. The card and the servo drive was configured that the voltage value it can get has to be in the interval of [-10;+10]. In the right side of the figure we can see this protection function. We always calculate the current output value and if it is out of the interval we send 0 instead. We note that if the functions work properly it can not happen to get these critical values.
The other important thing is to choose the update time of the control loop. It can affect the system because this interval can be different depend on the other processes running on the computer. To solve this problem we should choose a proper interval which is big enough to finish all the calculation and small enough to be able to control the cart. It is possible to use a time delay in the while loop. If we don't use this timer, the loop will repeat itself when all the procedures have finished inside the loop. Our camera can get pictures with 25 fps (frame per second). It means that it gives pictures in 40 ms intervals, so we decide to choose this value for the timer.
The other point we have to note is the use of the integrator. This component calculates the integral of the angle. We made a lot of experiment on the pendulum in the final phase and we found the factor which effects most of the trouble is the integrator component. The problem was that it increased too fast and was not be able to decrease its value in a proper time. We found that we have to divide the input of the integral by an adequate value to get the result we want. During the experimentation we found this value to be 25.
We can note one more thing on the figure. This is the AVGMeter constant. This constant is calculated by the VI that makes the calibration of the camera. This gives the ratio of pixels and meters.

10.4 Rebuild the VI with Formula Node

The main advantage of LabView is that we can use a lot of predefined VIs. In a simple program we can easily follow the dataflow if we use the „Highlight

Execution" option on the block diagram. If we have complicated task to do, which needs a lot of components we had better to use some faster methods. Fortunately LabView gives us this opportunity by „Formula Node". This component can be found on the Functions Palette/All Functions/Structures. The main feature of the formula node is that we can use C code to realize an algorithm. The only thing we have to do is to wire the inputs and outputs we want to use. It is practical to rewrite the methods we use many times in a loop. If we take a look to the whole system we can find that the edge detection function runs in every cycle of the main loop. So first we try to realize this with formula node. The original SubVI can be found in the 10.2.1 chapter.
In this algorithm the main task is to look for the first pixel in a line which has a greater difference from the white colour pixel than the predefined threshold value. The searching method begins from the element we define. Our three inputs are the line of the picture, the threshold value and the start position of the search. The output is only the index of the element we are looking for.

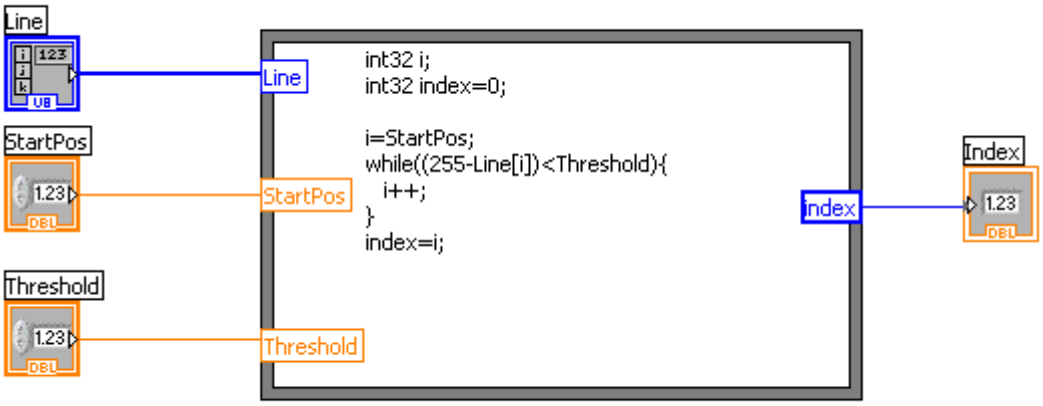


Figure 50: The edge detector function realized by formula node

We can see that this form is more simply than the other was (**Figure 50**). We only need a while loop which compares the current pixel with the difference of the white value and the threshold value. If it is greater it will stop and we get the right index we are looking for.
The next step: calculation of the state variables. We saw in the 10.2 chapter how we can calculate the state variables in every loop. Now we will show the calculation of the states by formula node (**Figure 51**).

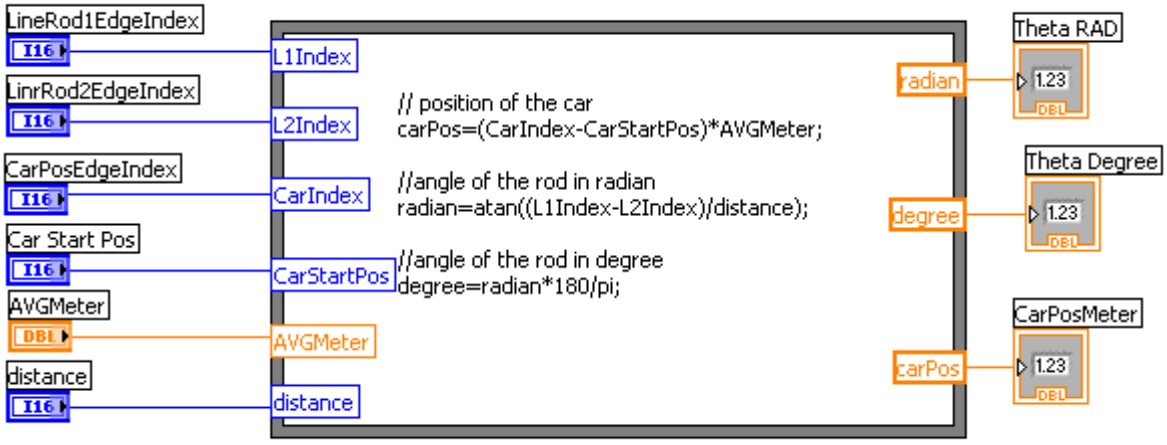


Figure 51: Calculation of the state variables with formula node

We can see that we need six inputs to get the correct output values. This function will be repeated in every cycle in the main loop. As we mentioned, calculating the angle of the rod needs two different lines. If we take a look at the code we notice an unknown element.

We can see the whole VI rewritten with formula nodes in the **Appendix H**. The matter of the program is the same only the realization is different. We substitute the SubVIs written in the other application with the above mentioned codes using formula nodes.

11 – RESULTS AND CONCLUSION

During the three month we got familiar with the different concepts of modern control theory, we came to know the methods can be used in this subject. With the basis of this we tried to realize a physical system in practice. Our task was to make an adequate model first. The next step was to choose the state variables of the system. To a first approximation we used four variables. To make the model simpler we took a simplification to reduce the 4th order system to a 3rd order system. It is not always the best way since our system will be more simply but it is more difficult to make so "sensitive" application. In the end we chose the simply way instead of the "perfect". After we got the state variables we used the vector-matrix form to write the system with differential equations. We used Matlab and Simulink to model the open and the closed loop system later. To make the closed loop system we used the control law and pole placement design. In this method we try to change the poles of the original system to be stable. To achieve it we had to decide the wanted "characteristic" of the system. This gave us how the system should behave. We approach this from two sides. We used the Butterworth-filter and an optimal approximation with LQR (Linear Quadratic Regulator). These methods help us to find the correct feedback gains called "K values" according to the control law. After these calculations we examined the response of the "new" closed loop system. Since we found that the feedback gains calculated by the Butterworth-filter were better, we decided to use these results.

In the final phase we "only" had to build the application use our calculations. We used LabView to make the final application. We wrote several VI-s to get familiar with all parts of the system for example reading frames from the camera, moving the cart, use the I/O card. From these parts we were able to make the whole system.

During the realization we found that the most important thing to calibrate the camera properly because every state variables are measured by it. If it gives wrong value the system will not work however the model and the application are adequate. The first thing which caused problem was the measurement of the angle. It has to be the most precise value. The minimum angle we could measure first was approximately 0.5°. We found it was too large to get an adequate result in the application. So we changed the parameters of the measurement by increasing the distance between the two lines we were looking for pixels in.

In the end we achieved our goal and our program was able to keep the pendulum in balance.

However we are rather adept in structured and object oriented programming languages we found that it is an easy way to realize problems in LabView. During this scholarship we came to know control theory from a different view and we saw how usable it is nowadays.

12 – APPLIED LITERATURE

Charles L. Phillips, Royce D. Harbor: Feedback Control Systems
Prentice Hall, USA, 2000

Katsuhiko Ogata: Modern Control Engineering
Prentice Hall, USA, 1997

Richard C. Dorf, Robert H. Bishop: Modern Control Systems
Prentice Hall, USA, 2005

Thomas Klinger: Image Processing with LabView and IMAQ Vision
Prentice Hall, USA, 2003

National Instruments: LabView7 Express – User Manual
National Instruments: IMAQ PCI/PXI – 1407 – User Manual
National Instruments: LabView7 Express – Measurement Manual

Ingolf-Martin Salen, Thor-Arne Voldsund: Regulering av Invertert Pendel Rapport
Ålesund University College, Norway, 2004

Vidar Øverås, Øystein Undertun: Regulering av Invertert Pendel Rapport
Ålesund University College, Norway, 2003

URL-s in LITERATURE REVIEW:

http://www.engin.umich.edu/group/ctm/examples/pend/invpen.html
http://www.aptronix.com/fuzzynet/applnote/twostage.htm
http://wwwa.mpi-magdeburg.mpg.de/research/pendel/index_e.html
http://www.obrador.com/EE471Design/EE471Design.htm
http://4north.no-ip.com:8080/pics/pendulum/
http://www.mathworks.com/matlabcentral/fileexchange/loadFile.do?objectId=3790&objectType=file
http://home.earthlink.net/~botronics/index/balibot.html

ATTACHMENTS AND APPENDICES

A	–	Matlab M-files
B	–	Matlab SimuLink Model files
C	–	LabView programs
D	–	I/O card
E	–	Servo drive
F	–	JAI AV50/60 CCD Camera
G	–	LabView program: David2InvertedPendulum.vi
H	–	LabView program: David2InvertedPendulumFormulaNode.vi

A – Matlab M-files

These files are available on the CD in the Attachement/Matlab directory.

- DIP01_StateSpaceAnalysisM.m
- DIP02_StateSpaceAnalysisM.m
- DIP03_PolePlacementM1.m
- DIP04_PolePlacementM2.m
- DIP05_PolePlacementM3.m
- DIP06_PolePlacementM4.m

B – Matlab SimuLink Model files

These files are available on the CD in the Attachement/SimuLink directory.

- DIP01_StateSpaceAnalysisMDL.mdl
- DIP02_StateSpaceAnalysisMDL.mdl
- DIP04_PolePlacementMDL.mdl

C – LabView programs

These files are available on the CD in the Attachement/LabView directory.

- IMAQPCI_1407.vi
- ServoDrive.vi
- EdgeIndex.vi
- Calibration.vi
- David2InvertedPendulum.vi
- David2InvertedPendulumFormulaNode.vi

E Series – Low-Cost

- 16 analog inputs at up to 200 kS/s, 12 or 16-bit resolution
- Up to 2 analog outputs at 10 kS/s, 12 or 16-bit resolution
- 8 digital I/O lines (TTL/CMOS); two 24-bit counter/timers
- Digital triggering
- 4 analog input signal ranges
- NI-DAQ driver simplifies configuration and measurements

Families

- NI 6036E
- NI 6034E
- NI 6025E
- NI 6024E
- NI 6023E

Operating Systems

- Windows 2000/NT/XP
- Real-time performance with LabVIEW
- Others such as Linux and Mac OS X

Recommended Software

- LabVIEW
- LabWindows/CVI
- Measurement Studio
- VI Logger

Other Compatible Software

- Visual Basic, C/C++, and C#

Driver Software (included)

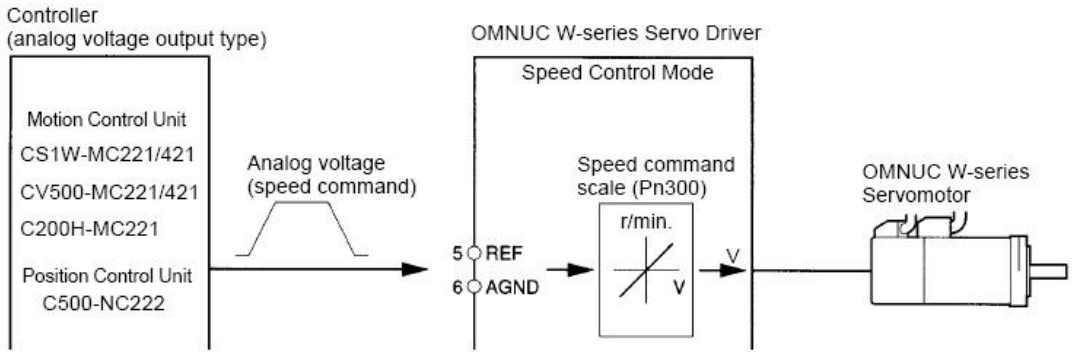
- NI-DAQ 7

Calibration Certificate Included

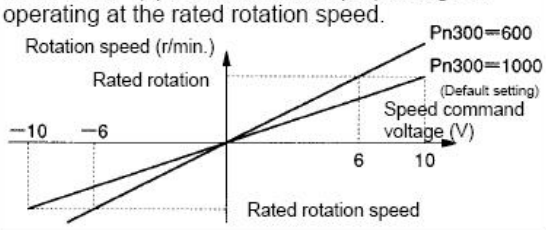
4-5-2 Speed Control (Speed)

■ Function

- Performs Servomotor speed control using analog voltage input from the speed command (REF: CN1-5, 6). You can also perform position control by combining speed control with the controller mounted to the position control function.
- You can change the relationship between the speed command and the rotation speed by setting the speed command scale (Pn300).



■ Parameters Requiring Settings

Parameter No.	Parameter name	Explanation	Reference
Pn000.1	Function selection basic switch 1	Set the control mode for speed control (Settings: 0, 4, 7, 9, A)	*4-4-3 Important Parameters*
Pn300	Speed command scale	Set the REF (speed command input) voltage for operating at the rated rotation speed. 	*4-4-4 Parameter Details*

CV-A50/A60

Compact Industrial Monochrome CCD Camera

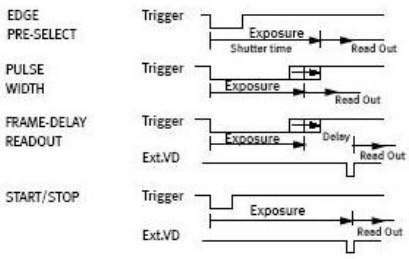
- Compact size 1/2" and 1/3" monochrome CCD cameras
- CCIR: 752 (h) x 582 (v) pixels. EIA: 768 (h) x 494 (v) pixels
- Improved sensitivity, dynamic range and smear performance
- High S/N ratio up to 60 dB
- Extended range of functions and options compared with CV-M50
- Shutter speeds from Off to 1/10,000 sec.
- Programmable exposure
- Interlaced or non-interlaced scanning. Field or frame accumulation
- Internal, external HD/VD or random trigger synchronization
- Edge pre-select, pulse width control, start/stop trigger modes
- Long time exposure with external VD pulse interval
- Frame-delay read out in connection with pulse width controlled shutter
- Exposure enable EEN, write enable WEN and pixel clock output
- Short ASCII commands for fast mode setup via serial port
- Setup by Windows NT/2000/XP software via RS 232C

The leading manufacturer of high performance camera solutions

Specifications for CV-A50/A60

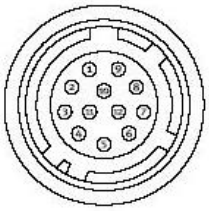
Specifications		CCIR	EIA
Scanning system		625 lines 25 frames/sec.	525 lines 30 frames/sec.
CCD sensor CV-A50		Monochrome 1/2" IT CCD	
CCD sensor CV-A60		Monochrome 1/3" IT CCD	
Sensing area	CV-A50	6.4 (h) x 4.8 (v) mm	
	CV-A60	4.9 (h) x 3.7 (v) mm	
Effective pixels		752 (h) x 582 (v)	768 (h) x 494 (v)
Pixels in video output		737 (h) x 575 (v)	758 (h) x 486 (v)
Cell size	CV-A50	8.6 (h) x 8.3 (v) µm	8.4 (h) x 9.8 (v) µm
	CV-A60	6.5 (h) x 6.25 (v) µm	6.35 (h) x 7.4 (v) µm
Resolution horizontal		570 TV lines	
Sensitivity on sensor			
	CV-A50	0.03 Lux, Max gain, 50% video	
	CV-A60	0.05 Lux, Max gain, 50% video	
S/N ratio	CV-A50	59 dB	60 dB
	CV-A60	58 dB	59 dB
Video output		Composite VS signal, 1.0 Vpp, 75 Ω	
Gamma		0.45 or 1.0	
Gain		Manual – Automatic	
Gain range		0 to +15 dB	
Accumulation		Field – Frame	
Synchronization		Int. X-tal. Ext HD/VD or random trigger	
Scanning		2:1 interlaced, non-interlaced	
HD sync. input/output		4 V, 75 Ω	
Trigger input		4 V, 75 Ω	
WEN output (write enable)		4 V, 75 Ω	
EEN out. (exposure enable)		4 V, 75 Ω	
Pixel clock output		4 V, 75 Ω	
Shutter		(Off), 1/100, 1/250, 1/500, 1/1000, 1/2000, 1/4000, 1/10,000 sec.	
Pulse width control		1.5 H to 1000 H	
Start/stop		1/50 to 1/10,000 sec. 1/60 to 1/10,000 sec.	
Long time exposure		1 field to ∞	
Programmable exposure		2.5 H to 252.5 H	
Frame delay readout		Time from PWC trigger input to ext. VD input	
Functions controlled by RS 232C		Scanning format, Trigger mode, Shutter speed, Trigger/WEN polarity, Accumulation, Shutter mode, Programmable exposure, AGC level, White clip, Setup, Manual gain, AGC/manual gain, Internal/potentiometer gain set, Gamma, Pixel clock, Commands and internal adjustments.	
Functions controlled by DIP switches		VD input/output, HD input/output, internal HD and VD 75 Ω termination on/off	
Operating temperature		-5°C to +45°C	
Humidity		20 - 80% non-condensing	
Storage temp./humidity		-25°C to 60°C./20% - 90 %	
Power		12V DC ± 10%. 1.5 W	
Lens mount		C-mount	
Dimensions		29 x 44 x 66 mm (HxWxD)	
Weight		150 g	

Trigger/Readout Modes



Connection Description

DC-IN/SYNC.

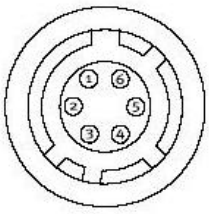


HIROSE HR10A-10R-12P

Pin		
	1	Ground
	2	+12V DC
	3	Ground
	4	Video output
	5	Ground
	6	HD input/HD output
	7	VD input/VD output
	8	Ground
	9	Pixel clock output
	10	WEN output
	11	Trigger input
	12	Ground

(Pin configuration compatible with EIAJ standard)

TRIGGER



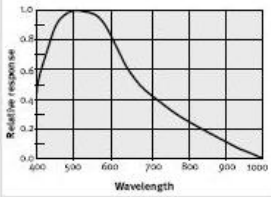
HIROSE HR10A-7R-6P

Pin		
	1	TXD
	2	RXD
	3	Ground
	4	Ground
	5	Trigger input
	6	EEN output

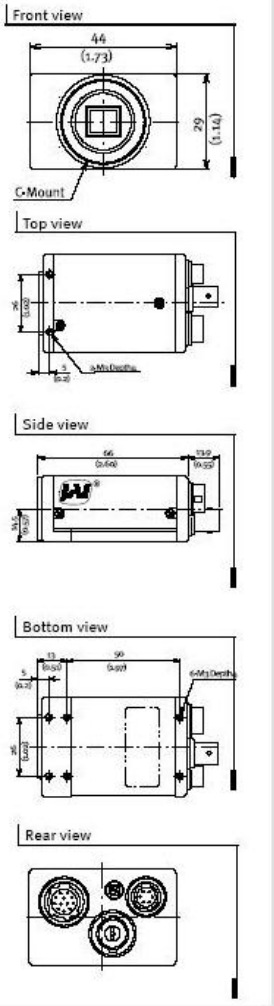
Plugs for cable:

12 pin: HIROSE HR10A-10P-12S
6 pin: HIROSE HR10A-7P-6S

Spectral Sensitivity



Dimensions



Ordering Information

CV-A50C 1/2" Compact Industrial Monochrome CCD Camera. CCIR
CV-A60C 1/3" Compact Industrial Monochrome CCD Camera. CCIR
CV-A50E 1/2" Compact Industrial Monochrome CCD Camera. EIA
CV-A60E 1/3" Compact Industrial Monochrome CCD Camera. EIA
MP-40 Tripod Adapter (must be ordered separately)
MP-70 Angle Adapter (must be ordered separately)

JAI A-S, Denmark
Phone +45 4457 8888
Fax +45 4491 8880
www.jai.com

JAI Corporation, Japan
Phone +81 45 440 0154
Fax +81 45 440 0166
www.jai-corp.co.jp

JAI UK Ltd., England
Phone +44 1895 821481
Fax +44 1895 824433
www.jai.com

JAI PULNiX Inc., USA
Phone (Toll-Free) +1 800 445 5444
Phone +1 408 747 0300
www.jai.com

JAI PULNiX, Germany
Phone +49 (0) 6055 9379 10
Fax +49 (0) 6055 9379 11
www.jai.com

THE MECHADEMIC COMPANY

Visit our web site on www.jai.com