UiT The Arctic University of Norway

Faculty of Science and Technology
Institute of Technology and Safety

Implementation and Control of an Inverted Pendulum on a Cart

Halvard Hanekam
Master's thesis in Technology and Safety in the High North, Tek-3901 June 2021

PREFACE

This master's thesis is written at the Department of Technology and Safety, Faculty of Science and Technology at UIT The Arctic University of Norway. The master's thesis concludes the 2-year master's program in Technology and Safety in The High North.

I would like to thank my supervisors Anne Mai Ersdal and Per Anton Øverseth Olsen for ideas, support, and feedback in the process of writing this thesis. I would also like to thank my girlfriend Inga Elda for support and motivation.

Halvard Hanekam, 30.05.2021

ABSTRACT

An Inverted Pendulum on a Cart is a common system often used as a benchmark problem for control systems. The system consists of a cart that can move in one direction on the horizontal plane and a pendulum attached to the cart through a hinge point. The pendulum can rotate 360° on the plane made up of the vertical direction and the direction the cart can move. The system is controlled by applying a force to the cart, to make it move.

This thesis consists of two goals. The first goal is to build a lab model of the Inverted Pendulum on a Cart system. The second goal is to create a controller that can swing the pendulum from a pendulum down position to a pendulum up position, and balance it in this position.

The lab model is built using a track that the cart can move along, a stepper motor for applying force to the cart and a microcontroller for controlling the system. The pendulum angle and the cart position are measured using incremental encoders.

A Mathematical model of the system have been derived. This forms the basis for the design of the controller and is also used for simulating and testing the system and controller in MATLAB/Simulink before it is implemented on the real system.

The controller consists of three parts. An extended Kalman filter is implemented to estimate the non-measurable state. An energy-based controller is used to swing the pendulum from the down position to the up position. This controller regulates the energy in the pendulum to be close to the energy the pendulum should have when it is balanced in the upright position. When the pendulum is close to the upright position the controller will switch to a linear quadratic regulator to balance the pendulum. This controller is based on a linearized version of the mathematical system model.

The lab model and the controllers have been successfully built and implemented.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

COG	Centre Of Gravity
COM-port	Communication port
DC	Direct Current
EKF	Extended Kalman Filter
GPIO	General Purpose Input or Output
IPC	Inverted Pendulum on a Cart
KF	Kalman Filter
LED	Light Emitting Diode
LQR	Linear Quadratic Regulator
LTI	Linear Time Invariant
PC	Personal Computer
PID	Proprotional Integral Derivative (controller)
PWM	Pulse Width Modulation
SPI	Serial Peripheral Interface
UIT	Universitetet iTromsø
USB	Universal Serial Bus
VDC	Volt Direct Current

1 INTRODUCTION

The inverted pendulum on a cart (IPC) is a classical control system problem that is widely used booth for education and testing of control algorithms. The problem consists of a pendulum mounted on a cart. The goal is to balance the pendulum in an upright position using a force acting on the cart in the horizontal plane as the control input, as illustrated in Figure 1.1.

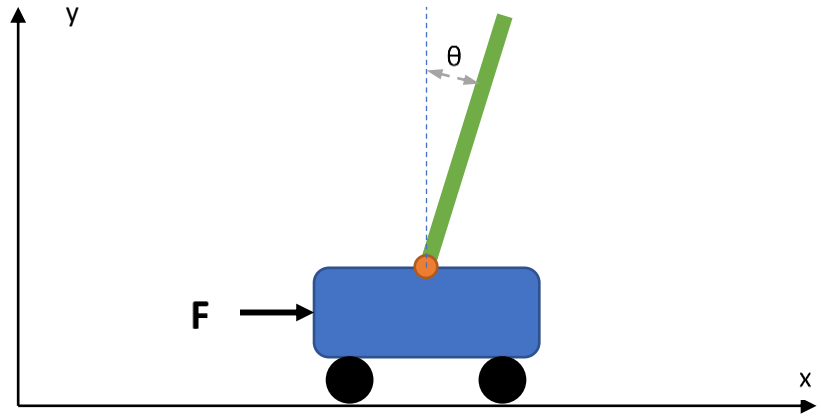


Figure 1.1 Inverted Pendulum on a Cart.

The inverted pendulum problem is often seen as an analogy to balancing a stick on your hand, or a person standing upright. The problem has many similarities to real world applications such as the two wheeled balancing transportation device, commercially known as the Segway[1].

From a control system perspective, the IPC system has many interesting properties. Due to the rotation of the pendulum, the system is highly nonlinear. The system is underactuated as it only has one input, the force acting on the cart in the horizontal direction, and two degrees of freedom, the cart position, x, and the pendulum angle, θ. With the pendulum in an upright position, the system is unstable and therefore requires active control to maintain this position.

1.1 Project background

The idea for this project was given by Per Anton Øverseth Olsen, lecturer at UIT. Olsen has planned to make a lab model of the IPC system for use in lab exercises in courses teached at UIT. It was therefore decided that this master thesis project would be to build an IPC for implementing control algorithms for this thesis, while the model also can be used later, for educational purposes.

This project was found interesting as it covers a large part of the topics in the master's degree program, such as electronics, embedded systems, and control theory. The project is diverse, as it contains designing of the pendulum system, selecting suitable components, the practical part with designing the model, coding of microcontrollers and control theory.

1.2 Literature review

Numerous lab models of the IPC system has been implemented ranging from simple low-cost solutions to expensive solutions with high performance. Examples of low-cost solutions are [2], where the system is controlled with a microcontroller and the cart input is generated by a stepper motor and timing belt, or [3], where parts salvaged from a printer are used for the cart sliding mechanism, a DC-motor powered by a servo-drive amplifier is used for generating the force input to the cart, and the system is controlled by a PC with a data acquisition and control board where the controller is running in Simulink. On the high-cost end of the specter there are system such as the one implemented in [4]. This system has a triple pendulum, where each pendulum is attached to the previous one. With the extra pendulums the requirements measurement resolution, control input and computing power increases, making the system more expensive. Other IPC system used as references in this thesis, such as the ones in [5] and [6], will be closer to the low-cost end of the specter, as they have a single pendulum.

Some of the control problems associated with the IPC system are to balance the pendulum in the upright position and to swing it from a downward position to the upright position. For balancing the pendulum the solution used in [2] is among the simpler ones. Here a PID[1] controller is used to control the pendulum angle. In this case the cart position cannot be controlled. In one of the solutions in [7] an PID controller is also added to control the cart position. This makes it possible to balance the pendulum and control the cart position. In [3] a LQR[2]-controller, with an augmented state to generate integral action, is used. The feedback gain for the LQR controller is found based on a mathematical model linearized about the upright pendulum position.

There are several possible solutions for swinging up the pendulum. A popular method is the energy based controller first described by Åstrøm and Furata [8]. Here the energy in the pendulum is controlled and the energy the pendulum will have in the upright position is used as a reference. In [9] the method is further developed to also include methods that ensures that system limitations such as track length and maximum velocity is not violated. Another option is to use a Fuzzy logic controller such as the one used in [5]. This is a rule-based controller where the control input is such that the pendulum energy is increased while not violating system limitations. In both these cases the controller must switch to a controller for balancing the pendulum when it is close to the upright position.

A third option for the swing up controller is to use a combination of a feedforward and a feedback controller such as in [6]. Here a feedback controller is used to make the system follow a trajectory calculated by the feedforward controller. The feedback controller is similar to a LQR controller, where the feedback gain is recalculated at every timestep, based on a linearization around the current state. The reference trajectory for the feedforward controller is calculated in advance by an optimization problem that includes the mathematical model and system limitations as constraints.

[1] PID - Proportional, Integral, Derivative
[2] LQR – Linear Quadratic Regulator

1.3 Project description

This project has the following two goals:

- Build a model of the Inverted Pendulum on a Cart system that can be used for testing of control algorithms.
- Implement a controller that can swing the pendulum from a pendulum down position to a pendulum up position and balance the pendulum in this position.

The IPC model will serve as a test platform in this project and will after the project end be given to UIT, for use in lab exercises.

Figure 1.2 shows a sketch of the IPC model. A system similar to what is presented in [10] and [3] is used. The cart is sliding on a low friction railing, and the force input is applied to the cart using an electric motor and a timing belt and pulley system. The pendulum will be mounted out from the side of the cart, allowing it to rotate a full 360°. A microcontroller will be used to control the motor in real time based on measurements of the cart position, pendulum angle and the implemented control algorithm. A mathematical model of the system will be derived. This is used both to develop the control algorithm and for testing and tuning the algorithm in MATLAB/Simulink. C++ code is generated from the control algorithm implemented in Simulink using *Embeded Coder*. This is an extension to MATLAB/Simulink that can generate *c* or *c++* code from MATLAB or Simulink code for use in embedded processors [11]. This code is then used on the microcontroller for implementation on the real model.

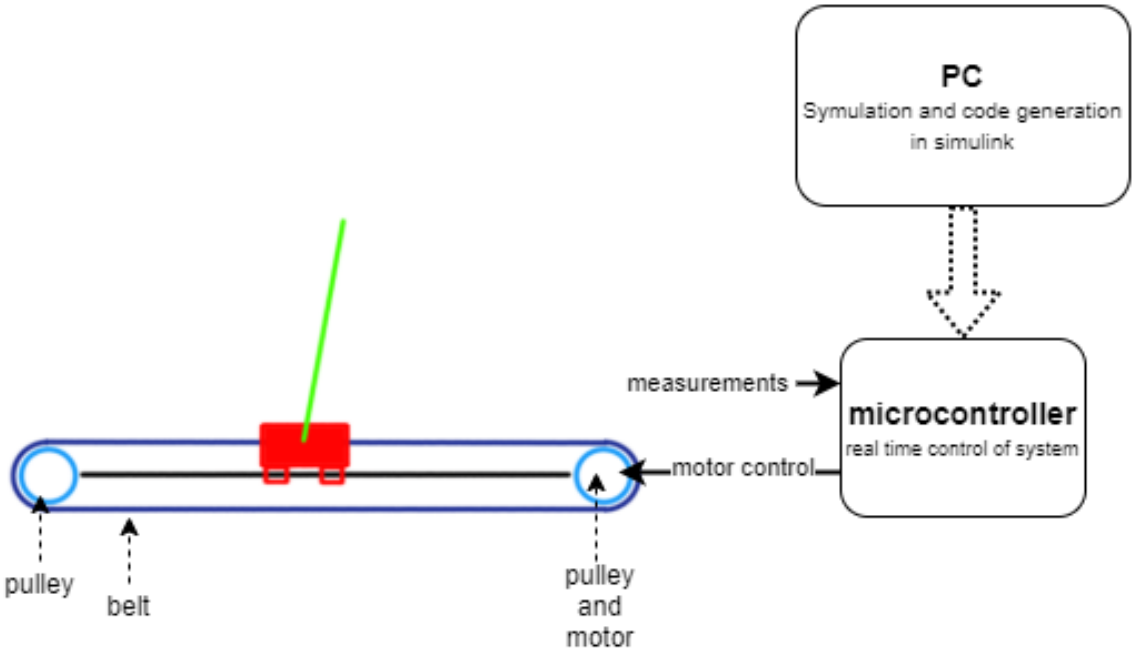


Figure 1.2 IPC model principal sketch.

1.4 Thesis structure

Chapter 2 describes the building of the IPC-model. This includes descriptions of design and component choices and the system control structure, but not the actual control algorithm that calculates the control input. This chapter will cover the first goal of the project. For any future users it should be sufficient to read this chapter for understanding how to use the model.

Chapter 3 covers the derivation of the mathematical model of the system. It also includes theory about the methods used and results and discussions of tests used to find parameters for the mathematical model. The mathematical model is finally verified against the actual system.

Chapter 4 describes the control algorithm used to calculate the system input. The controller consists of the following parts: a *Kalman Filter* based state estimator, a control algorithm for swinging up the pendulum and a *Linear Quadratic Regulator* used to balance the pendulum in the upright position. Each part has its own section, describing the theory of the part, how it is implemented, and tests and analysis used for making decisions and tuning.

Throughout the thesis tests, analysis and discussions will be presented in the chapters and subsections, as these are used to design the system and the controller. This means that the results in chapter 5 and the discussion in chapter 6 will be for the final implementation and will mainly not include any results and analysis used in the previous chapter.

A final summary is given in chapter 7.

2 IPC-MODEL

This chapter describes the planning, design, and building of the inverted pendulum on a cart model. The chapter starts with section 2.1 about the design basis and description of the major components used. Here the necessary specifications for the system will be defined based. Then the components will be described along with the reasoning for choosing the components. In section 2.2 the design of the system is presented. This will show how components are mounted together. Section 2.3 describes the electrical system and how the system components are connected. Section 2.4 describes the control structure of the system. This describes how the system is initialized, started, and stopped, how the motor is controlled and how sensor data is read. It does not include the actual control algorithm for calculating the control input, this is found in chapter 4.

2.1 Design basis and components

The project has a budget limit at 5000 NOK. To reduce costs the cart and most mounting brackets are 3D-printed. It has also been emphasized to keep the weight of the cart and the pendulum low, as this will decrees the necessary input force, and then also reduce the cost of the motor, motor controller and power supply. Components for this project have been supplied by Norwegian suppliers or suppliers with quick delivery to Norway and a system for collecting Norwegian Vat. This is to reduce the risk of delays from long shipping or Norwegian customs declaration. This has had a large impact on the design phase, as many cheap parts has been excluded.

Based on research papers concerning the use of an IPC model , such as [3, 6, 10], it was decided to aim for making the system capable of a cart acceleration of $20m/s^2$, a maximum cart velocity of $2m/s$, a track length of $80-100\text{cm}$, a cart weight around 0.350kg and a linear pendulum with weight $100-250\text{g}$ and length 0.40m. These numbers are not absolute requirements, but they will be sufficient for most IPC control problems, while also being achievable within the budget limits.

One major design decision was the choice of the motor. As a brushless DC-motor and controller was considered too expensive, there were two options left, a brushed DC-motor or a stepper motor. Both alternatives are capable of the task, but they have quite different characteristic that will affect the rest of the design and the modelling of the control system.

Brushed DC-motor

The stator[3] of the brushed-DC motor has a set of fixed permanent magnets, creating a magnet field. The rotor[4] consists of coils that can carry a current received through brushes and commutators. When a current is passing through the coils inside the stator magnet field, they will experience a force normal to the current. This will make the rotor rotate. The coil current

[3] Stator is the non-rotating outer part of a motor.
[4] Rotor is the rotating part of the motor.

is changed continuously through the brushes and commutator, causing the motor to rotate continuously [12, pp. 279-285].

The torque output from a DC-motor is given by

$$T_m = \frac{k_t}{R} U - \frac{k_t^2}{R} \omega \tag{2.1}$$

where $k_t[Nm/A]$ is the motor torque constant, $R[\Omega]$ is the motor coil resistance, $U[V]$ is the motor input voltage and $\omega[rad/s]$ is the motor angular velocity[5][12, p. 284]. The first term shows that the output torque is increasing linearly with the voltage input. The second term shows that the torque is decreasing linearly with the angular speed. This is caused by back-emf generated when a conductor is moved in a magnetic field.

The force produced by the motor and pulley system can then be found by

$$F_m = \frac{T_m}{r} \tag{2.2}$$

where r[m] is the pulley radius, acting as the arm length in the torque formula [13]. Further in the IPC model the force acting on the cart will depend on force any inertial forces and friction forces generated in the system

This means that when using a DC-motor the force acting on the cart will be a function dependent on the motor characteristics, inertia in the system, the friction in the system and the angular velocity of the motor. To achieve good performance of the control system, this will have to be modelled as there will not be a linear relationship between the input voltage and the force acting on the cart.

Since the system will be controlled by a microcontroller, an amplifier circuit or voltage/current control unit will be needed to control the larger voltage and current going to the motor.

Stepper motor

Stepper motors are generally built with permanent magnet or soft iron motor and a stator consisting of multiple coils. When a set of coils are energized the permanent magnet in the rotor will align with the generated magnet field. The rotor will stay in this position until the magnet field is changed. The magnet field is altered by changing the energization of the coil sets. If a set of coils are continuously energized, the motor stays still. This gives the motor the stepping characteristics, where the energization of the coil sets must be continuously altered to get a continuous rotation. This means that a controller, usually called a stepper driver, is necessary to control the rotation of the stepper motor [12, pp. 287-303].

The stepping characteristics of a stepper motor and the use of the stepper motor driver means that the angle, angular velocity, and angular acceleration of the stepper motor is controlled, and not the output torque. This is under the assumption that the motor can produce enough torque

[5] This is a simplified equation not including the motor reactance, which usually has a low impact on the outcome.

to move the load. If not, the motor will stall, and the rotation stops. This means that the chosen stepper motor and driver combination must be able to produce enough torque at all possible requirements.

The torque of a stepper motor is at its maximum at zero velocity, called the holding torque. At this state there is no change to the energization of the motor coils or any movement of the rotor. This means that no back-emf is generated. Because a stepper motor must accelerate, move, and decelerate at every step, the stepper motor torque output will drop when the velocity is increased, due to the generated back-emf and the forces needed to move the rotor. Typically the torque output of a stepper motor will drop at a lower velocity than a comparable DC-motor [12, pp. 298-302].

2.1.1 Motor and drive system components

It was decided to use a stepper motor for IPC model. There was no clear advantage of using a stepper motor over a brushed DC-motor, so the decision was made mainly on the available parts for timing belt, pulley, motor, and controller had a better fit for the desired specification of the system. A quite limited choice of long enough timing belts, and the corresponding timing belt pulleys did not fit the motors or had a large radius that made the motor torque to small.

It was considered an advantage that the stepper motor option doesn't require modeling of the motor, inertia, and friction of the cart rail and timing belt system. The necessity of the modelling itself is no problem, as this is a normal procedure for control systems. The issue is that it is expected that the IPC system will need adjustments. This can affect the modeled parameters. Anny disassembly and reassembly could also change the timing belt tension and then also it's friction. With a stepper motor the control input will always produce the same cart movement output. This also reduces the effect that a low-quality car rail, with nonlinear friction, will have on the system performance. On the backside, additional programming of the microcontroller for interface and setup of the stepper motor driver is required.

Table 2.1 to Table 2.4 shows some of the specifications of the stepper motor, stepper motor driver, timing belt and timing belt pulley used. These components were considered sufficient for the desired velocity of $2m/s$ and $10\mathrm{m/s^2}$ based on the following calculations. The expected total weight of the cart and pendulum is $600g$. To add forces from friction, inertia and dynamic movements in the pendulum, motor and drive system should fulfill the acceleration and velocity requirements for a weight of 1.2kg with no friction, inertia, or other forces acting on the system. This is a very crude estimation, but it will give a good indication if the motor fulfills the requirements.

Figure 2.1 shows the torque vs speed curve for the motor, and the required torque and speed required by the IPC model is marked with a red box. The required torque is calculated to

$$T_{requiered} = m * a * pulley\ radius = 1.2kg * 20\frac{m}{s^2} * 0.0255m = \underline{0.612\ Nm} \qquad (2.3)$$

The velocity in motor steps per second is calculated to

$$v_{step} = \frac{v * steps\ per\ round}{pulley\ circumference} = \frac{2\frac{m}{s} * 200\ steps}{0.16m} = \underline{2500\frac{step}{s}} \qquad (2.4)$$

It should be noted that the torque curve for the motor is made for a phase current of 3.0A, while the motor will be used with the nominal phase current of 2.7A. This means that the real motor output torque will be slightly less, as the motor torque is proportional to the current [14]. Figure 2.1 shows that the motor can supply the required torque through the whole velocity range with around 30% margin.

Table 2.1 Trinamic QSH-6018-45-28-110 stepper motor specifications [14].

Stepper motor: Trinamic QSH-6018-45-28-110	
Rated Voltage	2.1 V
Max voltage	75
Nominal phase current	2.8A
Max phase current	3.0A
Holding Torque	1.1 Nm
Rotor Inertia	275 gcm^2
Weight	0.6 kg
Step Angle	1.8°
Steps per round	200
Shaft diameter	8mm
Nema size	24

Table 2.2 Trinamic TMC 5160 BOB stepper motor driver specifications and features[15, 16]

Stepper driver: Trinamic TMC5160 BOB	
Break out board based on the TMC5160-TA stepper controller	
Supply voltage	9-36V
Max nominal phase current	2.8A
Configuration	Through SPI communication
Control	Through SPI communication or step/dir input
Highest resolution	256 microsteps per full step
StealthChop2™ for quiet operation and smooth motion	
Resonance Dampening for mid-range resonances	
spreadCycle™ highly dynamic motor control chopper	
dcStep™ load dependent speed control	

Table 2.3 Timing belt specification.

Timing belt	
length	2.525m
width	9mm
pitch	5mm

Table 2.4 Timing belt pulley specification.

Timing belt pulley	
pitch	5mm
No of teeth	32
Effective circumference	160mm
Effective radius	25.5mm

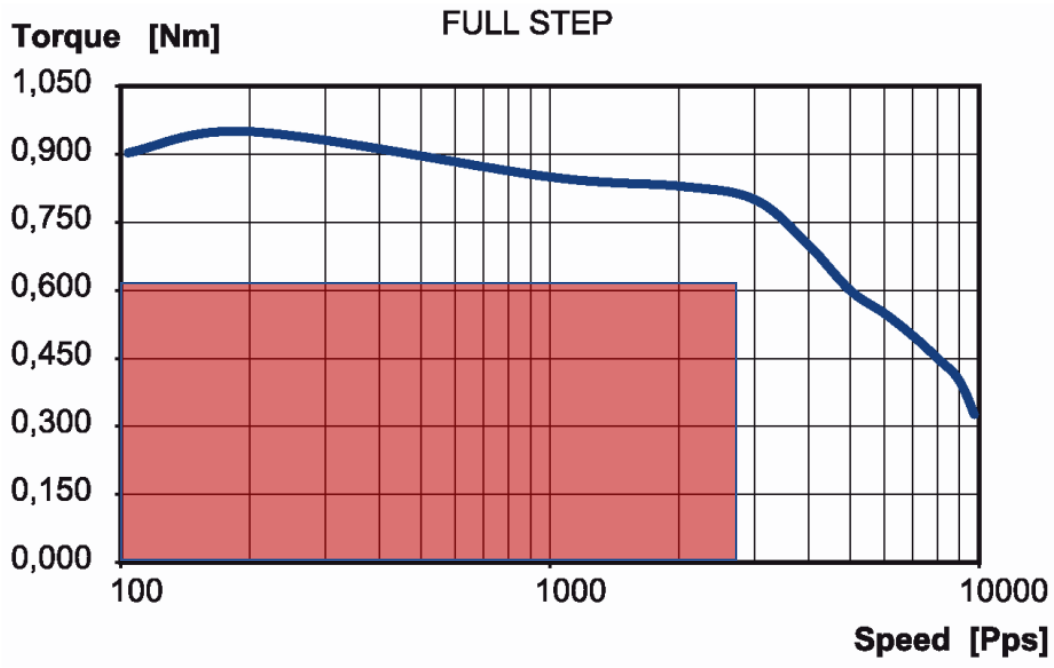


Figure 2.1 QSH6018-45-28-110 torque curve with 30V supply and 3.0A RMS phase current. Blue line shows motor torque vs speed curve. Red box shows necessary working characteristics for the IPC model. [14]

2.1.2 Microcontroller

The microcontroller used in this project is a Teensy 4.1 development board based on an ARM Cortex-M7 processor running at 600MHz. This is a very fast microcontroller that can do 64bit double precision math in hardware, making it suitable for the high number of math computations that are necessary in control systems. It has 4 hardware implemented quadrature decoders that are used for decoding input from encoders. This is considered a big benefit, as the encoders otherwise must be decoded via interrupts. With the two encoders used in this project, this would generate many interrupts that may slow down other processes in the code [17].

Further the Teensy fulfils all other requirements[17] such as:

- Communication to PC via USB and virtual serial comport or through hardware serial ports.
- SPI for communication to stepper motor driver
- Total of 55 I/O pins with properties such as interrupts, PWM output, pullup/pulldown resistors, analog input for some of the pins.

The microcontroller can be programmed using the *Arduino IDE* with the *Teensyduino* ad on. This simplifies the code development as it includes many libraries for interfacing the low-level settings and use of the microcontroller.

2.1.3 Encoders

The encoders used for this project is Broadcom AS22-M560-4A12 encoders. These are low cost and low size two channel (A and B) encoders. The channels have 2000 pulses per rotation and the waveforms have a 90° phase difference [18]. With quadrature decoding this gives 8000 steps per rotation. For the cart position we then get a resolution of

$$cart\ pos\ resolution\ = \frac{pulley\ circumference}{8000} = \frac{160mm}{8000} = \underline{0.2mm}. \qquad (2.5)$$

And for the pendulum angle we get a resolution of

$$pendulum\ angle\ resolution = \frac{2\pi}{8000} = \underline{7.85 * 10^{-4} rad} = \underline{0.045°}. \qquad (2.6)$$

The resolution of the encoders is considered to be sufficient for the intended use. The chosen encoders are the ones found with highest resolution at a price range within the budget. Further the encoders have very low weight compared to alternatives, this helps to reduce the weight of the cart.

The encoder channel output is 5V, while the maximum input voltage on the microcontroller is 3.3V. The encoder output is therefore stepped down to 3.3V using a SN74LVC245A buffer.

2.1.4 Cart track

The chosen cart track is an aluminium profile with V-shaped groves. The cart is connected to the track with delrin[6] wheels that fit in the V-grove. The wheels are mounted on ball bearings to ensure low friction. This solution was chosen as it has low friction between the cart and the track, and low price compared to comparable options.

2.1.5 Cost

Table 2.1 shows the cost of the components used in the IPC-model.

Table 2.5 Cost of IPC-model.

Component	price (NOK)
stepper motor	754
stepper motor driver	230
microcontroller	316
timing belt and pulley	401
aluminium profile and wheels	540
encoders (2 pieces)	705
miscellaneous electronics and mechanical parts	600
sum	3546

2.2 Design

Figure 2.2 shows a picture of the built inverted pendulum on a cart model. As shown in the figure the cart slides on the track. The timing belt connecting the cart to the motor, runs above and below the track, and is connected to the cart above the track. The motor is mounted with a housing around its timing belt pulley on the right side of the track. On the left side of the track the other pulley is mounted to a similar housing and mounted with the encoder for cart position measurements. The timing belt pulleys are capsulated with the housing to reduce the risk of anything, for example hair, to get caught in the rotating pulleys and to give protection. Both housing have flat pads that can be used to fasten the model to a table with a clamp. The length between the two housings is 1.05 m.

[6] Delrin is a thermoplastic often used in high-load mechanical applications.

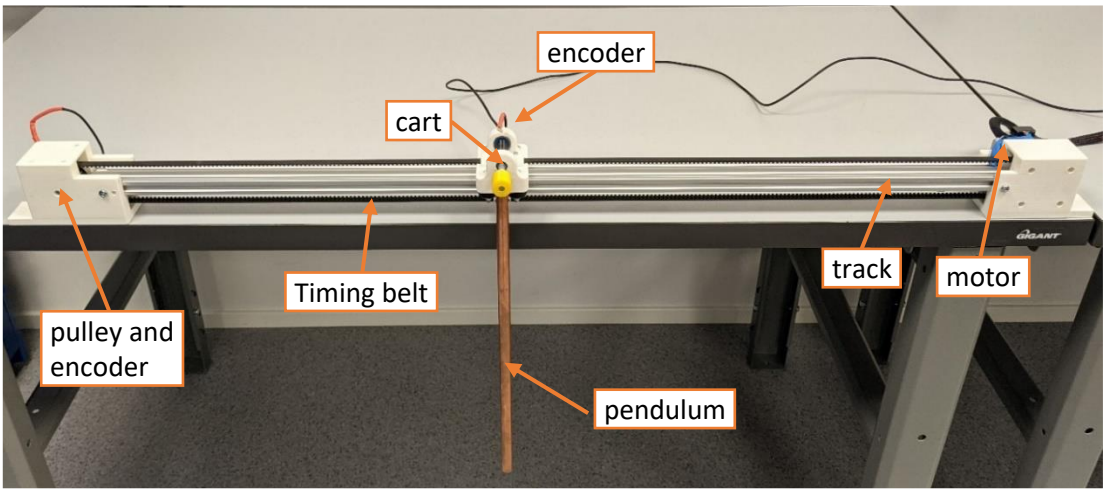


Figure 2.2 Picture of the built IPC model.

2.2.1 Cart and pendulum design

Figure 2.3 shows pictures of the cart, which is 3d-printed printed. Picture *b)* shows how the cart is connected to the cart with two delrin wheels on each side of the track. To fasten the timing belt, a pattern matching the timing belt is made on the top of the cart. The belt is secured in this track by a plate.

The pendulum is connected to the cart by a shaft made of an 8mm threaded rod. The pendulum is fastened to the shaft with and 3d-printed joint. The joint is screwed on the shaft and secured with a nut. The shaft is connected to the cart via two ball bearings. The shaft is secured in the ball bearings by a 3d-printed clamp that snaps onto the shaft in the middle of the ball bearings. As seen in Figure 2.3 *a)* the encoder code wheel is fastened to the shaft with a set screw. There is no contact between the encoder electronics board and the code wheel. This means that the encoder does not create any friction.

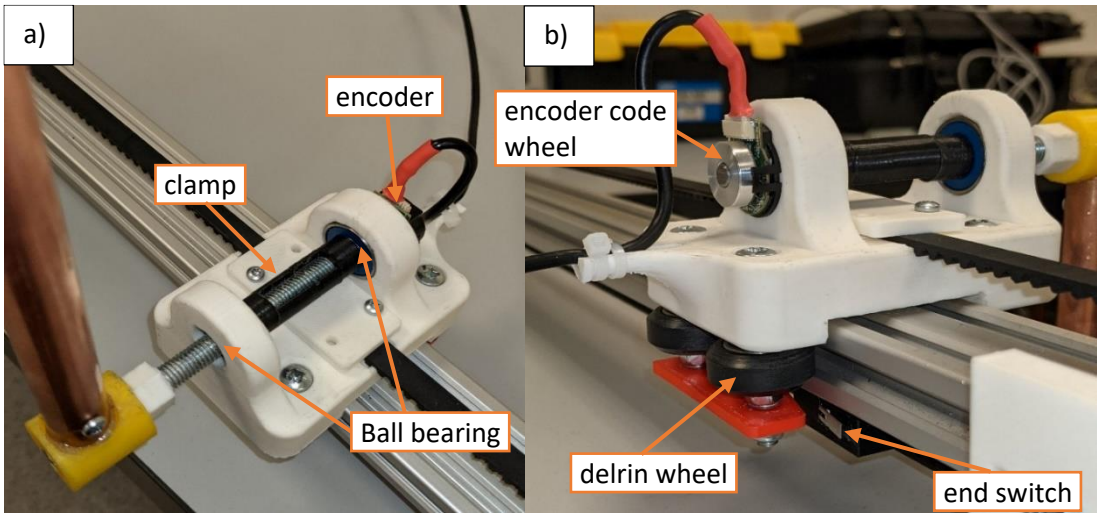


Figure 2.3 Pictures of the cart.

Figure 2.3 *b)* shows the mounting of the end switch underneath the track. The switch will be triggered by the red plate under the wheels. One switch is mounted on each side in position such that it will be 4cm between the cart and the track end.

The cart is designed with a focus on keeping the weight low, to ensure that motor can achieve a high acceleration. The total weight without the pendulum is measured to 0.170kg. The low weight of the cart means that its strength is reduced, and that the weight of the pendulum also must be low. Several pendulums have been tested and it was found that a too heavy pendulum made the cart flex and bend, while a too light pendulum will be very affected by the friction in the ball bearings. The pendulum that is used is made of a copper pipe. It weighs 0.14kg and has a length of 45.9 cm from the centre of the shat to the end of the pendulum.

2.2.2 Motor and pulley housing design

Figure 2.4 shows the mounting of the motor to the housing around the timing belt pulley. It is made such that the motor is covered as little as possible, to ensure good cooling of the motor. The timing belt pulley is fastened to the motor shaft by a set screw.

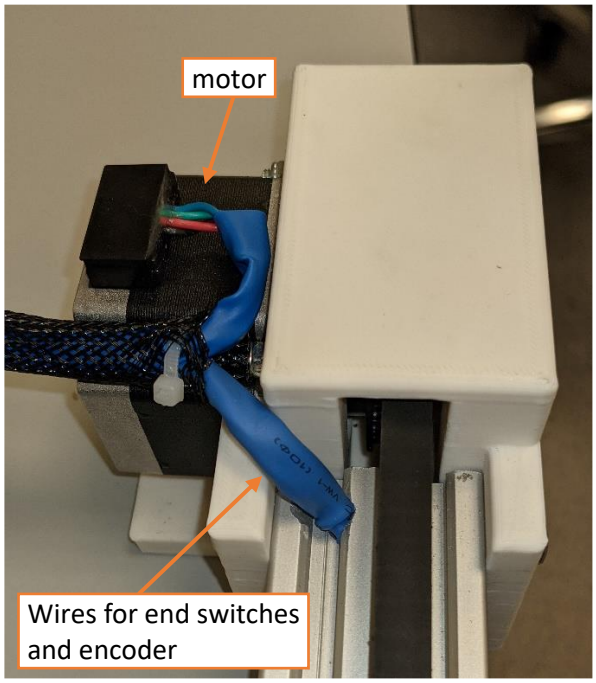


Figure 2.4 Picture of stepper motor mounting.

Figure 2.5 shows how the timing belt pulley at the left side of track and the encoder is mounted. The timing belt pulley is fastened to a screw going through its centre hole, with ball bearings on both sides. The ball bearings are seated in the housing. The housing cover will be pushing against the ball bearings, making the whole assembly fixed. The encoder code wheel is mounted at the end of the screw.

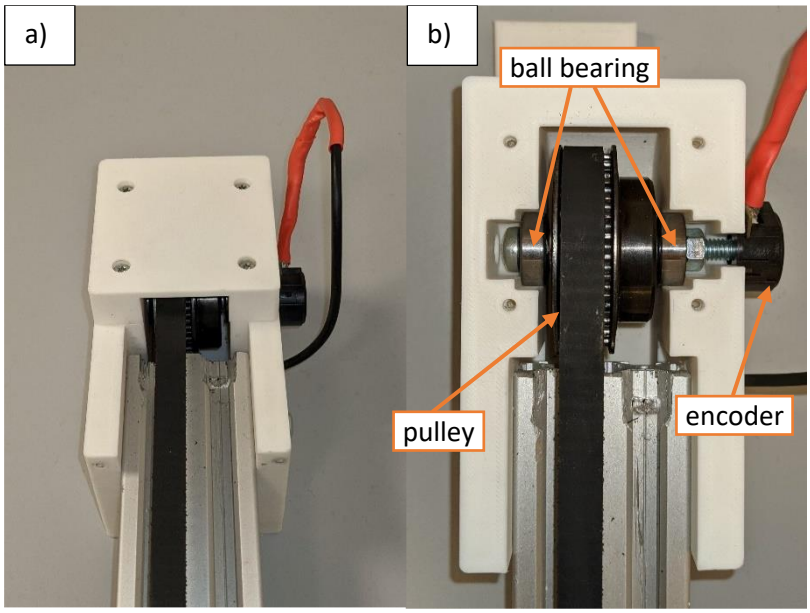


Figure 2.5 Picture of mounting of timing belt pulley with encoder at the left end of the track.

2.2.3 Control box

Figure 2.6 shows the system control box. This contains the electronic board with the microcontroller and the stepper motor driver, start/stop buttons, status LEDs and power supply connections. The box is made without a lid because a button on microcontroller in some cases must be pushed when code is uploaded. The electronics board will be described in the following section.

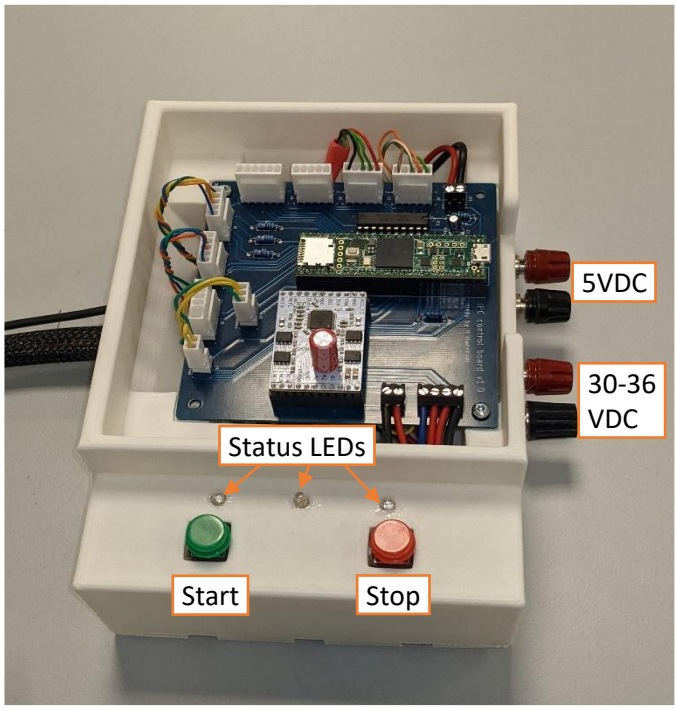


Figure 2.6 Picture of control box with electronic board, start/stop button, and status LEDs.

2.3 System electronics

Figure 2.7 shows a block diagram of the electronic components in the system. A circuit board has been made for holding the microcontroller, stepper motor driver, the buffer, and connectors for all the peripheral components. The circuit board needs a 30-36VDC supply that can deliver 3A for powering the stepper motor driver and the stepper motor and 5VDC capable of up to 0.5A to power the rest of the electronics. The 5VDC can be supplied from the USB or from an external 5VDC power supply. The microcontroller board generates 3.3VDC.

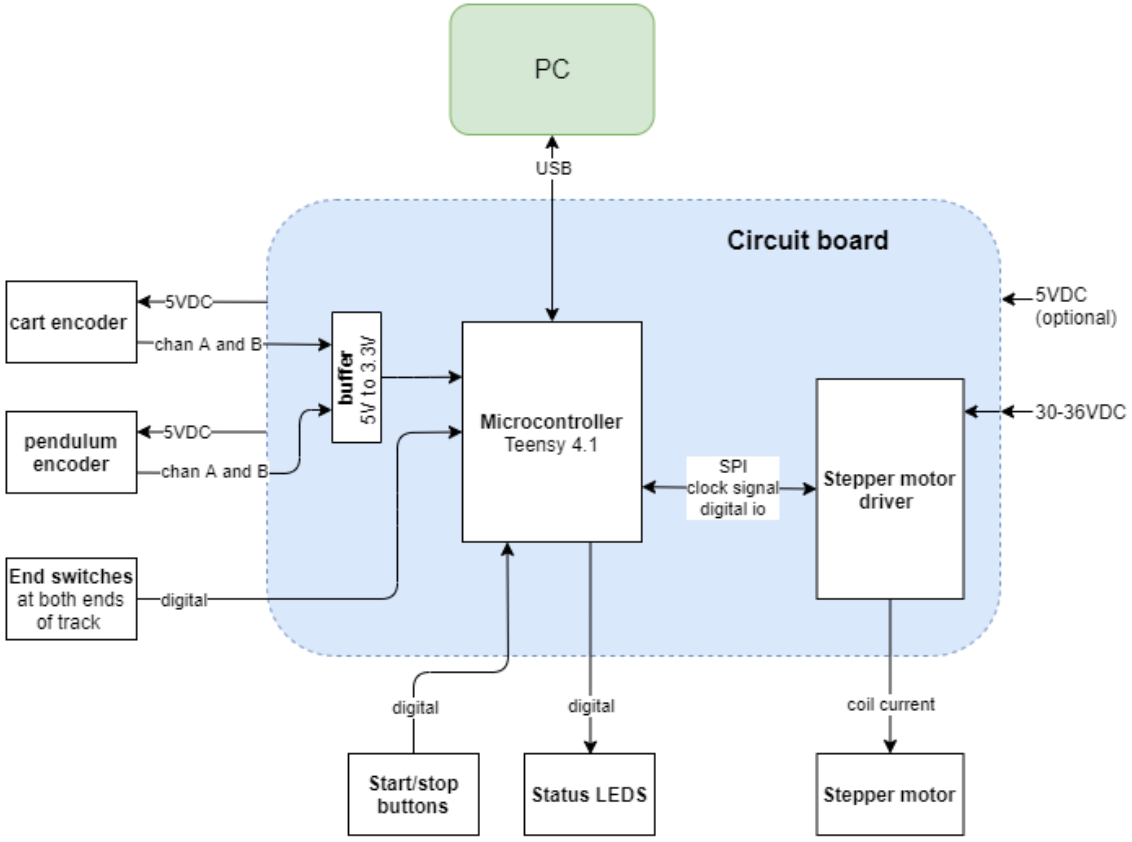


Figure 2.7 Block diagram electronic components.

The PC is connected to the microcontroller using USB. This is used to upload code to the microcontroller and serial data is sent back to the PC for logging through a virtual COM-port. The system can run without the PC connected, if an alternative 5VDC power supply is used.

The encoders are supplied by 5VDC from the circuit board. The A and B channel signals are passed through the buffer to reduce the voltage from 5V to 3.3V. The signals are then passed to the microcontroller.

The end switches and the buttons are connected to the microcontroller as source inputs and the status LEDs [7] are connected to the microcontroller as source outputs. The LEDs have

[7] LED – light emitting diode

330Ω resistors in series, to limit current. The LEDs have a diode voltage of 1.5V and the microcontroller output voltage is 3.3V. this gives a maximum current of

$$I = \frac{U}{R} = \frac{3.3V - 1.5V}{330\Omega} \approx 5.5mA. \quad (2.7)$$

This is well within the microcontroller maximum source/sink current of 10mA.

There are several connections between the microcontroller and the stepper motor driver:

- SPI[8] serial communication running at 5MHz is used for communication between the microcontroller and the stepper motor driver. This is used for controlling the stepper motor driver.
- 15MHz clock signal. This is used by the stepper motor driver to improve the accuracy control of stepper motor velocity and acceleration.
- vcc_io is used for turning on and setting a voltage reference for the input to the stepper motor driver. This is set to 3.3V
- DRV_enn is used to enable the output from the stepper motor driver to the stepper motor. The output is active if DRV_enn is low. The DRV_enn pin is therefore connected to 3.3V by a pull up resistor and must be driven low by a source output on the microcontroller. This means that the motor is normally disabled.
- The stepper motor driver also has digital step and direction control inputs and a digital output diagnosis flag that can be used for generating interrupts on the microcontroller. these are not used in this project, but they are connected.

Figure 2.8 shows a picture of the circuit board. In addition to the above-mentioned connection between components, some spare connectors for the following has been added for future use:

- One extra encoder input.
- Connector to the microcontrollers SPI communication.
- Connector to microcontroller pins for GPIO[9] and various serial communication protocols.
- Connector for emergency stop button. A normally closed switch should be used here. This will be connected in series to the DRV_enn control from the microcontroller. If it is activated, it will disable the stepper motor driver directly. This has not been used in this project because the power supply already had an emergency switch. The pins in the connector has therefore been shorted.

The complete electric schematic of the circuit board and connection to all components are added in the appendix.

[8] SPI – serial peripheral interface is a serial communication protocol.
[9] GPIO – general purpose input

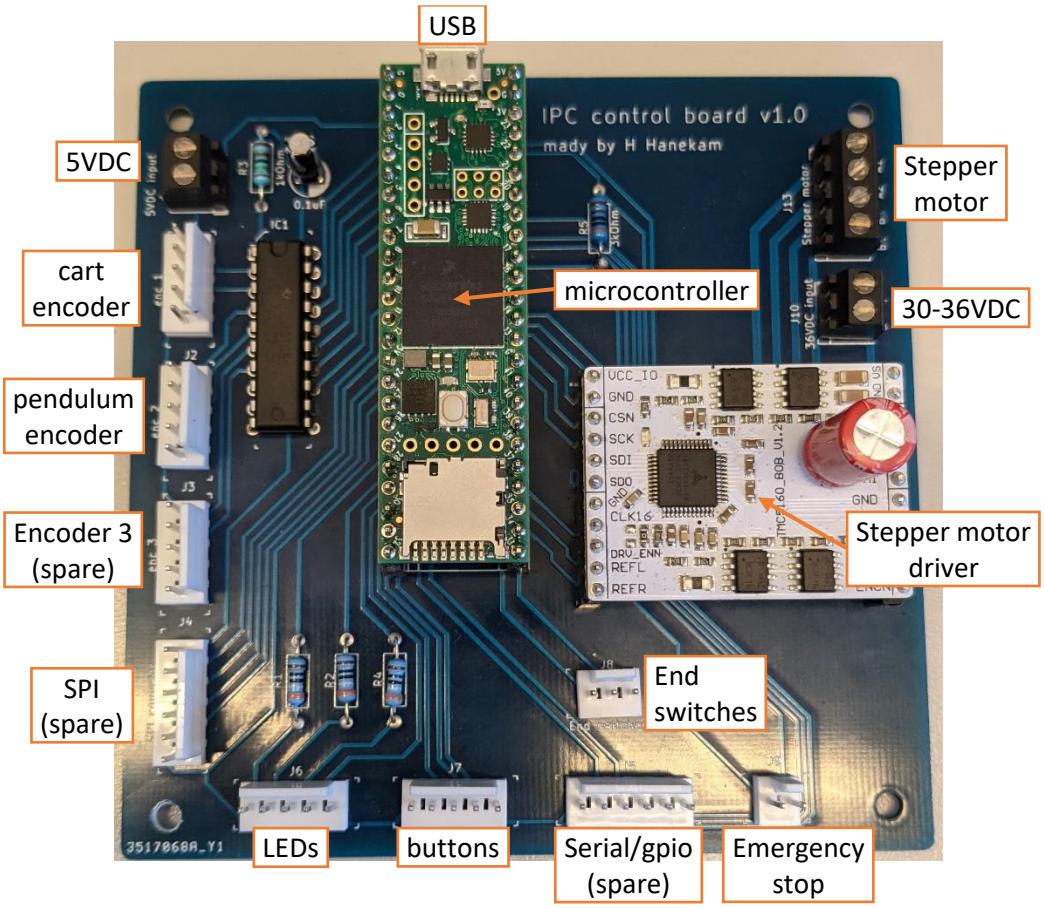


Figure 2.8 Circuit board for the IPC system.

2.4 System control structure

The control system is based on the sequential repetitive illustrated in Figure 2.9. The loop will run at a free frequency but blocks in the loop will only update at the frequency indicated in the figure.

The loop starts by updating the control algorithm. This is the controller that calculates the control input given to the cart, based on the encoder measurements. This control algorithm will be covered in chapter 4. The calculated control input will however be assigned in the next block, the update of the system state machine. This block runs at an update frequency of 200Hz.

The state machine governs the overall function of the system, such as starting and stopping, homing/initialization of encoders and motor, and stopping the system if the cart approaches the end of the track. The state machine is presented in section 2.4.1. The state machine is updated at a free frequency.

In the final block the states of the pendulum and the cart and the control input is sent to the PC through USB serial communication for logging. This is running at a 200Hz update frequency synchronized with the update of the control algorithm.

With regards to the update frequencies it should be noted the update state machine block computational cost is low. Its execution time has been measured with the microcontroller on

board clock to be less than $10\mu s$. This means that its running time should not create any significant variation in the update frequency of the update algorithm block. The execution of the loop when all three blocks are updated is measured to be less than 100μs.

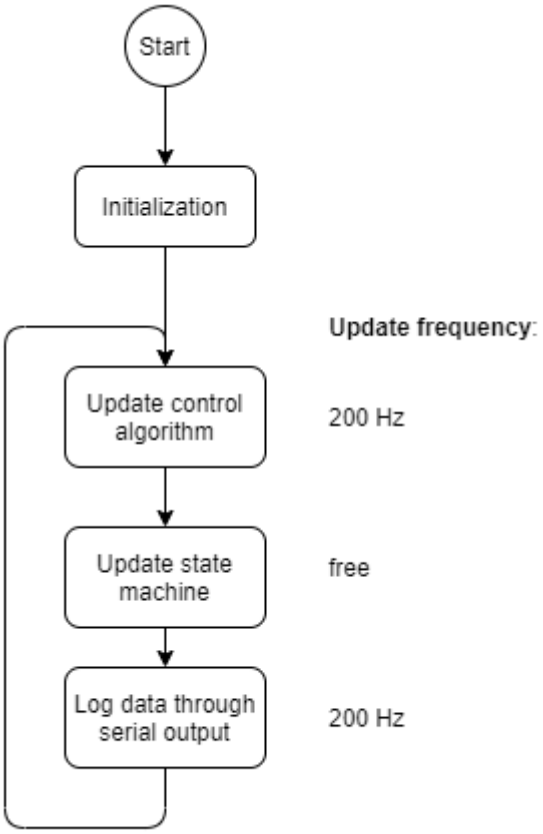


Figure 2.9 System control loop.

2.4.1 System state machine

The system state machine has four states: *Standby*, *Homing*, *Control* and *Stop*. The state transitions are shown in Figure 2.10.

At start up the state machine enters the *Standby* state. In this state the motor is disabled, and the cart can be moved around by hand. The state machine will transition if the start button is pushed. If the homing sequence has not been completed the homing sequence will start. If the homing sequence has been completed, the system enters the control state.

In the *Homing* state the homing sequence presented in section 2.4.2 is executed. The state machine will transition to the *Standby* state if the stop button is pushed or the homing sequence is completed.

In the *Control* state the control input calculated in the update control algorithm block is assigned to the stepper motor driver. The state machine will transition to the *Standby* state if the stop button is pushed or one of the end switches are activated. The state machine will enter the *Stop* state if the cart distance from the centre of the track is larger than a limit.

In the *Stop* state the motor is stopped with the maximum acceleration setting in the stepper motor driver. This will be an acceleration around $26m/s^2$. The state machine will transition to the *Standby* state if the stop button is pushed or one of the end switches are active or if the system has been in the *Stop* state for more than 1 second.

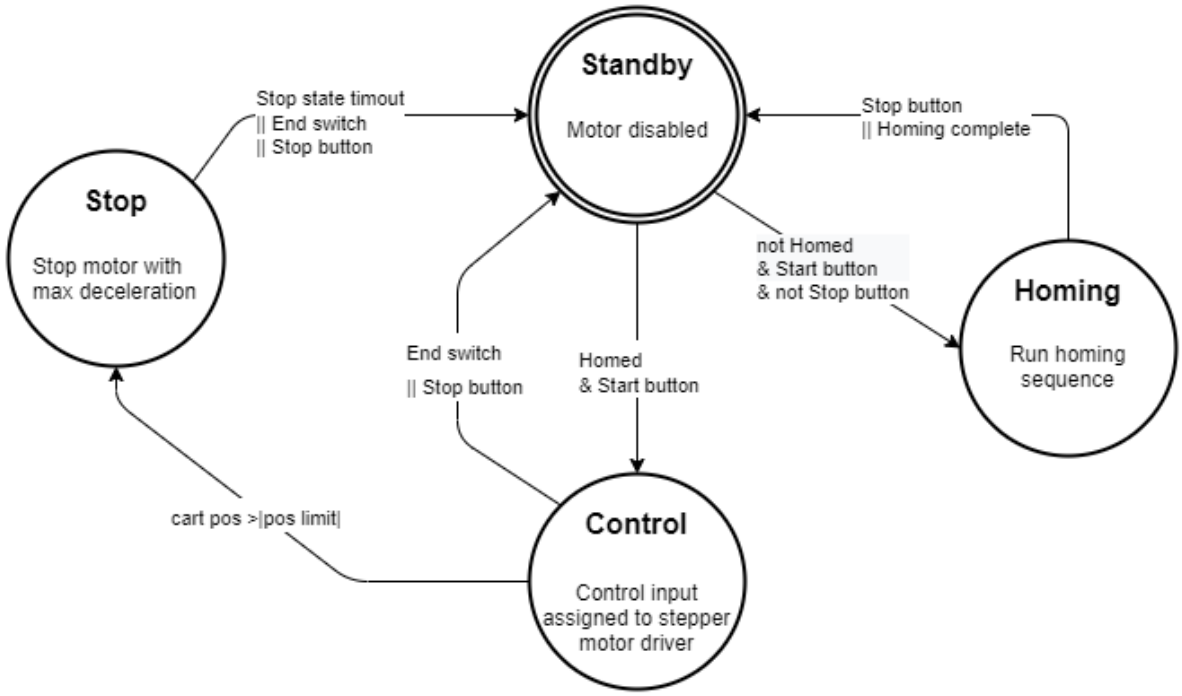


Figure 2.10 System state machine.

From the above it is clear that there are two ways of stopping the system if the cart approaches the end of the track. There are two reasons for this. Firstly, using both an end switch and a position limit gives redundancy. Secondly, it was found that only disabling the motor does not reduce the cart velocity quickly. This means that the cart could hit end of the track with a high velocity. The cart velocity is reduced quicker by using the maximum acceleration of the stepper motor driver. The system is therefore set up with the position limit being 43 cm from the middle of the track. This means that it is triggered 4cm before the end switch. If the cart approaches the end of the track it will therefore first be decelerated, before the motor is disabled.

Because of track length necessary to stop the cart at high velocity, the maximum velocity is set to 1.8 m/s. This is slightly less that the planned maximum velocity of 2.0 m/s but is still considered acceptable. The cart will in some cases still hit the end of the track, but with a significantly lower velocity. Ideally the cart should have been stopped before it can hit the end, but this would mean a significant reduction of either the maximum velocity or the usable track length. It has been tested to run the cart at full speed to the end of the track, and it does not damage any parts in the system, but it could create damage if done repetitively. There is also a risk of pinching for example a finger between the cart and one of the housings at the end, and this must be taken to account when using the model.

2.4.2 Homing procedure

The homing procedure is used to find a reference to the measurements from the incremental encoder. A block diagram of the homing procedure is given in Figure 2.11.

At the start the current pendulum angle is set to π. At start of the procedure the pendulum should be hanging straight down. This means that the pendulum angle will be 0 if the pendulum is rotated clockwise to the upright position.

The cart will then be moved left until the left end switch is triggered. If the left end switch is already triggered the cart will first be backed of the switch. The position of the left end switch is stored. The cart is then moved right until the right end switch is triggered. The length between the switches is calculated and the current position is set to be half of this. The cart is then moved to position 0, which will be the middle of the track.

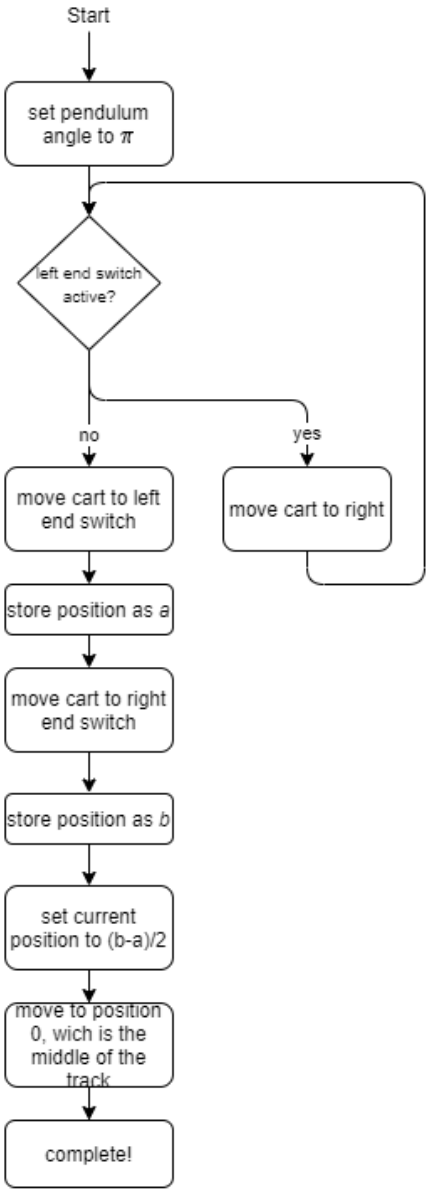


Figure 2.11 Homing procedure block diagram.

2.4.3 Encoder interface

To read the signal from the encoders the *Teensy4.x-Quad-Encoder-Library* provided by user *mjs513* on *github.com* is used [19]. This a library made for the decoding a quadrature such as the two-channel encoder output, using the *Teensy 4.1* microcontrollers hardware quadrature decoder. Using the library, the encoder position can be read as the number of steps moved from a reference position.

The interface is implemented as a *c++* class and one object is created for each encoder. The cart position and the pendulum angle is found by scaling the number of steps the encoder has moved with the resolution per step as found in (2.5) and (2.6). In addition, the class has functions for setting the current position and taking the derivative of the measurement. The derivation is implemented using finite differences and an average mean filter with a length of two. This is equal to what is presented in section 4.1.2.

2.4.4 Stepper motor driver interface

The *Trinamic TMC5160* stepper motor driver controls a stepper motor by alternating the current going through the spools in the stepper motor. The signal is current regulated, meaning that any back emf generated by the motor will not affect the torque output, if the back emf voltage is close to the supply voltage to the stepper driver. The current going through the motor spools is PWM[10] with a very high frequency and will be close to a sine wave as the motor is rotated. The stepper driver generates 256 micro steps, meaning that one step of the motor is divided into 256 steps. This means that the motor will rotate smoothly even at very low velocities. The velocity of the motor is directly controlled by the alternation of the current in the motor coils. The acceleration is controlled by gradually changing the frequency of the spool current. The generated current to the motor spools are from now referred to as the *chopper signal* [16] [14].

The *TMC5160* is mainly controlled by setting shift registers through SPI communication. In addition, toggling the *vcc_io* pin on the board for a few clock cycles will reset the entire stepper motor driver. The *DRV_enn* activates or deactivates the stepper driver. When disabled, the motor can move freely. However, when disabled, the *chopper signal* setting cannot change. This means that if the cart is moving at for example 1m/s, and then disabled through the *DRV_enn* pin, any reactivation of the *DRV_enn* pin will cause the signal to the motor coils to instantly be equal to the *chopper signal* for a cart velocity of 1 m/s. This means that the acceleration theoretically is infinitely, and the motor cannot generate enough force to start. This means that after the motor has been disabled, the stepper motor must be reinitialized as described in section 2.4.4.2.

2.4.4.1 SPI communication

The SPI communication to and from the *TMC5160* uses a 40-bit long datagram as shown in Table 2.6. When sending data to the *TMC5160* the first bit indicates a read (0) or write (1) command, the next 7 bits defines the register address to read from/write too and the last 32-bits is the data. If a read command is issued dummy data is used. The returned datagram consists of 8 status bits and 32-bits, that is the current data in the chips shift register. Reading a value from

[10] PWM: pulse width modulation

a *TMC5160* register is therefore a two-step procedure. First a read command is issued to the chip. This will load the data in the chosen register to the shift register. At the next datagram transaction this data is sent out from the chip. In this second transaction the command and data value does not matter. [16]

Table 2.6 TMC5160 SPI datagram structure.

SPI Datagram Structure			
Bit no:	39	38 … 32	31 … 0
To *TMC5160:*	R/WR	Register add.	data
From *TMC5160:*	status		data

2.4.4.2 Stepper driver configuration and initialization

The initialization of the stepper driver starts with switching of the *vcc_io* pin for $2ms$, before switching it back on. This will reset the whole driver, including setting the *chopper signal* to zero. Then all necessary shift registers will be set to the values corresponding to the configurations bellow. Finally, the motor is enabled [16].

The *SpreadCycle* chopper mode is enabled. In this mode the stepper driver automatically optimize the generated of *chopper signal* to fit the stepper motor [16].

The phase output current is set to 2.7A while the motor is running and to 1.6A if the motor is standing still. This is slightly below the motors nominal phase current of 2.8A. The current was reduced because the motor will get hot if it is running at low velocity for longer period. Reducing the phase current reduce the heat generation in the motor [14].

2.4.4.3 Motor control

The *TMC5160* has some options of how the position, velocity and the acceleration of motor is controlled. This is referred to as *ramp modes*. In this project a linear ramp is used, ehen the acceleration, maximum velocity and direction of acceleration is set. The motor will then accelerate with the set acceleration, in the chosen direction, until the maximum velocity is reached [16]. This is chosen because the acceleration can be controlled directly, meaning that there is a linear relation between the control input and the actual acceleration of the cart if the velocity is below the chosen maximum. This simplifies the mathematical modelling of the system.

Changing the control input to the system consist of three steps. First the *ramp mode* register is set to choose the direction. Then write the register value for the maximum velocity is set. Finally, the acceleration register is set. In the control state the maximum velocity is always the maximum velocity for the system, which is 1.8m/s, and only the acceleration changed. In the homing procedure, the acceleration is set to the maximum acceleration for the system, which is $20m/s^2$, and only the velocity is changed.

The setting of the maximum velocity register (VMAX) and the acceleration (AMAX) are found by a scaling of the velocity and acceleration in microsteps, given by [16]

$$VMAX = \frac{vel_{\mu steps} * f_{clk}}{2^{24}} \quad (2.8)$$

and

$$AMAX = \frac{acc_{\mu steps} * f_{clk}^2}{2^{41}}. \quad (2.9)$$

f_{clk} is here the frequency of the clock signal to the stepper driver, which is $15MHz$. The velocity in microsteps is found by

$$vel_{\mu steps} = \frac{vel * nsteps * n\mu steps}{pulley_c} \quad (2.10)$$

and the acceleration in microsteps is found by

$$acc_{\mu steps} = \frac{acc * nsteps * n\mu steps}{pulley_c}. \quad (2.11)$$

$nsteps$ is here the motors number of steps for a full rotation (200), $n\mu steps$ is the number of microsteps per motor step, $pulley_c$ is the circumference of the timing belt pulley ($0.16m$), vel is the cart velocity $[m/s]$ and acc is the cart acceleration $[m/s^2]$.

2.4.4.4 Read motor velocity

Since the motor velocity is controlled directly by the stepper motor the velocity of the cart can be found by reading the setting of the generated chopper signal that controls the motor. This value is stored in the VACTUAL register in the *TMC5160.* The register value can be scaled to a velocity in microsteps by

$$vel_{\mu steps} = \frac{VACTUAL * f_{clk}}{2^{24}} \quad (2.12)$$

and the cart velocity is found by

$$vel = \frac{vel_{\mu steps} * pulley_c}{nsteps * n\mu steps}. \quad (2.13)$$

2.4.5 File structure

The control system consists of the files described in Table 2.7.

Table 2.7 Control system file structure.

File structure	
CartPendulum.ino	This is the main Arduino file and contains the code for the control loop, state machine, homing procedure, logging, and the control algorithm.
pin_config.h	Contains constants for the connections to the microcontroller
Global_definitions.h	Contains global definitions for the system such as for setting the maximum acceleration and velocity, and the track length.
EncoderInterface.cpp *EncoderInterface.h*	C++ class for interface to the encoders.
MotorInterface.cpp *MotorInterface.h*	C++ class for interface to the *TMC5160* stepper driver.
estimation.h *estimation.cpp* *rtwtypes.h*	Contains c++ code generated in MATLAB/Simulink for the control algorithm described in chapter 4. For future use with a different control algorithm these can be removed.

3 MATHEMATICAL MODEL

There are two common approaches for modelling of mechanical systems. One method is based on using forces in the system, often referred to as Newtonian mechanics. The other is to use energy-based methods such as Lagrangian mechanics. From [7] and [20] where an approach based on Newtonian mechanics is used, and [6] where an Lagrangian mechanics approach is used, it is seen that both approaches will give a similar model, but with some variations due to differences in the representation of the system.

The Newtonian mechanics approach is based on utilizing Newtons three laws, to set up dynamic equations for the individual parts in the system, based on the forces acting on the parts. This is normally done with reference to a cartesian coordinate system. The individual parts will be affected by constraint forces, in this case from the fact that the cart and pendulum is connected at the hinge point. Trough manipulation and substitution of the dynamic equations for the individual parts, such as done in [20], the constraint forces can be removed and the equation of motions for the complete system is obtained.

With the Lagrangian approach the dynamical equations are derived using energy considerations described in general coordinates, that can be chosen freely, but must be independent. As this method is energy-based, it does not concern forces in the system. This means that we will not have to deal with the constraint forces between the cart and the pendulum. This makes the approach less tedious for a system such as the IPC [21, pp 237-257].

Based on this, the mathematical model will be derived using Lagrangian mechanics.

3.1 Lagrangian mechanics

Lagrangian mechanics is a formulation of classical mechanics and is based on the principle of stationary action, also known as the principle of least action [20p 30]. The principle states that for a system consisting of point masses, the systems trajectory from time a to time b is such that the action function

$$S(q) = \int_a^b L(q, \dot{q}, t)\, dt \tag{3.1}$$

is stationary, with respect to variations in the path. Here L is the Lagrangian function, given by the difference between the kinetic energy, T and the potential energy V,

$$L(q, \dot{q}, t) = T(q, \dot{q}, t) - V(q, \dot{q}, t) \tag{3.2}$$

and $q = [q1, \dots, qn]$ are the generalized coordinates and $\dot{q} = [\dot{q}_1, \dots, \dot{q}_n]$ their time derivatives [22, pp23-24].

From the calculus of variations, we have that for a stationary value of the action function, the Lagrange equations for a generalized system, including any external forces acting on the system, can be written as

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}} - \frac{\partial L}{\partial q} = Q, \tag{3.3}$$

where Q is here any external forces acting on the system [23p. 45]. This means that by expressing the kinetic and potential energy in the system in terms of the generalized coordinates, the Euler-Lagrange equation can be utilized to derive the systems equations of motions.

3.2 Derivation of equations of motions

Figure 3.1 shows a sketch describing the system and its variables. The generalized coordinates for the system are the carts position on the rail, $x[m]$, and the pendulum angle, $\theta[rad]$. The cart has mas $M[kg]$. The pendulum is a linear rod with mass $m[kg]$ and with $l[m]$ as the distance from the hinge point to the pendulum centre of gravity. In addition, we have $I[kgm^2]$ as the moment of inertia for the pendulum around the pendulum centre of gravity, with $2l$ as the pendulum length, this is given as

$$I = \frac{1}{12}m(2l)^2 = \frac{1}{3}ml^2, \tag{3.4}$$

for a linear rod [23 p. 64].

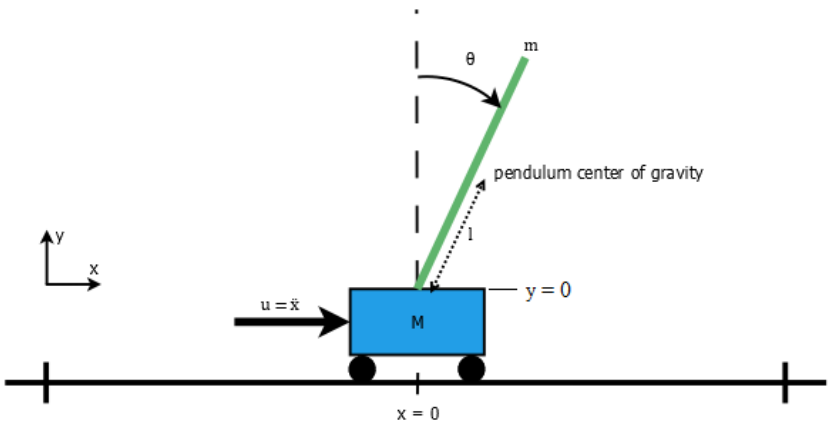


Figure 3.1 IPC figure.

We start by noting that because of the stepping characteristics of a stepper motor, the control input for the system will be the cart acceleration. This assumes that the motor always has enough torque to achieve the desired input acceleration, independent of any force from the swinging pendulum acting on the cart, friction or any other force acting on the system. This is a fair assumption given that the chosen motor is strong enough and the acceleration is limited to achievable values. Based on this we can state that system of equation for the *x* coordinate is fully given by

$$\ddot{x} = u. \tag{3.5}$$

This means that only the equations of motions for the θ coordinate needs to derived using the Euler-Lagrange equation.

Since the cart only can move in the *x* direction, we start by recognizing the kinetic energy for the cart as [23p. 41]

$$T_c = \frac{1}{2} M\dot{x}^2. \tag{3.6}$$

The kinetic energy for the pendulum can be described with two components, the kinetic energy from linear motion of the rod as a point mass at the pendulum centre of gravity, and the kinetic energy from rotational motion around the hinge point, generated by the moment of inertia around the pendulum centre of gravity [23p. 41]. This gives

$$T_p = \frac{1}{2} m v_p^2 + \frac{1}{2} I\dot{\theta}^2. \tag{3.7}$$

Here $v_p[m/s]$ is the is the pendulum centre of gravity velocity, consisting of components in both the x and y direction. By setting the y coordinate to zero at the pendulum hinge point and, the position of the pendulum centre of gravity (COG) is given by

$$\begin{bmatrix} x_p \\ y_p \end{bmatrix} = \begin{bmatrix} x + lsin(\theta) \\ lcos(\theta) \end{bmatrix}. \tag{3.8}$$

Differentiating (3.8) gives the velocity components as

$$\begin{bmatrix} \dot{x}_p \\ \dot{y}_p \end{bmatrix} = \begin{bmatrix} \dot{x} + lcos(\theta)\dot{\theta} \\ -lsin(\theta)\dot{\theta} \end{bmatrix}. \tag{3.9}$$

From this we can write

$$\begin{aligned} v_p^2 &= \left(\dot{x} + lcos(\theta)\dot{\theta}\right)^2 + \left(-lsin(\theta)\dot{\theta}\right)^2 \\ &= \dot{x}^2 + 2lcos(\theta)\dot{\theta}\dot{x} + l^2\, cos^2(\theta)\, \dot{\theta}^2 + l^2\, sin^2(\theta)\dot{\theta}^2 \\ &= \dot{x}^2 + 2lcos(\theta)\dot{\theta}\dot{x} + l^2\dot{\theta}^2 \end{aligned} \tag{3.10}$$

This give the kinetic energy for the pendulum as

$$T_p = \frac{1}{2} m\dot{x}^2 + mlcos(\theta)\dot{\theta}\dot{x} + \frac{1}{2} ml^2\dot{\theta}^2 + \frac{1}{2} I\dot{\theta}^2. \tag{3.11}$$

With the inertia for a pendulum consisting of a linear rod, given by (3.4), we could combine the two last terms in (3.11). However we will keep them separate, as this makes it possible to modify the final model, in case the pendulum is changed to a pendulum consisting of a linear rod and a weight, simply by updating the centre of gravity and recalculate the moment of inertia around the COG.

We then have the kinetic energy for the whole system as

$$T = T_c + T_p = \frac{1}{2}M\dot{x}^2 + \frac{1}{2}m\dot{x}^2 + mlcos(\theta)\dot{\theta}\dot{x} + \frac{1}{2}ml^2\dot{\theta}^2 + \frac{1}{2}I\dot{\theta}^2. \quad (3.12)$$

The potential energy in the system is given by the gravitational force acting on the system components. Because the cart only can move in the x direction, which is perpendicular to the gravitational force, the cart cannot have any change in potential energy. The potential energy of the cart is therefor set to

$$V_c = 0. \quad (3.13)$$

The potential energy is given by the height of the pendulum centre of gravity. We use the hinge point as a potential reference and give the pendulum potential energy as

$$V_p = mglcos(\theta). \quad (3.14)$$

This give the total potential energy as

$$V = V_c + V_p = mglcos(\theta). \quad (3.15)$$

From (3.12) and (3.15) we get the Lagrangian as

$$\begin{aligned} L &= T - V \\ &= \frac{1}{2}M\dot{x}^2 + \frac{1}{2}m\dot{x}^2 + mlcos(\theta)\dot{\theta}\dot{x} + \frac{1}{2}ml^2\dot{\theta}^2 + \frac{1}{2}I\dot{\theta}^2 - mglcos(\theta). \end{aligned} \quad (3.16)$$

We then insert the Lagrangian into the Euler-Lagrange equation and get

$$\begin{aligned} Q &= \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}}\right) - \frac{\partial L}{\partial \theta} \\ &= \frac{d}{dt}\left(mlcos(\theta)\dot{x} + ml^2\dot{\theta} + I\dot{\theta}\right) - \left(-ml\dot{\theta}\, sin(\theta)\, \dot{x} + mglsin(\theta)\right) \\ &= mlcos(\theta)\ddot{x} - ml\dot{\theta}sin(\theta)\dot{x} + ml^2\ddot{\theta} + I\ddot{\theta} + ml\dot{\theta}\, sin(\theta)\, \dot{x} - mglsin(\theta) \\ &= (ml^2 + I)\ddot{\theta} + mlcos(\theta)\ddot{x} - mglsin(\theta). \end{aligned} \quad (3.17)$$

3.3 External forces

External forces acting on the cart will be the input force from the motor and belt system, friction forces from the railing, and any force from air resistance/drag. As stated in the start of section 3.2 the input is controlling the acceleration in the *x* coordinate directly, as shown in Figure 3.1. This can be seen as that the motor will input the necessary force to overcome any other force in the system and the force necessary to get the input acceleration. Further we must consider that any force acting in the *x* coordinate, could affect the θ coordinate. From equation (3.17) we see the term $mlcos(\theta)\ddot{x}$ will take care of the coupling between the coordinates, as this describes how the θ coordinate will react to an input affecting the acceleration in the *x*

coordinate. This means that we will only have to consider the external forces acting on the θ coordinate.

The external forces acting on the pendulum are friction in the ball bearings at the hinge point between the cart and the pendulum, and air resistance/drag acting on the pendulum. We will continue the discussion of these forces in the following sections. For now we will note that they act on the θ coordinate and that they are a function of the pendulum angular velocity, $\dot{\theta}$, but will act in the opposite direction, and therefore will be referenced as $-F_{ext}(\dot{\theta})$.

By inserting the external forces into (3.17) we get

$$(ml^2 + I)\ddot{\theta} + mlcos(\theta)\ddot{x} - mglsin(\theta) = -F_{ext}(\dot{\theta}). \tag{3.18}$$

With some manipulation we get the equation for the angular acceleration of the pendulum

$$\ddot{\theta} = \frac{mglsin(\theta)}{ml^2 + I} - \frac{mlcos(\theta)}{ml^2 + I}\ddot{x} - \frac{F_{ext}(\dot{\theta})}{ml^2 + I}. \tag{3.19}$$

We now have the full mathematical model for the IPC system from the two coupled second order differential equations (3.1) and (3.19). By introducing the state vector

$$\underline{x} = \begin{bmatrix} x \\ \dot{x} \\ \theta \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}, \tag{3.20}$$

we can write the model as a set of the following first order differential equations

$$\dot{\underline{x}} = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} = \begin{bmatrix} x_2 \\ u \\ x_4 \\ \frac{mglsin(x_3)}{ml^2 + I} - \frac{mlcos(x_3)}{ml^2 + I}u - \frac{F_{ext}(x_4)}{ml^2 + I} \end{bmatrix} \tag{3.21}$$

When deriving a mathematical model, it is necessary to consider how accurate it needs to be. For the IPC this will mainly be how accurate we model the friction forces in the system. Is a simple linear model consisting only of a linear damping term good enough, or do we benefit from having a more accurate and complex model?

A very accurate model will often include an increased number of terms and in many cases, nonlinear and discontinuous terms. This makes the job of determining parameters more tedious and we run the risk of overfitting the model to test data. The behaviour of the physical system may vary, for example because of temperature change or wear and tear. This can leave us with a model that is less accurate than we expect. A model containing nonlinear and discontinuous terms, can create problems when we need the derivative of the model.

The model we derive here will be used for creating a controller for swinging the pendulum from a downward position, to an upright position, and for creating a LQR controller for stopping and balancing the pendulum as it is swung up close to the upright position. The model will also be used for simulation and tuning of the controllers in MATLAB/Simulink.

The LQR controller described in section 4.3.3 is based on a model linearized around the upright equilibrium; hence it needs a linear model for deriving the controller. The linear model primarily needs to be accurate at low velocities to achieve good balancing, and slightly higher velocities for catching the pendulum as it is swung up.

The swing up controller described in section 4.2 does not utilize knowledge about friction in the system for deriving the control law, but it relies heavily of parameter tuning after testing. This means that an accurate model for simulation will be beneficial, as parameters can be tuned in Simulink before it's implemented on the real system. The pendulum will be swinging at higher velocities. This means that the model used for simulation should be accurate both at high and low angular velocities.

Based on the discussion above and the further discussion and elaboration below, it was decided to make two models. One model where the external forces are modelled with a linear damping term, to be used for deriving the LQR controller. And one model used for simulation which model the external forces with terms for air drag, linear damping, and Coulomb friction. We will refer to these models as the linear damping model and the simulation model in the following sections.

We will start with looking at linear damping and presenting the linear damping model, as this will show the limitations of this model. Further we will look at including air drag and coulomb friction in the simulation model.

3.3.1 Modelling linear Damping

In mechanical systems damping is an influence on a system by reducing its oscillation. In our case: the pendulum oscillation. This can be modelled as a force proportional to motion, acting in the opposite direction. This is analog to what is referred to as the viscous friction model [24]. Since we are dealing with the angular rotation of the pendulum, we have the damping force

$$F_d(t) = k_d\dot{\theta}(t) \tag{3.22}$$

Here k_d is the damping coefficient, with unit [Ns/m]. Figure 3.2 shows a simulation and measurement of the real system of free oscillation of the pendulum, starting at three different angles. The simulation use the model in (3.21) with $F_{ext} = F_d$ and $k_d = 0.001$.

The linear model is a good representation of the system, but it can be seen in Figure 3.2 that is has some shortcomings. First, we see in plot *b)* and *c)* that the model fails to completely stop the oscillations. This indicates that there is friction in the system that isn't velocity dependent or is only present at low velocities. Secondly, we see in the middle part of plot *a)* and first part of plot *b)* that there is too much damping at the medium velocities. Further we see some difference in the phase between the real system and the simulation in plot *a)*. We don't see the same difference in plot *b)* and *c)*. This indicates that the system has some influence that is mainly present at higher velocities, indicating a higher order expression.

k_d is chosen to make the model fit at low oscillations and angular velocities, as this is most relevant for the LQR controller. For reference, the maximum angular velocity of the first swing in Figure 3.2 plot *b)* is 2 rad/s. In section 4.2.1 the swing up controller is tuned to give an angular velocity of less than 2 rad/s as the pendulum is coming to the upright position. This means that we mainly consider the accuracy in plot *b)* and *c)* for the linear damping model, as these cover the expected pendulum angular velocities while the LQR controller is active. The figure shows that the chosen k_d is a good compromise between the two.

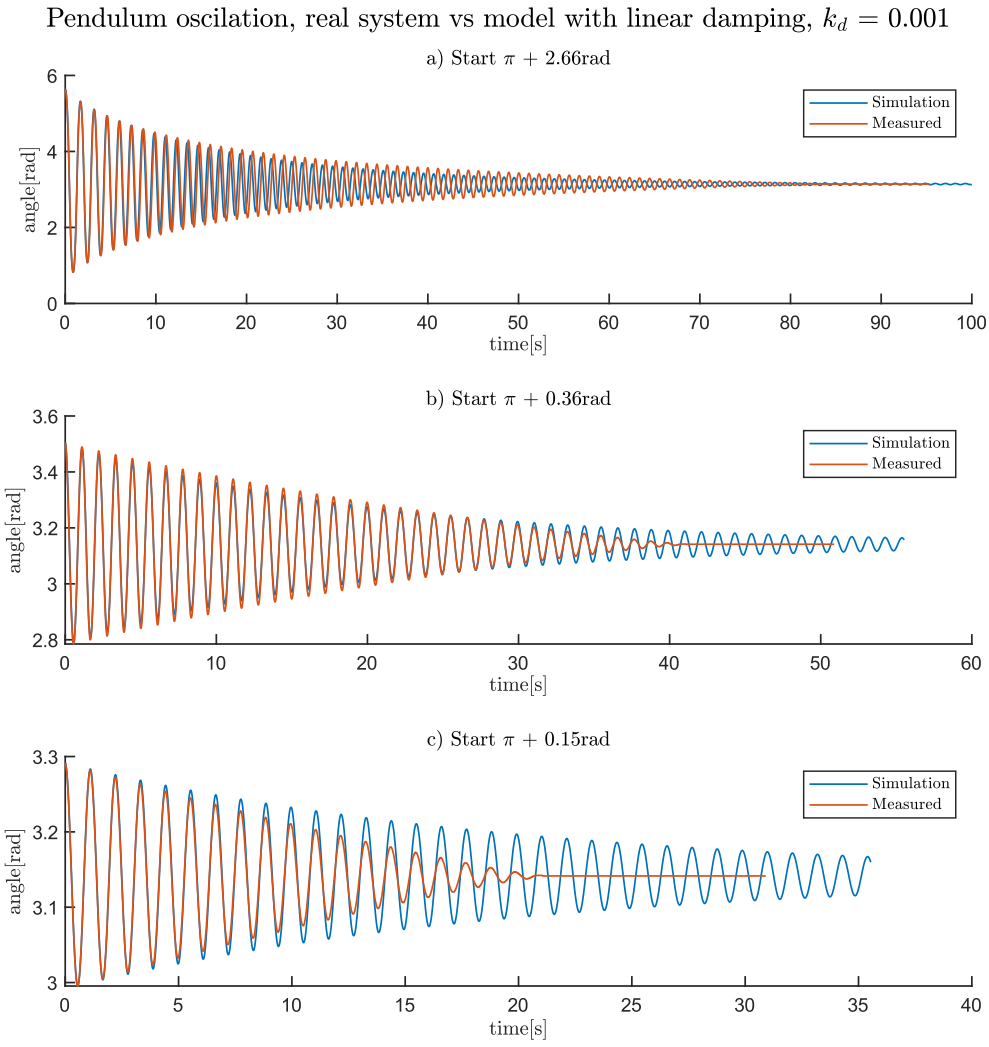


Figure 3.2 Free oscillation of the pendulum for the real system and model with linear damping. k_d = 0.001. The figure shows three graphs with oscillations starting from different angles. The cart is in a fixed position.

3.3.2 Air drag

In Fluid Mechanics drag is known as the force produced on a body as it passes through a fluid. The drag force can be modelled as proportional to the flow velocity at low Reynolds number and proportional to the square of the flow velocity at higher Reynolds number. Reynolds number is given found by

$$Re = \frac{\rho d v}{\mu}, \tag{3.23}$$

where ρ is the fluid density, μ is the fluid viscosity, d is a characteristic linear dimension and v is the flow velocity [25 pp. 21-33].

In the case for the IPC we consider all parameters in (3.23), except the velocity, as constants. Based on the discussion at the end of section 3.3.1, this indicates that the pendulum is rotating at an angular velocity high enough for the drag force to be proportional to the square of the velocity. And a term for this should be added to the model. The linear damping term will model the drag force in the cases where the velocity is enough for modelling the force proportional to the velocity.

Since the pendulum is rotating the linear velocity will vary through the length of the pendulum, but it will be proportional to the angular velocity at all points. This means thar we can model the force from the air drag as proportional to the squared angular velocity, giving

$$F_{dr}(t) = k_{dr} * \dot{\theta}(t)^2 \tag{3.24}$$

where k_{dr} is a constant with unit $[Ns^2/m^2]$, which will be found by parameterization/fitting to test measurement of the real pendulum.

3.3.3 Coulomb friction

The coulomb sliding friction force is a force acting on an object that is sliding on a surface. The force will act in the opposite direction of the object's movement parallel to the sliding surface. The Coulomb friction is defined as

$$\mathcal{F}_c = \mu_c N \tag{3.25}$$

where μ_c is the sliding friction coefficient and $N[F]$ is the normal force in the contact. The Coulomb friction is often modelled as

$$F_C(t) = \begin{cases} \mathcal{F}_c sign(v(t)) & if\ v(t) > 0 \\ F_{app}(t) & if\ v(t) = 0\ and\ F_{app}(t) < \mathcal{F}_c \\ \mathcal{F}_c sign(F_{app}(t)) & if\ v(t) = 0\ and\ F_{app}(t) > \mathcal{F}_c \end{cases} \tag{3.26}$$

where $F_{app}(t)$ is the applied force to the object and $v(t)$ is the velocity of the object[24]. As we will not have knowledge about the applied force, we use the simplification

$$F_C(t) = \mathcal{F}_c sign(v(t)) \tag{3.27}$$

This model can cause problems in simulations, as the sign function is discontinuous at zero velocity. We will therefore replace the *sign* function with a *tanh* function to get a continuous function around zero velocity. And by replacing the velocity with angular velocity we have the Coulomb friction model we will use as

$$F_C(t) = \mathcal{F}_c tanh\left(k_t \dot{\theta}(t)\right) \quad (3.28)$$

k_t is here a scaling constant to increase the steepness of the *tanh* function. Figure 3.3 shows the difference of modelling the Coulomb friction with a *sign* function, as in (3.27), and with a *tanh* function, as in (3.28), with differ values for k_t. Initially $k_t = 200$ was used, but this resulted in poor performance for the *extended Kalman filter* because of the steeper gradient around 0 angular velocity. This was improved by reducing k_t to 100. It wasn't noticed any change in the simulation model because off this change.

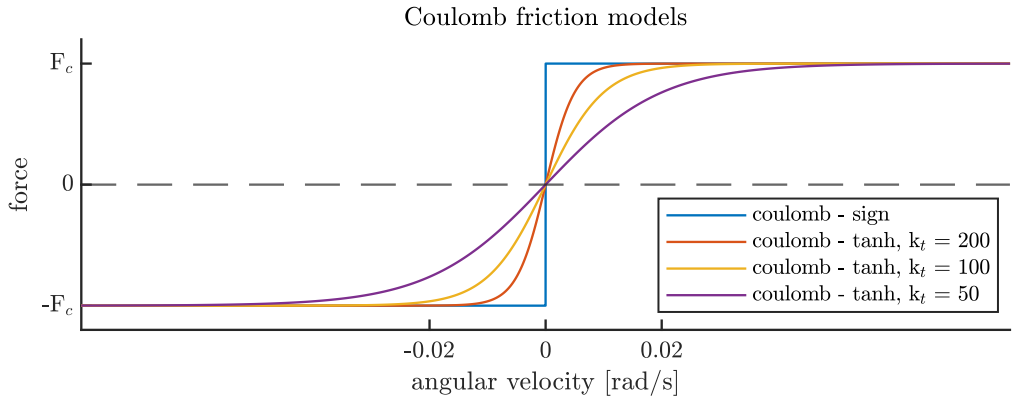


Figure 3.3 Coulomb friction models.

Figure 3.4 shows the result of combining coulomb friction and linear damping around zero angular velocity.

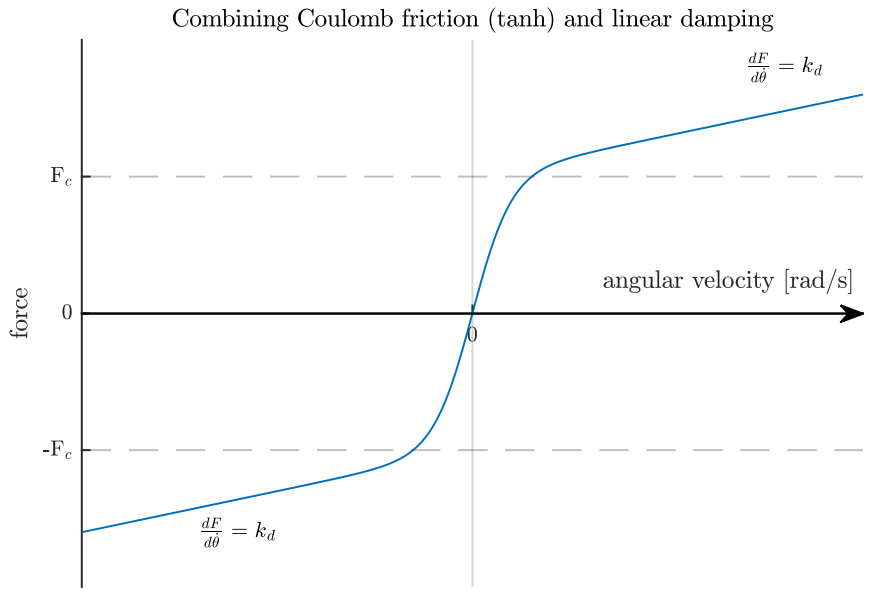


Figure 3.4 Combining Coulomb friction (tanh) with linear damping around zero angular velocity. The damping coefficient, k_d, is chozen for good visualization and does not correspond with the chosen one for the model, as this would give a very flat slope angle.

3.3.4 Static friction

Static friction is a force acting on objects in contact with a surface when the velocity of the object is zero. The static friction force is acting in opposite direction as any force acting on the object parallel to the sliding surface [24]. Hence the Static friction force can be seen as a force that must be overcome before the object starts moving.

The static friction force has the same problem as the Coulomb friction, as it is discontinuous at zero velocity. It was considered to model the static friction with

$$F_s(t) = (\mathcal{F}_c - \mathcal{F}_s)e^{\left(-\frac{|v(t)|}{v_s}\right)} tanh\,(v(t)), \tag{3.29}$$

where $F_s(t)[N]$ is the resulting static friction, $\mathcal{F}_s$[N] is the static friction and v_s is a coefficient for setting the decay of the exponential term. The result would be a model that has static friction at low velocity, but no static friction at zero velocity. The combination of linear damping, Coulomb friction and static friction as modelled by (3.24),(3.28) and (3.29) would lead to a common simplification of the *Stribeck friction model[24].*

It was considered to include static friction in the simulation model, however during parameterization and testing including static friction in the model didn't result in any noticeable improvement. It was therefore determined to consider the static friction in the system as negligible and it is therefore excluded from the model.

3.4 Parameter estimation

The parameters used in the system model is found mainly by fitting a simulation of the pendulum oscillations to corresponding measured oscillations of the real system.

The pendulum weight, m, is found by weighing of the pendulum. The distance from the pendulum centre of gravity to the hinge point, named l is approximated by disconnecting the pendulum and finding the point where it can be balanced horizontally. This gives l as the distance between the hinge point and the balance point. As the part mounting the pendulum will affect the balance point slightly, it was expected to get some errors with this measurement. The length has been adjusted to make the oscillation of the model and the pendulum match at oscillations at low magnitude, corresponding to the magnitudes seen in *plot b)* in Figure 3.2. This is based on Christiaan Huygens's law, that states that for low oscillations the period of a friction less pendulum can be found by

$$T_0 = 2\pi\sqrt{\frac{l}{g}}, \tag{3.30}$$

meaning that only dimensions affecting the period is the gravity and the distance between the hinge point and the pendulum centre of gravity [26]. The IPC pendulum is not friction less, but we assume it to be close enough for (3.30) to be fairly accurate.

The parameters for the friction modelling was first found for the linear damping model by adjusting k_d until the model is as described in section 3.3.1.

For the Coulomb friction we have that $\mathcal{F}_c = \mu_c N$. The normal force, N, will be the force from the shaft connected to the pendulum, acting normally on the ball bearings. This force depends on the weight of the pendulum and dynamics depending on the cart movement and pendulum swing. This means that the normal force will by time varying, but for the sake of simplification it is considered constant. N and μ_c will not be estimated/calculated. As both are considered constant, we will rather estimate a constant value for $\mathcal{F}_c$.

Estimation of $\mathcal{F}_c$ is done by fitting the model to measured pendulum oscillations at medium to low amplitude, where the air drag was considered neglectable. This was done by using the linear damping model as a starting point. The Coulomb friction model in (3.27) was added and $\mathcal{F}_c$ gradually increased and k_d gradually decreased. The result is shown in Figure 3.5 illustrates that the combination of linear damping and coulomb friction model gives a good representation at low pendulum amplitudes and angle velocities, corresponding to plot *b)* and *c)*. The model lacks accuracy at the higher angular velocities, corresponding to plot *a)*.

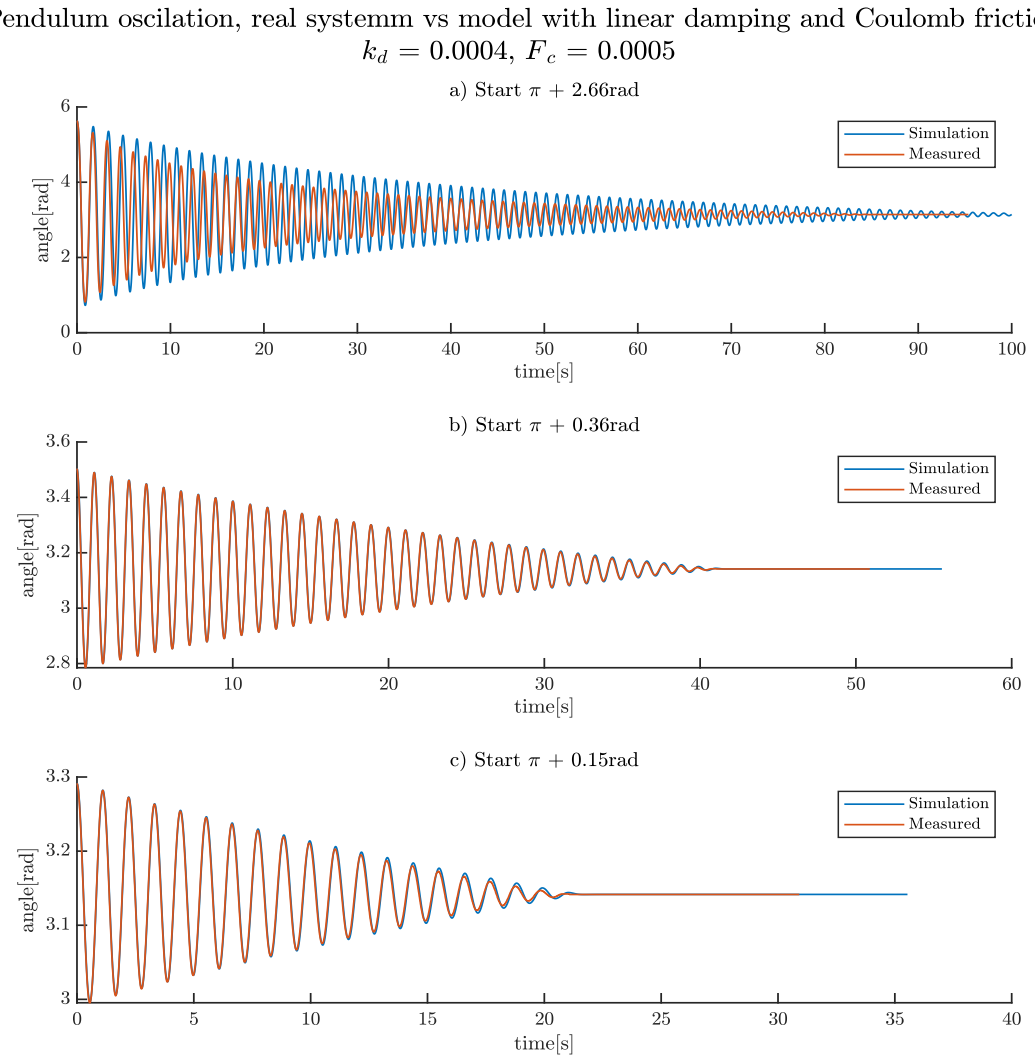


Figure 3.5 Free oscillation of the pendulum for the real system and model with linear damping and Coulomb friction. k_d = 0.0004 and $\mathcal{F}_c$ = 0.0005. The figure shows three graphs with oscillations starting from different angles.

The air drag modelled with (3.24)was then included and k_{dr} gradually increased, together with some minor adjustments of k_d and $\mathcal{F}_c$. The final resulting oscillation for the simulation model is shown in Figure 3.6. The figure shows that the model fits the measured oscillations through the whole range of amplitude and angular velocities.

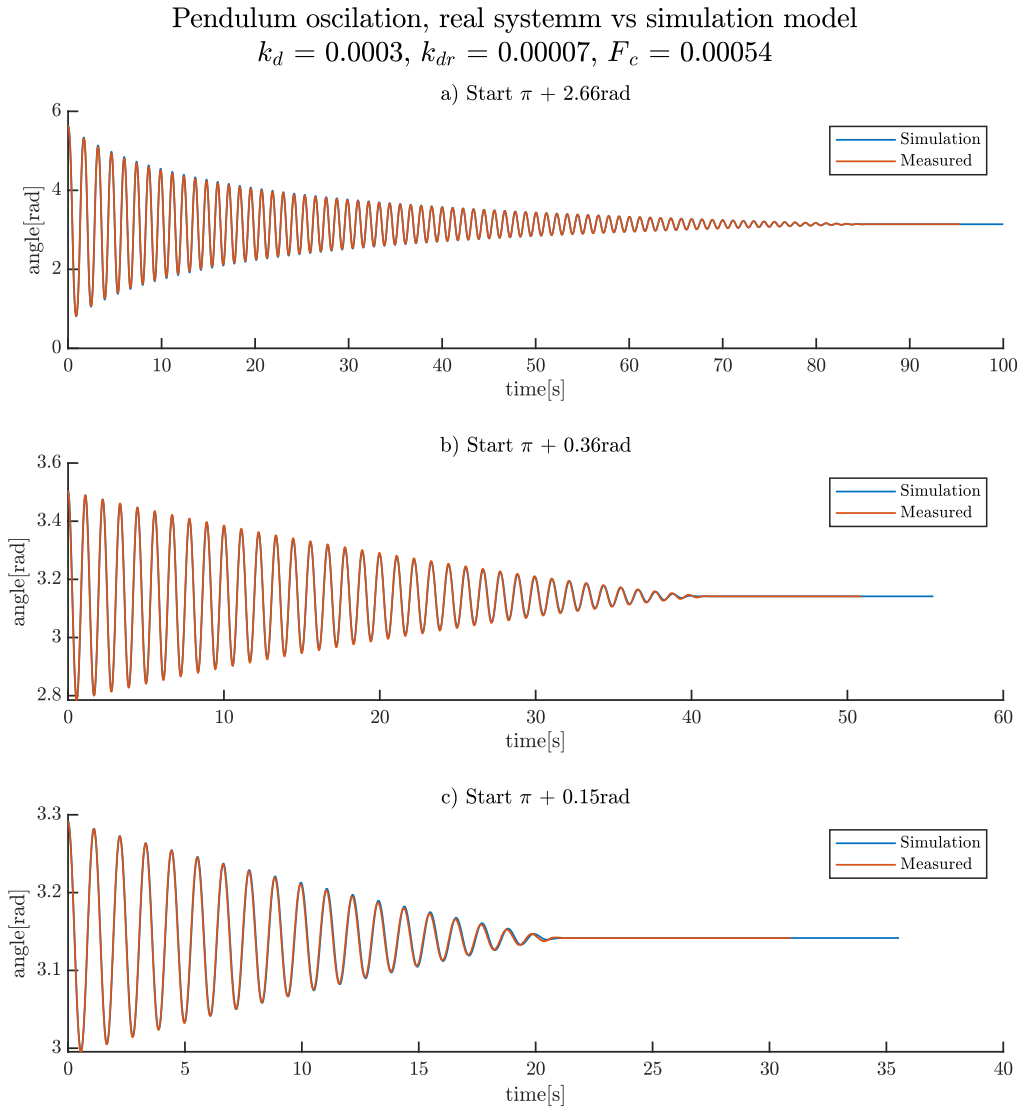


Figure 3.6 Free oscillation of the pendulum for the real system and simulation model. $k_d = 0.0003$ *and* $F_c = 0.00054$ *and* $d_{dr} = 0.00007$. *The figure shows three graphs with oscillations starting from different angles.*

3.5 Final models and parameters

We will now rewrite the model from (3.21) by inserting the damping term F_d from (3.22), the air drag term F_{dr} from (3.24) and the Coulomb friction from (3.28) into the term for the external forces. This gives the external forces in the system as

$$\begin{aligned} F_{ext}(x_4) &= F_d(x_4) + F_{dr}(x_4) + F_c(x_4) \\ &= k_d x_4 + F_{dr} x_4^2 + \mathcal{F}_c tanh(k_t x_4) \end{aligned} \tag{3.31}$$

And the complete model as

$$\begin{aligned} \dot{\underline{x}} &= \begin{bmatrix} x \\ \dot{x} \\ \theta \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} \\ &= \underbrace{\begin{bmatrix} x_2 \\ u \\ x_4 \\ \dfrac{mglsin(x_3)}{ml^2 + I} - \dfrac{mlcos(x_3)}{ml^2 + I} u - \dfrac{k_d}{ml^2 + I} x_4 - \dfrac{k_{dr}}{ml^2 + I} x_4^2 - \dfrac{\mathcal{F}_c tanh(k_t x_4)}{ml^2 + I} \end{bmatrix}}_{f(\underline{x},u)}. \end{aligned} \tag{3.32}$$

The system parameters are given in Table 3.1. The right-hand side of (3.32) is denoted f$(\underline{x}, u)$ for later reference.

Table 3.1 System and model parameters.

System parameters			
Symbol	**Description**		**Value**
$\boldsymbol{m}$	Pendulum mass		$0.14\ kg$
$\boldsymbol{l}$	Length from hinge point to pendulum centre of gravity		$0.228\ m$
$\boldsymbol{I}$	Pendulum Inertia around centre of gravity		$0.0025\ kg\ m^2$
$\boldsymbol{g}$	Gravitational acceleration		$9.81\ m/s^2$
$\boldsymbol{u_{max}}$	Max acceleration		$20\ m/s^2$
$\boldsymbol{x_{1\ max}, x_{1\ min}}$	Max and min allowed cart position with reference to track length		$0.43m, -0.43m$
$\boldsymbol{x_{2\ max}}$	Maximum cart velocity		$\pm 1.8\ m/s$
Friction parameters			
		Linear damping model	**Simulation model**
$\boldsymbol{k_d}$	Linear damping constant	$0.001\ Ns/m$	$0.0004\ Ns/m$
$\boldsymbol{k_{dr}}$	Air drag constant (quadratic damping)	0	$0.00007\ Ns^2/m^2$
$\boldsymbol{F_c}$	Coulomb constant force	0	$0.00054\ N$
$\boldsymbol{k_t}$	Constant in *tanh* function	0	100

3.6 Model verification

The accuracy of the model has been verified by applying an input to the real system and the mathematical model and comparing the cart position and pendulum angle. The applied input and results are shown in Figure 3.7, indicating a near perfect match.

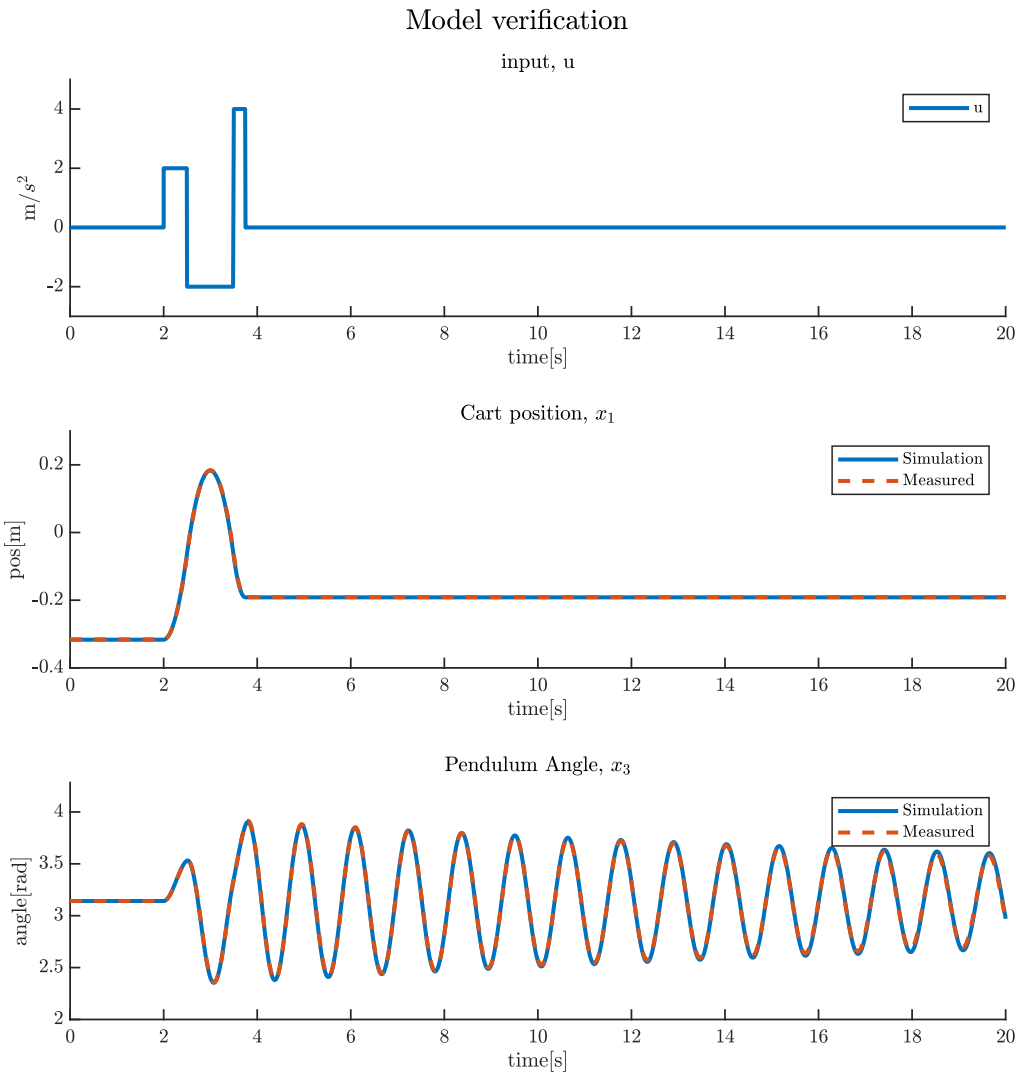


Figure 3.7 Verification of simulation model. An equal input is applied both to the real system and the mathematical model and the resulting cart position and pendulum angle is compared.

4 CONTROL ALGORITHM FOR THE IPC

We will now look at the control system for the IPC. We will divide the controller in two sub controllers, one for swinging the pendulum from a pendulum down position to a upright position, and one for balancing the pendulum in the upright position. These controllers will be separate, and the system will switch between the controllers depending on the pendulum angle.

Both controllers are based on feedback loop and will need the full state of the system, $\underline{x}$, for calculation of the control input. The system has encoders for measuring the cart position and pendulum angle, but the cart velocity and pendulum angle velocity will have to be estimated or obtained in other ways. We will therefore start this chapter with looking at state estimation before we continue with the swing up control and the *linear quadratic regulator* used to control the pendulum at the upright position.

The control system described here will run at a 200Hz update frequency. This choice is based on the quick response time of the system, for example as seen in Figure 3.6, where one oscillation takes around 1 second. We could probably have managed with a lower frequency than 200Hz, but this was chosen because it would most likely be fast enough, it was expected that we could estimate the missing parameters without much noise, and the microcontroller was expected to be quick enough. The later argument has been verified by measuring time for one update cycle, and the former argument will be verified in the section for state estimation.

In this section we will present multiple results from simulation in Simulink. Simulations are performed using the simulation model presented in section 3.5. Measurements from the encoders have been simulated using a *quantizer* block on the measured state. The *quantizer* block will discretize the output corresponding to the resolution the encoders give for the measurement, as given by (2.5) and (2.6) .

4.1 State estimation

We can measure the cart position and the pendulum angle directly using the encoders. For the cart velocity we have to options, we can estimate it, or we can use the velocity obtained from the stepper motor driver. It was decided to use the velocity from the stepper motor because this is based on the same data that the stepper motor uses for ramp generation for the motor current. This means that as long as the motor functions normally, this data is accurate and does not have any delays. Further the data contains little noise. This means that we can treat this velocity obtain from the stepper driver as a direct measurement. We can then write the measurement model for the system as

$$\underline{y} = C\underline{x} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \underline{x}. \qquad (4.1)$$

Since the cart position and pendulum angle is obtained from encoders the measurements will contain quantization noise. This will be described in the following section, before we continue with the estimation of the last state, x_4.

For the estimation of x_4 we will examine two options. First, we will use numerical differentiation to estimate the derivative of x_3 and use this as the estimate of x_4. The second option is to do the estimation with an *extended Kalman filter*.

4.1.1 Encoder quantization error

As stated, the cart position and pendulum angle are measured with encoders. Because of the resolution of the encoder, the encoder output will contain a quantization noise. This is illustrated in Figure 4.1, showing that the quantized measurement has a stepping shape, where the quantization step will have a height corresponding to the encoder resolution. In the figure the quantized relation is placed bellow the correct relation, but it could also be above, or somewhere in the middle. This depends on the encoder position at initialization and is unknown.

With the measurement resolutions of 0.2mm for the cart position and $7.85 * 10^{-4} rad$ for the pendulum angle, the quantization noise is considered to be insignificant for the position and angle measurement. However, it will affect the differentiation of angle, to get the angle velocity.

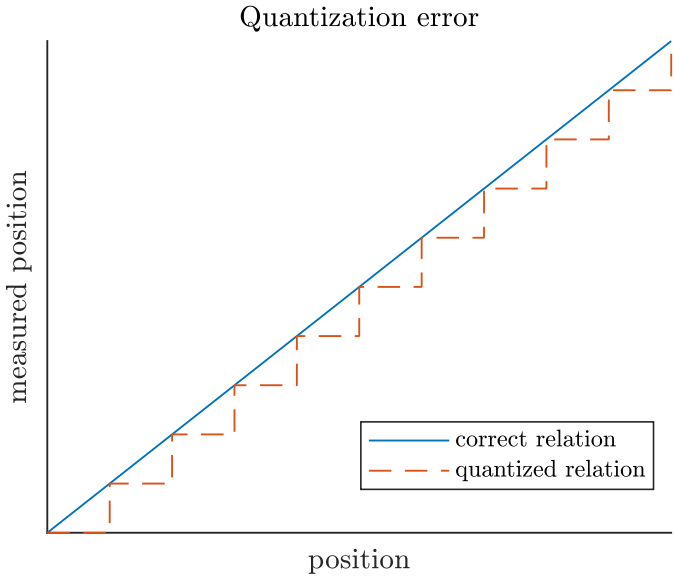


Figure 4.1 Illustration of error in a quantized measurement.

4.1.2 Numerical differentiation

The pendulum angular velocity is the derivative of the pendulum angle. A common way to approximate the derivative is to use *finite differences* and since calculations are needed in real time, we must use *backward difference* as only present and previous measurements are available. This is equal to what is referred to as frequency measurement in [27]. The approximation for the derivative of the pendulum angle is

$$\frac{d\theta(t)}{dt} = \lim_{\Delta t \to 0} \frac{\theta(t) - \theta(t - \Delta t)}{\Delta t} \approx \frac{\theta_k - \theta_{k-1}}{T_s} \approx \dot{\theta}_k. \tag{4.2}$$

Here T_s is the sampling interval, which will correspond to the control loop update frequency of 200Hz, and θ_k denotes the angle at time instance $k * T_s$. This approach can give problems with noisy measurements, as this can create big derivatives when the sampling time is short. This means that the numerical differentiation acts as a high pass filter and lets noise through. Increasing the sampling time or using a low pass filter will reduce this noise, but it will increase the phase delay of the estimate and should be used with care.

The encoders are considered to only contain quantization noise and (4.2) show that this quantization noise will be included in the numerical differentiation and the quantisation step size is dependent on T_s. To reduce the quantization step, we will use an average mean filter. This is chosen as it is analogue to increasing the sample time. We can then calculate the quantization step as

$$\delta\dot{\theta} = \frac{\delta\theta}{T_s * n} \tag{4.3}$$

where $\delta\theta = 8.85 * 10^{-4} rad$ is the quantisation step for the theta measurement and n is the filter length. Using a filter with length n=2 will then give a quantization step of $0.0785 \frac{rad}{s} \approx 4.5°/s$.

The quantization step size is considered to mainly be a problem when the pendulum is balance in the upright position. In this case the angular velocity will be close to zero, this means that the estimated angular velocity will alternate between 0 and the value of the quantization step size. The performance of the numerical differentiation will be further discussed in section 4.1.4 were it is compared with the estimate from the extended Kalman filter.

The numerical differentiation and the average mean filter will be implemented in Simulink as *finite impulse response* (FIR) filters.

4.1.3 Extended Kalman filter

The *Kalman filter* (KF) is a discrete time algorithm for estimating the full state of a processes based on a partial measurement of the state of a linear process model and the noise present in the system and measurements. The noise is considered zero mean gaussian. The Kalman filter is named after Rudolf E. Kálmán, one of the primary developers of the filter. The filter is used in a wide variety of applications both for estimating missing states and improving noisy measurements. Given a linear time invariant process and zero mean gaussian noise the Kalman filter is an optimal estimator [28 pp 1-5],[29 p 515].

The KF is a recursive algorithm using the following steps to estimate the optimal estimate. Prior to a new measurement the state is estimated, using the process model, along with an estimate of the uncertainty of the estimate. When a new measurement is available, the final estimation is given by a weighting between the a priori estimate and the measurements. The weighting, referred to as the Kalman gain, is calculated based un the uncertainties of the a priori estimate and the uncertainties of the measurement.

Because the nonlinearities in the IPC model, a linear model cannot describe the system satisfactory. Therefore, an *extended Kalman filter* (EKF) is used. In the EKF a linear model is

created by linearization about the current state at each iteration. This enables the Kalman filter to be used for nonlinear systems, however it is required that the linear model is a good approximation of the system around the point of the current estimate[30 pp 125-128]. This mean that the filter is best suited for processes that are slightly nonlinear and differentiable. It should be noted that the extended Kalman filter is not an optimal estimator, as it is based on a linear approximation. But it will generally perform good on processes that are slightly nonlinear around the estimation point.

The noise in the IPC system is a quantization noise and is technically not a zero mean gaussian noise. In the Kalman filter the noise is expected to be zero mean gaussian, but the Kalman filter has shown to give good results also for the quantization noise, as produced by encoders [27].

The implementations of the extended Kalman filter can have some slight variations depending on the use. Here the implementation is done in Simulink using the *extended Kalman filter block* in the *control system toolbox*. The description of the filter given below is therefore based on the MATLAB online documentation. The MATLAB documentation is general, as it can be modified in the block settings. The description here will be slightly modified, to describe the functionality of the settings used.

A nonlinear discrete-time system with additive noise can be described by the following difference equation for the state and measurement respectively

$$x_{k+1} = g(x_k, u_k) + w_k \tag{4.4}$$

$$y_k = h(x_k) + v_k \tag{4.5}$$

where the subscript k denotes the time instance $t = k * T_s$, $g(x_k, u_k)$ is a nonlinear state transition function evolving the state from timestep k to timestep $k+1$ and $h(x_k)$ is a nonlinear measurement function relating the states, x, to the measurements, y, at timestep k. w_k is the process noise and v_k is the measurement noise at timestep k. Both w_k and v_k are gaussian with zero mean described as

$$w_k \sim (0, Q) \tag{4.6}$$

$$v_k \sim (0, R) \tag{4.7}$$

Q is the process noise covariance and R is the measurement noise covariance. We consider both to be time invariant. The Jacobi matrix for the nonlinear state transition function is defined as

$$G_k = \left.\frac{\partial g}{\partial x}\right|_{\hat{x}_{k|k}, u_k} \tag{4.8}$$

and the Jacobi matrix for the measurement function as

$$H_k = \left.\frac{\partial h}{\partial x}\right|_{\hat{x}_{k|k}}. \tag{4.9}$$

In the following $\hat{x}$ is the state estimate and P is the estimation error covariance matrix. $\hat{x}_{k_a|k_b}$denotes the estimate at timestep k_a using the measurements at time steps $0,1,\dots,k_b$.

At start-up of the algorithm an initial estimate of the state $\hat{x}_{0|-1}$ must be given along with the associated initial estimation error covariance matrix $P_{0|-1}$. The initial estimate is the "best guess" of the state at start up and is specified in advance. The algorithm then uses the following two steps to estimate the step at each iteration [31].

Update Kalman gain, estimate and covariance estimate

$$K_k = P_{k|k-1}H_k^T\left(H_k P_{k|k-1}H_k^T + R\right)^{-1} \quad (4.10)$$

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k\left(y_k - h\left(\hat{x}_{k|k-1}\right)\right) \quad (4.11)$$

$$P_{k|k} = P_{k|k-1} - K_k H_k P_{k|k-1} \quad (4.12)$$

Predict next estimate and next estimate covariance

$$\hat{x}_{k+1|k} = g\left(\hat{x}_{k|k}, u_k\right) \quad (4.13)$$

$$P_{k+1|k} = G_k P_{k|k} G_k^T + Q \quad (4.14)$$

At the update step the Kalman gain K_k is updated based the estimated state covariance, the measurement Jacobian, and the measurement covariance as stated in (4.10). The Kalman gain will dictates how much the estimate should rely on the prediction prior to the measurement and the measurement, as shown in (4.11) where the estimated state is updated. The error covariance is updated by calculating how much the predicted error covariance is propagated through the measurement Jacobian, H_k and the Kalman, K_k gain used at the state estimation as shown in (4.12).

At the prediction step the prediction of the estimate at the next step is calculated by (4.13)(4.12) using the nonlinear state transition equation, the current estimate and the current input. The estimate error covariance at next step is calculated based on how the current estimate error covariance will propagate in the system through the transition function Jacobian, G_k, and adding the process noise covariance, as shown in (4.14).

4.1.3.1 Discrete time system model

As seen above the EKF requires a discrete time system model. The model described in by (3.32) is a continuous model and must therefore be discretised. This will be done using the *forward Euler method [32]*. The simulation model including coulomb friction and air drag will be used in the Kalman filter.

It is only necessary to estimate the angular velocity, x_4, as measurements for the other states are accurate and has low noise. The mathematical model in (3.32) shows that the pendulum angle, x_3, and pendulum angular velocity, x_4, are not affected by the cart position, x_1, and the

cart velocity, x_4. This means the model used in the Kalman filter can be reduced to model for the states concerning the pendulum angle, x_3 and x_4. This give the state vector used in the Kalman filter

$$\underline{x}_P = \begin{bmatrix}\theta \\ \dot{\theta}\end{bmatrix}\begin{bmatrix}x_3 \\ x_4\end{bmatrix}. \tag{4.15}$$

And the continuous model for reduced system as

$$\underline{\dot{x}}_P = f_P(\underline{x}_P, u) \tag{4.16}$$

where $f_P(\underline{x}_P, u)$ is the two bottom rows of $f_P(\underline{x}, u)$ in (3.32).

The discrete time model for the reduced system is then

$$\underline{x_P}_{k+1} = g\left(\underline{x_P}_k, u_k\right) + w_k \tag{4.17}$$

where

$$g\left(\underline{x_P}_k, u_k\right) = \underline{x_P}_k + T_s * f_P\left(\underline{x_P}_k, u_k\right) \tag{4.18}$$

is the discretization of $f_P(\underline{x}_P, u)$, using the *forward Euler method.* The measurement function will be equal to the measurement model in (4.1), giving

$$y_k = h\left(\underline{x_P}_k\right) + v_k$$

with

$$h\left(\underline{x_P}_k\right) = x_3. \tag{4.19}$$

4.1.3.2 Derivation of Jacobi matrices

The EKF uses the Jacobi matrixes G_k and H_k. The EKF block in Simulink can compute these numerically or functions for computing these can be added. It was decided to include functions for calculation of the Jacobi matrixes as this reduces the number of computations needed at each update. This means that the Jacobians must be calculated analytically.

The Jacobi matrix for the measurement function is straight forward to derive and is

$$H_k(\underline{x_P}_k) = \frac{\partial h\left(\underline{x_P}_k\right)}{\partial \underline{x}_{P_k}} = \begin{bmatrix}1 & 0 \\ 0 & 0\end{bmatrix} \tag{4.20}$$

For the Jacobi matrix for the transition function we have

$$G_k(\underline{x}_{P_k}, u_k) = \frac{\partial g}{\partial \underline{x}_{P_k}} = \frac{\partial \underline{x}_{P_k}}{\partial \underline{x}_{P_k}} + T_s * \frac{\partial f_P\left(\underline{x}_{P_k}, u_k\right)}{\partial \underline{x}_{P_k}}$$

with

$$\frac{\partial \underline{x}_{P_k}}{\partial \underline{x}_{P_k}} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{4.21}$$

and

$$\frac{\partial f_P\left(\underline{x}_{P_k}, u_k\right)}{\partial \underline{x}_{P_k}} = \begin{bmatrix} 0 & 1 \\ -\frac{ml\left(g\cos\left(x_{3_k}\right) + \sin\left(x_{3_k}\right) u_k\right)}{I + ml^2} & -\frac{k_d - 2k_{dr}x_{4_k} - F_C k_t \, sech^2\left(k_t x_{4_k}\right)}{I + ml^2} \end{bmatrix} \tag{4.22}$$

4.1.3.3 Tuning covariance matrices

The covariance matrices Q and R can be seen as tuning variables for the EKF, as the relation between them will affect how much of the prediction is based on the measurement and the a priori estimate. If both the measurement noise covariance and the process noise covariance is known, these should normally be used. In many cases the covariances in the system are unknown or are uncertain. In these cases, the Kalman filter must be tuned, to achieve the desired response. For the EKF used here, it is desirable to smoothen out the effect that the noise from the quantified measurement of the angle will have on the angular velocity, while still keeping the estimate responsive to any external noise acting on the pendulum, for example a push.

As a starting point $R = 0.002$ is chosen, as this in some degree correspond to the quantisation steps for the angle measurement. We know that this measurement is accurate and therefore use larger values for Q. This result is that the estimate will rely more on the measurement, than the predicted estimate. For the Q matrix we will only assign values along the main diagonal.

In the simulation a wide variety of Q values gave good results, but when the complete control system was implemented on the real IPC system, more tuning was needed. Initially the filter had a slight bias on the estimated angular velocity. This was removed by increasing the Q matrix. Indicating that the real IPC system has greater uncertainties than the simulation model, as should be expected. The final tuning of Q was done by looking at the response of the estimate when a small push was given to the pendulum while it was balance in the upright position. This will be further discussed in the next section and illustrated in Figure 4.3, where the two estimation options are compared. The final value of the process noise covariance is

$$Q = \begin{bmatrix} 0.01 & 0 \\ 0 & 8 \end{bmatrix} \tag{4.23}$$

The initial value is set to $\hat{x}_{0|-1} = [\pi\ 0]^T$, corresponding to the pendulum down position. The initial estimation error covariance is set to $P_{0|-1} = diag[1\ 1]$.

4.1.4 Comparing numerical differentiation and EKF

Figure 4.2 shows the real and estimated velocities of a simulated free oscillation of the pendulum starting at 0.2 rad, just after it has passed the bottom point of the first swing. The angular velocities have been estimated with the EKF and numerical differentiation. The differentiation estimate has been filtered through two different average mean filters with length $n = 2$ and $n = 6$. The figure shows that the Kalman filter nearly matches the real velocity. For the estimate using differentiation, it is clear that using a short filter has some noise, and if the filter is longer, there is a considerable delay on the estimate.

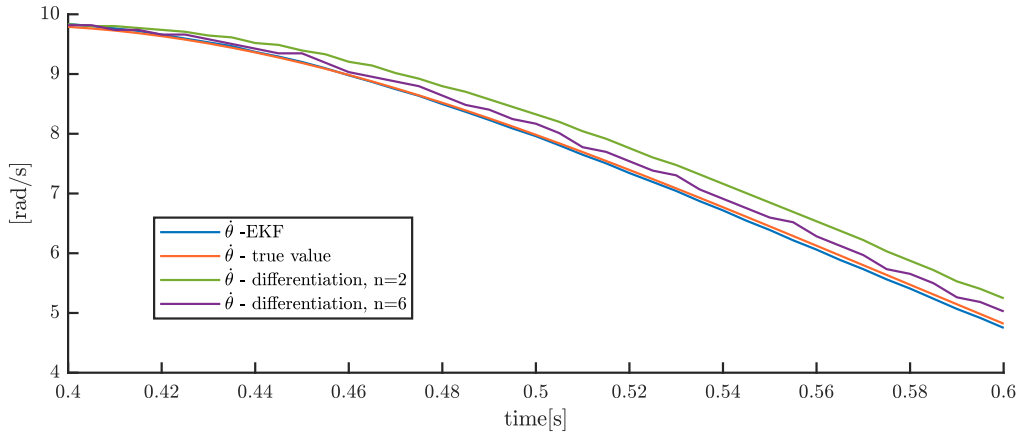


Figure 4.2 Phase delay in velocity estimation. The figure shows a simulation of real and estimated angular velocities of the pendulum during a swing from 0.2 rad, as it is just past the bottom point. The angular velocity is estimated with the EKF and differentiation and an average mean filter with two different lengths(n).

Figure 4.3 shows the difference of the angular velocity estimate using EKF and differentiation with filter length of 2, when the pendulum is expose to noise, in the form of a push in the negative direction. The pendulum is here controlled by the final controller, using the EKF estimate. The figure shows that the noise in the EKF estimate is significantly reduced compared to the differentiation estimate. The EKF does however not perform as good as the differentiation when the pendulum es exposed to a push. This is mainly because the controller responds with a strong control input based on the other measurements, leading the EKF predicted estimate based on the model to move in the wrong direction. The EKF filters ability to give a good estimate after a push will therefore also depend on the controller.

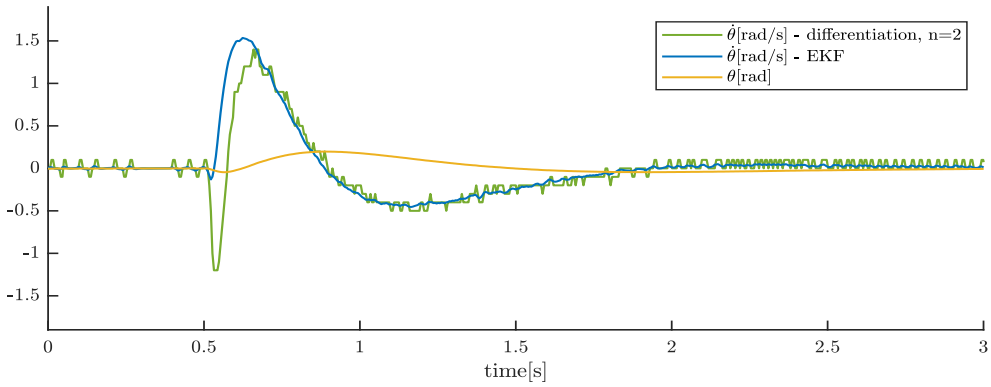


Figure 4.3 Velocity estimation after noise push to pendulum). The figure shows the measured angle and the estimated angular velocity using EKF and differentiation filtered with average mean filter with length n=2, from the real system while the pendulum is balanced in upright position with the complete controller. The EKF estimate is used as angular velocity input to the controller. At time=0.5 s the pendulum is given a short push in the negative direction.

Figure 4.2 and Figure 4.3 show that both estimation methods have their advantage, and the choice between them should be done by prioritising low noise at normal operation or the ability to detect an external disturbance. It is therefore chosen to use EKF because low noise is desired when the pendulum is balanced. The system will also measure disturbance directly through the pendulum angle.

4.1.5 Final estimator

For the final input to the controller it is desirable to have the angle measurement in the range $[-\pi, \pi]$. This ensures that the angle is 0 when the pendulum is in the upright position regardless which direction it comes up or if it is rotated several rotations. This is done by the following

$$\theta_{-\pi,\pi} = \big((\theta + \pi)\%2\pi\big) - \pi. \tag{4.24}$$

% is here the floating point modulo operator, implemented with the MATLAB *mod()* function.

Figure 4.4 shows a block diagram of the complete state estimation.

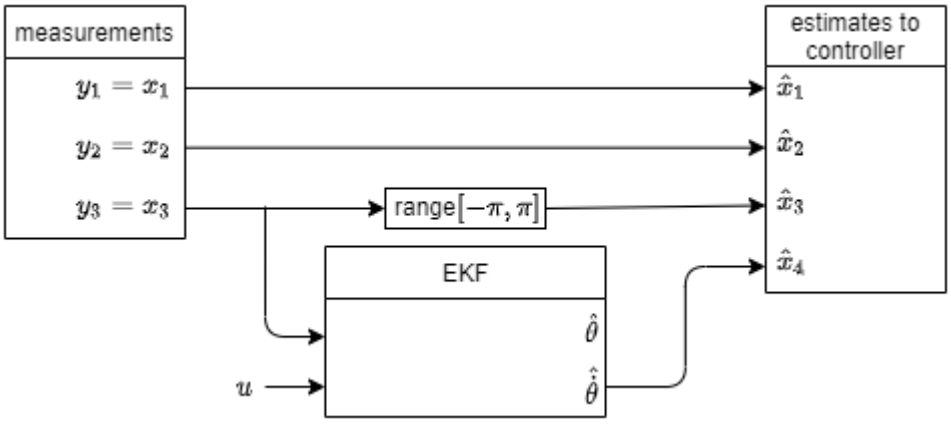


Figure 4.4 Block diagram state estimation

4.2 Swing up control

The swing up controller used here is an energy-based controller. This concept was first introduced by Åström and Furata [8] and is based on controlling the pendulum energy, instead of controlling the system states directly. If an energy corresponding to the pendulum energy in the upright position is given to the pendulum, the pendulum will swing to the upright position. Hence by adding energy to the pendulum through the cart acceleration input, u, and controlling it to be close to the pendulum angle in upright position, the pendulum will swing up and can be caught by the LQR controller presented in section 4.3 in the upright position.

The method presented by Åström and Furata does not take any consideration with regards to limitations in the system, such as the track length and maximum cart velocity. Therefore a extension to this method, presented by Chatterjee et al. [9] will be used. In this extension the problem with limitations in the system is handled with what they refer to as an "energy well". If the cart approaches the end of the track or the velocity is close to its maximum, a penalty will be given, to reduce the control input.

In this section the pendulum is considered as frictionless. This is of course not the case for the real pendulum, but it simplifies the explanation of how the swing up controller functions and the effects of the friction in the system will be compensated for by tuning. The equations of motions for the pendulum in (3.18) can then be restated as

$$I_H\ddot{\theta} + mlcos(\theta)u - mglsin(\theta) = 0 \tag{4.25}$$

Note that $\ddot{x}$ is replaced by u and $ml^2 + I$ is replaced with I_H, which is the pendulums moment of inertia around the hinge point, given by

$$I_H = \frac{1}{3}m(2l)^2 = \frac{4}{3}ml^2 = ml^2 + I\,. \tag{4.26}$$

The pendulum energy is defined as a sum of the rotational energy and the potential energy. The translational movement of the pendulum is not included as it will play a negligible role[9]. By defining the potential energy to be 0 at the height of the hinge point, the pendulum energy is then given by

$$E_{pen} = \frac{1}{2}I_H\dot{\theta}^2 + mglcos(\theta).$$

The energy of the pendulum in the upright position, with 0 angular velocity, for the system is

$$E_{up} = mglcos(\theta) = mglcos(0) \approx 0.3131\,J. \tag{4.27}$$

Since the goal of the swing up controller is to get the pendulum to an upright position by increasing the pendulum energy, the control input $u = \ddot{x}$ must influence the time derivative of the pendulum energy.

$$\frac{dE_{pen}}{dt} = I_H\dot{\theta}\ddot{\theta} - mgl\dot{\theta}\,sin(\theta) \tag{4.28}$$

By inserting an expression for $I_H\ddot{\theta}$ from (4.25) into (4.28) the time derivative is rewritten as

$$\begin{aligned}\frac{dE_{pen}}{dt} &= -mlcos(\theta)u\dot{\theta} + mglsin(\theta) - mgl\dot{\theta}\,sin(\theta) \\ &= -mlcos(\theta)\dot{\theta}u.\end{aligned} \tag{4.29}$$

This shows that the pendulum energy can be controlled through the control input, u. Further it is seen that the rate of energy change is dependent both on the pendulum angle and angular velocity. The control input as largest effect when θ is 0 or π, corresponding to the upright and down position. The control input has zero effect if $\theta = \frac{\pi}{2}$, corresponding to the pendulum in the horizontal position. It is also clear that the control input has zero effect if $\dot{\theta} = 0$. This is however not a problem, because any control input will make the pendulum swing, making $\dot{\theta}$ nonzero. Equation (3.28) forms the basis for the functionality of the energy-based swing up controller.

The swing up controller consists of two modes. The first mode is an energy injection mode, where the rotational energy is increased up to the value of E_{up}. The second mode is an energy maintenance mode where the controller seeks to keep the pendulum energy close to E_{up}. The complete controller is given as the sum of the following four terms [9], which are further described in the following,

$$u = u_{cw} + u_{vw} + u_{su} + u_{em} \tag{4.30}$$

The first term, u_{cw}, is referred to as the "cart potential well" and is used to keep the cart inside the restricted track length. This is done by introducing a repulsive force on the cart as it approaches the end of the track, in the form of an acceleration input towards the middle of the track. It is desirable that the cart can oscillate with larger amplitudes. Therefore, a logarithmic function is used, giving a low penalty input when the cart is in the middle part of the track and a rapidly increasing penalty when the cart approaches the track limit. The expression for the "cart potential well" is

$$u_{cw} = k_{cw}sign(x_1)\,log\left(1 - \frac{|x_1|}{x_{1\,max}}\right). \tag{4.31}$$

k_{cw} is a constant used for tuning the penalty input and $x_{1\,max} > 0$ is the maximum distance the cart can be from the middle of the track in either direction? The sign function is used to get the penalty input acting in the direction towards the middle of the track [9].

The second term, u_{vw}, is referred to as the "velocity potential well" and is used to keep the cart velocity below the limitations of the system. This act similar to the "cart potential well", but with regards to velocity. This means that the term will have a low effect while the velocity is well inside the limitations of the system, and a rapidly increasing penalty as the velocity approaches the maximum velocity for the system. The expression for the "velocity potential well" is

$$u_{vw} = k_{vw}sign(x_2)\,log\left(1 - \frac{|x_2|}{x_{2\,max}}\right). \tag{4.32}$$

k_{vw} is a constant used for tuning the penalty input and $x_{2\,max} > 0$ is the maximum velocity for the system [9].

The last two terms determine how the energy is added into the pendulum and how E_{pen} is maintained around the value of E_{up}. Their functions are closely related, so they will first be presented, before a description of result of combining them is presented.

The third term, u_{su}, is a term for injecting energy into the pendulum, and is given by

$$u_{su} = -k_{su} sign(\dot{\theta}\, cos(\theta)). \qquad (4.33)$$

The constant k_{su} is here at constant for determining the rate of energy injection into the pendulum. If (4.33) was the only term defining the control input u, the time derivative of the pendulum energy in (4.29) would always be positive, meaning that the pendulum energy would always be increased [9].

The fourth term, u_{em} is referred to as an energy maintenance mode that is used to maintain E_{pen} close to E_{up}. The term is given by

$$u_{em} = k_{em}\left(e^{|E_{pen} - \eta E_{up}|} - 1\right) * sign\left(E_{pen} - E_{up}\right) * sign\left(\dot{\theta} cos\theta\right), \qquad \eta > 1 \qquad (4.34)$$

The first *sign* function defines the input direction depending on if the pendulum energy should be increased or decrease. The last *sign* function decides the direction for the input which would give a positive derivative for the pendulum energy, depending on the pendulum angle and direction of angular velocity. The magnitude of the u_{em} term is defined by $k_{em}\left(e^{|E_{pen} - \eta E_{up}|} - 1\right)$. Since the magnitude depends on an exponential of the difference between E_{pen} and E_{up}, the magnitude of u_{em} will be small in a band around $E_{pen} = E_{up}$. The width of this band can be tuned using η. k_{em} is a tuning variable for the magnitude of the input. Because of the exponential term, k_{em} has largest effect when the error in the pendulum energy is larger, while η has largest effect when the error is small [9].

Because $|E_{pen} - \eta E_{up}|$ is used in an exponential term, the input from u_{em} is very large when E_{pen} is small, causing E_{pen} to rapidly increase. However, the increase of energy decreases as E_{pen} approaches E_{up}. If the term u_{su} was not included in the control input u described by (4.30), it is a risk that E_{pen} never reaches E_{up}, depending on the energy reducing effects from u_{cw}, u_{vw} and friction in the system. By also including the u_{em} term with a large enough value of k_{em} it can be guaranteed that E_{pen} will reach E_{up} [9]. When the necessary pendulum energy is reach, the u_{em} term will work to reduce the energy, meaning that u_{su} and u_{em} will cancel each other out and E_{pen} is slightly above E_{up}.

This means that a swing up controller as described by (4.30)-(4.34) and proper tuning will result in a rapid increase in the pendulum energy at the start. The rate of energy increase is reduced as E_{pen} approaches E_{up}, but it is guaranteed that E_{pen} will reach E_{up}. The terms u_{cw} and u_{vw} ensures that limitations on track length and cart velocity are not violated. Because of the reduced rate of energy increase when E_{pen} is close to E_{up}, the u_{cw} and u_{vw} terms will have a larger importance on the input. This leads to the cart being close to the centre of the track and at low velocity when the pendulum swings to the upright position, giving the LQR controller a good starting point for catching the pendulum.

4.2.1 Tuning the swing up controller

The tuning has been done by simulations in Simulink. The IPC system is simulated using the simulation model in section 3.5. Encoder measurements has been simulated with quantization blocks and the system states has been estimated as described in section 4.1, making it equal to what will be used on the real system. The input u is limited to $\pm 20 m/s^2$, as it will be for the real system.

In the final controller the system will switch from the swing up controller to the LQR controller if $|\theta| < 0.2\, rad$. This switch has not been included in the tuning. This is done to ensure that E_{pen} will stabilize around E_{up}, even after the point where the controller switch is normally done. It has been sought to ensure that pendulum will reach at least $|\theta| < 0.15\, rad$ or that $\dot{\theta} < 2\, rad/s$, if the pendulum does a full rotation, on every swing after the first that crosses $|\theta| < 0.2\, rad$. This ensures that the LQR controller should manage to stop and balance the pendulum. See section 4.3.4 for the selection of these values.

Further it has been sought to reduce the time and number of swings necessary for the swing up, while keeping the cart well within the track length of $\pm 0.43 m$ and keeping the cart velocity below maximum of $1.5 m/s$. The controller is tuned to have some tolerance to these limits to allow for some differences in the real system and the model and for cases where the controller is started while the pendulum already swinging.

The tuning of the swing up controller proved to be a cumbersome procedure, as it has a total of five tuning parameters, $k_{cw}, k_{vw}, k_{su}, k_{em}$ and η. No good way of determining the tuning parameters by calculation have been found. The tuning is therefore done as an iterative process where the tuning parameters have been modified based on intuition, understanding of the function of the controller, and simulation results. A general way of obtaining a decent result is to start using only the u_{cu} term. Then increase k_{su} until E_{pen} can reach E_{up}. Then introduce u_{cw} and u_{vw} and increase their parameters until the limitation of track length and maximum cart velocity is not violated. Also adjust k_{su} to ensure that E_{pen} can reach E_{up}. Finally, the u_{em} is introduced and its weight gradually increased. All tuning parameters needs to be gradually modified until a good result is achieved. The final tuning parameters are shown in Table 4.1.

Table 4.1 Parameters for swing up controller.

Swing up control parameters	
Symbol	**Value**
$\boldsymbol{k_{su}}$	0.57
$\boldsymbol{k_{cw}}$	6.5
$\boldsymbol{k_{vw}}$	3.2
$\boldsymbol{k_{em}}$	9.3
$\boldsymbol{\eta}$	1.265

Figure 4.5 shows the pendulum angle and angle velocity for a simulation of the swing up controller using the above parameters. Figure 4.6 shows the corresponding calculation of the pendulum energy and Figure 4.7 shows the corresponding input, cart position and cart velocity.

As seen in Figure 4.5 the pendulum crosses the controller switching angle on the third swing after approximately 2 seconds. The pendulum almost reaches an angle of $\theta = 0$, before it swings back. At the next swing the pendulum passes $\theta = 0$, with an almost zero angular velocity. This indicates that the LQR controller most likely should catch the pendulum in the upright position.

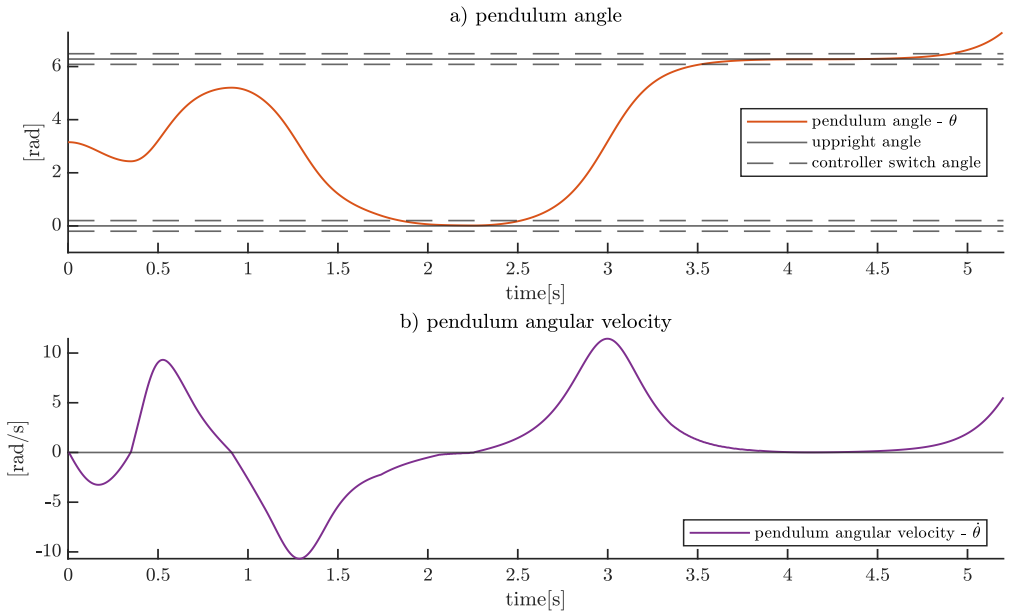


Figure 4.5 Pendulum angle and angular velocity for tuned swing up controller. θ is not limited to $[-\pi, \pi]$, as this makes it easier to interpret the figure.

Figure 4.6 shows that the pendulum energy increase rapidly at the start. From 0.6 seconds the rate of increase smoothens out and from 1.7s is maintained around E_{up}. The pendulum energy shows a stepping increase. The flat parts of the curve are there because either $\dot{\theta}$ is close to zero, θ is close to 0 or π or because the cart is near the track end and must change direction. To achieve the pendulum to swing up in as little as three swings, the controller is tuned quite aggressively. This leads to the overshoot of pendulum energy around 3 seconds. This is however considered to be acceptable.

The rate of energy increase between 0.6s and 1.7s could have been increase with a larger value of k_{su}, but this reduced the controllers ability to stabilize E_{pen} around E_{up}.

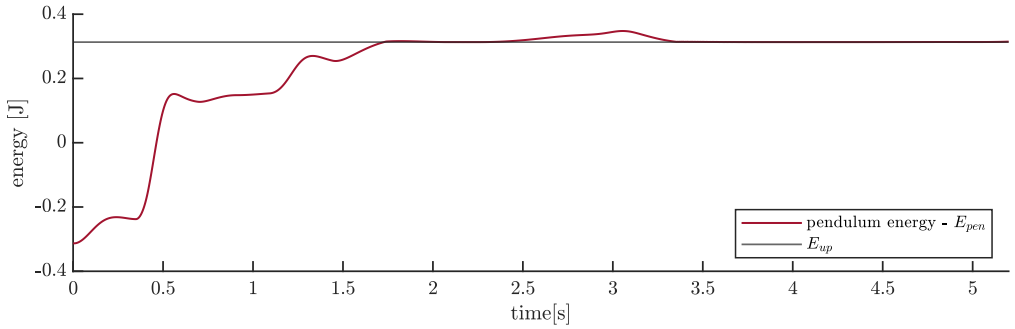


Figure 4.6 Pendulum energy for tuned swing up controller.

Figure 4.7 shows that the cart velocity is well within the limits. The cart position is at minimum 5cm from its limitation, which is acceptable. The input is saturated at a small part of the time and is otherwise considered to have low noise most of the time.

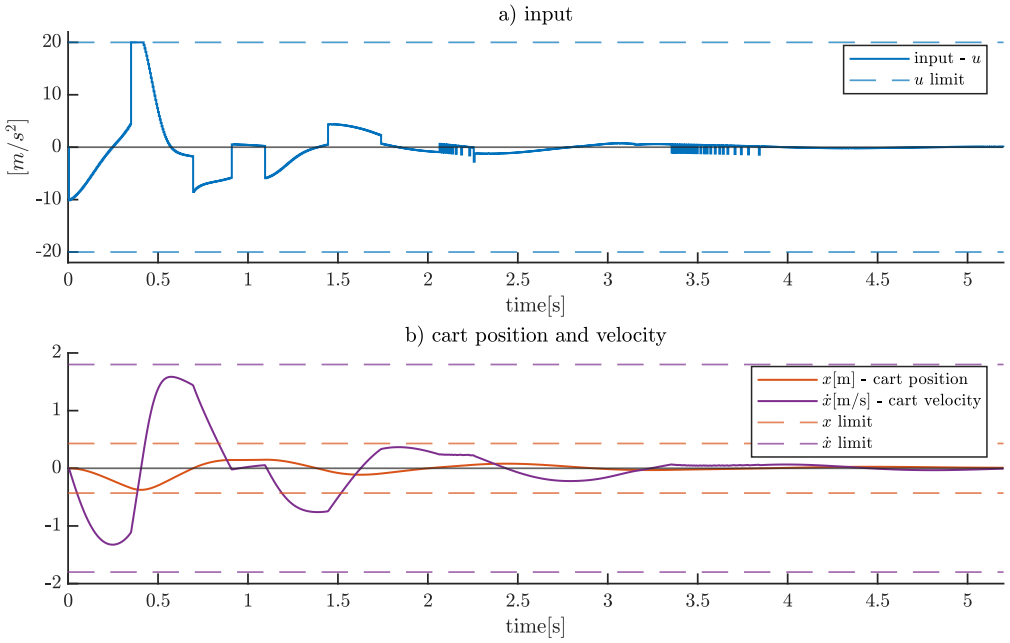


Figure 4.7 Input, cart position and cart velocity for tuned swing up controller.

It will not be performed any stability analysis here, but it is referred to the stability analysis done in the by Chatterjee et al. [9]. Here they state that the controller ensures stability under reasonable conditions, such that the system has the necessary performance regarding track length and input, that the tuning variables are chosen properly and that the LQR controller manages to “catch” the pendulum.

4.3 LQR control for balancing in upright position

To balance the pendulum in the upright position a *linear quadratic regulator* is used. This a state-feedback controller for linear systems, where the control input is calculated using a feedback gain on the errors in the state. The feedback gain is found as a solution to a *quadratic programming* optimization problem, where errors in the state and the control input are associated with a cost, and the system model is used as an equality constraint. The optimization problem therefore finds the feedback gain that minimizes the cost over a chosen predicted future time horizon. The calculated gain is considered as the optimal feedback gain

The LQR comes in some variations. In this implementation the version often referred to as *infinite-horizon, discrete- time LQR*. This means that optimization problem finds the gain that will minimize the cost for an infinite future. The system will be modelled as a discrete-time linear time invariant, based on a linearization around the equilibrium where the pendulum is in the upright position. The timesteps are $0.005s$, corresponding to the update frequency of $200Hz$.

Considering a discrete-time LTI[11] system described by

$$x_{t+1} = A_d x_t + B_d u_t, \tag{4.35}$$

where t denotes the discrete timesteps, the optimization problem for the *infinite-horizon, discrete- time LQR* is defined as

$$\min_z f^{\infty}(z) = \frac{1}{2}\sum_{t=0}^{\infty} x_{t+1}^T Q_c x_{t+1} + u_t^T R_c u_t \tag{4.36}$$

subject to the equality constraints

$$x_{t+1} = A_d x_t + B_d u_t \tag{4.37}$$

$$x_0 = given. \tag{4.38}$$

Further we have that $Q_c \geq 0$ and $R_c > 0$. Both are diagonal matrices containing the cost for errors in the states and cost for the input respectively. The subscript, c, is used to differentiate these from the Q and R in the Kalman filter. The dimension of the optimization problem is given by the number of states and inputs, $x_t \in \mathbb{R}^{n_x}$ and $u_t \in \mathbb{R}^{n_u}$. As seen the from the summation, the problem spans over an infinite horizon. This means that the problem has an infinite dimension, as $z^T = (u_0^T, \dots, u_\infty^T, x_1^T, \dots, x_\infty^T)$ [33 p 64].

Given that the optimization problem in (4.36)-(4.38) is bounded above, meaning that $f^{\infty}(z) < \infty$ for some feasible z, and that the system given by (4.37) is stabilisable[12], the problem is a solution of the *algebraic Riccati equation* [33 p 64-65].The feedback control law given by

$$u_t = -K_C x_t \tag{4.39}$$

where K_C is here the state feedback gain, can then be found by

$$P = Q_c + A_d^T P(I + B_d R_c^{-1} B_d^T P)^{-1} A_d \tag{4.40}$$

$$P = P^T \geq 0 \tag{4.41}$$

$$K_c = R_c^{-1} B_d P(I + B_d R^{-1} B_d^T P)^{-1} A_d \tag{4.42}$$

(4.40) is the *algebraic Riccati equation*, which may have several solutions, but only one is positive semidefinite. Is shown by (4.41) the positive solution is chosen. The feedback gain matrix is calculated by (4.42) [33 p 64-65]. Note that when reference tracking is used, the feedback control law is modified to

[11] LTI – linear time invariant.
[12] Stabilizable is a weaker form of controllable, and states that any modes that are not asymptoticly stable should be controllable.

$$u_t = -K_C(x_{ref_t} x_t), \tag{4.43}$$

where x_{ref} is the state reference.

In the following linearization of the system around the pendulum up position is shown. The model will then be extended to include integral action for the cart position. In both cases the model will be a continuous time model. The discretisation of the system model is done in MATLAB using the *c2d* function and the gain matrix is found using the *dlqr* function. In the end a discussion around the switching angle between the swing up and LQR controller is given.

4.3.1 Linearization around upright equilibrium

Since we are going to linearize the mathematical model, we will do this based on a model where the pendulum friction only consists of a damping term, as this is already linear. The system model is then given as

$$\underline{\dot{x}} = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} = \begin{bmatrix} x_2 \\ u \\ x_4 \\ \frac{mglsin(x_3)}{ml^2 + I} - \frac{mlcos(x_3)}{ml^2 + I} u - \frac{k_d}{ml^2 + I} x_4 \end{bmatrix}. \tag{4.44}$$

The system has two equilibrium points, one with the pendulum down, and one with the pendulum up. We will linearize the model in the pendulum up position. At this equilibrium we have the following state and input

$$\underline{\bar{x}} = \begin{bmatrix} * \\ 0 \\ 0 \\ 0 \end{bmatrix}, \bar{u} = 0. \tag{4.45}$$

Note here that the cart position is irrelevant, and therefore marked with a star. The linearization is performed using a first order Taylor series expansion around the equilibrium point, given by

$$f(\underline{x}, u) \approx f(\underline{\bar{x}}, \bar{u}) + \left.\frac{\partial f}{\partial \underline{x}}\right|_{\underline{x}=\underline{\bar{x}},\ u=\bar{u}} (x - \bar{x}) + \left.\frac{\partial f}{\partial u}\right|_{\underline{x}=\underline{\bar{x}},\ u=\bar{u}} (u - \bar{u}). \tag{4.46}$$

Note that (4.44) only contains two nonlinear terms, the terms containing $\sin(x_4)$ and $\cos(x_4)$, in the differential equation for the x_4 state. We will therefore only perform the linearization for these two terms and then put the result back into (4.44).

For the first term,

$$f_a(\underline{x}, u) = \frac{mglsin(x_3)}{ml^2 + I}, \tag{4.47}$$

we get the linear approximation

$$\begin{aligned} f_a(\underline{x}, u) &\approx \frac{mglsin(0)}{ml^2 + I} + \left(\frac{\text{mglcos}(0)}{\text{ml}^2 + \text{I}}\right) * (x_3 - 0) + 0 \\ &\approx \frac{\text{mgl}}{\text{ml}^2 + \text{I}} x_3 \end{aligned} \tag{4.48}$$

For the second term,

$$f_b(\underline{x}, u) = -\frac{mlcos(x_3)}{ml^2 + I} u, \tag{4.49}$$

we get the linear approximation

$$\begin{aligned} f_b(\underline{x}, u) &\approx -\frac{mlcos(0)}{ml^2 + I} * 0 + -\frac{mlsin(0)}{ml^2 + I} * (x_3 - 0) - \frac{mlcos(0)}{ml^2 + I} * (u - 0) \\ &\approx -\frac{ml}{ml^2 + I} u \end{aligned} \tag{4.50}$$

From the linear approximations (4.48) and (4.50), and the linear terms in (4.44), we can write the system as a linear state space model on the form

$$\dot{\underline{x}} = A\underline{x} + Bu,$$

with

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & \frac{mgl}{ml^2 + I} & -\frac{k_d}{ml^2 + I} \end{bmatrix} \tag{4.51}$$

and

$$B = \begin{bmatrix} 0 \\ 1 \\ 0 \\ -\frac{ml}{ml^2 + I} \end{bmatrix}. \tag{4.52}$$

The linearization around the equilibrium point consist only of linearizing $\cos(\theta)$ to be equal to 1, as this is the only nonlinear part of the original model. Figure 4.8 shows a comparison of $cos(\theta)$ and the linearization, around $\theta = 0$. The figure shows that the linearization will have an error less than 0.03 for when $\theta \leq \pm 0.25$, making the linearization a good approximation for expected values of θ expected while the pendulum is balanced in the upright position.

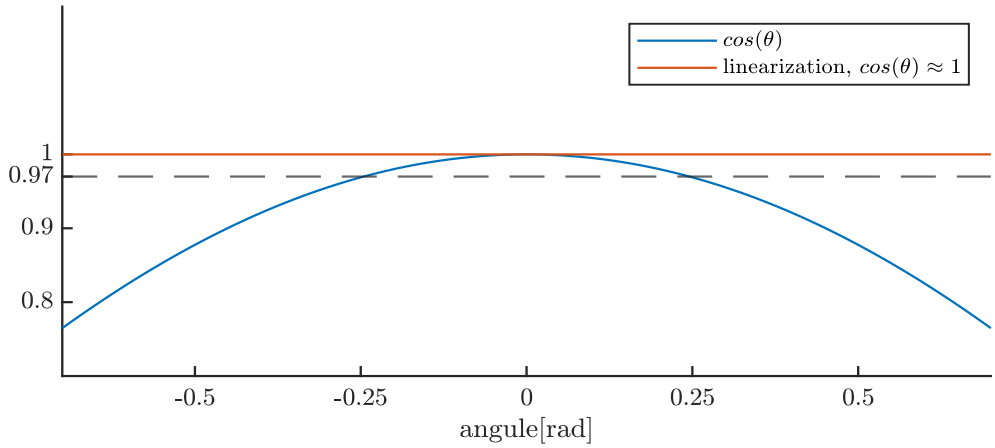


Figure 4.8 Comparison of $cos\ (\theta)$ and the linearization $cos\ (\theta) \approx 1$ around $\theta = 0$.

4.3.2 Adding integral action on the cart position

The LQR controller was first implemented using a system described by the A and B matrix in (4.51) and (4.52). On the implementation on the real system this gave a small constant offset on the cart position. This indicates that the system has a constant disturbance or measurement error. To compensate for this, integral action on the cart position is added by extending the system with what is often referred to as an augmented state [34]. The system is extending by adding a state described by the model

$$\dot{w} = x_{ref_1} - x_1 \tag{4.53}$$

This means that the state w is equal to the integral of the error in the cart position. The system can then be described using the extended state matrix

$$\gamma = \begin{bmatrix} x \\ w \end{bmatrix} \tag{4.54}$$

and the state space model

$$\dot{\gamma} = A_1\gamma + B_1 u \tag{4.55}$$

with

$$A_1 = \begin{bmatrix} & & A & & \begin{matrix}0\\0\\0\\0\end{matrix} \\ -1 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{4.56}$$

and

$$B_1 = \begin{bmatrix} B \\ 0 \end{bmatrix}. \tag{4.57}$$

This model is used for calculating the LQR gain, K_c. The states of the actual system $\underline{x}$ are estimated as described in section 4.1 and the extended state is calculated from this by

$$w = \int_0^t x_{ref_1} - \hat{x}_1 \, dt \tag{4.58}$$

The complete LQR control system is described by the block diagram in Figure 4.9.

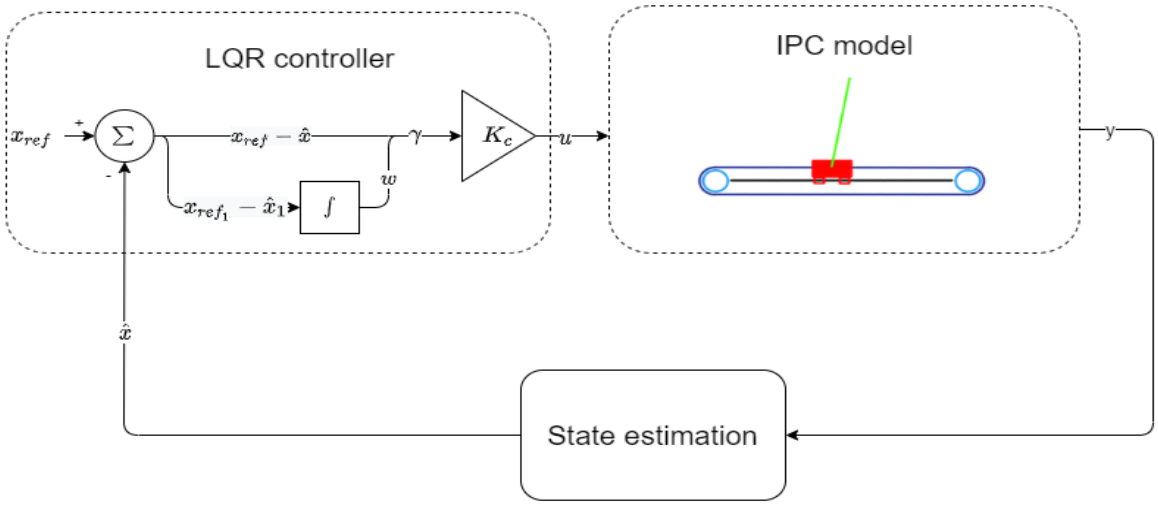


Figure 4.9 Block diagram of complete LQR controller.

4.3.3 Tuning LQR controller

The LQR controller is tuned by adjusting Q_c and R_c. The tuning was first done in simulation until a decent result was achieved before it was fine-tuned on the real model. For the controller we are mainly concerned with keeping the pendulum angle and the cart position close to their reference. This would lead to the cart velocity and pendulum angular velocity also being close to 0, which will be their reference while the pendulum is balanced in the upright position. It was further decided that an error in the pendulum angle of $0.02rad$ should have approximately the same cost as an error in the cart position of $0.06m$. This means that the cost associated with an error in the x_3 should be 3 times larger than the cost associated with the x_1. This was used as a starting point. The cost associated with input was adjusted to get pendulum to balance and response quickly to a disturbance in the form of a small push, while not getting a control input that is noise when the pendulum is balanced. The cost associated with the augmented state was then adjusted so that the cart was stable at the reference point. The cost for the cart velocity and the pendulum angular velocity was increase slightly, as this seemed to give a slightly smother response after a push. The final cost values are

$$Q_c = \begin{bmatrix} 4 & 0 & 0 & 0 & 0 \\ 0 & 0.1 & 0 & 0 & 0 \\ 0 & 0 & 11 & 0 & 0 \\ 0 & 0 & 0 & 0.2 & 0 \\ 0 & 0 & 0 & 0 & 0.16 \end{bmatrix} \tag{4.59}$$

And

$$R_c = 0.002 \tag{4.60}$$

This gives the state feedback gain of

$$K_c = [-46.5 \quad -35.1 \quad -139.7 \quad -23.2 \quad 8.0\,] \tag{4.61}$$

The eigenvalues for the closed loop system are found by $e = eig(A_{1d} - B_{1d}K_c)$, where A_{1d} and B_{1d} are the discretised model corresponding to A_1 and B_1. It should be noted that these eigenvalues do not include the dynamics generated by the EKF estimation of $\dot{\theta}$. This gives the eigenvalues

$$e = \begin{bmatrix} 0.999 \\ 0.9908 - 0.0075\boldsymbol{i} \\ 0.9908 + 0.0075\boldsymbol{i} \\ 0.9635 \\ 0.8449 \end{bmatrix} \tag{4.62}$$

All eigenvalues have a magnitude off less than one, and the system should therefore be stablem

4.3.4 Pendulum angle for switching between swing up control and LQR

The pendulum angle used for switching between the swing up controller and the LQR controller is $\theta \leq |0.2|$. This angle is chosen because the error of the linearization is quite small for the whole working range of the LQR controller. Further the LQR controller will in most cases manage to recover the pendulum to the balance point if it is within this range and it manages to catch the pendulum a it is swung up. This is based on testing in simulations and verified on the real system.

4.4 Final control system

The final control system can then be summarised by the block diagram in Figure 4.10.

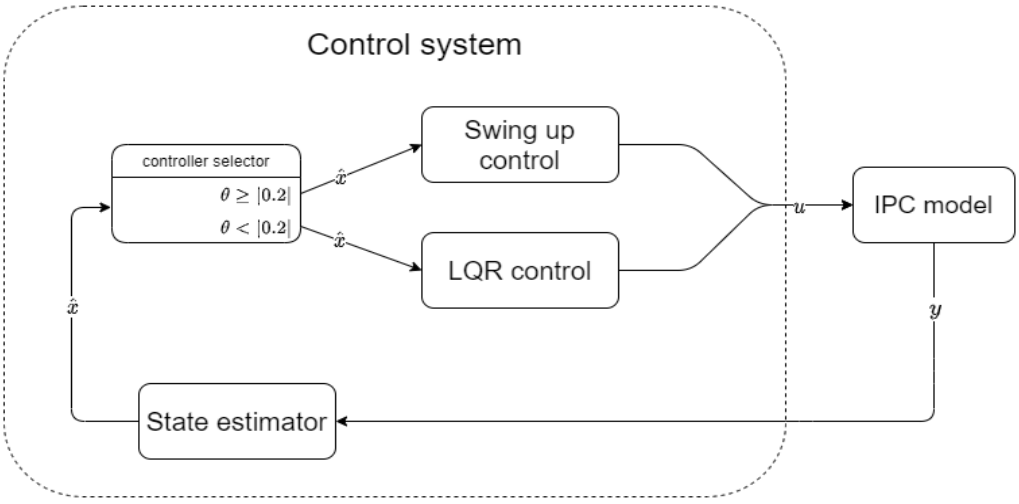


Figure 4.10 Control system block diagram.

5 RESULTS

Figure 5.1 shows a plot of all states and the control input both for the real system and a simulation. In the figure the pendulum is swung up from the pendulum down to the pendulum up position. After 10 seconds a sequence of changes to the cart position reference is performed as indicated in the figure.

Figure 5.2 shows all states and the control input as the pendulum is balanced in the pendulum up position, with the reference for all states as 0. The pendulum is exposed to disturbances in the form of a small push at the times indicated by the grey lines. At the time marked with a red line an additional weight of 170g is added to the end of the pendulum. 170g equals approximately 1.2 times the weight of the pendulum. The added weight is a screwdriver inserted into the end of the pendulum.

A video demonstration of the system can be found in the digital appendix. The video will show

- Homing procedure.
- Swing up of the pendulum.
- Balancing the pendulum while the cart position reference is changed.
- Balancing of the pendulum in the upright position, while being exposed to pushes and adding additional weight to the end of the pendulum.
- Stopping and restarting the controller at random times, to show how the system recovers to the reference.

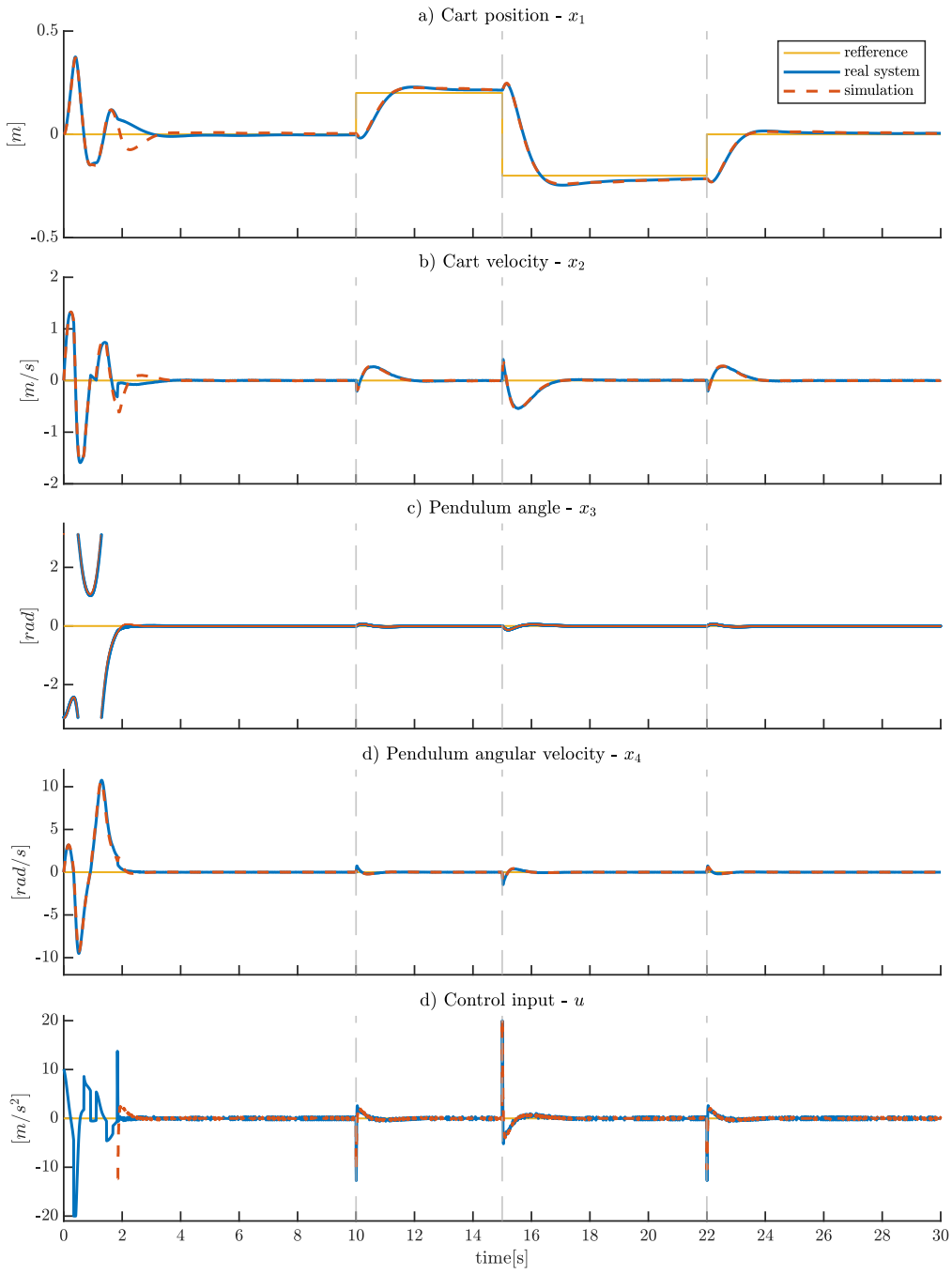


Figure 5.1 Swing up of the pendulum and change in cart position reference. The figure shows plots of all states and the input for the real system and a simulation, along with their refference values. The Pendulum is swung up from the pendulum down position to the pendulum up position starting at time = 0s. At times 10s, 15s and 22s the refference for the cart position is changed.

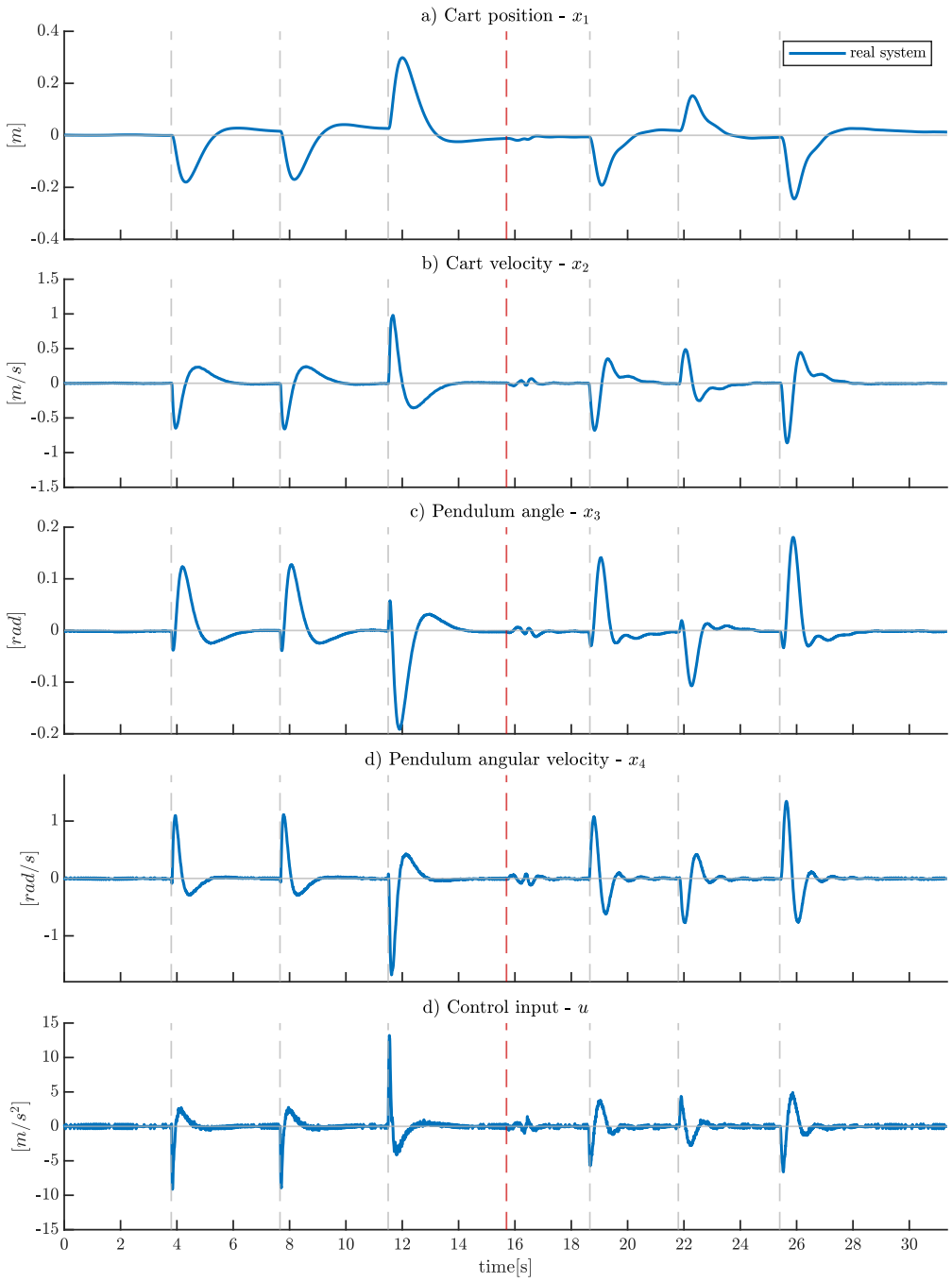


Figure 5.2 Balancing pendulum with LQR controlling while being exposed to disturbance. The figure shows all states and control input as the pendulum is balanced in the upright position. The pendulum is exposed to disturbance in the form off a push at times marked with grey vertical lines. At the time marked with the red vertical line an additional weight of 170g is added to the end of the pendulum.

6 DISCUSSION

Figure 5.1 shows that the swing up controller manages to swing the pendulum from the pendulum down position to the pendulum up position, where the LQR controller takes over and balances the pendulum in the pendulum up position. The swing up controller use approximately 2 seconds and three swings to get the pendulum to an angle of $\theta \leq 0.2\ rad$, where the controller is switch to LQR control. After approximately 3 seconds the LQR controller has stabilized all states close to their reference. This shows that the control system generally functions as planned.

6.1 Accuracy of mathematical model and simulation

It is seen in Figure 5.1 that the real system and the simulation is a very close match. This shows that the mathematical model is a very good representation of the real system. This has been very beneficial through the controller design process, as the controllers could be tested in simulations, and the performance would be very similar on the real system. The only mismatch seen in Figure 5.1 is around $time = 2s$, this will be commented on in section 6.3.

6.2 Swing up controller

The chosen swing up controller manages to swing up the pendulum to the upright position in three swings. The controller generally performs well both when starting with the pendulum hanging straight down or when the pendulum already is swinging when the controller is started. This can be seen in the demonstration video. If the pendulum already is swinging when the controller is started there is some variations of the number of swings needed before the upright position is reached. In some cases, it has been necessary with up to 5 swings after the controller is switched on. This seems to depend on cart position and pendulum angle when the controller is started. It has been hard to reproduce this, as the controller is started manually by pushing a button and is therefore added as a general observation.

It has been a couple of cases when the controller is started with the pendulum already swinging, where both the cart and the pendulum oscillate. In this case the pendulum will oscillate with amplitudes of $\theta > |2|\ rad$. The oscillations are of constant magnitude, and the pendulum will never reach the upright position. The problem is not illustrated by a plot in this report because it is hard to reproduce and have only occurred a few times while data has not been logged. For the same reason, the problem has not been investigated thoroughly. However, the swing up controller was first implemented to swing up in 4 swings, in this case the mentioned oscillations was never been observed. To achieve a swing up in 3 swings the controller is tuned considerably more aggressive than for a swing up in 4 swings. Chatterjee et al. [9] concludes that the swing up controller is stable under reasonable conditions. It is therefore considered that the aggressive tuning needed to achieve a swing up in 3 swings is on the limit of the criteria for a stable controller, and therefor has some edge cases where the pendulum will never reach the upright position.

It has also been tried to achieve a swing up in 2 swings in simulations. This has been achieved without violating the IPC model limitations, but the LQR controller could not "catch" the pendulum, as it had to high angular velocity when the upright position was reached. It is therefore concluded that the IPC model should be capable of a swing up in less than 3 swings, but the energy-based controller is not the right controller for this.

6.2.1 Alternative swing up controller

As an alternative method to swing up the pendulum, the method presented by Tum et.al. in [6] was considered. Their approach is to use a feed forward controller where a planned trajectory of the system states is used as a reference input to a state feedback controller.

The planned trajectory is found by solving an optimization problem which includes the system model and system limitations as constraints. This means that the system should be able to follow this optimal trajectory [6]. The trajectory optimization problem can be solved in advance to reduce the computation cost of the controller.

The feedback controller used by Tum et.al. in [6] is similar to the LQR controller presented in section 4.3. However, since it is used for the whole range of pendulum angles, the feedback cannot be calculated in advance as the linearization is only valid around $\theta = 0$. Instead the system is linearized around the current state and the LQR optimization problem is solved at each iteration. This means that the feedback controller will have a considerable computational cost.

This method of swinging up the pendulum was not implemented mainly because it was uncertain if the microcontroller would handle the computational cost. It was also considered to be a bit more work with implementing this method and it was a limited time before the deadline. With the controller as implemented in this project one update of the control input is calculated by the microcontroller in $100\mu s$. This means that the microcontroller can take a significant increase in computational cost and still achieve an update frequency of 200Hz. It is therefore likely that it would manage the computational cost of the method prese Tum et.al. Using this approach, it might be possible to achieve a swing up in less than 3 swings.

6.3 "Catching" the pendulum with the LQR controller

The only part of Figure 5.1 were there is a difference between the simulation and the real system is around the point where the controller switch from swing up control to LQR control. This should not be seen as an difference in the mathematical model and real system, but rather as small random variations in the real system and the swing up controllers ability to control the pendulum energy at its setpoint, E_{up}. This result in the pendulum having slightly different angular velocities as the controller is switched to LQR control. These variations are expected and does not cause a problem, but very small variations can create significantly different responses in the control input. If the pendulum comes in at a velocity too low for θ to reach 0, the cart must move underneath the pendulum, to compensate for this. If the pendulum comes in at a velocity that would make it go past $\theta = 0$, the LQR must slow down the angular velocity, by moving the cart away from the pendulum. This is seen in the video demonstration. It should be mentioned that during testing it has not been a single incident where the LQR controller has not managed to stop and balance the pendulum if it has passed the switching angle.

6.4 Balancing the pendulum in upright position

Both Figure 5.1 and Figure 5.2 shows that the pendulum is balanced well in the upright position. The pendulum angle is steady close to 0 and the cart position is close to its reference if there is no disturbance, or reference change. The cart has small movements of around $\pm\ 1cm$ while the pendulum is balanced. This is not visible in the figures but can be seen in the video demonstration.

After the reference changes in the cart position in Figure 5.1 it is seen that the cart position has a slight overshoot of the reference, and it takes a few seconds before it slowly approaches the reference. This overshoot and slow approach to the reference is because of the integral action in the controller. The integral action was added to compensate for a constant offset from the cart position reference. After some investigation it was found that also the pendulum was often balanced with a constant offset from $\theta = 0$. The offset is usually in the negative direction with a magnitude of less than $0.01\ rad$, but would change after the system was restarted. This indicates that the offset on the cart position is caused by the constant offset in the pendulum angle, which is probably caused by the two following possible reasons. One possibility is that the pendulum has it centre of gravity slight to one side, for example from a slight bend. This would create an offset that will always be in the system. The second possible reason is that the angle of the pendulum in the stable pendulum down position can vary, because of friction forces. This means that the initial pendulum angle set in the homing procedure will have some variations.

This means that the LQR control system must include integral action to compensate for the offset, as its direction and magnitude will vary. Figure 5.1 shows that the chosen cost value in associated with the integral action Q_c is a good compromise between compensating for the offset quickly and reducing the overshoot after the reference change. The maximum overshoot in the cart position in the figure, around 17 seconds, is found to be $4cm$ by checking the data.

6.5 Performance with disturbances

Figure 5.2 shows how the system responds to disturbances. At the times marked with the three grey lines to the left in the figure, it is seen that the system manages to recover back to the reference values after a push is applied to the pendulum. The force in the push is not possible to measure, but it is quite significant. This can clearly be seen in the demonstration video.

At times to the right of the red line in Figure 5.2 an additional weight of 170g is added to the end of the pendulum. This is approximately 1.2 times the pendulum weight. The added weight is a screwdriver inserted into the pendulum and is not fixed, meaning that it moves slightly inside the pendulum pipe. The system still manages to balance the pendulum, but it oscillates slightly more. This shows that the LQR control is robust to changes in the system.

When a push is given to the pendulum with added weight, the pendulum angle is recovered close to $0\ rad$ faster than without the added weight. This is because the add weight both increases the pendulum inertia and moves the pendulum centre of gravity further away from the cart. This means that the angular acceleration of the pendulum will generally be slower for force, than it would be without the added weight. The control input will then “overcompensate” for any error in the θ and $\dot{\theta}$ states, causing a quicker response, but also oscillations.

6.6 IPC-model

The IPC-model built in this master's thesis project generally functions well. The track length of $86cm$, the maximum control input of $20m/s^2$ and the maximum cart velocity of $1.8m/s$ makes the model a capable test bench for control problems related to the system. With the implemented control algorithm, the microcontroller calculates the new control input in less than $100ms$. This means that the microcontroller has a significant reserve regarding computation power, which makes the model suitable also for computational demanding control algorithms.

The motor will get hot while it is enabled. With the control algorithm implemented in this project the system has been running almost constantly for periods of up to an hour. The motor is hot, but it is found acceptable as it is okay to touch it with a hand. This can however depend on the control algorithm implemented and the possibility of overheating should be considered for future users. If it is found necessary, the heat generation of the motor can be reduced by reducing the motor coil current in the stepper motor driver setup [14]. This will reduce the force generated by the motor and it might also be necessary to reduce the maximum acceleration.

The control system for the model is ready for connection of an extra encoder. This means that pendulum can easily be extended to a double pendulum. Further the model has connections for a variety of serial communication protocols. This means that the system can easily be modified to be controlled from an external controller.

7 CONCLUSION

In this thesis a model of an inverted pendulum on a cart system has been built and controllers have been implemented on the system to swing the pendulum from a pendulum down position to a pendulum up position and balance it there. A mathematical model has of the system have been derived and used to tune the controllers in simulations in MATLAB/Simulink

The IPC-model consists of a pendulum connected to a cart sliding on a low friction railing. Pendulum angle and cart position is measured with encoders, and the cart can be moved with a stepper motor. The system is controlled by a microcontroller. The system has high computation power and high maximum control input to the cart, making it capable as a test bench for many control problems associated with the system.

The implemented controller consists of an extended Kalman filter for calculating the pendulum angular velocity, an energy-based controller for swinging the pendulum from a down position to a up position, and a LQR controller for balancing the pendulum in the up position.

The implemented extended Kalman filter ensures a low delay estimation of the angular velocity with low noise compared to a numerical differentiation of the measured angle. The estimation does however have some error if the pendulum is exposed to noise.

The swing up controller swings the pendulum from a down position to an up position in 3 swings and around 3 seconds.

The LQR controller successfully balances the pendulum in the upright position. Further it successfully handles any change in the cart reference position well. If the pendulum is exposed to noise such as a push or added weight, the controller successfully maintains the pendulum in the upright position.

BIBLIOGRAPHY

[1] M. Fiacchini *et al.*, "Design and Experimentation of a Personal Pendulum Vehicle," 09/11 2006.

[2] R. Sbresny, A. Getler, N. Felker, and C. Frederickson, *Implementation of an Inverted Pendulum PID Control System Using a Stepper Motor*. 2016.

[3] K. Mouratis, J. Fasoulas, and M. Sfakiotakis, *Development and Control of a Low-Cost Cart-Pendulum Educational Platform*. 2018.

[4] V. Šetka, R. Čečil, and M. Schlegel, "Triple inverted pendulum system implementation using a new ARM/FPGA control platform," in *2017 18th International Carpathian Control Conference (ICCC)*, 28-31 May 2017 2017, pp. 321-326, doi: 10.1109/CarpathianCC.2017.7970419.

[5] N. Muskinja and B. Tovornik, "Swinging up and stabilization of a real inverted pendulum," *IEEE Transactions on Industrial Electronics*, vol. 53, no. 2, pp. 631-639, 2006, doi: 10.1109/tie.2006.870667.

[6] M. Tum, G. Gyeong, J. Park, and Y. Lee, "Swing-up Control of a Single Inverted Pendulum on a Cart With Input and Output Constraints," *ICINCO 2014 - Proceedings of the 11th International Conference on Informatics in Control, Automation and Robotics*, vol. 1, pp. 475-482, 01/01 2014, doi: 10.5220/0005018604750482.

[7] L. B. Prasad, B. Tyagi, and H. O. Gupta, "Optimal control of nonlinear inverted pendulum dynamical system with disturbance input using PID controller & LQR," in *2011 IEEE International Conference on Control System, Computing and Engineering*, 25-27 Nov. 2011 2011, pp. 540-545, doi: 10.1109/ICCSCE.2011.6190585.

[8] K. J. Åström and K. Furuta, "Swinging Up a Pendulum by Energy Control*," *IFAC Proceedings Volumes*, vol. 29, no. 1, pp. 1919-1924, 1996/06/01/ 1996, doi: https://doi.org/10.1016/S1474-6670(17)57951-3.

[9] D. Chatterjee, A. Patra, and H. K. Joglekar, "Swing-up and stabilization of a cart–pendulum system under restricted cart track length," *Systems & Control Letters*, vol. 47, no. 4, pp. 355-364, 2002/11/15/ 2002, doi: https://doi.org/10.1016/S0167-6911(02)00229-3.

[10] P. Bakaráč, M. Kalúz, and L. Čirka, "Design and development of a low-cost inverted pendulum for control education," in *2017 21st International Conference on Process Control (PC)*, 6-9 June 2017 2017, pp. 398-403, doi: 10.1109/PC.2017.7976247.

[11] MathWorks. "Embedded Coder Documentation." https://se.mathworks.com/help/ecoder/ (accessed 18.05, 2021).

[12] S. B. Niku, *Introduction to Robotics: Analysis, Control, Applications* Second ed. John Wiley & Sons, 2011.

[13] S. E. Pedersen, *Teknisk formelsamling med tabeller*, 7. utg., bokmål Svein Erik Pedersen ... [et al.]. ed. Oslo: Universitetsforl., 1998.

[14] TRINAMIC Motion Control GmbH & Co. KG, "QMOT QSH6018 MANUAL v 1.08," ed. Hamburg, Germany, 2014, Available: https://www.trinamic.com/fileadmin/assets/Products/Motors_Documents/QSH6018_manual.pdf.

[15] TRINAMIC Motion Control GmbH & Co. KG, "TMC5160 BOB Description," V1.10 ed, 2019, Available:

https://www.trinamic.com/fileadmin/assets/Products/ICs_Documents/TMC5160-BOB_datasheet_Rev1.1.pdf.
[16] TRINAMIC Motion Control GmbH & Co. KG, "TMC5160 / TMC5160A DATASHEET Rev. 1.14," ed. Hamburg, Germany, 2020, Available: https://www.trinamic.com/fileadmin/assets/Products/ICs_Documents/TMC5160A_Datasheet_Rev1.14.pdf.
[17] PJRC. "Teensy® 4.1 Development Board." https://www.pjrc.com/store/teensy41.html (accessed 12.03, 2021).
[18] Broadcom, "AS22 Series, Mineature Kit Encoder (Data sheet)," ed, Available: https://docs.broadcom.com/doc/AS22-Kit-Encoder-DS102.
[19] mjs513 (github.com user name). "Teensy-4.x-Quad-Encoder-Library." github.com. https://github.com/mjs513/Teensy-4.x-Quad-Encoder-Library (accessed 13.05, 2012).
[20] M. Magdy, A. El Marhomy, and M. A. Attia, "Modeling of inverted pendulum system with gravitational search algorithm optimized controller," *Ain Shams Engineering Journal*, vol. 10, no. 1, pp. 129-149, 2019/03/01/ 2019, doi: https://doi.org/10.1016/j.asej.2018.11.001.
[21] M. W. Spong, S. Hutschinson, and M. Vidyasagar, *Robot modeling and control*. Hoboken, N.J: Wiley, 2006.
[22] A. Deriglazov, *Classical Mechanics: Hamiltonian and Lagrangian Formalism*. Berlin, Heidelberg: Berlin, Heidelberg: Springer Berlin / Heidelberg, 2010.
[23] F. Irgens, *Formulas in mechanics : statics, strength of material, dynamics, fluid mechanics* (Formulas mechanics). Trondheim: Tapir Academic Press, 2008.
[24] S. Andersson, A. Söderberg, and S. Björklund, "Friction models for sliding dry, boundary and mixed lubricated contacts," *Tribology International*, vol. 40, no. 4, pp. 580-587, 2007/04/01/ 2007, doi: https://doi.org/10.1016/j.triboint.2005.11.014.
[25] D. J. Tritton, *Physical Fluid Dynamics, second edition*. 1988.
[26] B. N. Biswas, S. Chatterjee, S. Pal, and S. Mallick, "ON PERIODIC MOTION OF SIMPLE PENDULUM: "AND YET, IT MOVES."," *IJP*, vol. 1, 02/01 2012.
[27] R. Petrella, M. Tursini, L. Peretti, and M. Zigliotto, "Speed measurement algorithms for low-resolution incremental encoder equipped drives: a comparative analysis," in *2007 International Aegean Conference on Electrical Machines and Power Electronics*, 10-12 Sept. 2007 2007, pp. 780-787, doi: 10.1109/ACEMP.2007.4510607.
[28] M. S. Grewal and A. P. Andrews, *Kalman filtering : theory and practice using MATLAB*, 4th ed. ed. Hoboken, New Jersey: John Wiley & Sons, 2015.
[29] S. L. Brunton and J. N. Kutz, *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control*. Cambridge: Cambridge University Press, 2019.
[30] C. K. Chui and G. Chen, *Kalman Filtering: with Real-Time Applications*, 5th ed. 2017 ed. Cham: Cham: Springer International Publishing.
[31] MathWorks. "Extended and Unscented Kalman Filter Algorithms for Online State Estimation." https://se.mathworks.com/help/control/ug/extended-and-unscented-kalman-filter-algorithms-for-online-state-estimation.html (accessed 22.04, 2021).
[32] B. N. Biswas, S. Chatterjee, S. Mukherjee, and S. Pal, "A DISCUSSION ON EULER METHOD: A REVIEW," *Electronic Journal of Mathematical Analysis and Applications*, vol. 1, pp. 294-317, 06/19 2013.
[33] T. B. Foss and A. N. Heirung, *Merging Optimization and Control*. NTNU Norwegian University of Science and Technology, 2016.

[34] H. G. Malkapure and M. Chidambaram, "Comparison of Two Methods of Incorporating an Integral Action in Linear Quadratic Regulator," *IFAC Proceedings Volumes*, vol. 47, no. 1, pp. 55-61, 2014/01/01/ 2014, doi: https://doi.org/10.3182/20140313-3-IN-3024.00105.

APPENDIX

Digital Appendix

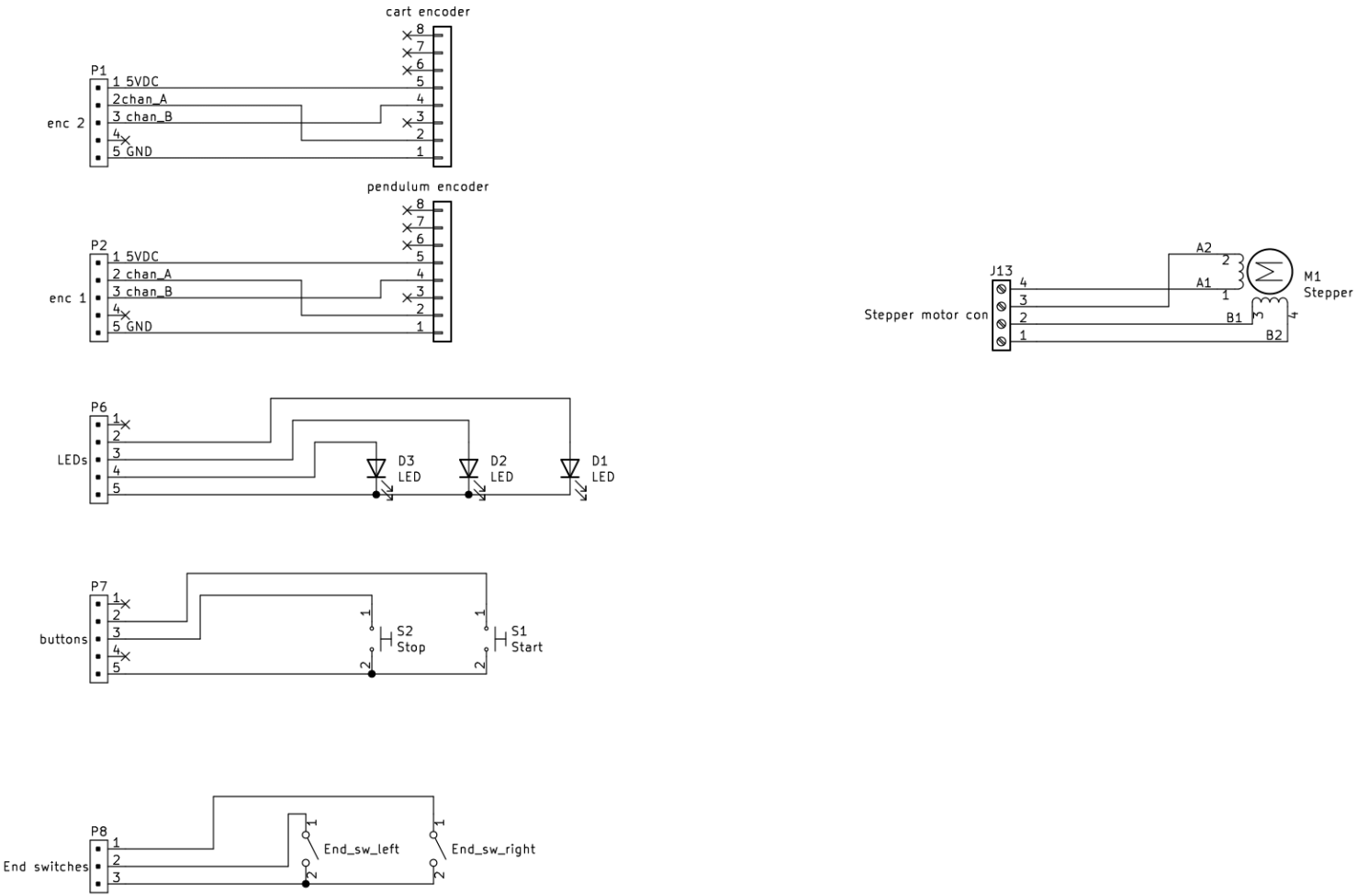
The digital appendix delivered with this thesis contain:

A demonstration video. This can also be found at
https://www.youtube.com/watch?v=hghaIoHLJYs
The code used on the microcontroller.
MATLAB/Simulink files used for the final simulation of the system.

Thesis Appendix

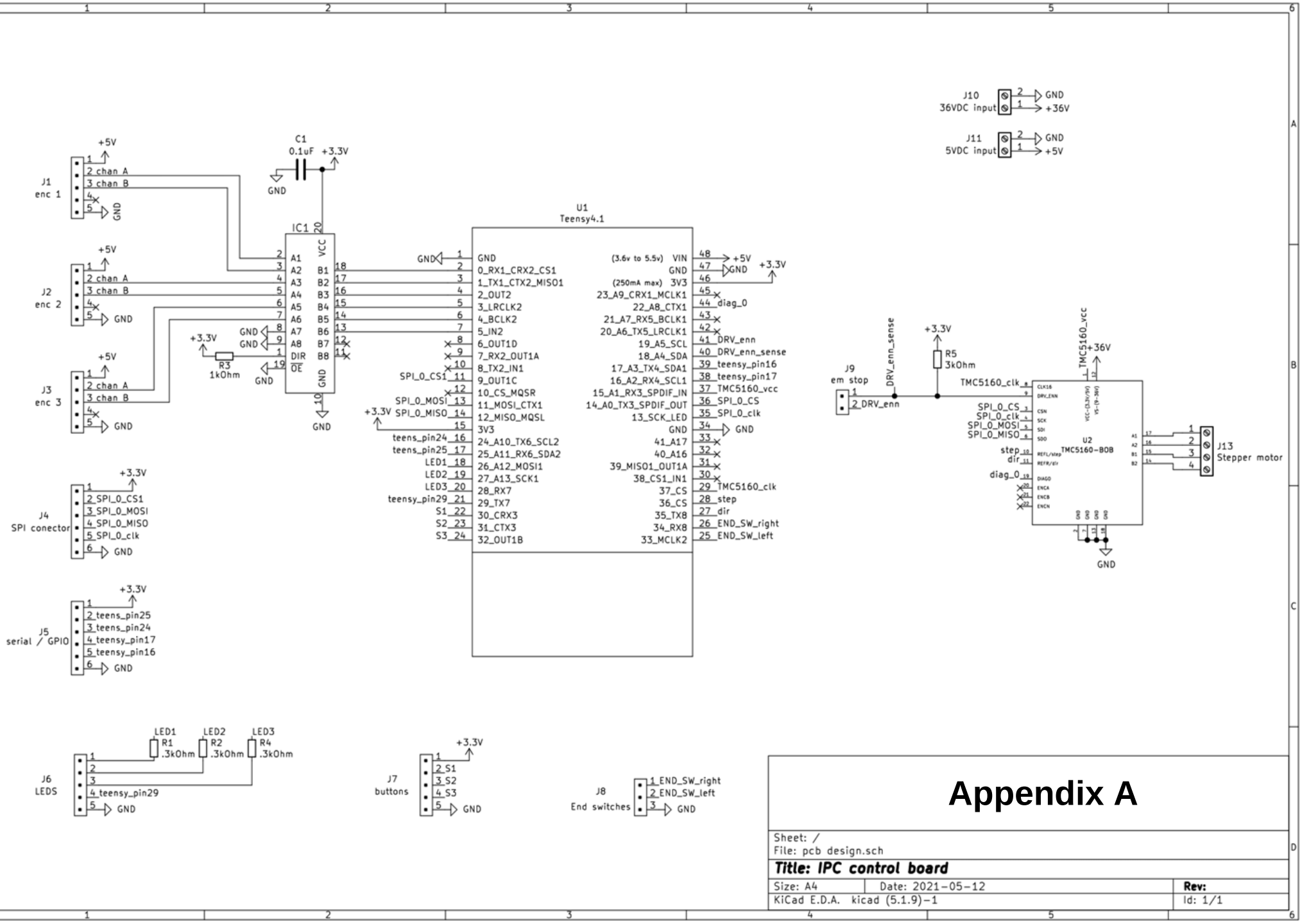
The appendix included in the thesis are:

A. Electrical drawings of the circuit board and connections to components.



Appendix A

Sheet: / File: wiring.sch		
Title: IPC model wiring diagram		
Size: A4	Date: 2021-05-12	**Rev:**
KiCad E.D.A. kicad (5.1.9)-1		Id: 1/1



Appendix A

Sheet: /		
File: pcb design.sch		
Title: IPC control board		
Size: A4	Date: 2021-05-12	**Rev:**
KiCad E.D.A. kicad (5.1.9)-1		Id: 1/1