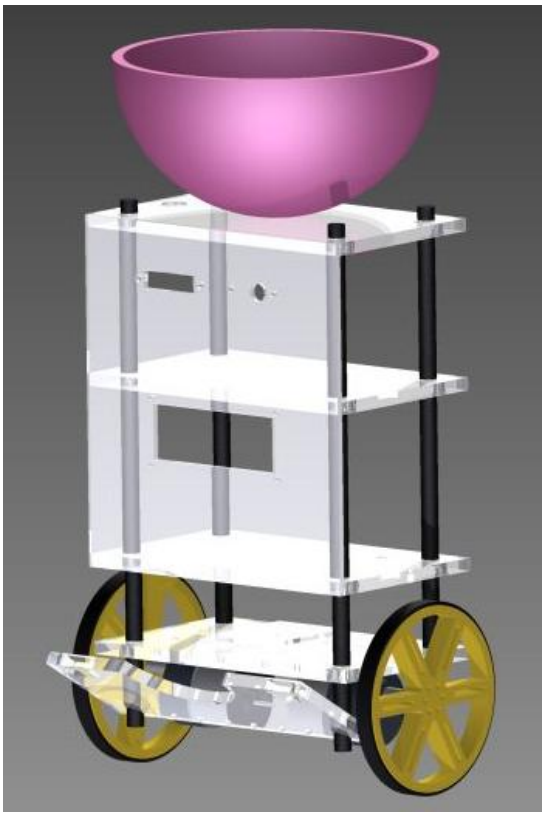
CHALMERS



Autonomous balancing robot

Design and construction of a balancing robot

Master of Science Thesis in the Master Degree Programme, Mechanical Engineering

CHRISTIAN SUNDIN
FILIP THORSTENSSON

Department of Signals and Systems
CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg, Sweden, 2012

Report No. EX060/2012

REPORT NO. EX060/2012

Autonomous balancing robot

Design and construction of a balancing robot

CHRISTIAN SUNDIN
FILIP THORSTENSSON

Department of Signals and Systems
CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg, Sweden 2012

Autonomous balancing robot

Design and construction of a balancing robot

Technical report no EX060/2012
Department of Signals and Systems
Chalmers University of Technology
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Cover: CAD drawing of the robot.

Chalmers Reproservice
Göteborg, Sweden 2012

Abstract

This projects purpose was to design and build a two wheeled upright robot. The robot was designed for use on display tables at exhibitions. It has visible components and features some functions, like the display and some sensors, whos task is to draw interest from the surroundings. It interacts in a small extent to the surroundings by using a distance sensor in combination with a temperature sensor which makes it possible to distinguish a living being from an object. The robot also has a bowl on top for carrying a load. The budget of the project was set to $5000SEK$ which was sufficient, with some money left, to complete building the robot.

A mathematical model was made for simulations and to test and dimension the controllers for standing upright and for movement. Both a PID and a LQG-controller were implemented and tested on the robot as well as different filters. The PID controller was only used for standing upright. For movement and standing upright a LQG controller was used. The robot has no sensors to get the speed of the wheels so for the LQG controller a Kalman observer was designed. For sensor fusion between the accelerometer and gyro both a complementary filter and a Kalman filter were tested. The ultimate choice of filter was the Kalman filter which is an adaptive filter that can compensate sufficient for both linear movement and noise of the accelerometer as well as estimate and compensate for the gyros drift and bias. Because the intention for the robot was to move on a table, two sensors were implemented to detect edges of the table and make the robot stop. A temperature sensor and a distance sensor in front of the robot were implemented to let the robot interact with the surroundings together with a display. This project also served as a feasability study to investigate if more advanced control systems could be implemented on a small Arduino microcontroller board, a board built with a lot of simplifications for easy use by a common person for hobby purposes.

Unfortunately it was not possible to implement the LQG controller with a Kalman observer so it was not possible to make it move. An analysis was carried out to narrow down why the LQG-controller failed when implemented. The results however are ambiguous due to several possible reasons why the LQG failed, but the one likely reason is that the Arduino is not fast enough to do the calculations for the LQG controller in the required loop time. The controller implemented on the robot is a PID controller which can balance the robot. The controller manages both instant fillup as well as instant depletion or someone picking candy from the bowl.

Acknowledgments

This project serves as a master thesis at the master program Systems, Control and Mechatronics at Chalmers University of Technology. The work was carried out at the department of Embedded systems of ÅF Technology in Gothenburg which funded the robot. Thanks goes out to the examiner at Chalmers Maben Rabi and the supervisor at ÅF Per Örbeck who both supervised the project and to the people at ÅF Technology department. Also thanks to Mikael Arvidsson for reflecting ideas.

Key abbreviations

ADC	Analog-Digital Converter
CPU	Central Processing Unit
DMP	Digital Motion Processor
I/O	Input/Output
IDE	Integrated Development Environment
I^2C	Inter-Integrated Circuit
IMU	Inertial Measurement Unit
LCD	Liquid Crystal Display
LiPo	Lithium Polymer
LQG	Linear-Quadratic-Gaussian
LQR	Linear-Quadratic Regulator
PID	Proportional Integral Derivative
PWM	Pulse-Width Modulation
RGB	Red Green Blue
SCL	Serial Clock Line
SDA	Serial Data Line
SPI	Serial Peripheral Interface
UART	Universal Asynchronous Receiver/Transmitter

Contents

1 Introduction

1.1 Statement of technical problem

The inverted pendulum is a common problem to solve in control theory. The pendulum is usually mounted on a cart and is balanced by controlling the movement of the cart. This setup can be varied in many ways to make the sytem more complex and it can be controlled in many ways. The company Segway introduced their solution of a transport vehicle in 2001, the Segway Personal Transporter, which is based on the same dynamics as the inverted pendulum. It has two wheels with a platform between them where the passenger stands. The passenger then acts as the pendulum to be balanced and the passenger runs it by leaning forward or backward. Since Segway introduced it's solution the area has gained more momentum both through public interest and in the transportation field.

This project aimed to construct a robot that works as a small, autonomous, segway. It was designed to be small to fit on a table so that people more eaily would see it, and built with a combination of bright colors and see-through glass to make it possible for spectators to see all components that the robot consists of.

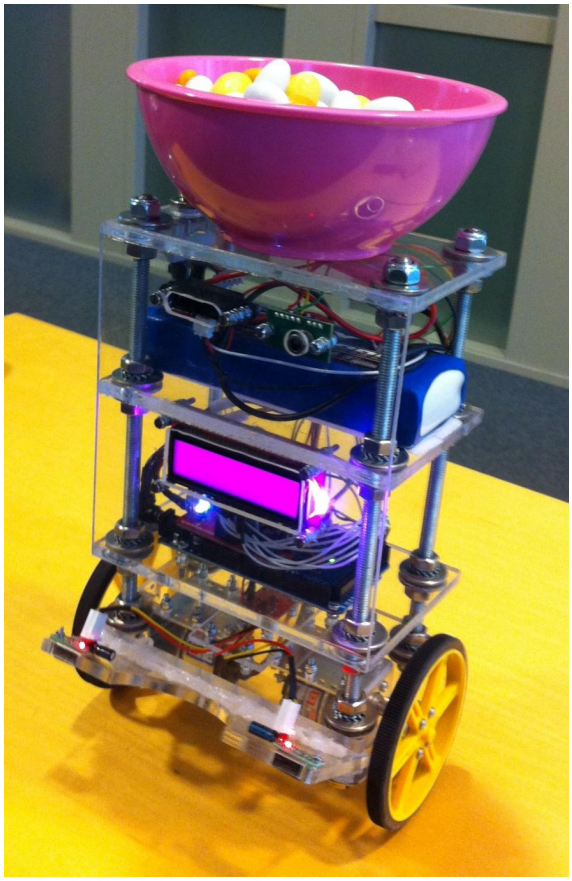


Figure 1: The robot loaded with candy.

1.1.1 Purpose and goals

The main purpose was to design and construct a robot which will balance on two wheels and it should be able to carry a load like a small bowl of candy or similar. The robot should be able to work on its own without any supervisor. After the start button is pressed the robot should be autonomous. It should also be small so it doesn't require much space, is easy to carry and is flexible to display at the small space available at an exhibition. It should be easy to see the components to arouse interest for spectators. The goals are as follows.

The robot should:

- be able to balance on two wheels,
- be able to carry a small load,
- be small so it is easy to carry,
- be stable and not make any large unwanted movements,
- be able to move on a small table without falling off the edges,
- be able to detect and interact with people,
- cost less than $5000SEK$.

To further increase the interest of the expected audience it should be built from parts common for hobby purposes that are easy available for the common person.

1.1.2 Delimitations

For the parts chosen and for the project to be achievable in the given time scope following constraints have been set.

- **Human detection**
  Because human detection is a very complex area it would be beyond the scope of this project to use a camera and image processing to achieve this goal. So the constraints on human detection was that the robot will use a distance sensor to detect objects in front and stop for a given limit. Then to determine whether it is a human or a wall in front of it it should use a infrared temperature sensor and respond accordingly.
- **Table constraints**
  The table should not be of any black or nonreflecting color since then the robot's distance sensors for detecting edges of the table will not work. It will also be horizontal and stable which means it will not move when the wheels are rotating. The table will also not be slippery.

- **Sensor constraints**
  The distance sensors to the table will work for distances between 0 and 10cm. They are mounted at 6cm from the table but the distance will change when the robot balances and tilts. The infrared sensor for human identification will work between 10 and 80cm and will not detect objects of black color or person wearing black clothes.

1.2 Method

An early design was developed and different subsystems were identified. This way, it made it clear that all functions were covered in the solution. For some parts many alternatives were available, such as when choosing electronic hardware, the Kesselring matrix was used [6]. The control systems were developed using model-based design and by simulating the system using Matlab and Simulink before implementing it on the robot.

The design of the robot was made by first analyzing the dynamics of the inverted pendulum so that it had a good design for making it easier to control from the beginning. To achieve this a mathematical model was built by using free body diagrams.

Having good knowledge of the dynamics of the system the mechanical design was made with help by using the CAD-software Autodesk Inventor. Drawings of the complete mechanical solution is found in appendix B. The production of the mechanical construction were partly made at Chalmers Robot Societys workshop and partly in ÅF:s workshop. All electronic hardware were put together in ÅF:s workshop as well as the final assembly of the system. Figure 1 shows a picture of the robot with candy in the bowl.

The original plan for the project was to use a small developmentboard for hobby purposes that run at 80MHz clockspeed, a common board of the type that have gained a lot of popularity in later years, and should be up for the task. But unfortunately the board broke during the project and had to be replaced. But by than the board was not available for a reasonable time, it was sold out, so to be able to get the replacement in time, stick to the budget and stay to the plan to use a developmentboard for hobbypurposes a Arduino Mega2560 [1] board was chosen instead. This board is the original inspiration board to the first used. The main difference is that the Arduino CPU only runs at 16MHz and is of AVR type instead of ARM.

The microcontroller comes with a preloaded bootloader that automatically starts everytime the board is powered. So when the board starts up it will first read any global variable declarations and then run a short part called ”setup”. In the setup communications such as I^2C and sensors are initiated. After that it will go into the ”main” loop where the full program is written. It will run through ”main” as an infinite loop from top and down. It is not possible to use any other operating system like a realtime operating system (RTOS) such as TinyOS. The code was developed using the procedural programming paradigm. The infinite loop allows

for interrupts caused by external signals or timers. The software was developed as the project went on. When a new part, such as the IMU, arrived a small testing program were written for testing the device and learn how it operates. The program was then, when possible, optimized for reuse and written as function blocks. This made it possible to use the same program to test other devices as well, such as the thermometer sensor which used the same communcation as the IMU, without the need of writing a new program. Also it gave information of how the signal looked like and gave a preview of which sensors that would probably need some filtering and also it meant that when the final assembly was made and the full program were to be written for the robot all the functionblocks were already written and ready to be configured. For the final software structure a flowchart was made for the full system as well for the subfunctions.

1.3 What has been done before

At ÅF Technology a previous master thesis has been done where a robot was designed and constructed using open source real-time operating system. The goal was to stabilize an inverted pendulum on two wheels. One of the main tasks of that project was to use an open souce Linux distribution. The robot was controlled by an operator who controlls it via Bluetooth from a mobile phone or a laptop PC. LQR controller and an observer were used to be able to balance and move the robot [2]. There are some differences between the two works. While the previous robot, hereby named the Råfbot, was only moving when actively controlled by an user the new robot was designed to be completely autonomous and interact through sensors and a display with the surroundings. Also as mentioned the Råfbots main goal was to run on a Linux-system using a $700MHz$ processor while the new robot investigated if a similar controller could be implemented on a smaller hobby purpose developmentboard running at $16MHz$. The new system should also be a physically smaller and more compact system.

Balancing robots is a common project to build using the Arduino board. But it has not yet been seen documented to be used for a LQG controller with an observer, only for a PID controller and solving the movement control by using encoders. So it is not clear if the Arduino has been used for LQG with observer and if it is possible for the Arduino to handle the heavier calculations in a sufficient speed. This project aims to implement an LQG controller and use an observer to observe the wheel velocity. The reason for this is to not have to spend money on encoders and to make the project more challenging.

1.4 Why is it an interesting project?

Because of the European Union's regulations and demands for reduction of CO_2 emmisions and cities, like Stockholm and Gothenburg, wish to reduce city traffic and CO_2 emmisions the transportation sector puts more money than ever into developing environmentally friendly vehicles. This includes to look for different solutions to transportation, such as the segway. To reach the demands the companies needs

knowledge in the field of hybrid vehicles technology or fully electric vehicles. And that puts a demand on new engineers today to have a wide set of knowledge in both mechanics, power electronics, vehicle dynamics, software, microcomputers, automatic control and different communication techniques among other. This is backed up by, for instance, a project carried out at Volvo Cars during 2012 to investigate the possibility of the use of two wheeled load carrying trucks in cities [4].

ÅF Technology is a consulting company that strives to be in front of competitors when it comes to knowledge and new technology. They are also a company that is a regular visitor at different technical exhibitions and Universities in order to promote and recruit. For this purpose they would like to have something eye-catching to show that draws interest and shows that ÅF Technology is a company that has a lot of knowledge and interesting projects.

2 The mechatronic system

2.1 Mechanical system

The general design of the robot was a rectangular body on two wheels. The wheels are placed parallell to each other. On top of the body a bowl was placed for carrying loads. The battery was placed as high as possible on the body to place the center of gravity as high as possible which is desirable. The body can be described as bookshelf were the components were placed on four different shelves. Figure 2 shows CAD drawings over the robot from front and side view.

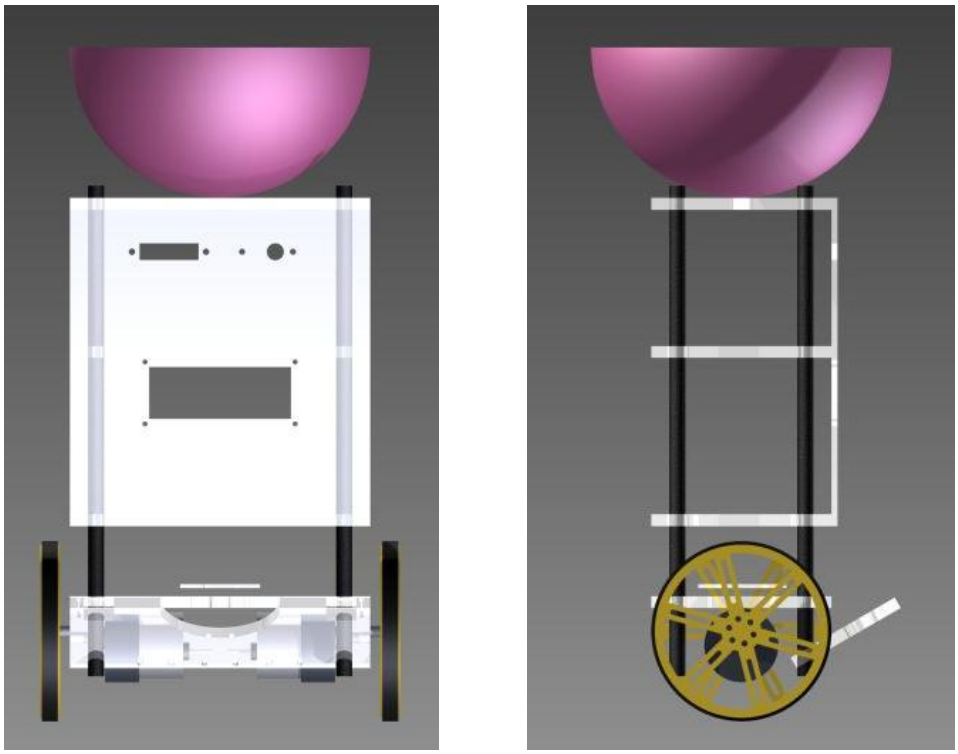


Figure 2: The robot from front and side view.

2.1.1 Main frame

The main frame of the body consists of four threaded steel rods that can represent the four legs of the shelf. The shelves was made out of $6mm$ thick plexiglass. On the bottom shelf there was a smaller plane of $3mm$ thick plexiglass mounted across the shelf to hold the IMU. The reason for having a separate smaller plate for the IMU was because of simplification of placement and connecting the wires. On the next shelf the microprocessor and motordriver was placed so it was close to the center of the robot for easy wiring. Shelf three holds the battery and is placed as high it can be underneath the top shelf. On the top shelf the bowl made out of hard plastic was placed together with the powerswitch. Placed in the front of the robot glued onto the edge of the three top shelves sits a $3mm$ thick plexiglass plate where the sensors for distance in front and temperature were mounted together with the

display. Beneath the bottom shelf mounted on the two front rods and to an angle of the rods and the shelf sits another smaller plane of $6mm$ thick plexiglass where the sensors for edge detecting were placed.

2.1.2 Wheel base

For wheels two Pololu $90 \times 10mm$ were choosen because of the diameter on the wheels were good by consideration of the motors angular velocity and the desired speed of the robot. The wheels were made out of light plastic with a rubber tire and atached to the motor axis with an aluminium hub. The motors were then mounted on the bottom shelf between the front and the back rods.

2.2 Electrical system

Here the complete electrical system is described part by part. To get a full overview of the system see appendix C.

2.2.1 Microprocessor Arduino

As the main CPU board a small Arduino board was chosen. This was a small development board created for experimenting and hobby purposes. It had 54 I/O pins which includes 14 PWM outputs, 16 analog inputs, 4 UARTs with 1 I^2C port and 3 SPI ports available. The card consists of a ATmega2560 microcontroller with a clockspeed of $16Mhz$. It requires an input voltage between $7 - 12V$ for optimal functioning and can be powered and be programmed through a USB-cable. It also comes with a programming IDE and a preprogrammed bootloader. This makes the board good for getting started quickly and be able to reprogram the board fast, however it comes with the cost of limiting programming possibilities and forces the programmer to use their very limited programming IDE. The IDE uses both Arduinos adaptation of the processing language and/or c++ for writing code. Processing was a simplified version of c++ with that had some of the math functions and some classes included. Some drawback with Arduinos IDE were that it lacks line-numbering, it did not give full error messages and it was not possible to get a fast overview of a functions syntax. These drawbacks were not known when considering to use Arduino.

2.2.2 Motor driver

To power the motors a motor driver was needed and the one of choice was the Ardumoto for two $12V$ DC-motors. It was choosen because it fits easily on the Arduino-card and the only connections needed except for the pins already connected through the shield were the main power to the motors of $12V$ and two motors. It was based on the L298 H-bridge and could supply up tp $2A$ per channel. It worked with both $3.3V$ and $5V$ logic, where $5V$ was the level in the system. The motors were controlled by setting a digital 0 or 1 on the direction pin for deciding which direction the motor would go and by supplying a PWM signal for determine the speed of the motors where 255 was full power of supply voltage V_{in} and 0 is completely turned

off. This allows for 256 different levels for back and forth of the motor wich gives a total span of 511 speeds. It also had room for attaching additional pins or devices on the board as an small perfboard making it good for attaching devices such as the level converter or additional groundpoints or $5V$ terminals. The drawback of the Ardumoto was that the L298:s makes some high pitch noise when active.

2.2.3 Level converter

As the system uses I^2C devices of different logic levels a level converter was needed to make the I^2C network possible. The one of choice was a small device from Sparkfun which has two channels meant for SPI. It can be used for I^2C as well, but since SPI needs only one bidirectional line only one of the lines of each channel is bidirectional so to use it for one channel of I^2C both channels of the level converter was needed. This is not a problem even if the system would have more than two I^2C slaves because the level converter was only needed to translate the level of a branch of the network and not for each device.

2.2.4 Accelerometer and gyro

The IMU choosen consist of a 3 axis accelerometer and a 3 axis gyroscope. It has a onboard processor for sensor fusion to get a good estimate of the angle called "digital motion processor", DMP provided by Invensense. However since good sensor fusion algorithms are hard to obtain all providers guard their algorithms well. So to get use of the DMP you must use their software. But to do this the code must be ported to the Arduino enviroment and then the problem would be a software engineering problem. The problem with porting the code is that you don't actually get access to the code, you only get a AVRStudio target file for a specific processor. Also, then there would not be any control over the sensior fusion, the sensor readings would have to be trusted as they were. But since the sensors were possible to access without using the DMP it was choosen to do so.

- Accelerometer ADXL345

  The accelerometer measured the acceleration in x-, y- and z-directions for the robot. Two of these were used and the angle ψ of the pendulum were calculated by the arctan function. This sensor outputs an analog signal, a voltage, that corresponds to an acceleration. To communicate with the ADXL345 I^2C protocol was used and it returns first low bytes followed by high bytes. It measures the acceleration in units of gravitational pull, g, but since the angle was calculated using arcus tanges and the quote between two axis the scaling to g:s or degrees wasn't necessary since the ratio between them were independent of unit. The noise of the accelerometer were modeled as white noise. The range of the accelerometer could be set between $0-2g$ up to $0-16g$. To achieve the highest resolution the range should be set as low as possible. Since the accelerometer only should be used to estimate the angle the maximum output of the accelerometer should be expected to be $1g$. Therefore it was set to $0-2g$ since it was enough for the purpose.

- Gyro IMU3000

  The gyro was a IMU3000 unit that measured the angular velocity around x-, y- and z-axis in $^{\circ}/s$ although only one axis was actually used for the robot. It communicated through I^2C protocol and had the ADXL345 connected as a slave making it possible for the IMU3000 to act as a master and read the ADXL345. But when this function was used the ADXL345 value was first stored in an internal memory so to be sure to control when to read the ADXL345 and be sure of getting the right value the IMU3000 was set to passthrough so that the ADXL345 could be accessed at it's own adress as a separate I^2C unit. The noise of the gyro were modeled as white noise. The range of the gyro could be set to between $0 - 250°/s$ up to $0 - 2000°/s$ and to achieve the highest resolution it should be set as low as possible so it was set to $0 - 250°/s$.

Both the accelerometers angle and the gyros angle were used to get the pendulums angle of the robot because the accelerometer has a better measurement for small variations and the gyro has a better measurement for large variations and also the gyro has some drift caused by sampling, so to filter the gyro signal and get the correct angle the accelerometer values were needed.

2.2.5 Distance sensors

In the system two different types of proximity sensors was used. Two for detecting edges and one for measuring the distance to objects in front of the robot.

- Table detection: Sharp GP2Y0D810Z0F infrared digital 2-10cm

  The edge detecting sensors are of infrared type, the same type that can be used for line following robots where the robot uses two sensors to scan for the line. They have a infrared light emitting diode and a photo diode. They are of short distance types measuring distances between $2 - 10cm$. They use triangulation to measure the distance which makes them robust against disturbances as sunlight. Because they rely on reflecting light they can be sensitive to angles between the measured surface and the sensor. Also for them to be able to read the distance the surface needs to be of a light reflecting material making black an unsuitable color for operation. The onboard signal processing unit samples the sensor with a speed of $400Hz$ and returns a logic 1 when it fails to detect any surface, that is the distance is either below $2cm$ or above $10.25cm$ or a logic 0 otherwise making them good for detecting e.g. the edge of a table. Because it only returns a high or a low value and uses onboard signal processing the actual signal read by the microprocessor is close to noise free.

- Human detection: Sharp GP2Y0A21YK0F infrared digital 10-80cm

  For detecting objects in front of the robot such as walls or people another infrared proximity sensor was used which outputs an analog voltage. This one

measures distances in the span $10 - 80cm$ by sending out a beam of infrared light and then by using the reflection using an onboard signal processor it calculates a responding output voltage. The voltage can than be read by the ADC and used to calculate a distance to the object detected in centimeters. Because of sampling and imperfections of reflections and ambient light such as sunshine the sensor has some noise that needs to be filtered to accurately measure distance. Of course the measurement will not be fully accurate since the robot itself and also the target will be moving, but this is not considered as a problem since the actual distance itself was not important since instead of using actual distance a threshold was set for when the distance was small enough for the robot to respond, whether it was a wall or a person it sees.

The sensors had an input filter capacitor of $10\mu F$ as recomended in the datasheet. This was because of the sensor can draw high peaks current that can damage the sensor and also it improves the readings of the sensor by reducing noise.

2.2.6 Temperature sensor

For human identification a temperature sensor was used. For this purpose the TPA81 was chosen which is a thermopile array that consist of 8 pyro-electric sensors and 1 onboard ambient temperature sensor. It detects infra-red light in the range $2-22\mu m$ and has a field of view of 100°, but with a use of silicon lens this field of view is reduced to 5.12° by 6° and the thermopile gets a total field of view of 41° by 6°.

The range $2 - 22\mu m$ is the wavelength of radient heat, the same waveband that are used in burglar alarms or lightswitches activated by movements. These sensors can only detect changes in heatlevels so if the system would be completely static it would not read the temperature of a static heatsource. But since both the system will be moving as well as the subjects that are to be detected they are good for the purpose. It works as a thermoucouple which means that a conductor that is subjected to thermal gradient will generate a voltage. Thermopiles will not return absolute temperature, but generate an output that responds to a change in temperature. So by using an onboard sensor that senses the ambient temperature by traditional means, like resistance thermometer, the device can calculate what temperature is read by the thermopile. So by sampling the voltage returned by the sensor the device returns a digital number responding to the themperature. Because the way the device works there is a lot of noise sources such as fluctuations of temperture in the room, calcutation errors in temperature and quatization errors so there is need for filtering.

A human in a room of $20°C - 22°C$ appears around $29°C - 33°C$. The TPA81 communicates through I^2C and returns 1 byte containing the temperature in $°C$.

2.2.7 Battery

As a power source for the robot a 3 cell $5000mAh$ Lithium-polymer battery was chosen. The reason for using 3 cells is because 1 cell of a LiPo battery has the nominal voltage of $3.7V$ and a maximum voltage when fully charged of $4.2V$. The

Arduino reguires an input voltage between $9 - 12V$ to operate optimal and also because of the motors being rated at $12V$ so this means that there was no need for an external voltage regulator. Since the robot was supposed to be able to operate actively for at least 8 hours before recharging $5000mAh$ was chosen to guarantee this. The consumption was estimated as the consumed currents in normal operation which gave

$$I_{tot} = \sum I_{parts} \tag{1}$$

and as the battery had a discharge rate of 20C which means $20 \times C$ where the unit of C is $1/mh$ which gives that the battery can deliever a maximum of $100A$ for 1 hour or with the estimated current consumption it can run approximately the time t between charging where t is given by

$$t = \frac{I_{tot}}{C} = 35.062h \tag{2}$$

which shows that the battery was largely overdimensioned. The reason for this was mainly due to uncertainity of the power consumption of the system, prevent the battery to unexpected run out of charge causing the robot to fall off the table and to be able to guarantee the robot to work for several days at exhibitions even if the charger is left behind. The discharge curve of the battery can be seen in appendix C which shows that the battery charge is almost constant until it is discharged and the voltage suddenly drops.

2.2.8 Motors

Two GHM-16 $12V$ DC-motors, which is a two pole brushed DC-motor, were choosen to get the desired torque and speed. They have a no-load speed of $200rpm$ and rated torque of $0.078Nm$, which is enough for the small system it should drive. They are geared 30:1 which caused a small backlash. As usually for small, low priced DC-motors there was no complete datasheet so many motor constants were unknown. But in the specifications available the ideal speed/torque, current/torque, power/torque and efficiency/torque-curves were given along with rated torque, rated current, rated speed, no-load speed and no-load current. Since they are a 2 pole DC-motor the armature resitance can be measured directly by with a multimeter assuming the inductance can be neglected, which was true for small DC-motors and low power applications. So by using ideal curves and the ratings for the motor together with the motor equations the constants were calculated.

2.2.9 Display

For use during the development and also for some interaction a LCD withc built in driver was added. It made it possible to read data while running the robot without connecting to a computer. It had two rows and 16 columns which each makes it possible to write a maximum of two 16 letter long words at once without rolling the display. The Arduino has a ready to go library suitable for this display which made it a good choice for easy use. To use it however it was needed to figure out where each pin connection would go since once again for low price parts there was

no datasheet available to explain how the LCD driver was configured. The LCD also has a RGB-backlight wich is PWM-controllable which makes the display useful for reacting to different colors as well as printing out messages. The backlight of the display was set to react to the temperture read in front of it so if a person were standing in front of it the display would turn red as in figure 3.

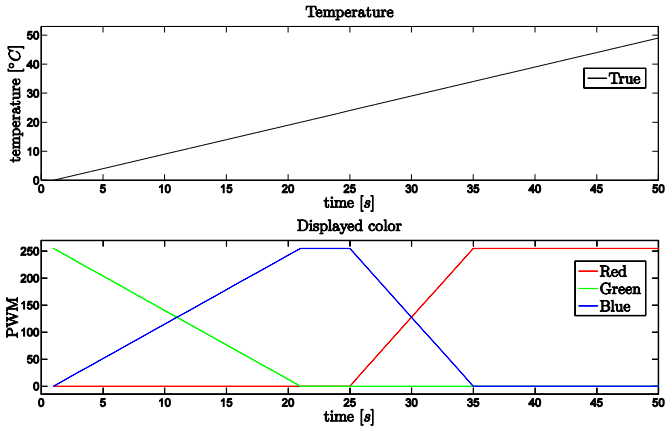


Figure 3: Colour change of display.

2.3 Software

Because the system is built upon an Arduino there was no option to use any operating system so the system is designed as an infinite loop. The software was first modeled as a statemachine shown in figure 5 showing all the states and the cases leading to the states. All states in the statemachine that does not have a specific case describing the transition to the state there is a time trigger controlling the transition. Also it should be noted that the case ”Edge interrupt” is, as the name suggest, a transition that actually can occur at any point in the loop since it is triggered by an interrupt caused by the edge sensors. Statemachine was designed only for the LQG controlled system since the PID would not require more than one state due to it's lesser abilities. A simple flowchart descibing the PID system can be viewed in figure 4.

2.4 Filter

Most of the sensors used on the robot will needed some filtering. This section will briefly go through the filters used in the system.

To get a good estimation of the angle of the robot both the accelerometer and the gyroscope had to be used. The sensors have different properties that affects the angle estimation in different ways. The main bullets are listed below.

- **Accelerometer**
  The accelerometer is good for getting a good reference of the pitch angle when it is uneffected by other forces such as linear movements. To get a perfect

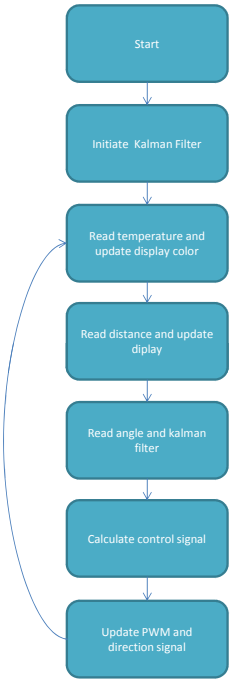


Figure 4: A flow chart describing the system with PID controller for the upright pendulum, the one ultimately used.

pitch it needs to be effected by gravity alone or otherwise it may not return the correct angle.

- **Gyroscope**
  The gyroscope can estimate the pitch angle by integration of the sensor output. This angle will not be affected by linear motions by the robots movement but since the sensor has some bias and other defects the estimation will drift and then the integrated signal will not be the true angle.

Using this knowledge different filtering can be used to get a good reliable reading of the angle. Two different filters were tried out in the system and will be described next.

2.4.1 Complementary filter

This is a filter often used in hobby applications for a good and easy fusion of the two sensors. It is a simple filter that is easy to implement, experimentaly tune and demands very little processing power. It is basically a high pass filter and a low pass

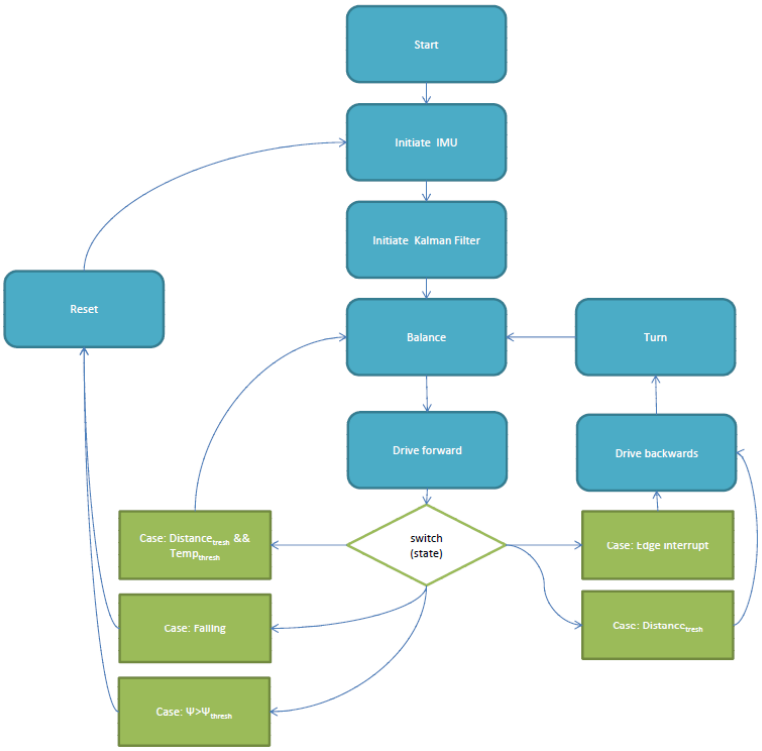


Figure 5: A statemachine of the system when using LQG for a moving robot.

filter combined where the high pass acts on the gyro and the low pass acts on the accelerometer to use the gyro for short term estimation and uses the accelerometer for absolute reference correct the estimation of the angle. The pitch angle ψ is then given by

$$\psi_k = (1-\alpha)(\psi_{k-1} + \dot{\psi}_{k,gyro}dt) + \alpha\psi_{k,acc} \tag{3}$$

where dt is the sampletime and α is a filter constant. Then by tuning the constant α until the result is good the time constant of the filter is changed and the bias of the gyro is removed. This filter is good to give a good estimate of the angle, however there are some drawbacks where the most significant is that while it removes the bias from the angle estimate it does not give any estimate of the actual bias of the gyro. So a controller using the angular rate as an input cannot use the gyro value directly because of the bias.

2.4.2 Kalman filter

A filter that does give an estimate of the gyro drift is the Kalman filter. The Kalman filter comes from optimization theory and is the otimal estimator for a system with disturbances in the system that has the characteristics of normal distributed white noise. The filter is a model of the sensors behavior and includes the gyros bias and drift meaning that it gives a good estimate of both the angle and the gyro bias which improves the measurement of the angular rate since the bias can be removed. This Kalman filter does not use the pendulum model to predict the angle or angular rate.

It only use a model of the sensors to calculate the estimations. The reason for this is to save calculations in the processor. This filter is described as

$$h_{k+1} = Eh_k + Fu_k \tag{4}$$

where h is the filter states, ψ and $\dot{\psi}_{bias}$, E and F is the filter state matrices describing the senors, which for the filter model becomes

$$\begin{bmatrix} \psi \\ \dot{\psi}_{bias} \end{bmatrix}_{k+1} = \begin{bmatrix} 1 & -dt \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \psi \\ \dot{\psi}_{bias} \end{bmatrix}_k + \begin{bmatrix} dt \\ 0 \end{bmatrix}_k \dot{\psi}_{gyro} \tag{5}$$

By using the E and F matrices and a output matrix G matrix as

$$G = \begin{bmatrix} 1 & 0 \end{bmatrix} \tag{6}$$

the filtering is done as follows.

$$\begin{aligned}
&\text{First the states are updated} \\
&h_k = Eh_{k-1} + F\dot{\psi}_{gyro} \\
&\text{Calculate predicted covariance} \\
&P = EP_{k-1}E' + Q \\
&\text{The the innovation } h_{inn} \text{ is calculated} \\
&h_{inn} = \psi - Gh_k \\
&\text{Update the prediction covariance} \\
&S = GPG' + R_\psi \\
&\text{Calculate the Kalman gain} \\
&K = EPG'S^{-1} \\
&\text{Calculate the state estimate} \\
&h_{k+1} = h_k + Kh_{inn} \\
&\text{Calculate estimate covariance} \\
&P = E'PE' - KGPE' + Q
\end{aligned} \tag{7}$$

So what is done is that first the state h_k and the covariance P is updated. Then the innovation, h_{inn}, is calculated which is the difference between the prediction and the angle calculated by the accelerometer measurement. After that the covariance S and the Kalman gain K is calculated to correct the prediction, h_{k+1}, of the state and lastly the covariance matrix P of the prediction error is calculated. R_ψ is the measurement noise covariance, the expected noise of the accelerometer, and Q is a 2×2 matrix of the process noise described as

$$Q = E\left(\begin{bmatrix} \psi & \dot{\psi}_{bias} \end{bmatrix} * \begin{bmatrix} \psi \\ \dot{\psi}_{bias} \end{bmatrix} \right) \tag{8}$$

note that here E is not the state matrix, but the operator for ”expected”. A good thing with using the Kalman filter is that it gives the optimal prediction based on previous guess and do not need the entire history of the states. Only the information from the last prediction and the new sensor readings is used to predict the next states [7]. This Kalman filter was used in combination with the PID-controller to get a good reading of the angle.

2.4.3 Butterworth filter

For the sensors used in front of the robot, the temperature and distance sensors, filtering was also needed. For the distance sensor the expected noise is of high frequency so it needed low pass filtering. The normal noise of the temperature sensor is also of high frequency character, however noise due to temperatur fluctations such as wind is of different character hard to predict. In the expected normal operation area of the robot however the room is assumed to have a close to constant temperature with no wind. Therefore only filtering considering the high frequency was made. The filter of choice was a Butterworth IIR low pass filter because it is easy to implement in code and uses low computation power and is designed to have a steep transition region and as flat frequency response as possible in the passband even for low frequencies as seen in figure 6. The transfer function of a general

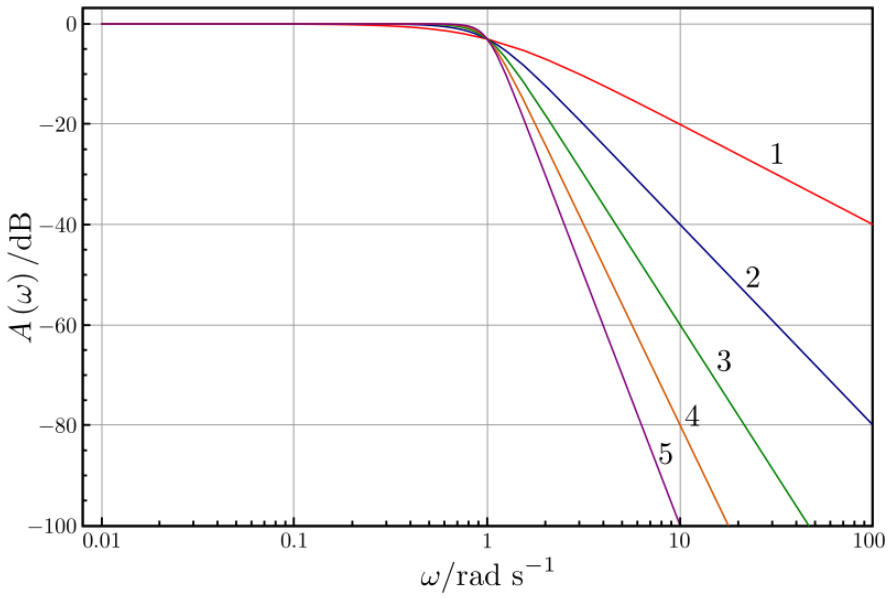


Figure 6: Characteristics of Butterworthfilters up to order 5.

Butterworth low pass filter looks like.

$$H_{filter}(z) = \frac{b_1 + b_2 z^{-1} + ... + b_{n+1} z^{-n}}{1 + a_2 z^{-1} + ... + a_{n+1} z^{-n}} \tag{9}$$

The Butterworth filter is used for the temperature-sensor and for the distance sensor in the front. Since these are no time-critical measurements there is no problem with time delays caused by filtering. Therefor the cut-off frequency could be chosen to achieve a good clean signal without any concern of instability caused by delay. The cut-off frequency was set to $1Hz$ since this gave the best result and the order of the filter was set to 1 to save computational power and because it was sufficient for to get a good result. The coefficients then became

$$\begin{aligned} b &= 0.030468747091254 \\ a &= -0.939062505817492 \end{aligned} \tag{10}$$

2.5 Communication

2.5.1 PWM

The battery delievers a close to constant voltage and to control the motors the source voltage needed to be controlled. For this pulse width modulation was used since it is a simple method with low power losses and has hardware and software support in the Arduino. The power was then drawn from the battery and switched on and off by the motorcontroller. Because of electrical inertia the voltage will not go from zero to max between each switching period but instead be close to constant of the resulting mean depending on the dutycycle D. The voltage level is described as

$$U = \frac{1}{T}\int_0^T f(t)dt \tag{11}$$

and $f(t)$ is a pulsewave described as

$$f(t) = \begin{cases} U_{max} & \text{for } 0 < t \leq D \times T, \\ 0 & \text{for } D \times T < t \leq T. \end{cases} \tag{12}$$

which then yields a voltage mean U

$$U = D \times U_{max} \tag{13}$$

The function can be seen in figure 7 [8].

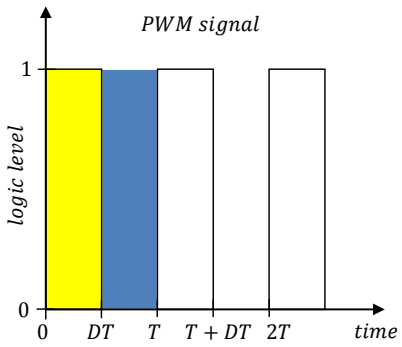


Figure 7: A graph describing the PWM signal function. The yellow block showsi when the signal is high and the blue when the signal is low.

2.5.2 I^2C communication protocol

I^2C stands for ”inter-integrated circuit” and is also sometimes referred to as ”two wire interface”. As suggested it is a communication protocol that uses only two wires to communicate and supports multiple masters and slaves. A setup of a I^2C network can look like figure 8. The two wires are bi-directional, open drain lines called Serial Data Line, SDA, and Serial Clock Line, SCL, that are pulled up with pull-up resistors [3]. It is possible to adress devices with a 7-bit, which is most common, or a 10-bit adress. The Arduino uses 7-bits adress at a speed of $100kHz$

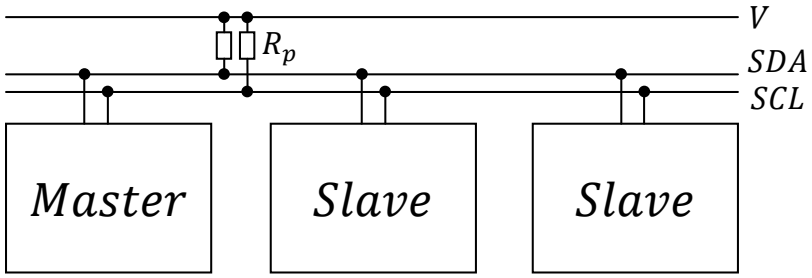


Figure 8: Example of a I^2C network.

and also have internal pull-up resitors making it unnecessary for another pair in the same network. The I^2C bus has two types of node-roles: master and slave. The master issues the clock and adresses the slaves and the slave recieves the clock and adress. To recieve new data from a slave the master first will send out an adress on the network and the slaves listens. If there is a slave with the sent adress on the network it will respond with a "Acknowledge" and master than requsts what register it wants to read and how many bytes it wants to read. Between each byte read the master sends a "Acknowledge" bit to confirm that the data was read properly. The transmission is ended by the master sends a stop-bit or, if it wants to retain the control of the network, another start bit. How messages is sent can be seen in the timing diagram in figure 9. The master first sends a startbit (S, yellow) when SDA is pulled low while SCL stays high. Then the SDA sets the transferred bit (B_i, blue) while SCL is low and the bit is received (green) when SCL goes high again. When the transfer is complete the stopbit (P, yellow) is sent by letting SDA go high while SCL is high.

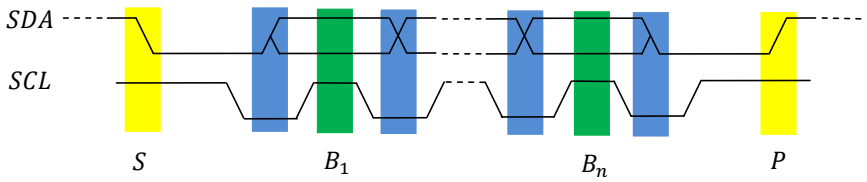


Figure 9: Example of timing diagram of I^2C communication.

The Arduino operates at a level of $U = 5V$ so the high voltage in the I^2C network is at this level which yields that for the master to be able to communicate with all devices in the network, they all need to operate at a $5V$ level. Since this is not the case as the IMU operates at a $3.3V$ level there is need for a level converter between the IMU and the rest of the network. The level converter works as a interpreter for the IMU making it possible for it to understand the master. The level must be done in such a way that the message, or pulses, is not changed as the voltage level is changed. This is done by using mosfets for switching down the levels as can be seen in appendix C.

2.5.3 ADC

To read analog sensors such as the proximity sensor in front for human detection the Arduino uses a $10bit$ analog to digital converter. This converter reads a voltage that is proportional to the reading of the sensor and, as the name suggest, converts it into a digital representation, a number that is proportional to magnitude of the voltage. The resolution, i.e the number of points N that can be represented digitally, is then determined from the number of bits, m, used in the ADC as

$$N = 2^m - 1 \tag{14}$$

which gives that the value read from the sensor will be represented by a number between $0 - 1023$. The voltage resolution, i.e how small steps of change in voltage the ADC will read, is determined as

$$Q = \frac{E_{fsr}}{2^m} \tag{15}$$

where E_{fsr} is the full scale voltage span determined from the maximum and the minimum voltage of the ADC

$$E_{fsr} = E_{max} - E_{min} \tag{16}$$

which for the arduino is $5V$ which gives a voltage resolution of $4.9mV$.

2.6 Data logging

To log data one of the UART's of the Arduino was used. The UART was converted with the onboard USB device and sent to a computer through a USB cable. Data was then displayed in the "Serial monitor" of the Arduino IDE and could be used for plotting in Matlab. The good with this way was that it was a simple and fast way to do it, but the drawbacks was that the cable had to be connected to be able to logg data which affected the system when the robot was operating. Another drawback was that the Arduinos implementation of sending data through the USB was slow, so if to much data was sent at the same time, such as readings from more than one sensor, the control loop was slowed down significantly.

3 Mathematical model

The robot's mathematical description was devided in three parts, one for the inverted pendulum, one for the wheels and one for the electrical motor system. The pendulum and the wheel have three equations each, one for rotational direction and two for x- and y-direction. The axes are fixed in the wheels and therefore follows the wheels when the robot is moving. The pendulum have ψ for angle and $\dot{\psi}$ for angular velocity and the wheels have θ and $\dot{\theta}$ for the angle and angular velocity. Table 1 shows the parameters that were used in the mathematical model.

Parameters	Explainations	Units
Pendulum		
g	Gravity acceleration	m/s^2
J_p	Inertia pendulum	kgm^2
m_p	Mass pendulum	kg
L	Distance centre wheel and centre pendulum	m
ψ	Angle of the pendulum	rad
$\dot{\psi}$	Angular velocity of the pendulum	rad/s
x_p	x-direction for the pendulum	m
y_p	y-direction for the pendulum	m
N_x	Force between the pendulum and the wheel in x-direction	N
N_y	Force between the pendulum and the wheel in y-direction	N
Wheel		
J_w	Inertia wheel	kgm^2
m_w	Mass wheel	kg
r	Radius of the wheels	m
θ	Angle of the wheel	rad
$\dot{\theta}$	Angular velocity of the wheel	rad/s
x_w	x-direction for the wheel	m
y_w	y-direction for the wheel	m
N	Normal force from the table to the wheels	N
F	Friction force between the table and the wheels	N
Electrical		
R_a	Nominal terminal resistance	Ω
k_t	DC motor torque constant	Nm/A
k_e	DC motor back EMF constant	$Vsec/rad$
T_m	Torque from the motors	Nm
U	Input voltage to the motors	V
n	Gear ratio	

Table 1: Parameters with explinations and units.

3.1 Pendulum model

Figure 10 shows the forces acting on the pendulum.

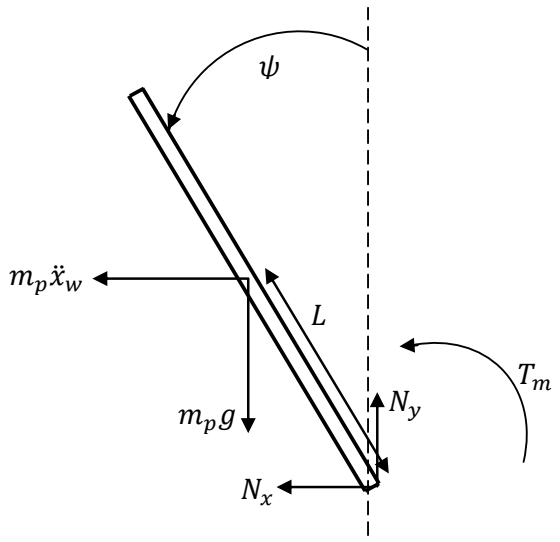


Figure 10: Forces acting on the pendulum.

The pendulum equations are as follows. Equation (17) is momentum equation counterclockwise around the mass centre of the pendulum. Equation (18) is the forces acting in x-direction and equation (19) is the forces acting in y-direction.

$$T_m + m_p g L sin\psi + m_p \ddot{x}_w L cos\psi = J_p \ddot{\psi} \tag{17}$$

$$-N_x - m_p \ddot{x}_w = m_p \ddot{x}_p \tag{18}$$

$$N_y - m_p g = m_p \ddot{y}_p \tag{19}$$

The nonlinear accelerations $\ddot{x}_p$ and $\ddot{y}_p$ has to be transposes from the X-Y coordinate system to the rotational coordinate system. First the equations for x_p and y_p are expressed

$$x_p = -Lsin\psi \tag{20}$$

$$y_p = Lcos\psi \tag{21}$$

these are derivated to get the expression for the velocities

$$\dot{x}_p = -L\dot{\psi}cos\psi \tag{22}$$

$$\dot{y}_p = -L\dot{\psi}sin\psi \tag{23}$$

and then derivated one more time to get the expressions for the accelerations

$$\ddot{x}_p = -L\ddot{\psi}cos\psi + L\dot{\psi}^2 sin\psi \tag{24}$$

$$\ddot{y}_p = -L\ddot{\psi}sin\psi - L\dot{\psi}^2 cos\psi \tag{25}$$

To find the inertia of the pendulum the body was seen as a cuboid with uniform mass distribution. And by using classical mechanics the inertia is calculated as in [5]

$$J_p = \frac{1}{12}m_p W^2 + \frac{1}{3}m_p H^2 \tag{26}$$

where m_p is the mass, W and H is the witdth and height respctively.

3.2 Wheel model

Figure 11 shows the forces acting on the wheels.

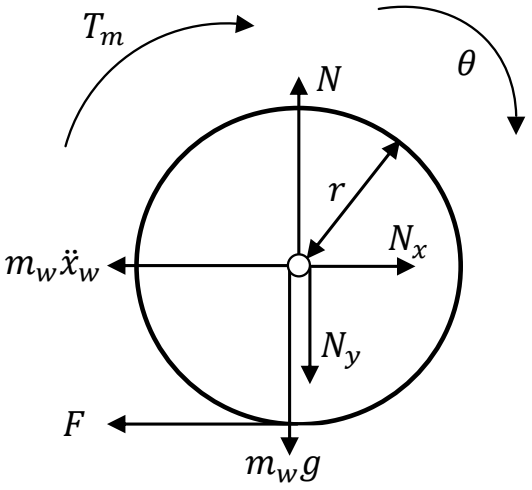


Figure 11: Forces acting on the wheels.

For the wheel the equations are as follows. Equation (27) is momentum equation clockwise around the mass centre of the wheel. Equation (28) is the forces acting in x-direction and equation (29) is the forces acting in y-direction.

$$T_m + Fr = J_w\ddot{\theta} \tag{27}$$

$$N_x - m_w\ddot{x}_w - F = 0 \tag{28}$$

$$N - N_y - m_w g = m_w\ddot{y}_w \tag{29}$$

The transportation from X-Y coordinate system to the rotational coordinate system are

$$\ddot{x}_w = \ddot{\theta}r \tag{30}$$

$$\ddot{y}_w = 0 \tag{31}$$

And in the same fashion as the pendulum but using the formula for a circle the inertia is found to be

$$J_w = \frac{2r \sin 2\alpha}{3\alpha} \tag{32}$$

where r is the radius and α is the angle which for the full circle is 2π.

3.3 Electrical motor model

For the system two small DC-motors were used. To use them in the mathematical model of the system a mathematical description of the motors were needed to provide a relationship from input voltage, the control signal, and the torque delievered from the motors. By applying Kirchoffs laws for the electric circuit, figure 12, describing the motor equation (33) is attained.

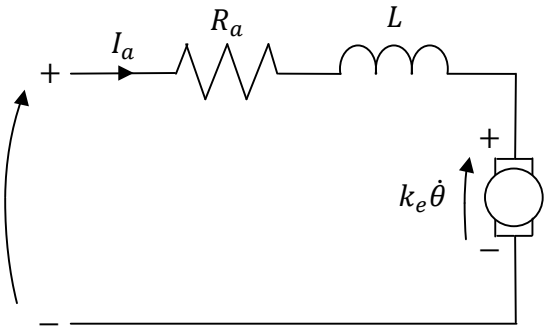


Figure 12: Electrical circuit for the DC-motors.

$$U = R_a I_a + k_e \dot{\theta} + L\frac{di}{dt} \tag{33}$$

Since the electrical time constant is much smaller than the mechanical the inductance can be neglected which gives equation (34).

$$U = R_a I_a + k_e \dot{\theta} \tag{34}$$

Then the following equation describes the shaft torque.

$$T_m = n k_t I_a \tag{35}$$

The shaft torque also needs to overcome the inertia of the motor as well as the viscous damping and motor friction. They are all hard to estimate and very small. Therefore both the friction and viscous damping as well as the inertia was chosen to be neglected after testing some simulations to see the effect of using small values compared to fully neglect them. Then the shaft torque can be described as

$$T_m = n k_t I_a \tag{36}$$

then the motor constants k_t and k_e needs to be found. These can for bigger and more expensive motors be found in the datasheets. However when datasheets are not available these can be estimated by using the rated values of the motor and making simplifications. This gives for k_t

$$k_t = \frac{T_r}{I_{a,r}} \tag{37}$$

where T_r and $I_{a,r}$ is the rated torque and the rated armature current. To find the constant k_e the equation for the back-emf was used. The back-emf is described as

$$V_e = \omega k_e \tag{38}$$

or since the datasheet gives the speed in rpm

$$V_e = \frac{N60}{2\pi} k_e \tag{39}$$

and this gives the equation for k_e

$$k_e = \frac{(V_r - I_{a,r}R)60}{2\pi N_r} \tag{40}$$

for rated speed, voltage and current. Since both motor constants were estimates these were uncertainties that can have different effects on the performance of the robot since they directly effects the torque and back-emf description of the motors. Since the motors were to be controlled thrue PWM, meaning that they were voltage controlled, the current needed to be eliminated from the torque equation. This is done by using equation (34) which gives

$$I_a = \frac{U - k_e\dot{\theta}}{R_a} \tag{41}$$

Then the torque-equation becomes for each motor

$$T_m = \frac{nk_tU}{R_a} - \frac{nk_ek_t\dot{\theta}}{R_a} \tag{42}$$

3.4 Nonlinear model

From the equations gives the nonlinear equations for the pendulum's angular acceleration $\ddot{\psi}$ and the wheels angular acceleration $\ddot{\theta}$.

$$\begin{aligned}\ddot{\psi} =&(J_w(gLR_am_psin\psi + 2nk_t(U - k_e\dot{\theta})) + r(LrR_am_psin\psi(m_p(g \\ &- L\dot{\psi}^2cos\psi) + gm_w) + 2nk_t(U - k_e\dot{\theta})(m_p(Lcos\psi + r) + rm_w))) \\ &/(R_a(J_p(J_w + r^2(m_p + m_w)) - L^2r^2m_p^2cos^2\psi))\end{aligned} \tag{43}$$

$$\ddot{\theta} = \frac{Lrm_pcos\psi(gLR_am_psin\psi + 2nk_t(U - k_e\dot{\theta})) + J_p(2nk_t(U - k_e\dot{\theta}) - LrR_am_p\dot{\psi}^2sin\psi}{(R_a(J_p(J_w + r^2(m_p + m_w)) - L^2r^2m_p^2cos^2\psi))} \tag{44}$$

3.5 Linearized model

The states were defined as

$$x = \begin{bmatrix} \theta \\ \psi \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \tag{45}$$

This gives the linearized state space model

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx + Du \end{aligned} \tag{46}$$

where

$$\begin{aligned} A &= \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \frac{gL^2rm_p^2}{-L^2r^2m_p^2+J_p(J_w+r^2(m_p+m_w))} & \frac{-2nJ_pk_ek_t-2Lnrk_ek_tm_p}{(-L^2r^2m_p^2+J_p(J_w+r^2(m_p+m_w)))R_a} & 0 \\ 0 & \frac{gLJ_wm_pR_a+Lr^2m_p(gm_p+gm_w)R_a}{(-L^2r^2m_p^2+J_p(J_w+r^2(m_p+m_w)))R_a} & \frac{-2nJ_wk_ek_t-2nrk_ek_t((L+r)m_p+rm_w)}{(-L^2r^2m_p^2+J_p(J_w+r^2(m_p+m_w)))R_a} & 0 \end{bmatrix} \\ B &= \begin{bmatrix} 0 \\ 0 \\ \frac{2nJ_pk_t+2Lnrk_tm_p}{(-L^2r^2m_p^2+J_p(J_w+r^2(m_p+m_w)))R_a} \\ \frac{2nJ_wk_t+2nrk_t((L+r)m_p+rm_w}{(-L^2r^2m_p^2+J_p(J_w+r^2(m_p+m_w)))R_a} \end{bmatrix} \quad C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad D = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \end{aligned} \tag{47}$$

The eigenvalues of the open loop system is the eigenvalues of the A matrix

$$eig(A) = \begin{bmatrix} 0 & -29.6 & 5.18 & -5.04 \end{bmatrix} \tag{48}$$

and because there were eigenvalues in the right half plane and in zero the system was unstable. To stabilize the system a controller was needed which gives the closed loop systems eigenvalues in the left half plane.

3.6 Open loop test

The mathematical model was programmed in Matlab Simulink. Both the nonlinear and the linear model have the voltage u as input signal and the angle ψ, the angular velocity $\dot{\psi}$, the angle θ and the angular velocity $\dot{\theta}$ as output signals. With the model tests could be made to verify how the systems behaves and to design controllers for stabilizing the inverted pendulum.

The maximum input voltage was $11.1V$ and this was modeled with the saturation block to realize how much torque the motors can give. A backlash block is used to realize the backlash of the motors. Measurements for dead zone takes care of the dead zone block. All these three parts will complement the mathematical model and make it more similar to the reality.

An open loop test was made to see how the mathematical model behaves and if it fulfills the expected behaviour. The angle ψ has an initial value of $-1°$ in the beginning. All other have initial values 0. The pendulum is expected to fall over and the angle ψ will be $-180°$ for the nonlinear model.

Figure 13 shows the angle ψ and the angular velocity $\dot{\psi}$ for the nonlinear and the linear model.

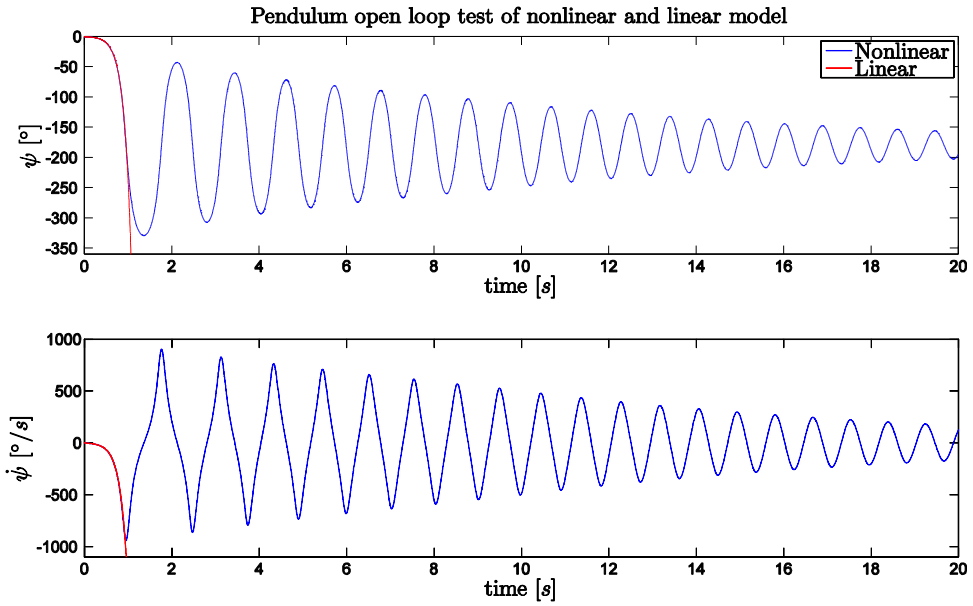


Figure 13: Angle ψ and angular velocity $\dot{\psi}$ of the pendulum for the open loop nonlinear and linear model. The initial value for ψ is $-1°$.

The angle ψ for the nonlinear model behaves as excepted and fall over and starts to oscillate around $-180°$ which is a stable point. The amplitude was decreasing with time because of energy losses in the motors when they forces to rotate. The angular velocity $\dot{\psi}$ oscillates around $0°/s$ for the nonlinear model. The linear model follows the nonlinear model around the operating point which is $0°$ and $0°/s$ which is seen. When the pendel moves away from this operating point the linear model can not be valid anymore.

Figure 14 shows the angle θ and the angular velocity $\dot{\theta}$ for the nonlinear and the linear model.
Because of the pendulums movement effects the wheels the angle θ and the angular velocity $\dot{\theta}$ starts to oscillate around 0 for the nonlinear model. The oscillation was decreasing which was aspected because of the friction. The linear model follows the nonlinear model close to the operating point.

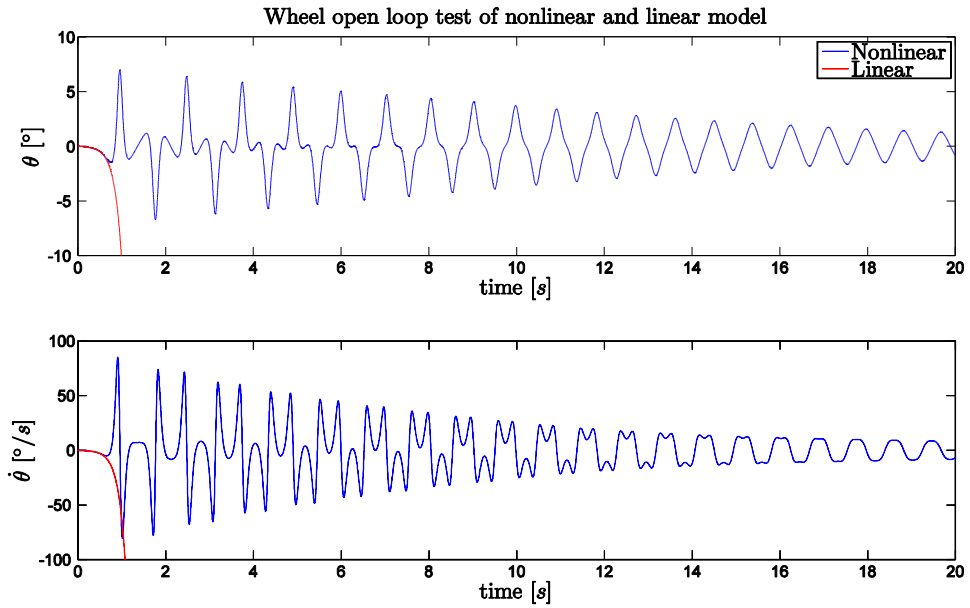


Figure 14: Angle θ and angular velocity $\dot{\theta}$ of the wheel for the open loop nonlinear and linear model.

4 Control design

The robot is an inverted pendulum control problem which need a controller. The states of the model were

$$\begin{bmatrix} \text{Wheel angle} \\ \text{Pendulum angle} \\ \text{Wheel angular velocity} \\ \text{Pendulum angular velocity} \end{bmatrix} = \begin{bmatrix} \theta \\ \psi \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \tag{49}$$

4.1 Robot standing upright

To stabilize the robot first a PID controller was designed and verified and then a LQG controller was designed and verified. For the PID controller a Kalman filter was used to remove the noise from the accelerometer and gyro. For the LQG a Kalman observer was needed to estimate the speed of the wheel and also to filter the sensor noise. The control system was sampled and therefore were the controllers designed in discrete time with the same sample time that was used in the real system. Both the PID and the LQG were simulated with varying sampling times to find the one best suited for the real system. The PID turned out to be much less sensitive to the length between samples while the LQG needed a shorter sample time to be able to stabilize the system. The system ultimately used was discretized with Zero-order-hold method and the sampling time $T_s = 10ms$. The eigenvalues of the open loop discrete time system were

$$eig(A) = \begin{bmatrix} 1.00 & 0.743 & 1.05 & 0.951 \end{bmatrix} \tag{50}$$

which shows that the system was unstable because eigenvalues were outside the unit circle. To be able to control the system the controller needs to move the eigenvalues inside the unit circle to stabilize at its operating point straight up. The Simulink models are shown in appendix C. In both the PID case and the LQG case noise was added in the simulations to realistically simulate the measurements of ψ and $\dot{\psi}$ of the real system. The noise was approximated as normal distributed white noise. The amplitude of the noise was tuned by measuring the real signal and compare the real noise with the simulated noise.

4.1.1 Kalman filter

For the PID controller the Kalman filter described in section 2.4.2. This filter removes the noise and drift of the measurements and gives a good signal to the controller but it does not give any estimates of the states not measured so it cannot be used for controlling the movement of the robot. The simulations was compared with measurments of the real system with the filter implemented and both simulations and real measurements showed that the filter worked as expected.

4.1.2 PID control

The PID controller was designed by using the linear system, a sensor model and the Kalman filter. The parameter to be controlled by the PID was the angle of the pendulum. The transfer function of the continuous time system from input u to output y is

$$G_{uy} = \frac{-22{,}28s - 1{,}789 * 10^{-14}}{s^3 + 29.5s^2 - 30.41s - 773.7} \tag{51}$$

The simulations showed that it was not possible to use any derivative part in the controller so the controller was a PI controller and the transfer function for a PI controller is

$$F_{PI} = \frac{K_p s - K_i}{s} \tag{52}$$

Matlabs Sisotool-toolbox was used to get starting values for the controller. Then by trial and error values for K_p and K_i was found

$$K_p = 271.8 \quad K_i = 180.77 \tag{53}$$

The feedback system was

$$G_{ry} = \frac{F_{PI} G_{uy}}{1 + F_{PI} G_{uy}} \tag{54}$$

which was discretized to

$$G_{ry} = \frac{0.2624z^2 - 0.02276z - 0.2363}{z^3 - 2.247z^2 + 1.994z - 0.7445} \tag{55}$$

and the eigenvalues for the feedback system with the choosen control parameters became

$$eig(G_{ry}) = \begin{bmatrix} 0.9946 \\ 0.6262 + 0.5970i \\ 0.6262 - 0.5970i \end{bmatrix} \tag{56}$$

which shows that the feedback system is stable. These parameters stabilized the system fine in the simulation as seen in figure 15. The figure shows the true signal, the filtered fignal and the noisy sensor signal. After $5.5s$ a disturbance was given as a push to the system to see how it responded. The push lasted for $0.5s$ and as seen the simulation shows that the controller manages to stabilize it. As seen the system is stable and continues to be stable after the push with a good filtered signal.

4.1.3 Observer

When using state feedback control all states needs to be known to be able to control them. To know the states they needs to be either measured, or estimated by an observer. The robot has an accelerometer and a gyro that measures ψ and $\dot{\psi}$. This corresponds to the two states x_2 and x_4. Since the wheel velocity needed to be controlled to be able to move the robot or make sure it stood still this state needed

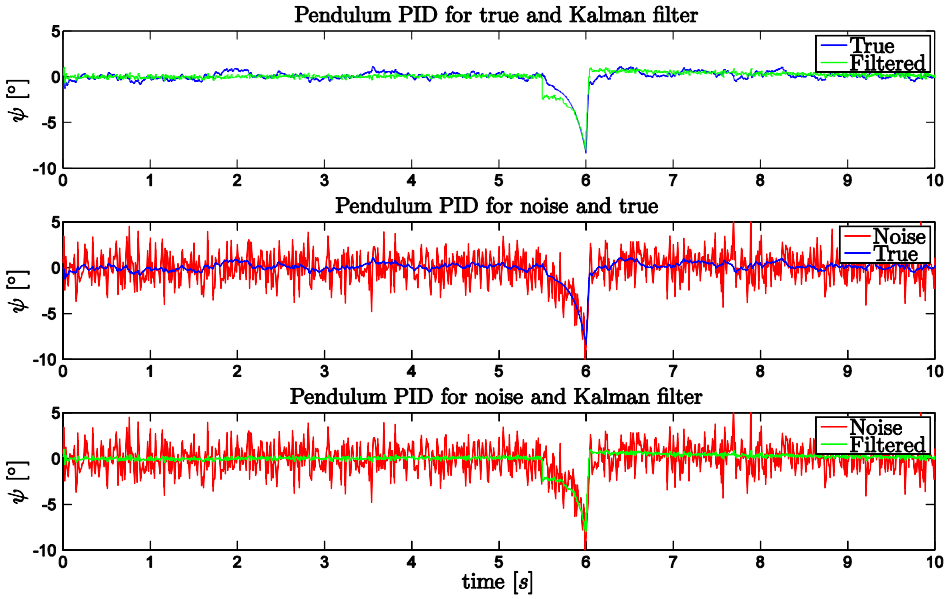


Figure 15: Simulation of PID controller for the true signal with noise and Kalman filtering. After $5.5s$ was a punch given to ψ for $0.5s$ in negative direction.

to be estimated. The angle θ of the wheel however is of no interest and could be removed from the state equations of the controller. The new states when θ was removed were

$$x = \begin{bmatrix} \psi \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \tag{57}$$

This yields a new state space model with three states

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx \end{aligned} \tag{58}$$

where

$$\begin{aligned} A &= \begin{bmatrix} 0 & 0 & 1 \\ \frac{gL^2rm_p^2}{-L^2r^2m_p^2+J_p(J_w+r^2(m_p+m_w))} & \frac{-2J_pnk_ek_t-2Lrnk_ek_tm_p}{(-L^2r^2m_p^2+J_p(J_w+r^2(m_p+m_w)))R_a} & 0 \\ \frac{gLJ_wm_pR_a+Lr^2m_p(gm_p+gm_w)R_a}{(-L^2r^2m_p^2+J_p(J_w+r^2(m_p+m_w)))R_a} & \frac{-2J_wnk_ek_t-2rnk_ek_t((L+r)m_p+rm_w)}{(-L^2r^2m_p^2+J_p(J_w+r^2(m_p+m_w)))R_a} & 0 \end{bmatrix} \\ B &= \begin{bmatrix} 0 \\ \frac{2J_pnk_t+2Lrnk_tm_p}{(-L^2r^2m_p^2+J_p(J_w+r^2(m_p+m_w)))R_a} \\ \frac{2J_wnk_t+2rnk_t((L+r)m_p+rm_w}{(-L^2r^2m_p^2+J_p(J_w+r^2(m_p+m_w)))R_a} \end{bmatrix} \quad C = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad D = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \end{aligned} \tag{59}$$

These state equations was then used to design the observer. The observer used was the Kalman observer which gives both the estimates of the states and also filters the measured signals so there is no problem with measurement noise or drift. The

Kalman observer is described as

$$\begin{aligned} K &= (APC^T + R_{12})(R_2 + CPC^T)^{-1} \\ &\text{Where } P > 0 \text{ is the solution to} \\ P &= APA^T + R_1 - (APC^T + R_{12})(R_2 + CPC^T)^{-1}(CPA^T + R_{12}^T) \end{aligned} \tag{60}$$

When designing the observer the covariance matrices of the measurement noise and the process noise had to be determined. R is the covariance matrix for the measurement noise and Q is the one for the process noise.

$$R = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{61}$$

and

$$Q = 200 \tag{62}$$

This gave the Kalman observer gain K

$$K_{Kalman} = \begin{bmatrix} 0.007202 & 0.01583 \\ 0.06804 & -0.8172 \\ 0.02786 & 0.6822 \end{bmatrix} \tag{63}$$

4.1.4 LQG control

For the LQG the cost matrices were selected given that main goal to stabilize the robot which is the pendulum angle ψ. But to decide position of the robot the angular velocity of the wheel $\dot{\theta}$ was given higher value as well. The Q matrix was designed to:

$$Q = \begin{bmatrix} 4x10^5 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{64}$$

and R was

$$R = 10 \tag{65}$$

This gave the control parameters for the gain L

$$L = \begin{bmatrix} -143.5 & -1.740 & -4.924 \end{bmatrix} \tag{66}$$

and K_r was calculated to

$$K_r = \begin{bmatrix} 0 & -0.740 & 0 \end{bmatrix} \tag{67}$$

To check the stability the eigenvalues of the closed loop system were calculated to

$$eig(A) = \begin{bmatrix} 0.549 + 0.264i & 0.549 - 0.264i & 0.998 \end{bmatrix} \tag{68}$$

which shows that the closed loop system was stable because the eigenvalues are inside the unit circle and the goal with the controller was fulfilled.

Figure 16 and figure 17 shows simulations for the LQG with noise and Kalman observer. Feedback were from the estimated states and the reference signals were all equal to 0.

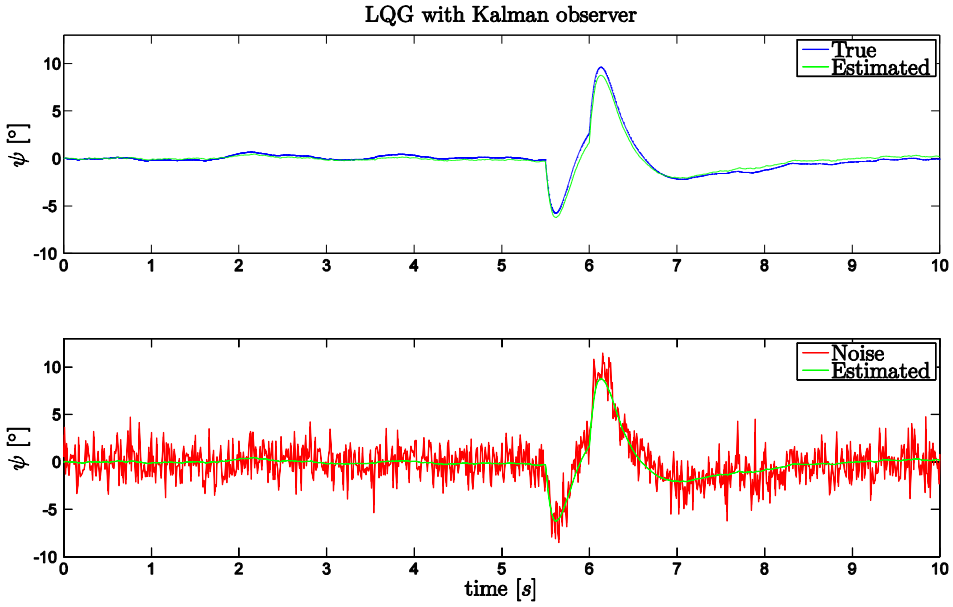


Figure 16: Simulation of LQG over ψ for true signal with noise and the estimated filtered signal with feedback from the estimated states with reference signal $\dot{\theta} = 0$. After $5.5s$ was a punch given to ψ for $0.5s$ in negative direction.

The figures shows that the noise was reduced and the estimated signal follows the true signal with a maximal differens for ψ of $1°$ when the disturbance was added. For $\dot{\theta}$ the estimated signal follows the true signal and keeps it close to $0°/s$. After the distrubance the true θ shows that it goes towards $0°$ again which was required from the reference signal.

4.2 Moving the robot

To see how the model works when it was driving forward, the reference signal for $\dot{\theta}$ was put equal to $180°/s$. Figure 18 and figure 19 shows simulation of the driving forward mode.
The angle of the pendulum goes down to $-2.5°$ in the start but after $0.5s$ the angle is 0 again. This is because the robot gets an acceleration. $\dot{\theta}$ is close to $180°/s$ and θ moves. After the distrubance θ continues in the old line.
The robot will be able to carry a load in the bowl. This was introduced to the model with an extra mass in the bowl of $0.9kg$ which is a realistic mass if the bowl is filled with candy. Figure 20 and figure 21 shows simulation of the driving forward mode with the extra mass.
With the extra mass the robot behaves as required, stabilized and moving forward.

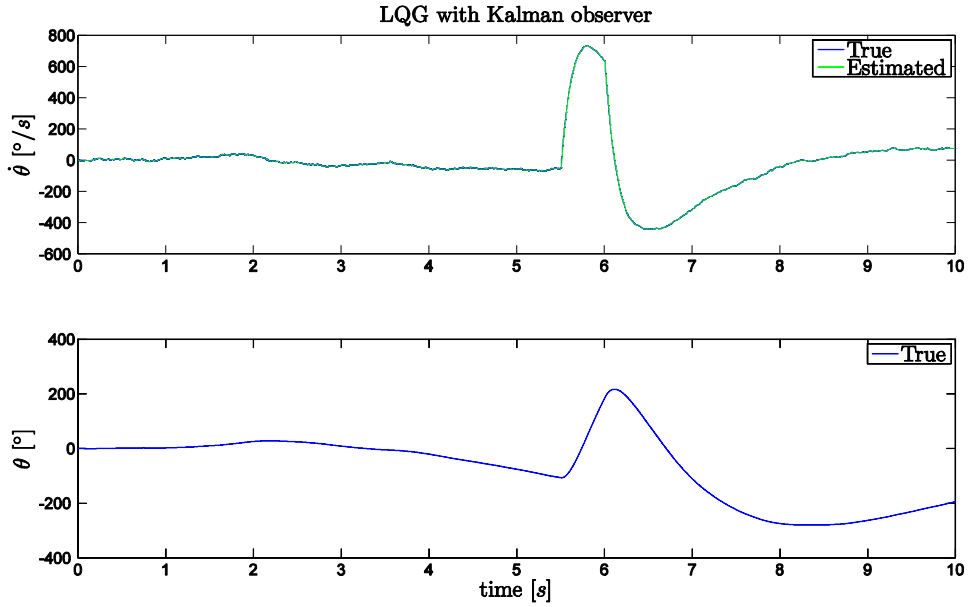


Figure 17: Simulation of LQG over $\dot{\theta}$ for true signal and the estimated signal and true signal for θ with feedback from the estimated states with reference signal $\dot{\theta} = 0$. After $5.5s$ was a punch given to ψ for $0.5s$ in negative direction.

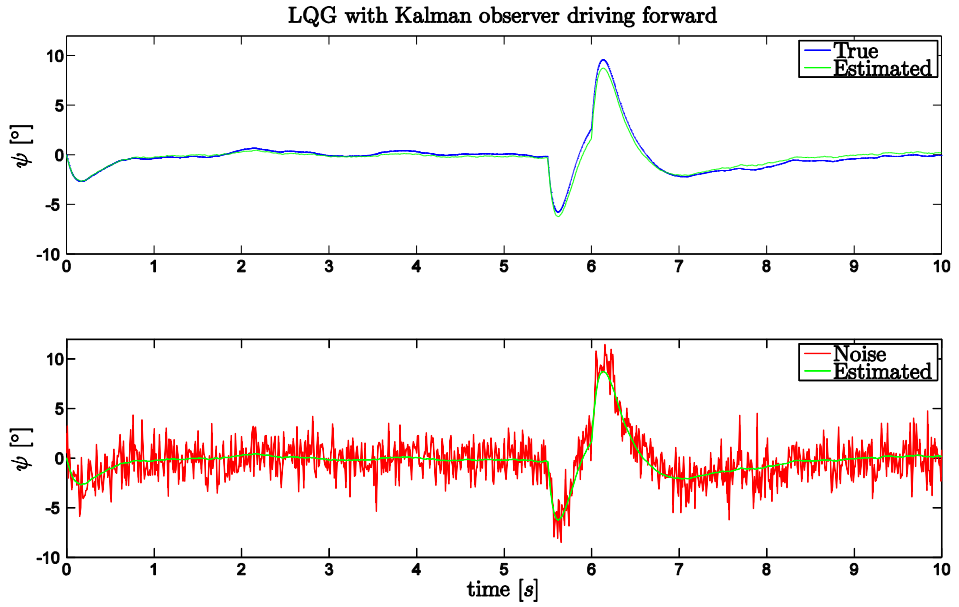


Figure 18: Simulation of LQG over ψ for true signal with noise and the estimated filtered signal with feedback from the estimated states with reference signal $\dot{\theta} = 180°/s$. After $5.5s$ was a punch given to ψ for $0.5s$ in negative direction.

4.3 Edge detecting

When an edge is detected the robot should change direction. This is done by changing the reference signal to a negative reference signal. Then the robot runs in the

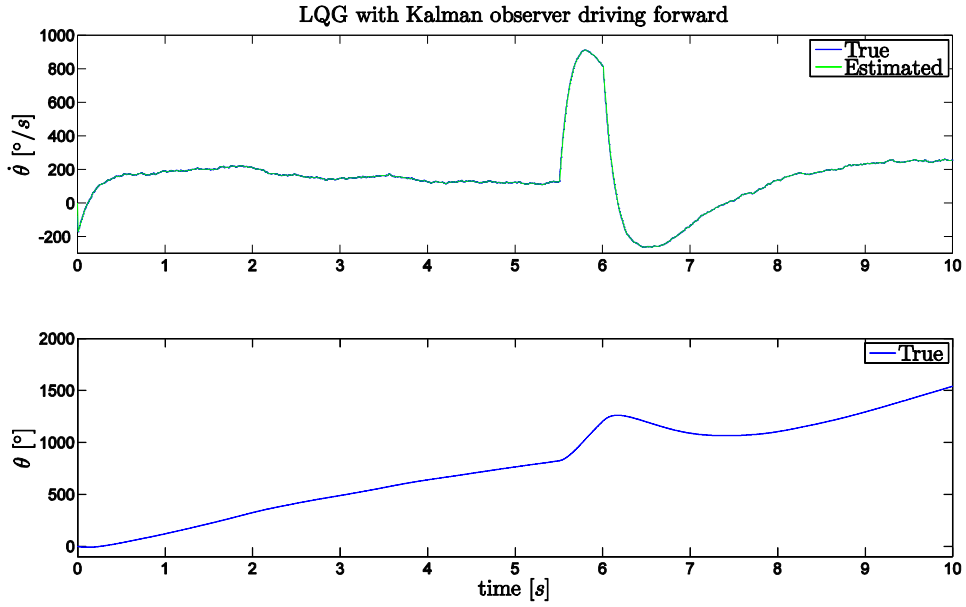


Figure 19: Simulation of LQG over $\dot{\theta}$ for true signal and the estimated signal and true signal for θ with feedback from the estimated states with reference signal $\dot{\theta} = 180°/s$. After $5.5s$ was a punch given to ψ for $0.5s$ in negative direction.

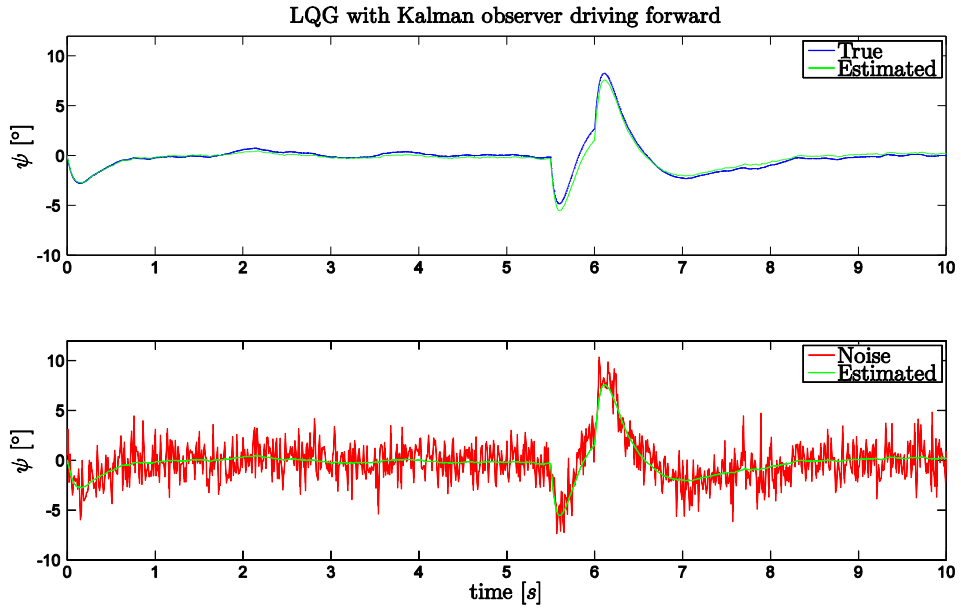


Figure 20: An extra mass $0.9kg$ is added in the bowl. Simulation of LQG over ψ for true signal with noise and the estimated filtered signal with feedback from the estimated states with reference signal $\dot{\theta} = 180°/s$. After $5.5s$ was a punch given to ψ for $0.5s$ in negative direction.

backwards direction for a time t_1 and then comes to a stop by setting the reference to 0. Then it should make a turn. This is achieved by once again giving a positive reference signal but then remove some control signal from one wheel and add the same

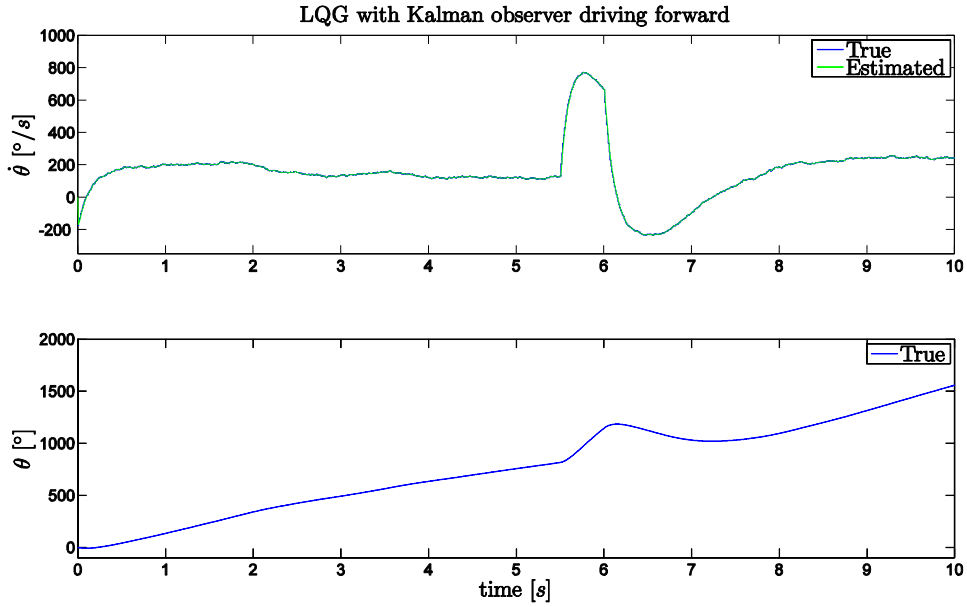


Figure 21: An extra mass $0.9kg$ is added in the bowl. Simulation of LQG over $\dot{\theta}$ for true signal and the estimated signal and true signal for θ with feedback from the estimated states with reference signal $\dot{\theta} = 180°/s$. After $5.5s$ was a punch given to ψ for $0.5s$ in negative direction.

amount to the other wheel for a time t_2. Then it should once again move forward by setting a positive reference signal. This function was however never implemented due to the robot never got to move so the actual turn algorithm was never tuned.

5 Experiments, results and analysis

5.1 Construction

The robot was constructed and built using plexi glass and threaded rods with a pink bowl for candy on top. The total height from the ground to the top was $33cm$ and the weight without candy $2{,}2kg$. A switch was placed on the side of the bowl to easily turn the robot on and off. The sensors and the display are protected by the plexi glass and all the electronics are clearly visible.

5.2 Sensor performance

5.2.1 Accelerometer and gyro sensor

The Accelerometer and gyro were read by the microprocessor and tests were made by tilting the robot by hand backward and forward. From the accelerometer the acceleration in y- and z-direction were used and from the gyro the angular velocity around x-axis was used. Data logging was done from the robot for the raw values of accelerometer and gyro angle without and with filtering. Figure 22 shows the angle ψ for raw values from the accelerometer, raw values from the gyro integrated to angle, Complementary filter and Kalman filter implemented on the robot. The Complementary filter shown in the figure is with $\alpha = 0.95$.

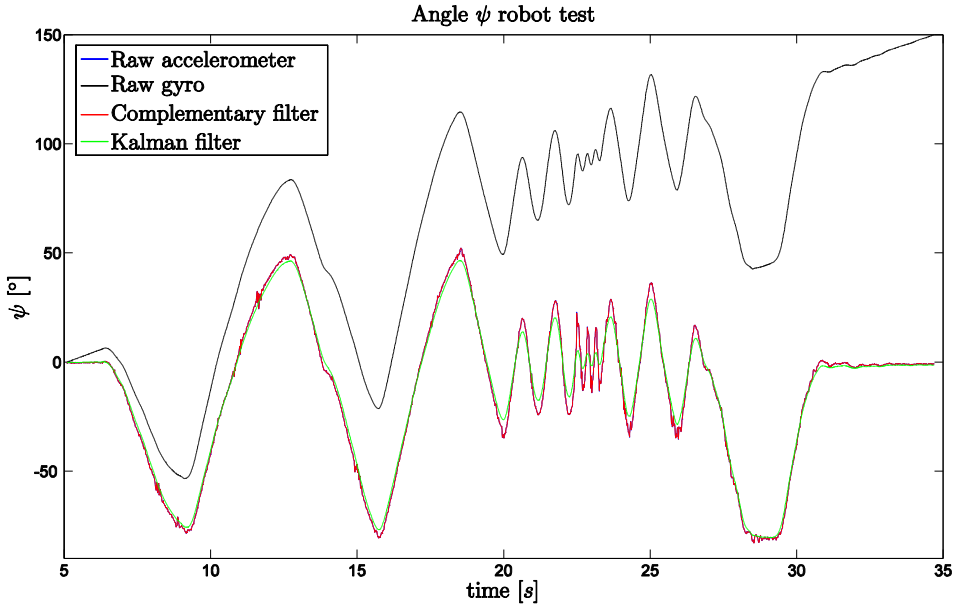


Figure 22: Robot test for angle ψ. The figure includes raw accelerometer values, raw gyro values integrated to angle, implemented Complementary and Kalman filter.

Figure 23 shows the raw gyro values around x-axis and the implemented Kalman filter for removing the offset.
The figure shows that accelerometer angle had high frequencies peakes but the gyro had no high frequence peaks. Instead the gyro was drifting with $5°/s$ and the gyro

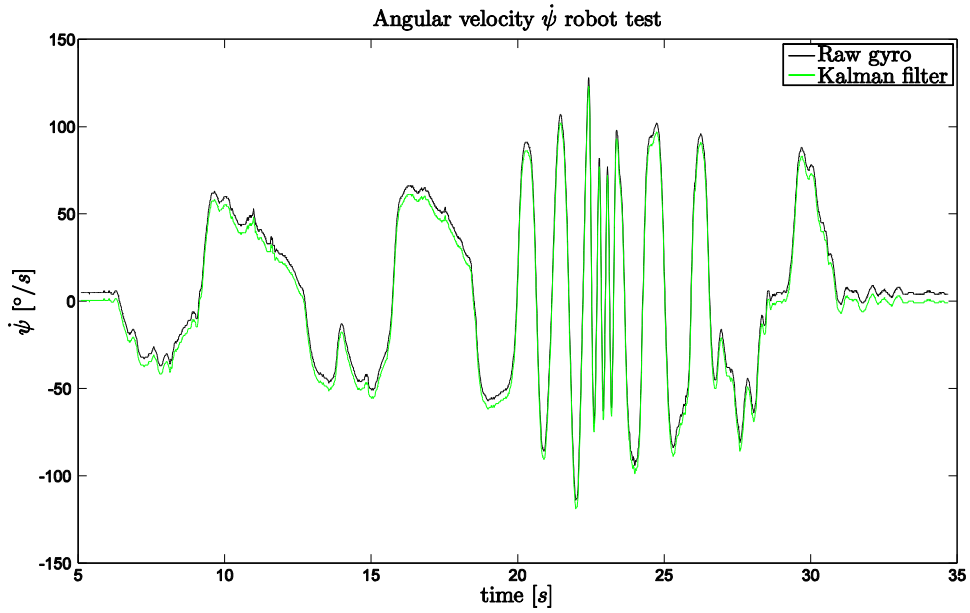


Figure 23: Robot test for angular velocity $\dot{\psi}$ for raw gyro and Kalman filter.

angle changed a lot from the true angle. The Complementary filter removed the high frequency peaks badly from the accelerometer but removed the low frequency drifting from the gyro. But it showed to still be sensitive to the noise. The raw accelerometer angle is very close to the Complementary filter angle and is therefore hard to see. The Kalman filter removed the noise from the high frequency peaks and also removed the gyro drifting. The Kalman filter for the gyro removed the offset sucessfully which is best seen in the beginning and the end of the test when the robot was still. Based on this the Kalman filter was choosen to be used.

5.2.2 Distance sensor

Figure 24 shows a distance sensor test when the object in front of the sensor is moving from $100cm$ to $0cm$. As seen in the figure the axis don't show the actual units of measurement in cm or V. This is because there was actually no need to know the distance in cm if there is some know thresholds for different distances. Figure 25 for instance shows the value of the sensor reading for $15cm$. The reason for not calculating the distance to actual cm was to unload the processor from unneccessary math wich would require floating points for no use. Also as seen in the graphs the original signals had some noise of which some of the noise was hard to capture because of it's rare appearance. However it caused some trouble because of high voltage peaks above the threshold value. The solution for this was to use a low-pass filter. Also as seen in the figure the filter had a risetime of about $0.5s$ which could have been to slow in other applications. But since the distance measurement in front of the robot is not time critical speed was not an issue so to achieve best performance in regards of clean signal the speed was not regarded as an important factor.

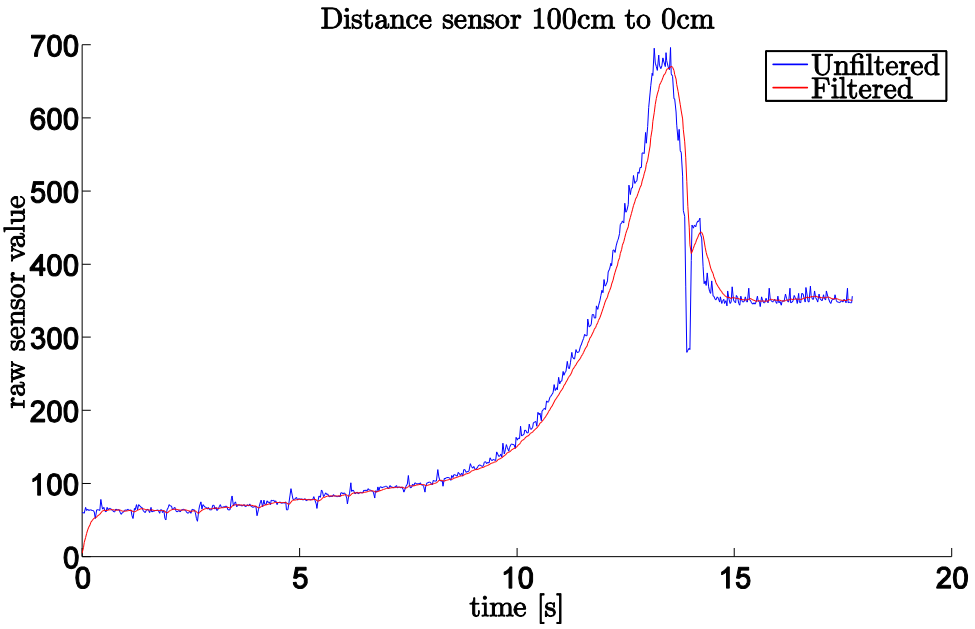


Figure 24: Distance sensor 100cm to 0cm unfiltered and Butterworth filtered.

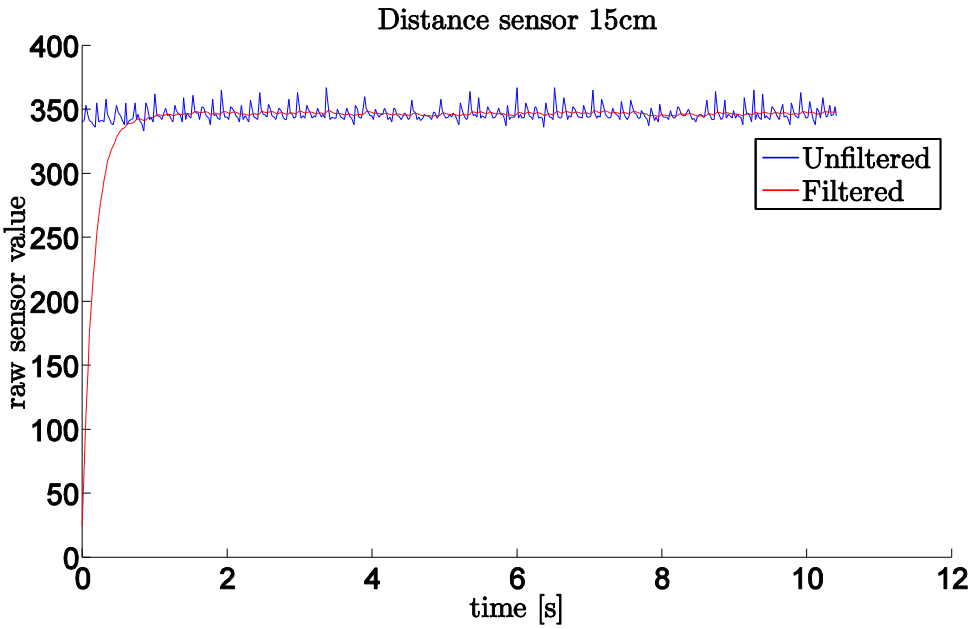


Figure 25: Distance sensor 15cm unfiltered and Butterworth filtered.

5.2.3 Table sensor

Since it was hard to actually measure the distance to get proof of concept the only measurements were to see if there ever was any noise that would cause the sensor to return a faulty reading in either when reading zeros or when reading ones. As it turned out it never returned a faulty reading out of 10 tests in either case. Then the sensor was moved back and forth from 25cm and 0cm to see the readings. The result shows in figure 26.

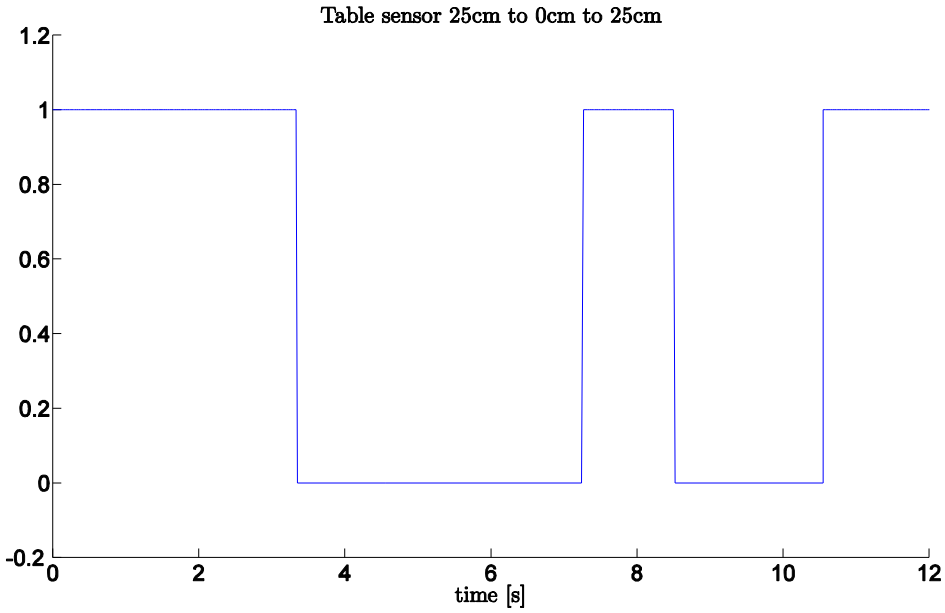


Figure 26: Table sensor $25cm$ to $0cm$ to $25cm$.

5.2.4 Temperature sensor

The following test was made with a window open on a hot and windy day, the temperature outside was hotter then inside but had some cool strong winds. As expected from the description of the sensor when there is a lot of fluctuations in temperature, because of the wind for example, there is a lot of noise. But the noise is not only because of temperature fluctuations but also of power source and communication noise which all can be seen as normal distributed white noise. But noise due to temperature fluctuations has no consistent character and is hard to filter. This was confirmed in figure 27 where the noise is seen to be quite severe and a person moves in front of the sensor at 1 meter range.
When looking at a power spectrum density graph in figure 28 of one of the signals from a sensor it can be seen that the frequency of interest was very low, but the noise was spread across the frequency band in such power that it was hard to filter even in the low frequencies. Knowing this the filter was designed with a cut-off frequency of $1Hz$. This would still prove to be a noisy signal but was the best that could be achieved with a Butterworth filter. Other filters were tried, like the Chebyshev-filter, without any improvements. The filtered signals is shown in figure 29.
Since the sensor is more likely to operate in a room with a close to constant temperature, no winds and much smaller temperature fluctuations it was also tested thoroghly in a closed, but hot, room as seen in figure 30 and figure 31.
Because of this behavior of the sensors the decision was made not to take any control actions, such as steering the robot, based upon temperature readings. Instead it was only used for interacting the display with the surroundings.

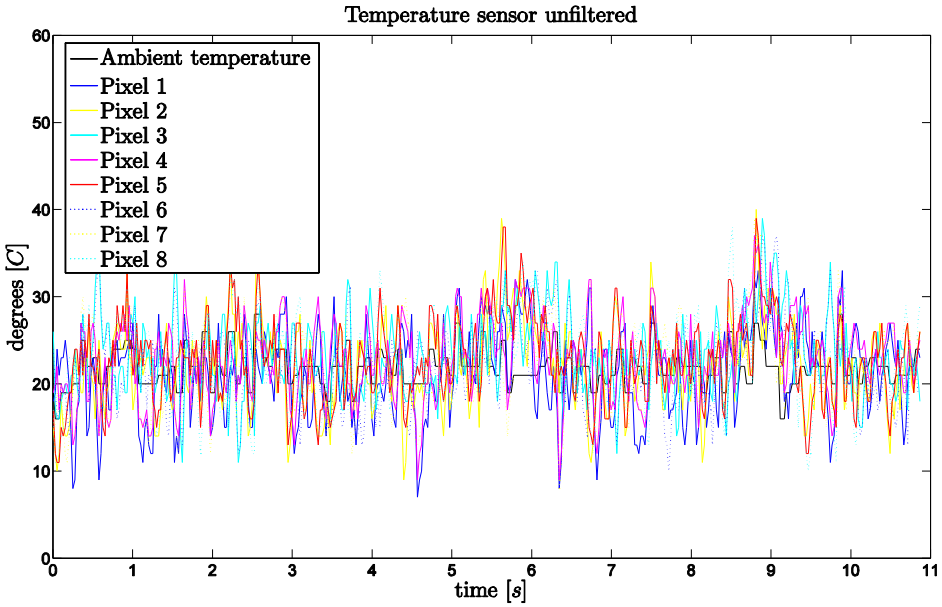


Figure 27: Temperature sensor influenced by temperture fluctuations.

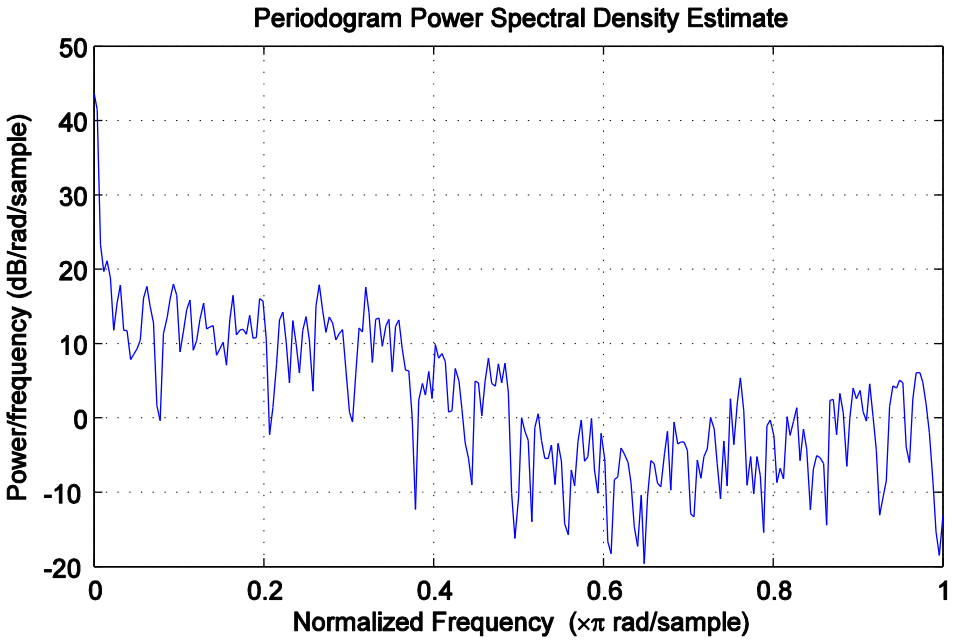


Figure 28: Power spectrum of one temperature signal.

5.3 Robot performance

5.3.1 Motor friction compensation

The electric motors were bought as identical, however there are always differences between two motors. Some of this differences can be showed when testing the performances of the motors and the motordriver. When starting the motors from 0 PWM and stepping up there were both differences between the motors and differences on each motor depending on the conditions of the test. To begin with the motors had

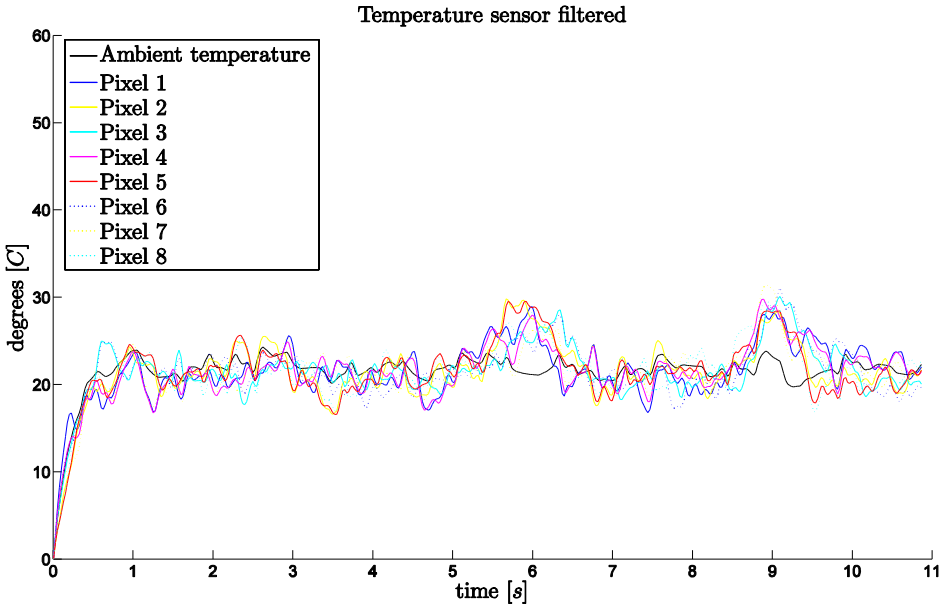


Figure 29: Temperature sensor influnced by temperture fluctuations with filter.

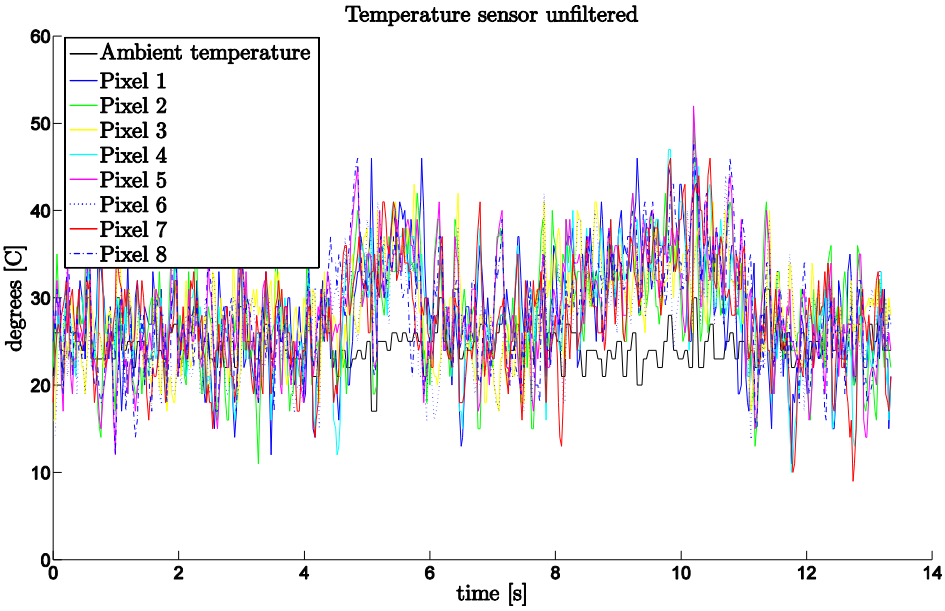


Figure 30: Temperature sensor in a closed room unfiltered.

different friction coefficients which can be felt when turning the motors by hand; one was turning easier than the other. The friction also showed as when starting the motors they needed different voltages to start. Hence there seems to be some need of friction compensation to make the motors more similar and also to make use of the full span of the available PWM signal as describe in figure 32.
This friction as well as the armature resistance changed with time as the motors heated up. The resistance used for simulation and modeling was the measured resistance when the motors had been operating for a while and had reached the

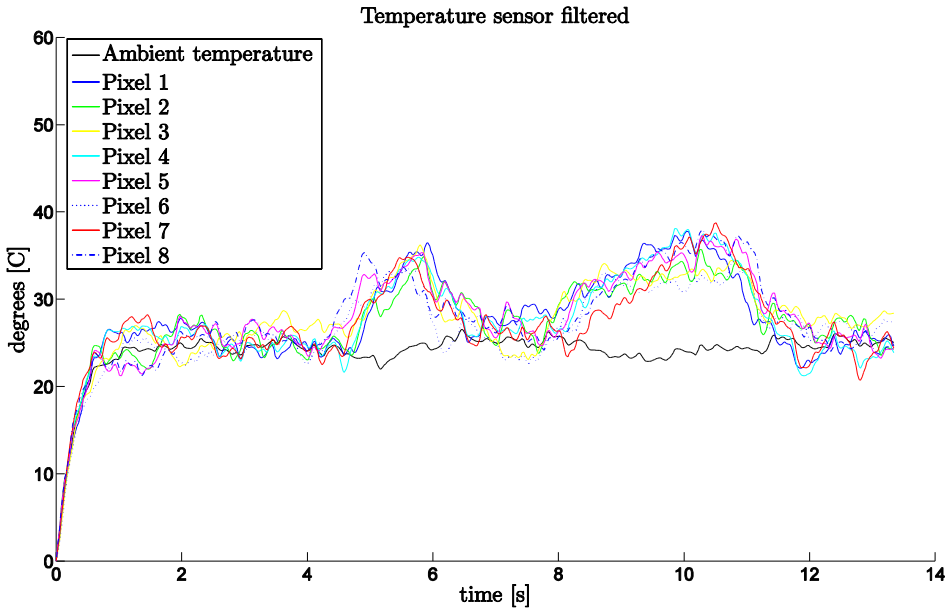


Figure 31: Temperature sensor in a closed room filtered.

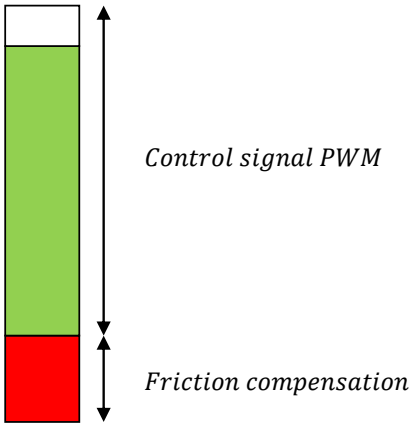


Figure 32: The span of the PWM signal. The red region shows the friction that needs to be overcomed to rotate the motors. By moving the PWM span to the green region this is achieved.

normal operating temperature as the resistance didn't change that much when they were operating in normal operation. The starting resistance was lower than the operating resistance and the effect of this was that when the robot was started the controller had a larger overshoot than when it had been operating for a while. This bigger overshoot was however so small when using low power motors that it was neglectable and was not considered as a problem. Based on tests it was seen that the motors behavior was not consistent which made it hard to compensate for differences between them. When the motors both made a cold start they both require about the same voltage to start but as they heat up there were differences between the two motors. Also it can be seen that one motors behavior is not consistent in different

tests but also varies. Some choosen datapoints which best represents the important results from the tests can seen in table 2.

PWM	Motor A (V)	Motor B (V)
Cold start test with forward direction		
50	0.359	0.34
100	6.4	6.32
150	8.5	8.4
200	9.6	9.6
255	10.85	10.85
Heated start backward direction		
50	1.8	0.35
100	6.3	6.6
150	8.44	8.62
200	9.55	9.6
255	10.85	10.85
Heated start forward direction		
50	0.36	2.2
100	6.29	6.6
150	8.4	8.62
200	9.5	9.6
255	10.85	10.85

Table 2: Motor test.

To be noted from the table is that in some of the tests the motors were running at $PWM = 50$, which can be seen by a lower voltage, and in other tests they were not why the voltage was significantly higher. It may seem from the table that motor A and B responds differently from backwards and forwards, but this was not the case. In some tests the results were the other way around that they started easier in the other direction than the table or that they did not start or started in both directions. Therefore it made it clear that it was going to be hard to compensate for this effect. What can be seen is that when the motors were running at speed they were quite consistent in power which says that if the robot is moving in one direction it will be moving close to a straight path when the same control signal is given to both motors. But if the robot is to balance at standstill then the same control signal will cause the robot to move back and forth with one wheel. This is because when the robot starts to fall a control signal will be given, the same to both motors, and when the signal gets high enough one of the motors will begin to move to upright the robot again. Since the other motor will still be standing still this will lead to the robot continues to fall and the control signal gets higher and the moving wheel will move faster so the robot moves back and forth on one wheel while the other stands still or makes a much smaller movement. This problem can be solved by either measuring the movement with encoders to see if the wheel is actually moving or by using current sensors to see if the motor is turning.

5.3.2 Reference signal update

When testing the system with a PID controller it became clear that the center of gravity was not above the center of the wheel axis. As the robot tried to balance it was always moving forward until the motors reached top speed and the robot would fall. This behavior indicatated that because of the center of gravity actually was at a point in front of the wheel axis so when the sensors told the robot it was at 0° and should be balanced the robot would be falling forward and the robot would need to supply a control signal to maintain this position. But since the control signal only tried to hold it at the position it would continue to fall and an even larger control signal was required etc. To get past this problem an offset for the angle was found by reading the values for the angle and by adjusting it by trial and error. Thats partly solved the problem but if the center of gravity was to move because of the amount of candy in the bowl or if the sensor would for some reason be slightly moved the problem would return. So to improve the angle offset an algorithm, seen in equation (69) for updating the angle offset was implemented. This algorithm looks at the given control signal and acts as a summer. When the sum gets sufficient large the angle offset would be either increased or decreased in small steps. If it would find the center of gravity perfect the summer would retain a sum pending about zero.

$$\begin{aligned} u_{sum} &= u_{sum} + u(t) \\ &\text{if} \\ |u_{sum}| &> u_{thresh} \\ &\text{then} \\ \psi_{offset} &= \pm\Delta\psi \end{aligned} \tag{69}$$

But when using this function the robot would get problems when it was supposed to move. To move the robot forward a reference angle were given and the control signal would rise and then the control signal sum would continuesly increment and cause the ψ_{offset} to decrease or increase. This results in that the robot would again would have to increase the control signal to hold the reference angle and keep moving and also when the robot would stop it would once again have a large offset that is not true to the real offset and causes the robot to either make a large move in the other direction of the movement or even make it fall. To come around this problem the ψ_{offset} update had to be turned off while the robot moves.

5.4 LQG controller analysis

As the simulations showed, the LQG controller worked good and was able to filter out even some heavy noise. However it never got to work in the real system. A large amount of attempts were done by changing parameters and testing. The first thing that seemed likely was that there had to be something wrong with the mathematical model since the controller is fully based on the model and noise estimation alone. If the mathematical model was wrong the observer would not give correct estimates of the three states and the controller would give the wrong signal. But by assuming the model was correct it seemed like the problem were likely to be in the parameters

of the model. This was a reasonable assumption since the model was designed with care and the simulation seemed to represent the reality well.

5.4.1 Sampling time

To try to narrow down the problem the best controller in simulation was calculated and then the controller was tested by changing parameters in the system, but not recalculating the observer or the controller. The simulations showed that for the intended sample time of $25ms$, corresponding to a $40Hz$ control loop, it was not possible to calculate a controller that stabilized the system. This could be explained by the dynamics of the system being faster than the sampling time which meant that for the sample time it was not possible to capture the full behaviour by the measured system. So different sample times were tested and the critical sampling time to be able to stabilize and control the system was $12ms$ or about $84Hz$, and even then the system was very sensitive to disturbances. So the limit for a robust controller was $10ms$ and since the controller is a very time critical task it meant that the sampling time needed to be changed to fit the system or else it would not be possible to control it. So the control loop was then set to $10ms$, $100Hz$, which according to simulations was sufficient to fully control the system. Since the Arduino only has a $16MHz$ processor and no floating point unit it is possible that the time it takes to do all calculations needed for both the controller, observer, filters etc. took to long time for the processor to be able to keep the set looptime, even if it seems unlikely. According to [1], calculations with floating points takes about 40 times as long as calculations with integers and the floats are only represented by 6 digits which may not be enough precision for all calculations. And for testing the controller none of the other tasks were implemented to ensure that the looptime was correct. Since the transfer through USB-cable was such a slow operation it was not possible to get the actual looptime since the looptime when logging data would be longer. Because of this it was not possible to confirm that the Arduino actually was able to stay within the looptime.

5.4.2 Pendulum parameters

With the problem with too slow control loop sorted out, the system was tested for other parameters to try to identify why the LQG failed to control it. The testing was done systematically by changing one parameter at the time and observing the behaviour and also then by changing more at the same time since many of the parameters were related so that if one was incorrect it would follow that another parameter was effected as well.

First the inertia of the body was tested. Simulations showed that it could be up to 4 times as big as calculated and the controller would still be able to stabilize, however it was more sensitive for lower inertias. This can be explained by the lower inertia would mean a faster system and then it could be faster than the controller could handle, similar to the problem identified with too slow control loop, and the looptime needs to be shorter. The inertia of the motors and the wheels assumed to be so small that they could be neglected and different simulations showed that this

was a valid assumption.

As for the mass the controller turned out to be quite sensitive. The robot was weighted to confirm the calculated weight. But the simulations showed that the weight could only have been wrong with a $\approx \pm 0.2kg$ limit to make the system unstable. It also proved to handle lower weights better as long it was within the limit, otherwise the same problem with dynamics would occur, which meant that the controller should be dimensioned for some weight in the bowl. Of course higher mass would naturally mean higher inertia as well, but still the weight could not be increased much for the system to remain stabilizable. This showed when testing for different amounts of candy in the bowl. When testing different amounts of candy the inertia of the system was recalculated as well, but not the controller. These simulations showed that the maximum amount of candy would be about $0.3kg$ and that limit proved be true even for controller calculated for the case with a load.

5.4.3 Center of gravity

The center of gravity also changes with the mass of course, but since the mass of the robot is quite high as it is, it did not move much. The calculated center of gravity for the robot was $11cm$ above the wheel axis which reponds well to the estimated $10.5cm$ by tests in the real world. This was all the tests of mechanical part of the mathematical model compared to the real world. Next the constants of the motors were tested.

5.4.4 DC-motor parameters

The constants of the electric motors were where most imperfections in estimates was expected since these are hard to measure and estimate when there is no good datasheet available. The datasheet available for the motors was of the typ with ideal curves and rated values for the general DC-motor of this size. This meant that there could be big differences from the reality and the calculated values.

The first parameter tested was the armature resistance. Since they were two poles DC-motors the resistance could be measured directly on the poles and was not expected to be different from the reality. Different tests however showed that the resistance changes as the motors runs. The value used in the model was the value measured when the motors had been running for a while and was heated. The simulations showed that the system was sensitive to low values of the resistance and would result in a shaky behaviour. A too high value on the other hand would not be as problematic and it could be vastly over estimated before causing any problems. This can be explained by the torque equation in section 3.3 where the equation shows that if the resistance was too low the torque would be greater than expected and cause an overshoot. However if it was too large it would mean that the torque would be too low and result in the error would increase and the torque would soon be higher and the system would act a bit slower but smoother, as confirmed in simulations. The same equations explained the behavior with incorrect estimations of the electric constant and the torque constant. The electric constant k_e did not effect

much, a too big value would result in lower torque. And for the torque constant a too big value would, as predicted in the equations, cause a too high torque and overshoots in the system. Since theese constants is calculated from the rated values and curves it is possible that they are not true to the true world and may be the cause of some problem.

5.4.5 Conclusions

As mentioned the it was decided to believe that the mathematical model was correct. The reasoning for this was that it was compared with results for both open loop tests and closed loop tests from works of others [9][2] with similar results. But unable to find a solution to the problems even with thoroughly testing the model might still be incorrect. The most likely problems beside the mathematical model are within the Arduinos ability to keep the correct control loop frequency or to correctly calculate the observer or control signal due to limited digits when using floats, and the model calculated in Matlab has twice the number of decimals. However the speed is more likely. The motor constants is not considered to be the cause of the problem. The reason for this is that if it was due to incorrect armature resistance the problem would disappear as the motors heated up since the measurements were based on warm motors and this was not the case. And for the electrical and torque constants they can not vary in such a large span so it was possible to thoroughly test different variations of these and confirm that they were not the cause. So in conclusion it can be said the problem is in the Arduino, the mathematical model, a combination or simply that the system it self can not be stabilized with a LQG due to it's physical shape and mass.

5.5 PID controller analysis

Because of the inability to design a LQG controller a PID had to be used. This controller was the first to be implemented. The calculated PID in the simulations turned out to just be a PI controller. However in the real system the D part was needed to stabilize the system and the I part needed to be much lower.

Because of reasons mentioned in 5.3.1 the friction and other imperfections of the motors, such as backlash, turned out to be hard to compensate for. This resulted in that the robot was able to balance at a standstill but one of the wheels would occasionally go back and forth.

The PID is a controller that not requires as heavy calculations as the LQG and is also not as sensitive to use a correct loop time which means that the other functions can be implemented without the risk of making the system unstable. But the drawbacks when using the PID is that to make the robot move the speed of the wheels needed to be controlled as well as the angle of the pendulum.

To do this either measurements of the wheelspeed of an observer was needed. Since the system was built with the intention of using an observer sensors for measuring wheelspeed was not purchased. And because of problems mentioned in 5.4 the ob-

server was not implemented. Another reason why, for instance encoders were not considered as a better solution than an observer is that an encoder measures ”ticks” for movement of the wheel. For each tick the system would come to a halt caused by an interrupt which means that if the robot moves a lot there would be a lot of ticks slowing the system down and may cause instability, and to get the angle and speed of the wheel some calculations would be required for each tick and further slow the system.

This meant that the controller was only able to stabilize the robot. The reason why movement without controlling the wheelspeed was impossible can be explained as to make the robot move a offset would be required in the angle making the pendulum lean in one direction. If the pendulum was leaning it would also have a falling movement, an acceleration equal to the gravity, in the downwards direction making it fall further. To keep the pendulum in the correct angle the motors would then require a speed and since the acceleration of gravity would be continous the motors would need to accelerate to compensate for this. Eventually the motors would reach their top speed and the robot would fall. One solution would be to pulse the angle offset, but since this would require the pendulum to straighten up before each pulse, where the time needed to rise up was unknown, and would do so with and overshoot this proved impossible to solve. This was also confirmed with simulations.

6 Summary

6.1 Findings

During the project a robot has been designed and built from scratch. Mechanically it looks and works as planned. A mathematical model of the robot and two control designs were calculated and simulated to verify the systems behaviour.

The PID controller was successfully implemented on the robot together with a Kalman filter and made it balance without moving more than $10cm$ on the table. The implemented PID controller were also able to handle a varying amount of candy in the bowl. It also handled disturbances when someone was picking candy from the bowl.

The robot is also small and autonomous, but the human interaction had to be altered from the original idéa because of the noisy temperature sensor. It could be shown that the sensors combined could be used for some interaction and also for different control actions if movement was successfully implemented. The total cost of the project was below $5000SEK$ which was required. The edge detection system was implemented and confirmed to work through testing.

Lastly a LQG and an observer was designed and simulated successfully. These were the goals that were fulfilled. However the build was not completely successful because of the inability to get the LQG controller and observer to work when implemented. So in the end the robot could balance, but it could not move. This fact left the edge sensors unused in practice.

6.2 Further work

To further improve this work the following suggestions can be done. For the problem with the calculations in Arduino a fixed point arithmetic routine could be written to improve speed. This must be written from scratch since there currently exist no libraries for the Arduino doing this. To solve the movement without an observer two encoders could be implemented. However then also a small circuit would be required to do the calculations for the encoder counts. When the movement is solved the filtering of the temperature sensor could be improved. This would possibly require to either solve the movement with encoders and a PID or implement fixed point arithmetics to improve calculation speed enough. If the filtering is improved however the sensor could use the advantage of all eight pixels to command the control signal to steer the robot and also use more advanced interaction. Lastly if the movement is solved another plexiglass plane with two more edge detection sensors could be mounted and implemented to allow the robot to move in both directions without risk to fall off the table.

References

[1] Arduino web site, version dated (2012-07-15), url: http://www.arduino.cc/, 2012.

[2] Björn Carlsson and Per Örbäck. Master thesis: Mobile inverted pendulum - control of an unstable process using open source real-time operating system, 2009.

[3] Robot Electronics. Using the i2c bus, version dated (2012-07-03), url: http://www.robot-electronics.co.uk/acatalog/i2c-tutorial.html, 2012.

[4] Andreas Jansson and Adnan Klempic. Master thesis presentation (2012-03-29): One axle light weight truck - dynamical modeling and implementation of a one axle vehicle, 2012.

[5] M.M. Japp. *Formelsamling i mekanik*. Teknisk mekanik, Chalmers, 2003.

[6] Hans Johannesson, Jan-Gunnar Persson, and Dennis Pettersson. *Produktutveckling: Effektiva metoder för konstruktion och design*. Liber, 2005.

[7] Tom Pycke. Kalman filtering of imu data, version dated (2012-05-23), url: http://tom.pycke.be/mav/71/kalman-filtering-of-imu-data, May 2006.

[8] Wikipedia. Pulse-width modulation, version dated (2012-07-03), url: http://en.wikipedia.org/wiki/pulse-width-modulation, 2012.

[9] Yorihisa Yamamoto. *NXTway-GS Model-Based Design - Control of self-balancing two-wheeled robot built with LEGO Mindstorms NXT*. 2009.

Appendix A

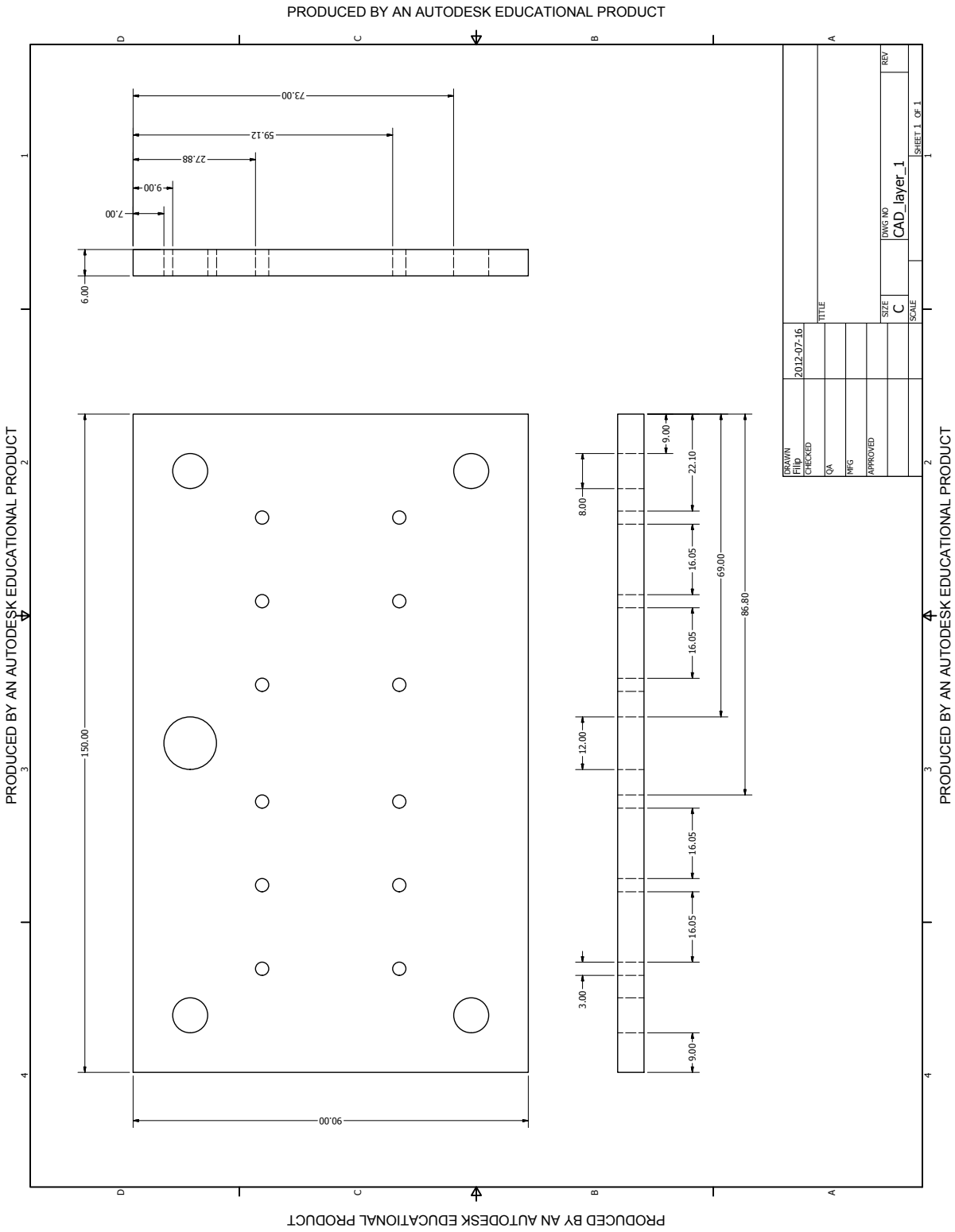
List of components

Components	Number	Price each SEK	Total price SEK
Lawicel			
IMU Fusion ADXL345 and IMU3000	1	486,25	486,25
AC/DC Adapter 12VDC, 1A	1	73,75	73,75
Sharp GP2Y0D810Z0 Distance Sensor	2	43,5	87
Sharp GP2Y0A21K0F Distance Sensor	1	123,75	123,75
Pololu Carrier for Sharp Sensor	2	36,25	72,5
Ardumoto - Motor Driver Shield	1	211,25	211,25
Arduino Stackable Header	1	20	20
Strip Header Str. 1x50 2.54mm	2	14,88	29,76
Strip Header Female 1x40 2.54mm	1	16,13	16,13
Pololu Mounting Hub for 6mm	1	54	54
Arduino Mega2560 R3, DEV-11061	1	486,25	486,25
Logic Level Converter	1	21,25	21,25
Capacitor 10uF/50V 10 pack	1	8,63	8,63
3pin JST cable for Sharp sensor	1	17,5	17,5
LCD 16x2 Ch- RGB BL	1	123,75	123,75
Resistor 330 Ohm 10 pack	1	6,13	6,13
Resistor 10 kOhm 10 pack	1	6,13	6,13
830-point Breadboard	1	55	55
Thermopile Sensor, I2C	1	661,25	661,25
Elfa Distrelec			
USB 2.0 kablage - mini 0.9m	1	39,875	39,88
Elektrolytkond TK 22uF/63V	2	0,9375	1,88
Kontakthus 3-pol 6471	2	4,5375	9,08
Kontaktelement typ 0032	10	1,925	19,25
Vippströmställare on-off 1P, 631H	1	60,4	60,40
DC-adapter, 2.1 mm, 5.5 mm	1	14,7	14,7
LED white, 100053-01	3	7,23	21,69
Light diodes red 3 mm (T1), 204HT	3	2,42	7,26
Screw UNC 4-40x9.53mm	8	1,86	14,88
USB 2.0 cablage 3m	1	26,8	26,8

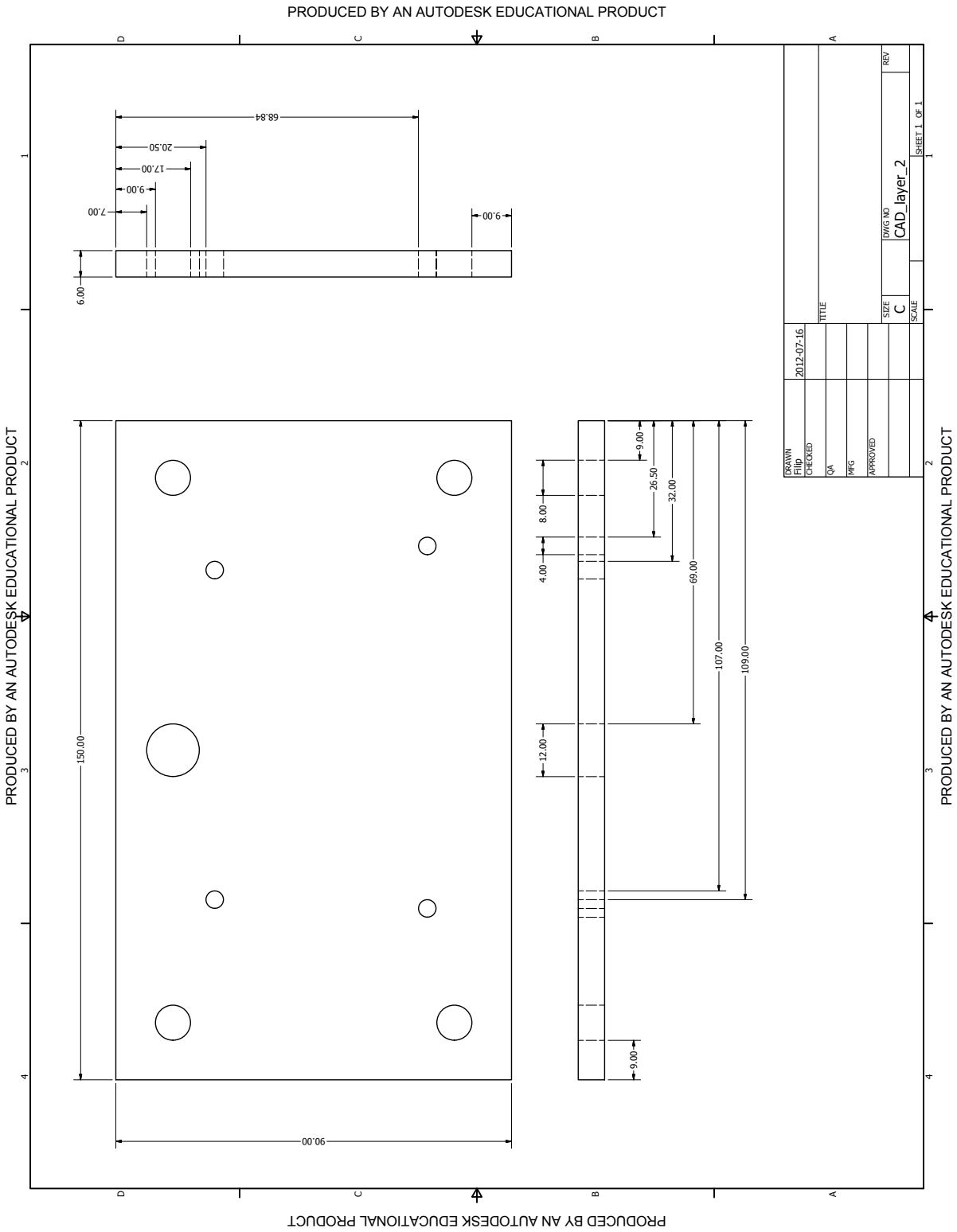
Components	Number	Price each SEK	Total price SEK
Electrokit			
GHM-16 motor - 12vdc 30:1 200rpm	2	213	426
Hylslist svarvad 1x40-pol brytbar	1	13,5	13,5
Hobbytronik			
Wheel Pololu 90x10mm 1 pair	1	78	78
Konsol för 37 mm motorer 1 pair	1	65	65
Clas Ohlson			
Threaded rod	1	16	16
Lock washer M8 25-pack	1	17	17
Screw-nut M8 rostfri 10-pack	3	18	54
Penny washer M8 rostfri 10-pack	3	26	78
Butt Hinge Kantgångjärn 6-pack	1	49	49
Plexiglass 6 mm	4	69	276
Lagerhaus			
Melaminbunke	1	39	39
Hobbyking			
HXT 4mm Gold connector w/ protector	1	24,57	24,57
IMAX B6 Charger 1-6 Cells	1	168,68	168,68
ZIPPY Flightmax 5000mAh 3S1P 20C	1	180,36	180,36
Total cost for the components			4305,20

Appendix B

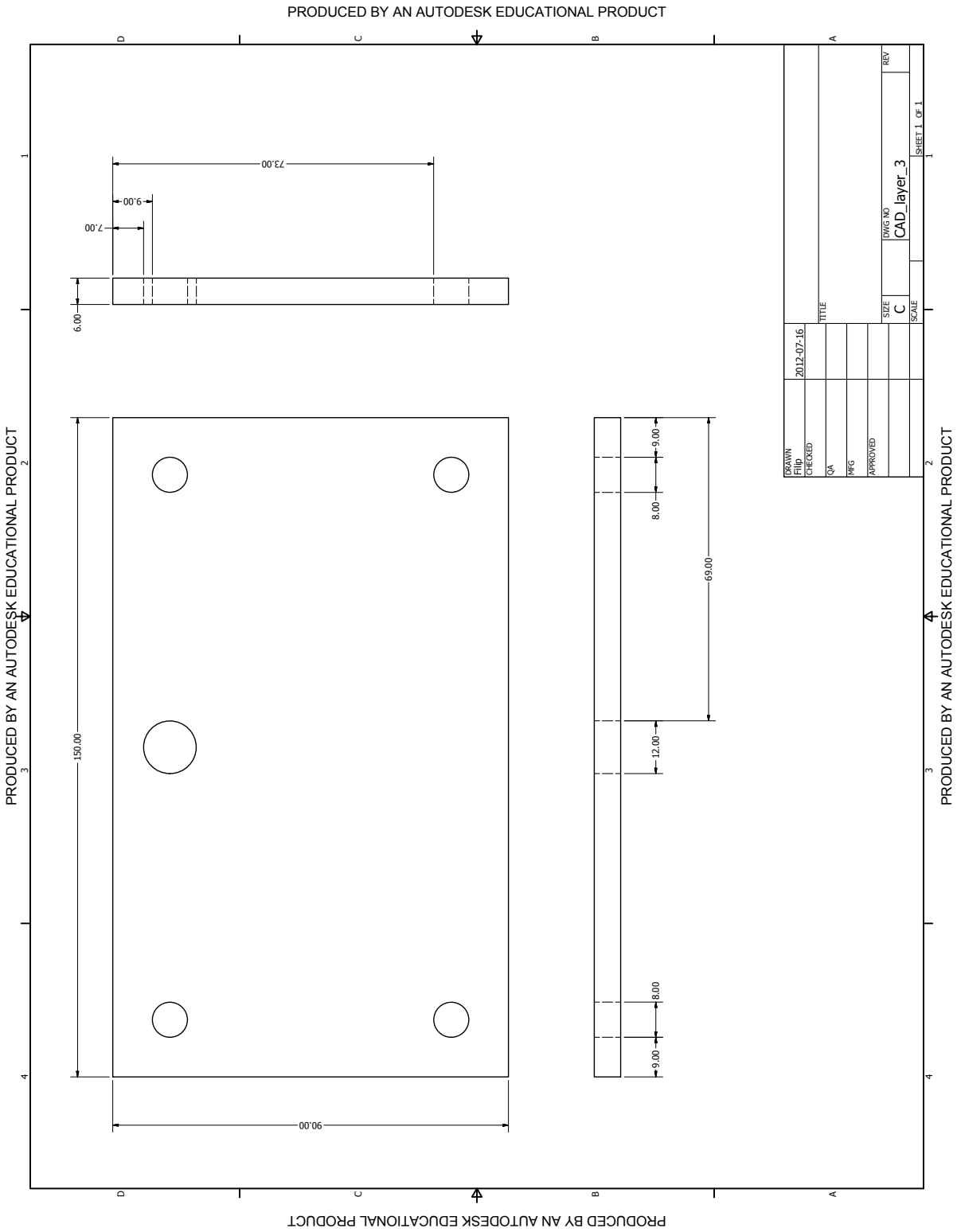
Cad drawing of plexi glass layer 1



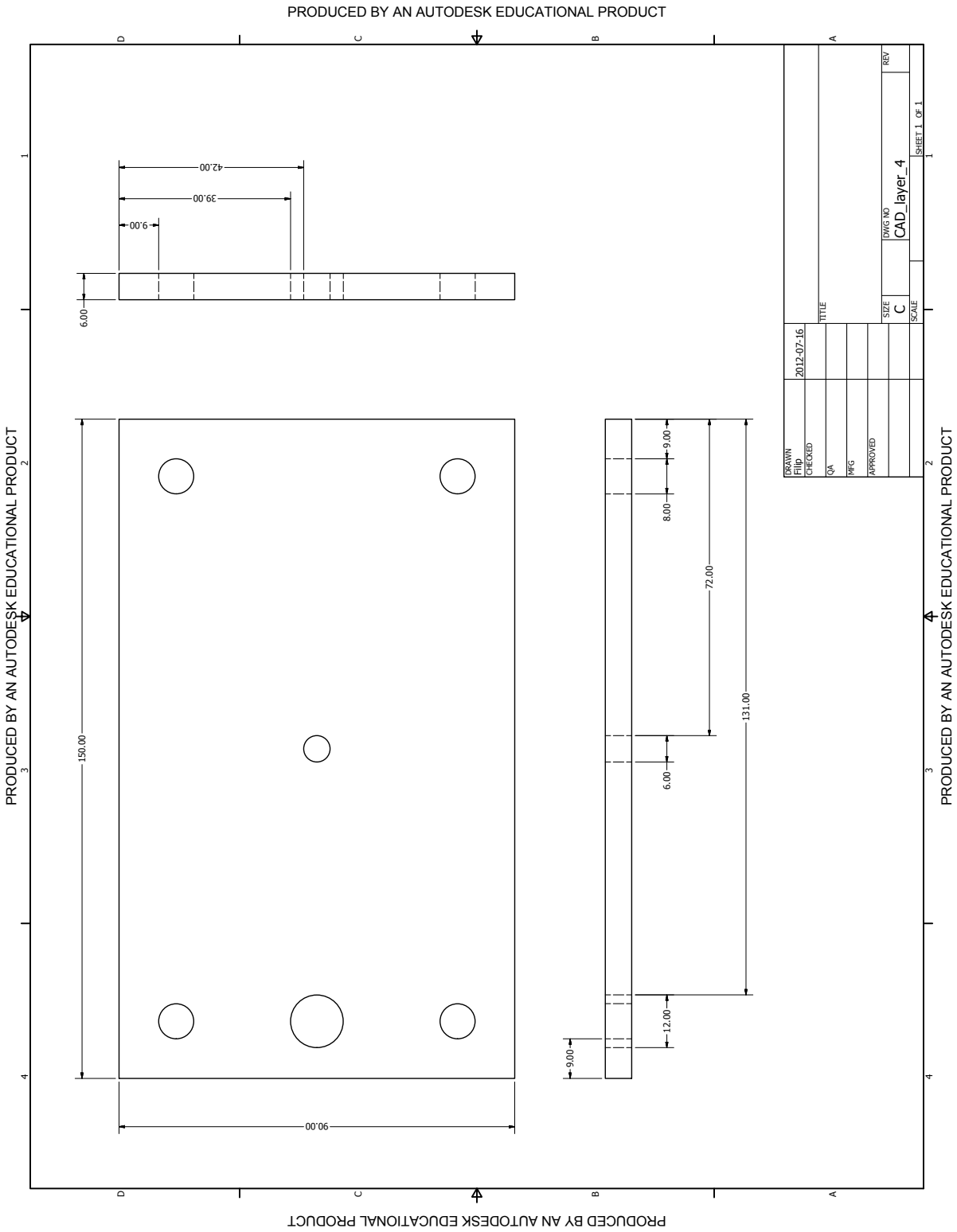
Cad drawing of plexi glass layer 2



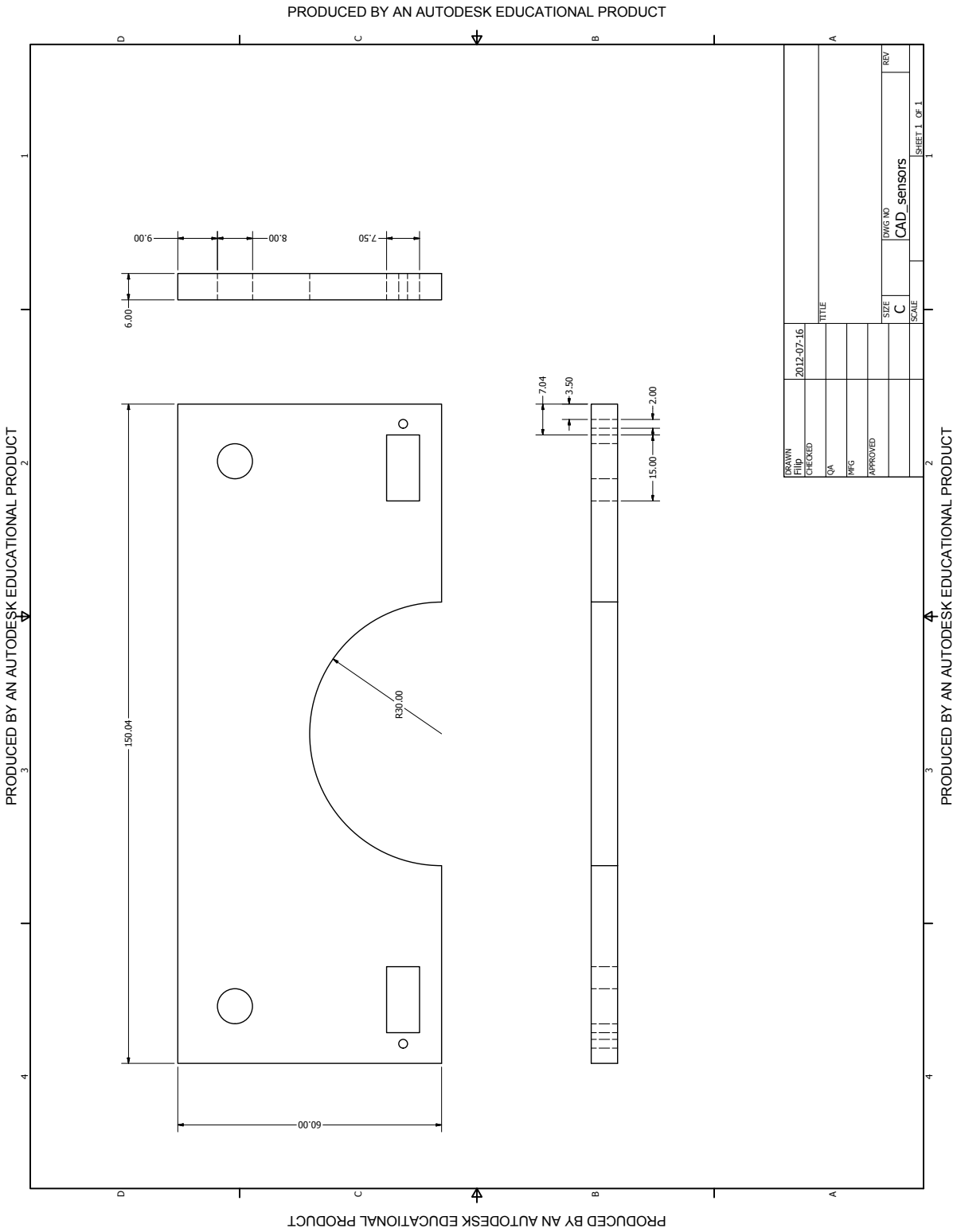
Cad drawing of plexi glass layer 3



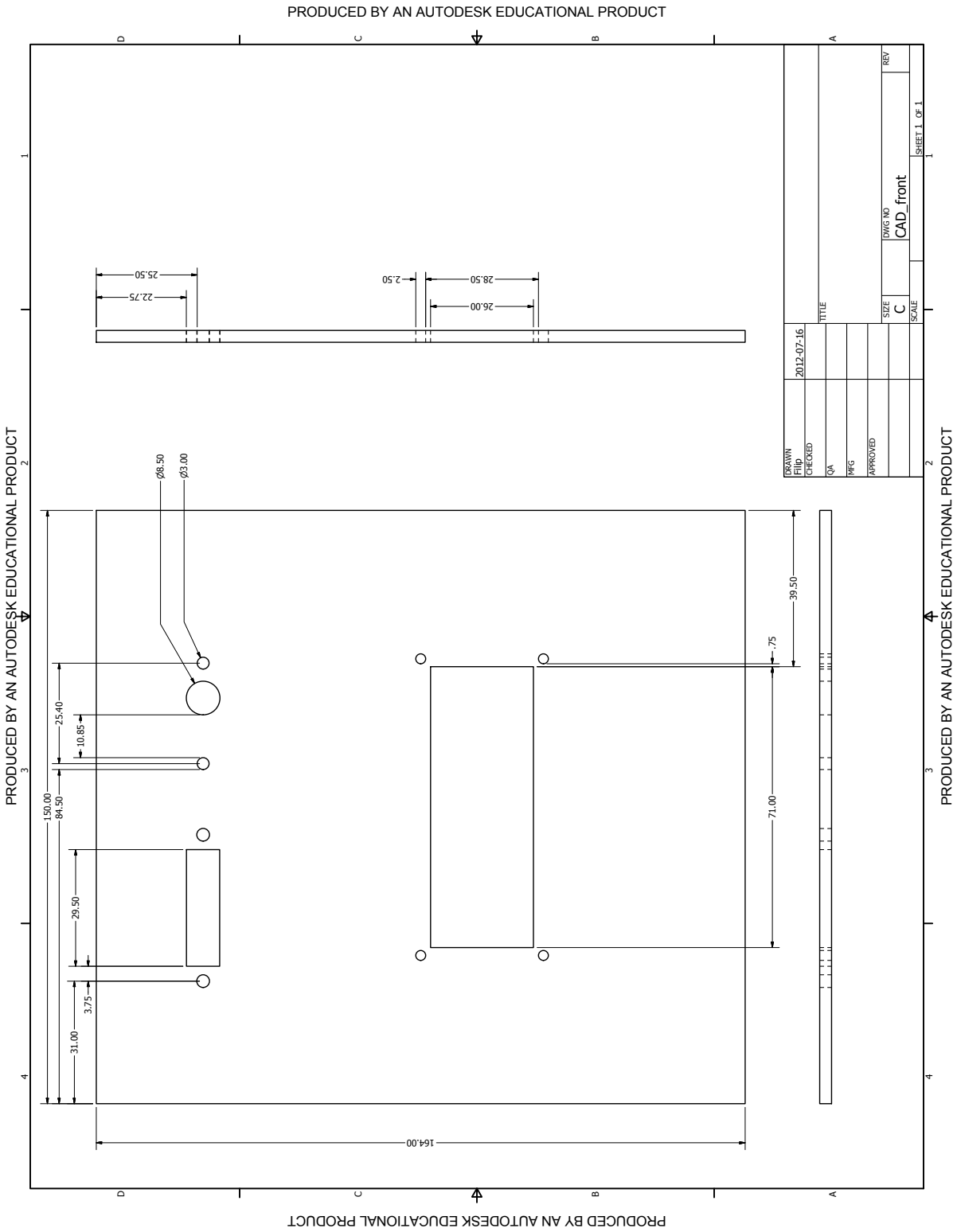
Cad drawing of plexi glass layer 4



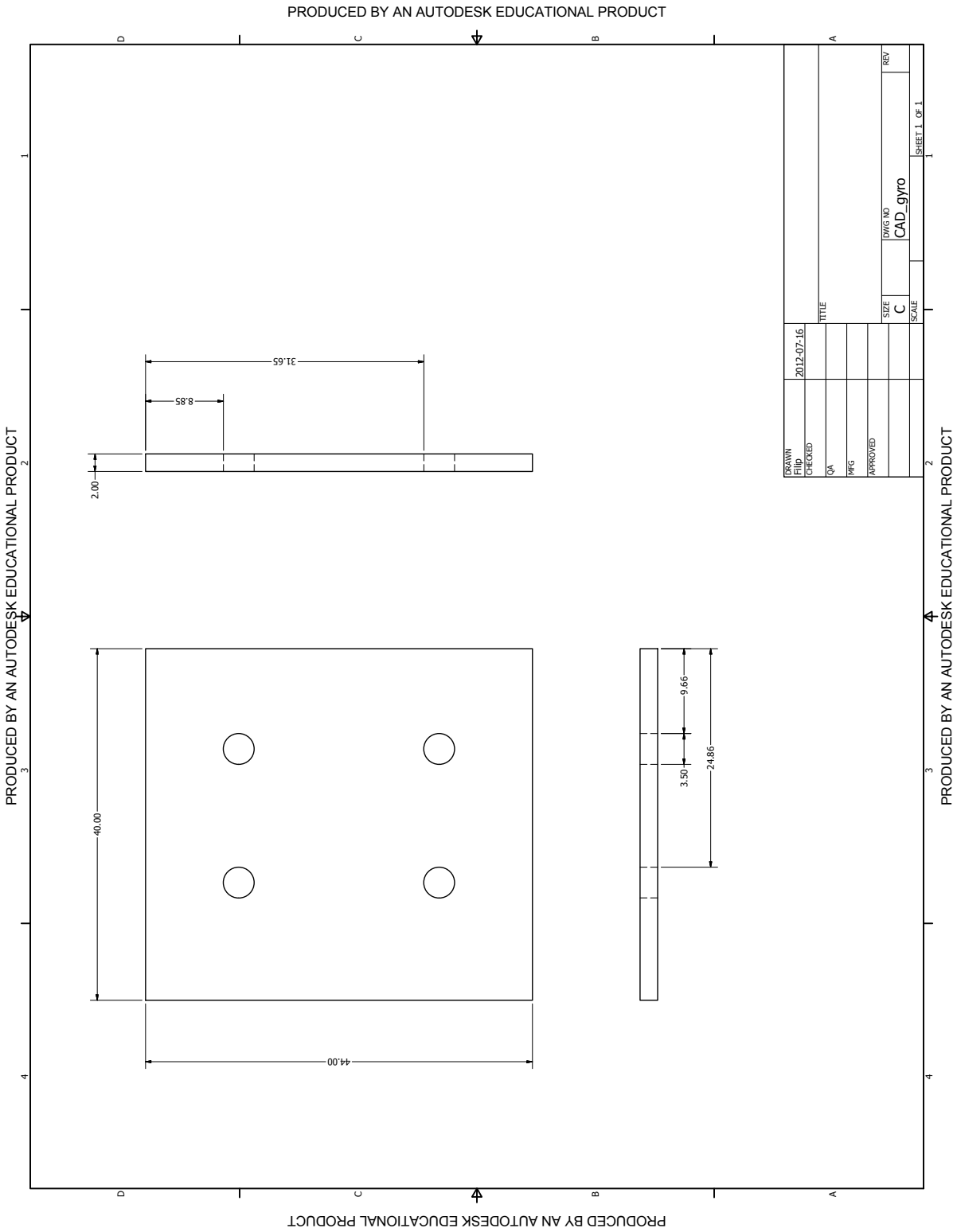
Cad drawing of plexi glass for table sensors



Cad drawing of plexi glass front

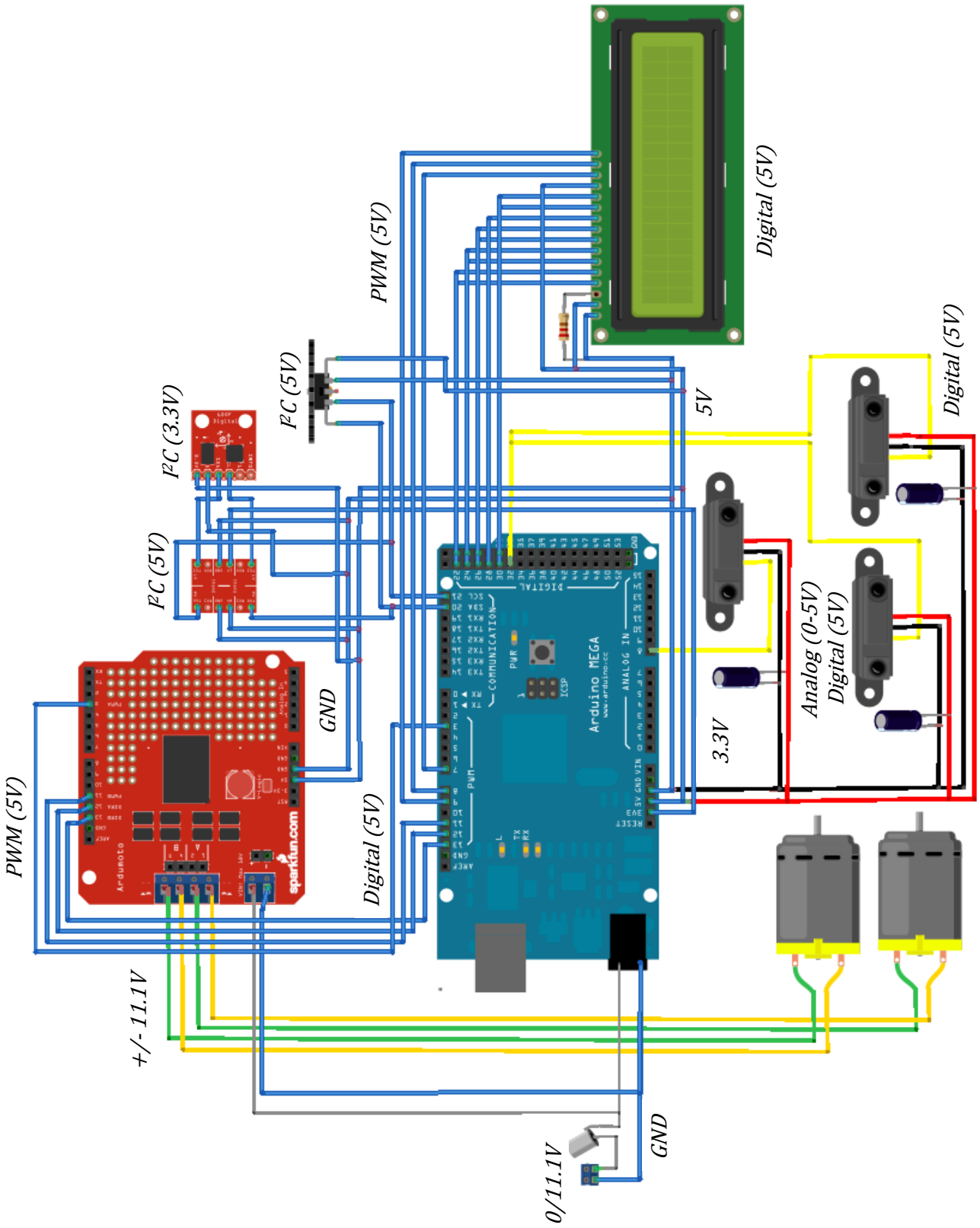


Cad drawing of plexi glass for accelerometer and gyro card

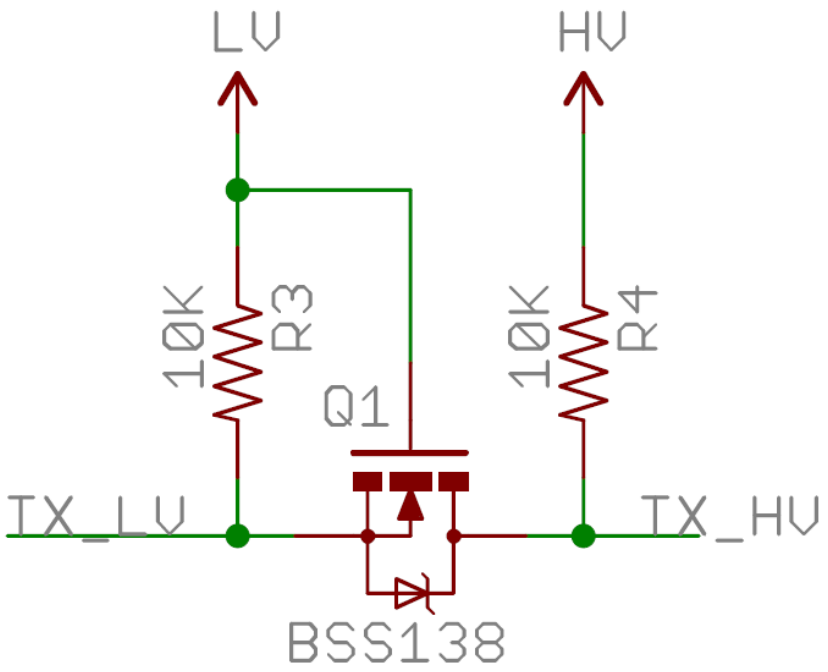


Appendix C

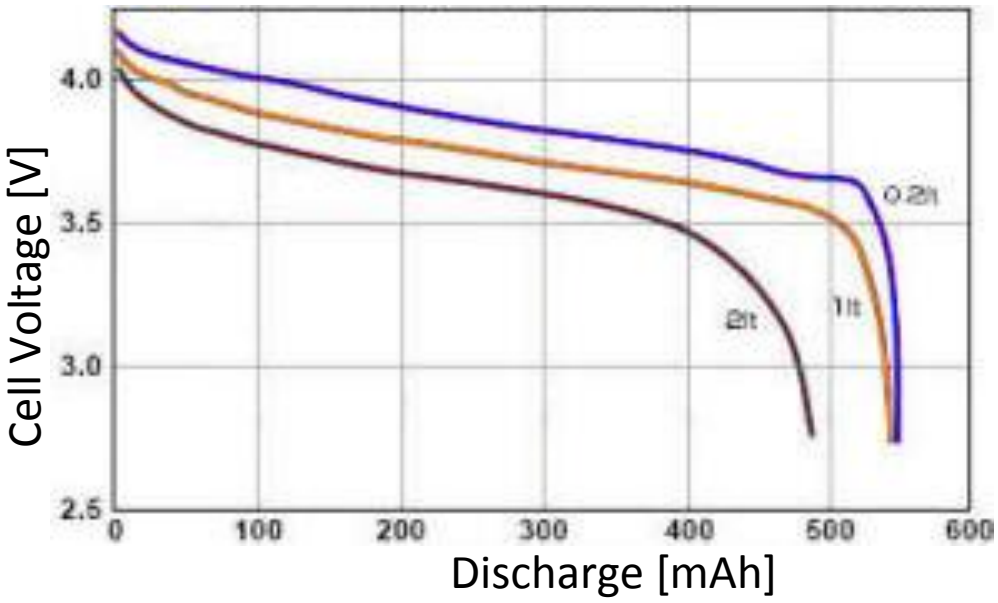
Full overview of the mechatronic system



Levelconverter

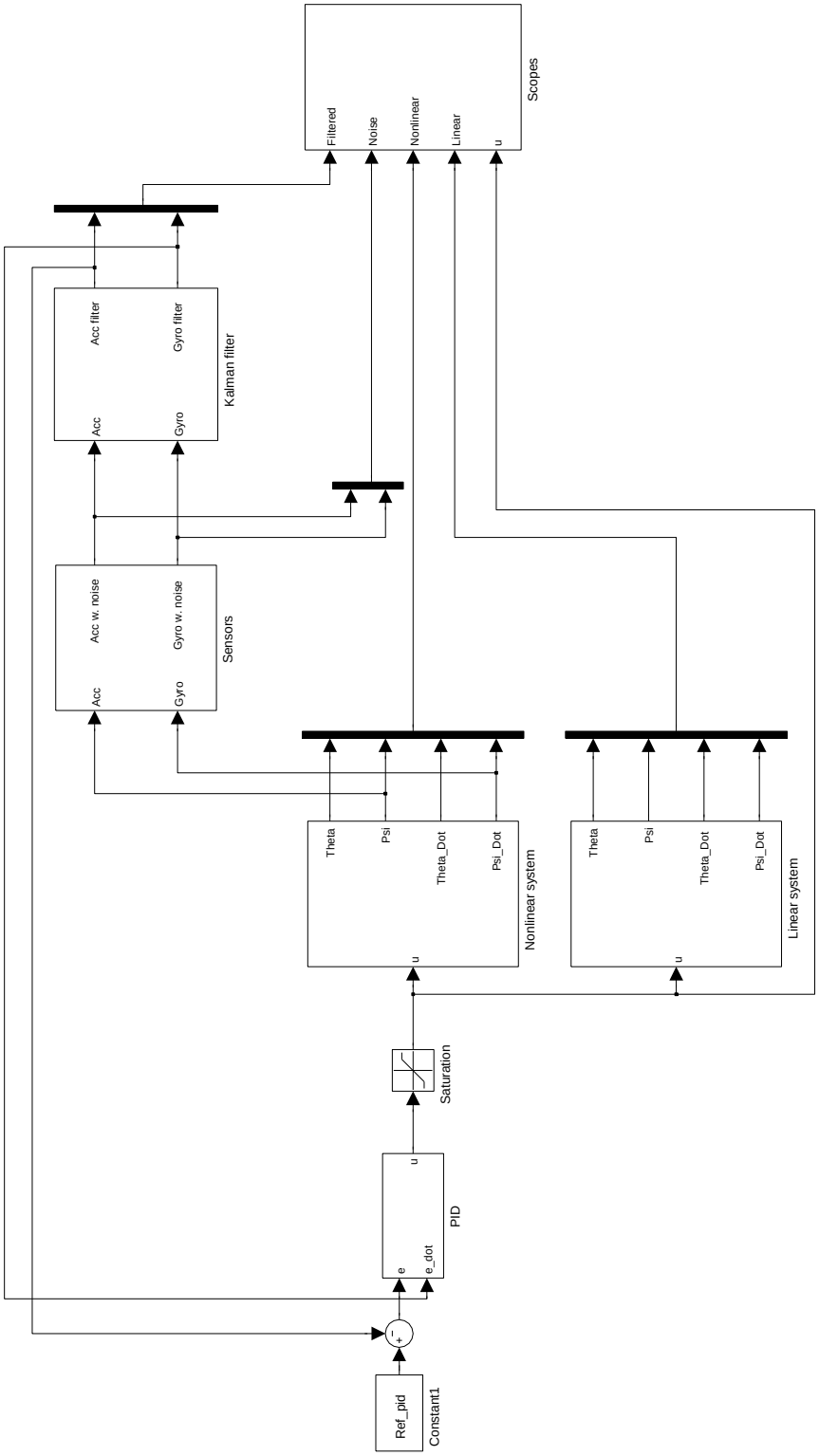


Battery discharge

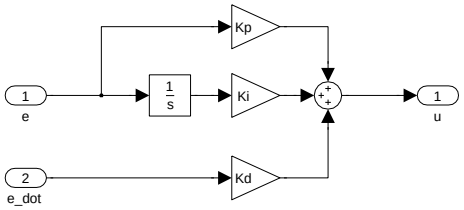


Appendix D

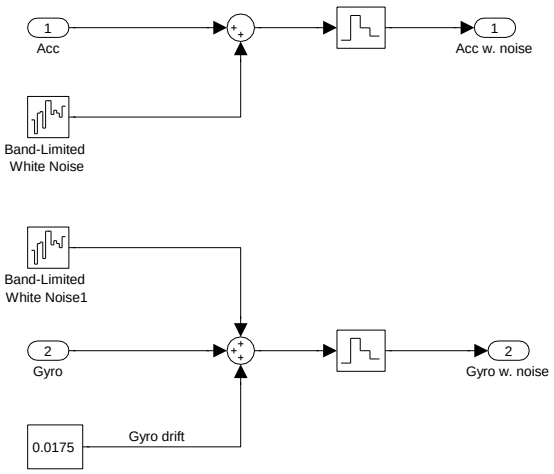
Simulink main model PID



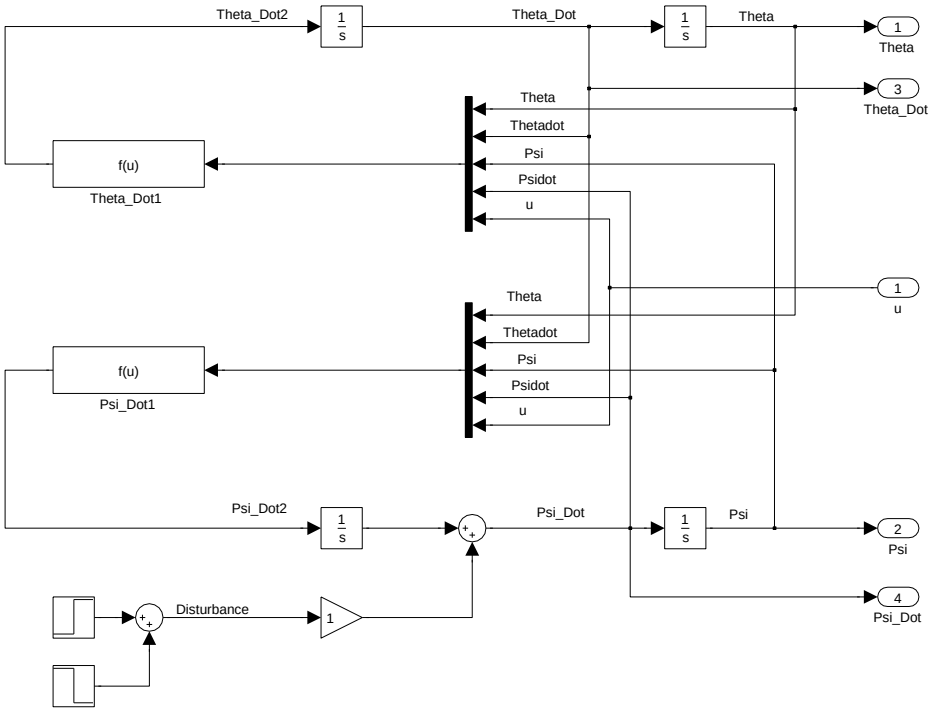
PID controller



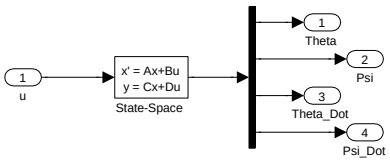
Sensors



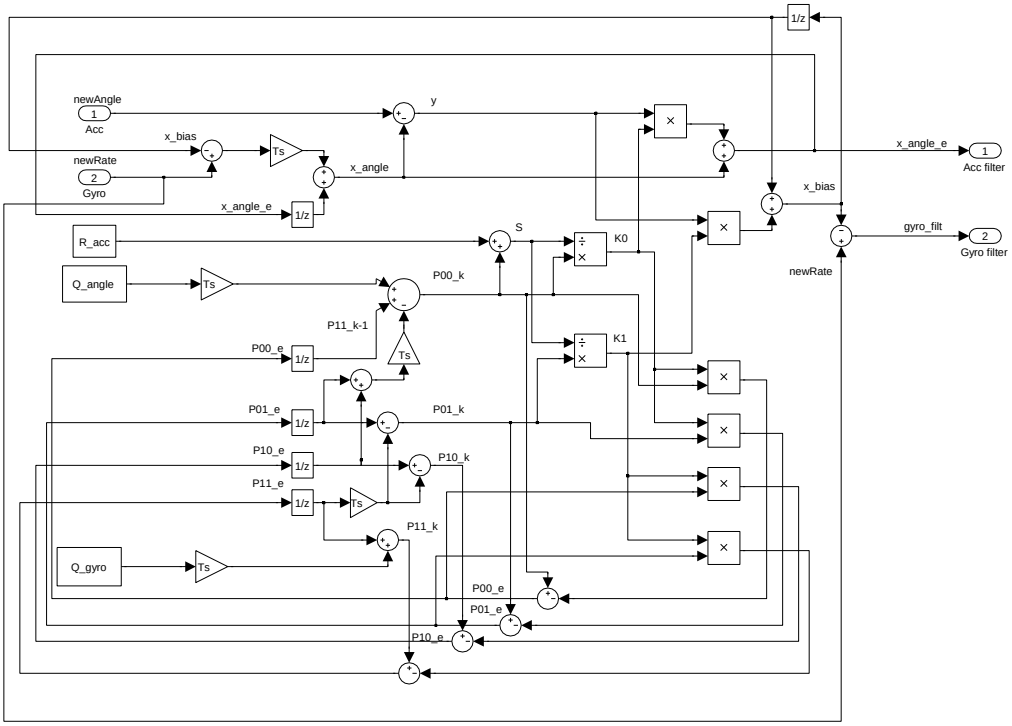
Nonlinear model



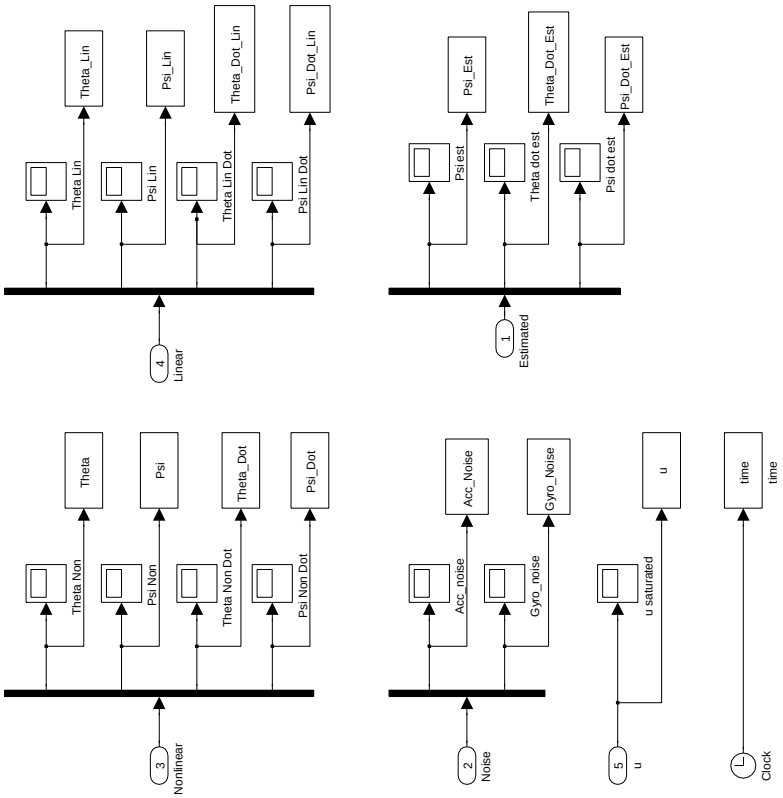
Linear model



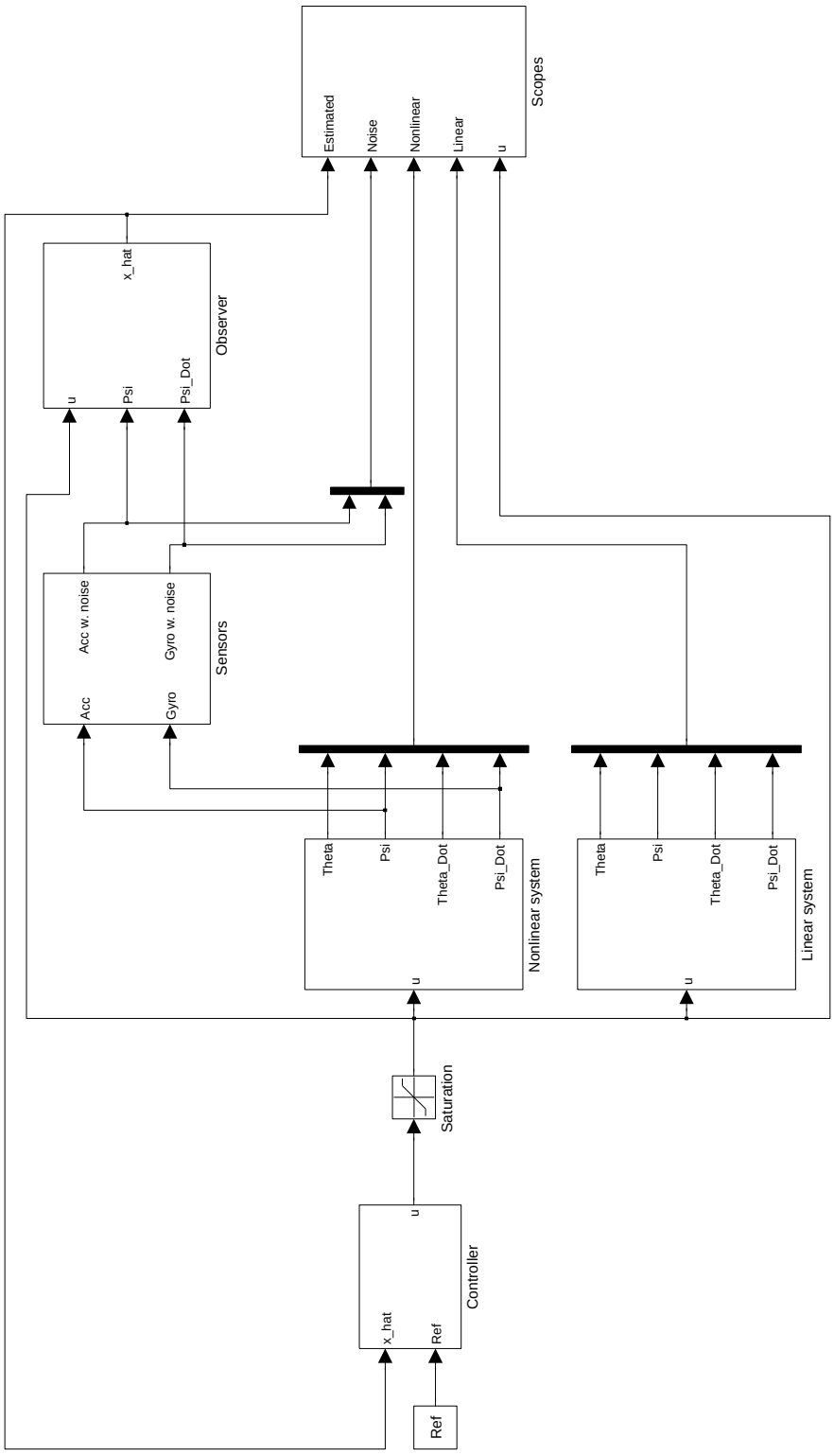
Kalman filter



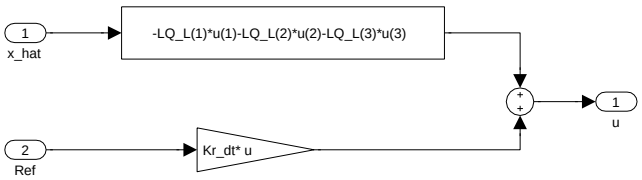
Scopes



Simulink main model LQG



LQ controller



Observer