Two-Wheeled Self-Balancing Robot

Design and control based on the concept of an inverted pendulum

HELLMAN, HANNA
SUNNERMAN, HENRIK

Bachelor's Thesis in Mechatronics

Supervisor:	Martin Edin Grimheden
Examiner:	Martin Edin Grimheden
Approved:	2015-05-20

TRITA MMK 2015:8 MDAB061

Abstract

In the last decade, the open source community has expanded to make it possible for people to build complex products at home. [13] In this thesis a two-wheeled self-balancing robot has been designed.

These types of robots can be based on the physical problem of an inverted pendulum [12]. The system in itself requires active control in order to be stable. Using open source microcontroller *Arduino Uno* and reliable angular and positional data the system can be made stable by implementing a controller.

A modern and efficient controller is the LQR - *Linear Quadratic Regulator* [12]. Being a state space feedback controller the model has to be a good representation of reality since the output signal depends on the model.

In this thesis, the validation process was performed using a PID-regulator. The results showed that the model is not yet reliable. The reasons for this are discussed and recommendations for future development are listed.

Sammanfattning

Framtagning av en tvåhjulig självbalanserande robot

Öppen källkod har under senaste årtiondet möjliggjort för intresserade att bygga avancerade produkter hemma [13]. I denna rapport avhandlas konstruktion och reglering av en tvåhjulig självbalanserande robot.

Denna kategori av robotar kan baseras på problemet för en inverterad pendel[12]. Systemet är i sig självt instabilt och kräver reglering för att balansera. Genom mikrokontrollern *Arduino Uno* och pålitlig vinkel- och positionsdata kan systemet regleras för att uppnå stabilitet.

En modern och effektiv kontroller är LQR - *Linear Quadratic Regulator* [12]. Eftersom denna bygger på tillståndsåterkoppling måste modellen av systemet vara pålitlig då utsignalen baserar sig på modellen.

I denna rapport har valideringsprocessen genomförts genom implementation av en PID-kontroller. Resultaten visade att modellen ännu inte är pålitliig. Anledningen till detta diskuteras och rekommendationer för fortsatt utveckling listas.

Preface

We would like to thank our supervisor Martin Edin Grimheden for guiding us through this project.

We are also grateful for the help from the assistants Sebastian Quiroga, Jimmy Karls and Tobias Gustafsson as well as Staffan Qvarnström for sharing his invaluable experience with us.

Henrik and Hanna
Stockholm, May 2015

Contents

Appendices

Nomenclature

Symbols

Symbols	Description
F	Force (N)
N	Normal Force (N)
x	Cart's position (m)
m	Mass (kg)
ψ	Angle deviation between pendulum and vertical axis (°)
θ	Angle from vertical axis to pendulum (°)
g	Gravitation constant (m/s^2)
l	Distance to pendulum's center of mass (m)
f	Friction constant (m)
M	Torque (Nm)
I	Moment of inertia (kgm^2)

Abbreviations

Abbreviation	Description
CAD	Computer Aided Design
MATLAB	Matrix Laboratory
PWM	Pulse width modulation
LQR	Linear Quadratic Regulator
IMU	Inertial Measurement Unit
I2C	Inter-Integrated Circuit
SISO	Single-Input Single-Output
MIMO	Multiple-Input Multiple-Output

Chapter 1

Introduction

This introductory chapter presents the Background, Purpose, Scope and Methodology of a project conducted within the context of a Bachelor Thesis at the department of Machine Design, division Mechatronics at KTH - Royal Institute of Technology. The project was conducted in the spring of 2015.

1.1 Background

Inverted pendulum applications are plenty; for example the human body is an inverted pendulum balancing the upper body around our ankle joints in every step. In the recent decade *Segways*, making use of bodily movements for steering, have emerged on the market. In common, these applications share that their centre of mass is located above their pivot points, thus requiring active control in order to balance. [9]

The open source community is full of instructions and code snippets, many making use of the open source micro controller *Arduino* for control algorithms. An example is the open source kit Balanduino as seen in Figure 1.1.

Figure 1.1. The open source balancing robot *Balanduino* supplied by TKJ Electronics

In order to balance a two-wheeled inverted pendulum robot it is necessary to have accurate information of the current tilt angle from using a measurement unit. Furthermore a controller needs to be implemented to compensate for said tilt [16]. A PID-controller is able to control the pendulum angle, since it is a SISO - *Single-Input Single-Output* system. [12]

If the robot should be able to be controlled in regard to position, x, as well as the angle, it is a MIMO *Multiple-Input Multiple-Output* system and one PID-controller is not enough. Controlling multiple states is conveniently made through a state space controller. [12]

1.2 Purpose

Prepare the demonstrator for a state space controller to be implemented, that can control the angle deviation, ψ and position, x. See Figure 1.2. A state space description is based on a model of reality. Several assumptions and simplifications must be made. The core part of this project will be to create a model for that purpose and validate whether the model is good enough for implementation of a state space control, this will be done by answering the question:

Using a manually tuned PID-controller, what conclusions can be made regarding the reliability of the model?

From answering this question it can be concluded if the model is accurate enough for future state space control.

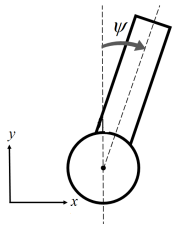

Figure 1.2. Simplified sketch showing the tilt angle, ψ and the position x

1.3 Scope

The process of making a balancing robot is widely documented and open-source code is available. In some parts of this project open-source code has been used and modified to fit the hardware. The given project requirements involved using the micro controller board *Arduino Uno*.

The robot will only be run and tested indoors on flat surfaces. The pendulum is assumed to have one degree of freedom. It will therefore only be controlled in one direction, regarding both the angle of tilt and the position of the robot.

1.4 Method

To fulfil the purpose the following method will be used:

- Derive dynamical equations based on theory of the inverted pendulum
- Form transfer functions for the angle deviation, ψ and position, x
- Find a controller that can control these two conditions
- Set up requirements for the demonstrator
- Design a demonstrator that fulfils these requirements, investigate the boundaries of the control signal

Chosen error sources will be investigated:

- Design a three dimensional model in CAD - *Computer Aided Design* that is as identical as possible to the physical demonstrator to acquire correct parameters needed for the simulated model
- Investigate the accuracy of the sensor that delivers the angular data

With an accurate model of the system and a functioning demonstrator this provides a platform for experiments in a simulated environment.

The simulated model in comparison to the demonstrator will be validated by implementation of a PID-controller in both in order to compare impulse responses.

Chapter 2

Theory

2.1 Model definition

The physical problem of the balancing robot is well described by the widely analysed inverted pendulum. It is commonly modelled as a rigid rod fastened by a frictionless joint to a rigid cart moving in one direction. The simplification that the wheel base can be seen as a cart sliding on a frictionless surface was made. This model definition is inspired by MathWorks tutorial about inverted pendulum [12].

See Figure 2.1 for the simplification steps in this project.

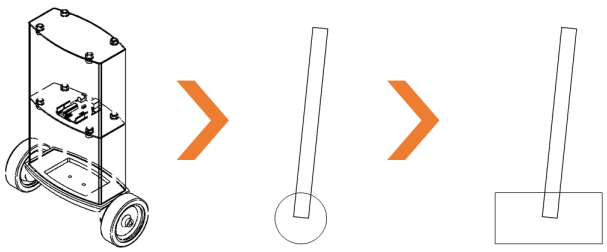

Figure 2.1. Simplification steps of the inverted pendulum

For convenience the pendulum's degrees of freedom are limited to one direction, the angle θ moving in the xy-plane, see Figure 2.2.

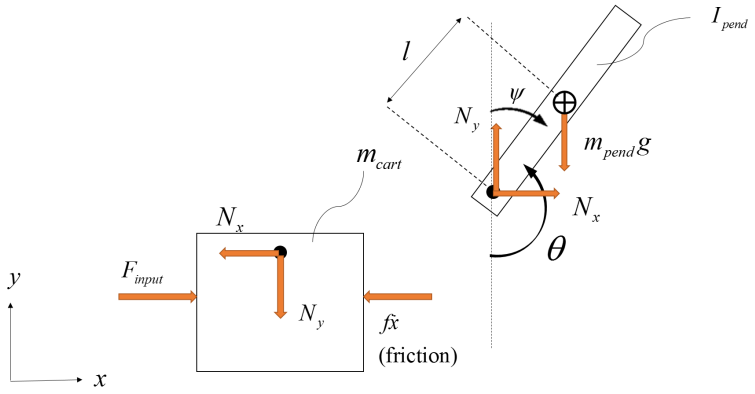

Figure 2.2. Exposure of the simplified inverted pendulum

2.2 Control Theory

Dynamic equations were derived from the forces in Figure 2.2, the complete derivation of the system's transfer function can be found in Appendix A - *Calculation of the system's transfer function.*

$$F_{input} = (m_{cart} + m_{pend})\ddot{x} + f\dot{x} + m_{pend}l\ddot{\theta}\cos\theta - m_{pend}l\dot{\theta}^2\sin\theta \tag{2.1}$$

$$(I_{pend} + m_{pend}l^2)\ddot{\theta} + m_{pend}gl\sin\theta = -m_{pend}l\ddot{x}\cos\theta \tag{2.2}$$

Where m_{cart} and m_{pend} represent the cart's and pendulum's weight respectively. Parameter, l, is the distance to the pendulum's center of mass, θ the angle between the pendulum and the vertical axis as seen in Figure 2.2. The parameter, f is a friction parameter. Rewriting the equations as transfer functions renders:

$$\Psi(s) = \underbrace{\frac{\frac{m_{pend}l}{q}s}{s^3 + \frac{f(I_{pend}+m_{pend}l^2)}{q}s^2 - \frac{(m_{cart}+m_{pend})m_{pend}gl}{q}s - \frac{fm_{pend}gl}{q}}}_{G_\Psi(s)} U_{input}(s) \tag{2.3}$$

$$X(s) = \underbrace{\frac{\frac{(I_{pend}+m_{pend}l^2)s^2 - gm_{pend}l}{q}}{s^4 + \frac{f(I_{pend}+m_{pend}l^2)}{q}s^3 - \frac{(m_{cart}+m_{pend})m_{pend}gl}{q}s^2 - \frac{fm_{pend}gl}{q}s}}_{G_x(s)} U_{input}(s) \tag{2.4}$$

Where Ψ represents the angle deviation, see Figure 1.2 and X the position of the cart. F_{input} from Equation (2.1) has been substituted for the more general control

effort U_{input}. The upper case letters mark the transformation from the time domain to the Laplace domain.

It is concluded that for all positive values of the parameters l, I_{pend}, m_{cart}, m_{pend} and g the system in itself is unstable since it has a pole in the right half plane as can be seen in the pole-zero map in Figure 2.3.

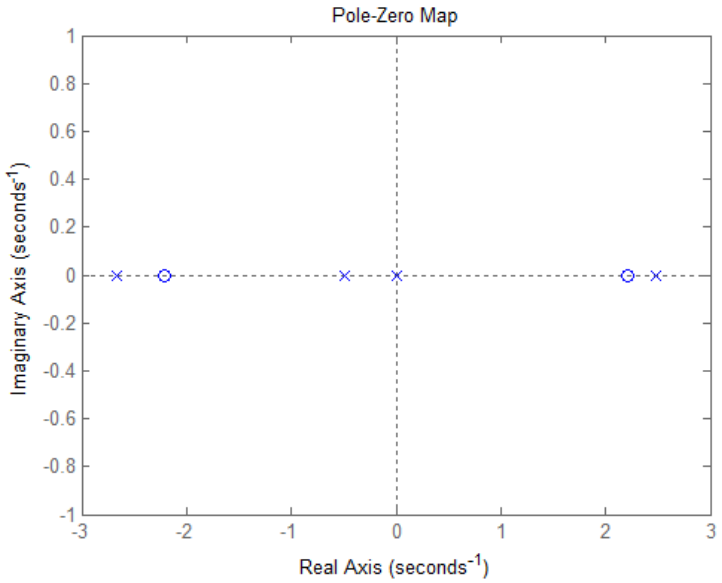

Figure 2.3. MATLAB-generated pole-zero map of the open loop system G_ψ revealing a pole in the right half plane

This arrangement is viable, since an inverted pendulum is intuitively unstable. The differential equations are linearised. According to Glad and Ljung, the system can then also be described in State Space form with the states being $\dot{x}$, $\ddot{x}$, $\dot{\psi}$ and $\ddot{\psi}$.[5]

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{\psi} \\ \ddot{\psi} \end{bmatrix} = A \begin{bmatrix} x \\ \dot{x} \\ \psi \\ \dot{\psi} \end{bmatrix} + B u_{input} \tag{2.5}$$

$$y = C u_{input} \tag{2.6}$$

Where u_{input} is the control effort in time domain. The system matrices A, B and C given by:

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & \frac{-(I_{pend}+m_{pend}l^2)f}{I_{pend}(m_{cart}+m_{pend})+m_{cart}m_{pend}l^2} & \frac{{m_{pend}}^2 gl^2}{I_{pend}(m_{cart}+m_{pend})+m_{cart}m_{pend}l^2} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \frac{-m_{pend}lf}{I_{pend}(m_{cart}+m_{pend})+m_{cart}m_{pend}l^2} & \frac{m_{pend}gl(m_{cart}+m_{pend})}{I_{pend}(m_{cart}+m_{pend})+m_{cart}m_{pend}l^2} & 0 \end{bmatrix} \tag{2.7}$$

$$B = \begin{bmatrix} 0 \\ \frac{I_{pend}+m_{pend}l^2}{I_{pend}(m_{cart}+m_{pend})+m_{cart}m_{pend}l^2} \\ 0 \\ \frac{m_{pend}l}{I_{pend}(m_{cart}+m_{pend})+m_{cart}m_{pend}l^2} \end{bmatrix} \tag{2.8}$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{2.9}$$

For complete calculation of the matrices A, B and C see Appendix A - *Calculation of the system's transfer function.*

The system is controllable if the matrix S, defined as

$$S = \begin{bmatrix} B & AB & A^2B & A^3B \end{bmatrix} \tag{2.10}$$

has full rank. Likewise, the system is observable if the matrix O has full rank, where O is defined as

$$O = \begin{pmatrix} C \\ CA \\ CA^2 \\ CA^3 \end{pmatrix} \tag{2.11}$$

The rank of S and O confirms that the system is controllable and observable for any positive value of l, I_{pend}, m_{cart}, m_{pend} and g.

There are several ways to control this thesis' system if the position x, is disregarded. A possible control method would be a PID-controller [12]. Since there are two states to be controlled, the deviation of the angle ψ and the position, x, cascaded PID-regulators can be used according to Glad and Ljung [5].

Another possibility is to implement some type of state space control. If the states of the system are known, the choice of in-signal can be based on the states inside the system leading to a state space feedback. The state space description holds information of the system's behaviour in past, present and future timespans according to Glad and Ljung. [5]

2.3 LQR, Linear Quadratic Regulator

To control a linear system, LQR-control can be applied. The states of the dynamical system are described by a state space model including the matrices A, B and C as seen in Figure 2.4.

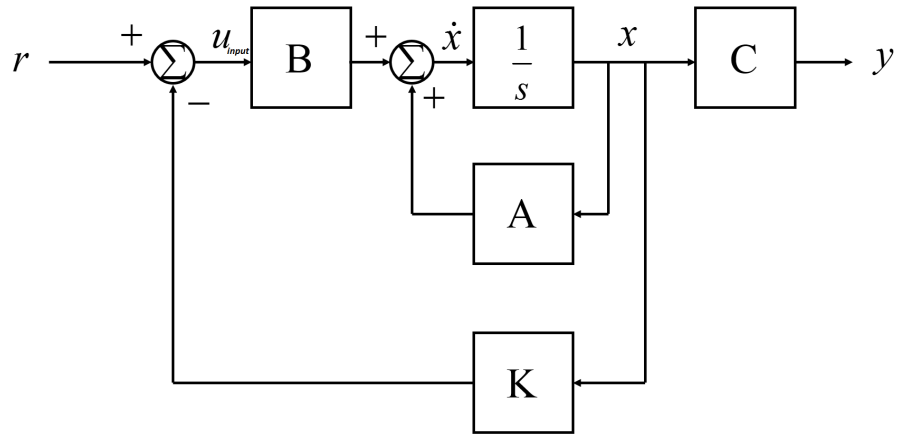

Figure 2.4. Block representation of state space system showing the system matrices A, B and C as well as the gain matrix K.

A cost matrix, Q, is introduced in order to weigh how fast each corresponding element in the state vector, x, need to reach their desired end values. The control effort, u_{input} is given as

$$u_{input} = -Kx \tag{2.12}$$

where K is the gain matrix defined as

$$K = R^{-1}B^T P \tag{2.13}$$

and P is solved from the *Riccatti Equation*

$$Q + A^T P + PA - PBB^T P = 0 \tag{2.14}$$

Since the state space system is observable and controllable, P has a unique solution so that the closed loop system poles are strictly in the right half plane according to Glad and Ljung [5]

It is however necessary to make sure that for a specified change in input, for example an impulse on the robot, that the control effort u_{input} calculated by the controller is within certain bounds.

2.4 Implications of the theoretical research

The system is observable and controllable for any positive value of the parameters l, I_{pend}, m_{cart}, m_{pend} and g. Since both angle and position are to be controlled, all four states of the state space definition need to be measured or some of the states must be estimated. [5]

How measuring the states is accomplished using hardware will be presented in the *Demonstrator* chapter of this thesis.

Chapter 3

Demonstrator

3.1 Problem definition

For this demonstrator an Arduino Uno will be used. As mentioned in 2.4 - *Implications of theoretical research*, in order to implement of an LQR-controller states need to be measured.

It is preferable that the two motors provide at least 300 rpm according to Lövgren. [10] The following needs to be addressed:

- Movement, boundaries of control effort
- Angular data of the pendulum, ψ and $\dot{\psi}$
- Position data of the cart, x and $\dot{x}$

This will be accomplished through hardware presented in this *Demonstrator* section. There will also be a validation process for the model.

3.2 Hardware

3.2.1 Arduino Uno board

The micro controller board used in this project is the *Arduino Uno* running on the ATmega328. The board is used for rapid prototyping of interactive applications through sensors monitoring the outside world. It has 14 digital in-/output pins and 6 analogue inputs. It operates on 5V and communicates via USB-port. The clock speed is 16 MHz and the flash memory is on 32 kB. [2] Chosen components will have to be compatible with the Arduino Uno.

3.2.2 Angular data, ψ and $\dot{\psi}$

As seen in section *Problem definition* some kind of measurement unit is needed to determine the angle deviation, ψ and the angular velocity $\dot{\psi}$. A sensor that fulfils

these requirements is the nine axis gyro/accelerometer, MPU-9150. The gyro can sense the angle velocity in three directions and the accelerometer senses angular acceleration. Only one axis is necessary for this application, although this IMU was available for this project and was therefore chosen.

According to the data sheet [6], the communication between the IMU and the processor is carried out through serial communication via protocol I2C at 400 kHz, with the accuracy of 16-bit which the Arduino can handle.

The MPU-9150 can track both slow and fast motions and it is programmable for different ranges depending on need. In this project the full scale range for the gyroscope is set to 250 °/s and the accelerometer to 2g. It runs on 3.3V (Vcc) which can be provided from the Arduino Uno board's pin with the same voltage. The MPU-9150 is connected to two of the Arduino Uno board's six analogue input pins.

The gyro values will be read in quids (a number between 0 and 1023), this can be translated into angular velocity, $\dot{\psi}$.

As mentioned in the problem definition the angle ψ is also needed. It is calculated with

$$\psi_{i+1} = \psi_i + \dot{\psi}_{i+1} dt \tag{3.1}$$

where dt is the time difference since the last loop.

State estimator - Kalman filter

As seen in Equation (3.1) the gyro angle tends to drift after time. A state estimator used in this project is the *Kalman Filter*. As mentioned in the theory section of this thesis, controlling the system depends on the angle ψ to be known. With better accuracy on this value the probability of attaining a stable system increases. To increase the precision of the deviation angle, ψ, a Kalman filter can be implemented.

Essentially, the Kalman filter is an algorithm that combines data and sorts out unreliable data by continuously estimating a prediction of the future, thus sorting out the data not consistent with this prediction. [4]

This project is accomplished with an open source Kalman Filter code. [8]

Minimising the error - Testing the accuracy of the angular data

The precision of the angular data is essential. Therefore it is necessary to experiment with the IMU together with the estimator, Kalman filter, to ensure that the output data is accurate.

The setup for this tests can be seen in Appendix C - *Test procedures with MPU-9150 and state estimator*.

The results from the experiments showed that the angular data from the sensor and the estimator follows sudden changes of tilt well, and the delay is small enough to be disregarded within the frames of this project. The magnitude of the noise was from statical tests concluded to be less than or equal to 0.03 degrees and is

also disregarded. For discussion of results see Appendix C - *Test procedures with MPU-9150 and state estimator*.

3.2.3 Movement - Motors and Driver shield

Two DC-motors of type *EMG30* were assigned for this project. The motors are equipped with encoders and a 30:1 reduction gearbox. Table 3.1 shows each motor's specifications from the datasheet [3].

Table 3.1. Motor specifications

Rated voltage	12 V
Rated torque	0.147 Nm
Rated speed	170 rpm or 17.8 rad/s
Rated current	530 mA
No load speed	216 rpm or 22.62 rad/s
No load current	150 mA
Stall current	2.5 A
Rated output	4.22 W

According to the problem definition each motor should deliver at least 300 rpm, although the *EMG30* delivers a rated speed of 170 rpm at the rated voltage 12V. Therefore these motors do not fulfil the recommendation regarding speed but they are used because they were the only available motors for this project with built-in encoders.

The motor driver used in this project is the *Velleman VMA03*. It is based on the L298P dual full bridge driver. It is design to fit the Arduino Uno as a shield. It runs on either external power supply or power from the Arduino board. The max current is 2.5 A for each channel.

The built-in encoders on the motors are *Hall sensors*, and they will use two digital pins on the Arduino. They have a resolution of 360° per rotation [15].

Control force limits

The control force u_{input} mentioned in theory and in Appendix A - *Calculation of the system's transfer function* is in fact an applied force that comes from the two DC motors. It can be described as

$$u_{input} = \frac{2M_{motor}}{r} \tag{3.2}$$

Where M_{motor} is the mechanical torque from one motor and r is the radius of the wheels. According to the book Elektroteknik [7] the torque can be expressed as

$$M_{motor} = K_2 \Phi I_A \tag{3.3}$$

Where $K_2\Phi$ is a torque constant of the DC-motor, and I_A is the armature current.

Applying Kirchhoff's law on the circuit of one DC-motor

$$U_A = R_A I_A + K_2 \Phi \omega \tag{3.4}$$

Where U_A is the armature voltage applied to each motor, R_A is the sum of all the resistance in the motor and ω is the angular velocity of the shaft. R_A was measured to be 8.5 Ω.

Parameter $K_2\Phi$ was theoretically calculated from (3.4) using the measured resistance and *no load speed* and *no load current* from the data sheet as well as an armature voltage of 12V. $K_2\Phi$ was calculated to be 0.474 Nm/A.

Equation (3.3) and (3.4) gives

$$M_{motor} = \frac{K_2\Phi}{R_A}\left(U_A - K_2\Phi\omega\right) \tag{3.5}$$

Equation 3.5 in 3.2 gives

$$u_{input} = \frac{2K_2\Phi}{R_A r}\left(U_A - K_2\Phi\omega\right) \tag{3.6}$$

It is assumed that ω at the moment of am impulse on the system, which is the moment that requires the largest control force, is 0 rad/s. Equation (3.6) with ω=0 shows the proportional relation between the control force u_{input} and the armature voltage of the DC-motor, U_A.

$$u_{input} = \frac{2K_2\Phi}{R_A r} U_A \tag{3.7}$$

The proportional relation between u_{input} and U_A is a factor of 1.9. This means that the limit of the control effort for the motors running on 12V is a force of 22N.

A summary of the used components and how they communicate can be seen in Figure 3.1.

3.2.4 Construction of physical demonstrator

The demonstrator was designed both physically and in CAD. The used CAD-program is *Solid Edge*[14]. The theoretical model is dependent on the exact values of the m_{cart}, m_{pend} and I_{pend}. *Solid Edge* assists in finding these parameters easy through the application *Physical properties*. The full assembly of the demonstrator can be seen in Figure 3.2.

The battery is the component with the greatest value of density. The placement of the battery at the top is to make sure that the center of mass is located above the pivot point. All the other components are placed close to each other to avoid interference that comes with longer wires. See Figure 3.3.

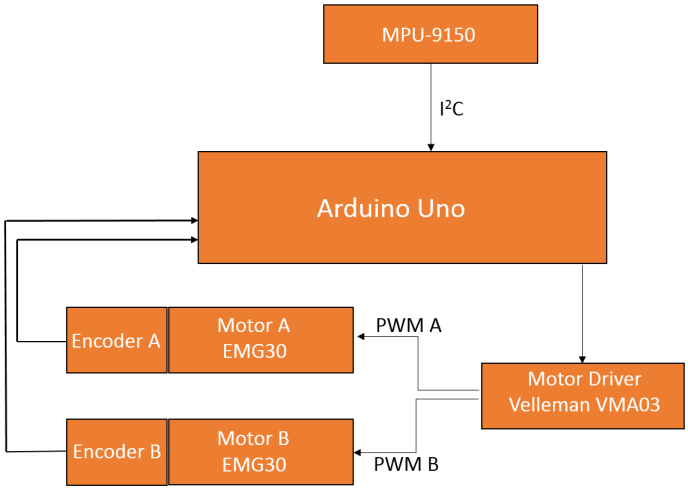

Figure 3.1. Simple schematics of the used hardware and their communication

The aim of the design has been to make the demonstrator and CAD-model as similar as possible, see Figure 3.3. For more information, see Appendix B - *Parameters from CAD-model*

To secure a stable foundation for the robot a bottom plate was designed. Its purpose is to fasten the motor brackets and also to align the gyroscope axis with the motor axis.

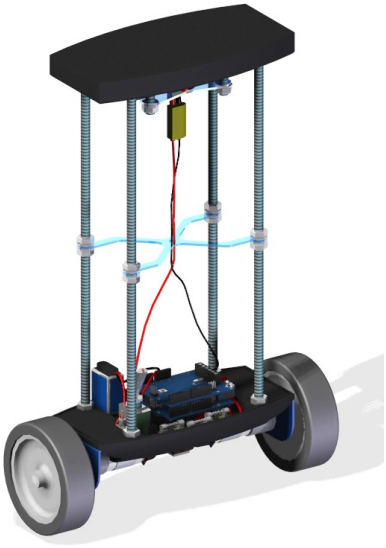

Figure 3.2. The demonstrator in CAD seen from a perspective view

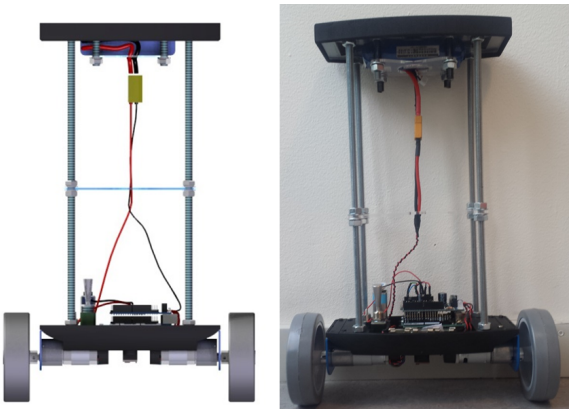

Figure 3.3. The CAD and physical demonstrator seen from front

3.3 Validation process

The derived model that forms the basis for the LQR-controller contains several assumptions and simplifications. It has been justified why these have been made, but it is not possible to determine if they will affect the stability of the system before implementing the controller to the demonstrator.

If the controller is implemented directly and it turns out that there are errors in the model and the demonstrator fails to balance, it is difficult to determine if the

model is close to work or not.

Therefore, a validation process with a PID controller will be made. This is because the demonstrator can be manually tuned to achieve balance, though it can only control the angle deviation as mentioned in the *Theory* chapter of this thesis. It is then possible to compare impulses from the demonstrator to simulated impulses with the same PID-controller. If they behave similarly, the transfer function for the angle deviation, $G_\psi(s)$ is assumed to be valid. Since the transfer function for the position, $G_x(s)$ is based on the same parameters and assumptions, it can also be assumed valid.

If it proves to be a noticeable difference between the real and the simulated impulses, it is assumed that it can be an indication of what the problem in the model is.

The validation process will be performed in these steps

1. Implement a manually tuned PID-controller on the demonstrator
2. Perform impulse experiments and store the serial data from the IMU
3. Insert the same PID-parameters to a simulated PID-controller with the transfer function for the deviation angle,$G_\psi(s)$. See Figure 3.4 for a block representation of the system. Do corresponding impulse tests as made on the physical demonstrator.
4. Compare the results and discuss the reliability of the theoretical model

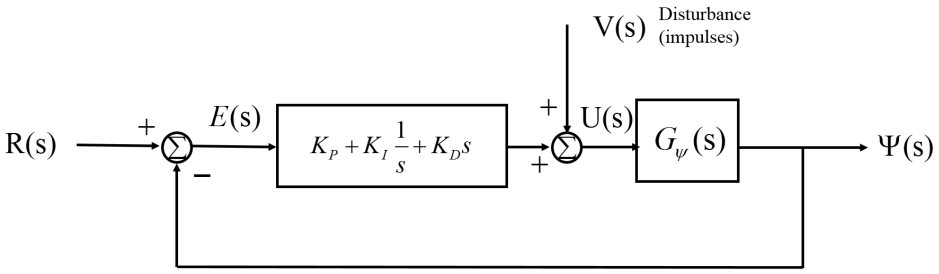

Figure 3.4. Block representation of system with PID controller controlling the angle Ψ. R(s) is the reference signal, E(s) is the error, V(s) is a disturbance signal, U(s) the control effort in the s-domain.

3.3.1 Software

Software for the demonstrator, assembled and created in the Arduino environment is presented schematically in Figure 3.5. An open-source library for Kalman filter

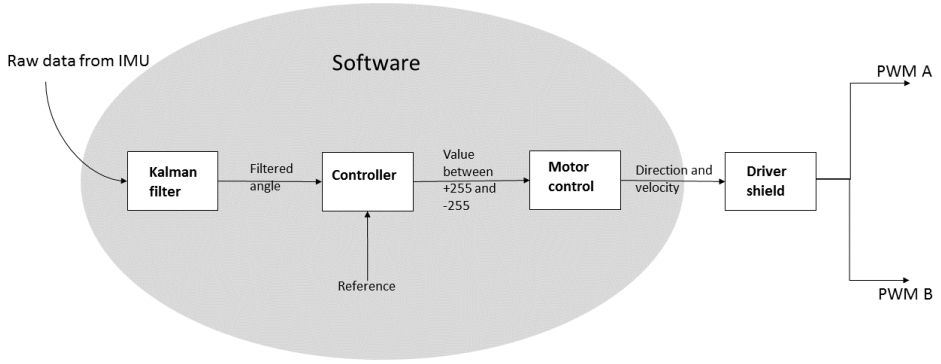

Figure 3.5. Simplified schematics of the software chain used for the validation process

and a PWM-library was used in order to transport the raw data from the IMU all the way to how the motors behave, in terms of direction and PWM.

Chapter 4

Results

The robot's parameters were approximated according to Appendix B - *Parameters from physical model*. The parameters concluded can be seen in Table 4.1.

Table 4.1. Parameters from CAD-model

Parameters	Values
m_{cart}	0.558 [kg]
m_{pend}	1.267 [kg]
I_{pend}	0.055 [kgm^2]
l	0.159 [m]

With these parameters inserted to the MATLAB model, the State Space system looks as

$$\begin{pmatrix} \dot{x} \\ \ddot{x} \\ \dot{\psi} \\ \ddot{\psi} \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & -0.007 & 3.363 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & -0.017 & 30.47 & 0 \end{pmatrix} \begin{pmatrix} x \\ \dot{x} \\ \psi \\ \dot{\psi} \end{pmatrix} + \begin{pmatrix} 0 \\ 0.736 \\ 0 \\ 1,704 \end{pmatrix} u_{input} \tag{4.1}$$

$$y = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} x \\ \dot{x} \\ \psi \\ \dot{\psi} \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \end{pmatrix} u_{input} \tag{4.2}$$

The unstable system can be stabilized by an LQR-controller. Using a cost matrix Q of

$$Q = \begin{pmatrix} \alpha & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \beta & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \tag{4.3}$$

where α and β are weight factors for the position and angle respectively. With their relative sizes the importance of the error and control effort for the respective states can be calibrated. [12] The control gain matrix, K , can be determined using for example the MATLAB function *lqr()* [11]:

$$K = \begin{pmatrix} k_1 & k_2 & k_3 & k_4 \end{pmatrix} \tag{4.4}$$

The proportional factor between the voltage given to the motors, U_A, and the control effort, u_{input}, was calculated to be 1.9. This results in a maximum control effort of $\pm$22N that can be delivered from the motors.

With these preparations an LQR-controller can be implemented for the model if it is ascertained that the control effort stays within the limits of $\pm$22N.

4.1 Validation of the model

Several different impulses were introduced to the demonstrator, resulting in impulse responses plotted from serial angular data. Two representative tests are shown in Figure 4.1. Both impulse responses show similar settling times of approximately 2.2 seconds.

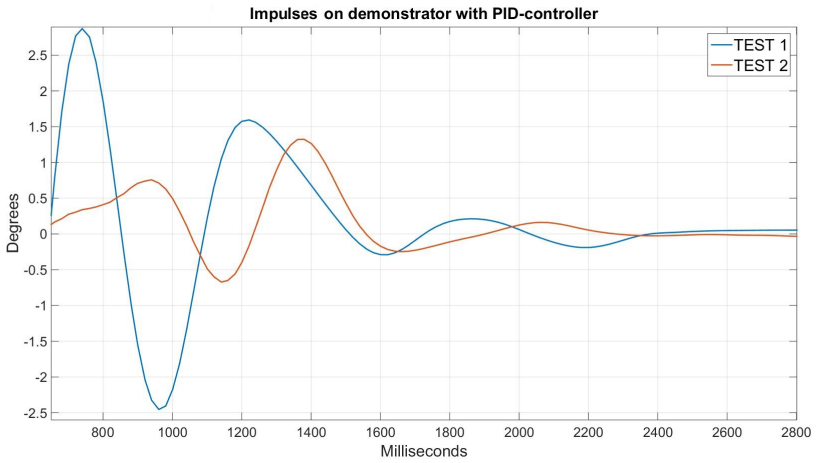

Figure 4.1. Impulse responses on demonstrator

Insertion of the manually tuned PID-parameters K_p=24, K_i=8 and K_d=15 into the simulated model resulted in the impulse response presented in Figure 4.2.

It is clear that a settling time of 12 seconds in the simulated system is not well enough and therefore the simulated model is currently deemed not to be an accurate representation of the demonstrator. Possible reasons for this will be discussed in the *Discussion* chapter of this thesis.

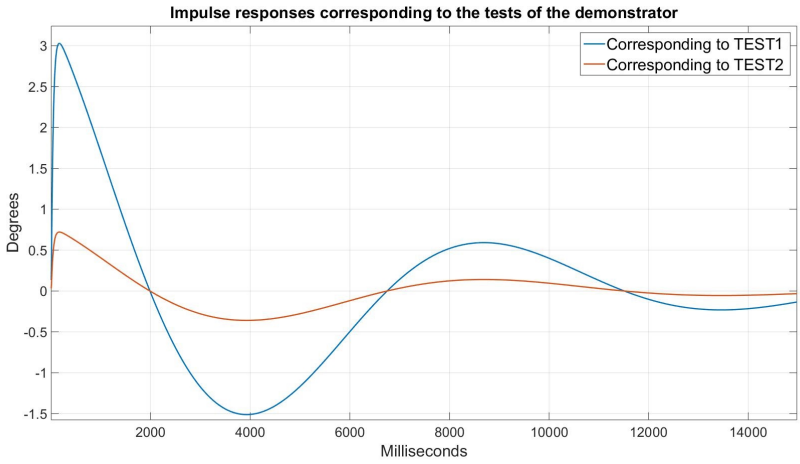

Figure 4.2. Impulse responses on simulated model

Chapter 5

Discussion and conclusions

5.1 Discussion

The results of the validation process show that the model is currently not precise enough to be implemented with LQR-control. The purpose of the validation was that its results could serve as an indication of what in the model causes the error. Since the discrepancy between the model and the demonstrator was concluded to be large regarding the settling time, no unequivocal conclusion of what causes the error was drawn. The cause of the large error most likely lies within some of the assumptions made in the development of the model as listed below:

- The robot only moves in one direction of freedom
- The robot was represented as a pendulum attached to a cart
- The cart was represented as the motors with wheels, and the pendulum was represented as the rest of the components
- The linearization is for angle deviation values close to its upright position
- The friction in the system is reduced to a constant proportional to the velocity of the cart
- The noise and delay in the angle measurement unit is insignificant

Furthermore a compromise regarding the motor speed was made.

The assumption that the robot only moves in one direction of freedom is considered to be realistic since it was seen in the demonstrator that the robot only moves as mentioned. The simplification that the robot consists of a cart and pendulum was made from it being a common assumption in similar projects. In these projects the reliability of that assumption has not been discussed, and that has been perceived as the simplification being reasonable.

The parameters m_{pend}, I_{pend}, m_{cart} and the distance to the centrum of mass l was drawn from the CAD-model. The difference between the demonstrator and the

CAD-model is not notable, so the values of these parameters are deemed correct. Although the assumption of what part of the model the components belong to is difficult to determine. The idea was that the components belonging to the pendulum should all follow the movement of the pendulum and the components related to the cart was rigid relative to the pendulum.

The linearisation of the equations, which can be seen in Appendix A - *Calculation of the system's transfer function* was based on the approximation that the pendulum only will move in small deviations according these two approximations

$$\cos\theta = \cos(\pi + \psi) \approx -1 \tag{5.1}$$

$$\sin\theta = \sin(\pi + \psi) \approx -\psi \tag{5.2}$$

The definition of a small deviation might seem unclear; See Table 5.1 where the limits for the approximation are investigated.

Table 5.1. Approximation errors from linearisation

Deviation	Approximation (5.1)	Error	Approximation (5.2)	Error
1°	-0.999	0.001	-0.017	0.000
5°	-0.996	0.004	-0.087	0.000
10°	-0.985	0.015	-0.174	0.001
20°	-0.940	0.060	-0.342	0.007
30°	-0.866	0.134	-0.500	0.024

Since the biggest impulse in the validation process lead to 8° deviation, the errors from the linearisation is barely noticeable. The table demonstrates that the linearisation leads to bigger errors with bigger deviations.

The friction in the system is summarized as a constant, f proportional to the velocity of the cart. The value of this parameter is taken from a similar project [12]. How they came up with this value was not motivated. It was tested to vary this parameter, but better results were not achieved. It is probably not the parameter itself that is inaccurate. It is rather the assumption that all friction in the system can be reduced to a parameter proportional to the velocity that is the problem. The friction in the system is more complex than that. In fact, there are several parts of the model where friction should be added. For example the motor has a static and dynamic friction, and this is probably an important factor since the motors change direction often. For future work, the mentioned needs to be investigated for a more accurate model.

The assumption that the noise and delay of the angular data could be ignored was made because adding it to the model would make it more complex, and it was assumed that it would not make a noticeable difference though these disturbances was minor. Though, for future work, this is recommended. Especially if other measurement unit will be used, the model will be more flexible.

The lacking speed of the motor is a problem. Though the demonstrator archived to balance with a PID-control it could not handle bigger impulses than shown in the results. For future work, a faster motor is recommended.

5.2 Conclusions

Modelling based on the inverted pendulum shows that the system is unstable without a controller. A simple PID-controller can be implemented, but only to control the angle deviation, ψ.

Since the demonstrator can receive angular data and position data, it is possible to get data for all the states that are required for a state space controller.

The results of the validation process demonstrates that the model is not reliable. That conclusion was made because of the big difference in settling time of the impulses between the physical demonstrator and the corresponding simulation. Reasons why the model in not reliable was discussed, where it was recommended that the friction of the system should be investigated more closely.

All required hardware for a robot controlled via LQR is mounted on the demonstrator, so if the errors in the model are handled, the demonstrator is fully prepared for implementation. Though, if the demonstrator is supposed to handle impulses bigger than 8°, faster motors are recommended.

Chapter 6

Recommendations and future work

6.1 Recommendations

The used motors, *EMG30* have a rated speed of 170 rpm, it was determined that it was too slow if the robot is going to handle impulses of greater magnitude than 8°. Eventually, when an LQR-controller is implemented, the robot should also be able to control steps. If the demonstrator cannot handle small impulses it would probably not be able to handle steps either. Therefore, the recommendation is to acquire faster motors.

6.2 Future work

As mentioned, the friction in the system needs to be investigated and added to the model. When the model works properly the demonstrator is ready to implement the theoretically derived LQR-control. With LQR the system will become faster and can handle both disturbances as impulses and steps. For example, see Figure 6.1. It is also possible to change the reference of the position that enables navigation of the robot.

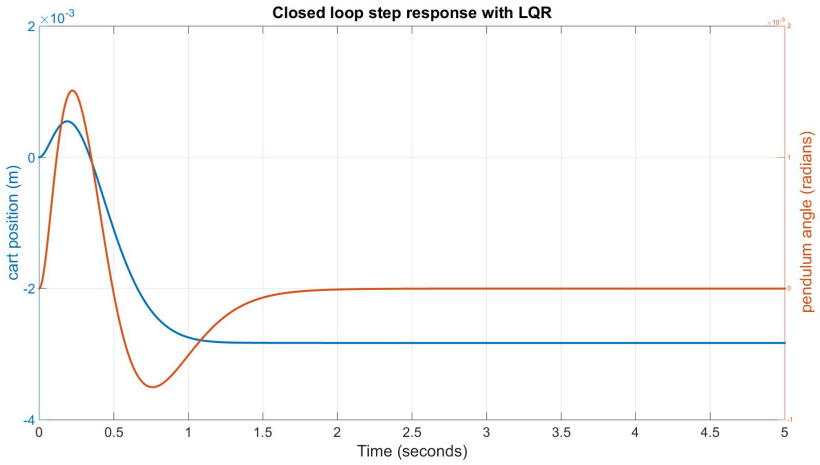

Figure 6.1. Step response of 0.2 rad with an LQR regulator

Here, a cost matrix of Q is used.

$$Q = \begin{pmatrix} 5000 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 100 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \tag{6.1}$$

A step of magnitude 0.2 radians on the pendulum was applied to the simulated LQR-controlled system. The step response showing the behaviour of the cart's position and the deviation angle of the pendulum is represented in Figure 6.1. The pendulum angle settles at 0° tilt, and the cart's position stabilises only 5 mm from its origin. All this within the first 2 seconds. It is important to make sure that the control effort is within the limits calculated to be ±22N.

Chapter 7

References

[1] AMCI Tech Tutorials. 2015. *Stepper vs. Servo.* Available at: http://www.amci.com/tutorials/tutorials-stepper-vs-servo.asp. [Accessed 13 May 2015].

[2] Arduino Uno. 2015. *Arduino - ArduinoBoardUno* . Available at: http://www.arduino.cc/en/main/arduinoBoardUno. [Accessed 22 April 2015]

[3] EMG30. 2010. *Data sheet.* Available at: http://www.robot−electronics.co.uk/htm/emg30.htm [Accessed 22 April 2015]

[4] Faragher, Ramsey. 2012. *Understanding the Basis of the Kalman Filter Via a Simple and Intuitive Derivation.* IEEE signal processing magazine.

[5] Glad, Torkel. Ljung, Lennart. 2006. *Reglerteknik - Grundläggande teori.* 4th edn. Studentlitteratur.

[6] InvenSense Inc. 2013. *MPU-9150 Product Specification.* Sunnyvale, USA. Revision 4.3. Available at: http://www.invensense.com/mems/gyro/documents/PS-MPU-9150A-00v43.pdf. [Accessed 22 April 2015]

[7] Johansson, Hans. Lindahl, Per-Erik. Meyer, Robert. Grimheden, Martin. Sandqvist, William. Paulsson, Magareta. 2013. *Elektroteknik.* KTH

[8] Lauszus, Kristian. 2012. *Kalman Filter.* Available at: https://github.com/TKJElectronics/KalmanFilter [Accessed 20 Mars 2015]

[9] Loram, I, 2002. *Human balancing of an inverted pendulum: position control by small, ballistic-like, throw and catch movements.* Journal of Physiology, 540.3, 1111.

[10] Lövgren, Samir. 2015. *Self balancing Robot.* Available at: http://samirlovgren.se/pageid647. [Accessed 11 May 2015].

[11] MathWorks Nordic. 2015. *MATLAB - The Language of Technical Computing*. Available at: http://se.mathworks.com/products/matlab/. [Accessed 21 April 2015]

[12] MathWorks. 2012. *Control Tutorials for MATLAB and SIMULINK, Inverted Pendulum*. Available at: http://ctms.engin.umich.edu/CTMS/index.php.exampleInverted Pendulumsection-SystemModeling. [Accessed 21 April 2015]

[13] Redhat. 2015. *What is open source software*. Available at: https://opensource.com/resources /what-open-source. [Accessed 13 May 2015].

[14] Siemens PLM Software. 2015. *Solid Edge*. Available at: http://www.plm.automation.siemens.com /en-us/products/solid-edge/. [Accessed 21 April 2015]

[15] Velleman. 2013. *Motor and power shield for Arduino*. Gavere, Belgium. Available at: http://www.velleman.co.uk/manuals/vma03.pdf [Acessed 29 Mars 2015]

[16] Jin, D. 2015. *Development of a Stable Control System for a Segway*. Available at: http://www.raysforexcellence.se/wp-content/uploads/2013/01/Dennis-Jin-Development-of-a-stable-control-system-for-a-segway.pdf. [Accessed 02 May 2015]

Appendix A

Calculation of the system's transfer functions

To simplify the model, we assume that it can be represented as an inverted pendulum attached to a cart. Calculations about inverted pendulum attached to a cart is common and can be found on several websites on the internet, this will help ensure that these calculations are correct. The following calculations were inspired by MathWorks [12].

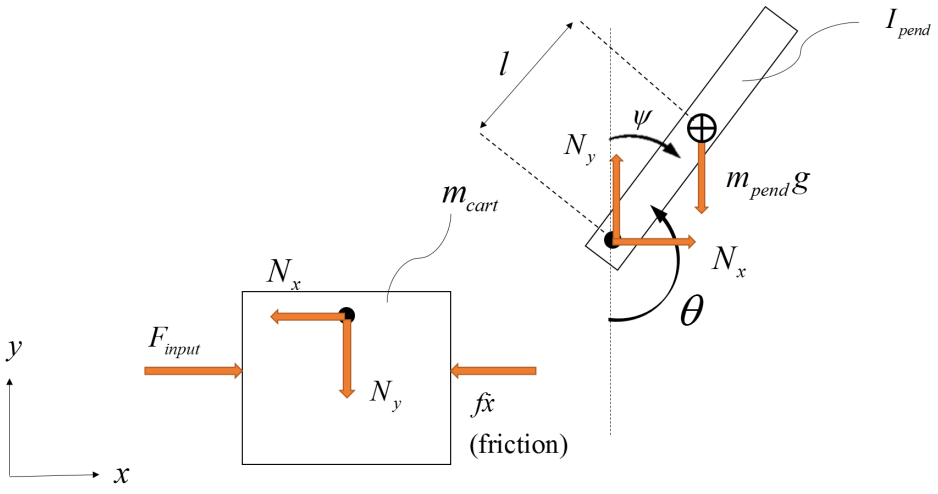

Figure A.1. Exposure of the simplified system with cart and pendulum

A.1 Equations for the cart

The only equation needed from the cart is the one who sums the forces in the x direction

$$F_{input} = m_{cart}\ddot{x} + f\dot{x} + N_x \tag{A.1}$$

where F_{input} is an applied force, m_{cart} is the mass of the cart, f is a constant of friction and N_x is the contact force in the axis between cart and pendulum in x-direction. See Figure A.1.

A.2 Equations for the pendulum

The sum of forces in the x direction on the pendulum is

$$N_x = m_{pend}\ddot{x} + m_{pend}l\ddot{\theta}\cos\theta - m_{pend}l\dot{\theta}^2\sin\theta \tag{A.2}$$

where m_{pend} is the mass and l is the distance to the center of mass. Variable θ is the angle between a vertical line and the pendulum. See Figure A.1 for details. The sum of all forces perpendicular to the pendulum is

$$N_y\sin\theta + N_x\cos\theta - m_{pend}g\sin\theta = m_{pend}l\ddot{\theta} + m_{pend}\ddot{x}\cos\theta \tag{A.3}$$

where N_y is the force in y direction and g is the gravity constant. Summarization the torque acting on the center of the pendulum gives the following equation

$$-N_y l\sin\theta - N_x l\cos\theta = I_{pend}\ddot{\theta} \tag{A.4}$$

where I_{pend} is the moment of inertia of the pendulum.

A.3 Combining the equations

Insert equation (A.2) in (A.1) gives the following equation

$$F_{input} = (m_{cart} + m_{pend})\ddot{x} + f\dot{x} + m_{pend}l\ddot{\theta}\cos\theta - m_{pend}l\dot{\theta}^2\sin\theta \tag{A.5}$$

Combine equations (A.2) and (A.3)

$$(I_{pend} + m_{pend}l^2)\ddot{\theta} + m_{pend}gl\sin\theta = -m_{pend}l\ddot{x}\cos\theta \tag{A.6}$$

A.4 Linearising the equations

Equation (A.5) and (A.6) is necessary to get transfer functions for the position x and the angle deviation ψ. To compute the transfer functions the equations need

to be linearised. A proper equilibrium point would be when the pendulum is in upright position. The angle will represent the deviation of the pendulum from the equilibrium. The following approximations for small deviations will be used in the nonlinear equations (A.5) and (A.6).

$$\cos\theta = \cos(\pi + \psi) \approx -1 \tag{A.7}$$

$$\sin\theta = \sin(\pi + \psi) \approx -\psi \tag{A.8}$$

$$\dot{\theta}^2 = \dot{\psi}^2 \approx 0 \tag{A.9}$$

Linearization with (A.7), (A.8) and (A.9) in (A.5) and (A.6) leads to the following approximated linear equations where F_{input} has been substituted for the more general control effort u_{input}.

$$(I_{pend} + m_{pend}l^2)\ddot{\psi} - m_{pend}gl\psi = m_{pend}l\ddot{x} \tag{A.10}$$

$$u_{input} = (m_{cart} + m_{pend})\ddot{x} + f\dot{x} - m_{pend}l\ddot{\psi} \tag{A.11}$$

A.5 Laplace transform

To obtain the transfer functions, equations (A.10) and (A.11) is transformed to the Laplace domain, the transformation is here denoted by upper case letters.

$$(I_{pend} + m_{pend}l^2)\Psi(s)s^2 - m_{pend}gl\Psi(s) = m_{pend}lX(s)s^2 \tag{A.12}$$

$$U_{input}(s) = (m_{cart} + m_{pend})X(s)s^2 + fX(s)s - m_{pend}l\Psi(s)s^2 \tag{A.13}$$

A transfer function is a relationship between a single input and a single output, therefore it is needed to solve $X(s)$ from equation (A.12)

$$X(s) = \left[\frac{I_{pend} + m_{pend}l^2}{m_{pend}l} - \frac{g}{s^2}\right]\Psi(s) \tag{A.14}$$

Substitute (A.14) into (A.12) gives

$$U_{input}(s) = (m_{cart}+m_{pend})\left[\frac{I_{pend} + m_{pend}l^2}{m_{pend}l} - \frac{g}{s^2}\right]\Psi(s)s^2 + f\left[\frac{I_{pend} + m_{pend}l^2}{m_{pend}l} - \frac{g}{s^2}\right]\Psi(s)s - m_{pend}l\Psi(s)s \tag{A.15}$$

If equation (A.15) is rearranged we get the transfer function $G_\psi(s)$ as the relation between $\Psi(s)$ and $U_{input}(s)$ as seen in (A.16).

$$\Psi(s) = \underbrace{\frac{\frac{m_{pend}l}{q}s}{s^3 + \frac{f(I_{pend}+m_{pend}l^2)}{q}s^2 - \frac{(m_{cart}+m_{pend})m_{pend}gl}{q}s - \frac{fm_{pend}gl}{q}}}_{G_\Psi(s)} U_{input}(s) \quad \text{(A.16)}$$

where

$$q = \left[(m_{cart} + m_{pend})(I_{pend} + m_{pend}l^2) - (m_{pend}l)^2\right] \quad \text{(A.17)}$$

The transfer function $G_x(s)$ that describes the cart position $X(s)$ looks as

$$X(s) = \underbrace{\frac{\frac{(I_{pend}+m_{pend}l^2)s^2 - gm_{pend}l}{q}}{s^4 + \frac{f(I_{pend}+m_{pend}l^2)}{q}s^3 - \frac{(m_{cart}+m_{pend})m_{pend}gl}{q}s^2 - \frac{fm_{pend}gl}{q}s}}_{G_x(s)} U_{input}(s) \quad \text{(A.18)}$$

A.6 State Space Modelling

It is possible to present the system in state space form. The matrix form is

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{\psi} \\ \ddot{\psi} \end{bmatrix} = A \begin{bmatrix} x \\ \dot{x} \\ \psi \\ \dot{\psi} \end{bmatrix} + Bu_{input} \quad \text{(A.19)}$$

where

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & \frac{-(I_{pend}+m_{pend}l^2)f}{I_{pend}(m_{cart}+m_{pend})+m_{cart}m_{pend}l^2} & \frac{{m_{pend}}^2gl^2}{I_{pend}(m_{cart}+m_{pend})+m_{cart}m_{pend}l^2} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \frac{-m_{pend}lf}{I_{pend}(m_{cart}+m_{pend})+m_{cart}m_{pend}l^2} & \frac{m_{pend}gl(m_{cart}+m_{pend})}{I_{pend}(m_{cart}+m_{pend})+m_{cart}m_{pend}l^2} & 0 \end{bmatrix} \quad \text{(A.20)}$$

$$B = \begin{bmatrix} 0 \\ \frac{I_{pend}+m_{pend}l^2}{I_{pend}(m_{cart}+m_{pend})+m_{cart}m_{pend}l^2} \\ 0 \\ \frac{m_{pend}l}{I_{pend}(m_{cart}+m_{pend})+m_{cart}m_{pend}l^2} \end{bmatrix} \quad \text{(A.21)}$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad \text{(A.22)}$$

Appendix B

Parameters from CAD-model

The simplified model mention in Appendix A - *calculation of the system's transfer function* is dependant of the parameters m_{cart}, m_{pend} and I_{pend} and l. These parameters are derived from a 3D-model assembled in the CAD program *Solid Edge* [14]. See Figure B.1 for the complete construction. All the parameters are determined from *Solid Edge*'s *Inspect* function.

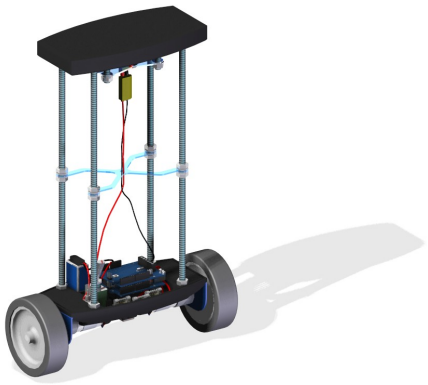

Figure B.1. Complete CAD-model

The total weight of the construction is: 1.83 kg. The mathematical model requires the weight of the cart and pendulum respectively, therefore these were estimated from the CAD-model.

B.1 Mass of cart, m_{cart}

The cart is approximated as the two motors together with their wheels. See Figure B.2. All components not built have weight data from their respective datasheet.

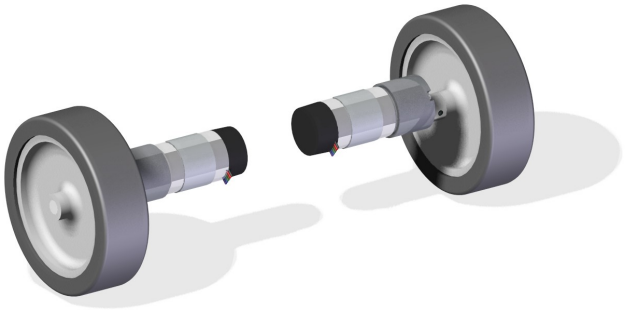

Figure B.2. CAD-model of the components considered to be the *cart*

The total weight of the cart is: 0.558 kg.

B.2 Mass of pendulum m_{pend} and the moment of inertia, I_{pend}

The pendulum is approximated as the base plate and motor brackets, the threaded rods, the two platforms, the battery, the Arduino with driver shield and the MPU-9150. See Figure B.3.

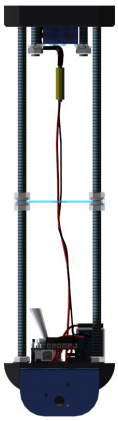

Figure B.3. CAD-model of the components considered to be the *pendulum*

The weight of the pendulum is 1.267 kg. The moment of inertia around the x

axis is 0.055 kg/m_2.

B.3 Distance to centre of mass, l

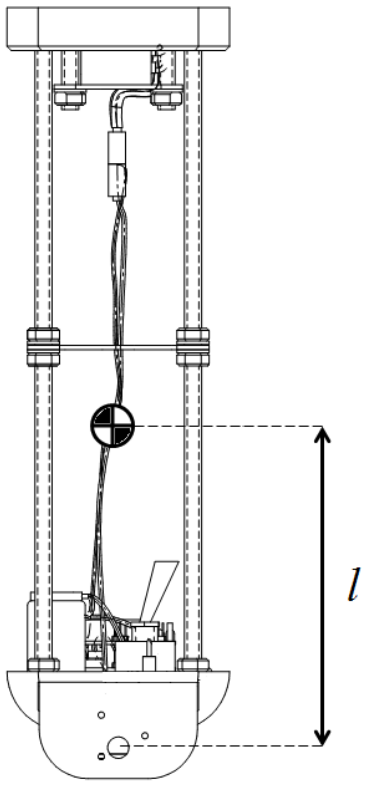

Figure B.4. Distance between the pendulum's pivot point and centre of mass

The distance, l is 0.159 m.

B.4 CAD compared to physical demonstrator

It is required that the parameters mentioned are exact. See the Figures B.5, B.6 and B.7 for comparison between CAD and demonstrator.

The physical demonstrator has been weighed and compared to the values in the CAD-model. There was no noticeable difference.

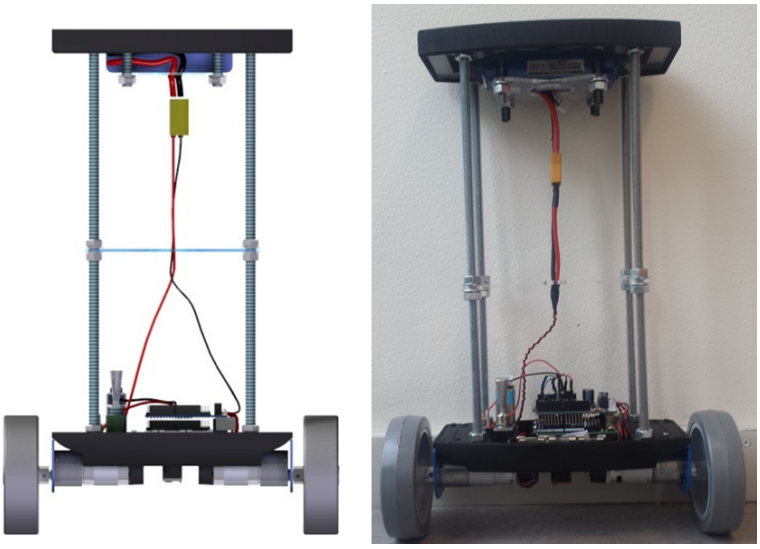

Figure B.5. The CAD and physical demonstrator scene from underneath, the MPU-9150 is visible in the middle

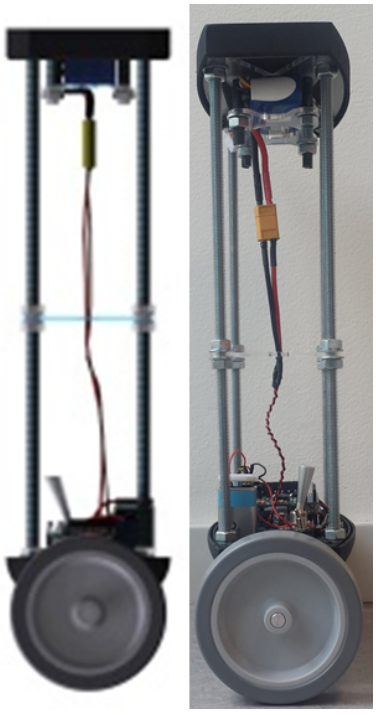

Figure B.6. The CAD and physical demonstrator seen from side

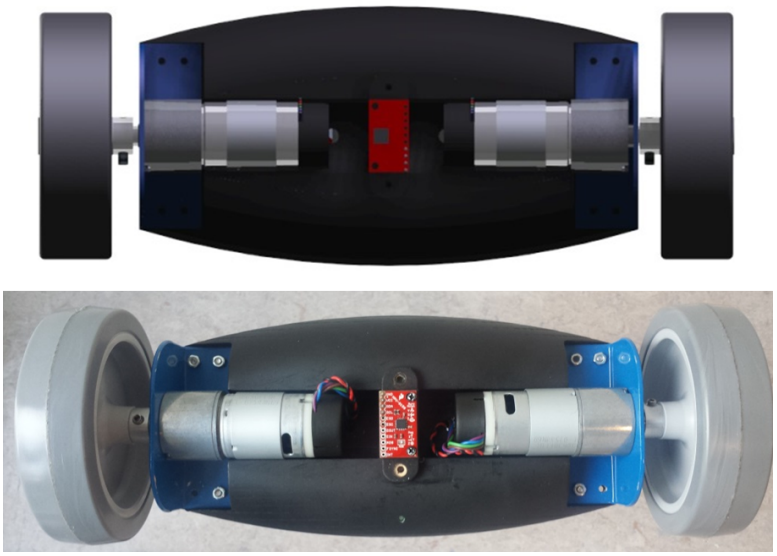

Figure B.7. The CAD and physical demonstrator scene from underneath, the MPU-9150 is visible in the middle

B.5 Results of parameters

Table B.1 shows the results of the parameters taken from the CAD-model.

Table B.1. Parameters from CAD-model

Parameters	Values
m_{cart}	0.558 [kg]
m_{pend}	1.267 [kg]
I_{pend}	0.055 [kgm^2]
l	0.159 [m]

Appendix C

Test procedures with MPU-9150 and state estimator

Tests to approximate the noise and delay of the MPU-9150 were conducted. The setup looked as can be seen in Figure C.1 below. The MPU-9150 sensor is securely mounted via a tight-fitting bracket to a servo, moving in predetermined patterns. Serial data was sent to and analysed in MATLAB.

The conducted tests included:

- Stationary test with MPU-9150 lying flat at 0° tilt.
- Servo moving periodically between 5° and 10° respectively
- Servo moving abruptly to create a step of 10°, 30° and 90°.

A stationary test was made where the sensor lays flat on a surface that has approximately 0° inclination. Many stationary tests were made, see Figure C.2 for representative results from two of the stationary readings.

Figure C.2 shows that the noise is approximately 0.03° in magnitude.

Using the Servo-library of Arduino the servo was then programmed to move in a periodical movement between +5° and -5° during 9 seconds. In Figure C.4 it can be seen how accurately the sensor follows the servo's motion.

In Figure C.4 the same test was performed but now moving periodically between +10° and -10°.

To test the speed of the sensor the servo was made to produce steps of different magnitudes produced the results presented in Figure C.5, Figure **??** and Figure C.7 presented below.

C.1 Discussion of results

As seen in the figures of the step responses, the angular data instantly reacts to the change in reference value and settles fast at a new reference level. Although the

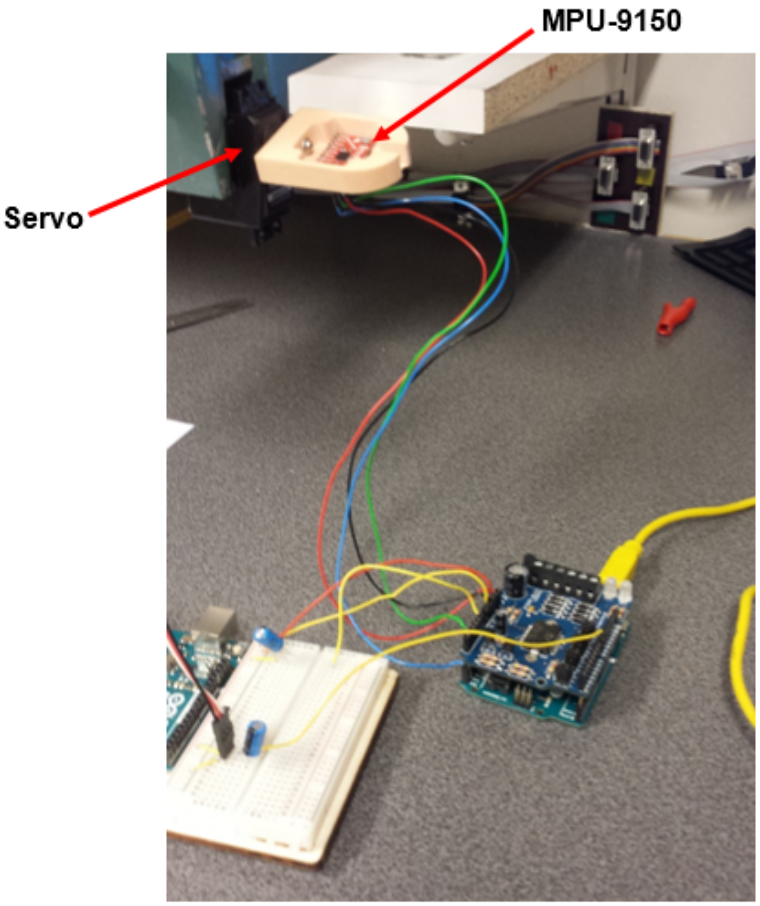

Figure C.1. Setup of experiments with MPU-9150 using servo, Arduino Uno and a driver shield

readings consistently settle at 20 percent above the angle reference. Since the error equal in terms of percentage between the tests, it is likely that the discrepancy is caused by the servo not being precise in its movements.

Servo motors often require an encoder to keep track of it's position, and its fewer poles compared to a stepper motor makes it less accurate. [1]

Since the servo is deemed not exact, conclusions can't be drawn from the graphs showing periodical movements. However from the step plots where it can be seen that the MPU followes the sudden change well, it is concluded that the delay is in the range of 5 ms.

The stationary test showed that the noise is approximately 0.03° in magnitude. The inaccurate servo was not a factor here, so this result is more valid. Apart from eventual white noise in the sensor, it can simply be a result of vibrations from the table.

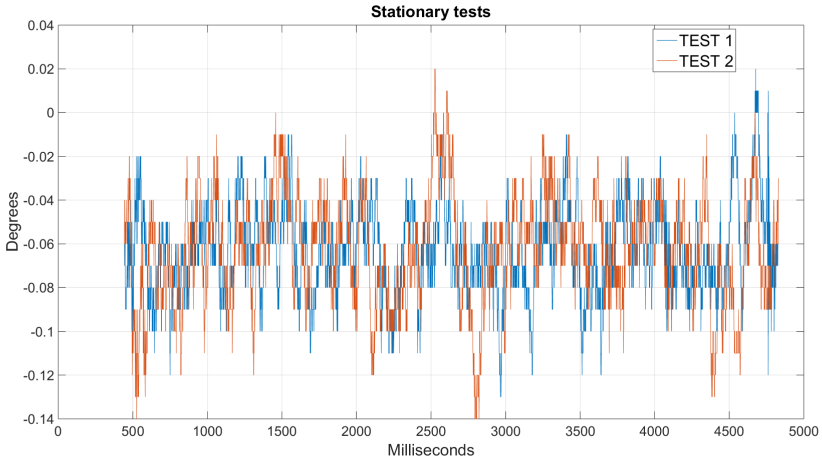

Figure C.2. Stationary tests of MPU-9150 lying flat on a table

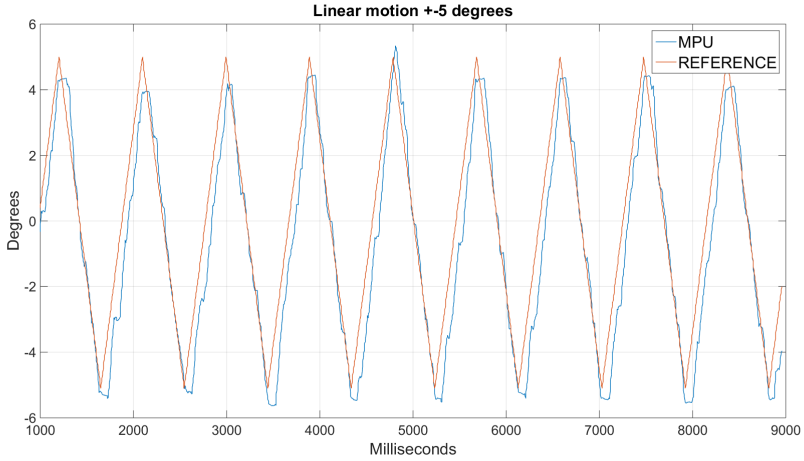

Figure C.3. MPU-9150 following the periodical servo movement between +5° and -5°

For future tests it would be beneficial to use either an encoder to track the servo's angular position, or possibly use a stepper motor. Although the sensor IMU-9150 was after the conducted tests deemed to be sufficiently precise in reading angle data.

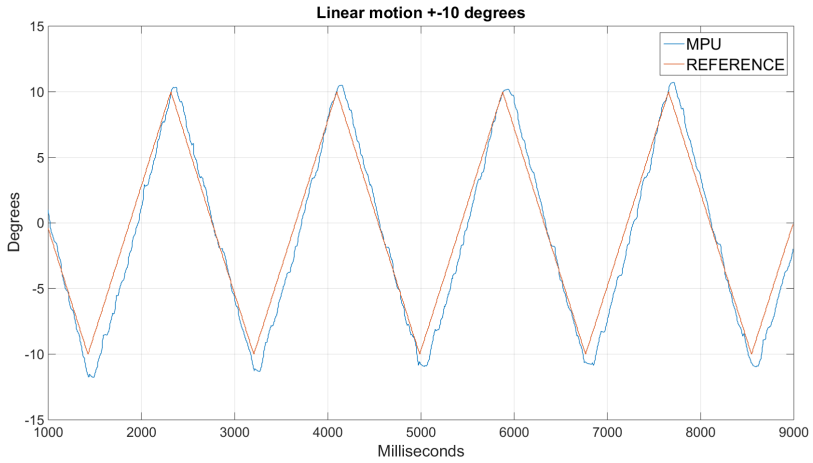

Figure C.4. MPU-9150 following the periodical servo movement between +10 ° and -10 °

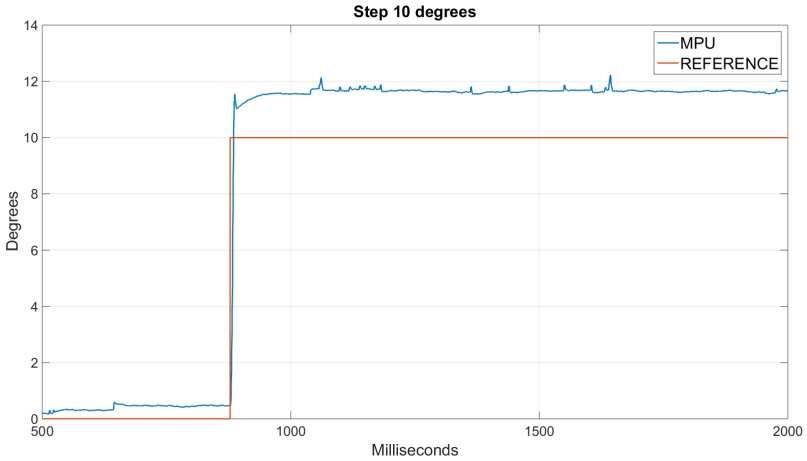

Figure C.5. MPU-9150 following a step of 10 °

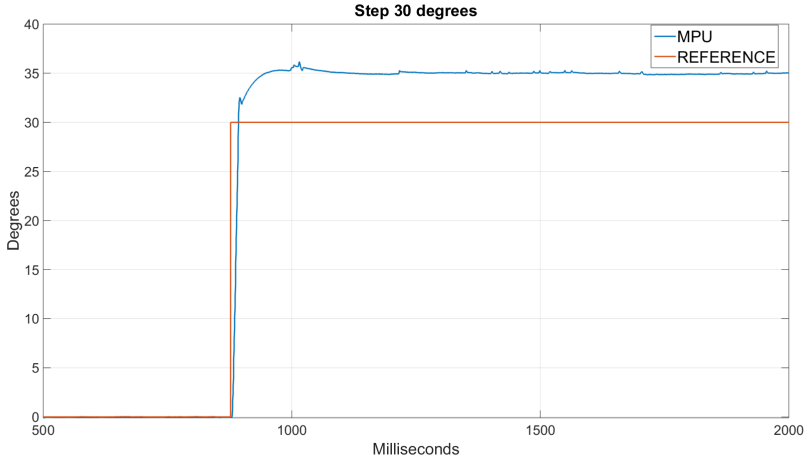

Figure C.6. MPU-9150 following a step of 30 °

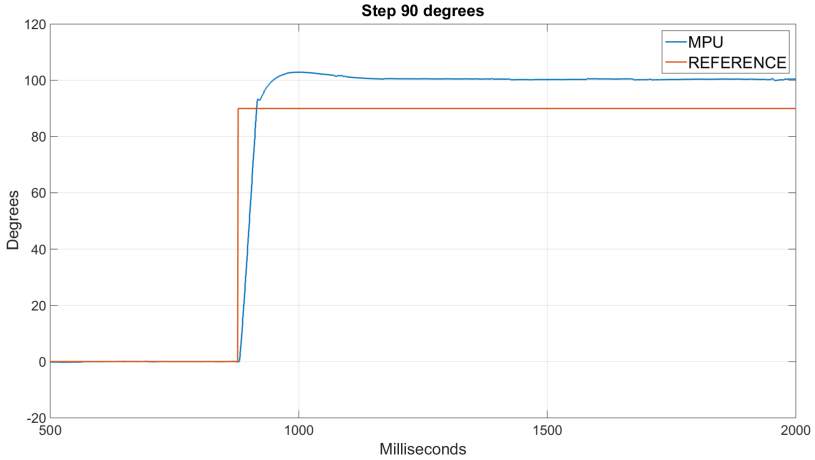

Figure C.7. MPU-9150 following a step of 90 °

TRITA xxx yyyy-nn
ISSN 1 23456789
ISRN 1 23456789