DEGREE PROJECT IN TECHNOLOGY,
FIRST CYCLE, 15 CREDITS
STOCKHOLM, *SWEDEN 2019*

Construction and theoretical study of a ball balancing platform

Limitations when stabilizing dynamic systems through implementation of automatic control theory

ALEXANDER HASP FRANK

MORGAN TJERNSTRÖM

KTH ROYAL INSTITUTE OF TECHNOLOGY
SCHOOL OF INDUSTRIAL ENGINEERING AND MANAGEMENT

Construction and theoretical study of a ball balancing platform

Limitations when stabilizing dynamic systems through implementation of automatic control theory

ALEXANDER HASP FRANK, MORGAN TJERNSTRÖM

Bachelor's Thesis at ITM
Supervisor: Nihad Subasic
Examiner: Nihad Subasic

TRITA ITM-EX 2019:35

Abstract

Control theory and its applications are crucial when operating within the area of dynamic systems. Compensating for disturbances and external actions imposed on a given system being inherently unstable or semi-stable.

Through the physical construction of a apparatus as a demonstrator of the theory further comparing the factual physical and computer simulated results derived from Newtonian mechanics. To enable comparison, designing a satisfactory controller capable of fulfilling the requirements set for the system is necessary. With regards to apparatus and control, the introduction of a proportional-integral-derivative controller for a system balancing a ball on a platform. Further allowing for analysis to determine the limitations when stabilizing a naturally unstable or semi-stable system. Also, examine how these differ from the theoretical expectations.

The control applied throughout the thesis is of the type linear, exclusively being able to operate properly within the linear spectrum of control. Using standard components and a microcontroller, a apparatus is constructed to maintain a ball on a platform. This is executed through programming with Arduino libraries and open source code. Hence, for research purposes, to see if the apparatus can operate satisfactory within the linear domain of control.

With the aforementioned stated, this thesis will first cover the theoretical model of the ball on platform scenario through computer aided programs. Then compare the theoretical results with the results acquired from a physical construction. Further examine why differences occur considering control theory and system implementation.

Keywords— mechatronics, control theory, balancing, PID, ball, platform, Arduino, servo

Referat

Konstruktion och teoretisk studie av bollbalanserande plattform

Reglertekniken och dess applikationer är centrala för att kontrollera dynamiska system och möjliggör för kompensering av störningar i system som är naturligt instabila eller semistabila.

Genom konstruktion av en apparat som demonstrerar reglerteknisk teori kan vidare jämförelser mellan resultat från apparaten och datorsimuleringar, erhållna från Newtonsk mekanik, tillhandahållas. Syftet är vidare att utveckla en regulator som uppfyller de krav som sätts upp för systemet. Med hänsyn till apparaten och regulatorn, introduceras en proportionell-integrerande-deriverande regulator för en bollbalanserande plattform. På så sätt kan begränsningarna vid stabilisering av ett naturligt instabilt eller semistabilt system bestämmas. Vidare studeras hur dessa skiljer sig från de teoretiska förväntningarna.

Endast linjär kontroll kommer att användas i detta projekt, därav en apparat som enbart är välfungerande inom ett linjärt domän. Genom användning av standardkomponenter och en mikrokontroll konstrueras en apparat för att bibehålla en boll på en plattform. Detta möjliggörs genom programmering med Arduinos bibliotek och öppen källkod. Således är, ur forskningssynpunkt, anordningens förmåga att fungera väl inom den linjära domänen av intresse.

Utifrån detta kommer examensarbetet först att redogöra för den teoretiska modellen av en boll balanserande på en plattform genom användandet av datorprogram. För att sedan jämföra de teoretiska resultaten med de resultat som erhålls från den fysiska bollbalanserande konstruktionen. Vidare undersöks varför skillnader uppstår med hänsyn till reglerteknik och systemimplementering.

Nyckelord— mekatronik, reglerteknik, balancerande, PID, boll, plattform, Arduino, servo

Acknowledgements

We would first like to thank Nihad Subasic, supervisor and course examiner, for his tireless efforts to help us through the project and ever so quick responses to all of our questions, regardless of time or day. We would further like to thank Mladen Cicic for his help with the control theory aspect of the project, providing a solid foundation to start working from. Moreover, a thank you is directed to Staffan Qvarnström for his consultation and supplying of the electrical equipment. A special thank you to Seshagopalan Thorapalli Muralidharan for his unfailing presence in the laboratory and ever helpful knowledge sharing and discussions.

Lastly we would like to thank the mechatronics department at KTH and all of our fellow class peers for the many interesting discussions and the feedback received throughout the project.

Alexander Hasp Frank & Morgan Tjernström
Stockholm, May 13th 2019

Contents

List of Figures

List of Tables

Nomenclature

Abbreviations

AC	Alternating current
DC	Direct current
DOF	Degrees of freedom
IDE	Integrated development environment
PID	Proportional-Integral-Derivative
PWM	Pulse width modulation

Miscellaneous

e_0 $[m]$	Static error for step input
F_{Plant} [-]	Transfer function of controller
F_{servo} [-]	Transfer function of built in servo controller
g $[\frac{m}{s^s}]$	Gravitational constant
G_i [-]	Transfer function of dynamic system
G_{servo} [-]	Transfer function of direct current servo motor
I_b $[kgm^2]$	Momentum of inertia of ball
K_D [-]	Derivative gain
K_I [-]	Integral gain
K_P [-]	Proportional gain
M [%]	Overshoot
m_b $[kg]$	Mass of ball
r $[m]$	Radius of ball

Variables

α_i $[rad]$	Inclination of platform
β_i $[rad]$	Angular displacement of ball relative platform
$\mathbf{a}_i$ $[\frac{m}{s^2}]$	Absolute acceleration of ball over x- and y-axis
ϕ $[rad]$	Angular displacement of link
θ $[rad]$	Angular displacement of servo arm
$\theta_{i,d}$ $[rad]$	Desired angle of servo motor
$\theta_{i,e}$ $[rad]$	Angular error of servo motor
$e(t)$ $[m]$	Error value
F_{ri} $[N]$	Friction on ball exerted by platform
N_i $[N]$	Normal force exerted on ball by platform
$u(t)$ $[V]$	Output signal
$u_D(t)$ $[V]$	Derivative output signal
U_i $[V]$	Voltage in control loop fed to servo motors
$u_I(t)$ $[V]$	Integral output signal
$u_P(t)$ $[V]$	Proportional output signal
x_d $[m]$	Desired setpoint on x-axis
x_e $[m]$	Distance error from setpoint on x-axis
x_p $[m]$	Position of ball relative origin on x-axis
y_d $[m]$	Desired setpoint on y-axis
y_e $[m]$	Distance error from setpoint on y-axis
y_p $[m]$	Position of ball relative origin on y-axis

Chapter 1

Introduction

Control theory is a sub-field of mathematics operating within the area of dynamic systems. The main purpose of developing a control model is to either stabilize an unstable system or modulating the desired response and outcome of a given system. Furthermore, in this project, control theory will be applied within the area of engineering, more specifically mechatronics.

Automatic control use dates back to the industrial revolution when the Centrifugal Governor was developed to control the speed of a steam engine by regulating fuel injection [1]. However, the theory of automatic control was not developed until second half of 19th century by James Clerk Maxwell [2]. With technological advancements and theoretical developments control theory can now be found in multiple fields and disciplines. Thus, the control theory is of great relevance within the engineering subject. Hence, the aspiration of this thesis is to demonstrate how theory is actualized in a physical model and how alterations and defects in implementation causes differences in physical response.

1.1 Automatic control theory

Although the disciplines within the area of automatic control theory varies, the theory and methods of analysis do remain, to some extent, unaltered. However, with various tasks of application the choice of automatic control design approach may vary. The controller used in the project is a proportional-integral-derivative controller, also referred to as a PID controller. The proportional part compensate for disturbances, but does not eliminate their effect. The integral part on the other hand does reduce the effect of the disturbances, but may cause issues with regards to system stability. Moreover, the derivative part of the controller does improve stability margins but at the expense of increased measurement errors.

1.2 Application and scope

For research purpose the aforementioned method of control design will be implemented, monitored, and analyzed through the use of a physical construction, a ball balancing platform. The objective is to balance a ball on a flat surface and adjust for ball displacement caused by e.g. human interaction or environmental disturbances. To actualize, generate data and acquire desired motion output of the construction, three types of hardware will be utilized apart from construction materials. A resistive touch panel, also referred to as the platform in the thesis, to register position of the ball, transceiving input data to a microcontroller. The microcontroller converting data input into velocity and acceleration of the ball, sending control signals to a pair of servo motors. The servo motors, responding to signal from the microcontroller, adjusting in accordance with the control signal generating an angular displacement. The acquired data from the actual physical model and a system simulation will then be used for the analysis. Testing will be conducted through manual ball position disturbance and observed through steady state stability. Ultimately the thesis aims to answer the following research questions:

- Can one design and construct a ball balancing platform with satisfactory performance solely practicing linear control?
- Looking at theoretical simulations and experimental results, what are the causing factor of these differences?
- What are the performance limitations when stabilizing a naturally unstable or semi-stable system and how do these differ from the theoretical expectations?

Chapter 2

Theory

The ball balancing platform introduces two separate theoretical areas of interest. The system description and control design. The system description covers the specific equations and mathematical fundamental deductions central to the ball on platform problem. The control design theory elaborates on the relationships between the PID, proportional-integral-derivative, controller components and the relevance of the respective components and their effects.

2.1 Mathematical system modelling

The model of the dynamical system is derived from the laws of physics and expressed as one or multiple differential equations. These equations can be acquired by either exercising Newtonian mechanics or Euler-Lagrangian mechanics to the system setup. As of this thesis Newtonian mechanics will be used. However, the Euler-Lagrangian method arrives to the same conclusion [3]. A accurate model of the system allows for better system analysis and prerequisites for control design. It is therefore a critical aspect in the process of stabilizing an unstable system in the way that this thesis will demonstrate.

2.1.1 Simplifications and assumptions

In order to achieve the equations of motion for a dynamical system consisting of a ball on a platform the following simplifications has to be assumed:

- The ball is rolling and not slipping on the platform
- No friction is considered
- The geometry of the ball is perfectly spherical and homogeneous
- The ball has no translation upwards relative to the platform

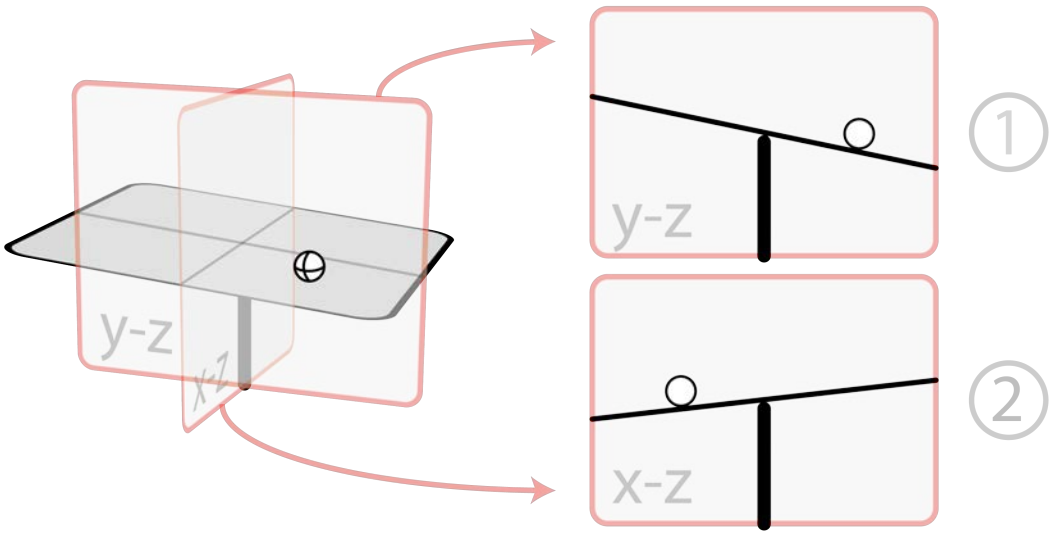

Figure 2.1. Three dimensional system represented as a pair of systems, created in Adobe Illustrator CC 2015

The three dimensional ball on platform system will from here on be separated into a pair of two dimensional ball on beam systems as illustrated in Fig. 2.1 consisting of the same equations but different variables. Index 1 for the system viewed in the x-z plane of the room and index 2 for the system viewed in the y-z plane of the room. The equations will be derived within the first system but later translated into its equivalent equation within the second system.

2.1.2 Equations of motion

The equation for absolute acceleration of the ball is given by [4].

$$\mathbf{a}_a = \dot{\omega} \times \mathbf{r} + \omega \times (\omega \times \mathbf{r}) + 2\omega \times \mathbf{v_{rel}} + \mathbf{a_{rel}} \tag{2.1}$$

The above can be rewritten as follows for one of the two dimensional ball on beam systems.

$$\mathbf{a}_1 = \ddot{\alpha_1}\mathbf{e}_{k1} \times x_p\mathbf{e}_{i1} + \dot{\alpha_1}\mathbf{e}_{k1} \times (\dot{\alpha_1}\mathbf{e}_{k1} \times x_p\mathbf{e}_{i1}) + 2\dot{\alpha_1}\mathbf{e}_{k1} \times \dot{x_p}\mathbf{e}_{i1} + \ddot{x_p}\mathbf{e}_{i1} \tag{2.2}$$

As seen in Fig.2.2, α_1 is the inclination of the platform in the first two dimensional system and x_p is the position of the ball relative to the coordinate system $\mathbf{e}_{i1}, \mathbf{e}_{j1}, \mathbf{e}_{k1}$ fixed to the platform in the current view. After simplifications and vector multiplication the following equation is derived for the x-z system as seen below.

$$\mathbf{a}_1 = (\ddot{x_p} - x_p\dot{\alpha_1}^2)\mathbf{e}_{i1} + (x_p\ddot{\alpha_1} + 2\dot{\alpha_1}\dot{x_p})\mathbf{e}_{j1} \tag{2.3}$$

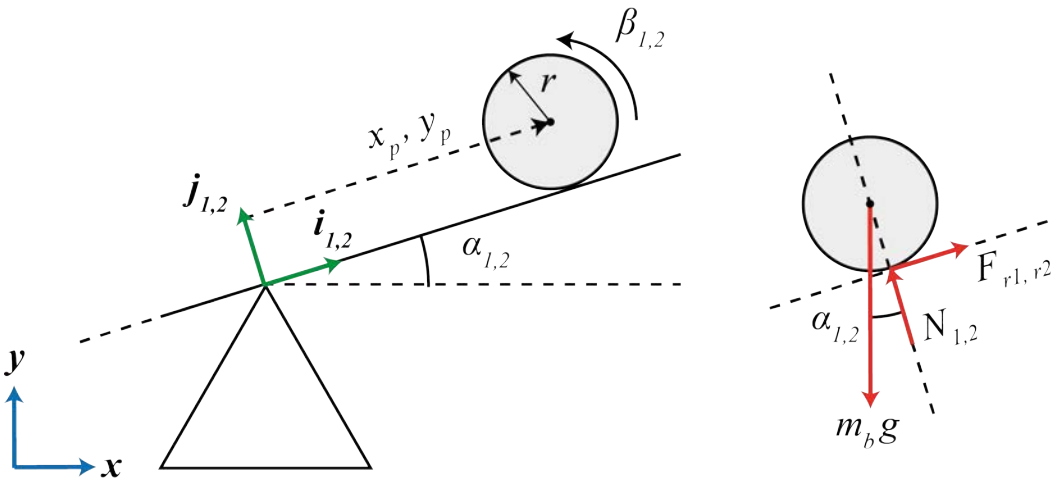

Figure 2.2. Two dimensional representation of the system and free body diagram of the ball, created in Adobe Illustrator CC 2015

From equilibrium of torques in the free body diagram seen in Fig.2.2 it is possible to derive the remaining forces on the ball.

$$I_b \ddot{\beta}_1 = F_{r1} r \tag{2.4}$$

Where I_b is the mass moment of inertia for the ball, β_1 is the angle of the ball relative to its initial position in the center of the platform and r is the radius of the ball. F_r is the force from the platform acting on the ball with fully developed friction. With regards to the assumption of no slip on the platform the relative angle β can be defined by the position as follows.

$$\beta_1 = -\frac{x_p}{r} \tag{2.5}$$

In order to solve (2.4) for F_r, the second time derivative of equation (2.5) is combined with (2.4). Resulting in equation (2.6) as stated below.

$$F_r = -\frac{I_b \ddot{x_p}}{r^2} \tag{2.6}$$

The equilibrium of forces exerted on and by the ball parallel to the platform given by acceleration in equation (2.3) and the force in (2.6) results in the two principal equations of motion of the dynamic system.

$$(\frac{I_b}{r^2} + m_b)\ddot{x_p} + m_b g sin\alpha_1 - m_b x_p \dot{\alpha_1}^2 = 0 \tag{2.7}$$

$$(\frac{I_b}{r^2} + m_b)\ddot{y_p} + m_b g sin\alpha_2 - m_b y_p \dot{\alpha_2}^2 = 0 \tag{2.8}$$

By rearranging the equations of motion the following equations suitable for Laplace transformation is achieved.

$$\ddot{x} = \frac{m_b {r_b}^2 (x_p \dot{\alpha}_1^2 - g sin\alpha_1)}{m_b {r_b}^2 + I_b} \tag{2.9}$$

$$\ddot{y} = \frac{m_b {r_b}^2 (y_p \dot{\alpha}_2^2 - g sin\alpha_2)}{m_b {r_b}^2 + I_b} \tag{2.10}$$

2.1.3 Linearization

The dynamical ball on platform system can thereafter be described by two nonlinear differential equations, one for each axis. Dealing with nonlinear systems is beyond the scope of this thesis and thus, the system needs to be linearized around a working point henceforth. The preferred and desired working point of the apparatus will be based in the center of the platform. Hence, equations (2.9) and (2.10) are linearized around $x_p = 0$, $\alpha_1 = 0$, $y_p = 0$, $\alpha_2 = 0$. The following linearized equations are valid for small deviations in α_1 and α_2.

$$\ddot{x} = \frac{m_b g \alpha_1 r^2}{m_b {r_b}^2 + I_b} \tag{2.11}$$

$$\ddot{y} = \frac{m_b g \alpha_2 r^2}{m_b {r_b}^2 + I_b} \tag{2.12}$$

Further, if the complete expression of the moment of inertia for the ball, I_b is inserted, the linearized equation can be written as follows below. Noteworthy is that the theoretical system is independent of the mass as well as of the radius of the ball.

$$\ddot{x} = \frac{3}{5} g \alpha_1 \tag{2.13}$$

$$\ddot{y} = \frac{3}{5} g \alpha_2 \tag{2.14}$$

2.1.4 Laplace Transformation

Equations (2.13) and (2.14) is defining the dynamics of the system within the time domain. To allow controller design of the system, these equations need to be translated into the frequency domain using Laplace transformation. The resulting Equations, now within the frequency domain as seen below.

$$s^2 X = \frac{3}{5} g A_1 \tag{2.15}$$

$$s^2 Y = \frac{3}{5} g A_2 \tag{2.16}$$

2.2 Control Theory

As aforementioned, a PID controller will be utilized to control the system. In this section its different parts will be explained in detail.

2.2.1 PID Controller

The PID, proportional-integral-derivative, controller is a control function which applies corrections automatically given a feedback loop mechanism. Through the PID, modulated systems requiring continuity in the control can be monitored and stabilized to desired response. The value evaluated in the controller is the error $e(t)$. Where the error value is given by the offset from a setpoint. However, any combinations of respective parts of the controller may be implemented e.g. P, PI, PD. The regulated output signal, discussed in detail below, is given by equation (2.17) [5].

$$u(t) = K_P e(t) + K_I \int_0^t e(\tau)d\tau + K_D \frac{de(t)}{dt} \tag{2.17}$$

2.2.2 Proportional control

The purpose of the proportional controller is to reduce the steady state error through increasing the proportional gain of the system. Through introduction of a constant K_P, the proportional gain, the proportional response is adjusted accordingly i.e. the steady state error behaves inversely of the proportional gain. The proportional term or output signal is further given by equation (2.18).

$$u_P(t) = K_P e(t) \tag{2.18}$$

Nevertheless, if the gain is increased too much it may alter the output causing instability in the system. Opposite, if a gain becomes too small it could possibly cause issues handling external or internal disturbances [5].

2.2.3 Integrating control

The integral controller is proportionate to the, desirably, finite duration time of the error and the magnitude. The accumulated offset caused previously in the system is corrected by the sum of errors over the finite time. Thus, the steady state error is eliminated. However, K_I, the integral control, could cause the response of the system to become worse e.g. crippling the system with regards to response causing transient or oscillatory behaviour. The output signal from the integral term is defined as equation (2.19) [5].

$$u_I(t) = K_I \int_0^t e(\tau)d\tau \tag{2.19}$$

2.2.4 Derivative control

The derivative controller is determined through the calculation of the errors response slope over time. The slope of the error is then multiplied with K_D, the derivative gain. The derivative term is defined in equation (2.20).

$$u_D(t) = K_D \frac{de(t)}{dt} \tag{2.20}$$

This term does alter the rate of change of the output, slowing it down. K_D will however further reduce overshoot, increase stability and enhance the transient system response [5].

Chapter 3

System Design

To acquire the desired function of stabilizing a ball on a flat surface the system must first be able to alter the inclination of the platform around the two orthogonal axes. This is actualized by two servo motors connected to the platform by two pairs of arms. A resistive touch panel, acting as a platform, will provide ball position as input to the microcontroller and as of the platforms inclination, servo motor will be used to calculate the necessary output. Fig.3.1 below visualizes the basic elements and flow of information through the system.

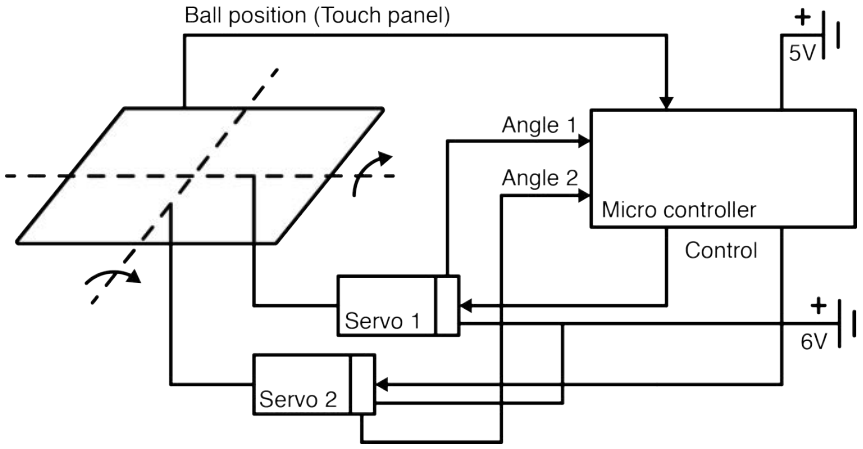

Figure 3.1. Essential system elements, created in Adobe Illustrator CC 2015

3.1 Requirements

Development of the previously illustrated system, in Fig. 3.1, is restricted by a series of requirements, both on mechanical construction and performance. These requirements are based on previous attempts to construct a similar system [6]. Following are the requirement necessary for the system design.

- Allow for the platform to incline 20° in every direction
- Minimize rotational play around the z-axis of the platform
- Settling time (5%) $<$ 3 s ($T_s < 3$)
- Overshoot $< 5\%$ ($M < 0.05$)
- Static error for a step input $<$ 5 mm ($e_0 < 0.005$)

3.2 Geometrical analysis

To allow for the platform to incline according to the requirements one servo motor for each rotational axis is connected to the platform as described in Fig.3.2 where point E is the motors center of rotation and point C is where forces are applied to the platform.

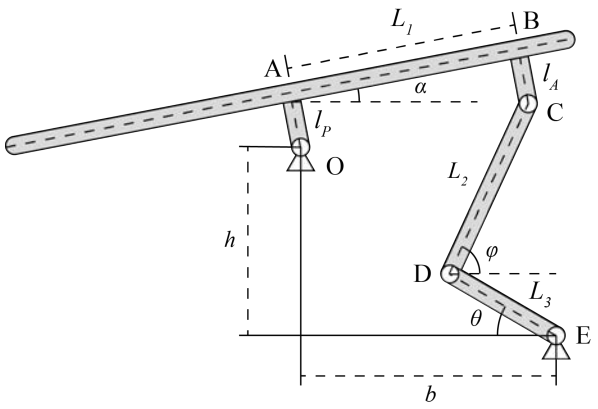

Figure 3.2. Geometry of platform actuation, created in Adobe Illustrator CC 2015

The dimensions of the servo arms, L_2 and L_3, the relative placement of the motors h and b as well as the distance to the joints on the platform L_1 are achieved using geometrical synthesis in MATLAB R2018a, henceforth referred to as MATLAB.

Except from the requirement of minimum inclination, the objective of the synthesis is also to obtain a close to linear relationship between servo angle and inclination of the platform to be used in the transfer function of the plant. Table 3.1 presents the resulting system dimensions and in Appendix B the system can be viewed in three positions corresponding to its extreme and natural equilibrium positions in one plane.

Table 3.1. System dimensions in millimeters

L_1	L_2	L_3	h	b
45	100	31	100	76

With the presented dimensions a close to linear relationship between the angles, α and θ, could be obtained. Further note that the dimensions are equal for both the x-z and y-z system. Fig.3.3 illustrates the differences between analytic angle relationship and a linear relationship based on the geometry of the system.

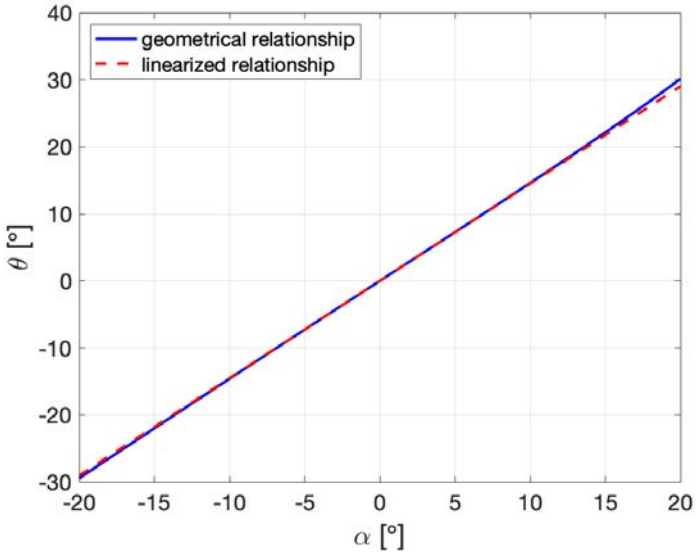

Figure 3.3. Relation between platform inclination and motor angle, graphed with MATLAB

However, the linear relationship, between α and θ, seen in Fig.3.3 is achieved though a scalar constant deducted from the geometric analysis. The relationship given by equation (3.1) below.

$$\alpha = \frac{L_1}{L_3}\theta \tag{3.1}$$

3.3 Hardware

The central working parts of the physical model, apart from construction material seen in Appendix B, are a 4-wire resistive touch panel, two servo motors and a Arduino Uno microcontroller. The panel generate input data, the microcontroller processes the input and generates an output sent to the servo motors to generate physical motion.

3.3.1 4-wire touch panel

To determine the position of the ball a 4-wire analog resistive touch panel is used as the platform. The components in the touch panels cross section as viewed below in Fig. 3.4.

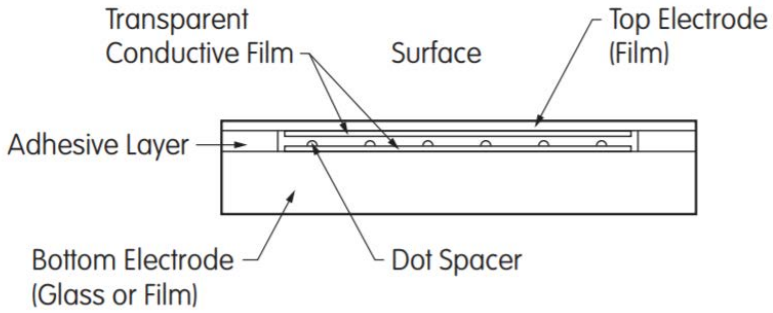

Figure 3.4. Cross section of touch screen [7]

The touch panel consists of two electrode films in which current can run. Amid the electrode top and bottom films, dot spacers are fixed with a transparent flexible conductive film and a adhesive layer to keep the electrode films separate. As the top electrode film is exposed to pressure a connection with the bottom electrode is made, responding to the position on the touch panel on which the pressure is applied. Hence, the top electrode film is utilized to examine the voltage at the pressure point, giving an analog reading of the position on the panel. Furthermore, given the two dimensional panel, one analog reading will subsequently provide one position for each axes. The electric analog signal thenceforth is converted into a digital signal in the microcontroller using a A/D-converter [7].

3.3.2 Servo motor

The basic components within the standard servo motor is a potentiometer, a DC motor and a control circuit. As the DC motor rotates the potentiometer administers a change in resistance. The control circuit can thereby regulate the rotation of the DC motor and determine direction. The standard servo motor has a maximum rotational output of 180°, or from the servo's zero position a ±90° freedom of rotation in each direction [8].

The servo motor is controlled through electrical pulses with pulse width as the variable. The PWM does furthermore regulate the orientation of the angular change, i.e. clock-wise or counter clock-wise [8].

However, the rotational velocity of the motor is proportional to the difference between he actual and desired position. Due to position differences, large displacement causes the servo to rotate faster whilst small differences cause the servo to rotate slower i.e. proportional control. If the servo is holding a position to which it was commanded to move but then exposed to external forces, the servo will attempt to resist further displacement.

3.3.3 Arduino

The core of system revolves around the microcontroller Arduino Uno based ATmega328P, programmable through Arduino's own IDE. The Arduino Uno possess fourteen digital outputs and inputs, whereof six are pulse width modulating. An additional 6 inputs are analog. Some other features are 16 MHz quartz crystal, USB connection to power jack, reset button and in circuit serial programming header [9].

3.4 Control loop and transfer functions

The dynamics and geometry of the system is considered well defined and can from Fig.3.1 be interpreted as a control system seen in Fig.3.5. Each hardware component described above serves a specific purpose and is associated with an element in the control loop according to Fig.3.5. The function of the control loop is as follows: The user firstly defines an input in the form of desired ball position or setpoint. A Difference between the setpoint and the actual ball position results in the error. The microcontroller then calculates the desired servo angle of the system in accordance with the error to reach the set position, to which the servo acts. The platforms inclination is geometrically coupled with the servo angle which serves as input to the dynamics of the system. The resulting ball position is registered by the touch panel and fed backwards resulting in a closed loop.

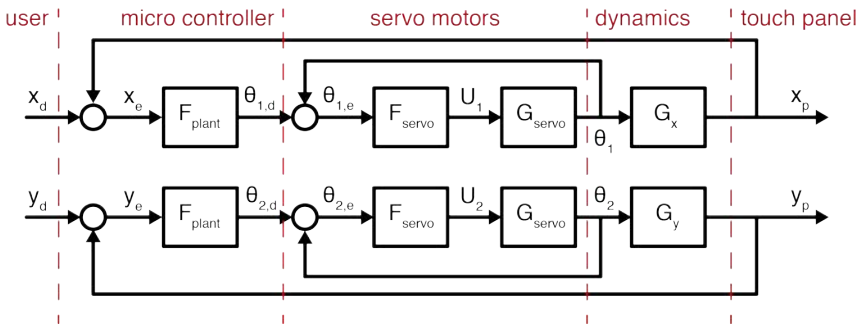

Figure 3.5. Closed loop system representation, created in Adobe Illustrator CC 2015

The control loop now consists of two controllers and two transfer functions for each axis of rotation. Since the servo motors themselves include a built in control circuit, these servo controllers does not have to be developed. F_{servo} and G_{servo} can instead be seen as an ideal unit with time delay that invariably execute the rotation to the desired angle. The transfer functions of the plant G_x and G_y can be deducted from equation (2.15) and (2.16) together with the linear relationship between θ and α as follows:

$$G_x = \frac{X_p}{\theta_1} = \frac{5g}{2s^2} \tag{3.2}$$

$$G_x = \frac{Y_p}{\theta_2} = \frac{5g}{2s^2} \tag{3.3}$$

3.5 Controller design

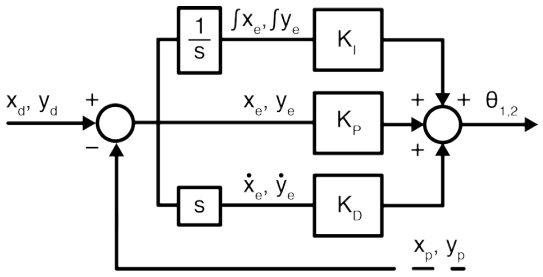

Figure 3.6. Structure of plant controller, created in Adobe Illustrator CC 2015

Looking at the denominator of the mathematical system formulated above in equations (3.2) and (3.3). The system is marginally stable since it contains a double pole in the origin. This results in the system having only non-decaying oscillatory components in its step response when introduced to a feedback loop [10]. To gain stability and enhance performance of the system a PID controller is introduced, regulating the output signal to the servo motors, seen in Fig. 3.5 represented as F_{Plant}. Values of K_P, K_I and K_D determines the control output that is required to attain a stable system response. Assuring that all system requirements are fulfilled to a satisfactory level, gaining system stability, values of K_P, K_I and K_D are established through a iterative process in MATLAB. In theory, due to the system having a embedded integrator, the static error is self-regulated by the system itself. Thus, a integrating part could be redundant in the PID controller.

Chapter 4

Implementation

Theoretical and actual physical implementation vary widely with regards to desired response and actual outcome of the systems. Disregarded parameters and simplifications are incorporated and thus compensated for to achieve desired response. As follows below is the methodology behind programming the physical system model. Moreover, note that the same process applies for each rotational axis.

4.1 Electrical circuit

As follows below in Fig.4.1 is the electrical circuit and the respective components. Besides wiring the components are the Arduino Uno microprocessor, two servo motors, a resistive touch panel, a small breadboard and a voltage step up component. Complete list of components in Table 4.1. The Arduino and touch panel is powered through USB and the servo motors from an external power source stepped up to 6V. The capacitors are used to prevent temporary power deficit and assure continuity in the power supply to the servo motors.

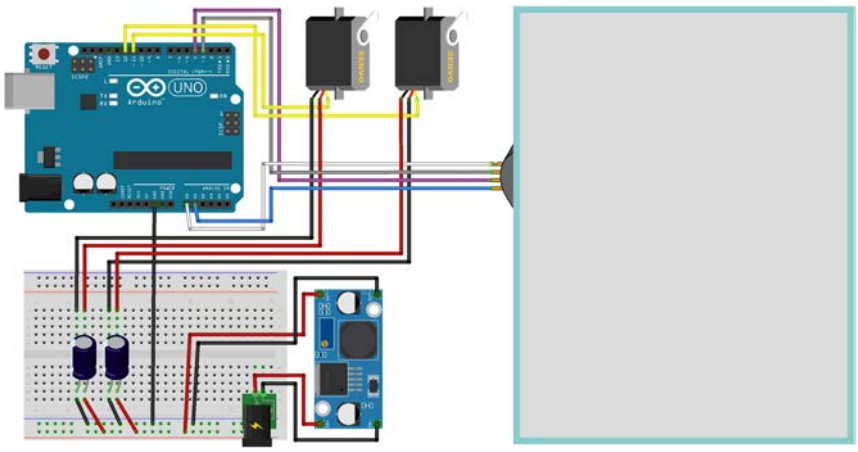

Figure 4.1. Electrical circuit to be implemented, created with Fritzing beta software

Table 4.1. List of components

#	Component
1	Arduino Uno Rev3
1	Hitec HS-303 standard analog servo
1	Futuba S3003 standard analog servo
1	NKK Switches 4-wire analog resistive touch screen, FTAS00-10.4AV-4A
1	LM2596 DC-DC Step-down voltage converter
2	470 μF capacitor

4.2 Arduino programming

The Arduino IDE allows for users to apply and implement functions and programs found in the preexisting Arduino library [9] or in open source GitHub repositories. Through these libraries, touch panel input signal processing, filters, PID controllers and servo maneuvering can be automatically performed, executed and applied. However, the functions are dependent on hard coded system parameters established in code to ascertain the necessary response. The used concepts and strings of code implemented in the Arduino are explained below. Moreover, the full code can be seen in Appendix C.

4.2.1 Touch panel input

Input data to the system is acquired through analog readings of voltage levels over the touch panel converted to digital signal in the Arduino Uno microprocessor. The analog to digital signal conversion are performed utilizing Adafruit Touchscreen library [11]. Beyond conversion, the library is also set to establish double sampling of the input signal, thereby reducing the readings of inaccurate data. The signal from the touch panel does however still contain noise which cause undesired frequencies in the control input signal. To generate accurate input, with regards to center of platform, a manual four point calibration is carried out.

4.2.2 Low pass filter

The mathematical system modeling does not account for disturbances and noise due to simplifications, assumptions and continuous time. However, data sampling is largely affected with regards to the touch panel discussed above. The panel emits control input to the Arduino Uno with added noise causing fictional reading spikes. Further causing unnecessarily large output signals to the servo motors given by the PID controller. To eliminate or regulate the input transmitted from the touch panel a low pass filter is implemented. The filter is active when frequencies exceeds a given

cut off frequency before sampling [5]. Signals with frequencies above the cut off frequency is hence attenuated. The necessary code in the Arduino microprocessor is executed through utilization of the Arduino real time digital signal processing library [12].

4.2.3 PID controller and tuning

As the input from the touch panel has been filtered a time discretization is executed by utilizing functions in the Arduino PID Library [13]. This is due to the electronic components inability to collect data continuously. As time discretization is implemented, a finite time is assigned to the program, sampling data points for each time interval. In order to form a continuous curve from the sampling data, methods e.g. Cranck-Nicolson or Euler are used to create unconditionally stable algorithms [14]. The continuous curve formed is comprehensible to the Arduino enabling for further calculations and signal processing.

The filtered signal is next sent to the servo motors. However, the signal must first be processed through the PID. This to ascertain the correctional regulatory motion of the platform balancing the ball. This entails the calibration and tuning of K_P, K_I and K_D.

4.2.4 Servo motor maneuvering

Finally the processed control signal is sent to each servo motor independently. The control signal enables angular displacement of the motors by utilization of the servo maneuvering Arduino library [15]. Different methods within the library enables e.g. attaching, detaching and writing explicit motor angles given a calculated PWM signal. Each motor is calibrated to maintain a horizontal platform initially. Output limitations is also set to reflect the mechanical constraints of the physical model. If limitations and constraints are not set, mechanical components could malfunction, damage or both.

Chapter 5

Results

As follow below are the results of the factual physical model and the theoretical modelling simulations. Note that the factual results are approximations collected over a finite time i.e. some variations are not included in the samples presented. However, the collected data is considered to be a good representation despite being an approximation. Seen below in Fig. 5.1 is the complete construction of the ball balancing platform.

Figure 5.1. Final construction of ball balancing platform

5.1 Simulation results

A simulation of the nonlinear mathematical ball on beam model together with of a servo motor model for one axis [16] is performed with the SIMULINK software included in MATLAB. The SIMULINK system model can be viewed in Appendix C. The simulation and PID tuning is performed under the following constraints: Controller output limited to 6 V and servo motor output limited to ±30°. The resulting proportional, integrating and derivative gains are presented in Table 5.1 below. Using the above, now, PD control to eliminate abrupt constant position

Table 5.1. Simulation PID tuning values

Gain	K_P	K_I	K_D
	0.2	0	1

disturbance of 60 mm produce a damped oscillating response as seen in Fig.5.2. When the system is in a settled state there is no static error present. This system behaviour has been discussed in previous chapters and was expected.

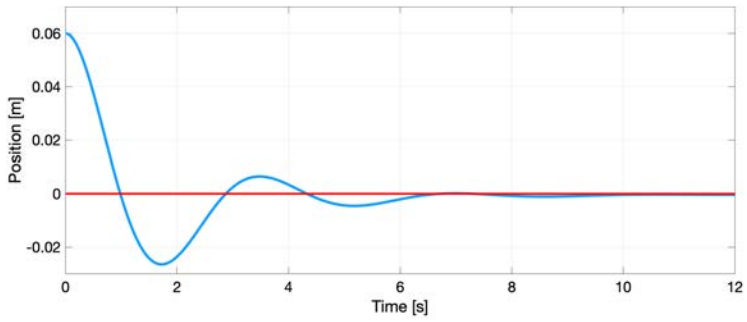

Figure 5.2. Simulation results of abrupt position disturbance, graphed with MATLAB

5.2 Experimental results

Through iteration, trial and error seen in Appendix D, a PD controller is satisfactory to balance the ball on the platform. As follow in Table 5.2 are the gain values for each respective servo motor.

Table 5.2. PID tuning values

Gain	K_P	K_I	K_D
Servo x-axis	0.09	0	0.0532
Servo y-axis	0.205	0	0.05

The sampling time, T_s, is set to 15 milliseconds, which is the fastest processing time in the Arduino void loop. The minimal sampling time gives the fastest response to ball displacement and hence reducing potential delay in reaction and angular displacement of the servo arms. Due to shattering and erratic behaviour in the servo motors the low pass filter attenuates high frequencies in the signal. As follows below in Table 5.3 are the cut off frequency values for each respective servo motor.

Table 5.3. Low pass filters

Axis	Attenuating frequency
Servo x-axis	4
Servo y-axis	2.3

The introduction of a low pass filter gives a smoother reaction response of the servo motors and reduced the shattering behaviour in the construction. Shattering caused by the servo motors inability to move to a set position but rather quickly oscillating around the set position with small deviations.

With regards to stability the construction is unable to position the ball exactly in the setpoint of origin but keeping in within the center square area of the platform seen in Fig. 5.2. Furthermore, the approximated maximal deviation from origin in each direction was 30 mm along the x-axis and 15 mm along the y-axis.

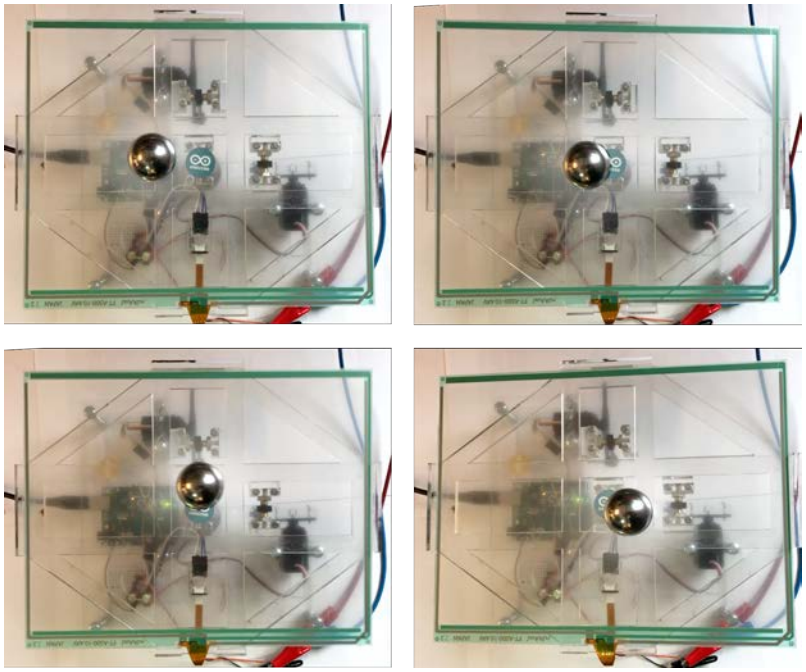

Figure 5.3. Ball stability around center setpoint

To analyze the response time of the system when exposed to disturbance, the ball was placed 60 mm from origin i.e. adding abrupt disturbance through a dislocated starting point.The placements of the ball on the respective axes is further seen in Fig. 5.3. A approximate response time in seconds to dislocation disturbance was determined through multiple samples seen Table 5.4.

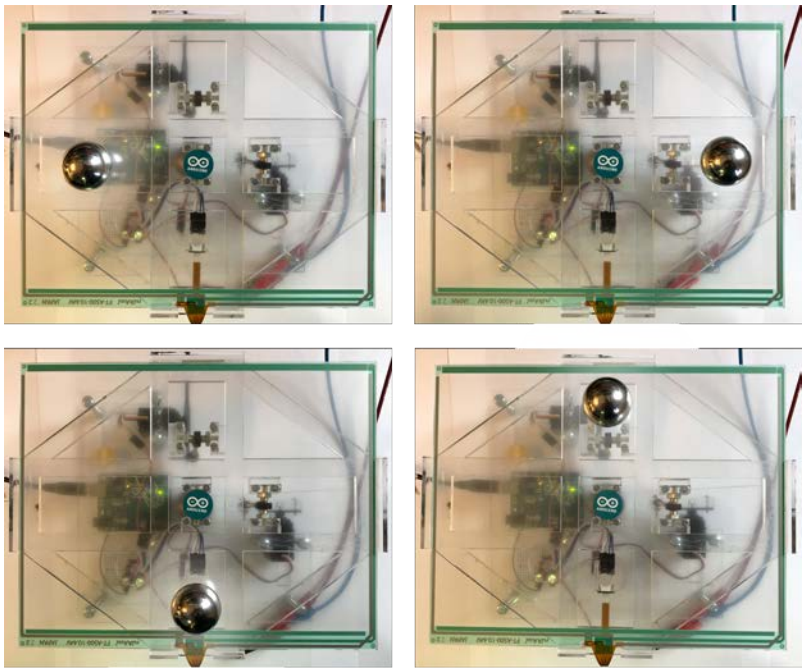

Figure 5.4. Initial abrupt position disturbance for both axes

Table 5.4. Response time to ball dislocation

Axis	Average response time
x_{max}	2.68
x_{min}	2.58
y_{max}	1.63
y_{min}	1.81

When changing the desired set position or setpoint, i.e. not being in origin, the construction was crippled and unable to perform the task of balancing the ball on the platform. However, it was able to fulfill the action when the coordinates of the setpoint were relatively small and close to origin.

Chapter 6

Discussion

Analyzing the simulation and experimental results, several factors have apparent difference with regards to control, system modulation and physical properties being neglected. The following discussion will cover the differences and causing factors between actual and simulated control, physical modelling and construction and their difficulties.

Mention worthy is the way results where extracted from the apparatus. All results are approximated by sampling video observations of different scenarios. This is due to limited capacity of Arduino serial communications being unable to sample data points. In other words, when trying to extract data points while running the apparatus, performance was significantly worsened.

6.1 Physical discussion

As follows below is a list of the physical aspects and differences that are relevant to the simulation and experimental result incoherence, i.e. not having these accounted for in the simulations.

- Play in the servo motors and arm joints and links
- Rotational allowance of the platform around the z-axis
- Vibrations in the whole structure due to shattering in the servo motors
- Geometrical construction errors and tolerance inaccuracies
- Inability to maintain the platform in a fixed horizontal position of exactly 0°
- Mounting errors of the platform allowing play
- Deformation and give in solid parts

The accumulated physical aspects, unaccounted for in the simulation results which is consider ideal, does hence cause a large impact with regards to tuning and balancing the ball in origin. Likewise, ability to control disturbances of ball displacement. Though the constructional features impacts the outcome of the model to a great extent, the electrical components, i.e. servo motors and touch panel, also induce undesired properties to the system. The touch panel is unable to read the position of the ball accurately and subsequently causing input data to contain noise. Processing inaccurate input information fed to the micro processor, Ardunio Uno, will hence give imprecise and erroneous responses in the servo motors. In conclusion, considering physical properties, the physical differences are both caused by the constructional errors and electrical components inability to function close to ideally.

6.2 Control discussion

As stated above, there are several factors embedded within the physical model but not in the mathematical model which engendered discrepancy between results. These discrepancies can be observed in multiple aspects and this part of the chapter will first discuss the discrepancy in results between the orthogonal axes in the physical model and then aspects of control theory. Discussion is based on results provided in Chapter 5.

Hardware wise, both the x- and y-axis are dependent on the same input, the touch panel and the same microcontroller, the Arduino Uno. What gives the axes their individual characteristics is therefore essentially the servo motors. The project has been on a strict budget, hence, a compromise to use second hand servo motors provided by the Royal Institute of Technology, Stockholm over buying new motors was made. Subsequently, two servo motors from different brands are used to actuate the platform inclination. The fact that the x-axis oscillations are larger in amplitude at its steady state, in contrast to the y-axis, is mainly due to the difference in motor specifications. The motor coupled with the x-axis is a Futuba S3003 with a maximum angular velocity of 0.19 sec/60° while the motor coupled with the y-axis is a Hitec HS-303 with a maximum angular velocity of 0.15 sec/60°. This effect is also enhanced by the fact that the platform by its aspect ratio has a larger moment of inertia around the vertical shorter axis than it has around its horizontal longer axis.

Earlier chapter implied that no integrating control would be needed to gain satisfactory performance in the physical model. This is confirmed by the results since the observed oscillations are symmetrical around the center for both axes. Experiments showed that introduction of integrating control only increased amplitude of oscillation, as expected and in accordance with theory. Further note that the controller applied is a proportional-derivative, PD, with the K_I gain set to zero in accordance with Chapter 5.

With regards to control, the system was exposed to setpoint offsets i.e. not being in origin. The inability to attain desired responses is likely due to the static

and linear tuning of the control. However, working fairly well with a setpoint close to origin.

Another factor is the use of low pass filter on the input signal. The filter was applied in order to reduce shattering in the apparatus generated by signal noise fed to the controller. However, it also introduced a delay in the system. Large dislocations from origin in ball position over short periods of time where attenuated and thereby caused the the motors not to react to the change.

Chapter 7

Conclusion

To conclude the thesis the research questions stated in the introduction are to be answered with provided motivation.

- Can one design and construct a ball balancing platform with satisfactory performance solely practicing linear control?

This mainly depends on how satisfactory performance is defined. To be able to answer the question, satisfactory performance was defined by the system requirements set in Chapter 3. Comparison between results and the requirements confirms that the physical model operate satisfactory with regards to settling time but not static error. Regarding overshoot no observations with adequate accuracy has been made and therefore cannot be discussed further.

Considering the static error, it is possible to argue that the main reason for not attaining a small enough error may not be the solely linear control but rather the motor specifications as aforementioned. Optionally, nonlinear or model based controller design would offer flexibility in the construction by allowing the system to function properly even for setpoints that are not close to the center. But since this is not stated as satisfactory performance it is not within the scope of this thesis.

- Looking at theoretical simulations and experimental results, what are the causing factor of these differences?

The factors causing discrepancy between simulation and reality are naturally, as discussed in previous Chapter 6, pure construction errors. The inability to foresee and mimic these in simulations are further not to be ignored. Hence, necessary measures should be taken and encouraged if to build a construction of this nature to minimize errors significantly.

- What are the performance limitations when stabilizing a naturally unstable or semi-stable system and how do these differ from the theoretical expectations?

After constructing a fully functional ball balancing platform, some insights during the project may help in answering the above stated question. A naturally unstable or semi-stable system is always in need of state adjustment to some extent. In the case of a ball balancing platform, the state to be adjusted is the platform inclination and it is actualized through servo motors. Thus, the adjustment has to be controlled. This can often be executed manually but within the field of control theory the actions are executed by the controller e.g. PD.
A controller is however solely dependent on an input. In the case of the ball balancing platform the input is given by the position of the ball. On a more general note, with regards to other systems, the input could be temperature, pressure, velocity and so forth.

The ability to adjust states, the controller design and the input constitutes the limitations of stabilizing systems and are the key elements to consider. Hence, the system performance is reliant, yet limited, on state adjustment agility, specified controller limitations and accuracy and speed of input data gathering. In the case of a ball balancing platform these limitations are realized by the angular velocity through which the servo motors can operate. Further the PD controller design, the Arduino capabilities and the resolution of which the analog touch panel can register position accurately.

Gain values, K_P and K_D, in the PD controller affects performance to a great extent. Covered in Chapter 2, the K_D gain term does alter the rate of change of the output, slowing it down and further reducing overshoot, increasing stability and enhancing the transient system responses. Whereas K_P increases the proportional response of the system, making it faster the higher the value of K_P.

From the results in Table 5.2, despite being small, incremental increases or decreases caused instability of the system or severe difference in performance for the worse. Hence, being a significant factor with regards to performance limitations. Further notice the great difference in values of K_D between the experimental model and theoretical simulation. The derivative gain in the theoretical simulation was not applicable in the experimental model. Hence, strengthening the conclusion with regards to controller design.

All of these limitation exist in theory as well as in practice. However, experimental limitations are inherently different than theoretical limitations due to time discretization, possibly limited computing power and non-ideal physical construction parameters.

Chapter 8

Future Work

The future work and necessary improvements that would be relevant are several, with regards to both controller design and construction. Foremost the implementation of powerful and accurately reading and responding servo motors of the same brand and model to avoid different calibrations of the PD controller gains.

Secondly, using motors that are analog does impose difficulties with regards to reading PWM. However, this issue could be resolved by using digital servo motors having better PWM reading properties.

Another improvement is a improved touch panel, or other ball tracing equipment, with higher reading accuracy considering data collection to achieve better input. These components could likely improve the construction significantly. However, with increased strength in the motors and new reading equipment the construction model would require modifications.

From a construction stand point, the play in the joints and links and the tolerances should be improved since this is a large causing factor to shattering of the touch panel or platform. The play is as aforementioned, caused by the servo motors but further enhanced by the play and insufficient tolerances in the joints.

Completely accurate results and data readings could not be attained due to slow serial communication with the Arduino through USB connection. When attempting to extract data, the consequence was a slower response in the servo motors and thus, inability to maintain the ball on the platform. To avoid this, another system to extrapolate data should be further investigated. Subsequently allowing for a better analysis and discussion. Due to time constraints this was not possible in the project.

Another additional improvement could be to introduce a more advanced filter, rather than the low pass filter e.g. Kalman filter. This would give better readings from the touch panel or the altered reading equipment.

Lastly, increasing the degrees of freedom of the platform would, presumably, improve the handling of the ball by a smoother reaction to ball displacement. The introduction of additional degrees of freedom would however require a more advanced controller and system due to additional motors and axes involved in the

process. The principles of the ball on platform does however remain the same regardless of degrees of freedom.

Bibliography

[1] M. Sakharov and V. Tarabarin, *Explorations in the History of Machines and Mechanisms*, vol. 15. Springer, Dordrecht, 2012.

[2] S. Bittanti, "James clerk maxwell, a precursor of system identification and control science," *International Journal of Control*, vol. 88, no. 12, pp. 2427–2432, 2015.

[3] C. G. Bolivar-Vincenty and G. Beauchamp-Baez, "Modelling the ball-and-beam system from newtonian mechanics and from lagrange methods," *12th Latin American and Caribbean Conference for Engineering and Technology*, 2014.

[4] N. Apazidis, *Mekanik II, Partikelsystem, stel kropp och analytisk mekanik*, vol. 1:3. Studentlitteratur AB, 2017.

[5] T. Glad and L. Ljung, *Reglerteknik, Grundläggande teori*, vol. 4:15. Studentlitteratur AB, 2016.

[6] J. Blomqvist and N. Osterman, "Automatic control of a dynamic system. positioning of a spherical object on a flat surface.," bachelor's thesis, Royal Institute of Technology, Stockholm, Sweden, 2016.

[7] "Ft series 4-wire analog resistive touch screens with fpc tails." https://eu.mouser.com/pdfdocs/CN-0320FTSerieswithFPCTail.pdf, 2017. Accessed: 04.04.2019.

[8] "Lab 4: Motor control - design of electromechanical robotic systems, fall 2009." https://ocw.mit.edu/courses/mechanical-engineering/2-017j-design-of-electromechanical-robotic-systems-fall-2009/labs/MIT2017JF09lab4.pdf, 2009. Accessed: 15.03.2019.

[9] A. AG, "Arduino uno rev3." https://store.arduino.cc/arduino-uno-rev3. Accessed: 2019-03-26.

[10] MIT, "Understanding poles and zeros - analysis and design of feedback control systems." http://web.mit.edu/2.14/www/Handouts/PoleZero.pdf. Accessed: 11.04.2019.

[11] Adafruit, "Arduino adafruit touchscreen library." https://github.com/adafruit/AdafruitTouchScreen. Accessed: 04.04.2019.

[12] JonHub, "A realtime digital signal processing (dsp) library for arduino." https://github.com/JonHub/Filters. Accessed: 09.04.2019.

[13] B. Beauregard, "Arduino pid library." https://playground.arduino.cc/Code/PIDLibrary. Accessed: 08.04.2019.

[14] P. Dular and P. Kuo-Peng, "An efficient time discretization procedure for finite element - electronic circuit equation coupling," *COMPEL - The international journal for computation and mathematics in electrical and electronic engineering*, vol. 21 Issue: 2, 2002.

[15] M. Margolis, "Servo library." https://www.arduino.cc/en/Reference/Servo. Accessed: 03.03.2019.

[16] K. Heaning, S. Sohail, W. Kerbel, R. Trafford, P. Georgieva, N. Bouaynaya, and R. Polikar, "Dual axis solar panel control system," *2016 4th International Conference on Control Engineering Information Technology (CEIT-2016) Tunisia*, 2017.

Appendix A

Geometrical analysis

Fig. A.1 shows the geometrical analysis confirmation of the 20° rotation of the platform requirement. Thereby being able to set satisfactory dimensions of the servo arms and links between platform and servo arms.

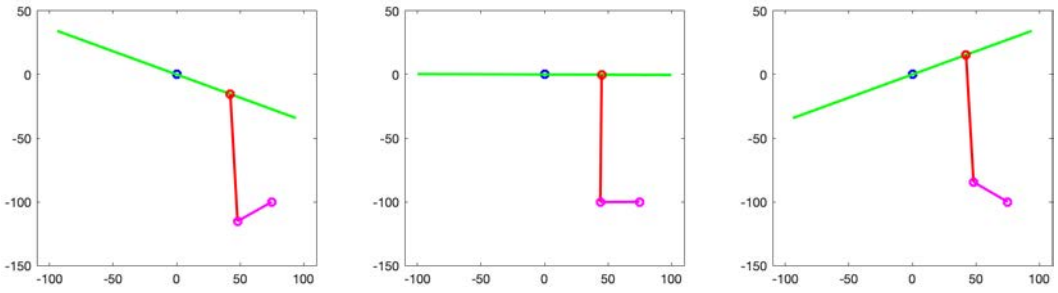

Figure A.1. Synthesis of boundary values, graphed with MATLAB

Appendix B

Construction drawings

A full view of the construction and detailed drawing are produced in Solid Edge ST9 as seen below. As follows in this appendix are the full detailed views of the essential solid non-electrical components. Moreover, the screws used are of the type M6x65mm and M2x25mm with their respective nuts.

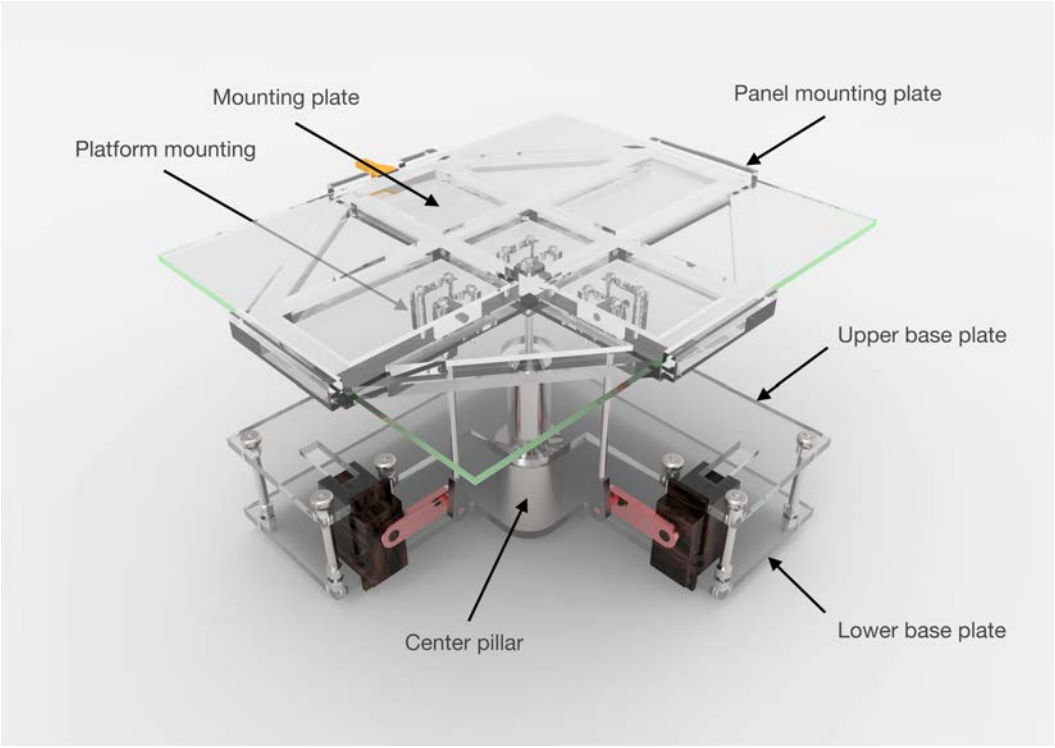

Figure B.1. Full view of construction, rendered in KeyShot 6

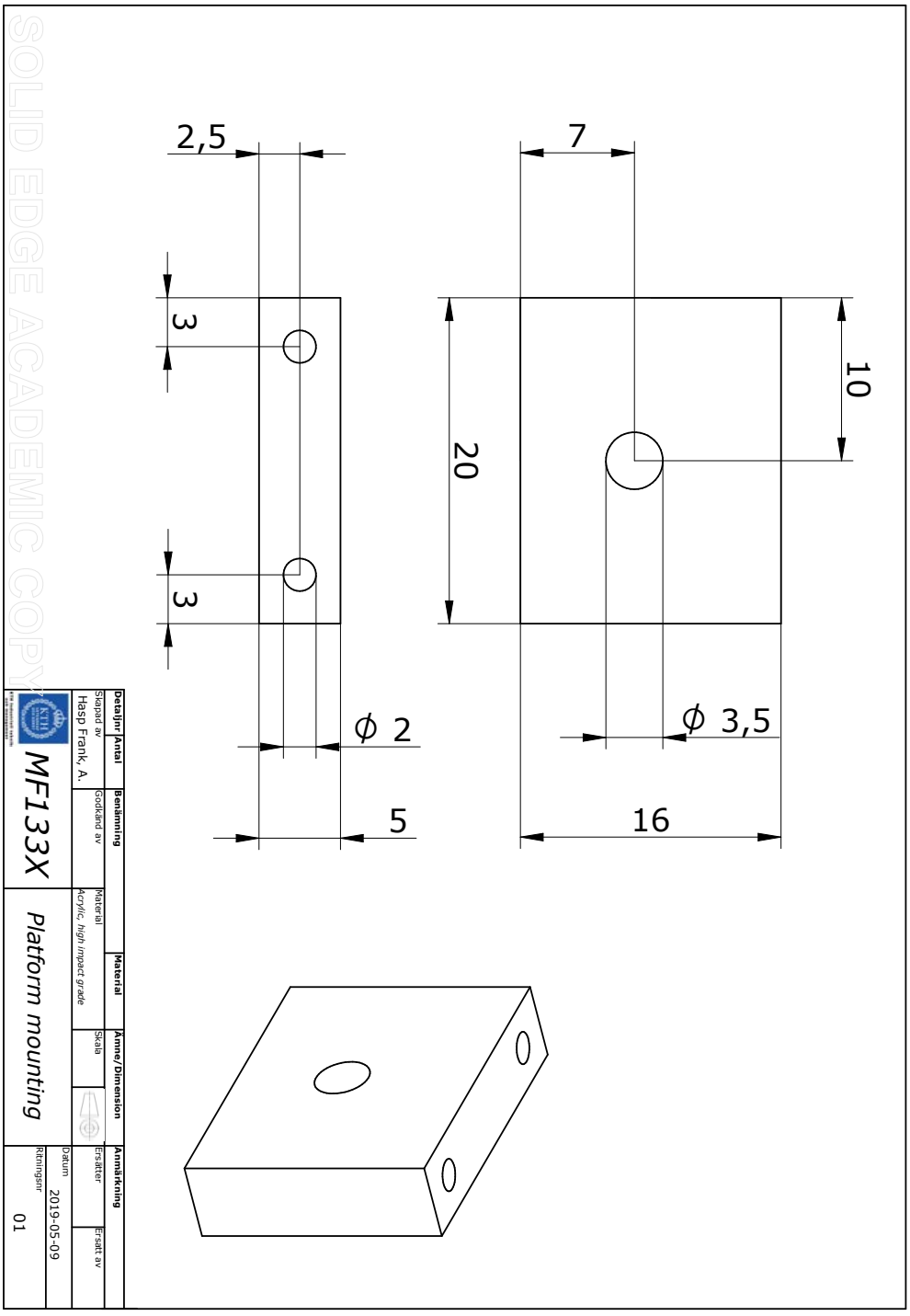

Figure B.2. Detail view of platform mounting

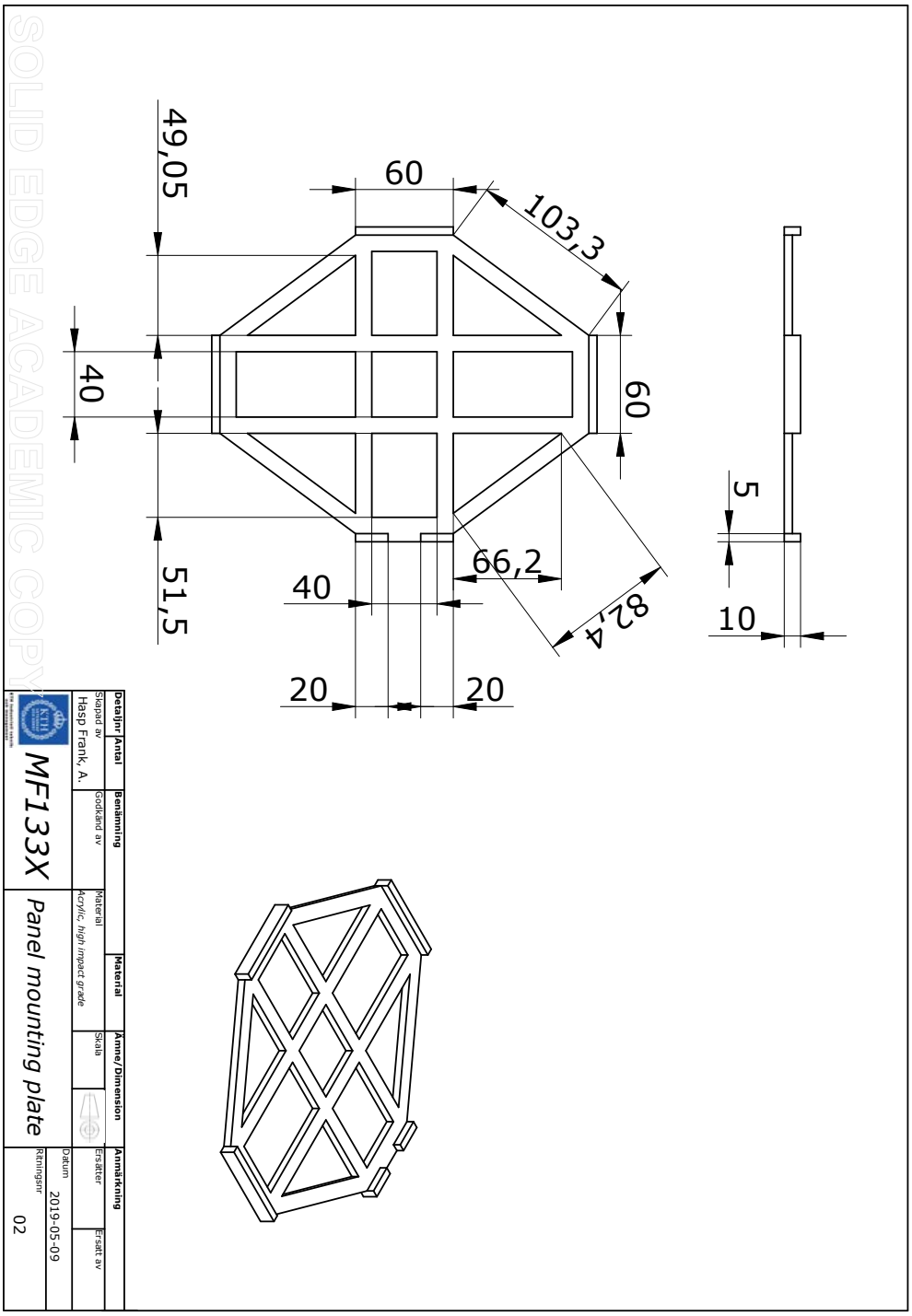

Figure B.3. Detail view of panel mounting construction

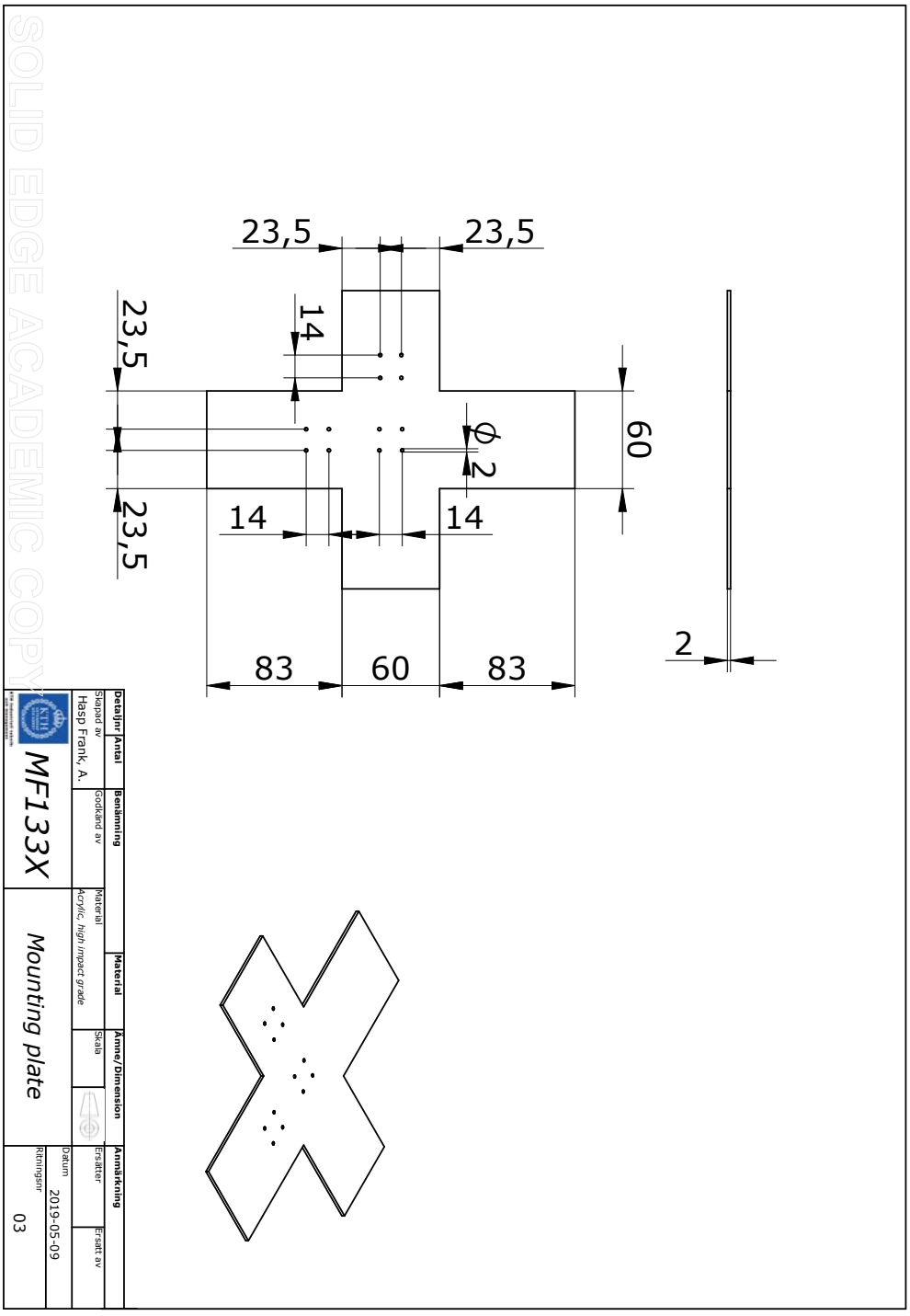

Figure B.4. Detail view of panel mounting base

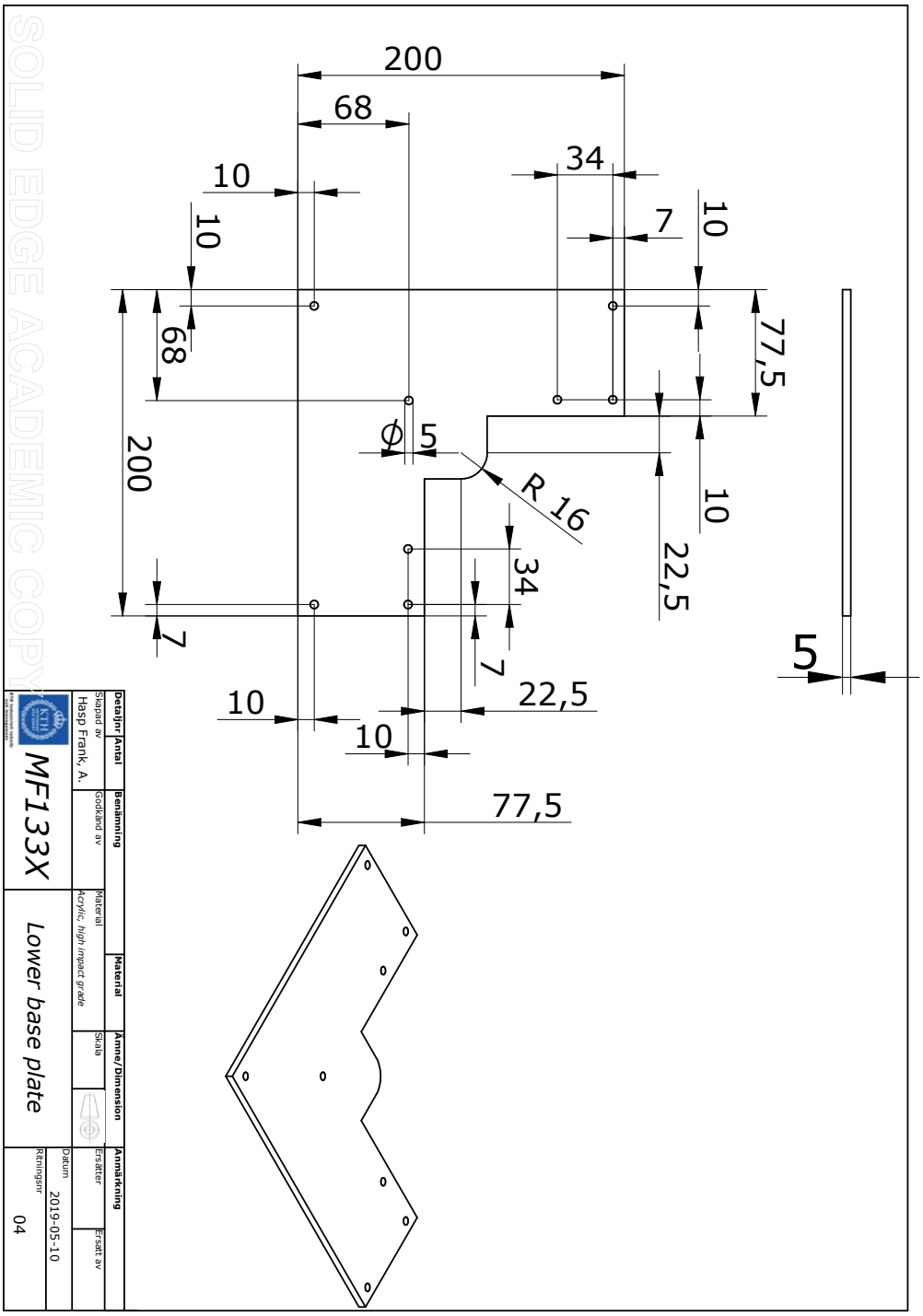

Figure B.5. Detail view of lower base platform

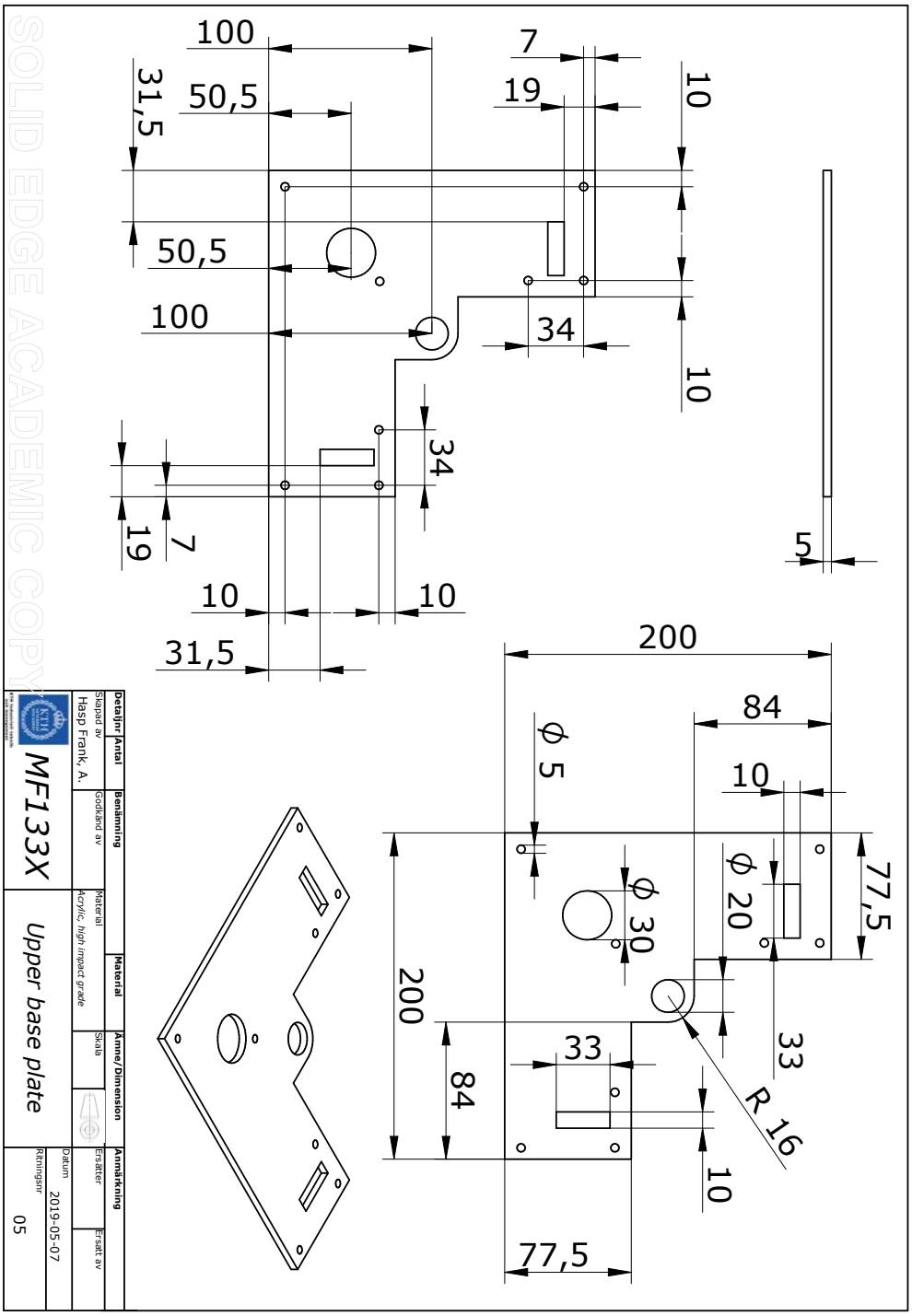

Figure B.6. Detail view of upper base platform

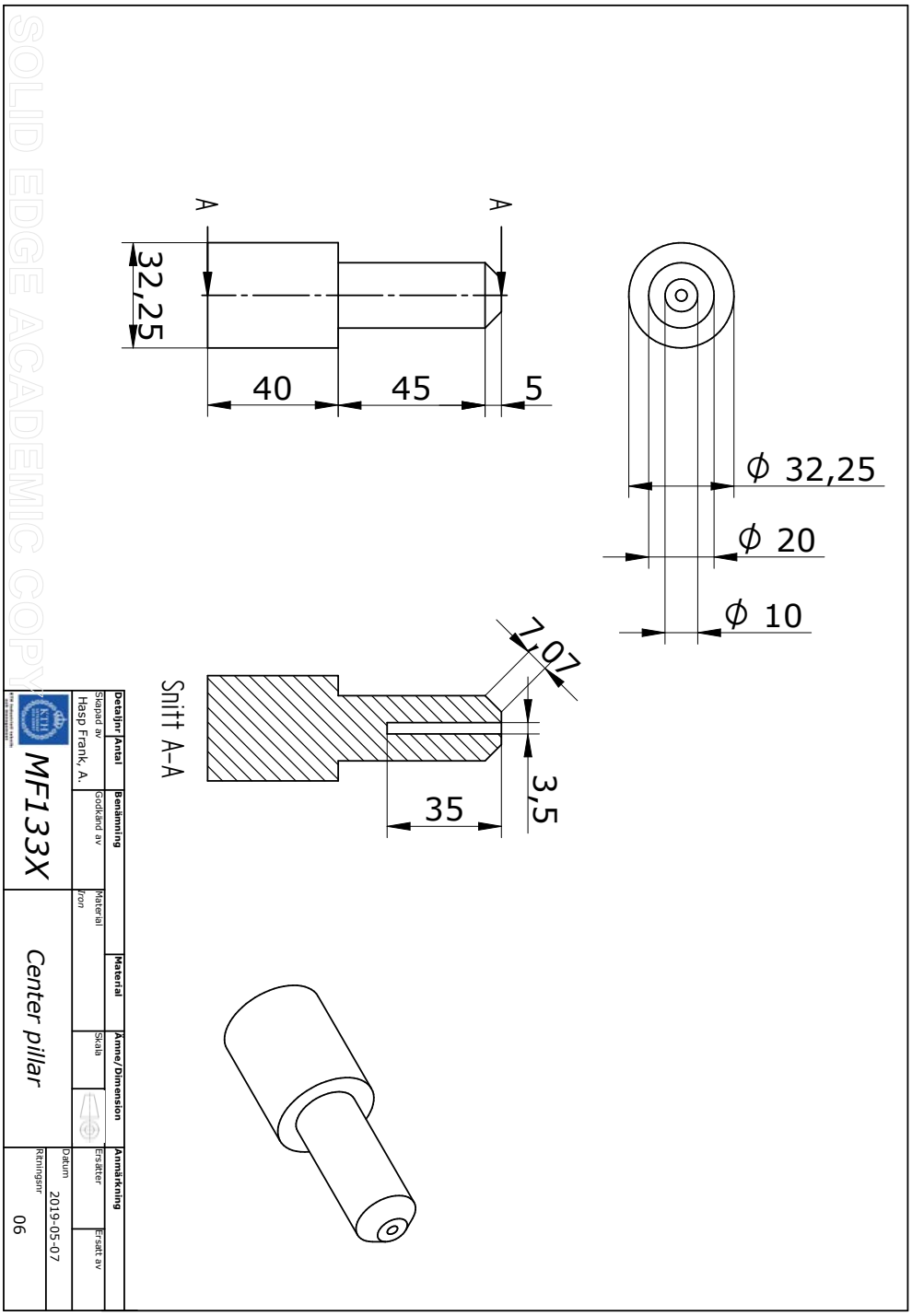

Figure B.7. Detail view of center pillar

Appendix C

System simulation

Fig. B.1 shows the full model of the SIMULINK structure and the respective scopes to read data. There within positions, angles, speeds and accelerations in the system.

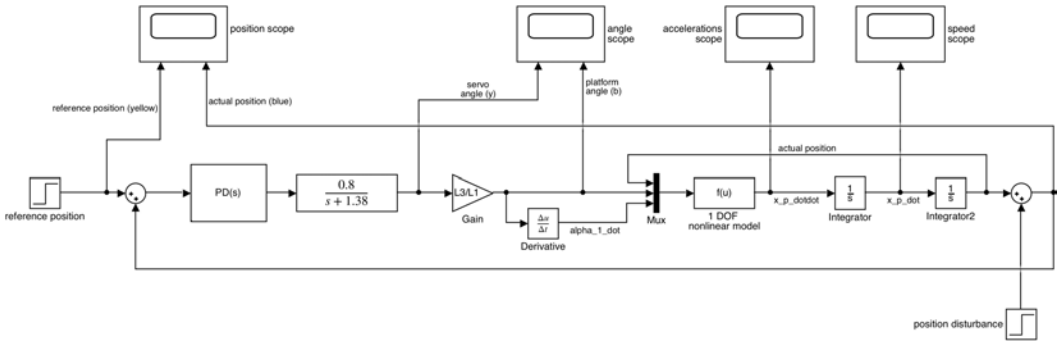

Figure C.1. SIMULINK model of 1 DOF system

Appendix D

Arduino code and flowchart

The following appendix contains the full source code used in the Arduino program for the ball balancing platform. The full Arduino library codes used are however not included but can be seen under INCLUDED LIBRARIES below in the code. Furthermore, a flowchart visualizing the code.

```
/* Ball_on_plate.ino
by Morgan Tjernstrom & Alexander Hasp Frank
modified 02-05-2019, KTH, Stockholm

A touch panel provides input to a PID controller.
The PID output is fed to two separated servo motors
which are controlling the rotation of two
orthogonal axes of a plate.
A ball is now being balanced by the plate.*/

//--- INCLUDED LIBRARIES ---//

#include <TouchScreen.h>
#include <PID_v1.h>
#include <Servo.h>
#include <Filters.h>

//--- PID_v1.h Installation ---//

// Defining pin numbers for output to be sent to
#define PinOutputX 11
#define PinOutputY 12

// Seting up variables for PID
double SetpointX, InputX, OutputX;
double SetpointY, InputY, OutputY;
```

```
27
28 // Tuning parameters for PID controller
29 float KpX = 0.162;
30 float KiX = 0.00;
31 float KdX = 0.039;
32
33 float KpY = 0.2;
34 float KiY = 0.0;
35 float KdY = 0.0505;
36
37 // Specified sampling time in milliseconds
38 int Ts = 15;
39
40 // Introduce PID from library, Reverse for counter-clockwise
       motor
41 PID myPIDX(&InputX, &OutputX, &SetpointX, KpX, KiX, KdX,
      DIRECT);
42 PID myPIDY(&InputY, &OutputY, &SetpointY, KpY, KiY, KdY,
      REVERSE);
43
44 int InputXfiltered, InputYfiltered;
45
46 //--- TouchScreen.h Installation ---//
47
48 // Defines Analog inputs/outputs for touch panel
49 #define YP A1      // Blue
50 #define XM A0      // White
51 #define YM 3       // Grey
52 #define XP 4       // Purple
53
54 // Touch panel variables
55 unsigned int noTouchCount;
56
57 // Initializing touch screen with pins
58 TouchScreen ts = TouchScreen(XP, YP, XM, YM, 575);
59
60 // Conversion from bits to centered bits
61 double Xcenter = 238;
62 double Ycenter = 873;
63
64 // Maximum/Minimum bits on touch panel
65 double Xmax = 282;
66 double Ymax = 910;
67 double Xmin = 195;
```

```
68 double Ymin = 836;
69
70 // Touch panel working area dimensions (mm)
71 double Xmm = 60;
72 double Ymm = 60;
73
74 //Conversion from bits to mm constant
75 double Xconversion = Xmm / (Xmax - Xmin);
76 double Yconversion = Ymm / (Ymax - Ymin);
77
78 //--- Filters.h Lowpass filter ---//
79
80 // filters out changes faster that 2.3 Hz. (Carefully tuned
      parameter)
81 float filterFrequencyX= 4;
82 float filterFrequencyY= 2.3;
83
84 // Initializing a one pole (RC) lowpass filter
85 FilterOnePole lowpassFilterX( LOWPASS, filterFrequencyX );
86 FilterOnePole lowpassFilterY( LOWPASS, filterFrequencyY );
87
88 //--- Servo.h Instllation ---//
89
90 // Introducing servo motors controlled
91 Servo myservo1;  // create servo object to control servo 1
92 Servo myservo2;  // create servo object to control servo 2
93
94 double theta1;
95 double theta2;
96 int servo_delay = 20;
97
98 // Servo calibration for a flat surface
99 double theta1flat = 90;
100 double theta2flat = 85;
101
102 // Maximum angular displacement values of servo motors (
      mechanical limit)
103 double max1 = 30;
104 double max2 = 30;
105
106 //--- LOOP SETUP ---//
107
108 void setup() {
109
```

```
110   // Attach Servo motors to pins and set to flat position
111   myservo1.attach(PinOutputX);
112   myservo2.attach(PinOutputY);
113
114   OutputX = theta1flat;
115   OutputY = theta2flat;
116
117   myservo1.write(OutputX);
118   myservo2.write(OutputY);
119
120   // Setting starting inputs and setpoints for PID in x,y-
         coordinates
121   InputX = 0;
122   InputY = 0;
123
124   SetpointX = 0;
125   SetpointY = 0;
126
127   myPIDX.SetMode(AUTOMATIC);
128   myPIDY.SetMode(AUTOMATIC);
129
130   // Setting limiting parameters for servo motors
131   myPIDX.SetOutputLimits(theta1flat-max1, theta1flat+max1);
132   myPIDY.SetOutputLimits(theta2flat-max2, theta2flat+max2);
133
134   myPIDX.SetSampleTime(Ts);
135   myPIDY.SetSampleTime(Ts);
136
137   theta1 = theta1flat;
138   theta2 = theta2flat;
139
140   delay(1000);
141
142 }
143
144 void loop() {
145
146   //--- TOUCH PANEL READ ---//
147
148   // Extracting pressure point value to initialize if loop
149   TSPoint p = ts.getPoint();
150
151   // if something is touching the plate compute PID output
         of input
```

```
152  if (p.z < 0) {
153
154
155    // Activate servo motors
156    myservo1.attach(PinOutputX);
157    myservo2.attach(PinOutputY);
158
159    // reset noTouchCount
160    noTouchCount = 0;
161
162    // Extracting position data
163    TSPoint p = ts.getPoint();
164
165    // Filtering touchpanel signal
166    InputXfiltered = lowpassFilterX.input(p.x);
167    InputYfiltered = lowpassFilterY.input(p.y);
168    InputX = ((InputXfiltered - Xcenter) * Xconversion);
169    InputY = ((InputYfiltered - Ycenter) * Yconversion);
170
171    // Computing output to servo motors
172    myPIDX.Compute();
173    myPIDY.Compute();
174
175    }
176
177    else {
178
179      noTouchCount++;
180
181      // Make platform flat if no touch
182      if(noTouchCount > 30) {
183        OutputX = theta1flat;
184        OutputY = theta2flat;
185
186      }
187
188      // Detach motors if no touch for a long time
189      if(noTouchCount = 300) {
190        myservo1.detach();
191        myservo2.detach();
192
193      }
194
195    }
```

```
196
197     // Write dynamic output to servo motors
198     myservo1.write(OutputX);
199     myservo2.write(OutputY);
200
201 }
```

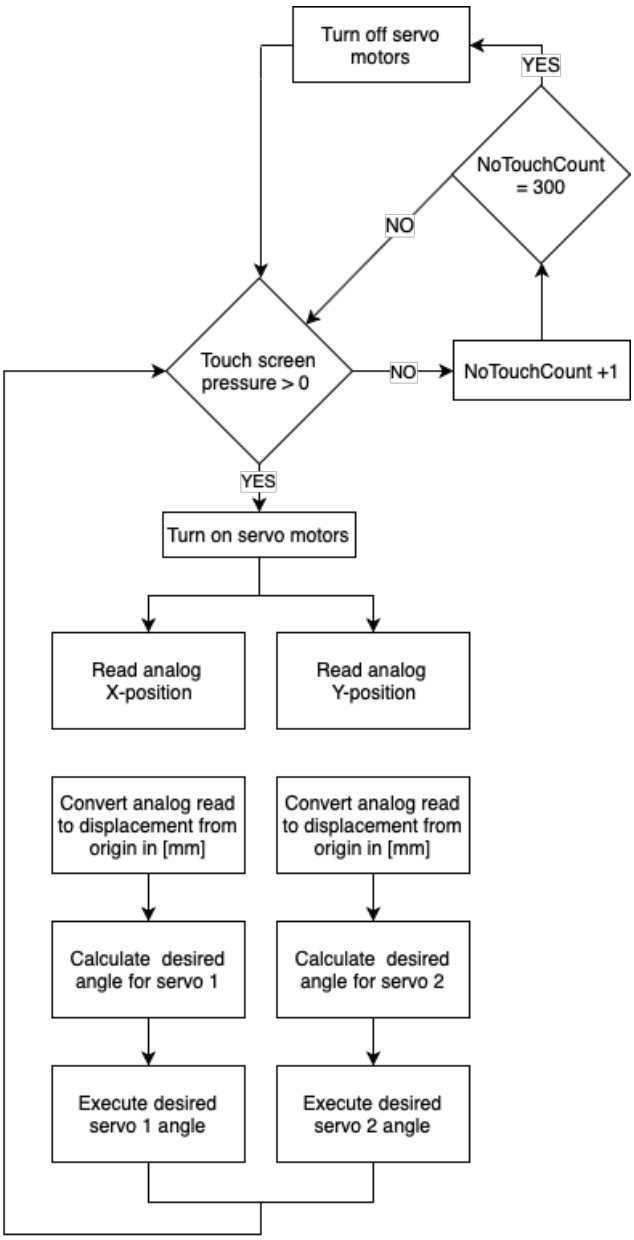

Figure D.1. Flowchart visualizing the Arduino code, created in draw.io

Appendix E

PID tuning

The iterative process of tuning the PID controller, or more specifically a PD controller, can be seen below. Noteworthy is that the integrating part of the controller is implemented when a static error is present. In the case of the ball balancing platform in the thesis, this was not the case. In order to achieve appropriate gain values the process used is listed below.

- Use initial values from system simulation
- Decrease values of K_D to gain oscillatory behaviour where ball remains on platform
- Reduce K_P until ball oscillates around setpoint
- Fine tune to achieve desired response

Table E.1. PID tuning iterations

Iteration Servo 1	K_P	K_I	K_D	Iteration Servo 2	K_P	K_I	K_D
1	0.2	0	1	1	0.2	0	1
2	0.2	0	0.5	2	0.2	0	0.1
3	0.2	0	0.25	3	0.15	0	0.1
4	0.2	0	0.1	4	0.3	0	0.1
5	0.2	0	0.075	5	0.2	0	0.1
6	0.2	0	0.05	6	0.2	0	0.075
7	0.15	0	0.05	7	0.2	0	0.05
8	0.1	0	0.05	8	0.2	0	0.06
9	0.17	0	0.05	9	0.2	0	0.0505
10	0.17	0	0.04	10			
11	0.16	0	0.04	11			
12	0.165	0	0.04	12			
13	0.162	0	0.04	13			
14	0.162	0	0.039	14			

TRITA ITM-EX 2019:35