University POLITEHNICA of Bucharest
Faculty of Electronics, Telecommunications and Information Technology

Cubic Structure Capable of Balancing

Bachelor's Thesis

submitted in partial fulfillment of the requirements of the degree of *Engineer*, in the domain *Electronics and Telecommunications*, study program *Microelectronics, Optoelectronics and Nanotechnology*

Coordinating professors
prof. Corneliu Burileanu
lect. Mihai Stafe

Student
Elena Sorina Lupu

Year 2015

Statement of Academic Honesty

I hereby declare that the thesis *Cubic Structure Capable of Balancing*, submitted to the Faculty of Electronics, Telecommunications and Information Technology in partial fulfillment of the requirements for the degree of *Engineer*, in the domain *Electronics and Telecommunications*, study program *Microelectronics, Optoelectronics and Nanotechnology* is written by myself and was never before submitted to any other faculty or higher learning institution in Romania or any other country.

I declare that all information sources sources I used, including the ones I found on the Internet, are properly cited in the thesis as bibliographical references. Text fragments cited "as is" or translated from other languages are written between quotes and are referenced to the source. Reformulation using different words of a certain text is also properly referenced. I understand plagiarism constitutes an offence punishable by law.

I declare that all the results I present as coming from simulations and measurements I performed, together with the procedures used to obtain them, are real and indeed come from the respective simulations and measurements. I understand that data faking is an offence punishable according to the University regulations.

Bucharest, July 2015.
Student: Elena Sorina Lupu

.....

Dedication

To my aunt, Emily

Acknowledgements

I would like to acknowledge the continuous support given by my parents, Cristina and Valentin Lupu and by my boyfriend, Valeriu Balaban, during the preparation of my thesis.

I would like to offer my special thanks to my supervisors, **professor Corneliu Burileanu** and **professor Mihai Stafe** for their constructive support in the development of this research work, their enthusiastic encouragement and their useful critique. My grateful thanks are also extended to **Mr. Michael Muhlebach**, one of the inventors of the Cubli project for providing me with useful bibliography and advice.

Special thanks should be given to **Marius Dima** and **Victoras Anghel**, engineers at AFT Design, to **Camil Muresan** from INCAS, to **Mr. Cristian Dobre**, engineer at FORHydraulic for their benevolent help to manufacture the prototype and for their brilliant mechanical hunts that led to a successful project. Great appreciation should be give to **George Bitlan**, for 3D printing more than 20 iterations of components of this project.

I would like to show my gratitude to **Ioana Josan-Drinceanu** for her lectures in Control Theory and for helping me with Matlab simulations. In addition, **Ion Ciobanu** from European Space Agency and **Cristian Soare** from Automation Control and Computer Science should be very thanked for their advices. Special thanks should be given to **prof. Mihnea Udrea** for providing me with a laboratory where I could build my experiment. **Mr. Istvan Templi** from MAXON must be thanked for his advice in choosing the right motor and motor driver for my project.

Last, but not least, I want to express my deep gratitude to **Valentin Preda** from Université de Bordeaux. IMS Laboratory, as his help came in a crucial moment of my project. He gave me advice of how a new method of control can be implemented on the microcontroller, a method which stabilized the system and made my thesis a success.

Contents

List of figures

List of tables

List of acronyms

BLDC	Brushless Direct Current
CCW	Counter-Clockwise
CW	Clockwise
DC	Direct Current
GUI	Graphical User Interface
LQR	Linear Quadratic Regulator
PID	Proportional Integral Derivative
RW	Reaction Wheel
RWIP	Reaction Wheel Inverted Pendulum
RWP	Reaction Wheel Pendulum
UART	Universal Asynchronous Receiver/Transmitter

Abstract

This thesis is concerned with the development of a reaction wheel inverted pendulum. A DC motor acts on the reaction wheel and provides sufficient torque to maintain the system at an equilibrum position. Two types of feedback controls were simulated and implemented - the Proportional-Integral-Differential Control and the Linear Quadratic Regulator. The results proved positive in simulation for both control. However, in practice, only the latter was able to provide stabilization for the system. This practical result was expected because the second control uses more variables to characterize the system: the tilt angle of the pendulum, the angular velocity of the pendulum and the angular velocity of the wheel. In constract, the PID control uses only the tilt angle.

Preamble

Time is not a river.
Time is a pendulum.

Arna Bontemps

The Universe is considered to be infinite. A circle is infinite - one can go around forever. Yet, we are limited by the illusion that everything around us has a beginning and an end. A perpetual motion which works forever is epistemically impossible. Still, to dream of a gyroscope that never stops rotating and a pendulum which swings eternally must not be forbidden.

At the age of 15, I bought a Levitron - a toy which is able to spin and float in the air due to an induced magnetic field. I was then fascinated by the unseen forces which kept the spinning top in the air for a couple of minutes.

Several years later, I had discovered the CUBLI project, developed by engineers at University in Zurich, which implies the development of a cube able to stabilize on one edge and corner. I was so amazed that I started to dream building my own cubic structure.

This dream is now translated into my bachelors thesis

1. Introduction

1.1 The Pendulum Concept

Pendulums are present everywhere around us, translated in different forms. To give a definition, a pendulum is simply an object swinging freely relative to its equilibrium position. Thus, its motion can be easily defined as a simple harmonic oscillation. The idealised pendulum contains a weight (usually called a bob) at the end of an ideal chord suspended from a pivot. In this case, frictions are neglected.

1.1.1 History

When discussing about pendulum and its properties, it is requisite to value the discoveries done in the past in this direction. One of the earliest approaches of a pendulum starts early in the first century, during the Chinese Han Dynasty and it was used to determine for the first time the direction of an earthquake. An article from 2009 [4] points out that "it was over 1700 years later that one similar instrument was invented in Europe".

Figure 1.1: First Seismograph

As depicted in [5], "the seismoscope, which was approximately 1.82 m tall, resembled a gigantic bronze urn with eight tubes shaped as dragons' heads around the top. Surrounding the seismoscope were four bronze toads which were positioned in the four directions. Inside the casing was a pendulum, a crank and right-angle lever, and some bronze balls. In the event of an earthquake, the pendulum would swing, the crank and lever would raise one of the dragons' heads, and the balls would drop out of the mouth of the dragon into the mouth of a bronze toad. This would create a large 'clang', which served as a warning alarm, and the toad indicated the direction of the quake."

Galileo Galilei was the first to study the properties of a pendulum and discovered an important aspect: the period of the swing is independent of its amplitude. This fact enabled the application of pendulum in time measurements. In 1973, Huygens provided the most accurate measurement of time by using a pendulum. The accuracy of these clocks increased to few seconds per day, which enabled precision measurements in astronomy, navigation etc. [6]

1.1.2 Types and applications

The simple pendulum is a type of pendulum which swings only back and forth. A spherical pendulum has its bob experiencing a circular motion (e.g. scary machines in a park which spin you in a circle). Another specific example of a pendulum is an inverted pendulum which is an unstable system because its center of mass is above its pivot point. It can be stabilized using proper applied force, thus it is considered one of the most difficult system to be controlled in engineering. [7]

A relevant example of an inverted pendulum is you. A person in an upright position must constantly adjust his or her moves to maintain balance while performing different activities. This is easily extrapolated to humanoid robots which relay on simplified models of inverted pendulums to perform bipedal walking. NAO, 1.2 an autonomous, programmable humanoid robot developed by Aldebaran Robotics is an example of this fact. [8]

1.1.3 The Challenge

In general, engineers try to find simplifications of the real-time issues by considering the approach of *linear systems*. In contrast, the dynamics of the pendulum is interesting to be studied due to its non-linearity behavior (the output of the system is not directly proportional to the input). My main interest that will be emphasized in this thesis is in terms of **stabilization of an inverted pendulum** through different control algorithms.

Figure 1.2: NAO Humanoid Robot

1.2 The Reaction Wheel Concept

A reaction wheel is a spinning mass which provides a reaction torque coming from the rotational acceleration. The principle of reaction wheel is used in space technology because it controls the spacecraft attitude ("the angular orientation of a spacecraft body vector with respect to an external reference frame"). [9]. Mechanically, a motor is attached to a flywheel. When the motor starts rotating, the spacecraft starts counter-rotating. By using a single reaction wheel, one axis can be stabilized . For 3D attitude control, three reaction wheels are needed.

The Challenge

Improvements in the research of attitude control using reaction wheels is mandatory. Why? Because a mal-function of a RW can lead to a space mission failure. This is what happened in the Kepler Project which is mainly a telescope searching for Earth-size exoplanets orbiting solar type stars (or in other words to see if there are any planets like Earth in the distant Universe). [10]. Two of the Kepler's reaction wheels have failed. This lead to a mission failure because the telescope was not able to point accurately to its target.[11]

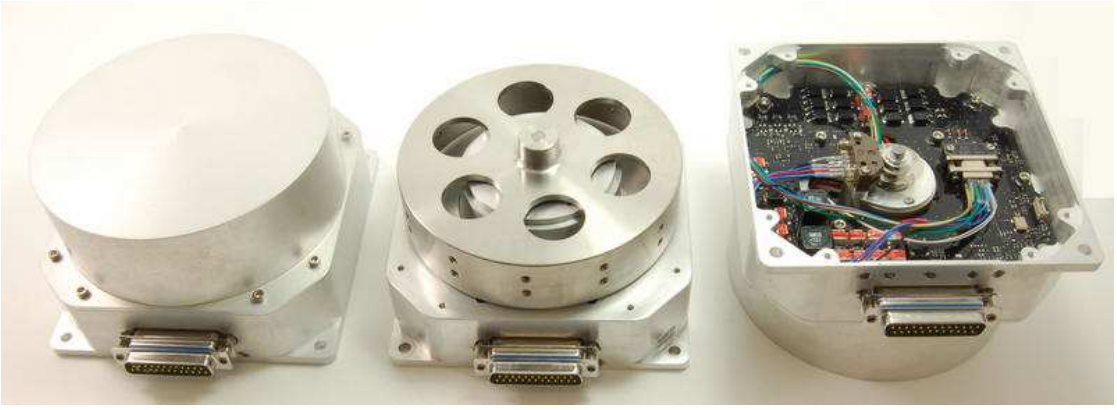


Figure 1.3: SmallSat Reaction Wheel as designed by CLYDESPACE

1.3 Reaction Wheel Inverted Pendulum Concept

If we combine the previous two concepts, we will obtain a reaction wheel inverted pendulum which is an innovative inverted pendulum device firstly introduced in 1999 by Mark W. Spong, Peter Corke and Rogelio Lozano in the paper *"Nonlinear control of the inertia wheel pendulum"* [12]. This platform can be considered useful from two points of view. On one hand, it offers a pedagogical approach, and on the other hand it is a practical instrument to test different control algorithms.

1.3.1 Problem definition and Requirements

Statement: Design and built a reaction wheel inverted pendulum capable of balancing on one edge and corner.

The problem aforementioned implies two steps:

1. One dimensional prototype

   A reaction wheel is attached to a cubic face (considered the pendulum body). By applying torque, the system is controlled to stay in upright position and in equilibrium, respectively. The system is also insusceptible to any external force applied. This system can start from two positions:

   (a) From Ground Level

   In this case, a jump up maneuver is required to bring the system to the upright position. Then, the algorithm for stabilization starts, as in Figure 1.4

   (b) From Equilibrium

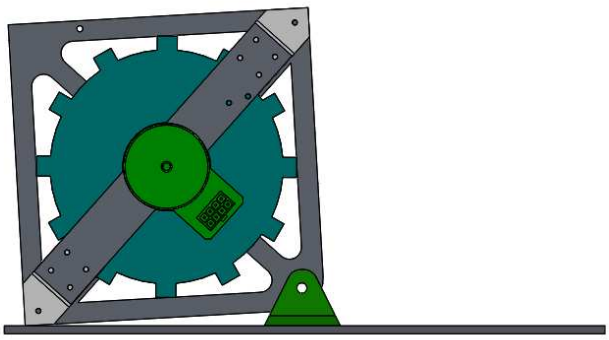


Figure 1.4: One dimensional system from ground level

In this case, the system must constantly adjust the motor torque to keep the pendulum in balance. Any external force will move the pendulum from its equilibrium state. In this case, a feedback control algorithm is used. 1.5

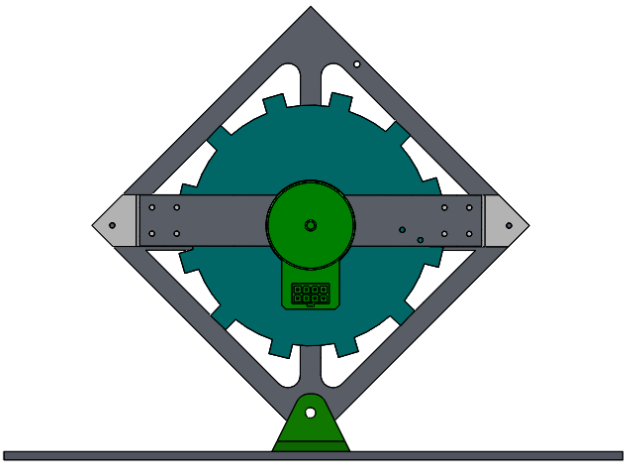


Figure 1.5: One dimensional system in balance

2. Three dimensional prototype

   Three reaction wheels are attached in a cubic structure. Through a control algorithm, the cubic is made to balance on its edge and corner In this thesis, this part is only theoretically presented, without any practical implementation. A similar version was developed and implemented by Mohanarajah Gajamohan et. al. from University of Zurich. 1.6

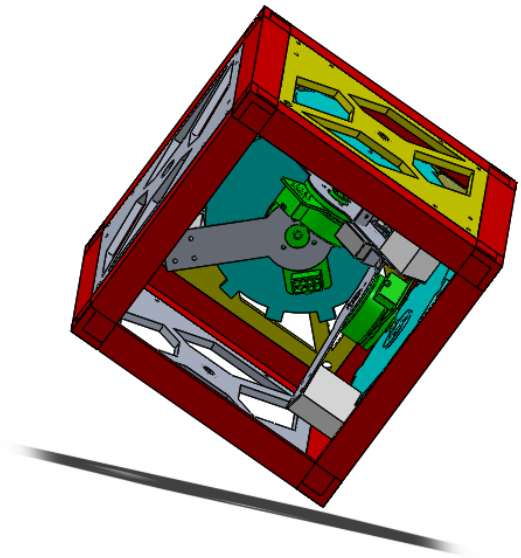


Figure 1.6: Three Dimensional System

The cube will always perform in one of the following two states:

- **Jump - Up**: from ground level, using a mechanical brake, the prototype will jump to the upright position.
- **Balance**: if experiencing external forces, the cube will be able to respond and counteract the motion.

In this thesis, these states will be referred either as **JUMP-UP** or **BALANCE**. The RWIP is called CUBalance.

The challenge of this project stands in the sphere of using only one parameters to control the entire complex system: the velocity of the DC motor. The next diagram outlines the input and states on this system, as well as the method of measuring the variables.

After stating the problem, the next step done in developing the thesis was to clearly outline the requirements:

1. *A cubic structure capable of balancing on one corner and edge should be designed and built.*
2. *The cube should contain reaction wheels controlled by motors, as well as motor drivers, sensors, RC servomotors, but should not include the power source.*

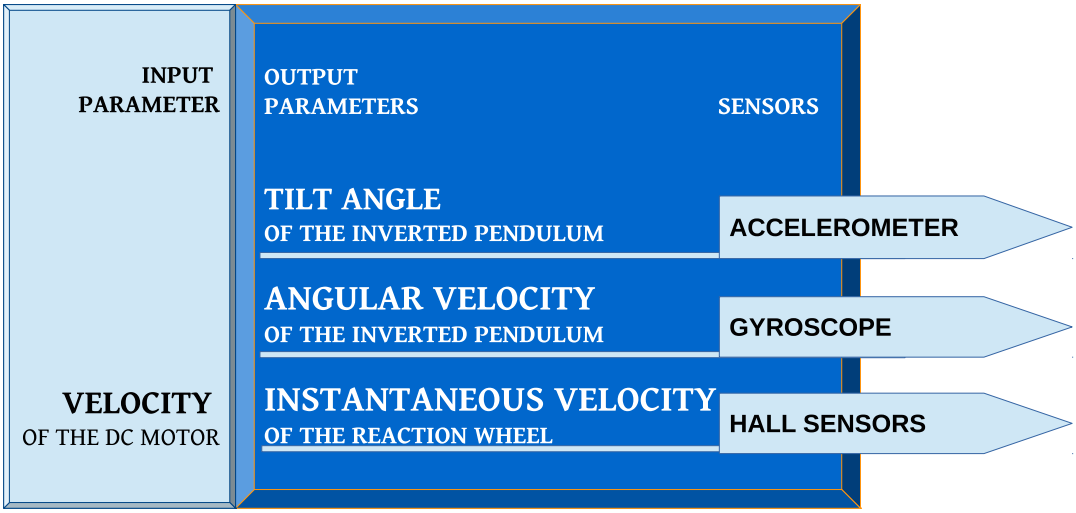


Figure 1.7: Input and Outputs of the System

3. *The INPUT parameter (the only parameter controlled in the cube) is the velocity of the motor.*
4. *The STATE parameters are the angular velocity of the cube, the angle relative to the ground and the instantaneous velocity of the motor.*
5. *The cube should respond by combining the INPUT and OUTPUT parameters in a control algorithm developed on the microcontroller.*
6. *The JUMP UP of the cube must be done using a mechanical braking.*
7. *The cube should receive commands from the ground station and should respond accordingly.*
8. *The cube should have a dimensions of 20 x 20 x 20 cm maximum.*
9. *The cube should not exceed 1.5 kg.*
10. *The cube chassis should be made of aluminum and plastic.*

1.4 Related projects

The following innovative project was taken as reference in the development of this bachelor thesis.

1.4.1 Cubli - a cube able to jump up and balance

"The Cubli was developed by the ETH Institute for Dynamic Systems and Control located in Zurich, and is part of ongoing research into inverted pendulum systems. Cubli is a cube (15 x 15 x 15cm) with the ability to jump up from a resting

position without the aid of an external support and balance on one corner or edge. It consists of an aluminum housing, light but strong enough to withstand the jump up and controlled falling. Within the housing are three reaction wheels through the motors, a STM32 discovery board (ARM7 Cortex-M4, 168MHz) as its main controller, and six IMUs (one for each face) consisting of a rate gyro and accelerometer connected to the main controller via a I2C Bus. Also included are a three 50 W brushless DC motor (used to drive the reaction wheels), three DEC 36/2 modules, digital 4 quadrant brushless DC motor controllers. The motor and main controller communicate via the CANopen protocol. Power to the Cubli is provided by a constant external voltage supply. For the software, the STM32 port of the FreeRTOS scheduler is used to estimate and control algorithms, and an Eclipse based tool chain is used for any development. The system of spinning wheels is similar to what stabilizes satellites in space, and it is hope that the technology can be used to assist in future planetary exploration." [13]

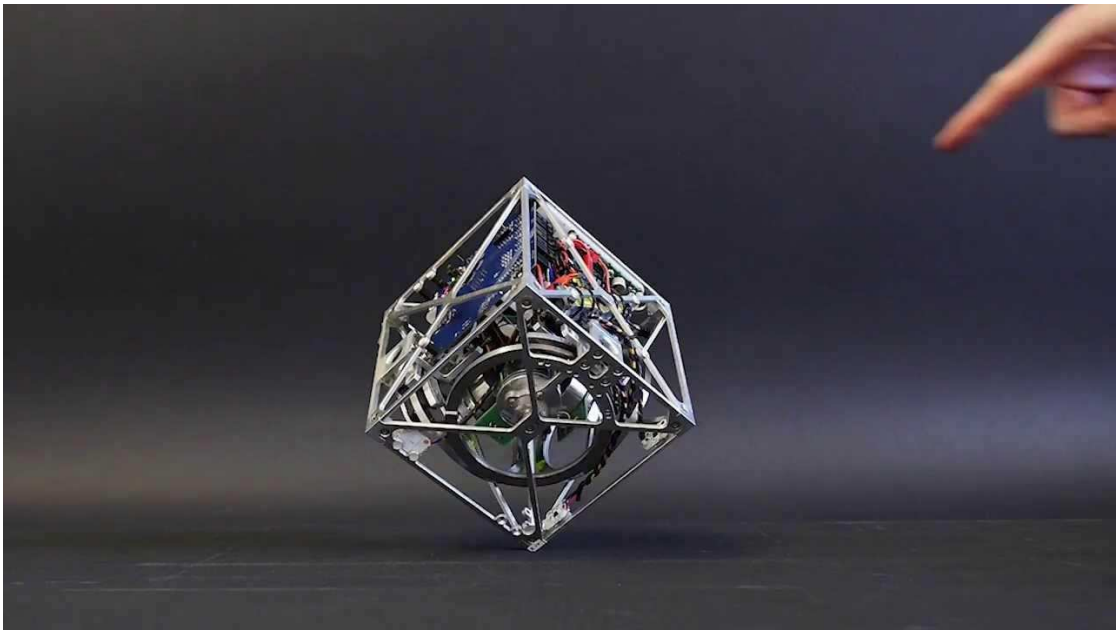


Figure 1.8: Structure of Cubli

My project, entitled Cubalance, aims to replicate the Cubli prototype, by innovating in terms of structure and stabilization system. Also, by gaining knowledge in control theory, further innovative stabilization system can be developed for the CUBalance.

The following components were taken as a reference from Cubli

- The concept

- The type of the DC motor and its driver
- The equations of motion for the system
- The braking system in order to make the Cube jump up

The following components were different and added in my project:

- The mechanical structure of the reaction wheel and materials used
- The demonstration of the equations of motion
- The development board for the microcontroller and the sensors
- The control software

2. Background

This chapter aims at providing an overview of the concepts used to develop CUBalance.

2.1 Rotational Motion

To understand the kinematis of circular motion and further equations developed in this thesis, several basic notions must be introduced.

2.1.1 Uniform Circular Motion [14] [1]

Velocity vector in linear motion is directed along the line of movement. However, in circular motion, the direction of velocity is always tangent to the circle, and it moves with the object. [14] [1]

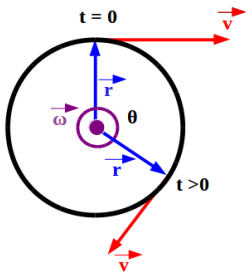


Figure 2.1: Circular Motion

Figure 2.1 outlines a particle executing a circular orbit of radius r with uniform tangential speed v. The instantaneous position is easily defined in terms of angle θ. Firstly, we should decide that at t = 0, the object finds itself at $\theta = 0$ °. Extrapolating in time, we obtain

$$\theta = \omega t \tag{2.1}$$

with ω the angular velocity of the object. We consider that the particle is tracing a circular arc with length s. If θ is measured in radians, we obtain

$$s = \theta r \tag{2.2}$$

Let's consider the motion in short time interval dt. During this time, the object moves

$$\begin{aligned} d\theta &= \omega dt \\ ds &= r d\theta \end{aligned} \tag{2.3}$$

However $\frac{ds}{dt}$ is the tangential velocity. Thus, we have

$$v = r\omega \tag{2.4}$$

To obtain the direction of angular velocity ω, we will use the vectorial equation

$$\vec{v} = \vec{\omega} \times \vec{r} \tag{2.5}$$

2.1.2 Angular Momentum [1]

Let's consider a particle of mass m with a velocity v. The linear momentum can be written as:

$$\vec{p} = m\vec{v} \tag{2.6}$$

The second law of Newton can also be written in terms of linear momentum, because it is known that the derivative of velocity with respect to time is the acceleration:

$$\vec{F} = m\vec{a} = \frac{d\vec{p}}{dt} \tag{2.7}$$

Objects performing a circular motion posses a quantity named angular momentum (L). This is important because all the experiments show that this quantity can be conserved: it can be transferred, but it cannot be destroyed or created.

$$\vec{L} = \vec{r} \times \vec{p} \tag{2.8}$$

where $\vec{r}$ is the position vector.

In terms of magnitude,

$$L = rp\sin\theta, \tag{2.9}$$

,where θ is the angle between $\vec{r}$ and $\vec{p}$. The direction of $\vec{L}$ is perpendicular to the plane obtained with $\vec{r}$ and $\vec{p}$, in the direction given by the right-hand rule.

If we differentiate equation (2.8) with respect to time

$$\frac{d\vec{L}}{dt} = \frac{d\vec{r}}{dt} \times \vec{p} + \frac{d\vec{p}}{dt} \times \vec{r} = \vec{v} \times \vec{p} + \vec{F} \times \vec{r} \tag{2.10}$$

Using equation (2.6) and $\vec{v} \times \vec{v} = 0$, we get:

$$\frac{d\vec{L}}{dt} = \vec{F} \times \vec{r} = \vec{T} \tag{2.11}$$

where T is the torque. This torque is equivalent to the force in linear motion.

2.2 Moment of Intertia [1]

A system is made of N elements. The ith element has mass m_i, position vector r_i and velocity v_i. Then, the kinematic energy of the system is

$$K = \frac{1}{2}\sum_{k=1}^{N} m_i v_i^2 \tag{2.12}$$

If $\vec{v_i} = \vec{\omega} \times \vec{r_i}$ or, scalar, $v_i = r_i\omega$. Then,

$$K = \frac{1}{2}\sum_{k=1}^{N} m_i r_i^2 \omega^2 = \frac{1}{2} I\omega^2 \tag{2.13}$$

where $I = \sum_{k=1}^{N} m_i r_i^2$ is the moment of inertia of the object with respect to the rotational axis.

Observations [1]

- The moment of intertia is an aditive quantity and is equal to the moment of intertia of the constituent elements.
- The role of mass in the linear motion is replaced by the moment of intertia in the rotational motion.
- The dimension of the moment of intertia is related not only to the mass of constituent particles, but to the way they are distributed (the distance to the rotational axis). For the same total mass, the object which has its mass distrubuted far from its axis point experiences a higher moment of intertia than an object with it mass distribution closer to the axis point.

- In rotational motion, the angular momentum is $L = I\omega$
- An important theorema is **Steiner Theorema** (or Parallel Axis Theorema)

  Suppose a body with mass m which rotates about an axis Z, passing through the body's center of mass. *The moment of intertia regarding an arbitrary axis (Z') parallel to the Z axis is equal to the sum between the moment of inertia of a central axis, parallel to the arbitrary one and the product between the object's mass and the square distance between these axes.*

$$I = I_0 + mR_0^2$$

2.3 Control Algorithms [2]

In this section, the control concepts used to stabilize the system are presented.

Process control has a fundamental role in manufacturing different products. Flow rates, speed, temperatures are just simple examples of variables that must be controlled to get a desired product. But why do we need actually to control these parameters? In general for safety. Imagine what can happen in a plant if all the parameters are not working as required. To overcome this issue, thoughtout the time, several algorithms were implemented to control these variables. One of the most popular is the PID algorithm due to its simplicity and reliability. [15].

The feedback control diagram of any system is represented in Figure 2.2, where u is the control command, y the output/measurements, r is the reference input and e the response error.

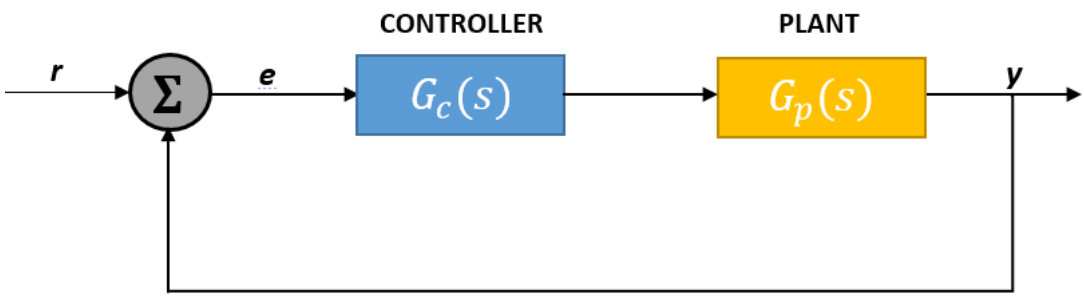


Figure 2.2: Feedback Control Diagram - Closed Loop

In control theory, there are several methods to determine the stability of a system. One solution is to test the stability of a linear system from its transfer

function. [3]. Therefore, let the plant transfer function and the controller transfer function be:

$$G_p(s) = k_{pl}\frac{N_p}{D_p} = k_{pl}\frac{\prod_{i=1}^{n_{pz}}(s - z_{pi})}{\prod_{i=1}^{n_{pp}}(s - p_{pi})} \tag{2.14}$$

$$G_c(s) = k_c\frac{N_c}{D_c} = k_c\frac{\prod_{i=1}^{n_{cz}}(s - z_{ci})}{\prod_{i=1}^{n_{cp}}(s - p_{ci})} \tag{2.15}$$

The loop transfer function is

$$L(s) = G_c(s)G_p(s) \tag{2.16}$$

Using the following two relationships

$$\begin{aligned} y &= G_pG_ce \\ e &= r - y \end{aligned} \tag{2.17}$$

we obtain that

$$\frac{y}{r} = \frac{G_pG_ce}{e + y} = \frac{G_pG_ce}{e + G_pG_ce} = \frac{G_pG_c}{G_pG_c + 1} = G_{cl}(s) \tag{2.18}$$

where

$$G_{cl}(s) = \frac{k_ck_{pl}N_cN_p}{D_cD_p + k_ck_{pl}N_cN_p} \tag{2.19}$$

is the closed loop transfer function.

A transfer function is defined by poles and zeroes because they can be used to reconstruct the system. Because they are complex numbers, they need to be represented graphically in the complex s-plane. Zeroes are represented by a circle, whereas the poles by a cross. Their location provides insights into the response of the system , as in Figure 2.3. If any pole as a positive real part, then there is a component in the output that increases to infinity with time. Thus, the system is unstable. In order to have stability, all the poles must be in the left-half of the s-plane.[3]

2.3.1 P.I.D Algorithm

As stated before, the P.I.D. Controller can be considered the most known form of feedback control. P.I.D. comes from Proportional - Integral - Derivative. Nowadays, more than 95% of the control loops use P.I.D. It is sometimes called the

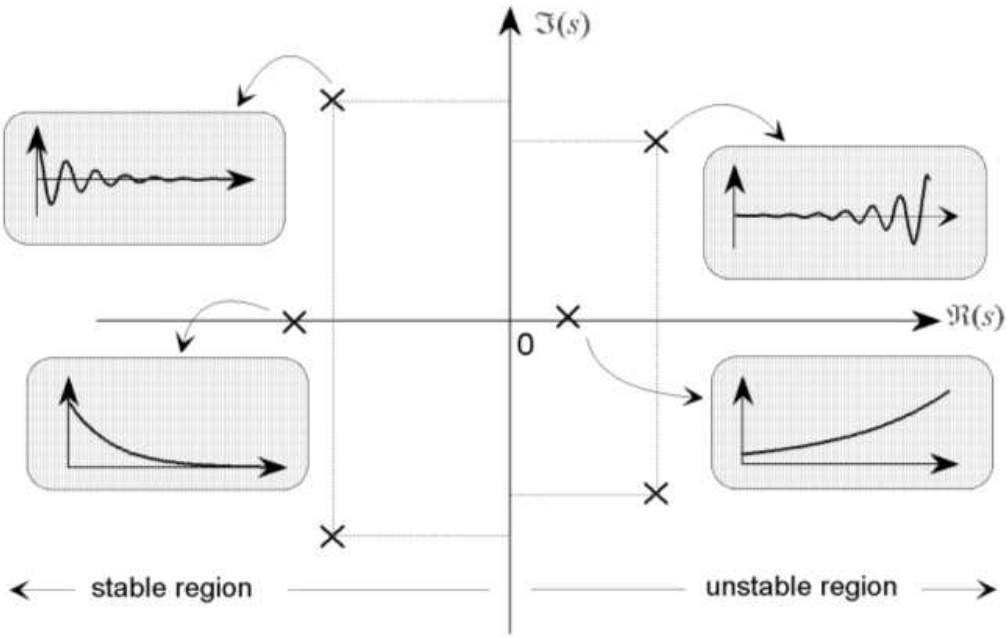


Figure 2.3: Pole-Zero Plot[3]

"Bread and butter in control engineering" and dates from 1890. [16] As an overview, a P.I.D. controller calculates an error between a measured value and a desired one. The equation that stands behind a P.I.D algorithm is: [16]

$$u(t) = K\Big(e(t) + \frac{1}{T_i}\int_0^t e(\tau)d\tau + T_d\frac{de(t)}{dt}\Big) \tag{2.20}$$

,where u is the control signal, e is the error, K is the proportional gain, T_i is the integral time and T_d is the derivative time. [16]

If we apply Laplace transformation to the previous equation, we obtain

$$u(s) = K\Big(1 + \frac{1}{sT_i} + sT_d\Big) \tag{2.21}$$

where the transfer function is $H(s) = K(1 + \frac{1}{sT_i} + sT_d)$. Therefore,

$$\begin{aligned} K_p &= K \\ K_i &= \frac{K}{T_i} \\ K_d &= KT_d \end{aligned} \tag{2.22}$$

Interesting to mention, the mathematical relationship was obtained *after* the controller had been developed. The integral mode was, at the early stages of PID development, called *automatic reset* while the derivative was entitled *hyper-reset* or *pre-act*. [17]

Practically, the control signal is a sum of three variables:

- the P-term (proportional to the error)
- the I-term (proportional to the integral of the error)
- the D-term (proportional to the derivative of the error)

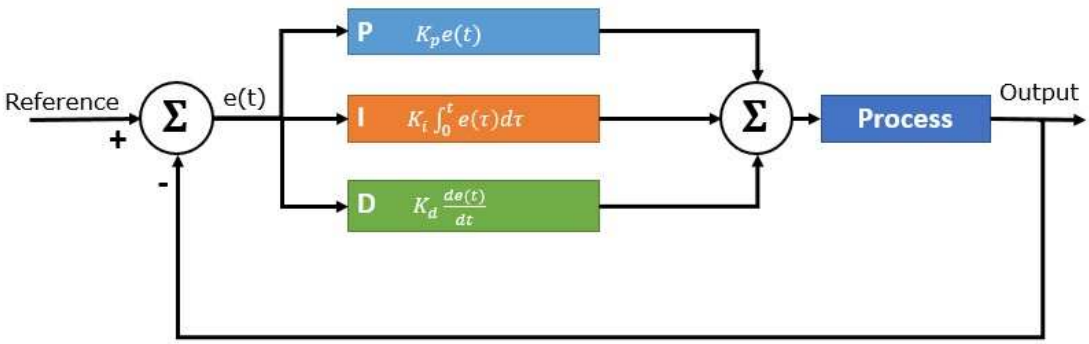


Figure 2.4: PID control

The setting of the coefficients is done using the practical testing. The process of choosing the P, I and D to get an ideal response is called tuning. [16]

To start the P.I.D. algorithm, we should set I and D to 0, and then increase P until the output of the loop oscillates. The integral coefficient has a role in stopping the oscillations. The derivative coefficient has a role in overshotting, yet can cause the system to be sensitive to noise.[16]

Because the P.I.D. controller is implemented on a microcontroller, the continuous form of the equation cannot be used. We need therefore to approximate the integral and the derivative terms as

$$\frac{de(t)}{dt} \simeq \frac{e(t) - e(t-1)}{T} \tag{2.23}$$

and

$$\int_0^t e(\tau)d\tau = T \sum_{i=1}^{t} e(i) \tag{2.24}$$

2.3.2 Linear Quadratic Regulator [2]

The Linear Quadratic Regulator assumes a linear dynamic system

$$\dot{x}(t) = Ax(t) + Bu(t) \tag{2.25}$$

where $x(t)$ is the state at time t and $u(t)$ is the input at time t.

$$u(t) = -Kx \tag{2.26}$$

The quadratic cost function is

$$\int_0^\infty (x^T Qx + u^T Ru)dt \tag{2.27}$$

with Q and R positive definite matrices. The objective is to find the matrix K which minimizes the cost function. K is defined as $R^{-1}B^T P(t)$, where P can be found by solving the Riccati differential equation.

$$A^T P(t) + P(t)A - P(t)BR^{-1}B^T P(t) + Q = -\dot{P}(t) \tag{2.28}$$

3. Experimental Design

3.1 Mathematical Model

The most important approach in understanding the behaviour of a reaction wheel inverted pendulum is to build a rigorous mathematical model. Firstly, a one dimensional model was built in SolidWorks to fully understand how every component interacts with each other. Then, using Motion Analysis from SolidWorks, dynamics was simulated to observe the system behavior at different motor speeds. These results were validated through mathematical calculations and are presented in the next pages.

Classical dynamics considers two approaches: *vectorial dynamics* and *analytically dynamics*. The former has its roots in the Laws of Motion, developed by Newton, whereas the latter is concerned with the system as a whole and uses kinetic energy (T) and potential energy (V) to characterize the system. [18]

We will start with the consideration that the system contains two important parts: **the reaction wheel** (abbreviated with w) and the **pendulum body** (abbreviated with p). Figure 3.1 contains the one dimensional model, with the following notations:[19] [8]

- θ_p = angle of the pendulum with respect to the surface normal
- θ_w = rotational displacement of the momentum wheel with respect to the pendulum axis
- m_p = mass of the pendulum body
- m_w = mass of the momentum wheel
- m_t = total mass of the system
- I_p = moment of inertia of the pendulum about its center of mass
- I_w = moment of inertia of the momentum wheel about its center of mass
- L_p = the distance between the pivot and the center of mass of the pendulum body

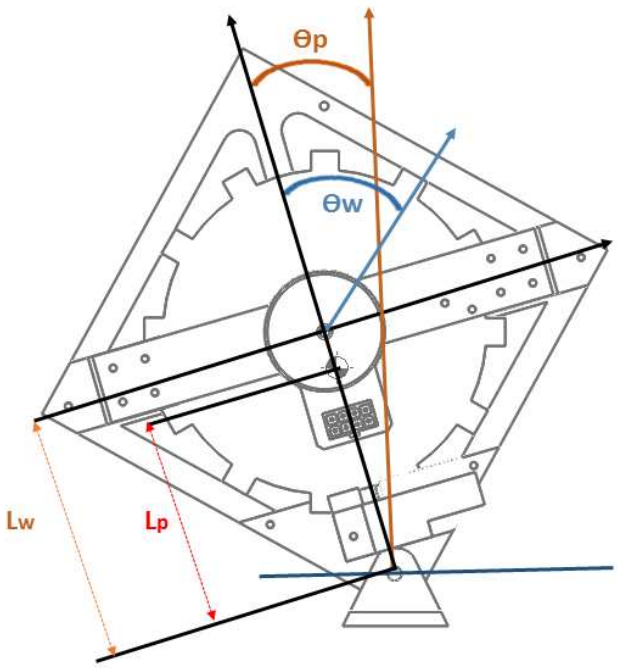


Figure 3.1: Notations for mechanical analysis

- L_w = distance between the pivot and the center of mass of the momentum wheel

3.1.1 Modeling using Vectorial Dynamics

The second Law of Newton for rotational motion, considering the forces presented in Figure 3.2 is:

$$\overrightarrow{M_F} + \overrightarrow{M_{G_w}} + \overrightarrow{M_{G_p}} = I\vec{\epsilon} \tag{3.1}$$

where M_F is the moment of force F and M_G corresponds to the moment due to gravity for pendulum and wheel, respectively. The acceleration was noted with ϵ, whereas I is the moment of inertia.

Each of the term can be written as:

$$\begin{aligned} \overrightarrow{M_F} &= \vec{R} \times \vec{F} \\ \overrightarrow{M_{G_w}} &= \overrightarrow{L_w} \times \overrightarrow{G_w} \\ \overrightarrow{M_{G_p}} &= \overrightarrow{L_p} \times \overrightarrow{G_p} \end{aligned} \tag{3.2}$$

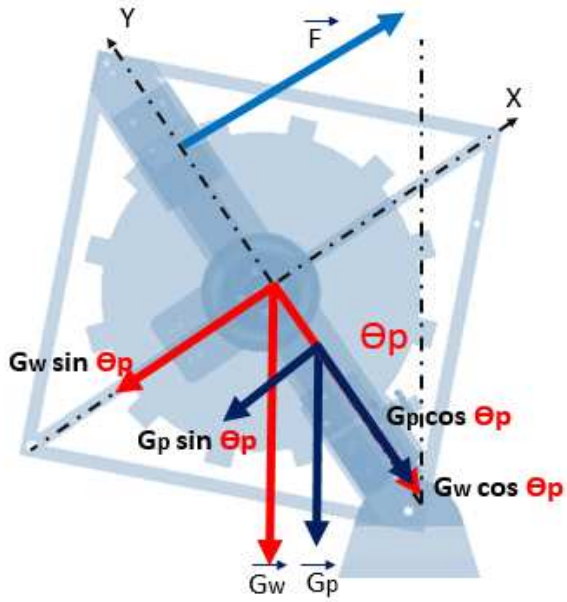


Figure 3.2: Forces on the cubic face

where R is the radius of the reaction wheel and the other factors were aforementioned. We can approximate that the momentum of force F is exactly the same as the torque generated by the DC motor in this system. With this approximation, the previous equation becomes

$$-T_m + G_w L_w \sin\theta_w + G_p L_p \sin\theta_w = \ddot{\theta}_p I \tag{3.3}$$

where $\ddot{\theta}_p = \epsilon$.

To obtain the equation of motion of the reaction wheel, we need to consider the relative acceleration equation:

$$\epsilon_{rel} = \epsilon_{absolute} + \epsilon_{translation} \tag{3.4}$$

In terms of moment of forces, we obtain:

$$\epsilon_{rel} = \frac{T_m - M_{fr_w}}{I_w} - \frac{M_{G_w} + M_{G_p} - T_m + M_{fr_w} - M_{fr_p}}{I_p + m_w L_w^2} \tag{3.5}$$

where M_{fr_w} is the moment of the friction force for the wheel and is equal to $C_w\dot{\theta}_w$ and M_{fr_p} is the moment of the friction force for the pendulum and is equal to $C_p\dot{\theta}_p$.

3.1.2 Modelling using Analytical Dynamics

To validate the equation 3.3, we used another approach of the classical dynamics: the analytical dynamics.

Let's assume $q_1, q_2, ...q_n$ generalized coordinates which represent n degrees of freedom for a system. The generalized coordinates can usually be angles or distances. Kinetic(T) and potential energies(V) can be calculated and then added together to form the total energy of the system. If we compute the difference between kinetic and potential energy, the Lagrangian is obtained: [18]

$$L = T - V \tag{3.6}$$

The equations of motion are then expressed in terms of the *Lagrangian*, as following:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_j}\right) - \frac{\partial L}{\partial q_j} = Q_j \tag{3.7}$$

where Q_j is the dissipative force. If the force is conservative, then

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_j}\right) - \frac{\partial L}{\partial q_j} = 0 \tag{3.8}$$

To start the computation, we consider the moment of inertia is additive and that generalized coordinates are θ_w and θ_p.

$$m_t = m_p + m_w \tag{3.9}$$

$$m_t L_t = m_p L_p + m_w L_w \tag{3.10}$$

$$I_{total} = I_p + I_w = m_p L_p^2 + m_w L_w^2 \tag{3.11}$$

The potential energy of the system (assume V = 0 at the Ground Level) and the kinetic energy are:

$$V = m_t g h = m_t g L_t \cos\theta_p = g(m_p L_p + m_w L_w)\cos\theta_p \tag{3.12}$$

$$T = \frac{1}{2} I_{total} \dot{\theta}_p^2 + \frac{1}{2} I_w \dot{\theta}_w^2 \tag{3.13}$$

Now, let's consider the relationships made in the first part of this section.

Thus, the Lagrangian for our system can be calculated as following:

$$L = T - V = \frac{1}{2} I_{total} \dot{\theta}_p^2 + \frac{1}{2} I_w \dot{\theta}_w^2 - m_t g L_t \cos\theta_p \tag{3.14}$$

For the first generalized coordinate θ_p, we will calculate the partial derivatives:

$$\frac{\partial L}{\partial \dot{\theta}_p} = I_{total} \dot{\theta}_p \tag{3.15}$$

$$\frac{d}{dt}\Big(\frac{\partial L}{\partial \dot{\theta}_p}\Big) = I_{total} \ddot{\theta}_p = (I_p + m_w L_w^2)\ddot{\theta}_p \tag{3.16}$$

In the previous equation, Steiner Theorema was used for the moment of inertia.

$$\frac{\partial L}{\partial \theta_p} = m_t g L_t \sin\theta_p \tag{3.17}$$

From the previous three equations and equation (3.7), we obtain

$$(I_p + m_w l_w^2)\ddot{\theta}_p - m_t g L_t \sin\theta_p = Q_j \tag{3.18}$$

We should take into consideration the Torque generated by the DC motor. Therefore, the previous equation becomes:

$$I_{total}\ddot{\theta}_p - m g L_t \sin\theta_p = T_m \tag{3.19}$$

Considering equation 1 from [19] and the previous equation, we can observe that the frictional forces are missing in my equation. By adding them, we obtain

$$\ddot{\theta}_p = \frac{m_t g L_t \sin\theta_p - C_p\dot{\theta}_p + C_w\dot{\theta}_w - T_m}{I_p + m_w L_w^2} \tag{3.20}$$

$$\ddot{\theta}_p = \frac{g(m_p L_p + m_w L_w)\sin\theta_p - C_p\dot{\theta}_p + C_w\dot{\theta}_w - T_m}{I_p + m_w L_w^2} \tag{3.21}$$

Considering 3.5, we can derive the equation of motion of the reaction wheel

$$\ddot{\theta}_\omega = \frac{(I_p + I_\omega + m_w L_w^2)(-C_w\dot{\theta})}{I_w(I_p + m_w L_w^2)} - \frac{g(m_p L_p + m_w L_w)\sin\theta_p - C_p\dot{\theta}_p}{I_p + m_w L_w^2} \tag{3.22}$$

Therefore, 3.21 and 3.22 are two coupled equations, that will be modelled accordingly in the previous sections.

3.2 Mechanical Design

The practical implementation of the reaction wheel was performed. For the mechanical design, the project Cubli was analyzed in detail and a model in Solidworks was developed. The main components for the one-dimensional prototype are:

- The Reaction Wheel
- The Braking System
  - Servomotor
  - Mounting for Servomotor
- The Cubic Face (the pendulum body)
- DC Motor
- Mounting for the DC Motor
- Flange to connect the DC Motor with the Reaction Wheel
- Mounting for the assembly

The first component to be designed was the reaction wheel profiled with teeth for the braking system. The reaction wheel was manufactured of textolite glass. Unfortunately, this material does not have a constant density distribution, so further adjustments for equilibrium were performed. Then, the pendulum body was designed. The dimensions were kept as in the [19] reference. The mounting for servomotor and for the DC Motor were 3D printed, whereas the flange was made of aluminum. The weight of the reaction wheel is two times the weight of the pendulum.

The Figure 3.3 shows an isometric view of the assembly. In Figure 3.4, the braking mechanism is presented. A servomotor is operated, in order to block the teeth of the reaction wheel.

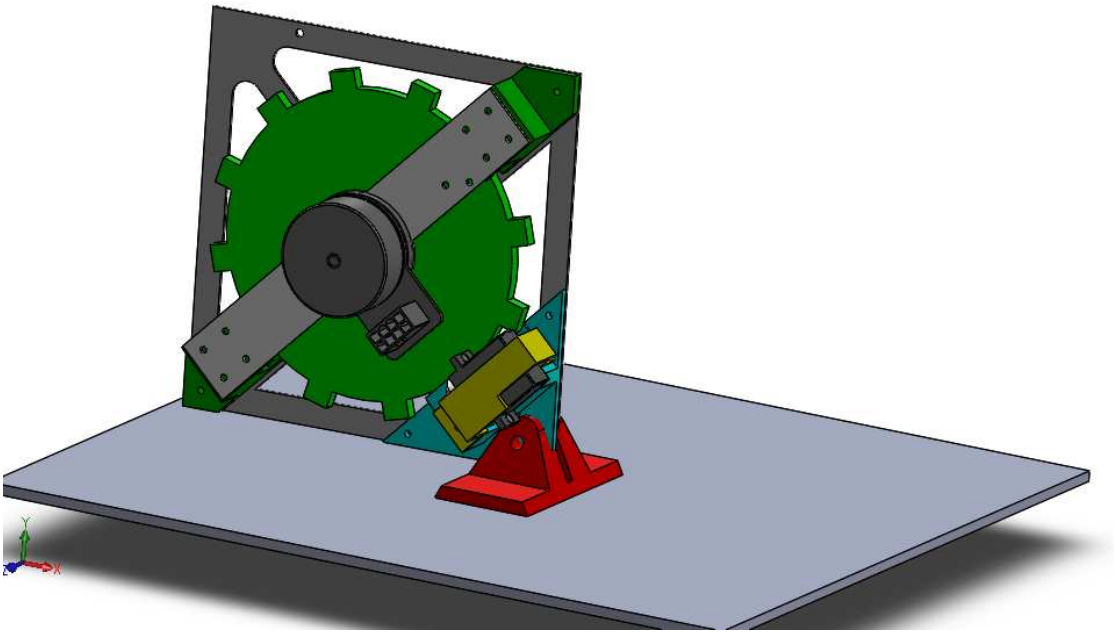


Figure 3.3: Isometric View of CUBalance

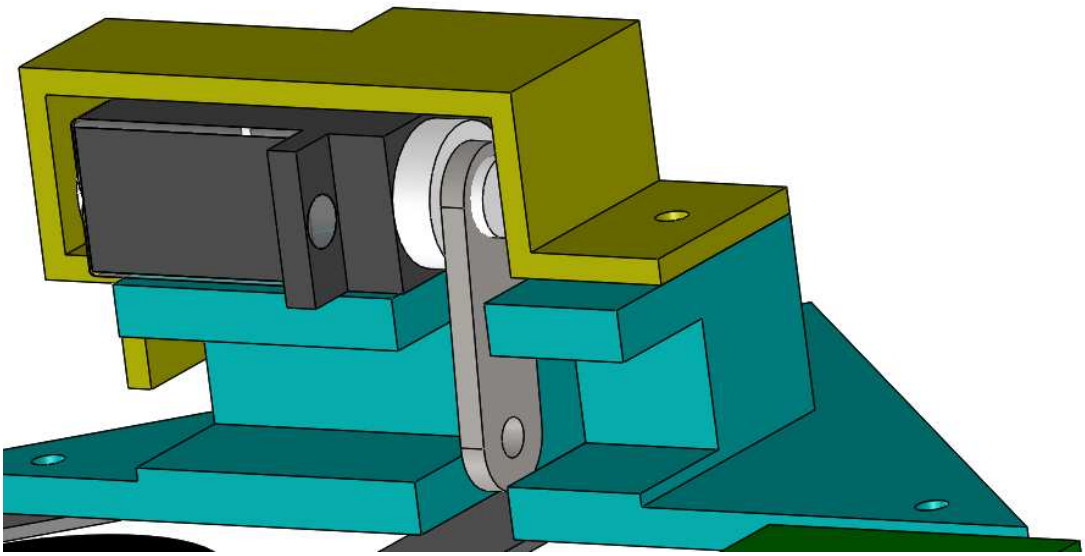


Figure 3.4: The braking system for CUBalance

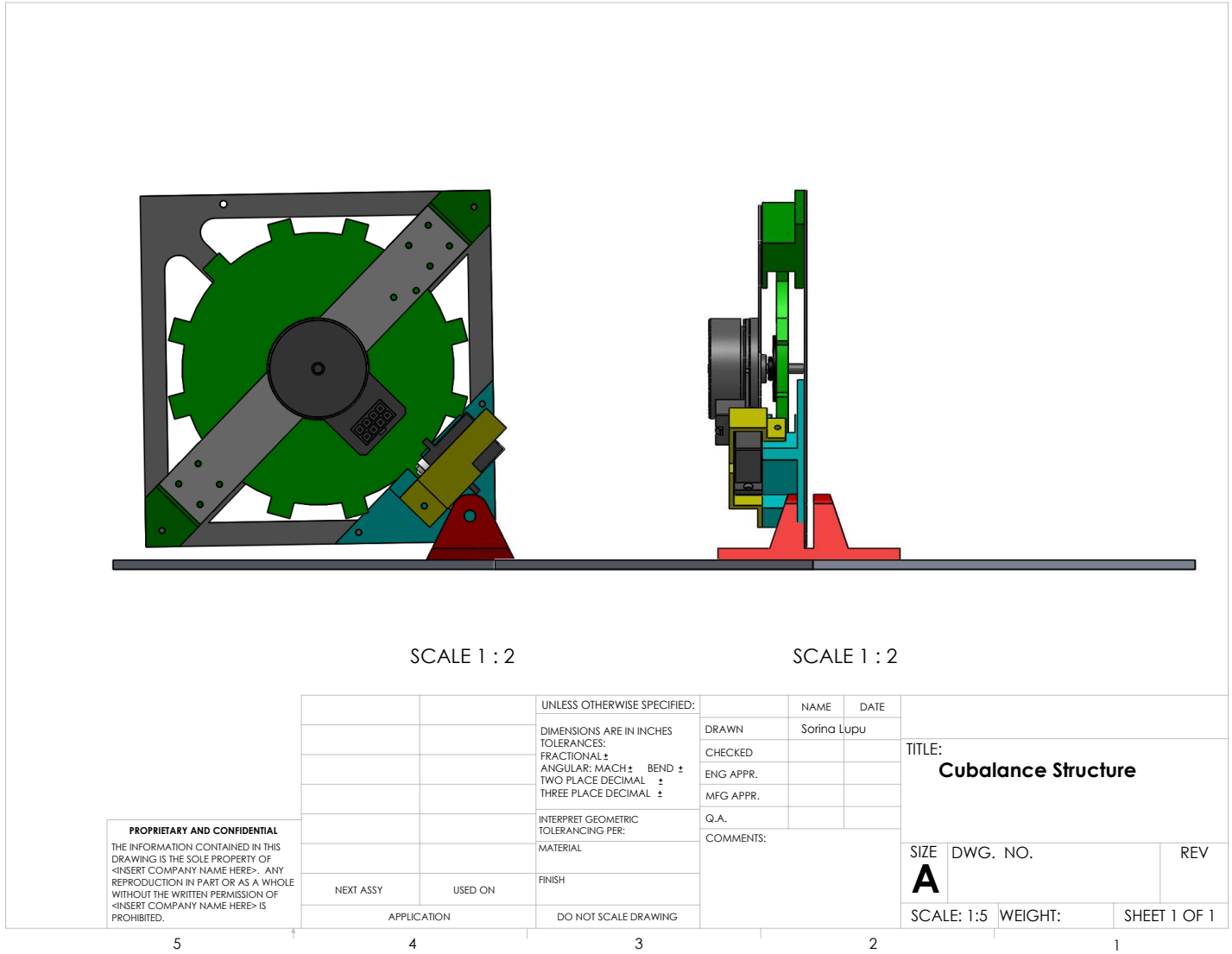


Figure 3.5: Mechanical Drawing of CUBALANCE

3.3 Electronics Design

The electronics for this project is one of the most important and critical aspects that must be taken into considerations. Not only the microcontroller should process the instructions at high frequency, but also the sensors should be accurate enough to provide correct data for the control algorithm.

3.3.1 Development Board

I had chosen the FDRM-KL25Z which is a low-cost development board made by Freescale with a 32-bit ARM Cortex-M0+ core running at 48MHz. It includes 128KB FLASH, 16KB RAM and many interfaces including SPI, I2C, ADC, DAC, PWM, Touch Sensor and other I/O interfaces. The code running on the microcontroller is written in C++, using the CodeWarrior Embedded Software with Processor Expert.

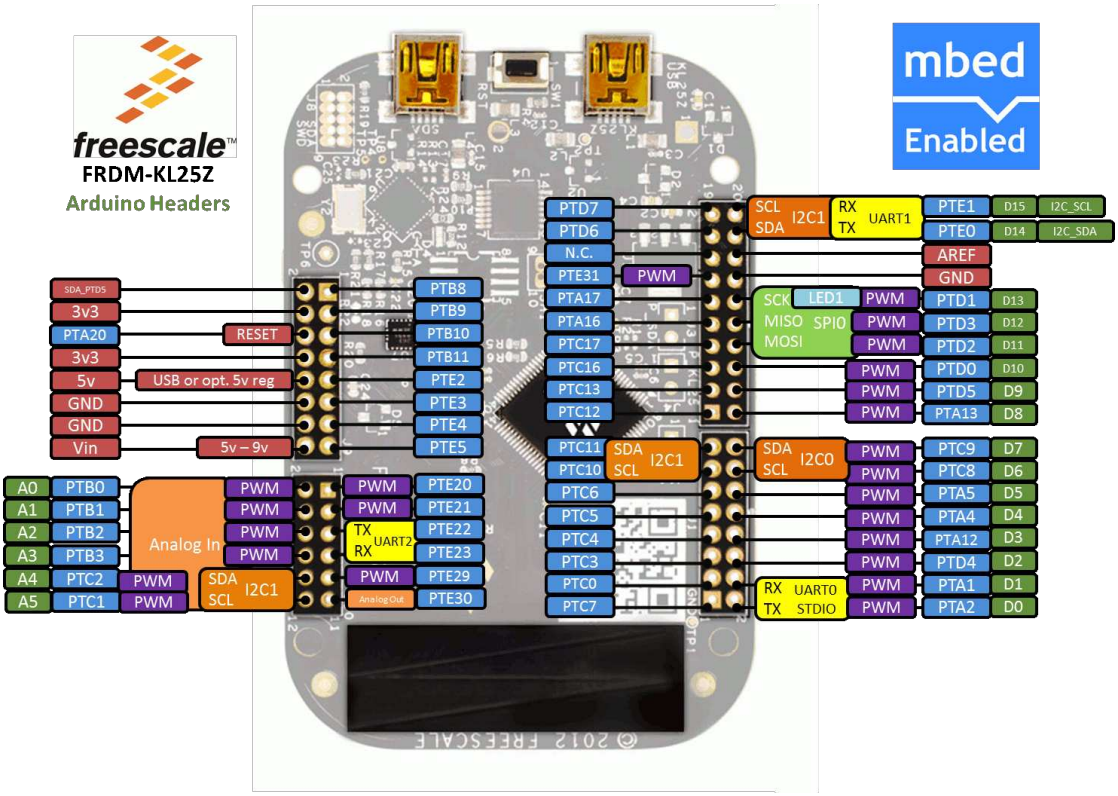


Figure 3.6: Features of the Development Board

3.3.2 Sensor

I had used an MPU-6050 Triple Axis Accelerometer and Gyroscope which features three 16-bit ADC convertors for the gyroscope outputs and three 16-bit ADCs for the acceleration outputs. The communication was done using I^2C.

Figure 3.7: MPU-6050 accelerometer and gyro

3.3.3 Motor and Motor Driver

Firstly, a normal DC motor was considered for prototyping, then, due to the space constraints, the solution of a flat motor was chosen.

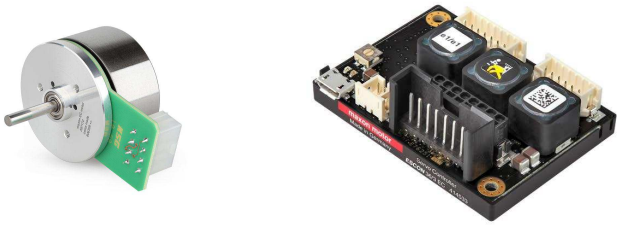


Figure 3.8: Motor and Motor Driver

The following requirements for the motor were taken into consideration, after calculations and simulations were performed:

1. *The motor should have a velocity of minimum 2000 rpm.*
2. *The motor should have a voltage of maximum 24V.*

3. *The weight of the motor should not exceed 100g.*
4. *The motor should contain Hall Sensors to determine the speed.*

The EC 45 flat (42.8 mm diameters, brushless, 50 Watt, with Hall sensors) from MAXON was chosen as the best option. As seen from the datasheet in the appendix, the motor has a velocity of 6710 rpm, a maximum voltage of 24V and a weight of 110g.

For this type of motor, the following driver was chosen: ESCON 36/3 EC, 4-Q Servocontroller for EC motors, 2.7/9 A, 10 - 36 VDC (Figure 3.8)

Figure 3.3.3 shows the interface of the sensors, the motor driver and the motor with the microcontroller.

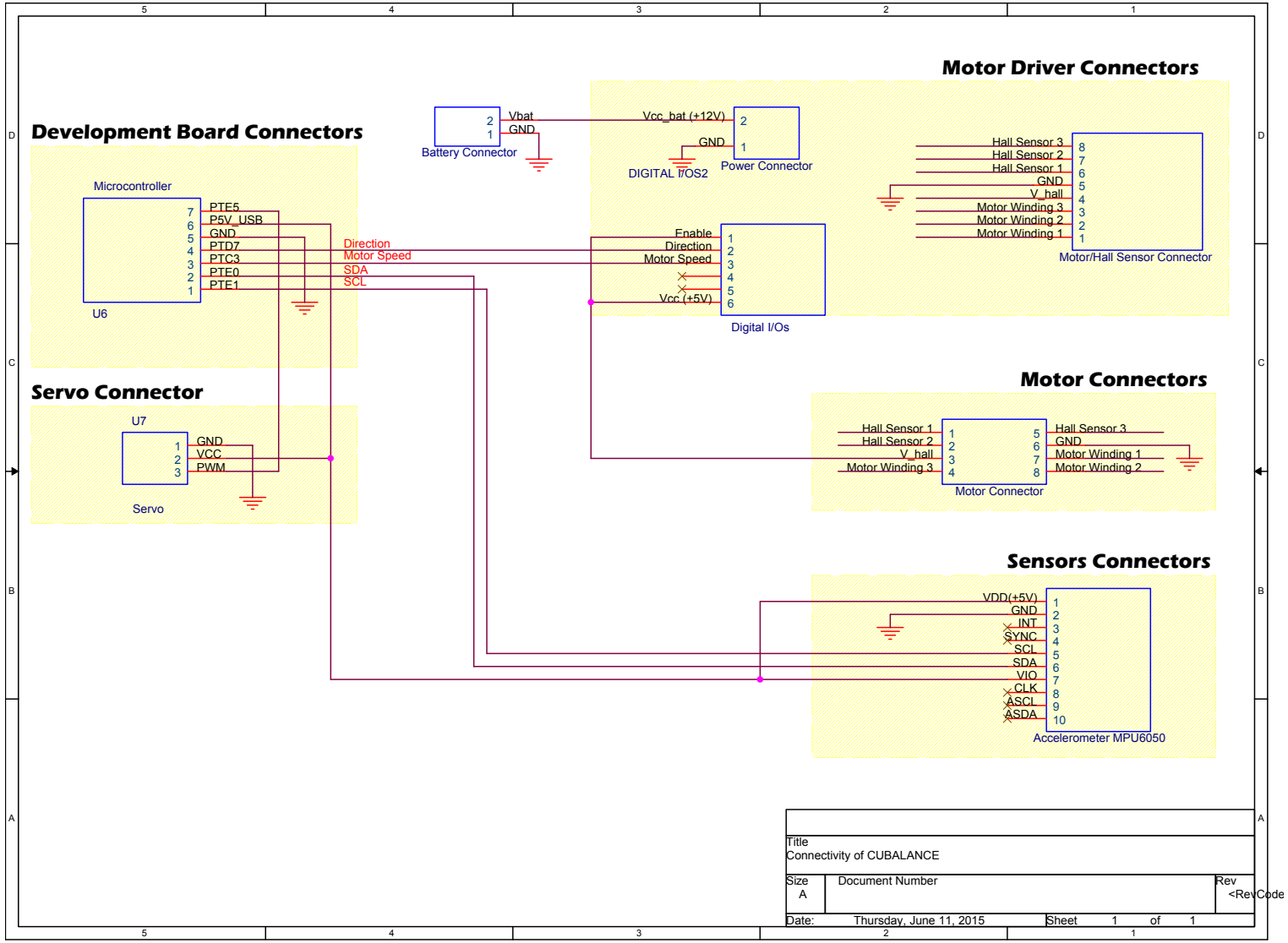


Figure 3.9: Connectivity of CUBALANCE

3.4 Software and Control

The software is divided into two parts:

- the algorithm running on the microcontroller which controls the system
- the Graphical User Interface (GUI) able to send and receive data in real time via UART.

3.4.1 Output Parameters and Control Algorithm

An overview of input/output parameters is presented in Figure 1.7.

3.4.1.1 Tilt Angle of Pendulum

Accelerometer sensor measures any difference between the linear accelerometer and the Earth gravity. In addition, the accelerometer can be used to determine the orientation of any object in space. [20] Using the following relationship [21], the tilt angle was obtained

$$\theta_y = \arctan \frac{A_x}{\sqrt{A_y^2 + A_z^2}} \tag{3.23}$$

The code written in C++ is presented below. The 57.295 term comes from the transformation between degrees and radians. Because the tilt angle read from the accelerometer was influenced by the vibrations of the system, a Complementary Filter was implemented which is a method to combine the accelerometer and gyroscopic signals into a stable angle estimate.

The formula of the Complementary Filter is:

$$angle_{filtered} = 0.98(angle_{filtered} + gyro_{data}T_s) + 0.02angle_{accelerometer} \tag{3.24}$$

The $angle_{filtered}$ was used as a reference in the PID and LQR Algorithms.

```
ACCEL_XANGLE = 57.295*atan((float)Acc_Y/ sqrt(pow((float)
    Acc_Z,2)+pow((float)Acc_X,2)));
ACCEL_YANGLE = 57.295*atan((float)-Acc_X/ sqrt(pow((float
    )Acc_Z,2)+pow((float)Acc_Y,2)));
```

The control algorithm for the PID is presented in Figure 3.10, whereas the control algorithm for the LQR Algorithm is outlines in Figure 3.11.

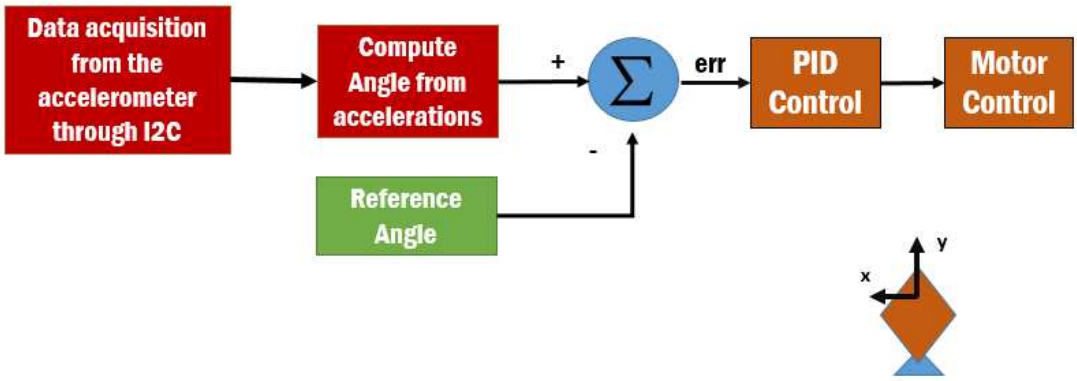


Figure 3.10: Control Algorithm(PID)

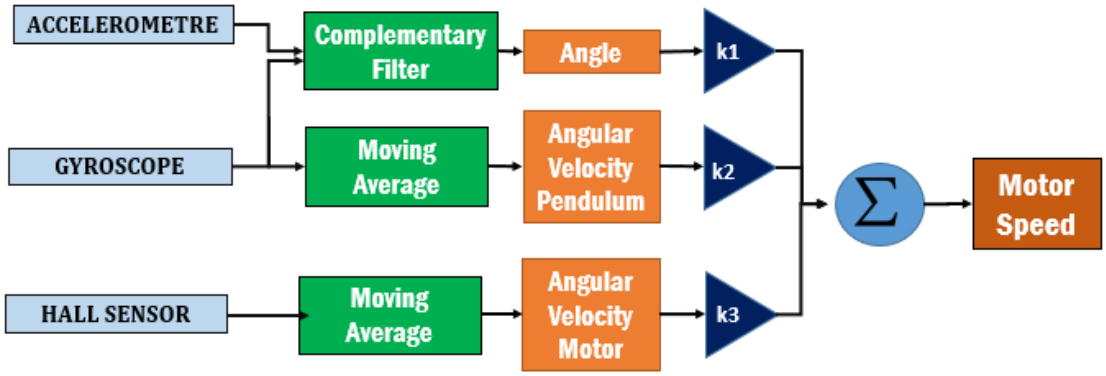


Figure 3.11: Control Algorithm (LQR)

3.4.1.2 Angular Velocity of the Pendulum

A gyroscope was attached to the pendulum body to measure the angular velocity and to provide accuracy in the tilt angle. By detecting small changes from the equilibrum position, this quantity can be used in the Linear Quadratic Regulator algorithm to stabilize the system.

The code for gyro readings is presented below

```
int32_t GYRO_XOUT_OFFSET = -448;
int32_t GYRO_YOUT_OFFSET = 64;
int32_t GYRO_ZOUT_OFFSET = -192;

volatile uint32_t i = 0;

float Get_Gyro_Rates()
{

        int16_t Gyro_X;
        int16_t Gyro_Y;
        int16_t Gyro_Z;


        Gyro_X = MPU6050_Get_Data (MPU6050_RA_GYRO_XOUT_H
           , MPU6050_RA_GYRO_XOUT_L);
        Gyro_Y = MPU6050_Get_Data (MPU6050_RA_GYRO_YOUT_H
           , MPU6050_RA_GYRO_YOUT_L);
        Gyro_Z = MPU6050_Get_Data (MPU6050_RA_GYRO_ZOUT_H
           , MPU6050_RA_GYRO_ZOUT_L);

        gui_gyro_x = (Gyro_X - GYRO_XOUT_OFFSET) /
           GYRO_SENSITIVITY;
        gui_gyro_y = (Gyro_Y - GYRO_YOUT_OFFSET) /
           GYRO_SENSITIVITY;
        gui_gyro_z = (Gyro_Z - GYRO_ZOUT_OFFSET) /
           GYRO_SENSITIVITY;

        return gui_gyro_z;
}
```

3.4.1.3 Angular Velocity of the Reaction Wheel

This type of motor has three Hall Sensors to detect the angle of the rotor, and further to determine its position and its velocity. This motor speed is read from an analog pin on the motor driver which is connected to an ADC pin on the Freescale development board.

3.4.2 Graphical User Interface

In any system, it is necessary to be able to obtain and plot real-time data from the sensors. Therefore, a GUI was built to solve this need, and also to configure the parameters of the PID and LQR in real time. I had chosen FreeMASTER, developed by Freescale. "FreeMASTER is a user-friendly real-time debug monitor and data visualization tool that you can use for any application development and information management. FreeMASTER supports both cooperative and non-intrusive monitoring of variables on a running system. You can display multiple variables changing over time on an oscilloscope-like display, or view the data in text form. As well, FreeMASTER supports additional capabilities and targets with an on-target driver for transmitting data from the target to the host computer."[22]

Name	Value	Unit
gui_acc_y	-0.1875	unit
gui_acc_z	-0.25	unit
gui_total_acc	1.00936	unit
gui_angle	-11.877	unit
KP	100	unit
KD	150	DEC
KI	0	DEC
gui_pid_output	1141.14	unit
gui_gyro_x	0	unit
gui_gyro_y	1.9542	unit
gui_gyro_z	0	unit
gui_k_angle	0	unit
gui_computed_angle	-11.894	unit
gui_servo_position	1	DEC
gui_motor_speed	0	DEC
gui_acc_x	-0.953125	unit

Figure 3.12: Parameters read from the system

In Figure 3.4.2, the parameters logged are presented and are explained in the previous tables.

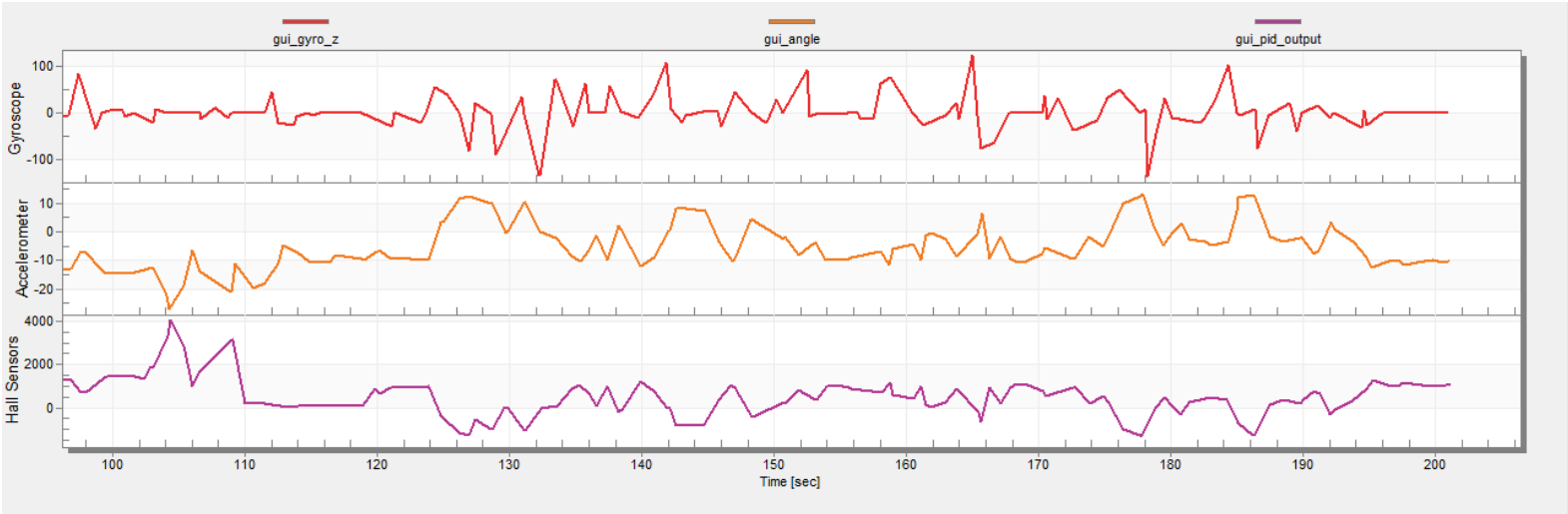


Figure 3.13: Data Displayed on the GUI

Table 3.1: Explanation of OUTPUT Parameters from the GUI

Parameter	Importance
gui_acc_x	Acceleration on the X axis
gui_acc_y	Acceleration on the Y axis
gui_acc_z	Acceleration on the Z axis
gui_gyro_x	Rate of rotation on the X axis
gui_gyro_y	Rate of rotation on the Y axis
gui_gyro_z	Rate of rotation on the Z axis
gui_computer_angle	Angle of the System relative to the ground (In equilibrium, 0deg is expected)
gui_servo_position	Position of the servomotor for the braking maneuver
gui_motor_speed	Motor speed as read from the Hall Sensors
gui_pid_output	Speed of the motor as calculated using the PID Algorithm

Table 3.2: Explanation of INPUT Parameters from the GUI

Parameter	Importance
KP	Proportional coefficient for the PID
KD	Differential coefficient for the PID
KI	Integral coefficient for the PID
gui_k_angle	Coefficient to link the acceleration with the rate of rotation (default is 0)

4. Results

4.1 Practical Approach

In order to validate the mathematical modelling, the reaction wheel inverted pendulum was physically built.

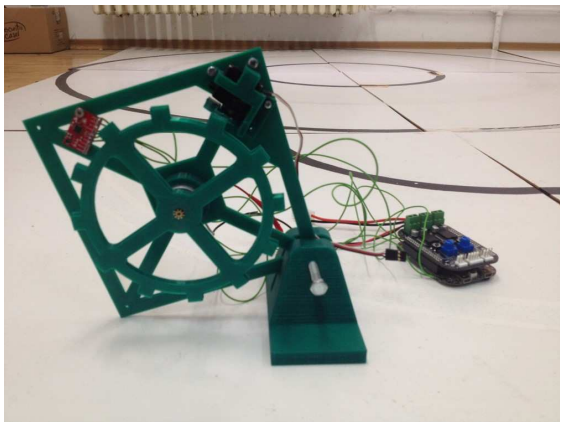


Figure 4.1: First iteration of the Cubalance

Figure 4.1 shows the first iteration of the one-dimensional structure, built with 3D printed components. A DC motor was used to control the reaction wheel. The main disadvantage of this structure was its inaccuracy. In addition, the DC Motor was not providing enough torque to give stability to the system (low switching between direction of rotation). However, this prototype was useful in terms of understanding how the system works and for sensor calibration.

The second iteration, as presented in Figure 4.2, was completely different from the previous one. The reaction wheel and the cubic face were made of textolite glass, not 3D printed, in order to increase the accuracy. In addition, the DC motor was replaced with a more powerful one (a flat DC motor with a maximum velocity of 6000 rpm). The mounting of the *DC Motor - reaction wheel - cubic face* was done through a flange and a ball bearing.

The third iteration and the final one is presented in Figure 4.3. Here, the ball bearing was eliminated and the flange was reduced in side. The cubic face was made of Aluminum with a thickness of 2 mm.

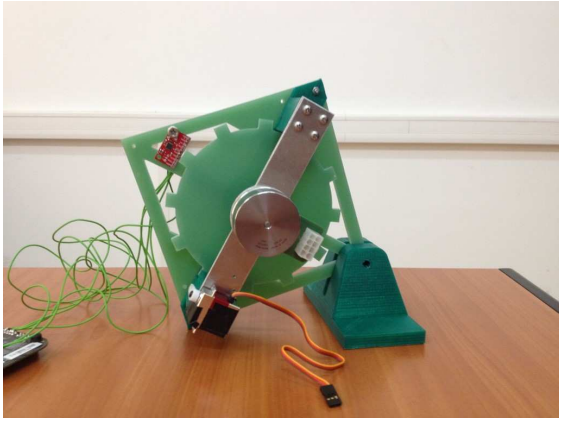


Figure 4.2: Second iteration of the Cubalance

The following important parameters (Table 4.1) were obtained for the one-dimensional structure. When referred to analytic, either Solidworks computation or Matlab calculations were performed. Experimental means that the subject was measured.

Table 4.1: Parameters of Cubalance

Parameters	Determination	Values
Reaction wheel mass (m_w)	Analytic and Experimental	0.115 kg
Pendulum mass (m_p)	Analytic and Experimental	0.058 kg
Distance between the CoG of the pendulum and the pivot (l_p)	Analytic	0.077 m
Distance between the CoG of the wheel and the pivot (l_w)	Analytic and Experimental	0.101m
Moment of Inertia of the Wheel (I_w)	Analytic	0.287e-3 kgm^2
Moment of Inertia of the Pendulum (I_b)	Analytic	0.6e-3 kgm^2
Torque constant (k_m)	From datasheet	33.5 mNm/A

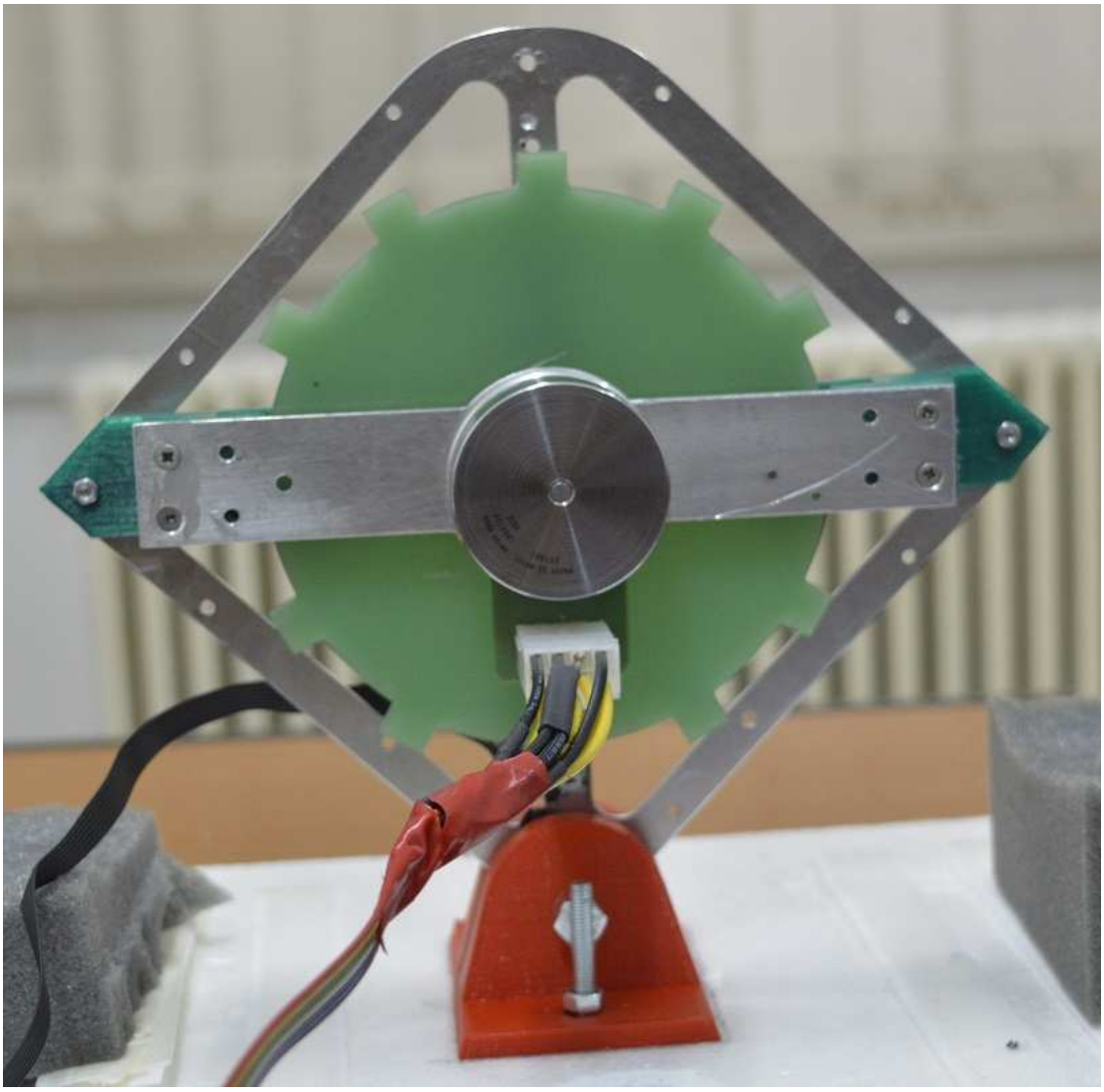


Figure 4.3: Final iteration of the Cubalance

4.2 Theoretical Validation. Equations of Motion

With the parameters from Table 4.1 and the equations from Chapter section 3.1, the results of the theoretical validation are presented in the next pages.

4.2.1 System without any applied torque

Firstly, I will consider the system of equations derived in section 3.1 without any applied torque. The system now becomes:

$$\ddot{\theta}_p = \frac{g(m_p L_p + m_w L_w)\sin\theta_p - C_p\dot{\theta}_p}{I_p + m_w L_w^2} \tag{4.1}$$

$$\ddot{\theta}_\omega = 0 \tag{4.2}$$

The initial conditions considered for this system were: $\theta_w(0) = 0$, $\dot{\theta}_w(0) = 0$, $\theta_p(0) = \frac{\pi}{10}$ and $\dot{\theta}_p(0) = 0$.

The non-linear differential system was simulated using Matlab/Simulink. The results are in agreement with the model (an harmonic oscillation of a simple pendulum) as seen in Figure 4.4. The attenuation observed on the plot is caused by the friction coefficients.

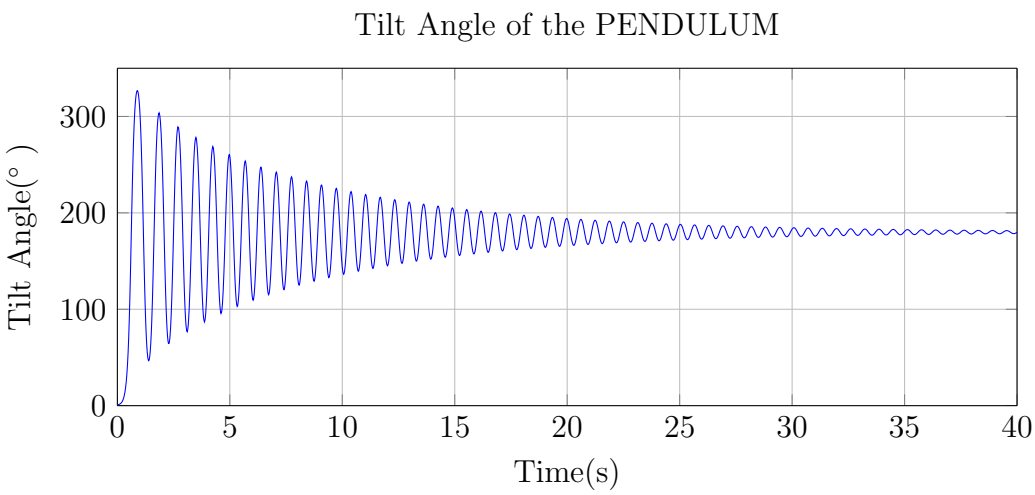


Figure 4.4: Simple Pendulum Behaviour

4.2.2 System with applied torque

My goal is to design and implement a controller to make the system achieve the desired state: stabilization when disturbance is applied. I have develop two approaches to solve this issue: **a PID Control** and a **Linear Quadratic Regulator**.

This torque is provided by the motor attached to the system and is proportional

to the current, according to the next formula:

$$T_m = k_m i \tag{4.3}$$

where k_m is the Torque Constant and its value is 33.5 mNm/A. The system now becomes:

$$\ddot{\theta}_p = \frac{g(m_p L_p + m_w L_w)\sin\theta_p - C_p\dot{\theta}_p + C_w\dot{\theta}_w - T_m}{I_p + m_w L_w^2} \tag{4.4}$$

$$\ddot{\theta}_\omega = \frac{(I_p + I_\omega + m_w L_w^2)(T_m - C_w\dot{\theta}_w)}{I_w(I_p + m_w L_w^2)} - \frac{g(m_p L_p + m_w L_w)\sin\theta_p - C_p\dot{\theta}_p}{I_p + m_w L_w^2} \tag{4.5}$$

The initial conditions for this system were considered: $\theta_w(0) = 0$, $\dot{\theta}_w(0) = 0$, $\theta_p(0) = \frac{\pi}{10}$ and $\dot{\theta}_p(0) = 0$. The system was build in Simulink (Figure 4.5) and a P.I.D. control was applied using the coefficients from 4.2.

Table 4.2: P.I.D Values

P	400
I	345
D	232

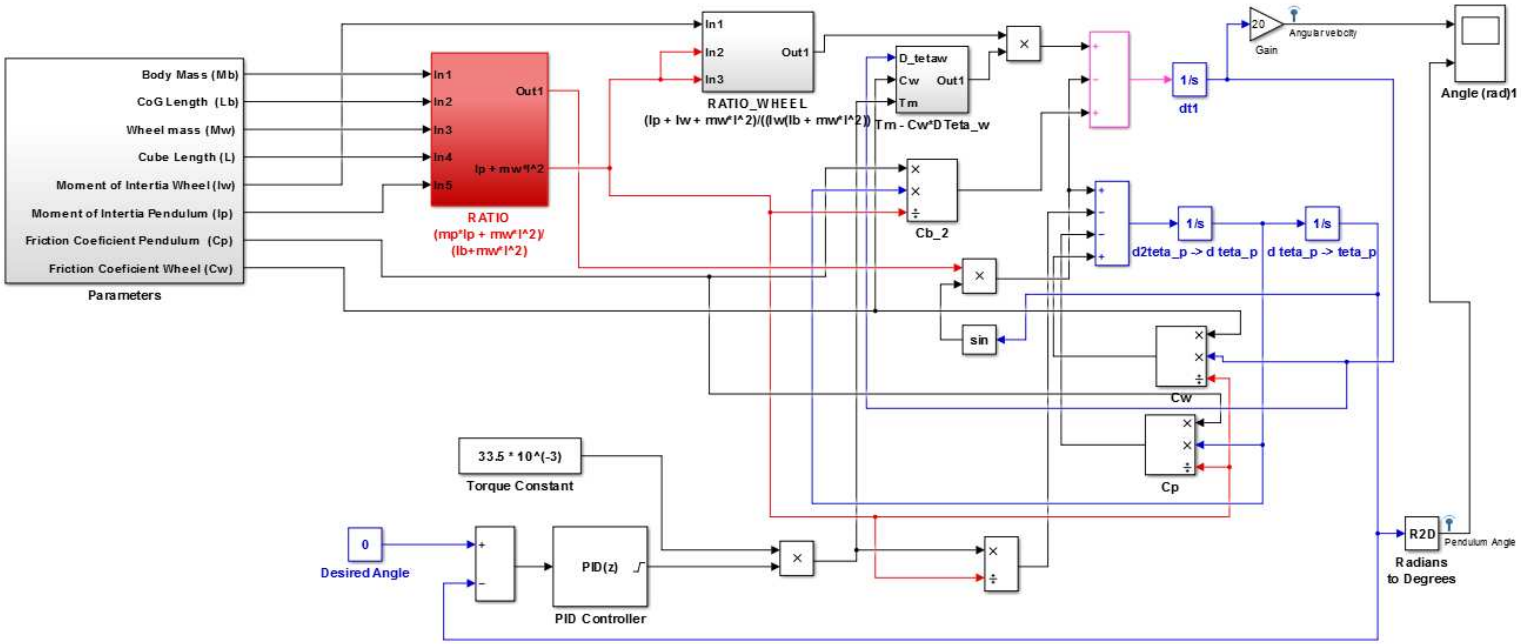


Figure 4.5: Simulink Block for equations and P.I.D modelling

The monitored signals from the previous Simulink block are the *tilt angle* with respect to the surface normal and the *angular velocity* of the motor. It can be observed (Figure 4.6) that the system is brought to an equilibrium state in 1.5 seconds for an initial condition of $\frac{\pi}{10}$ for the tilt angle. The motor response is presented in Figure 4.7.

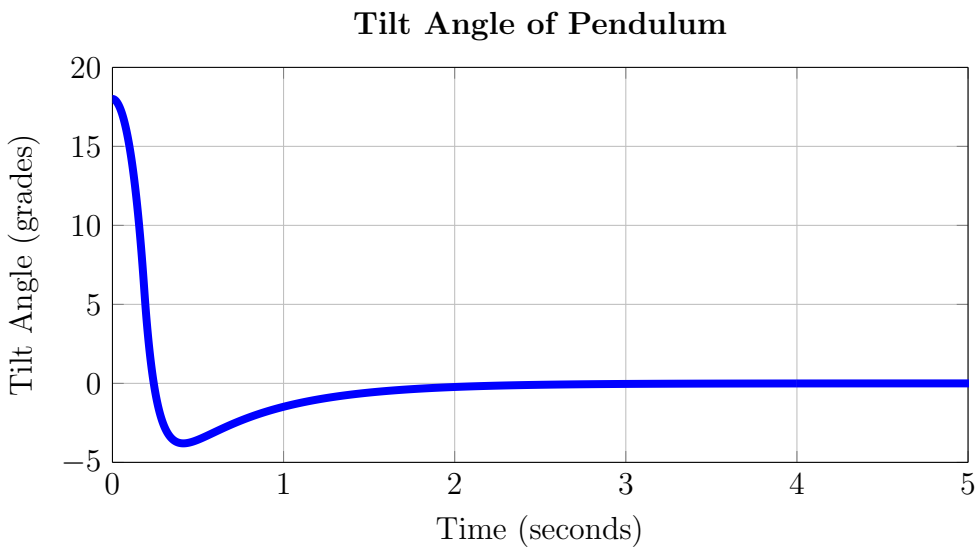


Figure 4.6: Tilt Angle when PID control is applied

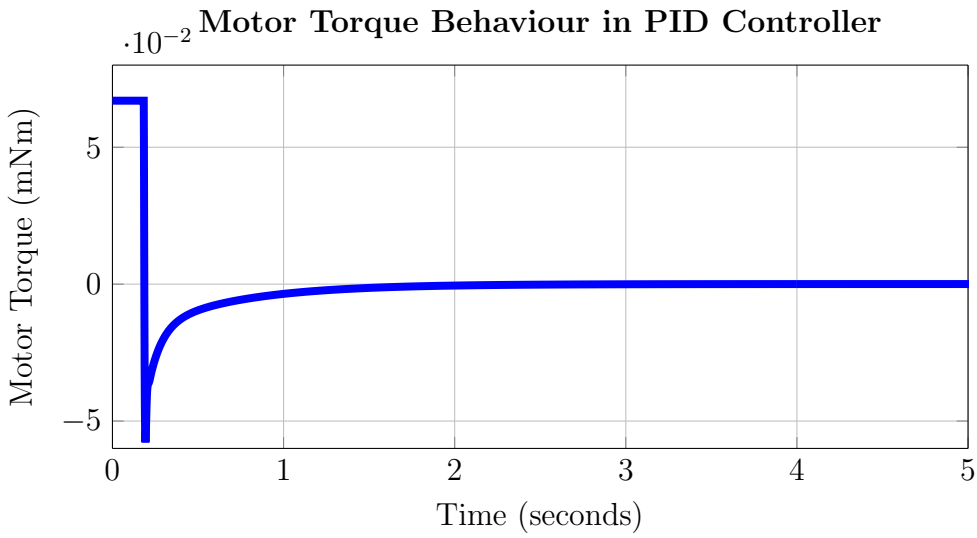


Figure 4.7: Motor Torque when PID Control is applied

4.2.3 State Space Representation

Another approach to valide the system was the State Space Representation, and it was implemented in order to develop the Linear Quadratic Regulator.

The state space model[2] is the representation of the dynamics of the Nth order system as a first order differential equation in the N-vector, called the state.

$$\begin{aligned} \dot{x}(t) &= A(t)x(t) + B(t)u(t) \\ y(t) &= C(t)x(t) + D(t)u(t) \end{aligned} \tag{4.6}$$

Many systems can however be modelled as a Linear Time Invariant State Dynamics.

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) + Du(t) \end{aligned} \tag{4.7}$$

Where $x(t) \in \mathbb{R}$ is the state vector, $u(t)$ is the input/control and $y(t)$ is the output. A is the dynamics matrix $\in \mathbb{R}^{n\times n}$, B is the input matrix $\in \mathbb{R}^{n\times m}$, C is the output or sensor matrix $\in \mathbb{R}^{p\times n}$ and D is the feedthrough matrix $\in \mathbb{R}^{p\times m}$. $x(t)$ is called "state" because future outputs depend only on the future input and current state. Also, it can be considered as a "memory of the system".

To obtain the space state representation, the following approximation (liniarization) for small angles was considered:

$$\sin(x) \approx x \tag{4.8}$$

In addition, the matrix A,B and C were obtained and calculated from the equations of motion with the parameters from Table 4.1:

$$A = \begin{bmatrix} 0 & 1 & 0 \\ \frac{g(m_pL_p+m_wL_w)}{I_p+m_wL_w^2} & -\frac{C_p}{I_p+m_wL_w^2} & \frac{C_w}{I_p+m_wL_w^2} \\ -\frac{g(m_pL_p+m_wL_w)}{I_p+m_wL_w^2} & \frac{C_p}{I_p+m_wL_w^2} & -\frac{C_w(I_w+I_p+m_wL_w^2)}{I_w(I_p+m_wL_w^2)} \end{bmatrix} \tag{4.9}$$

$$B = \begin{bmatrix} 0 \\ -\frac{k_m}{I_p+m_wL_w^2} \\ -\frac{k_m(I_w+I_p+m_wL_w^2)}{I_w(I_p+m_wL_w^2)} \end{bmatrix} \tag{4.10}$$

$$C = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} \tag{4.11}$$

Numerically, A,B and C are:

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 95.67 & -0.54 & 0.02 \\ -95.67 & 0.54 & 0.20 \end{bmatrix} \tag{4.12}$$

$$B = \begin{bmatrix} 0 \\ 17.93 \\ 137.57 \end{bmatrix} \tag{4.13}$$

$$C = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} \tag{4.14}$$

The states of the system are $((\theta_p, \dot{\theta}_p, \dot{\theta}_w))$ which mean (the tilt angle, the angular velocity of the pendulum and the angular velocity of the motor). The input $u(t)$ is used to control the motor. For the sake of simplification, we will consider D = 0_3. By applying Laplace transformation to the previous equations, we obtain:

$$\begin{aligned} sX(s) &= AX(s) + BU(s) \\ Y(s) &= CX(s) \end{aligned} \tag{4.15}$$

$$\begin{aligned} (sI_3 - A)X(s) &= BU(s) \\ Y(s) &= CX(s) \end{aligned} \tag{4.16}$$

By combining the previous two equations, we get

$$Y(s) = (sI_3 - A)^{-1}BU(s) \tag{4.17}$$

So, the transfer function is

$$G_{pl}(s) = \frac{Y(s)}{U(s)} = C(sI_3 - A)^{-1}B \tag{4.18}$$

A Matlab code was written in order to test the stability of the system using Plot-Zero Map (as presented in Figure 2.3).

```
sys_ss = ss(A, B, C, D);
figure
isstable(sys_ss)
[NUM, DEN] = ss2tf(A, B, C, D);
sys = tf(NUM, DEN)
isstable(sys)
figure; pzplot(sys)
```

In Figure 4.8, it can be observed that the pendulum is not stable without any applied control (there are poles in the right-half region).

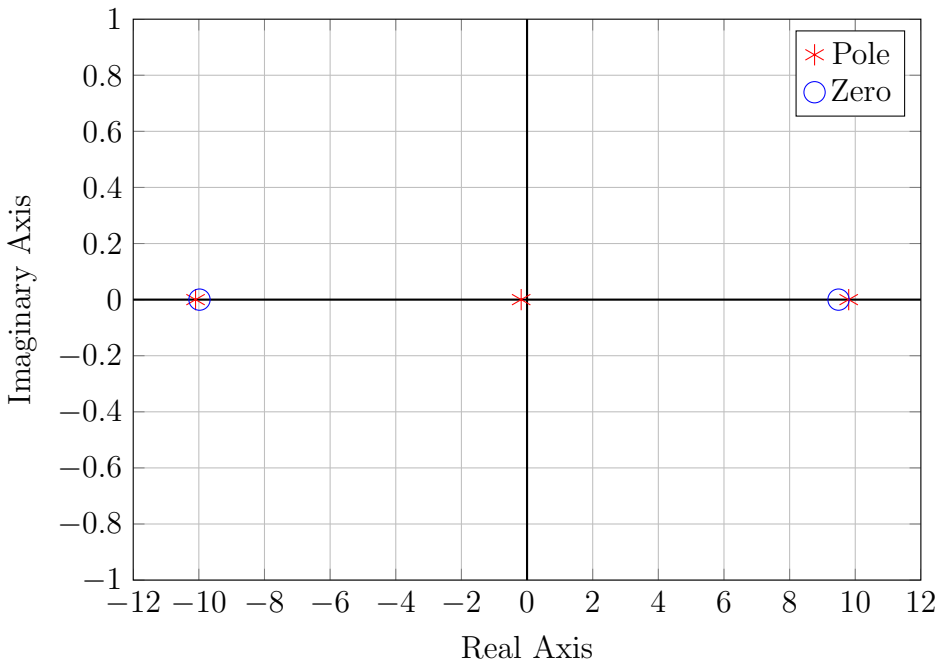


Figure 4.8: Pole-Zero Map when no control is applied

4.2.4 PID Control

To try to stabilize the system, I have tried in Matlab at first the PID Control with both automatic tuning of parameters and iterative tuning, as presented in the following Matlab code.

```
%% PID control with automatic tuning of parameters

control = pidtune(sys, 'PID')
set(gco,'LineWidth',6)
sys3 = feedback(control * sys,1)
isstable(sys3)
zpk(sys3)
figure; pzplot(control*sys)
figure, rlocus(control*sys)
zpk(control*sys);

```

```
%% PID control with iterative tuning

s = tf('s');
Kp = -400;
Kd = -250;
Ki = 0;
G_c = Kp + Ki / s + Kd * s;
sys4 = sys * G_c
    for Kp = (-1000):1:1000
        for Kd = (-1000):1:1000
            if isstable(sys4) == 1
                zpk(sys4)
                break;
            else
                isstable(sys4)
                disp(Kp)
                G_c = Kp + Kd * s;
                sys4 = sys * G_c;
            end
        end
    end

S = stepinfo(sys)
```

However, it can be seen from Figure 4.9 that in both cases, the system is not stable.

4.2.5 LQR Control

In the case of Linear Quadratic Regulator (Figure 4.10 and Figure 4.11) it is proven that the system is stable.

```
%% PID control with automatic tuning of parameters

control = pidtune(sys, 'PID')
set(gco,'LineWidth',6)
sys3 = feedback(control * sys,1)
isstable(sys3)
zpk(sys3)
figure; pzplot(control*sys)
figure, rlocus(control*sys)
zpk(control*sys);

```

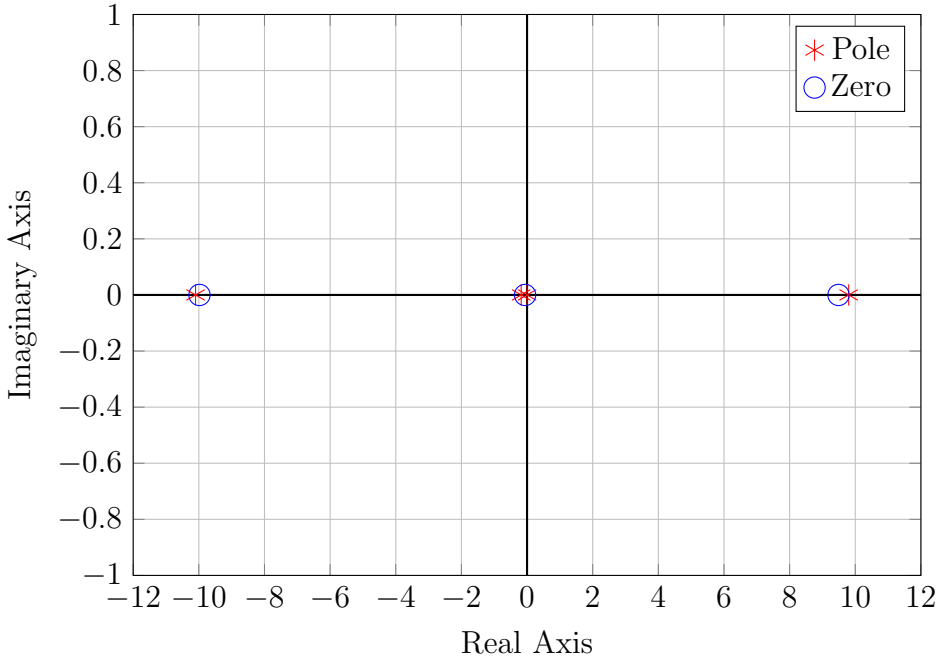


Figure 4.9: Pole-Zero Map when PID Control is applied

```
%% PID control with iterative tuning

s = tf('s');
Kp = -400;
Kd = -250;
Ki = 0;
G_c = Kp + Ki / s + Kd * s;
sys4 = sys * G_c
    for Kp = (-1000):1:1000
        for Kd = (-1000):1:1000
            if isstable(sys4) == 1
                zpk(sys4)
                break;
            else
                isstable(sys4)
                disp(Kp)
                G_c = Kp + Kd * s;
                sys4 = sys * G_c;
            end
        end
    end

```

```
S = stepinfo(sys)
```

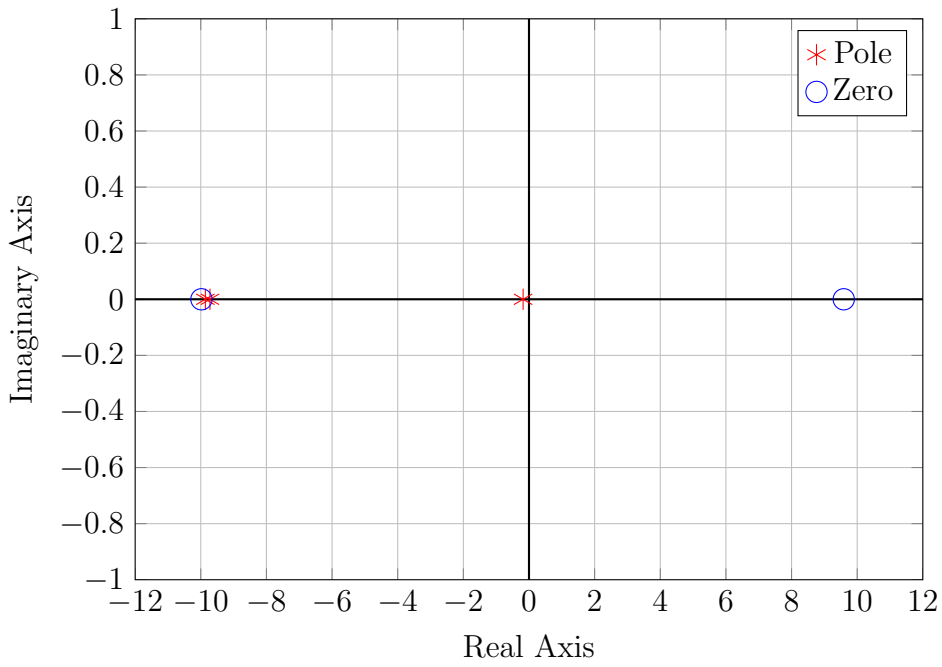


Figure 4.10: Pole-Zero Map when Linear Quadratic Regulator is applied on the system with matrix defined from equation of motion

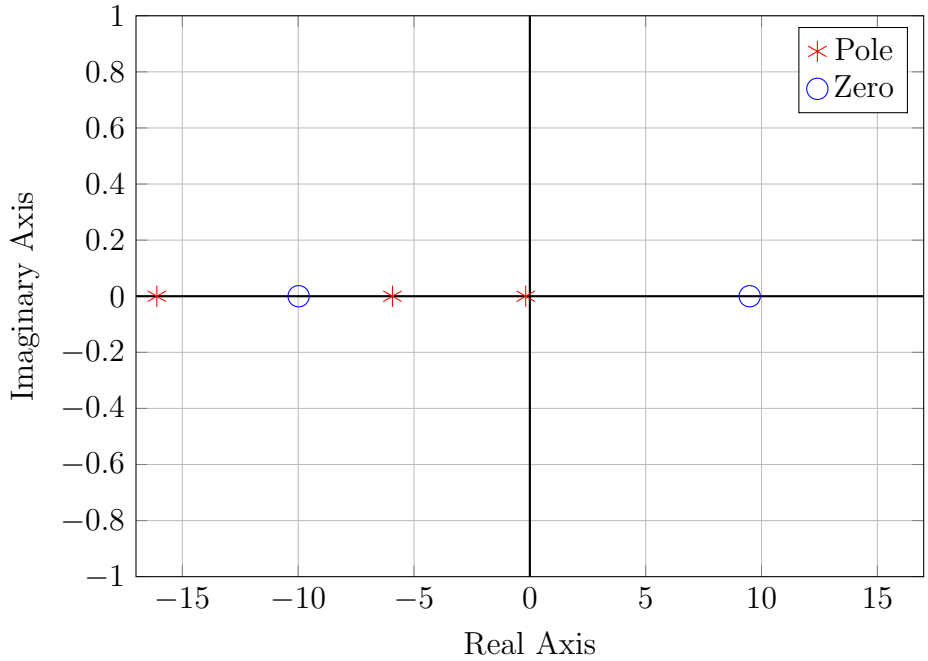


Figure 4.11: Pole-Zero Map when Linear Quadratic Regulator is applied on the system with state space matrix obtained from the transfer function

5. Conclusions and Overview

My interest in space technology had driven me to choose this topic as part of my bachelor degree in electronics engineering. Nowadays, more and more private companies are interested in sending satellites into space. Moreover, CubeSat startkits can be easily bought for a price of 55,000 EUR. In this framework, the attitude control of a satellite is a crucial aspect for the success of space mission. In the future, the near-Earth space will become more and more crowded, and satellite collisions are not desired. There are several types of satellite control used in space technology, but their complexity increases with the complexity of the mission. In this thesis, I focused on the Proportional-Integral-Derivative Control and on the Linear Quadratic Regulator in order to stabilize a Reaction Wheel Inverted Pendulum.

The main steps performed in this direction were:

1. Design of the assembly in solidworks which led to the construction of three iterations of the system (See Chapter 4, section 4.1)

2. Built the one dimensional prototype

3. Investigate the behaviour of the PID and LQR controllers in Simulink/Matlab

4. Program the microcontroller with the PID algorithm and the LQR algorithm. The following code is open-source and can be accessed from my GITHUB repository.

5. Validate the system, such as the pendulum is able to restore to the initial position at 0° after a shift of 13°.

6. Implement the Jump-Up without mechanical brake from a tilt angle of 45°.

In terms of control, even if from a theoretical point of view, the P.I.D control algorithm might be enough to obtain stabilization of the system in certain initial conditions, in practice the system is difficult to control. Therefore, LQR is a reliable solution.

The three dimensional implementation of the system is expected to be finished for the Master Thesis.

Appendices

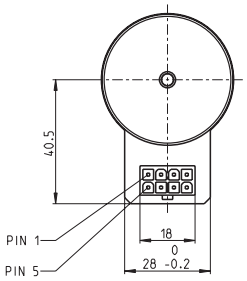
A. Relevant Datasheets

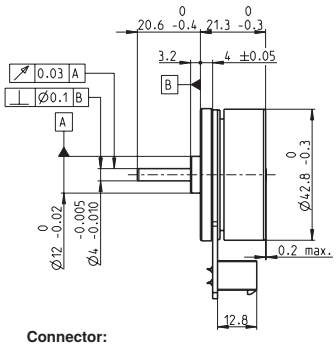
Some relevant aspects from the datasheet of the components are present in the following pages

- MOTOR: *EC 45 flat diameters 42.8 mm, brushless, 50 Watt, with Hall sensors*
- MOTOR DRIVER: *ESCON 36/3 EC, 4-Q Servocontroller for EC motors, 2.7/9 A, 10 - 36 VDC*

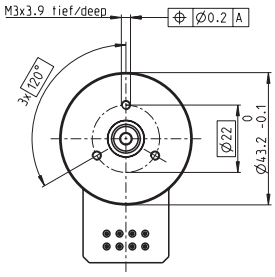
maxon f at motor

EC 45 f at ∅42.8 mm, brushless, 50 Watt





Connector:
39-28-1083 Molex



M 1:2

Stock program
Standard program
Special program (on request)

Part Numbers

	with Hall sensors	339285	251601	339286	339287
Motor Data					
Values at nominal voltage					
1 Nominal voltage	V	18	24	24	36
2 No load speed	rpm	6720	6710	4730	3360
3 No load current	mA	247	185	106	42.3
4 Nominal speed	rpm	5190	5240	3480	2360
5 Nominal torque (max. continuous torque)	mNm	97.1	83.4	69.6	90.5
6 Nominal current (max. continuous current)	A	3.52	2.33	1.41	0.828
7 Stall torque	mNm	975	780	402	484
8 Stall current	A	38.8	23.3	8.47	4.81
9 Max. eff ciency	%	85	83	79	82
Characteristics					
10 Terminal resistance phase to phase	Ω	0.464	1.03	2.83	7.48
11 Terminal inductance phase to phase	mH	0.322	0.572	1.15	5.15
12 Torque constant	mNm/A	25.1	33.5	47.5	101
13 Speed constant	rpm/V	380	285	201	95
14 Speed/torque gradient	rpm/mNm	7.02	8.77	12	7.07
15 Mechanical time constant	ms	9.92	12.4	17	10
16 Rotor inertia	gcm^2	135	135	135	135

Specif cations

Thermal data

17	Thermal resistance housing-ambient	4.53 K/W
18	Thermal resistance winding-housing	4.75 K/W
19	Thermal time constant winding	17.7 s
20	Thermal time constant motor	227 s
21	Ambient temperature	-40…+100°C
22	Max. winding temperature	+125°C

Mechanical data (preloaded ball bearings)

23	Max. speed	10000 rpm
24	Axial play at axial load < 4.0 N	0 mm
	> 4.0 N	0.14 mm
25	Radial play	preloaded
26	Max. axial load (dynamic)	3.8 N
27	Max. force for press f ts (static)	53 N
	(static, shaft supported)	1000 N
28	Max. radial load, 5 mm from f ange	20 N

Other specif cations

29	Number of pole pairs	8
30	Number of phases	3
31	Weight of motor	110 g

Values listed in the table are nominal.

Connection

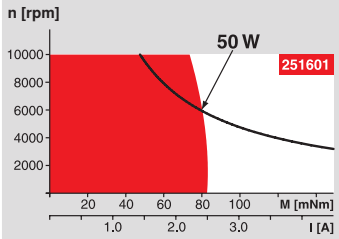
Pin 1	Hall sensor 1*
Pin 2	Hall sensor 2*
Pin 3	V_{Hall} 4.5…18 VDC
Pin 4	Motor winding 3
Pin 5	Hall sensor 3*
Pin 6	GND
Pin 7	Motor winding 1
Pin 8	Motor winding 2

*Internal pull-up (7…13 kΩ) on pin 3
Wiring diagram for Hall sensors see p. 35

Cable

Connection cable Universal, L = 500 mm	**339380**
Connection cable to EPOS, L = 500 mm	**354045**

Operating Range



Comments

Continuous operation
In observation of above listed thermal resistance (lines 17 and 18) the maximum permissible winding temperature will be reached during continuous operation at 25°C ambient.
= Thermal limit.

Short term operation
The motor may be brief y overloaded (recurring).

Assigned power rating

maxon Modular System

Overview on page 20–25

Planetary Gearhead
∅42 mm
3 - 15 Nm
Page 316

Spur Gearhead
∅45 mm
0.5 - 2.0 Nm
Page 317

Encoder MILE
256 - 2048 CPT,
2 channels
Page 342

Recommended Electronics:

Notes	Page **24**
ESCON Module 24/2	378
ESCON 36/3 EC	379
ESCON Mod. 50/4 EC-S	379
ESCON Module 50/5	379
ESCON 50/5	380
DEC Module 24/2	382
DEC Module 50/5	382
EPOS2 24/2	386
EPOS2 Module 36/2	386
EPOS2 24/5, 50/5	387
EPOS2 P 24/5	390
EPOS3 70/10 EtherCAT	393
MAXPOS 50/5	396

Option
With Cable and Connector
(Ambient temperature -20…+100°C)

April 2015 edition / subject to change

ESCON Feature Comparison Chart

NEW

	ESCON Module 24/2	ESCON 36/2 DC
DC motors up to	48 W	72 W
EC motors up to	48 W	–
Sensors		
	Digital Incremental Encoder (2 channel with or without Line Driver)	Digital Incremental Encoder (2 channel with or without Line Driver)
	DC Tacho	DC Tacho
	Without sensor (DC motors)	Without sensor (DC motors)
	Digital Hall Sensors (EC motors)	–
Operating Mode		
	Current controller (torque control), Speed controller (closed and open loop)	Current controller (torque control), Speed controller (closed and open loop)
Electrical Data		
Nominal operating voltage V_{CC}	10 - 24 VDC	10 - 36 VDC
Max. output voltage	0.98 x V_{CC}	0.98 x V_{CC}
Max. output current	6 A (<4 s)	4 A (<60 s)
Continuous output current	2 A	2 A
Pulse width modulation frequency	53.6 kHz	53.6 kHz
Sampling rate PI current controller	53.6 kHz	53.6 kHz
Sampling rate PI speed controller	5.36 kHz	5.36 kHz
Max. efficiency	92%	95%
Max. speed (DC)	limited by Max. speed (motor) and max. output voltage (controller)	limited by Max. speed (motor) and max. output voltage (controller)
Max. speed (EC; 1 pole pair)	150 000 rpm	–
Built-in motor choke	–	300 µH / 2 A
Inputs/Outputs		
Hall sensor signals	H1, H2, H3	–
Encoder signals	A, A\, B, B\	A, A\, B, B\
Max. encoder input frequency differential (single-ended)	1 MHz (100 kHz)	1 MHz (100 kHz)
Potentiometers	–	1
Digital inputs	2	2
Digital inputs/outputs	2	2
Analog inputs Resolution, Range, Circuit	2 12-bit, -10…+10 V, differential	2 12-bit, -10…+10 V, differential
Analog outputs Resolution, Range	2 12-bit, -4…+4 V	2 12-bit, -4…+4 V
Auxiliary voltage output	+5 VDC (IL ≤10 mA)	+5 VDC (IL ≤10 mA)
Hall sensor supply voltage	+5 VDC (IL ≤30 mA)	–
Encoder supply voltage	+5 VDC (IL ≤70 mA)	+5 VDC (IL ≤70 mA)
Status Indicators	Operation: green LED / Error: red LED	Operation: green LED / Error: red LED
Environmental Conditions		
Temperature – Operation	-30…+60°C	-30…+45°C
Temperature – Extended range	+60…+80°C; Derating: -0.100 A/°C	+45…+81°C; Derating: -0.056 A/°C
Temperature – Storage	-40…+85°C	-40…+85°C
Humidity (condensation not permitted)	20…80%	20…80%
Mechanical Data		
Weight	Approx. 7 g	Approx. 30 g
Dimensions (L x W x H)	35.6 x 26.7 x 12.7 mm	55.0 x 40.0 x 16.1 mm
Mounting holes	Plugable (socket headers with 2.54 mm pitch)	for screws M2.5
Part Numbers		
	466023 ESCON Module 24/2	**403112** ESCON 36/2 DC
	Order accessories separately, from page 398	Order accessories separately, from page 398

B. Source Code - PID and LQR Implementation

```

#include <math.h>
#include "MPU6050.h"
#include "stdio.h"
#include "TPM0.h"
#include "GPIO1.h"
#include "GPIO2.h"
#include "globals.h"
#include "math.h"
#include "stdlib.h"
typedef struct{
        float kp;       //proportional  coefficient
        float kd;       //differential coefficient
        float ki;       //integral coefficient
        float integral_acc;     //acumulator to calculating the integral
        float err_old;  //reminder for calculating the differential
}Pid_params;

float pid(float err, Pid_params *param)                                    // PID algorithm
{
        float err_dif;
        float output;

        param->integral_acc = param->integral_acc + err;
        err_dif = err - param->err_old;

        output = param->kp * err + param->ki * param->integral_acc + param->kd * err_dif;

        gui_old_error = param->err_old;
        param->err_old = err;

        return output;
}
```

```
void set_motor_speed(float speed)                                                    //
    Algorithm to control the motor speed and direction
{
        uint32_t duty_cycle;

        //TPM0_C5V = (uint32_t) speed;

        if(speed >= 0){
                        duty_cycle = (uint32_t) speed;
                        TPM0_C5V = duty_cycle;
                        GPIO1_SetFieldValue(GPIO1_DeviceData,
                            Motor_Direction, 0);
                        //      GPIO1_SetFieldBits(
                            GPIO1_DeviceData, Motor_Direction, 0);
                } else {
                        duty_cycle = (uint32_t) (-speed);
                        GPIO1_SetFieldValue(GPIO1_DeviceData,
                            Motor_Direction, 1);
                        TPM0_C5V = duty_cycle;
                }
}

void stabilize()

            // main
{

        //uint16_t motor_speed = 0;
        float vel = 0.0f;
        float pid_output;

        gui_motor_speed = Average_Motor_Speed();
        gui_gyro_z = Average_Gyro();
        //gui_computed_angle = MPU6050_Read_Angle() +
            gui_angle_offset;
        gui_filter_angle = 0.98*(gui_filter_angle + gui_gyro_z
            /100) + 0.02 * MPU6050_Read_Angle();



        switch (gui_set_mode){
        case 1:
                        // LQR Mode
                        // set LQR coefficients
```

```
                        param.kd = KD;                                                    // parameters controlled by GUI
                        param.ki = KI;
                        param.kp = KP;

                        vel = param.kp * gui_filter_angle + param.kd * gui_gyro_z + param.ki * gui_motor_speed;
                        gui_vel = vel;
                        set_motor_speed(vel);
                break;
        case 2:
                // PID Mode
                param.kd = KD;                                                    // parameters controlled by GUI
                param.ki = KI;
                param.kp = KP;

                pid_output = pid(gui_filter_angle, &param);

                gui_pid_output = pid_output;
                set_motor_speed(pid_output);
                break;
        }


}
```

C. Source Code - Sensor Filtering

```
float Average_Motor_Speed()
{
        float sum = 0;
        uint8_t j;
        uint16_t motor_speed = 0;
        float computed_speed = 0;
        AD1_Measure(TRUE);
        AD1_GetValue16(&motor_speed);
        computed_speed = 0.18*(motor_speed - 32657) - 36;

        motor_hist[filter_pos_motor] = computed_speed;
        filter_pos_motor++;
        if (filter_pos_motor == 4)
                filter_pos_motor = 0;
        for (j = 0; j < 4; j++)
                sum = sum + motor_hist[j];
        return sum / 4;
}

int32_t filter_pos_gyro = 0;
float   gyro_hist[4] = {0, 0, 0, 0};
float Average_Gyro()
{
        float sum = 0;
        uint8_t j;
        gyro_hist[filter_pos_gyro] = Get_Gyro_Rates();
        filter_pos_gyro++;

        if (filter_pos_gyro == 4)
                filter_pos_gyro = 0;

        for (j = 0; j < 4; j++)
                sum = sum + gyro_hist[j];
        return sum / 4;
}
```

D. Source Code - Matlab Simulations

```
clear all
close all
%% Constants
PENDULUM_MASS = 0.419; %kg
WHEEL_MASS = 0.204; %kg

COG_PIVOT_LENGTH = 0.075; %m
MOTOR_PIVOT_LENGTH = 0.085; %m

G = 9.8; %m/s^2
PENDULUM_INERTIA = 3.34*10^(-3); %kg*m^2
WHEEL_INERTIA = 0.57*10^(-3);

TORQUE_CONSTANT =    33.5*10^(-3) / ...
                     (PENDULUM_INERTIA + WHEEL_MASS *
                        MOTOR_PIVOT_LENGTH ^ 2);

RATIO = (PENDULUM_MASS*COG_PIVOT_LENGTH + WHEEL_MASS*
    MOTOR_PIVOT_LENGTH) * G / ...
          (PENDULUM_INERTIA + WHEEL_MASS * MOTOR_PIVOT_LENGTH ^ 2)


C_B = 1.02*10^(-3) %kg*m^2 / s
C_W = 0.05*10^(-3) %kg*m^2 / s

RATIO_WHEEL  = (PENDULUM_INERTIA + WHEEL_INERTIA + WHEEL_MASS*
    MOTOR_PIVOT_LENGTH^2) / ...
                  (WHEEL_INERTIA*(PENDULUM_INERTIA + WHEEL_MASS*
                     MOTOR_PIVOT_LENGTH^2))

FRAC =  (PENDULUM_INERTIA + WHEEL_MASS * MOTOR_PIVOT_LENGTH ^ 2)
%% No control torque

time_period = [0 100];
initial = [pi/180; 0; 0 ; 0];

[t, teta_p] = ode45(@(x,y)pendulum_equation(x,y, RATIO,
    TORQUE_CONSTANT, ...
```

```
    RATIO_WHEEL, C_B, C_W, FRAC), time_period, initial);



figure
plot(t, radtodeg(teta_p(:,1)));
title('Angle of the PENDULUM \newline Initial condition \theta_p(0) = 45* Deriv(\theta_p)(0) = 0   - No torque')
xlabel('Time(s)');
ylabel('Angle PENDULUM (*)')


figure
plot(t, teta_p(:,2));
title('Angular velocity of the PENDULUM \newline Initial condition \theta_p(0) = 45* Deriv(\theta_p)(0) = 0   - No torque')
xlabel('Time(s)');
ylabel('Angular velocity of PENDULUM (m/s)')

figure
plot(t, radtodeg(teta_p(:,3)));
title('Angle of the WHEEL \newline Initial condition   \theta_w(0) = 0* Deriv(\theta_w)(0) = 0   - No torque')
xlabel('Time(s)');
ylabel('Angle WHEEL (*)')

figure
plot(t, teta_p(:,4)/5);
title('Angular velocity of the WHEEL \newline Initial condition   \theta_w(0) = 0* Deriv(\theta_w)(0) = 0   - No torque')
xlabel('Time(s)');
ylabel('Angular velocity of WHEEL (rpm)')
```

The functions implemented

```
function solution = pendulum_equation(t, theta_p, K, TQ, K2, CB, CW, FRACTION)
CURRENT = 0;
%- TQ * CURRENT
solution = [theta_p(2); K * sin(theta_p(1)) - CB*theta_p(2)/FRACTION + CW*theta_p(4)/FRACTION - TQ*CURRENT/FRACTION; ...
    theta_p(4); K2*(- CW* theta_p(4) + TQ*CURRENT) - K * sin(theta_p(1)) + CB * theta_p(2)/FRACTION ];
```

Bibliography

[1] Anatolie Hristev. *Mechanics and Acoustics*. APH Bucharest, 1999.

[2] Prof. Jonathan P. How and Prof. Emilio Frazzoli. Feedback control systems. Massachusetts Institute of Technology: MIT OpenCouseWare, Fall 2010 (accessed May 10, 2015). Creative Commons BY-NC-SA.

[3] Lecture: Understanding poles and zeros. pages 1–13. Department of Mechanical Engineering, MIT.

[4] *Zhang Heng and the Seismograph*, 2009 (accessed June 03, 2015). `http://people.chinesecio.com/en/article/2009-11/04/content_81237.htm`.

[5] *Zhang Heng and the World's First Seismometer*, 2012 (accessed June 03, 2015). `http://historysinnovations.blogspot.ro/2012/03/zhang-heng-and-worlds-first-seismometer.html`.

[6] Arthur Stinner Michael R. Matthews, Colin F. Gauld. *The Pendulum - Scientific, Historical, Philosophical and Educational Perspectives*. Springer, P.O. Box 17, 3300 AA Dordrecht, The Netherlands, 2005.

[7] Leon Blitzer. Inverted Pendulum. *American Journal of Physics*, 33:1076, 1965.

[8] Mark W. Spong Daniel J. Block, Karl J. Astrőm. *The Reaction Wheel Pendulum, SYNTHESIS LECTURES ON CONTROL AND MECHATRONICS*. Morgan and Claypool Publishers, Mineola, New York, 2007.

[9] Klaus Wittmann Wilfried Ley and Willi Hallmann. *Handbook of Space Technology*. 2009.

[10] Report: Call for White Papers : Soliciting Community Input for Alternate Science Investigations for the Kepler Spacecraft. pages 1–9, 2013.

[11] *NASA Ends Attempts to Fully Recover Kepler Spacecraft, Potential New Missions Considered*, August 15, 2013 (accessed June 25, 2015). `http://www.nasa.gov/content/nasa-ends-attempts-to-fully-recover-kepler-spacecraft-potential-new-missions-cons`

[12] Rogelio Lozano Mark W. Spong, Peter Corke. Nonlinear control of the inertia wheel pendulum. *Automatica*, 1999.

[13] *Cubli - The Robotic Cube that can Jump up, Balance and Walk*, 2014 (accessed June 09, 2015). `http://makerflux.com/cubli-robotic-cube-can-jump-balance-walk/`.

[14] Richard Fitzpatrick. *Classical Mechanics An introductory course The University of Texas at Austin*. 1998.

[15] John G. Webster. *Measurement, Instrumentation, and Sensors Handbook*. 1999.

[16] Karl Johan Astrom. PID Control. Technical report, 2002.

[17] J. G. Ziegler and N. B. Nichols. *Optimum settings for automatic controllers*. Trans. ASME, November 1942.

[18] Donald T. Greenwood. *Classical Dynamics*. DOVER PUBLICATIONS, INC, Mineola, New York, 1997.

[19] Igor Thommen Mohanarajah Gajamohan, Michael Merz and Raffaello D'Andrea. The cubli: A cube that can jump up and balance. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012.

[20] Marc Pedley. Tilt Sensing Using a Three-Axis Accelerometer. *Freescale semiconductor application notes*, pages 1–22, 2013.

[21] BIOPAC System INC. Application note 273 - using bionomadix tri-axial accelerometer as a tilt sensor inclinometer. April 2013.

[22] *FREEMASTER: FreeMASTER Run-Time Debugging Tool*, accessed June 16, 2015). `http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=FREEMASTER`.