POLITECNICO DI MILANO

School of Industrial and Information Engineering

Master of Science in Automation and Control Engineering

POLITECNICO
MILANO 1863

BALANCE CONTROL OF A SEGWAY THROUGH TEXAS INSTRUMENTS HARDWARE

Supervisor:	Prof. Francesco CASTELLI DEZZA
Co-supervisor:	PhD. Nicola TOSCANI

Master Thesis dissertation of:

Gabriele LISSONI 921119

Academic Year 2020-2021

Abstract

The aim of this thesis is to study and implement the equilibrium control of a self-balancing segway. It has to autonomously swing up starting from rest position, and then reaching and maintaining the equilibrium around a stable reference angle.

The adopted control board is a LaunchXL-F28069M, produced by Texas Instruments. This hardware is programmed in Simulink environment. The setup includes a BoostXL-DRV8301, managing the power side of the control action. This driver has 3 legs only. For this reason, a particular PWM control technique has been developed in order to fully control both the motors.

The final scheme consists of a cascade control, in which the inner loops are devoted to the current control of the two motors, while the external loop regulates the roll angle of the segway.

The roll angle is measured combining data coming from three accelerometers and a gyroscope through a complementary filter.

Current PIs coefficients are based on electrical parameters of the motors, experimentally obtained during some laboratory tests.

The roll angle adopted controller is a PD. It comes from some on-field adjustments on the preliminar regulator, that was based on the simulations performed on the mathematical model of the entire system. This model is the result of a mechanical analysis of the segway structure, modeled as an inverted pendulum.

To make the segway able to autonomously swing up from rest position, a dedicated initial procedure has been implemented. It basically consists of feeding the motors with an acceleration and deceleration profile that drives up the segway. Once the roll angle reaches a defined threshold, the controller starts keeping the segway in equilibrium position.

In the last part of the dissertation, an early step towards regarding motion control is treated.

In particular, it consists of unbalancing the segway acting on the reference angle. In this way it is set to motion. Then, it is stopped by restoring the equilibrium position.

Sommario

Lo scopo di questa tesi consiste nella realizzazione di uno schema di controllo in grado di mantenere in equilibrio un segway autobilanciante. Il segway deve essere in grado di sollevarsi autonomamente dalla sua posizione di riposo, fino a raggiungere e mantenere un equilibrio stabile nell'intorno dell'angolo di riferimento imposto.

La scheda di controllo utilizzata è una LaunchXL-F28069M, prodotta da Texas Instruments e programmata in ambiente Simulink. La scheda è equipaggiata con un BoostXL-DRV8301 per permettere l'interfacciamento con i motori. Questo driver è dotato di soli 3 canali di uscita, di conseguenza è stato necessario sviluppare un particolare schema per la gestione del PWM, in modo da avere il controllo completo su entrambi i motori.

Lo schema definitivo si basa su un controllo a cascata, in cui l'anello interno è dedicato al controllo in corrente dei due motori, mentre l'anello esterno regola l'angolo di inclinazione del segway. L'angolo viene misurato combinando con un filtro complementare i dati ottenuti da tre accelerometri e un giroscopio.

I coefficienti dei PI di corrente sono basati sui parametri elettrici dei motori, ottenuti mediante alcuni test di laboratorio.

Per l'angolo di inclinazione è stato utilizzato un regolatore PD, ottenuto modificando, sulla base di alcuni test sperimentali, un precedente regolatore sintetizzato attraverso uno studio teorico. Il regolatore di partenza era basato sul modello dell'intero sistema, derivante da un'analisi meccanica della struttura del segway, modellato come un pendolo inverso.

Per rendere il segway in grado di raggiungere autonomamente la posizione di equilibrio, è stato necessario introdurre un'apposita procedura iniziale. In questa fase i motori vengono alimentati secondo un profilo di accelerazioni e decelerazioni che consente al corpo del segway di sollevarsi. Quando l'angolo supera una certa soglia, il controllore inzia ad agire sul sistema, fino a portarlo nella sua posizione di equilibrio.

Nell'ultima parte di tesi viene introdotto l'argomento del controllo del moto.

Nello specifico, il segway viene sbilanciato agendo sull'angolo di riferimento. In questo modo il controllore mobilita i motori nel tentativo di recuperare l'equilibrio, producendo un avanzamento. Il segway si arresta quando viene ristabilito il corretto angolo di riferimento.

Ringraziamenti

Desidero ringraziare tutte le persone che mi hanno affiancato durante questo percorso.

Ringrazio Nicola, per la disponibilità e la pazienza dimostrate.
Il Prof. Castelli Dezza, per avermi dato la possibilità di lavorare a questa Tesi.

Un ringraziamento particolare va ai miei genitori, che mi hanno aiutato a rendere possibile il raggiungimento di questo traguardo.

Infine, grazie di cuore a tutti gli amici con cui ho condiviso alcuni dei momenti più importanti della mia vita, ma soprattutto per esserci ogni giorno.

Grazie anche a te, perché non dimenticherò mai. $_{27}$

Contents

Chapter 1

Introduction

A Segway is a two wheels self-balanced vehicle. It was first introduced on the market by Dean Kamen in 2001.

Figure 1.1: Commercial Segway.

The control of this kind of vehicle is based on the position of the center of mass. The user can easily move forward or backward changing his/her position on the platform.

The main purpose of this thesis is to build a control scheme able to balance the segway. The adopted device is the Elegoo Tumbller shown in Fig 1.2.

In its original configuration, it was possible to identify four different levels:

- Level 1, including motors and mechanical joints;
- Level 2, composed by the metallic plate and the support leg.

- Level 3, containing the electronic boards and sensors.
- Level 4, composed by the battery case and the two plastic plates.

The original control is performed through an Elegoo Nano V3.0 equipped with a TB6612FNG module for the control of the motors. Elegoo Nano is a board equivalent to the Arduino Nano and it can be programmed through the Arduino IDE. The original version of the source code is provided by the constructor.

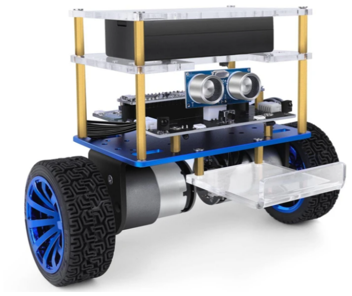


Figure 1.2: Elegoo Tumbller.

The original structure was a bit rearranged in order to allow the installation of the Texas Instruments microcontroller board. In particular, a new level was added, as support for the new electronic.

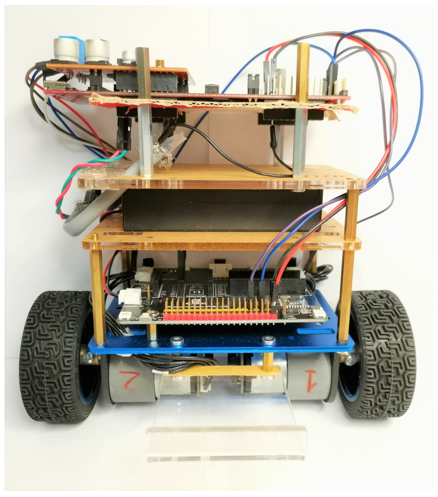


Figure 1.3: Customized segway structure.

In order to achieve the prefixed results, the thesis work was organized starting from the following considerations.

First of all, it was necessary to write a physical model of the vehicle referring to an inverted pendulum case study. In order to obtain a transfer function able to represent the mechanical behavior of the real segway, the structure was geometrically analyzed finding mass and intertia values.

The same approach was used to study the electrical part. Indeed, some tests were performed on the motors in order to characterize them, allowing the construction of a theoretical model.

Referring to the complete model, some control schemes were progressively implemented and tested through Matlab & Simulink software simulations. After that, the controllers were adapted to the real segway based on testing on the real hardware.

Texas Instruments hardware was adopted in this setup, in particular a LaunchXL-F28069M as microcontroller board and a Boost-DRV8301 as converter board. The source code was generated through Simulink using a dedicated library.

Chapter 2

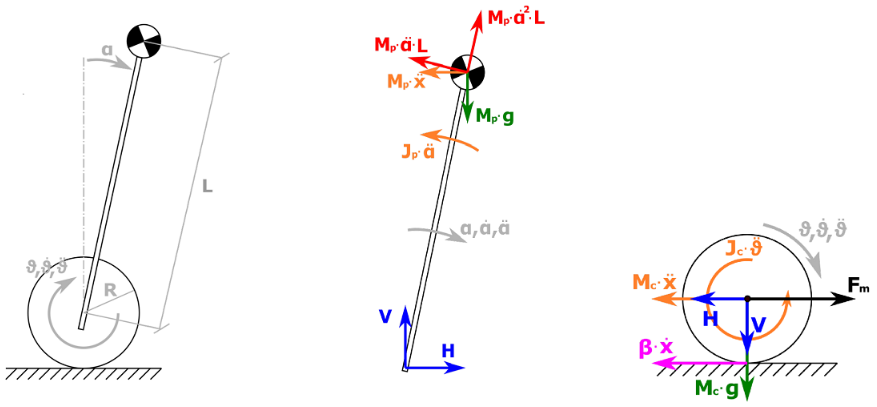
Mechanical structure

This chapter shows how the transfer function between vertical angle and required force to keep the equilibrium was obtained.

The first section reports the required steps to obtain the analytical expression of the aforementioned transfer function, whereas the explanation on how numerical values have been derived from the real structure is proposed in the second part.

2.1 Inverted pendulum model

The behavior of the segway can be easily modeled referring to the wheeled inverted pendulum.



α
L
R
$\vartheta, \dot\vartheta, \ddot\vartheta$
$M_p \cdot \dot\alpha^2 \cdot L$
$M_p \cdot \ddot\alpha \cdot L$
$M_p \cdot \ddot x$
$M_p \cdot g$
$J_p \cdot \ddot\alpha$
$\alpha, \dot\alpha, \ddot\alpha$
V
H
$J_c \cdot \ddot\vartheta$
$\vartheta, \dot\vartheta, \ddot\vartheta$
$M_c \cdot \ddot x$
F_m
H
V
$\beta \cdot \dot x$
$M_c \cdot g$

Figure 2.1: Inverted pendulum force diagram.

In the following, the list of symbols used in Fig 2.1 and in the equations is reported:

- ϑ: wheel angle.
- x: cart displacement.
- α: roll angle of the pendulum.
- R: wheel radius.
- L: distance from wheel axis and pendulum center of mass.
- M_p: pendulum mass.
- M_c: cart mass.
- J_p: pendulum moment of inertia.
- J_c: cart moment of inertia.
- β: friction coefficient between wheels and ground.
- V: vertical constraint reaction between cart and pendulum.
- H: horizontal constraint reaction between cart and pendulum.
- F_m: driving force.

Looking at the force balances along the Horizontal axis of cart and pendulum, it is possible to obtain the following equations:

$$\sum H_{Cart} = 0 \rightarrow F_m = H + M_c\ddot{x} + B\dot{x} \tag{2.1}$$

$$\sum H_{Pendulum} = 0 \rightarrow H = M_p\ddot{x} + M_pL\ddot{\alpha}\cos(\alpha) - M_pL\dot{\alpha}^2\sin(\alpha) \tag{2.2}$$

Substituting 2.1 into 2.2 a new equation for the driving force is obtained.

$$F_m = M_p\ddot{x} + M_pL\ddot{\alpha}\cos(\alpha) - M_pL\dot{\alpha}^2\sin(\alpha) + M_c\ddot{x} + B\dot{x} \tag{2.3}$$

Now, considering the force balance along the pendulum vertical axis and at the momentum balance on the pendulum center of mass:

$$\sum V_{pendulum} = 0 \rightarrow V = -M_pL\ddot{\alpha}\sin(\alpha) - M_pL\dot{\alpha}^2\cos(\alpha) + M_pg \tag{2.4}$$

$$\sum M_{Gpendulum} = 0 \rightarrow J_p\ddot{\alpha} + HL\cos(\alpha) - VL\sin(\alpha) = 0 \tag{2.5}$$

Replacing 2.2 and 2.4 in 2.5 :

$$J_p\ddot{\alpha} + (M_p\ddot{x} + M_pL\ddot{\alpha}\cos(\alpha) - M_pL\dot{\alpha}^2\sin(\alpha))L\cos(\alpha) +$$
$$-(-M_pL\ddot{\alpha}\sin(\alpha) - M_pL\dot{\alpha}^2\cos(\alpha) + M_pg)L\sin(\alpha) = 0$$

Performing some simplifications, the resulting equation becomes:

$$(J_p + M_pL^2)\ddot{\alpha} + M_p\ddot{x}L\cos(\alpha) - M_pgL\sin(\alpha) = 0 \tag{2.6}$$

In principle, equations 2.3 and 2.6 are enough to describe the whole system, but they are non-linear equations.

The aim of this analysis is to obtain some transfer functions as system descriptions and for synthesize the required controllers in a proper way.

To this purpose, considering that the control objective is to maintain the segway in his equilibrium position, it is possible to linearize the motion equations around $\alpha = 0$.

Performing the linearization ($cos(\alpha) \approx 1$ and $sin(\alpha) \approx \alpha$) the equations become:

$$F_m = (M_p + M_c)\ddot{x} + M_pL\ddot{\alpha} + B\dot{x} \tag{2.7}$$

$$0 = (J_p + M_pL^2)\ddot{\alpha} + M_pL\ddot{x} - M_pgL\alpha \tag{2.8}$$

Starting from the linearized equations 2.7 and 2.8, it is possible to apply the Laplace transform, moving from time domain to Laplace domain.

The driving force F_m was chosen as control variable, so it was defined $u(t) = F_m$.

Applying Laplace tranform

$$\mathcal{L}[\alpha(t)] = \mathrm{A}(s)$$

$$\mathcal{L}[x(t)] = \mathrm{X}(s)$$

$$\mathcal{L}[u(t)] = \mathrm{U}(s)$$

It follows that 2.7 and 2.8 become

$$\mathrm{U}(s) = (M_p + M_c)\mathrm{X}(s)\cdot s^2 + M_pL\mathrm{A}(s)\cdot s^2 + B\mathrm{X}(s)\cdot s \tag{2.9}$$

$$(J_p + M_pL^2)\mathrm{A}(s)\cdot s^2 + M_pL\mathrm{X}(s)\cdot s^2 - M_pLg\mathrm{A}(s) = 0 \tag{2.10}$$

From equation 2.10, it is possible to obtain the transfer function $\frac{\mathrm{X}(s)}{\mathrm{A}(s)}$:

$$\mathrm{X}(s) = \frac{M_pLg - (J_p + M_pL^2)\cdot s^2}{M_pL\cdot s^2}\mathrm{A}(s)$$

$$\frac{\mathrm{X}(s)}{\mathrm{A}(s)} = \frac{M_pLg - (J_p + M_pL^2) \cdot s^2}{M_pL \cdot s^2} \tag{2.11}$$

In conclusion, replacing 2.11 in 2.9 it is possible to obtain the transfer function between the vertical angle and the driving force.

$$\frac{\mathrm{A}(s)}{\mathrm{U}(s)} = \frac{M_pL \cdot s}{-C_1 \cdot s^3 - C_2 \cdot s^2 + C_3 \cdot s + C_4} \tag{2.12}$$

where:

$$C_1 = (M_pJ_p + M_cJ_p + M_pM_cL^2)$$
$$C_2 = B(J_p + M_pL^2)$$
$$C_3 = M_p(M_p + M_c)Lg$$
$$C_4 = M_pBLg$$

2.2 Mechanical parameters characterization

In order to tune the controllers, it was necessary to assign coherent numerical values to the model. To achieve this purpose, the physical structure of the segway was analyzed.

2.2.1 Pendulum center of mass

First of all, each component was weighed and the structure was divided in cart and pendulum. Then, the pendulum body was split in different levels.

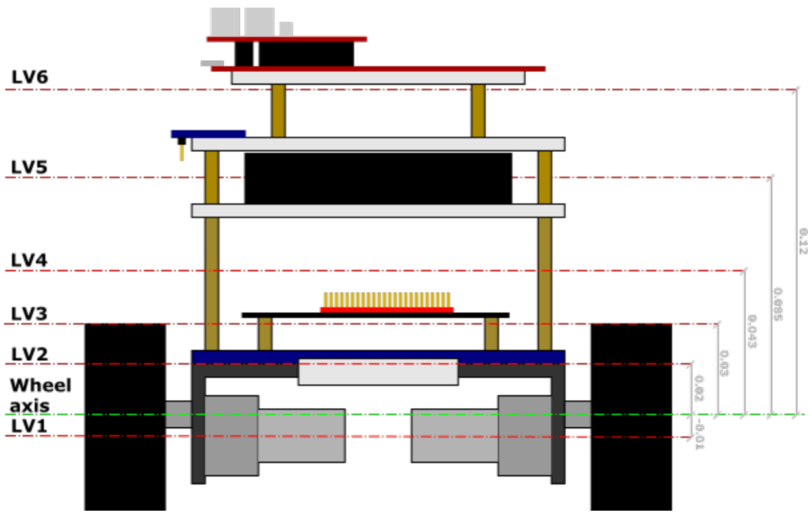


Figure 2.2: Segway levels representation.

Fig 2.2 shows how the levels are identified in the segway structure. Each axis passes through the center of mass of the level which is referred to. The customized configuration is very similar to the original one, the main difference can be found in the additional level.

- Level 1 includes the motors and the mechanical joints;
- Level 2 is composed by the metallic plate and the support legs.
- Level 3 contains the Elegoo segway electronic board and four 11 mm long metallic supports.
- Level 4 is composed by the four longest holders, 46 mm each one.
- Level 5 includes the battery case, two plastic plates and four 23 mm supports.
- Level 6 was added in order to house the control electronics. It is composed by the LaunchXL-F28069M, the BoostXL-DRV8301 and a cardboard plate with its metallic holders.

Component	Weight [kg]
Motor	0.171
Wheel	0.037
Elegoo electronic board	0.039
Batteries	0.124
Support leg	0.018
Plate 1	0.041
Plate 2	0.034
Blue plate	0.074
Wheel joint	0.014
LaunchXL-F28069M	0.045
BoostXL-DRV8301	0.025

Table 2.1: Segway components weight.

The cart is composed by wheels and their joints, so the cart mass results:

$$M_c = 0.102\,\text{kg}$$

Alle the other components are considered as part of the pendulum, so its mass is:

$$M_p = 0.790\,\text{kg}$$

Each level is assumed to be homogeneus and it is characterized by weight, width, length and distance from wheels axis. Levels characteristics are shown in Tab 2.2.

Level	mass [kg]	width [m]	length [m]	distance [m]
1	0.342	0.04	0.12	-0.010
2	0.110	0.20	0.12	0.020
3	0.039	0.08	0.09	0.030
4	0.034	0.08	0.12	0.043
5	0.199	0.08	0.12	0.085
6	0.076	0.08	0.13	0.120

Table 2.2: Size and weight of each level of the segway.

Because of homogeneity hypotesis, the center of mass of each level corresponds to the geometrical barycentre. Moreover, all the centers of mass are aligned on the vertical axis. Relying on this characteristics, the center of mass of the entire pendulum can be easily computed as weighted average of the centers of mass of each level.

$$z_G = \sum_{liv} m_{liv} \cdot z_{liv} = 0.034\,\mathrm{m}$$

2.2.2 Inertia analysis

As specified before, the segway was divided in different levels and each one of them is represented by a parallelepiped like the one shown in Fig 2.3.

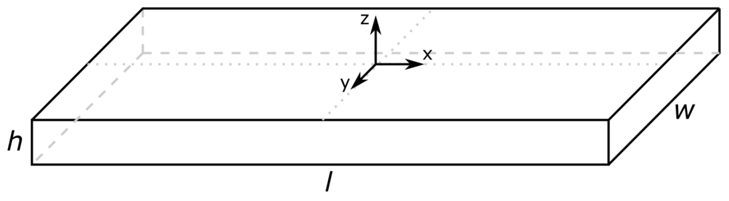


Figure 2.3: Generic representation of a level.

Because of this approximation and also considering homogenity, it is possible to compute the moment of inertia of each level through a simple double integral along x and y coordinates. This value is referred to the center of mass of each level.

$$J_{lev} = \int_{\frac{-l}{2}}^{\frac{l}{2}} \int_{\frac{-w}{2}}^{\frac{w}{2}} (x^2 + y^2) dm$$

where $dm = \rho \cdot h \cdot dx \cdot dy$, and ρ is the density of the material.

Developing the integral, it is possible to obtain a simple expression for the calculation of the inertia:

$$J_{lev} = \frac{\rho hlw}{12} \cdot (l^2 + w^2) = \frac{M_{lev}}{12} \cdot (l^2 + w^2)$$

To obtain the moment of inertia of the entire pendulum, it is necessary to refer the moment of inertia of each level to the center of mass of the pendulum z_G and then sum the obtained values. To this aim, it's possible to apply the Huygens-Steiner theorem, transporting the moment of inertia of each level on z_G through the following equation:

$$J_{Glev} = J_{lev} + M_{lev} \cdot d^2$$

where d represents the distance between the center of mass of the considered level and the center of mass of the pendulum. Results for each level are shown in Tab 2.3.

Level	J_{lev} [kg·m^2]	J_{Glev} [kg·m^2]
1	$0.456 \cdot 10^{-3}$	$0.5 \cdot 10^{-3}$
2	$0.499 \cdot 10^{-3}$	$0.5 \cdot 10^{-3}$
3	$0.047 \cdot 10^{-3}$	$0.1 \cdot 10^{-3}$
4	$0.042 \cdot 10^{-3}$	$0.1 \cdot 10^{-3}$
5	$0.345 \cdot 10^{-3}$	$1.8 \cdot 10^{-3}$
6	$0.148 \cdot 10^{-3}$	$1.2 \cdot 10^{-3}$

Table 2.3: Levels inertia values.

In conclusion, the moment of inertia of the entire pendulum can be easily computed as the sum of all the J_{Glev}.

$$J_p = \sum J_{Glev} = 0.0042\,\text{kg·m}^2$$

Chapter 3

Electrical hardware

In addition to the mechanical structure, the segway has some electrical components. In particular, there are a control board, a power electronic shield, a module for accelerometers and gyroscopes, two encoders and batteries for energy supply.

3.1 Control board LaunchXL-F28069M

The LaunchXL-F28069M (Fig 3.1) is an evaluation microcontroller board produced by Texas Instruments for prototyping applications. The adopted version is provided with a 32-Bit Real Time Microcontroller working at 90 MHz, 256 KB Flash memory and 96 KB of RAM.

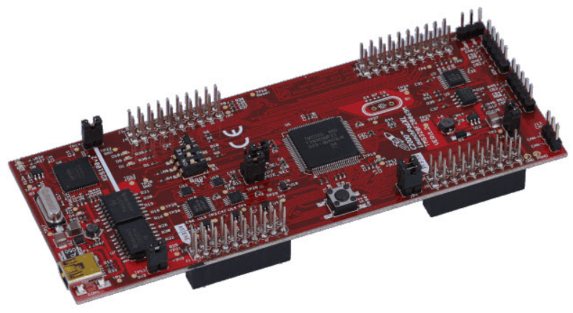


Figure 3.1: TI LaunchXL-F28069M.

One of the main reasons for choosing this board was that it can be programmed through Matlab & Simulink by using a dedicated library and allowing an easy transition from simulations to real system control.

The LaunchPad is predisposed for the reading of two encoders through the EQEP modules. Moreover, it has two pins dedicated to the I2C communication, that were adopted for the MPU6050 reading.

3.2 BoostXL-DRV8301

For what concerns the power side of the control, the choosen board was the BoostXL-DRV8301 (Fig 3.2). Also this board is produced by Texas Instruments and it is perfectly compatible with the LaunchPad.

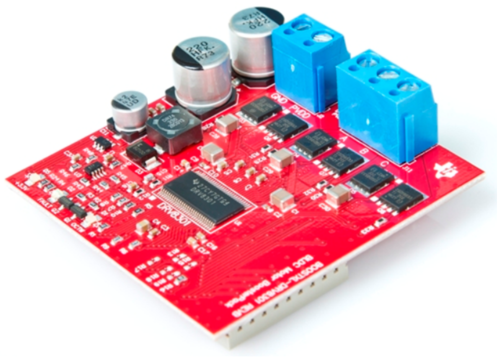


Figure 3.2: TI BoostXL-DRV8301.

It consists of a 3-legs inverter able to operate with 6 to 24 V DC supply and up to 10 A rms current.

The most intuitive choice could have been a 4-legs inverter, allowing to control the two motors with two separate H-bridges, but considering that the desired movement for this segway was on a straigth line, i.e., excluding curves, a 3-legs inverter and a dedicated control technique were enough to achieve the result. The adopted scheme is shown in Fig 3.3.

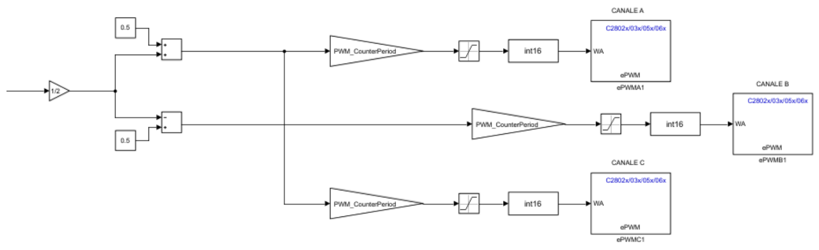


Figure 3.3: Actuation of the control logic on the three legs of the converter.

Having only 3 legs, it is necessary to use the central channel as modulable common reference for both the motors. The adopted strategy consists of simmetrical modulation of the reference and of the other two channels with respect to 0.5 of pwm value. In this way, it is possible to generate a voltage difference, between common reference and the other channels, going from $-V_{dc}$ to $+V_{dc}$ on the two motors, allowing the full movement in both directions. The only limit is that the applied voltage is the same on the two motors, but this is not relevant for this application.

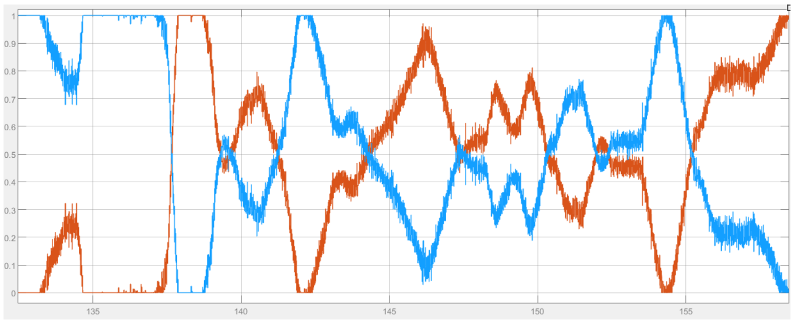


Figure 3.4: Example of PWM applied to common reference (Red) and other two channels (Blue).

In Fig 3.4, the red curve shows the duty-cycle applied to the common reference, in blue the ones applied to the other two channels. As explained before they are simmetric with respect to 0.5 and bounded between 0 and 1.

Moreover, this converter board is equipped with a shunt resistance on each channel, allowing the measurement of current absorption which is useful for the current control loop.

3.3 Mpu6050 gyroscope and accelerometer sensor

In order to stabilize the segway, it was strictly necessary to know his inclination with respect to the vertical axis. This angle corresponds to the roll angle of the MPU. For this purpose, a gyroscope and three accelerometers are combined generating a good sensing of the required angle.

The MPU6050 board is equipped with three gyroscopes, three accelerometers and one temperature sensor. Each one of them sends a package of 2 bytes once the sensor is interrogated.

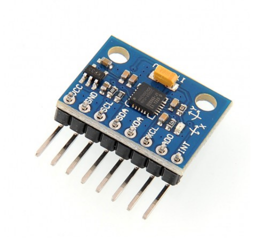


Figure 3.5: Elegoo MPU6050 Accelerometers and gyroscopes board.

This kind of sensor communicates with the control board through the serial communication, more precisely, adopting I2C protocol. A specific part of the Simulink code is dedicated to the communication management and an other one to the angle computation. It is shown in Fig 3.6.

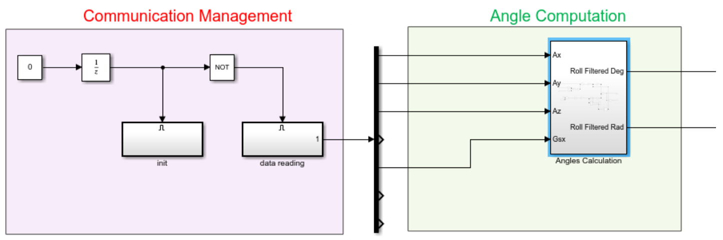


Figure 3.6: I2C communication and angle computation blocks.

3.3.1 I2C communication management

In order to set the communication, an initialization procedure is performed, sending to the sensor some strings to set the right functioning of the MPU. Fig 3.7 shows what's inside the init block.

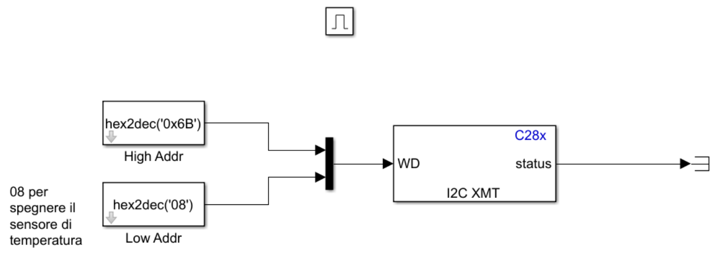


Figure 3.7: Init block.

After this one-shot initialization, the MPU is regularly set and there is an other piece of code continuosly running to manage the communication. It is called every 1 ms through a function-call generator, as it can be seen in Fig 3.8.

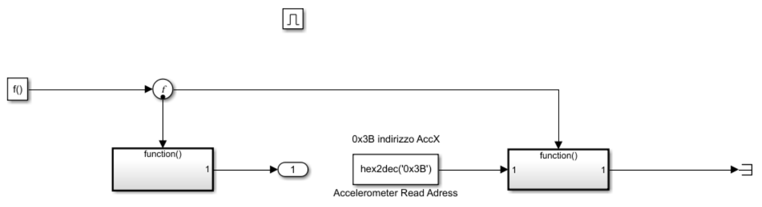


Figure 3.8: Data reading block.

First, it performs a reading of the data sent by the sensors. Once the data are collected they pass through a subsystem in which they are re-ordered. This procedure is necessary because the mpu6050 sends 2 bytes in Little Indian, for each of the sensors,. This operations are executed inside the first function block, that is expanded in Fig 3.9.

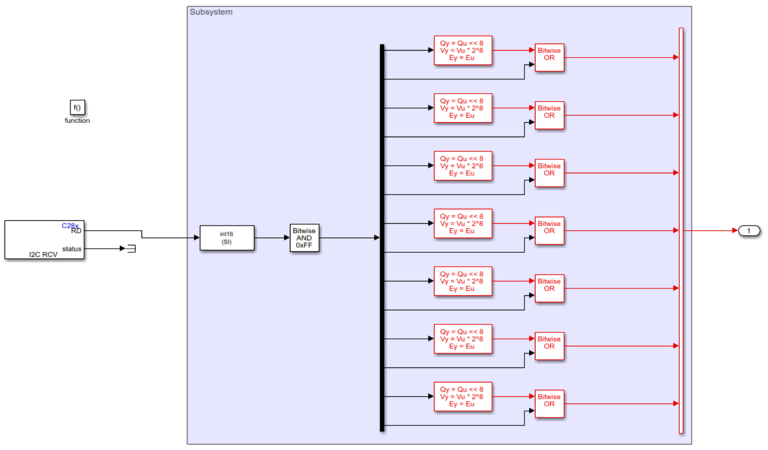


Figure 3.9: First function block.

The second function block simply consists in a Serial Send block communicating to the MPU which registers have to be read (Fig 3.10).

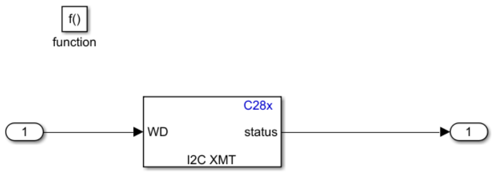


Figure 3.10: Second function block.

3.3.2 Angle computation

Once data from accelerometers and gyroscopes are ready to be used, it is necessary to elaborate them in the right way, obtaining a good estimate of the angle. First of all, the raw values must be converted from number of bits to analog values through a simple gain. The right value of this gain is determined by the fullscale value of each sensor found on the datasheet.

Having both accelerometers and gyroscope, it is possible to compute the angle in two different ways. For the sake of accuracy, the idea was to combine these two different results, obtaining a better estimate of the angle. To this aim, a complementary filter was implemented. It basically consists of a weighed average of the two estimates.

First method for the estimation of the angle uses the values coming from accelerometers. Roll angle is provided by the following operation:

$$\alpha_{acc} = atan2(\frac{\sqrt{A_x^2 + A_z^2}}{A_y})$$

Where A_x, A_y, A_z are the accelreations along MPU axes.

The other method performs a discrete integration of the angular acceleration provided by the gyroscope. Theoretically, it should be enough to calculate the angle through the following formula:

$$\alpha_{gyro}(k) = \alpha_{gyro}(k-1) + Gs_x \cdot T_s$$

Where Gs_x represents the angular acceleration along x axis measured by the gyroscope. T_s is the sample time, and it is set to 1 ms .

In practice it was necessary to face some non-idealities. In particular, the gyroscope presents a bias and a drift of the reading. The first one is easily compensated through an offset. Instead, the second one is limited through the introduction of a dead zone, avoiding to consider small variations due to the sensor drift.

It was also necessary to set a starting condition for the integration, because the segway starting position is different from 0. The initial value comes from an average of the very first values of α_{acc}. To make it possible, the discrete integration is enabled by a step after 20 ms. The overall scheme is shown in Fig 3.11.

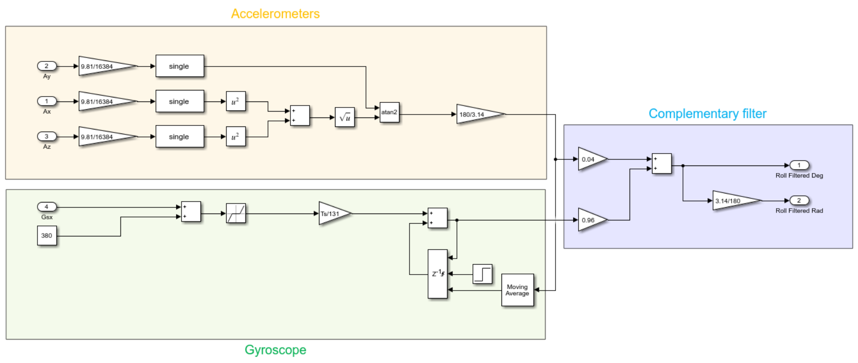


Figure 3.11: Overall angle computation scheme.

In the following graph (Fig 3.12) α_{acc}(green), α_{gyro}(blue) and the result of the complementary filter (red) are shown.

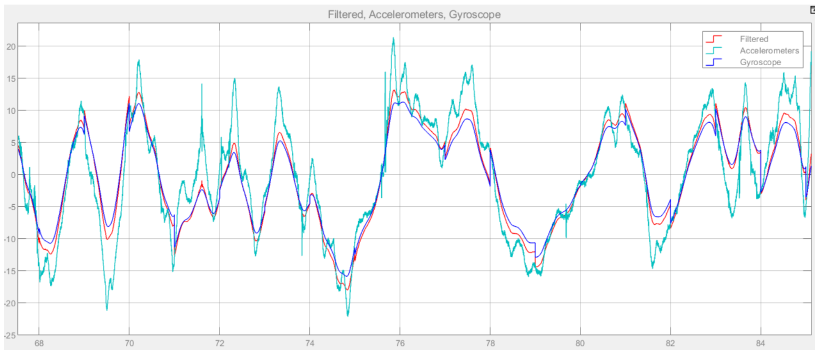


Figure 3.12: Comparison between non-filtered and filtered angles.

3.4 Encoders

Sensors for the speed acquisition were provided by Elegoo and they are directly mounted on the motor shafts. These sensors are based on hall effect, they have a 13-wire strong magnetic code disk and they can be used as encoders. By the way, they are called encoders from now on. Two output channels are provided by each sensor. Using quadrature count method, 1560 pulses correspond to a complete round of the shaft.

In the implemented control logic, encoders cover a marginal role. In fact they are used only for back emf calculation.

Encoders are read through a dedicated block provided by simulink. Pulses are counted for a period of $10 \cdot T_s$, corresponding to 10 ms and then converted in angular speed through a simple gain. The scheme is shown Fig 3.13.

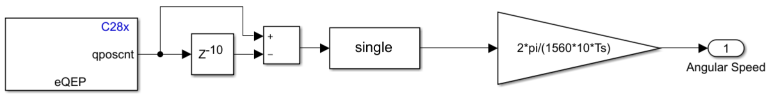


Figure 3.13: Wheels speed acquisition scheme.

3.5 Batteries

The two provided batteries have a nominal voltage of 3.7 V, they are connected in series providing 7.4 V, and real tested value is around 7.7 V. Batteries declared capacity is 2000 mAh.

Chapter 4

Electrical Motors

The adopted Motors are DC permanent magnet ones. They have a quite poor documentation. For this reason, it was necessary to characterize them through some laboratory tests.

A gearbox is directly mounted on the motors. The gear ratio is [1:30].

4.1 Transfer function of DC motors

In order to tune the PI current controller for the motors, it is good practice to know their transfer functions between armature voltage and current.

This transfer function can be found analyizing the DC permanent magnet model. Considering the armature winding, it can be found the following equivalent circuit.

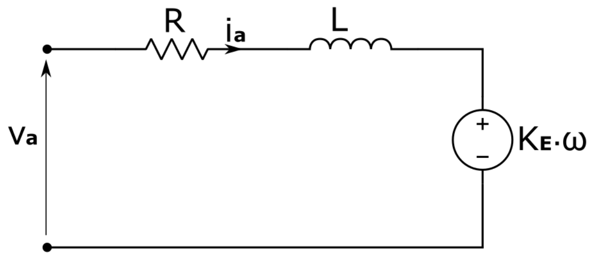


Figure 4.1: DC permanent magnet motor equivalent circuit.

Writing the KVL equation:

$$v_a - K_E \cdot \omega = R \cdot i_a + L \cdot \frac{di_a}{dt}$$

Moving in Laplace domain and rearranging the equation, it becomes:

$$I_a(s) = \frac{1}{R + L \cdot s} \cdot (V(s) - K_E \cdot \Omega(s))$$

Generated torque T_m can be easily obtained as product of the current and the mechanical constant K_T.

$$T_m(s) = K_T \cdot I_a(s)$$

Shifting the focus on the mechanical behavior of the motor, a different expression for T_m can be found. Indeed, it is possible to compute T_m as:

$$T_m = J \cdot \frac{d\omega}{dt} + B\omega + T_{rl}$$

where T_{rl} is the resistant torque of the load.
Again, applying Laplace transform and rearranging the equation, it becomes:

$$\omega(s) = \frac{1}{B + J \cdot s} \cdot (T_m(s) - T_{rl}(s))$$

4.2 Electrical parameters characterization

R and L were estimated through the analysis of data collected performing some standstill rotor tests. R can be easily derived looking at the steady state behavior of the motor and applying the Ohm law. In fact, when the rotor is locked the back emf is theoretically equal to zero.

$$V_a = R \cdot I_a + \cancelto{0}{K_E \cdot \omega}$$

Each motor underwent six standstill rotor tests, each one with a different voltage supply. For all the tests, data was measured by an oscilloscope and then analyzed through matlab. Fig 4.2 shows an example of the data collected during one test.

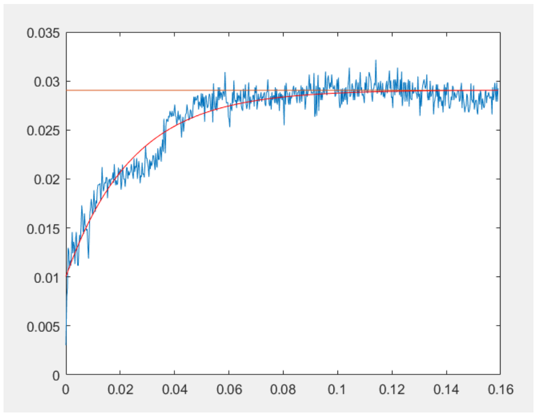
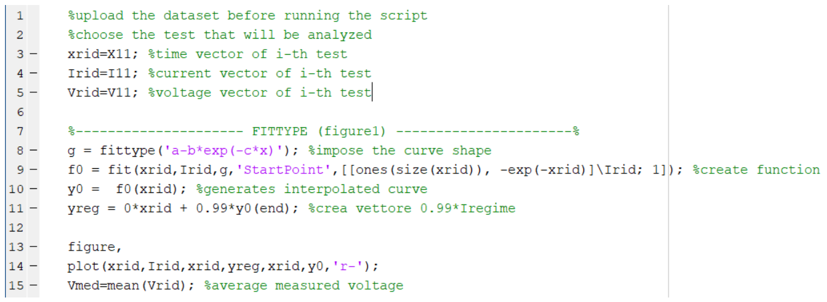


Figure 4.2: Example of data acquired during a standstill rotor test.

The graph represents the measured current during a test with 3 V applied to the motor. The exponential interpolation of the data is reported in red. The orange horizontal line represents the steady state value.

The interpolation is generated by the following matlab script (Fig 4.3):



```
%upload the dataset before running the script
%choose the test that will be analyzed
xrid=X11; %time vector of i-th test
Irid=I11; %current vector of i-th test
Vrid=V11; %voltage vector of i-th test

%--------------------- FITTYPE (figure1) ---------------------%
g = fittype('a-b*exp(-c*x)'); %impose the curve shape
f0 = fit(xrid,Irid,g,'StartPoint',[[ones(size(xrid)), -exp(-xrid)]\Irid; 1]); %create function
y0 =  f0(xrid); %generates interpolated curve
yreg = 0*xrid + 0.99*y0(end); %crea vettore 0.99*Iregime

figure,
plot(xrid,Irid,xrid,yreg,xrid,y0,'r-');
Vmed=mean(Vrid); %average measured voltage
```

Figure 4.3: Matlab script for data interpolation.

For each test, a value of R_{test} was computed through the following formula:

$$R_{test} = \frac{V_a}{I_a}$$

Then, the final value of R for each motor was calculated as the average of the six R_{test}. The final results are:

$$R_{mot1} = 6.28\,\Omega$$

$$R_{mot2} = 6.57\,\Omega$$

For what concerns L, the estimate is derived from the analysis of the time constant of each test. In particular, for a generic RL circuit, $\tau = \frac{L}{R}$. It is possible to obtain the settling time of each test from the graphs. It corresponds to the time in which the interpolated curve and the stady state value line intersect. It follows that:

$$\tau_{el} = \frac{T_{settling}}{5}$$

Consequently, it is possible to obtain an inductance value from each test

$$L_{test} = \tau_{el} \cdot R$$

Finally, the inductance of each motor is calculated as the average value of all the L_{test}.

$$L_{mot1} = 0.07\,\mathrm{mH}$$

$$L_{mot2} = 0.05\,\mathrm{mH}$$

In conclusion, the resulting transfer funcions are:

$$G_{El,mot1}(s) = \frac{1}{R_{mot1} + L_{mot1} \cdot s} = \frac{1}{6.28 + 0.07 \cdot s}$$

$$G_{El,mot2}(s) = \frac{1}{R_{mot2} + L_{mot2} \cdot s} = \frac{1}{6.57 + 0.05 \cdot s}$$

4.3 Mechanical parameters characterization

Mechanical constant K_T covers a fundamental role in the developed control scheme. For this reason, it was necessary to estimate it. From theory it's known that for permanent magnets DC motors, K_T has the same value of the electric constant K_E.

The performed test simply consists of feeding the motor with a fixed voltage and measuring armature current and rotor speed. Voltage supply V_a was imposed using the BoostXL-DRV8301 with a simple open loop pwm control, and measured through a tester. As specified in previous chapters, this board also offers a shunt resistance for current measurement, which was used to measure armature current I_a. Speed of the rotor was derived from encoders. The econder measures the rotational speed of the shaft, knowing the gear ratio [1 : 30] follows that $\omega_{rotor} = \omega_{shaft} \cdot 30$.

From steady state KVL equation, it comes:

$$V_a - R \cdot I_a - K_E \cdot \omega_{rotor} = 0$$

$$K_E = \frac{V_a - R \cdot I_a}{\omega_{rotor}}$$

Tests were performed six times on each motor and they gave really similar results for both of them. Tab 4.1 shows data collected testing motor 1.

Test	V_a [V]	I_a [A]	ω_{rotor} [rad/s]	$K_{E,test}$ [V·s/rad]
1	11.71	0.096	960	0.011
2	10.86	0.080	900	0.011
3	9.66	0.076	795	0.011
4	7.24	0.068	585	0.012
5	4.82	0.060	375	0.012
6	2.40	0.048	180	0.011

Table 4.1: $K_{E,test}$ results.

Calculating the average of all the $K_{E,test}$ for motor 1, it results

$$K_{E,mot1} = 0.011 \, \frac{\text{V·s}}{\text{rad}}$$

Applying the same procedure to the second motor the result is the same:

$$K_{E,mot2} = 0.011 \, \frac{\text{V·s}}{\text{rad}}$$

As previously specified, K_E has the same value of K_T. It follows that

$$K_{T,mot1} = 0.011 \, \frac{\text{N·m}}{\text{A}}$$

$$K_{T,mot2} = 0.011 \, \frac{\text{N·m}}{\text{A}}$$

For what concerns J and B, they were estimated by exploiting the data collected in the previous tests. Indeed, B can be obtained from the ratio between generated torque T_m and rotor speed ω_{rotor}. Knowing K_T it is easy to compute B as:

$$B = \frac{T_m}{\omega_{rotor}} = \frac{K_T \cdot I_a}{\omega_{rotor}}$$

Once again, the final values come from the average of the different tests, leading to:

$$B_{mot1} = 1.58 \cdot 10^{-6} \text{ N·m·s}$$

$$B_{mot2} = 1.31 \cdot 10^{-6}\ \text{N·m·s}$$

Moreover, analyzing the speed transient generated feeding the motors with voltage steps, it was possible to obtain the settling time $T_{settling}$ and consequently the time constant $\tau_{mecc} = \frac{T_{settling}}{5}$. Knowing B and τ_{mecc} of both the motors, it was possible to compute J_{mot1} and J_{mot2}.

$$J_{mot1} = \tau_{mecc,mot1} \cdot B_{mot1} = 0.381 \cdot 10^{-6}\ \text{kg·m}^2$$

$$J_{mot2} = \tau_{mecc,mot2} \cdot B_{mot2} = 0.363 \cdot 10^{-6}\ \text{kg·m}^2$$

In conclusion, the transfer functions describing mechanical behavior are:

$$G_{Mecc,mot1}(s) = \frac{1}{B_{mot1} + J_{mot1} \cdot s} = \frac{1}{1.58 \cdot 10^{-6} + 0.381 \cdot 10^{-6} \cdot s}$$

$$G_{Mecc,mot2}(s) = \frac{1}{B_{mot2} + J_{mot2} \cdot s} = \frac{1}{1.31 \cdot 10^{-6} + 0.363 \cdot 10^{-6} \cdot s}$$

Chapter 5

Controllers implementation: theoretical approach and simulations

As it was introduced in the previous chapters, the objective of this thesis is to implement a control scheme able to keep the segway in equilibrium.

To this aim, the first approach was to choose the control strategy to be adopted. In particular, PID controllers and pole placement were considered.

Controllers were built through Matlab scripts. The most relevant parts of the code are shown in the following.

5.1 Current control loop

Concerning the current loop, the choice was quite immediate. Indeed, the most used technique for current controls is the PI regulation on the error between reference and measured current.

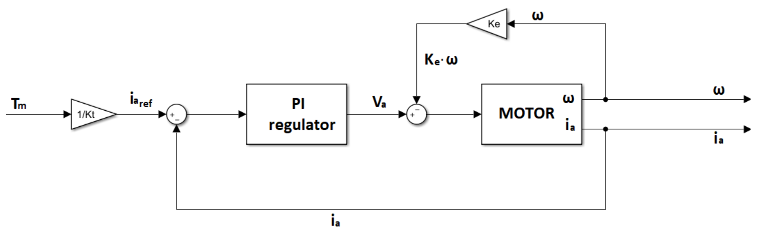


Figure 5.1: Current loop block scheme.

As shown in the parameters characterization chapter, the two motors are not exactly the same. For this reason, two different controllers were implemented.

Being motor transfer functions are already stable, PI regulation has the objective to improve performances, making the response faster.

Knowing that:

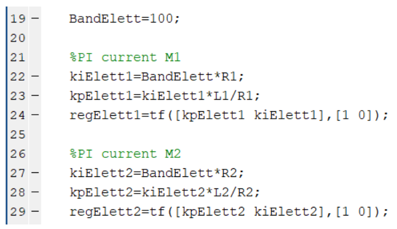
$$G_{El,mot1}(s) = \frac{1}{R_{mot1} + L_{mot1} \cdot s} = \frac{1}{6.28 + 0.07 \cdot s}$$

$$G_{El,mot2}(s) = \frac{1}{R_{mot2} + L_{mot2} \cdot s} = \frac{1}{6.57 + 0.05 \cdot s}$$

the idea was to cancel the stable pole of each transfer function and let:

$$L_{El,mot1}(s) = L_{El,mot2}(s) = \frac{\mu}{s}$$

In this way, the closed loop controlled system perfectly follows the imposed reference. The bandwidth can be easily tuned acting on μ. In order to automatically build the controllers on Matlab environment, the following piece of code was used.



```
BandElett=100;

%PI current M1
kiElett1=BandElett*R1;
kpElett1=kiElett1*L1/R1;
regElett1=tf([kpElett1 kiElett1],[1 0]);

%PI current M2
kiElett2=BandElett*R2;
kpElett2=kiElett2*L2/R2;
regElett2=tf([kpElett2 kiElett2],[1 0]);
```

Figure 5.2: Matlab script for current PI tuning.

Tuned controllers were immediately tested through another simple script, the following graph (Fig 5.3) shows the step response of the closed loop control. It's the same for both motors because, as specified before, $L_{El,mot1}(s) = L_{El,mot2}(s)$.

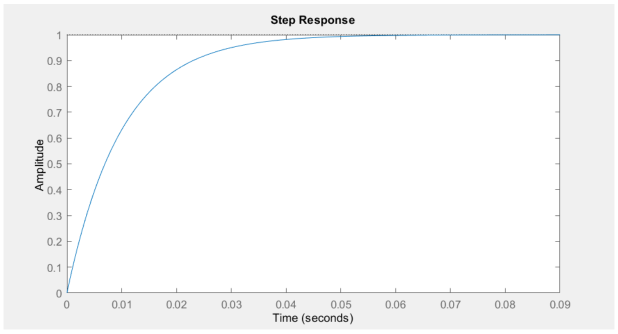


Figure 5.3: Current loop step response.

5.2 Current control loop Simulink simulations

Implemented controllers were preliminarily tested through Simulink simulations. In this way, it was easy to include also voltage saturation introduced by the batteries. Simulations are performed imposing a current step of 1 A as reference. This value was chosen considering that in normal operation the required current is lower. Simulation scheme is shown in Fig 5.4.

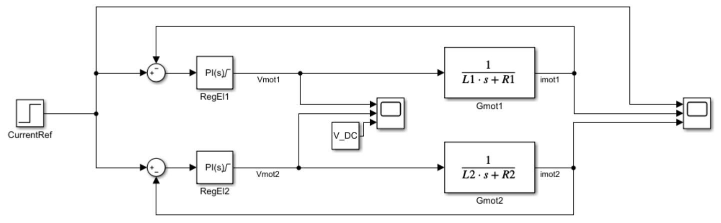


Figure 5.4: Current PI simulation scheme.

Graph in Fig 5.5 shows the current response of the closed loop system. Current flowing through the two motors is the same, this is due to the imposition of $L_{El,mot1}(s) = L_{El,mot2}(s)$.

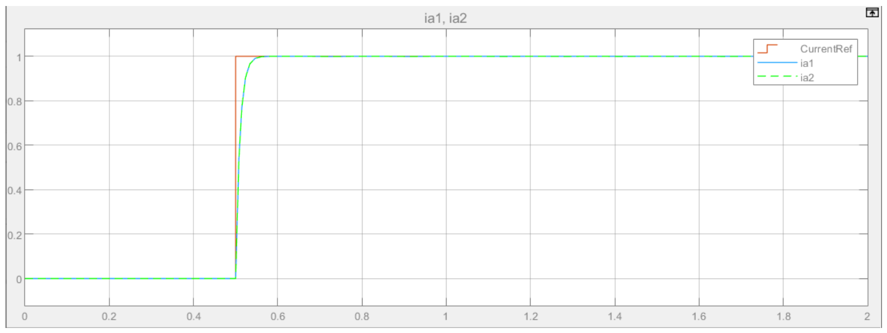


Figure 5.5: Current PI simulated step response.

Voltage is the control variable generated by the PI regulators.

To be sure that this solution was compatible with the adopted hardware, both the PI are saturated between $-7.5V$ and $+7.5V$, that is the voltage imposed by batteries.

The following graph (Fig 5.6) shows the behavior of the two different controllers.

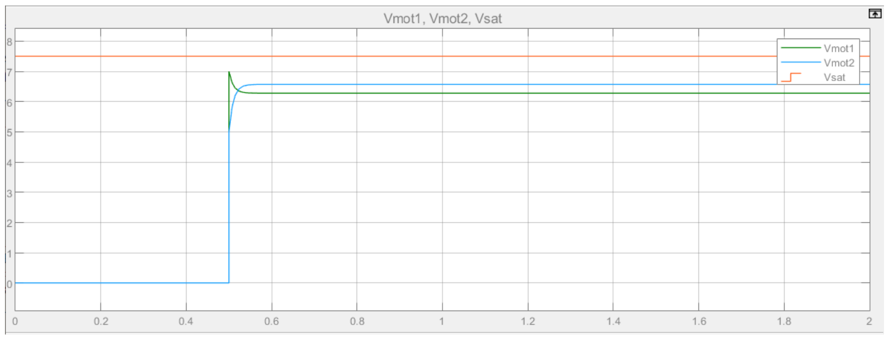


Figure 5.6: Effects on current PIs control variable during step response.

This demonstrates that both the controllers never reach the saturation value, proving that this kind of control should be sustainable for the adopted hardware.

The two curves are slightly different because, as demonstrated during the parameters identification, the two motors are not exactly identical.

5.3 Roll angle control loop

For what concerns the control of the roll angle, the situation is more complicated. Indeed, the transfer function between roll angle α and driving force F_m has an unstable pole.

$$\frac{\mathrm{A}(s)}{\mathrm{U}(s)} = G_{pendulum}(s) = \frac{-6.996 \cdot s}{s^3 + 0.1332 \cdot s^2 - 61.22C_3 \cdot s - 6.863} =$$
$$= \frac{-6.996 \cdot s}{(s-p_1)\cdot(s-p_2)\cdot(s-p_3)}$$

Where:

$$p_1 = 7.814$$

$$p_2 = -7.835$$

$$p_3 = -0.112$$

p_1 is positive, for this reason the system is clearly unstable.

In order to stabilize the system, two main approaches were initially considered.

The first one was about tuning a controller through the pole placement. Second one, was to implement a classic PID regulator.

Pole placement approach

Pole placement is based on the root locus. This technique allows to analyze how the poles vary their closed loop position, based on a gain variation. Applying root locus, the analyzed transfer function becomes:

$$L_{RootLocus}(s) = \rho \cdot \frac{-6.996 \cdot s}{s^3 + 0.1332 \cdot s^2 - 61.22C_3 \cdot s - 6.863}$$

where ρ is the introduced gain.

Matlab offers a dedicated function for root locus, graphs in Fig 5.7 show the results for both positive and negative ρ .

The aforementioned function is `rlocus()` and it takes as argument the function that has to be analyzed.

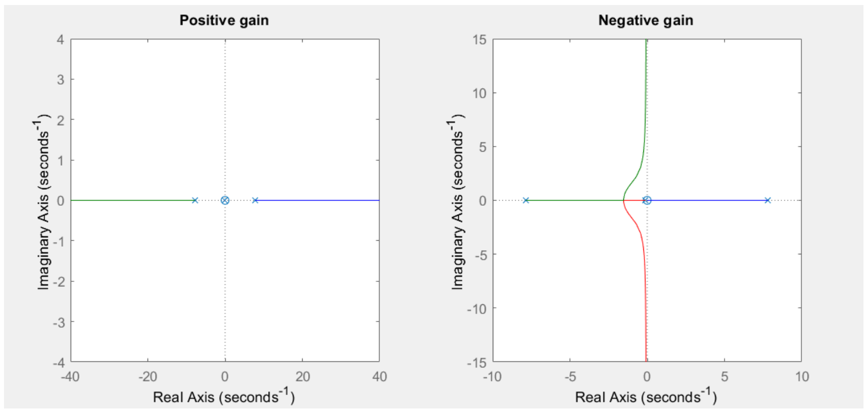


Figure 5.7: Root locus of $L_{RootLocus}(s)$.

These results demonstrate that a simple gain is not enough to asymptotically stabilize the system. Indeed, in both the graphs, the positive pole can't be brought in the left half plane. For this reason, it was necessary to introduce a more complex regulator.

Considering the need for a smooth control variable so that real systems can follow it, the idea was to tune the regulator such in the way to obtain a second order transfer function. $F(s)$ resulting from the control loop should have two stable coincident poles and no zeros. This kind of systems have a smooth response when following a reference.

To this aim, it was necessary to build up a regulator with the following form:

$$R_{PPlacement}(s) = \rho \cdot \frac{(s - p_2) \cdot (s - p_3)}{s \cdot (s - p_{new})}$$

Where p_2 and p_3 corresponds to the stable poles of the pendulum transfer function. All the pole - zero simplifications are non critical. p_{new} is a new pole, introduced with the aim of attracting the unstable pole p_1 in the left half plane. In fact, with a regulator like this, the loop transfer function becomes

$$L_{PPlacement}(s) = R_{PPlacement}(s) \cdot G_{pendulum}(s) = \rho \cdot \frac{-6.996}{(s - p_1) \cdot (s - p_{new})}$$

Analyzing the closed loop transfer function, it follows that

$$F_{PPlacement}(s) = \frac{L_{PPlacement}(s)}{1 + L_{PPlacement}(s)} =$$

$$= \frac{-\rho \cdot 6.996}{(s - p_1) \cdot (s - p_{new}) - \rho \cdot 6.996} = \frac{-\rho \cdot 6.996}{(s - p_{Fpp})^2}$$

As planned, $F_{PPlacement}(s)$ is a second order transfer function without any zeros.

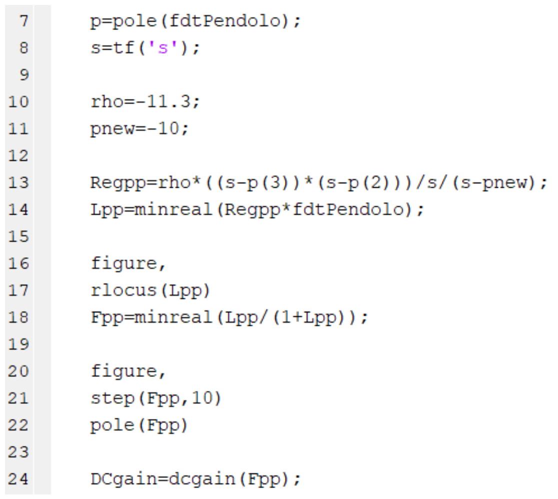
Following step is to determine ρ and p_{new} in such a way that $F_{PPlacement}(s)$ is stable and robust enough to keep the segway in equilibrium, even though initial conditions are slightly perturbed.

Considering $p_1 = 7.814$ and that the idea is to make the poles of $F_{PPlacement}(s)$ coincident, the two poles p_{Fpp} will be placed around the barycentrum

$$x_p = \frac{p_{new} + p_1}{2}$$

ρ must be chosen such in a way that poles of $F_{PPlacement}(s)$ are as close as possible to x_p.

In order to properly tune ρ and p_{new}, a Matlab script was used.



```
p=pole(fdtPendolo);
s=tf('s');

rho=-11.3;
pnew=-10;

Regpp=rho*((s-p(3))*(s-p(2)))/s/(s-pnew);
Lpp=minreal(Regpp*fdtPendolo);

figure,
rlocus(Lpp)
Fpp=minreal(Lpp/(1+Lpp));

figure,
step(Fpp,10)
pole(Fpp)

DCgain=dcgain(Fpp);
```

Figure 5.8: Matlab script for pole placement tuning.

First step is to choose where to place p_{Fpp}. Placing the poles far from 0 can make the system fast, but it may require too large values of torque that can't be provided by the available hardware. Vice versa, if the poles are too near to the vertical axis, the system may be too slow, being not enough responsive to keep the segway in equilibrium.

In the definitive version, p_{Fpp} was chosen around 1, and consequently $p_{new} = -10$. Applying the root locus, it is possible to find the right value of ρ to place the poles where desired.

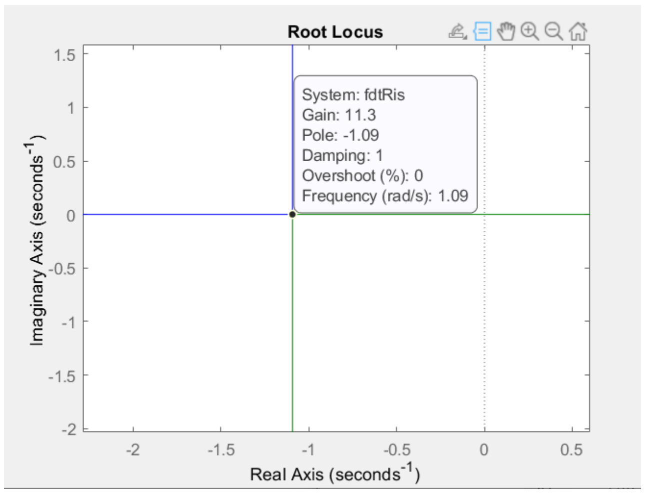


Figure 5.9: Proper gain tuning.

As indicated in Fig 5.9, $\rho = 11.3$ guarantees the poles p_{Fpp} in -1.09.

The second part of the script tests the step response of $F_{PPlacement}(s)$, it is shown in Fig 5.10.

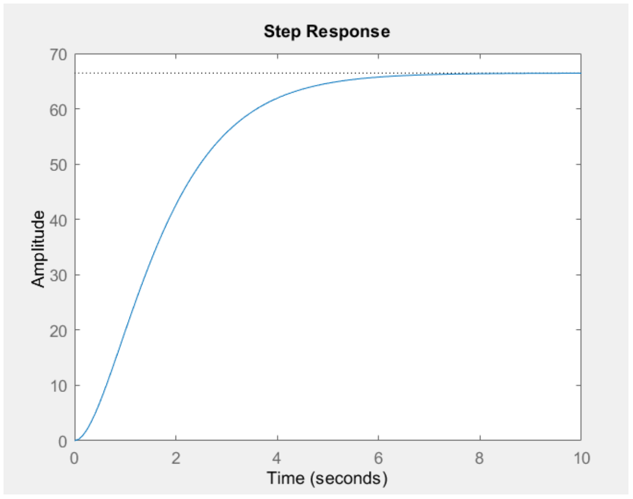


Figure 5.10: Step response of $F_{PPlacement}(s)$.

The closed loop system reaches the reference in 5.8 seconds. Gain is not unitary, for this reason, during simulations, the imposed reference is multiplied by $\frac{1}{DCgain(F_{PPlacement}(s))}$.

PID regulation approach

The second tested control strategy was a classical PID regulation.

This kind of control is largely used because of its simplicity and robustness. An ideal PID regulator has the following transfer function:

$$R_{PID}(s) = \frac{K_D \cdot s^2 + K_P \cdot s + K_I}{s} = K_D \cdot \frac{s^2 + \frac{K_P}{K_D} \cdot s + \frac{K_I}{K_D}}{s}$$

Even in this case, the adopted strategy was to cancel the stable poles of $G_{pendulum}(s)$, avoiding critical cancellations and reducing the order of the system. Then the gain is used in order to bring the unstable pole in the left half plane.

To perform this cancellation, it is necessary to impose

$$s^2 + \frac{K_P}{K_D} \cdot s + \frac{K_I}{K_D} = (s - p_2) \cdot (s - p_3) = s^2 - (p_2 + p_3) \cdot s + p_2 \cdot p_3$$

resulting in

$$K_D = gain$$

$$K_P = -K_D \cdot (p_2 + p_3)$$

$$K_I = K_D \cdot p_2 \cdot p_3$$

With this approach, there is no the introduction of a new pole. The resulting system is a first order one.

$$L_{PID}(s) = R_{PID}(s) \cdot G_{pendulum}(s) = \frac{-6.996 \cdot K_D}{s - p_1}$$

where $p_1$and -6.996 come frome $G_{pendulum}(s)$.

In order to analyze the stability of the closed loop transfer function $F_{PID}(s)$, Bode criterion can't be used, because $L_{PID}(s)$ has a positive pole. Anyway, $L_{PID}(s)$ is a first order transfer function, allowing an easy computation of $F_{PID}(s)$:

$$F_{PID}(s) = \frac{-6.996 \cdot K_D}{s - p_1 - 6.996 \cdot K_D} = \frac{-6.996 \cdot K_D}{s - p_{PID}}$$

where $p_{PID} = p_1 + 6.996 \cdot K_D$.

Looking at the denominator of this transfer function, it is possible to write the condition for the asymptotic stability of the closed loop system:

$$p_{PID} < 0$$

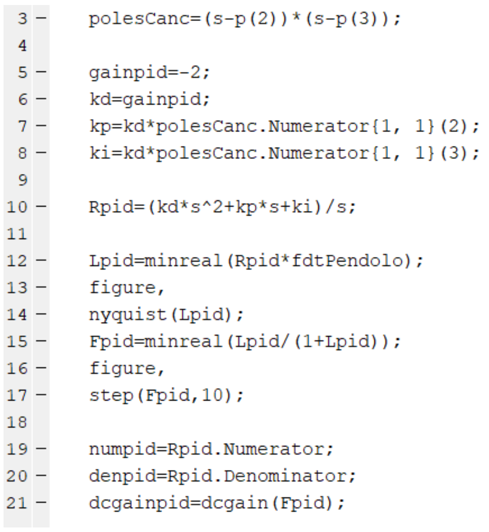
that is

$$p_1 + 6.996 \cdot K_D < 0$$

from which follows

$$K_D < \frac{-p_1}{6.996} = \frac{-7.81}{6.996} \simeq -1.12$$

For K_D tuning, a matlab script was used.



```
polesCanc=(s-p(2))*(s-p(3));

gainpid=-2;
kd=gainpid;
kp=kd*polesCanc.Numerator{1, 1}(2);
ki=kd*polesCanc.Numerator{1, 1}(3);

Rpid=(kd*s^2+kp*s+ki)/s;

Lpid=minreal(Rpid*fdtPendolo);
figure,
nyquist(Lpid);
Fpid=minreal(Lpid/(1+Lpid));
figure,
step(Fpid,10);

numpid=Rpid.Numerator;
denpid=Rpid.Denominator;
dcgainpid=dcgain(Fpid);
```

Figure 5.11: Matlab script for K_D tuning and $F_{PID}(s)$ analysis.

The script allows to easily modifiy and test different values of K_D. In fact, once K_D is set, it automatically tunes K_P and K_I.

Once the parameters are computed, the script generates $R_{PID}(s)$ and $L_{PID}(s)$. Then, it plots the Nyquist diagram of $L_{PID}(s)$, shown in Fig 5.12.

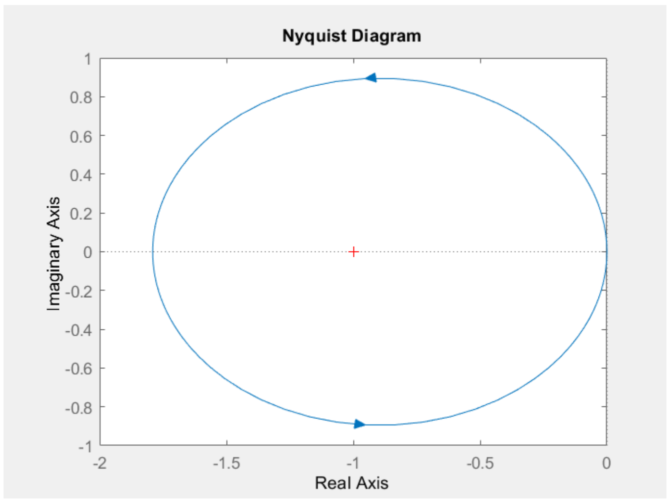


Figure 5.12: $L_{PID}(s)$ Nyquist plot.

For Nyquist criterion, a feedback loop system is asymptotically stable if the Nyquist plot encircles the point -1 as many times as the number of unstable poles of $L_{PID}(s)$.

Looking at the Nyquist plot of the system, it is possible to see that the closed curve encircles the critical point -1 exactly once. It means the feedback loop system is stable, because, as specified before, $L_{PID}(s)$ has one positive pole.

Moreover, the script computes $F_{PID}(s)$. The resulting transfer function is tested through a step reference imposition (Fig 5.13).

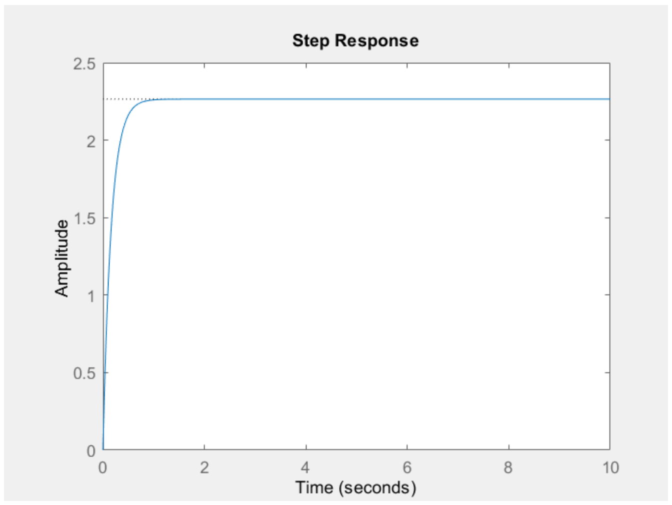


Figure 5.13: $F_{PID}(s)$ step response.

5.4 Roll angle control loop Simulink simulations

Controllers were tested through some simulations. To this aim, it was necessary to build a Simulink model of the segway. One possibility was to simply use $G_{pendulum}(s)$, but in this way the model is quite limited. In fact, using the transfer function, there was no possibility to access some intermediate variables.

For this reason, a block scheme was built starting from the linearized equations of the inverse pendulum model

$$F_m = (M_p + M_c)\ddot{x} + M_p L\ddot{\alpha} + B\dot{x}$$

$$0 = (J_p + M_p L^2)\ddot{\alpha} + M_p L\ddot{x} - M_p g L\alpha$$

This modeling strategy allowed to impose an initial condition to the roll angle α. There was also the possibility to monitor other involved variables, like cart position, speed and acceleration.

The resulting scheme is shown in Fig 5.14.

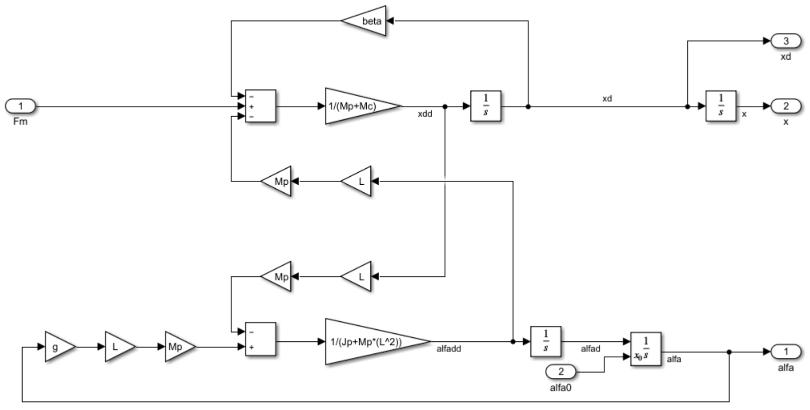


Figure 5.14: Pendulum model Simulink block scheme.

The final simulation is obtained after three intermediate steps, adding more and more details on the system, for both the tested control approaches. The aim of this subdivision is to better isolate problems. In fact, performing progressive tests was useful to understand in which part of the control startegy weaknesses were present.

All the tests were performed imposing $\alpha = 0$ as reference and $\alpha_0 = 0.05\text{rad} \simeq 3\,\text{deg}$ as initial perturbed condition.

Test 1: Segway model not considering motors

The starting point for simulation consists of a test of the roll angle controller applied to the model of the pendulum, considering ideal hardware. In practice, no models for the two motors were included in simulation. In this way, the correctness of the angle regulation can be tested.

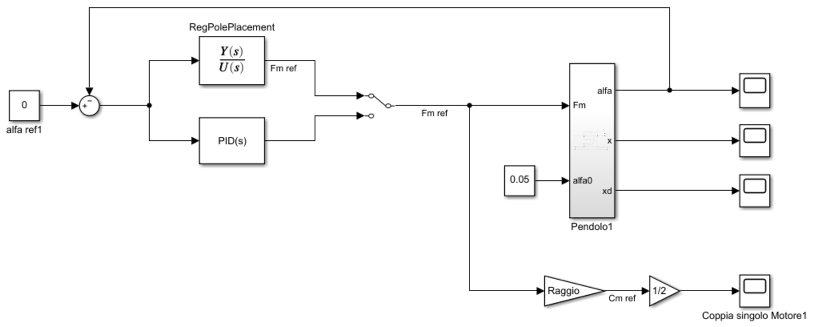


Figure 5.15: Test 1: Angle control not considering motors.

The scheme shown in Fig 5.15 has a switch in order to allow the easy testing of both the controllers. The subsystem contains the pendulum model presented in Fig 5.14.

Running the simulation, it is possible to monitor how the feedback loop system should behave. The following graph (Fig 5.16) compares the performances on α regulation of the two different approaches.

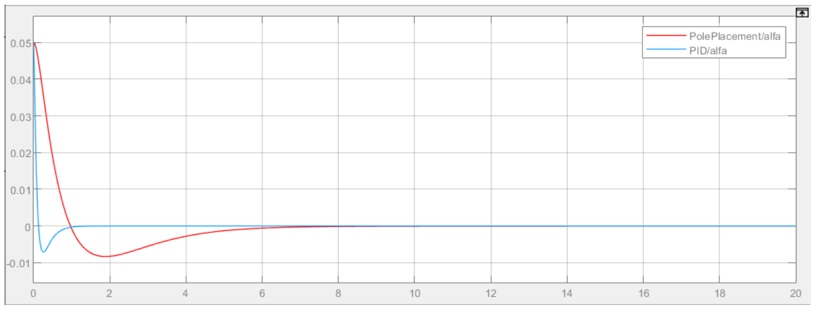


Figure 5.16: Test1: Comparison on α regulation through pole placement (red) and PID (blue).

From this plot, it is possible to see that both the controllers are able to stabilize the segway. Both of them have a soft overshoot. PID approach guarantees a faster response.

An other characteristic to take into account is the distance covered by the cart before keeping the segway in equilibrium position. Simulated diastances for both the approaches can be seen in Fig 5.17.

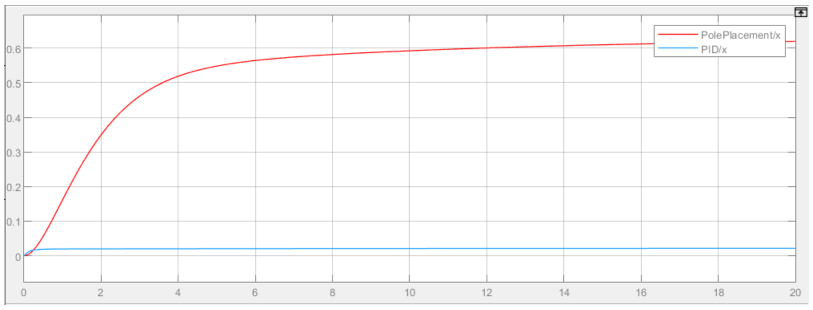


Figure 5.17: Test 1: Comparison on cart displacement x.

It can be noticed that the PID approach requires much less space to stop in the equilibrium position. In fact, with this approach, the segway should stop his run after 2 cm. Considering pole placemente approach, the required displacement grows up till 60 cm.

Anyway, this preliminar test was only useful to confirm that the implemented angle controllers were able to stabilize the segway, at least from a theoretical point of view. In fact, in this simulation, motors and hardware saturations are not considered.

Test 2: Segway model considering motors

By including the motors model, the possibility to monitor the voltage is introduced. This is important because voltage limitation due to the batteries is one of the main issues to deal with.

However, such limitations are not considered yet. In this way, it is possible to test if mechanical and electrical schemes are connected in the right way and if the cascade control works well. The block scheme for the test is the following (Fig 5.18):

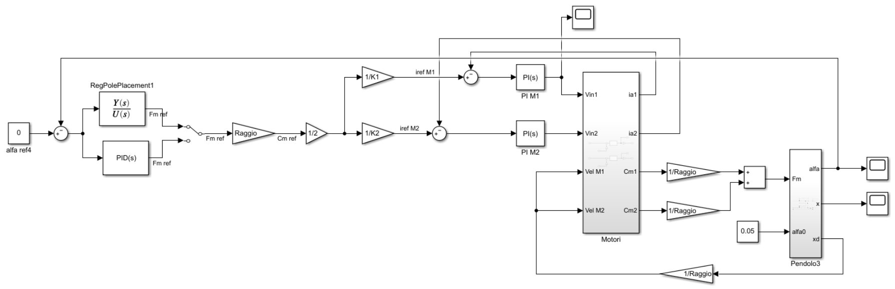


Figure 5.18: Test 2: Angle control including motors model.

Driving force reference F_{mref} is generated by the roll angle controllers and it is converted in current reference for the two motors. F_m is transformed in a torque passing through the radius r of the wheels.

$$C_{mref} = F_{mref} \cdot r$$

Then, the torque is split on the two motors:

$$C_{mref,mot1} = C_{mref,mot2} = \frac{C_{mref}}{2}$$

Finally, the two current references are computed:

$$i_{ref,mot1} = \frac{C_{mref,mot1}}{K_{T,mot1}}$$

$$i_{ref,mot2} = \frac{C_{mref,mot2}}{K_{T,mot2}}$$

These are the references for PI current regulators.

Performing the simulation, it is possible to look at α variation and compare it with the results of test 1. Variations for both the controllers are shown in Fig 5.19.

Moreover, in Fig 5.20 , the comparison between test 1 and test 2 is shown.

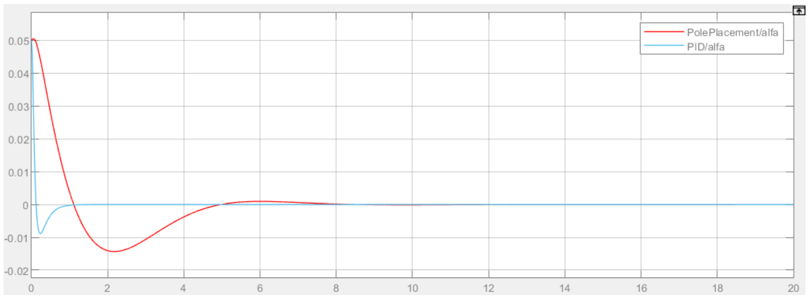


Figure 5.19: Test2: Comparison on α regulation through pole placement (red) and PID (blue).

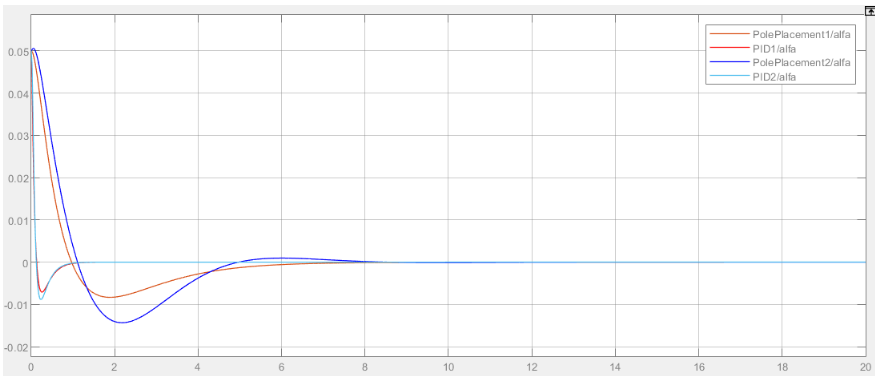


Figure 5.20: Comparison between Test 1 and Test 2.

For what concerns pole placement approach, it is possible to notice that overshoots are slightly amplified with respect to test 1, but completely acceptable.

Also in this case, the PID control appears faster and more accurate then the pole placement.

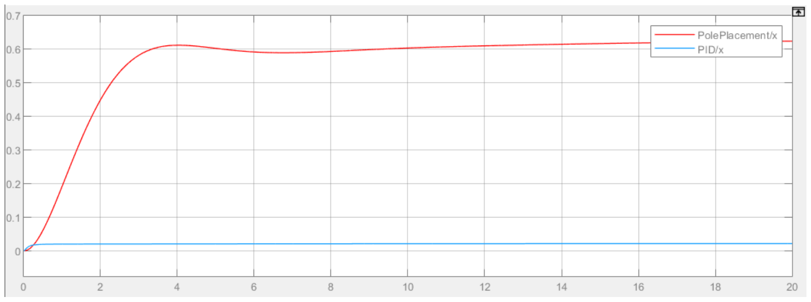


Figure 5.21: Test 2: Comparison on cart displacement x.

Looking at the cart displacements shown in Fig 5.21, the situation doesn't change. In fact, once again, the PID controller shows a better response, keeping the segway in equilibrium in 2 cm only, against the 63 cm of the pole placement.

This appreciable gap between the two approaches is the result of a large difference in control variables behavior. Indeed, looking at the voltage references generated by the current PIs (Fig 5.22), it's evident that the PID control is much more aggressive.

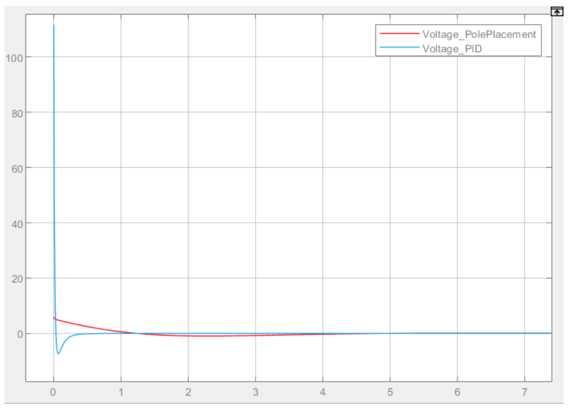


Figure 5.22: Test 2: Required voltage comparison.

At the the start-up, the PID controller commands a really high voltage, over 100 V. Of course this is not compatible with the adopted hardware, even if this amount of voltage is required for an infinitesimal time. For this reason, test 3 is performed maintaining the same PID values, testing the control considering the saturation.

For what concerns the pole placement control, the required voltage is much lower, around 5.86 V. This value is perfectly compatible with the batteries, being in the saturation range.

Test 3: Segway model considering motors and saturations

The purpose of this test is to include the voltage saturations in the model. The simulation scheme is the same of Fig 5.18, the only difference is that a saturation is imposed to the PI current controllers. It is set ranging from -7.5 V and +7.5 V, which is the maximum range obtainable with the equipped batteries. For what concerns the anti wind-up method, a back-calculation approach was used. In particular, the back-calculation coefficient was set as $K_b = \frac{K_i}{K_p}$.

Once again, the first analysis regards the comparison on regulation of roll angle α, shown in Fig 5.23.

Moreover, in Fig 5.24, the three tests results are compared.

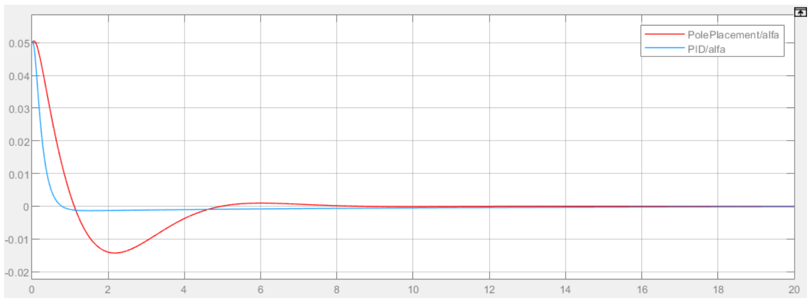


Figure 5.23: Test3: Comparison on α regulation through pole placement (red) and PID (blue).

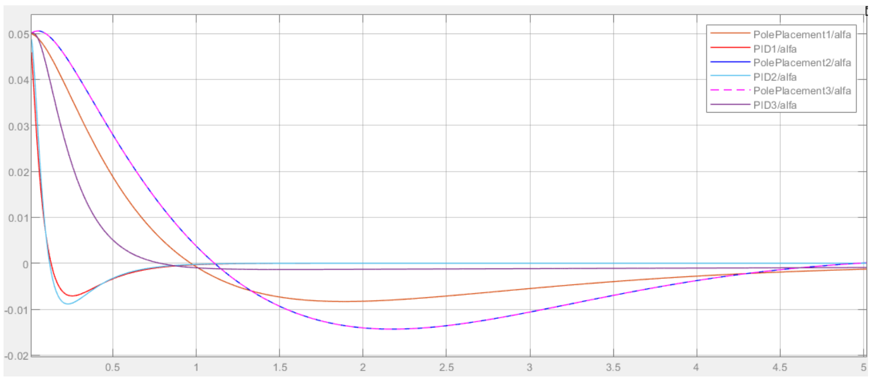


Figure 5.24: Comparison between Test 1, Test 2 and Test 3.

As in the previous tests, both the control strategies are able to keep the segway in equilibrium with good performances. This test confirms that PID approach seems to be much better than pole placement. Saturations don't affect the angle control in a significative way.

Next step is to analyze the segway behavior in terms of cart displacement.

Graphs in Fig 5.25 show how long the segway has to move before reaching the equilibium and stopping its walk.

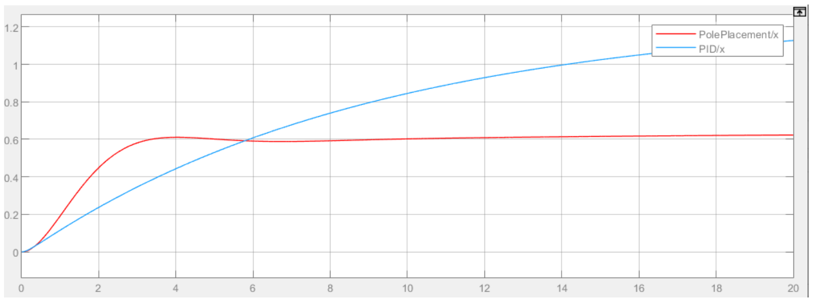


Figure 5.25: Test 3: Comparison on cart displacement x.

Differently from previous tests, the PID control approach need much more space than the pole placement one. In fact, with pole placement, the segway stops after 63 cm. With PID control, a 116 cm displacement is needed.

Focusing on PID controlled segway, the huge difference between cart displacement in test 2 and test 3 is clearly due to saturations. In fact, as shown in Fig 5.22, the required voltage at the start-up is much greater than 7.5 V.

In Fig 5.26, it is possible to see that for the first 50 ms, the PID works in saturation.

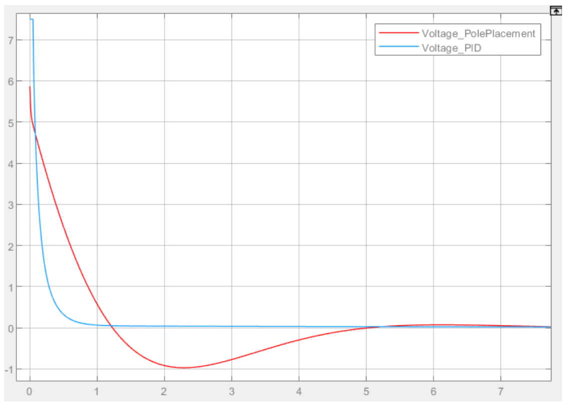


Figure 5.26: Test 3: Required voltage comparison.

Pole placement requires some voltage levels which are exactly the same as the values obtained in previuos tests and it never reaches saturation values. For this reason, angle and displacement are identical to the ones found in test 2.

There was the possibility to adjust the PID tuning and the anti wind-up scheme such in a way to obtain better results, but it was avoided observing that the model does not represent the real system in an accurate way. In fact, as it will be shown in next chapter, the behavior of the real segway is quite different from the one obtained through models and simulations.

Chapter 6

Real system

After simulations, it was necessary to evaluate the real system response. To this aim, the previously defined controllers were uploaded on the microcontroller board and progressively tested.

As a matter of fact, before running the entire control scheme, the current control loop of each motor was separately tested and fine-tuned.

6.1 Current control loop

In order to perform the current control on the real system, it is necessary to measure the current flowing through the motors and the wheels speed.

As specified in Chapter 3, the BoostXL-DRV8301 is equipped with three shunt resistances, allowing to measure the current supplied by each channel of the board. Moreover, each motor has its own encoder; Fig 3.13, shows how they are used to compute the wheels speed.

For what concerns the current measurement, the LaunchXL-F28069M provides analog input channels, each one having its own 12 bits ADC converter. Thanks to the ADC converter, it is possible to read the voltage on the shunt resistances and, then, to convert it in a current value through Simulink. This value corresponds to the current flowing into the motor attached to the channel.

Voltage range varies from 0 to 3.3 V and it is split into 4096 levels. The first operation is to subract 2048 to the measured number of levels n_i , in order to have negative readings at disposal. Moreover, each level corresponds to 8 mA and the current i_{ch_i} flowing through each channel can be theoretically computed as:

$$i_{ch_i} = (n_i - 2048) \cdot 8\,\mathrm{mA}$$

In real case, it was necessary to consider also a small offset for each channel. In order to estimate its value, it is enough to measure the current flowing through the channel while nothing is connected to it. The measured value corresponds to the offest for the tested channel. Indeed, it is enough to add a block for subtracting this value to i_{ch_i}. In this way all the measures are truly centered in 0 mA.

Current measuring is fundamental for closing the current loop of the motors.

Differently from the simulated scheme of Fig 5.4, also back emf was taken into account. The control scheme in Fig 6.1 shows how the current control loop for the real system was built.

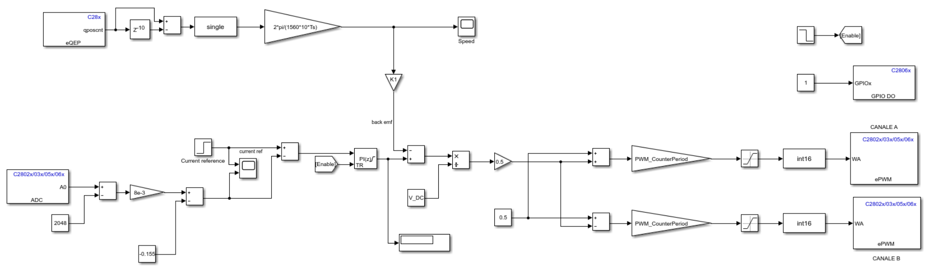


Figure 6.1: PI current loop control scheme.

In order to test the right functioning of the regulation, a step current reference was imposed while the rotor was locked and the response was measured through a scope.

When the current regulator works well, the reference has to be followed even if the rotor is locked.

Fig 6.2 shows the electrical behavior of motor 1 under the descripted test, it is the same for motor 2.

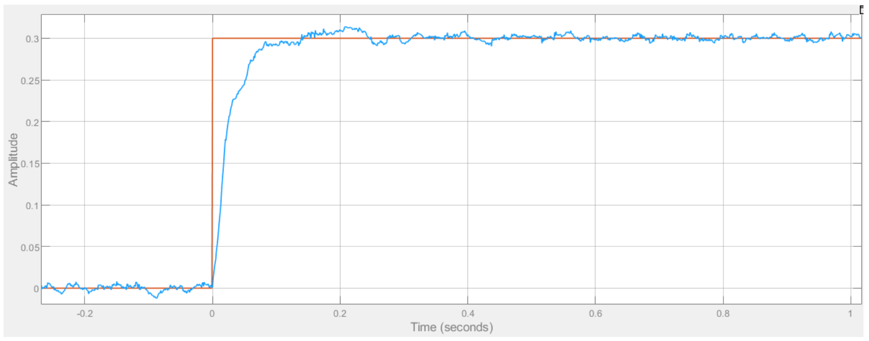


Figure 6.2: Current measured during the test.

From the graph, it is possible to see that the reference is perfectly followed, even when the wheel is blocked.

The adopted PIs have exaclty the same values that were computed through the simulation approach, this means that the models were good enough to well approximate the real behavior of both the motors.

6.2 Roll angle control loop

For what concerns the roll angle control on the real system, the situation was slightly more complicated. In fact, the controllers obtained during the simulation phase were tested on the segway, but they did not give the expected results.

This is probably due to some factors like imperfections in the model, limitations of the real hardware and inaccuracies in angle reading.

Moreover, because of these inaccuracies, the controllers showed some polarization issues. In particular, during tests on the real system, it was possible to observe that integral components were introducing some drift in the control, indeed, the control variables were considerably affected by polarization. Practically speaking, wheels continued to rotate in the same direction even if the inclination of the segway changed its sign, generating a huge delay in the response, that made the system unstable.

For this reason, after some tuning trials, the idea was to concentrate on a PD controller for the roll angle α, instead of pole placement or PID.

Also for PD controller, the intial values were model based but they had to be adjusted in order to build a controller tailored to the real system.

In the tuning phase, it was fundamental to determine the right values for proportional and derivative coefficients, but it was just as important to identify the right value for the reference angle α_{ref}. In fact, it theoretically should have been equal to 0 deg, but after some trials it was estimated to be around -1.7 degrees.

6.2.1 External mode for control scheme tuning

In a first approach to the tuning phase, it was really useful to use the "External mode" execution offered by Simulink. It gives the possibility to connect the microcontroller board to the PC and then to run the program on the real hardware, maintaining an active communication between Simulink and the microcontroller board. In this way, it was possible to perform a real time monitoring of the data and also to modifiy some control parameters without interrupting the execution.

This kind of execution gives the possibility to test different combinations for the PD controller and for α_{ref}, allowing in the meantime to check how the system responds to these variations.

The final result of the tuning procedure is the control scheme shown in Fig 6.3.

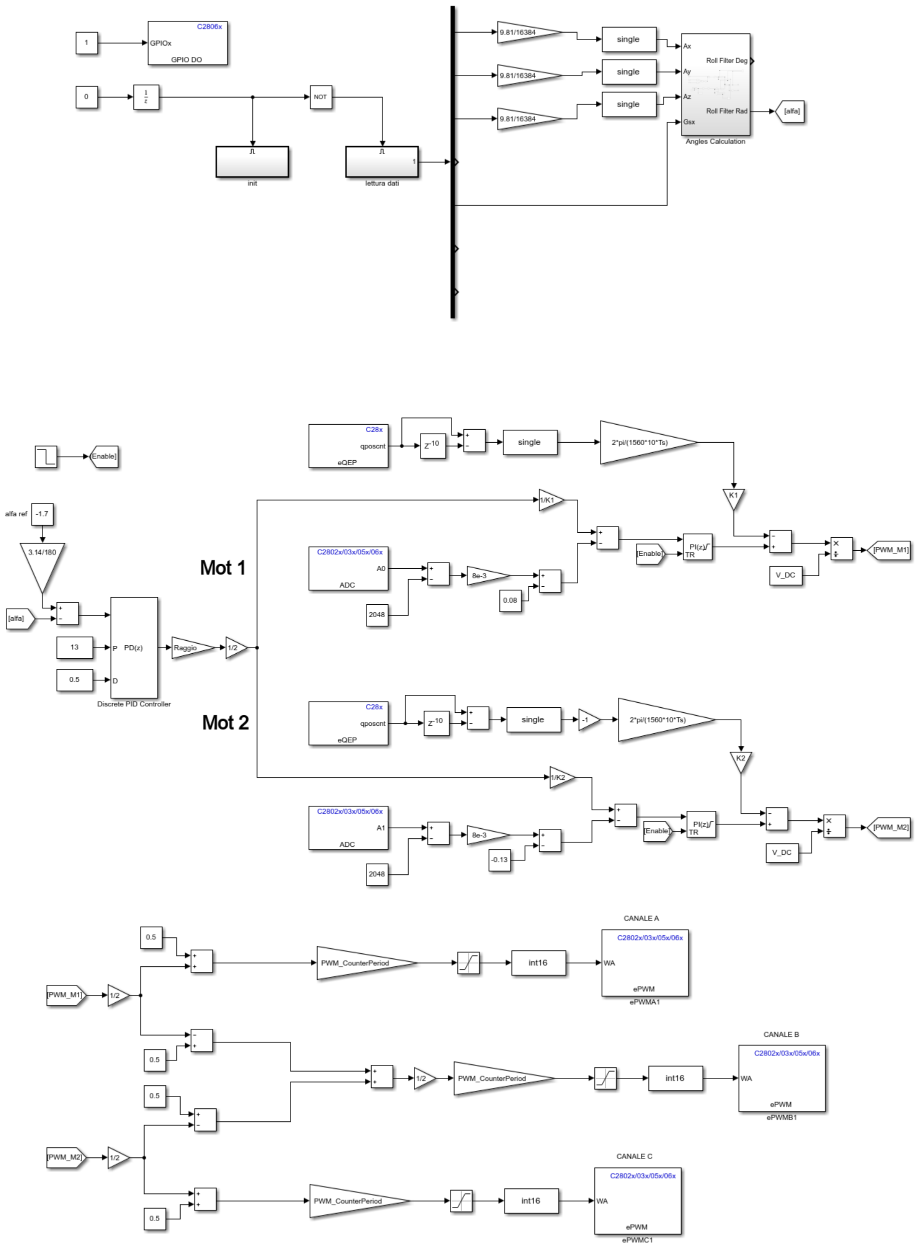


Figure 6.3: Real system control scheme.

This version of the firmware perfectly stabilized the pendulum during external mode execution. Being in external mode, it was possible to acquire the roll angle α while the segway was trying to keep the equilibrium. The result of this acquisition is reported in Fig 6.4 and it shows how the segway was perfectly stable around the imposed reference.

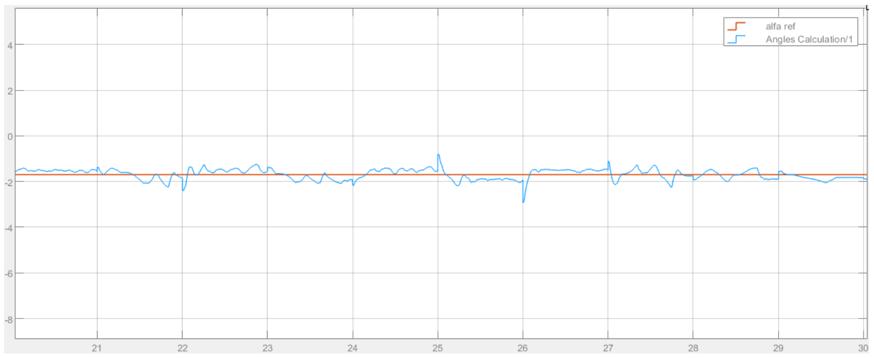


Figure 6.4: Roll angle α acquired during external mode execution.

6.2.2 Firmware upload for stand-alone operation

In order to make the segway able to autonomously work, a last step was required.

It was necessary to upload the firmware on the microcontroller board and detach the system from the PC.

The same scheme of Fig 6.3 was uploaded on the microcontroller board and tested. The result was quite unexpected, in fact, the controller was no longer able to stabilize the pendulum. It was necessary to further refine PD parameters and roll angle reference, but after some trials the obtained version was able to keep the segway in equilibrium with similar performances as before.

The difference between external mode and uploaded firmware behaviors is probably due to the fact that during external mode execution, some operations are delegated to the PC, while during stand-alone functioning all the computation is performed by the microcontroller board.

6.3 Swing up

Designed controller was able to keep the segway in equilibrium starting from a slightly perturbed condition. Next step was to make the segway able to autonomously stand up from rest position.

To this aim, it was necessary to introduce a dedicated initial procedure. It consists of imposing an acceleration and deceleration profile to the motors. In this way, the segway tries to overturn, but, before it happens, the controller takes action stabilizing the roll angle.

This procedure is managed through a specific block scheme integrated with the one showed in Fig 6.3.

In Fig 6.5, it is possible to see that for generating the acceleration profile, a pulse generator is used.

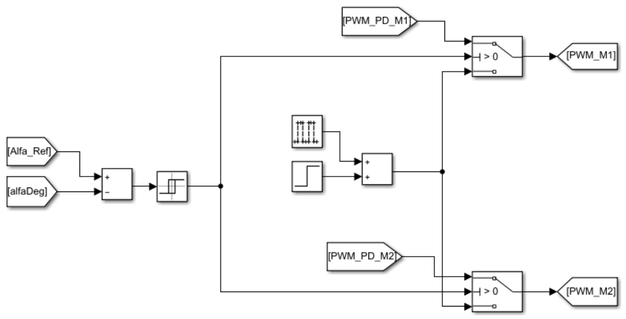


Figure 6.5: Swing up procedure Simulink scheme.

Moreover, an hysteresis relay determines the right moment for the controller intervention. It turns to 1 when the difference between roll angle α and the reference exceeds the treshold of 2 degrees. Turning to 1, it commands the switches to change their position, allowing the controller to take over the system.

Fig 6.6 shows the PWM profile applied to motor 1, before and after the PD controller intervention.

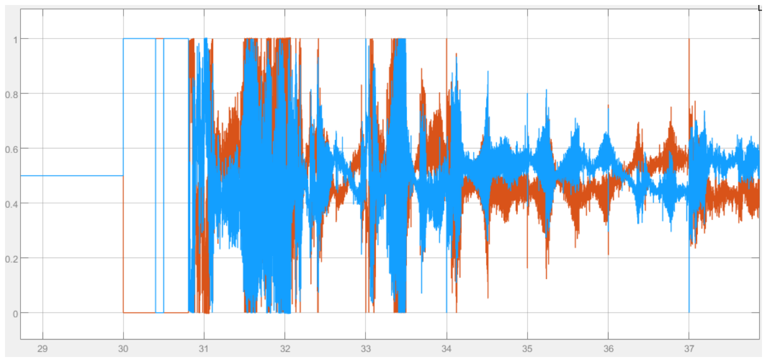


Figure 6.6: PWM command for swing up and equilibrium positioning.

In the first part, it is possible to identify the profile produced by the pulse generator. In the second one, the PWM values imposed by the PD controller.

6.4 Directed motion control

Having reached all the initial objectives, an additional challenge was faced.

It consists of trying to make the segway perform a straight movement, withouth loosing its equlbrium.

The adopted strategy consists of unbalancing the segway. To this aim, the idea was to impose an unstable reference angle α_{ref} for a short time. In this way, the segway becomes unbalanced and once α_{ref} comes back to its original value, the controller imposes a mild acceleration to keep the segway in equilibrium again, resulting in a straight movement.

The problem with this kind of approach is that it is not easy to find the right way to unblance the segway. After some trials, the best result was obtained through the scheme in Fig 6.7.

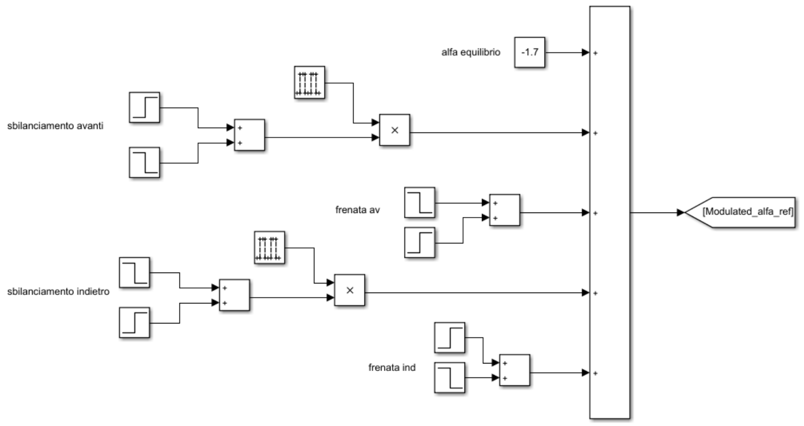


Figure 6.7: α_{ref} modulation scheme.

This scheme generates the α_{ref} profile shown in Fig 6.8.

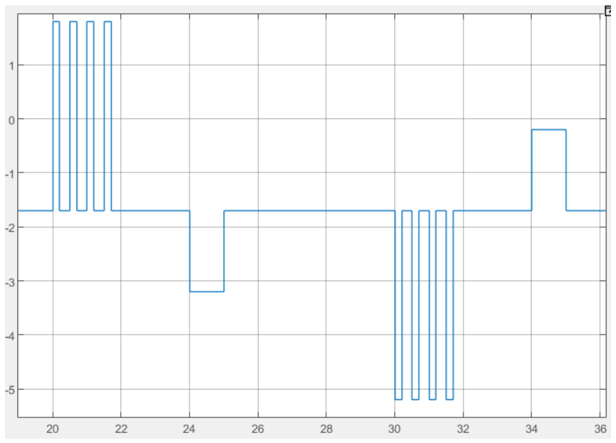


Figure 6.8: α_{ref} generated profile.

Imposing this kind of reference to the system, it is possible to obtain a forward unbalancing through the first train of pulses, generating a forward directed movement.

Pulses have a module of 3.5 degrees. Then, the segway stops its walk thanks to the single negative pulse. The braking pulses have a module of 1.5 degrees.

After 5 seconds in equilibrium, the segway is unbalanced in the opposite direction, performing a backward movement until it is stopped again by a braking pulse.

Chapter 7

Conclusions

The work at issue was meant to realize a controller for a self-balancing segway, able to stabilize its roll angle around the equilibrium position.

Some different control solutions were analyzed and tested through the usage of a specific model of the real system. The model was built according to the physics dominating the inverted pendulum, combined with a mechanical analysis of the segway structure. Moreover, electrical motors had to be analyzed and modeled.

Merging mechanical and electrical parts, it was possible to obtain a complete model suitable for controller theoretical tuning.

First approach was to progressively test these solutions through simulations on the already mentioned model.

Passing from simulations to the real system, some difficulties had to be faced. For this reason, the controllers obtained through the theoretical approach have been refined, in order to obtain a properly functioning controller for the real system.

Experimental tests were essential to refine the tuning of the control parameters and the roll angle reference.

At the end of the tuning procedure, the segway was perfectly able to maintain the equilibrium around the reference angle.

In order to make to segway able to autonomously reach the equilibrium, also starting from its rest position, a specific initial procedure was implemented.

A possible improvement for future works could be the implementation of a speed control strategy. An initial step in this direction was already faced. In fact, in the last part of the thesis, a direction control for the motion was implemented, allowing to make the segway move in forward or backward direction for a fixed time.

Other possible improvements for future works could be related to the accuracy of the model. To this aim, it could be possible to better characterize the motors, leading to completely avoid drift issues. From the mechanical side, there is the possibility to reorganize the last level of the segway. Actually, it is not perfectly simmetrical because of BoostXL-DRV8301 positioning.

Bibliography

[1] Ahmed Fahem Albaghdadi, Abduladhem Abdulkareem Ali. An Optimized Complementary Filter For An Inertial Measurement Unit Contain MPU6050 Sensor. *Iraqi Journal for Electrical And Electronic Engineering*, Vol. 15(No. 2), 2019.

[2] Akira Shimada, Naoya Hatakeyama. High-Speed Motion Control of Wheeled Inverted Pendulum Robots. *Proceedings of International Conference on Mechatronics*, 2007.

[3] Elegoo. Elegoo tumbller. Available at `https://www.elegoo.com/products/elegoo-tumbller-self-balancing-robot-car`.

[4] Elegoo. Gb37 dc geared motor with encoder. Technical report.

[5] Francesco Castelli Dezza. *Dispense "Azionamenti CC"*.

[6] Giuseppe Lorenzini. *Dispense "Setting and acquisition of a digital accelerometer"*.

[7] Kaustubh, Jaume Franch, Sunil K. Agrawal. Velocity and Position Control of a Wheeled Inverted Pendulum by Partial Feedback Linearization. *IEEE transactions on robotics*, Vol. 21(No. 3), 2005.

[8] Lalo Magni, Riccardo Scattolini. *Advanced and multivariable control*. 2014.

[9] Mathworks. Using the i2c bus to access sensors. Available at `https://it.mathworks.com/help/supportpkg/texasinstrumentsc2000/ug/using-the-i2c-bus-to-access-sensors.html`.

[10] Nicolo Bachschmid, Stefano Bruni, Andrea Collina, Bruno Pizzigoni, Ferruccio Resta, Albero Zasso. *Fondamenti di meccanica teorica e applicata*. Iii edizione edition, 2015.

[11] Paolo Bolzern, Riccardo Scattolini, Nicola Schiavoni. *Fondamenti di controlli automatici*. Iv edizione edition, 2015.

[12] Paolo Rocco. *Dispense "Decentralized Control"*.

[13] Texas Instruments. Boostxl-drv8301 hardware user's guide. Technical Report SLVU974, 2013.

[14] Texas Instruments. Launchxl-f28069m overview. Technical Report SPRUI11B, 2019.