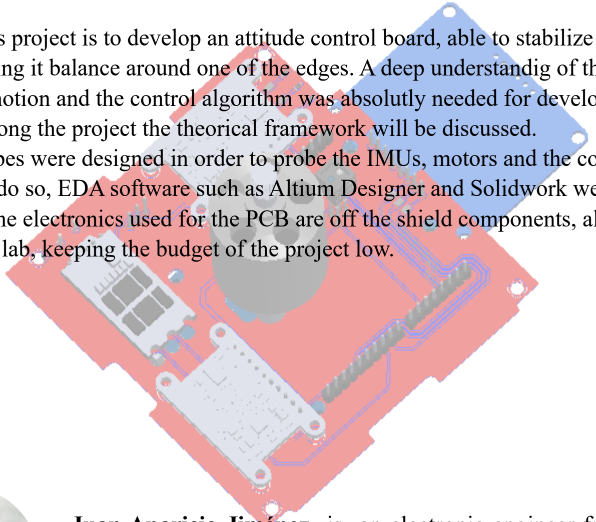


The aim of this project is to develop an attitude control board, able to stabilize a CubeSat, making it balance around one of the edges. A deep understandig of the equations of motion and the control algorithm was absolutly needed for developing the system. Along the project the theorical framework will be discussed.
A few prototypes were designed in order to probe the IMUs, motors and the control algorithm. To do so, EDA software such as Altium Designer and Solidwork were a usefull tool. The electronics used for the PCB are off the shield components, already existing in the lab, keeping the budget of the project low.

Juan Aparicio Jiménez is an electronic engineer from Granada, Spain. He finished his Bachelor's Studies in 2019 at the University of Granada.

Andrés María Roldán Aranda is the academic head of the present project, and the student's tutor. He is professor in Departament of Electronics and Computer Technology at University of Granada.

BACHELOR THESIS

Attitude Control Board for CubeSat

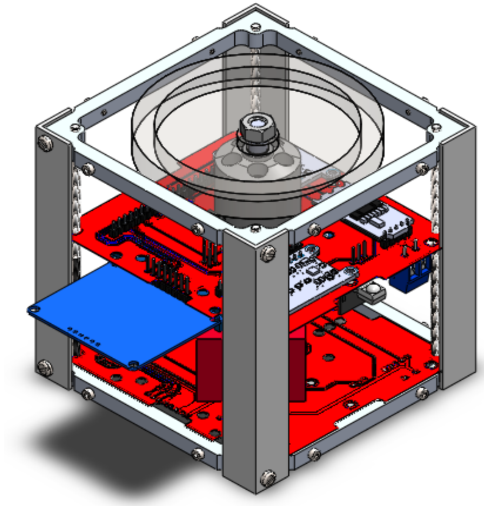
Juan Aparicio Jiménez

ELECTRONIC ENGINEERING

2018/19

UNIVERSITY OF GRANADA

Bachelor Degree in Electronic Engineering



Attitude Control Board for CubeSat

Juan Aparicio Jiménez

Academic year 2018/2019

Tutor: Andrés María Roldán Aranda

UNIVERSIDAD DE GRANADA

INGENIERIA ELECTRÓNICA INDUSTRIAL

NOMBRE DEL TRABAJO

Autor: Juan Aparicio Jiménez.

Directores: Andres María Roldán Aranda.

Departamento: Departamento de Electrónica y Tecnología de Computadores.

Palabras clave: CubeSat, Estabilidad, Torque, Momento de Inercia, PCB, Altium, SolidWorks.

Resumen:

El objetivo final de este proyecto es diseñar una PCB orientada al control de estabilidad de un CubeSat. El propósito es conseguir un algoritmo de control preciso que, acompañado de una electrónica correctamente elegida y configurada, permita levantar el cubo haciéndolo oscilar entorno a una de sus aristas. Para ello, un análisis profundo de la física de control, así como de fusión de señales y caracterización de componentes era necesaria.

Se han desarrollado varias prototipos, para conseguir un correcto entendimiento de la física detrás del control y modelado de los componentes. Para ello, herramientas de Diseño Asistido por Ordenador, como Solidworks o Altium han sido muy útiles.

Los componentes utilizados en la PCB han sido en su mayoría *off the shield*, es decir, componentes que ya estaban disponibles en el laboratorio. Además, se ha intentado que todos los componentes sean reutilizables para futuras PCB o modelos, haciendo que el presupuesto del proyecto sea lo más bajo posible.

Abstract:

The aim of this project is to develop a attitude control board, able to stabilize a CubeSat, making it balance around one of the edges. A deep understandig of the equations of motion and control algorithm was absolutly needed for developing the system. It is detailed on the first part of the project, devoted to the phisics.

A few prototypes were designed in order to probe the IMUs, motors and the control algorithm. To do so, EDA software such as Altium Designer and Solidwork were a usefull tool.

The electronics used for the PCB are off the shield components, already existing in the lab. One of the purpose of the project was to make the components reusable for further utilisation in other PCBs or models, keeping the budget of the project low.

UNIVERSIDAD DE GRANADA

INGENIERIA ELECTRÓNICA INDUSTRIAL

AUTORIZACIÓN DE LECTURA DE TRABAJO FIN DE CARRERA

D. Andres María Roldán Aranda profesor del Departamento de Departamento de Electrónica y Tecnología de Computadores de la Universidad de Granada, como director/es del Trabajo Fin de Grado titulado “Attitude Control Board for CubeSat” y realizado por el alumno D. Juan Aparicio Jiménez.

CERTIFICA/N: que el citado Trabajo Fin de Grado, ha sido realizado y redactado por dicho alumno y autorizan su presentación.

Granada,

Fdo. Prof Andres María Roldán Aranda.

UNIVERSIDAD DE GRANADA

INGENIERIA ELECTRÓNICA INDUSTRIAL

AUTORIZACIÓN DE DEPÓSITO EN LA BIBLIOTECA

Yo, D/Dña. Juan Aparicio Jiménez con DNI 45337739R, autor del Trabajo Fin de Grado titulado “Attitude Control Board for CubeSat” realizado en la Universidad de Granada.

DECLARO: explícitamente que asumo la originalidad del trabajo, entendida en el sentido de que no ha utilizado fuentes sin citarlas debidamente.

AUTORIZO: al depósito de dicho Trabajo en la Biblioteca de la Universidad de Granada, y de la visualización a través de Internet.

Granada,

Fdo. D/Dña. Juan Aparicio Jiménez.

Attitude Control Board for CubeSat

Juan Aparicio Jiménez

KEYWORDS:
CubeSat, Estabilidad, Torque, Momento de Inercia, PCB, Altium.

Abstract

El objetivo final de este proyecto es diseñar una PCB orientada al control de estabilidad de un CubeSat. El propósito es conseguir un algoritmo de control preciso que, acompañado de una electrónica correctamente elegida y configurada, permita levantar el cubo haciéndolo oscilar entorno a una de sus aristas. Para ello, un análisis profundo de la física de control, así como de fusión de señales y caracterización de componentes era necesaria.

Se han desarrollado varias prototipos, para conseguir un correcto entendimiento de la física detrás del control y modelado de los componentes. Para ello, herramientas de Diseño Asistido por Ordenador, como Solidworks o Altium han sido muy útiles.

Los componentes utilizados en la PCB han sido en su mayoría *"off the shield"*, es decir, componentes que ya estaban disponibles en el laboratorio. Además, se ha intentado que todos los componentes sean reutilizables para futuras PCB o modelos, haciendo que el presupuesto del proyecto sea lo más bajo posible.

Attitude Control Board for CubeSat

Juan Aparicio Jiménez

KEYWORDS:
CubeSat, Attitude, Torque, Inner Moment, PCB, Altium.

Abstract

The aim o this project is to develop a attitude control board, able to stabilize a CubeSat, making it balance around one of the edges. A deep understandig of the equations of motion and control algorithm was absolutly needed for developing the system. It is detailed on the first part of the project, devoted to the phisics.

A few prototypes were designed in order to probe the IMUs, motors and the control algorithm. To do so, EDA software such as Altium Designer and Solidwork were a usefull tool.

The electronics used for the PCB are off the shield components, already existing in the lab. One of the purpose of the project was to make the components reusable for further utilisation in other PCBs or models, keeping the budget of the project low.

Agradecimientos:

La cantidad de sentimientos presentes en mi pensamiento estos días es abrumadora. Mis éxitos y fracasos están indudablemente ligados a mi familia. Sin ellos, nada hubiese sido posible. Agradezco a mis padres y a mi hermana Teresa haber sido un apoyo fundamental a lo largo de mi vida, siendo exigentes cuando tenían que serlo y afectuosos el resto del tiempo. Me han educado en un ambiente de estudio y superación constante, siendo un referente cada uno de ellos. Desde que tengo uso de razón los he visto estudiar y esforzarse para conseguir sus objetivos. Me han transmitido una educación en valores y un pensamiento crítico. Además, me han regalado la pasión por la lectura y la música. Aprovecho ahora para agradecer a la literatura y a la música haberme ayudado cuando quería tirar la toalla. Todos esos personajes de las novelas que me apasionan forman parte de mí.

No puedo dejar de lado a mis amigos, jamás podría. Durante las diferentes etapas de mi vida siempre he estado rodeado de personas maravillosas, a las que considero imprescindibles. Ya sea en El Puerto de Santa María, ciudad donde me crié; en Granada, donde he madurado; o en Lyon, mis amigos me han apoyado y aportado una felicidad incalculable. Así que GRACIAS, Lucías, Helena, Ana, Martas, Marina, Gonzalo, Laura... Espero no perderos nunca.

Quiero agradecer el trabajo de todos esos profesores que me han permitido superarme en cada curso académico. El esfuerzo y dedicación de los docentes es increíble, nosotros vamos pasando, pero ellos hacen que merezca la pena. Gracias a Andrés, por retarnos cada día con cosas nuevas e inculcarnos una cultura del trabajo muy necesaria estos días.

En resumen, ¡muchas gracias a todas aquellas personas que han estado presente en mi vida!

Acknowledgments:

The amount of feelings in my head these days is overwhelming. My success and failure is undoubtedly linked to my family. Without them, nothing would have been possible. I would like to appreciate my parents and my sister Teresa for being a fundamental support throughout my life, being demanding when they had to be and charming the rest of the time. I've been taught in an atmosphere of study and constant improvement, being a reference each of them. Since I was a child I have always seen them studying and striving to achieve their goal. They have transmitted me values and critical thoughts. Moreover, they gave the passion for reading and listen to music which is undoubtedly a gif. I take this opportunity to thank literature and music for helping me when I wanted to give up. All those characters in the novels I love are part of me.

I can not leave my friends beside, I could never do it. During the different stages of my life I have always been surrounded by wonderful people, whom I consider essential. Whether in El Puerto de Santa María, the city where I grew up; in Granada, where I have matured; or in Lyon, my friends have supported me and brought me incalculable happiness. So THANK YOU, Lucías, Helena, Ana, Martas, Marina, Gonzalo, Laura ... I really hope I'll never lose you.

I want to thank the work of all those teachers who have allowed me to excel on each academic year. The effort and dedication of the teachers is incredible, we go through, but they make it worthwhile. Thanks to Andrés, for challenging us every day with new things and inculcating the culture of work, really useful these days.

All in all, thank you so much to all the people who have been part of my life!

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The following Bachelor Thesis ends the studies of the Electronic Engineering degree in University of Granada. The objective of this work was to design and implement a Attitude Control Board for the *GranaSAT-I* CubeSat.

GranaSAT is an aerospace development group from the University of Granada (UGR), which is made up entirely of students and is under the supervision of professor Dr. Andrés María Roldán Aranda.

Figure 1.1 – *Granasat logo*

1.1 State of the Art & Motivation

Theory of control is a topic that mixes up maths, phisics and engineering. The aim is to obtain an appropiate response of the system, given a feedbak of its estate. When focusing on the attitude stabilization, the purpose is to keep the system either with a exact tilt angle, a stablished orientation...

This seemed really interesting when appliying to satellites. A efficient way of setting an orientation of the satellite is needed so it can achieve its purpose. When satellites are staying longer in the space, propellants reservoirs don't seem like a practical option. At the beginning they can be useful for establishing a certain orientation of the satellite, but as they are not refillable, at some point the satellite can be useless.

For that purpose, efficient mechanism such as reaction wheels have been developed. There are some universities developing attitude control system by using reation wheels, such as the popular ”Cubli”, developed by the ETH Zurich [11], which consists of a 15x15x15 cube that is capable of balancing around one of the corners, been stable at all time. It is based itself on the inverted pendulum control, as seen in [12].

To achieve the success of our project, a deep understanding of the functional working mode of the Cubli, and some other systems, such as the inverted pendulum was absolutly needed.

A Cubesat is a 10x10x10 cubic satellite popular in the educational and amateur community, due to its features and standarization. The motivation of this project is to implement the attitude control on a PCB that can be fitted in a CubeSat, making it standarized.

1.2 Objectives

Since building a completly functional model involved many tasks and objetives, the main objetive of this Degree Thesis is to set a base for future improvement and implementation of a fully working Attitude Control on a CubeSat designed by GranaSAT.

Building from scratch and implement a fully working Attitude Control Board in a semester is an impossible task, due to the big amount of tasks and previous understanding needed. The main objetives ofthe project were set as follows:

(1) To study and understand the theory behind the equations of motion of the system. Involving the lagrange equation of motion; the dissipative energies; motor torque; gravitational torque.

(2) To study the already existing approaches when designing attitude control algoritm, state state model; digital PID implementation.

(3) To identify the system requirements in terms of accuracy; implementation; mechanical features.

(4) To develop a first prototype useful for a first approach to the control algoritm implementation; the calibration of the IMUs; and the motor torque characterisation.

(5) To learn using a milling CNC machine, useful in terms of developing future prototypes.

(6) To understand the theory behind signal merging and implementation of signal filtering to get the position by using IMUs.

(7) To develop a method for the characterisation of the motor’s torque when controlled by a PWM signal.

(8) To use EDA software tools such as Solidworks and Altium for real projects designing.

(9) To develop a working attitude control PCB through Altium Designer.

(8) To get used to working in a lab while being part of a team.

(10) To identify problems during designing or development. Learn to solve problems by developing an analitic view of the issues.

(11) To set a theorical explanation of every achievement in terms of designing, firmware, or setting the theorical framework, increasing the GranaSAT team know-how.

Therefore, the main purpose will be to set a starting point of a fully working attitude control board to be implemented in future CubeSats designed by GranaSAT. Future students could finish the implementation, solve mistakes, test it and improve the design.

1.3 Workflow & Chapter Description

Once the objectives of the project have been detailed, the workflow has to be set, so a sistematic approach is followed. The basic workflow that every engineer should follow is shown in figure 1.2

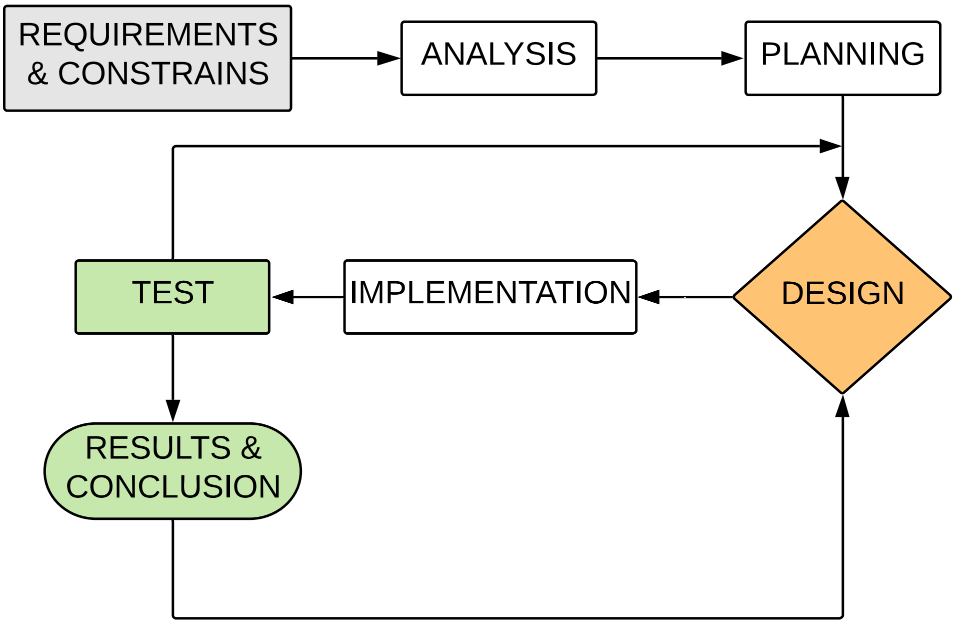


Figure 1.2 – *Engineering design workflow.*

Following the diagram of the workflow, each chapter will be briefly summarized:

- **Chapter 2: System Analysis & Theorical Framework**
  Along this chapter the phisics of the system are explained, setting the requirements of the system under the theorical framework. First, the dynamics equations of motion are gotten combining Euler-Lagrange equations. Also, the method for the characterisation of the PWM signal vs Torque will be exposed. The mechanical features are also treated. The PID tunning developement and the stability of the system are deeply analyzed.

- **Chapter 3: Mechanical Design.**
  The different prototypes and the aim of each one of them is explained. Also, the milling features learnt during the utilization of the CNC machine are shown. The explanation of the actual PCB design will be discussed.

- **Chapter 4: Electronic Design.**
  Each component used along the project is analyzed and its working mode is shown. Therefore by a quick review, anyone could set each component to its operation mode. The rpm measuring system is deeply explained, along with the phisics behind the signal merging and filtered needed for a proper position estimation through an IMU. The firmware update done to the ESC variator used for the BLDC motor is also analyzed.

- **Chapter 5: Software Design.**
  The coding of the project is cut up in different functions, implementing the requirements and working mode of each component shown in chapter 4. Some diagrams are avaliable for an easy understanding.

- **Chapter 6: Final PCB Design.**
  The PCB design pdf, including hierarchical schematics and the PCB itself is shown. Each schematic has a description of the component. The signal connection schematic is set as the top level entity.

- **Chapter 7: Conclusions & Future Work.**
  Brief conclusion and future overview.

Chapter 2

System Analysis & Theorical Framework

When starting the project, the theorical framework was absolutly essential. For this reason, along this chapter the dynamics equation of the system are analyzed, making possible to understand the constrains and the relationship between each part of the system. To do so, Euler-Lagrange equation will be exposed. Later on the chapter, the PID control algorithm is analyzed and the tunning parameters are set for a parametrizable set of values. The mechanical features are also set, allowing a successfull design.

2.1 Dynamics of the system

The equations of motion are absolutly need when designing a system. The aim of this section is to get the proper equation of the dynamics of the system, in order to be used for developing a successful control algorithm and mechanical design. The final purpose of this project is to develop a Attitude Control Board that allows a cube to self stand. The dynamics of a cube balancing around one of the edges can be described just by considering it as a 2D object, as it will only balance around one edge by now, as said before. Therefore, as a first approach to the system's dynamics, we will focus on the system shown in Figure 2.1.

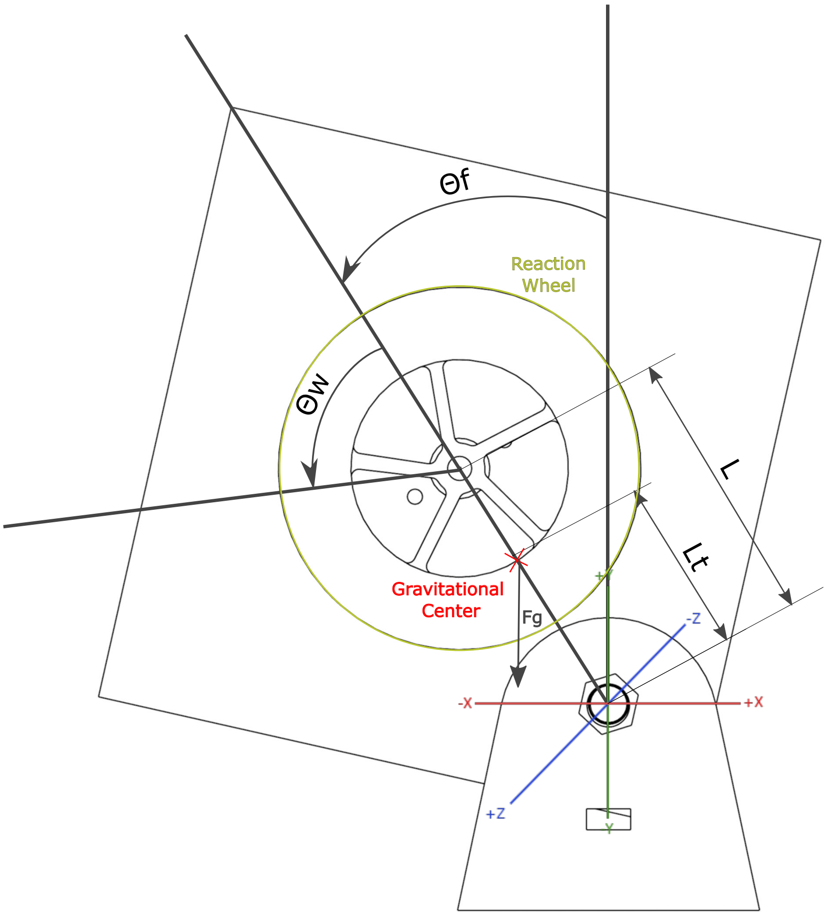


Figure 2.1 – *2D Model of the Balancing System*

For understanding the phisics within our project Euler Lagrange equations had to be deeply understood and studied. In order to model our system, according to D'Alembert's principle formulation of nonsmooth mechanics which include systems with nonconservative forces, Euler-Lagrange as used. They given by [13]:

$$\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{q}_i}\right) - \frac{\partial T}{\partial q_i} = Q_i \tag{2.1.1}$$

Where q_i is the angular position, $\dot{q}_i$ is the angular velocity, Q_i is the generalised force associated with coordinate q_i, and T is the total kinetic energy of the system.

Moreover, the Lagrange function (L) is defined as [13]:

$$L = T - V \tag{2.1.2}$$

Where T is the total kinetic energy of the mechanical system. V is the total potential energy of the mechanical system considering the whole system which is characterised. The generalised force associated with each coordinate can be defined as [14]:

$$Q_i = \sum_{i=1}^{N} F_i \frac{\partial r_i}{\partial \dot{q}_i} \tag{2.1.3}$$

On each coordinate of our system those generalised forces are:

$$Q_i = -\frac{\partial V}{\partial q_i} + \tau_i - \frac{\partial R}{\partial \dot{q}_i} \tag{2.1.4}$$

Where τ_i are the nonpartial torques and $\frac{\partial R}{\partial \dot{q}_i}$ models the dissipative forces. The potential Energy of the system isn't a function of either θ_f or θ_w, therefore:

$$\frac{\partial V}{\partial \dot{q}_i} = 0 \tag{2.1.5}$$

By combining the equations given in 2.1.1 and 2.1.2, Euler-Lagrange equations are given by the expression:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \frac{\partial L}{\partial q_i} = \tau_i - \frac{\partial R}{\partial \dot{q}_i} \tag{2.1.6}$$

2.1.1 Potential Energy

The potential Energy of our system is related to effect of gravitational force on our system. We consider that gravity affects on the center of mass, so we model the potential Energy caused by the gravitational force as:

$$V = L_t m_t g \cos\theta_{face} \tag{2.1.7}$$

2.1.2 Kinetic Energy

The total kinetic energy of the mechanical system can be seen as the sum of all the moving objects in the system. For our purpose, we consider the of the kinetic energy of the reaction wheel, and the kinetic energy of the face itself, which also turns. Therefore:

$$T_{tot} = E_{Face} + E_{wheel} \tag{2.1.8}$$

As said in [15], for a single particle rotating around a fixed axis we can relate the angular velocity to the magnitude of the translational velocity using the relation $v_t = \omega r$, where r is the distance of the particle from the axis of rotation and v_t is its tangential speed. Substituting into the equation for kinetic energy, we find:

$$K = \frac{1}{2}mv_t^2 = \frac{1}{2}m(\omega r)^2 = \frac{1}{2}(mr^2)\omega^2. \tag{2.1.9}$$

If we consider a rigid body as the sum of small particles, the angular velocity will be the same for all the particles. Thus, the Kinetic Energy of the rigid body will be:

$$K = \frac{1}{2}(\sum_j m_j r_j^2)\omega^2. \tag{2.1.10}$$

Which can be derived to:

$$K = \frac{1}{2} I \omega^2. \tag{2.1.11}$$

Where I represents the moment of inertia, with units of kgm^2, $I = \sum_j m_j r_j^2$.

For our purpose, I_{face} is considered as the inner momentum of the whole system but the inner wheel, around the pivot point. I_{wheel} is the inner momentum of the wheel around its center which, considering the hole system, rotates with a rotation speed that is the sum of the face rotational speed and the wheel apinning velocity itself. Therefore:

$$E_{face} = \frac{1}{2} I_{face} \dot{\theta}_{face}^2 \tag{2.1.12}$$

$$E_{wheel} = \frac{1}{2} I_{wheel} (\dot{\theta}_{wheel} + \dot{\theta}_{face})^2 \tag{2.1.13}$$

2.1.3 Dissipative Energies

The dissipation function represents the power lost to friction, so it is often a quadratic function of the generalized velocities. The Rayleigh dissipation function, is a function used to handle the effects of velocity-proportional frictional forces in Lagrangian mechanics [16]. It is defined for a system of N particles as:

$$R = \frac{1}{2} \sum_{i=1}^{N} (k_x v_x^2 + k_y v_y^2 + k_z v_z^2) \tag{2.1.14}$$

Therefore, we model the lost caused by the motion of the face and the wheel by:

$$R = \frac{1}{2} K_w \dot{\theta}_w^2 + \frac{1}{2} K_f \dot{\theta}_f^2 \tag{2.1.15}$$

The dissipative forces interaction within the system can be modeled by $\frac{\partial R}{\partial \dot{q}_i}$. Thus:

$$\frac{\partial R}{\partial \dot{\theta}_w} = K_w \dot{\theta}_w \tag{2.1.16}$$

$$\frac{\partial R}{\partial \dot{\theta}_f} = K_f \dot{\theta}_f \tag{2.1.17}$$

2.1.4 Lagrange Function

First, all the Inner moments have to be referenced to the same coordinate system. By implementing the Steiner's Theorem, the Inner moment of the wheel can be seen from the pivot point as [14]:

$$I_{wface} = I_{wheel} + m_w L^2 \tag{2.1.18}$$

Putting it all together, the Lagrange function is computed as:

$$L = -L_t m_t g \cos\theta_{face} + \frac{1}{2} I_{face}\dot{\theta}_{face}^2 + \frac{1}{2}(I_{wheel} + m_w L^2)(\dot{\theta}_{wheel} + \dot{\theta}_{face})^2 \tag{2.1.19}$$

The generalized momentas, derivated from the Lagrange function are:

$$M_{\theta_{face}} = \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}_{face}}\right) = I_{face}\ddot{\theta}_f + (I_{wheel} + m_w L^2)(\ddot{\theta}_w + \ddot{\theta}_{face}) \tag{2.1.20}$$

$$M_{\theta_{wheel}} = \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}_{wheel}}\right) = (I_{wheel} + m_w L^2)(\ddot{\theta}_f + \ddot{\theta}_w) \tag{2.1.21}$$

The partial derivative of the Lagrangian are given by the expression:

$$\frac{\partial L}{\partial \theta_f} = L_t m_t g \sin\theta_{face} \tag{2.1.22}$$

$$\frac{\partial L}{\partial \theta_w} = 0 \tag{2.1.23}$$

2.1.5 Motor Torque vs PWM

A Brushless DC Motor (BLDC) motor delivers a torque while turning which depends on the RPM and the power given to the motor. In order to modelise our system, the torque characterizarion was needed for a proper adjustment. The power profile and torque of a motor follows the folloring rules:

- The higher the spin velocity is, the lower the torque the motor is capable of delivering.
- The higher value of torque is found when working in intermediate velocities.

To measure the relation between both values and the angular velocity for our particular motor, an object with known inertia moment is attached to the motor axis. Therefore, a set of specific pulse width modulation (PWM) signals are used to comunicate to the motor and a full velocity profile is obtained. For this purpose, the CNY70 device, deeply analised on section 4, was used for measuring each spin of the wheel. The CNY70 consists of a IR-emitting diode and a phototransistor, if a surface absorbs the light, then a 5V signal will appear on the collector. Hence, four black stickers were placed around the item attached to the motor axis. On a first approach to characterization of the motor, the collector of the CNY70 was observed from the oscilloscope in the lab, as seen in 4.18. We could not get the data fast enough from the oscilloscope in the lab through the script shown in Appedix C, which ask the oscilloscope to send the frequency of the pulses seemed in the collector of the phototransistor, once it is stabilized, so the angular acceleration couldn't be taken from this measure.

For this reason, another solution had to be chosen. Audio signal processing programs, such as Audacity can record the voltage pulses created on the microphone through the magnets or condensers. For our purpose, through a coupling capacitor, the collector of the CNY70 phototransistor was plugged into the laptop's microphone jack for recording the rising edges on the collector. The purpose of the audio coupling capacitor is to block DC, allowing AC through it. Of course the ground had to be the same for both, the jack microphone input, and the CNY70. The schematic of the circuit implemented is shown in figure 2.2.

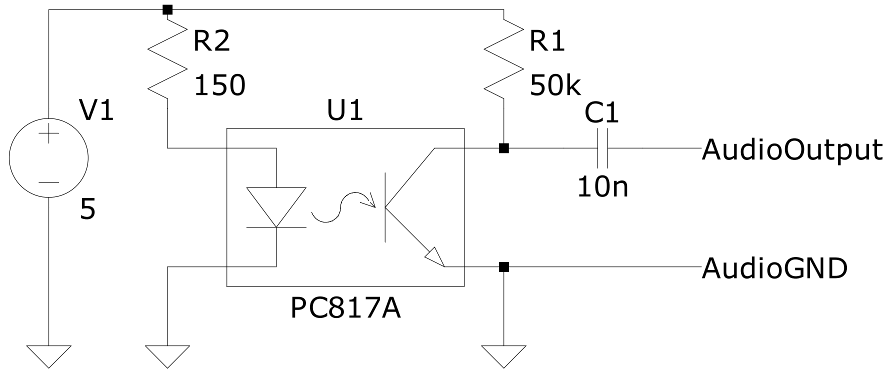


Figure 2.2 – *Schematic CNY70 Audio Reading.*

The implementation in the lab is shown in the following figure 2.3.

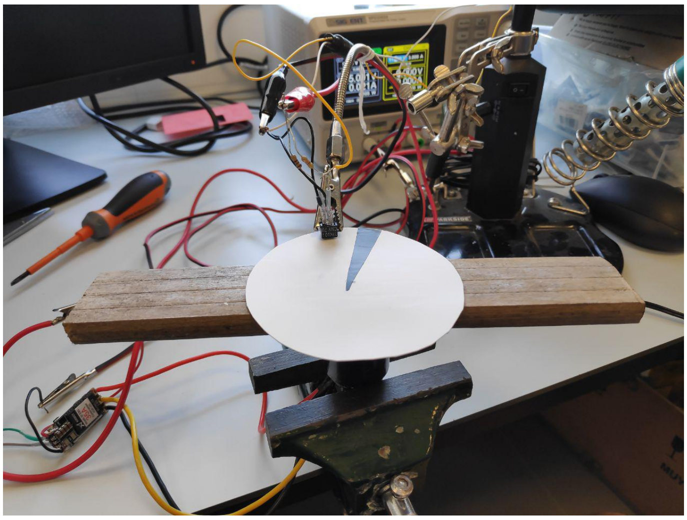
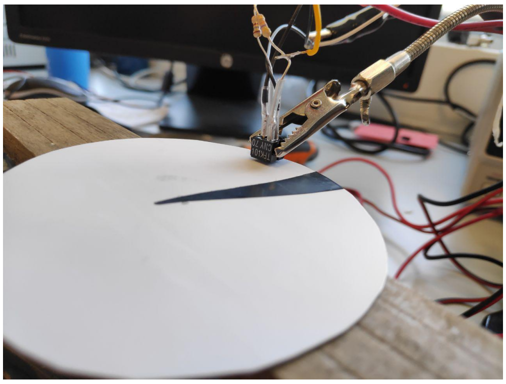


Figure 2.3 – *CNY70 implementation for reading through Audacity.*

Table 2.1 – *CNY70 implementation for reading through Audacity.*

The results read from Audacity are shown in figure 2.4. It can be seen that from the starting point, where the angular velocity is 0, the frequency of the pulses get higher as time goes by, caused by the angular acceleration of the spinning object.

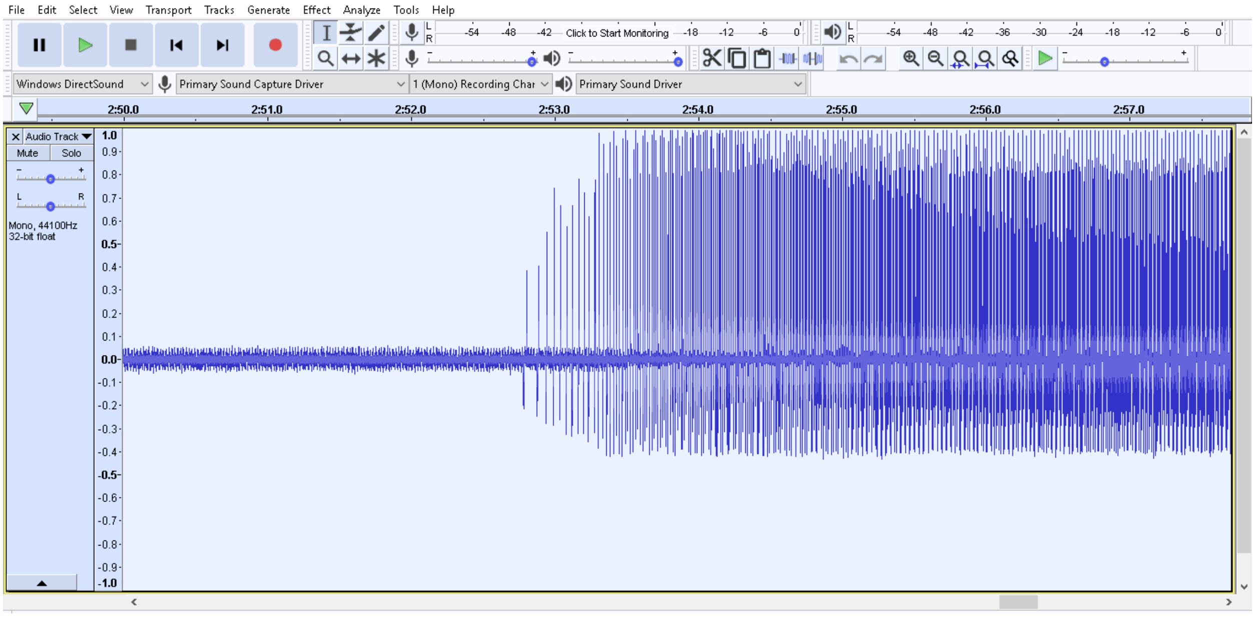


Figure 2.4 – *Signal read in Audacity.*

An algorithm is then created in Matlab to extract the time for each signal peak in order to perform the analysis. The signal can therefore be seen clearer in order to get the acceleration of the object caused by the motor's torque.

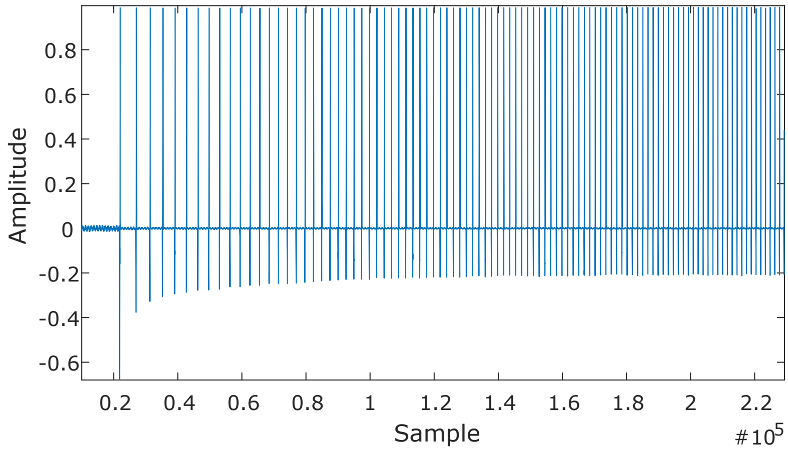


Figure 2.5 – *Signal proccessed in Matlab.*

The velocity profile presents different zones: a first part where the acceleration is linear with time, which comprises the lowest 80% portion of the total velocity range, and a second part where the acceleration decreases until the motor reaches a saturated state. It can be seen in Figure 2.6, where the results shown in Figure 2.5 have been further proccessed to get the velocity profile.

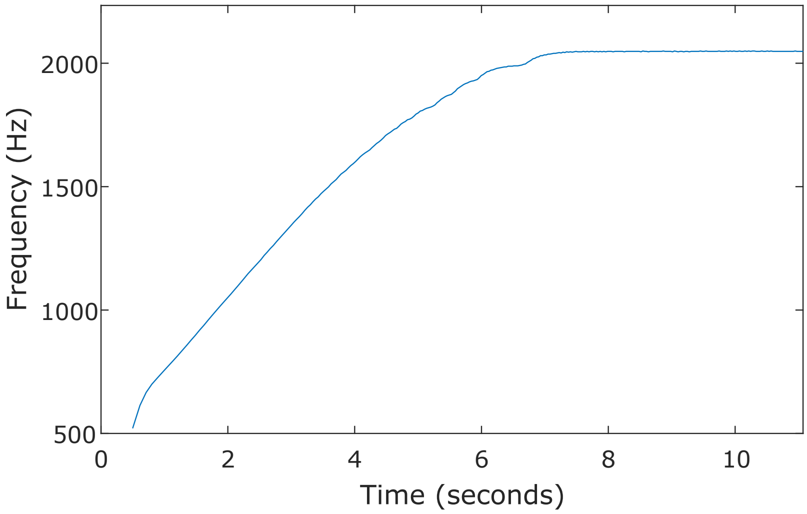


Figure 2.6 – *Velocity profile.*

Given our specific control purpose, the analysis of the almost-saturated and saturated regimes will be neglected, and the study will be focused in the linear behavour parts.

The same measurement is made for different values of the PWM Arduino control signal for a 11.1 volts power supply to the BLDC 3 phase motor, and from the slope of the linear part and the inertia moment of the rotating object, the torque can be obtained:

$$\tau = I\frac{d\omega}{dt} \tag{2.1.24}$$

As sais before, for our purpose, only the slopes of the linear part, at the beggining of the velocity profile will be consider. The object we placed on the motor axis was a wood stick, so the the inner moment is the same as for a solid cuboid rotating around the heigth axis, given by [17]:

$$I_h = \frac{1}{12}m(w^2 + d^2)(kg * m^2) \qquad (2.1.25)$$

Where m is the mass of the cuboid; w, the width and d, the depth.

The results when analizing the differents slopes is shown in Table 2.2.

PWM Input (%)	110	115	120	125	130
Slope (rad)	3737.49	3960.54	4117.36	4322.82	4490.55
Torque $(kg \cdot m/s^2)$	1.87	1.98	2.06	2.16	2.25

Table 2.2 – *Torque values in relation PWM control signal*

We can now stablish a lineal dependence between the PWM control signal and the Torque given to the system by the motor. In Figure 2.7 the linear trend can easily be seen, given by:

$$Tm(PWM) = 0.0187PWM - 0.1792 \qquad (2.1.26)$$

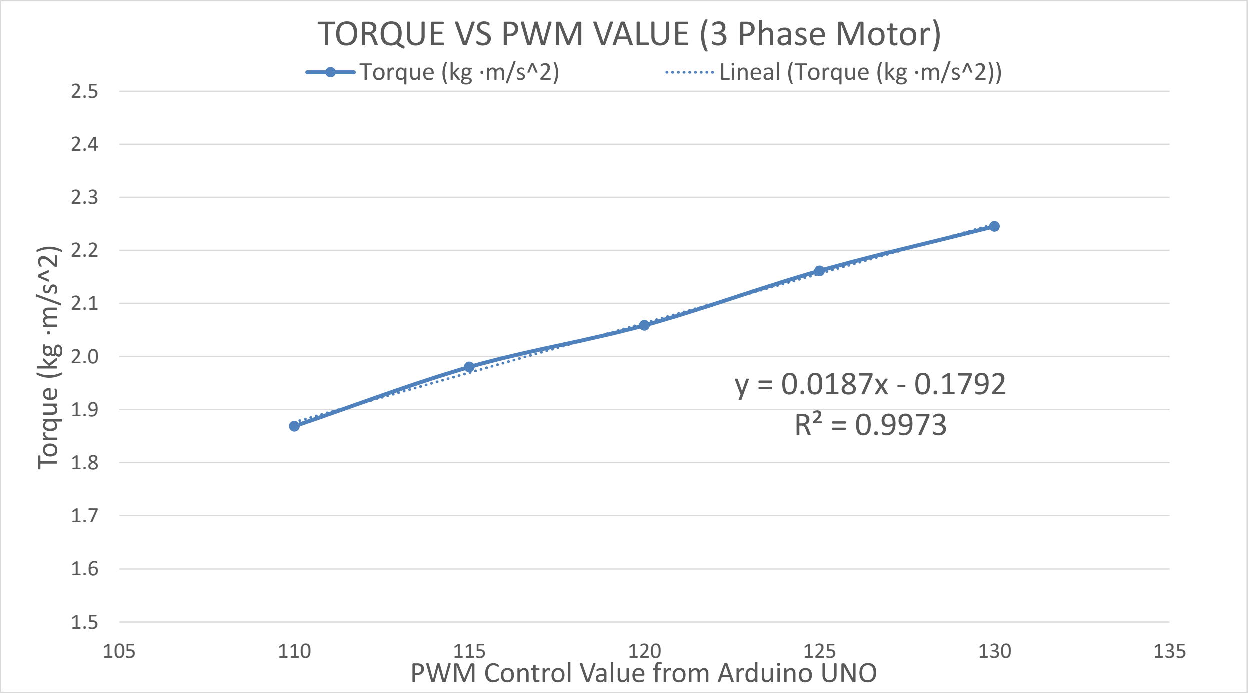


Figure 2.7 – *Torque values obtained for a PWM sweep range (3 Phase Motor).*

2.1.6 System's Dynamics

Using Euler-Lagrange equations 2.1.6, the motion equation for each coordinate in our system can be seen as:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}_w}\right) - \frac{\partial L}{\partial \theta_w} = \tau_{\theta_w} - \frac{\partial R}{\partial \dot{\theta}_w} \tag{2.1.27}$$

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}_f}\right) - \frac{\partial L}{\partial \theta_f} = \tau_{\theta_f} - \frac{\partial R}{\partial \dot{\theta}_f} \tag{2.1.28}$$

By combining Euler-Lagrange equations, the generalised momenta, dissipative energies, and the motor torque, the equations of motion of the system on the θ_w coordinate can be seen as:

$$(I_{wheel} + m_w L^2)(\ddot{\theta}_f + \ddot{\theta}_w) + K_w \dot{\theta}_w = T_m \tag{2.1.29}$$

When focusing on the θ_f coordinate:

$$I_{face}\ddot{\theta}_f + (I_{wheel} + m_w L^2)(\ddot{\theta}_w + \ddot{\theta}_f) - L_t m_t g \sin\theta_f + K_f \dot{\theta}_f = 0 \tag{2.1.30}$$

Reflecting the variables $\ddot{\theta}_f$ and $\ddot{\theta}_w$, interesting for the State Space Model 2.2 the dynamics of the face of the cube can be described by the equation 2.1.32 [11].

$$\ddot{\theta}_b = \frac{(m_t L_t) g \sin\theta_f - T_m - K_f \dot{\theta}_f + K_w \dot{\theta}_w}{I_b + m_w L^2}, \tag{2.1.31}$$

$$\ddot{\theta}_w = \frac{(I_f + I_w + m_w L^2)(T_m - K_w \dot{\theta}_w)}{I_w (I_f + m_w L^2)} - \frac{m_t L_t g \sin\theta_f - K_f \dot{\theta}_f}{(I_f + m_w L^2)} \tag{2.1.32}$$

This two equations will be used when setting the control algorithm, where the plant of the system is need in order to stabilize it, getting the desired response. Also, during the Mechanical Features section 2.3, this phisical approach to the system will be used to stablish the constrains in terms of designing.

2.2 State Space Model

A state space model can be derived form the equations of motion, 2.1.32 and 2.1.31. Therefore, we consider the standarized Controllable Canonical Form. Given a system transfer function, it can be obteined a canonical model, useful for the pole placement controller design technique. The observable canonical form is defined in terms of the transfer function coefficients as follows: [18]

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ . \\ . \\ . \\ \dot{x_n} \end{bmatrix} = \begin{bmatrix} 0 & 0 & . & . & -a_n \\ 1 & 0 & . & . & -a_n \\ . & . & . & . & . \\ . & . & . & . & . \\ . & . & . & . & . \\ 0 & 0 & . & . & -a_1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ . \\ . \\ . \\ x_n \end{bmatrix} + \begin{bmatrix} b_n - a_n b_0 \\ b_{n-1} - a_{n-1} b_1 \\ . \\ . \\ . \\ b_1 - a_1 b_0 \end{bmatrix} u \qquad (2.2.1)$$

$$y = \begin{bmatrix} 00...01 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ . \\ . \\ . \\ x_n \end{bmatrix} + b_0 u \qquad (2.2.2)$$

Therefore, our system can be described by a observable canonical form, useful when designing the control plan, where:

$$\dot{x} = Ax + Bu \qquad (2.2.3)$$

$$y = Cx + Du \qquad (2.2.4)$$

Where $\dot{x}$ represents the derivated angular velocity of the wheel and the face, considering the equations of motion; and y represents the observable parameters read from the sensors which will be explained later on. Thus, the system can be seen as [11]:

$$x = (\theta_b, \dot{\theta}_b, \dot{\theta}_w) \qquad (2.2.5)$$

$$A = \begin{bmatrix} 0 & 1 & 0 \\ \frac{(m_t L_t)g}{I_f + m_w L^2} & -\frac{K_f}{I_f + m_w L^2} & -\frac{K_w}{(I_f + m_w L^2)}(m_w L^2) \\ -\frac{(m_t L_t)g}{I_f + m_w l^2} & \frac{K_f}{I_f + m_w L^2} & -\frac{K_w((I_f + I_w + m_w L^2)}{I_w I_f + m_w L^2} \end{bmatrix} \qquad (2.2.6)$$

$$B = \begin{bmatrix} 0 \\ -\frac{K_m}{I_f + m_w L^2} \\ \frac{K_m(I_b + I_w + m_w L^2)}{I_w(I_f + m_w L^2)} \end{bmatrix} \tag{2.2.7}$$

For linearizing the equations of motion, the equilibrium point where $(\theta_f, \dot{\theta}_f, \dot{\theta}_w) = (0, 0, 0)$ is considered, therefore:

$$\sin\theta_f \approx \theta_f \tag{2.2.8}$$

In terms of the observable parameters, the angle of the face θ_b; and the angular velocity of the face, $\dot{\theta}_b$ and the wheel, $\dot{\theta}_w$, are considered perfectly meassured.

$$C = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} \tag{2.2.9}$$

$$D = \begin{bmatrix} 0 \end{bmatrix} \tag{2.2.10}$$

Thus, the system's reading on the variables will be:

$$y = (\theta_b, \dot{\theta}_b, \dot{\theta}_w) \tag{2.2.11}$$

In the section devoted to the PID control algorithm design, this equation will be the set base for solving the stabilitation of the system.

2.3 Mechanical Features & Constraints

When the face is laying on one of the edges (resting position), there is no angular movement in any of the objects within the body, meaning the inner wheel and the face itself. Therefore, by using the equation of dynamics shown in 2.1.31 and 2.1.32, assuming that there's no movement:

$$(\dot{\theta}_f, \dot{\theta}_w) \approx (0, 0) \tag{2.3.1}$$

The dissipative forces, due to the friction, approach zero. The dynamics equations for the face laying on one of the edges are then:

$$\ddot{\theta} f(I_{face} + I_{wheel}) = L_t m_t g \sin\theta_f - T_m \tag{2.3.2}$$

When the face hasn't moved and it's laying on the corner, the torque is then the full responsible of moving the face against gravitational force:

$$T_m = L_t m_t g \tag{2.3.3}$$

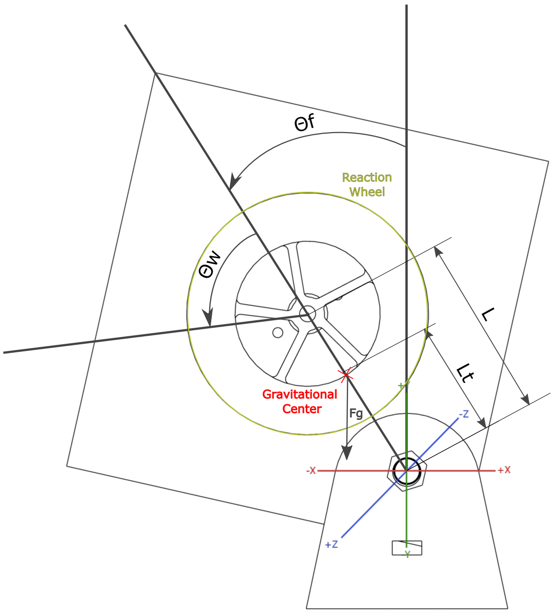


Figure 2.8 – *2D Model in rest position.*

Also, when the face is balancing around the desired equilibrium point, $\dot{\theta}_f, \dot{\theta}_w$ should be close to zero, making the system stable. The equations of motion will be then the same as when the face is laying on one of the edges, and the motor torque will be then again the responsible of movement. Consequently a gentle and smooth torque will be needed when the face is balancing around zero and a strong torque will be needed when the face is laying on the edge.

The aim of the mechanical design is to make the torque needed to move the face the smallest possible: assuring that the power consumption will be small, the control algorithm fast and strong enough and the motor capable of moving the system. As shown in equation 2.3.2, the effect of gravitational force on our system is multiplied by the mass of the whole system and by the distance between the bearing point and the gravity centre of the system. Therefore the system is required to be light and the center of gravity as closer to the bearing point as possible. Making possible to counter, by using the motor torque, the force applied by the gravity on our system.

$$T_m = L_t m_t g - \ddot{\theta}f(I_{face} + I_{wheel}) \sin\theta_f \tag{2.3.4}$$

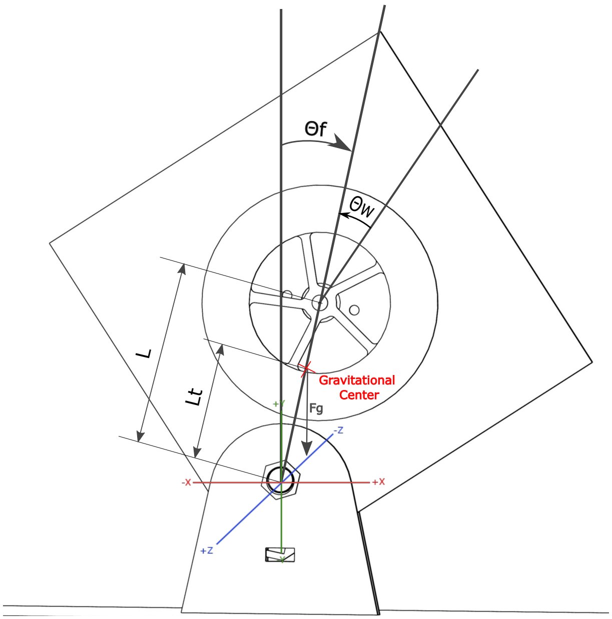


Figure 2.9 – *2D Model balancing around target point.*

In terms of the inner wheel, regarding the equation 2.3.4,once the initial torque given by the motor has counted the gravitational force, $\ddot{\theta}_{face}$ will be different to zero. The inner moment of the wheel (I_w) multiplies the angular acceleration of the face which decreases itself the effect of the gravitational force when the face is balancing up. Therefore, the inner moment should be big enough to make the torque needed smaller, the power consumption lower, and the softness of the control higher.

Assuming that the inner wheel desired will consist of a cylinder turning around its centre, the inner moment is given by the expression [14]:

$$I = \frac{1}{2}m(r_2^2 + r_1^2) \tag{2.3.5}$$

Where M is the mass of the wheel and r_1, r_2 can be seen on figure 2.10:

Given the characteristics of our system, the parameter that can make the torque needed smaller is the reaction wheel inner moment. As said before, the weight of the system should be the lighter possible, making the gravitational force acting on the system small. Taking into account the equation 2.3.5, the size of the inner wheel should be as big as possible and the weight relevant but light enough not to make the body weight too high.

The position of the motor within the system is another parameter to be discussed. If the motor is placed to high from the bearing point on the one face prototype, the center of gravity will be then moved up, making the gravitacional torque higher. On the other hand, the size of the inner is constrained by the position of the bearing ring on the face, so the higher the motor is, the bigger the inner wheel can be.

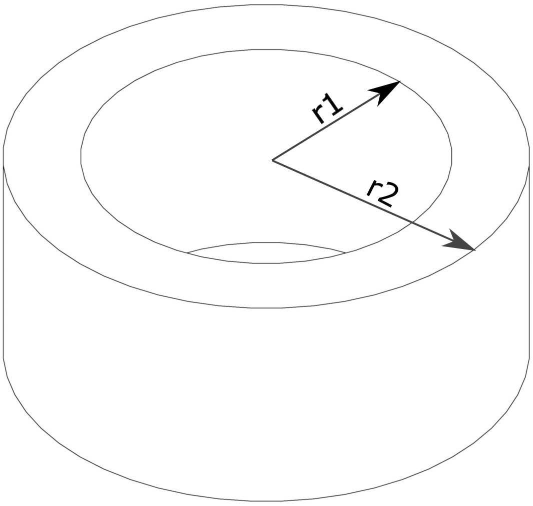


Figure 2.10 – *Cylinder Measures.*

2.4 PID Control Algorithm

Once we have the dynamics equations of the system, shown in section 2.1, the stability of the system has to be probed. The State Space Representation 2.2 of our system described by a observable canonical form, is given by:

$$\dot{x} = Ax + Bu \tag{2.4.1}$$

$$y = Cx + Du \tag{2.4.2}$$

$$A = \begin{bmatrix} 0 & 1 & 0 \\ \frac{(m_t L_t)g}{I_f + m_w L^2} & -\frac{K_f}{I_f + m_w L^2} & -\frac{K_w}{(I_f + m_w L^2)}(m_w L^2) \\ -\frac{(m_t L_t)g}{I_f + m_w l^2} & \frac{K_f}{I_f + m_w L^2} & -\frac{K_w((I_f + I_w + m_w L^2)}{I_w I_f + m_w L^2} \end{bmatrix} \tag{2.4.3}$$

$$B = \begin{bmatrix} 0 \\ -\frac{K_m}{I_f + m_w L^2} \\ \frac{K_m(I_b + I_w + m_w L^2)}{I_w(I_f + m_w L^2)} \end{bmatrix} \tag{2.4.4}$$

For linearizing the equations of motion, the equilibrium point where $(\theta_f, \dot{\theta}_f, \dot{\theta}_w) = (0, 0, 0)$ is considered, therefore:

$$\sin\theta_f \approx \theta_f \tag{2.4.5}$$

In terms of the observable parameters, the angle of the face θ_b; and the angular velocity of the face, $\dot{\theta}_b$ and the wheel, $\dot{\theta}_w$, are considered perfectly meassured.

$$C = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} \tag{2.4.6}$$

$$D = \begin{bmatrix} 0 \end{bmatrix} \tag{2.4.7}$$

A Matlab code was developed in order to get the transfer function of the system. The values of the inner moment, frictional forces etc... are taken from the system explained in [11]. [NUM, DEN] = ss2tf(A,B,C,D) creates an object [NUM, DEN] representing the transfer function of the continuous-time state-space model. From [NUM, DEN] we can get the simplify transfer function ”sys” of the system by sys = tf (NUM, DEN), which construct transfer function or convert to transfer function. Hence:

```
clc;
l = 0.085;
lb = 0.075;
mb = 0.419;
mw = 0.204;
Ib = 3.34*(10^-3);
```

```
Iw = 0.57*(10^-3);
Cb = 1.02*(10^-3);
Cw = 0.05*(10^-3);
g = 9.81;
Km = 0.0251;
A = [0 1 0; ((mb*lb + mw*l)*g)/(Ib +(mw*(l^2))) -Cb/(Ib + mw*(l
    ^2)) Cw/(Ib + mw*(l^2)); -((mb*lb + mw*l)*g)/(Ib +(mw*(l^2))
    ) -Cb/(Ib+(mw*(l^2))) -Cw*(Ib+Iw+(mw*(l^2)))/(Iw*(Ib + (mw*(
    l^2))))];
B = [0; -Km/(Ib+(mw*(l^2))); Km*(Ib+Iw+(mw*(l^2)))/(Iw*(Ib + (
    mw*(l^2))))];
C=[1 1 1];
D=[0];
[NUM, DEN]=ss2tf(A,B,C,D)
sys = tf (NUM, DEN)
```

The transfer function for the values shown above, in Laplace domain is therefore given by:

$$G(s) = \frac{44.04s^2 + 6.326s - 4376}{s^3 + 0.31s^2 - 99.35s - 8.717} \tag{2.4.8}$$

The Pole-Zero Map of the open loop system G(s), where no feedback and no control are applied, is represented by Matlab's pzplot() function, which shows the Pole-Zero map given the transfer function. The result is shown in figure 2.11, where as suppossed there are poles in the positive part of the Real Axis, meaning that the system without any control will be unstable. There is no way the system can get to the target point if no feedback of the angular position of the body is given. Hence, it is obvious that a control algorithm is needed in order to stabilize the system.

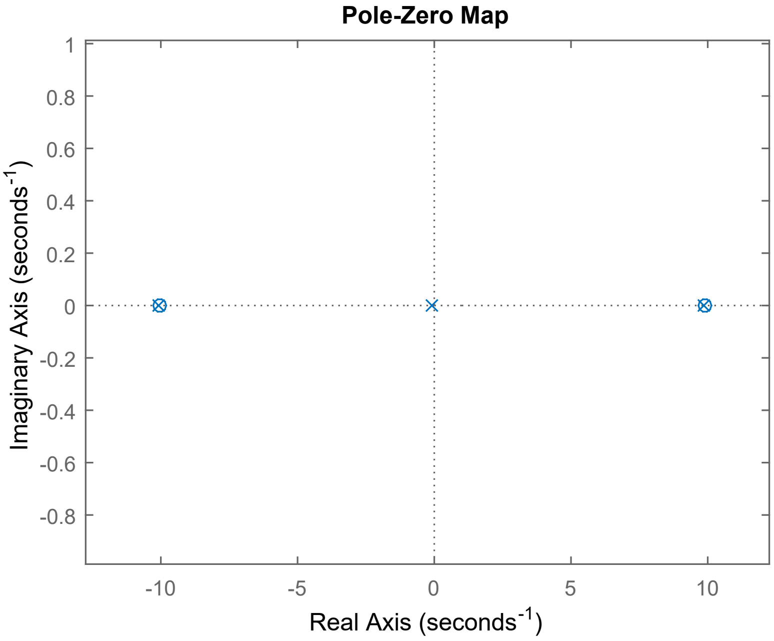


Figure 2.11 – *Pole-Zero map of the system's plant.*

"A system is controllable at a time t_0 *if it is possible to transfer through the use of a control vector without restrictions to the system from the initial state* x_{t_0} *to any other state in a finite interval of time."*[19]. If the output is the variable to be controlled, then a S matrix is defined $S == [B; AB; A^2B]$, for which it must be fulfilled that the range must be equal to m; the number of output variables, so the system is controllable. Matlab's function ctrb(sys), "calculates the controllability matrix of the state-space LTI object given by $\dot{x} = Ax + Bu$". Therefore, the code implemented in Matlab for knowing the controllability of the system is:

```
clc;
[NUM, DEN]=ss2tf(A,B,C,D)
sys = tf (NUM, DEN)
S=ctrb(sys);
rank(S)
```

The answer to rank(S) gave 3, so the rank is equal to the number of output variables, $(\theta_f, \dot{\theta}_f, \dot{\theta}_w)$.

For estabilizing the system, once it is known that the system is controllable, a PID control approach will be used. Probably, in future prototypes a LQR as well. The transfer function of a PID control algorithm is given by [20]:

$$u(t) = K_p \left[e(t) + \frac{1}{T_i} \int e(t)dt + T_d \frac{de(t)}{dt} \right] \tag{2.4.9}$$

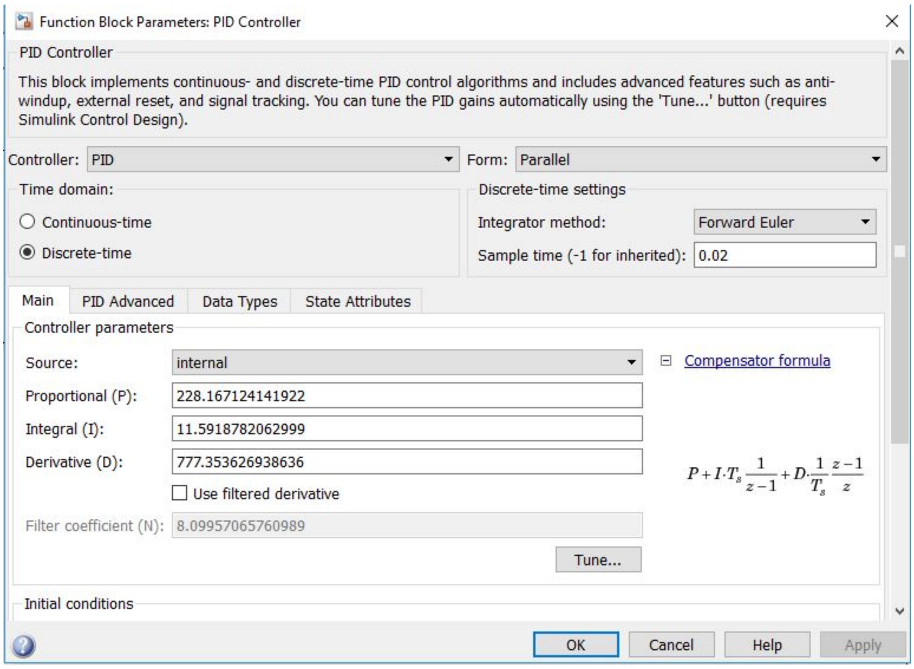
"Where u(t) is control input to the plant model, e(t) is error which is difference between actual output (y(t)) and reference input (r(t)), Kp is proportional gain, Ti is integral time constant and Td is derivative time constant."[20].

For our purpose, the control that will be implemented is actually digital, the system variables won't be read continiously, but discretely. The sampling time of the digital controller,T_i, implemented in Arduino UNO, furthered detailed in section 5.3, will be able to develop a sampling time of 20ms. The digital approach to the PID algorithm is given by [21]:

$$u(n) = K_P \left(e(n) + \frac{1}{T_i} \sum_{k=0}^{n} e(k) \cdot T + T_d \left(\frac{e(n) - e(n-1)}{T} \right) \right) + u_0 \tag{2.4.10}$$

A simulink model of the system to be implemented, with the PID algorithm in discrete time is developed. This model will be useful in terms of the further determination of the PID parameters for our system, once the PCB model shown in section 3.3 is fully developed. The PID block used is set to be discrete, with a sampling time of 20ms, which will correspond to the actual sampling time. As seen in figure 2.12, the compensator formula make use of the Backward Euler aproximation [22], for the Z transform for the PID control:

$$\frac{1}{s} \rightarrow \frac{T_S}{z-1} \tag{2.4.11}$$



Function Block Parameters: PID Controller

PID Controller

This block implements continuous- and discrete-time PID control algorithms and includes advanced features such as anti-windup, external reset, and signal tracking. You can tune the PID gains automatically using the 'Tune...' button (requires Simulink Control Design).

Controller: PID ▾ Form: Parallel ▾

Time domain:

○ Continuous-time

● Discrete-time

Discrete-time settings

Integrator method: Forward Euler ▾

Sample time (-1 for inherited): 0.02

Main | PID Advanced | Data Types | State Attributes

Controller parameters

Source: internal ▾

Proportional (P): 228.167124141922

Integral (I): 11.5918782062999

Derivative (D): 777.353626938636

□ Use filtered derivative

Filter coefficient (N): 8.09957065760989

Compensator formula

$$P+I{\cdot}T_s\frac{1}{z-1}+D{\cdot}\frac{1}{T_s}\frac{z-1}{z}$$

Initial conditions

Figure 2.12 – *Simulink PID Block parameters.*

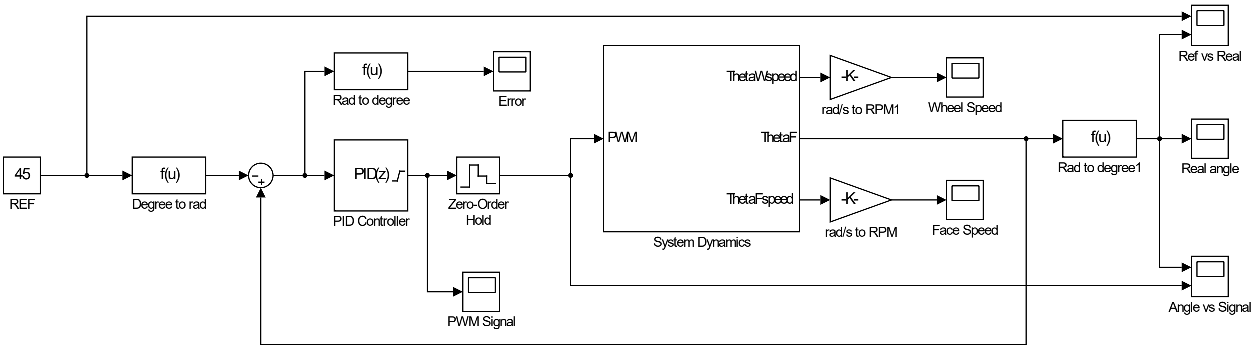


Figure 2.13 – *System's simulink model*

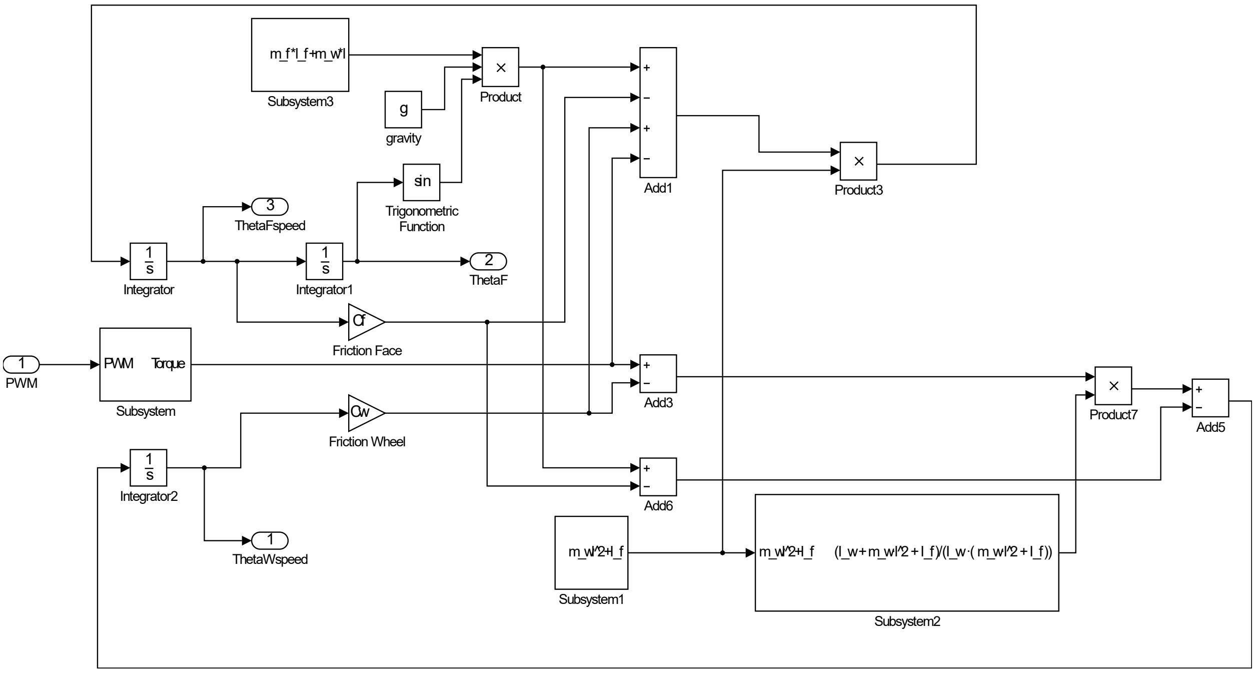


Figure 2.14 – *System's Plant simulink model.*

The PWM Signal entering the plant dynamics block, is acctually the control signal created by the Ardunino, see section 5.2, which has to be converted to the actual torque given to the system by the motor.

In section 5.2, devoted to the development the software for controlling the motor, we get to the conclusion that the motor is controlled through a PWM created through the library Servo.h [23]. Whenever a pulse width is needed, the funcion ESC.write(PWMValue) can be used. From 0 to 90 PWMValue the sense of rotation will be Clockwise (CCW) and from 90 to 180, Anticlockwise (ACW). This comes from the explanation shown in figure 5.4, always regarding the throttle set in the BLHELI firmware update, figure 5.3. Therefore, we can modelise our Torque ouput, regarding the signal signal created from the Arduino UNO 4.1.

The discrete PID block's output is also limited between 90, meaning full power in ACW rotation; and -90 meaning full power in CCW mode.

Therefore, as seen in section 2.1.5, devoted to the caracterisation of the motor, the relation between the PWM signal created through the funcion ESC.write(PWMValue) and the actual torque is given by:

$$Tm(PWM) = 0.0187PWM - 0.1792 \tag{2.4.12}$$

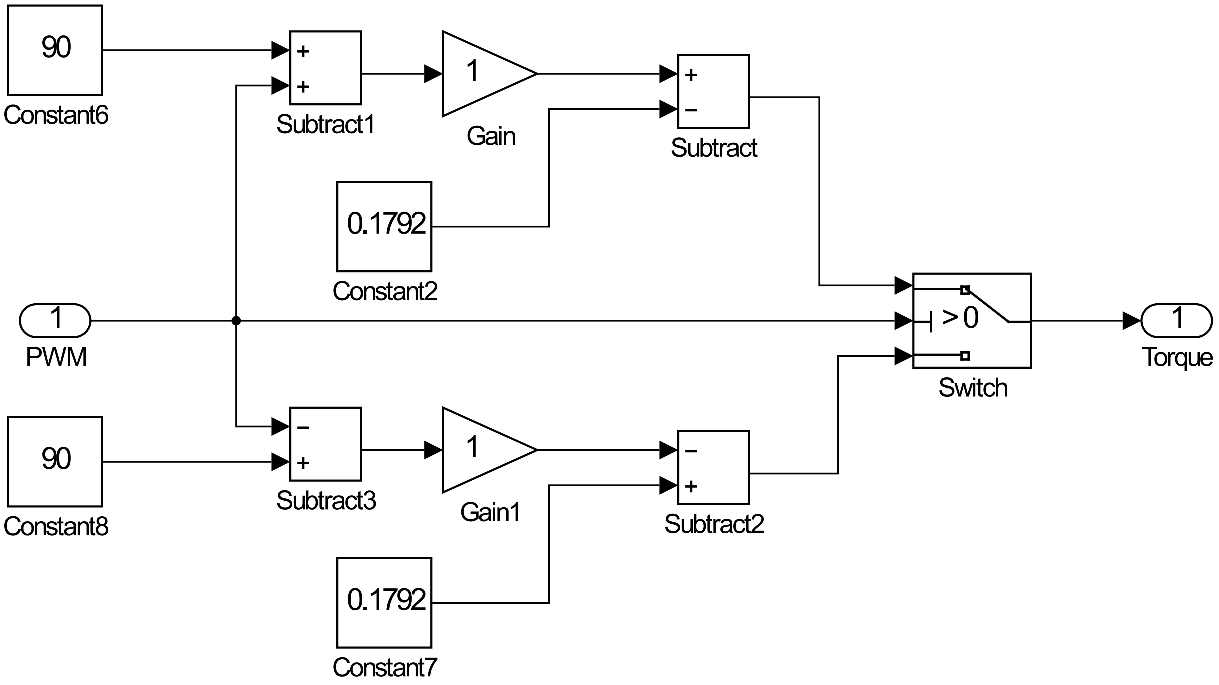
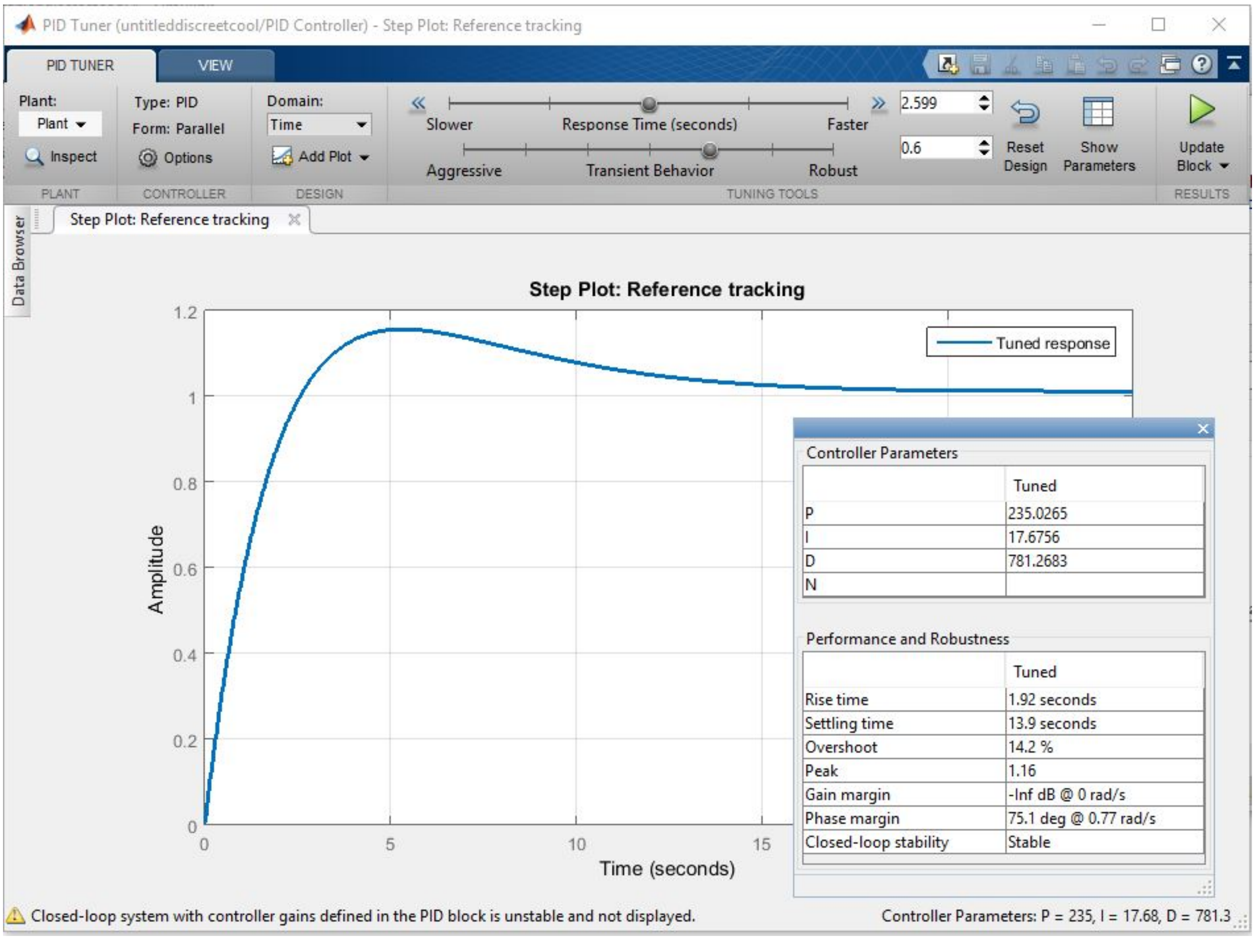


Figure 2.15 – *Torque vs PWM signal.*

For getting the correct tunning parameters, the PID Tunner tool avaliable in Simulink was used. It allows to tune the PID, visually checking parameters like overshoot or rise time. For the values of the system described in [11], the results of the tune through the PID Tunner are shown in the following figures:

Rise time	1.92 seconds
Settling time	13.9 seconds
Overshoot	14.20%
Phase Margin	75.1 de
Closed-loop stability	Stable

Table 2.3 – *Response obtained for the PID parameters set with PID Tuning tool.*



Controller Parameters

	Tuned
P	235.0265
I	17.6756
D	781.2683
N	

Performance and Robustness

	Tuned
Rise time	1.92 seconds
Settling time	13.9 seconds
Overshoot	14.2 %
Peak	1.16
Gain margin	-Inf dB @ 0 rad/s
Phase margin	75.1 deg @ 0.77 rad/s
Closed-loop stability	Stable

⚠ Closed-loop system with controller gains defined in the PID block is unstable and not displayed.

Controller Parameters: P = 235, I = 17.68, D = 781.3

Figure 2.16 – *Torque vs PWM signal.*

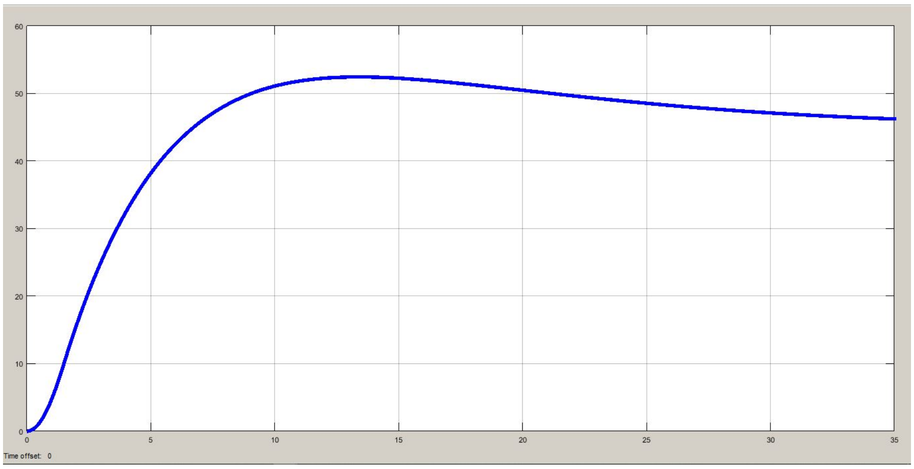


Figure 2.17 – *Response of the system.*

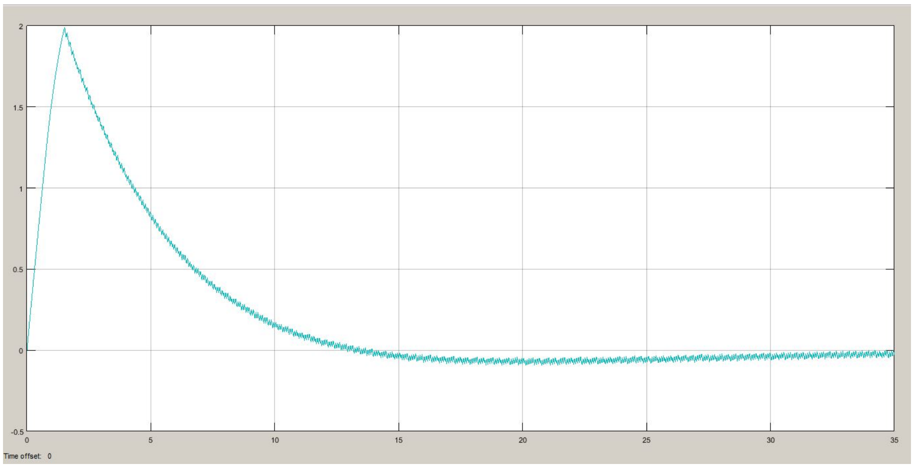


Figure 2.18 – *Angular velocity of the system.*

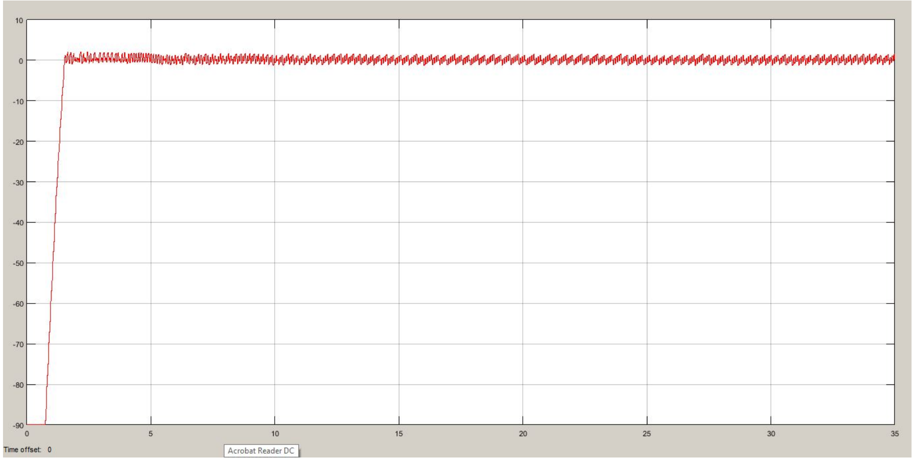


Figure 2.19 – *Signal generated by the PID discrete controller.*

Chapter 3

Mechanical Design

For a better comprenhension of the mechanical restriction and features, regarding the results shown in 2.3, a few prototypes were built.

Making possible the development of a successful mechanical design, a deep understanding of 3D modelling programs was needed. The program chose for this purpose was SolidWorks due to it's simulation possibilities. Many hours were spent in learning all the functionalities of this program, allowing at the end a good understanding of its functionalities and capabilities.

As a first approach to the behaving of the dynamics, the prototype made in first place was a thick 3D printed 2D body, that as said in section 2.1 can be used to model the dynamics of the system. It was meant to be heavy making control hard and the torque needed high. Also, due to the position of the motor and the size of the face, the constriction caused by the bearing point made the size of the inner wheel small. Consequently, the inner moment of the wheel wasn't big enough.

For the following iteration of the mechanical design development, a body face in steel will be developed due to the high young module (70GPa), and the relatively low density. Holes will be placed all along the face in order to make it lighter, but strong enough to withstand the mechanical efforts exerted by the system. Once the CNC coding an the milling machine working method was understood, the decission made was to use a already developed by GranaSAT model of a CubeSat.

The final prototype will therefore consists on PCB that can be fitted in this CubeSat. The measures were taken from the already existing cube and all the components were fitted in it. The design took a long time, due to the constrains and requirements.

3.1 Plastic Prototype

For the realisation of the first prototype, a really thick plastic body was created in order to have a heavy face to move. We wanted to confirm that our motor had enough torque to move the heavy body created, with no holes on it.

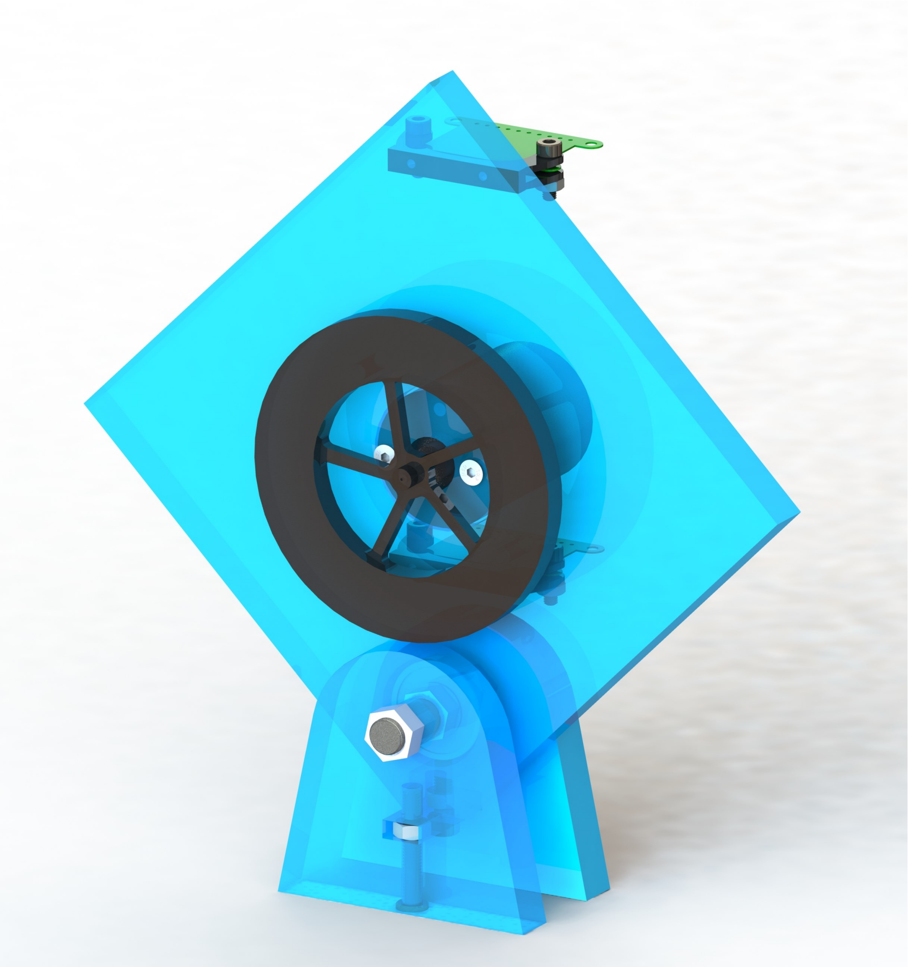


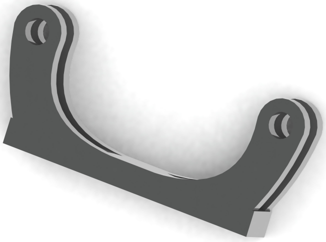
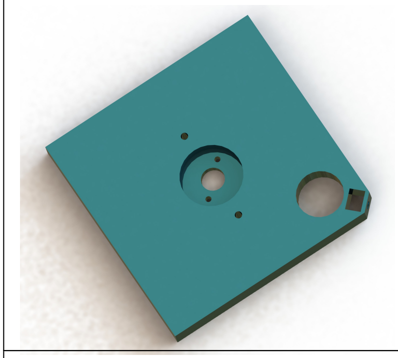
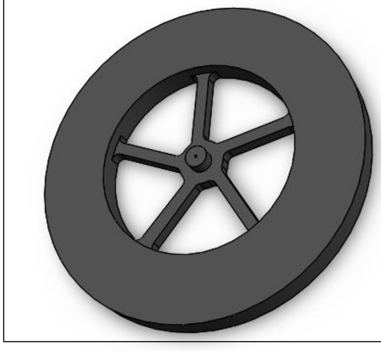
Figure 3.1 – *2D Plastic Prototype*

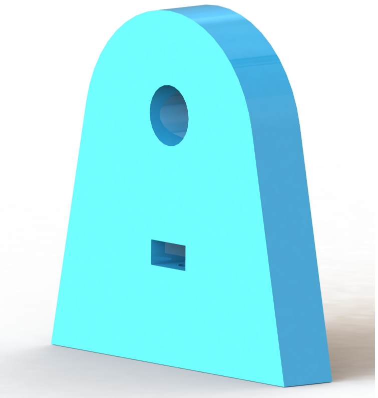
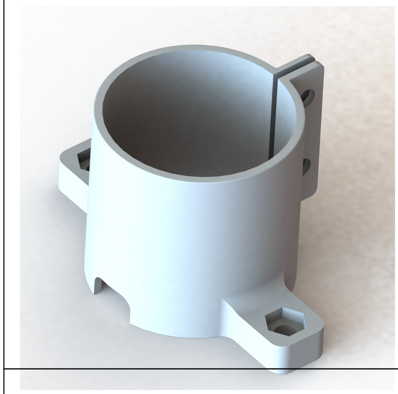
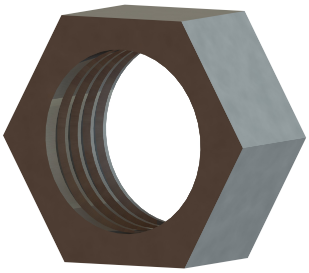
The prototype was made out of off the shield components. For that reason, and being a first approach to the problem, the motor used was a DC motor found in the lab. The characteristics of the motor will be expose in chapter 4.2.1 . In order to know the weight and position of the components within the prototype, 3D models of all the components which are part of the future assembly were made.

It was printed and assembled in the lab, so it could serve its purpose.

The components needed for modeling this prototype and further print it can be seen in Table 3.1 on the next page.

Table 3.1 – *3D Components Plastic Body*

Render 3D Model	Description
	- BLDC Motor 3D Model.
	- 3D designed holder for the Adafruit 10DOF IMU.
	- Plastic Face designed so the DC motor and the bearing ring could fit in.
	- 3D designed Inner Wheel, afterwards printed with high density, so the mass could be big enogh.

	• 3D designed mount.
	• 3D printed holder for the DC motor.
	• M8 Nut.
	• M8 Rod.

3.2 Aluminium Prototype

Trying to make the system look alike to the final iteration of it, a aluminium cube face CNC code was developed. For this purpose, a CNC machine avaliable in the lab was used. A few tries on the CNC machine were needed in order to understand the working method: commands, speed and feed rate. It was the first time using a CNC machine, therefore, the working mode and the coding was unknown. It took some time to understand its working principle.

3.2.1 Calculation for Spindle Speed and Feed Rate

When milling or drilling, creating the path needed to make our piece, feed rate and spindle speed must be determined. Materials have rated surface speed depending on the broach, the harder the material the slower the speed. We consider the parameters shown in the table bellow, specified for our cutters. The material used is an aluminium alloy, short chipping and no cover considered. The rest of the table, for other materials is specified in Appendix A.

Material	Rm/UTS (N/mm^2)	Cutting speed Vc (m/min)		Feed per tooth (fz cutter ø)			
		coated	uncoated	2 - 4	5 - 10	11 - 16	>16
Aluminum alloys, short chipping	-400	300	240	0,030	0,060	0,10	0,15

Table 3.2 – *Cutting data recommendations. Source: JHV Tools B.V. [10].*

Accordig to the manufacturer [10], the calculations to be made are:

$$n = \frac{Vc * 100}{\pi * d_1}(rpm) \tag{3.2.1}$$

$$V_f = f_z * 3 * n(mm/min.) \tag{3.2.2}$$

Where $n = SpindleSpeed$, $V_f = FeedRate$, $V_c = CuttingSpeed$, $d_1 = Diameter(mm)$, $f_z = Feedpertooth(mm)$, $z = Noofteeth$.

4mm, 3mm and 2mm diameter drills are used as milling cutters. Therefore:

Tool	Diameter ø (mm)	Spindle Speed (rpm)	Feed Rate (mm/min.)
Drill	4	$n = \frac{800*300}{\pi*4} = 19099$	$V_f = 0.03 * 3 * 19099 = 573$
Drill	3	$n = \frac{800*300}{\pi*3} = 25465$	$V_f = 0.03 * 1 * 25465 = 763$
Drill	2	$n = \frac{800*300}{\pi*2} = 38197$	$V_f = 0.03 * 1 * 38197 = 1146$

In order to make it look alike to the real and final cube that will be developed, and condering the restrictions found in the firs prototype, a code was developed for making a aluminium cube face.The final look of this face prototype would be really similar to the plastic cube face shown in figure 3.1, solving the problems related to the size of the Inner Wheel due to the position of the inner wheel.

The code developed for making the face of cube isn't shown, because of the large number of lines, but through the simulation mode on the CNC402 SW, used for controlling the CNC machine, we can get the path of the cutting. The simulation mode of CNC402 shows the paths in blue and the drilling in red. The drilling will be used for the holes to hold the motor, and its holder, in case it's not enough with the screws. In terms of the path, as shown in the code, each path and trajectory had it's own purpose, basically for engraving the motor inside the aluminin so it doesn't move; making the hole for the bearing ring and the screw used to fix it so it doesn't move while balancing.

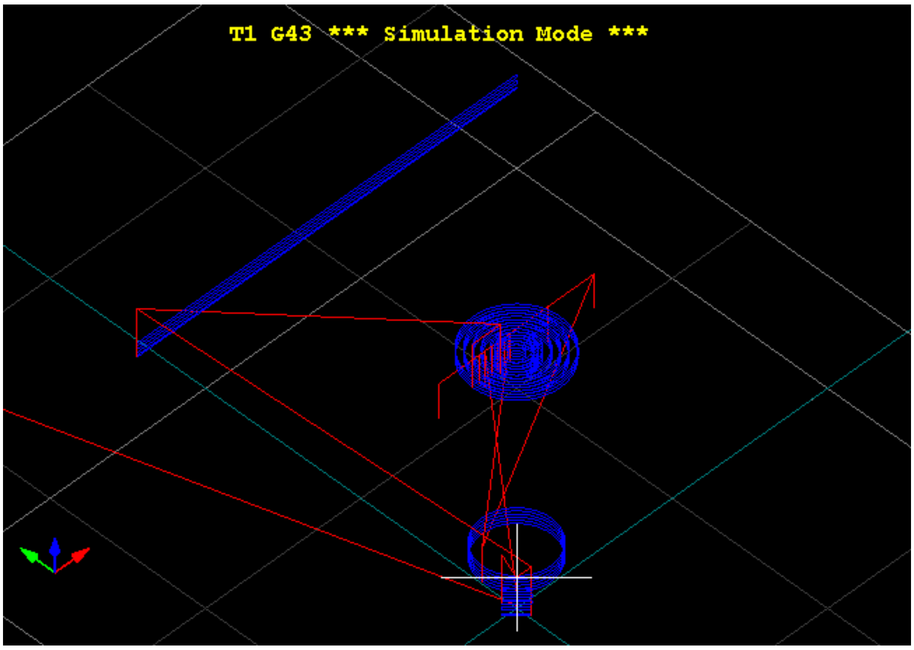


Figure 3.2 – *2D Aluminium Face Designed*

3.3 CubeSat Momentum Board

In this last iteration of our prototype a CubeSat aluminium model, developed by GranaSAT, was used. The model considers the 10x10x10cm measures standarized for CubeSats and was developed by the CNC machine avaliable in the lab, it can be seen in figure 3.3. In order to develop a real and practical control system a PCB was designed, fitting the size and restrictions of the CubeSat, so a unique body could be possible. This PCB will be used in future satellites developed by GranSat for controlling the Momenta and the position if the satellite within a exact target. Therefore, the control has to be smooth enough and accurate.

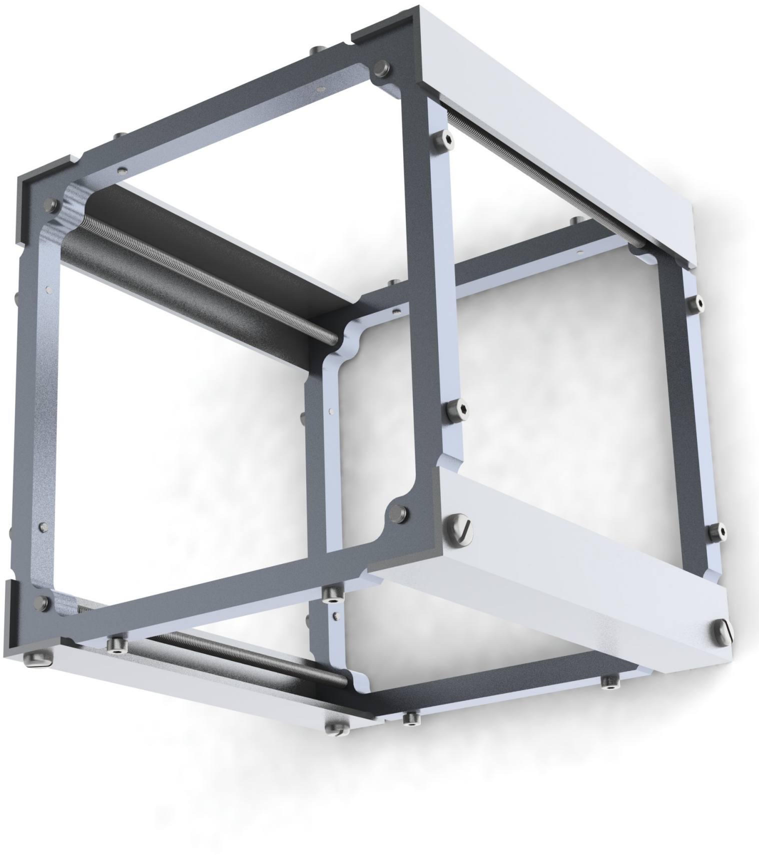


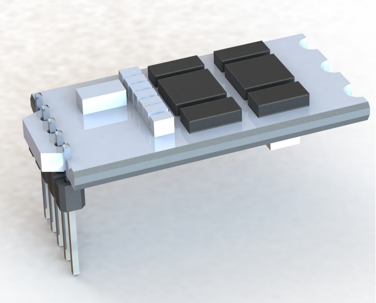
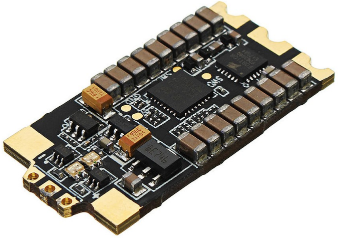
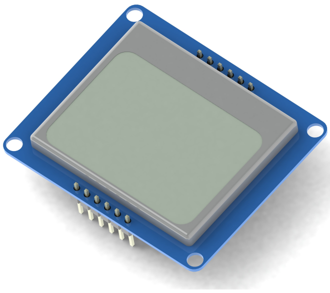
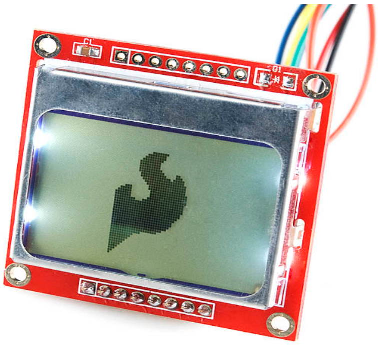
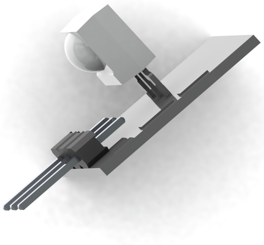
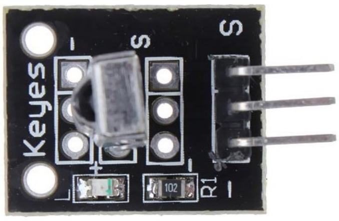
Figure 3.3 – *2D Aluminium CubeSat designed by GranaSat*

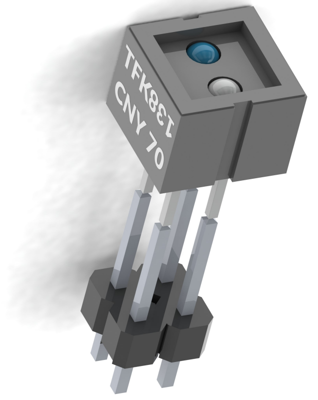
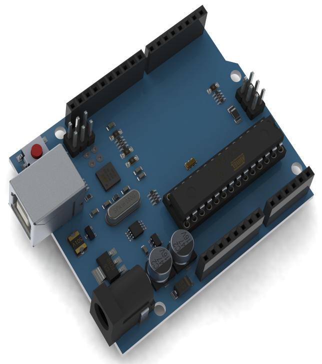
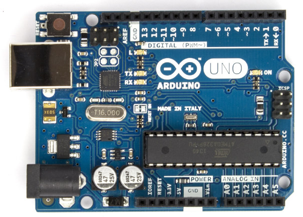
The components used for the PCB will be discussed in chapter **??**, but it basically cosists of a Arduino Uno microcontroller Shield, where two 10DOF Adafruit IMUs are used to get the body orientation; a PWM controlled ESC variator, used for powering a BLDC motor; a CNY70 phototransistor used as the encoder; and user interface components, such as an IR receiver and a Graphic LCD. The final assembly can be seen in figure 3.6.

For this prototype, in order to develop correctly the PCB and get the physical parameters that will be used for fitting the PCB in the CubeSat structure all the components had to be modeling through SolidWorks with the exact sizes and weights.

The electronics on it will be detailed on the next chapter, devoted to the Electronic Design, chapte 4. Also, all the schematics, detailing each of the components and its connections are shown in chapter 6.

Table 3.3 – *3D Components CubeSat Design*

Render 3D Model	Component	Description
		- BLDC 3-Phase Induction Motor. 3D Design made in Solidworks.
		- Wraight32 ESC variator. 3D Design made in Solidworks.
		- Nokia 5100 48×84 pixels matrix LCD. 3D Design downloaded from GrabCAD.
		- KY-022 IR Receiver. 3D Design made in Solidworks.

		• CNY70 Reflective Sensor. 3D Design downloaded from GrabCAD, but modified to fit our requirements.
		• Arduino UNO. 3D Design downloaded from GrabCAD.

The following aspects have to be discussed in the design of the PCB:

- The design of the PCB is made in two layers, bottom and top layer, so the components will be placed either on the top or the bottom of the board.
- On the top layer, the 3 phase motor, the ESC Airbot Wraigth variator 4.6 and the two IMUs 4.3 will be placed.
- The two IMUs are placed along the diagonal of the PCB, making possible the utilisation of the algorithm developed for the tilt angle estimation 5.1.
- The BLDC 3 phase motor 4.6 will be attached to the PCB through four M3 screws. The motor itself allows this implementation. In order not to cut the wires of the motor, holes are placed in the PCB, so the wires go through them before been soldered to the ESC variator.
- The Nokia 5100 LCD 4.7 LCD will be placed on the bottom layer, so whenever the cube is balancing, the printed information can be read.
- The Arduino UNO is placed on the bottom layer, the headers on it will be erased, so it can be soldered directly to the PCB through new headers. This way we prevent the overheating of the regulator in the Arduino.

- The IR Receiver is place so when the cube is balancing it is still accesible for the user, making easy to point at it with the IR Remote controller.
- A double implementation was designed for the printed PCB. When the PCB Gerber file is sent to be printed, a few copies are received, therefore, the idea is to use one of these copies for placing the electronics, and the other one for placing the battery.
- All the nets, as well as the buses are correctly commented on the top overlay layer, so whenever placing the components will be easy.
- All the nets were sized according to the results given by PCB Trace Width Conversion Calculator.

The final design can be seen in the following figures:

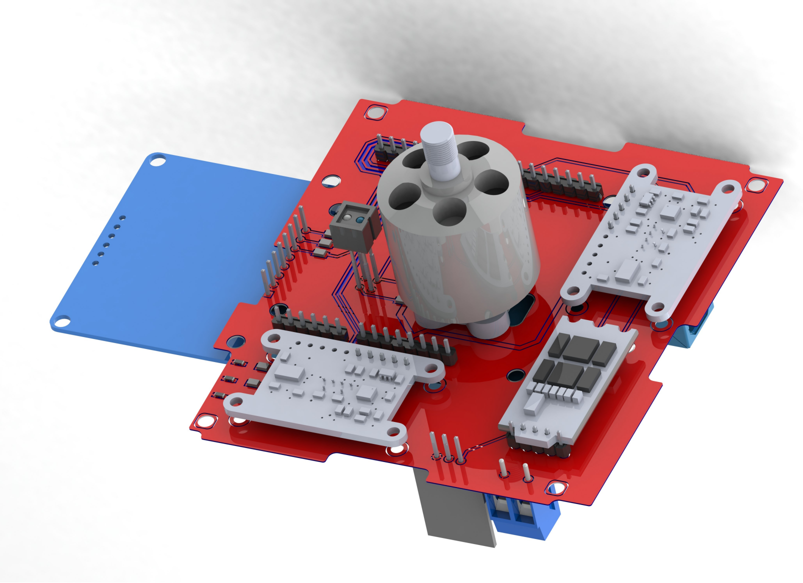


Figure 3.4 – *Top View of the PCB*

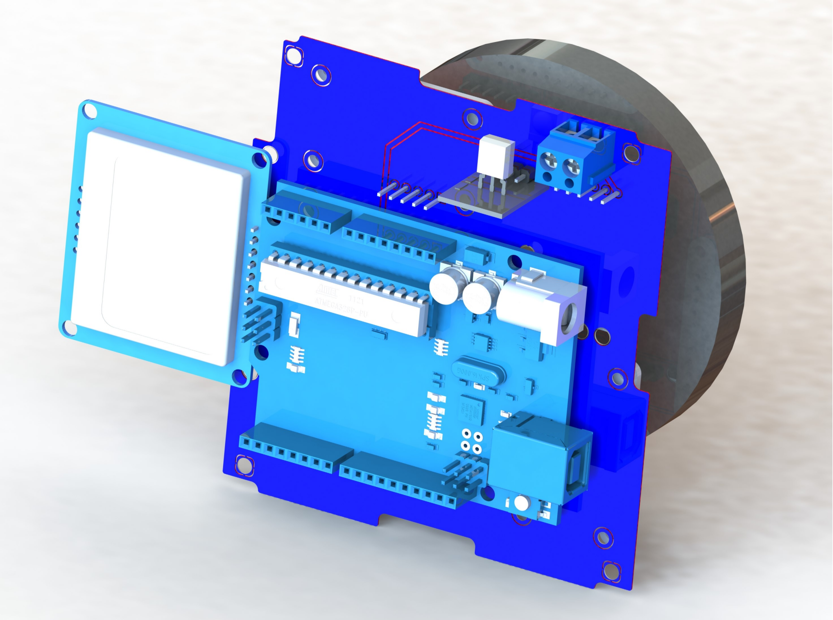


Figure 3.5 – *Bottom View of the PCB*

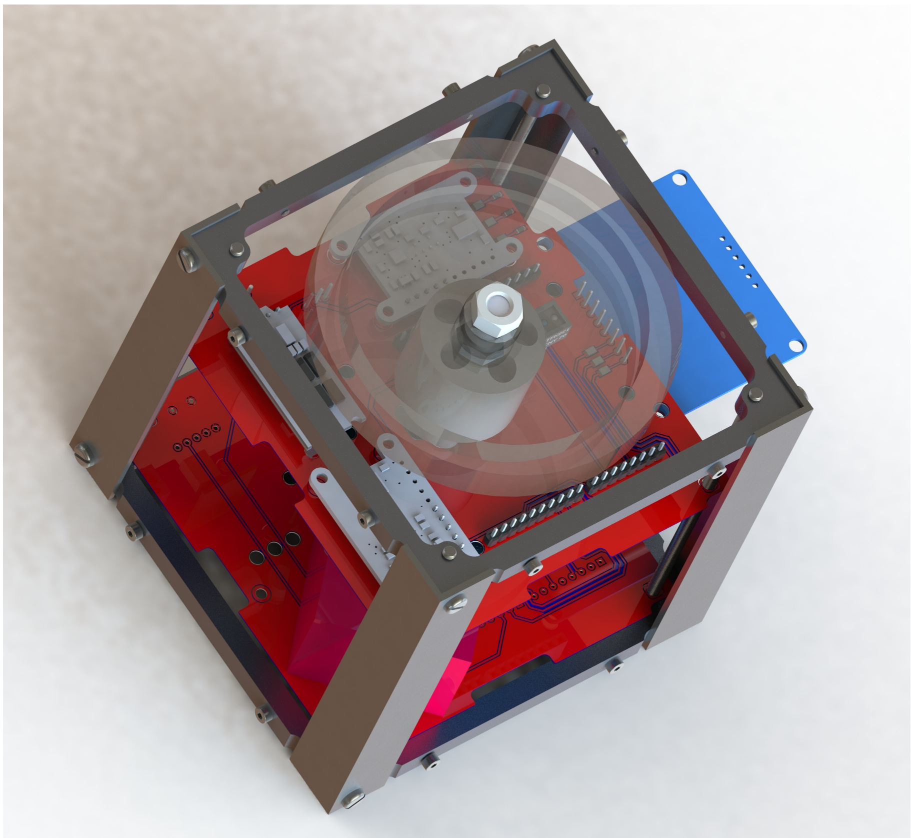


Figure 3.6 – *Top View of the Resulting System*

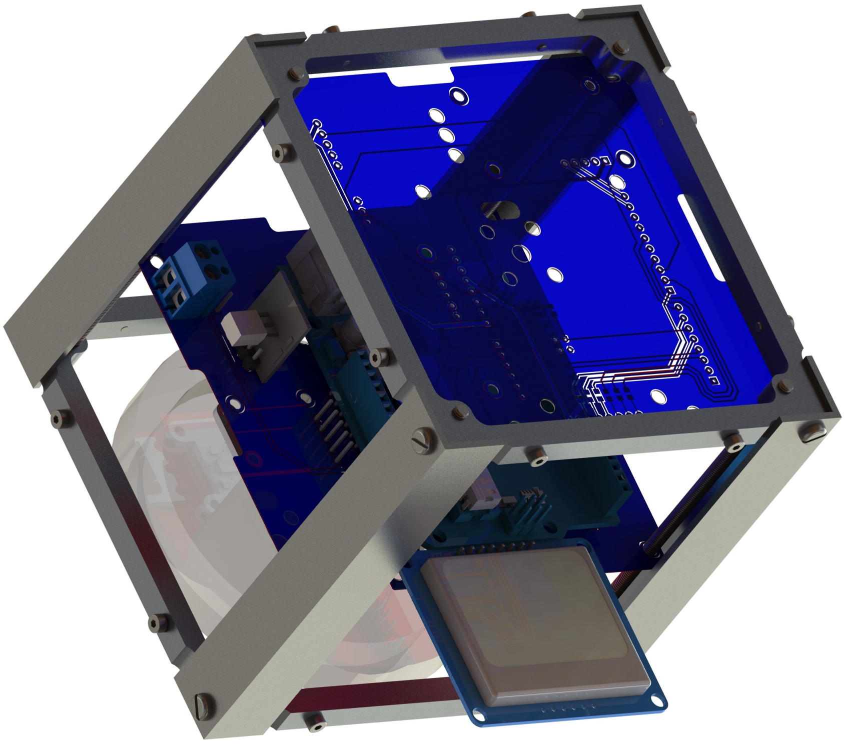
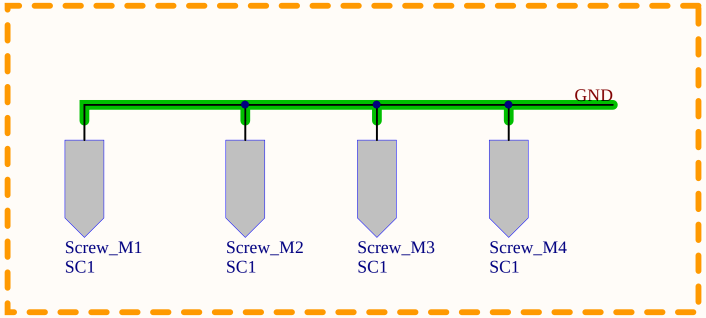
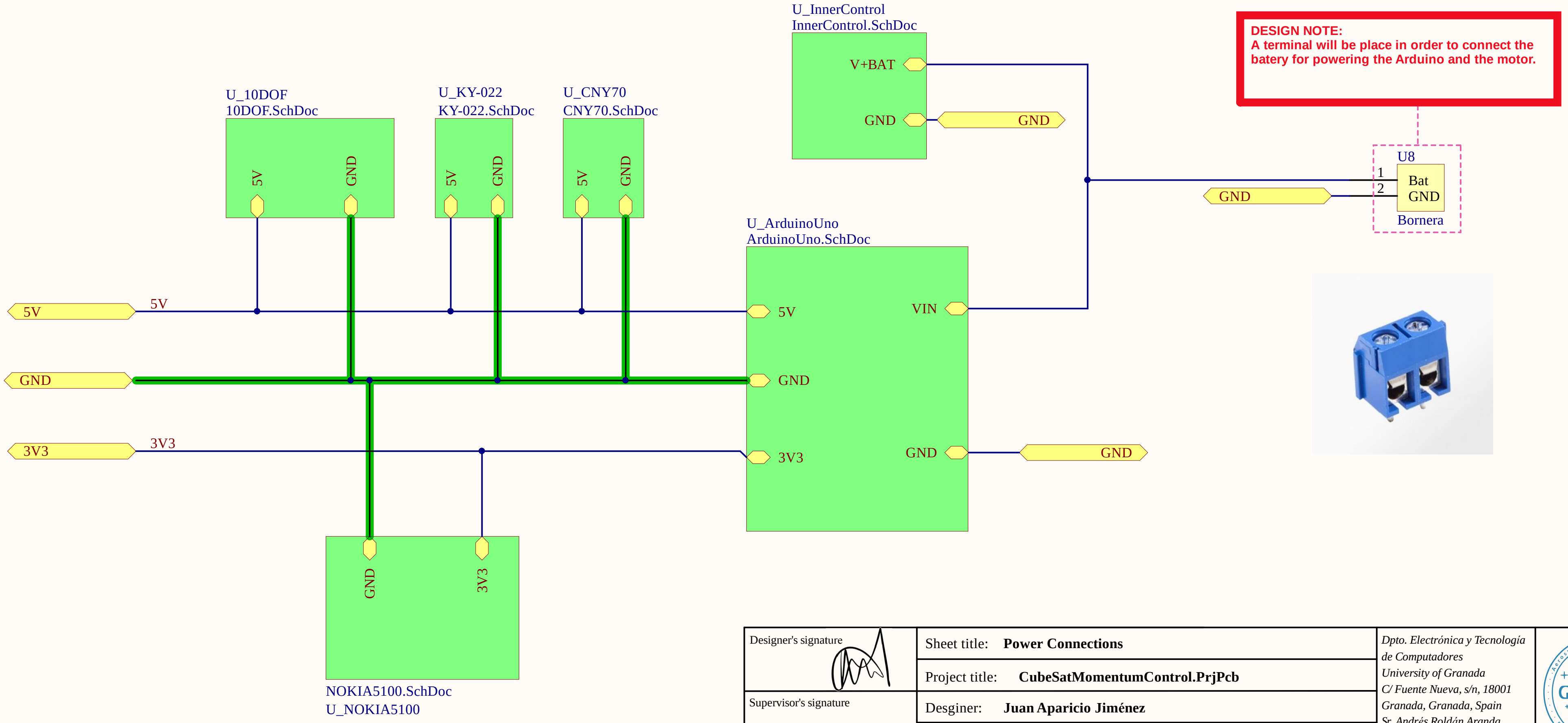


Figure 3.7 – *Bottom View of the Resulting System*

Power Connections



DESIGN NOTE:
For powering our PCB and the motor a 3S LIPO battery was chosen. It gives 11.1 volts and it has a relatively high power density, 1500mAh in 130g. Also, it squared shape allow us having the battey inside the steel CubeSat developed, making the system indepent from external power.



DESIGN NOTE:
A terminal will be place in order to connect the batery for powering the Arduino and the motor.

Designer's signature

Supervisor's signature

Sheet title: **Power Connections**

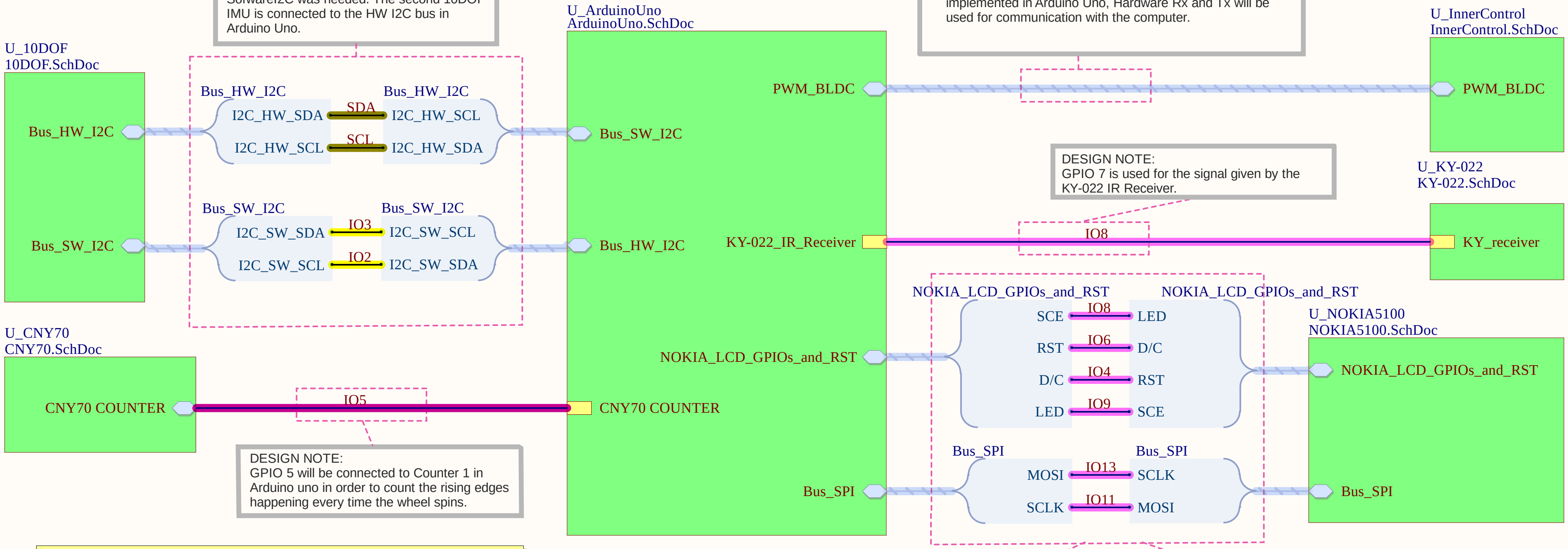
Project title: **CubeSatMomentumControl.PrjPcb**

Desginer: **Juan Aparicio Jiménez**

Date: **11/06/2019** Revision: ***** Sheet * of *

Dpto. Electrónica y Tecnología de Computadores
University of Granada
C/ Fuente Nueva, s/n, 18001
Granada, Granada, Spain
Sr. Andrés Roldán Aranda

Signal Connections



DESIGN NOTE:
GPIOs 2-3 are used as SDA-SCL in a SoftwareI2C protocol in order to communicate with one of the 10DOF IMUs. The accelerometers and gyroscopes had the same non-configurable I2C address so the SofwareI2C was needed. The second 10DOF IMU is connected to the HW I2C bus in Arduino Uno.

DESIGN NOTE:
GPIO 10 (Arduino Uno) will be used for controlling the input PWM signal, as well as for any change or updating on the Wraith 32 ESC firmware, through BLHELI 32. GPIO 12 will be used to communicate a software Rx implemented in Arduino Uno, Hardware Rx and Tx will be used for communication with the computer.

DESIGN NOTE:
GPIO 7 is used for the signal given by the KY-022 IR Receiver.

DESIGN NOTE:
GPIO 5 will be connected to Counter 1 in Arduino uno in order to count the rising edges happening every time the wheel spins.

Arduino Uno was chosen for the project trying to make the Inner Control Platform reusable and only with off-the-shelf components.

Also, it could be powered by a 11.1V LIPO battery, needed for the BLDC Motor and ESC. The internal regulator in Arduino Uno allowed us to connect 5V and 3.3V components to it.

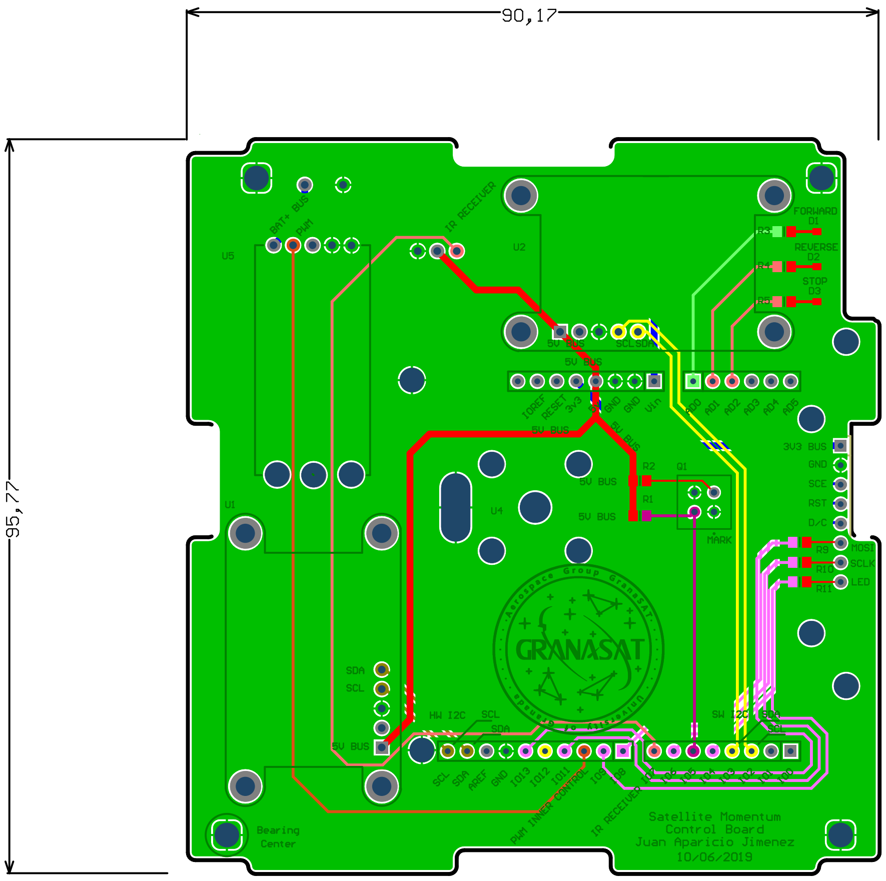
Furthermore, libraries were avaliable for some components, such as for the Nokia 5100 LCD and the KY-022 IR Receiver.

DESIGN NOTE:
GPIOs 8-6-4-9 are used for: chip select (SCE) , reset (RST), data/command (D/C). The LED pin should be connected to a PWM-capable Arduino pin, so we can dim the backlight as we please, therefore IO9 was chosen.

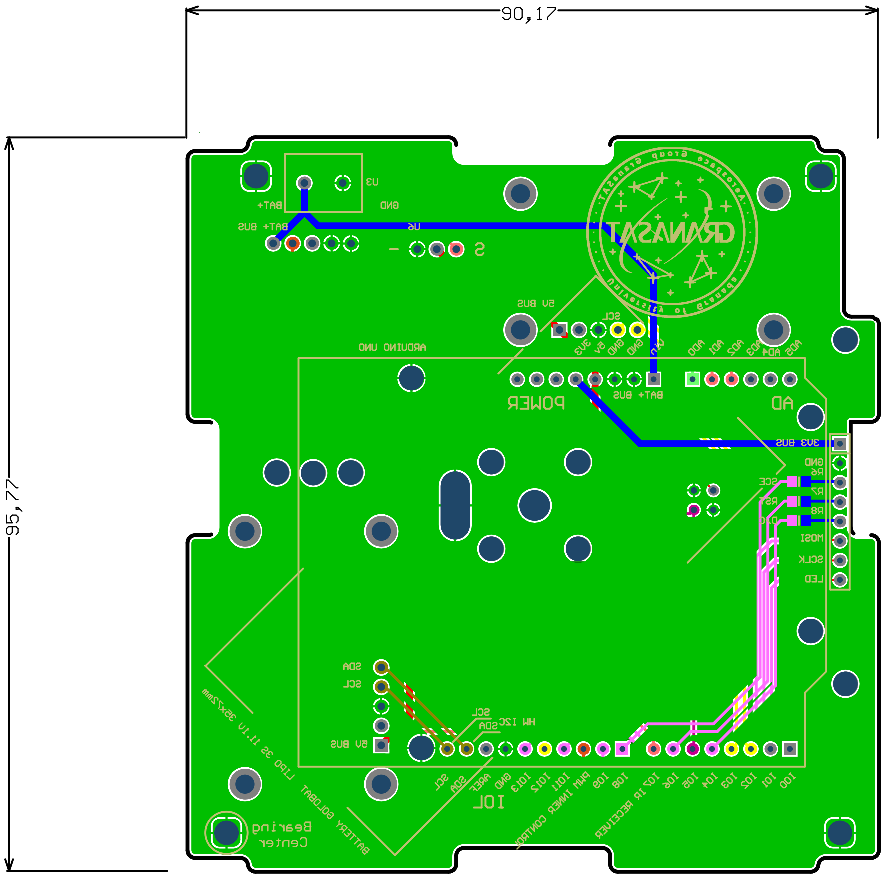
DESIGN NOTE:
For the data transmission with the Nokia 5100 LCD, pins -- SCLK and DN(MOSI) -- Arduino's hardware SPI pins will be used, which will help to achieve a faster data transfer.

DESIGN NOTE:
General Power Connections are detailed on PowerConnections.SchDoc

Designer's signature	Sheet title: **Signal Connections**		*Dpto. Electrónica y Tecnología de Computadores* *University of Granada* *C/ Fuente Nueva, s/n, 18001* *Granada, Granada, Spain* *Sr. Andrés Roldán Aranda*
	Project title: **CubeSatMomentumControl.PrjPcb**		
Supervisor's signature	Degsigner: **Juan Aparicio Jiménez**		
	Date: **11/06/2019** Revision: *****	Sheet * of *	



Designer's signature:	Sheet title: PCB	Dpto. Electronica y Tecnologia de Computadores University of Granada C/ Fuente Nueva, s/n, 18001 Granada, Granada, Spain Sr. Andres Roldan Aranda
	Project title: GranaSAT	
Supervisor's signature:	Designer: Juan Aparicio Jimenez	
	Supervisor: Andres Roldan Aranda	**Date: 21/05/2019 Revision: 2** **Sheet X**



Designer's signature:

Supervisor's signature:

Sheet title: PCB

Project title: GranaSAT

Designer: Juan Aparicio Jimenez

Supervisor: Andres Roldan Aranda

Dpto. Electronica y Tecnologia
de Computadores
University of Granada
C/ Fuente Nueva, s/n, 18001
Granada, Granada, Spain
Sr. Andres Roldan Aranda

Date: 21/05/2019 Revision: 2
Sheet X

Chapter 4

Electronic Design

The electronic design will determine whether the project is succesfull or a failure. Therefore, for the realisation of this project different sensors and motors where considered. Not only the speed and accuracy of sensors was important, but also how fast the controller could read the values on them and using the control algorithm give an output. The final choice will be explained for each case.

4.1 Arduino UNO

Arduino UNO seemed like a simple and affordable option for this project. The idea was to make a Arduino UNO shield which could be used for controlling the momentum of a CubeSat. It allowed us to easily communicate with the computer through the Serial Port, really useful when developing the code and the calibration and filtering for the IMUs, futher explained 5.1. Also, many libraries for some of the components were avaliable online, in chapter 5, devoted to the Software design, the utilisation of this libraries will be detailed.

Moreover, it could be powered with a 11.1V Battery, needed for powering the 3 phase motor. The recommended input voltage goes from 7V to 12V, according to its datasheet [1], and the power consumption is really low. Some of our components and sensors work with a 5V supply, so the internal regulator in the Arduino UNO seemed really useful. The maximum power supply through this 5V regulator is stimated in 1W, when a external supply is used. The DC current for 3.3V Pin is maximum 50 mA but, for our purpose, where only the Nokia5100 4.7 works at 3.3V, the power will be more than enough.

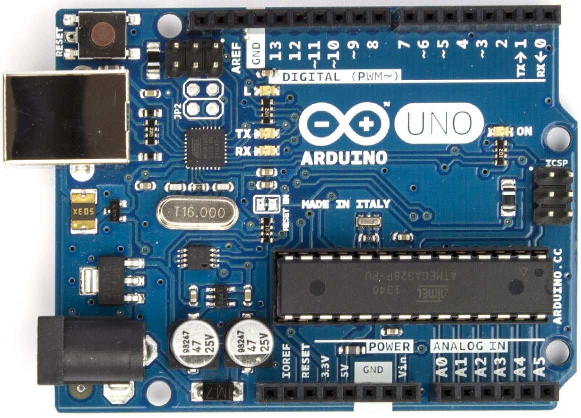


Figure 4.1 – *Arduino UNO. Source [1].*

4.2 First Prototype

This basic first approach to the project was developped using exclusively off the shield components that were found in the lab. As said previously in section 3.1 the aim of this first prototype developed and 3D printed was to understand the phisics and developing the basic algorithms and IMU based estimation of the position. The components that were used only in this prototype, and later discared are shown in the following sections 4.2.1 and 4.2.2.

4.2.1 DC Mini-Motors M30N

According to the research on internet, probably used in a old scanner. Being a DC motor, CCW/ACW movement was possible, regarding the polarity.

The voltage range goes from 28 34 V, therefore, it wont be really useful when not connected to a power supply. Also, the size and weight of this motor won't make it useful for our project in terms of the smooth control needed. As seen in section 2.3 when the weigth of our system is too high, the torque needed and the consumption of energy will be higher. Even though it wasn't the best options in terms of the control, in order to understand better the meaning of the torque shown in 2.1.5 and how to get the constant for a given input, this motor was usefull.

The dimmensions of this motor are shown in figure 4.2 and a SolidWorks model was developed as well to size the 3D printed face and the holes on it.

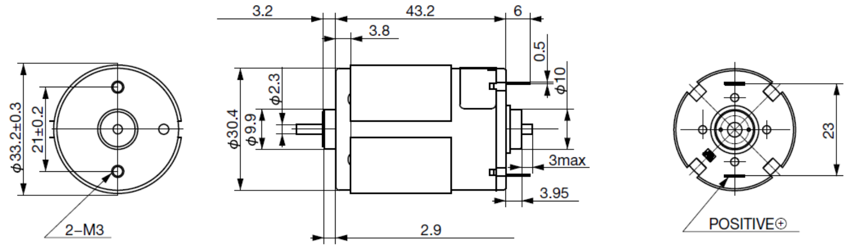


Figure 4.2 – *Measures DC Motor. Source [2].*

DC motors draw high current, the copper loss is high and they are not smooth enough for our purpose. Therefore, for the following prototype, the DC motor won't be used because it decreased power and increased vibration compared to a 3-phase motor, which runs smoother. Also, when using a 3 phase motor, we can get maximum torque at the beginning considering the high efficiency.

4.2.2 L298N H-Bridge

The L298N bridge was chosen for this prototype, considering that the operating supply voltage was up to to 46V an the high current admitted. It integrates two power output stages, for our purpose only one will be used, but thinking about the future and the aim of this project, set the base for a further attitude control of a satellite, it seemed interesting to control a dual H-Bridge.

"Each bridge is driven by means of four gates the input of which are In1 ; In2 ; EnA and In3 ; In4 ; EnB. The In inputs set the bridge state when The En input is high ; a low state of the En input inhibits the bridge." [24].

The regulator only works with voltages up to 12V in Vin, therefore, for our purpose the jumper between Vin and the regulator had to be removed and the logical part of the module had to be feed from another 5V source.

As supposed before using, once it was understood how to use an H-Bridge through PWM pins in Arduino as an output of the control algorithm implemented, PID as seen in section 2.4. The DC motor was changed for a 3 phase motor and therefore, the H-Bridge wasn't used for the following prototype.

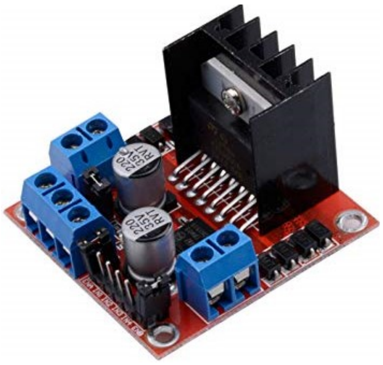


Figure 4.3 – *L298N H-Bridge. Source [3].*

4.3 Adafruit 10DOF IMU

A Adafruit breakboard was used in order to get the position of the face fast enough and with high accuracy. The basic components within the 10DOF breakboard, as seen is its datasheet [4] are:

- L3GD20H 3-axis gyroscope: ±250, ±500, or ±2000 degree-per-second scale.
- LSM303 3-axis compass: ±1.3 to ±8.1 gauss magnetic field scale.
- LSM303 3-axis accelerometer: ±2g/±4g/±8g/±16g selectable scale.
- BMP180 barometric pressure/temperature: -40 to 85 °C, 300 - 1100hPa range, 0.17m resolution.

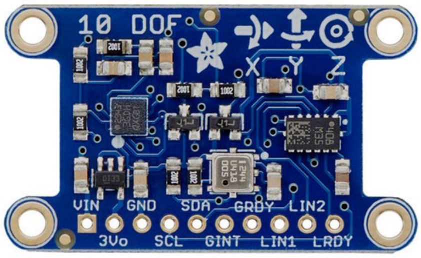


Figure 4.4 – *Adafruit 10DOF IMU. Source [4].*

The actual IMU breakboard we had was a Chineese clone, hence, some of the components weren't the components they were suppossed to be, specially the gyroscope was an old version. That caused delays in terms of developing the code, many of the 10DOF libraries couldn't be used and had to be modified to make it work.

For our project, two IMUs were avaliable in the lab, and they were used in order to make the interference immunity higher. Consequently, we had to connect two IMUs to the I2C bus in Arduino Uno. The I2C address of each component on this breakboard wasn't programable, so both IMUs couldn't be plugged into the same I2C Bus.

The solution chosen was to develop a SoftwareI2C Bus on other pair of pins in the Arduino Uno, acting as sda/scl. Hence, depending on which I2C bus we read, either the HW or SW, we will get the data from the components on one of the breakboards or the other. For this purpose, the library SoftwareI2C for arduino was used. [https://github.com/felias-fogg/SoftI2CMaster——SoftwareI2C Master Library].

The libraries given by Adafruit for the 10DOF breakboard were adapted to the new SoftwareI2C protocol so the read was done properly. When using SoftwareI2C, the commands are not the same as when using HW I2C, thus, a library for SoftwareI2C had to be created.

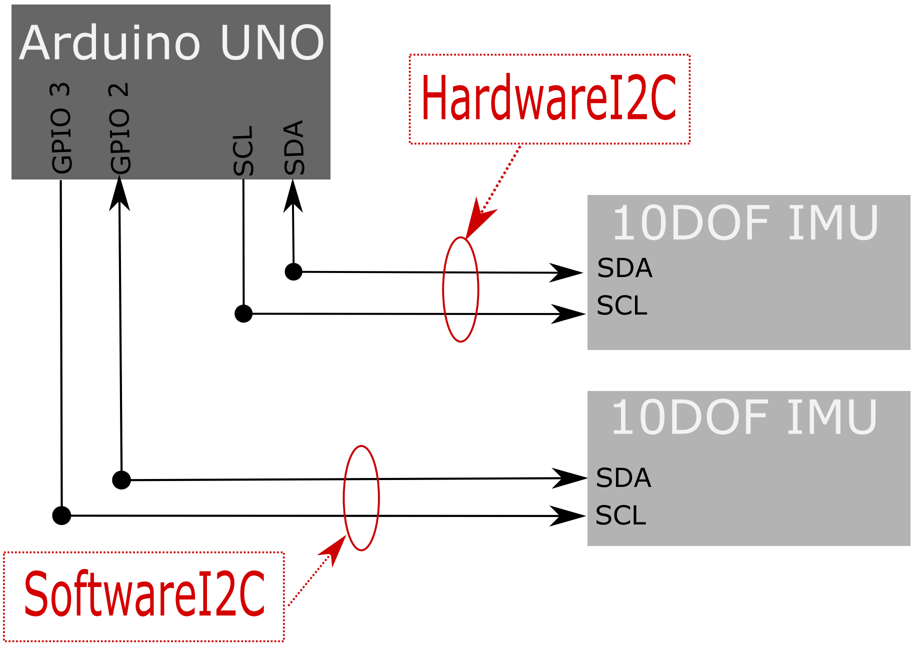


Figure 4.5 – *HW and SW i2c Bus for comunnication with two 10DOF IMUs.*

For a proper position and tilt angle estimation, some filtering and calibration is needed to compensate the error caused by the inertial sensors. After considering a few options, such as AHRS filter, Mahony filter, Kalman filter, the decision made was to use geometry to stimate the tilt angle of the body and a complementary filter, which combines accelerometer and gyroscope signals for proper estimation. Both, calibration of the components and the tilt estimation algorithm are explained in subsection 4.3.1 and 4.3.2.

4.3.1 IMU Calibration Process

Inertial sensors may be subject to different types of measurement errors, not only due to the limitations on the precision with which those measurements are made, but also because of a possible lack (or excess) of sensitivity of the sensor and steady-state related errors among others.

Those two can be modelled as follows:

$$X_{meas} = \Delta_{sens}\left(X_{real} + \Delta_{steady}\right) \tag{4.3.1}$$

that is, Δ_{sens} represents an error that scales proportional to the measurement magnitude, and Δ_{steady} is the bias that comes out when the real measured magnitude is zero.

Depending on the sensor, there is a variety of plausible methods for obtaining both errors.

4.3.1.1 Gyroscope

A gyroscope measures the angular velocity vector of the system. In particular, in the process of obtaining the orientation of a given body, the gyroscope signal is integrated over time. This is done by adding up the product of the angular velocity times the sampling period for every iteration:

$$\theta(k) = \theta(k-1) + \omega_{\theta}(k-1) \cdot \Delta t \tag{4.3.2}$$

Is easy to see how each error affects the measurement of the attitude.

First, the sensitivity error results in an augmented or diminished value of the angle respect to the initial value. Secondly, the steady-state error sums up in every iteration and increases over time.

The last is well seen after computing the angle value for a fixed system state:

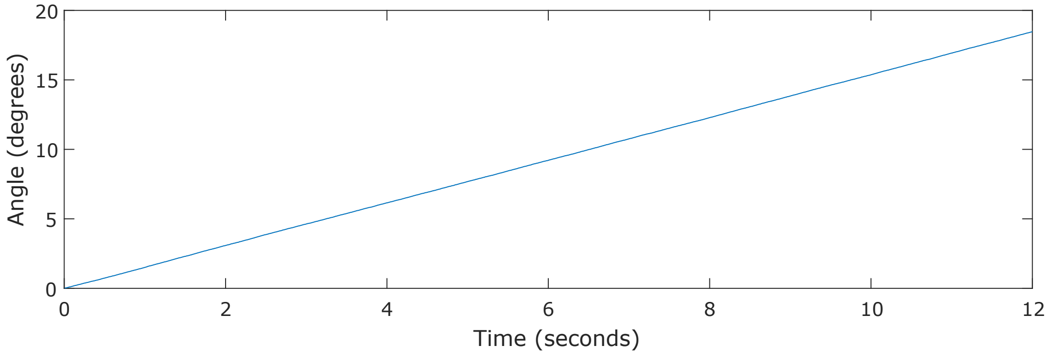


Figure 4.6 – *Angle bias for fixed state*

To obtain the steady-state error, the sensor signal is measured over a period of 120 seconds (20.000 iterations), and a mean of $-1.867°$ is obtained.

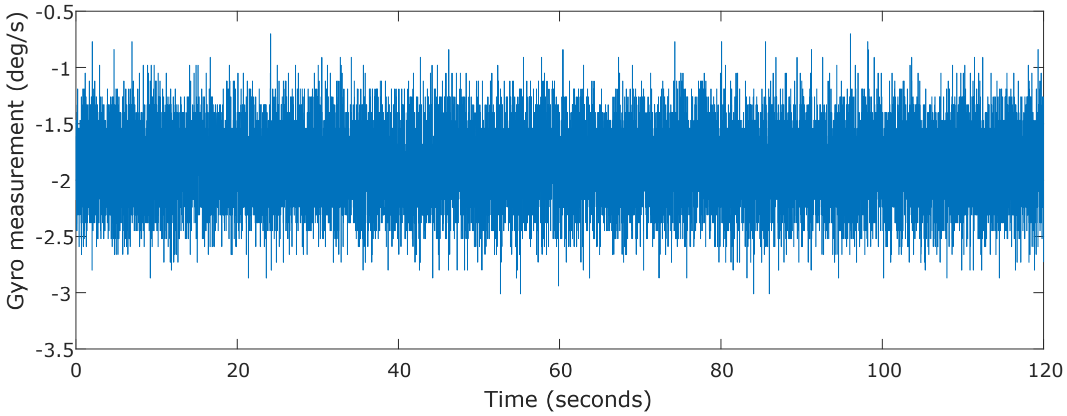


Figure 4.7 – *Gyroscope measurement for fixed state*

This value has to be substracted to each measurement of the angular velocity. To obtain the sensitivity error, a rotation of 90° is performed on the system and then the angle is measured.

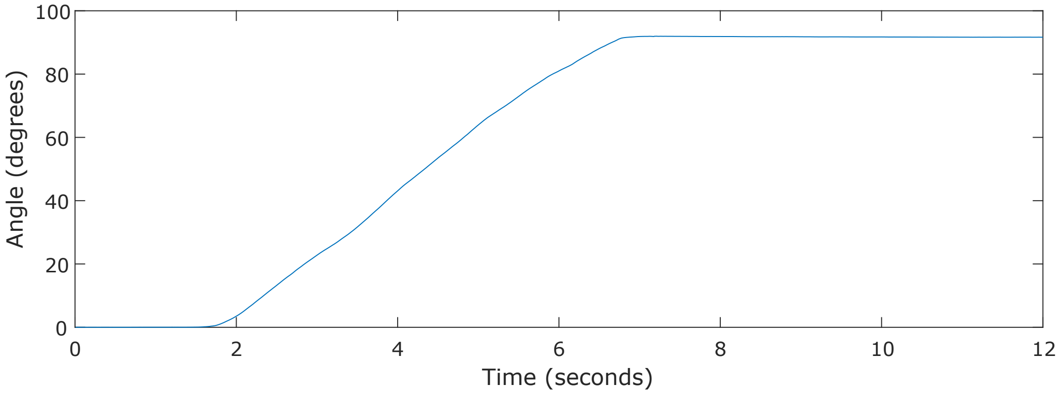


Figure 4.8 – *Gyroscope measured rotation*

The value stabilises at 91.5° and the sensitivity error is given by $\frac{91.5}{90} = 1.0167°$.

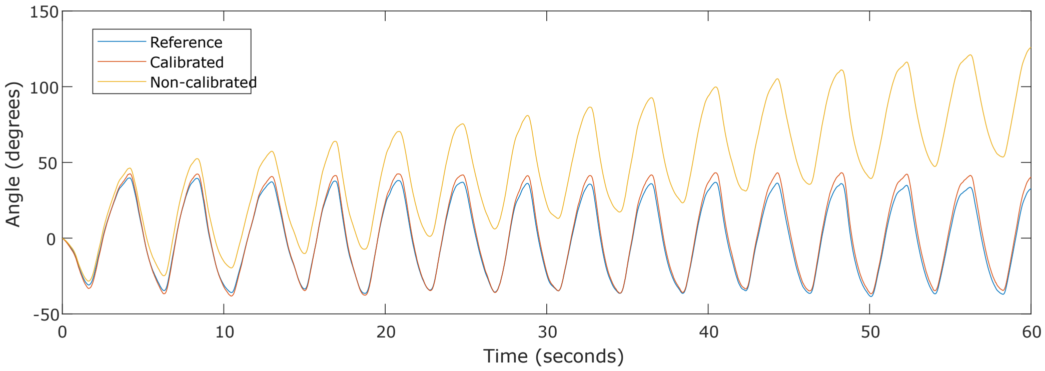


Figure 4.9 – *Orientation measurement before and after calibration compared to reference*

Although the calibration results in a considerable improvement of the body orientation measurement, the effect of the intrinsic noise 4.7 results in a random walker-type error that increases with time. This error and the one produced by the discrete integration generates a bias that can't be reduced by any calibration, and needs from further methods like sensor fusion for improvement. This methods, particularly the complementary filter will be explain on section 4.3.2.

4.3.1.2 Accelerometer

An accelerometer measures the acceleration of the body along it's three axes. The accelerometer measurements are subjected to a large amount of noise compared to the gyroscope, and it varies depending on the static or dynamic state of the body, getting larger as the change rate of the acceleration vector grows.

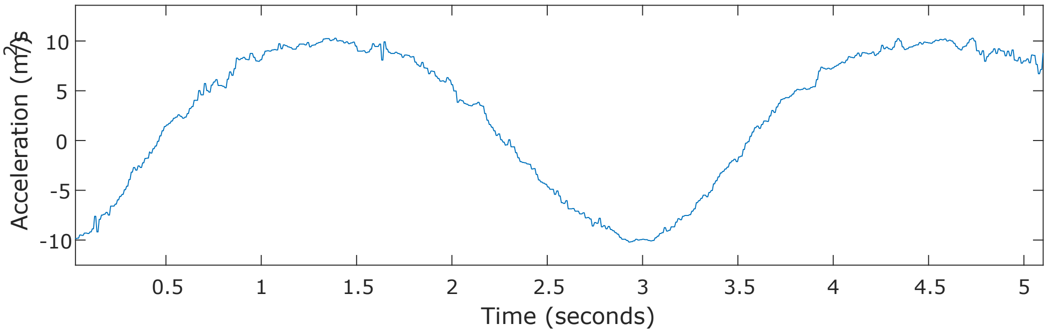


Figure 4.10 – *Accelerometer noise in dynamic state*

As in the previous case, the accelerometer presents sensitivity scaling and static offset errors.

A quick method to calibrate the accelerometer is to use the orthogonality property of the axes to determine the maximum gravity output value for each individual axis. This is done by slowly rotating the sensor in a way that the measured acceleration in two of the axes becomes zero at some point, in which the third axis displays the absolute value of the gravitational acceleration.

When the positive and negative value of acceleration is computed for a given axis, both the scaling and steady-state error can be obtained.

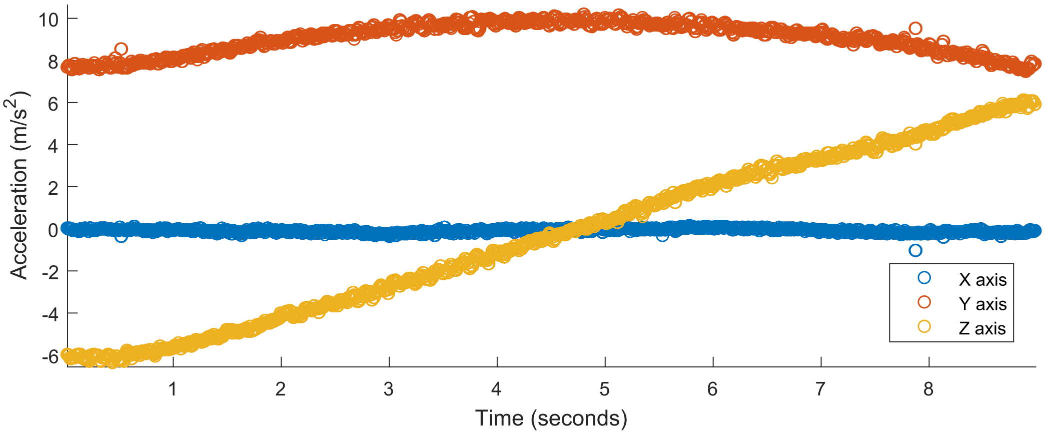


Figure 4.11 – *Acceleration measurement during rotation*

As seen above, when two of the accelerations cross at 0, the third reaches it's maximum value. This is done for both positive and negative part of each axis.

Accelerometer 1	Positive (m/s^2)	Negative (m/s^2)	Steady state bias (m/s^2)	Scaling
X axis	9.60	-10.31	-0.35	1.02
Y axis	10.06	-10.15	-0.05	1.03
Z axis	10.00	-10.37	-0.19	1.04

Accelerometer 2	Positive (m/s^2)	Negative (m/s^2)	Steady state bias (m/s^2)	Scaling
X axis	9.96	-10.25	-0.16	1.04
Y axis	9.80	-10.06	-0.13	1.01
Z axis	9.65	-9.92	-0.14	9.99

Table 4.1 – *Maximum values of acceleration, static bias and scaling error*

4.3.2 Complementary Filter

When measuring a system's magnitude, each sensor gives some information with different accuracy. For example, if two measurements are avaliable for the same magnitude: x_1, x_2, each one with the error $\triangle x_1, \triangle x_2$. Moreover, we can assume that the probability distribution associated with the measures is given by a gaussian function, with a standard deviation $\sigma_i = \Delta x_i$.

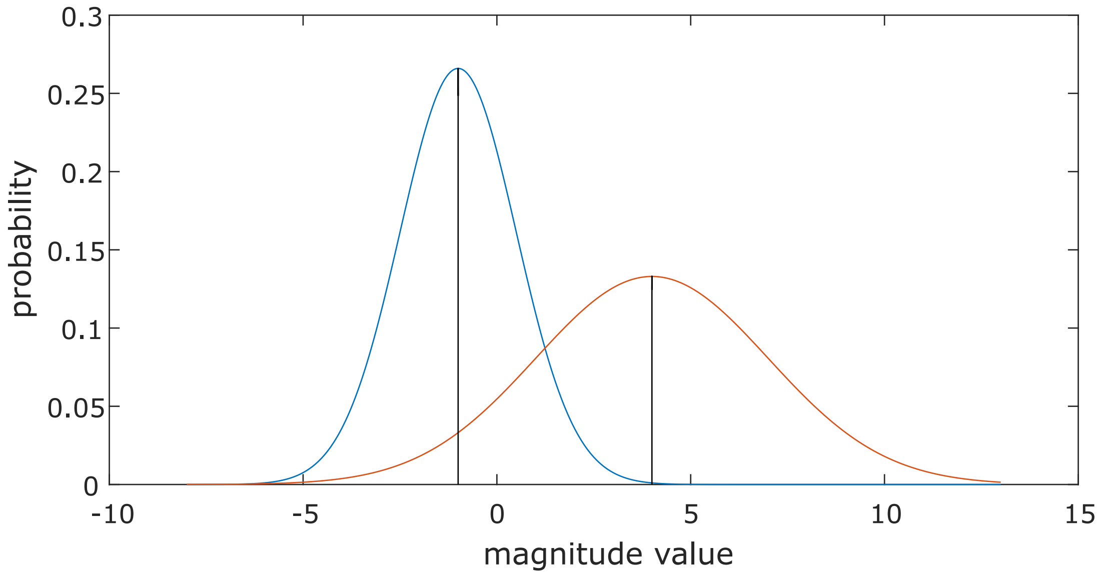


Figure 4.12 – *Two meassures with its associate Error*

Then, according to Bayes' theorem [25], the distribution of the optimal estimatimation of the magnitude is given by the product of the two probability functions (normalized):

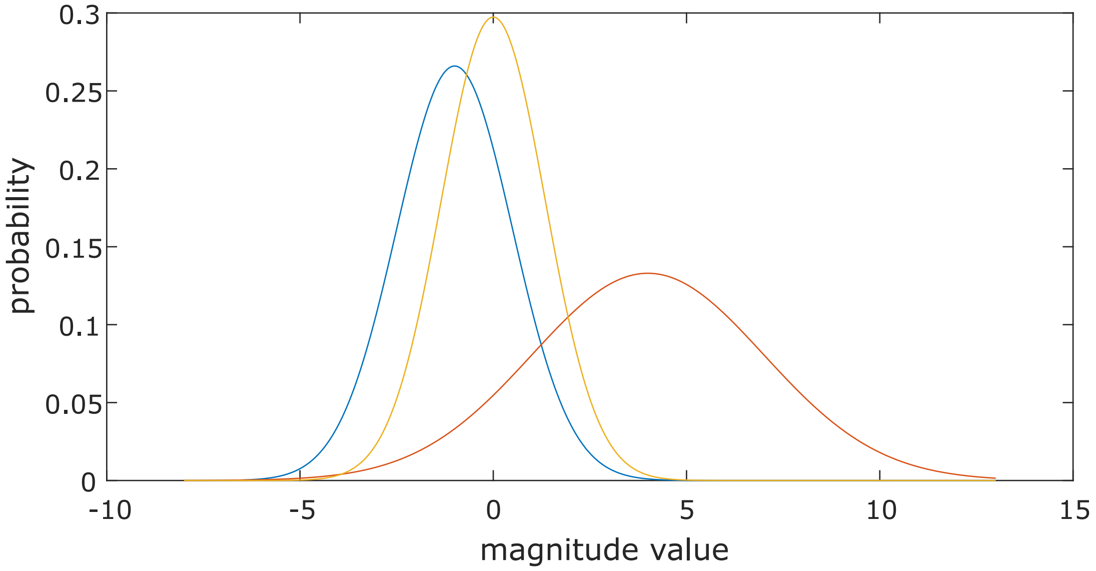


Figure 4.13 – *Optimal estimation merging two measures.*

Where the mean and variance of the new distribution is given by [25]:

$$\mu = \mu_1 + \frac{\sigma_0^2(\mu_1 - \mu_{0)}}{\sigma_0^2 + \sigma_1^2} \tag{4.3.3}$$

$$\sigma^2 = \sigma_0^2 - \frac{\sigma_0^4}{\sigma_0^2 + \sigma_1^2} \tag{4.3.4}$$

Therefore, for every pais of measures, the variance associated to the optimal measure is always less than the variance of the measures before merging.

$$\sigma \leqslant \sigma_0,\ \sigma_1 \tag{4.3.5}$$

This implies that no matter how bad a measure is, the information that it contains will always be useful when estimating the real value of the magnitude.

In this statistical principle, optimal filters are based, such as Kalman.

However, these filters have a high computational cost due to the large number of calculations involved, and for certain applications you can use other simpler sensor fusion methods but also valid and that require a shorter execution time. A simple implementation filter is the so-called complementary filter.

The complementary filter is based on calculating the value of a quantity making a weighted average of all the different measures carried out:

$$x = \sum_n C_n x_n \qquad \sum_n C_n = 1 \tag{4.3.6}$$

The weighting of said measures is done based on the estimate of the error of each of them; If a measurement has a lot of error, it will associate a constant of small proportionality and the other way around.

For our purpose, two sensors are avaliable in the IMU, that as seen before in sectio 4.3.1:

- Gyroscope: has a very precise measurement and with little rotation noise of the system, but is subject to a bias that increases with time.
- Accelerometer: through the measurement of the gravitational vector returns a constant update of the orientation of the system without presenting bias, but the measurement is subject to a large amount of noise.

Making a weighted sum of the two measures, we managed to obtain a value with a very low noise similar to that the one given by the gyroscope but with a temporary correction of the bias thanks to the measure given of the accelerometer.

The values of the proportionality constants were obtained in a empirical way:

$$\theta = 0.96\theta_{gyro} + 0.04\theta_{acc} \tag{4.3.7}$$

In the algorithm, the change in the angular position of the body in a iteration is given by the angular velocity θ^{gyro} by the sampling time, $\omega\Delta t$:

$$\triangle\theta^{gyro} = \omega\Delta t \tag{4.3.8}$$

The change in angular position measured by the accelerometer is the previous measure minus the current one:

$$\triangle\theta^{acc} = \theta_k - \theta_{k-1} \tag{4.3.9}$$

The update in the angle calculated by complementary filter is then:

$$\triangle\theta = 0.96\triangle\theta_{gyro} + 0.04\triangle\theta_{acc} \tag{4.3.10}$$

4.4 CNY70

Meassuring the speed of the wheel was a challenge in terms of accuracy and cost. A tachometer was needed for characterizing the motor and it's torque at the beginning, and later on, when a LQR control algorithm was considered, the spin velocity of the inner wheel was crucial. In order to make it useful for future projects, no matter the kind of motor used (BLDC or brushed motor) the choice made was to make the tachometer by using a CNY70.

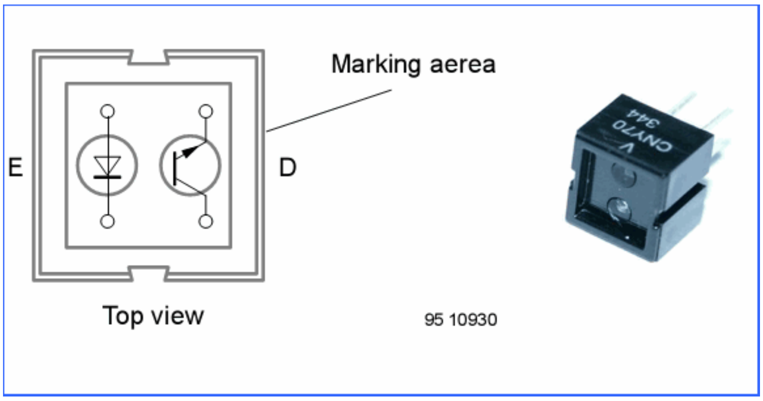


Figure 4.14 – *CNY70. Source: [5]*

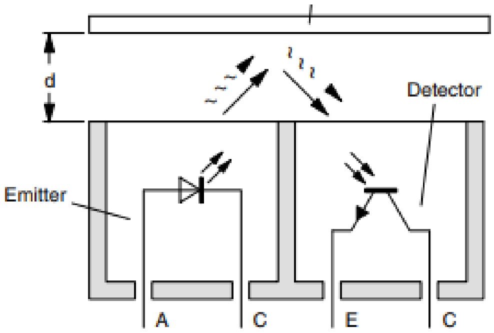


Figure 4.15 – *CNY70 phototransistor working mode. Source: [6]*

Summing up the working principle shown in Vishay Semiconductors, the manufacturer, Application of Optical Reflex Sensors [26]: The CNY70 contains a IR-emitting diode as transmitter and a phototransistors as receiver. The transmitter emit radiation of a wavelength of 950 nm. Depending on the reflection surface the output will represent a different logic value. If there's reflection of the infrared then the gate of the transistor will allow the flow of electrons, and thank to the pull down resistor, a 5V signal (1 in logic level) will appear on the output. If on the contrary the surface absorbs the light, then the logic level on the output will be 0. The Cut-off frecuency is 40KHz, according to the manufacturer, so it will be fast enough for our purpose.

We place four black stickers along the Inner Wheel, absorbing the infrared, so on every spin there will be four changes in the electrical signal. IO5 is connected to the phototransistor's collector. Counter 2 on the Arduino Uno will then count every rising edge on IO5, and therefore the number of spins within a meassured time can be known. The internal structure of the Counter 2 in the ATmega328 can be seen in the block diagram shown in figure 4.16.

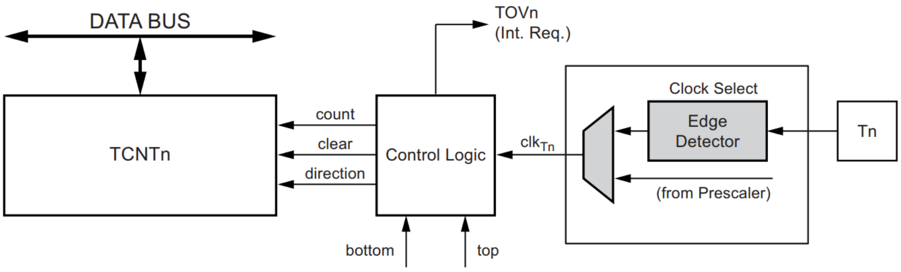


Figure 4.16 – *Counter2 ATmega328 block diagram. Source: [7]*

Some registers have to be programmed in order to use the collector in the CNY70 as the input in Counter 2 of ATMega 328 (Arduino Uno HW). During the setup of the Arduino program, this code has to be added, in order to count the rising edges on the counter 2, which will be connected to the collector of the CNY70 phototransistor, this is further explained in section 5.4, devoted to the RPMs measurement code for measuring through the Arduino UNO.

The measurement setup is shown in 4.17 for meassuring the torque and later on the CNY70 will be placed on the PCB close to the inner wheel so it can meassure the same way the inner wheel spinning velocity in real timing.

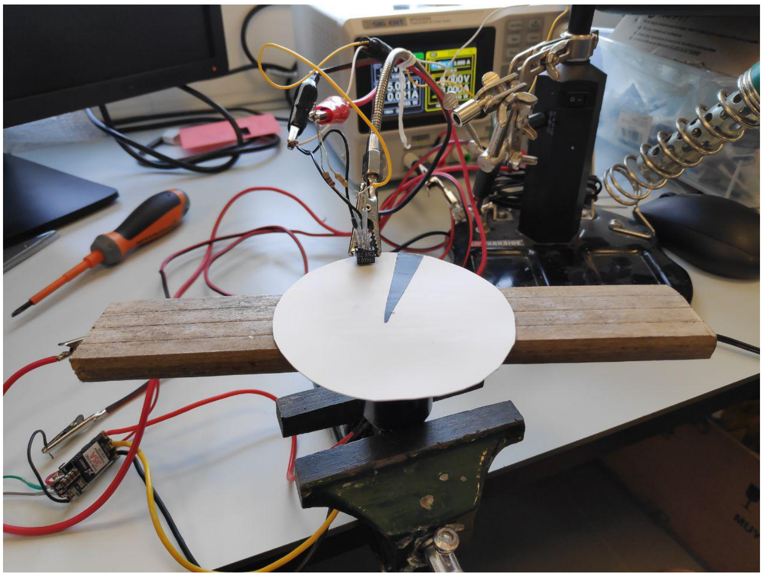
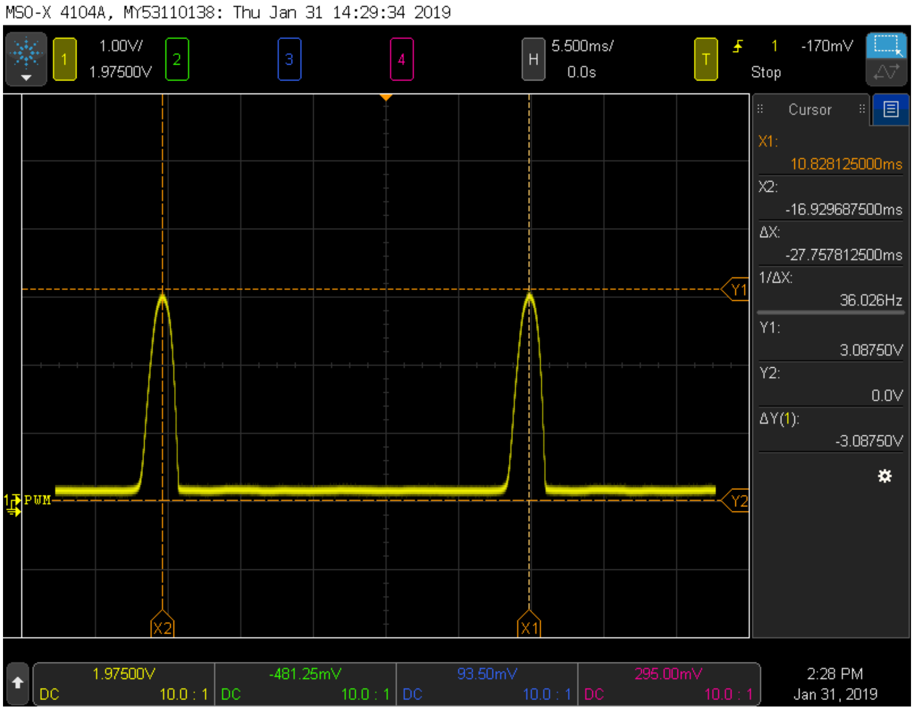


Figure 4.17 – *CNY70 implementation for reading RPMs.*

To probe the correct operation of the CNY70 in rpms meassurement, the output was plugged into the oscilloscope. The spinning frecuency meassured was then compared with the frecuency given by a laser tachometer found in the lab.



MSO-X 4104A, MY53110138: Thu Jan 31 14:29:34 2019

1.00V/ 1.97500V

5.500ms/ 0.0s

1 -170mV Stop

Cursor

X1: 10.828125000ms

X2: -16.929687500ms

ΔX: -27.757812500ms

1/ΔX: 36.026Hz

Y1: 3.08750V

Y2: 0.0V

ΔY(1): -3.08750V

1.97500V DC 10.0 : 1

-481.25mV DC 10.0 : 1

93.50mV DC 10.0 : 1

295.00mV DC 10.0 : 1

2:28 PM Jan 31, 2019

Figure 4.18 – *Osciloscope scope CNY70 output.*

For setting up the measurement of spinning frequency with the CNY70, the oscilloscope was used as a first approach. A python script shown in Appen C. was implemented for giving the frequency of rising edges on the output of the CNY70, connected to the osciloscope, while the voltage input was changed automatically. As it can be seen in the code, a volt sweep was made, and the measured frequency, the current drawn and the voltage applied for each case where written in a .csv file. The script was run 6 times in order to detect any disturbances. During the voltage sweep, using a manual tachometer the rpms were measured as well. The results are shown bellow in table 4.4.

The data obtained in the experiment is shown in the following figures.

Table 4.2 – *Rpms measured by the CNY70 while a voltage sweep was made to the DC motor.*

CNY70 Measure											
Sample	Vsupply	Current	Power	Error	Std deviation	Freq.	Error	Std deviation	RPM	Error	Std deviation
0	0.5	0.012	0.006	0.002181	0.005	7.765	0.115	0.564	465.888	2.462	33.864
1	1	0.017	0.017	0.000000	0.000	17.439	0.007	0.032	1046.328	0.004	1.923
2	1.5	0.020	0.030	0.000008	0.000	26.930	0.008	0.037	1615.800	0.003	2.213
3	2	0.025	0.050	0.000006	0.000	36.096	0.012	0.060	2165.760	0.006	3.592
4	2.5	0.032	0.081	0.000005	0.000	44.696	0.018	0.090	2681.760	0.011	5.401
5	3	0.034	0.103	0.000005	0.000	54.388	0.030	0.149	3263.280	0.025	8.943
6	3.5	0.041	0.144	0.000004	0.000	62.724	0.019	0.093	3763.440	0.008	5.598
7	4	0.048	0.193	0.000003	0.000	70.880	0.020	0.098	4252.800	0.008	5.879
8	4.5	0.058	0.260	0.000010	0.001	78.500	0.032	0.155	4710.000	0.018	9.295
9	5	0.071	0.357	0.000003	0.000	84.800	0.018	0.089	5088.000	0.006	5.367
10	5.5	0.093	0.512	0.000000	0.000	88.900	0.018	0.089	5334.000	0.005	5.367
11	6	0.077	0.463	0.000002	0.000	103.340	0.040	0.196	6200.400	0.022	11.758
12	6.5	0.084	0.549	0.000003	0.000	111.160	0.048	0.233	6669.600	0.029	13.994
13	7	0.093	0.651	0.000004	0.001	118.680	0.042	0.204	7120.800	0.021	12.238
14	7.5	0.102	0.768	0.000006	0.001	125.860	0.040	0.196	7551.600	0.018	11.758

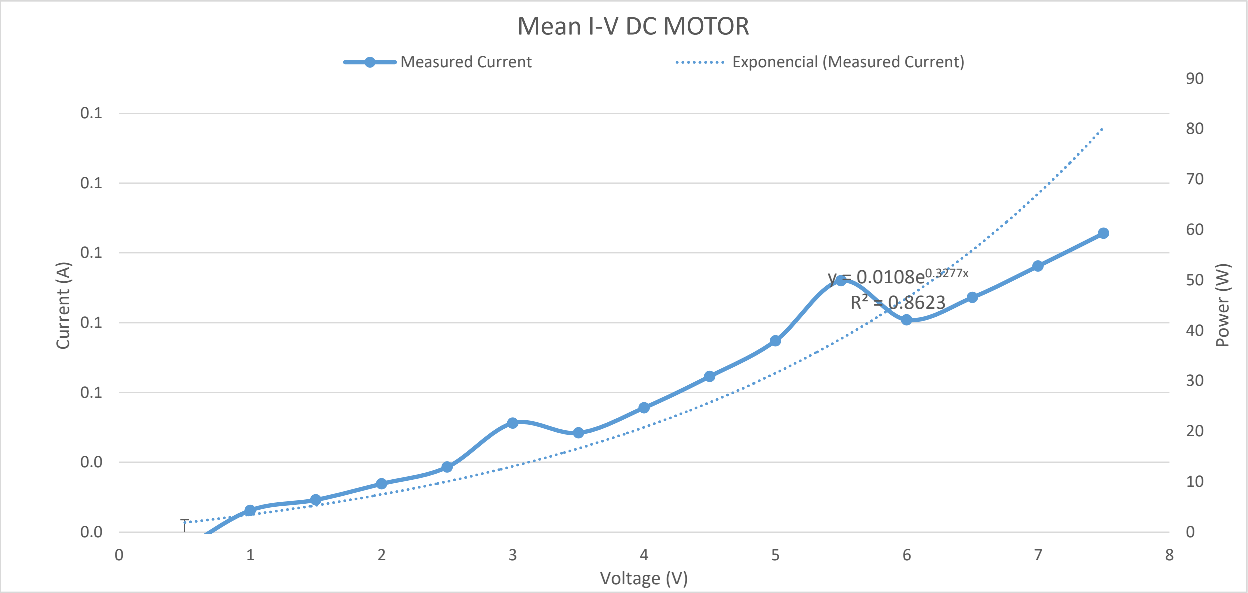


Figure 4.19 – *Mean I-V DC MOTOR. It shows how the higher the voltage is, the more current the DC motor drawns.*

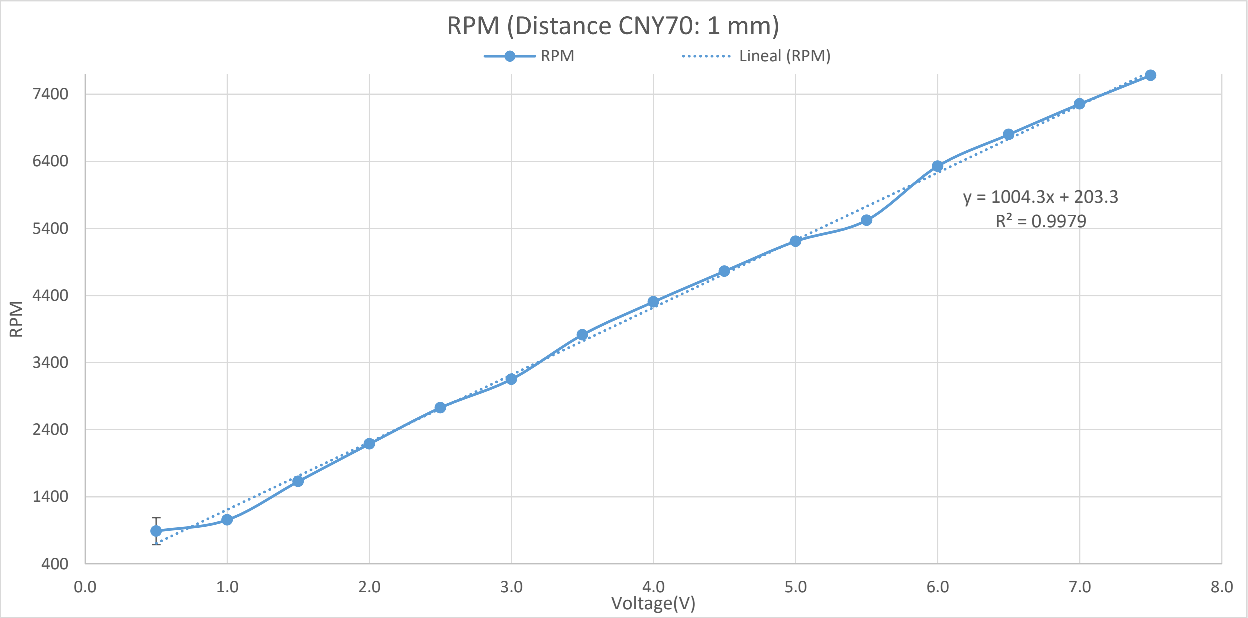


Figure 4.20 – *Measured RPMs during voltage sweep. It shows how both measures, the one taken by the CNY70 and read by the oscilloscope, and the one gotten from the hand tachometer are really similar, the error is almost 0, probing the correct operation of the CNY70.*

4.5 KY-022 IR Receiver

KY-022 IR Receiver is used for IR communication with a remote controller. Each key on the remote controller has an unique coding. In order to make the system start/stop and move the target point when balancing the IR controller seemed like an easy and practical way on control.

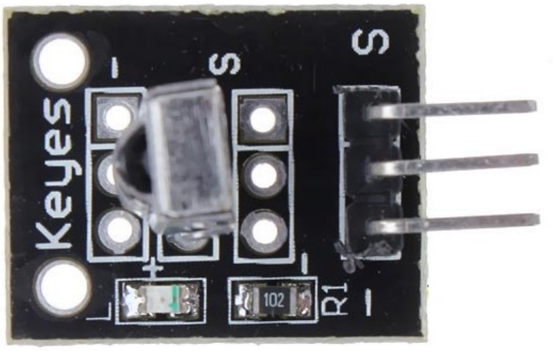


Figure 4.21 – *KY-022 IR Receiver. Source [8]*

.

The signal is trasmitted to the Arduino Uno through this device. The modulated IR signal is a series of IR light pulses switched on and off at a high frequency known as the carrier frequency. The pattern in which the modulated IR signal is converted to binary is defined by a transmission protocol. The NEC protocol used in our project considers the following: *Logical '1' starts with a 562.5 µs long HIGH pulse of 38 kHz IR followed by a 1,687.5 µs long LOW pulse. Logical '0' is transmitted with a 562.5 µs long HIGH pulse followed by a 562.5 µs long LOW pulse.* [8]

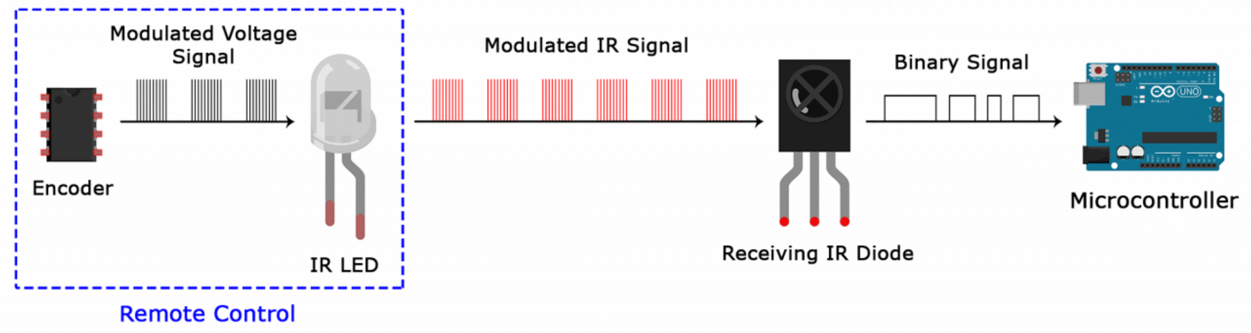


Figure 4.22 – *KY-022 IR Receiver working method. Source: [8]*

4.6 Airbot Wraith32 ST

For the final prototype a 3-phases BLDC motor was chosen due to the smoother control, low weight and high torque in low rpms. A cheap and light motor was chosen for this purpose, trying to reduce the weight of the system and the budget. In order to move the 3 phase induction motor, a variator driver was needed.

The aim of the algorithm is to control the torque given by the motor to the system in order to stabilize the tilt angle. PMW control was therefore used as the output variable of the algorithm and the input of the variator. The Pulse Width Modulation is then the responsible of controlling the system, when using a ESC variator, the PWM range goes from a pulse of 1ms which would be consider the lower, to 2ms, in a 50Hz wave. The PWM range can be seen in figure 4.23.

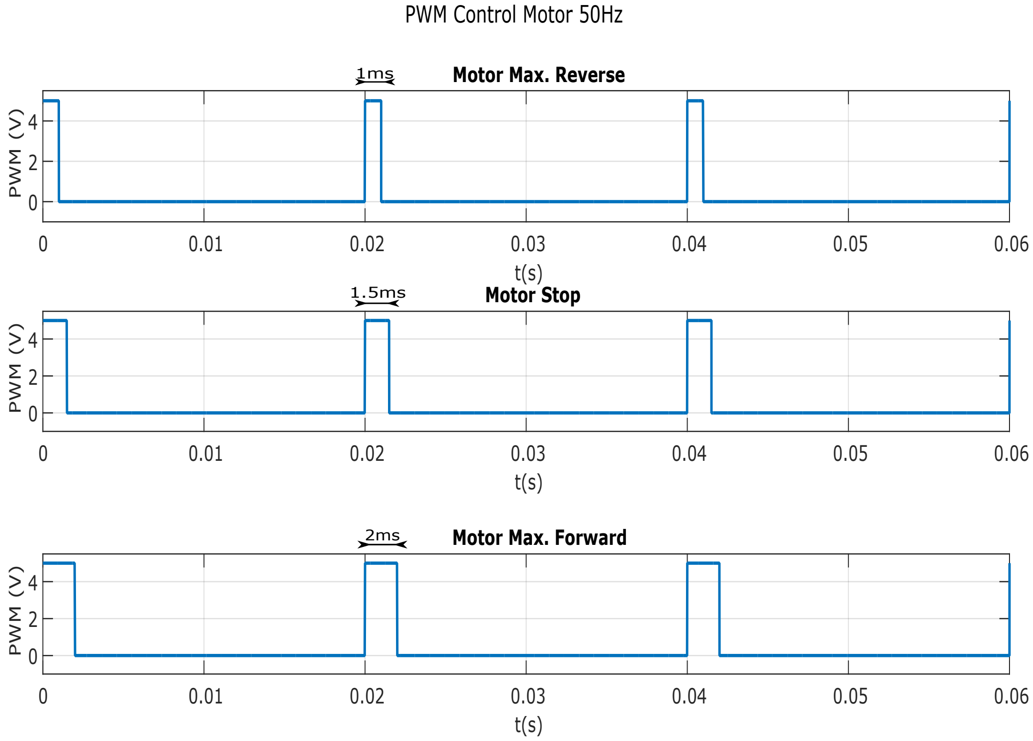


Figure 4.23 – *50Hz PWM control wave*

The variator chosen was the Airbot Wraith32 ST because it allowed up to 6S LIPO batteries, useful for the experiment, and a current limiter was also avaliable. The frequency is PWM programmabled and up to 48kHz, which makes the motor control smooth.

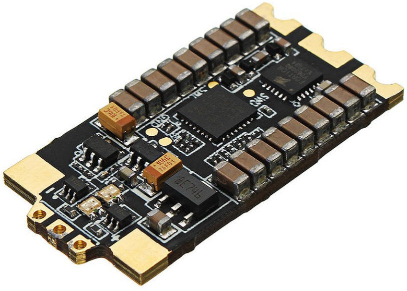
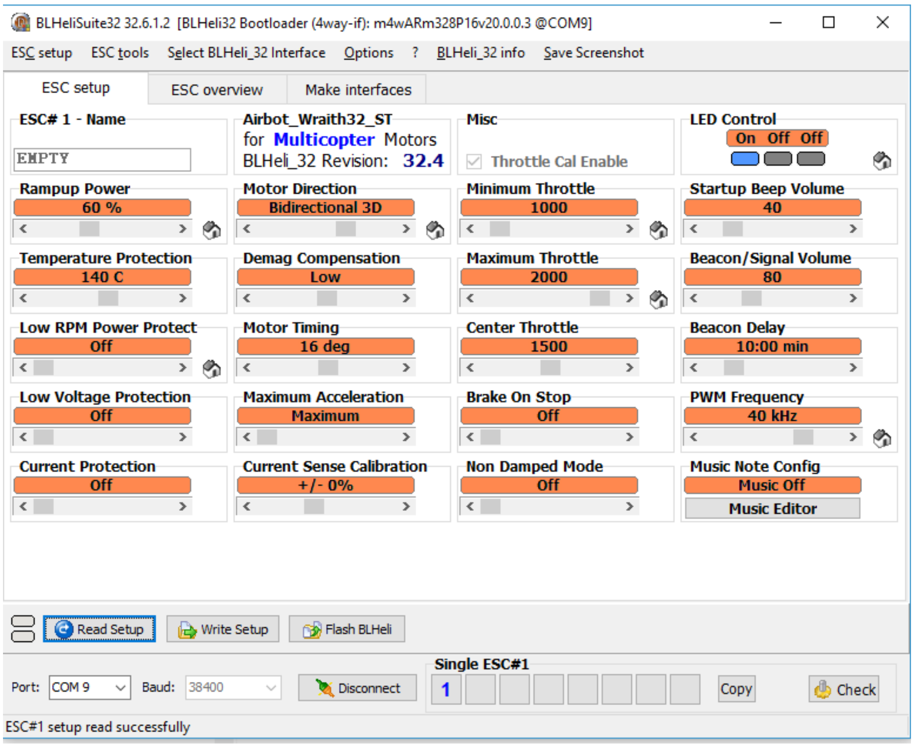


Figure 4.24 – *Airbot Wraith32 ST ESC variator*

A firmware update for the Wraight32 controller was first needed in order to make the BLDC motor work properly for our purpouse. Once the Wraight32 is connected to the Arduino an update can be done through the Software BLHELI32, avaliable in GitHub BLHeli32. The parameters shown in figure 4.25 were the parameters chosen for a bidirectional working mode, smooth stop and high torque. Some other parameters where chosen according to the working mode needed and the programming parameters shown in GitHub.

- Bidirectional 3D mode, so CW and CCW rotation will be avaliable. The minimum Throttle, meaning full power on reverse, will be set by a 1000ns width pulse in the 50Hz PWM wave; stop 1500ns width pulse; and full fordward power by a 2000ns width pulse.
- Low RPM Power Protect was disabled so full power was avaliable even if the supply voltage is low. Same for Low Voltage Protection.
- Brake On Stop was disabled, so a brake force won't be applied, an the stop will be smooth, so no negative torque is created (needed for our purpose).
- Rampup Power was set up to 60%, so the power consumption wouldn't be to high.



BLHeliSuite32 32.6.1.2 [BLHeli32 Bootloader (4way-if): m4wARm328P16v20.0.0.3 @COM9]

ESC setup   ESC tools   Select BLHeli_32 Interface   Options   ?   BLHeli_32 info   Save Screenshot

ESC setup   ESC overview   Make interfaces

ESC# 1 - Name: EMPTY

Airbot_Wraith32_ST
for Multicopter Motors
BLHeli_32 Revision: 32.4

Misc: ☑ Throttle Cal Enable

LED Control: On Off Off

Rampup Power: 60 %

Motor Direction: Bidirectional 3D

Minimum Throttle: 1000

Startup Beep Volume: 40

Temperature Protection: 140 C

Demag Compensation: Low

Maximum Throttle: 2000

Beacon/Signal Volume: 80

Low RPM Power Protect: Off

Motor Timing: 16 deg

Center Throttle: 1500

Beacon Delay: 10:00 min

Low Voltage Protection: Off

Maximum Acceleration: Maximum

Brake On Stop: Off

PWM Frequency: 40 kHz

Current Protection: Off

Current Sense Calibration: +/- 0%

Non Damped Mode: Off

Music Note Config: Music Off

Music Editor

Read Setup   Write Setup   Flash BLHeli

Port: COM 9   Baud: 38400   Disconnect

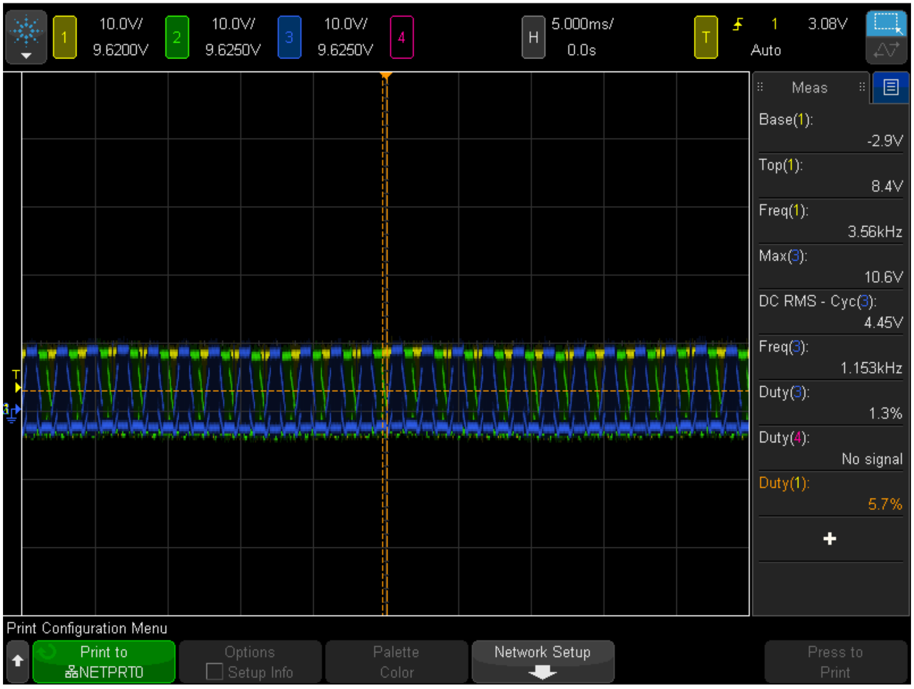
Single ESC#1: 1   Copy   Check

ESC#1 setup read successfully

Figure 4.25 – *BLHeli 32 Firmware for the ESC.*

Immitating a 3S LIPO battery, which would feed the variator with 11.1V, the power source was used so the correct operation could be checked, and the BLDC 3 Phase motor could be modelled, as seen in 2.1.5. The three-phase electric power on the output of the variator for a given PWM control signal is shown in figure 4.26.

MSO-X 4104A, MY53110138: Fri May 10 18:24:33 2019



Meas

Base(1): -2.9V

Top(1): 8.4V

Freq(1): 3.56kHz

Max(3): 10.6V

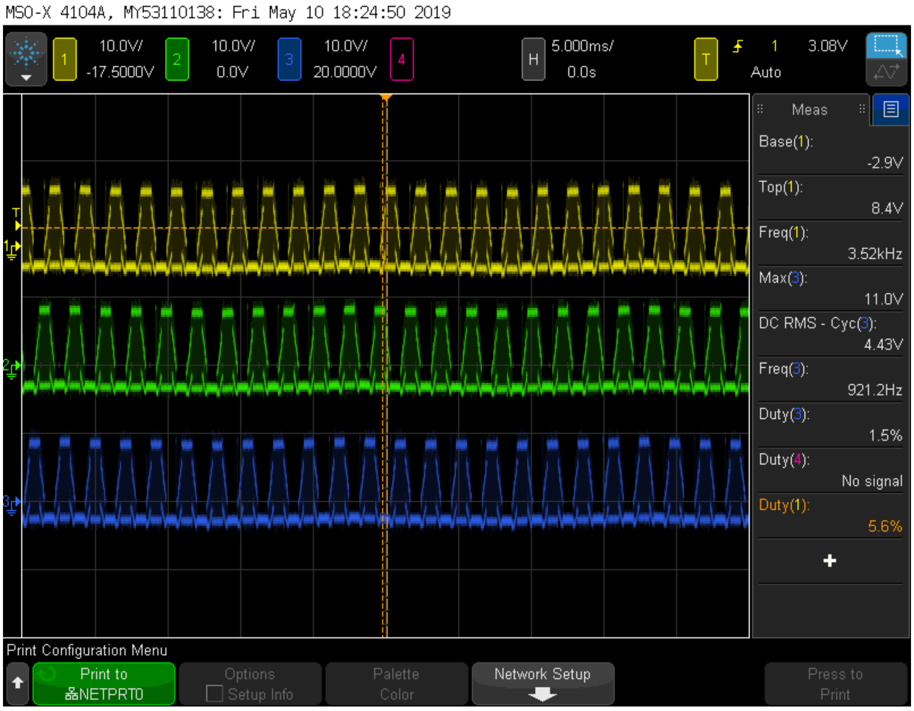
DC RMS - Cyc(3): 4.45V

Freq(3): 1.153kHz

Duty(3): 1.3%

Duty(4): No signal

Duty(1): 5.7%



MSO-X 4104A, MY53110138: Fri May 10 18:24:50 2019

1 10.0V/ -17.5000V
2 10.0V/ 0.0V
3 10.0V/ 20.0000V
4
H 5.000ms/ 0.0s
T 1 3.08V Auto

Base(1): -2.9V
Top(1): 8.4V
Freq(1): 3.52kHz
Max(3): 11.0V
DC RMS - Cyc(3): 4.43V
Freq(3): 921.2Hz
Duty(3): 1.5%
Duty(4): No signal
Duty(1): 5.6%

Figure 4.26 – *Three-phase electric power on the output of the ESC.*

For the interacting between the Arduino Uno and the ESC, a PWM pin will be used. A 50Hz PWM wave is needed as seen in figure 4.23, for that purpose, a 50Hz wave will be created from the 490Hz standar signal in Arduino Uno. Also, on every start of the system, the maximum and minimum throttle has to be set. Therefore, in the setup of our code, a 1ms width pulse; and a 2ms width pulse are sent to the ESC variator. Later on, the control will determine the pulse width to be sent.

4.7 Nokia 5100 LCD

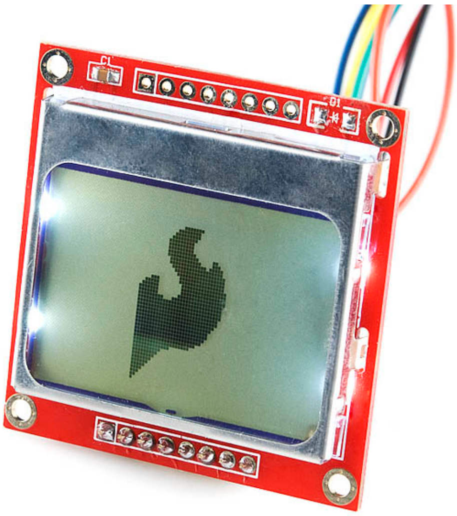


Figure 4.27 – *Nokia 5100 48 × 84 pixels matrix LCD*

This Nokia 5100 LCD was designed as a cellphone screen. It comes with the PCD8544 controller driving a 48x84 LCD matrix, which allows Serial Peripheral Interface, instead of a huge parallel interface. Therefore, the pinout characteristic are [27]:

- Vcc, which range goes from 2.7V to 3.3V, so we will feed the screen through the Arduino UNO 3V3 supply.
- SCE Chip Select.
- RST Reset pin.
- D/C Input mode selection, either Command (low) or Data (High).
- DN(MOSI) and SCLK, both inputs for the SPI. Arduino UNO Hardware SPI pins will be used for this purpose, making the trasmission faster.
- LED, which can be pluged in a PWM pin, so the dim or bright can be set through the Arduino.

The lines are supposed to work at 3V, so limiting resistors will be used to buck the 5V lines in Arduino UNO.

As indicated on the PCD8544 controller datasheet [9], the instruction for SPI protocol are:

"The instruction format is divided into two modes: If D/C (mode select) is set LOW, the current byte is interpreted as command byte. Figure 4.28 shows an example of a serial data stream for initializing the chip. If D/C is set HIGH, the following bytes are stored in the display data RAM. After every data byte, the address counter is incremented automatically. The level of the D/C

signal is read during the last bit of data byte. The serial interface is initialized when SCE is HIGH. In this state, SCLK clock pulses have no effect and no power is consumed by the serial interface. A negative edge on SCE enables the serial interface and indicates the start of a data transmission." [9]

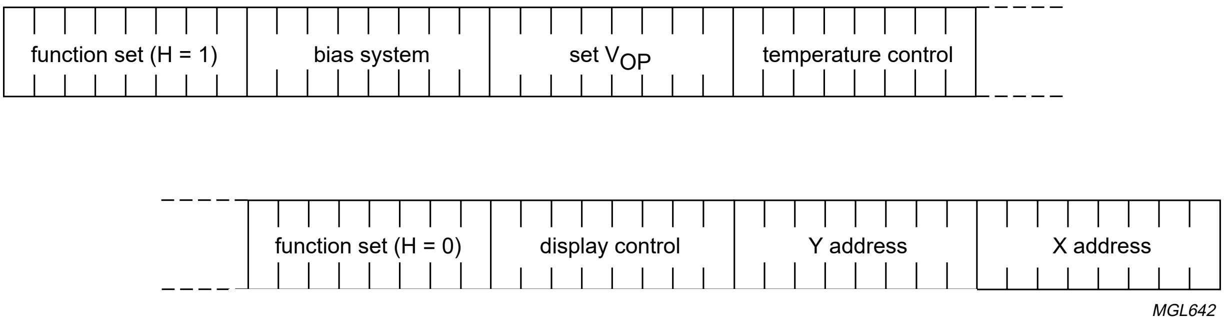


Figure 4.28 – *Serial Data stream, PCD8544. Source [9]*

Chapter 5

Software Design

Along this section, the main Software developed will be analysed. The code has been developed for Arduino Uno, therefore for flashing the device the Arduino IDE was used during the realisation of the project. The chose of the Arduino Uno as controller for this project was influenced also for the amount of libraries avaliable online, as seen in section 4.1.

5.1 IMU Calibration and Complementary Filtering

As seen before, in section 4.3 devoted to the Adafuit 10DOF utilisation in this project, the main issues when coding regarding the correct operation of the IMU are:

- The actual gyroscope avaliable in the 10DOF breakout bought wasn't the original one it was supposed to be. For this reason, the libraries avaliable for this breakout couldn't be used and had to be modified.
- The I2C addresses of the components within the 10DOF weren't programmable. Two IMUs were needed to get an accurate tilt angle. To communicate with both, only one HW I2C bus was avaliable in Arduino Uno, so a SW I2C bus was developed through the library [https://github.com/felias-fogg/SoftI2CMaster——SoftwareI2C Master Library].
- The calibration and signal filtering had to be implemented in the code, once it was understood the theorical framework, explained in sections 4.3.1 and 4.3.2.

5.1.1 Software i2c

For solving the first problem, an i2c scanner code was used in order to obtain the real i2c address of the component on the 10DOF breakout bought. One of the addresses found was 0x69, which wasn't suposed to be within the addresses found. A look online showed that this address is used for the Gyroscope L3G4200D, which is actually the one on the breakouts. The specified library for this Gyroscope is then used for sensing the angular velocity.

In terms of the implementation of the Software I2C needed for the utilisation of the two IMUs, the libraries for each component within the breakout had to be adapted for working with both, HW and SW I2C.

When debugging the Arduino, the sensors have to be initialised, making sure there is a pair of each one, one on the HW I2C bus, and another on the SW I2C bus. To do so, I2C protocol will be inisialised and each component will be enable and set up as slaves. To probe the identity of each component, the WHOAMI (Who am I?) register will be checked for the Gyroscope, and CTRL_REG1_A; CRA_REG_M will be checked for the Accelerometer and Magnetometer. Example for the magnetometer:

```
uint8˙t reg1˙a = read8(LSM303˙ADDRESS˙MAG,
    LSM303˙REGISTER˙MAG˙CRA˙REG˙M);
if (reg1˙a != 0x10) // the default value is (0b00010000/0x10)
–
           return false;
”
```

5.1.2 Tilt Estimation

As explained in [11], *to estimate the tilt angle of the body* θ_{face}*, a accelerometer-based tilt estimation was implemented using two accelerometers*. As the two accelerometers are placed along the diagonal of the pendulum body, there will only be angular acceleration on two axis, as seen in figure 5.1.

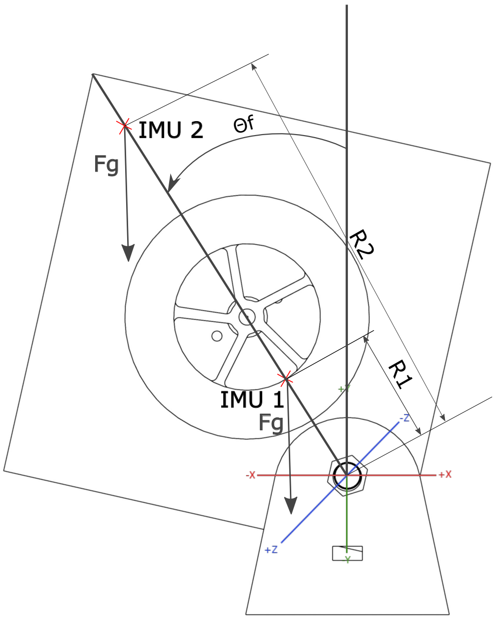


Figure 5.1 – *2D Model of the Balancing System IMUs along the diagonal.*

For our purpose, the IMUs will be place so the Z Axis is perpendicular to the body face, so no acceleration will be sensed in this axis when balancing. Therefore, the gravity affects only on the X and Y axis, and the angular acceleration of the system to both axis, as follows [11]:

$$ai^m := (r_x\ddot{\theta}_b + g\sin\theta_b, -r_i\dot{\theta}_b^2 - g\cos\theta_b, 0), i = 1, 2 \tag{5.1.1}$$

$$a_1m - \mu_{a2}m = ((1-\mu)g\sin\theta_b, -(1-\mu)g\cos\theta_b, 0) = (m_x, m_y, 0) \tag{5.1.2}$$

where $\mu = r_1/r_2$ *and the estimated tilt angle of the pendulum body is given by* $\hat{\theta}_b = \tan^{-1}(-m_x/m_y)$ [11]

Everytime the data is read from the accelerometers, this calculations have to be made in order to get the actual tilt angle of the body, according to the accelerometers reading. This is achieved by the following code:

```
//Trigonometry
mx=accelerationX1-4.17*accelerationX2; //mx=a1mx-(r1/r2)a2mx
my=-(accelerationY1-4.17*accelerationY2); //my=a1my-(r1/r2)a2my
angle=-mx/my;
theta˙b=atan (angle)-kappa;  // arc tangent of -mx/my ; kappa
   will be explained in function calibrate2imu().
theta˙b˙degree=theta˙b*360/(2*3.1416);
```

5.1.3 Calibration

To calibrate the IMUs on every debugging, a function was developed regarding the theory explained in section 4.3.1. The tilt angle of the system is referenced by the balancing point, where the θ_{face} has to be 0. To do so, when debugging the Arduino, the system has to be placed around the balancing point, so we can get the $kappa$ angle. Also, Gamma will be defined as the angle between the gravitational acceleration read by two IMUs, needed for making the vector paralels by using a rotation matrix. The function is called calibrate2imu().

```
void calibrate2imu() –
int i;
//Calculate gamma, the angle between the two IMUs and Kappa,
   the angle correction around the balancing point.
x1=0;
x2=0;
y1=0;
y2=0;
for (i=1;i¡4;i++)– //Three measures are taken, so the mean can
   be made later on, getting higher accuracy.
```

```
          sensors˙event˙t eventSWi2c;
          acceli2c.getEvent(&eventSWi2c);

          sensors˙event˙t eventHWi2c;
          accel.getEvent(&eventHWi2c);

          x1 += eventSWi2c.acceleration.x;
          y1 += eventSWi2c.acceleration.y;
          x2 += eventHWi2c.acceleration.x;
          y2 += eventHWi2c.acceleration.y;
”

x1=x1/3;
y1=y1/3;
x2=x2/3;
y2=y2/3;

gamma=((x1-x2)/(y2));

//The gravitational vector read from the IMU in the SWi2c bus
    is made paralel to the one on the HWi2c bus, by multiplying
    for the rotational matrix.
x1=cos(gamma)*x1+sin(gamma)*y1;
y1=-sin(gamma)*x1+cos(gamma)*y1;

eme1=x2-(4.17*x1);
eme2=-(y2-(4.17*y1));
float angl=-eme1/eme2;
kappa=atan(angl);

”
```

5.1.4 Complementary Filter

As seen in section 4.3.2, making a weighted sum of the measures given by the accelerometer and the gyroscope, we managed to obtain a value with a very low noise similar to that the one given by the gyroscope but with a temporary correction of the bias thanks to the measure given of the accelerometer.

The values of the proportionality constants were obtained in a empirical way:

$$\theta = 0.96\theta_{gyro} + 0.04\theta_{acc} \qquad (5.1.3)$$

Therefore, the complementary filter implementation can be coded as:

```
//Complementary filter implementation
deltaAcc=theta˙b-AnguloComp; //The angle just read and
    calculated by the tilt angle estimation algorithm minus the
    old angle.
deltaGyro=-norm.YAxis*samplingtime; //Angular acceleration of
    the body in the Y axis by the sampling time.
AnguloComp+=0.96*deltaGyro+0.04*deltaAcc; //Proportionality
    constants obtained in a empirical way.
AnguloComp˙deg=AnguloComp*360/(2*3.1416);
```

5.1.5 Code Diagram

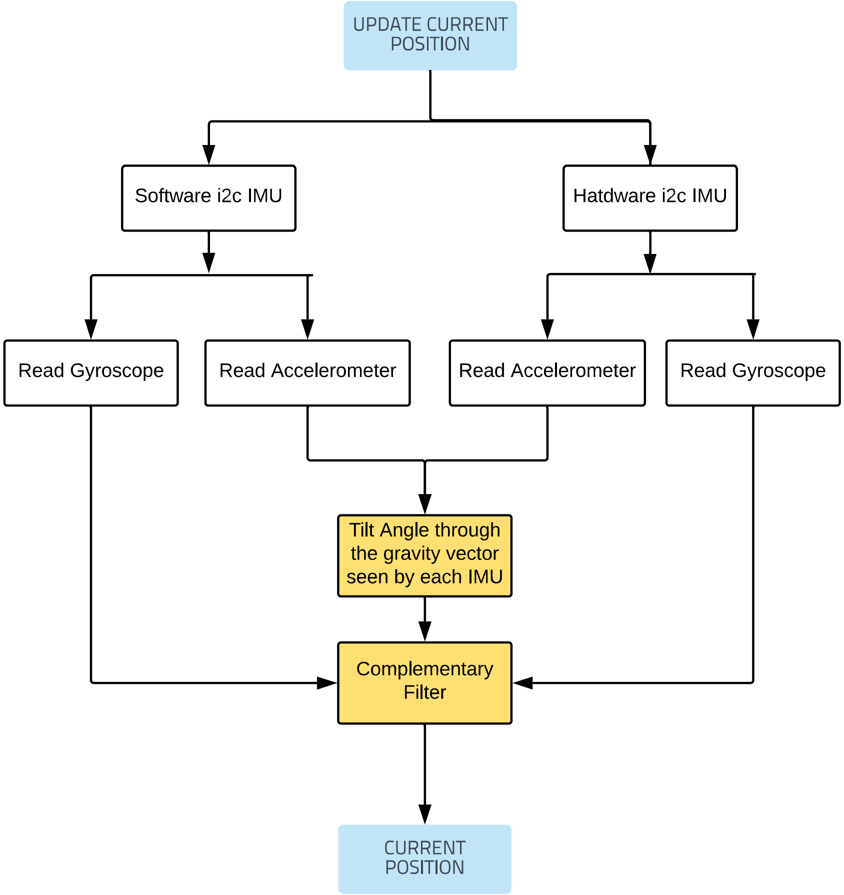
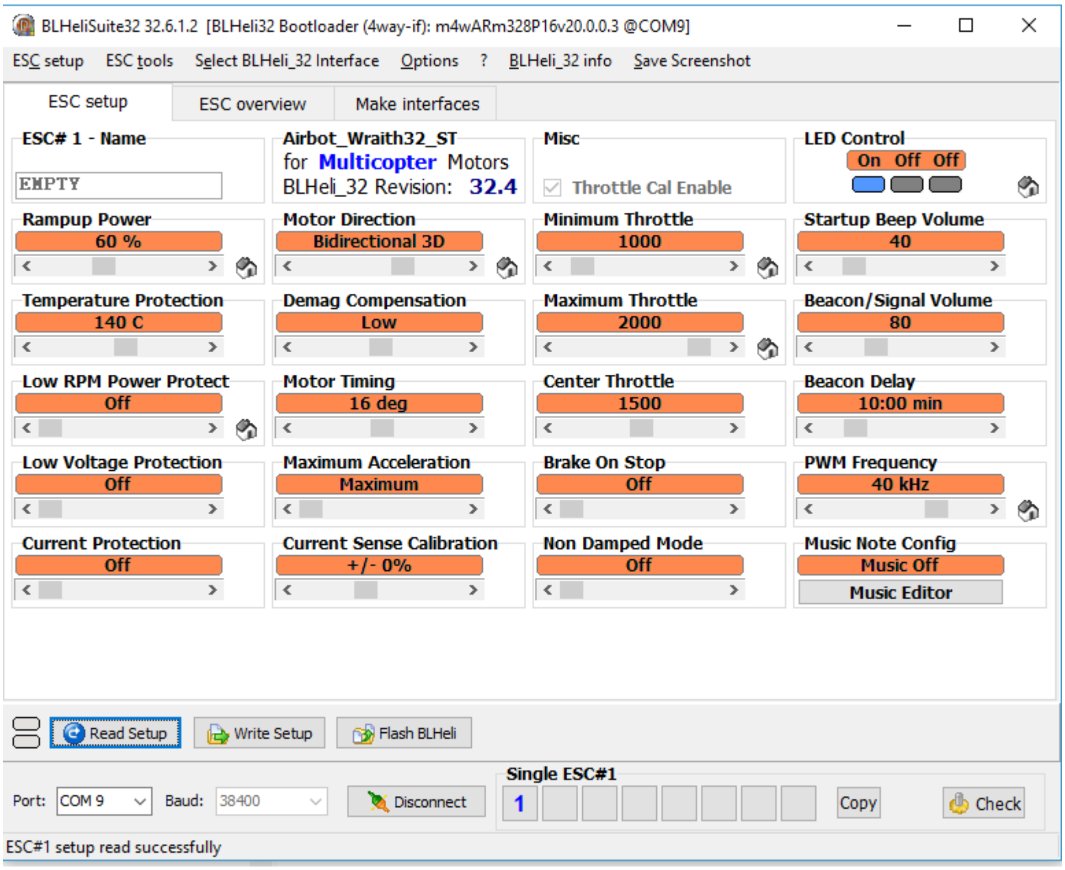


Figure 5.2 – *Updating Position code diagram.*

5.2 Inner Control Communication

As seen in Section 4.6, the Airbot Wraith32 ST ESC variator was chosen for this project, for it's low price and weight and the good features. The firmware update shown in Figure 5.3 set up the minimum and maximun throttle. Hence, figure 5.4 shows the communication protocol between a Pulse Width Modulation wave an the output of the ESC.



ESC# 1 - Name: EMPTY

Airbot_Wraith32_ST
for **Multicopter** Motors
BLHeli_32 Revision: **32.4**

Misc: ☑ Throttle Cal Enable

LED Control: On Off Off

Rampup Power: 60 %	Motor Direction: Bidirectional 3D	Minimum Throttle: 1000	Startup Beep Volume: 40
Temperature Protection: 140 C	Demag Compensation: Low	Maximum Throttle: 2000	Beacon/Signal Volume: 80
Low RPM Power Protect: Off	Motor Timing: 16 deg	Center Throttle: 1500	Beacon Delay: 10:00 min
Low Voltage Protection: Off	Maximum Acceleration: Maximum	Brake On Stop: Off	PWM Frequency: 40 kHz
Current Protection: Off	Current Sense Calibration: +/- 0%	Non Damped Mode: Off	Music Note Config: Music Off

Port: COM 9   Baud: 38400

Single ESC#1: 1

ESC#1 setup read successfully

Figure 5.3 – *BLHeli 32 Firmware for the ESC.*

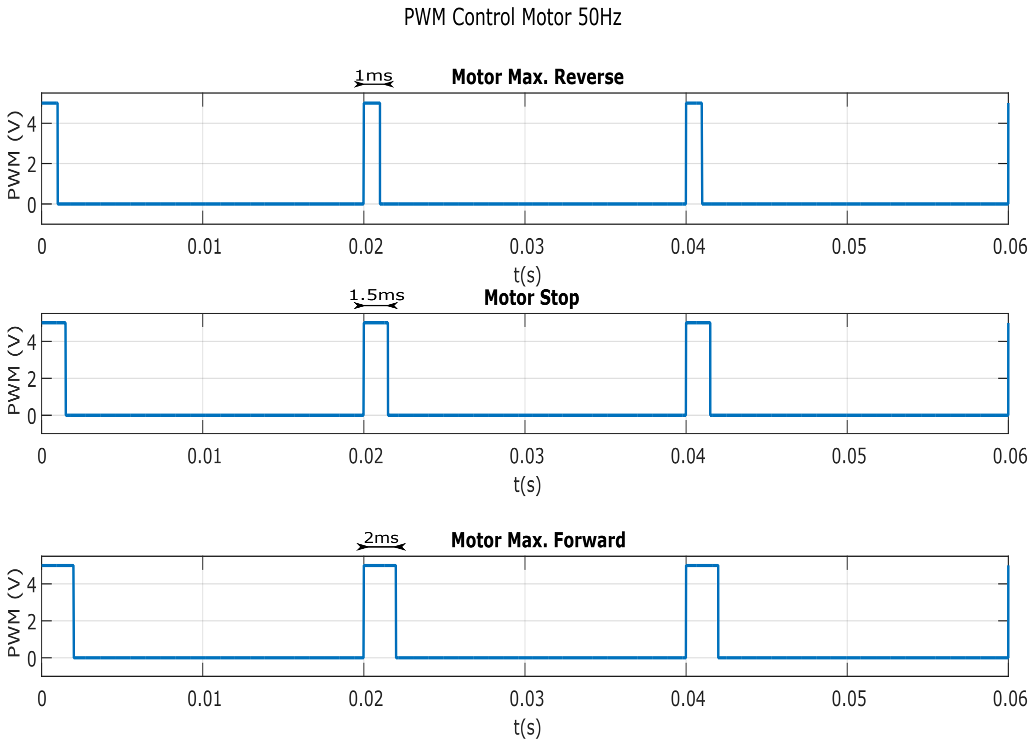


Figure 5.4 – *50Hz PWM control wave*

For succesfully communicating with the ESC, a 50Hz carrier wave was needed. To do so, thanks to the library Servo.h [23], we just have to declare a Servo object, called ESC in our code. It is meant to be used for communication with servos, which use a 50Hz carrier wave, for a Pulse Width Modulated signal. During the set up some adjustment has to be made:

```
//ESC Initation
ESC.attach(10,1000,2000); //The PWM pin is chose, IO10. Minimum
    Throttle is 1000ns and maximun Throttle is 2000ns. This
    Servo.h library allows us to map this 1000ns-2000ns range to
    0-180. Hence, if a 1000ns width pulse is needed, ESC.write
    (0) would make it.

ESC.write(0); //On every debugging for communication with the
    Airbot Wraith32, updated with BLHELI32 firmware, the minimum
    and maximum throttle have to be sent.
delay(10);

ESC.write(90); //Maximun throttle for forward sense of rotation
    CCW.
delay(10);
```

```
”
```

Later in the main code, whenever a pulse width is needed, the funcion ESC.write(PWMValue) can be used. From 0 to 90 PWMValue the sense of rotation will be CCW and from 90 to 180, ACW. This comes from the explanation shown in figure 5.4, always regarding the throttle set in the BLHELI firmware update, figure 5.3.

5.3 PID Implementation in Arduino

For the realisation of the code, the basic principle of PID controller was asumed and inplemented in the code. As seen in 2.4, the PID control algorithm digital aproximation is:

$$P(k) = K_p E(k) + K_i T \sum_{i=1}^{k-1} E(i) + K_d \frac{E(k-1) - E(k-2)}{T} \tag{5.3.1}$$

A function was created to be called from the Main loop everytime it runs, so the PID algorithm is executed. The inputs are the current position, read from the IMU and properly adjusted and filtered; the target position, that can be set by the IR remote controller seen in section 4.5; and the threshold, which determines the range around the target point where the error is accumulated for the integral control.

```
void PID(int targetPosition , float currentPosition , int thresh)
–
error = targetPosition - currentPosition;
if (abs(error) ¡ thresh)
–
        Integral += error; //Accumulate Error
”
else
–
        Integral = 0;
”

P = Kp * error;
I = Ki * Integral;
D = Kd * (error - last˙error);
last˙error = error;
correction = P + I + D;

motor˙pwm = constrain(correction , -90, 90);
```

```
ESC.write(90+motor˙pwm); //The maximum power CCW is transmitted
    to the ESC through ESC.write(0), as explained in the
   previous section devoted to the ESC coding. The maximum ACW
   power is transmitted by ESC.write(180). Hence, as the
   correction is constrained to -90  90 range, adding 90 to its
    value will make it transmittable.
”
```

5.4 Counting RPMs

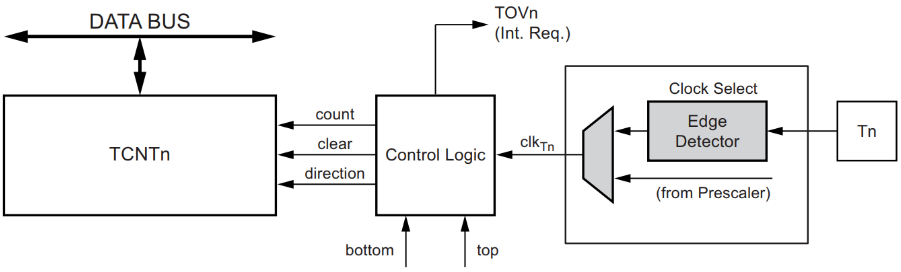


Figure 5.5 – *Counter2 ATmega328 block diagram.[7]*

Some registers have to be programmed in order to use the collector in the CNY70 as the input in Counter 2 of ATMega 328 (Arduino Uno HW). During the setup of the Arduino program, this code has to be added, in order to count the rising edges on the counter 2, which will be connected to the collector of the CNY70 phototransistor. The count will be saved in register TCNT2, as seen in figure 5.5.

The code needed for programming the Counter 2 to read from the Pin D5 was understood from the source [28].

```
DDRD &= (1 ¡¡ DDD5); // Clear the PD5 pin
// PD5 is now an input

PORTD —= (1 ¡¡ PORTD5); // turn On the Pull-up
// PD5 is now an input with pull-up enabled

TCCR2B —= (1 ¡¡ CS12) — (1 ¡¡ CS11) — (1 ¡¡ CS10);
// Turn on the counter, Clock on Rise
```

Once the Counter is set up for reading rising edges from the Pin D5 in Arduino Uno, and the CNY70's collector is connected to this pin, the count has to be transformed into RPMs. To do so, we make use of the sampling time of our system. As explained before, in section 5.3, the samplig time is determined in the code, so the data read from the sensors is actualized every exact amount of milliseconds. Therefore, if we divide the number of rising edges counted in the counter's register (TCNT2) by the time it has been counting, we can get the number of spins per milliseconds. The code for the implentation is described in funcion rpmget(), which returns the rpm of the Inner Wheel:

```
float rpmget(void)
–
        // Update sensor readings
        if (oldcount¡TCNT2) //If we have gotten to its maximum
           (which is unlikely to happen) then we start counting
            by 0.
        –
                oldcount=0;
        ”
        oldcount=counter;
        counter=TCNT2;
        Serial.println(counter-oldcount);
        rpms=(oldcount-counter)/(SamplingTIME); //Revolutions
           per millisecond
        rpm=rpms*(60*1000) //Revolutions per minute
        Serial.println(rpm);
        return rpm;
”
```

5.5 IR Receiver

The IR Receiver will be used as the interface between the user and the balancing cube. For that purpose, a remote controller will be given to the user. By pressing buttons the target point will be set and it can be turned off and on. The NEC protocol, shown in figure 5.6 is used for decoding the infrared signal sent by the remote controller.

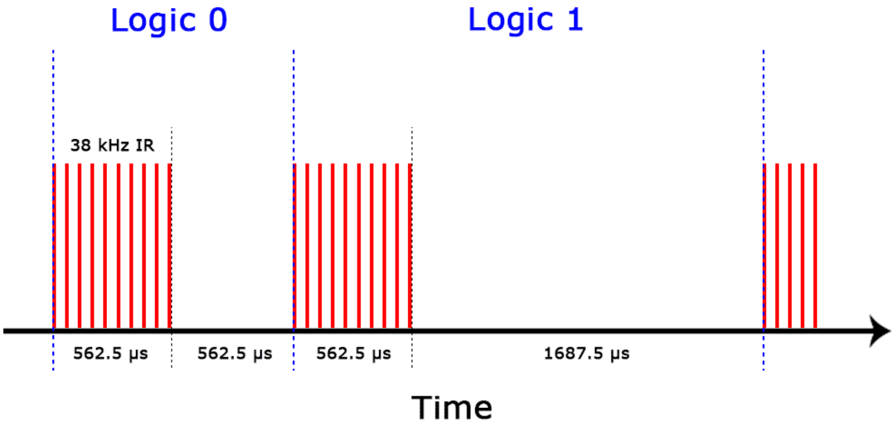


Figure 5.6 – *NEC protocol. Source [8]*

The IR Receiver is commonly used with the Arduino, hence, some libraries were avaliable. For our purpose the library IRremote.h [29] will be used.

The code is basically this [30]:

```
int RECV˙PIN = 7; // define input pin on Arduino. As shown in
   the connector diagram, for our purpose IO7 will be used.
```

```
IRrecv irrecv(RECV˙PIN);
decode˙results results; // decode˙results class is defined in
   IRremote.h

void setup() –
        irrecv.enableIRIn(); // Start the receiver
”

void loop() –
        if (irrecv.decode(&results)) – // Check if there are
           received data. The code is saved in results.value.
        irrecv.resume(); // Receive the next value
        if(result.value==0xFF02FD)– //Forward button has been
           pushed (¿¿)
        targetPosition=targetPosition+1;
        ”

        if(result.value== 0xFF22DD)– //Backward button has been
            pushed (¡¡)
        targetPosition=targetPosition+1;
        ”

        if(result.value== 0xFF22DD)– //Backward button has been
            pushed (¡¡)
        targetPosition=targetPosition+1;
        ”

        if(result.value==  0XFF6897)– //If 0 button has been
           pushed, then we stop the system
        stopper=1;
        ”

        if(result.value==  0XFF6897)– //If 1 button has been
           pushed, then we turn on the system
        stopper=0;
        ”
        ”
”
```

The code for each key in the remote controller is shown in Table ??, found in [29]:

Key	Code	Purpose
<<	0xFF22DD	Target Point +1°.
>>	0xFF02FD	Target Point -1°.
0	0XFF6897	Stop the motor.
1	0xFF30CF	Start balancing around the Target Point.

5.6 Main Function

A diagram was developed so the final main loop code could be easily understood.

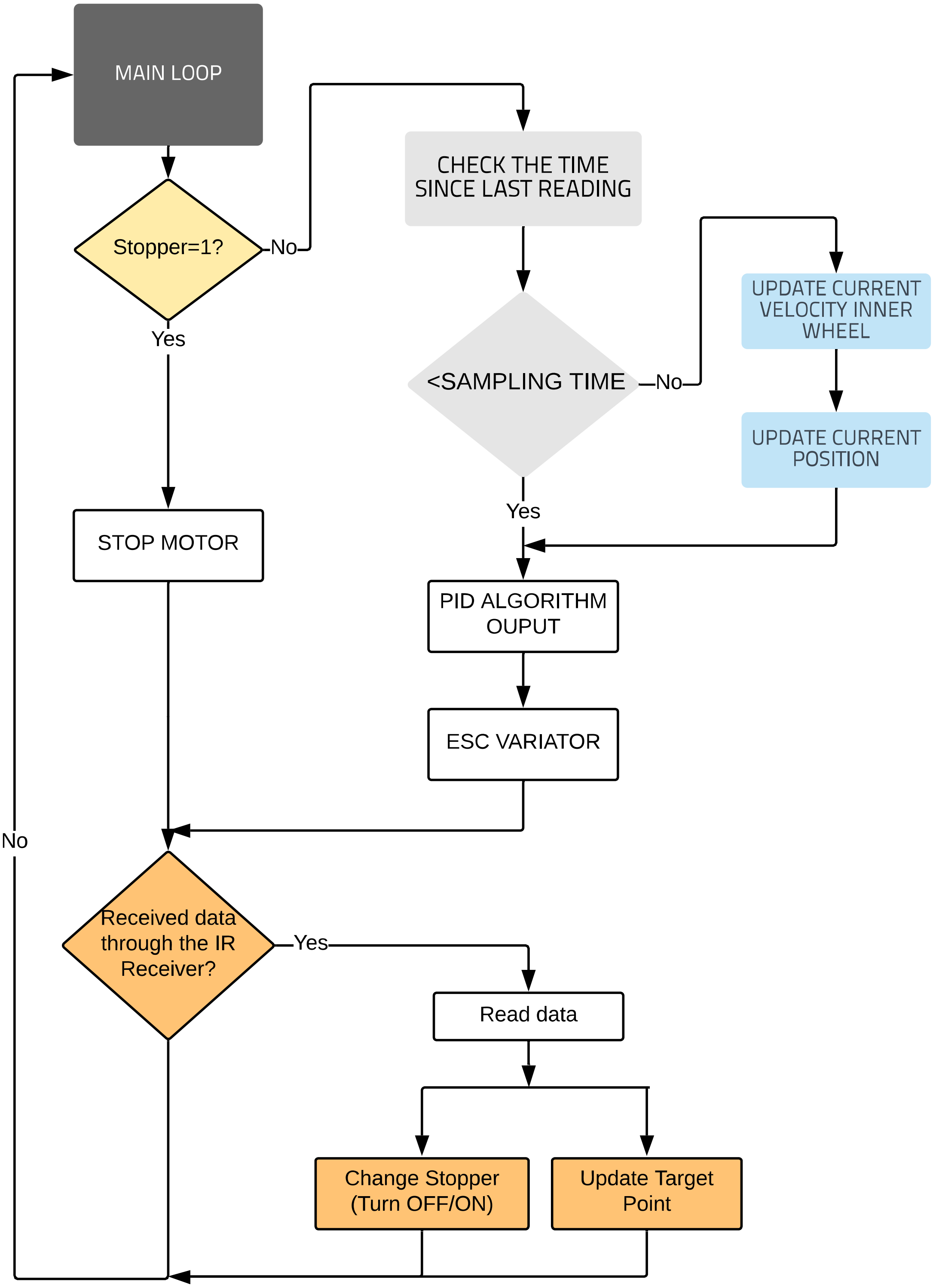


Figure 5.7 – *Main Loop code diagram.*

The main loop code consists basically in implementing all the function described before in the previous sections. After the setup loop is executed, the code goes to a loop that keeps doing till it is disconnected.

On every time the main loop is executed, if the stop variable hasn't been set to 1, we check if the time since the last time the sensors were read. To do so, everytime the sensors are read, a variable is set with the current time, millisends precision. The current time on every execution of the loop is compared then whith the variable where we saved the last time the sensors were read. If the last time the sensors were read is now bigger than the sampling time, the current position and the velocity of the wheel are updated. That means that these values are updated each sampling time milliseconds.

This current time is now used as the input of the PID algorithm, which creates an output. This output, as said before in section 5.2, is a PWM signal which is sent to the ESC variator, which controls the motor.

Now it is checked if there was any received information in the IR receiver, as said in section 5.5. If the stop button has been pushed, then we set the variable *"stopper"* to 1, meaning that at the beginning of the loop the sampling time won't be checked; the sensors' information won't be updated; and the motor will just stop. On the contrary, if te start button is pushed, the *"stopper"* is set to 0, allowing the normal PID algorithm based inner control. Also, the target point of the system can be modified from the IR remote controller.

The code is exposed:

```
void setup(void)
–
        //10DOF IMU (SW i2C and HW i2C)
        //  Serial.println(F(”Adafruit 10DOF two magnetometers
           and two accelerometers working on different i2c
           buses (HW and SW)”)); Serial.println(””);


        /* Initialise the sensors */
        if(!bmp.begin())
        –
                /* There was a problem detecting the BMP085 ...
                   check your connections */
                Serial.print(”Ooops, no BMP085 detected ...
                   Check your wiring or I2C ADDR!”);
                while(1);
        ”
        else
        –
                //    Serial.print(”Bien! Sensor T y Presión
                   BMP085 detectado.!“n”);
```

```
˝

/* Initialise the sensors */
if(!acceli2c.initSoftwareI2C(&WireS1, 3, 2))  //
   initSoftwareI2C, sda, scl
–
        /* There was a problem detecting the ADXL345
           ... check your connections */
        Serial.println(F("Ooops, no LSM303 detected ...
            Check your wiring!"));
        while(1);
˝
else
–
        //     Serial.print("Bien! Sensor Acelerómetro
           LSM303 detectado mediante SW i2c.!“n");
˝


if(!magi2c.initSoftwareI2C(&WireS1, 3, 2)) //
   initSoftwareI2C, sda, scl
–
        /* There was a problem detecting the LSM303 ...
            check your connections */
        Serial.println("Ooops, no LSM303 detected ...
           Check your wiring!");
        while(1);
˝
else
–
        //     Serial.print("Bien! Sensor Magnetómetro
           LSM303  detectado mediante SW i2c.!“n");
˝

if(!accel.begin())
–
        /* There was a problem detecting the ADXL345
           ... check your connections */
        Serial.println(F("Ooops, no LSM303 detected ...
            Check your wiring!"));
        while(1);
˝
else
–
        //     Serial.print("Bien! Sensor Acelerómetro
           LSM303 detectado mediante HW i2c.!“n");
˝
```

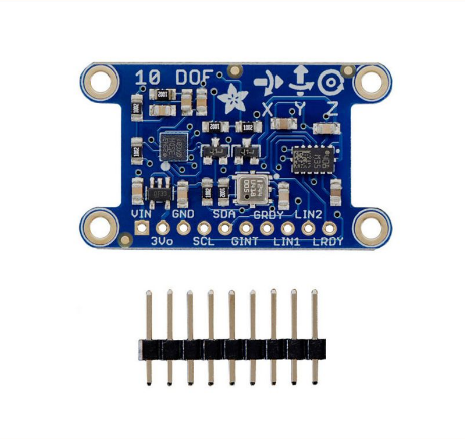
```
if(!mag.begin())
–
        /* There was a problem detecting the LSM303 ...
            check your connections */
        Serial.println(”Ooops, no LSM303 detected ...
           Check your wiring!”);
        while(1);
”
else
–
        //     Serial.print(”Bien! Sensor Magnetómetro
           LSM303  detectado mediante HW i2c.!“n”);
”
// Initialize L3G4200D
//  Serial.println(”Initialize L3G4200D”);
// Set scale 2000 dps and 400HZ Output data rate (cut-
   off 50)
if(!gyro.begin(L3G4200D˙SCALE˙2000DPS,
   L3G4200D˙DATARATE˙400HZ˙50))
–
        /* There was a problem detecting the L3G4200D
           ... check your connections */
        Serial.print(”Ooops, no L3G4200D gyroscope
           detected ... Check your wiring or I2C ADDR!”
           );
        while(1);
”
else
–
        //     Serial.print(”Bien! Sensor Giróscopo
           L3G4200D detectado.!“n”);
”
// Calibrate gyroscope. The calibration must be at rest
   .
// If you don't want calibrate, comment this line.

//  gyro.calibrate(100);
deltaGyro=0;
AnguloComp=0;
displaySensorDetails();
calibrar2imu();


//ESC Initation
ESC.attach(10,1000,2000);
```

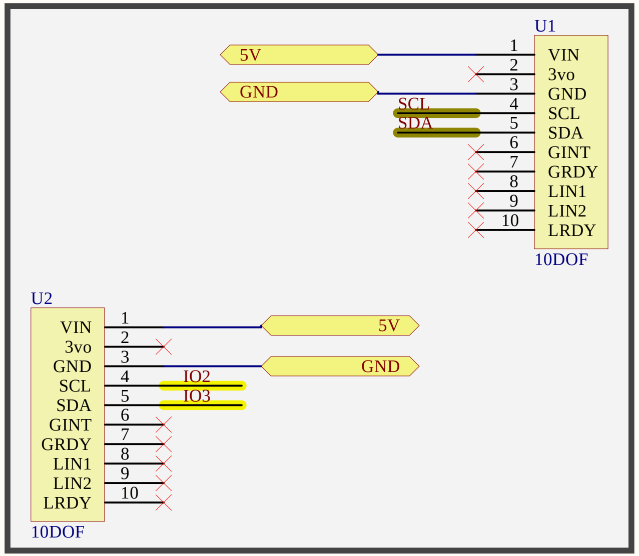
```
          //   Serial.print(0);
          ESC.write(0);
          delay(10);
          //   Serial.print(90);
          ESC.write(90);
          delay(10);

          //Start measuring time
          timestamp = millis();

          //Counter1 setup for CNY70

          DDRD &= ˜(1 ¡¡ DDD5);      // Clear the PD5 pin
          // PD5 is now an input Digital Pin n 5 in Arduino Uno

          PORTD —= (1 ¡¡ PORTD5);    // turn On the Pull-up
          // PD5 is now an input with pull-up enabled

          TCCR2B —= (1 ¡¡ CS12) — (1 ¡¡ CS11) — (1 ¡¡ CS10);
          // Turn on the counter on Rise

          irrecv.enableIRIn(); // Start the receiver
"


// Main loop
void loop()
–
          if (stopper=1)
          –
                    ESC.write(90); //Stop the motor
          "
          if (stopper=0)
          –
                    // Time to read the sensors again?
                    if((millis() - timestamp) ¿=
                       OUTPUT¨DATA˙INTERVAL)
                    –
                              timestamp = millis();
                              // Update sensor readings
                              currentPosition=updatecurrentposition()
                                  ; //Update Current Position
                              rpms=rpmget(); //Update RPMs
                    "
                    PID(targetPosition, currentPosition, thresh);
          "
          if (irrecv.decode(&results)) – // Check if there are
             received data. The code is saved in results.value.
                  irrecv.resume(); // Receive the next value
                  if(result.value==0xFF02FD)– //Forward button
                     has been pushed (¿¿)
                          targetPosition=targetPosition+1;
                  ”
                  if(result.value== 0xFF22DD)– //Backward button
                     has been pushed (¡¡)
                          targetPosition=targetPosition+1;
                  ”
                  if(result.value== 0xFF22DD)– //Backward button
                     has been pushed (¡¡)
                          targetPosition=targetPosition+1;
                  ”
                  if(result.value==  0XFF6897)– //If 0 button has
                      been pushed, then we stop the system
                          stopper=1;
                  ”
                  if(result.value==  0XFF6897)– //If 1 button has
                      been pushed, then we turn on the system
                          stopper=0;
                  ”
          ”

  ”
```

Chapter 6

Final PCB Design. Altium Designer.

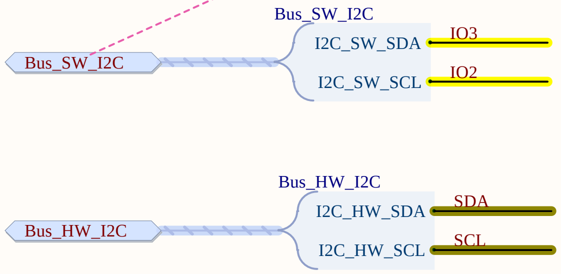
ADAFRUIT 10DOF IMU:





DESIGN NOTE:

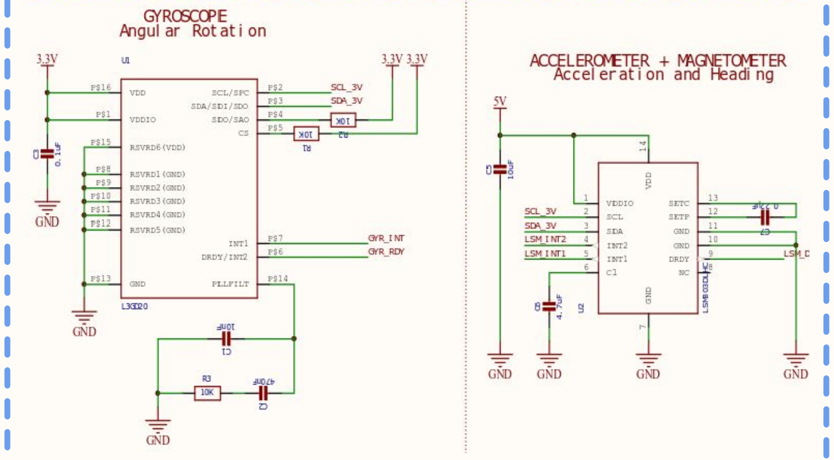
SoftwareI2C protocol was needed in order to communicate with one of the 10DOF IMUs. The accelerometers and gyroscopes had the same non-configurable I2C address so only one IMU could be use on each I2C bus.



DESIGN NOTE:
For our purpose, due to the geometry and high precission needed, two Adafruit 10DOF IMUs were needed. This breakout settled on the following devices:

- LSM303DLHC - a 3-axis accelerometer (up to +/-16g) and a 3-axis magnetometer (up to +/-8.1 gauss) on a single die.
- L3GD20 - a 3-axis gyroscope (up to +/-2000 dps).
- BMP180 - A barometric pressure sensor (300..1100 hPa) that can be used to calculate altitude, with an additional on-board temperature sensor.

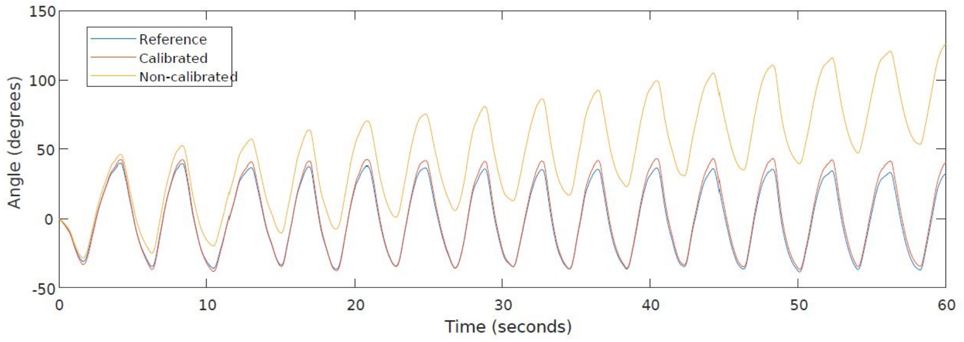
DEVICES USED TO GET ORIENTATION



DESIGN NOTE: CALIBRATION:

- An accelerometer measures the acceleration of the body along it's three axes. The accelerometer measurements are subjected to a large amount of noise compared to the gyroscope, and it varies depending on the static or dynamic state of the body, getting larger as the change rate of the acceleration vector grows. A quick method to calibrate the accelerometer is to use the orthogonality property of the axes to determine the maximum gravity output value for each individual axis. When the positive and negative value of acceleration is computed for a given axis, both the scaling and steady-state error can be obtained.

- A gyroscope measures the angular velocity vector of the system. In particular, in the process of obtaining the orientation of a given body, the gyroscope signal is integrated over time. To obtain the steady-state error, the sensor signal is measured over a period of 120 seconds (20.000 iterations), and a mean of -1.867°/s is obtained. This value has to be substracted to each measurement of the angular velocity.



Designer's signature

Supervisor's signature

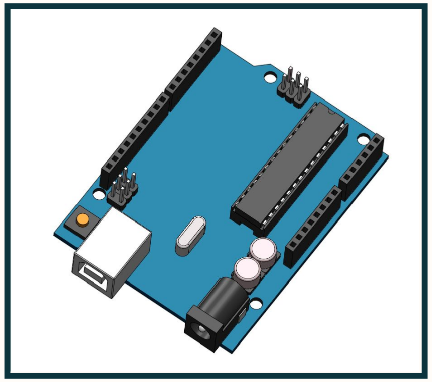
Sheet title: **Adafruit 10DOF IMU**

Project title: **CubeSatMomentumControl.PrjPcb**

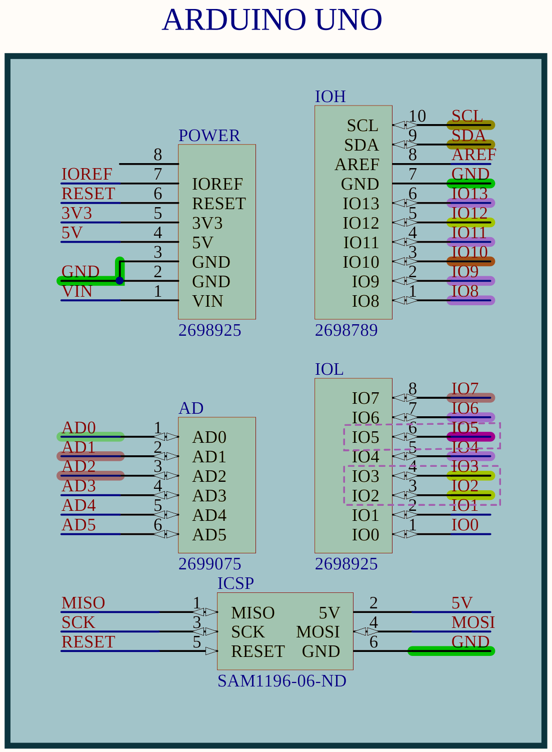
Desginer: **Juan Aparicio Jiménez**

Date: **11/06/2019** Revision: ***** Sheet * of *

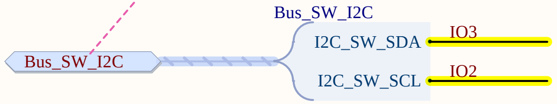
Dpto. Electrónica y Tecnología
de Computadores
University of Granada
C/ Fuente Nueva, s/n, 18001
Granada, Granada, Spain
Sr. Andrés Roldán Aranda

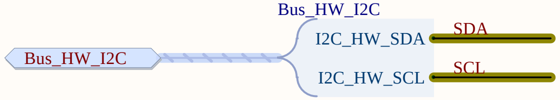


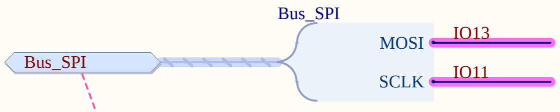
ARDUINO UNO CONNECTIONS



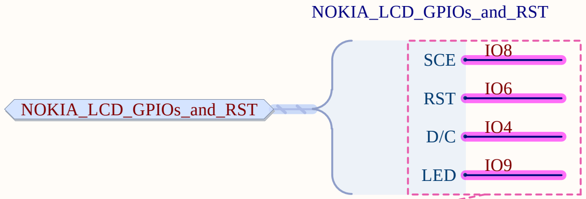
DESIGN NOTE:
GPIOs 2-3 are used as SDA-SCL in a SoftwareI2C protocol in order to communicate with one of the 10DOF IMUs. The accelerometers and gyroscopes had the same non-configurable I2C address so the SofwareI2C was needed.



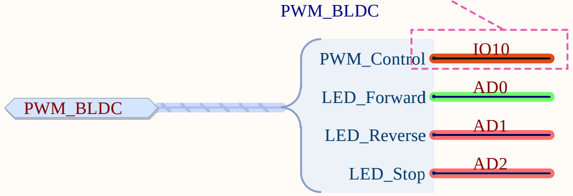




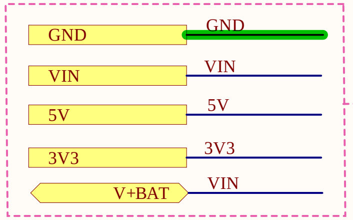
DESIGN NOTE:
For the data transmission with the Nokia 5100 LCD, pins -- SCLK and DN(MOSI) -- Arduino's hardware SPI pins will be used, which will help to achieve a faster data transfer.



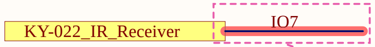
DESIGN NOTE:
GPIO 10 (Arduino Uno) will be used for controlling the input PWM signal, as well as for any change or updating on the Wraith 32 ESC firmware, through BLHELI 32. GPIO 12 will be used to communicate a software Rx implemented in Arduino Uno, Hardware Rx and Tx will be used for communication with the computer.



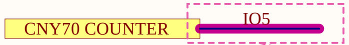
DESIGN NOTE:
GPIOs 8-6-4-9 are used for: chip select (SCE) , reset (RST), data/command (D/C). The LED pin should be connected to a PWM-capable Arduino pin, so we can dim the backlight as we please, therefore IO9 was chosen.



DESIGN NOTE:
The Arduino Uno will be powered through a 11.1V LIPO battery. Input Voltage (recommended) 7-12V. Using the internal regulator in Arduino Uno, both IMUs, the CNY70 and the LCD will be powered. Further details will be explained in Power schematic.



DESIGN NOTE:
GPIO 7 is used for the signal given by the KY-022 IR Receiver.



DESIGN NOTE:
GPIO 5 will be connected to Counter 1 in Arduino uno in order to count the rising edges happening every time the wheel spins.

Designer's signature

Supervisor's signature

Sheet title: **Arduino UNO**

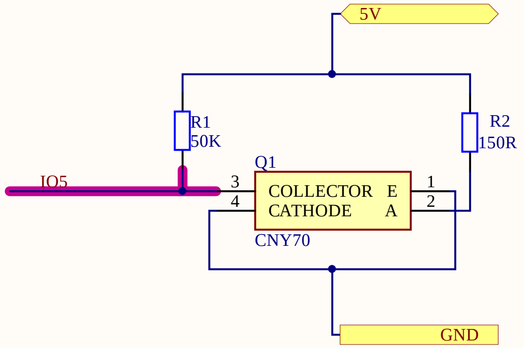
Project title: **CubeSatMomentumControl.PrjPcb**

Desginer: **Juan Aparicio Jiménez**

Date: **11/06/2019** Revision: ***** Sheet * of *

Dpto. Electrónica y Tecnología
de Computadores
University of Granada
C/ Fuente Nueva, s/n, 18001
Granada, Granada, Spain
Sr. Andrés Roldán Aranda

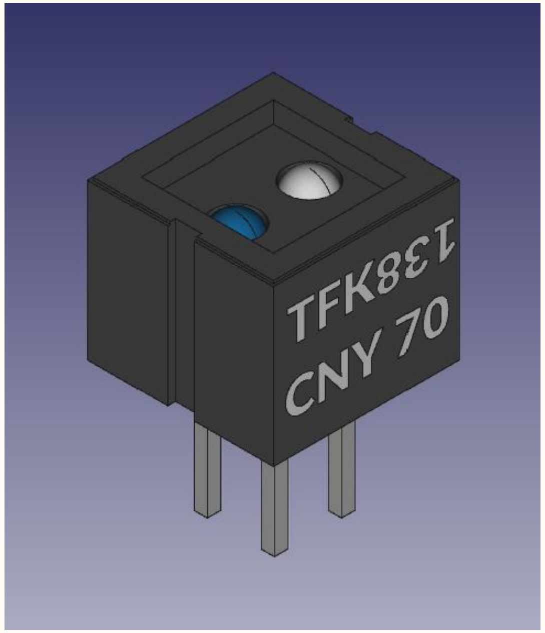
CNY70 TACHOMETER

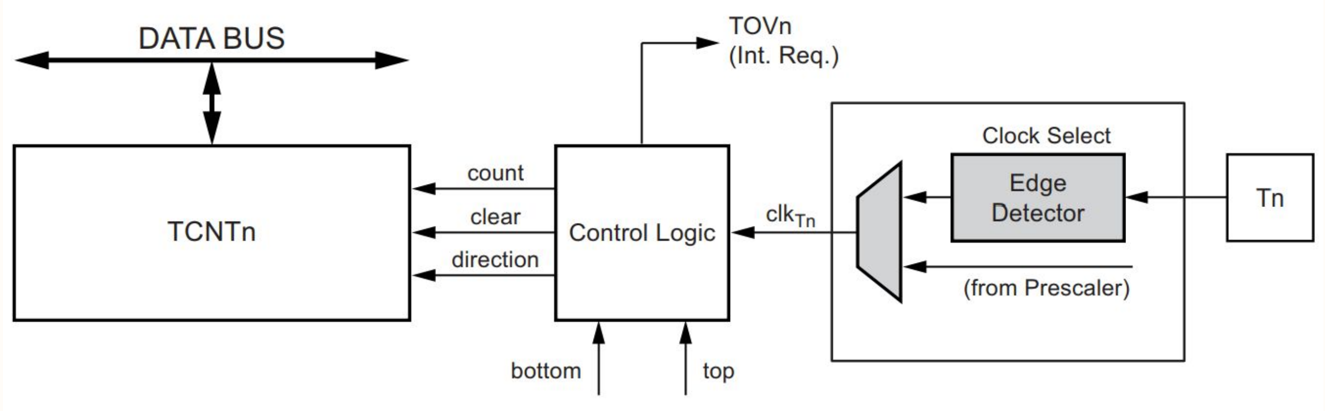


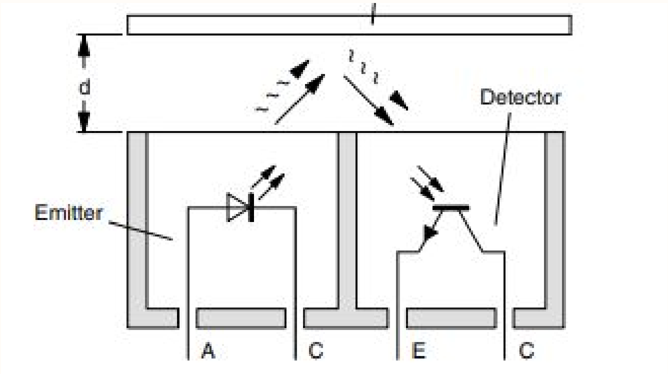
A CNY70 was chosen for measuring the spinning velocity of the Inner Wheel.

CNY70 contain a IR-emitting diode as transmitter and a phototransistors as receiver. The transmitter emit radiation of a wavelength of 950 nm. The working method is basically this, the light emitted by the transmitter is influenced by an object or a medium on its way to the detector. The change in the light signal caused by the interaction with the object then produces a change in the electrical signal in the optoelectronic receiver.

We place four black stickers along the Inner Wheel, so on every spin there will be four changes in the electrical signal. IO5 is connected to the phototransistor's collector. Counter 2 on the Arduino Uno will then count every rising edge on IO5, and therefore the number of spins within a meassured time can be known.







CNY70 COUNTER
IO5

DESIGN NOTE:
GPIO 5 will be connected to Counter 1 in Arduino uno in order to count the rising edges happening every time the wheel spins.

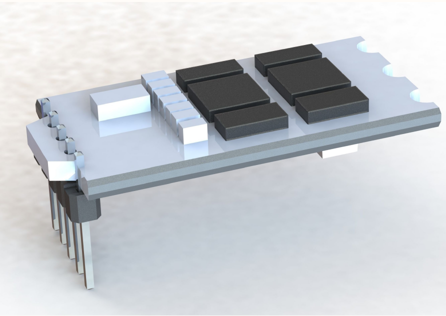
DESIGN NOTE:
Some registers have to be changed in order to use the collector in the CNY70 as the input in Counter 2 of ATMega 328 (Arduino Uno HW).

```
DDRD &= (1 << DDD5); // Clear the PD5 pin // PD5 is now an input

PORTD |= (1 << PORTD5); // turn On the Pull-up
// PD5 is now an input with pull-up enabled

TCCR1B |= (1 << CS12) | (1 << CS11) | (1 << CS10);
 // Turn on the counter, Clock on Rise
```

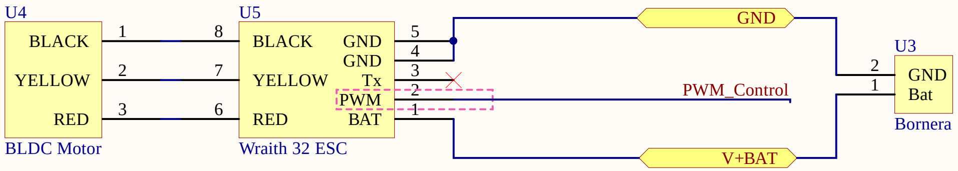
Designer's signature	Sheet title: **CNY 70**		*Dpto. Electrónica y Tecnología de Computadores* *University of Granada* *C/ Fuente Nueva, s/n, 18001* *Granada, Granada, Spain* *Sr. Andrés Roldán Aranda*
	Project title: **CubeSatMomentumControl.PrjPcb**		
Supervisor's signature	Desginer: **Juan Aparicio Jiménez**		
	Date: **11/06/2019** Revision: *****	Sheet * of *	

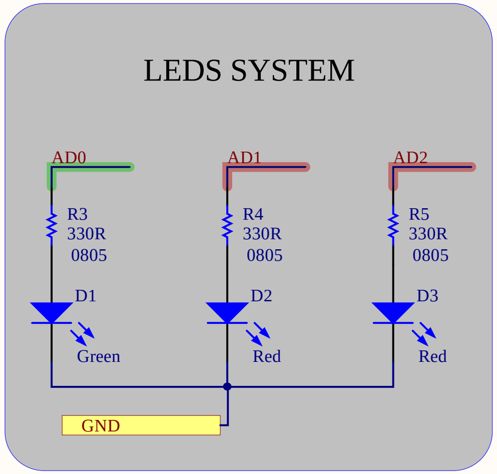
BLDC Motor Control

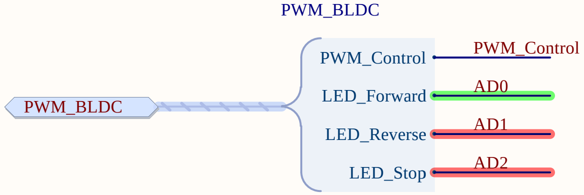
3 Phase BLDC Motor

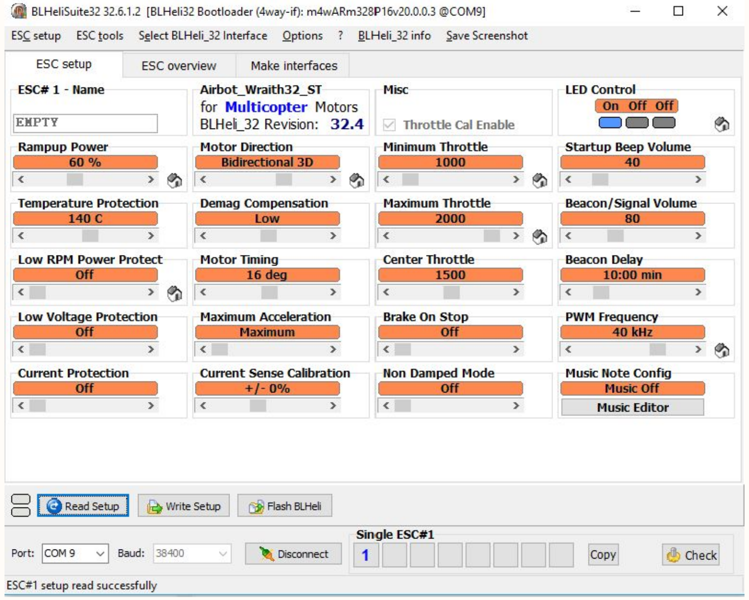


Airbot Wraith32 ST









ESC# 1 - Name: EMPTY

Airbot_Wraith32_ST
for **Multicopter** Motors
BLHeli_32 Revision: **32.4**

Misc: ☑ Throttle Cal Enable

LED Control: On Off Off

Rampup Power: 60 %
Motor Direction: Bidirectional 3D
Minimum Throttle: 1000
Startup Beep Volume: 40
Temperature Protection: 140 C
Demag Compensation: Low
Maximum Throttle: 2000
Beacon/Signal Volume: 80
Low RPM Power Protect: Off
Motor Timing: 16 deg
Center Throttle: 1500
Beacon Delay: 10:00 min
Low Voltage Protection: Off
Maximum Acceleration: Maximum
Brake On Stop: Off
PWM Frequency: 40 kHz
Current Protection: Off
Current Sense Calibration: +/- 0%
Non Damped Mode: Off
Music Note Config: Music Off

Port: COM 9 Baud: 38400

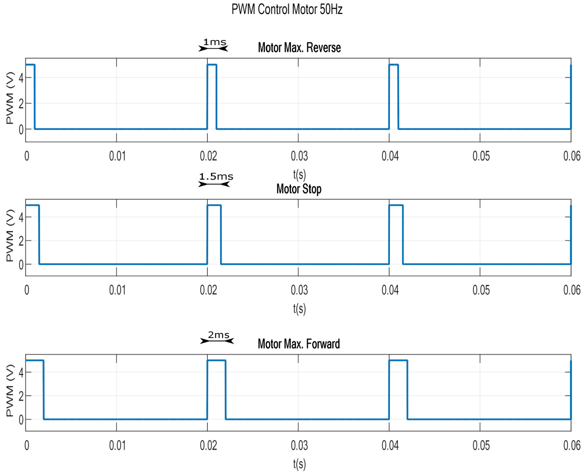
Single ESC#1: 1

ESC#1 setup read successfully

DESIGN NOTE:
A firmware update fot the Wraight32 controller was needed in order to make the BLDC motor work properly for our purpouse. Once the Wraight32 is connected to the Arduino an update can be done through the Software BLHELI32, avaliable in:

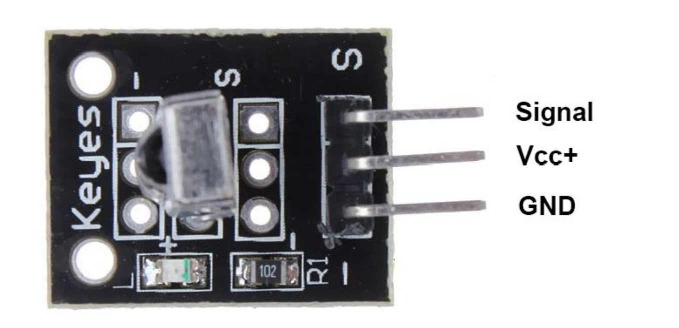
https://github.com/bitdump/BLHeli/blob/master/BLHeli_32%20ARM/BLHeli_32%20manual%20ARM%20Rev32.x.pdf

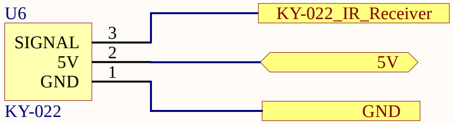
The parameters shown in the screenshot were the parameters chosen for a bidirectional working mode, smooth stop and high torque. Some other parameters where chosen according to the working mode needed and the programming parameters shown in GitHub.



Designer's signature	Sheet title: **BLDC Motor Control**	*Dpto. Electrónica y Tecnología de Computadores* *University of Granada* *C/ Fuente Nueva, s/n, 18001* *Granada, Granada, Spain* *Sr. Andrés Roldán Aranda*
	Project title: **CubeSatMomentumControl.PrjPcb**	
Supervisor's signature	Desginer: **Juan Aparicio Jiménez**	
	Date: **11/06/2019** Revision: ***** Sheet * of *	

KY-022 INFRARED RECEIVER





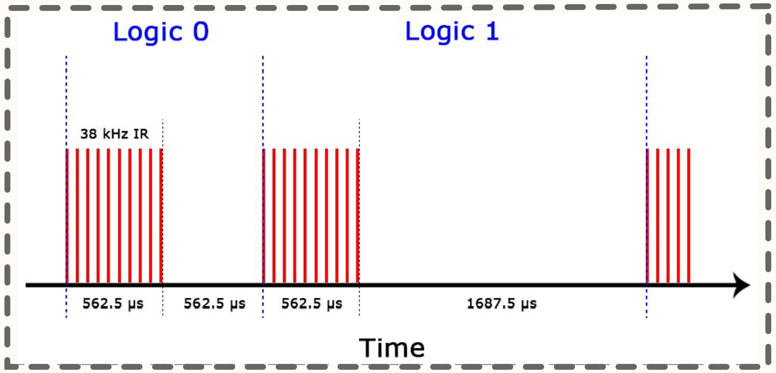
KY-022 IR Receiver is used for IR communication with a remote controller.

Each key on the remote controller has an unique coding. In order to make the system start/stop and move the target point when balancing the IR controller seemed like an easy and practical way on control.

The signal is trasmitted to the Arduino Uno through this device. The modulated IR signal is a series of IR light pulses switched on and off at a high frequency known as the carrier frequency.

The pattern in which the modulated IR signal is converted to binary is defined by a transmission protocol. The NEC protocol used in our project Logical ‘1’ starts with a 562.5 µs long HIGH pulse of 38 kHz IR followed by a 1,687.5 µs long LOW pulse. Logical ‘0’ is transmitted with a 562.5 µs long HIGH pulse followed by a 562.5 µs long LOW pulse.

DESIGN NOTE:
The pattern in which the modulated IR signal is converted to binary is defined by a transmission protocol. The NEC protocol used in our project Logical ‘1’ starts with a 562.5 µs long HIGH pulse of 38 kHz IR followed by a 1,687.5 µs long LOW pulse. Logical ‘0’ is transmitted with a 562.5 µs long HIGH pulse followed by a 562.5 µs long LOW pulse.

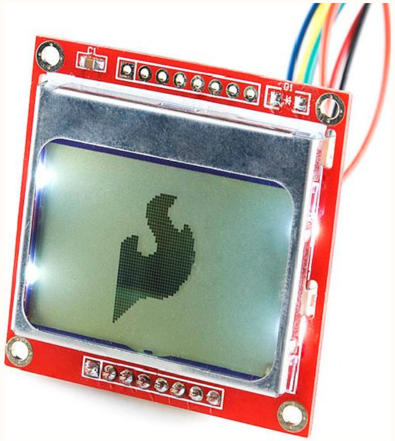


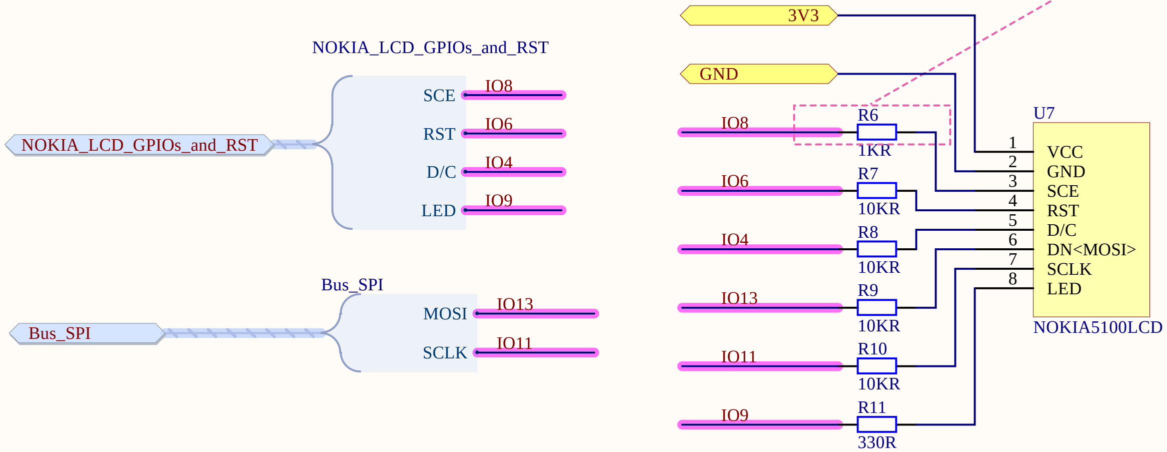
Parameter	Symbol	Test Conditions	Min	Typ	Mnx	Unit
Operating Voltage	Vcc		2.7		5.5	V
Receiving distance	L	L5IR = 300MA (test signal)	10	15		M
Carrier Frequency	f0		38K			HZ
Acceptance angle	01/2	Distance attenuation 1/2		+ / -35		Deg
BMP width	FBW	-3Db andwidth	2	3.3	5	kHz
Quiescent Current	Icc	When there is no signal input	----	0.8	1.5	mA
Low output	VOL	Vin = 0V Vcc = 5V		0.2	0.4	V
High-level output	VOH	Vcc = 5V	4.5			V
The output pulse width	TPWL	Vin = 500µVp-p ※	500	600	700	µs
	TPWH	Vin = 50mVp-p ※	500	600	700	µs

Designer's signature	Sheet title: **KY-022 IR Receiver**	*Dpto. Electrónica y Tecnología de Computadores* *University of Granada* *C/ Fuente Nueva, s/n, 18001* *Granada, Granada, Spain* *Sr. Andrés Roldán Aranda*
	Project title: **CubeSatMomentumControl.PrjPcb**	
Supervisor's signature	Desginer: **Juan Aparicio Jiménez**	
	Date: **11/06/2019** Revision: ***** Sheet * of *	

NOKIA 5100 LCD

This LCD has a maximum input voltage of 3.6V, so we can't hook up a standard 5V Arduino straight to it. Sticking resistors in-line with the data signals is an easy way to add some protection to the 3.3V lines.





The Nokia 5100 LCD is a 48 × 84 pixels matrix LCD. 4032 pixels (84*48=4032).
The PCD8544 is a low power CMOS LCD controller/driver, designed to drive a graphic display of 48 rows and 84 columns. All necessary functions for the display are provided in a single chip, including on-chip generation of LCD supply and bias voltages, resulting in a minimum of external components and low power consumption. The PCD8544 is manufactured in n-well CMOS technology.

The PCD8544 interfaces to microcontrollers through a serial bus interface. For the data transmission with the Nokia 5100 LCD, pins -- SCLK and DN(MOSI) -- Arduino's hardware SPI pins will be used, which will help to achieve a faster data transfer.

GPIOs 8-6-4-9 are used for: chip select (SCE) , reset (RST), data/command (D/C). The LED pin should be connected to a PWM-capable Arduino pin, so we can dim the backlight as we please, therefore IO9 was chosen.

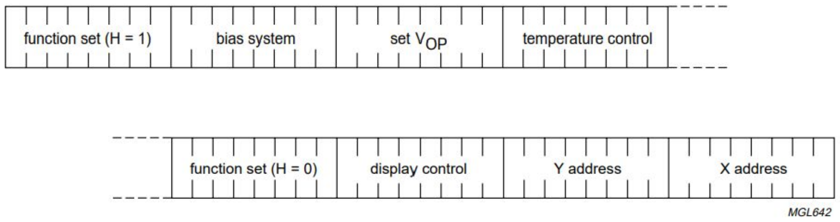
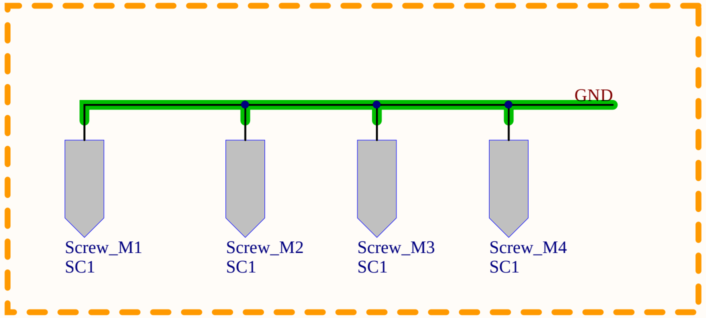


Fig.9 Serial data stream, example.

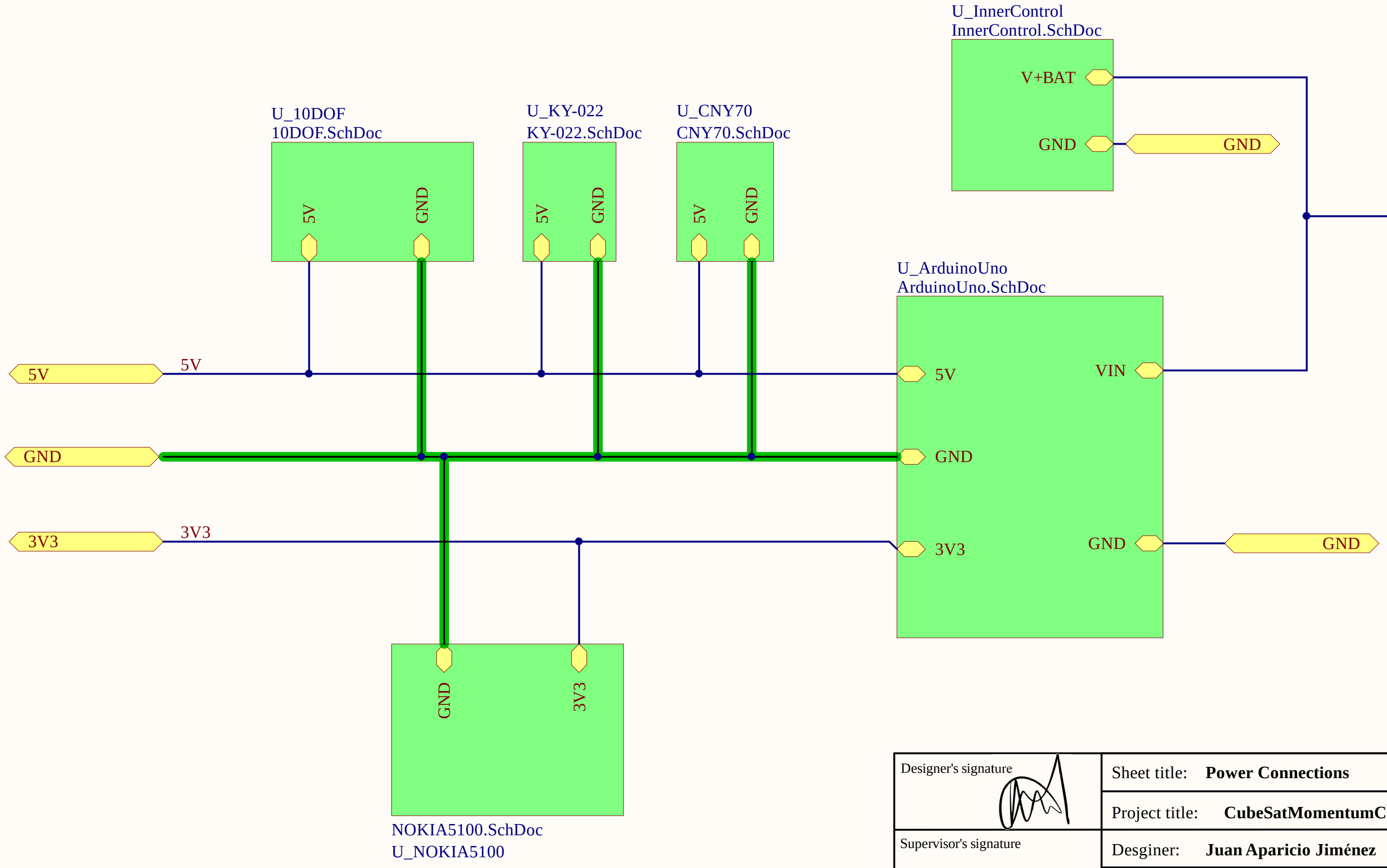
The serial interface is initialized when SCE is HIGH. In this state, SCLK clock pulses have no effect and no power is consumed by the serial interface. A negative edge on SCE enables the serial interface and indicates the start of a data transmission.
- When SCE is HIGH, SCLK clock signals are ignored; during the HIGH time of SCE, the serial interface is initialized.
- D/C indicates whether the byte is a command (D/C = 0) or RAM data (D/C = 1); it is read with the eighth SCLK pulse.
- If SCE stays LOW after the last bit of a command/data byte, the serial interface expects bit 7 of the next byte at the next positive edge of SCLK.
- A reset pulse with RES interrupts the transmission. No data is written into the RAM. The registers arecleared. If SCE is LOW after the positive edge of RES, the serial interface is ready to receive bit 7 of acommand/data byte.

Designer's signature	Sheet title: **Nokia 5100 LCD**		*Dpto. Electrónica y Tecnología de Computadores* *University of Granada* *C/ Fuente Nueva, s/n, 18001* *Granada, Granada, Spain* *Sr. Andrés Roldán Aranda*
	Project title: **CubeSatMomentumControl.PrjPcb**		
Supervisor's signature	Desginer: **Juan Aparicio Jiménez**		
	Date: **11/06/2019** Revision: *****	Sheet * of *	

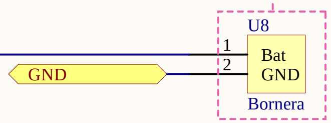
Power Connections



DESIGN NOTE:
For powering our PCB and the motor a 3S LIPO battery was chosen. It gives 11.1 volts and it has a relatively high power density, 1500mAh in 130g. Also, it squared shape allow us having the battey inside the steel CubeSat developed, making the system indepent from external power.



DESIGN NOTE:
A terminal will be place in order to connect the batery for powering the Arduino and the motor.



Designer's signature

Supervisor's signature

Sheet title: **Power Connections**

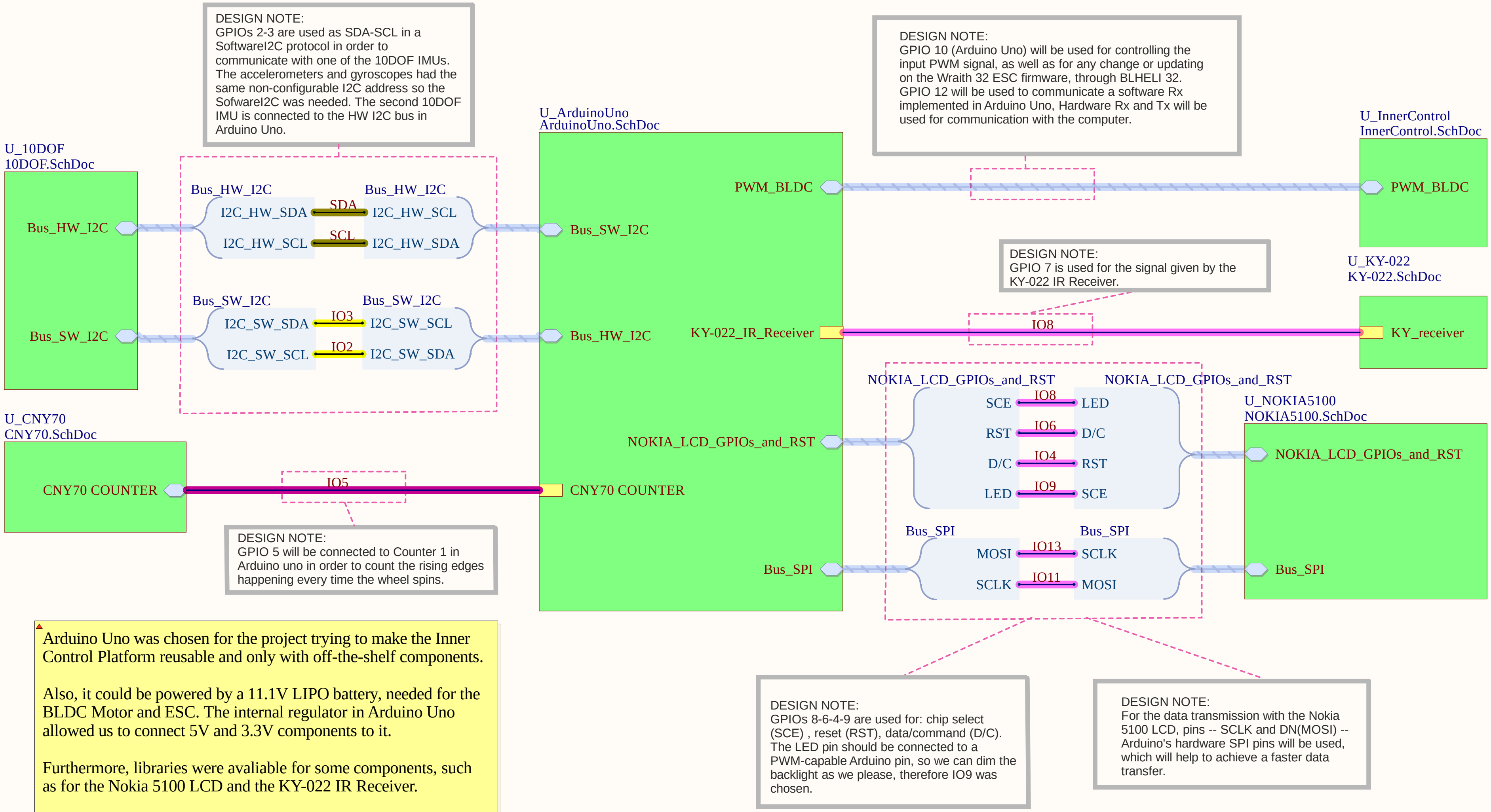
Project title: **CubeSatMomentumControl.PrjPcb**

Desginer: **Juan Aparicio Jiménez**

Date: **11/06/2019** Revision: ***** Sheet * of *

Dpto. Electrónica y Tecnología de Computadores
University of Granada
C/ Fuente Nueva, s/n, 18001
Granada, Granada, Spain
Sr. Andrés Roldán Aranda

Signal Connections



DESIGN NOTE:
GPIOs 2-3 are used as SDA-SCL in a SoftwareI2C protocol in order to communicate with one of the 10DOF IMUs. The accelerometers and gyroscopes had the same non-configurable I2C address so the SofwareI2C was needed. The second 10DOF IMU is connected to the HW I2C bus in Arduino Uno.

DESIGN NOTE:
GPIO 10 (Arduino Uno) will be used for controlling the input PWM signal, as well as for any change or updating on the Wraith 32 ESC firmware, through BLHELI 32. GPIO 12 will be used to communicate a software Rx implemented in Arduino Uno, Hardware Rx and Tx will be used for communication with the computer.

DESIGN NOTE:
GPIO 7 is used for the signal given by the KY-022 IR Receiver.

DESIGN NOTE:
GPIO 5 will be connected to Counter 1 in Arduino uno in order to count the rising edges happening every time the wheel spins.

DESIGN NOTE:
GPIOs 8-6-4-9 are used for: chip select (SCE) , reset (RST), data/command (D/C). The LED pin should be connected to a PWM-capable Arduino pin, so we can dim the backlight as we please, therefore IO9 was chosen.

DESIGN NOTE:
For the data transmission with the Nokia 5100 LCD, pins -- SCLK and DN(MOSI) -- Arduino's hardware SPI pins will be used, which will help to achieve a faster data transfer.

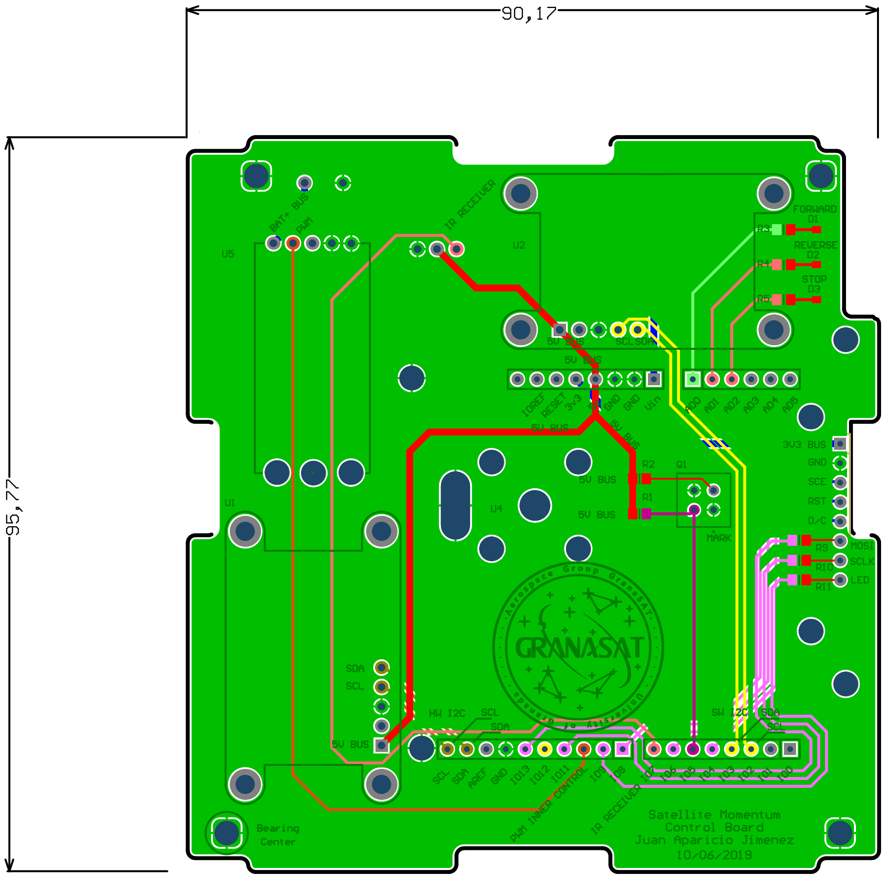
Arduino Uno was chosen for the project trying to make the Inner Control Platform reusable and only with off-the-shelf components.

Also, it could be powered by a 11.1V LIPO battery, needed for the BLDC Motor and ESC. The internal regulator in Arduino Uno allowed us to connect 5V and 3.3V components to it.

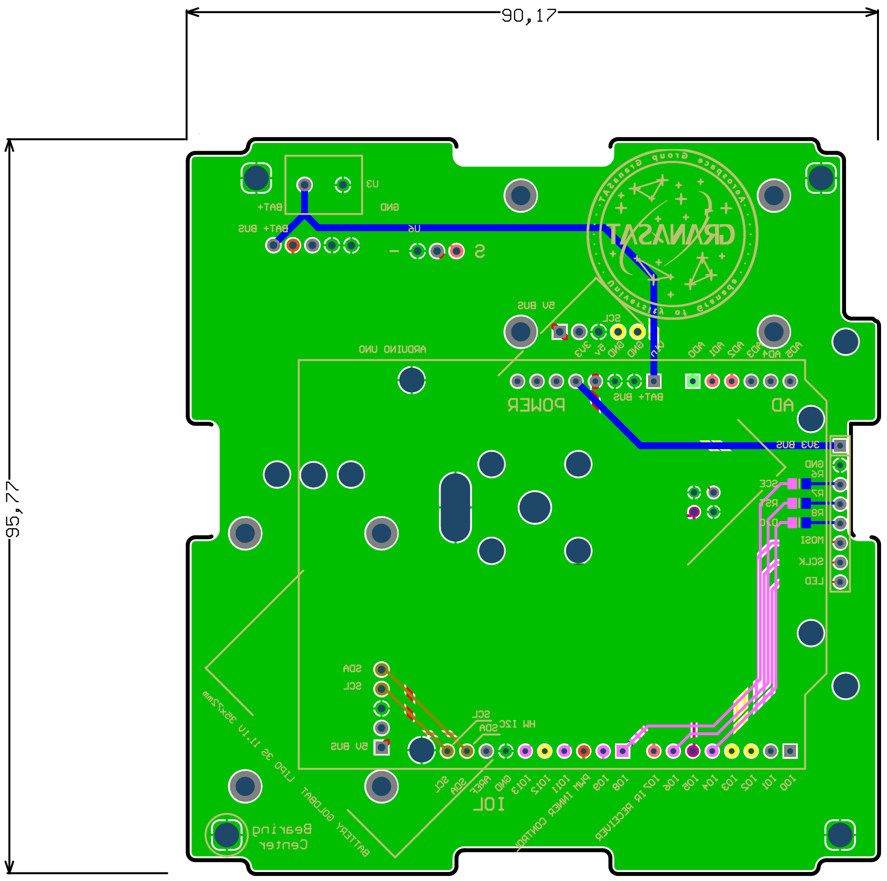
Furthermore, libraries were avaliable for some components, such as for the Nokia 5100 LCD and the KY-022 IR Receiver.

DESIGN NOTE:
General Power Connections are detailed on PowerConnections.SchDoc

Designer's signature	Sheet title: **Signal Connections**		*Dpto. Electrónica y Tecnología de Computadores* *University of Granada* *C/ Fuente Nueva, s/n, 18001* *Granada, Granada, Spain* *Sr. Andrés Roldán Aranda*
	Project title: **CubeSatMomentumControl.PrjPcb**		
Supervisor's signature	Desginer: **Juan Aparicio Jiménez**		
	Date: **11/06/2019** Revision: *****	Sheet * of *	



Designer's signature:	Sheet title: PCB	Dpto. Electronica y Tecnologia de Computadores University of Granada C/ Fuente Nueva, s/n, 18001 Granada, Granada, Spain Sr. Andres Roldan Aranda	
	Project title: GranaSAT		
Supervisor's signature:	Designer: Juan Aparicio Jimenez		
	Supervisor: Andres Roldan Aranda	**Date: 21/05/2019** **Revision: 2** **Sheet X**	



Designer's signature:	Sheet title: PCB	Dpto. Electronica y Tecnologia de Computadores University of Granada C/ Fuente Nueva, s/n, 18001 Granada, Granada, Spain Sr. Andres Roldan Aranda	
	Project title: GranaSAT		
Supervisor's signature:	Designer: Juan Aparicio Jimenez		
	Supervisor: Andres Roldan Aranda	**Date: 21/05/2019 Revision: 2** **Sheet X**	

Chapter 7

Conclusions & Future Work.

In this Degree Thesis an overview of all the necessary items for a successful development of a Inner Control System has been made, following a reasonable and useful path.

Fist, a deep understading of the Euler-Lagrange dynamics, with its associated parameters and equations was absolutely needed for developing the dynamics of the system. This came together with the need of characterization of the components, such as the motor's torque and its relation with the PWM signal created with an Arduino UNO. The fact of using the software Audacity and the jack in our computer for this purpose allowed me to understand a new way of approaching the problems, at the beginning only the oscilloscope seemed like the right tool to be used, but later on a whole new set of tools were avaliable in my personal knowledge. This happen similar with many of the things we developed in the lab.

The PID controlled simulink model will be really useful for future students in GranaSAT, only the parameters have to be set for each prototype, and the tunning will be easily accomplished. The previous knowledge in Matlab allowed me to understand and perfom the model, but it was really useful in terms of remembering thigs that had been learnt a while ago.

During the Mechanical Design, a few prototypes were designed and built. The aim of this was not only to get a better prototype on each iteration but also to learn using different tools. For each prototype a SolidWorks model was developed, including all the components within it. That meant many hours learning on using SolidWorks, not compulsory during the Bachelor Degree but really useful in terms of a future employment. Also, after being designed thanks to EDA tools, the prototypes were built. For this purpose, I had to learn using a milling CNC machine, with all it comes with, learning how to code, actually use the machine and its interface, the utilization of the software CNC402 that allows to simulate the path of the machine and move it later on... Moreover, many of the pieces designed were 3D printed, meaning that many parameters had to be set. I had never printed 3D models, so some of the objects weren't properly printed at the beginning, but the final result was the expected.

Many electronic components have been deeply analyzed and the code is now avaliable for future students. Components such as the IMU; the ESC variator, the CNY70... were in the lab, but had never been used for this purpose. The development of the signal merging filtering and the calibration was succesfull and usefull in terms of getting a proper tilt stimation. It happens similar with the rest of the components, the result were succesfull, and they are now avaliable for further utilisation.

A PCB which can fit in a model of CubeSat developed by GranaSAT has been design. That involved many hours using EDA software such as Altium and SolidWorks. The result is shown through renders and good quality schematics, that allow an easy comprehension of the utilisation of each component and its working mode. The Altium schematics were designed hierarchical for this reason, makin easy to understand how it works and the connections between the different components.

The fact of writting the Bachelor Thesis in English made it hard, but absolutely worth it. It allowed me to practice for a future employment were I would be assumed the writting abilies in English. Also, it was all typed and coded in LaTeX, never used before.

The proposed future work is a main part of this project, as it was all taken from zero towards the design of an attitude control board for future satellites developed by GranaSAT. The future guidelines that will allow the implementation of the knowledge achieved along this projects are:

- **Board Implementation:** Assembly the board, or a similar one for an specific satellites, as all the components are modelled and integrated in Altium.
- **Test & Verification:** Tune through the Simulink model and test the correct operation of the PID algorithm. Develop a LQR algorithm, by implementing the dynamics equation gotten for the system.
- **Update Firmware:** Some of the components used along this project are commonly used in some other areas, such as the ESC, that it's mainly used for drones. That means that firmware updates are being released really often, so it is interesting to keep the components updated, for a better operation and maybe a smoother control.

From a personal point of view, this Degree Thesis was a real challenge. Almost all the EDA software tools and fabrication proccess were unknown before starting digging online for tutorials. This meant many hours of dedication and learning, but the result was successful, achieving the objectives of the Thesis. Nevertheless, after this months of dedication I see myself more prepared for a successful professional career, where all the knowledge achieved during the realisation of this project will be surely emplyed.

Appendix A

Cutting data recommendations

Material	Rm/UTS (N/mm²)	Cutting speed Vc (m/min)		Feed per tooth (fz cutter ø)			
		coated	uncoated	2 - 4	5 - 10	11 - 16	>16
Metals, soft	400 -800 (120 - 310HB)	400	320	0,020	0,040	0,08	0,14
Aluminum alloys, long chipping	100 -400 (120 - 260HB)	1000	800	0,030	0,060	0,10	0,15
Aluminum alloys, short chipping	-400	300	240	0,030	0,060	0,10	0,15
Copper alloys, long chipping	150 -250 (160 - 230HB)	300	240	0,025	0,050	0,09	0,14
Copper alloys, short chipping	-500	250	200	0,025	0,050	0,09	0,14
Magnesium alloys	160 -300	400	320	0,030	0,060	0,09	0,14
Thermoplastics	350 -700 (150 - 280HB)	250	200	0,030	0,060	0,10	0,15
Duroplastics	20 -40	350	280	0,025	0,050	0,10	0,15
Graphite	-	400	320	0,040	0,080	0,15	0,20

Table A.1 – *Cutting data recommendations. Source: JHV Tools B.V. [10].*

Appendix B

Adafruit 10DOF HW and SW CODE

```
/*
 *
  ········    ········    ········    ········     ········    ········    ········
        ·········
___“     ····“___“    ··   “___“    ··   “___“    ···   “___“    ··   “___“    ····“___“    ··
    “___“···      ···“
 “  “   “···___“ “   “___“   “  “   “___“   “  “   ““ “   “  “   “___“   “  “   “···___“ “   “___“
    “___···  “   “·___
  “  “   “   ··“  “     ·   ·“  “    ··    “  “   ““ “   “  “    ··    “  “·····   “  “    ··
       “    “  “   “
   “  “   “___“   “  “   ““   ““ “   “  “   “  “   ““ “   “  “   “  “   “___····___“   “  “   “  “
         “    “  “   “
    “  “········ “  “··““ ·““ “··“ “··“ “··““ “··“ “··“ “··“·····“·“   “  “··“
        “··“    “  “··“
     “___·······___“___··___“___··___“___··___“___··___“___··___ “___··___“___··___“___··___“·········“___··
         ___“___··___     “___··___
                                                                                  “___·········___
 * Typing help from: http://patorjk.com/software/taag/#p=display&f=
    Graffiti&t=Type%20Something%20
 *
 * https://granasat.ugr.es
 *
 * Programmers:
 *   Prof. Andrés Roldán      1/03/2019 (amroldan@ugr.es)
 *   ·························  02/04/2018 (????????????@gmail.com)
 *
 *  Uso:
 *
 *   Esta versión inicial de uso del IMU 10 DOF  https://www.adafruit.
    com/product/1604
```

```
*   que tiene disponible:
*
*       LSM303 3-axis compass: +-1.3 to +-8.1 gauss magnetic field scale
*       LSM303 3-axis accelerometer: +-2g/+-4g/+-8g/+-16g selectable scale
*       BMP180 barometric pressure/temperature: -40 to 85 C, 300 -
   1100hPa range, 0.17m resolution
*
* Versions:
*
*  V02:
*    La dirección de los integrados es fija, y no hay patilla de
   configuración de direcciones. Así para poder leer de dos IMUs es
   necesario tener dos canales de
*    i2c diferentes. Para conseguirlo recurrimos a la librería
   SoftwareI2C.h, que nos permite crear un bus i2c virtual en los
   pines digitales que determinemos.
*    En nuestro caso utilizamos el pin digital 3 para SDA y 2 para SCL
   .
*
*    He modificado la biblioteca del accelerómetro y del magnetómetro
   (Adafruit˙LSM303˙U.h) para que, dependiendo de la clase que
   definamos, poder leer del
*    bus i2c HW del arduino, o del SW. Las funciones de lectura del
   magnetómetro y del accelerómetro son idénticas pero modificamos la
   interacción con el bus i2c.
*
*    Para modificar la libreria seguimos el tutorial que encontramos
   en: "http://wiki.seeedstudio.com/Arduino˙Software˙I2C˙user˙guide/".
*
*    Modificación preparada por Juan Aparicio Jiménez.
*
*  V01:
*    Metido la cabecera y la descripción de emails.
*    Versión portable del código para ver los valores de los sensores
   incluidos en el IMU, preparada por el prof. Andrés Roldán
*
* Scanning... I2C devices:  Estos son los dispositivos encontrados.
*  I2C device found at address 0x19  !    #define
   LSM303˙ADDRESS˙ACCEL           (0x32 ¿¿ 1)         // 0011001x
*  I2C device found at address 0x1E  !    #define LSM303˙ADDRESS˙MAG
              (0x3C ¿¿ 1)         // 0011110x
*  I2C device found at address 0x69  !
*  I2C device found at address 0x77  ! BMP085˙ADDRESS (0x77)
*  done
*/
```

```
/*
   ------------------------------------------------------------------------
   */
/*------------Librerías modificadas para soportar HW i2c y SW i2c
   simultánamente---------------*/
/*
   ------------------------------------------------------------------------
   */
#include ¡SoftwareI2C.h¿
SoftwareI2C WireS1; //We have to define SoftwareI2C object

#include "Adafruit_Sensor/Adafruit_Sensor.h"
#include "Adafruit_LSM303DLHCboth/Adafruit_LSM303_Uboth.h"
#include "Adafruit_LSM303DLHCboth/Adafruit_LSM303_Uboth.cpp"



/* Assign a unique ID to the SW i2c sensors */
Adafruit˙LSM303˙Accel˙UnifiedSWi2c acceli2c =
   Adafruit˙LSM303˙Accel˙UnifiedSWi2c(30301);
Adafruit˙LSM303˙Mag˙UnifiedSWi2c   magi2c   =
   Adafruit˙LSM303˙Mag˙UnifiedSWi2c(30302);

/* Assign a unique ID to the HW i2c sensors */
Adafruit˙LSM303˙Accel˙Unified accel = Adafruit˙LSM303˙Accel˙Unified
   (30301);
Adafruit˙LSM303˙Mag˙Unified   mag   = Adafruit˙LSM303˙Mag˙Unified
   (30302);


// Timers
unsigned long timer = 0;
float timeStep = 0.01;



void displaySensorDetails(void)
–
  sensor˙t sensor;

  acceli2c.getSensor(&sensor);
  Serial.println(F("———————— ACCELEROMETER ————————"));
  Serial.print  (F("Sensor:       ")); Serial.println(sensor.name);
  Serial.print  (F("Driver Ver:   ")); Serial.println(sensor.version);
  Serial.print  (F("Unique ID:    ")); Serial.println(sensor.sensor˙id)
     ;
  Serial.print  (F("Max Value:    ")); Serial.print(sensor.max˙value);
     Serial.println(F(" m/sˆ2"));
  Serial.print  (F("Min Value:    ")); Serial.print(sensor.min˙value);
     Serial.println(F(" m/sˆ2"));
  Serial.print  (F("Resolution:   ")); Serial.print(sensor.resolution);
      Serial.println(F(" m/sˆ2"));
  Serial.println(F("————————————————————————————————————"));
  Serial.println(F(""));


  magi2c.getSensor(&sensor);
  Serial.println(F("——————————— MAGNETOMETER ———————————"));
  Serial.print  (F("Sensor:       ")); Serial.println(sensor.name);
  Serial.print  (F("Driver Ver:   ")); Serial.println(sensor.version);
  Serial.print  (F("Unique ID:    ")); Serial.println(sensor.sensor˙id)
     ;
  Serial.print  (F("Max Value:    ")); Serial.print(sensor.max˙value);
     Serial.println(F(" uT"));
  Serial.print  (F("Min Value:    ")); Serial.print(sensor.min˙value);
     Serial.println(F(" uT"));
  Serial.print  (F("Resolution:   ")); Serial.print(sensor.resolution);
      Serial.println(F(" uT"));
  Serial.println(F("————————————————————————————————————"));
  Serial.println(F(""));
˝

void setup(void)
–
  Serial.begin(115200);
  Serial.println(F("Adafruit 10DOF two magnetometers and two
     accelerometers working on different i2c buses (HW and SW)"));
     Serial.println("");


  /* Initialise the sensors */
  if(!acceli2c.initSoftwareI2C(&WireS1, 3, 2))  // initSoftwareI2C, sda
     , scl
  –
    /* There was a problem detecting the ADXL345 ... check your
       connections */
    Serial.println(F("Ooops, no LSM303 detected ... Check your wiring!"
       ));
    while(1);
  ˝
  else
  –
  Serial.print(”Bien! Sensor Acelerómetro LSM303 detectado mediante
     SW i2c.!\n”);
˝


if(!magi2c.initSoftwareI2C(&WireS1, 3, 2)) // initSoftwareI2C, sda,
   scl
–
  /* There was a problem detecting the LSM303 ... check your
     connections */
  Serial.println(”Ooops, no LSM303 detected ... Check your wiring!”);
  while(1);
˝
else
–
  Serial.print(”Bien! Sensor Magnetómetro LSM303  detectado mediante
     SW i2c.!\n”);
˝

if(!accel.begin())
–
  /* There was a problem detecting the ADXL345 ... check your
     connections */
  Serial.println(F(”Ooops, no LSM303 detected ... Check your wiring!”
     ));
  while(1);
˝
else
–
  Serial.print(”Bien! Sensor Acelerómetro LSM303 detectado mediante
     HW i2c.!\n”);
˝



if(!mag.begin())
–
  /* There was a problem detecting the LSM303 ... check your
     connections */
  Serial.println(”Ooops, no LSM303 detected ... Check your wiring!”);
  while(1);
˝
else
–
  Serial.print(”Bien! Sensor Magnetómetro LSM303  detectado mediante
     HW i2c.!\n”);
˝
```

```
  displaySensorDetails();
˝

void loop(void)
–
  /* Get a new sensor event, for the sensor connected through SW i2c*/
  sensors˙event˙t eventSWi2c;

  /* Display the results (acceleration is measured in m/s^2) */
  acceli2c.getEvent(&eventSWi2c);
  Serial.print(F("ACCEL "));
  Serial.print("X1: "); Serial.print(eventSWi2c.acceleration.x); Serial
      .print("  ");
  Serial.print("Y1: "); Serial.print(eventSWi2c.acceleration.y); Serial
      .print("  ");
  Serial.print("Z1: "); Serial.print(eventSWi2c.acceleration.z); Serial
      .print("  ");Serial.println("m/s^2 ");

  /* Display the results (magnetic vector values are in micro-Tesla (uT
      )) */
  magi2c.getEvent(&eventSWi2c);
  Serial.print(F("MAG   "));
  Serial.print("X1: "); Serial.print(eventSWi2c.magnetic.x); Serial.
      print("  ");
  Serial.print("Y1: "); Serial.print(eventSWi2c.magnetic.y); Serial.
      print("  ");
  Serial.print("Z1: "); Serial.print(eventSWi2c.magnetic.z); Serial.
      print("  ");Serial.println("uT");

  /* Get a new sensor event, for the sensor connected through HW i2c */
  sensors˙event˙t eventHWi2c;

  /* Display the results (acceleration is measured in m/s^2) */
  accel.getEvent(&eventHWi2c);
  Serial.print(F("ACCEL "));
  Serial.print("X2: "); Serial.print(eventHWi2c.acceleration.x); Serial
      .print("  ");
  Serial.print("Y2: "); Serial.print(eventHWi2c.acceleration.y); Serial
      .print("  ");
  Serial.print("Z2: "); Serial.print(eventHWi2c.acceleration.z); Serial
      .print("  ");Serial.println("m/s^2 ");

  /* Display the results (magnetic vector values are in micro-Tesla (uT
      )) */
  mag.getEvent(&eventHWi2c);
  Serial.print(F("MAG   "));
  Serial.print("X2: "); Serial.print(eventHWi2c.magnetic.x); Serial.
     print("  ");
  Serial.print("Y2: "); Serial.print(eventHWi2c.magnetic.y); Serial.
     print("  ");
  Serial.print("Z2: "); Serial.print(eventHWi2c.magnetic.z); Serial.
     print("  ");Serial.println("uT");


  /* Display the pressure sensor results (barometric pressure is
     measure in hPa) */


  Serial.println(F(""));
  delay(1000);
"
```

Appendix C

Python Script for comunnication with Oscilloscope and Power Source

```
import visa
from visa import constants
import vxi11
import csv
import pandas as pd
import time
import math
import os
import numpy as np

def toIS(m, unit):
    if m ¡ 1E3:
        return "%.3f" % (m) + " " + unit
    elif m ¡ 1E6:
        return "%.3f" % (m/1E3) + " k" + unit
    elif m ¡ 1E9:
        return "%.3f" % (m/1E6) + " M" + unit
    elif m ¡ 1E12:
        return "%.3f" % (m/1E9) + " G" + unit

# GPIB INIT
# visa.log˙to˙screen()
SG =  vxi11.Instrument("192.168.1.119")
OSC = visa.ResourceManager('@py').get˙instrument('TCPIP0
    ::192.168.1.121::insto::INSTR')
OSC.timeout=2500000
```

```
# IDENTIFYING
print("SG found: " + SG.ask("*IDN?").strip())
print("OSC found: " + OSC.query('*IDN?').strip())

#OSC Setting-up

OSC.write(':CHANnel1:DISPlay ON')
OSC.write(':DISPlay:SIDebar MEASurements')
OSC.write(':MEASure:FREQuency')

#SG Setting-up

SG.ask("CH1:VOLT 3.3")
SG.ask("CH2:VOLT 1")

SG.ask("CH1:CURRent 2")
SG.ask("CH2:CURRent 3")

SG.ask("OUTPut CH1,ON")
SG.ask("OUTPut CH2,ON")

volt˙Sweep=np.arange(0.5,8,0.5)
current˙Sweep=[]
measured˙Osc=[]

for V in volt˙Sweep:

    SG.ask("CH2:VOLT %f" %V)

    if V==8.5 or V==9:

        time.sleep(60)

    else:

        time.sleep(10)

    current˙Sweep.append(SG.ask("MEASure:CURRent? CH2"))
    measured˙Osc.append(OSC.query(':MEASure:FREQuency? CHANnel1'))


voltage˙Data=pd.DataFrame(volt˙Sweep)
current˙Data=pd.DataFrame(current˙Sweep)
measured˙Freq=pd.DataFrame(measured˙Osc)
```

```
pd.concat([voltage˙Data, current˙Data, measured˙Freq], axis=1).to˙csv(”
    data18.csv”)

print(”Data written to CSV”)

SG.ask(”OUTPut CH1,OFF”)
SG.ask(”OUTPut CH2,OFF”)
```

Bibliography

[1] Farnell, "Arduino uno." Arduino UNO DATASHEET.

[2] MITSUMI, "M30n-2 series." M30N-2 Series DATASHEET.

[3] "Photo l298n h-bridge from supplier.." https://www.amazon.com/Qunqi-Controller-Module-Stepper-Arduino/dp/B014KMHSW6, 2019.

[4] K. Townsend, "Adafruit 10-dof imu breakout," 2018. Adafruit 10DOF details.

[5] C. Electrónica, "Sensor cny70 reflectivo de un canal." https://www.carrod.mx/products/sensor-reflectivo-de-un-canal-cny70, 2019.

[6] V. Semiconductors, "Cny70, reflective optical sensor with transistor output," 2012. CNY70 Datasheet.

[7] Atmel, "Atmega328p." ATmega328P DATASHEET.

[8] Adminlgl, "How to set up an ir remote and receiver on an arduino." http://sin.lyceeleyguescouffignal.fr/how-to-set-up-an-ir-remote-and-receiver-on-an-arduino, october 2018.

[9] P. Semiconductors, "Pcd8544," 1999. Product specification PCD8544.

[10] J. T. B.V., "Cnc milling tools," 1999. Product specification CNC milling tools.

[11] M. Gajamohan, M. Merz, I. Thommen, and R. D'Andrea, "The cubli: A cube that can jump up and balance," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3722–3727, Oct 2012.

[12] M. Gajamohan, M. Muehlebach, T. Widmer, and R. D'Andrea, "The cubli: A reaction wheel based 3d inverted pendulum," in *2013 European Control Conference (ECC)*, pp. 268–274, July 2013.

[13] D. Pekarek and T. D. Murphey, "A projected lagrange-d'alembert principle for forced nonsmooth mechanics and optimal control," in *52nd IEEE Conference on Decision and Control*, pp. 7777–7784, June 2013.

[14] M. Cederwall and P. Salomonson, "Fundamental physics," in *An introduction to analytical mechanics*, vol. 4th edition, Chalmers University of Technology, April 2009.

[15] OpenStax, "Openstax university physics," Aug 2016.

[16] O. Paulsson, *The Rayleigh dissipation function. Theory and application.* PhD thesis, 0AD.

[17] fxsolver, "Moment of inertia of a solid cuboid (axis of rotation at the height)." https://www.fxsolver.com/browse/formulas/Moment+of+inertia+of+a+solid+cuboid+%28+Axis+of+rotation+at+the+height+%29, 2019.

[18] J. P. Millan, *State-Space Canonical Forms*. PhD thesis.

[19] B. C. KUO, "Automatic control systems," in *Automatic Control Systems*, vol. 6th edition, Prentice-Hall, Inc., 1991.

[20] M. Korkmaz, . Aydoğdu, and H. Doğan, "Design and performance comparison of variable parameter nonlinear pid controller and genetic algorithm based pid controller," in *2012 International Symposium on Innovations in Intelligent Systems and Applications*, pp. 1–5, July 2012.

[21] L. Feng, "The study of the digital pid control algorithm," in *Journal of Liaoning University*, vol. 32, pp. 367–370, 2005.

[22] M. Comanescu, "Influence of the discretization method on the integration accuracy of observers with continuous feedback," in *2011 IEEE International Symposium on Industrial Electronics*, pp. 625–630, June 2011.

[23] M. Margolis, "Servo.h library."

[24] STMicroelectronics, "L298." L298 DUAL FULL-BRIDGE DRIVER DATASHEET.

[25] J. V. Stone, "A tutorial introduction to bayensian analysis," in *Baye's Rule*, vol. 4th edition, Sebtel Press, 2013.

[26] V. Semiconductors, "Application of optical reflex sensors." Application of Optical Reflex Sensors.

[27] JIMBLOM, "Graphic lcd hookup guide." https://learn.sparkfun.com/tutorials/graphic-lcd-hookup-guide/all, june 2019.

[28] QEEWiki, "Counters on the atmega168/328." https://sites.google.com/site/qeewiki/books/avr-guide/counter-atmega328, 2019.

[29] ZetosLab, "Irremote library."

[30] ArduinoModules, "Ky-022 infrared receiver module." https://arduinomodules.info/ky-022-infrared-receiver-module/, 2016.