UNIVERSIDAD DE GRANADA

FINAL DEGREE THESIS

TELECOMMUNICATIONS TECHNOLOGIES ENGINEERING

Control system in open FPGAs for autonomous robots.

Author
Juan Ordóñez Cerezo

Directors
Encarnación del Castillo Morales
Jose María Cañas Plaza

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

Granada, November 2018

Control system in open FPGAs for autonomous robots.

Author
Juan Ordóñez Cerezo

Directors
Encarnación del Castillo Morales
Jose María Cañas Plaza

Control system in open FPGAs for autonomous robots.

Juan Ordóñez Cerezo

Key words: FPGA, IceStudio, IceZumAlhambra, microcontroller, Self-Balancing, quadcopter, OpenSource.

Abstract

This paper proposes a new use of FPGAs in educational robotics field. For that, different robotic behaviors are being developed creating a module library that is useful and giving their user the possibility to work at a high abstraction level. As an example, a self-balance robot is proposed and which explanation consists about this project. Without loosing sight of the initial finality, tools used for that will be in the open source community, as can be IceStudio and IceZum Alhambra.

Me, **Juan Ordóñez Cerezo**, student of the degree in Telecommunications Technologies Engineering from **Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación de la Universidad de Granada**, with DNI 77143207-B, authorize the location of my Final Degree Thesis in the library of the center so that it can be consulted by people who need it.

Fdo: Juan Ordóñez Cerezo

Granada on November 19, 2018.

D. **Encarnación del Castillo Morales**, Teacher from Electronic Area of the Electronic Department and Computer Technology from Universidad de Granada.

D. **Jose María Cañas Plaza**, Teacher from Theory of Signal and Communications and Telematic Systems and Computing department from Universidad Rey Juan Carlos, Madrid.

Inform:

That the present project, titled ***Autonomous control system for Robots in FPGA***, has been developed under supervision by **Juan Ordóñez Cerezo**, and we authorize the defense from this project in front of the corresponding tribunal.

And for the record, the report has been issued and signed in Granada on November 19, 2018.

The directors:

Encarnación del Castillo Morales **Jose María Cañas Plaza**

Acknowledgements

In first place, and for that reason the most important, I want to thank my family, **Juan, Paqui and María**, who have suffered and enjoyed this project as much as I have, for their credibility, patience, for the moments of weakness supplied with joy and above all for believing in the capacity of someone who never gave reasons for it.
Of course, to my tutors Jose María Cañas Plaza and Encarnación del Castillo Morales who believed and made me believe since the beginning in this project and who have taken me to what I am now.
To my colleagues and friends from Munich, from Eesy-Innovation and Infineon Technologies, who turned last summer into the best summer in the world and who helped me without asking why, because if something has to go right, it would come out all right.
To my friends from Granada whom I have had to say so many times not because of spending time, but because I never get tired of being with them and because they cheered me up with laughter and jokes.
Finally, I remember part of the speech of two professors in my graduation, two professors, who not just for being teachers forget to treat us as people and who saw us cry, laugh and above all, to grow up. And, if I have to take something from here, it's not only the knowledge learned, it's the friends and those people who help you climb when you do not have the strength to keep climbing, people who believe in you even when you've fallen a few times.
For all of that, thanks.

Granada on November 19, 2018.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.0.1 Motivation and Objectives

Digital electronics[1] is part of the most modern branch of electronics and is evolving rapidly nowadays mainly due to the advantages it offers and which will be analyzed in the present project. Therefore, it is important to bring this technology to the whole world, to make it friendly and provide society with the necessary tools for its correct understanding, without this implying of higher education achievement.

Getting involved in a project not so developed and with lack of information, it may seem a bit overwhelming at first, especially when problems start to come out, but for any engineer it is quite a challenge to be able to start working on a determined field, offering society some useful tools for future implementations and improvements.

Working on a platform native from Granada, Spain, such as the Ice-Zum Alhambra board, considering the lack of development in this field, is also an honor that should not be forgotten when it comes to naming the motivations and objectives.

Digital electronics, FPGAs[2], microcontroller, hardware description language[3], educational robotics[4], etc, are only some of the most important concepts on which this work is based and are also part of one of the most important research lines from engineers from all over the world. It is an objective to take this concept to the classrooms, specially to little kids, making use of the libraries of implementation hardware blocks that would help in the successive works and widening the educational robotic concept.

The most important objectives from this work and which must be achieved are presented:

- Ability to understand and implement all kinds of robotic behaviors through hardware implementation languages.
- Current robotic systems featuring.
- Open tools usage for educational robotics such as IceStudio.

- Ability to develop Printed Circuit Boards (PCB) with Altium Designer[5].
- Ability to design and construct mechanical structures by using SolidWorks[6] and 3D printers.
- Understating about the importance of the coexistence between Microcontroller and FPGA.

1.0.2 Planning (Gantt Diagram)

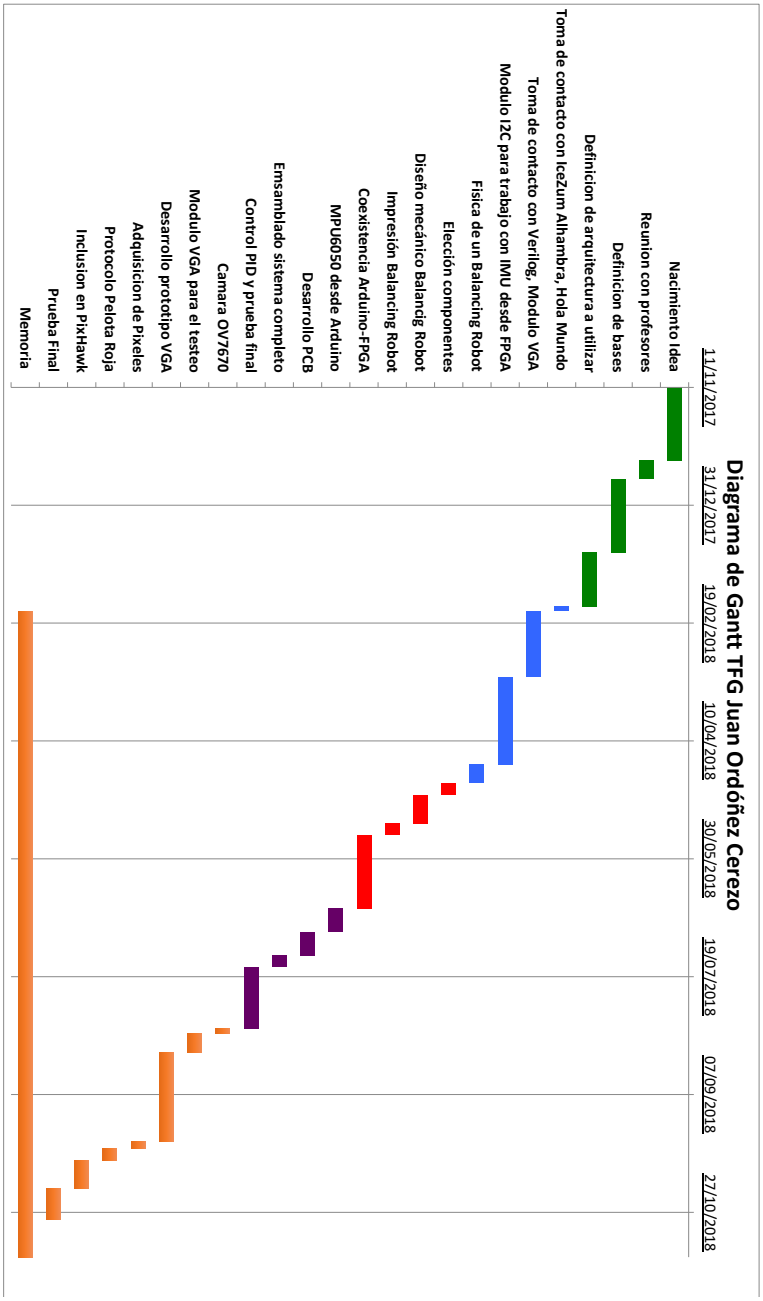

Figure 1.1

1.0.3 Work Methodology

In order to introduce the work methodology, GitHub tool will be introduced[7] (figure 1.2).

Figure 1.2: GitHub Logo.

GitHub is a collaborative software development platform for hosting projects using the Git version control system. GitHub hosts your project in a repository and provides very useful tools for teamwork.
It also provides the possibility of a Wiki for the maintenance of the versions and information about them.
In the present project GitHub has been used as a container, where everything has been partially uploaded, normally, when a stable version was obtained on any of the branches.
This way, and to be wide open, anyone has been able to follow the progress of this, doubts, problems, or even use some of the modules or material uploaded.

This project can be found at [8].

An example of the trajectory of this project is represented in the screenshot of GitHub in Figure 1.3.

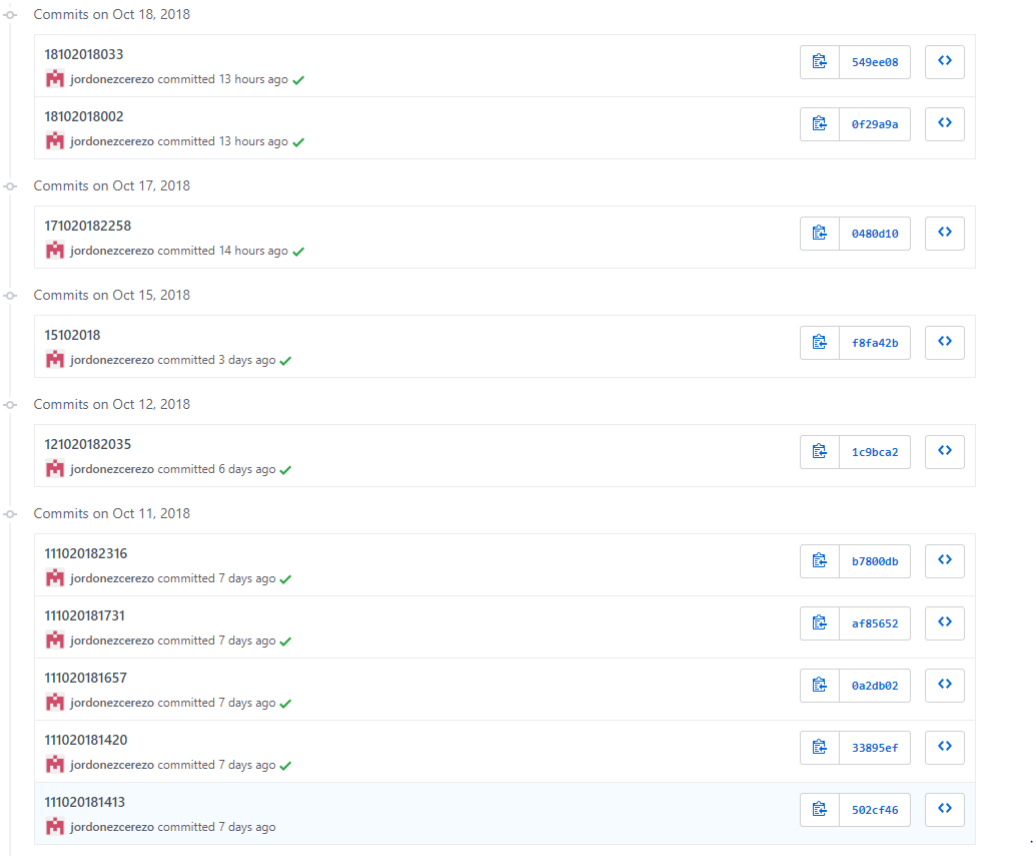

Figure 1.3: Commits GitHub.

For the good fulfillment of the objectives settled out at first instance and taking into account the different location of the components of the work, it was necessary to propose weekly meetings (usually on Fridays) where the work done during the week was put together and new objectives were settled.

For that, Appear.in[9] tool, was used, which is a free software and offers videoconferences between several users at the same time.(figure 1.4).

Figure 1.4: Appear.in.

1.0.4 Memory Structure

Memory is divides in three chapters or differenced sections.
In section 2 will be briefly explained the whole theoretical part and basic knowledges for the understanding of this project. Also, the evolution through nowadays of some proposed systems will be commented.

In section 3 3 the Balancing Robot problem would be covered and a design part and an implementation part will be differentiated. The self-balance robot will be able to stay stable on two horizontal wheels.

In section 4 the vision problem on board a quad-copter will be covered as a first aproximation.

At last but not least, in section 5 conclusions from the whole work done will be exposed, as well as a possible future work to be done, errors to be corrected, etc.

Chapter 2

State of the art

In the following section, theoretical aspects will be considered for a correct understanding of the work done. In addition, the evolution of some systems will be presented, as well as its advantages and disadvantages.

2.1 FPGA concept

An FPGA[2] (Field Programmable Gate Arrays) is a reconfigurable device that can be electrically programmed to implement a high variety of logic circuits. It consists in a uniform logic programmable structures array which are interconnected by configurable routing network. A routing network example is shown in 2.1.

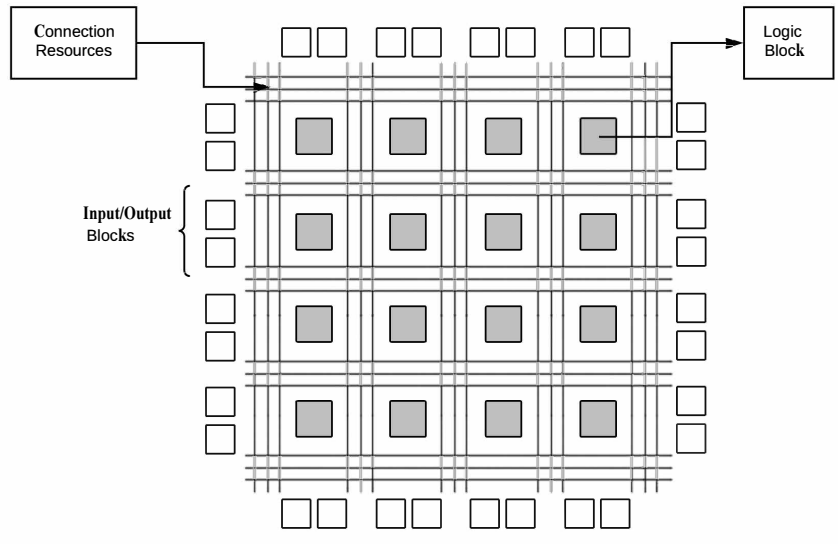

Figure 2.1: PLD vs FPGA

The logic and structures interconnection can be configured thanks to the powerful CAD tools, which allow that the final user could define this physical logic blocks interconnection through Hardware Description Language (HDL). Some of the best known are the VHDL or ABEL. Although it is not the most used nowadays, along the 2.1.4 section a brief introduction to Verilog will be done, discussing and analyzing its main features.

2.1.1 Evolution and scenario

FPGAs were invented in 1984 by Ross Freeman and Bernard Vonderschmitt, co-founders of Xilinx[10]. They were mainly designed to work as prototypes or demonstrations of digital electronic circuits. FPGAs conform the maximum evolution of PLDs (Programmable Logic Device), defined as integrated circuits which can be programmed Boolean Logic equations. Some usage examples nowadays are:

- Artificial Vision Systems
- Medical Image Systems
- Coding and Encryption
- Voice Recognition
- Aeronautics and Defense

Revolutionary success from this programmable device can be attributed to the flexibility in design implementation. This way, the capacity to instantly reprogram the FPGA with several circuits without additional cost, promotes the reuse of the device and allows a fast design verification y because of that, reduces the cost in the developing stage. Even though they do not actually represent the whole system from a final product, its advantages in the properties of the design are making them to become much more important in the electronic products developing.

In order to try to understand the importance of hardware implementation devices, it would be adequate to introduce the Moore's Law. Moore's Law expresses that approximately every 22 years, the number of transistors is duplicated in a microprocessor. As it is shown in 2.2, this law formulated by the Intel cofounder, E. Moore, in April the 19th of 1965, is not so far from reality.

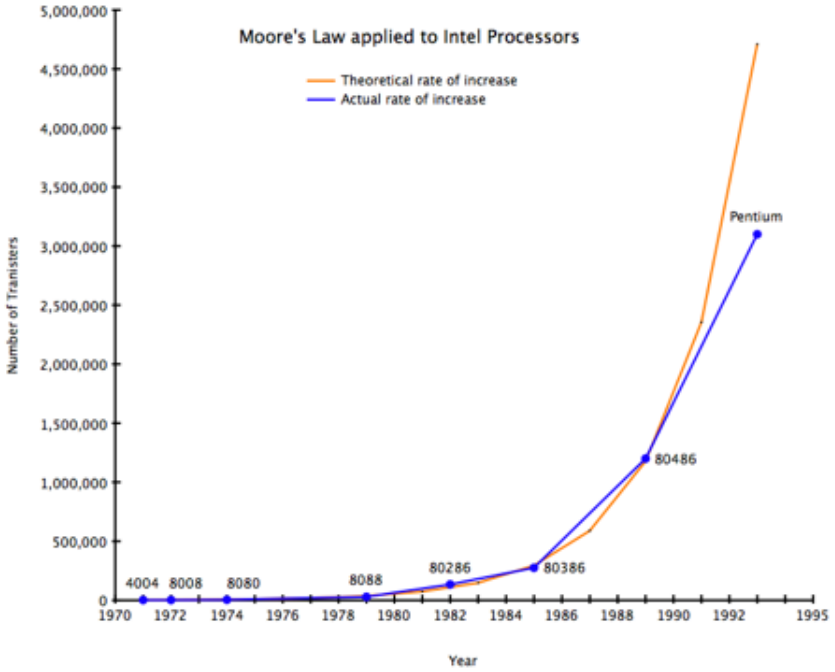

Figure 2.2: Moore's Law

However, as it has been announced form the main microprocessors enterprises like Intel and AMD, in their technological roadmap for semiconductors, the Moore's Law will get to its end in 2021. After this date, it will not be economically efficient to keep reducing the silicon transistor's size. The industry prediction is not only that its reduction rate will become slower and slower, but that will definitely stop. At that point, digital electronics will start to have an important role with microprocessors. This is forcing Multinational Enterprises dedicated to Microprocessors manufacturing like Intel or AMD, to implement those behaviors in FPGAs. Its advantages and disadvantages will be presented in 2.1.2 and 2.1.3.

2.1.2 FPGA Architecture

In a PLD, interconnections between elements is already prefixed and it is only possible to enable or disable this connection. Otherwise, FPGAs connections are no prefixed, making possible to the final user to decide the logic blocks interconnections. (figura 2.3).

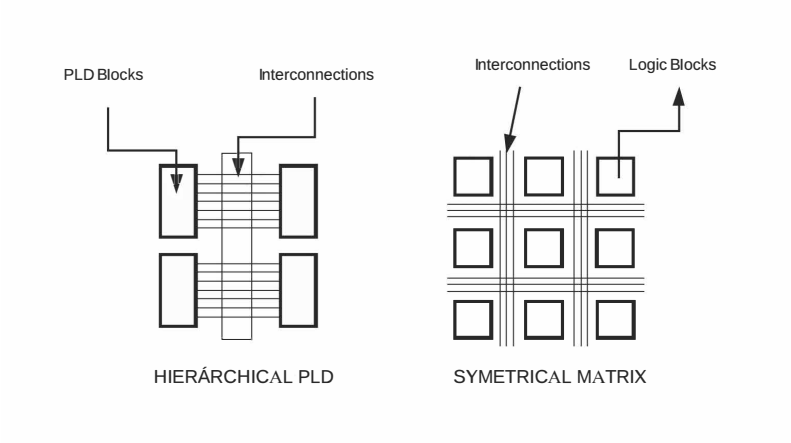

Figure 2.3: PLD vs FPGA

When choosing a FPGA for a specific project, it is important to keep in mind one of its most important property: The number of logic blocks. This number determines the device capability and is part of one the most limited features from nowadays FPGAs/ Logic blocks are independent between them and are able to interconnect in order to build a more complex module.

Now fine-grained and coarse-grained meanings will be introduced, which will lead to the correct understanding to the FPGA architecture.

This complex module performs the basic operation which together represent the function that will operate in a FPGA. It is said that FPGAs are Advanced Architecture devices because of the density of its components and because of the different interconnection paths between modules. According to the module types that conform it, we have two structural configurations:

- Fine-grained
- Coarse-grained

Logic modules in a coarse-grained architecture are big modules generally consistent which have one or more query tables and two or more Flip-Flops. The query table is also known as the Look Up Table (LUT). It performs like a memory where the truth table, which represents the circuit logic function is located, this way, any desired function could be implemented in a LUT. Depending of the LUT size, more or less variable functions can be implemented.

On the other hand, a fine-grained architecture is structured by a huge quantity of small logic modules which perform relatively simple tasks. Each module has a two-input circuit that performs a determined logic function, or in some other cases by a multiplexor. It may also contain any Flip-Flop.

The most used FPGAs nowadays have the coarse-grained technology, which allows to increase the abstraction level with respect to fine-grained FPGAs.

The first case allows less detailed implementations since from a very low level there are much more complex modules and the fact of using LUT suggests that larger designs can be created. Along this project the first case will be considered, as an example of that is the FPGAs used, IceZum Alhambra II.

The designer proposes the logic function to be performed throughout the description hardware methods and define the parameter to its design. This is done through programmable code, that can be a description hardware language, which is introduced in section 2.1.3.

2.1.3 Hardware Description Languages

A Hardware Description Language[3] (HDL) is a language used to model the hardware block functionality in a textual way. As we can find differences between the different type of programming languages according to their coding syntax and their simulation methods and synthesis, we can find several differences between HDLs. There are two different hardware description languages types:

- Low level: They do not allow the hierarchy of modules and they are able to perform only simple descriptions.
- High level: They are the most used currently, they allow to design more complex systems. Examples of high modeling are Verilog (Verify Logic) and VHDL (VHSIC Hardware Description Language).

This work would be based on Verilog for allowing a higher level of implementation than VHDL, as would be discussed in section 2.1.4.

Both languages share some common characteristics, such as support for any level of modeling and abstraction, and that each design element has well defined interphase to allow fast connection with other logical elements.

2.1.4 Verilog

Due to its use throughout this project, some important characteristics, whose knowledge would be basic to understand the code, will be analyzed.

Type of data

There are two types of data in Verilog which must be understood in order to reach the desired functionality:

- reg: Represent variables with information storage capacity.

- wire: They represent connections between components, they do not have storage capacity.

Modules Implementation

In most cases and in order not to lose the abstraction level, a project in Verilog[11] is usually composed of a set of modules which form a complete functionality, and each of them, a specific one.

The main features of the digital implementation by modules are:

- Each module has a series of inputs and outputs whose main function is to interconnect other modules, although it may not have inputs and outputs.
- Each module can be described architecturally or behavioral.

This implementation by Verilog modules allows the configuration of the abstraction level desired by the end user. The development of specific functionality modules can have their advantages and disadvantages depending on their final use. If they want to be reused for other workflows (example: 8-bits adder), it is good to have very defined modules according to their functionality (example: 2-bit adder). However, polarization does not have to be essential, although advised.

Process Parallelism

One of the most important features that differentiates Verilog from the rest of the procedural languages [1] is the possibility of executing several processes in parallel, a fundamental aspect in the language which provides a great part of the advantages of using hardware implementation.

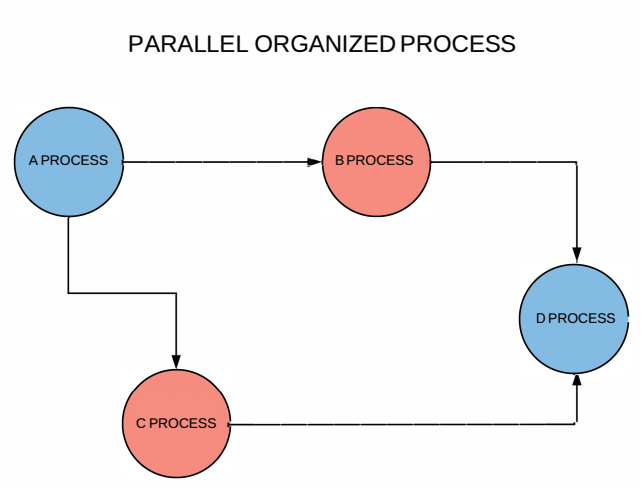

Figure 2.4: Process Parallelism

[1] Procedural Languages: The application execution starts in the code's first line and follows a predefined root through the application, calling procedures as it is necessary.

The whole implementation of Verilog is declared inside a process that can be of two types:

- Initial: This type of process is executed only once starting its execution at the beginning, therefore, there are no delays. This process is not synthesizable, which means it cannot be used in an RTL description.
- Always: This type of process is continuously executed as a loop, as its own name indicates, is continually running. This is a synthesizable process and is controlled by timing or events. If this block is executed by more than one event, its set is called sensitive list.

Control Structures

Just like other procedure languages, Verilog has a series of control structures:

- if - else
- Case. This is one of the most used along this project. It allows the state machines generation.
- For
- While
- Forever
- Wait

Continuous Assignment

Through continuous assignment a logic combinational can be modeled, that means, there's no need a sensibility list to complete the task. It can only be declared outside of any process.

Procedural Assigment

Variables have a value assigned inside an always or initial process, the type of variable to which the value is assigned can be of any type.

2.2 Open FPGAs evolution

Many hardware implementation languages as well as its FPGA architecture used, are linked to important companies such as Xilinx, Intel (formerly Altera), etc, and being able to work with them requires a large budget.
This leads us to the fact that not many companies or individuals can benefit from the advantages of using FPGAs, and at the same time, that technological progress is at a slower rate. One of the keys of success of companies like Arduino is not more than the community of people that stands behind creating new libraries, components, etc. This is all thanks to the low price of its products and the possibility of finding all the hardware and software on the web.

To understand the formation of free FPGAs it is important to know what the bytestream is.
A bytestream is a sequence of bytes that is used in telecommunications and IT. The term bytestream is used to describe the configuration with which a specific design will be implemented in an FPGA. This detailed bitstream format for a particular FPGA is typically owned by the FPGA provider.

This is why Clifford Wolf decided to interpret the bytestream of the Lattice[12] iCE40 model and developed the IceStorm[13] tool.
IceStorm was developed as a translate software from Verilog (Description Language in FPGAs, 2.1.4) and bytestream. This translation was possible thanks to the inverse engineering, this means that the usual usage is not give, but the inverse.
There's no longer dependency on any manufacturer and all knowledge is also available. From these tools you can create any interface or any application that has not been foreseen by the manufacturer.

Only Lattice iCE40 FPGAs (HX1K-TQ144 and HX8K-CT256 models) are at the moment the ones that can be worked with (figura 2.5), but since it is a open project [2],a lot of people are increasing their chances.

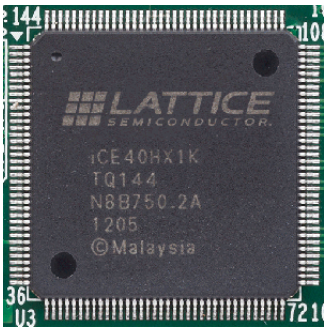

Figure 2.5: Lattice iCE40HX1K

[2]Open project: This refers to the user freedom to execute, copy, distribute, study, change and improve that hardware or software.

Some FPGAs examples that are now available for users are shown in figures 2.6, 2.7, 2.8

Figure 2.6: Tiny FPGA BX

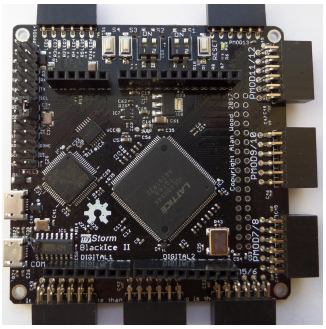

Figure 2.7: BlackIce II

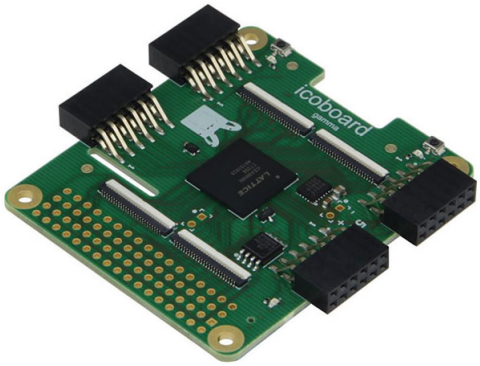

Figure 2.8: ico Board

2.2.1 IceZum Alhambra

For this project, IceZum Alhambra[14] has been used, which has been completely designed and assembled in Granada (Spain).
This is a free FPGA compatible with IceStudio (to be analyzed in section 2.2.2).
Some of its most important features are:

- Development FPGA board iCE40HX1K-TQ144 from Lattice company.
- Open hardware.
- Compatible with IceStorm toolchain.
- Compatible with Arduino Uno shields.
- 12MHz oscillator.
- ON/OFF switch to enable or disable digital pins.
- 20 Input/Output 5V pins.
- 8 Input/Output 3.3V pins.
- Micro-B USB to program FPGA from PC.
- Reset button.
- 8 general purpose LEDs.
- TX/RX LEDs.
- 4 analogic inputs available through i2c.

It is known that there are better features board, but the fact that it is Open Hardware and that can be implemented with IceStudio, has led to the final choice of this board for the development of this project.

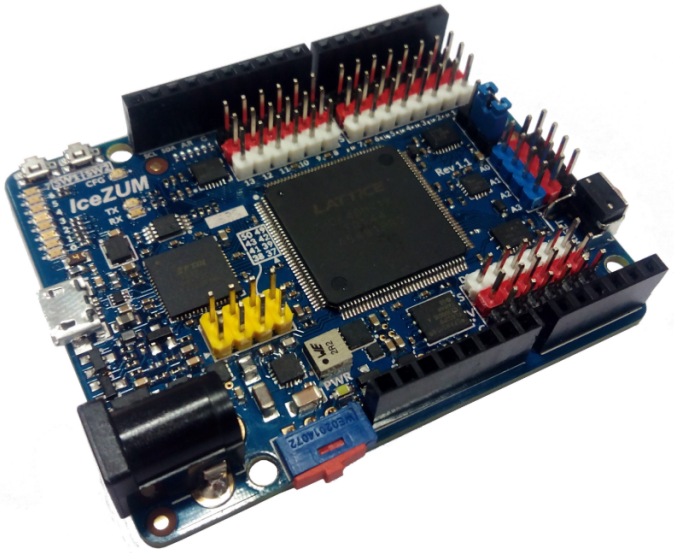

Figure 2.9: IceZum Alhambra Board

One point to keep in mind for the hardware development with this FPGA is its 1K memory, which has been a significant limitation in the development. For the present project, the new version of the IceZum Alhambra, IceZum Alhambra II, was also used, which was not in the market at the beginning of the project and which takes some improvements, such as the expansion of 8K in its memory, the improvement of the data bus i2c, the possibility of powering through LIPO. battery, etc. IceZum is presented in figure 2.10.

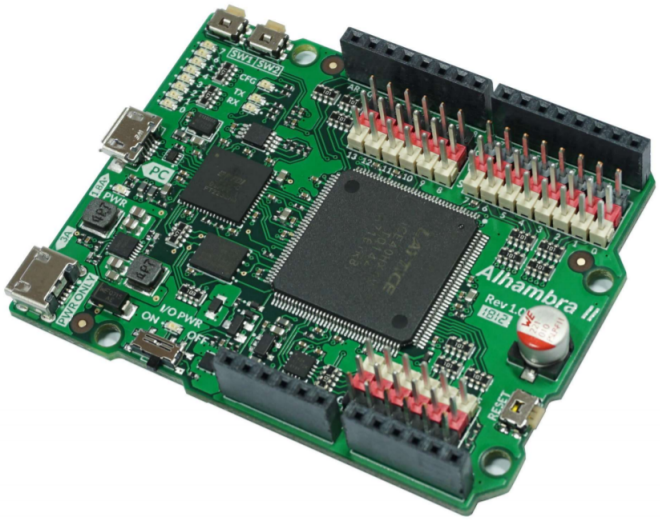

Figure 2.10: IceZum Alhambra II Board

2.2.2 IceStudio

HDL languages usually have a difficult learning curve, this is largely due to the low level of abstraction needed to design a particular system. It is necessary to know the hardware features of our system in order to work with this type of logic.

As previously developed, some manufacturers provide commercial tools to program their own FPGAs. Although in the present, they are complex environments, they have a great number of tools and functionalities. Unfortunately, most of them are not free and are linked to the architecture of a unique manufacturer.
With the evolution of FPGAs, languages that allow a higher level of abstraction have started to appear. In addition, tools focused on the implementation of graphic design have also appeared. An example of this type of implementation is LabVIEW FPGA or IceStudio[15].
IceStudio is an Open Source project developed by Jesús Arroyo Torrens, on which the present document will be based.

IceStudio is a graphic IDE for free FPGAs and is built on the IceStorm project. The objective of the IceStorm project is the reverse engineering and the bitstream format of the FPGA Lattice iCE40 (although a few more were emerging) documentation. It provides simple tools to analyze and create bitstream files, that is, the lowest level of implementation for an FPGA.

To get the reader closer to the knowledge and operation of IceStudio, a series of representative screenshots will be incorporated throughout the document so that the vision of what is being done is not lost. For example, the main window of IceStudio and on which everything else will be developed, is the same as shown in figure 2.11.

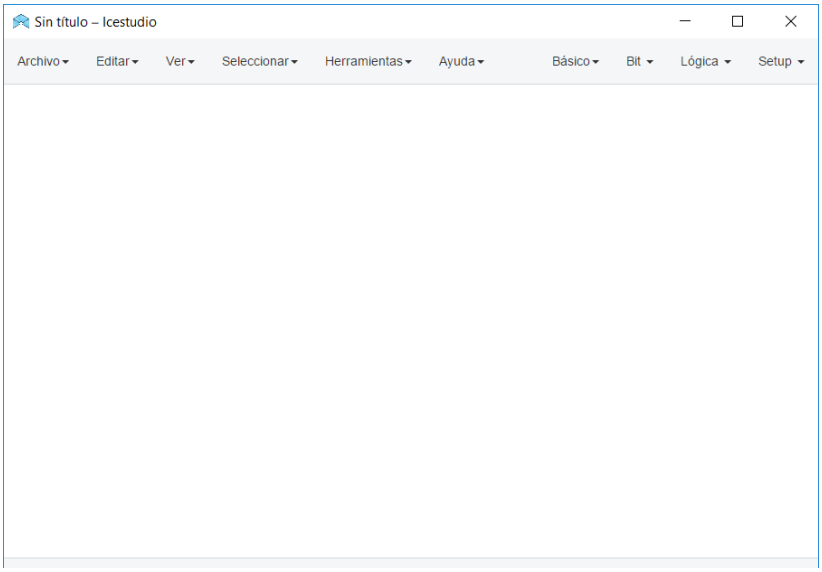

Figure 2.11: Main Window IceStudio.

The fact that IceStudio is a graphic editor can make believe that the abstraction level could be higher that desired, but the truth is that this a configurable level. The final user who decides which abstraction level will be working with, making necessary for this a large modules library as we will se ahead. In order to explain the IceStudio power, a practical case will be shown.

The module represented in figure 2.12 is a normal writing of i2c, in which the slave direction is parameterized and also the direction that wants to be read on (will be explained in detail ahead).

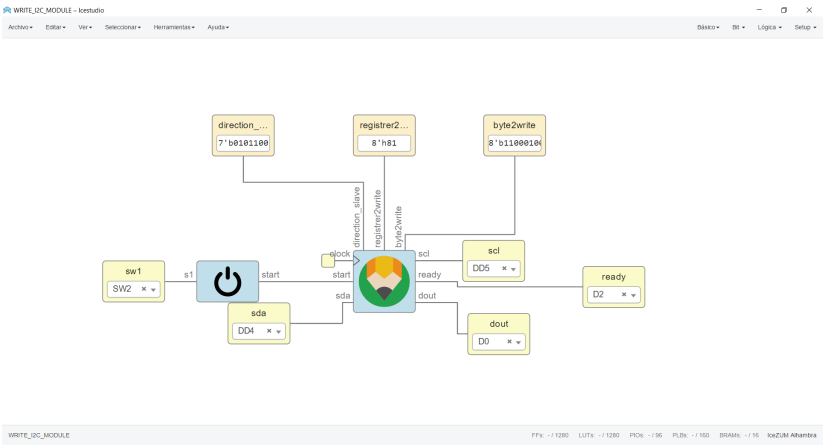

Figure 2.12: High-Level I2C writing

Thus, if someone who is not familiarized with this type of code and whose objective is not to understand it but wants to make use of it, it should not lower a lot of level. However, there is a possibility that it is necessary to change some values such as clock frequency, i2c operation mode, etc. To do this we can lower the level and enter the module being worked with, in this case, by double clicking, as shown in figure 2.13.

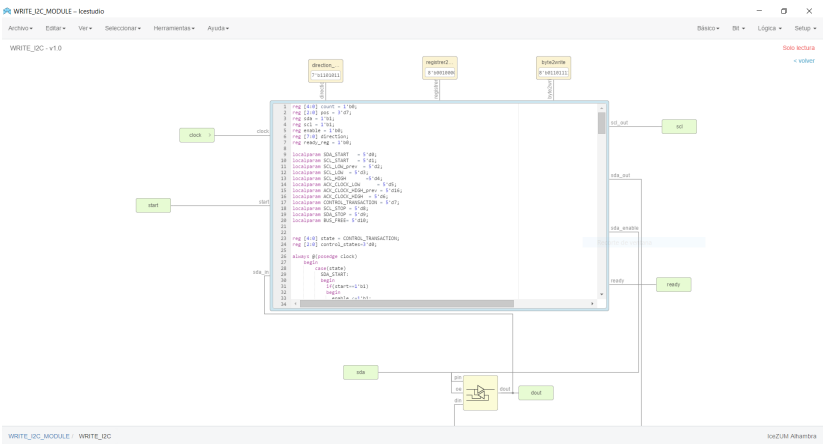

Figure 2.13: Low-Level I2C writing

The it could be said that a level of abstraction has been lowered, being now able to enter in more specific hardware details if necessary.

In the previous practical case, one of the advantages of IceStudio has been seen. Modularity allows to configure the abstraction level. For this, we need a module's library, some of which will be developed throughout this work, some other modules are being developed, and can be found at [16].

2.3 Microcontroller-FPGA coexistence

2.3.1 Microcontroller-FPGA differences

At first it may seem that a processor and FPGA are similar devices because both can perform certain pre-configured tasks. The truth is that digging deeper we can find more differences than similarities. Both are able to implement a transfer function, but the way they do it is different for each of them.

This, we could see FPGA and microcontrollers in a black box in which we have some inputs and several outputs as shown in figure 2.14.

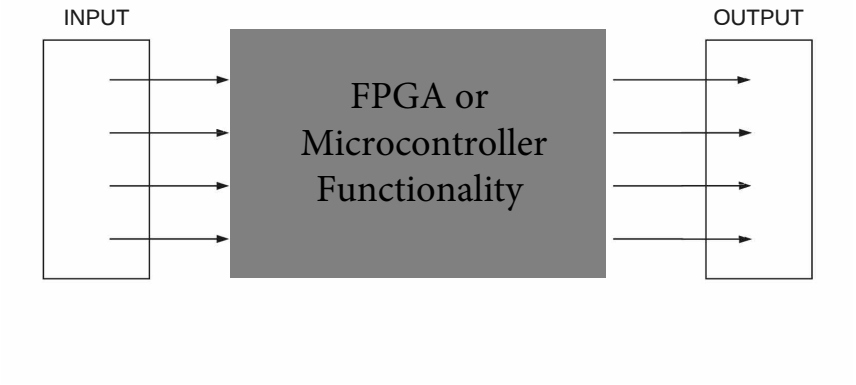

Figure 2.14: Funcionalidad FPGA y Micro-controlador.

To summarize how they implement this transfer function differently, the way to work with a processor will be briefly explained.
A processor contains a series of instructions that perform operations on a set of bits (add, increase, read and write in memory). Depending on the processor type and its architecture we have more or less associated instructions, this being one of the most important aspects that determine its performance.

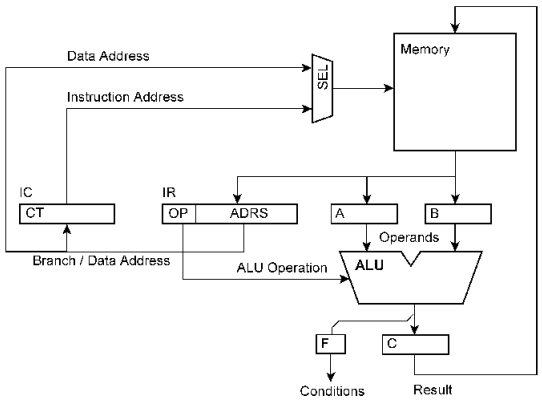

Figure 2.15: Basic architecture in a processor

It has a series of registers, a memory to store the information and a stack of instructions, which contains the program to be executed in machine code, in addition to a clock.

Its operation mode at a high level; in each clock cycle the processor reads from its instruction stack all necessary values, calls the appropriate instruction and executes a determining calculation.
As discussed in 2.1.2, when implementing a logic design in an FPGA, a matrix of physical connections is being modified. By modifying that connection matrix, different functionality blocks can be implemented, which means, it could be represented as several transfer functions in a same hardware system.
Figure 2.16 shows a real example of how the physical connections of logical gates are implemented in an FPGA, and how it allows to have independent modules from each other. Also, notice the problem of the memory in an FPGA, being this the total number of physical logical doors that can be used.

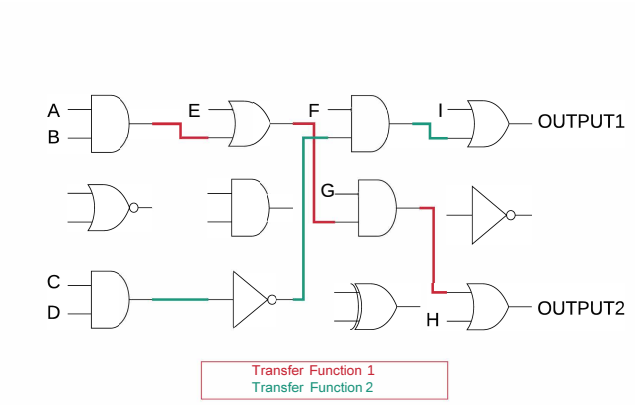

Figure 2.16: Logic gates in a hardware implementation.

2.3.2 Importance

The following practical case is proposed:

It is required to monitor with accuracy 4 different sensors coming from the outside, in an exact way, all four at the same time and at a specific speed by an external clock, being necessary in addition an actuation of the system (valve opening, closes gates, etc).

The workflow block diagram in a processor that implements the above, might look like figure 2.17

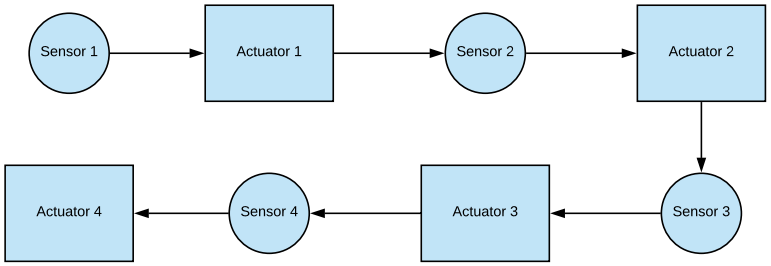

Figure 2.17: Flow diagram in a processor.

Thus, the end user of this system should be able to check in a cyclic way, each sensor and its following performance, thus failing to meet the specifications (the four sensors will not be monitored at the same time). If the work was done with interruptions, the different external interruptions would be configured or cyclic executives would be used in order to approach those final requirements. However, any of these solutions, they are still an approximation.
In contrast, with the use of an FPGA, the block diagram of the workflow would look like figure 2.18.

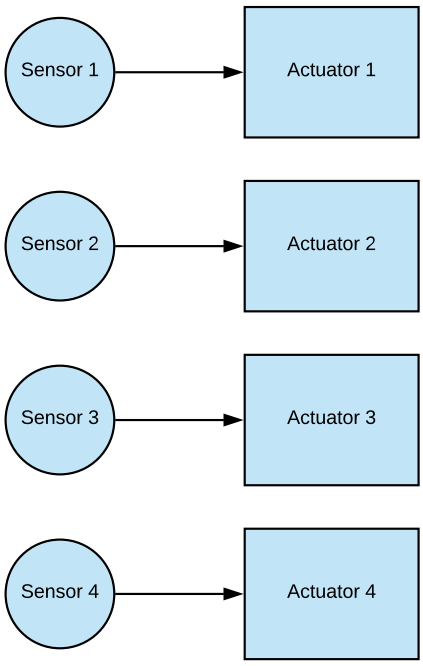

Figure 2.18: Flow diagram in a FPGA.

Different transfer functions will be implemented for each one of the blocks to be developed, which can be executed in parallel. However, it is not always necessary to have parallel execution, and not only may it not be necessary, but it could be harmful. When a system must be sequential, why to use a parallel nature implementation?

It is very common to have systems where it is convenient to be able to implement both types of operation, for that reason an FPGA/Microcontroller coexistence could be enough to adapt to the requirements.

In figure 2.19 a real example of a bipedal system can be seen, which would be more detailed explained in the following chapters.

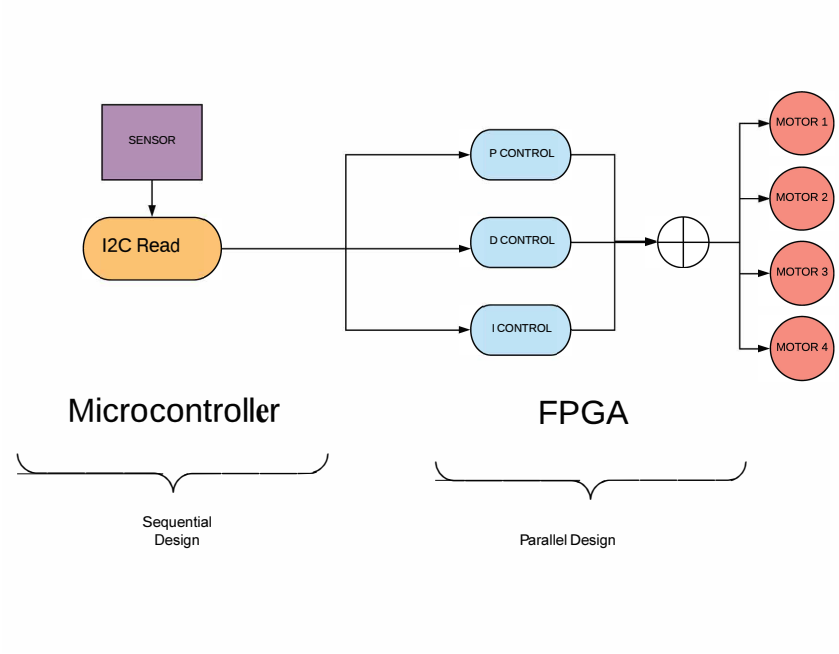

Figure 2.19: Bipedal System coexistence between microcontroller and FPGA.

2.4 Inertial Measurement Unit

The central component of the proposed system is defined as IMU or inertial measurement unit. It is composed of a series of sensors which will be used to know exactly many on-board location system aspects such as speed, orientation, gravitational forces, etc.
This information can be used for control or simple knowledge of the system at a specific time.

In the case of the present project, it will be used to obtain the navigation angles or Tait-Brain angles, in which the orientation is presented with three orthogonal rotations around the X, Y and Z axis. An example of this type of representation can be found in Figure 2.20.

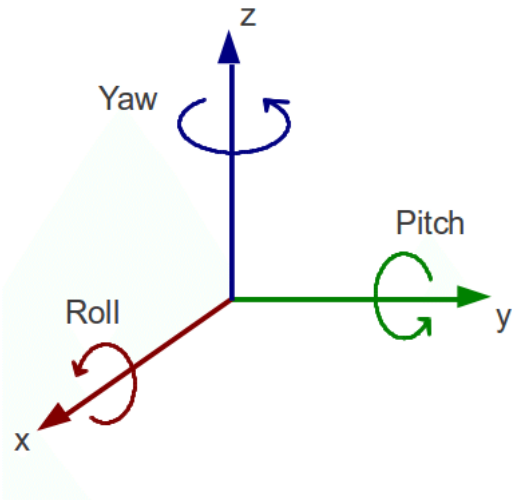

Figure 2.20: Tait-Brain angles.

An IMU is defined by its number of degrees of freedom (DOF) which would depend on the number of sensors on board (accelerometer, gyroscope) and the number of axis on which are applied. Thus, an IMU with six DOF (for example a 3-axis accelerometer and a 3-axis gyroscope) would be said to be 6DOF.

Figures 2.21, 2.22, 2.22 present some IMUs examples.

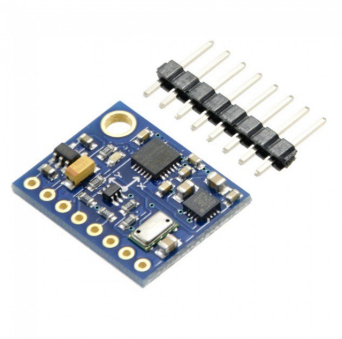

Figure 2.21: IMU.

Figure 2.22: IMU.

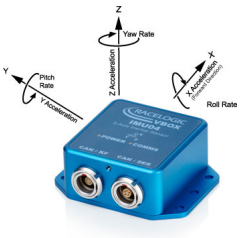

Figure 2.23: IMU.

Another important feature and one on which will largely depend the price of the product, is the range of the on-board sensors and the availability in the digital analogous converters system, which are in charge to convert the value of the sensor to a binary value that can be used for further analysis.
Following there are some of the most important aspects that can be found in an Inertial Measurement Unit and those that would be very useful throughout the present project.

Accelerometer

An accelerometer, as its name suggests, is a device that allows measuring the acceleration to which a body is subjected. It is a fundamental component in the inertial measurement units because it can detect, for example, free fall conditions, although the main use is to determine the sensor's orientation.

Normally accelerometer have 3 axis, which means, they are capable to independently measure the acceleration in X, Y and Z axis, which allows to know the magnitude and direction of the acceleration vector on each one of the axis.

The important case in this project is to be able to determine the sensor orientation. For that, trigonometry must be applied. Supposing a 2D system according to Figure 2.24.

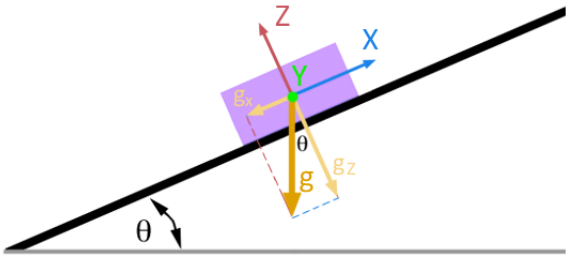

Figure 2.24: Trigonometry in accelerometer system 2D.

$$\theta = atan\frac{A_x}{A_z} \tag{2.1}$$

If it is applied in a 3D representation as shown on Figure 2.25

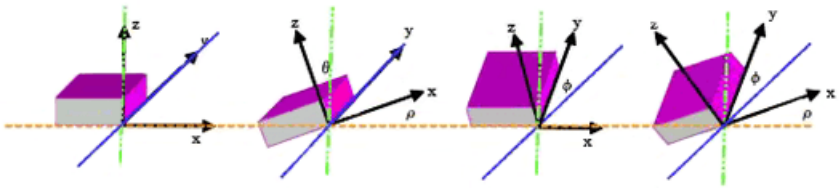

Figure 2.25: Trigonometry in accelerometer system 3D.

$$\theta_x = atan\frac{A_x}{\sqrt{A_{y^2} + A_{z^2}}} \tag{2.2}$$

$$\theta_y = atan\frac{A_y}{\sqrt{A_{x^2} + A_{z^2}}} \tag{2.3}$$

$$\theta_z = atan\frac{A_z}{\sqrt{A_{x^2} + A_{y^2}}} \tag{2.4}$$

Gyroscope

A Gyroscope is a device that allows to measure the rotation angle of a certain device.

In a gyroscope, relative angles are always measured to an arbitrary reference (unlike accelerometers). Gyroscope used in this project is called a Coriolis Vibratory Gyroscope.

As in the accelerometer in the previous section, gyroscopes usually employ three axes, that is, they are able to independently register the rotation in the X, Y and Z axis, so the module and direction of the rotation vector are obtained.

It is important to keep in mind that this kind of Gyroscopes based in Coriolis effect are not able to detect the rotated angle, but the angular speed, an important aspect to its depuration.

Reminding the angular speed concept (2.5):

$$\omega = \frac{\delta\omega}{\delta t} \tag{2.5}$$

In order to obtain the angle, it is necessary to perform the integration with respect to time (ecquation 2.6):

$$\theta_{gyro} = \omega_{gyro} * \Delta t \tag{2.6}$$

Accelerometers are devices that are often very sensitive to vibrations, so for its correct processing must take into account that it will present a lot of high frequency noise. A filtering at a certain frequency will solve part of this problem.

Having to make an integration with respect to time brings with it some problems which would be seen in the next section.

Drift Problem

A huge part of the Inertial Measurement Units on the market are at least composed with accelerometers and gyroscopes, the reason for this combination is that one complements the limitations of the other and vice versa.

The drift in an electronic sensor is a variation over time of the output of the meter (with respect to the real measurement) although the variable may be constant, is in a way cause by extent changes in the temperature or by the accumulation of errors.

In both accelerometer and gyroscope, we find drift associated errors:

- Accelerometers do not have medium or long-term drift; however, they are influenced by the movements of the sensor and the noise, reason why they are not reliable to medium or short term.

- As for gyroscopes, they work very well in sudden and short movements but when making an integration with respect to time the drift problem appears in the medium or long term.

After analyzing both accelerometers and gyroscopes problems, it seems reasonable to combine both measurements to obtain more precise orientations.

Possible solutions

To solve some of the above problems, it can be very useful to apply some of the following proposed solutions:

- Combine and filter the signals by using a complementary filter. It is the most used nowadays due to its not very high complexity. Its simplest expression is represented in equation 2.7.

$$\theta = A * (\theta_{prev} + \theta_{gyro}) + B * \theta_{accel} \tag{2.7}$$

- Combine and filter the signals using a Kalman filter, which evaluates the future value of the measurement. However, it uses complex calculations.
- Some inertial measurement units internally incorporate DMPs processors (Digital Motion Processor) which execute complex algorithms avoiding having to perform filters and freeing the processing system.

One explained the basic features of commercial IMUs, in section 3.3.2 will be developed the IMU chosen for this system and its own features.

2.5 Educational Robotics – Motivations

Robotics[4] can be considered as one of the technological areas with more boom nowadays and based on the study of robots, which are systems composed of mechanisms that allow to make movements and perform specific tasks, programmable and intelligent.

Depending on the application, therefore, robotics can be extended and generate benefits not only in the industry but also in classrooms, enabling the appearance of new learning systems.

In addition, in a world whose future is aimed at the use of robots for any activity, the approach from classrooms with these systems, enables the student's technological development at an early age, making their integration easier into an adult age.

Some robotics educational benefits are:

- Drives initiative and creativity
- Larger sociability
- Encourages algorithmic and mathematical thinking

- Teamwork
- Problem resolution
- Active learning.
- Self-Esteem increase.

However, so that the integration in the classroom of educational robotics could be easier, the systems must fulfill some characteristics:

- High technological integration level not recommended
- Robots must be sociable and fun
- Programming environments should not be complex, and even its functionality is kind of limited, it has to draw the attention of the student and make them feel comfortable.
- It is important that the robot has a series of sensors and actuators, inputs and outputs so that results must be visuals.

After analyzing the advantages of digital electronics, it is convenient to be able to bring these two knowledge fields closer together; educational digital-robotics electronics.

If digital electronics and the world of robots are called to be part of our lives in the near future, the need of an approach to these two concepts at an early age is basic for a correct technological improvement.

IceStudio was born with this idea, making digital electronics to be friendly so that the little aged students can use it, and so, meets the requirements before explained.

2.6 Sensors, actuators and control system

Before starting the development of the project, it is important to be clear about the concepts of sensors, actuators and elements of the control system, which are part of any mobile robotic platform.

Any control installation, being robotic or imnotic, is composed by three fundamental components.

- Sensors
- Actuators
- Control systems

Sensors are devices that collect information from the world around us and transform it into electrical signals that can be input to a control system.

Thus, the control system receives information from the environment on which we want to perform some kind of action through the sensors, which is the transfer

function of the system. From some known type inputs, outputs are generated, usually, dependent on the inputs.

These outputs are called actuators, which are devices that, following the parameters given by the control system, perform actions that affect the environment.

Example 2.1 *A sensor indicates to the control system the luminous intensity of a room. The control system recognized that the luminous level is not adequate for reading and activates an actuator, in this case, a light to balance that level.*

When choosing a specific sensor, it is important to know its operation module, in order to configure or maintain incorporated systems. There are different type of sensors according to:

- Ouput:
  - Analogic
  - Binary
  - Digitals
- Internal structure:
  - Passive
  - Active
- Kind of parameter able to detect.

Another possible classification is that of the application scope, that is, where and what for these sensors are used.
Between the most important technical features from a sensor and making an introduction to the possible vocabulary that will be used, there are:

- Measurement range: Domain to the measured magnitude to which the sensor can be applied.
- Precision: Expected maximum measurement error.
- Offset or zero deviation: Value of the output variable when the input variable is zero.
- Sensor sensitivity: Assuming it is input to output and the variation of the input magnitude.
- Resolution: Minimum variation of the input magnitude that can be detected at the output.
- Drifts: Are other magnitudes, apart from the measure as input magnitude, that influence the output variable.

In general, the output signal from these sensors is not suitable for direct reading and sometimes not for processing, which is why conditioning circuits are used Figures 2.26, 2.27 and 2.28 show some sensors examples.

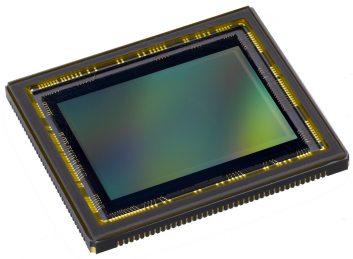

Figure 2.26: Sensor CMOS to image adquisition.

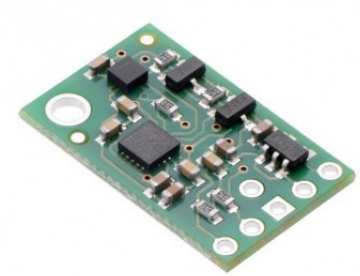

Figure 2.27: IMU.

Figure 2.28: Potenciometter.

Actuators are devices that allow the control system to "act" on the "real world" to perform desired actions. One of the best-known actuators are motors, which will be very used throughout this application.

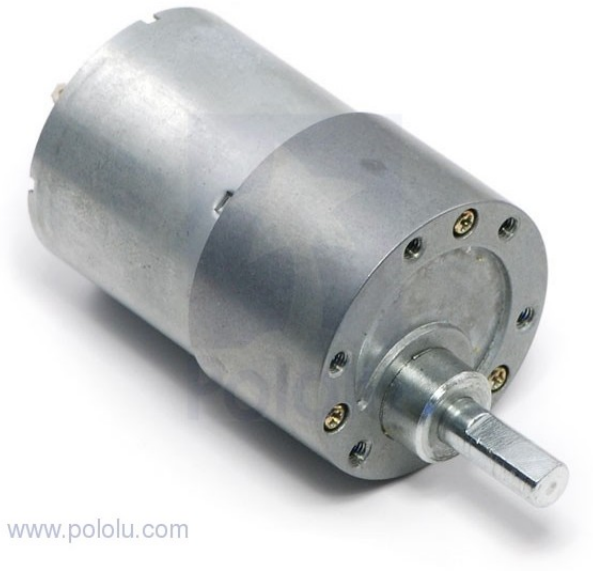

Figure 2.29: DC Motor.

There are many different types of control systems in relation to the application to be developed, normally a microcontroller is chosen as a control system and the transfer function to be performed is programmed.
In the case of this project, and in order to acquire the advantages of an FPGA and on the other hand the advantages of a microcontroller (seccion 2.3), both Arduino will be used for sequential tasks or the IceZum Alhambra for the tasks that can be parallelized.

2.7 Classic PID controller

Proportional, Integral and Derivative[17] (Proportional-Integral-Derivative, PID) is the most common and the most used control in the industry and has been universally accepted in the control industry. The popularity of this control is due to its robustness and ease of use, which allows engineers to work in a simple way.

As its name indicates, the algorithm consists in three basic operators: Proportional, Integral and Derivative which are the clue to obtain an optimal response.

The basic idea of a PID controller is to reed a sensor and calculate a proportional response, integral and derivative with the objective of calculating the best output from an actuator.

To know very well how a PID controller actuates and to know the meaning of each one of its components, it is necessary to know what a closed loop system is.

Closed loop system

A closed loop control system is s systems in which the action of the control is in function of the output signal. Closed loop systems use feedback from a final result to adjust the control action accordingly. The set point is the value that the system wants to reach.

Now it is digged a little deeper about is what's the meaning of each of the components by which the PID control is formed.

Proportional Response

The proportional component depends only on the difference between the current value and the setpoint. This difference is known as the "Error". Thus, the proportional gain K_p determines the relationship between the output response and the error signal. If the error has a magnitude of 10, a proportional gain of 5 will produce a proportional response of 50. The gain indicates the correction speed of that error, but if it is too fast, the system will oscillate.

Integral Response

The integral component adds the error-in-time component, a small increase in the error until the integral component increases slightly with time. The integral response will increase continuously over time unless the error is zero, so the effect is to bring the steady state error to zero.

Derivate Response

The derivative response causes the output to decrease if the process variable increases rapidly. The derived response is proportional to the rate of change of the process variable. Increasing the derivative constant K_d will make the control system stronger to changes in the error and will increase the speed of the overall response of the control system.

A block diagram of the complete system of a PID controller is shown in Figure 2.30.

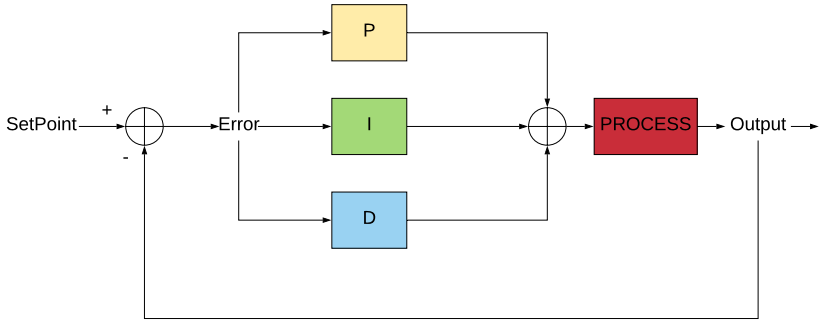

Figure 2.30: Block diagram PID controller.

Example 2.2 *A PID is used to control the temperature of a room. The set point is the temperature you want to reach, the plant is the room itself, which can be modeled as a transfer function. A temperature sensor closes the loop, indicating the temperature at each moment, while a valve opens and closes the air duct depending on the values of the PID controller, which will be calculated in relation to the setpoint value and to the value of the current temperature in the room.*

Chapter 3

Self-Balancing Robot

In this chapter, the objective is to address the inverted pendulum problem through the usage of an FPGA in coexistence with a microcontroller. For this, in the respective chapters, physics of a self-balancing robot, the calculation of its structure, the sensors and actuators used, the control system and the design and manufacture of a PCB that solves in a more adequate way some problems of those previously raised, will be treated.A communication between FPGA/Microcontroller will be used and a more global version of the proposed system will be given, with a general block diagram.

It starts with a problem description (section 3.1) to continue with a brief high level solution (section 3.2). To finish, an explanation of each blocks will be described (section 3.3).

3.1 Problem Description

To understand the work to be done, the problem of inverted pendulum will be briefly enunciated, whose solution has given rise to many very famous tools nowadays, one of them, called *SegWay* (Figure 3.1).

Figure 3.1: Commercial Segway.

Definition 3.1 *Pendulum [18]: It is a physical system that can oscillate under the gravitational action or other physical characteristic (elasticity, for example) and that is configured by a mass suspended from a point or a horizontal axis by*

a wire, a rod, or other device that is used to measure time.

As can be imagined, an inverted pendulum has the aspect shown in Figure 3.2.

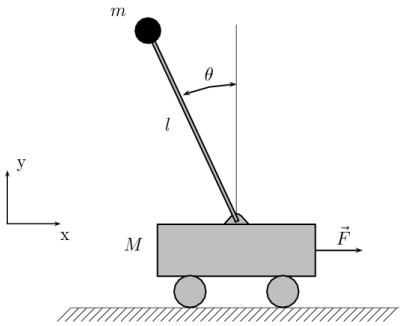

Figure 3.2: Inverted pendulum representation.

Consists of a pendulum where the mass center is located above the point or balancing axis. As expected, this layout gives the system static instability. We recall that a system is stable when its gravity center is closer to the support horizontal plane.

The base of this project will therefore be to correct this instability and is part of one of the most famous problems in terms of control theory and systems dynamics.

3.2 System Design

By i2c communication with an IMU sensor, the microcontroller obtains the current angle of the system. Obtained the angle by the microcontroller a communication of serial type sends it to the FPGA in a binary format of 1 byte for the integral part and 1 byte for the decimal part. A shield with a driver of DC motors connected to the FPGA brings the possibility of varying the speed and direction of two DC motors that allow the stabilization of the system. The speed of the motors for a correct correction of the angle is calculated by a basic PD controller implemented in the FPGA.

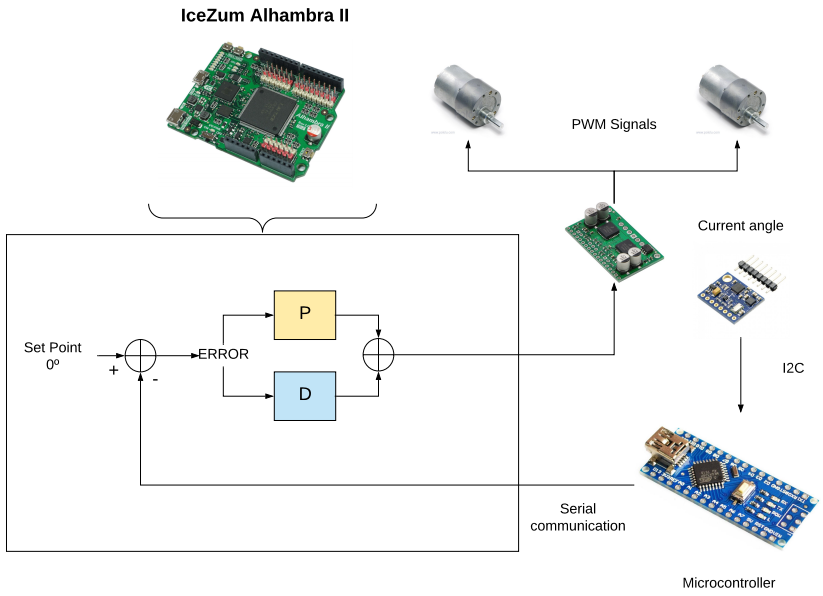

Figure 3.3: Final block diagram.

The following chapters will go deeper into each of the previous blocks that form the final solution (Figure 3.3).

3.3 System Implementation

3.3.1 Mechanical Structure Manufacturing

Knowing the physics of a self-balancing robot [19] [20] and with the aim, therefore, of solving the classic problem of the inverted pendulum, the mechanical structure of the Figures 3.4, 3.5 y 3.6,designed with SolidWorks is proposed and from which the rest of the components will be assembled.

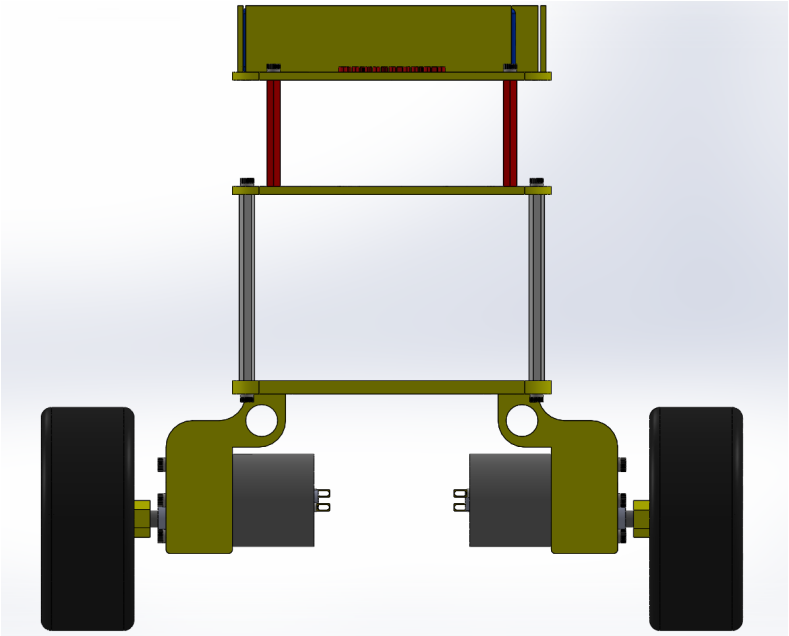

Figure 3.4: Frontal view Balancing Robot.

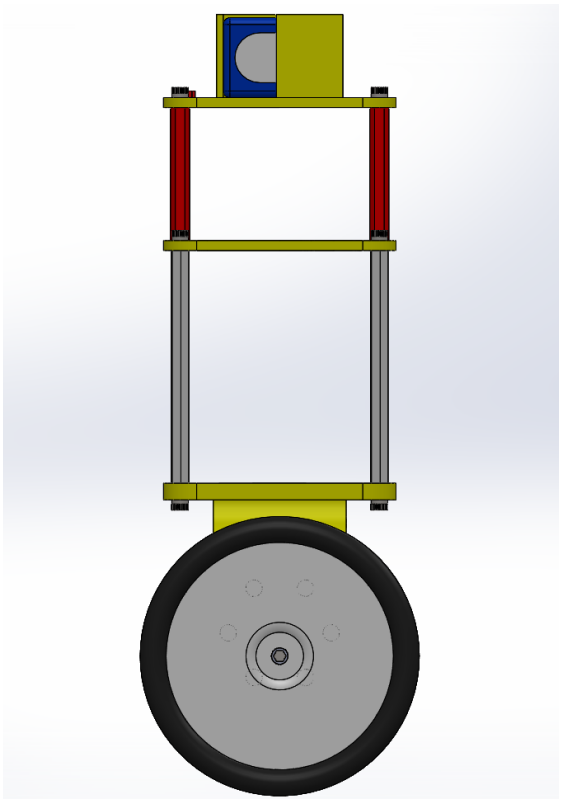

Figure 3.5: Lateral view derecha Balancing Robot.

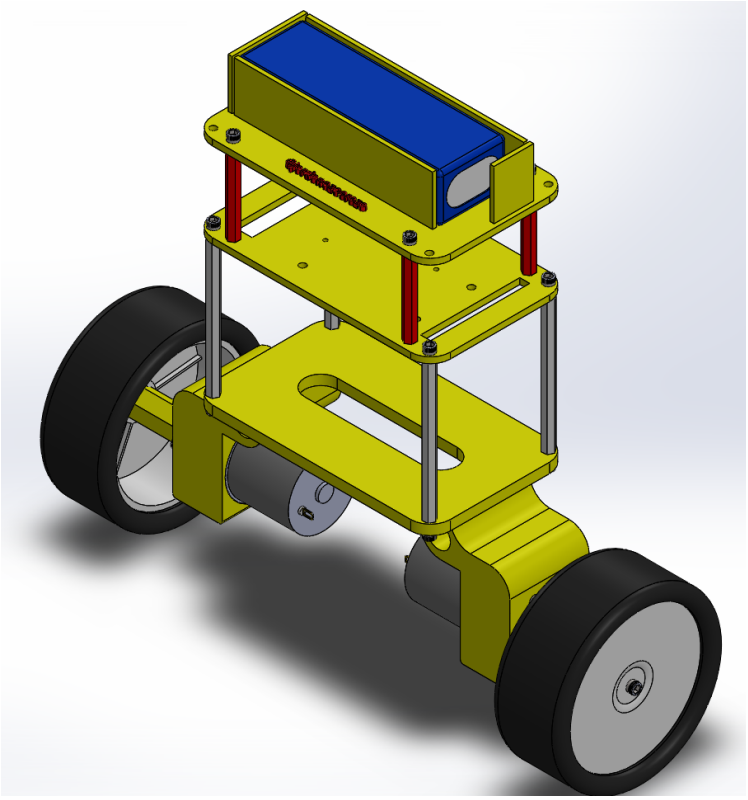

Figure 3.6: Balancing Robot perspective.

Different aspects of the design of this structure are considered, which are directly related to the physics of a self-balance Robot, and with it, of the inverted pendulum.
As argued in section 3.1, a system at rest is stable when its mass center is closer to the horizontal plane. If we consider that the nature of the proposed system is inherently unstable, it is necessary to know the best point where the mass center must be in order to allow a better stability.

Assuming the of mathematical modeling characterization, it is therefore assumed that to achieve greater ease in stabilization, the center of mass should be placed above the midpoint of the vertical axis of our system. Therefore, we must consider the weight of all components for a placement that allows the above.
In Figure 3.7, a SolidWorks calculation is represented from this mass center where there have only been considered the heavier components of the final system, which includes, DC motors, batteries, mechanical structures and wheels.

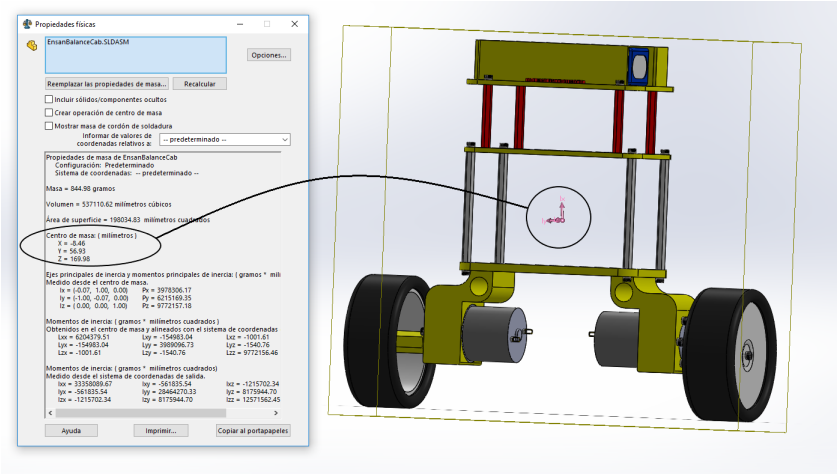

Figure 3.7: Center of mass final system.

3.3.2 Inertial Measurement Unit MPU6050 in Arduino Nano

A constant knowledge of the angle of the system is necessary for its analysis and correction, for this purpose the MPU6050 sensor has been used, connected by an i2c communication to an Arduino Nano.

The MPU6050 is an Inertial Measure Unit (IMU) with 6 degrees of freedom (6DOF) manufactured by Invensense. It has an accelerometer and gyroscope and allows communication by both SPI and i2c bus. To correct some of the data collection problems, mentioned in section 2.4 it incorporates an internal processor (Digital Motion Processor, DMP) that executes data fusion algorithms (Motion Fusion) to combine the measurements of the internal sensors, avoiding having to perform the filters externally.

Due to its low cost and big quality, this is one of the most used IMUs nowadays. In Figure 3.8 a MPU6050 image is shown.

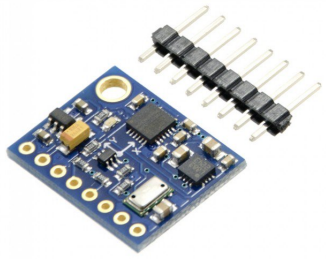

Figure 3.8: MPU6050 IMU.

Pin-out

Figure 3.9 shows the schematic connections diagram of the MPU6050.

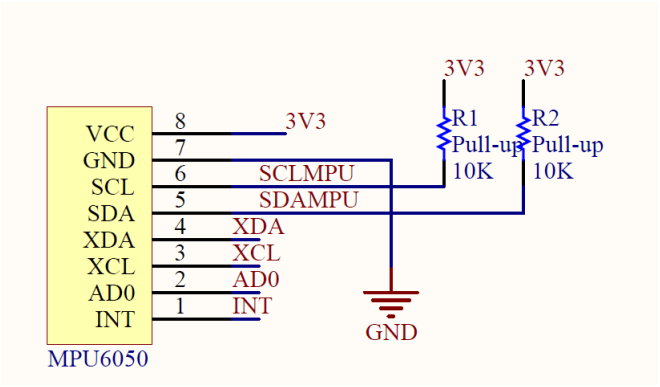

Figure 3.9: MPU6050 IMU.

It has a 3.3V power supply voltage. The clock pin for the I2C connection (Serial Clock Line, SCL) and the data pin (Serial Data Line, SDA) represent the connection for the bus with Arduino Nano. AD0 pin allows the user to change the address of the MPU (slave), which by default is 0x68h connected to GND. If it is connected to Vcc, the address changes to 0x69h. INT pin produces a signal on high when the data in issue is available from the MPU to be captured and will warn by an interruption to the Arduino Nano with the purpose to be obtained.

Arduino Nano Program

For the Arduino Nano implementation it is used a library developed by Jeff Rowberg. The reason of the use of this library is because it incorporates the usage of the DMP. This use exempts the microprocessor (Arduino Nano in this case) from a complex filtered and calculation with the purpose to obtain the values pitch, yaw and roll. A representation of this advantages are represented in Figure 3.10.

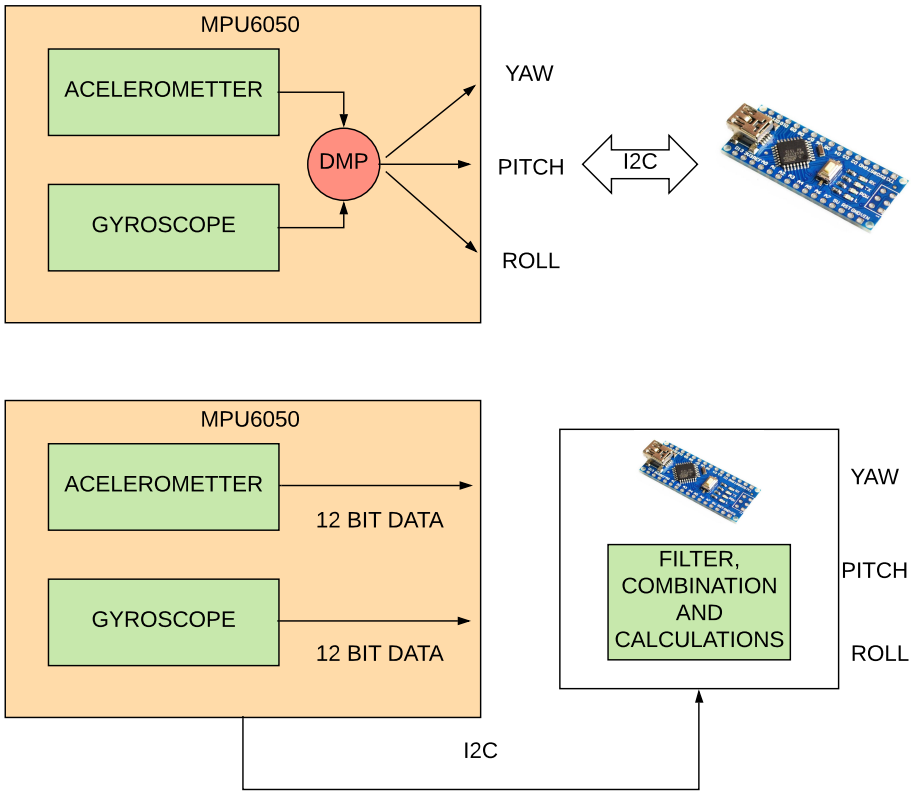

Figure 3.10: Advantaje in the use of DMP.

3.3.3 PCB Implementation

After featuring the entire system and considering the necessary connection diagram not only between the microcontroller and FPGA but also for the OV7670 (it uses in section 4) and the motor driver, it is convenient and appropriate a printed circuit that solves some noise problems, excessive cables, etc.

The printed circuit contains the following components and behaviors:

- A total of 28 pins on the outside of the PCB and arranged in the correct position for a fit in the IceZum Alhambra II board (Figure 3.11), which allows to use as inputs or outputs the pins of the FPGA. In order to know the exact position of the pins on the plate, the Altium project was used,

which is available on GitHub.

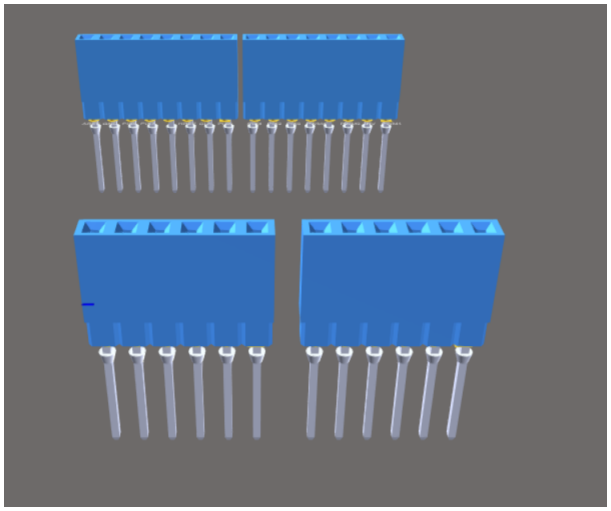

Figure 3.11: Pin headers for IceZum Alhambra II.

- 4 VDC and 12 Volt GND connections to power the ESCs of the brushless motors of the aerial vehicle. For this, the component of figure 3.12, has been chosen, because it has adequate features of maximum temperature to which it may be subjected and which would be analyzed further on.

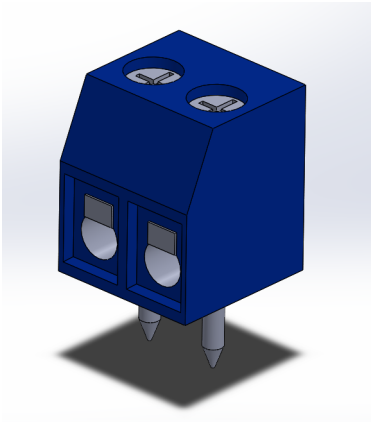

Figure 3.12: Connector GND and VCC.

- A VCC and GND connection to feed the previous connections. This connector would go directly to a LIPO battery of 11.1V (3 cells) and 2200mAh. The fact of choosing this battery is due to the minimum voltage by which the ESCs of the brushless motors are fed, as well like the motors and the motor driver used for the self-balancing Robot. A more detailed analysis of the battery can be found in subsection **??**

- Header pin modules for the MPU6050 connection previously described. (Figure 3.13).

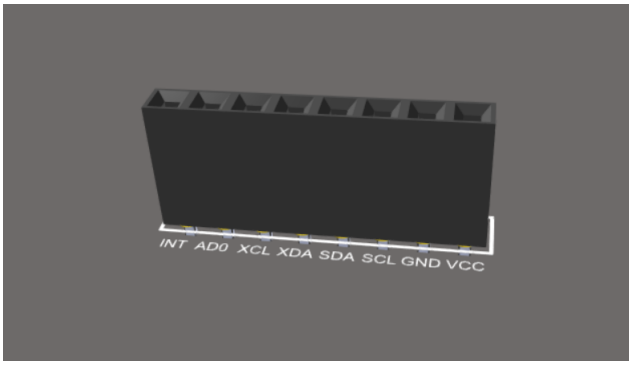

Figure 3.13: Module MPU6050.

- An extension of the most important pins of the MPU6050 is made so that they can be used by the microcontroller in the case of angle analysis, as in this project, is part of a process governed by the microcontroller.
- Two jumpers connection (Figure 3.14) in I2C bus allow the user to decide who governs the SDA and SCL line, microcontroller or FPGA.

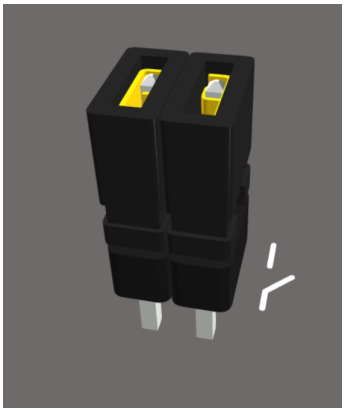

Figure 3.14: Jumpers to configurate i2c MPU6050.

When hosting a bus line, it is not necessary to consider the thermal characteristics of the connector in issue, and when working at a relatively small frequency, the noise that the jumpers can introduce into the I2C communication may be accepted.

- A great part of actual microcontrollers works at 3.3-5V but the input voltage supported can go high to 12V, because of that it is taken advantage of the LIPO battery power and a new connector with two header pin is implemented that goes to VCC and GND which will power the microcontroller.

- A module that can host the OV7670 camera (section 4)formed by male headers pin and each of which will be attached to one of the FPGA's in-out pins, as will be seen later in the general schematic.
- A module that can host the driver of the DC motor used in this case and that allows direct connection with IceZum Alhambra II pins, for example, the pwm signal that will define the speed of the engines will be an output on one of the FPGA pins, and it should be an input to the motor driver.
- So that there are no errors in the I2C transmission, there's a 4,7KΩ resistor in each of the lines.

In Figures 3.15, 3.16, 3.17, 3.18 is included a 3D representation of the final system with all its added component

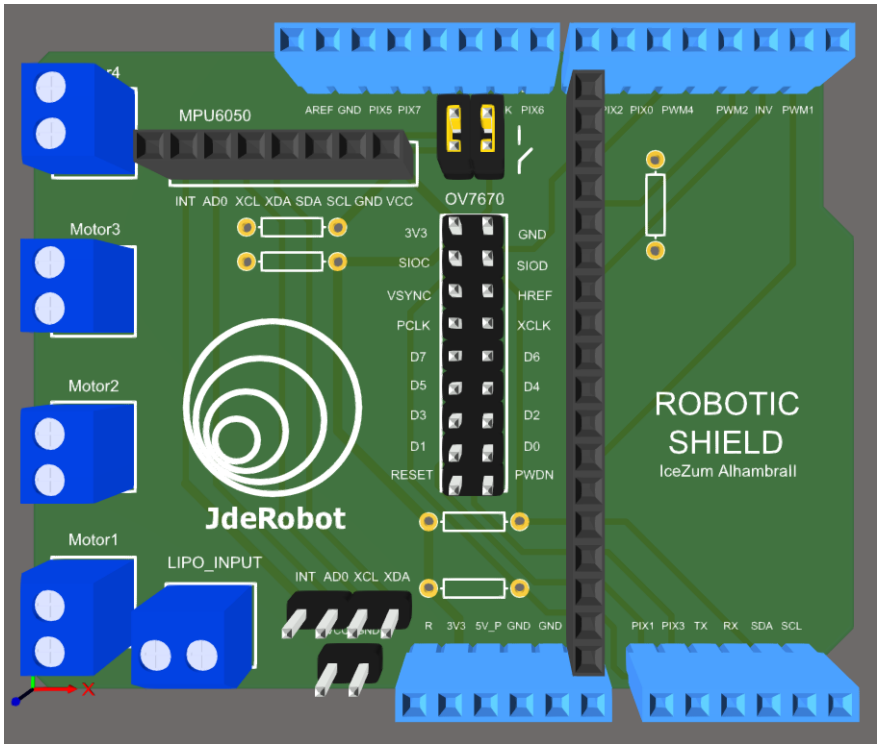

Figure 3.15: 3D View of Shield for IceZum Alhambra II.

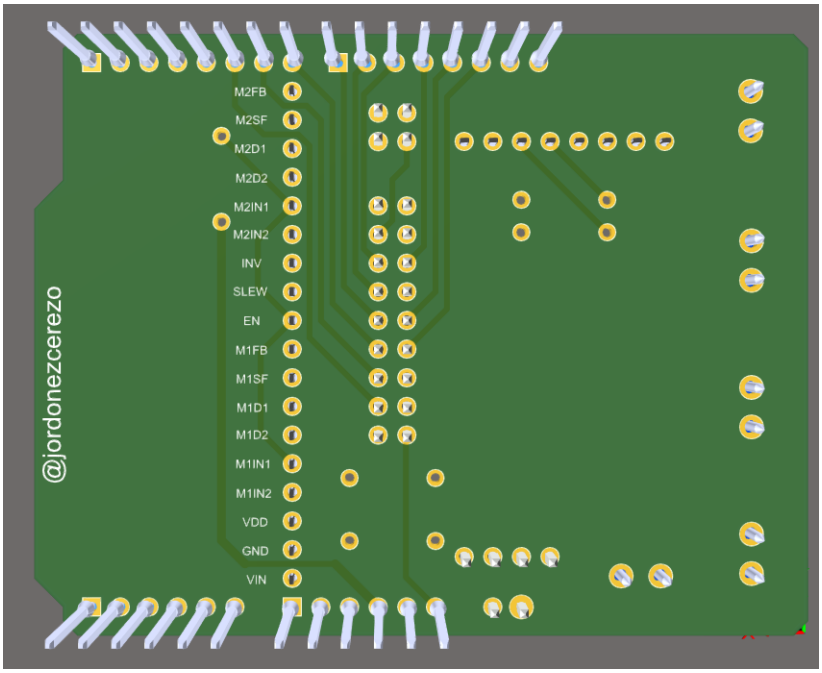

Figure 3.16: 3D View of Shield for IceZum Alhambra II.

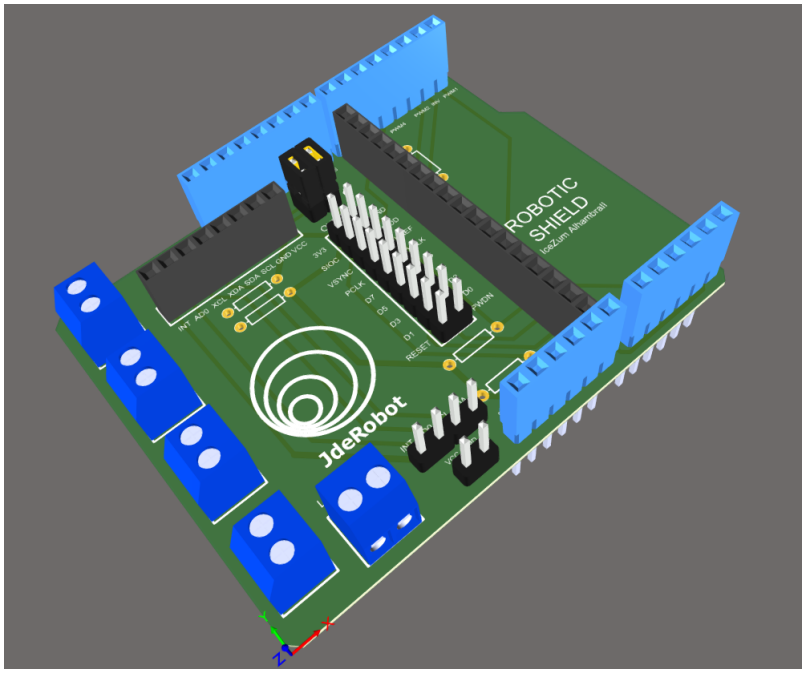

Figure 3.17: 3D of shield for IceZum Alhambra II.

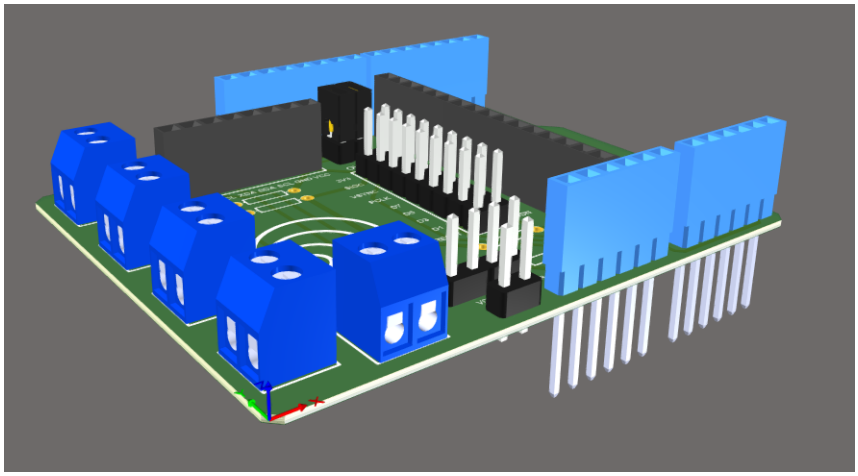

Figure 3.18: 3D of shield for IceZum Alhambra II.

For a PCB development with the described requirements it has been used Altium Designer. The elaboration process of a PCB in Altium can be different depending on the final user, but in this project the following root has been followed:

- At first the project in issue is created.
- For each of the components used a new library is created, formed by the schematic and the layout in the PCB.
- Once all the libraries are created, the schematic of the plate is designed, taking special care that the connections are adequate.
- With the schematic already created, the PCB can be implemented defining its edges, tracks, pads, etc.

The scheme is represented in 3.19.

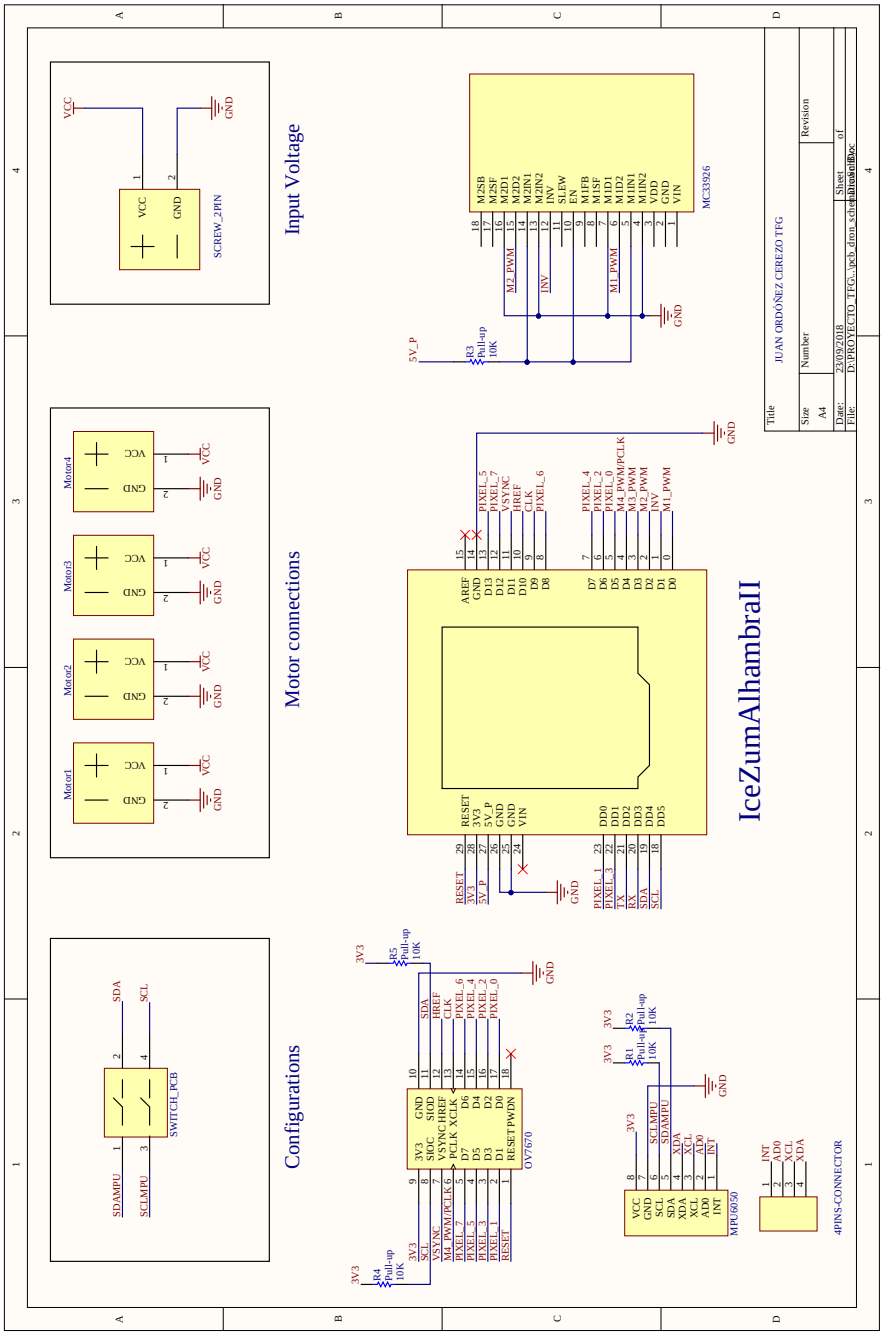

Figure 3.19: Schematic Shield IceZum Alhambra II

Trace size

When designing the PCB, the size of the copper trace should be considered, especially if the current could be very high. Otherwise, the PCB could suffer damage or even burn out.

To calculate the trace width, it is necessary first to know the maximum intensity that could circulate through it. It is based on the knowledge that one of the engines of the unmanned aerial vehicle (this is the worst case and for which the calculation of tracks is necessary) can consume a maximum of 20 amps. This amount has been extracted from the datasheet and is usually common for this type of motors.

The formula used to obtain the trace width has been extracted from the standar IPC-2221B, which establishes the generic requirements for the design of PCB.

The formula is defined as in equation 3.1:

$$I = K * dT^{0.44} * (W * H)^{0.725} \tag{3.1}$$

Donde:

- I = Max. Intensity (A).
- dT =Temperature rise over ambient (C^o)
- W,H = With and Height (mils)
- K = 0.024 for internal traces and 0.048 for external traces.

The following results were obtained either for inner and outer traces:

$$W_{externas} = 18.716mm$$
$$H_{externas} = 0.035mm$$
$$W_{internas} = 48.6876mm$$
$$H_{internas} = 0.035mm$$

The thickness of the layer is provided by the manufacturer. In this project, this web has been used for ordering the PCBs.
Trace thickness is commonly measured in "oz" and in this case, the manufacturing company allows a 1oz trace thickness in a PCB with, which is equal to a 0.035mm thickness.
If the results obtained from equation 3.1 are analyzed, the difference between a track in an inner layer and a track in an outer layer is clearly seen.
A PCB is divided into layers, each of the layers has its function and organizing them properly is a good practice to avoid bad behavior. Thus, once the requirements were analyzed, two layers were necessary for the connection of data pins, in addition a ground plane is always recommended in which all the pins connected to GND have a common layer. This avoids noise and interference besides ensuring a good ground reference.
In the used PCB provider, there is no price difference between a three-layer PCB

and a four-layer PCB and considering that in the best case the trace width for the brushless motors should be of 1.8716cm, it was reached the determination of the usage of one of the layers as a common plane for the powering traces of the motors. The distribution of the layers would therefore be represented in Table 3.1.

For the rest of data traces, it is assumed a width of 20 mm, which allows a 1.46 A current, enough for this application.

Top Overlay	Overlay
Top Solder	Solder Mask/Coverlay
Top Layer	Signal
Dielectric 1	Dielectric
VCC	Signal
Dielectric 2	Dielectric
GND	Signal
Dielectric 4	Dielectric
Bottom Layer	Signal
Bottom Solder	Solder Mask/Coverlay
Bottom Overlay	Overlay

Table 3.1: Layers composition in PCB.

A 2D image of the Top Layer, VCC, GND, Bottom Layer, Top Overlay and Bottom Overlay layers are shown in Figure 3.20.

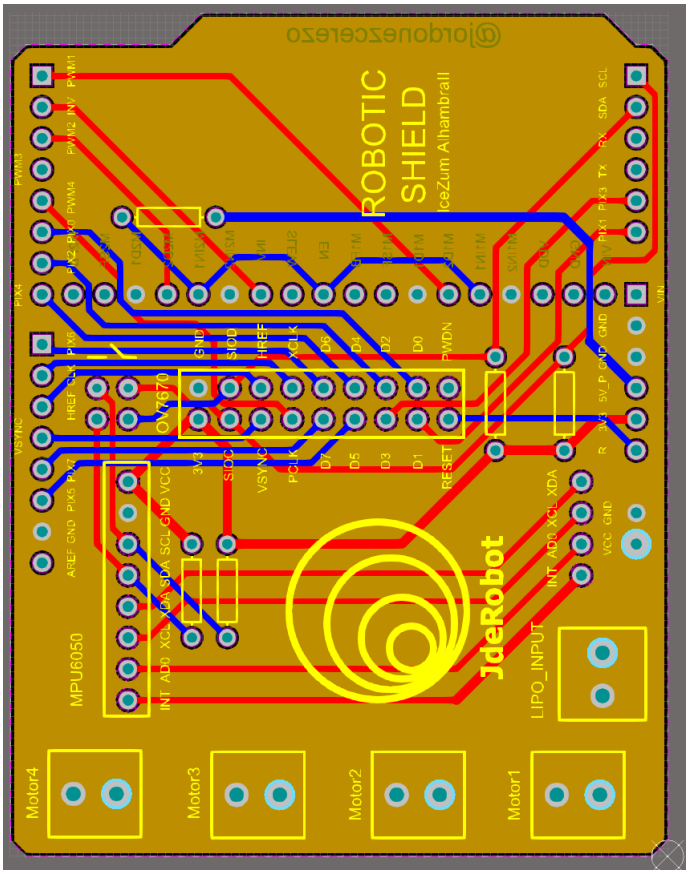

Figure 3.20: Composition layers in Altium.

3.3.4 IceZum Alhambra-Arduino Nano Implementation

An integration between a microcontroller and FPGA allows to distinguish sequential and parallel tasks, assigning each process to the microcontroller if it necessarily has to be sequential, or to the FPGA if the process can be parallelized and obtain with it some advantages

There are some options to make a Microcontroller/FPGA integration:

- Emulate the behavior of a microcontroller in an FPGA.
- Physical coexistence of an FPGA and microcontroller creating a communication between each one of them.

In this project the second one has been chosen as an option because there are not enough resources to carry out the first one, although this would be the most adequate in terms of saving resources and ease of use.

There are two possible communication types to this purpose:

- Serial communication: It is a sequential communication. The bits are sent one by one, sequentially and using only one data bus.

- Parallel communication: All bits of each symbol are sent at the same time.

To make usage of the communication in parallel you need as many channels as bits to have the information to be transmitted (if you want to send a byte you should use a total of 8 channels, corresponding to each bit). In this case there will be used 8 pins of the FPGA to be able to carry out this type of transmission. Therefore, and despite the fact that parallel communication is quicker than serial communication, the first one is chosen as the most convenient option.

The type of serial communication developed could be alike an SPI protocol, although it only has data capacity in one direction. The schematic of this developed communication is shown in Figure 3.21.

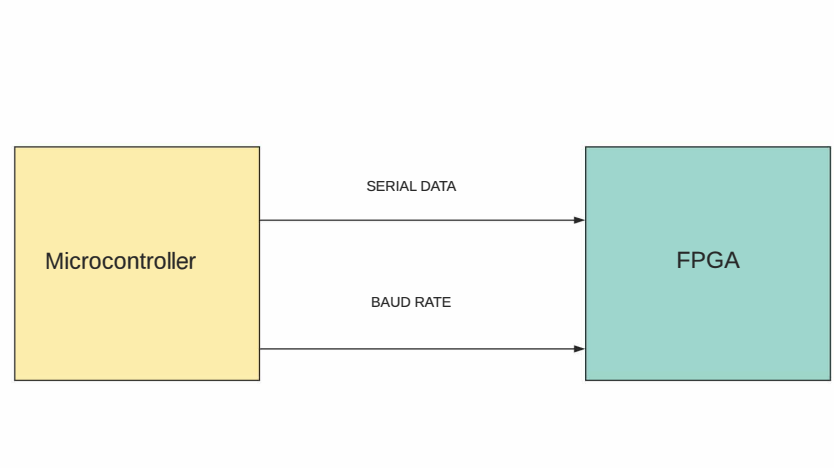

Figure 3.21: Hardware coexistence microcontroller-FPGA.

The general system has two connections:

- A data line to send the information.
- A clock line so that the FPGA could obtain at every time the information speed.

A possible example of a serial communication of a data byte could be shown in Figure 3.22.

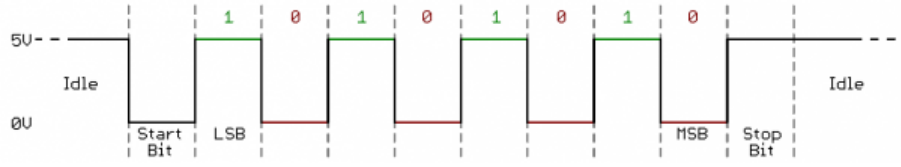

Figure 3.22: Serial communication example.

The angle has to be known in the IceZum-Alhambra in order to make decision both in direction as in the speed of the engines. It is therefore necessary a microcontroller/FPGA communication (Figure 3.23).

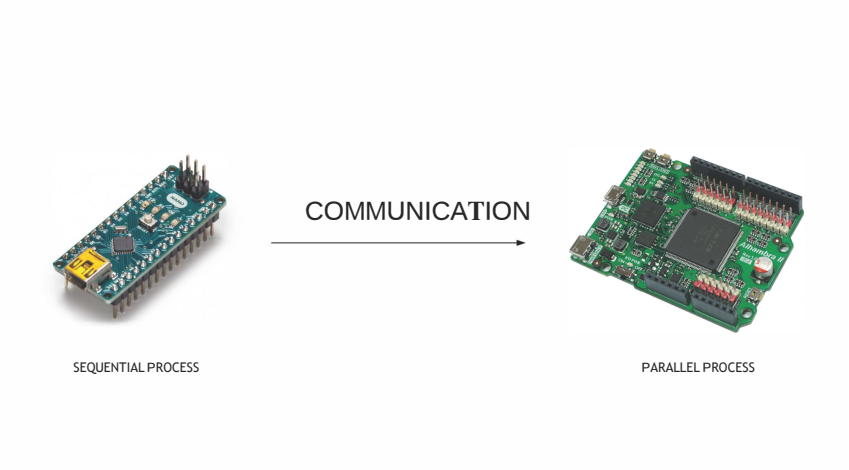

Figure 3.23: Separation microcontroller-FPGA.

The communication would be only unidirectional, the microcontroller will send information to the FPGA about the current angle of the object in issue with the purpose that the FPGA analyzes and actuates starting from that angle.

Therefore, there's a two-part separation: from the point of view of the microcontroller and from the point of view of the FPGA. Both will be explained specifically in subsection 3.3.4 y 3.3.4

Microcontroller POV

For the development of this sub-chapter, it is assumed that the current angle has already been obtained in the microcontroller. Thus, and for the clarity of the reader, an internal schematic of the Arduino Nano is shown in figure 3.24. In this section and according to the part of the microcontroller, only the shadowed part is analyzed.

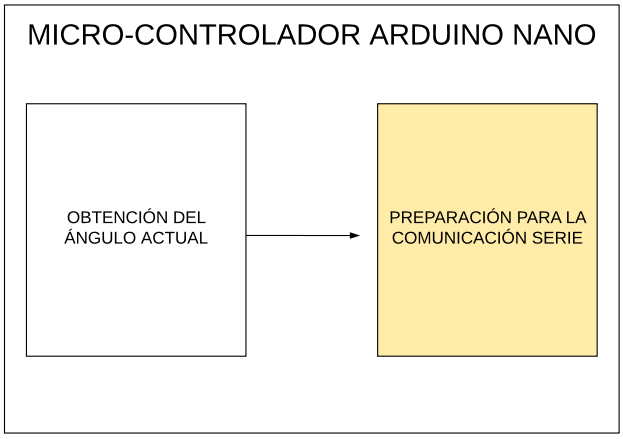

Figure 3.24: Inner diagram Arduino Nano.

The flow diagram in which the C code of the microcontroller is based on, is shown in Figure 3.25.

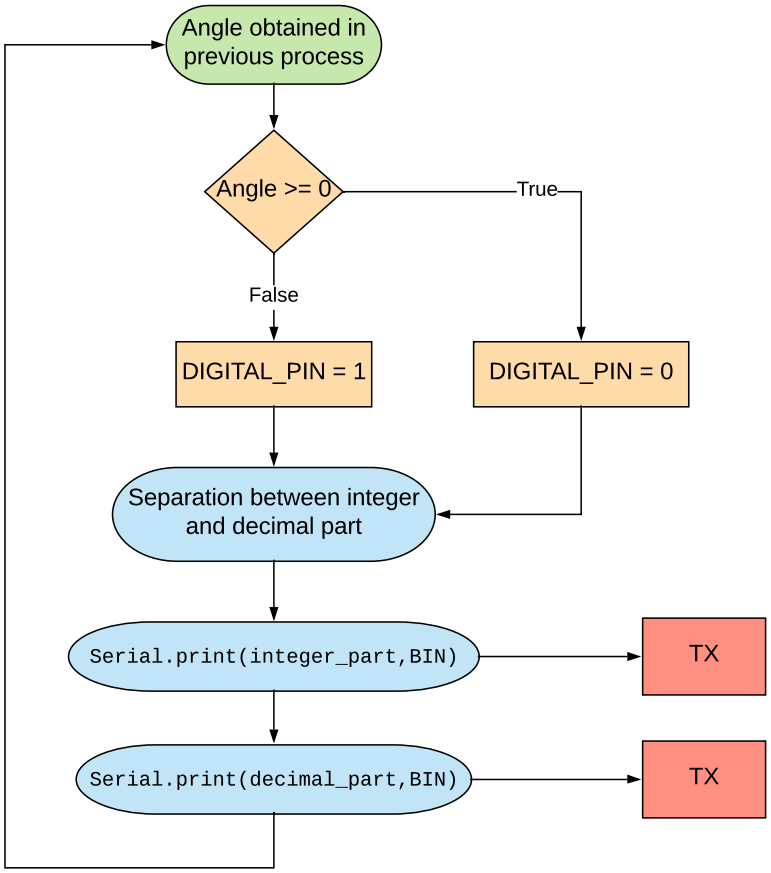

Figure 3.25: Flow diagram to send angle.

The most relevant features will be explained:

- It is based on the assumption that the angle has already been obtained and is correct. In section 3.3.2 you can find more information about it.
- As it has been said before 3.3.2, one of the most important parts to correct the control of the self-balance robot, is to know the inclination at every moment to correct this desviation (3.26). To do this, the FPGA must know if the angle is positive or negative. This aspect would not form part of the communication protocol itself, as will be seen below.

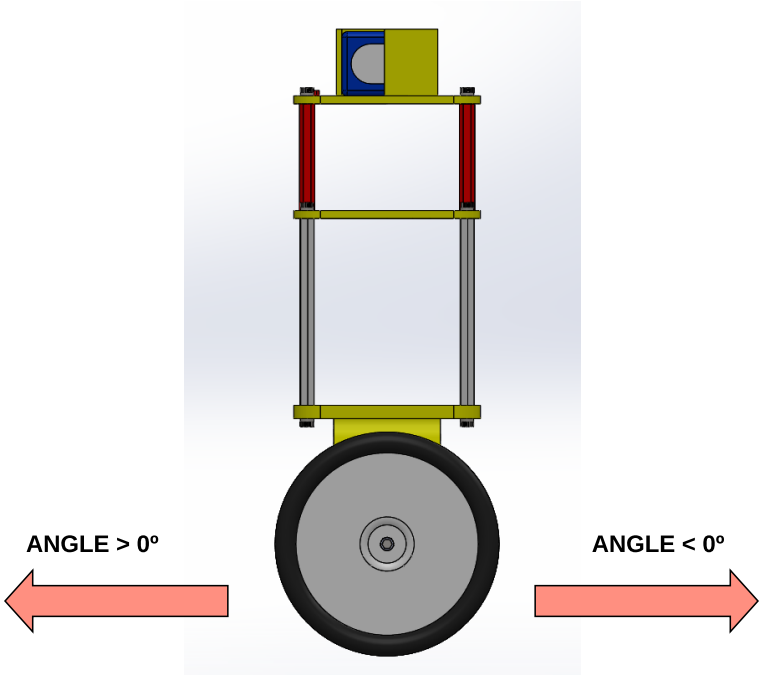

Figure 3.26: Correction angle in Self-Balance Robot.

- A single pin of the microcontroller is used, which will change its value to 1 or 0 depending on the sign of the angle at each moment. This way the FPGA will only have to read this information when it is necessary.
- For a correct understanding by the FPGA it is necessary to send the represented Angle in bytes and not in ASCII code. This is why for that sending the "Serial.print(Angle)" command will be used. This Arduino function sends by the serial port the representation in bytes of the angle in issue.
  If the whole angle is intended to be sent, both the symbol and the ASCII code of the comma will be sent, and this is an aspect that does not matter to be considered when processing the FPGA is referred. To correct it, first the module of the angle is made (the sign is not needed because there is already a pin designated for it) and later it makes a separation between the integer and the decimal part in order to eliminate the "comma" character.
- The integer part corresponding byte is sent by the serial port.
- The decimal part corresponding byte is sent by parallel port.
- This is a loop that will be reproducing each "x" seconds, this is, the angle that could be corrected each "x" seconds.

FPGA POV

For an input PIN to the FPGA it will be continuously entering data from the transmission pin and so that it can make a correct reading of the byte it is necessary to know:

- When a byte transmission starts.
- When a byte transmission ends.
- When a bit can be captured.
- When the necessary bits are saved in a buffed until the byte is complete.

In order to implement an intermediate module in the FPGA with the previous features (aspect in IceStudio Figure 3.27) it is necessary previously know the speed of the transmission by the microcontroller, which has been 16200 baud.

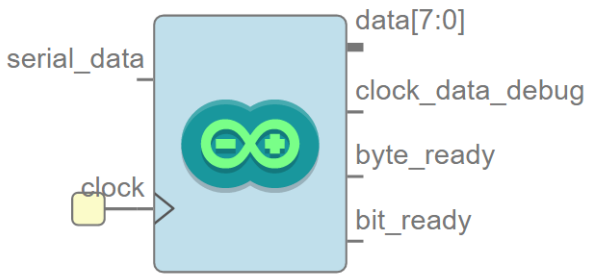

Figure 3.27: Appearance of Arduino Nano module in IceStudio.

The inputs and outputs from the previous module are detailed:

- data[7:0]:Consists in a buffer in which bits are being stored when it is necessary until we have the byte. It is important that the following module knows when the byte is prepared for its capture.
- clock_data_debug: The usage of this output is for depuration only.
- byte_ready: A clock flag will change its state when a byte is ready to be captured. As there has been seen in previous developments, this byte will be both integer or decimal part.
- bit_ready:The usage of this output is for depuration only.

The implementation in IceStudio of the previous behavior is implemented by two machine states with their corresponding sensitivity lists and the flow diagram is represented in Figure 3.28.

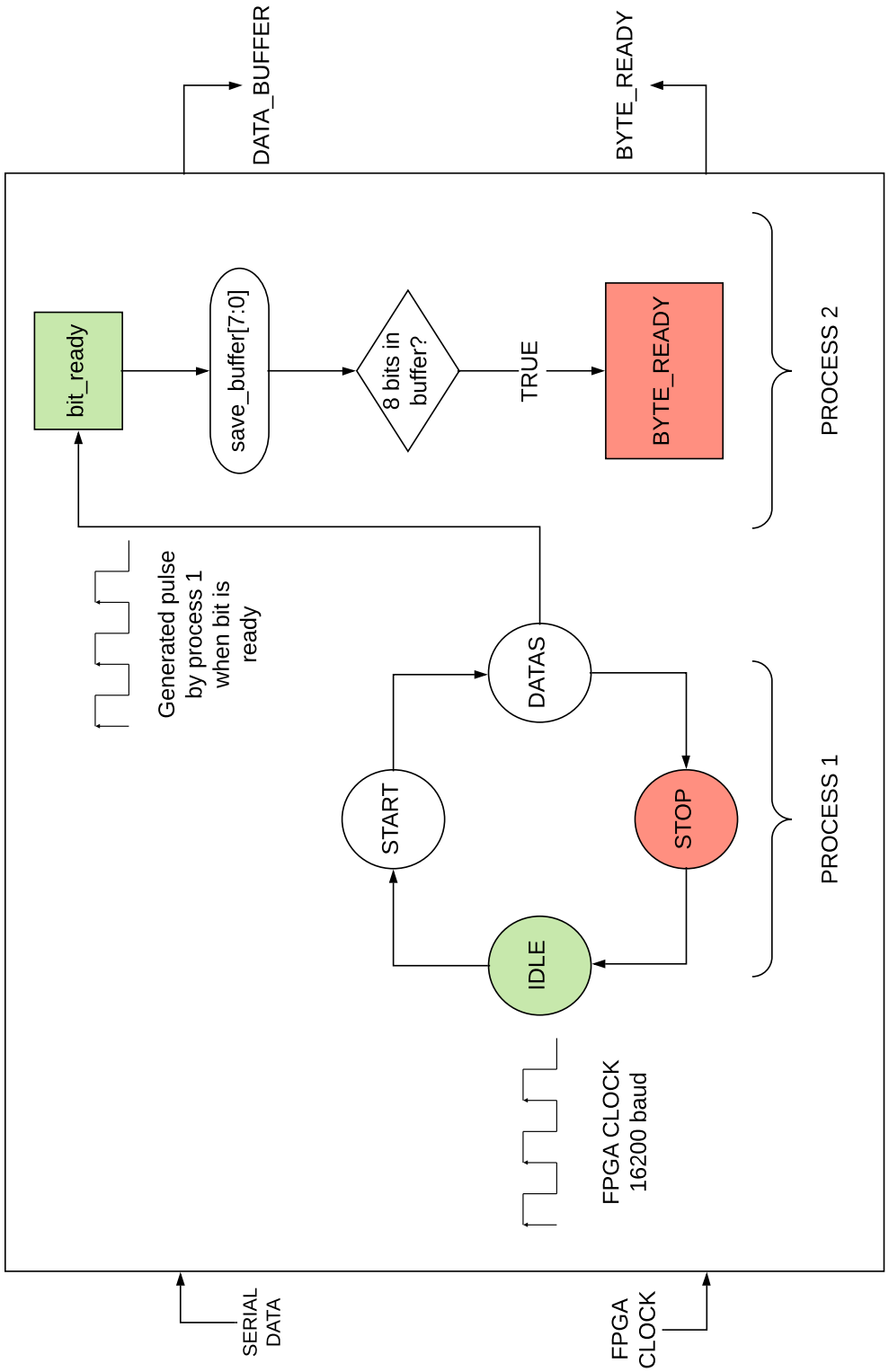

Figure 3.28: Flow diagram for Arduino interface.

Two well differenced processes are used:
Process 1: This process only gives the next system the exact moment at which it can capture a bit and save it in the buffer, for this, you must know the speed of the transmission discussed above. The states would be the following:

- IDLE: The process remains in this state until the transmission starts, which will go to the next state (START).
- START: How was it developed in the section ... The serial transmission protocol begins with a start condition, this state will allow to recognize when this condition ends to start saving bits in the buffer.
- DATA: Since the transmission speed is already known and the condition of START in the previous state has been recognized, in this state a flag will change its value when the bit is ready to be stored in the buffer, of which process 2 will be in charge of.
- STOP: In addition to a START condition, the serial transmission protocol used in Arduino has a STOP condition. This state allows to recognize the time it takes Arduino to carry out this last condition, then, it will return to the first state until a new transaction begins.

Process 2: This process activated by process 1. When process 1 determines that a bit is available on the bus to be captured, it will set a clock flag on, initiating process 1 through a sensitivity list. The flow diagram could be:

- Wait until the sensibility list is activates, this will indicate that a bit could be captured.
- Bits will be storing in a buffer that will form a byte, which will represent the integer or decimal part of the angle at that moment.
- When the byte is prepared to be captured by two consecutive modules, a channel will be on, being available both the outputs and the buffer with the 8 bits and the channel "byte_ready".

At this point the FPGA is able to differentiate when it can capture a byte (BYTE_READY) and from where it has to capture the data bus (DATA_BUFFER). However, an aspect that is not part of the communication itself, it is important to analyze if you want to get a correct operation. That is, if it has been previously said that the microcontroller continuously sends the integer and decimal part of the angle, if a good interpretation of this data is not made, it is possible that an angle on the FPGA is formed by a decimal part of an angle n and the integer part of the angle $n + 1$.

To do this, a module is created in IceStudio that is capable of ordering these values. The aspect of this module in IceStudio is shown in Figure 3.29. 3.29.

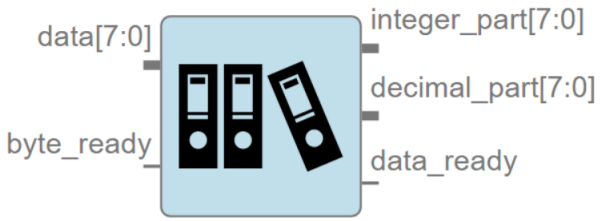

Figure 3.29: Module to arrange data from Arduino.

As Inputs there are:

- Data [7:0]: This is the output buffer of the previous module where all captured bits are being stored until the byte is completed.
- byte_ready : Clock flag which activates when the byte is available to be captured.

It is important to consider that the data will be available as long as the entire part as well as the decimal part of the angle in issue is available. Thus, as outputs are:

- integer_part[7:0] :Byte that represents the integer part of the angle.
- decimal_part[7:0] : Byte that represents the decimal part of the angle.
- data_ready : Clock flag that activates when the data (decimal and integer part) is waiting to be captured.

As has been previously discussed, the flow diagram of the code in Verilog that implements the previous behavior is shown in Figure 3.30.

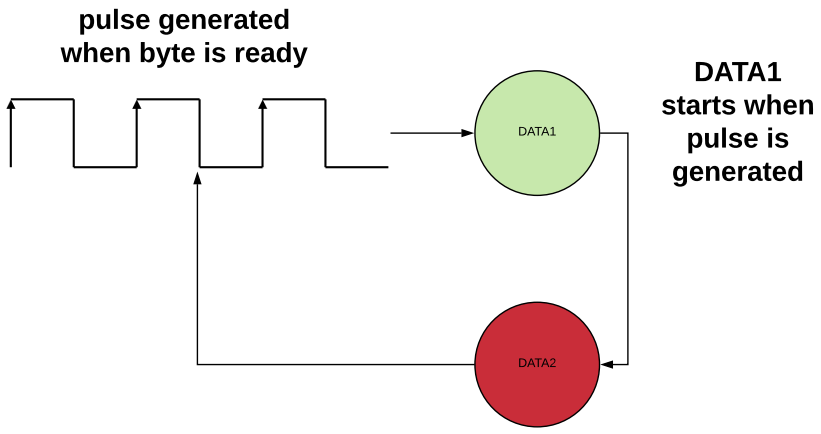

Figure 3.30: Flow diagram to arrange bytes.

In this case is a process with a sensibility list that counts as a sensible signal the "byte_ready", bus and which is the output from the previous module.

Each time the flag is activated that indicates that a byte Is ready to be captured, a cyclic state machine starts that counts with the following states:

- DATA1: It is the first data to be captured and corresponds to the whole part of the first angle. The second time the "byte ready" flag is activated, it corresponds to the decimal part of the first angle and therefore it will pass to the next DATA2 state.
- DATA2: In this state not only is the decimal part of the angle in issue captured, but also a new flag is activated, which indicates that the complete data (Angle) is ready to be captured.

Thus, the module will have as outputs, the buffer of the integer part, the buffer of the decimal part and a bus that will notify the successive modules of the complete data is ready to be captured. In this way, the problem explained above of the non-ordering in the data arrived from the microcontroller has been avoided.

The Arduino-FPGA communication protocol is terminated and the necessary tools are provided so that the successive processes and modules can know the angle at each moment represented by its integer part (8 bits), decimal part (8 bits) and a pin that indicates the value of the sign (positive or negative).

The final communication system between Arduino and IceZum Alhambra from a POV of the FPGA is represented in Figure 3.31.

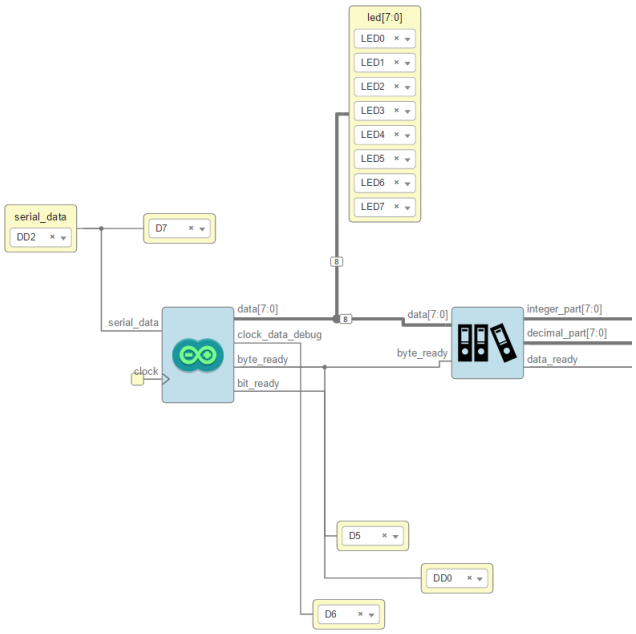

Figure 3.31: Communication between Arduino and IceZum Alhambra.

3.3.5 PID Control in IceZum Alhambra

As it has been explained in section 2.7, a PID controller can be used in a simple way to control the stability of the system.
One of the facilities in the use of this kinds of controllers is because its ease of implementation.

P Controller

The flow diagram that features the behavior of the P controller is shown in Figure 3.32.

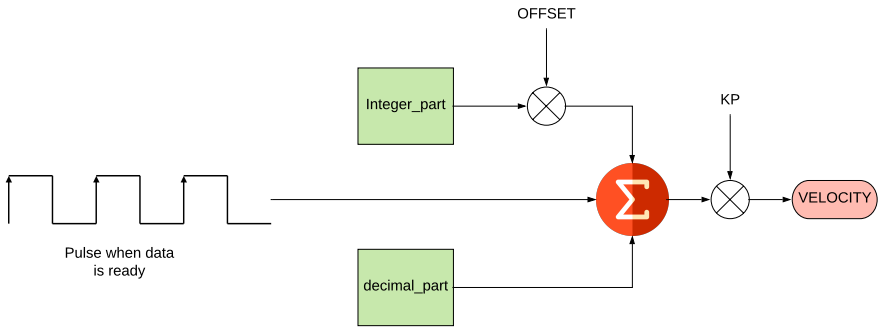

Figure 3.32: Flow diagram P control.

Due to its importance, the functionality is briefly explained ahead:

- 1.-The complete process will be refreshed every time there is a new datum, this is in this case, every time the angle system is changing.
- 2.-As inputs there are both the integer part as the decimal part from the angle.
- 3.-Both the integer part and the decimal part are represented as 8 bits data without sign. So, in order to give more importance to the integer part, there is the option to divide the decimal part by 100 or to multiply the integer part by 100. In the first option there is not a good behavior obtained due to the digital treat of the floating comma, which is why the second option is better.
- 4.- At the end, the two integer and decimal components are added and then it is multiplied by a Kp constant, defined as a parameter which can be didactically changed.

Its representation in IceStudio has the aspect as shown in Figure 3.33.

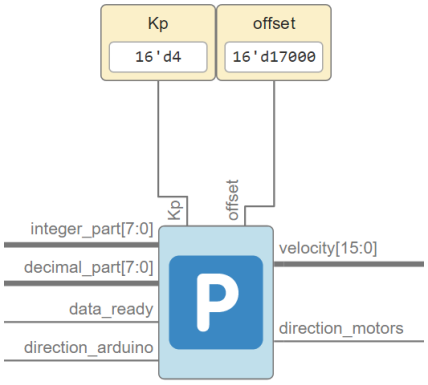

Figure 3.33: Appearance of P control in IceStudio

D Controller

Referring to the D controller, the flow diagram implemented in Verilog is shown in Figure 3.34.

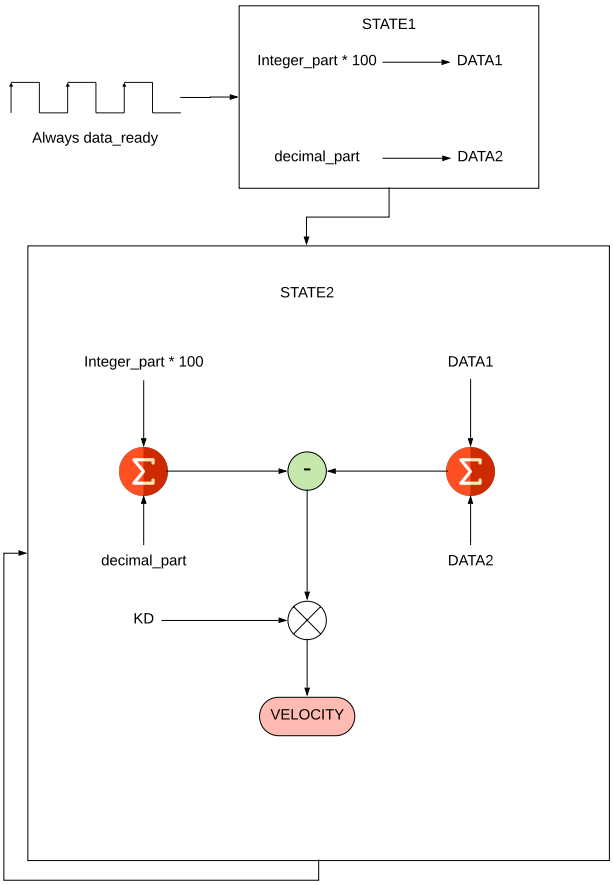

Figure 3.34: Flow diagram D control.

Its implementation is composed by a state machine with two states, which will change at each pulse on the $data_ready$, which means, will change whenever a new angle is available.

If remembered, D controller is based in its operation on the prediction of future errors. The derivative control action generates a control signal proportional to the derivative of the error signal.
A subtraction (derived from the error in time) is therefore carried out between the current error and the last error. Its result is multiplied by the constant Kd. Its representation in IceStudio is represented in 3.35.

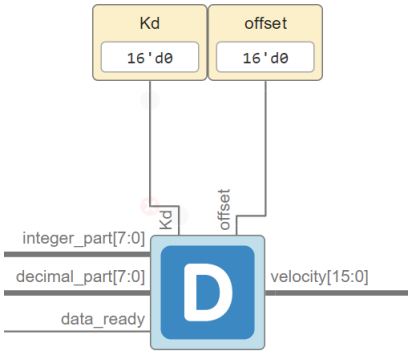

Figure 3.35: Appearance of D control module in IceStudio.

Controlador PD

Referring to the closed loop feedback system analyzed in section 2.7, it is important to show the final result of the aspect of the present project in IceStudio, making a direct comparison between this (figure 3.36) and figure 2.30.

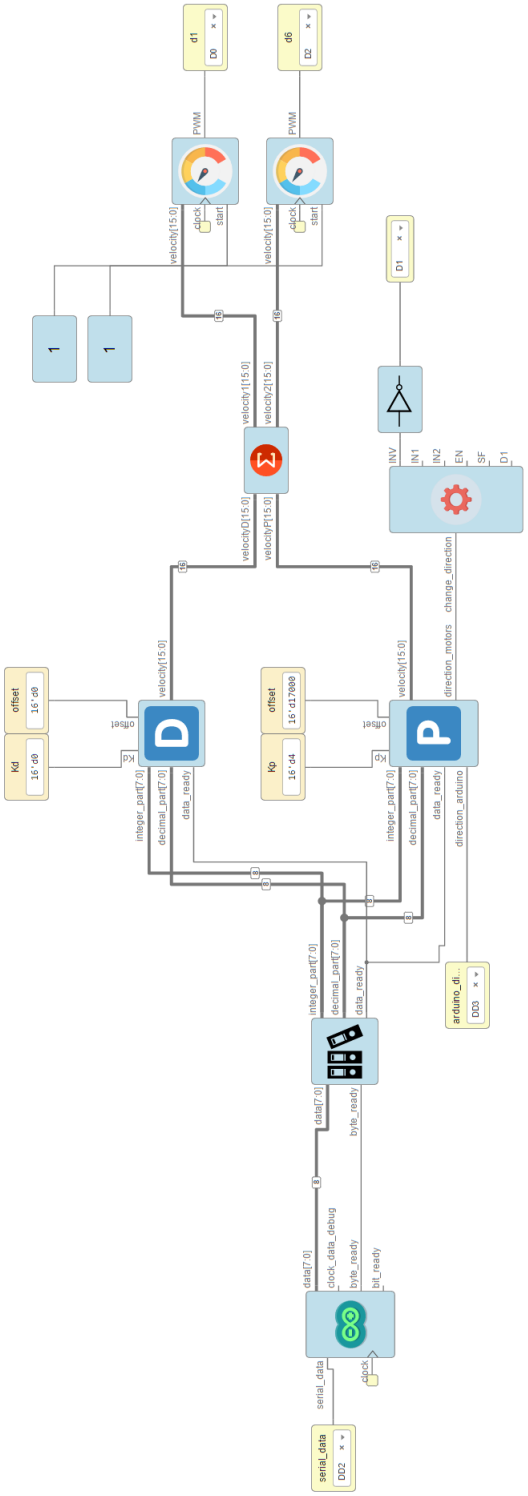

Figure 3.36: Final appearance of Self-Balancin in IceStudio.

3.3.6 Motor controller

Having a module with the ability to generate PWM solves many subsequent problems while improving the visibility of the code in the final system. As can be seen in the motor features, most of them are commanded by a PWM signal that, although it is true, depends on the motor in issue.
Pulse Width Modulation (PWM) is a technique that modifies the work cycle of a periodic signal (squared in our case) that is used to transmit information through a communication channel or to control the amount of energy that is sent to a load. An example of a PWM squared signal is shown in figure 3.37.

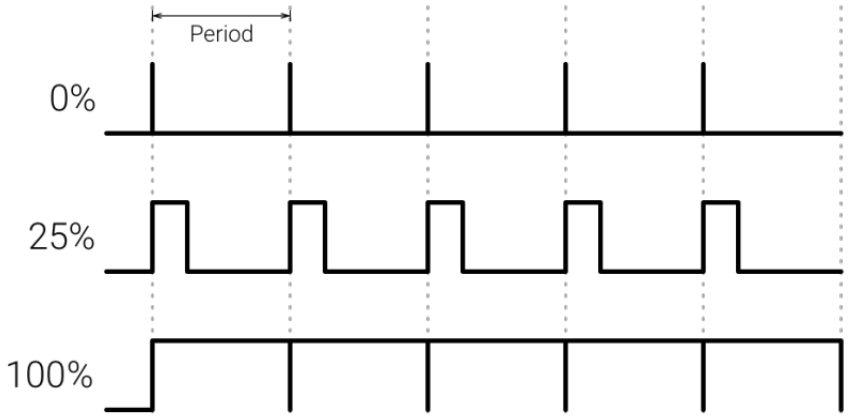

Figure 3.37: PWM signal example with different dutty.

Applying this signal, for example, to a classic DC motor the amount of energy that is applied to the load is varied, the motor in this case. It simply works as a switch in which a high logical level is open and a low level closed. If it is managed to vary the time the motor is being charged and the time in where there's no current, you can control its speed.

The features of a PWM signal are:

- D = Work cycle.
- τ = Time-lapse while the function is positive.
- T = Function period.

The motor driver used to control DC motors (MC3392 figure 3.38),needs a series of configurations to work according to the needs which could be obtained from its datasheet. The final inputs and outputs diagram and also the necessary connections are detailed in the scheme of the 3.39 figure.

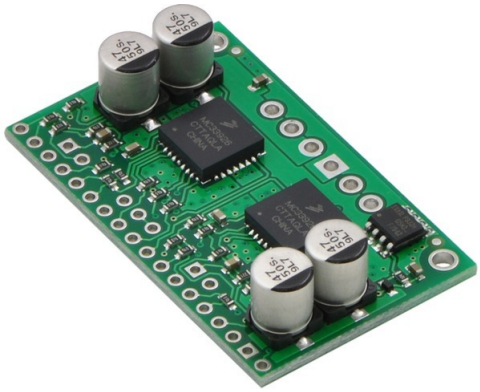

Figure 3.38: MC33926 to control DC motors.

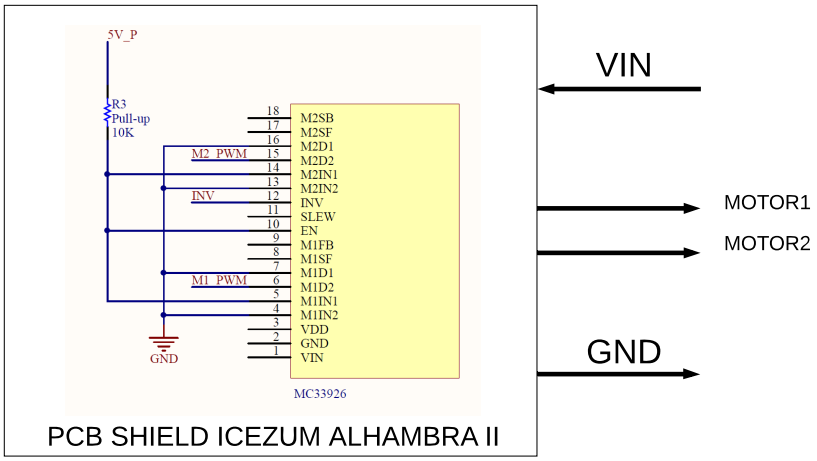

Figure 3.39: Schematic MC33926.

PWM Control

The speed of the motors is controlled by a PWM which is connected to pin 15 or M2D2, from figure 3.39 for each one of the motors and another PWM generator connected to pin 6 or M1D1 from the DC motor driver. This generator module has the aspect shown in figure 3.40 in IceStudio.

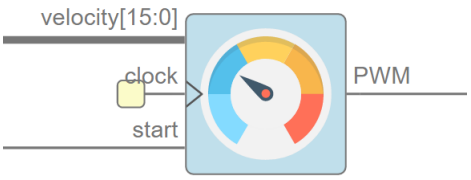

Figure 3.40: Appearance PWM module in IceStudio.

The block diagram of its performance is exposed in figure 3.41.

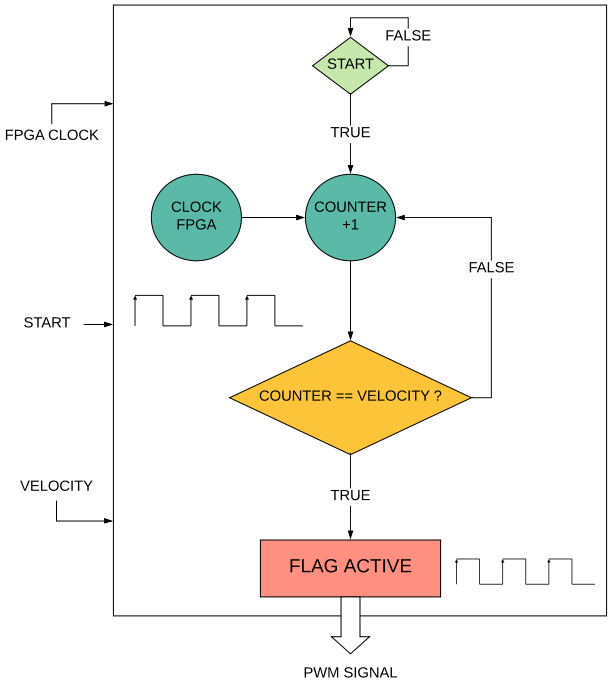

Figure 3.41: Flow diagram PWM generator in Verilog.

Its performance is based in an input clock pulse counter from the FPGA. A speed log will dictate how many pulses have to be counted before an output signal is settled on (desired PWM). As inputs are:

- FPGA Clock: Is the 12Mhz FPGA clock which is in charge of the pulse counter.
- Start: It is a common signal to the whole system, performing as a switch to give a general start.
- Velocity: Is a record that marks how many clock pulses need to be counted before activating the output signal PWM.

As outputs are:

- As outputs are:

3.3.7 Power Supply System

As with any electronic system, a power source is necessary to allow the correct operation of all the components.

As a fundamental task, an analysis of the requirements of these components that make up the complete system is a priority in order to choose an adequate power supply. In addition, it is considered that the purpose is to have a mobile system and that, as far as possible, a direct connection to the electrical network or a USB connection to a computer is avoided. So, it only remains to choose what type of battery is suitable.

Next, the different types of battery currently in the market are named, analyzing their most important advantages and disadvantages:

- Lead-acid Batteries: They are inexpensive and easy to manufacture but do not admit overloads or deep discharges, besides, they are heavy and have a big volume compared with the small amount of energy they are capable of storing.
- Nickel-cadmium Batteries (Ni-Cd): They work well over a wide range of temperatures and can be overloaded without damage. They allow deep discharges and provide a good number of cycles. As in the previous one, they have a very high weight and volume.
- Nickel-metal Hydride Batteries (Ni-MH): Features compared to the previous batteries are improved, however, it provides a fewer number of cycles.
- Lithium Ion Batteries (Li-ion): In comparison with the previous ones, these are from a recent development and have facilitated the existence of portable technologies. They have a high capacity in relation to their weight and volume, they have a very high self-discharge factor. They are almost unaffected by the memory effect and can be charged without having been previously discharged. On the other hand, they do not support temperature changes too good.
- Lithium Polymer Batteries (Li-Po): They are a variation of the Li-ion batteries that improve their weight and volume features as its discharge rate. They remain practically unused if they are discharged in excess.

Considering that the final and most restrictive system is a remotely piloted aerial vehicle, and that the self-balance robot needs a not very high weight, it is important that the weight features, volume, and discharge are adequate. For this reason, Li-po type batteries are chosen for the power supply of the systems in this project, which can store a large amount of energy and offer a very high discharge rate.

The Li-Po type batteries have a different nomenclature from the rest, which is necessary to analyze:

- Sort by number of cells "S": The number S corresponds to the number of cells, which are 3.7 volts but can reach 4.2 if they are fully charged. A 3-cell (3S) battery is composed of 3 sub-batteries placed in series, which is, a total of 11.1 volts.
- Capacity indicated in "mAh": The higher the number of mAh, the higher the load capacity. A common mistake is to think that the greater the capacity, the greater the possibility of lengthening the time of the system in issue. At higher capacity, weight and volume of the battery increase, so the best configuration for this system must be found.
- Download rate "C": The number from C corresponds to the battery discharge rate. If a battery is 1C, it means that the maximum discharge rate it can reach is the one corresponding to its capacity. If the number C is different from 1 means that we multiply the discharge rate by that value, reducing the discharge time proportionally, that is, a battery of 1000mAh 2C will be discharged to 2A in half an hour.

The following approach will be to choose which of the previous values is the most adequate for the system. For this and after analyzig the differents components separately, two independent subsystems are distinguished in terms of the power supply:

- DC Motors and Arduino Nano Power Supply.
- IceZum Alhambra II and other components Power Supply.

DC Motors and Arduino Nano Power Supply

For the DC motors and Arduino Nano power supply, a 11.1V and 2200mAh LIPO battery of is used as in figure 3.42. As it is presented in the final schematic from the controller shield (Figure 3.19), the battery is connected to the shield, which is in charge to supply both the motors and Arduino Nano.

Figure 3.42: LIPO Battery 11.1V y 2.2A.

Alhambra IceZum II and other components Power Supply

For the IceZum Alhambra II power supply and the rest of the components, a 3.7V and 4mAh LIPO battery has been used, as shown in Figure 3.43.

Figure 3.43: LIPO Battery 3.7V y 4mAh.

3.3.8 Materials and Prototype Cost

In the table 3.2 the total cost of the prototype along chapter 3 is pulled apart. It is differenciated by four columns called material, number of units, unit cost and total cost in euros.

MATERIAL	QUANTITY	UNIT COST (€)	TOTAL COST (€)
IceZum Alhambra II	1	60	60
Arduino Nano	1	8	8
Hexagonal Nylon 10 mm Separator	4	0.20	0.80
Metálico Hexagonal M3 25 mm Separator	8	0.25	2
Metálico Hexagonal M3 50 mm Separator	4	0.41	1.64
Nylon M3 Screw	16	0.09	1.44
Nylon M3 Nut	8	0.05	0.40
Nylon M1 Screw	4	0.09	0.39
Nylon M1 Nut	8	0.05	0.40
PCB 4 layers	1	5	5
Wheel 7 cm	2	7.90	15.80
Motor DC	2	24.95	49.9
Driver Motor DC Dual MC33926	1	30	30
IMU MPU6050	1	2.50	2.50
Mechanical 3D Structure	1	20	20
Screw 3.5 mm	5	0.80	4
Cable 5-pin	1	0.82	0.82
PIN 2,54 mm 4 contacts Macho	1	0.08	0.08
PIN 2,54 mm 8 contacts Shield	2	0.54	1.08
PIN 2,54 mm 6 contacts Shield	2	0.58	1.16
Jumper 2.54 mm	2	0.08	0.16
PIN 2,54 mm 10 contacts Female	2	0.10	0.20
PIN 2,54 mm 10 contacts Male	5	0.10	0.50
Resistance 4K7 5%	3	0.21	0.63
Tin	1	5.50	5.50
Retractable therman Tape	2	1.50	3
Cable 12 AWG	1	0.90	0.90
Battery Lipo 11.1 V 2200 mA	1	19.95	19.95
Battery Lipo 3.7 V 5 mA	1	4.20	4.20
PROTOTYPE TOTAL COST:			**240.45 €**

Table 3.2: Total cost of Self-Balancing Robot.

3.4 Experiments and final results

3.4.1 Self-Balancing Robot

A set of demostrative video of the correct behaviour in the Self-Balancing robot can be seen in [21]. Also, the process to the end in [22] [23].

In order to manufacture the mechanical structure, a 3D printer was used [24].

A set of images wich form the final system are shown in 3.44, 3.45.

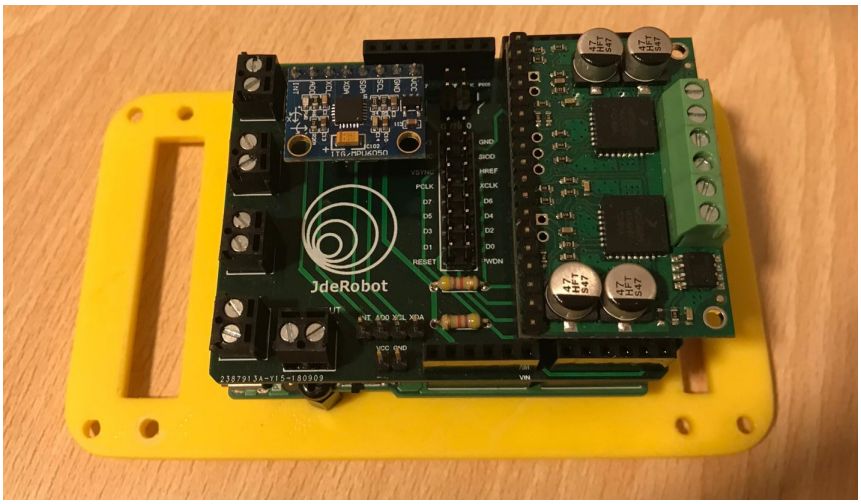

Figure 3.44

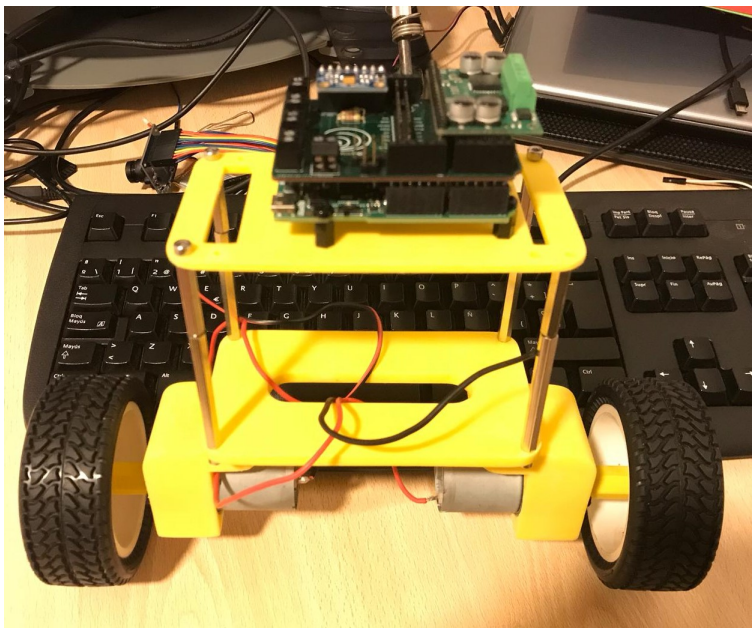

Figure 3.45

3.4.2 VGA Module

For a more adequate knowledge of the hardware implementation language Verilog, prior to the realization of this project and as an initial idea to use it in future projects, a shield is carried out for the connection with a screen, having to know for it this communication protocol. A video demonstration is found in [25].
The manufactured shield is represented in the figure 3.46

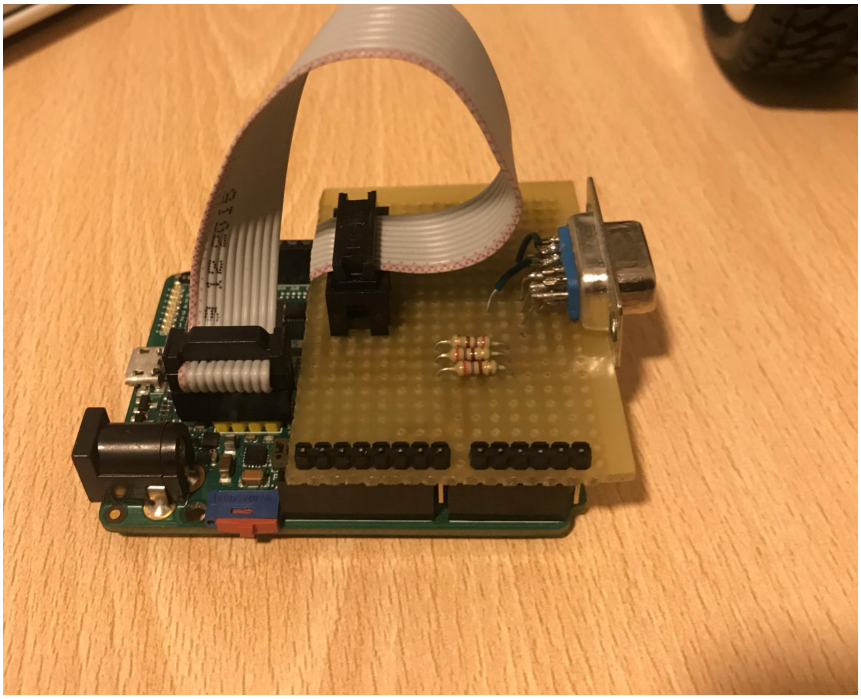

Figure 3.46

3.4.3 Motor brushless Controller

With the fundamental idea of using the PCB for a totally independent quadcopter system and as a first approximation to the control of brushless motors, the model of the figure is developed such that it allows a control over this type of actuators. A demonstration video can be found in [26].

Chapter 4

Quadcopter with artificial vision

In this chapter it is pretended to approach the design and development of a quad-copter which is able to follow an object, using for that a FPGA as a fundamental part and some other tools that will be exposed throughout this section.

4.1 Design

In the following figure, a high abstraction level scheme is presented in which a first approach about the general system functionality is given. (4.1).

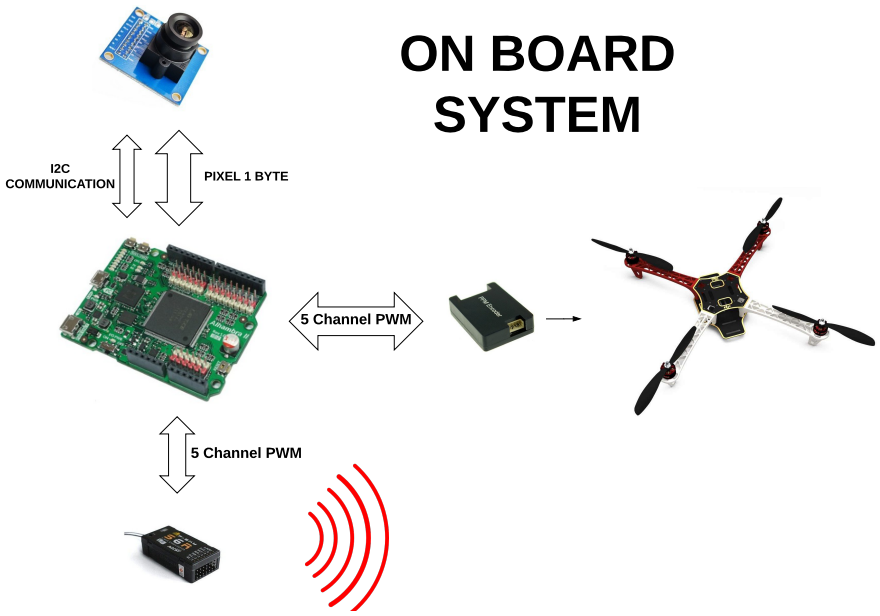

Figure 4.1: Vision quad-copter high level design.

The design of this project has two parts, the part of stabilization control and the part of perception. As for the perception part, a low-cost camera will connect to the FPGA, which should implement the entire object recognition algorithm. Prior to this, and through the i2c protocol, the camera will be configured in such a way to make this recognition simpler. Subsequently and as an output of this recognition algorithm, the FPGA generates 4 PWM channels corresponding to the yaw, pitch, roll and altitude. It makes use of a PPM encoder, which transforms these 8 PWM signals into a PPM channel, input of the stabilization system of the quad-copter "Pixhawk".

4.2 Perception Implementation

Module OV7670[27] has a CMOS VGA image sensor OV7670, which allows to work at a maximum of 30 frames per second and a resolution of 640x480 pixels. It is a System on Chip (SoC) that is capable of processing images, such as: exposure control, gamma, white balance, color saturation, tone control. These parameters can be configured through the SCCB interface[28] (Serial Control Bus Camera). Some of the most important features are presented below, and which could be obtained from its datasheet[27].

- 3.3 VDC Operation Voltage.
- Sleep state current. μA
- 8 bits parallel data transmission.
- SCCB Standard control Interface compatible with I2C.
- optic objective 1/6"
- Field of View (FOV): 25°C
- 640x480 VGA resolution
- 1.3V/(Lux-sex) sensibility
- Signal to Noise Ratio: 46 dB.
- High sensibility in low light environments.
- Low voltage, according to portable applications

An image from OV7670 module is pressent in Figure 4.2.

Figure 4.2: OV7670 camera.

The camera configuration is done through an SCCB interface[28], very similar to an I2C communication and whose registers can be found in the datasheet[27]. Therefore, it is necessary to choose a suitable configuration for this purpose:

- A 640x480 window size is chosen because of the limited resources of the FPGA card used.
- The type of output data will be RGB, for the simplicity of use. We will therefore obtain two bytes for each pixel captured in the order represented in 4.3.

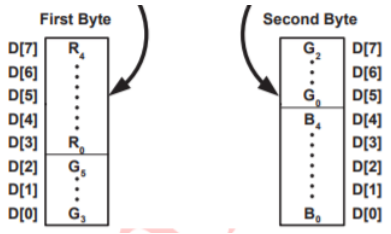

Figure 4.3: OV7670 Pixel Formation

- The velocity of adquisition in the outputs pixels has to be defined. This signal is in charge to notify when the next pixel can be captured. it is imposed that this signal is the half of the input clock. If the input clock frecuency is 12Mhz, the pixel signal generation will be 6MHz.

I2C Protocol

In order to achieve this configuration, an I2C communication is necessary with the OV7670 module that allows writing in certain records.

The appearance in IceStudio of this writing in I2C has the aspect of Figure 4.4.

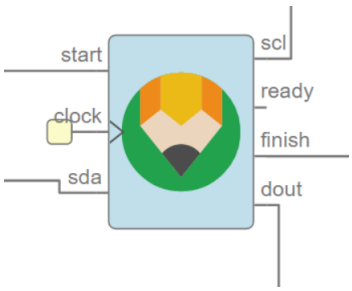

Figure 4.4: I2C writing module in IceStudio aspect.

Since it has been an entirely module developed for this application, it is represented in the flow diagram in figure 4.5.

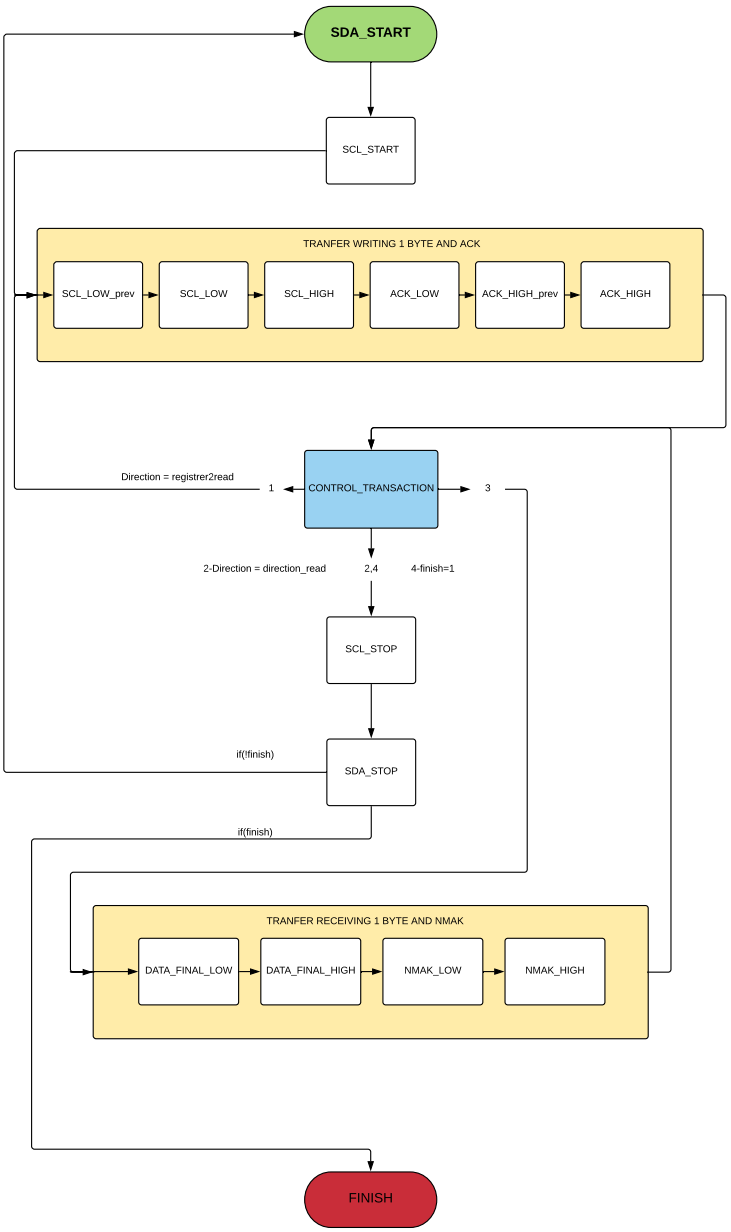

Figure 4.5: Flow diagram of I2C writing.

It is composed of a machine of 13 states in which each one has a fundamental function to achieve the desired result. You have to work at a very low level to avoid errors in the transmission and provide the system with the necessary tools for the detection of possible anomalies.

In an I2C communication it is important that both the master and the slave share the SCL and SDA buses, which will be always on high except when one of the two previous ones impose their path at a low voltage level, ground in this case. I2C bases it functionality in recognizing at every moment this voltage levels, which is why it is important to make sure that there is always the condition that if the bus is free, the voltage level is high. For that, both buses use to have a connection with a pull-up resistor that eases this feature and avoids at every moment possible clock glitches, bad ground reference, noise, etc. The connection schematics from the camera is below represented in figure 4.6.

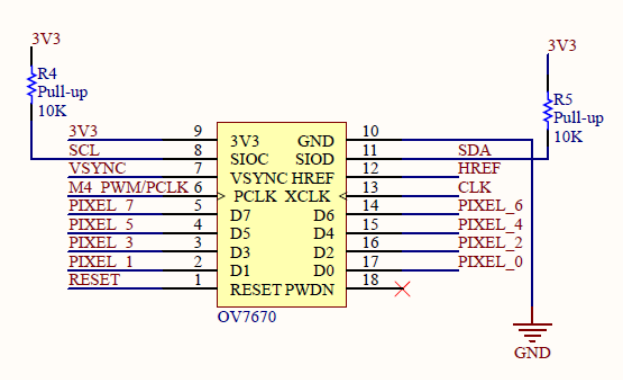

Figure 4.6: OV7670 camera schematics.

One of the most important problem is this module's development, which falls in temporal government of this buses by the slave and the master. For a better comprehension, it is described in example 4.1.

Example 4.1 *It is required at first instance that the master imposes the SDA bus at a high voltage level. Then, the master must be waiting for an answer coming from the slave, which is a non-controlled device and extern to our system. When the slave tries to write in the bus, it will find an imposed voltage level by the master which will not be able to be changed.*

In order to correct the problem exposed in example 4.1 the device waiting for a response much impose its high electric impedance state respect to the bus. This is why a tri-state bus is used which is represented in figure 4.7 and that allows this behavior in a controlled way.

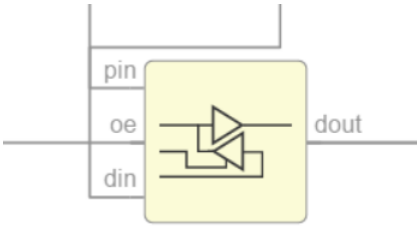

Figure 4.7: Tri-state buffer for I2C bus.

A writing example in I2C in OV7670 module is represented in Figure 4.8.

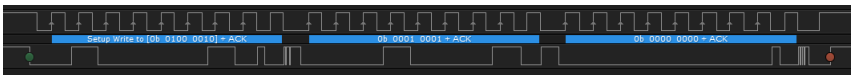

Figure 4.8: I2C on OV7670 writing example.

Pixels Storage

One of the most important problems that must be solved when working with an OV7670 module and with the intention of objects recognizing in real time without delay, is the need for a FIFO memory that stores the pixels that are getting in, since to obtain a complete image of 640x480 it is necessary to store in memory 307200 pixels. In this case and considering that each cell is formed by two bytes, the total size of a frame will be the one shown in equation 4.1.

$$(640 * 480) * (16) = 4915200 \text{ bits por fotograma} \tag{4.1}$$

The device that controls the camera must be able to store that quantity of information so that later could be analyzed.

In case of using a microcontroller, a FIFO memory could be incorporated to the system with the capacity to store that quantity of information. In this project it is used to the image analysis the FPGA IceZum Alhambra board and the shield exposed in 3.3.3. The operating mode changes considerably compared to the previous one.

After analyzing different alternatives, the most adequate and which a better performance could be reached, it is based in the no necessity to store a complete image, to understand this concept, it will be briefly explained in the section related to the position and volume of an object.

Volume and position recognize

The recognition of the volume and positioning of the object will be calculated pixel by pixel instead of when obtained the complete image, thus the algorithm would be as follows:

- At first instance, the color filter will be used, which will allow to recognize the color range of the pixel, in this case of the red ball. Therefore, a maximun and minimun will be defined for each red, green and blue component.
- Pixels will arrive sequentially, two bytes per pixels. The OV7670 module has as outputs pins D7, D6, D5, D4, D3, D2, D1, D0 corresponding to the bits of each component of the panel as shown in Figure 4.6.
- An internal counter will be in charge of keeping track of the number of total pixels that has passed through the filter. When the complete frame is finished, the object volume can be obtained, and with it, the distance to it, as shown in equation 4.2.

$$Volume = Num_{\text{filtered pixels}}/Num_{\text{total pixels}} \tag{4.2}$$

- Considering the position within the frame where the pixels that have passed through the filter are, the position in columns and rows and from which you can obtain an estimate of the position of the object within the area of view as explained in equations 4.5 and 4.8.

$$Acum_X = \sum \text{columns of filtered pixels} \tag{4.3}$$

$$X_{average} = \frac{Acum_X}{Num_{\text{filtered pixels}}} \tag{4.4}$$

$$Error_X = X_{average} - \frac{width}{2} \tag{4.5}$$

$$Acum_Y = \sum \text{row filtered pixels} \tag{4.6}$$

$$Y_{average} = \frac{Acum_Y}{Num_{\text{filtered pixels}}} \tag{4.7}$$

$$Error_Y = Y_{average} - \frac{height}{2} \tag{4.8}$$

To achieve the above behavior, it is therefore necessary to know at each exact moment what the value of the column and row in issue is.
There are two independent modules developed; whose outputs provide these required positions.

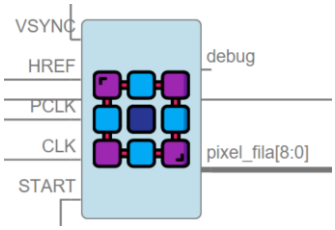

Figure 4.9: Row module in IceStudio.

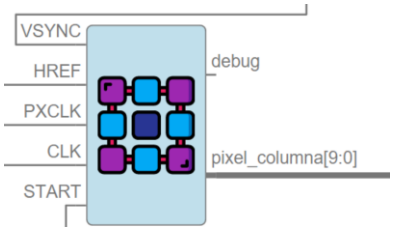

Figure 4.10: Column module in IceStudio.

Due to the importance of these modules, the development of one of them is explained through its flow diagram exposed in figure 4.11.

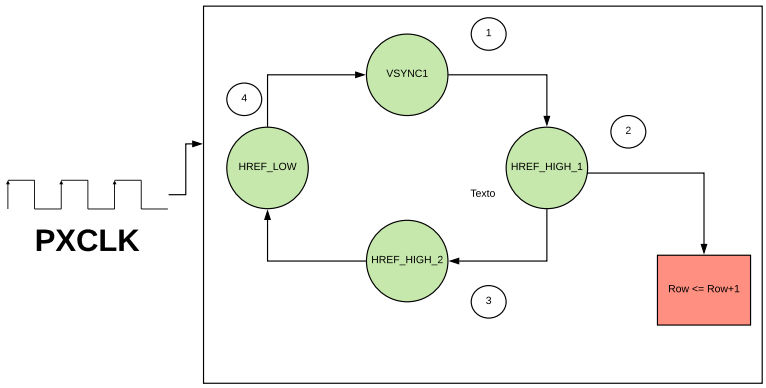

Figure 4.11: Flow diagram to the row counter.

As is known, the OV7670 module provides a clock signal called PXCLK which will provide the synchronization of this module. On the other hand, it is important to know the synchrony signals with which every VGA signal count and which is represented in the figures 4.12, 4.13 y 4.14.

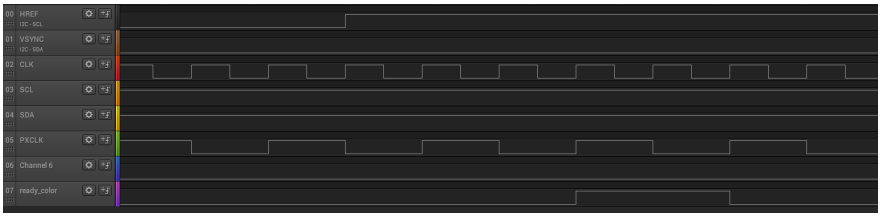

Figure 4.12: Synchronize signal OV7670.

Figure 4.13: Synchronize signal OV7670.

Figure 4.14: Synchronize signal OV7670.

There is a horizontal sync signal that will notify the change of the row in the frame and a vertical sync signal for the frame change.

In the first place in the process of rows recognizing, it is necessary to detect when it begins and when it ends. A state machine has therefore been proposed in which each state assumes a change in the bus of the horizontal synchronization signal.

Once known the actual row and column, an auxiliar module is in charge to store the color components in different register. D7,D6,D5,D4,D3,D2,D1,D0 bits will be received wich correspond with red component and the three most significative bits of green component as it is indicated in 4.9.

$$RED = (D7, D6, D5, D4, D3) \tag{4.9}$$

$$GREEN_{prev} = (D2, D1, D0)$$

In second place, the second byte will be received with the other green part and the hole blue component (4.10).

$$GREEN = (GREEN_{prev}, D7, D6, D5) \tag{4.10}$$

$$BLUE = (D4, D3, D2, D1, D0)$$

The behaviour showed in equations 4.9 and 4.10 can be developed with a state machine as it is represented in 4.15. PXCLK is used as sensibility list.

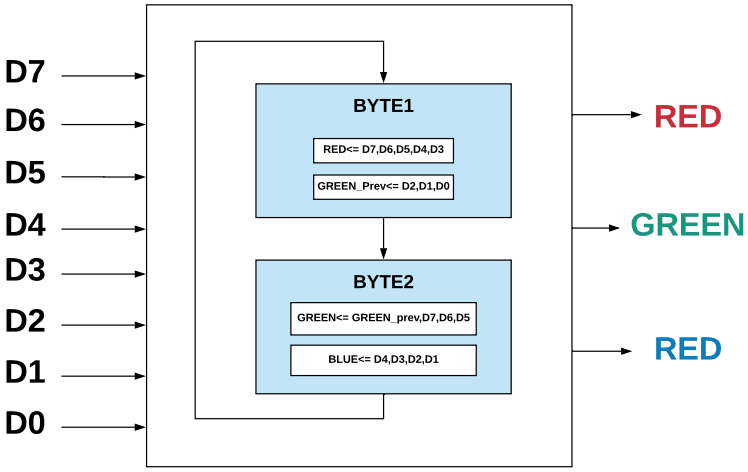

Figure 4.15: Bits assigments.

The final appearance of this module is represented in figure 4.16.

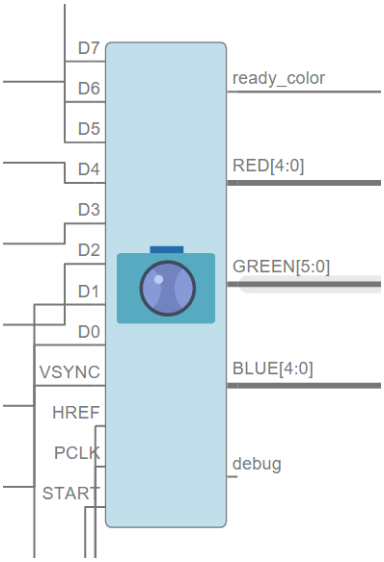

Figure 4.16: Bits assigments in IceStudio.

If it is observed in figure 4.16 in the output of this module the three color components will be obtained and a signal will be in charge to notify when the three color components are ready.

The next module has enough iformation to implement the whole showed algorithm in 4.2.

4.3 Control Design

Respect to the control implementation on quadcopter board, it is presented only the theorical basis in order to improves it in following works.

To understand this behaviour, the figure 4.17 is represented.

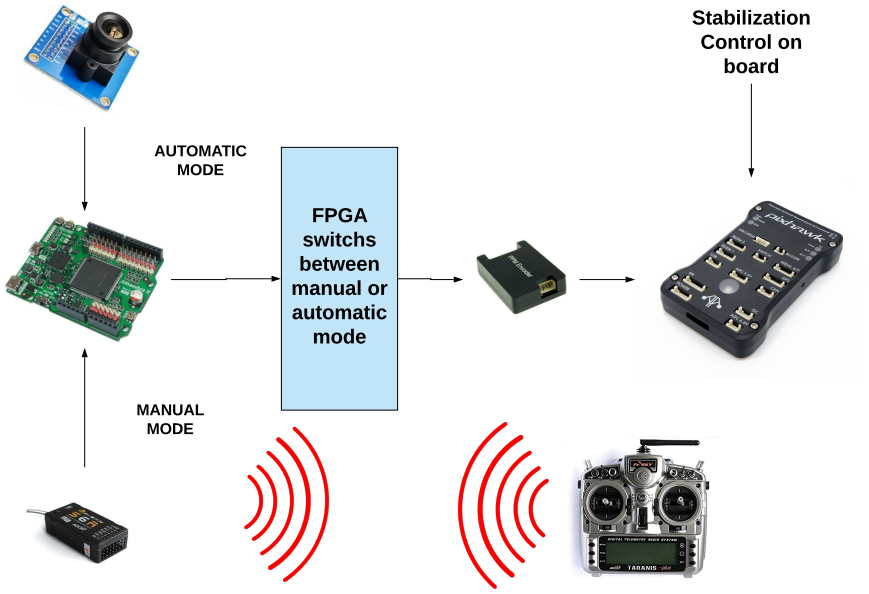

Figure 4.17: Diagram of control Implementation.

The stabilization control will take part of the inter processes from PixHawk board. This has a PPM signal connected corresponding to the reception of the signal coming from a frequency transmitter, which sends the PWM signal from the Yaw, Pitch and Roll.

The objetive is to modify this signal wich provoke the quadcopter movement in relation with the output recognition object protocol. The FPGA will be in the middle between signal receptor (FrSky) and PixHawk in order to allow the both behaviour.

Both behaviour can be:

- Automatic mode: The quad-copter will move in relation to the object, having as a principal objective it is positioned at the center of the vision area. To maximize this tracing a PID control an be used which is developed in this project. Quadcopter will move in relation to the position of the object being detected.

- Manual mode: Quadcopter will move in relation to the signal received from the radio transmitter.

Chapter 5

Conclusions and future work

With this chapter the project documentation is closed. Throughout this report, design and implementation from two robotic systems have been exposed by using open FPGAs. The key points developed in this project can be summarized ahead:

- Learning about educational robotic and open projects
- Design and implementation of the mechanical structure of the self-balance robot through the usage of SolidWorks and 3D printing
- Design and implementation of a PCB shield for IceZum Alhambra II by using Altium Designer
- Hardware implementation using PID controller Verilog and all other systems that compose the self-balance robot
- Design and implementation of the quad-copter ball recognition protocol
- Low-cost camera reading through FPGA and Verilog
- Design of the quad-copter control system

Through this project, there have been a lot of learned aspects that has nothing specifically to do with the work done. Between them the problem resolution capability and autonomous work can stand out, through which directors have served as a guide, bringing necessary tools for different problems resolution.

The objectives settled at the beginning of the work have been achieved, however, some of them remain as challenges that could not be covered or that have come out during the development of the same. The main challenges that came out for future work are presented below:

- Determine a more accurately the mathematical model of the balancing robot for an improvement in the mechanical structure.
- Integrate a PID controller (It is a PD controller until now) or use other control systems that could improve the stability.

- Improvement in the I2C protocol developed to allow some errors in transmission.
- Improvement in the FPGA-Microcontroller communication protocol to raise the speed and allow bidirectional communication.
- Improvement in the recognition of the red ball to allow a less ideal performance environment.
- Implementation of control on board in a quadcopter.

Special interest is shown, and is still working on it, about being able to take what has been learned to the classrooms, specially for little kids, allowing an early evolution in a technology whose future is evident and early.

Bibliography

[1] WikiPedia. Digital electonics. https://en.wikipedia.org/wiki/Digital_electronics, 2018. Accessed 20-03-2018.

[2] What is a fpga? http://fpgayxilinx.blogspot.com/2016/08/cuando-se-aborda-el-diseno-de-un.html, 2018. Accessed 23-05-2018.

[3] Wikipedia. Descripción hardware languaje. https://es.wikipedia.org/wiki/Lenguaje_de_descripci%C3%B3n_de_hardware, 2018. Accessed 01-10-2018.

[4] N. Eteokleous and D. Ktoridou. Educational robotics as learning tools within the teaching and learning practice. In *2014 IEEE Global Engineering Education Conference (EDUCON)*, pages 1055–1058, April 2014.

[5] Altium designer. https://www.altium.com/altium-designer/.

[6] Solidworks. https://www.solidworks.com/es.

[7] Github. https://github.com/.

[8] Juan Ordoñez. Repositorio github. https://github.com/RoboticsURJC-students/2017-tfg-juan-ordonez.

[9] Appear. https://appear.in/.

[10] Xilinx. https://www.xilinx.com/, 2018. Accessed 01-10-2018.

[11] C. Dawson, S. K. Pattanam, and D. Roberts. The verilog procedural interface for the verilog hardware description language. In *Proceedings. IEEE International Verilog HDL Conference*, pages 17–23, Feb 1996.

[12] Lattice Semiconductor. Fpga lattice. https://www.latticesemi.com/, 2018.

[13] A. Romanov, M. Romanov, and A. Kharchenko. Fpga-based control system reconfiguration using open source software. In *2017 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus)*, pages 976–981, Feb 2017.

[14] Tarjeta icezum alhambra ii. https://alhambrabits.com/alhambra/.

[15] Icestudio. https://icestudio.readthedocs.io/en/latest/.

[16] Google groups fpga wars. https://groups.google.com/forum/#!forum/fpga-wars-explorando-el-lado-libre.

[17] Rajesh Nema, Rajeev Thakur, and Ruchi Gupta. Design & implementation of pid controller based on fpga with pwm modulator.

[18] Wikipedia. Inverted pendulum. https://en.wikipedia.org/wiki/Inverted_pendulum, 2018. Accessed 01-10-2018.

[19] L. H. Yu and F. Jian. An inverted pendulum fuzzy controller design and simulation. In *2014 International Symposium on Computer, Consumer and Control*, pages 557–559, June 2014.

[20] L. M. Lizarraga Orozco, G. Ronquillo Lomeli, J. G. Rios Moreno, and M. Trejo Perea. Identification inverted pendulum system using multilayer and polynomial neural networks. *IEEE Latin America Transactions*, 13(5):1569–1576, May 2015.

[21] Juan Ordoñez. Demostrative video self-balancing robot. https://youtu.be/d_1bnjbpQks, 2018.

[22] Juan Ordoñez. Demostrative video self-balancing robot. https://youtu.be/dQg8NQP7CfQ, 2018.

[23] Juan Ordoñez. Demostrative video self-balancing robot. https://youtu.be/mLyxewOVGug, 2018.

[24] Juan Ordoñez. Demostrative video of printing 3d structure. https://youtu.be/rKoIdgaJU2k, 2018.

[25] Juan Ordoñez. Demostrative video vga module. https://www.youtube.com/watch?v=b3I2MBhlZ9g, 2018.

[26] Juan Ordoñez. Demostrative video brushless motor. https://youtu.be/OFD7vlm7f3A, 2018.

[27] Omnivision. Ov7670 vga module. https://www.voti.nl/docs/OV7670.pdf, 2018.

[28] NXP. Sccb interface information. https://www.nxp.com/docs/en/supporting-information/flexibleCameraInterfaceSolution_1.pdf, 2018.

[29] Driver para motor mc33926. https://www.pololu.com/product/1213.

[30] Julián Caro, Antonio Barrientos, and Enric Mayas. Hybrid bio-inspired architecture for walking robots through central pattern generators using open source fpgas. In *Intelligent Robots and Systems, 2018.(IROS 2018). 2018 IEEE/RSJ International Conference on*, volume -, pages –. IEEE, 2018.

[31] Freescale Semiconductor. Technical data mc33926. https://www.pololu.com/product/1213, 2014.

[32] K. Pathak, J. Franch, and S. K. Agrawal. Velocity and position control of a wheeled inverted pendulum by partial feedback linearization. *IEEE Transactions on Robotics*, 21(3):505–513, June 2005.

[33] Github. `https://github.com/`.

[34] Wikipedia. Sensor. `https://es.wikipedia.org/wiki/Sensor`, 2018. Accessed 01-10-201.