Universidad Autónoma de Nuevo León

Facultad de Ingeniería Mecánica y Eléctrica

Subdirección de Estudios de Posgrado

Sistema de Guía, Navegación y Retroalimentación Para un Misil Tierra-Aire a Través de Sistemas Híbridos

por

Pedro Antonio Limón Díaz

como requisito parcial para obtener el grado de

maestría en ingeniería aueronáutica

con orientación en Dinámica de Vuelo

Junio 2016

Universidad Autónoma de Nuevo León

Facultad de Ingeniería Mecánica y Eléctrica

Subdirección de Estudios de Posgrado

Sistema de Guía, Navegación y Retroalimentación Para un Misil Tierra-Aire a Través de Sistemas Híbridos

por

Pedro Antonio Limón Díaz

como requisito parcial para obtener el grado de

maestría en ingeniería aueronáutica
con orientación en Dinámica de Vuelo

Junio 2016

Universidad Autónoma de Nuevo León

Facultad de Ingeniería Mecánica y Eléctrica

Subdirección de Estudios de Posgrado

The members of the Thesis Committee recommend that the Thesis "Sistema de Guía, Navegación y Retroalimentación Para un Misil Tierra-Aire a Través de Sistemas Híbridos", performed by the student Pedro Antonio Limón Díaz, with registration number 1707080, to be accepted for its defense as a partial requirement to obtain the degree of Maestría en Ingeniería Aueronáutica con orientación en Dinámica de Vuelo.

The Thesis Committee

[illegible]

Mario Alberto García Ramírez, PhD

Asesor

[illegible]

Diego Francisco Ledezma García, PhD

Asesor

[illegible]

Dr. Romeo de Jesús Selvas Aguilar

Sinodal

[illegible]

Dr. Ángel Enrique Sánchez Colín

Sinodal

Vo. Bo.

Dr. Simón Martínez Martínez

Subdirección de Estudios de Posgrado

San Nicolás de los Garza, Nuevo León, junio 2016

UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN

Acta de Examen de Maestría

82203
001707080

REGISTRO
Y
TITULACIÓN

Acta Núm. 2789

En la Ciudad de Monterrey, capital del Estado de Nuevo León, al día 15 del mes de JULIO del año 2016 siendo las 11:00 horas, reunidos en las instalaciones de la FACULTAD DE INGENIERÍA MECÁNICA Y ELÉCTRICA de la Universidad Autónoma de Nuevo León los (as) señores (itas) DR. MARIO A. GARCIA RAMIREZ, DR. MARTIN CASTILLO MORALES Y DR. ROMEO DE JESUS SELVAS AGUILAR, catedráticos (as) de la misma, quienes fueron designados por las autoridades de la División de Estudios de Posgrado de la Facultad para integrar el jurado calificador del Examen de Grado de **PEDRO ANTONIO LIMON DIAZ**, quien cursó y aprobó todas las materias de la **MAESTRÍA EN INGENIERÍA AERONÁUTICA CON ORIENTACIÓN EN DINÁMICA DE VUELO,** tal como lo disponen la Ley Orgánica de la Universidad Autónoma de Nuevo León publicada en el Periódico Oficial el siete de junio de mil novecientos setenta y uno, el Reglamento General del Sistema de Posgrado y el Reglamento Interno de la Facultad. Se procedió a iniciar dicho examen sometiendo al sustentante a los interrogatorios de rigor para los diversos casos teóricos y posteriormente se llevaron a efecto las pruebas prácticas sobre los diferentes aspectos de los estudios. Una vez concluido este último acto, el jurado pasó a deliberar sobre las diversas pruebas a las que fue sometido el sustentante y después de discutidos todos los aspectos del examen, el jurado resolvió comunicar a **PEDRO ANTONIO LIMON DIAZ** que fue APROBADO en su Examen de Grado. Con lo anterior se dio por terminado el acto, y en cumplimiento de lo dispuesto por los preceptos legales y reglamentarios, firman la presente acta los señores sinodales, ante la presencia del Secretario del Jurado, que da fe.

PRESIDENTE	SECRETARIO
[illegible]	[illegible]
DR. MARIO A. GARCIA RAMIREZ	DR. MARTIN CASTILLO MORALES

VOCAL

[illegible]

DR. ROMEO DE JESUS SELVAS AGUILAR

El suscrito, Director de la FACULTAD DE INGENIERÍA MECÁNICA Y ELÉCTRICA, CERTIFICA que las firmas que aparecen en la presente acta son auténticas y las mismas que utilizan los C.C. profesores mencionados en ella.

FACULTAD DE INGENIERÍA MECÁNICA Y ELÉCTRICA
U.A.N.L.
DIRECCION

Monterrey, N.L., a 08 de AGOSTO del año 2016

[illegible]

DR. JAIME A. CASTILLO ELIZONDO

El C. Secretario General de la Universidad Autónoma de Nuevo León CERTIFICA que la firma del C. Director de la FACULTAD DE INGENIERÍA MECÁNICA Y ELÉCTRICA que aparece en la presente acta, es auténtica.

SECRETARÍA GENERAL
DEPARTAMENTO ESCOLAR Y DE ARCHIVO
UANL
DEyA

UANLT
005079

gilda

El recibir de conformidad este documento académico, compromete al interesado a dar el uso legal y correcto del mismo. Quien altere cualquiera de las partes que lo conforman, invalida inmediatamente su autenticidad, haciéndose acreedor a la aplicación de las sanciones previstas en las leyes y reglamentos de la UANL; independientemente de los efectos legales que procedan.

ORIGINAL PROFESIONES

07-11

This thesis is dedicated to all those whom offered their support in one way or another, directly or indirectly and that gave me something that I could not be able to compensate: confidence, knowledge, opportunities, time, patience and dedication.

Irreplaceable things once given away you have them forever, but once lost you can't get them back.

Contents

List of Figures

List of Tables

Acknowledgements

I deeply want to thank to my family without their support, understanding, collaboration and inspiration, it would have been almost impossible to complete this achievement. To my parents, Antonieta and Pedro, for their example of perceverance, tenacity in solving problems and humility at the moment of being successful. My sister Andrea for her observations and patience and my sister Ana for her optimism in combination with a bit of realism and intelligence to say the correct thing at the right time.

I must thank the graduate scholarship program of the *Consejo Nacional de Ciencia y Tecnología*, CONACYT, for giving me a scholarship that supported me throughout the period of my Degree. Likewise, I thank the academic and administrative staff and workers of *Centro de Investigación e Innovación en Ingeniería Aeronáutica*, CIIIA, which belongs to *Facultad de Ingeniería Mecánica y Eléctrica*, FIME, of *Universidad Autónoma de Nuevo León*, UANL, for giving me the opportunity and support needed to complete my studies.

My sincere gratitude to Mario Alberto García Ramírez, PhD for accepting me to do the thesis under his direction and tutelage. His support and guidance to connect me with the right people to comprehend, understand and complete this thesis project. The things that marked me the most during this timeslot were working out of my comfort zone, his rigor and freedom to work with my ideas Thank you very much and I hope to keep in touch with you after the end.

I am specially grateful to Dr. Mario Alberto Mendoza Bárcenas, whom in my times of crisis held out his hand for me. Not only did he accepted me as a student, but received me with open arms into his private circle. For the ears pull that he gave me several times which deeply influenced me to becoming a better student and person. His collaboration was of great value to me and a significant influence at the time of developing and improving my criterion. I am also grateful for his attentive and quick answers to my questions and concerns that arose during the time I worked with him. All this is reflected in the results. Thank you so much!

To my master colleagues whom joined me for a great part of this adventure. Wherever they are, all of them deserve the best words. Without you guys, this way would have been overwhelming and lonely. Each one of you taught me things that I will always keep. Daniel, for teaching me to say hello whenever I get to any place, whoever he is. Pao, for making me see that there is a life beyond work and study. Mary, for not forgetting the colleagues with whom you live with and occasionally make them notice that you remember them. Josué, for showing me that I should not tell everything to everyone. Arturo, for sharing your knowledge where I did not have the bases and you did. Orlando, for showing willingness wins over attitude, even when there is not time to complete tasks. Carlos, for sharing me your experience in machining and enlightening me about the decision I have to make in the near future.

Among my circle of friends, I must thank very attentively to the sisters De la Fuente Salas and Lulú, whom are my mainstay whenever it comes to put into words and letters one idea. Thank you for giving coherence to my sentences and the sense that I want to express them.

To all the friends who encouraged me during the course of the thesis, Marie, Josselyne, José, Sammy, Iván and Gus. Also thanks to those whom gave me their direct or indirect support and now are gone.

Summary

Pedro Antonio Limón Díaz.

Candidate for the degree of Maestría en Ingeniería Aueronáutica con orientación en Dinámica de Vuelo.

Universidad Autónoma de Nuevo León.

Facultad de Ingeniería Mecánica y Eléctrica.

Title of the study: Sistema de Guía, Navegación y Retroalimentación Para un Misil Tierra-Aire a Través de Sistemas Híbridos.

Number of pages: 91.

Objectives and study method: This thesis aims to design and to develop a platform that is capable to guide, control a system orientation as well as the position for a missile. The system designed can be also extrapolated to an unmanned aerial vehicle. The platform designed will help to generate tools to define the orientation and position of a vehicle by acquiring data from an inertial measurement unit in real time.

To confirm the platform operation, a co-simulation technique Hardware-In-The-Loop is used. By using this technique, it is possible to divide the platform into the following modules:

1) Sensor module. This module is composed by an accelerometer, gyroscope, mag-

netometer, barometer, temperature sensor (that are co-integrated in the same board) and an IR camera.

2) Flight computer module. This module has a distributed system architecture. One side is composed by a commertial development FPGA board for data acquisition, a personal computer for processing and displaying data results.

Contributions and conclusions: The developing process is to perform a simulation that, separately, communicates the sensor module block with the flight computer module. Once confirmed the data acquisition and the processes, we will proceed to co-integrate all the algorithms into a main one and test the platform with all the instruments. Additionally, we add a communication module that is co-integrated with a radio frequency transceiver to reinforce a wireless communication.

A detailed process to implement the embedded system within the FPGA board is included. The specifications for each element used as the sensor, flight computer and communication modules with a comprehensive description of the system generation are shown. Finally, the results of the test, conclusions and recommendations for future work are addressed.

Signature of assesor: ______________________________

Mario Alberto García Ramírez, PhD

Chapter 1

General Introduction

1.1 Description of the problem

Nowadays, the development of high speed either algorithms and computers for military applications has been increased to a steady step for last decades. Modern materials, high speed processors and state-of-the-art maths have helped in the development of several unmannered aerial vehicles such as drones or missiles. The issue that we are addressing in this research work is based on the development of a low cost, reconfigurable flight computer to control an earth to air missile.

In order to develop the system, we were focused on foreseen for many development boards based on a set of reconfigurable devices on the market such as Raspberry Pi, Arduino, Beagle, FPGA, etc. These boards can be programmed to perform a specific task. It is possible to design and to develop embedded systems. These specific tasks can perform almost any function or control several sets of arrays. Within the whole range of sensors, we must define those that integrate an inertial measurement unit. In a similar way, the FPGA board selected have the capability and resources to support data management (acquisition and transmission) and real-time processing for the missile or remote probe requirements.

A set of sensors and a reconfigurable FPGA development board to design and to develop a platform based on a navigation and control system are required. The data acquisition should serve as a support tool to provide among other for complex task orientation to a missile. It must be confirmed by a wireless communication. This platform will have the ability to acquire and to store data from a set of sensors in order to process them.

1.2 Hypothesis

The resources that a FPGA development board has are suitable for data acquisition as well as for data processing. Those can be implemented within a flight computer in order to control quite a few different system such as those that can drive a missile. Thus, we will confirm that the selected FPGA board can work with a set of sensors by using a set of different communication protocols. As a consequence, we will have a robust platform that can be modified with either an array of sensors or boards with broader capabilities by following a similar methodology. The reason for using a FPGA board is because compared with the other programmable boards this can work in parallel on assigned tasks and can be programmed remotely.

Those whom are interested into replicate this work should have a basic background in inertial navigation, avionics and electronics. It is due to the designer must understand the use of new tools for the development of an embedded system focused on navigation and proper understanding of the results. While updating for the use of new devices, strengthening the foundation in electronic knowledge and not being close mind to a single tool for designing platforms based on embedded systems. It will allow us to develop far more complex systems by using the tools that are currently being used.

1.3 Methodology

1. Documentation, classification and delimitation of the missiles.
2. Study of the basis for inertial navigation.
3. Search and selection of possible electronic instrumentation to be used in the sensor module.
4. Literature review and configuration of the selected FPGA board.
5. With the resources of the FPGA board and devices, we will design the first scheme of instrumentation for data acquisition.
6. Simulation and testing of the devices separately.
7. Integrated platform test with the sensor module and the flight computer module.
8. Analysis of mathematical functions such as quaternions in order to implement it to find the position and location of the navigation platform.

Chapter 2

Background

2.1 Introduction

Every technological artifact has a story to tell. This chapter recapitulates the key events in which missiles have been introduced, features, classification, contribution to other areas as well as the country in which they were developed. On the other hand, we have the revolution of modern electronics that is performed by the transistor invention and the development of integrated circuits that are used in reconfigurable systems. Finally, some definitions that will be used throughout this research work are described to identify them.

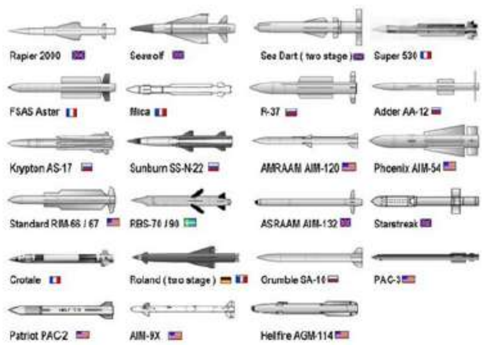
2.2 Missiles

2.2.1 Rockets and missiles

The key difference between a rocket and a missile should be stated. A rocket is a vehicle that is accelerated by a flow of particles (working fluid) in the opposite direction according to the second law of Newton [2]. So far, almost all rockets use gases, which are produced in the combustion chambers as the working fluid. Meanwhile, a missile is a rocket-powered vehicle and it is directed by a guidance system [2].

Missiles usually carry an explosive charge that can be directed towards a target. Figure 2.1 shows the difference between the emblematic rockets used for Apollo missions and the different types of surface-to-air and air-to-air missiles that are being manufactured by different nations.

(a)



(b)

Figure 2.1: Examples of different types of rockets and missiles. 2.1(a) Rockets used in space missions between 1960 and 1973. 2.1(b) Different air-to-air and surface-to-air missiles.

The discovery of a rocket takes us back to the middle of the Twelfth Century in China, when some fireworks accidentally turned out of its course and exploded when landing, leaving a crater on the ground. As a result of this accident, the idea of using these tools for military purposes was developed [2]. In the middle of the Thirteenth Century, the Arabs and Europeans were the first who used this technique as a military form. The first written reports that refer to rockets were found in the Fourteenth Century. So, most of the European civilizations knew and used rocket technology during the Fifteenth Century [2].

It was until the mid-Seventeenth Century that rockets were first mentioned in European and Chinese military manuals. In the late Eighteenth Century, the first conventional rocket appears in India and was used against the British. A few years later, the British, adapted the rocket created in India for European use [2].

The British military forces used this adaptation against the Danish, French and American settlers. Given the success, the rocket was modified for being used in rescue operations and whaling. In the mid-Nineteenth Century, a British marine developed the first spin-stabilized rocket. Then, the rockets that used gunpowder as fuel began to decrease as the “tube” artillery improved [2].

In the beginning of the Twentieth Century, without knowing the work of each other, a Soviet scientist, an American engineer and a German physicist mentioned almost at the same time the multistage rockets using propellant as fuel. The American engineer, Robert Goddard, built and flew the first propellant rocket. The success of this experiment, engineers, supported by government funds, invented stable and storable liquid propellants, better solid boosters and key technologies that would serve as a base for missiles. On the other hand, German engineers implement again rocket artillery for military use, that is how they revolutionized this technology with the creation of cruise missile V1 (Vergeltungswaffe 1) and the ballistic missile V2 (Vergeltungswaffe 2) [2].

At the end of the Word War II, some German engineers allied with the former Soviet Union Republic (USSR) and the United States (USA) began with the development, construction and testing of high-altitude research rockets. Sometime later, researchers armed the first operational combat aircraft and the first positive result for an air-to-surface missile was given during a war between Taiwan and China. At the same time, ex-USSR launched Sputnik I, the first artificial satellite to orbit the Earth and the first intermediate-range ballistic missile (IRBM) became operational and China acquired missile technology from ex-USSR [2].

Ex-USSR launched the first manned spacecraft. Meanwhile, USA made operational ballistic missiles of intercontinental range and tested flights for the rocket Saturn V [2]. At the time of Vietnam War, the first sustained use of surface-to-air, air-to-air and air-to-surface guided missiles in combat was confirmed. The first real achievement in the space race was when USA sent the first man to the moon in

July 1969. In the early 70's, USA developed the first independently targeted reentry vehicle (MIRV). ex-USSR exported FROG and Scuad missiles in order to verify results in actual combat and China made the first test flight of its first intercontinental ballistic missile.

The Apollo mission ended with a last landing on the Moon but for continuing the space race and made it a profitable business, USA launched the space shuttle program. Ex-USSR left the space race and its attempt to put a man on the Moon after several failed missions where spacecraft resulted in partial or completed destruction. By the end of the decade, the European Space Agency made its first successful flight of its space vehicle Ariane. While the first orbital flight tests for the shuttle program were made, small conflicts started and with them, the impact of tactical missiles in the modern era of wars appeared.

The catastrophic combination of the shuttles Challenger and Columbia, in 1986 and 2003, respectively, set new goals in the research of how far the humans could reach. A natural satellite was no longer seen as a target but other planets. China, India, Pakistan, Iran and North Korea expanded their nuclear programs and with them, the research of the system that integrated a missile [2]. These nations have continuously tested missiles in small conflicts with the continuous improvement of the technological advances they represent.

2.2.2 Classification of missiles

The missiles can be classified by the place of release, type, applications and guidance system. The classification of the launch site can be an air-to-air, air-to-surface, surface-to-air or surface-to-surface type. The kind of artifact comprises the division between ballistic and cruise missiles. The use can be dissuasive (preventive) or persuasive (on target). The guidance system is divided into programmable passive, semi-active, active and remote control [1, 2].

Similarly, the cruise missiles types are sub-classified according to their speed and these can be subsonic, transonic, supersonic or hypersonic. The ballistic missiles are sub-categorized by their range such as short range (SRBM), intermediate-range (IRBM), intercontinental (ICBM), from a submarine launched (SLBM) and antiballistic (ABM) [1]. The missiles with programmable guidance system have an inertial guidance as a main system which can also be sub-classified. The passive guidance systems may use heat detecting or terrain comparison [1]. The semi-active guidance systems use a laser and a radar or a ground based guide system. Missiles with active guidance systems are guided by commands [1]. Finally, missiles with remote guidance systems are guided by radio or wires to set the direction [2].

We can observe on a schematic diagram in Figure 2.2 closely, the given classifications. It should be noted that there are missiles with a combination of categories and depending on the target, the systems are changed for performance [1, 2].

2.2.3 Sections of a missile

The main components in a missile are warhead, flight computer, flaps, rudders and engines [1]. The warhead is the designed space containing the explosive material [2]. The flight computer is the brain of the missile, it contains the inertial navigation system and the electronic assistant to guide the missile to the target. The flaps

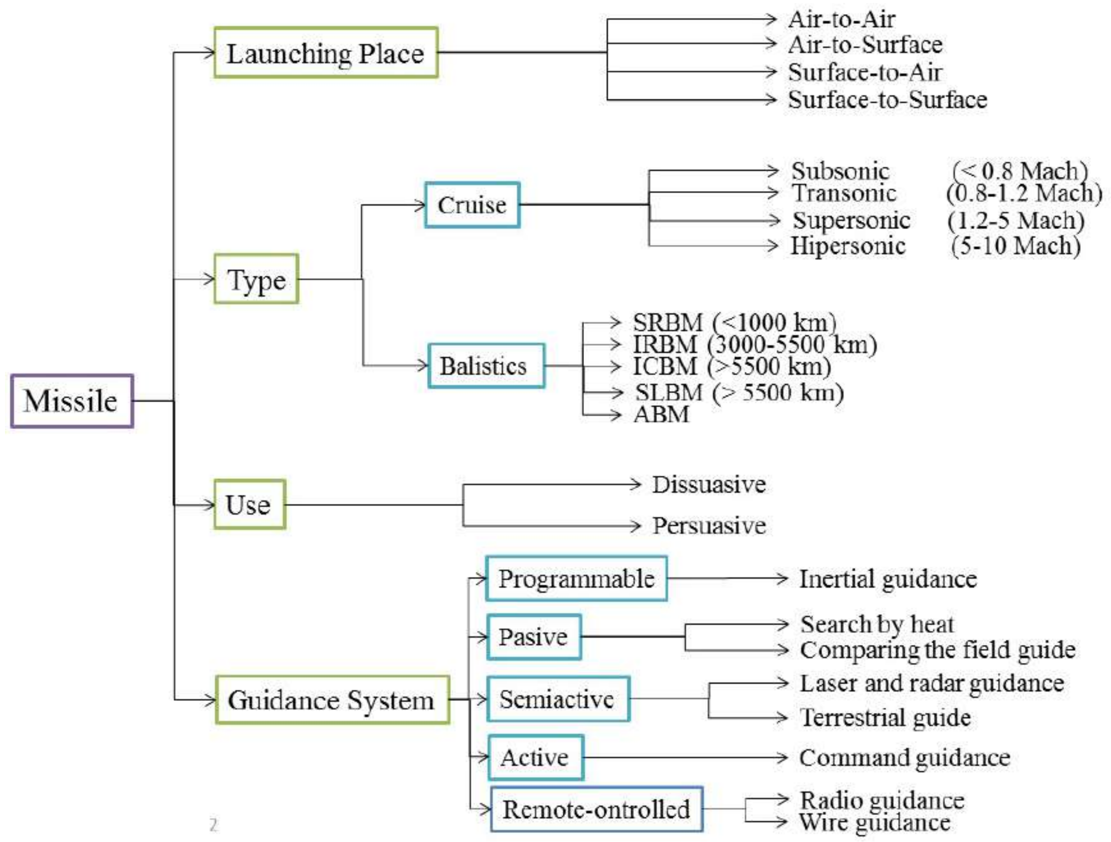


Figure 2.2: Missile classification and division.

and the rudders stabilize the missile during flight and redirect its trajectory. In the engine, the ignition stage, which gives take off power and speed during flight (that can be made of more than one stage), can be found [1].

This thesis is purely focused on the flight computer and to illustrate it. Figure 2.3 shows a supersonic, surface-to-air, passive IR homing missile named Chaparral, where the highlighted part is where the flight computer is located with its respective electronic, computer and sensor devices [1].

2.3 Integrated Circuits

To understand how important a reconfigurable board is, we must describe the road that integrated circuits had passed through years until today and demonstrate its

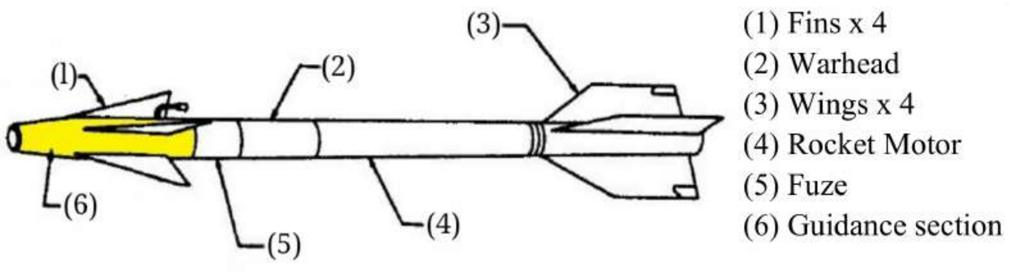


Figure 2.3: Parts of a Chaparral missile [1].

importance. An integrated circuit (IC), also known as microchips, are tiny chips in which we can find a large number of electronic devices (micro size) that interact with each other, mainly diodes and transistors. In special cases, passive components are added (such as resistors and capacitors). ICs are manufactured with a material called semiconductor. The first effect of a semiconductor was registered by Michel Faraday. He discovered that the rise of the temperature effect in the silver sulfide increased electrical conductivity. In other words, the temperature rises in the majority of the semiconductors and it raises the density of charge carriers contained in it, improving its conductivity. That is how the effect of the thermistors is explained [9].

Ferdinand Braund, a German scientist, discovered that in a crystal of lead sulfide with a thin metal wire tip, current flowed in one direction. This was the discovery of the rectification effect in the interface between metal and some crystalline materials. This device found an application in the beginning of the 1900s in radios and the set of signal detector will be found. The colloquial name of the detector is “cat’s-whisker”. Diodes are electronic devices that perform the rectification. William J. Hammer added another electrode a hot light bulb filament then, John Fleming used this base to create the “oscillation valve” unidirectional that converts an AC current of the radio signal into DC current. The rectifier contact point of Braund performs the same function by using semiconductors, without thermionic properties [27].

The engineer Greenleaf W. Pickard tested hundreds of samples of mineral to evaluate their properties for rectification [27]. Silicon crystals produced the best results [9]. Although crystal rectifiers allow radios at the time to operate with a little consuption of power source, predictable operation of vacuum tubes replaced them in most applications and peaked during the World War II because of its ability to operate in microwave frequencies. Vacuum tubes are composed of three elements: a thermionic cathode, a grid of parallel wires and a plate or anode. Figure 2.4 illustrates how these elements were incorporated before being introduced into the vacuum tube made of glass or metal. Lee De Forest was the inventor of the vacuum tube but it had a huge superficial resemblance to an invention of John Ambrose Fleming, motive enough to deny the Novel Prize to De Forest. However, the invention of Fleming could not amplify signals [7]. The American Telephone and Telegraph Company (AT&T) faced patent expiration that Alexander Graham Bell invented, as a solution Vail propused the transcontinental communications. Thus AT&T acquired the patent of the triode vacuum tube.

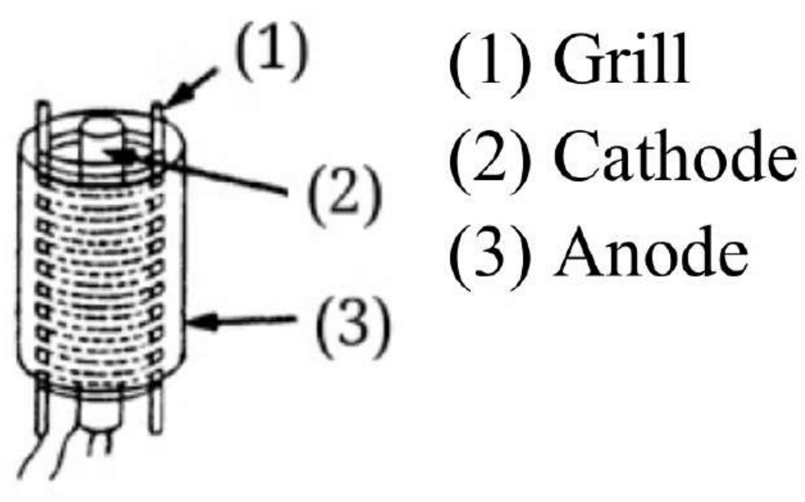
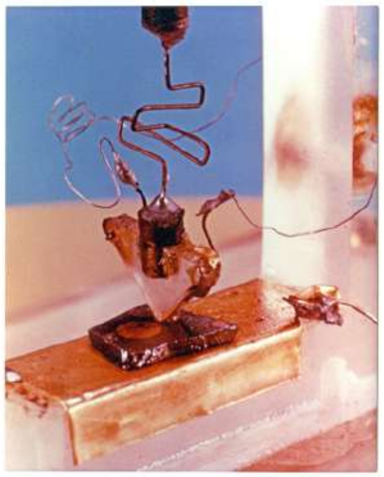


Figure 2.4: Triode form and its elements.

Russell Ohl began studying the use of silicon rectifiers as radar detectors [27]. The result was that the amount of pure silicon is directly related to the detection capability. Ohl also noted that the different parts of the crystal have opposite electrical effects when it was tested with a “cat’s-whisker” probe [27]. Ohl and his colleague Jack Scaff found a barrier into the plate which marked a split in the silicon with two different types of impurities. They called these regions n-type (negative) and p-type (positive). The surface of these connections is called “pn junction”. Nowadays, the pn junction is the most common way to rectify signals in the electronics industry and it is the main element in the design of electronic devices. With this, Ohl discovered the photovoltaic effect that feeds solar cells today.

During World War II, the semiconductor technology gave a breakthrough. Radar receivers required solid state rectifiers to detect and convert microwave signals at higher frequencies, a great advantage over vacuum tubes. Researches during this event yielded the result that silicon and germanium would be the semiconductor elements dominant in the area. At the end of WWII, Mervin Kelly, director of the research area at Bell Telephone Laboratories (belonging to AT&T), created a department dedicated solely to this science. Bill Shockley was put in charge of the department and hired Walter Brattain and John Bardeen. Its creation, the transistor, is considered the greatest invention for humanity since the wheel. Figure 2.5(a) shows the first point-contact transistor manufactured by Bardeen and Brattain. Advances made the transistor smaller, useful and practical as the Bell Labs “Type A” transistor 2.5(b).


(a)

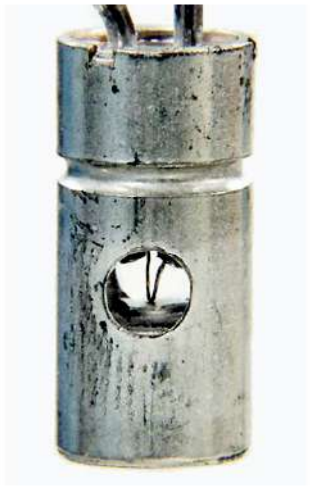

(b)

Figure 2.5: First transistors produced as a result of the successful test results. 2.5(a) First point contact transistor. 2.5(b) “Type A” transistor. [33]

The difference, at functional level, between the vacuum tubes and the transistor is that the first are based on the ability of electrons to travel freely in the vacuum with any energy and compared with vacuum tubes, transistors had a high initial cost. Transistors are based on free movement of electrons through semiconductor. The reasoning which Shockley gave on the junction transistor was thanks to the fortuitous discovery of Ohl in 1940. Shockley discovered the “minority carrier injection” a critical phenomenon for the operation of the junction transistor, a sandwich of three layers of n-type and p-type semiconductor separated by pn junctions. This concept serves in all “bipolar” transistors today [14].

Bell Labs organized the first transistor symposium which lasted nine days. This symposium was attended by over 100 representatives from 40 companies among them General Electric, RCA, Texas Instruments (TI) and Sony [27]. Bell Labs built a computer completely made by point-contact transistor for the Air Force which consumed less than 100 Watts. Later, it was fitted on an airplane. Bell Labs did not follow the research with silicon but TI. Bell Labs announced the creation of the first solar cell. With a new technique to work the purity of the silicon transistors, TI launched a mass production sales and silicon transistors would be the preferred semiconductor material in industry.

After some differences, Shockley broke relations with Bell Labs and founded Silicon Valley. Sometime later some workers of Shockley resigned over differences in their ways of working and new companies emerged as Fairchild (company that innovated the transistor with the bases of Silicon Valley), Advanced Micro Devices (AMD) and Intel (companies for high-level devices). The manufacturing process “step-and-repeat” was created and today it is a process particularly used in semiconductors with connections less than 0.1 micrometers. Leo Esaki developed a new diode that its current decreased as the tension increased causing a “negative resistance”. He called it the tunnel diode [10].

As a consequence of such knowledge, Autonetics developed the first digital control system and guidance device for a ballistic missile. Figure 2.6 shows the inside of the flight computer put on the ballistic missile Minuteman I. The guidance system was divided in many sections, each connected to each other in order to complete the a common task. Each section worked with capacitors, resistences and an occasional TTL (mostly an NOR gate).

Figure 2.6: Interior of the first guidance system computer of a ballistic missile [33].

RCA created the first transistors fabricated within a chip. Later, the shift register and logic gates CI were developed [11]. The most important innovation in the history of semiconductors industry came when a scientist in Fairchild introduced to the market the planar transistors. The method used for the manufacture of ICs is still in use until today. A new way to treat the wafer to protect the impurities improving electrical characteristics. TI adapted very quickly to the planar technique and by 1961 announced its first family of fully integrated circuits. Fairchild researchers presented progress of these chips and various miniaturizations that would help in a future to incorporate passive devices on the chip [11].

The aerospace industry and the Air Force were the first to spare no expense in smaller, faster and low-power transistors because the low-cost transistors were relatively slow. The guidance computer of the Apollo missions was the first most significant project for digital IC. It was a computer developed by the MIT and built by Raytheon, each system used 4000 IC (NOR gates). The first ICs into orbit were aboard the interplanetary probe surveillance of the NASA and were created by TI as its first planar family of IC. Later, TI won a contract with Autonetics Division of North American Aviation to design 22 tailored circuits for the guidance system of Minuteman II missiles [11, 27].

RCA laboratories were CMOS pioneers and together with the manufacturing Somerville developed the first IC of these transistors. Its first orientation was for the aerospace industry before moving on to commercial applications. Because of its smaller size and lower power consumption than bipolar transistors, almost all microchips produced today use MOSFET technology. CMOS provide the best solution to manage density issues resulting for packing thousands of transistors on an IC. The first mass production lines of IC were for the integrated version of discrete diode transistor logic (DTL). It is not until the late 60s that Transistor-Transistor Logic (TTL) takes popularity in the logic configuration and became the favorite for the following decades [12, 26].

TI, inspired by the work of Logo and successful military high profile design (the Phoenix missile), developed the TTL SN7400 family plastic packaged and low cost for industrial customers. Hybrid mounted circuits one or more passive components on an IC and ceramic substrates are interconnected by wires or conductive traces [15, 26]. Circuits with specific functions that could not be co-integrated by technical or economic reasons use monolithic IC and continued to be classified as hybrids. IBM changed the whole concept of computing to develop Solid Logic Technology (SLT) [16]. Two approaches to the production of large volumes of personalized designs, gate arrays and standard cells were developed. They are commonly known as application specific integrated circuits (ASIC). ASCI was born as a solution from IBM and TI to the problems generated by computers of the Air Force [15].

TI introduced the TTL 74S family using Shottky diodes to manage the change of state of a gate in an average time of 3 ns. Its brothers, the 74LS family, replaced the 7400 family offering the same speed but removing some of the power consumption [16]. These IC are used in universities to date. Radiation Inc. introduced to market the first programmable ROM (PROM) 512 bits in 1970. These devices allow designers to make changes in laboratories. Intel introduced its erasable PROM (EPROM). The 2048-bit EPROM could be reused many times. Designers of IC had the idea of integrating the central processing unit (CPU) functions of a computer in a handful of chips called micro processing unit (MPU). The idea was to integrate 8-bit ALUs. The first of its kind was the MP944 assigned to mainframe data from an F-14A and was manufactured by AMI [16].

A System-On-Chip (SOC) is an IC incorporating all electronic components, including analog devices and circuitry required to implement a system on a single chip [5]. The first SOC was in a digital wristwatch, everything was communicated by a CMOS IC Intel 5810. Many vendors chose to change the ASIC for SOC in embedded systems allowing handheld video games, communications data, computer peripherals, etc. Some logic semiconductor designers had the idea that smaller and faster PROMs could create products intended specifically for reprogrammable logic. Based on this idea and on migration induced by avalanche of PROMs, designer Bill Sievers created the first FPLA (Field Programmable Logic Array) that would be known to history as the first programmable logic array (PLA) could be electrically programmed in the field. Architecture PLA is more rational when making connections.

The range of applications extended when AMD used the CMOS technology for lower power consumption, adapting reprogrammable EPROM CMOS based devices compatible with computers and schematic support I/O designed by Altera. Other companies such as Xilinx, Actel and QuickLogic were born and they are the first companies to enter the market Field Programmable Gate Array (FPGA). These types of devices are known as Programmable Logic Devices (PLD). FPGA architecture allows applications to run higher degree of difficulty. Over the years other reprogrammable low-cost devices in an IC were added and became popular by offering resources, including Arduino, Raspberry Pi and Beagle Board. For their capabilities, easy programming, high degree of control, low power consumption, easy replacement and personalized approach to digital logic we can find these PLDs in production lines of assembly plants, supercomputers, satellites, links to microprocessors and more [8].

2.4 Nomenclature

2.4.1 Guidance system

From the control's point of view, the guidance system has the function to find the appropriate compensation network to be placed in series with the plant in order to achieve an interception. From a material point of view, it is an electrical and/or mechanical system that drives the vehicle. For missile specifications, the system varies according to the desired task, the size of the missile and the available technology [1].

The purpose is to determine and to decide the dynamics of the flight path (physical action) based on knowledge of the guiding capacity of the vehicle and chosen objectives to achieve for a specific target efficiently. The system main function is to mathematically co-integrate the separate functions of the hardware navigation and flight control system. To facilitate the reach of the target, the feedback control principles are used into the guidance system for better maneuvering. The system is designed to use a initially stabilized tracking program (e.g. seeker).

2.4.2 Navigation system

A navigation system is a set of components that provide position, speed and altitude for a guided vehicle with respect to the target. The information provided helps to establish the flight path according to a law of direction, in order to achieve the objective of the flight mission [1].

The guidance system in a missile includes sensors, a flight computer and a set of control components. The flight main goal is to reach a target. Accelerometers and gyroscopes are the sensors that are configured as an inertial system and feed data during the flight.

2.4.3 Embedded system

An embedded system is a combination of hardware, software and in some cases the inclusion of mechanical components designed to perform a specific function. Embedded systems commonly have features such as low manufacturing cost, effective use of components, improved performance of the task and a low power consumption [5]. Figure 2.7 shows the components distribution and general interconnection of an embedded system. Those are [25]:

1. Processor (μP, μC, DSP, FPGA)
2. Memory (InstMem, DataMem)
3. Buses (communication between components)
4. Peripherials (standar devices or customized interfaces)

2.4.4 FPGA

Field Programmable Gate Array (FPGA) is an array of logic blocks surrounded by input and output blocks (some with a general purpose and others with a specific purpose) programmable and connected through reprogrammable interconnections [5]. Among the advantages of an FPGA board, we can mention that the IC is reprogrammable, the designs are fast, high gate density, run any logic function,

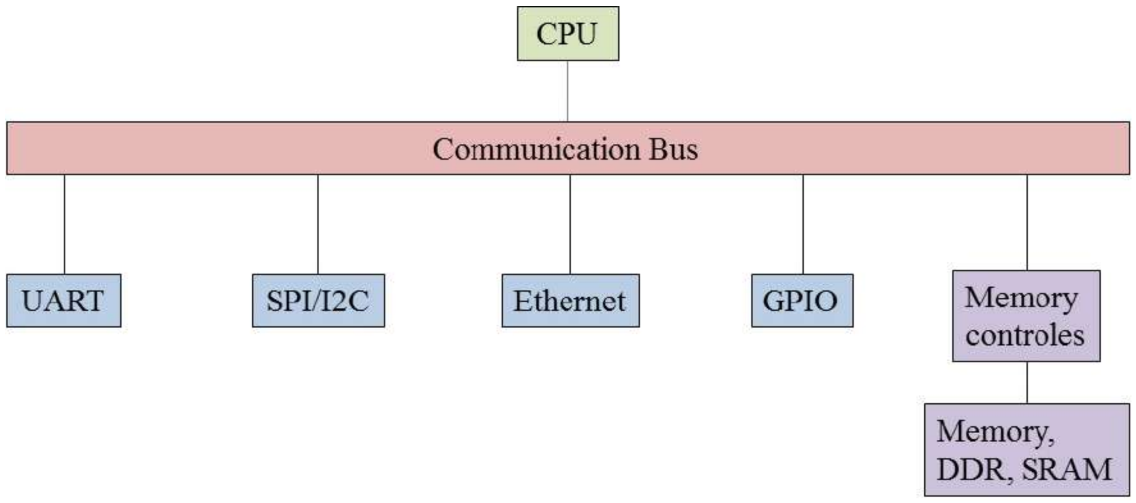


Figure 2.7: Distribution and interconnections of the components of a general embedded system [25].

process in parallel, have the ability to update their functionality and have a variety of communication languages with several softwares [5].

The high cost and low availability on the market are the main disadvantages of the FPGA boards. The most technical disadvantage is that they have a slow setting time. Table 2.1 shows a comparison of the main features of PLDs basic model for each brand.

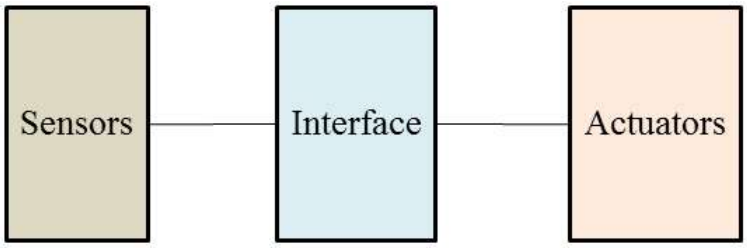
2.4.5 Hardware-In-The-Loop simulation

The Hardware-In-The-Loop (HIL) simulation is a technique in which a system, partially or completely, is subjected to a larger system that simulates an environment with controlled and measurable variables. HIL simulation offers advantages that other methods of analysis and testing do not provide. Among these advantages we can apoint the opportunity to investigate the same machine thoroughly on multiple occasions, the test conditions are faithful to nature and it reduces the cost and increases the risk to find hidden flaws in the hardware before causing damage to the functional system.

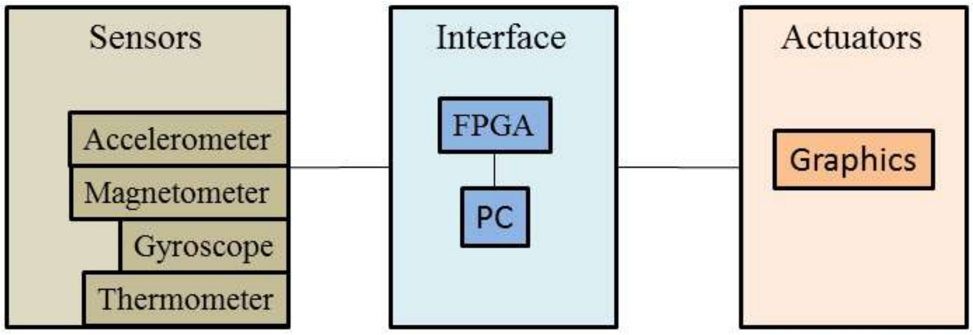
	Arduino	Raspberri Pi	Beagle Board	FPGA
Model	UNO	Model B	Rev. C4	Spartan 3-E
Operative System	×	Linux, RISC OS	Android, Linux, Windows CE, RISC OC	Linux
Programming Language	C, C++	Python, C, Basic	Python, C, etc.	VHDL, C, C++, Vivado
Sequential	✓	×	✓	×
Architecture	8 bits	32 bits	32 bits	32 bits
Processor	ATMEGA 328	BCM2835 (ARM)	TI DM3730 (ARM)	Atmel AT90USB2
Clock Frequency	16 MHz	700 MHz	720 MHz	25/50/100 MHz
RAM	2 kB	256 MB	256 MB	256 MB
ROM	32 kB	SD	256 MB flash	256 MB
ADC	6	8	Internal	Internal
Security	×	×	✓	✓
Cost	220 MXN	980 MXN	1900 MXN	2100 MXN

Table 2.1: Comparative characteristics of the comertial PLDs.

In some cases, the focus of HIL simulation adds the transient response of the system even without be entirely built. This technique has the potential to expose the full extent of the interactions to be gained in the final system. A HIL simulation consists of three main parts: a piece of system hardware to simulate, actuators that respond to disturbances and an interface that provides a part of the system with the actuators. Figure 2.8(a) shows the schematic design of the distribution prior mentioned elsewhere, which is a general scheme for any HIL simulation. On the other hand, Figure 2.8(b) is the specific scheme of HIL simulation applied to this



(a)



(b)

Figure 2.8: Schemes of the HIL simulation. 2.8(a) Genereal scheme of a HIL simulation. 2.8(b) Scheme of the HIL simulation to use.

work.

Chapter 3

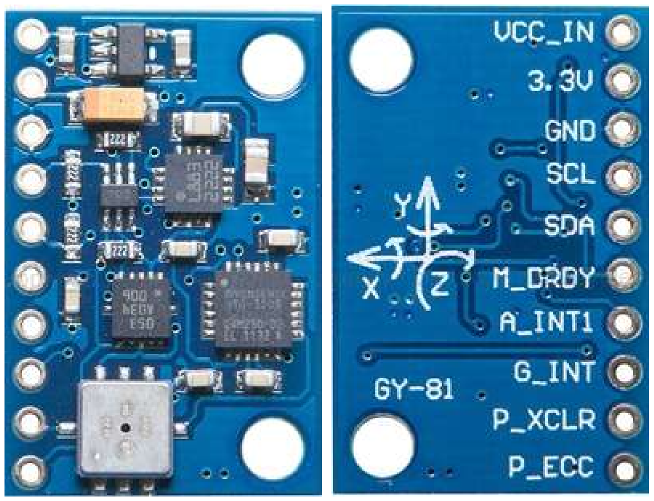
Development of the modules

3.1 Introduction

There are an endless number of electronic devices that can be used depending on the needs of the project to develop. There are high and low costs, high and low precision as well as boards with several integrated sensors or separated sensors, different protocol communication, etc. This chapter refers to the parts of the sensor module, the communications module and the computing module for integrated assembles the HIL simulation.

3.2 Sensor module

To assemble the sensor module, we have two options, find a card that already integrates the necessary sensors for the inertial measurement unit or obtain the sensors separately and make an arrangement to guide their axes in the same direction. We chose the card that has an integrated IMU (GY-81), shown in Figure 3.1.



(a) (b)

Figure 3.1: Flight control sensor GY-81 module. a) Front view. b) Back view.

3.2.1 Accelerometer

The accelerometer is a device that is widely used to measure the inclination and vibration, if the application is static. When the application is dynamic, the usefulness of the accelerometers is to determine the translational acceleration of the system. Accelerometers are based on one of three principles: piezo resistive, piezoelectric or capacitive transduction. GY-81 module contains a BMA180 accelerometer sensor and is of capacitive type (see Figure 3.2) [28]. The main advantages of capacitive accelerometers are easy mounting in a system and low temperature dependency. It is manufactured with microelectromechanical systems technology (MEMS). It is a digital sensor. Therefore, the output will be affected by an increase or a decrease in the numerical value. The accelerometer uses the communication protocol I^2C [28].

To measure the dynamic acceleration we need to stablish a relation between the sensor output that the sensor has when a known acceleration is aplied and a rule of three stablished; e. g. if one sensor axe is placed in the same direction of the gravity and it is know that the value delivered by the sensor will be the gravity value, the acceleration in the system can be calculated by the equation (3.1). It is

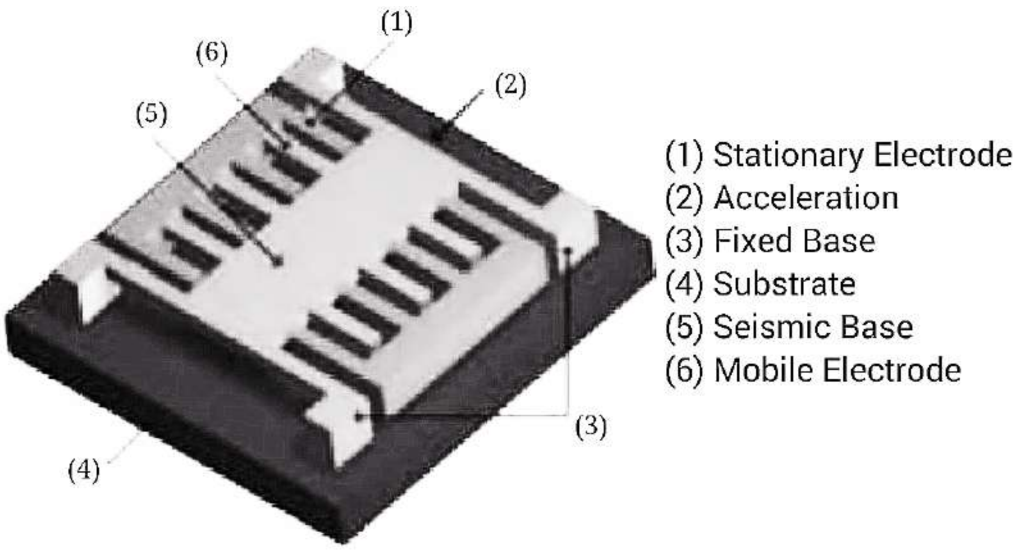


Figure 3.2: Capacitive accelerometer [17].

clear that both values can be either analog or digital. The main difficulty of this type of sensing is the noise due to the vibration of the system. It is considered that an inertial sensor with low cross sensitivity is a sensor with improved performance. It is helpful to have the units of the measurement in G at the time of the simulation, the data acquisition of the accelerometer facilitates the conversion to this measurement.

$$a = \frac{9.81 * X_{current}}{X_G} \tag{3.1}$$

Where:

$X_{current}$ Current value of an sensor axis.

X_G The value derivered by the sensor in the position of G.

3.2.2 Gyroscope

The gyroscope is an instrument that links the relative rotation with a voltage. The gyroscope in GY-81 module is ITG3205. Despite being excited by an inertial force, gyroscopes benefit from Coriolis forces of rotational movements. ITG3205 consists of a body that has symetry in its rotation, i.e., the mass of the sensor is moving at a velocity $\overrightarrow{V}$ and when a force moment is applied to the system, it rotates at an angular velocity $\overrightarrow{\Omega}$. The combination of linear and rotational movements generate the Coriolis force that is perpendicular to the initial linear movement shown in Figure 3.3. The gyroscope is a solid state sensor. The angles have symmetry with the BMA180 and are similarly to MEMS technology. ITG3205 architecture contains a pair of masses in reciprocating linear motion in tune [31]. This architecture has the ability to integrate into systems with first variable capacitance transduction. The ITG3205 uses a protocol I^2C to communicate with the microcontroller [31].

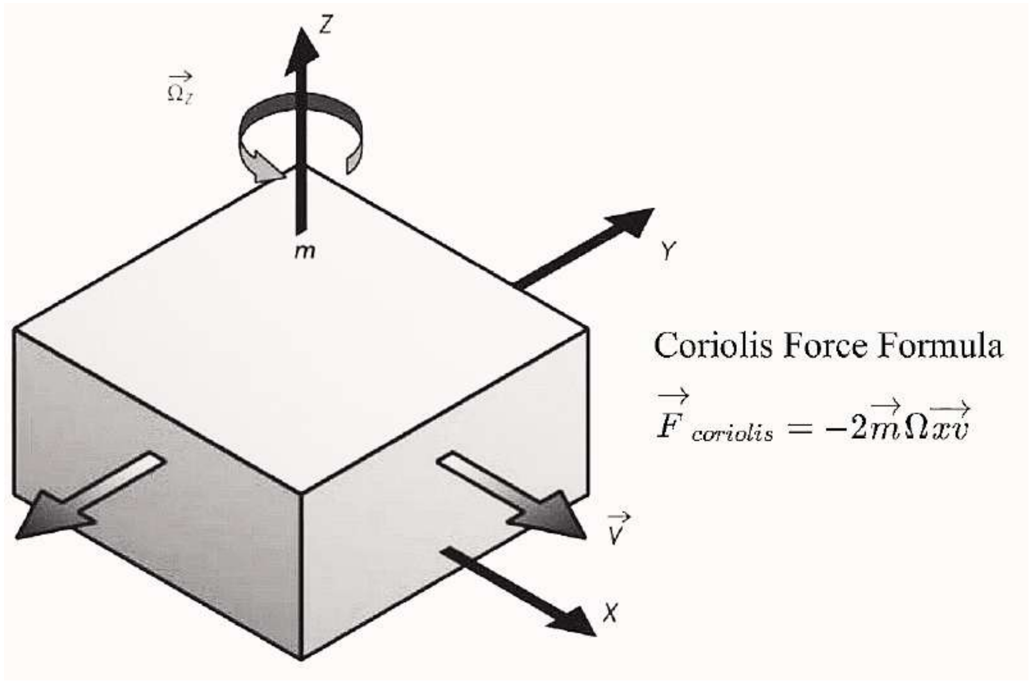

Z
$\overrightarrow{\Omega_Z}$
m
Y
Coriolis Force Formula
$\overrightarrow{F}_{coriolis} = -2\overrightarrow{m}\Omega\overrightarrow{xv}$
$\overrightarrow{V}$
X

Figure 3.3: Representation of the Coriolis force [17].

As most of MEMS sensors, gyroscopes are not free of residual stresses in the microstructure generated at potting. These efforts, though minimal, slightly change the properties of the sensor output when at rest and reading sensitivity. These changes should only be considered when the result expected have high accuracy. Otherwise, it should be ignored. The temperature would have little influence on the sensor output. ITG3205 has a linear behavior at the output, to obtain a value of the angular velocity is necesary to use the equation (3.2).

$$\omega = S(V_{current} - V_{resting}) \tag{3.2}$$

Where:

ω is the angular velocity.

S sensitiveness.

$V_{current}$ current value.

$V_{resting}$ resting value.

3.2.3 Magnetometer (compass)

The magnetometer sensor is responsible for quantifying the intensity and direction of a magnetic field. The transduction principle of magnetometers are diverse and vary according to the technology used by the manufacturer. To measure the magnetic field, some of them use Hall effect magnetometers and some others the Lorenz force or piezo resistive principle [30]. The magnetometer in GY-81 plate is HMC5883L. The three-axis magnetometer is aligned with the accelerometer and the gyroscope. The HMC5883L is solid state-based and manufactured with MEMS technology. It takes measurements using Hall Effect. Figure 3.4 shows how the Hall effect works. When a current flows in an axis, the voltage is perpendicular created to the current flow. When a magnetic field is applied perpendicular to the voltage and current planes. The more perpendicular the applied field, the greater the output voltage. The equation (3.3) calculates a voltage produced by the Hall effect over an axis. In order to establish communication, the protocol used by HMC5883L is I^{2}C [30].

$$E = BIL\sin(\theta) \tag{3.3}$$

Where:

B = applied magnetic field

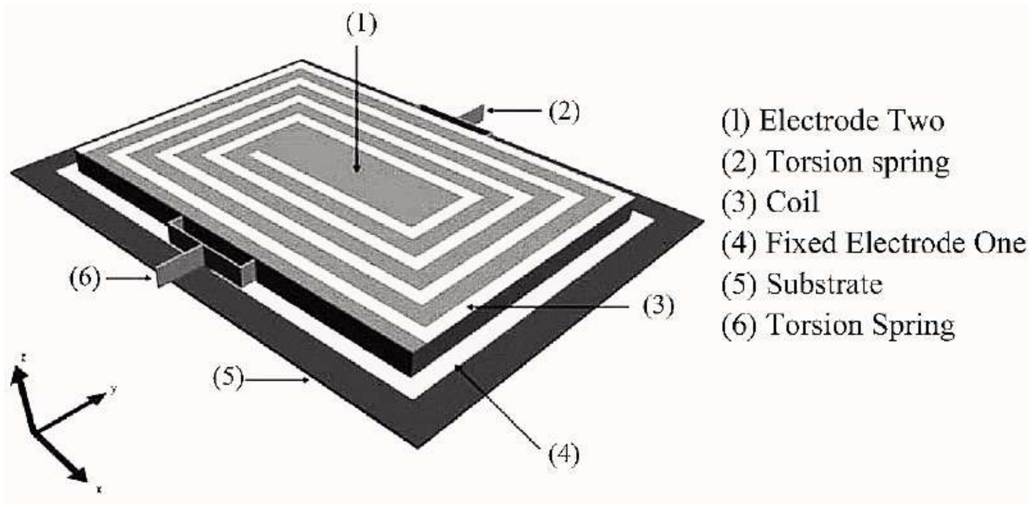


Figure 3.4: Base architecture of a magnetometer [17].

I = applied electric current

L = a constant for the Hall crystal material.

θ = angle between magnetic field and active Hall device element.

In the case of navigation systems, magnetometers measure the earth's magnetic field. The utility is to easily implement an electronic compass system. The main disadvantage is its sensitivity. The readings are very susceptible to external disturbances, e.g., the magnetic fields generated by the system or some object that attracts magnetism. The isolation of the sensor is the recommended solution for avoiding erroneous readings.

3.2.4 Barometer and thermometer

The barometer is the instrument used to measure atmospheric pressure. The atmospheric pressure can help to calculate an estimated altitude, as they are directly related by Ec. (3.4). The GY-81 module includes a BMP085 barometer. It is a solid state sensor, manufactured with MEMS technology and it is a piezo resistive sensor. Measurement is initially analog. Then it goes through an ADC and is communicated using a protocol I^{2}C [29].

$$altitude = 44330 * (1 - (\frac{p}{p_0})^{\frac{1}{5.255}}) \tag{3.4}$$

The BMP085 also provides a sensory temperature value. According to the program within the system, the reading is very precise to the decimal value. The sensitivity is large, which means that if the system overheats, the sensor will take a reading considered by the emitted heat. It is advisable to put the BMP085 outside for proper environmental reading.

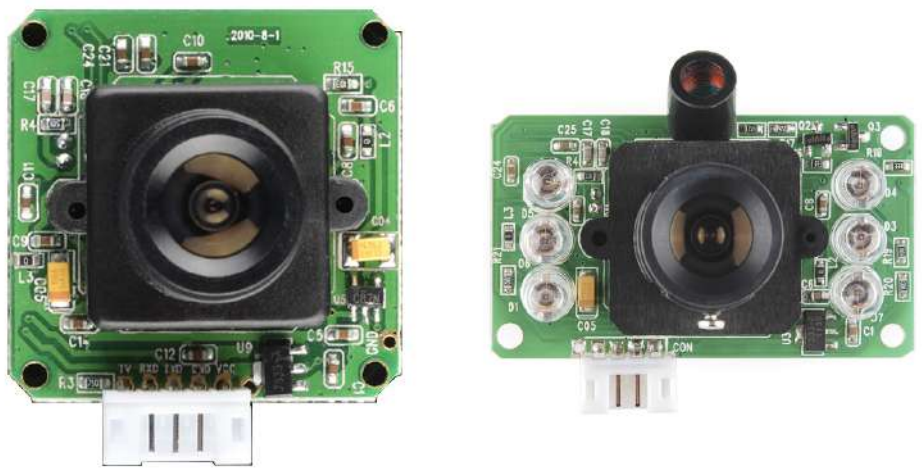
3.2.5 Camera

The most ideal way to obtain images in a hybrid embedded system is through a TTL camera. Two TTL cameras manufacturer by LinkSpire are available for embedded applications, the LS-Y201 and the SEN11610. The LS-Y201 camera is a module containing a high resolution camera of 2MP. It captures images by using the serial port Universal Asynchronous Receiver-Transmitter (UART). The images taken are in JPEG format. The SEN-11610 camera has, in addition to the same features as the LS-Y201, a light sensor and infrared LEDs to capture images in the dark [32].

The Figure 3.5 shows the between the two cameras. Both cameras are low-cost, compact and easy to program, have low power consumption and different image resolutions. For versatility, we used the SEN-11610. Also, the particular ability to take pictures depending on the lighting makes it suitable for application in the system.

3.3 Communication module

Communication via radio frequency (RF) allows wireless communication between the sensor module and the module of the flight computer. The module is divided



(a) (b)

Figure 3.5: Serial TTL compatible cameras. 3.5(a) LS-Y201 Camera RS232 serial communication. 3.5(b) SEN-11610 infrared camera serial communication RS232.

into two, the transmitter and the receiver. The transmitter part will be integrated in the sensor module while the receiver is taking a ground-based communication with the PC module flight computer.

3.3.1 Transceivers

TB387 is a wireless transparent data-transmission module based on 2.4GHz frequency band. The module supports most basic AT commands: baud rate, ID number, frequency settings and inquiries, factory settings, version information. When the module is in transparent data transfer mode, the user transmit data, frame number data module, add packaged rowcount and then automatically transmit at reliable range, the module will automatically re-transmit data to ensure successful transmission. The working voltage is 3.3V-5.5V. TB387 has a RS232 interface (3.3V/5V TTL level), a frequency range between 2402 and 2482MHz. The transparent transmission mode baud rate could be at 2400, 4800, 9600 (Default), 14400, 19200, 38400, 57600, 115200, 12800 or 25600bpm. The maximum distance range is 400m. TB387 is a device that can be embedded in any project that would need wireless transmission with any PLD.

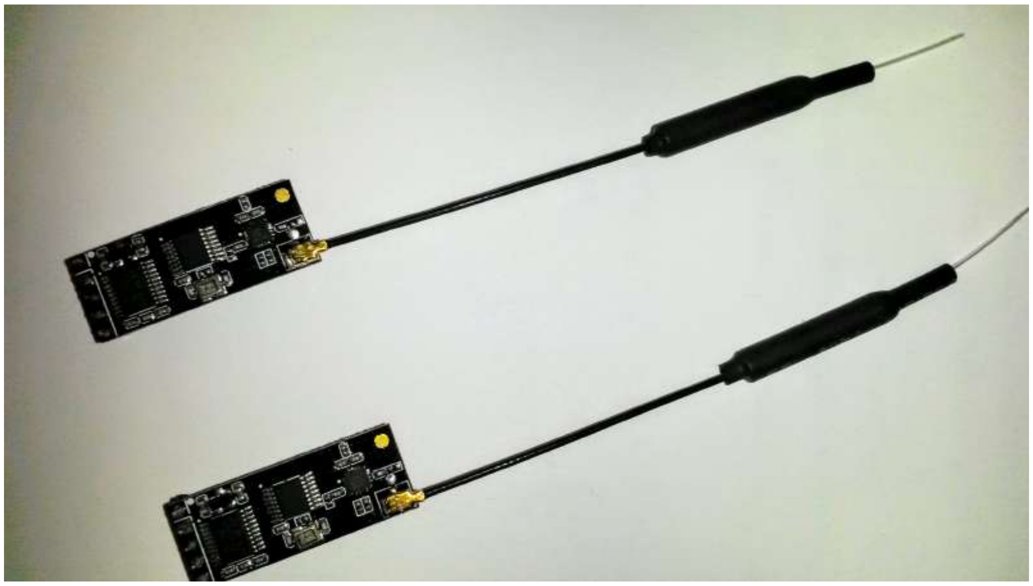


Figure 3.6: TB387 UART RF wireless transceiver module.

3.3.2 Serial-USB converter

Included in the wireless module kit, it is CH340G series USB interface integrated circuit. The CH340G is an USB bus adapter that provides serial or parallel interfaces over the USB bus. The CH340G integrate circuit provides common modem signals to allow adding a UART to a computer. Amoung the CH340G features contains a full-speed USB interface, compatible with USB 2.0 interface, it supports 5V and 3.3V operation, a RS232, RS422 and RS485 with external level shifting components and supports all existing applications by using serial ports without the need of changing the existing code. The device is embedded within the ground station.

3.4 Processing module

The module of the computer flight is divided into two stages, the stage of data acquisition as well as the processing section and display stage. The FPGA board will be responsible for making the acquisition data from the sensors module, then the information will be transferred to a PC for post-processing and being displayed.

3.4.1 Developing system

We can find the Artix 7 FPGA development board as an IC to be coupled to a SOC. Another way to find it is as 7-segment displays assembled in ready-to-be-programmed and input/output general purpose LEDs, among other cards. The Digilent Nexys 4, Figure 3.7, card has LEDs, switches, GPIO, audio and video connector; push buttons, 7-segment display, SD memory reader, among others [19]. It has a 100 MHz clock oscillator and programs that can be charged by using the USB/JTAG port depending on the configuration that we give to the jumpers. Table 3.1 makes a comparison between the characteristics of the different families Xilinx 7.

	Artix-7 Family	Kintex-7 Familia	Virtex-7 family
Logic Cells	215K	478K	1955K
Block RAM	13 Mb	34 Mb	68 Mb
Transceivers	16	32	96
Memory Interface	1066 Mb/s	1866 Mb/s	1866 Mb/s
I/O pins	500	500	1200
I/O Voltage	1.2V, 1.35V, 1.5V, 1.8V, 2.5V, 3.3V	1.2V, 1.35V, 1.5V, 1.8V, 2.5V, 3.3V	1.2V, 1.35V, 1.5V, 1.8V, 2.5V, 3.3V

Table 3.1: Comparative characteristics between Xilinx 7 families.

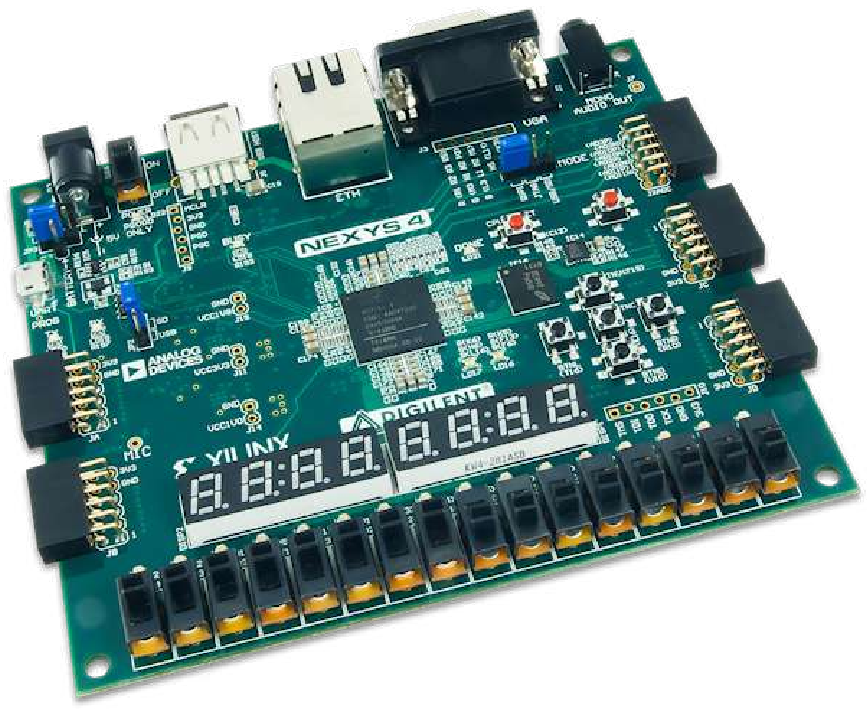


Figure 3.7: FPGA development board Nexys 4 by Digilent

3.4.2 Computer system

The personal computer (PC) is a Hewlett-Packard (HP) Pavilion model. It is loaded with a 64-bit Windows 8.1 operating system Single Language and Intel (R) processor CORE (TM) i5-4210U 2.4 GHz. It is recommended to have an Inter (R) processor over an AMD (R) since at the time of loading the software from Xilinx Inc. They have greater stability to support the installation and when execution of the program and it must be at least CORE i5 model.

Chapter 4

Simulation

4.1 Introduction

This chapter aims to describe the steps to follow in order to design a platform that is subsequently used for testing the functionality and simulations. To explain it, we divide this chapter into two sections. First, we describe the steps to generate the embedded system within the FPGA development board and in this way, we can use the necessary resources for the acquisition and communication of them. The next one describes how to load the program in C language within the FPGA board. We assemble the FPGA board, the sensor module and a power supply on a printed circuit board. Finally, for processing and data visualization, we make an interface in LabVIEW software.

4.2 Data acquisition

The microprocessors available for use in Xilinx FPGA boards with Xilinx EDK software tools can be divided into two. There are soft-core microprocessors (MicroBlaze) and the hard-core embedded microprocessor (PowerPC). We focus on the soft-core microprocessors because it is the one we use. The MicroBlaze is a virtual microprocessor that is built by combining blocks of code called cores inside a Xilinx FPGA. The beauty to this approach is that we only end up with as much microprocessor as we need. We can also tailor the project for the specific needs (i.e.: Flash, UART, General Purpose Input/Output peripherals and etc.). It allows us to balance the required performance of the target application against the logic area cost of the soft processor.

The MicroBlaze processor is a 32-bit Harvard Reduced Instruction Set Computer (RISC) architecture optimized for implementation in Xilinx FPGAs with separate 32-bit or more instruction and data buses running at full speed to execute programs and access data from both on-chip and external memory at the same time. The MicroBlaze core is organized as a Harvard architecture with separate bus interface units for data accesses and instruction accesses. The stack convention used in MicroBlaze starts from a higher memory location and grows downward to lower memory locations when items are pushed onto a stack with a function call. Writing software to control the MicroBlaze processor must be done in C/C++ language. Using C/C++ is the preferred method by most people and is the format that the Xilinx Embedded Development Kit (EDK) software tools expect. The EDK tools have built in C/C++ compilers to generate the necessary machine code for the MicroBlaze processor.

The MicroBlaze processor is useless by itself without some type of peripheral devices to connect to and EDK comes with a large number of commonly used peripherals. The processor system by EDK is connected by On-chip Peripheral Bus (OPB) and/or Processor Local Bus (PLB), so your custom peripheral must be OPB or PLB compliant. EDK uses Intellectual-Property Interface (IPIF) library to implement common functionality among various processor peripherals. Using the IPIF module with parameterization that suits our needs will greatly reduce our design and test effort [25].

Xilinxs Platform Studio (XPS) allows us to control the hardware and software development of the MicroBlaze system. It includes an editor and a project management interface for creating and editing source code and a Software tool flow configuration options. Software Development Kit (SDK) is an integrated development environment, complimentary to XPS, that is used for C/C++ embedded software application creation and verification. SDK is built on the Eclipse^{TM} open-source framework. According to the mentioned above and in the previous chapter, the HIL final scheme implemented is depicted in Figure 4.1. Figure 4.2 shows the set of connections between the modules just to know how they are embedded and how they are divided into the ground station and the flight computer [25].

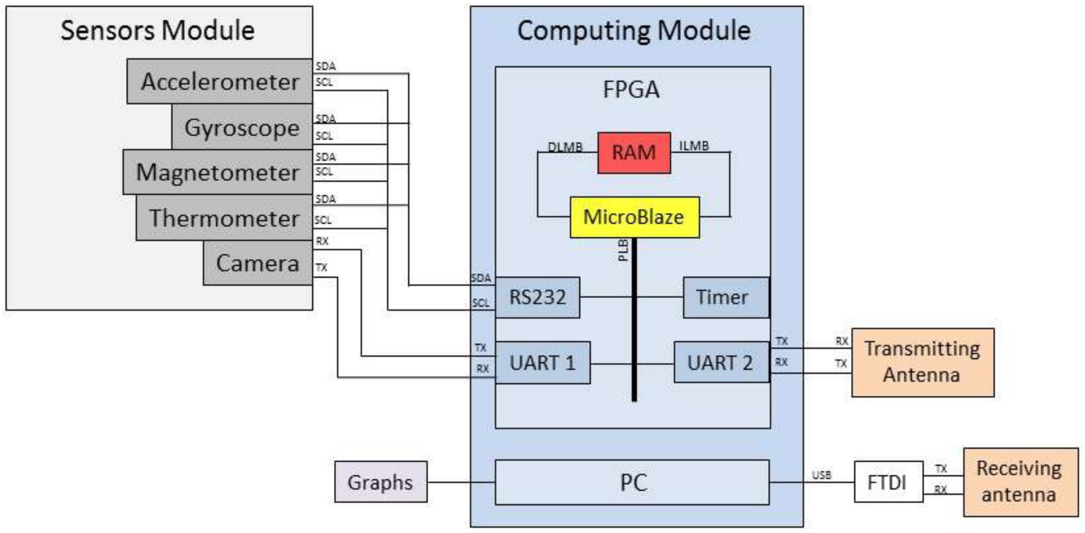


Figure 4.1: HIL final scheme implementation.

The Appendix A details the process to implement the embedded system in the EDK software. The instructions are based on [18, 19, 20, 21, 23, 24]. Meanwhile the appendix B describes step by step how to include the C program into the FPGA board for the data acquisition. Eventhough the programming language is C, the instructions to interact with the FPGA board need to be commands that the board understand and that comands are in [22].

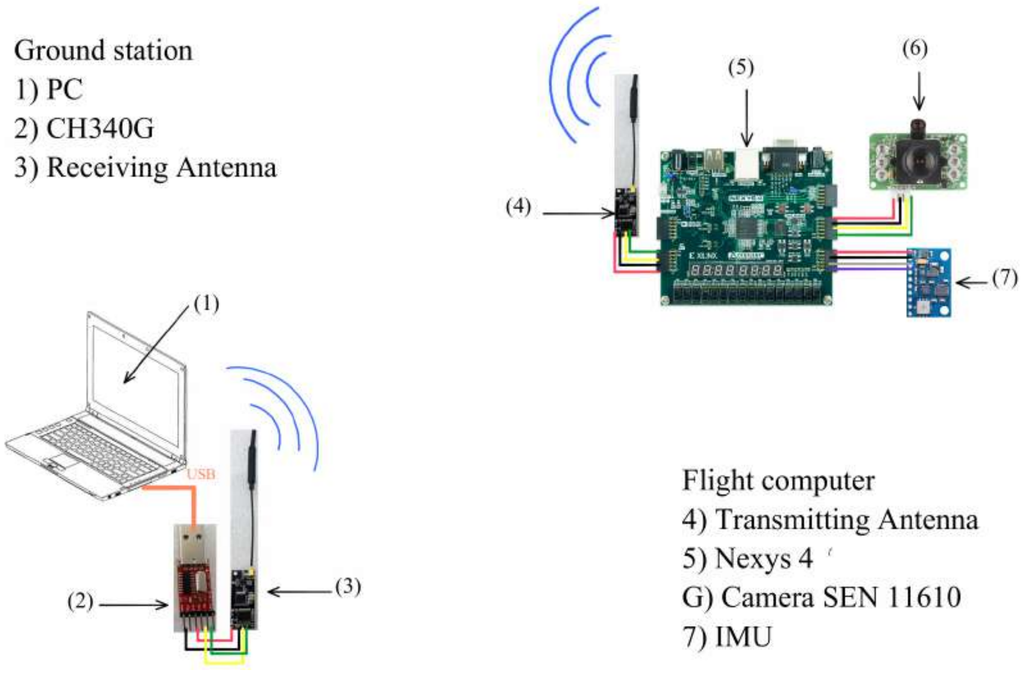


Figure 4.2: Final embedded system for the HIL simulation.

4.2.1 PCB Development

In order to proceed to the stage of validation and testing, we implement the components needed for a ground station and the flight computer. Then, we proceed to improve all the implementations and, taking account a power source, we elaborate a printed circuit board (PCB) so we can integrate all the components of the sensor module and the transmitting antenna. The PCB has the features to be placed together with the Nexys 4. The final printed design of the PCB is shown in Fig. 4.3. After polishing the PCB, the components that integrate the flight computer are welded, leaving the PCB as shown in Figure 4.4.

4.3 Data processing

In flight processing and post-processing navigation, the components sensed by the IMU have to be transformed onto a reference coordinate system. To introduce the

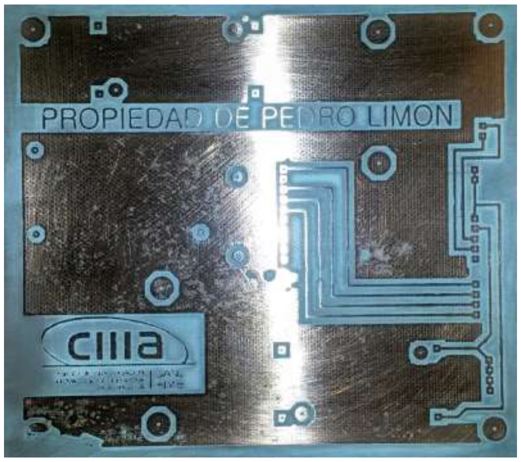


Figure 4.3: Back view of flight computer printed circuit board.

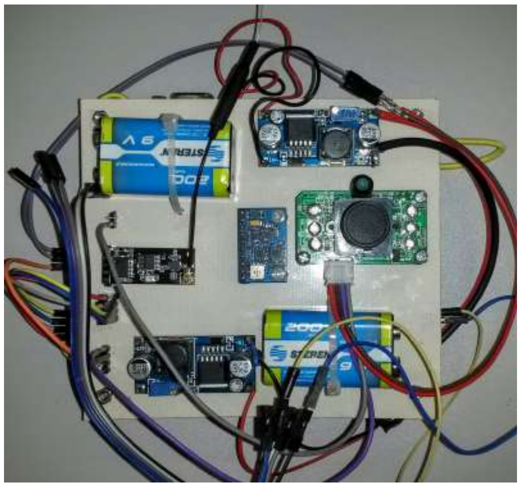


Figure 4.4: Frontal view of flight computer printed circuit board.

object motions into a computerized system there are several representation to characterized rotation, kinematics and orientation of an object in the space. The most common representations are through the Euler angles and quaternions. However there are other representations as Rodrigues parameters.

4.3.1 Euler angles

Euler angles represent the orientation of a body relative to a reference system "x y z" or relative to another rigid body [4, 6]. The rotations that describe the orientation in space of a body in terms of Euler angles are:

- ϕ (roll) denotates a rotation of X axis.
- θ (pitch) denotates a rotation of Y axis.
- ψ (yaw) denotates a rotation of Z axis.

We give an example of an Euler angle orientation in Figure 4.5 by [4]. By using notations such as $(\psi)_3$, $(\theta)_2$, $(\phi)_1$, we can specify the Euler angles and the axes of sequential rotations which denotes a rotation of (OXYZ) by angle ψ OZ, resulting in the intermediate orientation, (OX'Y'Z'), followed by a rotation by angle θ about OY', (OX''Y''Z''), and then a final rotation by angle ϕ about OX'', to produce the new orientation (OX'''Y'''Z'''). The rotation matrix to get the Euler angles is referred in equation (4.1), this is a general equation and it is applied when the vehicle can move and rotate in the direction of the 3 axis [1, 3]. The Euler angles of the roll, pitch and yaw can be determined from the values of the rotation matrix according to the inverse transformation in Eq. (4.2).

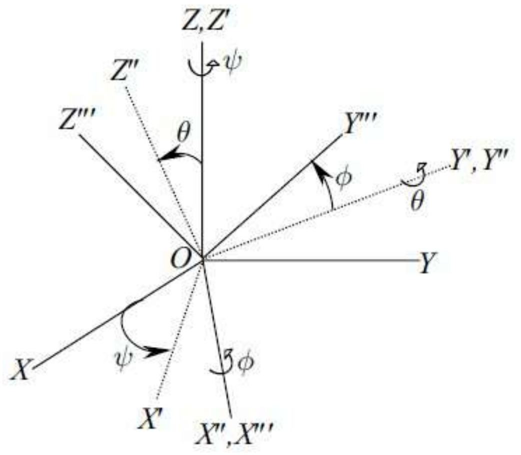


Figure 4.5: Schematic diagram of the Euler angle orientation.

$$C = \begin{pmatrix} \cos\theta\cos\psi & \cos\theta\sin\psi & -\sin\theta \\ (\sin\phi\sin\theta\cos\psi - \cos\phi\sin\psi) & (\sin\phi\sin\theta\sin\psi - \cos\phi\cos\psi) & \sin\phi\cos\theta \\ (\cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi) & (\cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi) & \cos\phi\cos\theta \end{pmatrix} \tag{4.1}$$

$$\begin{aligned} \phi &= \tan^{-1}\frac{c_{23}}{c_{33}} \\ \theta &= -\sin^{-1} c_{13} \\ \psi &= \tan^{-1}\frac{c_{12}}{c_{11}} \end{aligned} \tag{4.2}$$

where c_{ij} represent the element (i,j) of C.

Euler angles are not unique. There are certain orientations that can not be determinate by the rotation matrix and it becomes singular an useless. In a commercial aircraft this does not cause a problem. However, when dealing with fighter aircrafts, missiles or spacecrafts it is different because they can have vertical attitudes [4, 6].

4.3.2 Quaternion

The quaternion is a special set composed of four mutually dependent scalar parameters, q_0, q_1, q_2, q_3, such that the last three form a vector called vector part and it were discovered by W. R. Hamilton. It is represented in the Ec. (4.3). The first part, q_0, represents the scalar part. This constraint of the Ec. (4.4) implies that the quaternion yields only three independent, scalar parameters, as in the principal angle/Euler axis or the Euler angle attitude representations[1, 3]. Since the four elements of the quaternion satisfy the constraint equation, it can be said that attitude orientations vary along the surface of a four-dimensional unit sphere without any singularity [4]. Hamilton also discovered that the multiplication of the quaternions is not commutative, but it is associative and distributive when adding, it is checked in Ec. (4.5).

$$q = \left\{ \begin{array}{c} q_1 \\ q_2 \\ q_3 \end{array} \right\} \tag{4.3}$$

$$q_0^2 + q_1^2 + q_2^2 + q_3^2 = 1 \tag{4.4}$$

$$ij = k = -ji, \qquad jk = i = -kj, \qquad ki = j = -ik \tag{4.5}$$

In practice, quaternions are more useful as a means of representing navigation and orientation. The representation of relative orientation using Euler angles or rotation matrix are easy to develop and to visualize, but require computationally intensive trigonometric function evaluations when derived from the rotation matrix [4]. In other words, for a computer is more demanding to process the data acquired and make operation of a matrix of 3×3 than in a vector of 1×4. Therefore, using quaternion instead of Euler axis makes a shorter code allowing less storage space. Also the singularity problem, mentioned above, occurs when describing attitude kinematics in terms of Euler angles and therefore it is not an effective method for vehicles that can move in a vertical direction. Similarly, by implementing filters, such as Kalman, the mathematical operation are reduced and it is easier to calculate predictive motion or possibly for spacecraft an input vector.

The quaternion was calculated by taking a benchmarks from the accelerometer and the magnetometer in order to generate the rotation matrix (Euler angles, equation (4.1)). To make this possible data acquired from accelerometer and magnetometer is transformed into a vector of 1×3 while the benchmarks are taken as the first acquisition of these sensors. Once the previous points are set, the execution of the program is as follows:

- Divide the accelerometer vector into its norm

$$r_1 = \frac{acc\_vector}{\parallel acc\_vector \parallel} \tag{4.6}$$

- Calculate the cross product of Ec. (4.6) and magnetometer vector

$$X = r_1 \times mag\_vector \tag{4.7}$$

- Divide the resulting vector of Ec. (4.7) between its norm.

$$r_2 = \frac{X}{\parallel X \parallel} \tag{4.8}$$

- Calculate the cross product of resulting vector of Ec. (4.6) and Ec. (4.7)

$$r_3 = r_1 \times r_2 \tag{4.9}$$

- Divide the accelerometer reference into its norm

$$s_1 = \frac{acc\_ref}{\parallel acc\_ref \parallel} \tag{4.10}$$

- Calculate the cross product of the resulting vector of Ec. (4.10) and the magnetometer reference

$$q_u = s_1 \times mag\_ref \tag{4.11}$$

- Divide the resulting vector of Ec. (4.11) between its norm

$$s_2 = \frac{q_u}{\parallel q_u \parallel} \tag{4.12}$$

- Calculate the cross product of resulting vector of Ec. (4.10) and Ec. (4.12)

$$s_3 = s_1 \times s_2 \tag{4.13}$$

- By adding the results as a product of the transposed Ecs. i.e. (4.6), (4.8) and (4.9) with (4.10), (4.12) and (4.13) respectively, a 3×3 matrix results and the transpose is the rotation matrix, just as Ec. (4.14) shows:

$$C = rot\_mat = [r1^T * s1 + r2^T * s2 + r3^T * s3]^T \tag{4.14}$$

We can calculate the quaternion elements from the elements of the rotation matrix according to Ec. (4.15)

$$\begin{aligned} q_0 &= \pm\tfrac{1}{2}\sqrt{1+C_{11}+C_{22}+C_{33}} \\ q_1 &= \frac{C_{23}-C_{32}}{4q_0} \\ q_2 &= \frac{C_{31}-C_{13}}{4q_0} \\ q_3 &= \frac{C_{12}-C_{21}}{4q_0} \end{aligned} \tag{4.15}$$

4.3.3 Interface

The interface, in which we can see the results of the acquiring data, it is made in LabVIEW. In order to obtain a better performance, all the charts and matrices are displayed in an individual screen as show in Figure 4.6. The blue chart is for modifying the serial communication characteristics. The green chart graphs and displays the altitude (in meters) and the pressure (in Pascal) values. The yellow chart contains a thermometer to illustrate the temperature and display its value in a numeric form (in Celsius degrees). The three charts of below graph each axis of the accelerometer (in G), gyroscope (in rad/s) and magnetometer respectively. Lastly, there are three matrices, the first displays the quaternion, the second the rotation matrix and the third Euler angles (radians) of the roll, pitch and yaw.

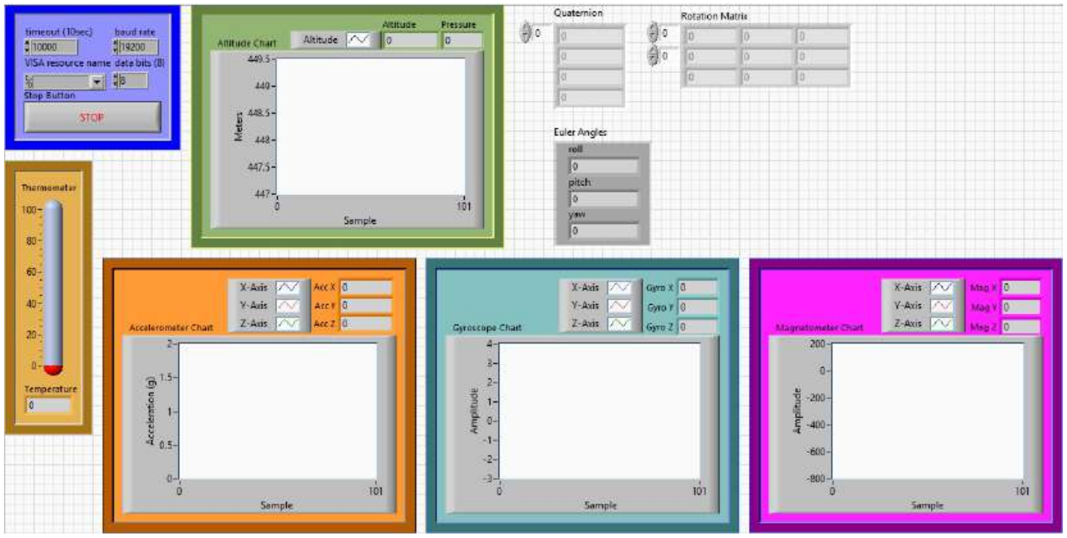


Figure 4.6: LabVIEW interface.

Chapter 5

Results and discussion

The experiments were set in order to get information in real time according to the simulation for a missile system. The experiments were performed by lifting and lowering the flight computer. These tests were done in the Channel 22 antenna located at the top of the hill "El Mirador". The flight computer has a mass of 290 grams and it was raised to a height of 30 meters. The antenna structure generates a Faraday cage effect making the communication cleaner and the only disturbance is caused by the engine of the air-conditioner of the building.

Figure 5.1 is divided in three graphs. The upper graph indicates the pressure, the graph in the middle shows the temperature and the graph below depicts the altitude. According to Ec. (3.4), pressure and altitude graphs shown in Figure 5.1. It states that the atmospheric pressure is inversely proportional to the altitude (for greater pressure, lower altitude). These two graphs indicate that the flight computer began in an altitude of 1114 meters, then it was gradually lifted to a middle height of maximum reached and it stood there for a couple of minutes. The next lifting helped the computer flight to reach the maximum height (1142 meters). Then, it was slowly descended to the midpoint. Once at this stage, it was lifted faster than the first time and returned to the initial height. These movements lasted for about 14 minutes because in previous experiments the power source capability was about 15-16 minutes. At the beginning of the experiments, the flight computer was at ground floor aside of the ground station and the engines. The initial temperature was at 30°C due to the heat emanated from the engines and as the computer flight moved away from the engines the temperature dropped gradually until it reached the ambient temperature which was at 24°C.

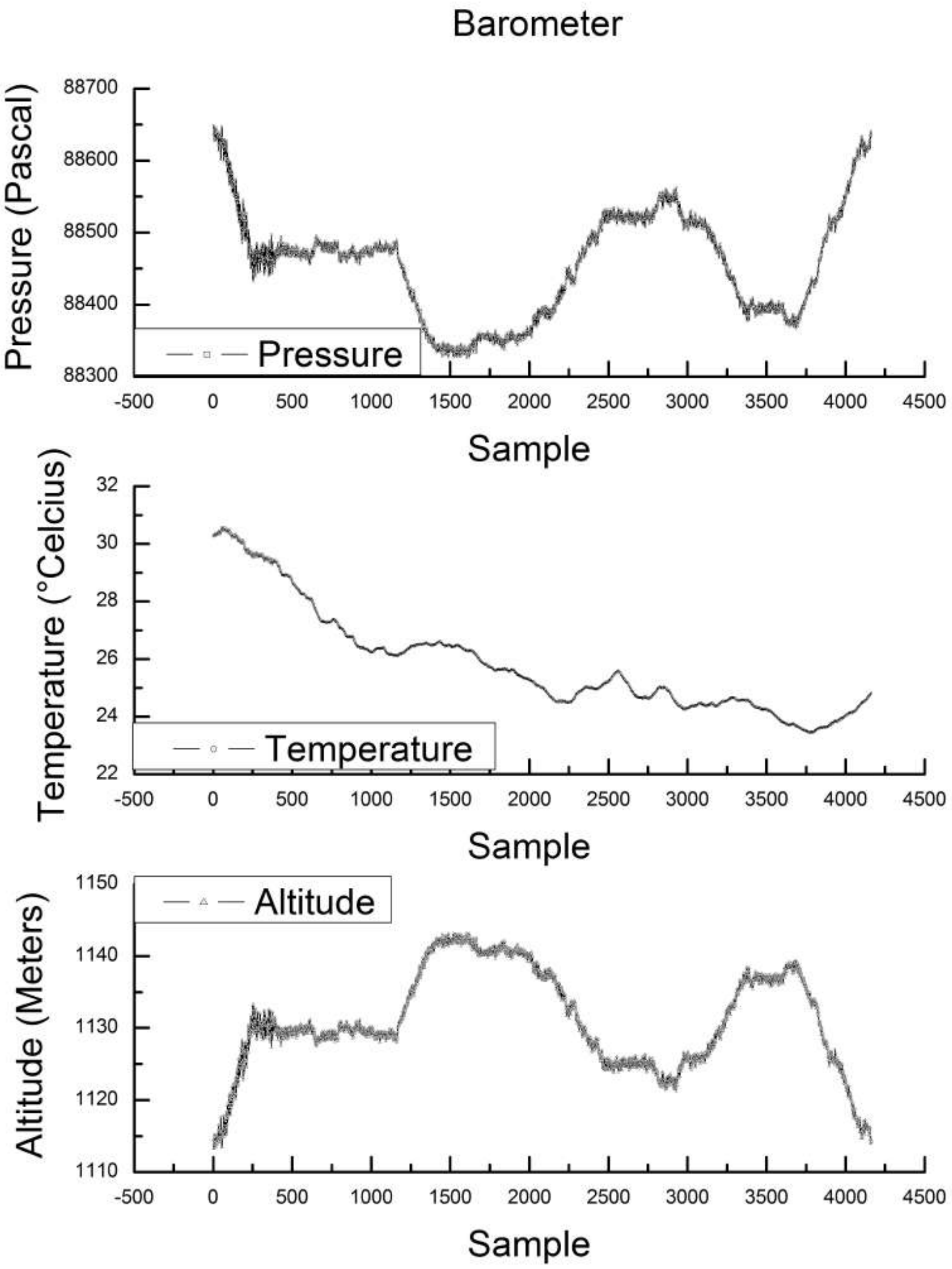


Figure 5.1: Data from the accelerometer indicating the motion of each axis.

Three conditions were taken in this test to interpret the data gotten by the accelerometer. The inclination produced in the flight computer due to the center of gravity not being centered; the rope used has a high elasticity coefficient and the position of the sensor. The IMU was centered as Figure 4.4 shows. In the beginning, the flight computer was leveled to the floor making the X and Y axis to stand near to zero and Y-axis opposite to the center of the Earth so the reading gave a value near to 2 as seen in Figure 5.2. These initial values mean that the frame if X and Y axis was almost in parallel with the ground and the Y axis was pointing to the sky.

When the flight computer started to lift, the inclination in the X-axis changed to an average of -0.7, Y-axis to an average of -0.2 and the Y-axis increased its value to an average of -1.8. These moves indicate that the axes were not leveled and the positive side had a direction opposite to the floor. Each time the rope was pulled to be lifted, the movement produced acceleration in the direction of the rope and as the three axes had certain value to that direction. All of them reacted but not at the same amplitude. As the rope had a high elasticity coefficient at the time of finishing pulling, the computer flight descended with a spring-like effect. It could be said that each time the rope was pulled, it generated acceleration to the direction of the rope (becoming a positive acceleration) and between each pull an opposite acceleration was generated by a spring-like effect becoming a negative acceleration.

Due to the center of gravity is not coupled to the time the computer flight was lifted, the three gyroscope axis presented similar angular velocity, especially X-axis and Y-axis. Figure 5.3 makes proof of this. The two physical actions that produce the angular movement are the yanks to lift the flight computer and the pounding of the wind. Seen from the ground station, if the flight computer rotated in counterclockwise the angular velocity was positive but if rotated in clockwise the velocity would be negative. By the conditions, in the beginning of the test the yanks produced the first angular movements; however, as the experiment progressed the wind began to blow continuously causing that the flight computer moved like a pendulum. The pendulum movement caused the flight computer positive rotated on its

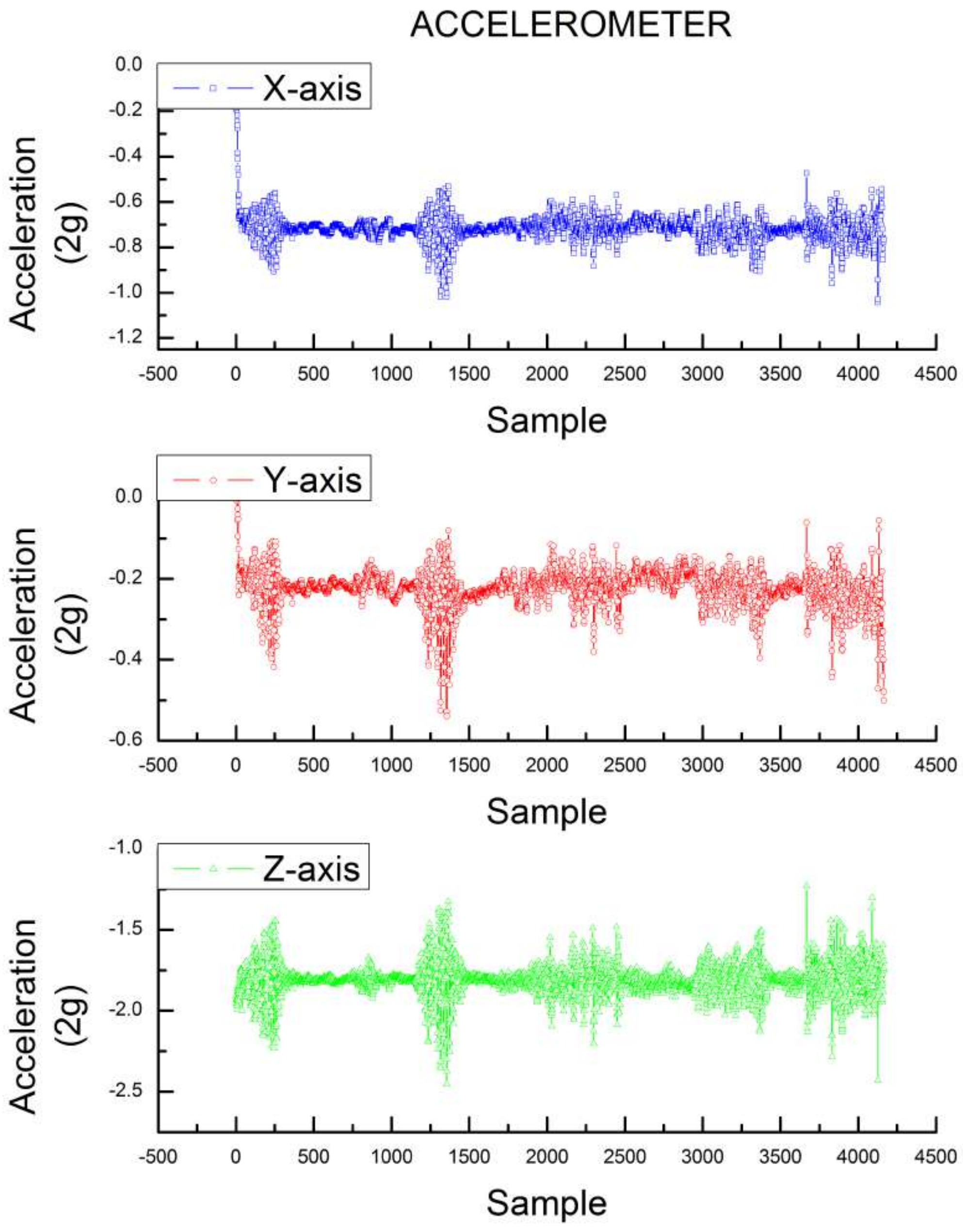


Figure 5.2: Data from the accelerometer indicating the motion of each axis.

axis while the tension of the rope made it back into position. These two movements produced the undulations in the axes graphs. The reason why the amplitude of Z-axis is greater than X and Y axes is because the rotation is mostly concentrated on this axis.

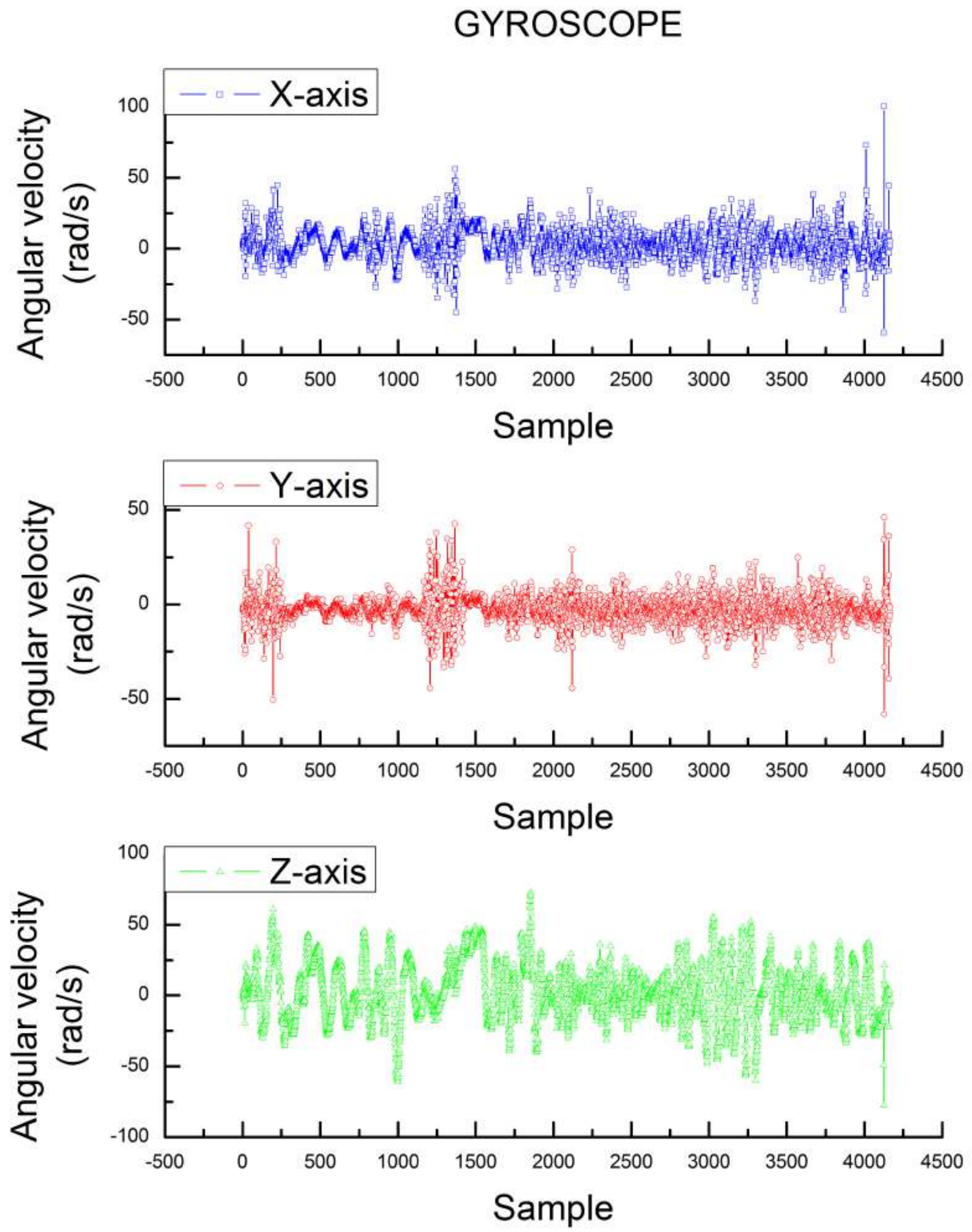


Figure 5.3: Data taken from the gyroscope indicating the angular velocity in each axis.

The magnetic field preset a constant disturbance generated by the building engines that are near the ground station. Figure 5.4 shows the magnetic field flow

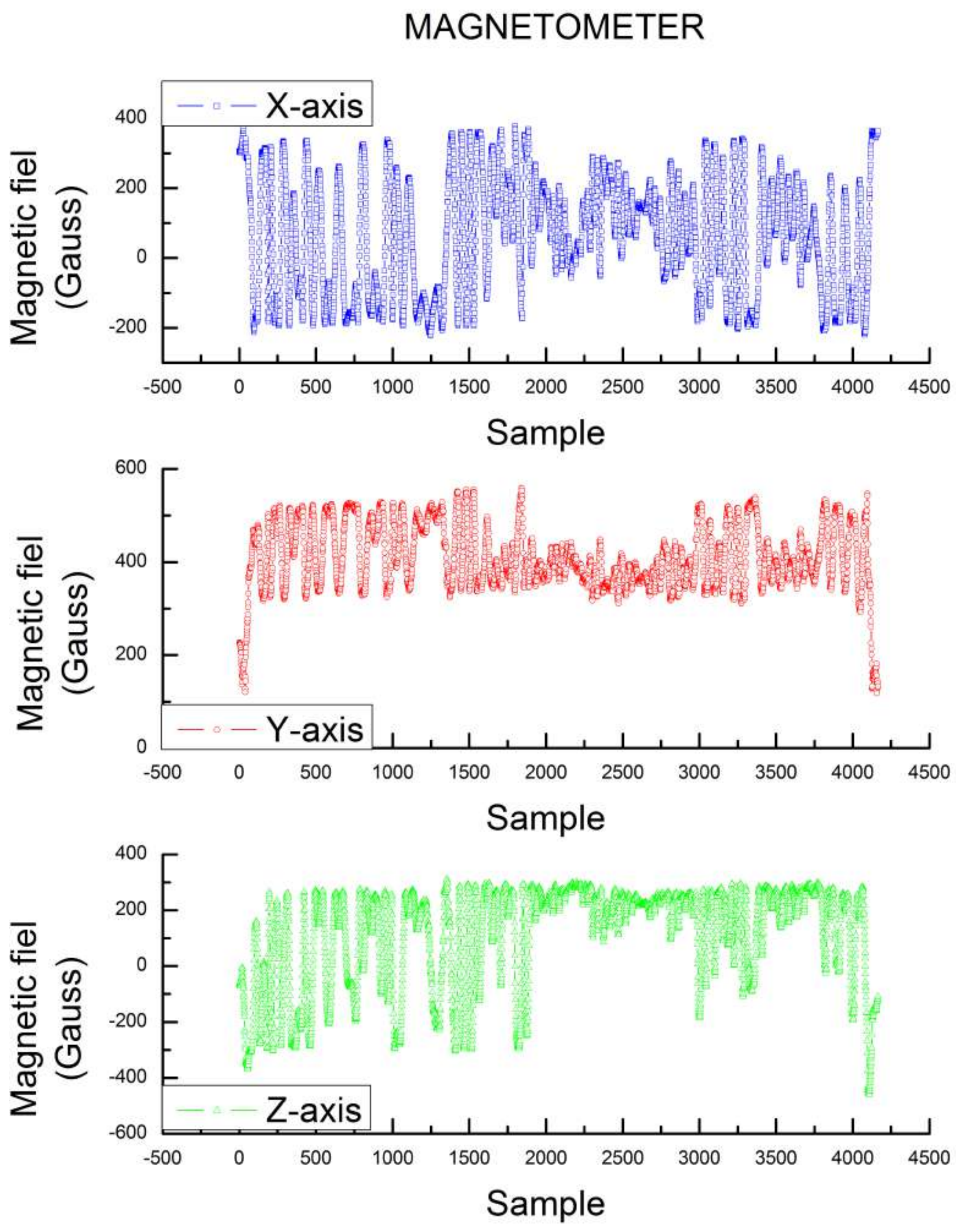


Figure 5.4: Data taken from the gyroscope indicating the disturbance of the magnetic fiel that each axis receive.

that each axis of the magnetometer received. The sinusoidal shape of each axis is due to the rotation that the flight computer made over the Earth magnetic field flow that pass through the area. The reason that some waver are smaller than other is because even though the flight computer was rotating on its axis, the movement was not considered enough to increase the amplitude of the value.

During the test, some images were taken but due to the disturbance in the transmission produced by engines. Therefore, some acquired data presented corruption in its code when it was received at the graound station. This corruption or disturbance produced a line code longer, shorter or none. In the images are three types of bars representing each of the disturbances. If the bar has colors that do not match, it means that the code line was longer. If the end of the bar is in black, the code line was shorter. If the bar is countiniusl in gray color, the code line was not transmitted. A total of 8 images were taken but the code of the eighth was interrupted. The camera is capable of take pictures with a resolution of 640×480 pxs, 320×240pxs or 160×120pxs [32]. The transmitted code size is directly related to the resolution of the image. For practical purposes, the images taken in the experiment were in 320×240pxs, which is the standard resolution.

The first image that the camera took occurred when the flight computer was at ground floor. The image in Figure 5.5 was taken at an altitude of 1130 m at the beginning of the first stage. Figure 5.6 was taken during the second lifting at a height of 1132 m. Figure 5.7 shows the flight computer standing at the mid-height for the second time and the altitude marked 1123 m. Figure 5.8 was taken while the flight computer made the final descent and it was taken at a height of 1138 m as last image.

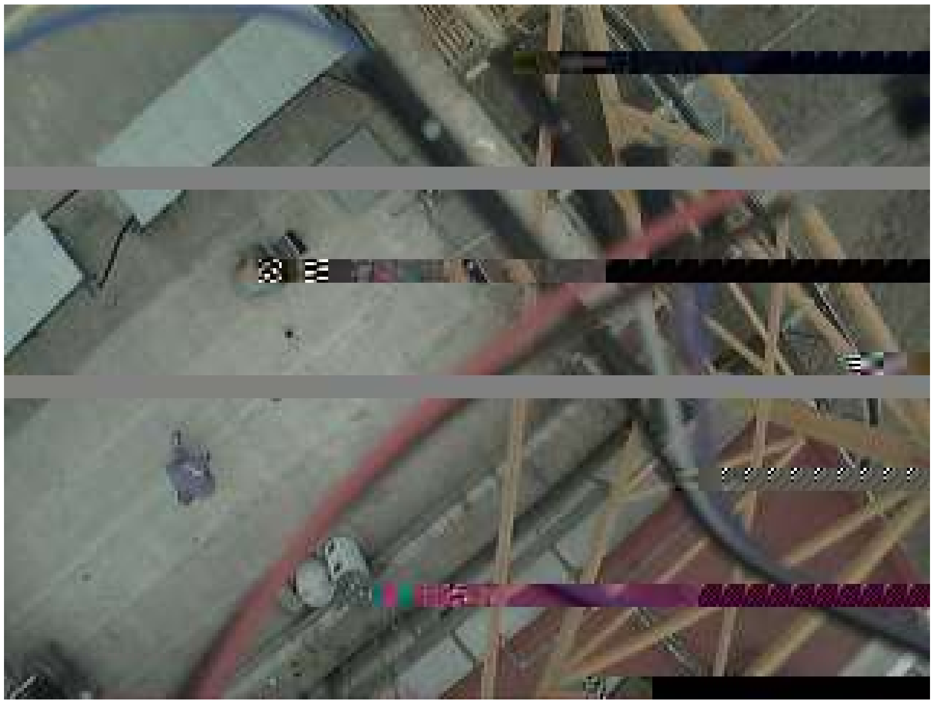


Figure 5.5: Image 1, 16 meters from floor.

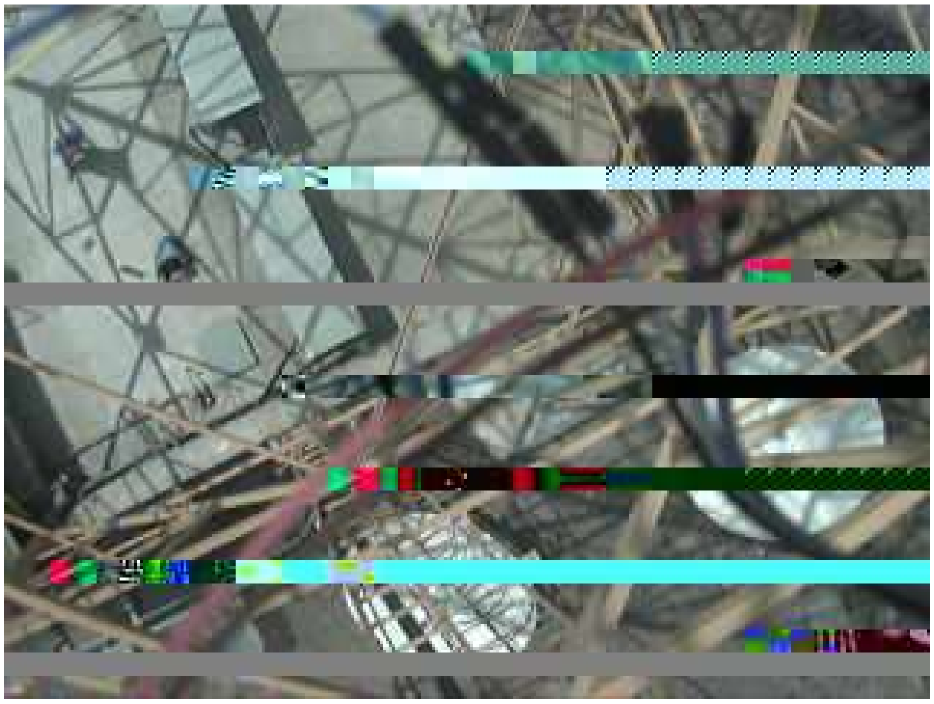


Figure 5.6: Image 2, 18 meters from floor.

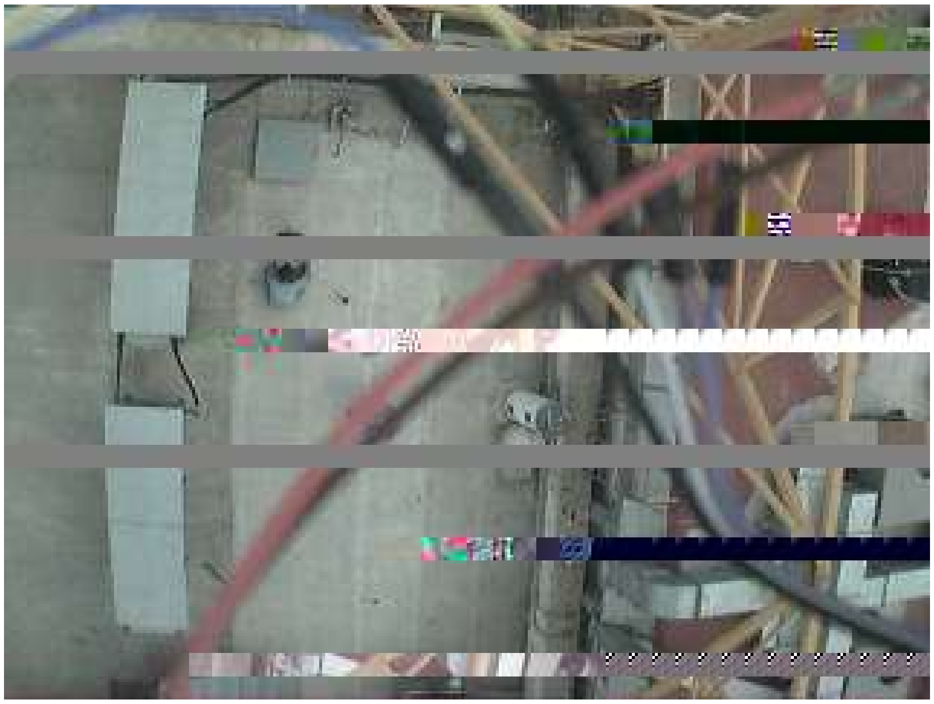


Figure 5.7: Image 3. 10 meter from floor

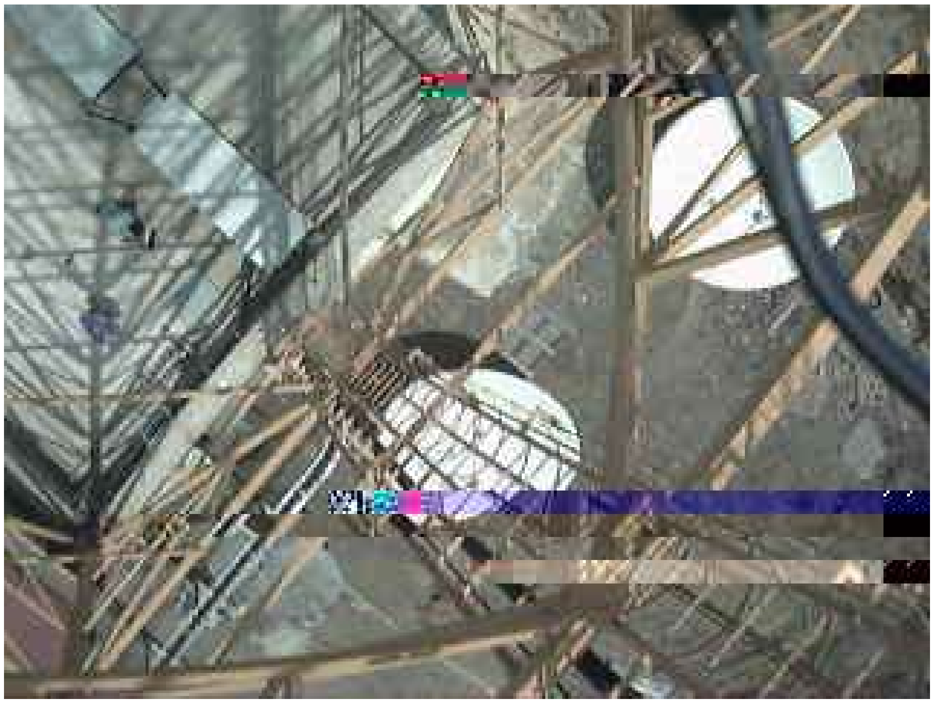


Figure 5.8: Image 4, 24 meter from floor.

Chapter 6

Conclusions and future work

6.1 Conclusions

By considering that the project (design, development and material selection) started from scratch, and given the results obtained, we can validate that a developed FPGA board performs data acquisition of navigation sensors as well as images and then transfer them through a wireless communication module to a ground station for processing. For a 14 minutes test, the ground station received 4183 data from each sensor of which 4165 data were correctly sent. The transmission had an efficiency of 99.57% on the acquisition and data transmission with a frequency of 293 data per minute. The computer flight weight was very light, 290 grams; thereby the results of the HIL simulation were more precise. Also it is considered that the assembly of the flight computer was handmade. As a consequence, a few of the signals and processes could produce noise at the time of the acquisition.

By regarding the sensors, the camera was the only sensor which had an error while sending the data and it was expressed by sending larger or shorter lines of code seen in the ground station. By considering its precedence, the communication module had a considerable efficiency in spite of being near an engine that produced disturbance. The results were displayed in real time in a LabVIEW interface and stored either in an .xls or .txt file. With this information, a quaternion was generated and the value of the quaternion changed as a function of the received information. The power supply supported a considerable time during acquiring and transferring data. However, also it should be mentioned that the camera IR LEDs were not in used. With higher quality sensors, data acquisition would be faster and more accurate.

By using a computer-based mathematical process, the results and some points of the next section, we can work with PWM signals to manipulate servo motors and thus expand the HIL simulation by adding these servos as the missile wings controllers.

6.2 Future work

A thesis work is always evolving, changing and improving, what is proposed in this paper is that way. One way to confirm that we are in the right path is by writing the same program but in a VHDL code and compare results in the data transmission speed and the difficulty to interact with the FPGA board. We could add a GPS to complement the sensor module, it would provide information of the route by which the missile traveled. We can also set the camera to take pictures faster or enable it to take video, but if it is not possible then we have to find and install an appropriate camera for this task.

Now then, if the flight computer orients the direction that the missile has and will have, it will have to direct the trajectory, and for doing that the missile needs actuators. In a flying missile, those actuators would be the wings or flats that are luckily controlled by servo motors, which are controlled, at the same time, by PWM signals generated within the flight computer. To calculate the position that the servos should have to guide the missile is necessary to calculate an estimate position of the target and with a Kalman filter the flight computer can archive this.

We must not forget that the sensors can be exchanged for others with higher quality or other communication protocol. Another module that could be improved is the flight computer. We can change the FPGA board by one with better resources and in turn build a PCB that contains the FPGA chip and the peripherals used for the tasks. At the same time, both the communication and the power modules can be improved by modules that transmit cleaner signals and with longer transmission distance meanwhile we can replace the power source with battery cells and a smaller electronic arrangement to regulate the current consumed by the sensor module and the flight computer.

Last but not least, the flight computer must be inserted into a compartment of the missile body, it should be attached to a chasing to resist the accelerations and movements caused by the missile. The modules must be distributed such that the space they occupy is the required, should not alter the center of gravity and have an interaction with the FPGA board equally efficient. By doing this, we can test with flying vehicles for interaction with the actuators (servos) and measuring the time it takes for the processing module to calculate a new route and take action to follow that new route.

Appendix A

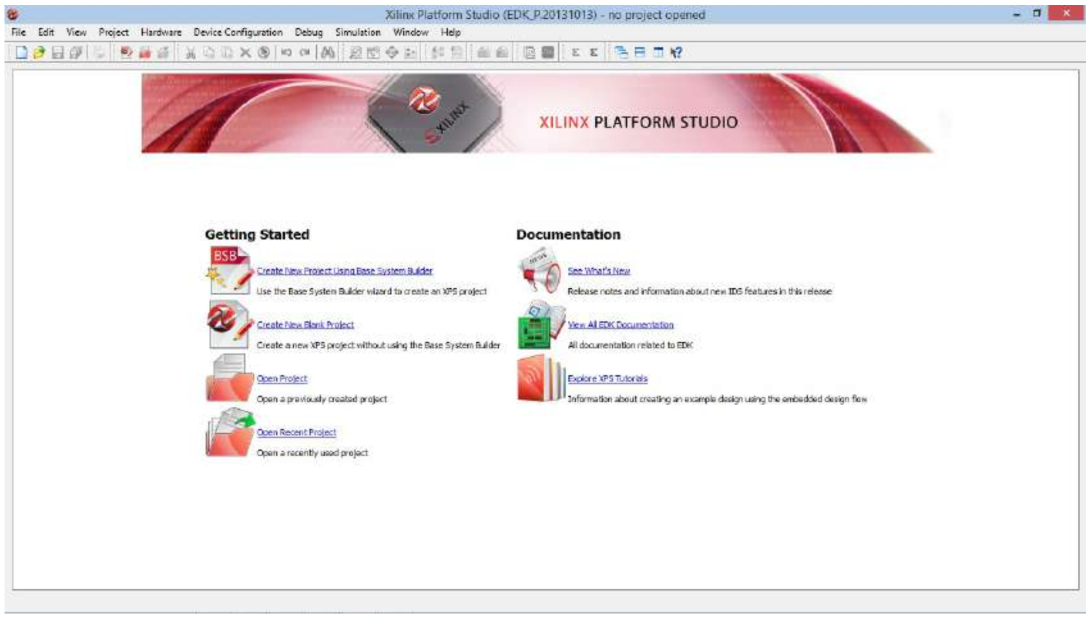
Implementation of the embedded system

To create a project file using XPS of the EDK software is necessary to follow the next instructions:

1.- Excecute as administrator XPS version installed in the computer. The XPS main window would like figure A.1.

2.- Click on **Create New Project Using Base System Builder** and the assistant window will appear. The project files must be the closest to C:\, it is recomended no more than 3 carpets. Select **AXI System** and the assistant window shoul be like A.2. Click **OK**.

3.- In the **Base System Builder** screen select **Create a System for a Custom Board** and the option in **Board Configuration** sould be enable.
Select **Architecture - artix7**
Select **Device - xc7a100t**
Select **Reference Clock Frequency - 100.00 MHz**
Select **Package - csg324**
Select **Speed Grade - -1**
Select **Reset Polarity- Active High**


Xilinx Platform Studio (EDK_P.20131013) - no project opened
File Edit View Project Hardware Device Configuration Debug Simulation Window Help
XILINX PLATFORM STUDIO

Getting Started

Create New Project Using Base System Builder
Use the Base System Builder wizard to create an XPS project

Create New Blank Project
Create a new XPS project without using the Base System Builder

Open Project
Open a previously created project

Open Recent Project
Open a recently used project

Documentation

See What's New
Release notes and information about new IDS features in this release

View All EDK Documentation
All documentation related to EDK

Explore XPS Tutorials
Information about creating an example design using the embedded design flow

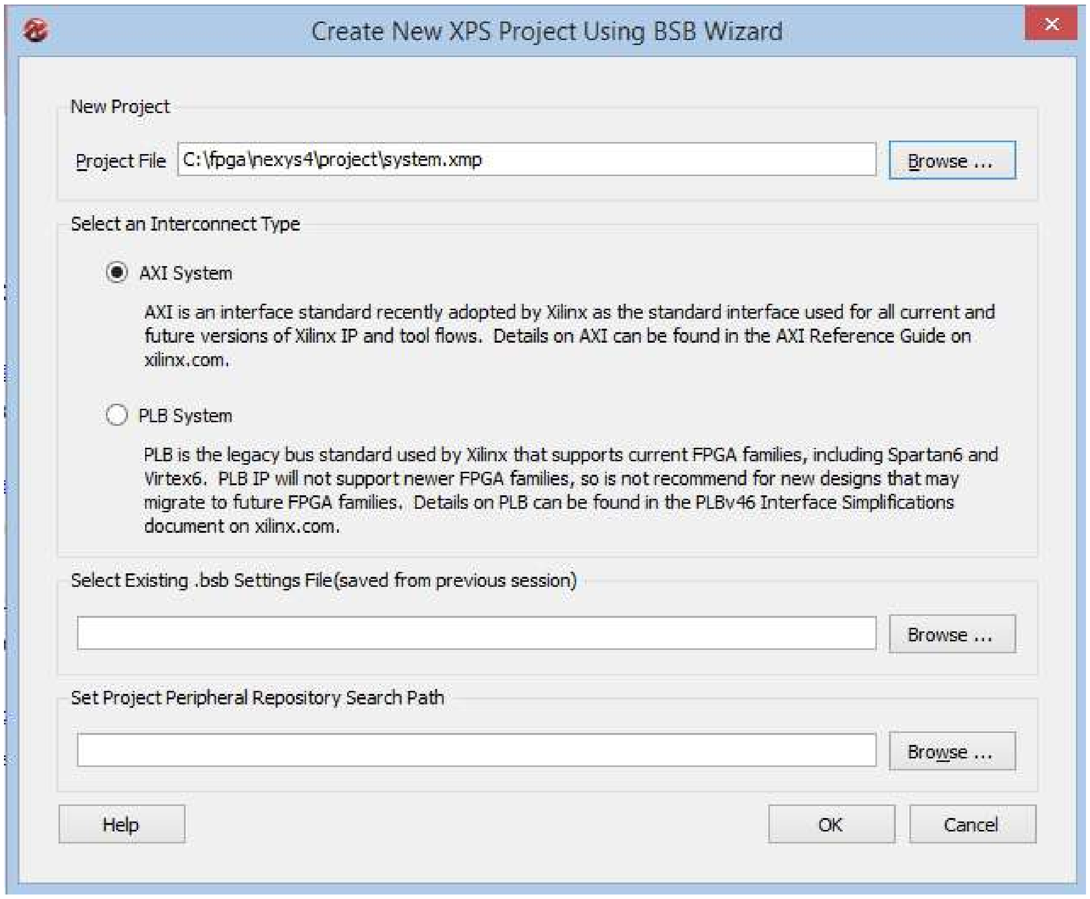
Figure A.1: Xilinx Platform Studio main window.

Select **Single MicroBlaze Processor System**

Select **Optimization Strategy - Throughput**. Figute A.3 shows the previus steps. Click **Next** and the **Processor, Cache, and Peripheral Configuration** auxiliar will appear.

4.- In the **Base System Builder - Processor, Cache, and Peripheral Configuration** panel select **Local Memory Size - 32 KB**. Click in **Add Device...** and the **Add IO Devices for Generic Board** will be displayed.
Select **IO Interface Type - UART**
Select **Device - RS232**
Click **OK**
Change **Include Peripherals for microblaze_0 - RS232 - Baud Rate - 19200**. Figure A.4 shows the panel modified.
click **Finish** and after several seconds the **Xilinx Platform Studio** editor window will appear.

5.- In the Xilinx Platform Studufio editor the first thing to do is to add the left peripherals which are other UART port, an I^{2}C interface and a timer/counter.



New Project

Project File: C:\fpga\nexys4\project\system.xmp

Select an Interconnect Type

● AXI System

AXI is an interface standard recently adopted by Xilinx as the standard interface used for all current and future versions of Xilinx IP and tool flows. Details on AXI can be found in the AXI Reference Guide on xilinx.com.

○ PLB System

PLB is the legacy bus standard used by Xilinx that supports current FPGA families, including Spartan6 and Virtex6. PLB IP will not support newer FPGA families, so is not recommend for new designs that may migrate to future FPGA families. Details on PLB can be found in the PLBv46 Interface Simplifications document on xilinx.com.

Select Existing .bsb Settings File(saved from previous session)

Set Project Peripheral Repository Search Path

Figure A.2: Window to create a XPS project using BSB wizard.

To add a second UART port which will communicate with the camera, focus in **IP Catalog**

Deploy the option **Communication Low Speed**

Left click on **AXI UART (Lite)** Select **Add IP**. Select **Yes**.

Select **UART Lite Baud Rate - 38400**

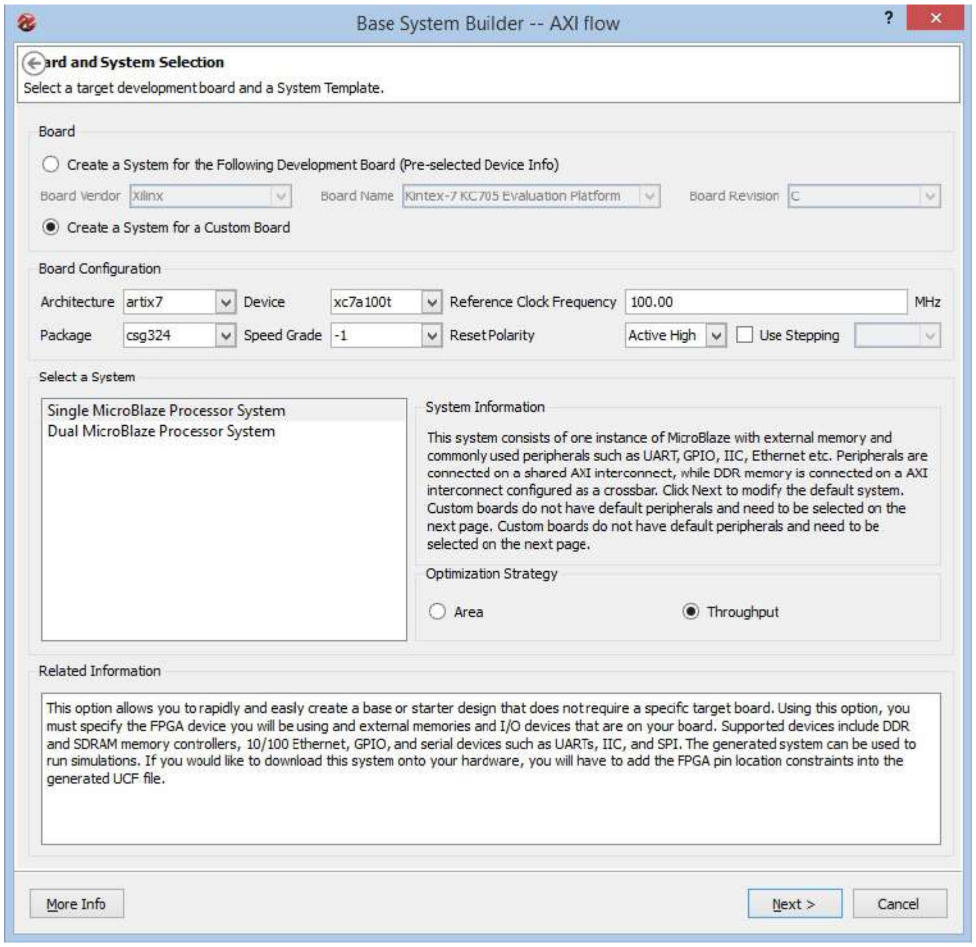
Select **Parity Type - Odd**. Figure A.5 shows how the options of the UART port for must be. Click **OK**

Click **OK**.

To add port available of process the communication I^{2}C: Deploy the option **Communication Low Speed**

Left click on **AXI IIC Interface**

Select **Add IP**. Select **Yes**. We left the characteristics as they are, figure A.6.


Base System Builder -- AXI flow

Board and System Selection

Select a target development board and a System Template.

Board

( ) Create a System for the Following Development Board (Pre-selected Device Info)

Board Vendor: Xilinx — Board Name: Kintex-7 KC705 Evaluation Platform — Board Revision: C

(●) Create a System for a Custom Board

Board Configuration

Architecture	artix7	Device	xc7a100t	Reference Clock Frequency	100.00	MHz	
Package	csg324	Speed Grade	-1	Reset Polarity	Active High	Use Stepping	

Select a System

Single MicroBlaze Processor System
Dual MicroBlaze Processor System

System Information

This system consists of one instance of MicroBlaze with external memory and commonly used peripherals such as UART, GPIO, IIC, Ethernet etc. Peripherals are connected on a shared AXI interconnect, while DDR memory is connected on a AXI interconnect configured as a crossbar. Click Next to modify the default system. Custom boards do not have default peripherals and need to be selected on the next page. Custom boards do not have default peripherals and need to be selected on the next page.

Optimization Strategy

( ) Area (●) Throughput

Related Information

This option allows you to rapidly and easly create a base or starter design that does not require a specific target board. Using this option, you must specify the FPGA device you will be using and external memories and I/O devices that are on your board. Supported devices include DDR and SDRAM memory controllers, 10/100 Ethernet, GPIO, and serial devices such as UARTs, IIC, and SPI. The generated system can be used to run simulations. If you would like to download this system onto your hardware, you will have to add the FPGA pin location constraints into the generated UCF file.

Figure A.3: Base System Builder for an AXI flow.

Select **OK**

Select **OK**

To add an inter timer/counter to the delays that the program should have: Deploy the option **DMA and Timer**

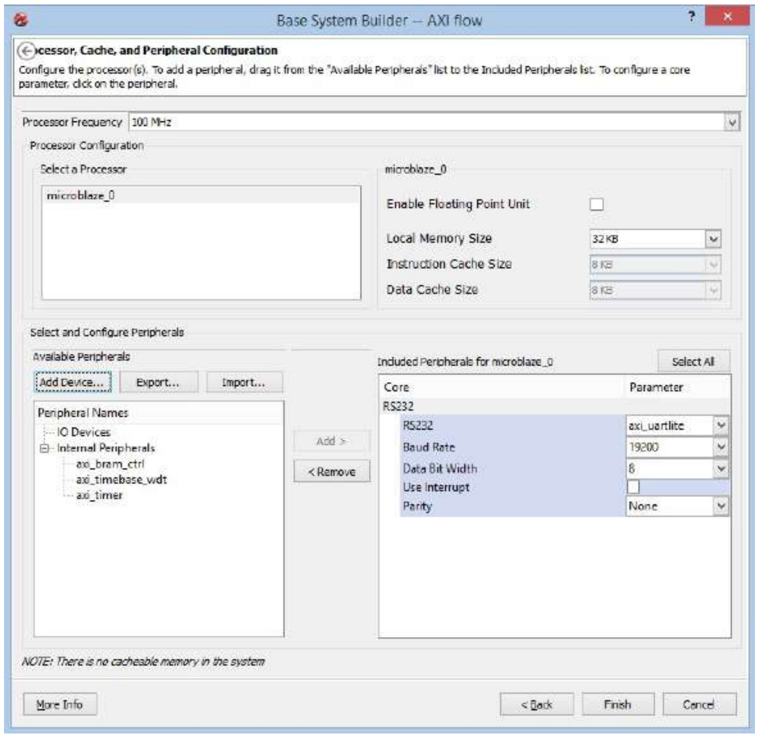
Left click on **AXI Timer/Counter**

Select **Add IP**. Select **Yes**. Figure A.7 shows the configuration we need to make the timer/counter a delay.

Select **OK**

Select **OK**

6.- Now the ports are connected to MicroBlaze and available but not addressed in the externat ports. To see the name of the external ports select the tab **Sys-**



cessor, Cache, and Peripheral Configuration

Configure the processor(s). To add a peripheral, drag it from the "Available Peripherals" list to the Included Peripherals list. To configure a core parameter, click on the peripheral.

Processor Frequency: 100 MHz

Processor Configuration

Select a Processor: microblaze_0

microblaze_0

Enable Floating Point Unit	☐
Local Memory Size	32 KB
Instruction Cache Size	8 KB
Data Cache Size	8 KB

Select and Configure Peripherals

Available Peripherals

Peripheral Names
- IO Devices
- Internal Peripherals
  - axi_bram_ctrl
  - axi_timebase_wdt
  - axi_timer

Included Peripherals for microblaze_0

Core	Parameter
RS232	
RS232	axi_uartlite
Baud Rate	19200
Data Bit Width	8
Use Interrupt	☐
Parity	None

NOTE: There is no cacheable memory in the system

Figure A.4: Base System Builder - Processor, Cache, and Peripheral Configuration panel.

tem Assembly View - Ports, There would be the characteristics of each port as figure A.8 shows.

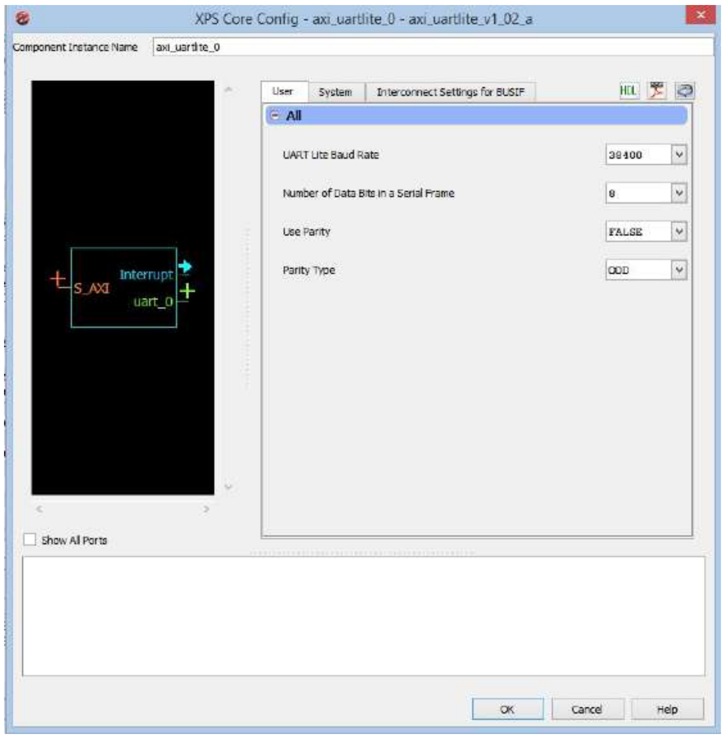
To relate the name of the ports with the output addresses that the FPGA board has [19] select the tab **Project**

Select **UCF File: data\system.ucf**

Write the code that relates the external ports with the output addresses that is shown in Appendix C.

7.- Click on **Generate BitStream** (Figure A.9) to create a BIT file cointaining the hardware design to be inported to SDK. The process would take several minutes.

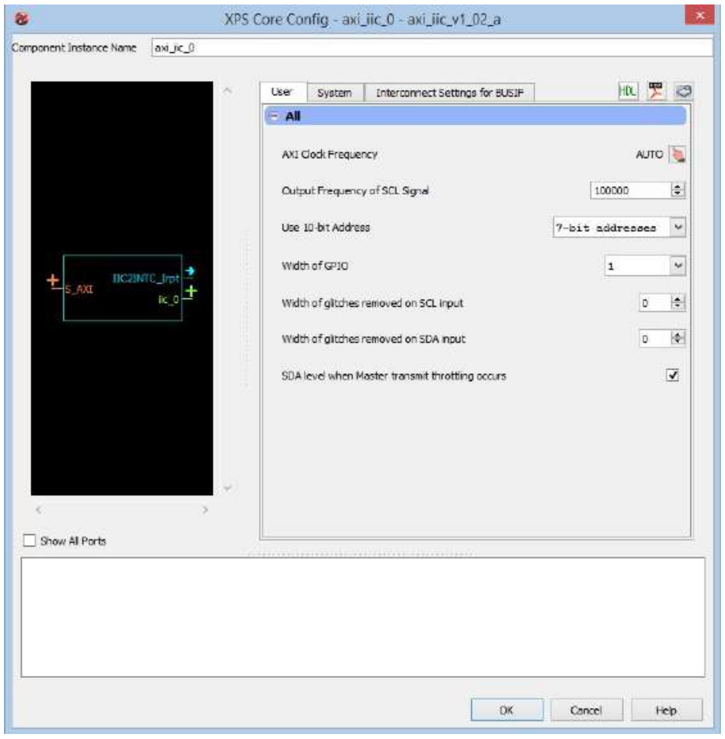
8.- Click on **Export Design** (Figure A.10), the window **Export to SDK/Launch SDK** will appear (Figure A.11). Select **Export & Launch SDK**. The next


XPS Core Config - axi_uartlite_0 - axi_uartlite_v1_02_a
User
System
Interconnect Settings for BUSIF
All
Interrupt
S_AXI
uart_0
Show All Ports
OK
Cancel
Help

Component Instance Name: axi_uartlite_0

UART Lite Baud Rate: 38400
Number of Data Bits in a Serial Frame: 8
Use Parity: FALSE
Parity Type: ODD

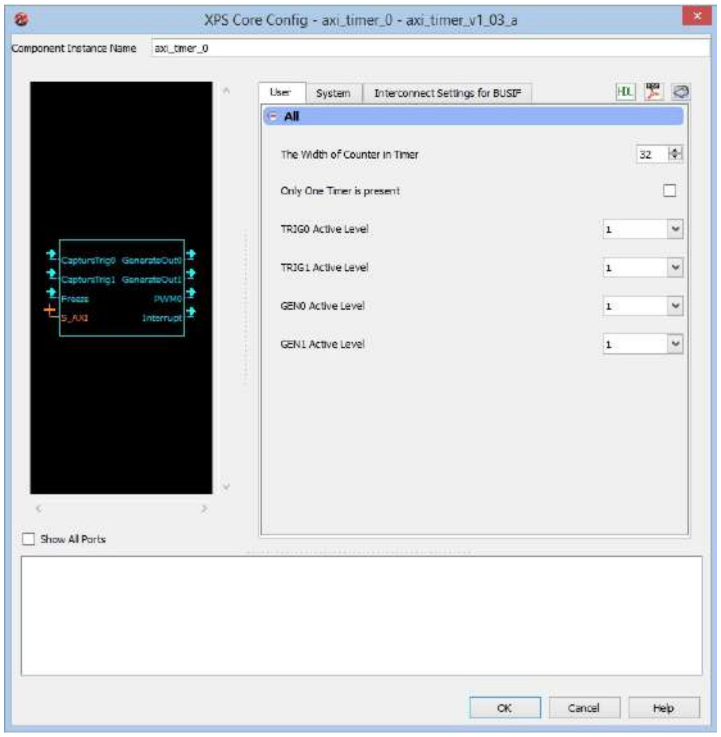
Figure A.5: Second UART port configuration.


XPS Core Config - axi_iic_0 - axi_iic_v1_02_a
User
System
Interconnect Settings for BUSIF
All
IIC2INTC_Irpt
S_AXI
iic_0
Show All Ports
OK
Cancel
Help

Component Instance Name: axi_iic_0

AXI Clock Frequency: AUTO
Output Frequency of SCL Signal: 100000
Use 10-bit Address: 7-bit addresses
Width of GPIO: 1
Width of glitches removed on SCL input: 0
Width of glitches removed on SDA input: 0
SDA level when Master transmit throttling occurs: ✓

Figure A.6: I^{2}C port configuration.

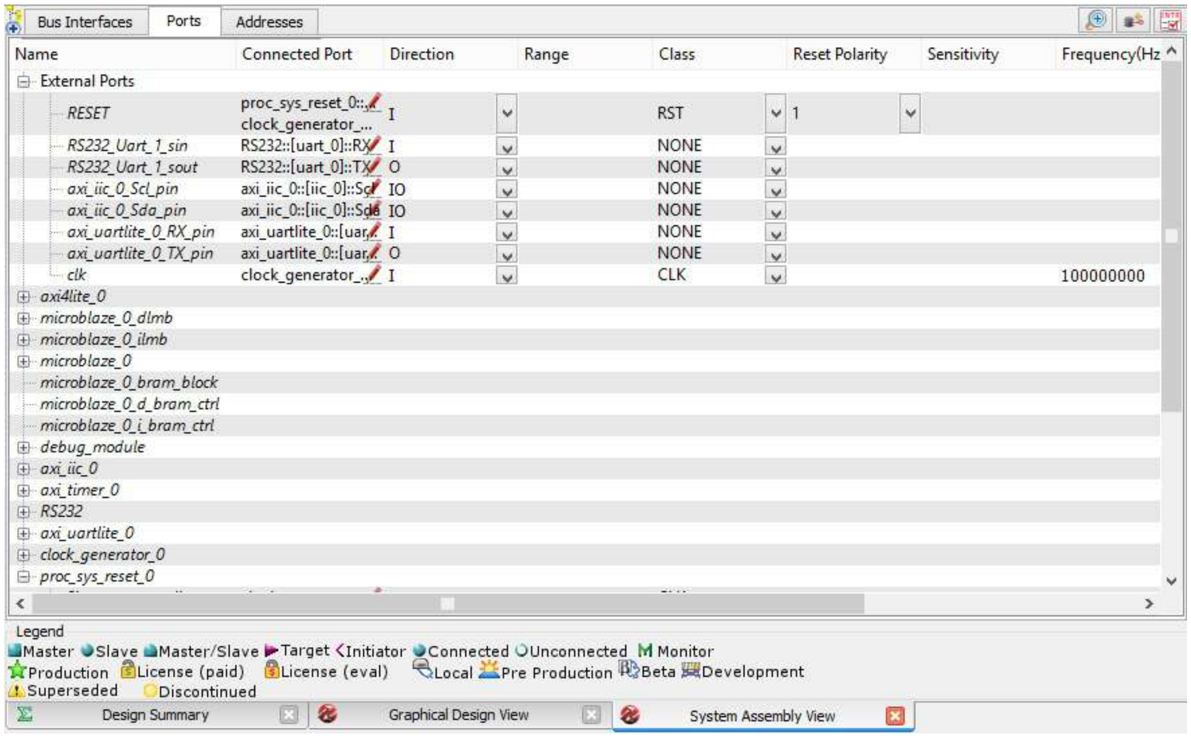

XPS Core Config - axi_timer_0 - axi_timer_v1_03_a

Component Instance Name: axi_tmer_0

All

The Width of Counter in Timer	32
Only One Timer is present	☐
TRIG0 Active Level	1
TRIG1 Active Level	1
GEN0 Active Level	1
GEN1 Active Level	1

Figure A.7: Timer/Counter configuration.



Name	Connected Port	Direction	Range	Class	Reset Polarity	Sensitivity	Frequency(Hz
External Ports							
RESET	proc_sys_reset_0::… clock_generator_…	I		RST	1		
RS232_Uart_1_sin	RS232::[uart_0]::RX	I		NONE			
RS232_Uart_1_sout	RS232::[uart_0]::TX	O		NONE			
axi_iic_0_Scl_pin	axi_iic_0::[iic_0]::Scl	IO		NONE			
axi_iic_0_Sda_pin	axi_iic_0::[iic_0]::Sda	IO		NONE			
axi_uartlite_0_RX_pin	axi_uartlite_0::[uar	I		NONE			
axi_uartlite_0_TX_pin	axi_uartlite_0::[uar	O		NONE			
clk	clock_generator_…	I		CLK			100000000
axi4lite_0							
microblaze_0_dlmb							
microblaze_0_ilmb							
microblaze_0							
microblaze_0_bram_block							
microblaze_0_d_bram_ctrl							
microblaze_0_i_bram_ctrl							
debug_module							
axi_iic_0							
axi_timer_0							
RS232							
axi_uartlite_0							
clock_generator_0							
proc_sys_reset_0							

Legend
Master Slave Master/Slave Target Initiator Connected Unconnected Monitor
Production License (paid) License (eval) Local Pre Production Beta Development
Superseded Discontinued

Figure A.8: Ports caracteristics.

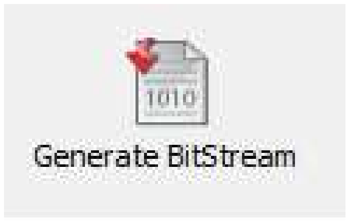


Figure A.9: Generate BitStream button.

window to appear is **Workspace Launcher** and it is to indicate the direction in which we will keep the programs to run on the board, it is recomended to save the programs in the same folder created for the embedded system, as shown in Figure A.12.

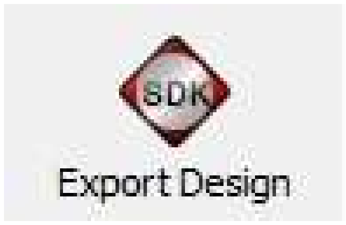
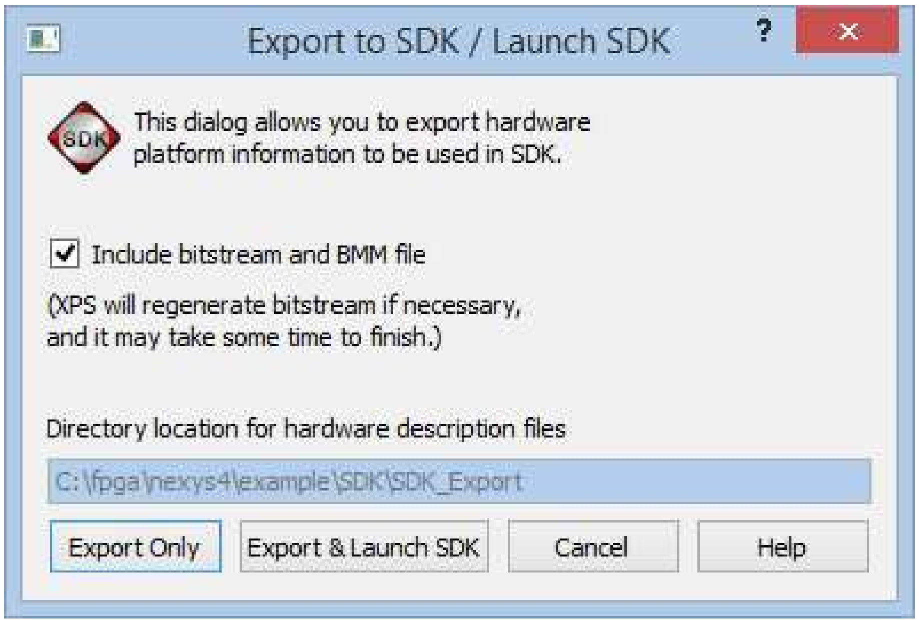


Figure A.10: Export Design button.



Export to SDK / Launch SDK

This dialog allows you to export hardware platform information to be used in SDK.

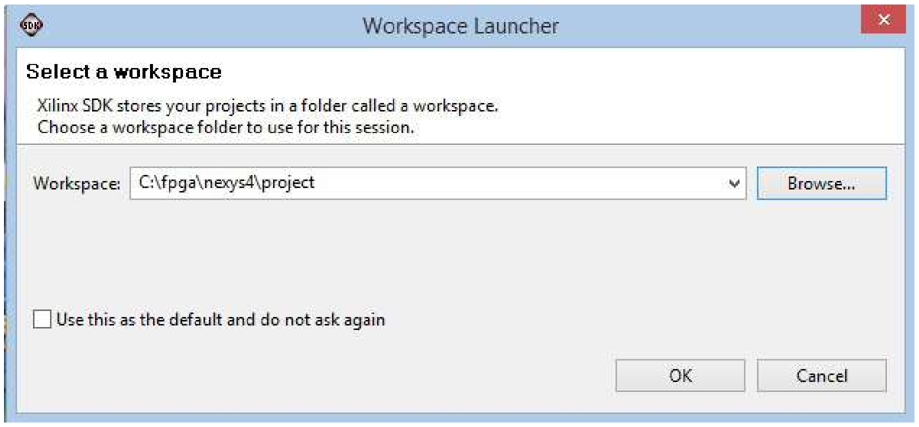
[✓] Include bitstream and BMM file

(XPS will regenerate bitstream if necessary, and it may take some time to finish.)

Directory location for hardware description files

C:\fpga\nexys4\example\SDK\SDK_Export

Figure A.11: Export to SDK/Launch SDK window.



Workspace Launcher

Select a workspace

Xilinx SDK stores your projects in a folder called a workspace.
Choose a workspace folder to use for this session.

Workspace: C:\fpga\nexys4\project

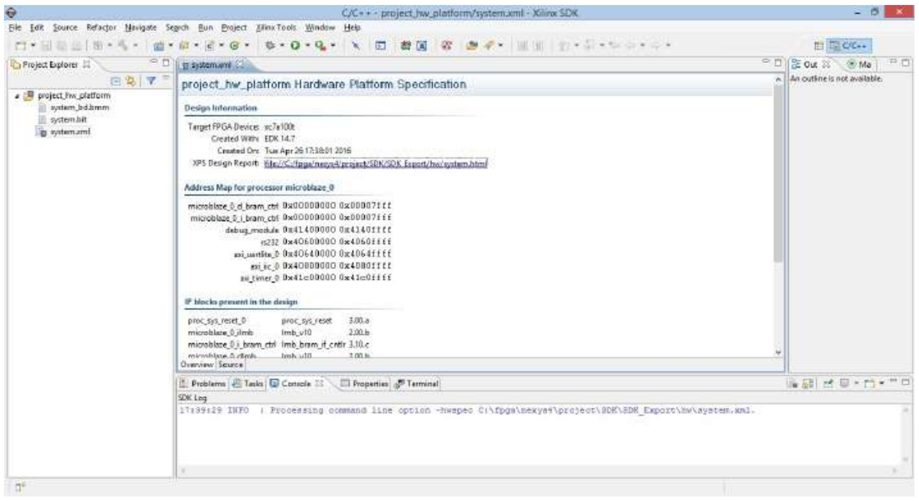
Figure A.12: Workspace Launcher window.

After this steps a new window will appear to implement the C program into the FPGA board.

Appendix B

Implementation of the C program

The new window is called **C/C++ - project_hw_platform/system.xml - Xilinx SDK**, Figure B.1.


C/C++ - project_hw_platform/system.xml - Xilinx SDK
File Edit Source Refactor Navigate Search Run Project Xilinx Tools Window Help
Project Explorer
project_hw_platform
system_bd.bmm
system.bit
system.xml
Outline is not available.

project_hw_platform Hardware Platform Specification

Design Information

Target FPGA Device: xc7a100t
Created With: EDK 14.7
Created On: Tue Apr 26 17:38:01 2016
XPS Design Report: file:///C:/fpga/nexys4/project/SDK/SDK_Export/hw/system.html

Address Map for processor microblaze_0

```
microblaze_0_d_bram_ctrl 0x00000000 0x00007fff
microblaze_0_i_bram_ctrl 0x00000000 0x00007fff
            debug_module 0x41400000 0x4140ffff
                   rs232 0x40600000 0x4060ffff
           axi_uartlite_0 0x40640000 0x4064ffff
                axi_iic_0 0x40800000 0x4080ffff
              axi_timer_0 0x41c00000 0x41c0ffff
```

IP blocks present in the design

```
proc_sys_reset_0          proc_sys_reset     3.00.a
microblaze_0_ilmb         lmb_v10            2.00.b
microblaze_0_i_bram_ctrl  lmb_bram_if_cntlr  3.10.c
microblaze_0_dlmb         lmb_v10            2.00.b
```

SDK Log

```
17:39:29 INFO  : Processing command line option -hwspec C:\fpga\nexys4\project\SDK\SDK_Export\hw\system.xml.
```

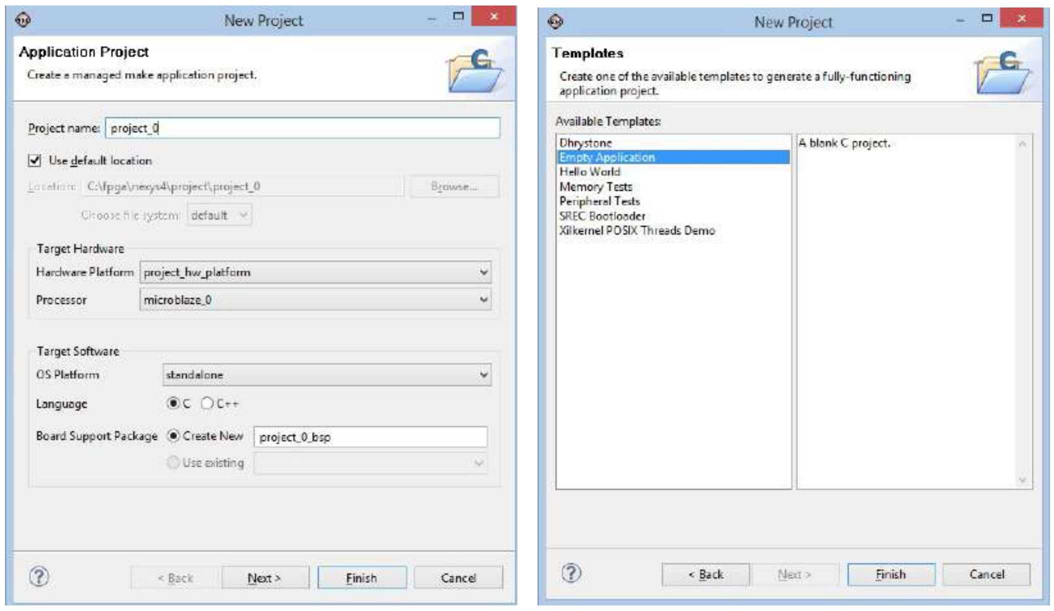
Figure B.1: Xilinx SDK main window.

The following steps describes the procedure to create and later to program the FPGA board with a program that acquire data from the IMU and the camera.

1.- Click on **File - New - Application Project**.

2.- The window to select the project features will appear, select a name for the project (preferably other one different than written in the storage folder) and click **Next**, Figure B.2(a).

The next part of the window is to choose the template to generate a fully-functioning application project. Select **Empty Application** and clock on **Finish**, Fugure B.2(b).


New Project

Application Project
Create a managed make application project.

Project name: project_0
☑ Use default location
Location: C:\fpga\nexys4\project\project_0
Choose file system: default

Target Hardware
Hardware Platform: project_hw_platform
Processor: microblaze_0

Target Software
OS Platform: standalone
Language: ● C ○ C++
Board Support Package: ● Create New project_0_bsp
○ Use existing

New Project

Templates
Create one of the available templates to generate a fully-functioning application project.

Available Templates:
Dhrystone
Empty Application
Hello World
Memory Tests
Peripheral Tests
SREC Bootloader
Xilkernel POSIX Threads Demo

A blank C project.

(a) (b)

Figure B.2: New Project window. a)Selecting the name and the lenguage of the programs. b)Selecting an empty application to crate multiple files.

3.- In the project explorer will appear two new files. Deploy the folder with the C. Right click on **src** folder.

Select **Next** and click on **File**

The new file will be the library to get access to the ROM memory within each sensor of the IMU.

Write **i2clib.h**

Click on **Finish**.

On the Project Explorer, doble click on **project_0 - src - i2clib.h**, wait a moment and a new lash will appear.

Write the code that is on Appendix D.

Save the changes in the file.

4- In the project explore, right click on **src** folder.

Select **Next** and click on **File**

The file created is the one that contain the main program. Write **main_project.c**

Click on **Finish**

Doble cick on **project_0 - src - main_project.c**

Write the code in the Appendix E.

Save all the changes.

5.- To program the FPGA with the programs, click on **Xilinx Tools - Program FPGA**

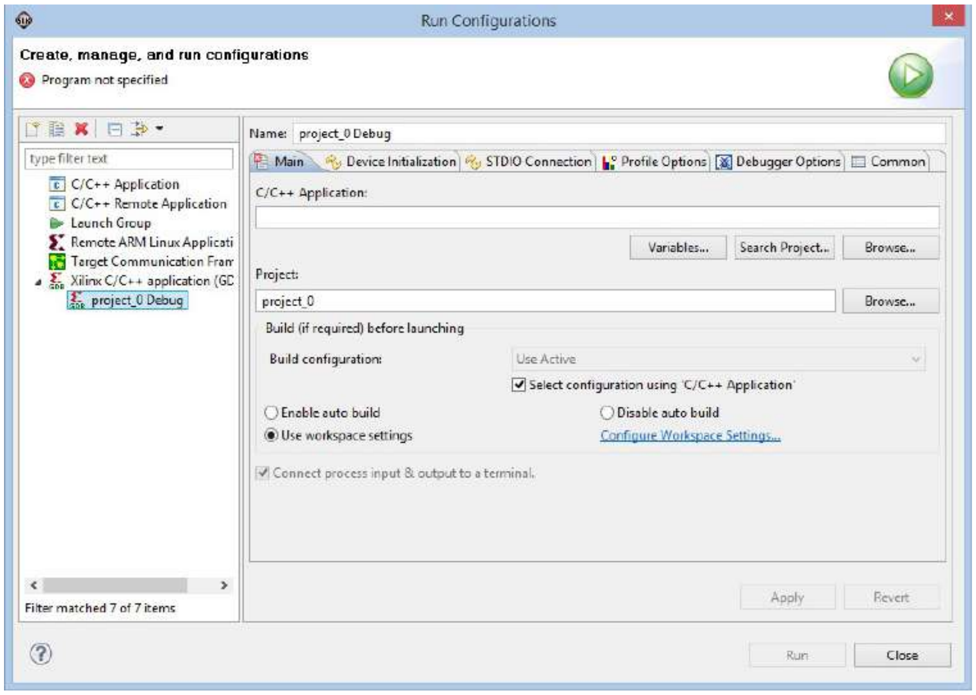
In the new window verify that the direction of the **system.bit** is the correct and click on **Program**.

To run the program, go to **Run - Debug Configuration** and a new window will appear, Figure B.3.

Select **project_0 Debug**.

Open the tab STDIO Connection.

If want to see the results in the console panel of Xilinx SDK enable the option **Connect STDIO to Console** choose the correct serial communication port and the baud rate correspondent. Click on **Apply**. Click on **Run**



Name: project_0 Debug

C/C++ Application:

Project: project_0

Build (if required) before launching

Build configuration: Use Active

[✓] Select configuration using 'C/C++ Application'

( ) Enable auto build ( ) Disable auto build

(●) Use workspace settings Configure Workspace Settings...

[✓] Connect process input & output to a terminal.

Figure B.3: Run configuration window.

Appendix C

Direction of the ports within the FPGA

# Clock signal

NET "clk" LOC = "E3" — IOSTANDARD = "LVCMOS33"; #Bank = 35, Pin name = IO_L12P_T1_MRCC_35, Sch name = CLK100MHZ

NET "clk" TNM_NET = sys_clk_pin;

TIMESPEC TS_sys_clk_pin = PERIOD sys_clk_pin 100 MHz HIGH 50%;

# Buttons

NET "RESET" LOC = "V10" — IOSTANDARD = "LVCMOS33"; #Bank = 14, Pin name = IO_L21P_T3_DQS_14, Sch name = BTND

# Pmod Header JA

NET "axi_uartlite_0_RX_pin" LOC = "D18" — IOSTANDARD = "LVCMOS33"; #Bank = 15, Pin name = IO_L21N_T3_A17_15, Sch name = JA9

NET "axi_uartlite_0_TX_pin" LOC = "E18" — IOSTANDARD = "LVCMOS33"; #Bank = 15, Pin name = IO_L21P_T3_DQS_15, Sch name = JA10

# Pmod Header JB

NET "axi_iic_0_Scl_pin" LOC = "T9" — IOSTANDARD = "LVCMOS33"; #Bank = 14, Pin name = IO_L24P_T3_A01_D17_14, Sch name = JB9

NET "axi_iic_0_Sda_pin" LOC = "U11" — IOSTANDARD = "LVCMOS33"; #Bank

= 14, Pin name = IO_L19N_T3_A09_D25_VREF_14, Sch name = JB10

# Pmod Header JC

NET “RS232_Uart_1_sin” LOC = “J2” — IOSTANDARD = LVCMOS33; #Bank = 35, Pin name = IO_L22N_T3_35, Sch name = JC9

NET “RS232_Uart_1_sout” LOC = “G6” — IOSTANDARD = LVCMOS33; #Bank = 35, Pin name = IO_L19P_T3_35, Sch name = JC10

Appendix D

Code reading of sensors EPROM

```
#include "xparameters.h"
#include "xutil.h"
#include "xiic.h"
#include "xiic_l.h"

writeI2C(u8 DISP_ADD, u8 reg_DISP, u8 data){
cu8 Data_Send[2];
    Data_Send[0] = reg_DISP;
    Data_Send[1] = data;
    XIic_Send(XPAR_AXI_IIC_0_BASEADDR, DISP_ADD, Data_Send, 2, XIIC_STOP);
};

readI2C(u8 DISP_ADD, u8 reg_DISP, u8 *Data_Rec, u8 lenght){
    u8 Data_Send[1];
    Data_Send[0] = reg_DISP;
    XIic_Send(XPAR_AXI_IIC_0_BASEADDR, DISP_ADD, Data_Send, 1, XIIC_STOP);
    XIic_Recv(XPAR_AXI_IIC_0_BASEADDR, DISP_ADD, Data_Rec, lenght, XIIC_STOP);
};
```

Appendix E

Main program

```
//Libraries
#include "xparameters.h"
#include "xutil.h"
#include "stdio.h"
#include "xtmrctr.h"
#include "xuartlite.h"
#include "xuartlite_l.h"
#include "i2clib.h"
#include "math.h"

//Global variables
#define icc_add          XPAR_AXI_IIC_0_BASEADDR
#define bar_add           0x77
const unsigned char OSS = 3;
short M_o_g[3], Measures_bmp[11], Measures_bma[4], Measures_itg[3], Measures_hmc[3];
short Ref[6];
long P[2], UT[2];
unsigned long UP[2];
int a=0x0000, k=0, count=0, EndFlag=0, StartFlag=0;
int ph[42];
Xuint8 MH,ML;

XIic IicInstance;
XTmrCtr DelayTimer;
void delay(Xuint32 time);
void SendResetCmd();
void SendTakePhotoCmd();
void SendReadDataCmd();
void SendReadCmd();        //Extra al Arduino
void SendImage640();       //Extra al Arduino
void SendImage320();       //Extra al Arduino
void StopTakePhotoCmd();
void init_bma();
void GetMeasures();
void PrintMeasures();
void References();

//Delay function in milliseconds
void delay(Xuint32 time){
   XTmrCtr_SetResetValue(&DelayTimer, 1, time * 100000);
   XTmrCtr_Start(&DelayTimer, 1);
   while (!(XTmrCtr_IsExpired(&DelayTimer, 1)))
   XTmrCtr_Stop(&DelayTimer, 1);
}

//Camera commands
void SendResetCmd(){
   XUartLite_SendByte(XPAR_AXI_UARTLITE_0_BASEADDR, 0x56);
   XUartLite_SendByte(XPAR_AXI_UARTLITE_0_BASEADDR, 0x00);
```

```
    XUartLite_SendByte(XPAR_AXI_UARTLITE_0_BASEADDR, 0x26);
    XUartLite_SendByte(XPAR_AXI_UARTLITE_0_BASEADDR, 0x00);
}

void SendTakePhotoCmd(){
    XUartLite_SendByte(XPAR_AXI_UARTLITE_0_BASEADDR, 0x56);
    XUartLite_SendByte(XPAR_AXI_UARTLITE_0_BASEADDR, 0x00);
    XUartLite_SendByte(XPAR_AXI_UARTLITE_0_BASEADDR, 0x36);
    XUartLite_SendByte(XPAR_AXI_UARTLITE_0_BASEADDR, 0x01);
    XUartLite_SendByte(XPAR_AXI_UARTLITE_0_BASEADDR, 0x00);
}

void SendReadData(){
    MH = a / 0x100;
    ML = a % 0x100;
    XUartLite_SendByte(XPAR_AXI_UARTLITE_0_BASEADDR, 0x56);
    XUartLite_SendByte(XPAR_AXI_UARTLITE_0_BASEADDR, 0x00);
    XUartLite_SendByte(XPAR_AXI_UARTLITE_0_BASEADDR, 0x32);
    XUartLite_SendByte(XPAR_AXI_UARTLITE_0_BASEADDR, 0x0C);
    XUartLite_SendByte(XPAR_AXI_UARTLITE_0_BASEADDR, 0x00);
    XUartLite_SendByte(XPAR_AXI_UARTLITE_0_BASEADDR, 0x0A);
    XUartLite_SendByte(XPAR_AXI_UARTLITE_0_BASEADDR, 0x00);
    XUartLite_SendByte(XPAR_AXI_UARTLITE_0_BASEADDR, 0x00);
    XUartLite_SendByte(XPAR_AXI_UARTLITE_0_BASEADDR, MH);
    XUartLite_SendByte(XPAR_AXI_UARTLITE_0_BASEADDR, ML);
    XUartLite_SendByte(XPAR_AXI_UARTLITE_0_BASEADDR, 0x00);
    XUartLite_SendByte(XPAR_AXI_UARTLITE_0_BASEADDR, 0x00);
    XUartLite_SendByte(XPAR_AXI_UARTLITE_0_BASEADDR, 0x00);
    XUartLite_SendByte(XPAR_AXI_UARTLITE_0_BASEADDR, 0x20);
    XUartLite_SendByte(XPAR_AXI_UARTLITE_0_BASEADDR, 0x00);
    XUartLite_SendByte(XPAR_AXI_UARTLITE_0_BASEADDR, 0x0A);
    a+=0x20;
}

void SendReadCmd(){
    XUartLite_SendByte(XPAR_AXI_UARTLITE_0_BASEADDR, 0x56);
    XUartLite_SendByte(XPAR_AXI_UARTLITE_0_BASEADDR, 0x00);
    XUartLite_SendByte(XPAR_AXI_UARTLITE_0_BASEADDR, 0x34);
    XUartLite_SendByte(XPAR_AXI_UARTLITE_0_BASEADDR, 0x01);
    XUartLite_SendByte(XPAR_AXI_UARTLITE_0_BASEADDR, 0x00);
}

void StopTakePhotoCmd(){
    XUartLite_SendByte(XPAR_AXI_UARTLITE_0_BASEADDR, 0x56);
    XUartLite_SendByte(XPAR_AXI_UARTLITE_0_BASEADDR, 0x00);
    XUartLite_SendByte(XPAR_AXI_UARTLITE_0_BASEADDR, 0x36);
    XUartLite_SendByte(XPAR_AXI_UARTLITE_0_BASEADDR, 0x01);
    XUartLite_SendByte(XPAR_AXI_UARTLITE_0_BASEADDR, 0x03);
}

void SendImage640(){
    XUartLite_SendByte(XPAR_AXI_UARTLITE_0_BASEADDR, 0x56);
    XUartLite_SendByte(XPAR_AXI_UARTLITE_0_BASEADDR, 0x00);
    XUartLite_SendByte(XPAR_AXI_UARTLITE_0_BASEADDR, 0x31);
    XUartLite_SendByte(XPAR_AXI_UARTLITE_0_BASEADDR, 0x05);
    XUartLite_SendByte(XPAR_AXI_UARTLITE_0_BASEADDR, 0x04);
    XUartLite_SendByte(XPAR_AXI_UARTLITE_0_BASEADDR, 0x01);
    XUartLite_SendByte(XPAR_AXI_UARTLITE_0_BASEADDR, 0x00);
    XUartLite_SendByte(XPAR_AXI_UARTLITE_0_BASEADDR, 0x19);
    XUartLite_SendByte(XPAR_AXI_UARTLITE_0_BASEADDR, 0x00);
```

```
}

void SendImage320(){
    XUartLite_SendByte(XPAR_AXI_UARTLITE_0_BASEADDR, 0x56);
    XUartLite_SendByte(XPAR_AXI_UARTLITE_0_BASEADDR, 0x00);
    XUartLite_SendByte(XPAR_AXI_UARTLITE_0_BASEADDR, 0x31);
    XUartLite_SendByte(XPAR_AXI_UARTLITE_0_BASEADDR, 0x05);
    XUartLite_SendByte(XPAR_AXI_UARTLITE_0_BASEADDR, 0x04);
    XUartLite_SendByte(XPAR_AXI_UARTLITE_0_BASEADDR, 0x01);
    XUartLite_SendByte(XPAR_AXI_UARTLITE_0_BASEADDR, 0x00);
    XUartLite_SendByte(XPAR_AXI_UARTLITE_0_BASEADDR, 0x19);
    XUartLite_SendByte(XPAR_AXI_UARTLITE_0_BASEADDR, 0x11);
}

//Get barometer and temperature measures
bmp085(short *Measures_bmp, long *UT, unsigned long *UP, long *P){
    u8 data_read[22];
    u8 read_data[2];
    u8 dataread[3];
    int ac1, ac2, ac3, b1, b2, mb, mc, md;
    unsigned int ac4, ac5, ac6;
    long b3, b5, b6, x1, x2, x3;
    unsigned long b4, b7;
    readI2C(0x77, 0xAA, data_read, 22);
    Measures_bmp[0] = ((short)data_read[1]) | ((short)data_read[0])<<8;     //ac1
    Measures_bmp[1] = ((short)data_read[3]) | ((short)data_read[2])<<8;     //ac2
    Measures_bmp[2] = ((short)data_read[5]) | ((short)data_read[4])<<8;     //ac3
    ac4 = ((short)data_read[7]) | ((short)data_read[6])<<8;     //ac4
    Measures_bmp[3] = ((short)data_read[7]) | ((short)data_read[6])<<8;     //ac4
    ac5 = ((short)data_read[9]) | ((short)data_read[8])<<8;     //ac5
```

```
Measures_bmp[4] = ((short)data_read[9]) | ((short)data_read[8])≪8;        //ac5
ac6 = ((short)data_read[11]) | ((short)data_read[10])≪8;        //ac6
Measures_bmp[5] = ((short)data_read[11]) | ((short)data_read[10])≪8;        //ac6
Measures_bmp[6] = ((short)data_read[13]) | ((short)data_read[12])≪8;        //b1
Measures_bmp[7] = ((short)data_read[15]) | ((short)data_read[14])≪8;        //b2
Measures_bmp[8] = ((short)data_read[17]) | ((short)data_read[16])≪8;        //mb
Measures_bmp[9] = ((short)data_read[19]) |((short)data_read[18])≪8;        //mc
Measures_bmp[10] = ((short)data_read[21]) | ((short)data_read[20])≪8;        //md
ac1 = Measures_bmp[0];
ac2 = Measures_bmp[1];
ac3 = Measures_bmp[2];
//ac4 = Measures[3];
//ac5 = Measures[4];
//ac6 = Measures[5];
b1 = Measures_bmp[6];
b2 = Measures_bmp[7];
mb = Measures_bmp[8];
mc = Measures_bmp[9];
md = Measures_bmp[10];
writeI2C(0x77, 0xF4, 0x2E);
delay(5);
readI2C(0x77, 0xF6, read_data, 2);
UT[0] = ((long)read_data[1]) | ((long)read_data[0])≪8;
writeI2C(0x77, 0xF4, (0x34 + (OSS≪6)));
delay((2 + (3≪OSS)));
readI2C(0x77, 0xF6, dataread, 3);
UP[0] = (((unsigned long)dataread[2]) | ((unsigned long)dataread[1])≪8 | ((un-
signed long)dataread[0]≪16)) ≫ (8-OSS);
x1 = (((long)UT[0] - (long)ac6)*(long)ac5) ≫ 15;
x2 = ((long)mc ≪ 11)/(x1 + md);
```

```
    b5 = x1 + x2;
    b6 = b5 - 4000;
    // Calculate B3
    x1 = (b2 * (b6 * b6) / 4096) / 2048;
    x2 = (ac2 * b6) / 2048;
    x3 = x1 + x2;
    b3 = (((((long)ac1)*4 + x3)<<OSS) + 2) / 4;
    // Calculate B4
    x1 = (ac3 * b6)>>13;
    x2 = (b1 * ((b6 * b6) / 4096)) / 65536;
    x3 = ((x1 + x2) + 2) / 4;
    b4 = (ac4 * (unsigned long)(x3 + 32768))>>15;
    b7 = ((unsigned long)(UP[0] - b3) * (50000>>OSS));
    if (b7 < 0x80000000)
        P[0] = (b7<<1)/b4;
    else
        P[0] = (b7/b4)<<1;
    x1 = (P[0] / 256) * (P[0] / 256);
    x1 = (x1 * 3038) / 65636;
    x2 = (-7357 * P[0]) / 65636;
    P[0] += (x1 + x2 + 3791) / 16;
}

//Initialize accelerometer
init_bma(){
    u8 data_read[2];
    writeI2C(0x40, 0x10, 0xB6);
    delay(10);
    writeI2C(0x40, 0x0D, 0x10);
    writeI2C(0x40, 0x37, 88);
    readI2C(0x40, 0x20, data_read, 1);
    writeI2C(0x40, 0x20, (data_read[0] & 0x0F));
    readI2C(0x40, 0x35, data_read, 2);
    writeI2C(0x40, 0x35, ((data_read[0]& 0xf1))—0x04);
}

//Get accelerometer measures
bma180(short *Measures_bma){
    u8 data_read[6];
    readI2C(0x40, 0x02, data_read, 6);
    Measures_bma[0] = ((short)data_read[0]) | ((short)data_read[1])≪8;
    Measures_bma[1] = ((short)data_read[2]) | ((short)data_read[3])≪8;
    Measures_bma[2] = ((short)data_read[4]) | ((short)data_read[5])≪8;
}

itg3200(short *Measures_itg){
    u8 data_read[6];
    readI2C(0x68, 0x1D, data_read, 6);
    Measures_itg[0] = ((short)data_read[1]) | ((short)data_read[0])≪8;
    Measures_itg[1] = ((short)data_read[3]) | ((short)data_read[2])≪8;
    Measures_itg[2] = ((short)data_read[5]) | ((short)data_read[4])≪8;
}

//Get magnetometer measures
hmc5883l(short *Measures_hmc){
    u8 Data_Read[6];
    readI2C(0x1E, 0x03, Data_Read, 6);
    Measures_hmc[0] = ((short)Data_Read[1]) | ((short)Data_Read[0])≪8;
    Measures_hmc[1] = ((short)Data_Read[3]) | ((short)Data_Read[2])≪8;
    Measures_hmc[2] = ((short)Data_Read[5]) | ((short)Data_Read[4])≪8;
```

```
}

GetMeasures(){
    bmp085(Measures_bmp, UT, UP, P); //Barometer
    bma180(Measures_bma); //Accelerometer
    itg3200(Measures_itg); //Gyroscope
    hmc5883l(Measures_hmc); //Magnetometer

    if (M_o_g[0] <= 0)
        Measures_itg[0] = Measures_itg[0] - M_o_g[0];
    else
        Measures_itg[0] = Measures_itg[0] + M_o_g[0];
    if (M_o_g[1] >= 0)
        Measures_itg[1] = Measures_itg[1] - M_o_g[1];
    else
        Measures_itg[1] = Measures_itg[1] + M_o_g[1];
    if (M_o_g[2] <= 0)
        Measures_itg[2] = Measures_itg[2] - M_o_g[2];
    else
        Measures_itg[2] = Measures_itg[2] + M_o_g[2];
}

PrintMeasures(){
    xil_printf("a%db%dc%dd%de%df%dg"
              "%dh%di%dj"
              "%dk%dl%dm"
              "%dn%do%dp"
              "%dq%dr%ds%dt%du%dv",
              Measures_bmp[4], Measures_bmp[5], Measures_bmp[9], Measures_bmp[10],
              UT[0], P[0],
              Measures_bma[0], Measures_bma[1], Measures_bma[2],
              Measures_itg[0], Measures_itg[1], Measures_itg[2],
              Measures_hmc[0], Measures_hmc[1], Measures_hmc[2],
              Ref[0], Ref[1], Ref[2], Ref[3], Ref[4], Ref[5]);
}

//Accelerometer and magnetometer references
References(){
    delay(500);
    bma180(Measures_bma);
    Ref[0] = Measures_bma[0];
    Ref[1] = Measures_bma[1];
    Ref[2] = Measures_bma[2];
    hmc5883l(Measures_hmc);
    Ref[3] = Measures_hmc[0];
    Ref[4] = Measures_hmc[1];
    Ref[5] = Measures_hmc[2];
}
//Main program
int main(void){
    //Local Variables
    int i, Status, x;
    int ans_rd[10], ans_rst[55], ans_tp[10], ans_stp[10];

    //Initialize delay's timer
    XTmrCtr_Initialize(&DelayTimer, XPAR_AXI_TIMER_0_DEVICE_ID);
    XTmrCtr_SetOptions(&DelayTimer, 1, XTC_DOWN_COUNT_OPTION);

    //Initialize i2c
    Status = XIic_Initialize(&IicInstance, XPAR_AXI_IIC_0_BASEADDR);
```

```
if (Status != XST_SUCCESS){
    return XST_FAILURE;
    xil_printf("\r\nFAIL \r\n");
}
else
    xil_printf("\r\nSuccess \r\n");
XIic_Start(&IicInstance);

init_bma();
writeI2C(0x68, 0x16, 0X1f);
writeI2C(0x1E, 0x02, 00);

//gyroscope's offset measure
for(x=0; x<=20; x=x+1){
    itg3200(Measures_itg);
    M_o_g[0]= M_o_g[0] + Measures_itg[0];
    M_o_g[1]= M_o_g[1] + Measures_itg[1];
    M_o_g[2]= M_o_g[2] + Measures_itg[2];
    delay(20);
}
M_o_g[0] = M_o_g[0] / 20;
M_o_g[1] = M_o_g[1] / 20;
M_o_g[2] = M_o_g[2] / 20;

References();

SendImage320();
xil_printf("v");
for(i=0;i<5;i++){
    ans_tp[i] = XUartLite_RecvByte(XPAR_AXI_UARTLITE_0_BASEADDR);
```

```
        if (i != 4){
            if (ans_tp[i]<=0x0F)
            xil_printf("0");
            xil_printf("%x", ans_tp[i]);
        }
        else{
            if (ans_tp[i]<=0x0F)
            xil_printf("0");
            xil_printf("%xcmd\r\n", ans_tp[i]);
        }
    }

    while(1){
        GetMeasures();
        PrintMeasures();
        //print("Reset Command\r\n");
        xil_printf("v");
        SendResetCmd();
        for(i=0;i<46;i++){
            ans_rst[i] = XUartLite_RecvByte(XPAR_AXI_UARTLITE_0_BASEADDR);
            if (i != 45){
                if (ans_rst[i]¡=0x0F)
                xil_printf("0");
                xil_printf("%x",ans_rst[i]);
            }
            else{
                if (ans_rst[i]¡=0x0F)
                xil_printf("0");
                xil_printf("%xcmd\r\n", ans_rst[i]);
            }
```

```
}

//4 sec delay
for(x=0; x<45; x++){
    GetMeasures();
    PrintMeasures();
    xil_printf("\r\n");
    delay(5);
}

GetMeasures();
PrintMeasures();
//print("Take Photo Command\r\n");
SendTakePhotoCmd();
xil_printf("v");
for(i=0;i<5;i++){
    ans_tp[i] = XUartLite_RecvByte(XPAR_AXI_UARTLITE_0_BASEADDR);
    if (i != 4){
        if (ans_tp[i]<=0x0F)
        xil_printf("0");
        xil_printf("%x", ans_tp[i]);
    }
    else{
        if (ans_tp[i]<=0x0F)
        xil_printf("0");
        xil_printf("%xcmd\r\n", ans_tp[i]);
    }
}

GetMeasures();
```

```
PrintMeasures();
//print("Read Command\r\n");
SendReadCmd();
xil_printf("v");
for(i=0;i<9;i++){
    ans_rd[i] = XUartLite_RecvByte(XPAR_AXI_UARTLITE_0_BASEADDR);
    if (i != 8){
        if (ans_rd[i]<=0x0F)
        xil_printf("0");
        xil_printf("%x", ans_rd[i]);
    }
    else{
        if (ans_rd[i]<=0x0F)
        xil_printf("0");
        xil_printf("%xcmd\r\n", ans_rd[i]);
    }
}

//print("Photo\r\n");
while(EndFlag==0){
    if(StartFlag==0){
        SendReadDataCmd();
        count=37;
        for(i=0;i¡42;i++)
            ph[i] = XUartLite_RecvByte(XPAR_AXI_UARTLITE_0_BASEADDR);
        delay(4);
        for (i=6;i<37;i++)
            if ((ph[i-1]==0xFF)&&(ph[i]==0xD8))
                StartFlag=1;
        GetMeasures();
        PrintMeasures();
        if(StartFlag==1){
            for (i=5;i<37;i++){
                if (ph[i]<=0x0F)
                    xil_printf("0");
                xil_printf("%x",ph[i]);
            }
        }
        xil_printf("cmd\r\n");
    }
    if(StartFlag==1){
        SendReadDataCmd();
        count=37;
        for(i=0;i<42;i++)
            ph[i] = XUartLite_RecvByte(XPAR_AXI_UARTLITE_0_BASEADDR);
        delay(4);
        for (i=6;i<37;i++)
            if ((ph[i-1]==0xFF)&&(ph[i]==0xD9)){
                StartFlag = 0;
                EndFlag = 1;
                count = i+1;
        }
        GetMeasures();
        PrintMeasures();
        for (i=5;i<count;i++){
            if (ph[i]<=0x0F)
                xil_printf("0");
            xil_printf("%x",ph[i]);
        }
        xil_printf("cmd\r\n");
        }
    }

    //4 sec delay
    for(x=0; x<45; x++){
        GetMeasures();
        PrintMeasures();
        xil_printf("\r\n");
        delay(5);
    }

    GetMeasures();
    PrintMeasures();
    //print("Stop Command\r\n");
    StopTakePhotoCmd();
    xil_printf("v");
    for(i=0;i¡5;i++){
        ans_stp[i] = XUartLite_RecvByte(XPAR_AXI_UARTLITE_0_BASEADDR);
        if (i != 4){
            if (ans_stp[i]<=0x0F)
                xil_printf("0");
            xil_printf("%x", ans_stp[i]);
        }
        else{
            if (ans_stp[i]<=0x0F)
                xil_printf("0");
            xil_printf("%xcmd\r\n", ans_stp[i]);
        }
    }
EndFlag=0;
```

```
        //4 sec delay
        for(x=0; x<45; x++){
            GetMeasures();
            PrintMeasures();
            xil_printf("\r\n");
            delay(5);
        }
    }
}
```

Bibliography

[1] Lin, Ching-Fang; *Modern Navigation, Guidance and Control Processing*; Prentice Hall; 1991.

[2] Van Riper, A. Bowdoin; *Rockets and Missiles: the life story of a technology*; Greenwood Technographies; 2004.

[3] Yanushevsky, Rafael; *Guidance of Unmanned Aerial Vehicles*; CRC Press; 2011.

[4] Tewari, Ashish; *Atmospheric and Space Flight Dynamics*; Birkhäuser; 2007.

[5] Romero, Rene; *Electrónica Digital y Lógica Programable*; Universidad de Guanajuato; 2007.

[6] V. Cook, Michel; *Flight Dynamics Principles*; Elsevier Ltd.; 2007.

[7] Pierce, John & Noll, A. Michel; *Señales: La Ciencia de las Telecomunicaciones*; Editorial Reverté, 2002.

[8] Trueba, José Luis; *Electromagnetismo, circuitos y semiconductores*; Dykinson; 2007.

[9] Bassett, R.; *To the Digital Age*; Baltimore: The Johns Hopkins University Press; 2002.

[10] Berlin, L.; *The Man Behind the Microchip*; New York: The Oxford University Press; 2005.

[11] Kilby, J. S.; *Invention of the Integrated Circuit*; IEEE Transactions in Electron Devices; 1976.

[12] Lécuyer, C.; *Making Silicon Valley: Innovation and the Growth of High Tech*; The MIT Press; 2007.

[13] Moore, G. M.; *The Role of Fairchild in Silicon Technology*; Proceedings of IEEE, Vol.86; 1998.

[14] Lojek, B.; *History of semiconductors diffusion engineering,*; 10th IEEE International Conference of Advanced Thermal Processing of Semiconductors; 2002.

[15] Misa, T. J.; *Military Needs, Commercial Realities, and the Development of the Transistor, 1948-58*; MIT Press; 1985.

[16] Smits, F. M.; *A History of Engineering and Science in the Bell System: Electronics Technology*; AT&T Bell Laboratories; 1985.

[17] Corona Ramírez, L. & Abarca Jiménez G; *Sensores y Actuadores, Aplicaciones con Arduino*; Grupo Editorial Patria; 2014.

[18] *EE4218 Embedded Hardware System*; Nacional University of Singapore.

[19] *Nexys4*^{TM} *FPGA Board Reference Manual*; Digilent; 2013.

[20] *Nexys4*^{TM} *PDM Filter Project*; Digilent; 2014.

[21] *Simple Microblaze Tutorial 2*; Xilinx Inc.; 2012.

[22] *Xilinx Device Drivers Documentation*; Xilinx Device Drivers; 2004.

[23] *Xilinx EDK Tutorial*; 2013.

[24] *EDK Concepts, Tools, and Techniques*, Xilinx, INC, 2009

[25] *Diseño de sistemas embebidos usando FPGAs*; CINVESTAV; 2010.

[26] Lee, Thomas H.; *The (Pre-) History of the Integrated Circuit: A Random Walk*; IEEE Solid-State Circuits Society Newsletter; 2007.

[27] Riordan, M. & Hoddeson, L.; *Crystal Fire: The Birth of the Information Age*; New York: W. W. Norton; 1997.

[28] "BMA180 data sheet"; Bosch Sensortech, Germany.

[29] "BMP085 data sheet"; Bosch Sensortech, Germany.

[30] "HMC5883L data sheet"; Honeywell; Plymouth, MN, USA.

[31] "ITG3205 data sheet"; InvenSense Inc.; Sunny Valley, CA, USA.

[32] "SEN11610 data sheet"; LinkSprite; Longmont, CO, USA.

[33] The Computer History Museum [Online]. Available: http://www.computerhistory.org/