Design, Development and Implementation of a UAV flight controller based on a State Machine approach using a FPGA embedded system

Noé Monterrosa
Instituto de Investigación e Innovación en Electrónica
Universidad Don Bosco
El Salvador, C.A.
noejose1992@gmail.com

Jason Montoya
Instituto de Investigación e Innovación en Electrónica
Universidad Don Bosco
El Salvador, C.A.
jmontoyaag1992@gmail.com

Fredy Jarquín
Instituto de Investigación e Innovación en Electrónica
Universidad Don Bosco
El Salvador, C.A.
fredyjarquin@gmail.com

Carlos Bran
Instituto de Investigación e Innovación en Electrónica
Universidad Don Bosco
El Salvador, C.A.
cbran@udb.edu.sv

***Abstract*— This article presents the development of a fixed-wing UAV flight controller using a complete parallelism embedded system as a FPGA. Many solutions for UAV´s flight controllers are based on embedded sequential systems. However, these systems are not perfect. The greater number of processes and tasks being executed simultaneously, the more variables such as precision, speed of response and synchronism may suffer. Our proposed flight controller solves this problem because it is based on a concurrent system and can therefore, execute many processes at the same time. The development of this flight controller represents just one part of the "Drone Bosco" project, where university students from Universidad Don Bosco are constructing the first UAV designed completely in El Salvador. The solution was designed and implemented taking into consideration specific characteristics of other areas of the project such as Radio Control Systems, Power Generation Systems and Aerodynamics. These considerations are outlined in this article. The flight controller is based on a state machine system that migrates from state to state depending on the stimulus received from sensors like accelerometers, tachometers, compass, pitot, GPS, etc. Another feature developed in this project is an emergency system that provides enough intelligence and robustness to secure the integrity of the aircraft in case a problem occurs during missions. Features like high speed of response, adaptable calibration and parallelism are achieved with our solution. Moreover, given that many parameters are generic, it has the flexibility to migrate to other fixed-wing UAVs with different characteristics. A similar approach could be applied in the future for the development of other devices that need navigation controllers with these characteristics, for example rockets or rovers. The results obtained in the simulations and tests of the flight controller system are described in detail in this article.**

***Index Terms* – UAV, FPGA, VHDL, Embedded system, PWM, Top Down, fixed wing, Finite State Machine.**

I. Introduction

The UAV's (Unmanned Aerial Vehicle) or drones as they are commonly known, are aircrafts which can be controlled remotely or fly autonomously. These devices have become very popular in recent years and can be used for several applications. United States' companies such as FedEx and UPS are even starting to use them to deliver packages. These companies are currently working with the government in order to regulate their usage in urban zones [1]. Around the world drones are being used to improve agricultural performance in what is called "Precision Agriculture". By using this technology farmers are able to know the state and health of their crops. The capability and autonomy of UAVs are proportional to the power of calculation of the Flight Controller System, these systems give judgment and guidance to the. The more complex these systems become the more independent from human control the aircraft becomes. The response speed and synchronism of the signals are important factors that influence the performance of the vehicle. Many solutions are based in embedded sequential systems, however the more processes these systems execute the greater impact it will have on variables such as precision, speed of response and synchronism. This article presents a solution for the UAV's Flight Controller integrated in an embedded concurrent system such as an FPGA. The FPGA provides enough power of calculation to extend the intelligence of the system with more autonomy, excellent synchronism and improved response times. The second part of the article describes the variables taken into account in making the design with Top-Down methodology and details the integration and interaction of the system's modules. The third part explains the implementation and functionality of the system, offering detailed information about all the components and tools that have been used to build the UAV's Flight Controller System. The fourth and fifth parts show the results obtained through experiments and the corresponding conclusions. .

II. Design of Concept

A Top Down design methodology was used for the conceptual design of the aircraft, with this methodology the full operation of the system can be described. Later, the system's design is divided in block units that can be worked and tested individually. This reduces debugging, prototyping and integration time. The UAV's development is divided into the following 4 areas:

- Propulsion and fuselage
- Energy generation
- Telecommunications and Video
- Flight Control

In the Propulsion and Fuselage area the dimensions of the electronic cards and equipment were taken into account so that they can fit inside of the aircraft's fuselage. A key point for the Flight Control Area is the aerodynamic design of the aircraft which will make it very stable in the air and will determine the speeds at which the maneuvers of the aircraft are taking place. Then, the Energy Generation Area provides all the electrical demands for the different electronic devices and circuits, this includes sensors and servomotors. Furthermore, this UAV will be powered by a brushless motor controlled with an ESC. Lastly, the area with the most influence in the Flight Controller design was Telecommunications and Video. The user and the UAV will communicate through RF. The UAV will follow the commands sent by the user and the telemetry will provide the user important information such as position, latitude, speed, etc. All incoming and outgoing data use a unique communication protocol.

The goal of the Flight Controller System is to help the user with automatic compensation in all the maneuvers of the aircraft in case of any disturbance. The inputs of the system are the signals delivered by the RF receiver which delivers the commands coming from the user and the sensor's signals which give the direction and position of the aircraft in real time. The outputs are connected to the actuators of the aircraft and to the emergency system. The block unit's design of the Flight Controller System with all its block units is shown in figure 1.

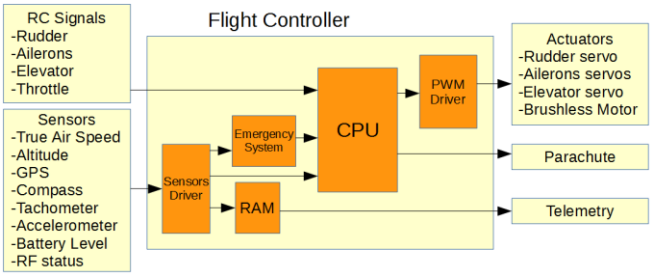


Fig. 1 Conceptual design of flight control system.

Each block unit is described in the next paragraphs.

A. Sensors

Their function is to indicate the Flight Control System about the condition of the aircraft. The data provided by sensors allows the controller to make the necessary adjustments in order to maintain the UAV in its route and detect possible malfunctions and hazards. These are the sensors used in this prototype:

- Accelerometer: This sensor dictates the current position of the aircraft regarding to pitch, yaw and roll axis.
- True air speed sensor: This sensor indicates the speed of the aircraft regarding air speed. The measurement of this variable is critical, if the speed is too low the UAV would go on stall. Stall would cause a severe decrease in the plane´s lift. This is a dangerous situation for the aircraft.
- Barometer: This sensor provides the current altitude of the UAV regarding sea level.
- GPS: This sensor provides the coordinates of latitude and longitude of the location of the aircraft in a map.
- Compass: This instrument, which is used for navigation and orientation, shows the direction of the aircraft regarding the cardinal points.
- Tachometer: This provides the revolutions per minute of the brushless motor in order to make the necessary compensations.
- Battery level sensor: This sensor shows the remaining power in the battery.
- Radio frequency sensors: The telecommunications and video area will send an alarm in case the radio controller signal is saturated with noise, out of synchrony or low power.

B. Sensor Driver

This block receives the data from the sensors and send it to the Central Processing Unit and the RAM Memory to be stored. Subsequently these stored signals will be sent through a RF transmitter to the user. Most of these sensors are connected to an ATMega processor, which receives the incoming data and send it to the Sensor Driver unit.

C. Central Processing Unit

This block controls the route of the aircraft according to the incoming commands from land. Besides transferring all the incoming commands to the actuators it makes all the necessary compensations to maintain the aircraft in route regardless of any disturbance, turbulence and gusts of wind. In order to do the compensations, the system uses all the data provided by the sensors. A Finite State Machine was needed in order to describe an abstract machine that represents all the possible conditions of the controller and its proper actions. This machine is in one state at a time, this state is known as "current state" and can make a transition into another state immediately after a previously defined condition changes.

The CPU starts with an initial state and monitors the input variables that can change the current state. These variables are set by the radio control (joysticks and switches values). With the data from the sensors and the commands from the user each state sets the UAV in the desired course and compensates the actuators. There are two types of compensations: the first type of compensation is based on the commands from the user, the

second type is based on theoretical pre-defined data. For the first case, if the user commands the aircraft to increase the pitch angle by 10 degrees, the incoming data cause the system to command the servomotors to rotate its lateral axis +10 degrees (this movement is verified by the accelerometer). The new direction of the UAV is maintained and compensated by the Flight Control System through the servomotors. The second compensation type works in a different way, for example, when the UAV is set in cruise mode, it maintains its course regarding pre-defined yaw, pitch and roll angles at an also pre-defined speed. The different states of the UAV, input conditions and output responses are detailed in table 1:

Code number	State	Inputs	Outputs
00	Takeoff-Free State	rst=0,se= 0	Actuators controlled by user with no compensation
01	Climb	rst=1 , sw1=0, sw2=0, ELE < 0 , AIL= 0,se= 0	Pitch controlled by user and compensated; Roll and speed compensated
02	Descent	rst=1 , sw1=0, sw2=0, ELE > 0 , AIL= 0,se= 0	Pitch controlled by user and compensated; Roll and speed compensated
03	Turn Right	rst=1 , sw1=0, sw2=0, ELE =0 , AIL>0 ,se= 0	Roll controlled by user and compensated; Pitch and speed compensated
04	Turn Left	rst=1 , sw1=0, sw2=0, ELE =0 , AIL<0 ,se= 0	Roll controlled by user and compensated; Pitch and speed compensated
05	Climb Right	rst=1 , sw1=0, sw2=0, ELE <0 , AIL>0 ,se= 0	Pitch and Roll controlled by user and compensated; Speed compensated
06	Climb Left	rst=1 , sw1=0, sw2=0, ELE <0 , AIL<0 ,se= 0	Pitch and Roll controlled by user and compensated; Speed compensated
07	Descent Right	rst=1 , sw1=0, sw2=0, ELE >0 , AIL>0 ,se= 0	Pitch and Roll controlled by user and compensated; Speed compensated
08	Descent Left	rst=1 , sw1=0, sw2=0, ELE >0 , AIL<0 ,se= 0	Pitch and Roll controlled by user and compensated; Speed compensated
09	Cruise 1 - Fixed Speed	rst=1, sw1=1, sw2=0,se= 0	Pitch, Roll and Speed compensated
10	Cruise 2 - Changeable Speed	rst=1, sw1=1, sw2=1,se= 0	Pitch and Roll compensated, Speed controlled by user and compensated
11	Emergency	se= 1 OR Emergency detected by sensors	Parachute and actuators emergency sequence activated

Table 1. CPU defined states, "sw1" and "sw2" are buttons switched by the user, "se" is the emergency state switch, "ELE" means elevator and "Ail" means aileron.

D. RAM Memory

Stores the telemetry data. These data are sent by the UAV´s transmitter, so that the user can receive real time information of the current aircraft's condition. This block is generic and possess the ability to be modified and expanded.

E. Emergency System

This block is in charge of detecting any event that could jeopardize the integrity of the aircraft and reacts executing a protocol were a parachute is released in order to help the UAV land safely. The alarms that trigger the emergency protocol are the following:

- Signal lost: This happens when the user's control signal is poor or has been lost.
- Lack of synchrony in RF signal: The signal sent by the user to the aircraft lacks of synchrony. Telecommunication and Video area designed a module to detect such event.
- Noisy signal: Noise in the signal that affects the aircrafts control.
- Critical Speed: Detected by the air speed sensor. If the aircraft´s speed is too low lift is lost. Data provided by sensors are compared with a critical theoretical value.
- Critical battery level: The emergency system compares the battery sensor´s signal with a theoretical critical value.
- Activated by the user: There will be a switch in the user's controller that will activate the emergency system if he thinks is necessary.

If any of the five emergency conditions are activated, a three seconds timer will be activated in the emergency system block. This time is a pre caution in case of possible sensor malfunction or misreading. If the variable that has been activated does not return to its normal value, a flag is set on high in the control unit by the emergency system module. This high value will cause the control unit to change its current state to emergency state.
The next sequence will be executed once the emergency state is on:

- Brushless motor off

- All servomotors change the control surfaces positions back to neutral position.
- Three seconds timer is activated in order to wait for the UAV's own aerodynamics stabilize it
- Parachute releasing mechanism is activated
- All servo motors are turned off

F. *PWM Driver*

This driver translates the binary data coming from the CPU into equivalent PWM signals. These signals control the servomotor's position and the brushless motor's speed, depending on the signal's duty cycle. Four identical modules were used for the servomotors controlling the ailerons, elevator and rudder. For the brushless motor, the configuration parameters of its ESC (Electronic Speed Controller) were different. These modules are generic and have been configured to work within the actuator's duty cycle range and frequency. This generic module can be used for any PWM controlled device. The operation of this module is described by the author in [2].

III. Implementation

The devices used to implement the design are the following:

A. *ARTIX–7 35T FPGA*

A FPGA (Field Programmable Gate Array), a concurrent embedded system, was used to develop the Flight Control System. The programming of this device was made with VHDL, a hardware descriptive language. This type of programming language describes the behavior of an electronic circuit or system. Among the advantages of using a FPGA with VHDL for our system include the following:

- Short prototyping time and code portability: This language defines the structure, design and operation of electronic circuits. A formal description of the circuit is obtained simplifying its automated analysis and simulation. Hardware description language allows engineers to design from the concept level to the circuit level, testing every phase until the final implementation, reducing prototyping time and costs while ensuring the portability to other architecture.
- Handles a number of different processes simultaneously: The code declarations are concurrent unlike the sequential embedded systems. This characteristic is very important for our Flight Control System because it must be able to execute different tasks at the same time such as reading the sensor's data and the user's commands, compensating the actuator's outputs and monitoring the critical variables that trigger the emergency system.
- Lower response time: The FPGA can process much more information simultaneously than embedded sequential systems. Also, the internal clock of our FPGA works at 100 MHZ, which provides the Flight Control System with the capability to react to external stimulus very rapidly and keep the aircraft in course.

The FPGA used for our system is the Artix-7 XC7A35T-L1CSG324I from Xilinx, on a development board Artix-35T that has the following characteristics:

- Four Pmod interfaces (32 I/O)
- Switches, Buttons, RGB LEDs
- Arduino/ChipKit "shield" connector (49 I/O)
- On-chip analog-to-digital converter (XADC).
- Programmable over JTAG and Quad-SPI Flash
- 256 MB DDR3L with a 16-bit bus @ 667 MHz
- 16 MB Quad-SPI Flash
- 10/100 Mb/s Ethernet
- USB-UART Bridge

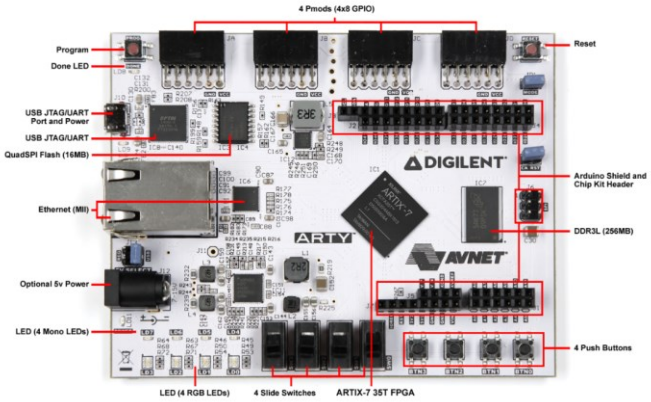


Fig. 2 ARTIX-7 35 T Development Board

B. *ATmega328P Processor*

The response time of the sensors is relatively high compared to the speed of the FPGA, for this reason a sequential processor was used to read the telemetry. Consequently, there was no need to develop the modules with the communication protocols that the sensors use (I2C, SPI). Instead, the drivers that read the data from the different sensors were already developed by the Open Source community and were easily modified and adapted to our system. The usage of these drivers reduced the implementation time and facilitated the processing of the telemetry data.

The correct operation of each sensor was corroborated, then all individual drivers were integrated in a program that uses a customized parallel communication protocol to facilitate the interaction with the FPGA. This program first executes reading functions for each sensor and stores the data in independent variables. Then, the communication between the ATmega328P and the FPGA begins loading all the data in the processor's Port B, with a 4 bit address to differentiate which sensor the data belong to. Lastly an enabler is activated to synchronize the

moment when the FPGA read the data. Figure 3 represents the temporal communication process diagram:

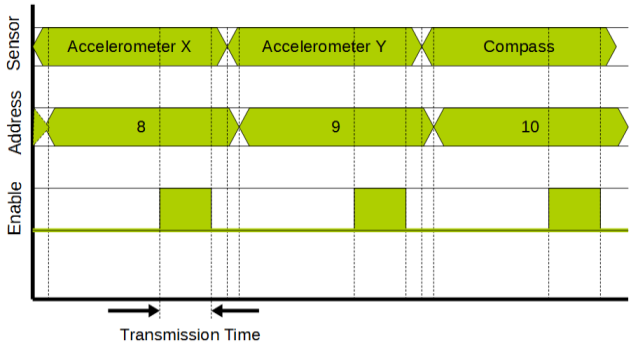


Fig. 3 Diagram of ATmega328 – FPGA temporal communication process diagram

C. *Optocouplers*

For the galvanic isolation between the electronic boards and the actuators (servomotors and brushless motor) an optic coupling circuit was designed. The electric diagram for one of the outputs is presented in figure 4:

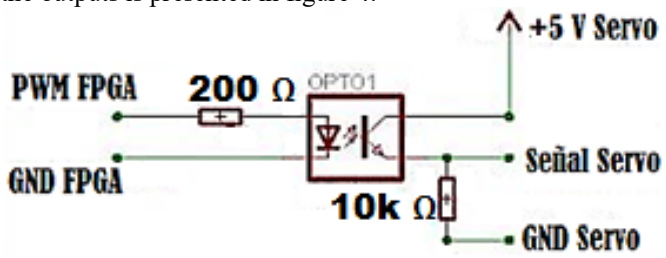


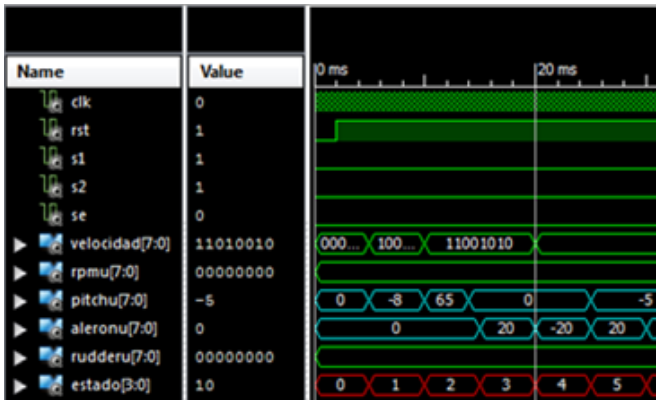
Fig.4 FPGA-actuators connection circuit

IV. TESTS

To corroborate the correct operation of the implemented solution the following tests were made:

A. *Tests and simulations of CPU*

VIVADO Design Suite 16.1 software was used to simulate and check the correct operation of the CPU module. Correct transitions between states and the corresponding outputs were verified. The inputs were simulated in the software and the following chart was obtained:



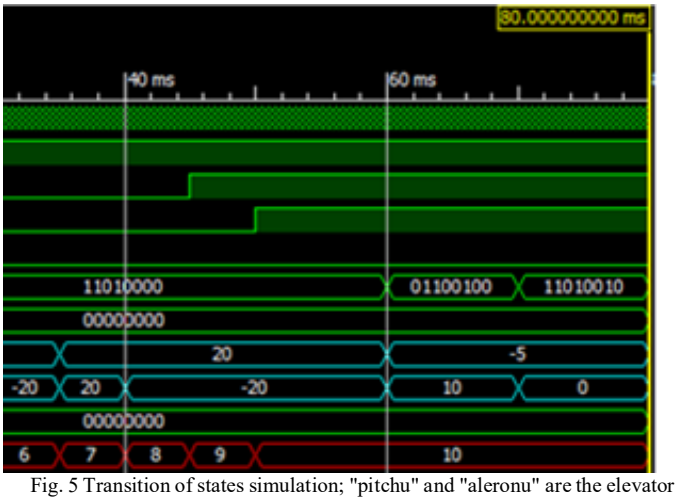


Fig. 5 Transition of states simulation; "pitchu" and "aleronu" are the elevator and aileron values sent by the user respectively,"estado" indicates the current state of the Finite State Machine in the CPU

As seen in the previous figure, the CPU changes the current state according to the inputs and the logic presented in Table 1. The correct compensation in each state was also checked. In order to do this, the accelerometer sensor was used and a test module was created and instantiated to simulate the external outputs from the user. The output signals from the Flight Control System were verified with an oscilloscope and the Duty Cycle time of the PWM signals for the servomotors was recorded.

The formulas used to make the compensation through the servomotors are the following:

Compensation with user's commands:

$$Output = \big(UserValue + (UserValue - Accelerometer)\big) + 90 \quad (1)$$

Compensation with pre-established data:

$$Output = (ExpectedOutput - Accelerometer) + 90 \quad (2)$$

All the variables in the formulas are in degrees. A 90 degree offset is added to the formulas because 90 degrees is the central position of the servomotors from which they move to positive or negative values. The value from the formulas is then transformed by the PWM Driver module in an equivalent PWM signal sent to the servomotors. These actuators, which move the control surfaces of the aircraft, use a 50 Hz signal with a Duty Cycle ranging from 0.8 ms (0 degrees) to 2.6 ms (180 degrees), these specifications were taken into account to shape the PWM signal. The inputs, compensation degree values, equivalent Duty Cycle time, current state and the oscilloscope data are presented in the following tables:

Compensation User Values					
User Value	Accel. Value	Output	Equivalent Time HIGH [ms]	Oscilloscope Time HIGH [ms]	State
15	0	120	1.736	1.7319	3
15	3	117	1.7126	1.7119	3
15	-37	157	2.0246	2.0199	3
8	-9	115	1.697	1.699	3
8	18	88	1.4864	1.4799	3
-3	-15	99	1.5722	1.56	6
-5	-21	101	1.5878	1.58	6
-5	-10	90	1.502	1.5	8
-9	-28	100	1.58	1.56	8

Table 2. Compensation with user's commands

Compensation with pre-defined values				
Accel. Value	Output	Equivalent Time HIGH [ms]	Oscilloscope Time HIGH [ms]	State
12	78	1.4084	1.399	3
-11	101	1.5878	1.6	3
-17	107	1.6346	1.636	4
16	74	1.3772	1.372	4
13	77	1.4006	1.4	4
5	85	1.463	1.464	1
2	88	1.4864	1.484	1

Table 3. Compensation with pre-established data

Multiple conditions were tested to guarantee the accuracy of the Flight Control System, the tables show correct compensation with both formulas in every state.

B. *Emergency System Simulation*

The system consists of two modules, one of them detects the emergency condition and sends an alarm to the CPU so it can make a transition from its current state to the emergency state. The other module detects the alarm, disables the servomotors and activates the parachute in accordance with the emergency protocol. The simulation of the first module is presented in figure 6:

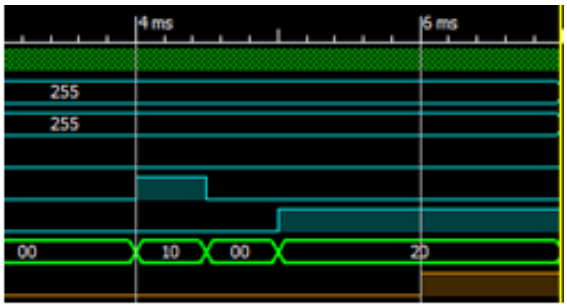


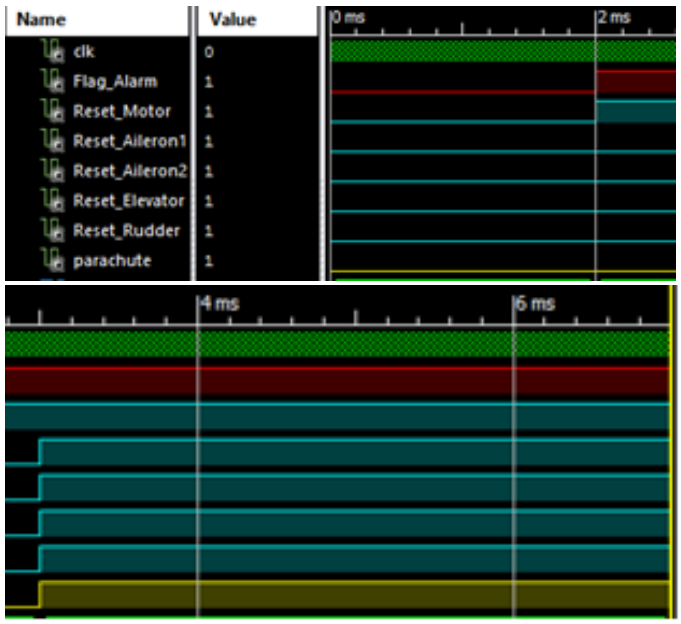
Fig.6 First module of Emergency System simulation

All the input values that trigger the alarm are in cyan color. When any of the bit inputs sent by the Telecommunications and Video Area are set HIGH or the values of Battery or RPM are below the minimum permitted, the system waits a short period of time for the signal to normalize, if it doesn't the module sets the alarm flag at HIGH and sends it to the CPU. The alarm will maintain its value even if the condition that has activated it changes. Also, all the conditions that trigger the alarm are stored inside a register in the RAM memory and later sent to the user so he can know in real time which condition triggered the emergency system.

The CPU receives the alarm signal and immediately changes its current state to the emergency state, in this state the actuators are set to the neutral position and no further compensation is made. The alarm is also detected by the second module of the emergency system, which turns off the motor, waits some time for the aerodynamics of the aircraft to help stabilize it, then turns off the actuators and finally activates the parachute releasing mechanism. A simulation of this module is shown in figure 7:

Fig. 7 Second module of Emergency System simulation

C. *Test on Telemetry sensors*



The purpose of this test was to corroborate the correct operation of some of the sensors used in the UAV, specifically the GPS and altitude sensors. We traveled a route by car with the system and sensors. The data stored in the RAM Memory were shown through the RGB LEDS in the FPGA development board and compared to the data than an Android App provided (This app is "GPS TEST"). The data from both sources were registered in intervals of 2 minutes simultaneously. Later, the data from our sensors were compared with the data provided by the app. The route points obtained are shown in figure 8 and figure 9:

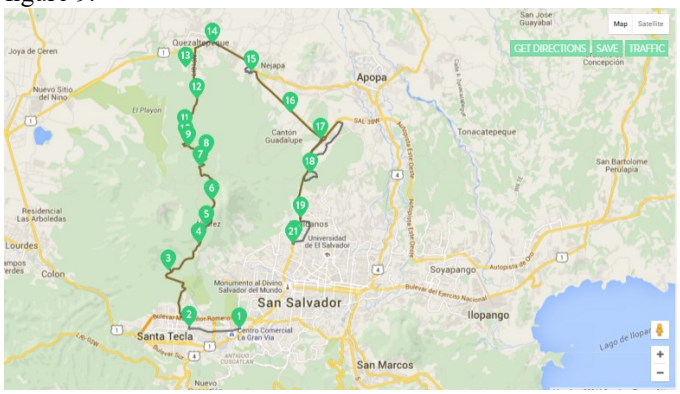


Fig.8 Route points taken from Android app.

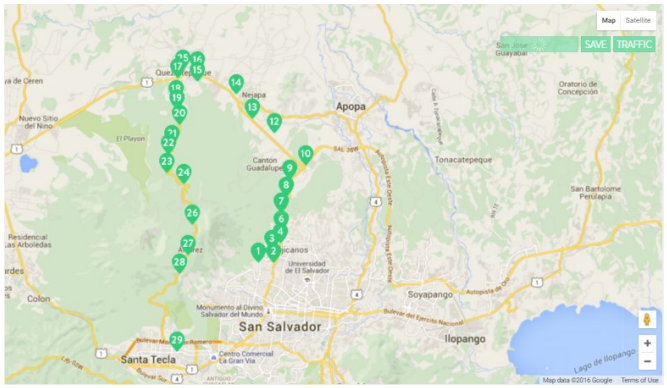


Fig. 9 Route points taken from the Flight Control System's Ram Memory

As shown above, both routes are very similar; 70% of the points taken from the RAM Memory have a corresponding Android App point. Also, the percentage of variation in the measurements of altitude between both systems is an average of 7%.

V. Conclusions

The Top-Down design methodology accelerates the prototyping time of any solution, helps to separate the functional modules so they can be tested individually and guarantees successful results. It also provides the capability to reuse, modify and integrate the existing modules with other designs. The implementation of the design in an FPGA represents a level of chip usage of 10% of its LUTs (Lookup Tables) and less than 1% of its Flip Flops, and small percentages due to the synthesis tools that optimize the use of the area of the chip. This results in reduced power consumption and increased processing speed.

In the solution proposed the integration between different embedded systems technologies was proven. This integration reduced the development time taking advantage of the benefits of each system and provided a solution within the parameters set in the design phase. The response times obtained with our solution are shorter than the ones obtained with fully sequential embedded systems, this is because of the capability of our system to execute different processes in a parallel way. Also, the compensation method is very practical considering that our UAV has been designed for reconnaissance and surveillance missions where great maneuverability is not necessary, instead what is demanded is to keep the course set by the user. Moreover, the Flight Control System has been designed to help the user fly the UAV in an easy way. Another important characteristic of our solution is the incorporation of an Emergency System which will allow the UAV to land in a secure way if any flaw/error is presented. This characteristic represents an innovation in comparison to the drones sold commercially.

VI. References

[1] A. Oppenheimer, ¡Crear o Morir!, Buenos Aires: Debate, 2014.

[2] N. Monterrosa and C. Bran, "Design and implementation of a motor control module based on PWM and FPGA for the development of a UAV flight controller," in *2015 CHILEAN Conference on Electrical, Electronics Engineering, Information and Communication Technologies (CHILECON)*, Santiago of Chile, 2015.

[3] P. Volnei A., Circuit Design with VHDL, Cambridge: MIT Press, 2004.

[4] Pololu Robotics & Electronics, «Power HD High-Torque Servo 1501MG,» 2015. [En línea]. Available: https://www.pololu.com/product/1057. [Último acceso: 15 Septiembre 2015].

[5] Y.-Y. T. Hsung-Hao Hsu, «FPGA control and implementation of a multiphase-interleaved PWM inverter for a segmented PMSM,» *Power Electronics and Drive Systems (PEDS), 2015 IEEE 11th International Conference,* pp. 224-230, 2015.

[6] H. V. H. R. Jainesh M.Patel, «Simulation and Analysis of Brushless DC Motor Based on Sinusoidal PWM Control,» *INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH IN ELECTRICAL, ELECTRONICS, INSTRUMENTATION AND CONTROL ENGINEERING,* vol. II, 2014.

[7] A. I. Carmona, Aerodinámica y Actuaciones del Avión, Madrid: PARANINFO, 2004.

[8] Y. Xu, J. Zhao y J. Huang, «Multiple linear motor control system based on FPGA,» *Electrical Machines and Systems (ICEMS), 2014 17th International Conference,* pp. 2327 - 2331, 2014.

[9] W. F. Phillips, Mechanics of Flight, Hoboken, New Jersey: WILEY, 2010.

[10] R. Austin, Unmanned Aircraft Systems UAVS Design, Development and Deployment, United Kingdom: WILEY, 2010.

[11] F. K. S. S. W. W. D. N. Kenzo Nonami, Autonomous Flying Robots: Unmanned Aerial Vehicles and Micro Aerial Vehicles, Springer, 2010.

[12] D. J. P. Steven F. Barrett, Atmel AVR Microcontroller Primer: Programming and Interfacing, Second Edition, Morgan & Claypool Publisher, 2012.

[13] S. Chattopadhyay, Embedded System Design, 2nd ed, PHI, 2013.

[14] K. Moudgalya, Digital Control, Wiley-Interscience, 2008.

[15] G. F. L. Israel Lugo-Cárdenas, «The MAV3DSim: A Simulation Platform for Research, Education and Validation of UAV Controllers,» de *19th World Congress The International Federation of Automatic Control, vol.*, Cape Town, South Africa, 2014.

[16] M. M. H. A. F. M. K. S. G. Atheer L. Salih, «Flight PID controller design for a UAV quadrotor,» de *Scientific Research and Essays Vol. 5(23), pp. 3660-3667*, 2010.

[17] J. Young y P. Andrew, «FPGA Based UAV Flight Controller,» de *AIAC-11 Eleventh Australian International Aerospace Congress*, Melbourne, 2005.