UKACC International Conference on Control 2012
Cardiff, UK, 3-5 September 2012

Development of an autopilot system for rapid prototyping of high level control algorithms

Matthew Coombes, Owen McAree, Wen-Hua Chen, Peter Render
Department of Automotive and Aeronautical engineering
Loughborough University
Loughborough, LE11 3TQ UK
Email: {ttmjc2@lboro.ac.uk, ttom@lboro.ac.uk, w.chen@lboro.ac.uk, p.m.render@lboro.ac.uk}

***Abstract*—This paper describes the development of a system for the rapid prototyping of high level control algorithms using an Arduino based commercial off the shelf autopilot called ArduPilot. It is capable of controlling multiple vehicle types, including fixed, and rotary wing aircraft as well as ground vehicles. The inner loop control is performed by ArduPilot, so the high level control can be rapidly prototyped and tested in Simulink, or an embedded system. The ability to conduct tests in software and hardware in the loop has also be developed, to enable safe testing of algorithms, which will speed up the development process. To show its functionality and ability to assist with the development process of algorithms, ArduPilot is used with a remote controlled aircraft in simulation and in real world testing to verify newly developed high level algorithms for UAVs.**

***Index Terms*—Autopilot; rapid prototyping; algorithm development; hardware in the loop; software in the loop.**

I. Introduction

It is important to be able to test high level algorithms, in simulation first, and then eventually in the real world. To be able to test high level control, a system needs to be in place for this to be tested on. The advantage of performing Software In the Loop (SIL) and then Hardware In the Loop (HIL) testing before performing real world testing is widely recognized as much cheaper, less time consuming, and safer [1]. A system that allows a seamless transition from one stage to the next without reprogramming, or reconfiguring any part of the system would hugely reduce the time from conception to the final design. This paper describes a system for developing high level autonomous control functions of UAVs.

Developing a full autopilot system from scratch is difficult, as specialist knowledge is needed in electronic engineering, systems integration, and software engineering. By purchasing an open source Commercial Off The Shelf (COTS) autopilot system much time and effort was saved. ArduPilot [2] is a hobbyist "do-it-yourself" autopilot meant for use on remote controlled aircraft. It is capable of flying simple waypoints, or more importantly taking roll (ϕ), pitch (θ), rudder, and throttle commands from an external source. By using a communication protocol called MAVLink [3], any device, or system can give ArduPilot these commands over a number of different connection types. As the low level control is performed by this autopilot, it leaves the user the freedom to concentrate on the high level control.

This paper describes the ArduPilot autopilot, and the system developed to control it. SIL, and HIL testing is discussed, as well as the X-Plane simulation environment used in this testing. The use of Simulink, and embedded systems to perform high level control is described. An example is given to illustrate the full development cycle from SIL to HIL to real world testing, where a simple PID waypoint tracking algorithm is developed and tested safely and easily.

II. ArduPilot

ArduPilot is a Arduino [4] based autopilot designed for the use by remote controlled aircraft hobbyists. It is designed to control fixed wing aircraft, and various rotary wing platforms, including single, tri, quad, hexa, and octa copters. The software is written in C++, and is completely open source with an active development community. The system enables the user to fly a series of predefined waypoints using a simple cross track error trajectory following algorithm. Or the user can fly the aircraft on a Fly by Wire (FBW) mode, where pitch and roll angles are commanded over the transmitter using ArduPilots inner loop instead of directly commanding servos. ArduPilot can communicate with a ground control station, where data can be gathered, waypoints, or even control gains can be updated. It communicates over a wireless serial connection, using a communication protocol called MAVLink [**?**], which was designed specifically for Micro Aerial Vehicles (MAVs).

The system hardware consists of two circuit boards shown in Fig. 1. The lower board has an ATmega2560 processor which runs the software, a failsafe multiplexer (that means there is always a hardware override if the processor fails) and all servo and receiver connections. The upper board houses the sensors, and telemetry ports. The sensors include a triple axis accelerometer, a dual and single axis gyro, barometric pressure sensor, triple axis magnetometer, also an externally connected GPS module. It is also capable of having an external pitot static sensor, for airspeed measurements. The full system on an aircraft set up for flight testing is shown in Fig. 2.

The ATmega2560 processor is relatively slow and only capable of executing 256 Kb of code. The autopilot software is written in C++ which is not that accessible a programming language. In its default state it is not an effective system for research.

978-1-4673-1558-6/12/$31.00

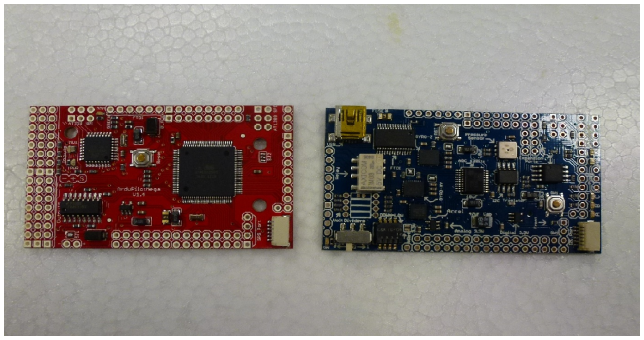

Fig. 1. Left: ArduPilot ATmega2560 processor board Right: IMU board

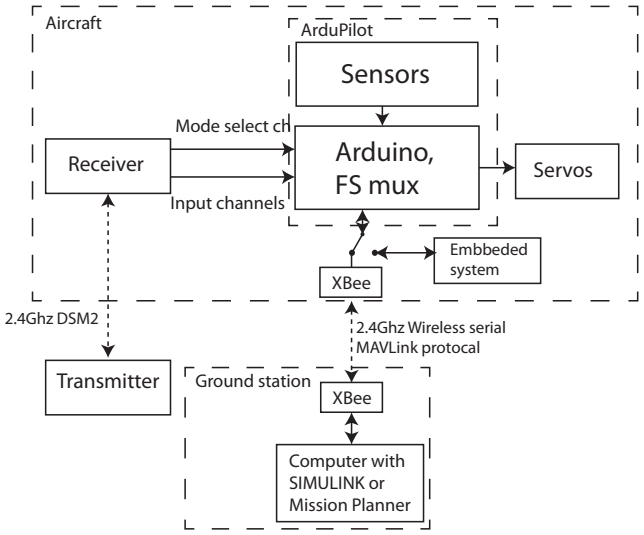

Fig. 2. Flow diagram for ArduPliot on and aircraft ready for flight testing

There are many examples of autopilot systems developed by research organisations, to assist with their research goals. They are often made in house at considerable expense, and many run on embedded systems making for high development times [1], [5], [6], and [7]. There are other autopilots available, like Paparazzi [8], or MicroPilot [9]. These systems are expensive, they are not easy to manipulate and interface with, and by no means as cross platform compatible. Paparrazi has no standard hardware as the schematics are open source [10], requiring one to be specially ordered, or made in house. MicroPilot is a very small open architecture autopilot, which has an integrated Inertial Measurement Unit (IMU) and GPS. Although all the parameters and control gains can be altered, the code is closed source and can not be altered. An autopilot system meant for hobbyists is Attopilot [11], this system is quite expensive and is quite dated. AttoPilot uses thermopiles for attitude control in instead of the much more accurate and reliable IMU [12].

III. System Configuration

ArduPilot has a number of different operating modes. The outer loop waypoint following mode (auto), the inner loop pitch, and roll angle hold (FBW), and manual override. If roll

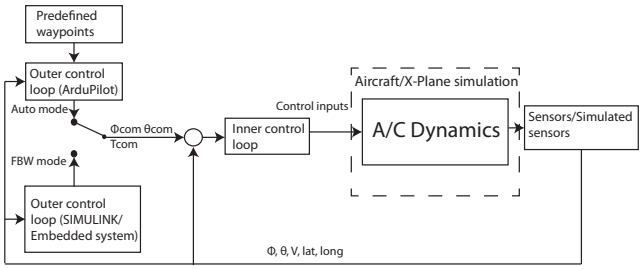

Fig. 3. Flow diagram of the control structure on the whole system

and pitch commands are sent to ArduPilot from an external source while in FBW mode, the higher level control loop can be performed on any platform of the users choice that is able to communicate over a serial connection. This gives huge flexibility to the user. As the ATmega2560 is simply not fast enough, and does not have enough storage, an external system to run more advance, more demanding code is needed.

The external system that this system has been primarily designed for is Simulink. A program that is used and understood by many engineers. It enables rapid prototyping of algorithms, and has an accessibility that is simply not offered when doing the same on an embedded system where the user must be familiar with C or C++.

The layout of the control system is shown in Fig. 3. ArduPilot's Attitude Heading Refrence System (AHRS) conducts sensor fusion and transmits the data over its serial telemetry port encoded in MAVLink format. A Simulink block has been developed for communication over MAVLink, it receives and decodes MAVLink messages, Simulink conducts high level control on the block outputs, to give pitch, roll angle, and throttle commands. The block then encodes these commands and resends them over the same serial connection. To enable wireless serial communication for flight tests, XBee wireless modules are used. As data and control signals are being transmitted wirelessly, this system relies on the wireless link to have high data integrity and low latency.

An embedded system can be used in place of Simulink if the wireless data link is a concern. By simply using the MAVLink wrapper discussed in Section V an embedded system can do exactly the same job as Simulink by directly connecting its serial port to that of ArduPilot's. This enables all the control to be performed on board the aircraft but over two components, mitigating XBee signal issues, also enabling a higher data rate.

IV. Software and Hardware in the Loop

Before performing real world testing, it is important to verify code functionality, which is commonly done through SIL, and then HIL testing. Putting algorithms through the full development cycle of SIL, HIL, and then flight testing is a good systematic debugging method, which significantly reduces risk. This process can be shown in Fig. 4. SIL enables any bugs in the software, to be ironed out, then HIL brings to light bugs which come about due to the software's interaction with the hardware, and the wireless communication

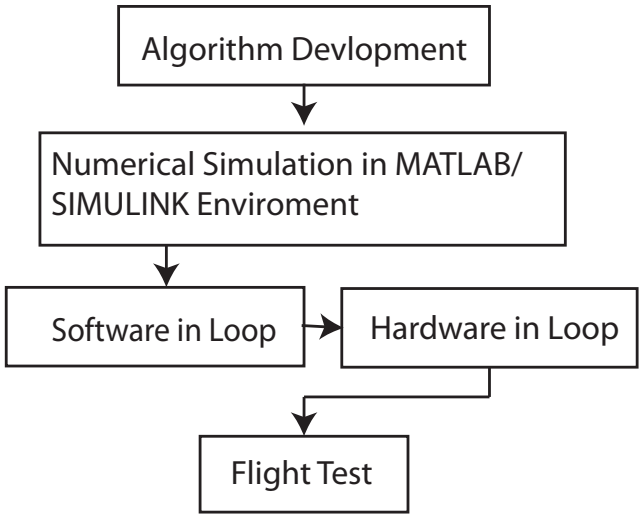

Fig. 4. The development cycle for high level control for UAS

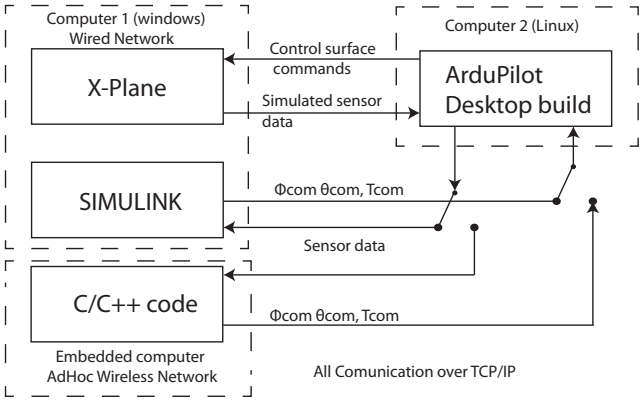

Fig. 5. Flow diagram for ArduPliots SIL testing, use with Simulink or and embedded system

system. Finally fight testing shows the functionality of the communications system at range, and issues the system has in the real environment, like wind.

The SIL system is shown in Fig. 5. The open source ArduPilot code has a desktop build, which means that, instead of the code being compiled on the Arduino on the ArduPilot, it is complied on a normal desktop computer running LINUX. This simulated ArduPilot communicates with a simulated aircraft in the X-Plane flight simulator using a plugin that enables communication over a TCP/IP network connection using the MAVLink Protocol. The X-Plane aircraft model is controlled by the desktop build of ArduPilot, and in turn ArduPilot is controlled by an external system like Simulink or an embedded system running the users high level control algorithms. This SIL method enables development to be undertaken without any ArduPilot hardware.

HIL (shown in Fig. 6) is much the same, but the actual ArduPliot hardware is used and sensor data is faked. The communication between X-Plane and ArduPilot is now done over a virtual serial connection, provided by the USB to

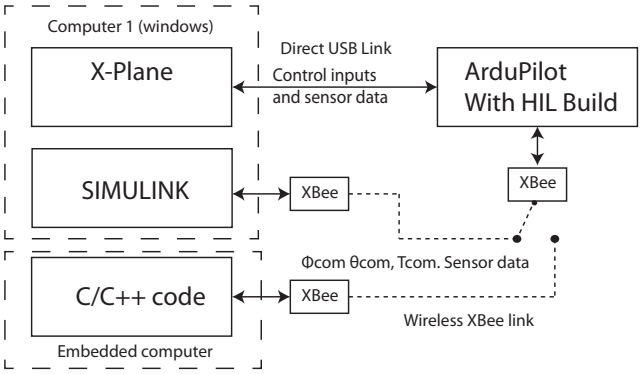

Fig. 6. Flow diagram for ArduPliots HIL testing, use with Simulink or and embedded system

serial chip. ArduPilot is now controlled over its telemetry port connected to Simulink, or a ground station using a XBee wireless serial transmitter, or to an embedded system directly.

When it comes time to conduct actual flight tests there is no need for reconfiguration of software hardware or communications, apart from ArduPilot needs to be mounted to the aircraft, and wired to the receiver and servos. As ArduPilot and MAVLink have been abstracted to such a high level, throughout the whole development cycle none of the ArduPilot code, or high level control ran externally needs to change at all to move to the next stage. All that changes is if the aircraft is real or simulated, where ArduPilot software is ran, and how the components of the system physically communicate with one another.

V. Communication

As has been mentioned previously, the communications between all components of the system use a protocol called MAVLink. MAVLink is a very lightweight, header-only message marshalling library for MAVs, in C/C++. It encodes data structures into high efficiency data packets which use binary instead of ASCII encoding, yielding faster data transfer and higher data integrity. Any device that can communicate in MAVLink can talk to ArduPilot. A C++ wrapper has been developed that abstracts MAVLink so it can communicate over any physical transport layer, currently available is communication over serial, TCP/IP, UDP, and Write to file. This has facilitated the seamless transition in the testing phase from SIL through HIL to flight testing, by simply having each component change the physical means of sending MAVLink encoded data.

XBee converts a serial stream to wireless using a protocol called Zigbee. A large outdoor 15 dB omni directional 2.4Ghz antenna is used on the ground, and a striped down 8 dB antenna on the aircraft. This means that the small 60 mW power of the XBee can communicate with an aircraft a mile away in any direction and orientation. The series 2.5 XBees used are point to multi point in a communication mesh. A master module is attached to the ground station receiving data from as many as 255 other XBees. Due to the high gain

Fig. 7. WOT4 testing aircraft

Fig. 8. X-Plane RC Plane model of WOT4

antennas, and the meshing ability of XBee 2.5 multi vehicle test are possible.

VI. Experimental Studies

The platform chosen for the initial experiments is a WOT4 model aircraft. The WOT4 is an inexpensive option for the initial flight tests. This aircraft has been chosen as it is large enough to carry Ardupilot but small enough to keep risk at an acceptable level during the initial flight testing. It is made from Expanded Poly Olefin (EPO) foam with is extremely strong, durable, and very light. The WOT4 is capable of carrying of 500g payload, so would be capable of carrying a camera and video transmitter equipment. Ardupilot is easily able to fit inside the fuselage, close to the center of gravity of the aircraft, and out of the airflow to minimise drag. A pitot static probe, and sensor is mounted half way along the wing out of prop wash, to measure the airspeed of the aircraft. The WOT4 is shown in Fig. 7.

For SIL and HIL, an X-Plane model of the WOT4 has been developed, so algorithms could be tested on a representative model. The model was developed using X-Plane's plane maker, using the aircrafts dimensions, power, and wing cross section, to give an approximate representation of the aircraft. A graphical representation of the model can be seen in Fig. 8.

To test the functionality of the full system from SIL to real world testing, step tests are performed as well as tests on speed hold, and altitude hold controllers. Finally a simple PID waypoint tracking algorithm is performed.

Using the MAVLink Simulink block aircraft state information can be read, and roll, pitch angle, rudder and throttle commands sent to and from ArduPilot. The speed, altitude hold, and waypoint tracking are also programed in Simulink.

Illustrated in Fig. 9 is the step response of the WOT4 in roll, in SIL, HIL, and real world. From flying straight and level a 60° roll angle is commanded, then a -60 ° angle is commanded. The response in SIL and HIL were of course similar as they use the same model in X-Plane, the real world response only differed slightly. This simple test shows the fundamentals of the system functioning, including Ardupilots

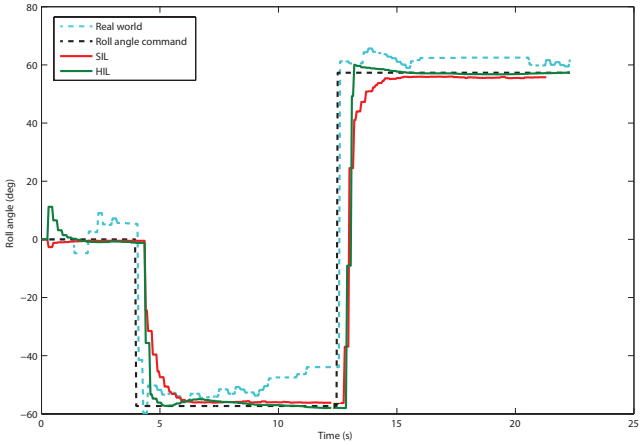

Fig. 9. Step response for roll angle for the WOT4, in SIL, HIL, and real world

inner loop control, MAVLink, and communications downlink functionality.

To demonstrate the ability of Simulink to perform outer loop control, both speed hold, and heading hold control are tested. Also they are needed for more advanced outer loop control. Both are based on simple PID controllers, where speed is controlled with pitch angle, and heading is controlled with roll angle. Once again this was put through the whole development cycle. Fig. 10 shows the aircraft responding to step commands in airspeed. Fig. 11 shows the aircraft responding to step commands in heading. As good data integrity, and speed is important in off board control, these tests show that Xbee, and Simulink are capable of this. As wireless communication is the weak link in this system and as simple off board control functions perfectly, there is no reason that the system could not now perform more complex control algorithms.

A simple PID waypoint tracking algorithm was implemented in Simulink, to make an aircraft fly around user defined waypoints. It uses the haversign formula [13] to calculate the heading between the aircraft's, and waypoint's latitude and

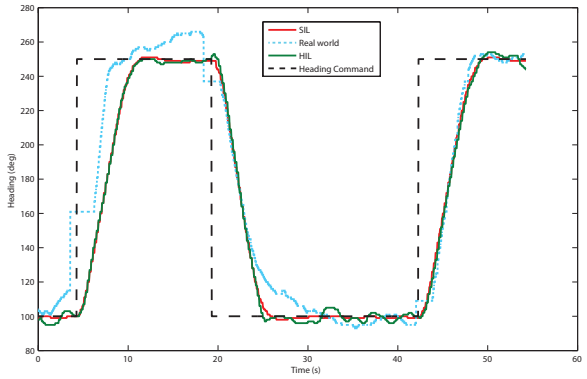

Fig. 10. Step response for heading hold controller for the WOT4, in SIL, HIL, and real world

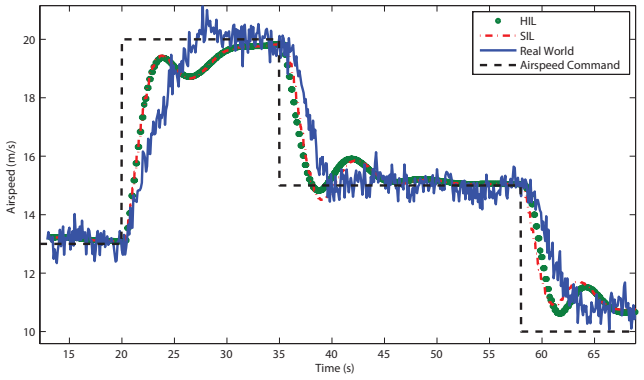

Fig. 11. Step response for speed hold controller for the WOT4, in SIL, HIL, and real world

longitude. Eq. (1), and Eq. (2) shows how the bearing (θ) and the distance (d) between two points on the earth surface defined by latitude and longitude can be calculated.

$$a = s^2(\tfrac{\delta lat}{2}) + c(lat_1)c(lat_2)s^2(\tfrac{\delta lng}{2})$$

$$c = 2\arctan(\tfrac{\sqrt{a}}{\sqrt{1-a}}) \quad (1)$$

$$d = R_e c$$

$$\theta = \arctan(\frac{s(\delta lng)c(lat_2)}{c(lat_1)s(lat_2) - s(lat_1)c(lat_2)c(\delta lng)}) \quad (2)$$

Where cos and sin are abbreviated to c and s respectively. Where R_e is the radius of the earth which is 6378.1 Km, and δlat, and δlng is difference between the aircraft's latitude and longitude and that of the origin or the waypoint the distance and heading is to be measured. lat_2, and lng_2 are the latitude, and longitude of the aircraft, and lat_1, and lng_1 are the latitude, and longitude of the next waypoint.

This bearing is then used as the heading command to make the aircraft fly directly at the next waypoint. The aircraft is flown around a sqaure circuit at 200 m height above ground, and at 15 m/s. Fig. 12 shows the 2D path flown by the WOT4

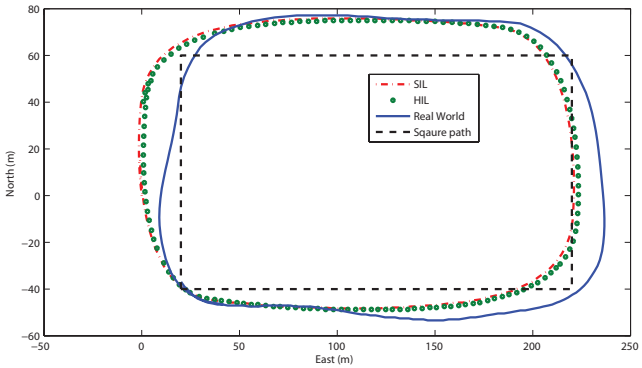

Fig. 12. Simple PID waypoint tracking algorithm tracking a 100x200m square path, in SIL, HIL, and real world

in SIL HIL, and in real world testing. It is in this flight test that the greatest difference between the simulated WOT4 and the real WOT4 is observed, but the simulated tests proved that the algorithms worked successfully.

VII. Conclusion

A system has been developed that enables research in the field of UAS to quickly develop and test high level control algorithms. By using a COTS autopilot, the full system is cheap and easy to integrate into a range of vehicles. As ArduPilot is a well developed system it is extremely reliable, and robust which makes flight testing swift and safe. The results from SIL, HIL testing have with relative accuracy, predicted the performance of the WOT4 in real world tests. The control system developed in Simulink needed no modification between simulation and real world tests. These show that this is a good technique for safely testing algorithms in simulation before moving the to actual vehicles. As the high level control was carried out off board, on a ground control station, and relied on a wireless data link occasionally there were a few data packets drops. The aircraft never got further than 300m away so if operating at further distances, or at higher data rates, an embedded system can be used to conduct the high level control with the same ease.

References

[1] G. Cai, B. M. Chen, T. H. Lee, and M. Dong, "Design and implementation of a hardware-in-the-loop simulation system for small-scale uav helicopters," *Mechatronics*, vol. 19, no. 7, pp. 1057 – 1066, 2009.
[2] [Online]. Available: http://diydrones.com/notes/ArduPilot
[3] [Online]. Available: http://qgroundcontrol.org/mavlink/start
[4] [Online]. Available: http://www.arduino.cc/
[5] D. Kingston, R. Beard, A. Beard, T. McLain, M. Larsen, and W. Ren, "Autonomous vehicle technologies for small fixed wing uavs," in *AIAA Journal of Aerospace Computing, Information, and Communication*, 2003, pp. 2003–6559.
[6] Y. C. Paw and G. J. Balas, "Development and application of an integrated framework for small uav flight control development," *Mechatronics*, vol. 21, pp. 789 – 802, 2011.
[7] A. Mehta and K. Pister, "Warpwing: A complete open source control platform for miniature robots," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, oct. 2010, pp. 5169 –5174.
[8] [Online]. Available: http://paparazzi.enac.fr/wiki
[9] [Online]. Available: http://www.micropilot.com/

[10] M. G. P.-S. H. Pascal Brisset, Antoine Drouin and J. Tyler. (2006, October) The paparazzi solution. ENAC.
[11] [Online]. Available: http://www.attopilotinternational.com/
[12] M. A. O. Simes, "Development of an aerial robot for inspection and surveillance," in *Mestrado em Engenharia Mecnica*. Universidade de Aveiro, 2009.
[13] R. W. Sinnott, "Virtues of the haversine," in *Sky and Telescope*, vol. 68, Dec 1987, p. 158.