FASt - An autonomous sailing platform for oceanographic missions

José C. Alves
Department of Electrical and Computer Engineering
University of Porto, Faculty of Engineering
Porto, Portugal
Email: jca@fe.up.pt

Nuno A. Cruz
Department of Electrical and Computer Engineering
University of Porto, Faculty of Engineering
Porto, Portugal
Email: nacruz@fe.up.pt

***Abstract*— Sailing has been for long times the only means of ship propulsion at sea. Although the performance of a sailing vessel is well below the present power driven ships, either in terms of navigation speed and predictability, wind energy is absolutely renewable, clean and free. Unmanned autonomous sailing boats may exhibit a virtually unlimited autonomy and be able to perform unassisted missions at sea for long periods of time. Promising applications include oceanographic and weather data collecting, surveillance and even military applications. The Microtransat competition, launched in Europe in 2006, has been a key initiative to promote the development of robotic unmanned sailing boats. Various regattas have taken place across Europe and the ultimate challenge will be a transatlantic race. This paper presents an autonomous sailing boat developed at the University of Porto, Portugal, with emphasis on the hardware and software computing infrastructure. This platform is capable of carrying a few kilograms of sensing equipment that can be hooked to the boat's main computer, also providing support for short and long range data communications.**

I. Introduction

Contrary to power driven ships, in sailing boats the fastest way to reach a given destination point may be a complex path that depends on various parameters, mainly the current and future wind direction and speed, the period and height of the waves and the sailing performance of the boat. In addition, due to the limited predictability of the weather conditions, the best solution has to be determined or adjusted dynamically during a journey.

Autopilots have been used for decades for automatic steering of sailing boats and are now an essential equipment for long sailing journeys, specially for solitaire sailors. The first autopilots for sailing boats were fully mechanical, providing a closed loop control system to maintain a constant course angle relative to the apparent wind direction. A popular system, still in use today by some sailors, is the windvane servomechanism [1] that has the great advantage of not requiring electric power for its operation. However, stability may be compromised due to the varying apparent wind direction induced by the waves, specially in downwind legs with high waves when the absolute boat speed can oscillate significantly. Current electronic autopilots rely on various input data to steer a sailing boat, including the heading, course, wind and heeling angle. Computerized control systems allow the application of sophisticated methods like fuzzy-sets [2], [3], neural networks [4] or other artificial intelligence techniques [5] as a means to handle the steering control much like a human does.

In addition to steering a sailing boat to keep it in the desired course, the control of sails is a key issue for maximizing its speed. In practice, it may be impossible to constantly tune the sail to follow all the fluctuations of the apparent wind direction, mainly due to restrictions on the power available (either electric or human) to move the sails and also practical limitations of the speed of such adjustments. This is effective in small dinghies where the sailor is continuously controlling the sail sheet by hand but may not be possible in large yachts where a sail adjustment is a slow process and consumes a significant amount of energy due to the high forces involved.

Tuning a sailing boat for minimizing the time required to reach a desired target is thus a complex task that involves course planning, steering control and sail tuning. These components are highly interdependent and cannot be determined in advance during real navigation conditions.

Although presently fully autonomous unmanned sailing boats do not fit in any legal category of ship, the potential they exhibit to carry out unassisted missions at sea for long periods of time may help change this scenario.

This paper introduces the Microtransat competition and presents the hardware and software architecture of FASt (FEUP[1] Autonomous Sailboat), a small scale sailing boat developed at the University of Porto, Portugal. In addition to this introduction, the rest of the paper is organized as follows. Section II presents the Microtransat contest, as a stimuli for the development of autonomous sailing boats. The FASt project is introduced in section III, including a brief presentation of the construction phases and physical data of the current configuration. The electronic system, including the main computer, sensors, actuators and communications, is detailed in section IV, and the organization of the software component is presented in section VIII. Finally, section IX concludes the paper and discusses the future developments planned for FASt.

II. The Microtransat competition

To motivate the research and development on autonomous sailing boats, a group of university professors in the United

[1]Faculty of Engineering of the University of Porto

Kingdom and France has launched, a few years ago, the Microtransat Challenge (www.microtransat.org). This is a competition of small fully autonomous sailing boats that will have as the ultimate challenge a crossing of the Atlantic ocean.

The Microtransat rules are simple: the only means of propulsion allowed is the wind, boats have to carry or generate all the electric energy they need and no external intervention is allowed during the navigation. In addition, rules establish a maximum boat length of 4 m and (originally) a maximum displacement of 40 kg. The objective of these bounds was to not harm any other type of boat, in the case of an eventual collision at sea.

Since 2006, three competitions have taken place across Europe in restricted waters, including the first World Robotic Sailing Championship in the lake Neusiedl, Austria, in May 2008, that joined four teams from Austria, United Kingdom, Canada and Portugal. These meetings represent excellent opportunities for testing the technologies developed, exchange ideas and share experiences.

A. Legal issues

In spite of the low risk of damage in a real ship resulting from a collision with one of the Microtransat sailboats, the current international maritime law forbids the autonomous navigation of such unmanned vessels. Because of this, a solution has still to be envisaged for accomplishing the transatlantic race planned in the Microtransat competition.

The international rules for avoiding collisions at sea (COLREGS) only address crafts under human control. Current technology has not yet proven to be capable of dealing with the ambiguity of some rules under real situations and, in some cases, the complexity of the procedures required to follow them. Some recent works [6], [7] address the problem of collision avoidance at sea, although a major challenge is still the difficulties associated with the visual perception of the local environment around a boat.

Present Microtransat participants still have no capability to watch their surroundings for identifying potential risks of collision with other ships. However, with the generalization of the AIS (Automatic Information System) and other electronic aids this may be surpassed in a near future by low cost and low power electronics, and open room for the inclusion of unmanned autonomous boats in the international sea regulations.

III. The FASt project

The FASt project was launched at the Electric and Computer Department of the School of Engineering of the University of Porto, Portugal, in the beginning of 2007, with the immediate objective of entering the Microtransat series of competitions. The project aimed to design and build a small-scale autonomous sailing boat, robust enough to withstand tough sea and wind conditions. The design length was set to 2.5 m, after scaling down in length and displacement some real oceanic modern sailing boats, and keeping the displacement not far from the 40 kg mark, in order to facilitate the launch and transportation.

The hull shape was inspired in the modern racing oceanic yachts and was developed with the free version of the boat design software DelftShip (www.delftship.org, former FreeShip). The rig is a standard Marconi configuration with the head sail mounted on a boom, as used in small radio-controlled sailing boats. For the moment, there is no mechanism to reef the sails, although this will be necessary in the oceanic version to make it capable of controlled navigation in a wide range of wind speeds. To increase stability, the boat includes a deep keel with a lead ballast. Main dimensions are presented in table I.

TABLE I
Main dimensions of the FEUP Autonomous Sailboat - FASt

Total length (LOA)	2.50 m
Length in the water line (LWL)	2.48 m
Maximum width (beam)	0.67 m
Draft	1.25 m
Displacement	50 kg
Wetted surface	1.0 m^2
Ballast	20 kg
Sail area	3.7 m^2
Mast height	3.4 m

A. Hull construction

The boat was built by a team of students and professors of FEUP, with the help of a kayak builder (Elio Kayaks, www.elio-kayaks.com) that made some of the parts in composite materials. The construction started in the beginning of March 2007 with the assembly of a model. This was built starting from a structure of plywood frames that was covered with a layer of strip planking reinforced with fibreglass and polyester resin. After several iterations of filling and sanding to obtain a smooth surface, this full scale model was then used to build a negative mould.

The hull was fabricated with a sandwich of unidirectional carbon fibre in the outer layer, a low density 3 mm honey comb core in the middle and a inner layer of fibreglass. This sandwich was then pressed with vacuum during the cure of the epoxy resin, following the same industrial process that is used to build high-performance and light weight racing kayaks. Additional reinforcements in plywood, fibreglass and carbon fibre were added at the points of major mechanical stress: the attachment of the keel, the foot of mast and the main bulkheads where the shrouds connect to the hull. Figure 1 shows key stages of the construction process and the final boat.

The keel was built from a block of rigid polyurethane foam reinforced with a wood core, shaped manually to a NACA profile and then laminated in vacuum with several layers of carbon fibre. The keel ballast was fabricated from several layers of a 3 mm lead sheet cut to the various sections of the bulb shape and glued together with epoxy. The rudders were made from a wood core covered by fibreglass with a stainless steel shaft. Mast and boom are tubes of carbon fibre

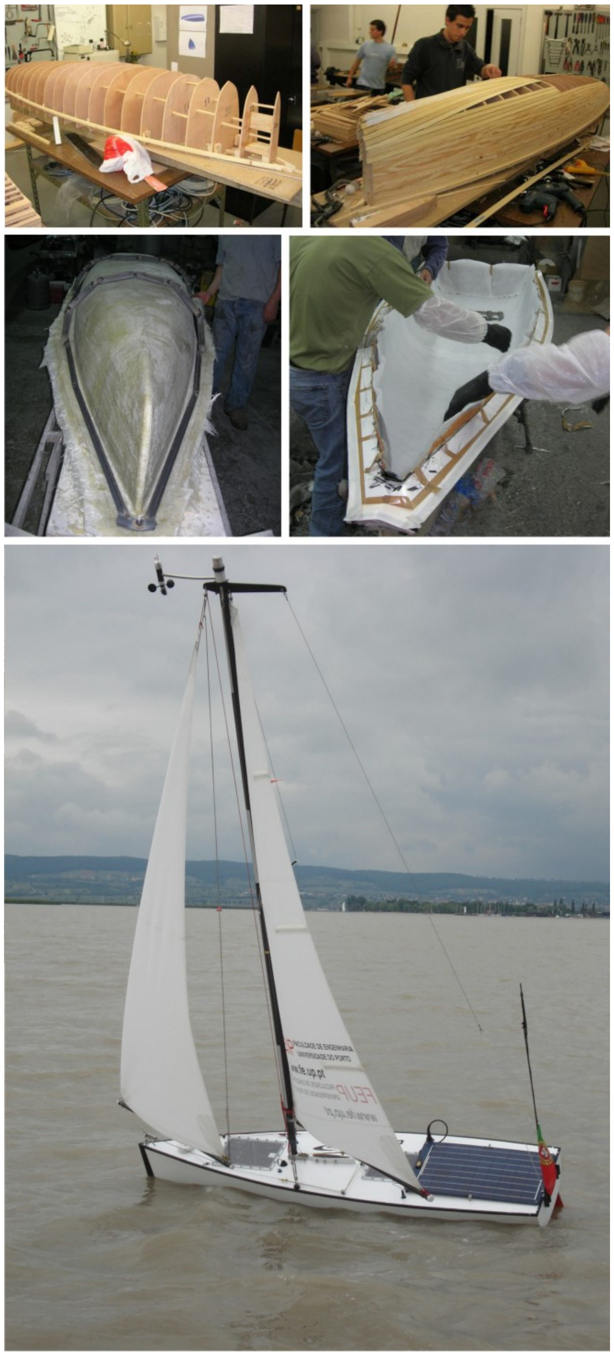

Fig. 1. The construction of FASt: (top-left to bottom) assembly of the frames; strip planking; the mould; building the hull; the final boat.

used in competition paddles and some standard hardware of masts of small dinghies.

IV. Electronic system

The electronic system used in FASt is assembled with various modules, some of them custom built for this application. These components are organized in 5 main subsystems: computing, communications, sensors, actuators and power management. Figure 2 presents a block diagram with the general organization of the various components and their approximate location in the hull.

The computing system is implemented in a small board computer based on a FPGA[2] reconfigurable digital integrated circuit [8]. The digital system implemented on this device includes a 32-bit RISC microprocessor running with a maximum frequency of $50\,\mathrm{MHz}$ (Microblaze [9]), ROM memory holding the bootstrap code and various custom designed digital modules that interface the processor with the sensors and actuators. The organization, characteristics and modes of operation of this system are detailed in the next sections.

The communications section includes a conventional WiFi router (LinkSys WRT54GC), GSM modem (Siemens MC35), IRIDIUM SBD modem (model 9601) and a radio control receiver. All these components are integrated as OEM modules and can be switched on and off under control of the software running in the computing system. The WiFi router provides a convenient data link for short range communication with a personal computer, mainly for software development, debug and configuration purposes. The radio control receiver allows a totally manual control of the sailboat using a four-channel proportional radio-control transmitter. While in sea missions, small volume communications are done by short data messages through a GSM modem and a IRIDIUM SBD modem. While the GSM network is only available within a few miles from shore, the IRIDIUM satellite service guarantees world wide coverage.

Sensors include the wind vane and anemometer, boom position, digital compass (Honeywell HMR3300), GPS (uBlox RCB-4H), inclinometer, voltage monitors, ambient light sensor, interior temperature and a set of water sensors distributed is various places inside the hull. The wind vane and boom position indicator were custom built with a magnetic field direction sensor from Austria Micro Systems (AS5040). This chip measures the orientation of the magnetic field created by a small magnet placed close to its case and provides $10\,\mathrm{bit}$ measures with 1 degree of accuracy. The chip was embedded in epoxy resin and is thus completely isolated from the water. The wind speed sensor is a conventional cup rotor actuating a hall-effect switch.

The sailboat includes only three actuators: two standard high-power RC servos provide independent control of the two rudders and a DC geared motor controls simultaneously the sheets of both sails. The interior layout has room for a second DC motor to allow independent control of the two sails or to provide some mechanism of reducing the sail area. These DC motors are standard window motors used in cars. Although this type of motor and the associated gearbox is known for its low efficiency, they are extremely robust and once positioned and unpowered the gearbox naturally locks the motor's shaft. As the sail angle only needs to be adjusted to set a new course or when the wind direction changes significantly, this represents an important saving of electric energy when comparing to other combinations motor-gearbox that would need power to

[2]Field-Programmable Gate Array

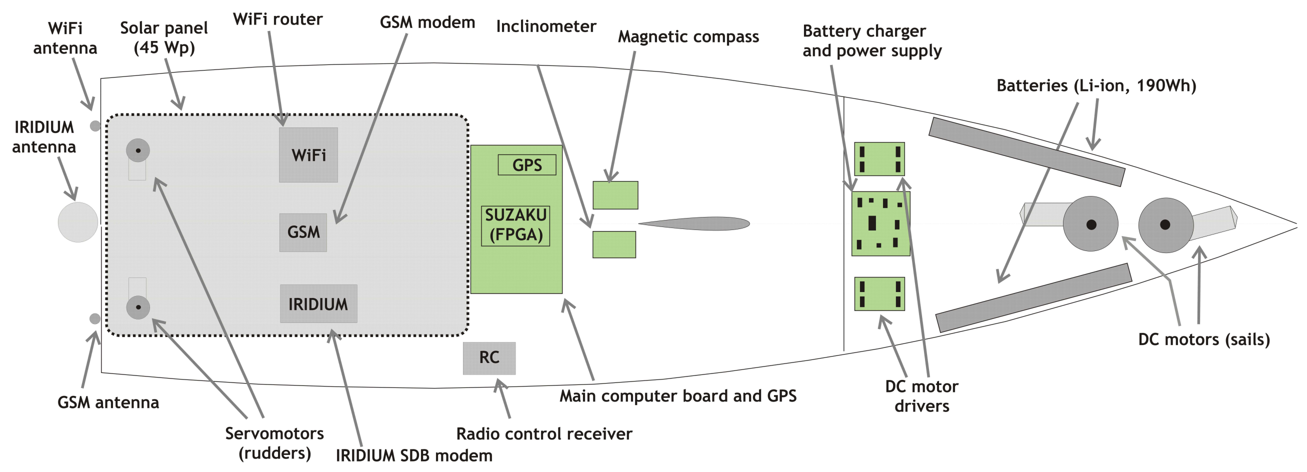

Fig. 2. The organization of the FASt electronic system and its distribution inside the hull.

react to the force of the sail sheet.

Finally, the power management section includes a $45\,\mathrm{Wp}$ solar panel (Solara SM160M), and a complete battery power solution with two $95\,\mathrm{W\,h}$ Li-ion batteries, battery charger and management module and a ultra efficiency power supply.

V. The computing platform

The computer board used in FASt is a commercial system from Suzaku (SZ130 [8]), built around a Xilinx FPGA, model Spartan3E S1200. The board includes $32\,\mathrm{MB}$ of SDRAM, $8\,\mathrm{MB}$ of SPI flash memory, serial interface and Ethernet port implemented by a dedicated chip external to the FPGA. A total of 86 digital I/O pins are available for the user application, directly connected to FPGA input/output pins and distributed in edge connectors around the board. The FPGA is only partially occupied by the base project (processor and essential peripherals), leaving roughly more than 1 million equivalent logic gates available for the user system. Figure 3 depicts the organization of the SZ130 board and the reference project implemented in the FPGA that is included with the development kit.

The system runs uCLinux (`www.uclinux.org`), a version of the popular Linux operating system that has been simplified and adapted for embedded applications running in processors with no memory management unit (MMU). The operating system provides an interactive command line console through a standard RS232 port, a structured file system, multitasking and basic TCP/IP services (FTP, HTTP and TELNET).

This board constitutes a convenient platform for building a digital integrated system that combines a hardware/software cooperative approach. In addition to a conventional microprocessor, a system may include virtually any custom designed digital circuit that may fit into the FPGA's free resources, combining sequential software execution with parallel custom processing carried out by dedicated hardware modules. By moving into custom hardware some of the processing and interfacing functions of a system, the computing load of the

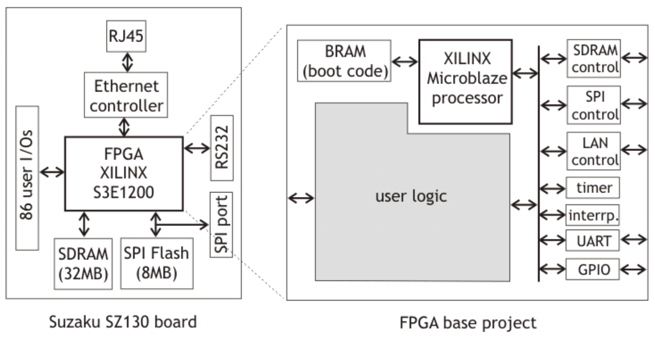

Fig. 3. Organization of the Suzaku SZ130 single board computer.

central microprocessor may be significantly alleviated. Besides simplifying the development of the software, this approach also allows to reduce the processor's clock frequency what impacts in the overall power consumption.

A. FPGAs

The computing system designed for FASt is based on a FPGA (Field-Programmable Gate Array), providing a flexible reconfigurable computing platform [10]. This solution allows the integration of almost all the required digital electronics into a single chip and simplifies significantly the design of the control software, removing from the central microprocessor the low-level interfacing and data processing tasks.

FPGAs are commercial integrated circuits that can be configured by the end user to implement any arbitrary digital system. The common configuration technology used in present FPGAs is based on SRAM and provides a virtually infinite number of re-configurations in very short times (tens to hundreds of milliseconds). The state of the art FPGA devices offer capacities equivalent to a few millions of logic gates, and include on-chip static RAM exceeding $10\,\mathrm{Mbit}$, dedicated functional blocks optimized for signal-processing applications, gigabit transceivers and, in some families, embedded high-performance processors. Furthermore, the digital systems implemented in such devices can run with clocks of a few hundreds of MHz and exceed one thousand input/output pins available for the user application. This is now a mature digital technology that offers flexible single chip platforms for targeting complex and high-performance digital systems without incurring in the high costs and long turnaround times of silicon fabrication.

Another interesting attractive of FPGA technology is the ability to quickly modify the digital system implemented in the chip. Different pre-compiled configuration files can reside in low cost off-chip flash memories (or even hard disk) and loaded on request into the FPGA to configure a completely different system. Some FPGA families even allow partial reconfigurations without disturbing the rest of the chip. This is particularly interesting in applications where the processing requirements may vary along the running time depending on external stimuli or operating conditions.

B. Software development

The software for this board is developed in ANSI C and compiled with a customized version of `gcc`. The development of programs to run on the uCLinux operating system can make use of the most common Linux standard libraries, including TCP/IP communication, file I/O and file system management. The compilation is done on a conventional Linux machine and transferred to the Suzaku board via FTP or through the RS232 serial interface.

The uCLinux file system is locally stored in the flash memory and loaded into a segment of the SDRAM (configured as a RAM disk) during the boot process. During application development, evaluation and debug, the support of the uCLinux operating system is a convenient solution because it eases the implementation of network communication processes, file management and multitasking of different program's parts. However, for applications requiring low power consumption, running on the top of an operating system may represent a significant overhead in terms of energy consumption. This platform supports easily both implementations that still have to be evaluated.

C. Hardware development

The development of the digital system implemented in the FPGA is done with the EDK/ISE software tools from XILINX. The XILINX EDK (Embedded Development Kit) is a design tool that builds a combined hardware/software design, targeted to a XILINX FPGA-based board. The hardware part is assembled with pre-designed parametrizable modules (microprocessor, SDRAM interface, USART, etc.) and user designed components. The software part is built as a C program that will be later embedded with the FPGA configuration data. The ISE tool suite performs the complete digital design flow for XILINX FPGAs and translates the circuit models produced by EDK into the final data file (bitstream) used to configure the FPGA.

D. FPGA reconfiguration

When the system is powered up, the FPGA chip is configured with data stored in the flash memory. Once the configuration is completed (within less than 1 second) the system implemented in the FPGA starts working and the Microblaze microprocessor runs the code stored internally in the FPGA memories. If the startup of the uCLinux is enabled, then a boot loader is executed to install the file system image into a virtual RAM disk, starting then the operating system kernel. The complete boot process, from power-up to system idle state, takes only approximately $42\,\mathrm{s}$, running the Microblaze at $50\,\mathrm{MHz}$.

The reconfiguration of the FPGA can be done easily under control of software. Under uCLinux, the section of the flash memory that holds the FPGA configuration data can be rewritten from a regular file stored in the file system, using one application included in the distribution. The running software application can thus choose a digital system from a batch of pre-built FPGA configurations, copy it to the flash memory and issue a `reboot` command to restart the system with a completely different digital system in the FPGA.

This unique feature of FPGA-based systems allows to change the digital circuit played by the FPGA, according to different processing needs that may be driven by several factors (eg. the availability of energy, environmental conditions or function to execute). This is not yet being exploited in FASt, although it may be a good strategy for reducing the energy consumption.

VI. The FASt computing system

The FASt computing system is implemented in the FPGA of the Suzaku board. Besides the central Microblaze processor, the system includes various dedicated controllers for interfacing the sensors and actuators used in the sailboat, some of them associated with custom computing modules. Figure 4 presents the general organization of the system.

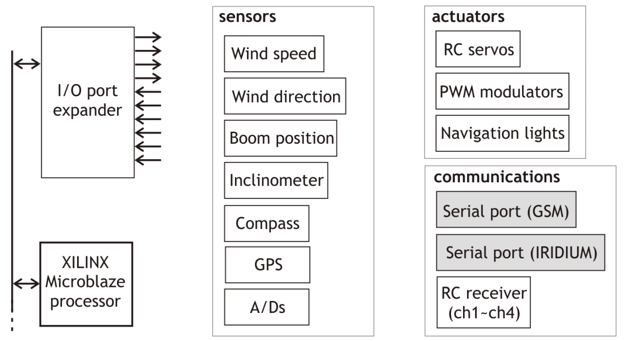

Fig. 4. Simplified view of the organization of the FASt computing system implemented in the FPGA. All the sensors and actuators are accessed by the microprocessor via dedicated interfaces. The two shaded blocks are standard serial port controllers implemented as pre-built modules from the Xilinx EDK library.

The global strategy adopted during the design of this system was to create a set of autonomous interfaces capable of delivering to the software the data retrieved from the sensors in a format easy to be used by the software control system. Besides the implementation of the sensor's specific interface protocols, this includes parsing messages from the sensors, data filtering, and conversion of units. Simple sliding window averaging filters have been implemented for the wind sensors and the inclinometer, although there is enough room to include higher quality filtering approaches coupled to the interfacing modules.

The access to the peripherals from the Microblaze is done through a port expander. This module uses only one pair of 32-bit memory-mapped bidirectional ports and makes available for the rest of the circuit a set of 16 output and inputs ports (32-bit each).

A. Interfaces with sensors

The wind direction interface reads the AS5040 sensor sampled at 50 Hz and averaged using a sliding window filter of 64 samples (the boom position sensor is interfaced by another instance of this module). The output to the software domain is an integer in the range [-180,+180] and the zero reference can be corrected with a value pre-loaded into the hardware interface. The wind speed sensor interface counts the number of 10 KHz clock periods during one revolution of the cup rotor, converts to knots and outputs an average value computed by a similar sliding window mean filter.

To measure the heel angle around the whole 360 degree range, a two-axis accelerometer is used as an inclinometer. This circuit outputs two PWM signals with a pulse width proportional to the acceleration along each axis. The duration of each pulse is measured by digital circuits based on counters and converted to the heel angle.

The magnetic compass provides 8 reads per second with the heading, roll and pitch angles in ASCII format, as variable sized messages. This interface is done by a small custom controller that implements a parser of the messages received from the compass and performs the conversion to binary. The GPS interface is built with independent finite state machines for extracting the relevant data (lat/lon, speed, time, course and status) from the different binary messages output from the GPS.

The interface with the radio-control receiver is done by 4 instances of the same controller, one for each channel of the radio. The standard control signal used in RC receivers and servos is a 50 Hz digital signal, where the high time defines the position of the servo (ranging from 0.8 ms to 2.2 ms). Each receiver module measures the high time of the corresponding channel and converts it to a two's complement 10 bit integer: zero means the control stick at the middle, +511 is full right (full front) and -512 is full left (full rear). One additional module monitors continuously the signal received from radio channel 1, looking for 8 consecutive valid pulses (criteria for valid pulses is frequency between 45 Hz and 55 Hz and high time between 0.5 ms and 2.5 ms) to assert a `radio present` signal. This notifies the rest of the system that the RC transmitter is in range and transmitting correct data.

TABLE II
SUMMARY OF THE FPGA SPARTAN 3E 1200 OCCUPATION

4-input LUTs	7314	(42%)
Flip-flops	3997	(23%)
Occupied slices	4730	(54%)
LUTs used as route-thru	324	(1.8%)
LUTs used as SRAM	256	(1.5%)
LUTs used as shift-registers	796	(4.6%)
Block RAMs	14	(50%)
Dedicated multipliers	8	(28%)
Equivalent gate count	1078257	

B. Actuators

The servo motors controllers receive a 10-bit two's complement number and generate the 50 Hz standard servo control signal, according to the timing referred above. For the moment, only two servos are being used for the two rudders. A hardware multiplexer selects the source of data that is routed to these servos: this can be the output of the RC channel 1 (corresponding to the left-right joystick) or the data sent by the software application running on the processor.

The sail sheet of both sails is commanded by the same DC motor linked to a multi-turn potentiometer for position feedback. This motor is controlled by a PWM modulator that receives from the software a signed 8 bit value that represents the desired speed for the motor. This interface includes a low-pass filter applied to the input data, to avoid high accelerations that would result in high current draw.

VII. FPGA IMPLEMENTATION

Current design, as represented in figure 4, uses less than 50% of the XC3S1200E FPGA resources and, according to the implementation reports generated by the Xilinx tools, corresponds approximately to 1 million equivalent logic gates. All the modules run with the maximum clock frequency of 50 MHz allowed for the Microblaze processor, although most of them may operate with much lower clock frequencies. Table II summarizes the occupation of the FPGA resources (LUT stands for Look-up table and is the elementary configurable block in Spartan 3E FPGA: a 16 bit SRAM implementing any 4-input combinational logic function).

A. Power consumption issues

Electric power consumption is one of the great concerns in an autonomous sailboat. For a small boat, the only reasonable sources of electric energy for long term navigation are photovoltaic panels and wind turbines. Best solution would be a combination of both but, as far as we know, the commercially available wind generators are too large and heavy for our sailboat. In both cases, the availability of energy always depends on the weather conditions which have a high degree of uncertainty. The electronic system must consume the lowest possible energy and whenever possible adapt its behaviour to the power budget available at each stage.

According to our first estimates, the computing system will account for more than 50% of the total energy consumed by all the electric components, assuming a continuous operation with the maximum power consumption measured for the present configuration. This represents aproximately $560\,\mathrm{mA}$ for the $3.3\,\mathrm{V}$ supply ($1.85\,\mathrm{W}$), including the GPS, digital compass and the wind sensors.

VIII. The control software

The control software runs on the top of the uCLinux operating system as a set of processes communicating by UDP sockets. The software architecture tries to mimic the command hierarchy that exists is real boats, associating to a process (software application) a key operational task that exists in a real boat. Each level receives commands (as UDP packets) from a higher hierarchical level and sends orders to the lower level applications.

Presently the software component is composed of 5 applications that implement the main 5 tasks: hardware interface, helm, sail, skipper and logger. The hardware interface implements all the communication with the boat's sensors, actuators and configuration parameters. The helm is responsible for controlling the rudders, according to the programmed control algorithms and orders received from the higher hierarchy (the skipper). The commands supported by this module include keeping a given heading or course, maintaining the angle to the apparent wind and performing the basic maneuvers of tacking or gibing. The sail application controls the position of sails according to the wind speed and direction and a set of decision rules that define the best sail angle. Automated functions embedded in this module include easing the sheet when the heel angle is greater than a certain value and ease completely the sail sheet when the boat is capsized. The skipper is presently the higher level control process. It has access to all the information collected from the sensors through the hardware interface but the commands to the actuators are only sent through the helm and sail processes. This module is responsible for deciding the course to take to reach a given point, when to tack or gibe, or when switch between the manual and autonomous control mode. Finally, the logger is a listen only application that interrogates all the other applications to collect a set of relevant data in a log file. This is specially important during the development and debug phases.

IX. Conclusions

This paper presented an autonomous unmanned sailing boat that was developed in the Faculty of Engineering of the University of Porto, Portugal, to enter the Microtransat competition. This boat provides an efficient sailing platform with virtually unlimited autonomy, capable of performing unassisted missions on the ocean for long periods of time. The boat can carry a few kilograms of additional equipment and it includes a custom designed computer system that can be easily interfaced to a wide variety of additional sensors or actuators.

The computing system includes a RISC microprocessor running a simplified version of the Linux operating system, surrounded by several custom designed peripherals that interface the processor with the sensors and actuators used in the sailboat. The hardware reconfigurability feature of the FPGA device used in this development enables a short design iteration and allows fast reconfigurations of the running hardware. This may be exploited for minimizing the energy consumption by adapting the control and computing logic circuits to the specific requirements of navigation under given wind and sea conditions.

Although presently the international rules do not allow the autonomous navigation of unmanned sailboats, such platforms may be a valuable resource for several applications domains. The generalization of low cost electronic surveillance and identification systems, such as the AIS in use today in commercial ships, may open room in a near future to include robotic boats as legal vessels in the international navigation rules.

Acknowledgements

The authors would like to thank the Department of Electrical and Computing Engineering (DEEC) of the Faculty of Engineering of the University of Porto, Portugal (FEUP), for the financial support of this project.

References

[1] B. Belcher, *Wind-vane Self-Steering*. International Marine Publishing Company, 1982.

[2] E. C. Yeh and J.-C. Bin, "Fuzzy control for self-steering of a sailboat," in *Proceedings of the Singapore International Conference on Intelligent Control and Instrumentation*, vol. 2, 1992, pp. 1339–1344.

[3] R. Stelzer, T. Prol, and J. Robert I, "Fuzzy logic control system for autonomous sailboats," in *Proceedings of the IEEE International Fuzzy Systems Conference*, 2007, pp. 1–6.

[4] A. Tiano, A. Zirilli, C. Yang, and C. Xiao, "A neural autopilot for sailing yachts," in *Proceedings of the 9th Mediterranean Conference on Control and Automation*, 2001, pp. 27–29.

[5] M. L. van Aartrijk, C. P. Tagliola, and P. W. Adriaans, "Ai on the ocean: the robosail project," in *Proceedings of the European Conference on Artificial Intelligence*, 2002, pp. 653–657.

[6] T. Statheros, G. Howells, and K. McDonald-Maier, "Autonomous ship collision avoidance navigation concepts, technologies and techniques," *The Journal of Navigation*, no. 61, pp. 129–142, 2008.

[7] M. R. Benjamin, J. J. Leonard, J. A. Curcio, and P. M. Newman, "A method for protocol-based collision avoidance between autonomous marine surface craft," *Journal of Field Robotics*, vol. 5, no. 23, pp. 333–346, 2006.

[8] Atmark-Techno, "SUZAKU-SZ130-U00, Hardware Manual (English version) V1.0.2, [online]. Atmark-Techno [cited 30 april 2008]. Available from World Wide Web: <http://www.atmark-techno.com>," 2006.

[9] Xilinx, "Microblaze Processor Reference Guide, EDK v6.2 [online]. Xilinx [cited 30 april 2008]. Available from World Wide Web: <http://www.xilinx.com/support/documentation/ip_documentation/microblaze.pdf>," 2004.

[10] K. Compton and S. Hauck, "Reconfigurable computing: A survey of systems and software," *ACM Computing Surveys*, vol. 34, no. 2, pp. 171–210, JUNE 2002.