Small and Inexpensive Single-Board Computer for Autonomous Sailboat Control

Mariano Alvira and Taylor Barton

Abstract. This work presents a computer board designed for autonomous robotic sailboat control. Taking advantage of the current availability of feature-rich processors such as the LPC3130 from NXP and the MC13224v from Freescale Semiconductor used in this work, our design emphasizes low cost and power consumption, as well as small size. At the same time, the system is not excessively specialized; it runs 32-bit Linux and has network capability via Ethernet, WiFi, cellular or Bluetooth USB sticks. The computing system presented in this work is applicable to a variety of robotic sailboat applications, including making a 0.5 m Graupner Micro Magic fully autonomous without relying on a shore-side base station for computation.

1 Introduction

For true autonomy, a robotic sailboat must do all course planning and control on-board the vessel, without relying on an on-shore base station for computation. For very small robotic sailboats, it has previously been difficult to incorporate sufficient computing power on-board given the size of the craft [13]. Class rules for the 0.5 m robotic racing Micro Magic (rrMM) Class at the World Robotic Sailing Championship (WRSC) 2011 and WRSC 2012 allow on-shore computation as a result of this constraint. Boats in the Sailbot and Microtransat classes at WRSC generally contain a more complete computing system to meet the requirement of full autonomy. These boats are typically larger and custom-designed vessels and thus have more relaxed form factor, cost, and power constraints compared to the rrMM.

For robotic sailing competitions like WRSC and Microtransat [8], teams commonly use commercial off-the-shelf (COTS) available electronics for both control hardware and sensors, for example to ease transitions between student groups

Mariano Alvira · Taylor Barton
Seascope, Everett, MA USA
e-mail: {mar,tbarton}@seascopetech.com

[9, 14]. Alternatively, evaluation kits or otherwise pre-existing hardware for a selected microcontroller are used [6, 13]. All of these systems contain to some degree a combination of COTS hardware mixed with custom elements, but the choice of that boundary is varied. While COTS hardware offers many advantages to the development process it has two main disadvantages: it often has either excessive or inadequate features, and typically has a form factor that is not exactly suitable.

For small boats, the disadvantages of COTS circuit boards become very pronounced. By designing a custom PCB for our computing system we were able to choose a main CPU that did not have an existing suitable COTS system but otherwise is very well suited for the application due to its low power, cost, and required board area. Designing a custom control board also allowed us to mitigate form factor concerns with the rrMM by using connectors and layout well suited for the vessel.

This paper presents a control board with enough computing resources for full autonomy suitable for Sailbot and even Microtransat class boats while meeting the tight power, size, and cost budgets required for the rrMM class. So that others can treat the board we have designed here as a COTS component, we have released the hardware design files and software used under suitable Open and/or Free Licences.

2 Hardware Overview

While the primary design goals of the system are cost, power, volume, and computing performance, a secondary consideration is the run-time operating system and choice of computing core. Autonomous sailing competitions often require a significant number of high-level functions such as route-planning, mission/game logic, and data logging. Although standard C on a microcontroller is appropriate for processing sensor samples and digital control, it quickly becomes cumbersome for these more general-purpose computational tasks. An operating system like Linux makes many tasks much easier, such as interfacing with mass storage SD cards, dealing with hardware drivers (e.g. for USB peripherals), and full-featured networking. These features typically require a "Distribution based-OS" (or "full OS") [1] such as any of the numerous OS distributions that use the Linux kernel (e.g. Debian, Arch, Ubuntu, Android). On the other hand, real-time operations such as sensor and control loops, are best done with a dedicated microcontroller running some kind of "real-time operating system" (RTOS). With this in mind, the hardware system is designed as a hybrid "full OS" + "RTOS" system. The system is made up of a baseboard with a main CPU (LPC3130) running Linux, as well as a general-purpose M12 MC13224v module from Redwire [10] running Contiki and serving as a real-time coprocessor. The system block diagram is shown in Fig. 1.

[1] The terms RTOS and OS have varied meaning and scope. For the purposes of this paper, we are calling a "full OS" anything that runs on CPUs with managed memory and provides a rich set of high level programs: i.e. a full Linux distribution. We are calling an "RTOS" any code that runs on a microcontroller or CPU incapable of running a "full OS" and whose purpose includes strict timing requirements.

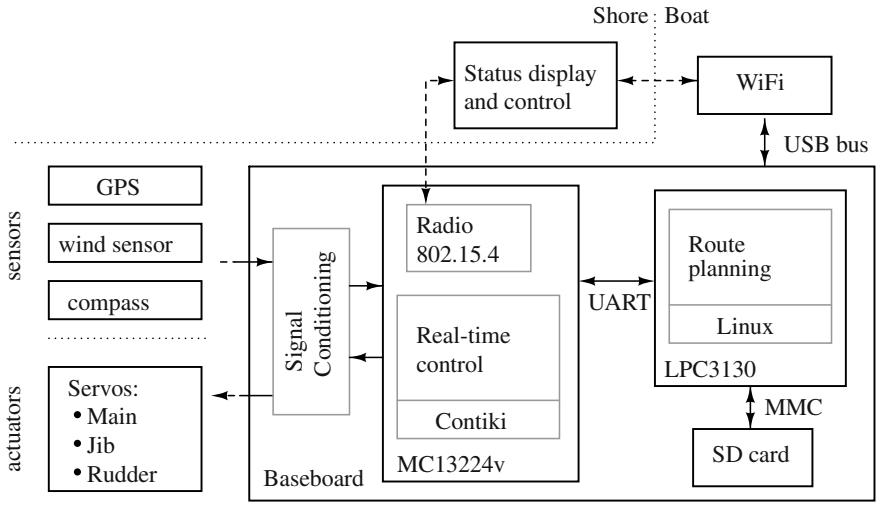

Fig. 1 Generalized system block diagram.

The baseboard is custom-designed for the robotic sailboat application. It contains all system-specific electronics as well as a full Linux system (running Arch Linux, although Debian is also possible) with the LPC3130 200MHz ARM9 with 32MB of RAM. Networking capability is achieved via a USB stick, an option which provides a great deal of flexibility and simplicity, and is also low cost (USB ethernet sticks can be obtained for approximately the same cost as the chips they contain). Wifi, Bluetooth, and cellular modems USB sticks can also be used. As they are swappable, ethernet can be used on shore for development and a wireless option can be used at runtime. Mass storage is accomplished with the SD card interface on the LPC3130. 2GB micro SD cards are used as they are inexpensive and offer plenty of disk storage for the system. The system boots from the SD card and all system files are contained there.

The power subsystem, not shown in the block diagram for clarity, also resides on the baseboard. Terminal pin headers receive wiring from the sailboat system, such as from sensors, actuators, and the power system which is located on the baseboard.

For the processor running the "RTOS" functions, we are using an MC13224v in the form of the M12 module and the Contiki Operating System [3]. The M12 is a general-purpose module that has been designed to be the smallest and easiest way to integrate an MC13224v into a design. The MC13224v is well supported in Contiki, which is a mature open-source "RTOS" written in standard C with a strong emphasis on networking. All real-time control and data processing of the sailboat is done using Contiki's "Protothreads" [4]. The LPC3130 and MC13224v communicate over serial line internet protocol (SLIP) with the UART on the LPC3130 connected to the first UART on the MC13224v. Then the MC13224v simply looks like another

computer on the LPC3130s IP network (and vice versa). This greatly simplifies the development system as all outward APIs can use standard IP based methods to interact.

Unlike many microcontrollers, the MC13224v also includes a wireless 802.15.4 radio (the same physical layer as Xbee or any Zigbee system). This gives an additional communication option that can be used as the primary communication or in tandem with a wireless option on the LPC3130. A system very similar to this with the M12 module alone and computation done on an on-shore computer running Linux was used by the authors at WRSC 2011 to control the "Friend" Micro Magic class sailboat.

With the system hardware in place, autonomous boat operation is achieved by running Python scripts in Linux. This allows for very quick prototyping and development of the boat's behaviors without impeding crucial real-time control operations. The Wifi connection provides a normal networking interface so the boat can be accessed through tools like SSH or the web page it serves. Logging is easily done by writing to a file on the SD card. See Section 4 for more details regarding the software system.

A breakdown of the power consumed by the subsystems of the board is shown in Table 1 below.

Table 1 Power consumption broken out by subsystem.

Subsystem	Power
LPC3130 + RAM:	165 mW
LEDs (3x):	51 mW
Power supply:	88 mW
Wifi:	844 mW
MC13224v:	17 mW
802.15.4 TX:	
@ 4 dBm:	100 mW
@ 20 dBm:	500 mW
Total without comm.	321 mW
+ 802.15.4 @ 4 dBm	421 mW
+ wifi	1165 mW
+ all comm.	1986 mW

3 Implementation Details

The Bill of Materials and approximate cost of the board is included below in Table 2. Photographs of the 79 mm $\times$ 108 mm board in Figs. 2 and 3 show the relative placement of the components.

Three LED indicators are present on the top of the board. A red LED indicates when the system is powered. Each processor has one green LED that it can control.

Also on the top side of the board and shown in detail in Fig. 4 is a debug connector that can receive a Tag-Connect cable [15]. This connector gives access to the UART

Table 2 Bill of Materials for the complete single-board computer.

Part No.	Description	Cost (USD)
LPC3130	ARM9 200MHz CPU	3.78
AS4C16M16S	32MB SDRAM	1.75
AMP-1010058159	micro SD card connector	1.05
	2GB micro SD card	2.79
TPS54140 (×2)	switching power supply	4.74
UE27AC	USB connector	0.25
	Wifi stick	10.00
M12	MC13224v module	12.00
OPA4330 (×2)	Operational amplifiers	3.96
	Ancillary components	6.00
	PCB	3.20
	Assembly	8.98
	Total cost	58.50

from the LPC3130 as well as reset lines for each CPU. This port can be used to work with the serial debug console from the OS running on the LPC3130. A multiplexer typically connects this UART to the MC13224v when the Tag-Connect cable is not present.

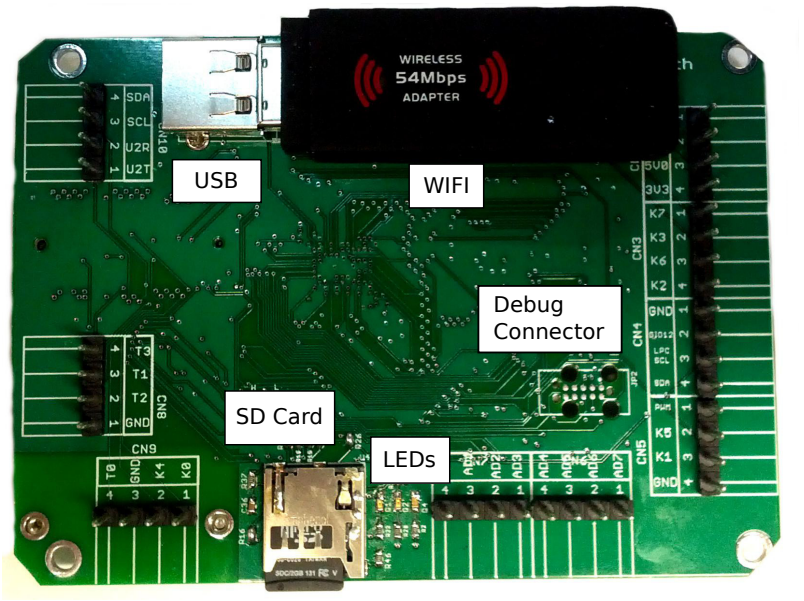

Fig. 2 Top of computer board. The board measures 79 mm × 108 mm.

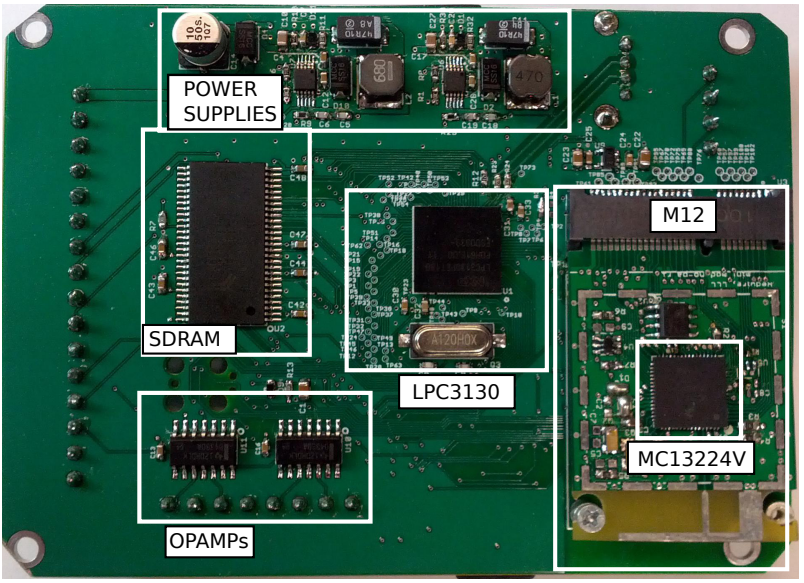

Fig. 3 Bottom of computer board.

Eight chopper stabilized and low-offset operational amplifiers are used to buffer analog input signals going to the MC13224v.

Two switch mode power converters are present on the board to generate 5.0V and 3.3V necessary for the on board components. The output of these regulators is also available on the external pin connections to be used by the system components. The switching converters were chosen for their very robust input ranges. The input voltage range is 7-48V. A polarity diode is included to protect the board from incorrect wiring.

The components that should be easily accessible have been placed on the top side of the PCB and include: the wire harness connectors, indicator LEDs, USB stick, SD card slot, and debug connector. We used pin-headers that can receive wire-to-board screw terminal blocks (see Fig. 5) so that the system wiring can be easily installed or changed.

The form factor of the computer board is sized to fit through the hatch opening in the deck of the rrMM, as shown in Fig. 6. With slight modifications to either the board or to the rrMM, it could be made to fit inside the hatch itself.

4 Software Overview

The control software performs two major functions: real-time control of the sailboats various actuators based on information from its sensors, and high-level

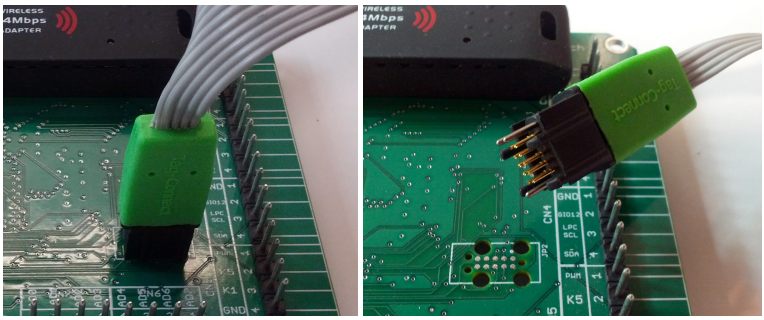

Fig. 4 Tag-Connect debug port.

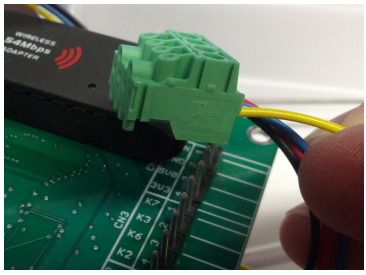

Fig. 5 The controller board has 36 connections for 3.5 mm spacing removable wire-to-board screw terminals.

functions including path planning. As discussed in Section 2, our design uses different CPUs and operating systems for these functions.

Low-level and real-time operations are performed by the MC13224v and the Contiki Operating system. Various hardware peripherals on the MC13224v are configured to interface with the sensors and systems on the robotic sailboats. Pulse width modulator (PWM) pins are configured to create the pulse train necessary to drive RC servos used for rudder and sail controls. The second serial UART interfaces with a GPS. The I2C peripheral communicates with a digital compass and accelerometer. Finally, integrated analog-to-digital converters are used to measure the output from wind sensors.

Once the peripherals are configured, the various inputs and outputs are processed using Contiki's "protothreads" mechanism. A "protothread" is a thread-like structure that is implemented using co-routines [4]. This mechanism results in isolated "processes" that run concurrently. Contiki's protothreads are implemented in standard C and do not need any special complier directives or hardware support. Separate protothreads are used to parse and convert GPS messages, operate real-time control loops, and to update information and control resources used by the next upper layer (i.e. code running on the main CPU).

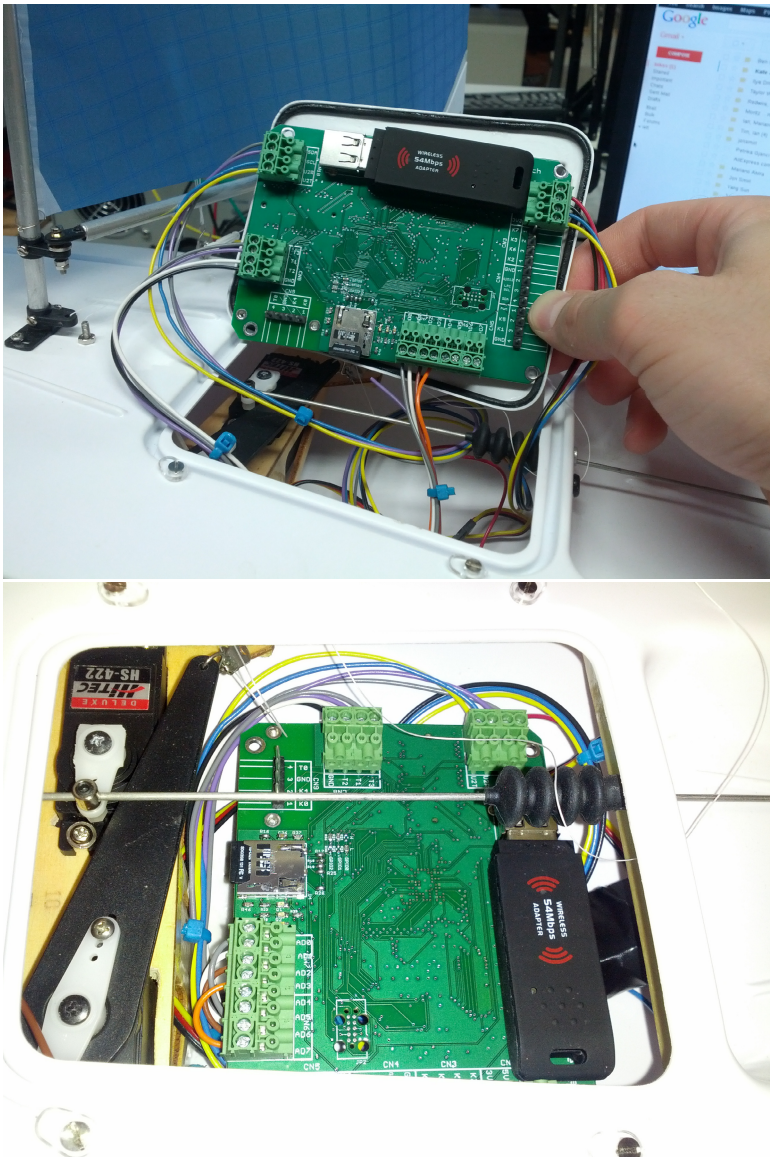

Fig. 6 Control board mounted in the hatch of a racing Micro Magic (top photograph) and inside the hull (bottom photograph).

Contiki has a rich set of networking features. We take advantage of these to expose a REST-ful interface to the sailboat [11]. The MC13224v receives an IP address from the networking environment. Then, other connected devices can perform GET and POST methods on these resources to query or set them appropriately.

Higher-level game functions are scripted in Python running in Linux on the LPC3130. The sailboat resource API is implemented as a "boat" class with state variables corresponding to the various parameters such as rudder angle and sensed wind speed. Logging and debug functions are implemented by writing files to the filesystem.

5 Design Motivation

This section describes in greater detail the motivation for and design decisions of the controller hardware. In particular, once a "full OS" + "RTOS" combination has been selected, there are a great number of available 32-bit ARM processors capable of running a "full OS." Furthermore, we examine whether there is a significant penalty in cost and/or power consumption in choosing to include "full OS" capability by comparing our approach to others used for robotic sailboat designs.

5.1 Choice of ARM Processor

In choosing a processor for the "full OS" processor, we opted to limit our search to 32-bit ARM processors capable of this task. Any of the available options satisfies the power and area constraints imposed by the rrMM; ARMs are widely used in mobile phones and consumer electronics for precisely this reason. Because every major semi-conductor company produces "full OS" capable ARMs, this design constraint does not significantly narrow down the possible options. The remaining design dimensions to explore are cost versus system performance.

When considering system cost and performance, the memory type and size compatible with a particular CPU must be considered in addition to the other CPU performance metrics. Because memory cost can vary by an order of magnitude depending on type and byte density, it has a significant impact on system cost. To characterize the tradeoff of cost versus performance, we have therefore grouped available ARM CPUs by the type of memory controller they have: SDRAM, DDR2, and DDR3. Table 3 shows the byte density and cost of the various memory types. The clock speed, core type, and cost are listed in Table 4 for a variety of CPUs with ARM cores ARM9, ARM11, and A8, in increasing order of performance. For cost numbers in these tables, we are using the qty 1000 prices from various common distributors (Digikey, Mouser, Future, etc...). Other criteria such as inclusion of a floating-point processor (FPU) or graphic processor, or specific peripherals are not considered. (Note: some designers may want an FPU on their computing platform. For our system we have decided that is not a requirement as software floating point emulation is fast enough on the selected CPU.)

We selected the lowest cost and performance corner to use in this design. This choice results in a LPC3130-based system which is a 200MHz ARM9 with 32MB of SDRAM, and an approximate bill-of-material (BOM) cost for the computing system of $5 ($3.5 CPU + $1.5 RAM).

Table 3 Comparison of byte density and cost of various types of RAM.

Memory type	SDRAM	DDR2	DDR3
Max bytes per chip	32 MB	128 MB	256 MB
Cost (USD per qty 1000)	$1.5	$4.8	$10.1

Table 4 Cost of various ARM CPUs grouped by performance class. For this work we selected the lowest cost and performance corner of the design space.

CPU	LPC31, IMX23	IMX25, 28, 35, 50, 51	IMX 53, TI OMAP
Memory Type	SDRAM	DDR2	DDR3
Memory Size (limited by type)	32MB	128 - 256MB	256-1024MB
Speed	200MHz	400MHz	600 - 1000 MHz
Core	ARM9	ARM9 - 11	A8
CPU price	$3.5 - $5	$7 - $12	$20 - $34
Total price w/ memory	$5 - $6.5	$11 - $20	$26 - $43

5.2 Comparison to Other Approaches

Table 5 contains a summary of selected computing systems used in other works, grouped according to a single “full OS” system, single “RTOS”/microcontroller system, and hybrid “full OS” + “RTOS” systems. There is a great amount of variability in the type of computing systems used for robotic sailing. Almost universally, however, a microcontroller (RTOS) is used somehow in the system. Often a “full OS” is not used. This work demonstrates that adding “full OS” capability is not detrimental to the power budget of a system, however, a single “RTOS”-only system is the lowest power option. It is important to note that the power consumption in this work is low enough to dominated by other system components such as communications power.

Table 5 also shows that communication systems used are also varied. As this work uses a USB interface for communication, a variety of different systems are easily supported. 802.15.4 is natively supported by the microcontroller we use for our RTOS functions.

Table 5 Computer system parameters from selected work. Power derived from datasheets when not explicitly cited by author. MOOP w/ gumstix shown [12].

System	Full OS	RTOS	Approx. power	Comm	Fits in rrMM?
[13]		X	70 mW	Bluetooth	yes
[6]		X	850 mW	Bluetooth	probably
[14]		X	200 mW	800-900MHz	probably
[5]	X		8000 mW	VHF	no
[12]	X	X	1250 mW	Wifi	probably
[9]		X	6 mW	802.15.4	no
This work	X	X	165 mW	802.15.4 + various	yes

6 Open Hardware and Software Release

All hardware and software files for the production of this board have been released under open hardware and open software licenses and are available at: `http://www.seascopetech.com/SBC`.

7 Conclusion

We have designed and constructed a small low-cost single board computer that is well suited for robotic sailboats including small boats such as the Graupner Micro Magic. The single-board computer uses both an ARM9 processor to run a complete operating system distribution such as Arch Linux or Debian, as well as an ARM7 microcontroller to perform real-time control operations and hardware interfacing. It supports a variety of communications systems such as Wifi, Bluetooth, or cellular modems via on-board USB; 802.15.4 is supported naively by the ARM7 coprocessor. The board can be constructed for approximately $60 (USD). The full system consumes about 250mW (excluding the power consumed for communication) which is comparable to other low-power controllers for robotic sailboats. Finally, we have released the hardware and software under open source licenses to benefit the robotic sailing community.

References

1. Alves, J., Ramos, T., Cruz, N.: A reconfigurable computing system for an autonomous sailboat. In: Proceedings of the International Robotic Sailing Conference, Breitenbrunn, Austria, pp. 13–20 (May 2008)
2. Barton, T., Alvira, M.: A Discrete-Component 2D-Array Wind Sensor without Moving Parts for a Robotic Sailboat. In: Sauze, C., Finnis, J. (eds.) Robotic Sailing 2012, vol. 121, pp. 95–104. Springer, Heidelberg (2013)
3. The Contiki OS: The Operating System for the Internet of Things, `http://www.contiki-os.org/` (accessed on July 8, 2012)

4. Dunkels, A., Schmidt, O., Voigt, T., Ali, M.: Protothreads: simplifying event-driven programming of memory-constrained embedded systems. In: Proceedings of the 4th International Conference on Embedded Networked Sensor Systems, Boulder, Colorado, USA, pp. 29–42 (2006)
5. Giger, L., Wismer, S., Boehl, S., Büsser, G.-A., Erckens, H., Weber, J., Moser, P., Schwizer, P., Pradalier, C., Siegwart, R.: Design and Construction of the Autonomous Sailing Vessel AVALON. In: Proceedings of the 2nd International Robotic Sailing Conference, Matosinhos, Portugal, pp. 17–22 (July 2009)
6. Koch, M., Petersen, W.: Using ARM7 and μC/OS-II to Control an Autonomous Sailboat. In: Schlaefer, A., Blaurock, O. (eds.) Robotic Sailing, vol. 79, pp. 101–112. Springer, Heidelberg (2011)
7. Kushalnagar, N., Montenegro, G., Schumacher, C.: IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals. Internet Engineering Task Force Request for Comments 4919 (2007), http://www.ietf.org/rfc/rfc4919.txt
8. Microtransat, http://www.microtransat.org/ (accessed on July 15, 2012)
9. Miller, P., Brooks, O., Hamlet, M.: Development of the USNA SailBots (ASV). In: Proceedings of the 2nd International Robotic Sailing Conference, Matosinhos, Portugal, pp. 9–16 (July 2009)
10. Redwire Store (Redwire is owned and operated by Mariano Alvira), http://redwirellc.com/store/ (accessed on July 8, 2012)
11. REpresentational State Transfer (REST) entry on Wikipedia, http://en.wikipedia.org/wiki/Representational_state_transfer (accessed on July 15, 2012)
12. Sauzé, C., Neal, M.: MOOP: A Miniature Sailing Robot Platform. In: Schlaefer, A., Blaurock, O. (eds.) Robotic Sailing, vol. 79, pp. 39–53. Springer, Heidelberg (2011)
13. Schlaefer, A., Beckmann, D., Heinig, M., Bruder, R.: A New Class for Robotic Sailing: The Robotic Racing Micro Magic. In: Schlaefer, A., Blaurock, O. (eds.) Robotic Sailing, vol. 79, pp. 71–84. Springer, Heidelberg (2011)
14. Sliwka, J., Reilhac, P.-H., Leloup, R., Crepier, P., De Malet, H., Sittaramane, P., Le Bars, F., Roncin, K., Aizier, B., Jaulin, L., et al.: Autonomous Robotic Boat of ENSIETA. In: Proceedings of the 2nd International Robotic Sailing Conference, Matosinhos, Portugal, pp. 1–7 (July 2009)
15. Tag-Connect Official Website, http://www.tag-connect.com/ (accessed on July 15, 2012)