Embedded Web Server for the AVR Butterfly Enabling Immediate Access to Wireless Sensor Node Readings

Konstantinos Samalekas, Evangelos Logaras, and Elias S. Manolakos

National and Kapodistrian University of Athens,
Department of Informatics and Telecommunications,
Panepistimioupolis, Ilissia, 15784 Athens, Greece
{k.samalekas,evlog,eliasm}@di.uoa.gr

Abstract. The "AVR Butterfly" (BF) is not just a microcontroller, but rather an autonomous embedded system kit including several sensors. Our main objective was first to provide Internet connectivity to the BF and then to evaluate its further capabilities as a sensor network node. To this end, we equipped the BF with a TCP/IP stack and a Zigbee transceiver. As a case study, we constructed a node and a gateway based on the BF and formed a simple wireless sensor network (WSN). In order to enable remote users to access, on demand, sensor node readings through their web browsers, an HTML Server application was developed for the BF-gateway. We demonstrate that despite the scarcity of the available resources, if we enhance the BF with a popular Ethernet chip and the optimal TCP/IP stack for 8-bit microcontrollers, we can obtain a powerful, yet simple and inexpensive, monitoring device with Internet connectivity capabilities.

Keywords: AVR Butterfly, TCP/IP, Web Server, Wireless Sensor Network, Microcontrollers, Zigbee.

1 Introduction

In our days, we are experiencing the dawn of a new information revolution. We are heading towards major changes that will affect the coexistence of humans and machines. Internet users will become billions, but still a minority in comparison to networked devices [1]. With the fast progress of computer technology and the advancements in communication protocols, such as the IPv6, the assignment of a unique address to every "small device" is now becoming possible. As a consequence, cameras, traffic lights, parking meters and household appliances of the near future may perform measurements and transmit information about themselves and peers in their neighbourhood. The huge development of networked devices, sensors and actuators is forming a "Sensor Nation" [2] which expands rapidly, transforming the way that people gather information and interact with their environment.

There are numerous applications of sensor networks, in a wide spectrum of fields such as in medicine for collecting patients' data [3], in industry for controlling manufacturing [4] and in transportation for effectively managing traffic [5]. One of the most interesting and challenging applications of sensor networks is wide-area environmental monitoring. It is now common to use multiple sensors in order to monitor the environmental conditions in large geographical areas, e.g. the activity of volcanoes, the eruption

N. Komninos (Ed.): SENSAPPEAL 2009, LNICST 29, pp. 145–158, 2010.

of wildfires etc. With the use of wireless protocols and inexpensive or even disposable sensors, real-time environmental monitoring becomes feasible even for low budget research projects.

In many situations, such as in wildlife habitat monitoring [6], it is undesirable that the researcher approaches the sensors physically. In these cases the installation and maintenance of information sinks that collect, compute and store data locally, might affect the flexibility and the energy requirements of the system. Thus, the network design must be accomplished in such a way that distributed data can be reached and evaluated from a remote location. The most interesting case, is to become able to utilize the largest and most widely available network, the Internet, to reach specific sensor measurements or reports from anywhere in the planet. The interest in distance monitoring and control is growing rapidly [7], bringing enhanced capabilities and efficiency. The development of microelectronics and embedded systems allows the creation of affordable, web-enabled boards and devices. The Internet as the means to access remote sensor nodes, not only provides flexibility to the monitoring system, but also facilitates the end-users needs, by replacing the physical or software control panels with their familiar web browser interface.

It is always vital for sensor networks to have the ability to communicate with other infrastructure networks in order to propagate their data. Apart from the structural and functional similarities that we notice between sensor networks and traditional networks, we also encounter some important differences. First of all, the nodes in a sensor network are usually co-operating, hence appropriate synchronization and data management is required. The protocols are mainly low-power wireless and the usage of the communication channel is limited. Additionally, as low power consumption is crucial, there are restrictions for the nodes, often limiting their performance and forcing them to sleep for most of the time. On the other hand, in traditional networks we have end-to-end connections among independent hosts. Moreover, the links are usually wired and power restrictions are unnecessary. The demand for throughput here is high, while both bandwidth and retransmissions are relatively inexpensive.

Modern tendencies and needs dictate that sensor networks should be integrated with the Internet. However, the best architectural solution for this integration is still debated. At the moment there is no perfect proposal, the scheme to be adopted though, must be efficient, reliable, extensible, and scalable. In general, we could classify the architectures that have been proposed in two categories: the Gateway approach and the Overlay approach [8].

The Gateway approach [9] is the “traditional” and simplest scheme to implement. The general idea is that a special device comes in between the sensor network and the Internet and performs the translation between the protocols of the two ends. Therefore, no additional workload is necessary for the nodes, since the (application-level) gateway is the only unit responsible for managing the communication overhead. However, due to the rapidly increasing number of sensors, this model may be heading towards a dead end [36]. As data loads rise inside the sensor networks, synchronization and real-time reporting becomes problematic. Apart from the drawback of high congestion, we should also consider that failure of the gateway entails isolation of the entire network. In addition, the final scheme is usually application dependent and not easily extensible[1].

[1] For the sake of completeness, we mention the Delay Tolerant Network (DTN) Gateway [5], which is a new, more extensible and performant type of Gateway-style architecture.

The Overlay approach attempts to overcome the issues discussed above. To this end, two different schemes have been proposed: a sensor network overlay on top of the Internet [8] and an IP overlay network over a sensor network [10]. The first case is data-centric and refers to the creation of virtual nodes outside the WSN, forming virtual sensor networks. According to this idea, Internet hosts are equipped with a "Sensor Network Layer" on top of their TCP/IP stack, which enables them to act as virtual nodes of the WSN. In that way, the virtual nodes can understand the sensor messages even if they do not belong "physically" to the WSN. Other special hosts handle the encapsulation of WSN messages into IP packets in order to reach other IP targets. As a disadvantage of this method, we could mention the protocol overhead of the nodes implementing the additional WSN layer. The need of protocol translation is also a drawback, however it is not as expensive as it is inside a single gateway.

The second case is address-centric. Each sensor-node is assigned a unique IP address, becoming equal with all other hosts and acting as a "First-class Citizen" of the Internet [10]. Thus, a remote host can send messages directly to a specific node equipped with a TCP/IP stack, without requiring any intermediate "translator". This particular architecture is of great interest, since it gives independence to each node, provides multiple-path access to the sensor network and relieves the network of congestion problems and expensive synchronization. Furthermore, it gives the opportunity to combine already successful Internet protocols and web services for the node communication. However, serious challenges appear, mainly due to memory and power restrictions of the tiny sensor-nodes. In addition, web protocols and services are often very demanding and the routing of TCP/IP messages may cause high overheads because of the large message headers..

Since recently, 8-bit microcontrollers have been regarded as outdated. However, thanks to features such as their simple architecture and low cost, they were brought back to the fore. Today, devices based on such microcontrollers seem to be the perfect choice for low-power and low-cost applications in sensor networking. The main objective of the work reported here was to expand the applicability of one such device, the well known "AVR Butterfly" [11]. The "AVR Butterfly" (to be called BF for short) is a simple and affordable (approx. 15€) embedded system demonstration kit provided by ATMEL, which is equipped with an 8-bit microcontroller and several sensors.

In this paper we show how we have managed to make the BF embedded system kit Internet-aware. We extended considerably its capabilities by transforming it into a very simple monitoring device and an inexpensive wireless sensor network node. In particular, we installed a TCP/IP stack, added a wireless transceiver and then designed a BF-based inexpensive system prototype which allows Internet users to have immediate access to remote sensor readings via their web browser. While creating a "mote" around the BF, some difficulties arose mainly due to the microcontroller's limited memory and other hardware limitations. After providing Internet connectivity to the BF, in order to evaluate its performance, we constructed a gateway and a Zigbee WSN node. The entities of this architecture, i.e. the Gateway device and the nodes in the Zigbee WSN, are both simple embedded systems built around a BF device. In our implementation we used one of the most efficient TCP/IP stacks for 8-bit and 16-bit microcontrollers: the uIP [12], the popular Ethernet chip: RTL-8019AS [13] and the Zigbee enabled radio transceiver: XBee [14]. The application we have

developed on top of the Gateway's TCP/IP stack, is a basic web server, able to serve static and dynamic pages and to communicate with the WSN nodes.

We should clarify that we do not claim that the AVR Butterfly can form the basis for a more suitable WSN node or gateway than other solutions. However, the metamorphosis of the popular BF kit to an inexpensive WSN mote (with total cost below 40 euros) is by itself a novel contribution and has not been demonstrated before. Furthermore, the applications presented here provide a list of ideas built around the web-enabled AVR Butterfly. The model system designed and developed in this work is inexpensive, highly configurable (it is easy to add more sensors if needed), easy to construct and combines some very popular off-the-shelf components.

After this introductory section, the rest of the paper is organized as follows. Next we describe in brief the main components of the basic scheme and then we refer to some implementation details and results. In the third section, we present the web server application and the user interface. Afterwards, we describe how the baseline system can be extended to implement a Zigbee WSN and present distance control applications. Finally, we conclude by summarizing the paper and point to interesting future work.

2 Basic Scheme

As mentioned previously, the BF is an inexpensive evaluation kit which demonstrates the major features and capabilities of ATMEL's AVR microcontrollers. The main onboard module is the ATmega169 which is an AVR 8-bit microcontroller, running at 8 MHz, with 1 KByte SRAM, 16 KBytes Flash, 512 Bytes EEPROM memory. It also includes a liquid crystal display screen (LCD), a serial port (RS-232), a real-time clock (RTC), temperature, light and voltage sensors, a miniature joystick and a piezo-electric speaker. Additionally, there is support for the Joint Test Action Group (JTAG) debugging interface, the Universal Serial Interface (USI) protocol and in-system Programming (ISP). The advantages of using this platform in applications are its low cost (approx. 15€) and low power consumption, the wide collection of peripherals and the ease of reprogramming due to the embedded Bootloader and the RS-232 converter. Because of the previous properties, the BF has been used as the core unit in several interesting projects such as an MP3 player device [15] and an educational robot called FlutterBot [16].

The RTL-8019AS is a very popular 10Base-T Ethernet controller chip provided by Realtek [13]. It supports Plug and Play (PnP) and full-duplex operation (i.e., sends and receives packets simultaneously), it is NE2000 compatible and has 16 KBytes embedded SRAM for packet buffering. There are many boards equipped with this specific Ethernet controller, which range from advanced commercial solutions, to very simple breakout boards. The Packet Whacker is a simple and inexpensive (approx. 20€) RTL-8019AS breakout board provided by EDTP [17], which contains a 20 MHz crystal, an RJ-45 port and indicating LEDs. This solution gives a convenient way to provide Ethernet connectivity to any microcontroller. Possible alternatives could be other similar boards such as the ETM121 from EMBIN [18] or any old RTL8019-based network card [19], with the appropriate interface to convert its ISA port to a standard input/output (I/O) port.

The uIP is an open-source TCP/IP stack, created by Adam Dunkels at the Swedish Institute of Computer Science (SICS) [20] and further developed by a world-wide community. Its implementation is general, aiming at 8-bit and 16-bit microcontrollers and its code is organized as a software library, providing functions for the communication between the TCP/IP stack and the lower layers. The strong points of the uIP are its small code size and low RAM memory requirements. Moreover, it is fully RFC standards compliant, supporting protocols such as ARP, SLIP, IP, UDP, ICMP, and TCP. It also provides basic mechanisms, such as flow control, packet reconstruction, retransmission time calculation and the ability to communicate with equivalent peers. Apart from its great performance, the uIP has very well structured code, extensive documentation, a wide spectrum of applications and ports for many different devices.

The current project uses the uIP-AVR-0.60 port of uIP, which was developed by Lois Beaudoin [21]. In addition, a web server application was placed on top of the TCP/IP stack, tailored to suit the BF. All software was compiled with the 7.16A version of the Imagecraft ICC-AVR compiler [22].

2.1 Related Work and uIP-AVR

The idea of combining the parts described above, to construct an Ethernet-enabled AVR microcontroller running a web server application is not new. Similar work has been done in: the AVRnet [23], the Embedded AVR WebServer [24], the AVR WebServer Project [19] and the Embedded ATMEL HTTP Server [25]. However, these projects utilize microcontrollers with higher memory capacities (ATmega32, ATmega128), their software is custom-made and the construction of the device is rather advanced. Some other complete solutions like the Ethernut [26] and the WS128 Amber Micro Web Server [27], are commercial products with a much higher price (approx. 100€), which do not allow access to their source code. Apart from the work of Louis Beaudoin, one project of special interest is the Tuxgraphics AVR Web Server [28], which embeds a web server application in the ATmega88, a microcontroller with very scarce resources (8 KBytes FLASH, 1 KByte RAM).

As for the uIP-AVR port of uIP, it contains a device driver implementation for the communication of the TCP/IP stack with the Ethernet controller, a checksum algorithm and a 32-bit addition function. The parts of the code which need modification to match the specific AVR device, include the hardware timer (used to generate periodic uIP calls) and the Ethernet device interface. Other projects which have used the uIP-AVR, are based on the following microcontrollers: ATmega161 [21], ATmega163 [29], ATmega128 [21] and AT90S8535 [21].

2.2 Structure and Difficulties

The Internet-user client makes a data request from a remote host using a web browser. The request message travels through the appropriate router to our Ethernet controller in the form of an IP packet. Then, the Ethernet controller collaborates with the TCP/IP stack and the microcontroller in order to extract the data from the IP packet. In the final step, the application (i.e., web server) processes the request, generates the proper data and forms an IP packet that goes back to the end-user following the opposite direction (Figure 1).

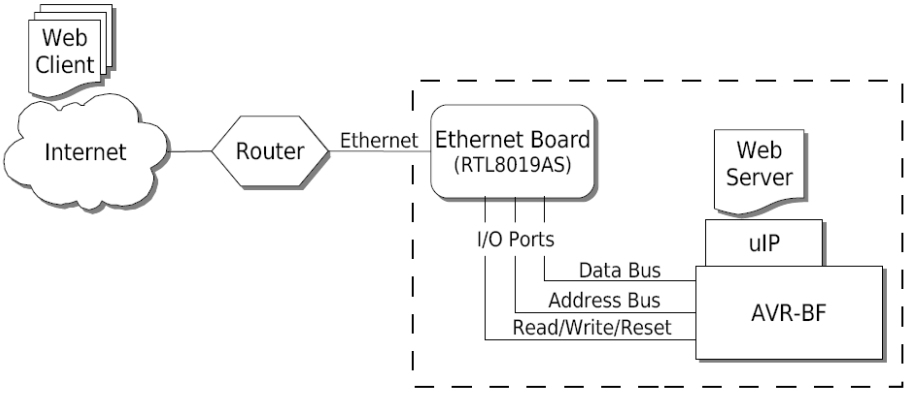

Fig. 1. The general structure of the baseline system

During the implementation of this simple model system, the main difficulties arose due to the low resources of the BF. Additionally, some device dependent modifications were necessary in the uIP-AVR code and special treatment was needed for the compilation of the application code with the ICC-AVR. There were also some hardware issues concerning the availability of free I/O pins and the power supply of the system. Finally, special care was given to the design of the user web-interface.

For the connection between the Ethernet controller and the AVR-BF we used the I/O ports interface approach. In particular, the necessary connections were two buses: one for data and one for addressing and three control signals: Read, Write and Reset. In total, we needed two 8-bit I/O ports for the buses and three I/O pins for the control signals. Unfortunately, most of the I/O ports of AVR-BF were already allocated to the LCD screen and the other peripherals, so not enough pins were actually free for our purposes. However, we could use the I/O pins anyway, changing their previous usage. Hence, we decided to use PortB and PortD, disabling the function of ISP, the rightmost part of the LCD and part of the joystick. In the same way, we managed to obtain three more pins by abandoning the USI feature. As for the device dependent parts of the code, we had to define the appropriate registers for the embedded timer (i.e. Timer0) which makes periodic checks to the uIP and the Ethernet device interface. Among the device dependent parts we must also mention the embedded clock speed and the overflow interrupts which control the hardware timer. Finally, we included the fix of Mattias Rosén for the ARP implementation, that allows packets from different subnets to be received [21].

3 Web Server Application

For the development of the web server application we utilized the sample HTTP server of the uIP-0.60 as the baseline, and then made modifications and simplifications. Some sort of file-system is necessary for a web server in order to be functional. In our case, the HTML pages served by the application are stored in the FLASH memory of the microcontroller converted in lists of hexadecimal numbers. In that

way, the static parts of the pages are stored in a file along with an index that maintains the structure of the files. The role of the compiler is important for the organization of the contents into the RAM and FLASH memory. In compilers such as the AVR-GCC, the "const" keyword can be used in two ways: to declare a constant or to give the instruction to store something in the FLASH memory. However, this could lead to less comprehensible code or misuse of memory instructions. In order to overcome these problems, the 7th version of ICC-AVR compiler introduces the "__flash" keyword, which indicates that a variable, a list or a structure will be stored in the FLASH memory. Therefore, the code of our application was modified according to the philosophy of the ICC-AVR compiler.

In order to get access to sensor data of the BF platform, we must define the source of the appropriate peripheral and then read the value from the equivalent analog-to-digital converter (ADC) channel. The light sensor measurement is in fact the voltage of the light sensor, which decreases as the light intensity increases. As for the temperature sensor, we use a table of constants to convert the voltage of the sensor to temperature in degrees Celsius. The web pages that contain sensor data should be updated dynamically before being served to the remote user. In the sample code of the uIP-0.60, there is an implementation of a simplified scripting language with a CGI-style operation. Another interesting feature would be to support server side includes (SSI), according to which we could use the syntax of HTML comments to import dynamic data [30]. However, neither of these approaches was appropriate for our platform because of memory limitations. According to our custom solution, a dynamic page is marked with a special character in the URL of the page. Then, the code of the web page is split into two pieces: the one preceding and the other following the dynamic entry part. The web page is constructed dynamically and then is divided in packets which are being sent to the recipient the one after the other.

According to the general structure of the code shown in Figure 2, the application first listens to a port. If there is a connection, and an active user makes a new request, then the requested web page is accessed from the file-system. In the case that the page

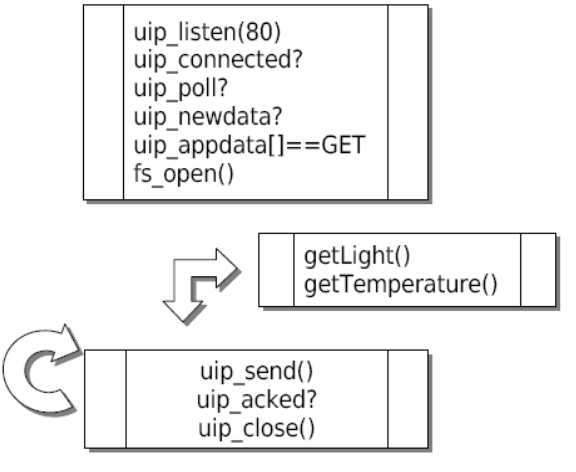

Fig. 2. The top level structure of the web server application code

has dynamic content, sensor data is generated and embedded into the page. Subsequently, successive packets and acknowledgements are being sent and received until all data is delivered to the remote client. Finally the connection is closed.

4 Zigbee Gateway Application

Zigbee is a wireless networking protocol built on top of the IEEE 802.15.4 standard for wireless personal area networks (WPANs). It was developed to meet the needs of low-cost, low-power WSNs and defines the Network, Application and Security layers, while the lower MAC and physical layers are defined by the IEEE 802.15.4. Some key features of this standard are: reliability, security, low power consumption, high number of supported nodes and inexpensive devices. It operates in the unlicensed 2.4 GHz, 915 MHz and 868 MHz ISM bands and enables interoperability between different products. The Zigbee enabled devices have long battery life and low complexity. Furthermore, the WSN becomes self-maintained, reliable and easily expandable thanks to the mesh networking support.

4.1 Baseline System Extension

Additional Hardware. At this point we will present an extension to the basic scheme, in which the Ethernet enabled device described in section 3, takes the role of a Gateway in a Zigbee WSN. The BF is again used to build the nodes of this WSN, equipped with an XBee RF module. The XBee is an RF-device provided by Digi, that implements the Zigbee protocol [14]. Its power consumption is very low and the outdoor line-of-sight of its embedded antenna can reach a range of 90 meters. The XBee is a great solution for the networking of small devices, especially because of its small size and low cost (approx. 15€). As part of another project, a number of special Zigbee WSN nodes were built around BF (XBee-BF) by the Embedded Systems Group at the University of Athens (Fig. 3a). These nodes are autonomous boards which provide interconnection between the BF and the XBee. In addition, the connection with a PC becomes easy, thanks to the serial RS-232 port and the onboard jumpers which set the mode of operation.

An important requirement for a monitoring device is to be power-line independent. Therefore we equipped the BF with a solar cell provided by Olimex, as shown in Figure 3b. This device is intended to be connected with the MSP430 family of Texas Instruments microcontrollers, however it can be arranged to supply power to the BF as well. Ten hours of exposure to sunlight are enough to fully recharge the onboard AA-type battery. Whenever there is no access to sunlight, the BF can use the stored energy for its operation.

Software. In order to combine all the components described above in our final application, we needed to manage the communication of the Gateway with a wireless node. At first, the Gateway was connected with an XBee via UART. Then, we configured the XBee firmware, so that it broadcasts all messages being received. Afterwards, we set up the XBee of the XBee-BF node to communicate with the Gateway, provided the appropriate code for the remote BF and modified the web server application. Finally, we added the code for the distant switch control application.

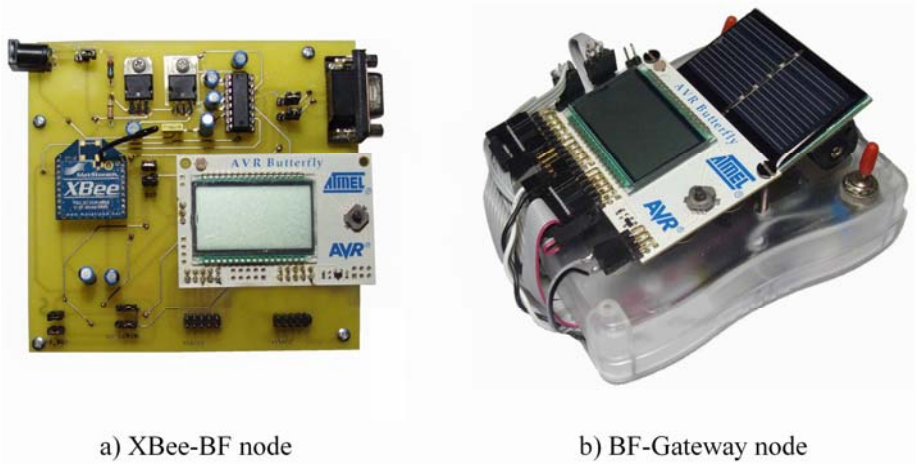

Fig. 3. The developed a) XBee-BF node and b) BF-Gateway node with solar panel

The Gateway was equipped with UART communication functions. When there is a request for remote measurements, the microcontroller sends a "GET" message serially to the XBee, which is then broadcasted. The message reaches the remote XBee-BF board over the Zigbee wireless network and then is forwarded from the XBee to the BF. The BF's microcontroller recognizes the message, collects the proper sensor measurements and creates a string of the form: "R{Temperature}{Light}", which is sent back to the Gateway as a reply. The measurements are then extracted from the message and imported to a dynamic web page.

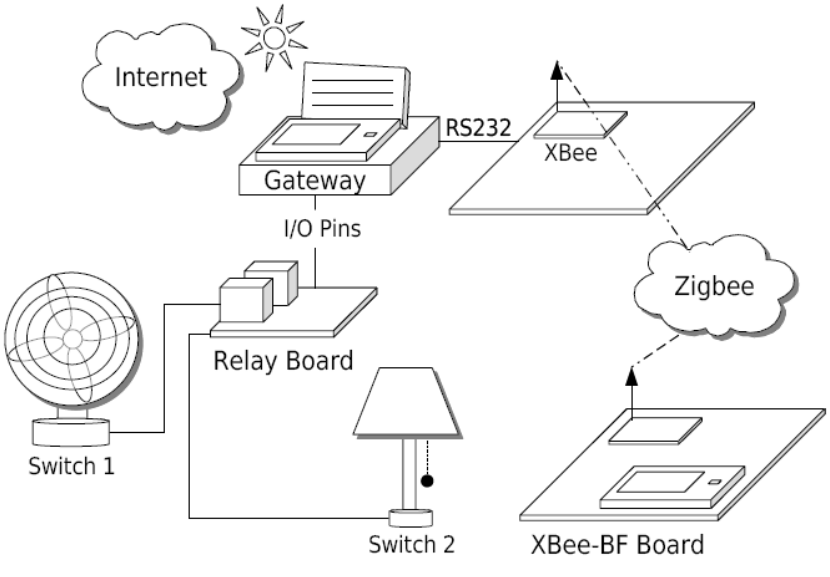

Fig. 4. The Zigbee WSN application structure: Gateway device (BF, uIP-AVR, Packet Whacker, solar cell), XBee-BF board, dual-relay board

We have also developed a relay-based control application. A relay is a switch that can start or stop the flow of electric current in a circuit, depending on the logic level of a signal that controls its input. It can be considered as a simple actuator which can control a high power output with a relatively small power input. Therefore, by connecting a BF to a relay, we can control a switch just by setting an output pin to logic "1". The idea is to allow users to control devices connected to the relay via their web browser. In our implementation, we used a board equipped with two 12V relays provided by Anykits [31]. Every input signal above 2V can trigger the relay and control an external circuit up to 230V. For instance, we could control an air conditioning system or a fire alarm when the temperature rises above a threshold, or manage the lights and shutters of a house according to the indoor light intensity. Two free I/O pins were necessary to connect the dual-relay board to the BF in order to control two devices independently. However, we already used all the free pins except for the JTAG port. In order to use the JTAG pins (i.e., PF4 – PF7) for I/O, we had to disable the JTAG feature which is supported by default. This can be done by writing a function [32] which instructs the appropriate registers to bypass the JTAG in every clock circle.

4.2 Final System Architecture

In the final configuration, we combine all the software and hardware components described above, forming a wireless network of BF nodes (Figure 4).

The web interface presented in Figure 5, was designed to be user friendly and functional, with low requirements in memory. During the system testing, the mean Gateway-to-node range was measured up to 9m. The number of users that can be

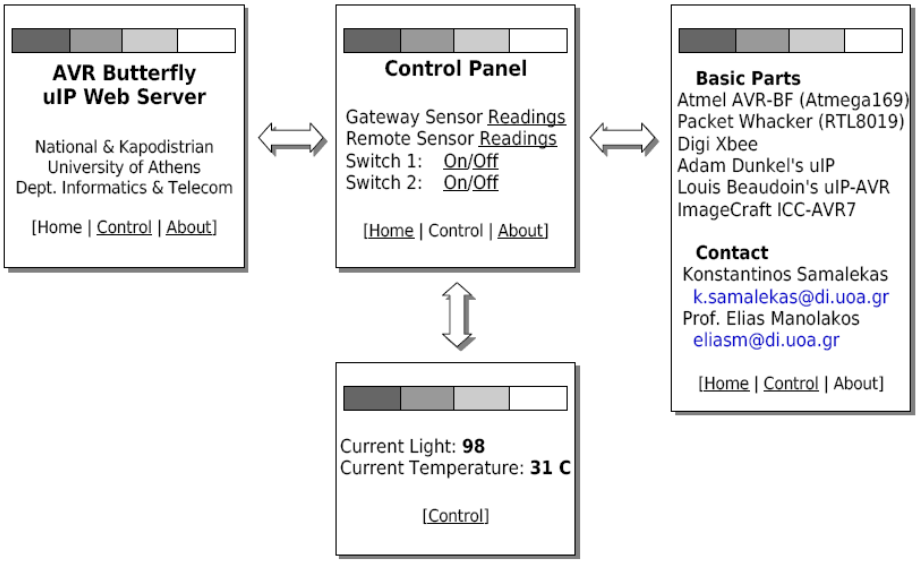

Fig. 5. The sitemap of the web interface

connected to the server simultaneously is restricted due to the limited RAM of the BF. This number depends on the configuration of uIP and the size of the static pages. The key advantage of a web-enabled monitoring and control system is that it can be accessed by any platform equipped with a simple browser. Therefore, the interface was also tested in a mobile phone browser simulator [33].

Another example that could be applied in our monitoring device is a simple wind meter. First we attach a wind propeller in the centre of a simple speedometer [34], then we connect the output of this device to an input pin of the BF. The speedometer is composed of an opto-interrupter and a plastic wheel with a small cut on its edge. The wheel is placed horizontally on a pin, in a way that is able to turn on its centre axe. The opto-interrupter is a component with two poles: an infrared emitter and a shielded infrared detector. By emitting a beam of infrared light from one pole to the other, the sensor can detect when an object passes between the poles, breaking the beam.

The system is arranged as shown in the Figure 6, so that the opto-interrupter can detect the cut in every turn of the spinning wheel. In the BF side we can estimate the wind-speed in Hertz, by counting the interrupts driven from the signals of the opto-interrupter every one second.

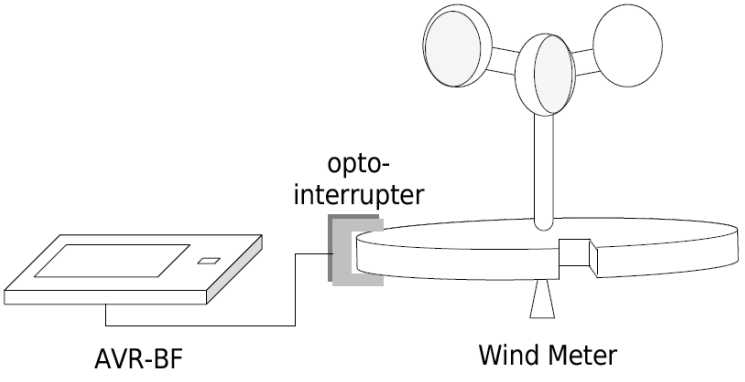

Fig. 6. Wind Meter application

Even though we already used a number of I/O pins related to the LCD screen, it could be possible to utilize its remaining part by writing a new display driver. We could then use text, such as the name of the file being served to the user, the temperature/light readings, or just “ON/OFF” when device’s power is switched on or off respectively.

Moreover, we could reallocate the I/O ports, in order to free and utilize the embedded Universal Serial Interface (USI), to connect additional components. USI is a serial interface protocol, which can provide external connections, operating as Serial Peripheral Interface (SPI) or as Inter-Integrated Circuit (I2C) controller. For instance, we could develop an e-Health application, by attaching devices able to monitor blood pressure or levels of glucose.

5 Conclusions and Future Work

The first contribution in the current study was to provide Internet connectivity to the AVR Butterfly, by installing the uIP-AVR TCP/IP stack. In order to evaluate its further capabilities, as a case study, we have presented an affordable and simple solution for monitoring physical conditions and controlling actuators via the Internet. All units were inexpensive and easy to design, built around the BF. Our final scheme supports wireless node reports, extending the functionality of our example application. We have shown, how BF-based sensor systems could be used for house automation or outdoor area-monitoring, despite their memory and power limitations. Combined with the appropriate components, the web-enabled BF played both the roles of a Zigbee WSN node and a WSN Gateway, providing immediate communication with external users. This simple system also provides an excellent educational platform, where students can learn in practice to develop Embedded-C applications and use different network protocols, getting valuable hands-on experience.

As mentioned before, in order to welcome the "Internet of things" era, we ought to overcome major hurdles and specify which is the most suitable architecture for integrating the WSNs with the Internet. Until recently, researchers have been expressing serious doubts about IP's aptness for low power sensor networks. The main concern was that IP seemed very "heavy" for such applications. Nevertheless, the emergence of light TCP/IP stacks such as the uIP, increased the supporters of this perspective. October 2008 may be the date that all the doubts started dissolving, as Cisco, Atmel, and SICS announced the uIPv6, an extension to uIP which combines IPv6 with the 6LowPan protocol [35]. With all these elements in place, TCP/IP may become a viable solution, even for sensor nodes with very low resources. Nowadays, sensors turn into independent producers of data, receive control, become web-enabled and serve data on demand. This is certainly more than a trend, it is the way of the future.

References

1. International Telecommunication Union: ITU Internet Reports 2005: The Internet of Things. The World Summit on the Information Society (WSIS), ITU, Tunis (2005)
2. Kumagai, J., Cherry, S.: Sensors & Sensibility. IEEE Spectrum 41(7), 22–26 (2004)
3. Park, S., Jayaraman, S.: On Innovation, Quality of Life and Technology of BodyNets. In: Proceedings of the ICST 3rd International Conference on Body Area Networks, ICST, Tempe, Arizona (2008)
4. Shen, X., Wang, Z., Sun, Y.: Wireless Sensor Networks for Industrial Applications. In: Fifth World Congress on Intelligent Control and Automation (WCICA), vol. 4, pp. 3636–3640. IEEE, Hangzhou (2004)
5. Coleri, S., Cheung, S.Y., Varaiya, P.: Sensor Networks for Monitoring Traffic. In: Forty-Second Annual Allerton Conference on Communication, Control, and Computing, Urbana-Champaign (2004) (invited paper)
6. Hac, A.: Wireless Sensor Network Design, pp. 349–367. John Wiley & Sons, Manoa (2003)
7. Römer, K., Mattern, F.: The Design Space of Wireless Sensor Networks. IEEE Wireless Communications 11(6), 54–61 (2004)

8. Zhang, M., Pack, S., Cho, K., Chang, D., Choi, Y., Kwon, T.: An Extensible Internetworking Architecture (EIA) for Wireless Sensor Networks and Internet. In: Asia-Pacific Network Operations and Management Symposium (APNOMS), Poster Sessions, Busan, S. Korea (2006)
9. Karl, H., Willig, A.: Protocols and Architectures for Wireless Sensor Networks, pp. 78–81. John Wiley & Sons, Chichester (2005)
10. Dunkels, A., Alonso, J., Voigt, T., Ritter, H., Schiller, J.: Connecting Wireless Sensornets with TCP/IP Networks. In: Langendoerfer, P., Liu, M., Matta, I., Tsaoussidis, V. (eds.) WWIC 2004. LNCS, vol. 2957, pp. 143–152. Springer, Heidelberg (2004)
11. ATMEL, AVR Butterfly, http://www.atmel.com/dyn/Products/tools_card.asp?tool_id=3146
12. Swedish Institute of Computer Science, Dunkels, A.: uIP, http://www.sics.se/~adam/uip/index.php
13. Realtek, RTL8019AS SA Full-Duplex Ethernet Controller with Plug and Play Function, http://www.realtek.com.tw/products/productsView.aspx?Langid=1&PFid=15&Level=4&Conn=3&ProdID=22
14. Digi, XBee 802.15.4 OEM RF Module, http://www.digi.com/products/wireless/point-multipoint/xbee-series1-module.jsp
15. Brokentoaster, AVR Butterfly MP3, http://www.brokentoaster.com/butterflymp3/index.html
16. Flutterbot, The FlutterBot an Educational Robot Kit, http://www.flutterbot.com
17. EDTP, Packet Whacker, http://www.edtp.com/whacker_page.htm
18. EMBIN, ETM121 Ethernet Connectivity, http://www.embin.com/products_etm121.html
19. Radig, U.: AVR web server project, http://www.ulrichradig.de/home/index.php/avr/webserver
20. Swedish Institute of Computer Science, Networked Embedded Systems Group, http://www.sics.se/nes
21. Louis Beaudoin, uIP-AVR, http://www.laskater.com/projects/uipAVR.htm
22. Imagecraft, ICCVAVR, http://www.imagecraft.com/devtools_AVR.html
23. AVR portal, AVRnet, http://www.avrportal.com/?lang=en&page=avrnet
24. Pfeifer, T.: Embedded AVR Webserver, http://thomaspfeifer.net
25. Tzeming Tan, J., Land, B.: Embedded ATMEL HTTP Server. Master's Thesis, School of Electrical and Computer Engineering, Cornell University (2004)
26. MicroController Pros Corporation, Ethernut, http://microcontrollershop.com/product_info.php?products_id=358
27. SOC Robotics, WS128 Amber Micro Web Server, http://www.soc-machines.com/product/Amber_Specs/Amber_Processor.html
28. Socher, G.: Tuxgraphics AVR web server, http://www.tuxgraphics.org/electronics/200611/embedded-webserver.shtml
29. Ben Zijlstra, Atmel163 With Ethernet, http://members.home.nl/bzijlstra/software/communication/Communication.htm
30. Apache, Server Side Includes, http://httpd.apache.org/docs/1.3/howto/ssi.html

31. AnyKits, Dual Channel Relay Board, http://www.anykits.com/catalog/product_info.php?products_id=310
32. Thomas, M.: AVR Butterfly JTAG Pins For General I/O, http://www.siwawi.arubi.uni-kl.de/avr_projects/BF_JTAG_disable.html
33. Opera Software, Opera Mini Simulator, http://www.operamini.com/demo/
34. Pardue, J.: C Programming for Microcontrollers Featuring AVR Butterfly. Smileymicros, 144–151 (2005)
35. Durvy, M., Abeillé, J., Wetterwald, P., O'Flynn, C., Leverett, B., Gnoske, E., Vidales, M., Mulligan, G., Tsiftes, N., Finne, N., Dunkels, A.: Making Sensor Networks IPV6 ready. In: Proceedings of the Sixth ACM Conference on Networked Embedded Sensor Systems (ACM SenSys). Poster Sessions. ACM, Raleigh (2008)
36. Ali, M., Langendoen, K.: A Case for Peer-to-Peer Network Overlays in Sensor Networks. In: The Sixth International Conference on Information Processing in Sensor Networks (IPSN 2007), Cambridge, MA, USA (2007)