Abstract—Embedded Systems assume an increasing importance in several biomedical applications. These applications present dissimilar requirements and characteristics, posing problems at the computing and communication levels. Low power consumption, size, weight and secure wireless communication are some of its key features.

This work proposes a platform that makes use of an autonomous communication module, and relies on personal digital assistants to act as Masters to interconnect the embedded systems to computer networks. Information is sent to a main server that maintains and provides access to a database. Data security is assured in all systems by using cryptographic algorithms and protocols. Two embedded systems have been developed based on this platform, a simple system for movement monitoring, and another for biomolecular recognition. Experimental results show that it was possible to implement a secure communication channel without sacrificing platform autonomy. Furthermore, we show that the optimization of the communication module lead to a significant increase in the total platform autonomy (about 70% for the movement monitoring system).

I. INTRODUCTION

In the last few years there has been a growing interest on embedded systems for biomedical applications, increasing the demand on computing and communication, but, at the same time, reinforcing the necessity to keep them portable and autonomous. Applications such as biochemical operations for clinical analysis (e.g, glucose/lactate analysis), DNA analysis and proteomics analysis for clinical diagnostics [1], and real-time pervasive patient monitoring and biomedical digital assistants [2] [3] [4] are typical examples where portability and computing power are important requirements. However, namely in the latter case, computing and communication requirements lead to the integration of wireless devices on the embedded systems, in order to communicate with general purpose computing systems. Therefore, the actual embedded systems for biomedical applications have to be designed with low power communication sub-systems and, on the other hand, have to be easily integrated with more general distributed computing platforms. Reliability and security issues have to be considered on those platforms, both at computation and communication levels [5].

This work proposes a general platform for portable biomedical systems supported on an autonomous communication module (ACoM) that relies on personal digital assistants (PDAs) to interconnect the embedded system to computer networks. Methods and techniques are proposed to improve the autonomy and security of the ACoM, which are crucial aspects in these kind of systems.

II. PLATFORM ARCHITECTURE

Figure 1 presents the block diagram of the embedded system architecture. The design comprises two fundamental modules: i) the autonomous communication module (ACoM); and ii) the sensing and processing module (SPM). These two modules compose an embedded system able to communicate with more general computing devices, such as laptops or PDAs.

The ACoM is used to interface any sensing and processing device using standard serial interfaces. It is also able to communicate with Masters, PDAs or laptops, through standard wire or wireless communications systems. In the core of the ACoM there is a data transfer manager and a set of standard communication interfaces. The transfer manager is responsible for communicating data and commands from and to a local SPM, and also to interface the ACoM with the Master (see Fig. 1). Two additional important blocks are present in the ACoM: the power manager and the cryptographic engine. The power manager is responsible for monitoring the state of the system battery and controlling its recharge. Data security is assured by the ACoM by encrypting the acquired data. Public-key or symmetric cryptosystems can be applied, and the secure sockets layer (SSL) can be adopted at the Master’s level. The Master can transfer the acquired data to a remote server using the simple object access protocol (SOAP) and WebServices.

The SPM architecture is generic, the only strict requirement is that it communicates with the ACoM using a standard serial protocol. A required block of this subsystem is a data formatter that serializes the data to transmit. The SPM usually includes a sensor interface block to perform the conditioning of the acquired signals and a programmable digital processor.

The Master device has full control over the embedded system. It must provide a user interface that allows the execution of complex preprogrammed tasks at the SPM. Although the interface is customized to fulfill the requirements of the different types of SPMs, a set of common tasks can be identified, namely the ability to: display the received data in real-time, store and retrieve data from a remote database or from a local repository, encrypt and decrypt the transferred data, communicate data and commands to the ACoM by using.
III. PROTOTYPES OF THE BIOMEDICAL SYSTEMS

In the core of the ACoM is a 16-bit microcontroller Fig. 2(1). The chosen microcontroller (PIC24FJ64GA002) can perform 16 MIPS and provides several communication peripherals as well as advanced power-saving features. It’s power consumption ranges from 50 mW at maximum clock speed to 12 mW in idle mode, or to 30 µW in sleep mode. Several communication peripherals are provided by the microcontroller. The communication signals that travel between the ACoM and the SPM must go through an optocoupler isolation block, in order to reduce the induced noise. This isolation block is also capable of voltage level adaption, allowing the ACoM to interface with devices operating at voltage levels from 3.0 V to 5.5 V.

The power supply circuits depicted in Fig. 2(2) are also managed by the microcontroller. The battery may be charged by either of the available external power sources: USB bus or DC adapter. The microcontroller is also capable of disconnecting entirely the external power supply to prevent the propagation of noise from this source to the SPM. The prototype is equipped with two low noise power rails. Each can provide a current up to 500 mA, and can be adjusted from 1.2 V to 5 V. The digital circuits are powered through a fixed high efficiency 3.3 V, 800 mA rail.

The prototype provides two communication mediums: Bluetooth, assured by a Bluetooth adapter, Fig. 2(3), and USB, assured by a USB serial converter, Fig. 2(4). Both devices are controlled via the microcontroller’s UART peripheral. The Bluetooth module (Bluegiga Tech WT11) requires an area of 35×14 mm and operates from a 3.3 V power rail. The module fully implements Bluetooth 2.0 standard with Enhanced Data Rate (EDR) allowing up to 3 Mbps data rates. The device is Class 1 compliant with a maximum range of 300 m. Even though the module has a peak power consumption of 560 mW during transmission, it has much lower power requirements in idle mode, 10 mW, or in sleep mode, 1.2 mW. The USB serial converter used (FT232RL) is USB 2.0 full speed compatible, supporting data rates of up to 1 Mbaud. This converter not only provides the standard UART signals but also the control signals that allow the microcontroller to monitor the USB connection status. The control signals are used to adjust the battery charge current and to determine which will be the active communication module.

Communication between the ACoM and the SPM, or between ACoM and a remote master, is achieved using a simple protocol. The protocol consists of two types of packets; firstly a command is sent using a fixed length packet, and then the sender waits for an acknowledgement from the receiver. Finally, the data is sent in a variable length packet. When communicating with a remote master this data should be encrypted by the ACoM’s cryptographic engine to ensure its confidentiality.

The ACoM software was mostly written in C, however some of its critical parts were coded in assembly language. The Encryption module is an optional software component that encrypts the information before sending it to a remote master. This module can be turned off in order to save power when the security requirements allow it.

A. Security

The encryption module is a software module that encrypts or decrypts data using the advanced encryption standard (AES) algorithm, which is a suitable symmetric key algorithm to protect sensitive electronic information up to SECRET level, if a 128 bit key is used [6], and TOP SECRET information if 192 bit keys or 256 bit keys are adopted. The prototype developed for this work uses a 128 bit implementation of AES. In order to prevent the key from being discovered, the master periodically generates a new key randomly, and uses the still secure connection to send this new key to the cryptographic module of the ACoM. This encryption module was built using Microchip’s cryptographic libraries, specifically designed for PIC microprocessors [7].

B. Techniques for lowering power consumption

Two of the four blocks depicted in Fig. 2 have significant power consumption: the microcontroller, Fig. 2(1), and the Bluetooth module, Fig. 2(3). The power supply circuits (Fig. 2(2)) are already optimized at the hardware level and no software optimizations are available. The USB interface is powered by USB, having no effect on the device’s autonomy.

The used Bluetooth module provides several options to lower power consumption, namely the use of low power modes. The module provides an active mode (default mode while communicating), idle mode (default mode when no connection is active), sniff mode, park mode and a deep sleep mode. When the device is in sniff mode it only listens to the piconet at certain intervals, thus trading bit rate for power consumption. These intervals can be configured by the user at run time, enabling the user to shape this interval according to the data rate required at that particular time. In the park state the device does not participate in the piconet traffic. This mode requires less power than sniff mode, but is less flexible. Furthermore the device can also enter a deep sleep mode, which requires the lowest power.

Another technique for lowering power consumption is lowering the device’s power class. Even though the used module
is a class 1 device with a range of up to 300m, it allows the user to change its class to a less power demanding one. In this prototype the bluetooth module was used as a class 2 device, which should be enough to achieve a 10 m range, deemed suitable for a large number of applications.

At the microcontroller (PIC) level, some techniques can also be applied for lowering power consumption, namely clock throttling, idle mode and sleep mode. Lowering clock speeds is a well known method of lowering power consumption, however lowering a processor clock speed does not necessarily increase a portable device’s autonomy, as a lower clock speed makes computations last longer. While the PIC is in sleep mode its clock source is shutdown, thus powering down the whole processor and all of its peripherals. Idle mode is somewhat similar to sleep mode, only the clock source is not shutdown, but the clock is prevented from reaching both the CPU and a configurable number of peripherals. Even though the idle mode consumes more power than sleep mode, it has far more flexibility.

C. Sensing and Processing Modules

The architecture described in the Section II was applied to develop an embedded portable system for biomolecular recognition and a movement monitoring system.

Biological Analysis: The microsystem for biological analysis is based on the architecture proposed in [1]. This system is used to perform assays that consist on a biological reaction that allows the detection of unknown biomolecules (e.g. human DNA strand for genetic disease detection or bacteria/cell detection).

In the core of this analysis system is a 16-bit Digital Signal Processor (dsPIC30F6014). This device can perform up to 30 MIPS and includes peripherals that are used to control the onboard electronic components and also to communicate with the ACoM. The interface with the ACoM is performed using one of the UART peripherals available in the processor. The digital signals are processed locally in order to reduce the bandwidth required to transmit the measured data. This process leads to only one or less samples per second per sensor. This allows the ACoM to use a lower bit rate, which increases the system’s autonomy.

Movement Monitoring: The movement monitoring is performed by a 3D accelerometer included in a small electronic module realized with 2 micro machined chips 2D accelerometers (placed in orthogonal planes) and a microcontroller [8].

D. Master Device

The Master device was implemented on a Pocket Loox 720, with an Intel XScale PXA 272 520 MHz processor, 128 MB RAM memory, Bluetooth 1.2 and USB1.1 host capabilities. To allow the user to interact with the ACoM, a Graphical User Interface was developed. This interface gives the user the possibility to configure the experience, watch the experience unfold by examining the incoming data in a real-time graphic and send the received data to a remote database through the internet using SOAP. Furthermore it is also possible to store the data locally on the PDA. As the ACoM is a generic platform for accessing all manner of biomedical SPMs, its interface must be easily extendable in order to accommodate the specific needs of every SPM. However, most of the underlying functionality remains the same for every application. The entire user interface was written in C#, using the .NET compact framework 2.0 for increased portability. For this implementation a MySQL database was set up on a remote machine, and the PDA communicated with the database through a PHP server (also set up on the remote machine). The connection between the master device and the PHP server was done in SOAP.

IV. Experimental Results

For the autonomy tests the ACoM was configured to reply to the master’s commands with a package containing 16 bytes worth of data (16 bytes is the block size used by AES). A command was sent every 5 seconds, until the battery fully discharged and the ACoM stopped replying. The PIC microcontroller should wait in idle between commands. The result can be seen in tab. I. In all tests the bluetooth device was configured as a class 2 device (range 10m), and to not reply to service discovery inquiries. The PIC’s UART peripheral was configured with a baud rate of 115000 symbols per second. The baseline test was done without any optimizations in both PIC and bluetooth module, serving as a reference for the other tests. In test 1 the bluetooth module was configured to operate in sniff mode, with a sniff parameter of 80, but the PIC was not optimized in any way. In test 2 the bluetooth module configuration was maintained (sniff mode 80), however the PIC was configured to shut down all peripherals when it entered idle mode, except the UART peripheral. The conditions for test 3 were the same as in test 2, but the bluetooth module was ordered to enter a deep sleep state whilst it waited between packets. In test 4 no optimizations were performed on the bluetooth module. The PIC was configured as in tests 2 and 3, shutting down all peripherals except the UART between packets. Test 5 used the same configuration as test 1 for both the microcontroller and the bluetooth module, only the sniff parameter was changed from 80 to 40. Finally, test 6 evaluates the impact of clock frequency: in this test the device was configured as in test 2, only the PIC’s clock frequency was scaled down to 1/2 of its value. All autonomy tests used the same 3.8 V Li-ion battery, with a stated capacity of 630 mAh.

<table>
<thead>
<tr>
<th>Test</th>
<th>Autonomy</th>
<th>Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>baseline</td>
<td>19h20m</td>
<td>-</td>
</tr>
<tr>
<td>test 1</td>
<td>21h16m</td>
<td>11,9%</td>
</tr>
<tr>
<td>test 2</td>
<td>33h16m</td>
<td>82,4%</td>
</tr>
<tr>
<td>test 3</td>
<td>34h45m</td>
<td>79,7%</td>
</tr>
<tr>
<td>test 4</td>
<td>19h57m</td>
<td>3,2%</td>
</tr>
<tr>
<td>test 5</td>
<td>33h55m</td>
<td>75,4%</td>
</tr>
<tr>
<td>test 6</td>
<td>35h41m</td>
<td>84,6%</td>
</tr>
</tbody>
</table>

Analyzing the table I it is possible to see that the best re-
sults occurred when low power consumption techniques were applied to both PIC and bluetooth module. The introduction of deep sleep in the bluetooth module produced no significant impact. Halving the sniff parameter produced only a slight decrease in the system autonomy. A 50% reduction of the device’s clock frequency resulted in a rather small increase in the system autonomy.

To determine the impact of the cryptographic engine the test 2 depicted in tab. I was redone, this time having the ACoM encrypt the data before sending it to the master device. The system’s autonomy was reduced to 34h 52m, meaning that encryption has no significant impact on system autonomy.

The effective data rate of the ACoM was also measured and results are presented in table II. In this test, test A, a fairly large amount of data was encrypted by the ACoM and sent over bluetooth to the master in 16 byte chunks using the protocol described earlier. The time between the sending of the request for data and the reception of the last data chunk was measured, determining the effective data rate. Test B measures the maximum effective data rate the chosen protocol allows, when the data is not encrypted and is gathered in chunks of 1020 bytes, the maximum allowed by the protocol. For both of these tests, the serial port being emulated by the bluetooth connection was configured with a baud rate of 9600 symbols per second.

<table>
<thead>
<tr>
<th></th>
<th>sniff mode off</th>
<th>sniff mode 40</th>
<th>sniff mode 80</th>
</tr>
</thead>
<tbody>
<tr>
<td>test A</td>
<td>4036</td>
<td>3098</td>
<td>2106</td>
</tr>
<tr>
<td>test B</td>
<td>5913</td>
<td>5098</td>
<td>4036</td>
</tr>
</tbody>
</table>

Table II shows that the data rate of the chosen protocol heavily depends on the amount of data sent in each packet. Furthermore it is also possible to conclude that, as expected, operation in sniff mode slows down the speed at which the data can travel between the master and the ACoM.

Figure 3 shows snapshots of the master’s GUI. In Fig. 3(a) one can see the configuration of the serial port, while in Fig. 3(b) it is possible to see data acquired by a Biochip SPM being displayed on a real-time graphic. This real-time display was built from scratch and supports several data series, zooming and scrolling.

**Biological Analysis:** The introduction of the techniques described earlier in the biomolecular recognition system improved it’s autonomy by 11,4%.

**Movement monitoring:** The improvement for the movement monitoring system was far more substantial, increasing the autonomy of this system by 71,7%, from 13h 50m to 23h 45m.

V. CONCLUSIONS

In this document we explored several techniques, mainly at software level, for increasing the autonomy of portable embedded biomedical devices. The results show an increase in autonomy of 11,4% in a biomolecular recognition system and 71,7% in a movement monitoring system. We have also implemented a cryptographic engine to ensure secure transmission of sensitive information between the portable system and a master device. Moreover, we have programmed a software library that implements on a PDA the core functionality of a master device. Amongst the implemented functions are secure communications, real-time data display and a connection to a database using web services. All these functions were wrapped in an user-friendly graphical user interface.

REFERENCES


