Sensors and Actuators A 156 (2009) 49–58

Contents lists available at ScienceDirect

Sensors and Actuators A: Physical

journal homepage: www.elsevier.com/locate/sna

A miniaturized wireless control platform for robotic capsular endoscopy using advanced pseudokernel approach

E. Susilo[a,b,*], P. Valdastri[a], A. Menciassi[a,b], P. Dario[a,b]

[a] CRIM Lab, Polo Sant'Anna Valdera, Scuola Superiore Sant'Anna, Pisa, Italy
[b] Italian Institute of Technology Network, Genova, Italy

ARTICLE INFO

Article history:
Received 26 September 2008
Received in revised form 30 January 2009
Accepted 31 March 2009
Available online 23 April 2009

Keywords:
Pseudokernel
Real time embedded system
Robotic capsular endoscopy
Wireless miniaturized control board

ABSTRACT

A ZigBee compliant wireless controller which manages multiple tasks, such as acquiring data from sensors and driving actuators, has been purposely developed for robotic capsular endoscopy applications. Preemptive priority pseudokernel, consisting of state-driven code, coroutine, and pooled-loop algorithm, has been implemented to perform hard, firm and soft real time applications. All the components have been placed on a miniaturized board ready to be integrated in a robotic capsule (max volume of 2 cm^3).

© 2009 Elsevier B.V. All rights reserved.

1. Introduction

Wireless capsule endoscopy is considered a dramatic breakthrough for gastrointestinal (GI) tract cancer screening [1]. Nevertheless, due to the passive peristaltic propulsion traditional capsule rely on, some interesting spots of the GI tract may be missed by the diagnostic device. This motivates increasing efforts in providing advanced on-board actuation and sensing capabilities to endoscopic capsular systems [2]. Having more than one independent means of actuation on-board would also enable tissue sampling or camera steering capabilities.

Capsular endoscopic variants have been developed by the authors to obtain better approach to different targeted sites. Based on the locomotion propulsion, there are the legged endoscopic capsule crawling along GI tract [2], the submersible capsule swimming and submerging in the liquid flooded stomach [3], and the magnetic locomotion capsule travelling along digestive tract utilizing external magnet [4]. In order to provide capsular endoscopes with enhanced features, the authors also developed a wireless clip releasing pill [5] and an auto-focus video capsule [6]. Peripherals used by all these capsules vary from microactuators to microsensors while the proposed main controller remains the same. Therefore, a fully customizable real time control system is proposed to well suit both hardware and software requirements.

A miniaturized wireless control system based on a ubiquitous 8051 architecture implementing a pseudokernel firmware structure is presented in this work. Aside from the low cost, this 8-bit Harvard architecture-based microcontroller has been widely used in many industries, medical equipments, and automotive applications. The developed platform is able to drive brushless direct current (DC) motors and/or brushed DC motors and to manage different sensor signals simultaneously. Furthermore, since a ZigBee compliant hardware has become one of the embedded features of the System on Chip (SoC), network applications for capsular endoscopy can also be envisaged, as suggested in [7].

This article is organized as follows. Sections 2 and 3 give hardware and software overview. Section 4 explains the essential of real time design with or without real time kernel. Section 5 introduces the architecture, abstraction layer, and the importance of preemption within pseudokernel approach. Section 6 briefly describes about wireless networking support. Sections 7–9 discuss about experimental results, discussion, conclusion and future work.

2. Hardware overview

Having many variants of wireless endoscopic capsule to carry particular tasks requires dedicated control systems. However, these control systems can be generalized from hardware viewpoint since most of capsular devices have common actuators and analog/digital sensors. A set of miniaturized control board was designed and developed in-house. The realization of the control board was based on state of the art off the shelf components to shorten time-to-fabrication.

* Corresponding author at: CRIM Lab, Polo Sant'Anna Valdera, Scuola Superiore Sant'Anna, Pisa, Italy. Tel.: +39 050883489.
E-mail addresses: ekawahyu@crim.sssup.it, ekawahyu@yahoo.com (E. Susilo).

0924-4247/$ – see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.sna.2009.03.036

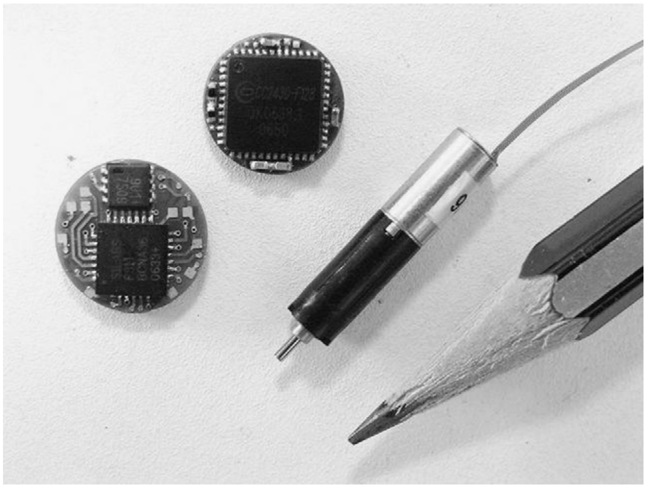

Fig. 1. Developed C8051F311 and CC2430 boards, next to the 4 mm in diameter and 16.2 mm in length Namiki brushless DC motor SBL04-0829PG79.

The latest well tested prototypes are described in this paper. The first one is built using C8051F311 microcontroller (Silicon Laboratories, USA) and 6 pieces of MOSFET drivers IRF7509 (International Rectifier, USA). The aim is to control common peripherals for capsule actuation and acquire data from sensors connected to digital/analog ports. The second prototype, also based on 8051 family, exploits CC2430 (Texas Instruments, USA) as the core component. The CC2430 was selected as for the wireless controllability over 2.4 GHz ZigBee compliant radio hardware, the small form factor of System on Chip (SoC) and qualified dimension to be deployed inside a capsular device. Needless to say that power management was also being considered as an important feature of this chip. This wireless device was proven to work from inside the GI tract during *in vivo* test [7]. As for the high electrical current driving demand, yet small in Surface Mount Device (SMD) package, A3901 (Allegro Microsystems Inc., USA) was selected. Thanks to the selected components it was possible to pursue the high miniaturization that is required for endoscopic capsular applications. Fig. 1 depicts both prototypes together with the SBL04-0829PG79 brushless DC motor having 4 mm in diameter and 16.2 mm in length.

The board design and specification are critical due to the space constraint, therefore circular shape was selected to conform tubular space of capsular devices or, more generally, endoluminal instruments for the GI tract. The former prototype is 10.8 mm in diameter, but it does not have embedded radio feature. Instead, the latter, which has slightly been reduced into 9.6 mm in diameter, has an IEEE 802.15.4 compliant transceiver. Both of them are constructed in four layers rigid flexible circuit material with total thickness ranging from 2.35 mm to 2.5 mm after components are mounted on top and bottom layers. This thickness varies during components mounting due to the in-house hand-soldering and on-site baking process. Regarding layer naming and functionality, all layers are being used as signal planes. Only a small region on the third layer is left out as a ground plane referring to the application note given by Anaren [8]. This ground plane is recommended to proper matching the impedance of BD2425N50200A00 (Anaren, USA) to the whip antenna.

Actuation devices attached to this system must conform the allowance of continuous direct current given by A3901 specification sheet [9]. The range of possible actuation devices are varying from 6 pieces of brushed DC motors, 2 pieces of brushless DC motors or combination of them. The aforementioned prototypes can drive all of those actuation devices simultaneously thanks to the pseudokernel approach.

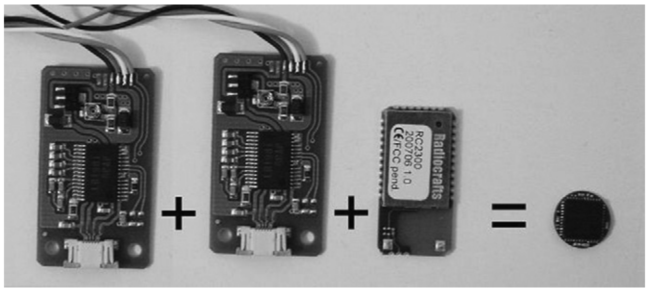

Fig. 2. Two Namiki motor drivers SSD04 and one Radiocraft RC2300 can be replaced by one developed wireless prototype.

The brushless DC motor control method can be selected either Back Electro-Motive Force (BEMF) feedback or slow speed stepping mode. The brushless DC motors selected for the system (SBL02-06H1PG79 having 2.4 mm in diameter and 10 mm in length; and SBL04-0829PG79 having 4 mm in diameter and 16.2 mm in length by Namiki Precision Jewel Co. Ltd., Japan) can be driven precisely with a resolution of 0.76° per step. Stepping resolution for different type of brushless DC motor can be obtained by calculation, based on the number of teeth of the built-in reduction gears and the number of commutation phase of the motor. On the other hand, the method of controlling brushed DC motor can be selected between standard ON/OFF or Pulse Width Modulation (PWM). This method is quite common to control brushed DC motor and is applicable as well to retract or elongate Shape Memory Alloy (SMA) wire if necessary.

For robotic applications, position sensors are required to implement a reliable closed loop control of the actuating system. In the present work, we used magnetic encoders and accelerometer, which do not demand so much hardware resources. Therefore, a proper serial or parallel connection would be sufficient to set them ready and data acquisition can be done through it. In some applications of wireless capsular endoscopy, for example endoscopic legged capsule, it is important to know the position of each leg during locomotion and this can be done by placing external position encoder. A proper gait locomotion has been studied in [10] to lessen accordion effect during locomotion in GI tract, but further improvements to eliminate this effect can be achieved by measuring the acceleration during locomotion through accelerometer. The acceleration data acquired in real time can be used by the locomotion gait generator to adapt itself to the arbitrary environment in the GI tract.

Fig. 2 shows the physical comparison of the small form factor aforementioned prototype which may substitute three commercially available boards (two boards SSD04, Namiki Precision Jewel Co. Ltd., Japan and one board RC2300, Radiocraft, Norway) at once to be used inside wireless endoscopic capsule.

3. Software overview

An embedded system for wireless capsular robot must deliver fast response time as possible to handle time critical data. This kind of system has very strict time deadlines that must be met every time [11]. Common peripherals in capsular endoscopy are actuators and sensors which are time critical and may not drop data during data acquisition or control process. Thus, a real time embedded system consisting of firmware for particular peripheral handlings is necessary.

An embedded system is defined as a fully functional hardware and software performing computation in order to resolve a specific tasks. Most of embedded systems implement real time tasks, but almost none of them comes with real time operating system (RTOS). Unlike personal computer (PC), embedded system does not allow different application to be loaded and peripheral to be attached during runtime. Therefore, implementation of functions, loops, and event-driven code merely work on embedded system. Instead in PC, an operating system is preferable to support a more wide range of applications.

Some devices lie between the world of embedded system and PC. A device may have a very high degree of computation and complexity, but its definition is far below the complexity of a PC. It might be tempting to take advantage of RTOS over an infinite loop and some interrupt funtions. However, the running task under an RTOS obviously would react less responsively for wireless endoscopy application. By definition, the kernel should support multiple applications running at the same time. This involves at least task creating, task scheduling and context switching. Compared to just a simple infinite loop and functions, deploying RTOS to an embedded system would require a lot of central processing unit (CPU) and memory resources.

According to the survey conducted in [12], deploying an RTOS is not a major requirement for an embedded system. The survey said that more than a quarter of embedded systems under development will not have operating system prior to the first production. A tiny scheduler or task switcher is more than satisfying and the reasons of not implementing real time operating system were varied. About 30% said an operating system would put too much strain on their system's processor and/or memory, with about half blaming the CPU and the other half the memory. A much smaller percentage, about 10%, said an operating system was too expensive. The remaining 7% complained that operating systems are too complicated to use. Note that this final group did not say that an operating system was too complicated for their system. It is said that it was too complicated for them personally. Whether it is true or not, this last response suggests that some OS-friendly marketing campaigns may be in order. Therefore, a pseudokernel approach is proposed in this paper to address a complex embedded system, yet real time, for wireless capsular robotic devices. Pseudokernel approach resembles, in a far simpler way, an operating system kernel. The number of tasks is decided to be fixed during source code compiling which obviously requires less resources. The complete architecture of the pseudokernel is explained in Section 5.

Regarding the depicted facts and the proposed hardware specification, implementing commercially available real time operating system in general is not a viable solution for wireless capsular devices. This is due to the fact that the amount of resources are limited, such as the amount of memory, 8-bit architecture only, and yet it should handle very strict time deadlines. A proprietary pseudokernel approach is the best option to manage particular tasks and peripherals. This work has been inspired by the famous RTX51 tiny [13,14], the opensource and royalty free FreeRTOS [15] and last but not least PaulOS [16].

4. Real time system design

Real time systems are classified as hard, firm, and soft depending on task deadline requirements. In hard real time systems, the most critical type, failure to meet its time constraints will lead to system failure. In firm real time systems, the time constraints must be met in most cases, but can tolerate missing a low number of deadlines. In soft real time systems, the performance is degraded when time constraints are not met, but the system will not catastrophically fail [11,12]. Some implementations can be very powerful and robust, while other implementations can be very simple, and suited for just one particular purpose. This section reveals the design of dedicated real time system for wireless capsular robotic applications.

4.1. Essentials of real time kernel

A real time kernel could act in a real time manner thanks to the support from its basic elements. There are five essential elements of a real time kernel: Task Manager, Time Manager, Memory Manager, Message Manager and Context Switcher. Comprehensive explanation on real time kernel is not within the scope of this article, but please note carefully that the dynamic application creation and dynamic content manipulations are the main issues why real time kernel is present in some of complex embedded system. The number of created application depends on the amount of memory available during runtime and it is important to destroy completed application afterward. Abandoning an application in the memory after its completion may prevent a new application to allocate itself, which surely leads to task creation failure. Task scheduling plays an important role in seamless multitasking process between applications, while context switcher preserves and recovers its context during application take-over. This makes all applications are perceived by the user as running simultaneously.

Real time kernel is embedded in every RTOS as the most fundamental component, disregard for commercial or opensource one. The dynamic application creations during runtime would require extra resources to be deployed in an embedded system. A new created application would occupy memory space at least for runtime variables, context switching and message queueing. The only advantage of having RTOS is that applications become more manageable in a structured manner. But system overhead appears to be unavoidable due to kernel-application performance, especially for a hard time critical application of wireless capsule robot. On the other hand, without RTOS, coding would become more complicated for large application and further development would probably require a total modification from scratch. The proposed pseudokernel approach can intermediate these shortcomings and find the right balance between RTOS and simple conventional coding.

4.2. Real time multitasking without RTOS

Real time multitasking can be achieved without interrupts and even without operating system per se. When feasible, these approaches are preferred because resultant systems are easy to analyze [17]. This method only resembles some kernel actions and is called pseudokernel. In this approach state-driven code, coroutine and pooled-loop are deployed with no preemption capability. Since some peripherals need to meet a certain deadline, preempting a running application is also preferred in the system. As far as authors' knowledge, there is no written documentation about the combination of both methods successfully applied in a real time system for biomedical application. In this work, preemptive priority pseudokernel, consisting of state-driven code, coroutine, and pooled-loop algorithm, is proposed to perform hard, firm, and soft real time applications for endoscopic capsular robot.

The proposed system can exhibit multitasking behavior without an RTOS. What is seen as a multitasking process is actually a CPU time sharing that resembles several tasks running at the same time. The 8051 architecture supports only one single task running at a time, therefore the idea of task scheduling is adopted for this system and implemented in a straightforward way.

Minimal memory footprint and fast response time have become main goals. Thus, predictions and calculations on the worst case time required for hard time critical application to reach its deadline is mandatory. This determines the lowest time interval between tasks, called as time tick. Time tick is customizable under pseudokernel configuration file and can be inserted during pseudokernel compilation.

Designing a real time system to support real time applications requires comprehensive knowledge on peripherals specification and behavior. Some peripherals may demand much CPU computing power in order to work properly and other may accept certain delay time. Therefore, real time programmers must know the worst case time required for each peripheral, keeping the CPU busy all the time, and decide the right balance among applications.

5. Pseudokernel approach

For all variants of the capsules, brushless and brushed DC motors are selected as actuators which demand a hard time critical handling. Motor control application comprising actuation devices and position sensing are categorized as hard time critical application. The brushless DC motor needs to know the internal rotor position to produce higher torque and the required speed. Failure to control motors to respective position may lead to catastrophic events during surgical intervention. Hence, preemptive priority must be implemented for actuation device task, comprising motor control and position feedback sensing. Other peripherals, such as lighting and liquid lens with auto-focus system [6] are considered as less critical (firm handling), meanwhile sensors reading comprising of temperature and battery monitoring are the least critical among all (soft handling) and dedicated as background processes.

This section explains in detail about the architecture, device drivers and fixed priority preemption development of the proposed pseudokernel.

5.1. Pseudokernel architecture

The architecture of proposed pseudokernel is presented in Fig. 3. There is nothing actually new with this kind of architecture at a glimpse, but the concept behind it makes pseudokernel approach running faster then a standard kernel. Pseudokernel does have task manager, but it does not contain task scheduler which also means that context switching does not necessarily exist. This ensures that very less memory footprint is created during runtime because context switching happens only for native function call. Since the number of task is fixed, task manager does a very simple job to activate, deactivate, pause, or resume the existing tasks. However this comes at a price, since every task should be event-driven and must be written in finite-state-machine. The event-driven finite-state-machine (or also called state-driven) is more pseudokernel-compliant than an infinite loop. Intertask communication is done by single bidirectional thread/bus messaging service. User applications may create their own message cache when necessary. Messaging service is also used to deliver system calls, which are indeed a more secure way to communicate to the pseudokernel. Direct system call would still be available and allowed as a quick system call, but it is not recommended for reliability reason. Coroutine implementation is inspired by Duff's device loop-unwinding theory [18], but further investigation is still in progress to merge it perfectly into the system. Fig. 4 gives a system overlook with some user application examples.

5.2. Device drivers

The pseudokernel provides an abstraction layer of system hardwares in the form of device drivers. They can be customized during compilation through one configuration file. Parameters in the configuration file, such as the time tick, are fixed and can not be changed during runtime.

5.2.1. Wireless communication protocol

Inside the CC2430, there is an embedded radio device implementing hardware finite state machine to alternate between radio data transmission and reception. Together with this state machine, wireless communication protocol was developed. Two tasks comprising radio tramission and radio reception work together in coroutine manner. The basic radio hardware protocol and pseudokernel wireless communication protocol can be seen in Fig. 5. The radio message of 128 bytes [19] consists of message header, message body, data body and message footer. In the message header, there are only information about message destination and the sender. The message body may contain anything up to 118 bytes, the data body is optional and reserved for future development. The last part of the protocol contains information about acknowledgement and number of retries generated by this communication driver, and hardware CRC generated by CC2430. Everytime a message is received in the RXFIFO buffer, it will be dropped automatically when the destination does not match to device address or it does not belong to the cluster. The communication protocol also supports network message broadcasting to all devices in all clusters or to all devices within one cluster.

5.2.2. Sensors and actuators drivers

Driving locomotion actuators, either using brushed or brushless DC motors, require additional components which are three pieces of dual H-bridge motor drivers A3901. The custom made device driver should manage and set device allocations referring to the configuration file. Several functionalities might be overlapped due to resources sharing between brushed and brushless DC motors. In order to avoid system crash, application must check for resources

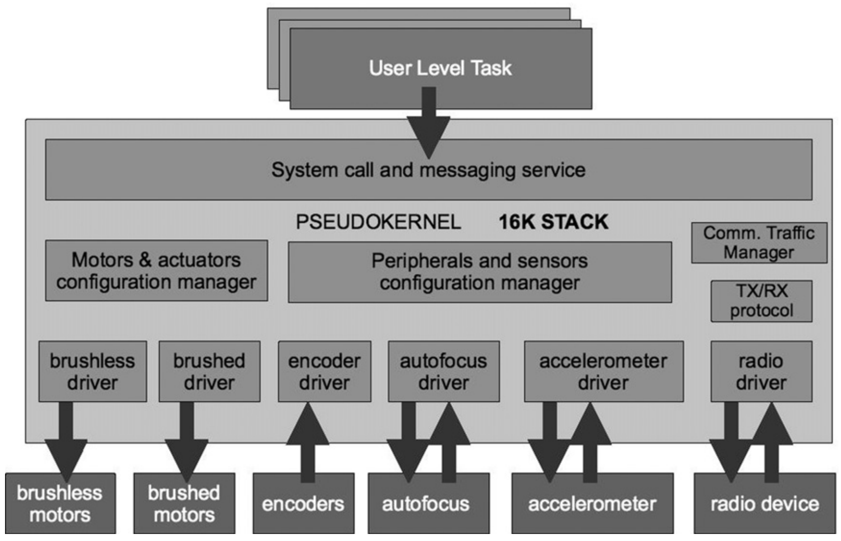

Fig. 3. The 16 kB pseudokernel architecture.

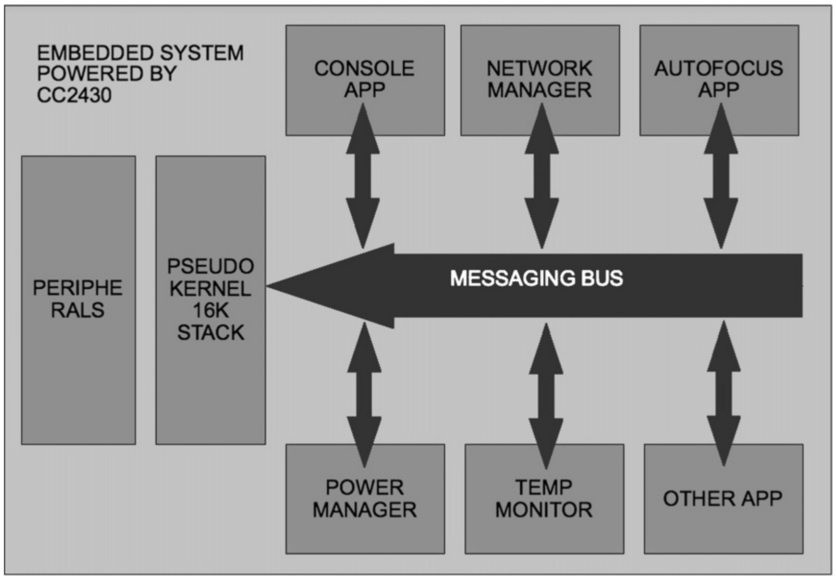

Fig. 4. Embedded system overlook with application examples.

availability and lock it. After the task is executed, the resources should be released so that other application may use them. Possible resources configuration can be selected among:

- One brushless and three brushed DC motors.
- Two brushless DC motors.
- Six brushed DC motors.

Management of the motors is done through setting up the appropriate pseudokernel system flags. Motor driver is categorized as hard time critical application since it implements BEMF commutation. The driver was built to drive brushless DC motor without using the neutral pin as described in [20,21]. This way, several brushless DC motors can be driven at once by just one board.

Position encoder used in this system is an incremental electronic sensor. The sensor delivers incremental pulse in the order of milliseconds and categorized as firm time critical task. However, since the incremental counter correlates to motor control, it would be better for the sensor to inherit the same priority level as motor control to avoid priority inversion. Therefore, motor driver along with external position encoders are placed in the same thread.

Liquid lens driving and accelerometer sensing utilize dedicated PWM and serial communication in the CC2430. Device drivers for both of them are categorized as less critical for current applications and can be reached by changing respective flags through messaging service.

5.3. Preemptive priorities

It is preferable to have a pseudokernel approach to run preemptive tasks for hard time critical applications. The CC2430 provides four levels of interrupt priority which correspond to hard, firm or soft time critical applications. In this approach, the pseudokernel is divided into four fixed priority threads. Soft time critical applications, such as battery and temperature monitoring, reside at the lowest level thread. User applications that are called as background processes should be placed in this level as well. At the first level thread, resides the majority of pseudokernel tasks such

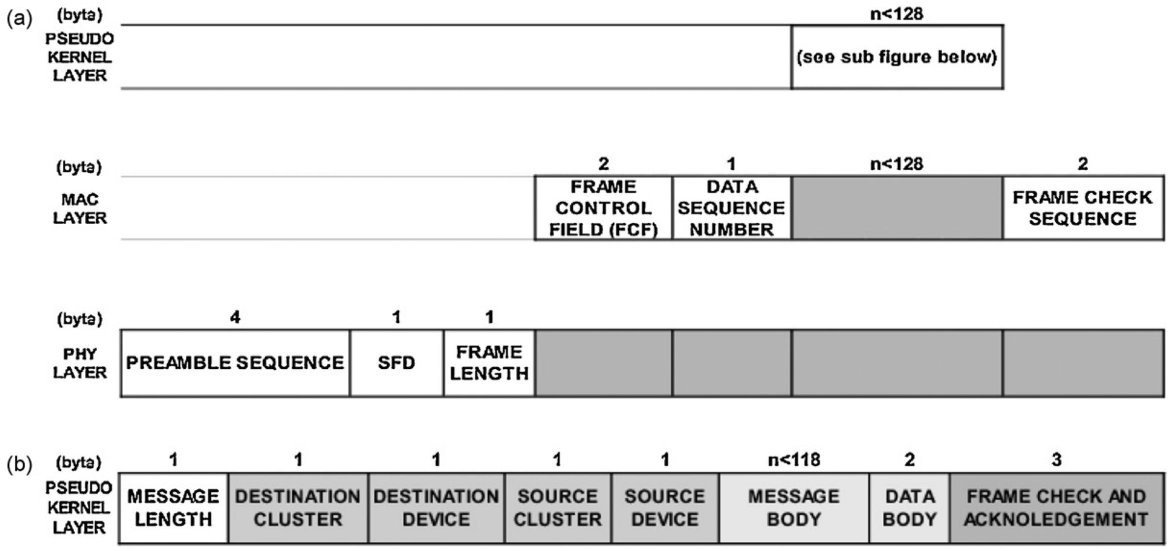

Fig. 5. (a) The CC2430 radio hardware protocol. (b) Pseudokernel wireless communication protocol.

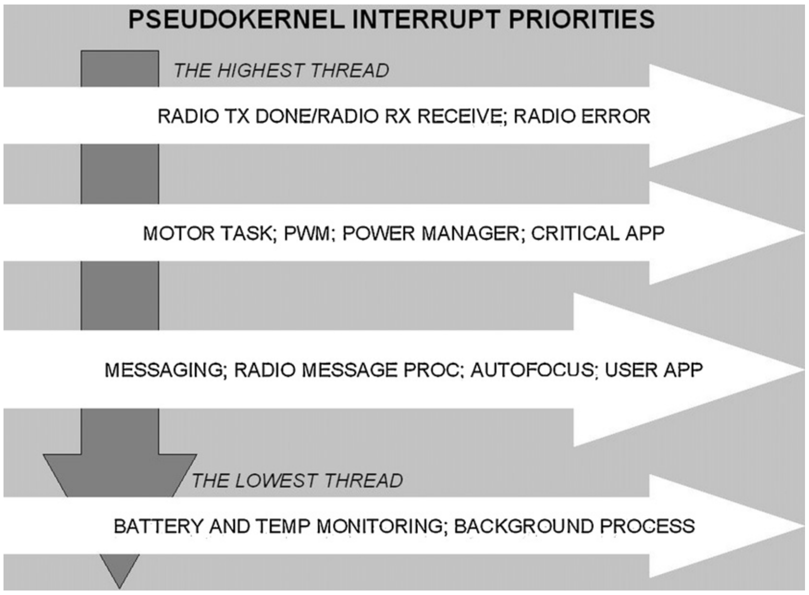
PSEUDOKERNEL INTERRUPT PRIORITIES
THE HIGHEST THREAD
RADIO TX DONE/RADIO RX RECEIVE; RADIO ERROR
MOTOR TASK; PWM; POWER MANAGER; CRITICAL APP
MESSAGING; RADIO MESSAGE PROC; AUTOFOCUS; USER APP
THE LOWEST THREAD
BATTERY AND TEMP MONITORING; BACKGROUND PROCESS

Fig. 6. Example of assigning tasks and applications to the proper thread.

as messaging service, radio message processing and user applications. The next higher level, the second level thread, contains motor control task and encoder sensing. Periodic task preemption happens in this thread conforming the time tick entry given in the configuration file. At the highest level thread, reside radio critical tasks, such as radio transmit/receive interrupt handling and radio error handling. Programmer has to pay attention during coding especially not to mismanage application to the priority it is not supposed to be. Preempting a running application is a priceless privilege given by pseudokernel, therefore programmer must take advantage of this feature by selecting the proper priority level for each user application. Otherwise, system overhead can not be tolerated anymore. Fig. 6 shows the threads from top to bottom as the highest to the lowest priority order.

6. Wireless networking support

Wireless networking support involves several capsular devices that work together in a group. In particular, this feature is necessary in order to distinguish between one capsular device to another by assigning wireless network cluster and wireless network address to each of them. This may involve different capsules used by different patients inside the same hospital, preventing cross-talk and interferences among devices. Further, a single patient may need to swallow more than one capsule for diagnostic or therapeutic purposes. Therefore, addressing and clustering are desirable features to control each single capsule within a group. For example, a cluster can be assigned to each single patient, so that all the capsules used by him belong to the same cluster.

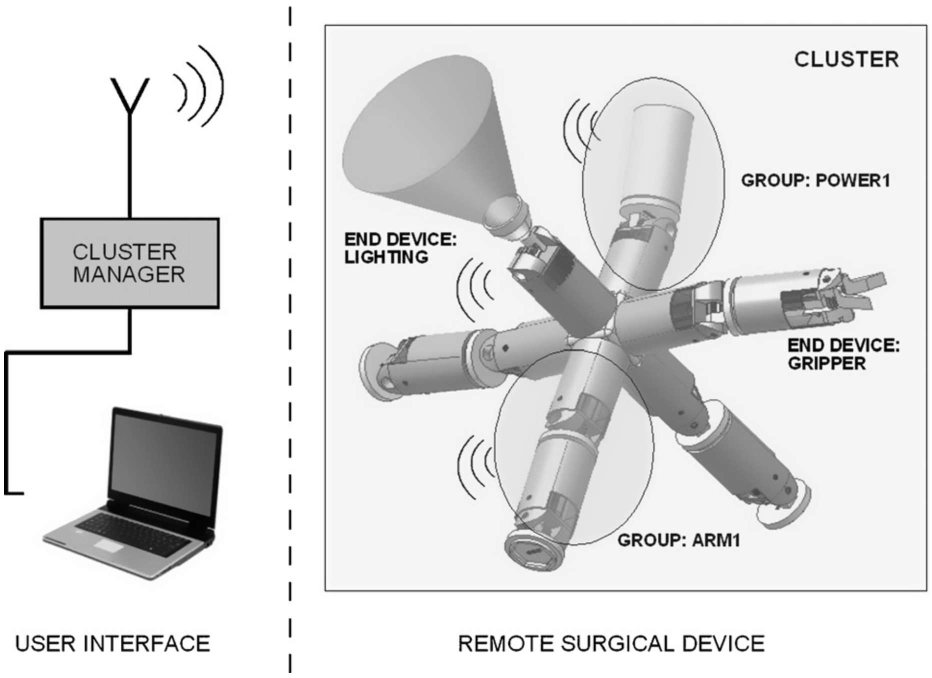

Fig. 7. Wireless networking application on reconfigurable wireless robotic platform.

Robotic module inside GI tract can be assigned as cluster manager or cluster device. As many as 253 clusters can be defined and 253 devices may stay under one cluster, due to the 8-bit addressing. Cluster manager must monitor and distributes messages continuously. Since this action withdraws more energy, it is preferably to have cluster manager wired directly to a PC, acting as communication interface to the surgeon.

Wireless network clustering can be useful when having multiple capsular devices assembling and reconfiguring themselves in the human digestive tract [22]. In such a futuristic scenario, the capsular devices are swallowed one by one and, once they arrive in a large cavity that needs intervention, for example the stomach, the capsular devices can be assembled to form a complex robotic structure. All segments have approximately the same size as one swallowable capsular device, but they have different functionalities. Each segment can be a gripper, lighting, biopsy tool, wireless camera, extra-power supply or just a simple bending module to have an extended arm. A prototype of this robotic platform has been reported in [23], implementing wireless networking support in the pseudokernel. Wireless networking application will not be described in detail here, since advanced application development is not within the scope of this paper. Fig. 7 illustrates the built-in clustering and addressing features on a developed prototype of reconfigurable wireless robotic platform.

7. Experimental results

Some experiments have been done to obtain optimal configuration parameters for the miniaturized wireless control platform with 16 kB of pseudokernel stack loaded inside the program memory. The experiments measured the optimal brushless DC motor speed, the response time of the pseudokernel and the wireless message bandwidth related to the pseudokernel time tick. Measurements related to the use of brushed DC motors were not reported in this paper, however the resources for 4 brushed DC motors and 2 brushless DC motors were included in the pseudokernel which was running during the test. The handling time of a brushed DC motor is significantly lower than the handling time of a brushless one. Driving on and off one brushed DC motor just takes 9 instruction cycles, which is calculated as 0.28 μs at 32 MHz clock of CC2430 microcontroller. Therefore, it is more relevant to look at the performance of the pseudokernel while running two brushless DC motors and some user applications, like sending radio command from console.

In order to test the pseudokernel response time during message request, message processing and acknowledgement, two software applications implementing wireless communication were added in the system. The test bench consists of one PC with hyperterminal, one miniaturized CC2430 board, one brushless DC motor SBL04-0829PG79, one evaluation board SmartRF04EB (Texas Instruments, USA) with CC2430 module and an oscilloscope to capture the program flow. The SmartRF04EB, acting as cluster manager, is connected to the computer via USB port and the miniaturized board, performing as cluster device, is connected to a brushless DC motor. The command is sent by PC through virtual serial port to cluster manager and then is forwarded to cluster device. Cluster device executes the command and replies with an acknowledgement.

A series of preliminary tests about worse case time required by brushless DC motor SBL04-0829PG79 showed very satisfying results. The state-driven tasks for driving two brushless DC motors give a maximum time interval of 7.6 μs to acquire BEMF and proceed commutation accordingly. Fig. 8 shows the current withdrawn by one brushless DC motor during BEMF commutation under different time ticks. When two brushless DC motors are running at the same time, the total amount of current is doubled. A region between 10 μs and 160 μs time tick is preferable due to less power required to drive the motor.

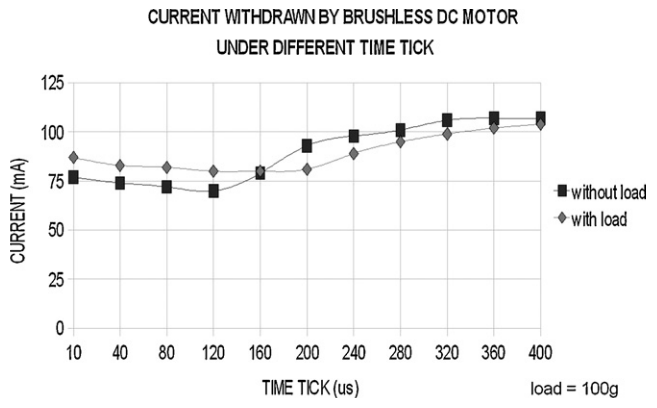

Fig. 8. Current withdrawn by brushless DC motor SBL04-0829PG79 under different time ticks.

Previous experience in testing and modeling legged capsules [24] indicates that for a 12-legged capsule robot as described in [2], approximately 0.66 N for each individual foot will be sufficient to propel the capsule along the colon. To be conservative, we tested our system with 100 g of load, representing a force of 1 N. Fig. 9 depicts the speed of the brushless DC motor under different time ticks. It is obvious that the highest speed can be obtained with small time tick interval, since BEMF is intensively read from the motor. This ensures that commutations happen at the right time and the brushless DC motor delivers as maximum torque as possible for actuation and locomotion. When Fig. 8 is associated to Fig. 9, the region between 120 μs and 160 μs time tick gives the optimal result for driving the motor. Therefore, in term of optimal energy consumption and speed, pseudokernel should be configured to operate at 140 μs time tick. It can be seen as well that the driver is able to commutate the brushless DC motor better and faster than the commercial board, within that region.

Implementing the optimal 140 μs of pseudokernel time tick, SmartRF04EB sent the longest message possible of 128 bytes from the radio buffer to test its response time and the wireless message bandwidth. Nevertheless, aside form it, the radio hardware itself sends another 11 bytes as header and footer. Within 21.5 ms response time from application request to application acknowledgement, the radio hardware sends actual 139 bytes of data. Therefore, a robust wireless communication at 52 kbps could be guaranteed by the pseudokernel approach. Fig. 10 shows detailed timing diagram of each process captured from SmartRF04EB and the miniaturized wireless control platform. The waveforms are acquired by debugging state logic transition through one available

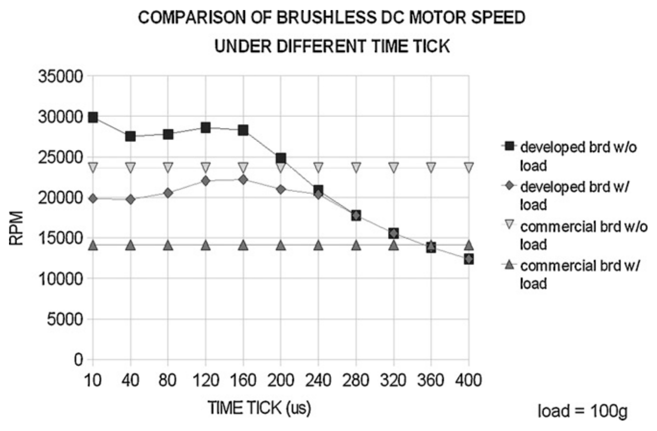

Fig. 9. Speed comparison of brushless DC motor SBL04-0829PG79 driven by the developed board and the commercial one.

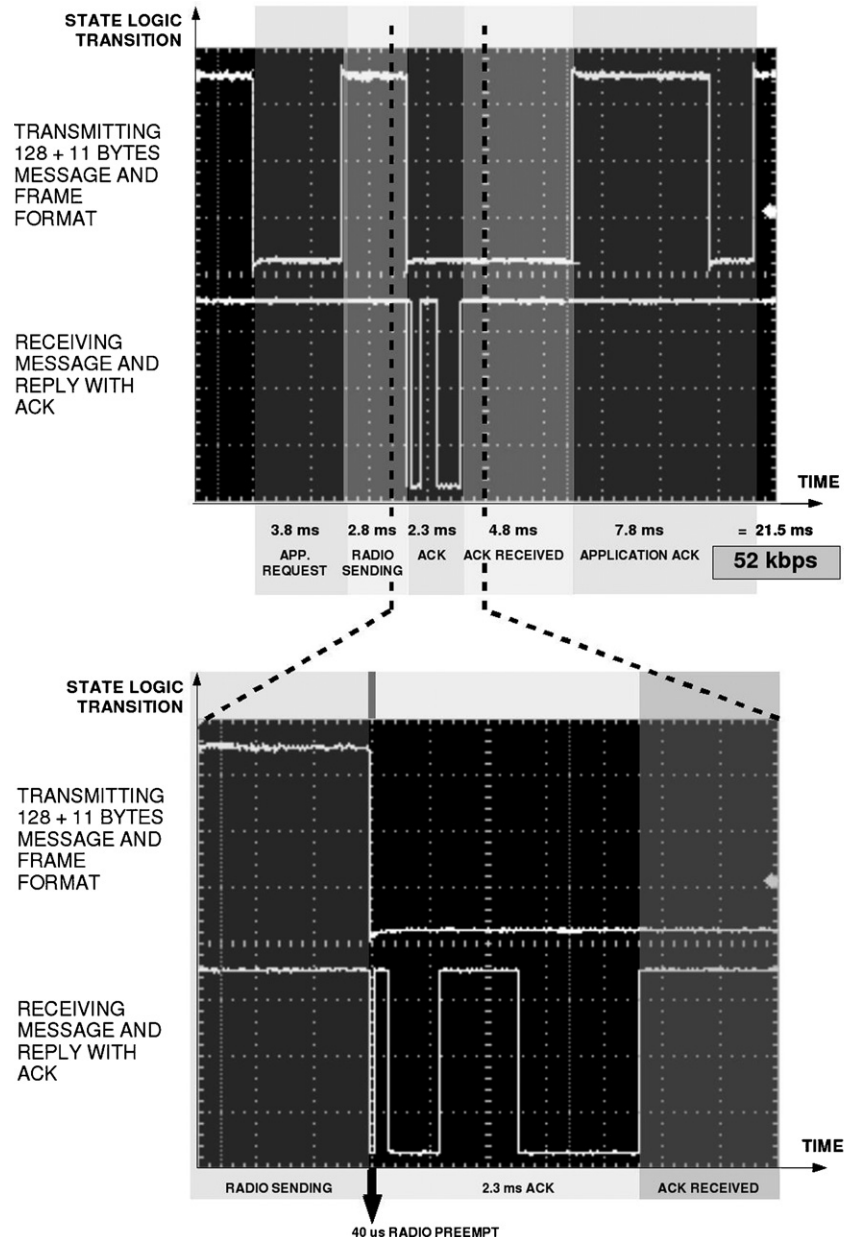

Fig. 10. Response time captured from SmartRF04EB (transmitting) and the miniaturized CC2430 platform (receiving) showing 52 kbps bandwidth and 40 μs radio preemption in the pseudokernel.

port of CC2430 microcontroller of SmartRF04EB evaluation kit. The radio preemption happens at about 40 μs.

8. Discussion

The correlation between the optimal speed and the withdrawn current of the brushless DC motor under different pseudokernel time ticks may introduce three different motor driving approaches. Region of time tick between 10 μs and 120 μs would give the highest speed and torque possible with less current, but the speed may vary depending on the load on the shaft of the motor. Region between 120 μs and 140 μs is the optimal driving method. Region between 140 μs and 240 μs is not recommended to do, since the withdrawn current with or without load. However, a particular application which uses a constant load on the shaft might consider to operate in this regime. The last region, more than 240 μs, has the advantage of having constant speed even if the load on the shaft varies. This constant speed can be achieved through varying the withdrawn current by the motor. It is worthy to note that the brushless DC motor is driven near to stepping mode in this region. The pseudokernel becomes less responsive and misses some important hard real time deadlines.

Thanks to the flexibility of the proposed solution, achieved both from a hardware and a software standpoint, the wireless miniaturized control board described in this paper was successfully applied to the different enhanced endoscopic capsules presented in Section 1. In particular, we were able to operate a 12-legged capsule for active locomotion in the colon [2](33 mm long and 11 mm in diameter, embedding a pair of SBL04-0829PG79 motor), a propeller based capsule for active orientation in a liquid environment, such as a gastric cavity filled with a transparent liquid [3](30 mm long and 11 mm in diameter, embedding 4 DC motors MK04S-24 by Didel, Switzerland), a video capsule controlled with external magnetic fields featuring a wireless camera and a triaxial inertial sensor [4](35 mm long and 11 mm in diameter), a therapeutic capsule, which is able to release a surgical clip on a desired spot of the gastrointestinal tract [5](33.5 mm long and 12.8 mm in diameter, embedding a single SBL04-0829PG79 motor), a video capsule with auto-focus capabilities [6](35 mm long and 14 mm in diameter, embedding a liquid lens system to achieve variable focus).

All these devices include in their dimensions the space for a standard wireless video module (almost 0.6 cm^3 as the one in the PillCam SB [25]), being still too large to swallow. The largest device released by Given Imaging is 11 mm in diameter and 31 mm long (PillCam Colon), roughly the size of a large vitamin pill. By further miniaturizing the camera module and by pursuing a custom approach for the electronics [26,27], such a swallowable size may be achieved in a near future.

9. Conclusions and future work

A preemptive priority pseudokernel, consisting of state-driven code, coroutine, and pooled-loop algorithm, has been implemented to perform hard, firm and soft real-time applications on a miniaturized platform for capsular robotic endoscopy and novel endoluminal devices. In particular, a state-driven method for two brushless DC motors under coroutine scheme was implemented on two purposely developed miniaturized boards and tested successfully. Furthermore the two prototypes were implemented in a capsule compatible layout, having a circular shape of 10 mm in diameter and 2.5 mm in thickness. These were applied to several different capsular devices with enhanced diagnostic and therapeutic features. The main advantage of the proposed wireless miniaturized control board is that the dimension does not change over any variant of endoscopic capsule robot. A unique hardware has been designed with well defined footprints so that changing the software driver may change completely the behavior of the system to control many types of actuators such as brushed DC motor, brushless DC motor, or even SMA actuator. The actuator selection and the number of actuators used for one variant of the capsule may be different from one to another, as detailed above.

Preemptive priority pseudokernel is proven working for 8051 compatible microcontroller to handle hard, firm, and soft real time applications. Further investigation is needed regarding the constant speed driving when the time tick is configured above 240 μs. Intensive study on the dynamic change of brushless DC motors current, while compensating a varying load in order to gain a constant speed, will require further investigation.

Subsequently, embedding these systems in active robotic capsules for wireless endoscopy will be the next step. Novel technologies or new integrated circuits, such as CC25xx family (Texas Instruments, USA), will come along to miniaturize even more the controller board.

Acknowledgments

The work described in this paper has been supported by European Commission in the framework of three FP6 European Projects, namely VECTOR Project(EU/IST-2006-033970), ARES Project(NEST-2003-1-ADVENTURE/15653) and P. Cezanne (EU/IST-2006-031867). This work is also supported by the Intelligent Microsystem Center, KIST, South Korea. Authors would like to thank Dr. Kanako Harada for the opportunity to test the developed networking application on the reconfigurable wireless robotic platform.

References

[1] A. Moglia, A. Menciassi, P. Oliver, Dario, Wireless capsule endoscopy: from diagnostic devices to multipurpose robotic systems, Biomedical Microdevices 9 (2) (2007) 235–243.
[2] M. Quirini, R. Webster, A. Menciassi, P. Dario, Design of a pill-sized 12-legged endoscopic capsule robot, in: Proc. IEEE International Conference on Robotics and Automation, 2007, pp. 1856–1862.
[3] R. Carta, B. Lenaerts, J. Thone, G. Tortora, P. Valdastri, A. Menciassi, R. Puers, P. Dario, Wireless power supply as enabling technology towards active locomotion in capsular endoscopy, in: Proc. Eurosensors XXII, Dresden, 7–10 Sept, 2008, p. 482.
[4] G. Ciuti, P. Valdastri, A. Menciassi, P. Dario, Robotic magnetic steering and locomotion of microsystems for diagnostic and surgical endoluminal procedures, Robotica, 2009.
[5] P. Valdastri, C. Quaglia, E. Susilo, A. Menciassi, P. Dario, C. Ho, G. Anhoeck, M. Schurr, Wireless therapeutic endoscopic capsule: in-vivo experiment, Endoscopy 40 (2008) 979–982.
[6] C. Cavallotti, F. Focacci, M. Piccigallo, E. Susilo, P. Valdastri, A. Menciassi, P. Dario, An integrated vision system with autofocus for wireless capsular endoscopy, in: Proc. Eurosensors XXII, Dresden, 7–10 September, 2008, p. 481.
[7] P. Valdastri, A. Menciassi, P. Dario, Transmission power requirements for novel zigbee implants in the gastrointestinal tract, IEEE Transactions on Biomedical Engineering 55 (6) (2008) 1705–1710.
[8] N.R. Subramanian, N. Kirkeby, Anaren 0404 balun optimized for Texas Instruments CC2430 Transceiver, August, 2007.
[9] A3901 Dual Full Bridge Low Voltage Motor Driver, 3901-DS, Rev. 3, 2007.
[10] M. Quirini, S. Scapellato, A. Menciassi, P. Dario, F. Rieber, C.N. Ho, S. Schostek, M.O. Schurr, Feasibility proof of a legged locomotion capsule for the GI tract, Gastrointestinal Endoscopy 67 (7) (2008) 1153–1158.
[11] W. Cedeno, P.A. Laplante, An overview of real time operating systems, Journal of the Association for Laboratory Automation 12 (2007) 40–45.
[12] J. Turley, Operating systems on the rise, Embedded Systems Design, 2006 http://www.embedded.com/showarticle.jhtml?articleid=187203732.
[13] Keil Software—Getting Started and Creating Applications, June, 2000).
[14] Keil real-time os/kernels. http://www.keil.com/rtos/.
[15] The freertos.org project. http://www.freertos.org/.
[16] P.P. Debono, Paulos an 8032–8051 rtos. http://staff.um.edu.mt/pdeb1/paulosrtos.htm.
[17] P.A. Laplante, Real-Time Systems Design and Analysis, 3rd edition, John Wiley & Son, Inc., 2004.
[18] B. Kernighan, D. Ritchie, The C Programming Language, 2nd edition, Prentice Hall, Inc., 1988.
[19] Texas Instrument CC2430 Datasheet, December, 2006.
[20] J. Shao, A novel microcontroller-based sensorless brushless dc (bldc) motor drive for automotive fuel pumps, IEEE Transactions on Industry Applications 39 (2003) 1734–1740.
[21] J. Shao, An improved microcontroller-based sensorless brushless dc (bldc) motor drive for automotive applications, IEEE Transactions on Industry Applications 42 (2006) 1216–1221.
[22] The ARES (Assembling Reconfigurable Endoluminal Surgical system) Project website. http://www.ares-nest.org.
[23] K. Harada, E. Susilo, N. Ng Pak, A. Menciassi, P. Dario, Design of a bending module for assembling recofigurable endoluminal surgical system, in: Proc. the 6th Intl. Conf. Intl. Soc. Gerontechnology (ISG'08), Pisa, Italy, 2008, paper ID-186.
[24] C. Stefanini, A. Menciassi, P. Dario, Modeling and experiments on a legged microrobot locomoting in a tubular, compliant and slippery environment, International Journal of Robotics Research 25 (5–6) (2006) 551–560.
[25] Given Imaging Ltd. http://www.givenimaging.com.
[26] O. Alonso, L. Freixas, J. Samitier, A. Dieguez, E. Susilo, Design of a brushless micro motor driver for a locomotive endoscopic capsule, in: 15th IEEE International Conference on Electronics, Circuits and Systems, ICECS 2008, Malta, 2008, pp. 986–989.
[27] A. Arbat, J. Samitier, A. Dieguez, P. Valdastri, Architecture of the integrated electronics for a wireless endoscopic capsule with locomotive and sensing and actuating capabilities, in: 15th IEEE International Conference on Electronics, Circuits and Systems, ICECS 2008, Malta, 2008, pp. 510–513.

Biographies

Ekawahyu Susilo received his Bachelor Degree in Electrical Engineering from Universitas Surabaya, Indonesia in July 2000. Six months later he joined Universitas Surabaya and is working lecturer since then. During 4 years time, he had been Head of Automation Laboratory in the same university and also Head of Microcontroller Division in School of Technology (SofT) in Surabaya, Indonesia. In May 2005 he started PhD Study in Scuola Superiore Sant'Anna, Pisa, Italy in the field of Bioengineering. Now he is in the final year of the PhD Study. His major is in real time embedded system, embedded system design and robotic applications. He has received many awards in scientific writings and robotic contests.

Pietro Valdastri received his Laurea Degree in Electronic Engineering (with Honors) from the University of Pisa in February 2002. In the same year he joined the CRIM Lab of the Scuola Superiore Sant'Anna in Pisa as PhD student. In 2006 he obtained his PhD in Bioengineering from Scuola Superiore Sant'Anna by discussing a thesis titled "Multi-Axial Force Sensing in Minimally Invasive Robotic Surgery". He is now assistant professor at CRIM Lab, with main research interests in the field of implantable robotic systems and active capsular endoscopy. He is working on several European projects for the development of minimally invasive and wireless biomedical devices.

Arianna Menciassi received her Laurea Degree in Physics (with Honours) from the University of Pisa in 1995. In the same year, she joined the CRIM Lab of the Scuola Superiore Sant'Anna in Pisa as a PhD student in bioengineering with a research program on the micromanipulation of mechanical and biological micro objects. In 1999, she received her PhD degree by discussing a thesis titled "Microfabricated Grippers for Micromanipulation of Biological and Mechanical Objects". Currently she is a professor of biomedical robotics at the Scuola Superiore Sant'Anna, Pisa. Her main research interests are in the fields of biomedical micro- and nano-robotics, microfabrication technologies, micromechatronics and microsystem technologies. She is working on several European projects and international projects for the development of micro- and nano-robotic systems for medical applications.

Paolo Dario received his Laurea Degree in Mechanical Engineering from the University of Pisa in 1977. Currently, he is a professor of biomedical robotics at the Scuola Superiore Sant'Anna, Pisa. He also established and teaches the course on Mechatronics at the School of Engineering, University of Pisa. He has been a visiting professor at the Ecole Polytechnique Federale de Lausanne (EPFL), Lausanne, Switzerland, and at Waseda University, Tokyo, Japan. He is the director of the CRIM Lab of Scuola Superiore Sant'Anna, where he supervises a team of about 70 researchers and PhD students. His main research interests are in the fields of medical robotics, mechatronics and microengineering, and specifically in sensors and actuators for the above applications. He is the coordinator of many national and European projects, the editor of two books on the subject of robotics and the author of more than 200 journal papers. He is a member of the Board of the International Foundation of Robotics Research. He is an associate editor of the IEEE Transactions on Robotics and Automation, a member of the Steering Committee of the Journal of Microelectromechanical Systems and a guest editor of the Special Issue on Medical Robotics of the IEEE Transactions on Robotics and Automation. He serves as president of the IEEE Robotics and Automation Society and as the co-chairman of the Technical Committee on Medical Robotics of the same society.