2015 IEEE International Conference on Robotics and Automation (ICRA)
Washington State Convention Center
Seattle, Washington, May 26-30, 2015

Toward Rapid Prototyping of Miniature Capsule Robots

Addisu Taddese, Marco Beccani, Ekawahyu Susilo, Péter Völgyesi, Ákos Lédeczi, Pietro Valdastri

Abstract— **Minimally invasive robotic surgery techniques are becoming popular thanks to their enhanced patient benefits, including shorter recovery time, better cosmetic results and reduced discomforts. Less invasive procedures would be achieved with the use of Medical Capsule Robots (MCRs). These devices are characterized by low power requirements and small dimensions as well as uncompromising safety. MCRs operate wirelessly in abdominal Minimally Invasive Surgery (MIS) and Natural Orifice Transluminal Endoscopic Surgery (NOTES) or in the Gastrointestinal (GI) tract. The design process of MCRs, however, is expensive and time consuming. A platform for rapid prototyping MCRs is needed so that MCR researchers can reduce development costs and spend more time in studying innovative MCR applications. In this work, we introduce an open source modular platform geared toward rapid prototyping MCRs. To speed up the prototyping process, the MCR is programmed using TinyOS instead of bare-bone C. We present the hardware architecture of the platform, and the motivation for using TinyOS. To show the viability of TinyOS, we present results from an experiment involving sensing, actuation and wireless communication. This work lays the foundation for our future goal of building an integrated design environment for the design, analysis and simulation of MCRs.**

I. Introduction

The use of robotic systems in surgical procedures has increased steadily with the design of customized medical robots for specific procedures such as abdominal surgery, urology, cardiac surgery and eye surgery [1]. In the last decade, the tendency towards solutions for consumer electronic devices brought the improvement of miniature circuits technology, wireless telemetry, battery technology and more powerful microcontrollers (MCUs). This resulted in the proliferation of Medical Capsule Robots (MCRs) in medicine [2]. An MCR is a biocompatible device designed to operate in the human body, a constrained environment, and has to fulfill three main requirements: safety, low power operation and small size. Common tasks of an MCR include sensing to monitor physiological parameters, communication to transfer data to an external agent and actuation for locomotion, therapy delivery and biopsy sampling [2]. MCRs aim to be less invasive and to improve surgeons' ability in diagnosing, detecting and curing diseases. They are designed to be inserted into the human body through natural orifices (e.g., mouth) to operate in the Gastrointestinal (GI) tract. Other possible areas of application for MCRs are Minimally Invasive Surgery (MIS) [3]–[6] and Natural Orifice Transluminal Endoscopic Surgery (NOTES) [7].

Since there are no widely available hardware development platforms for MCRs, research groups are usually forced to create their own MCRs from the ground up. MCRs have stringent power and space constraints, thus, researchers have to spend a lot of time and money on the difficult task of optimizing their hardware design before they can even begin work on their research. This amounts to a duplication of efforts by different research groups, which could be better applied to the innovation of MCR applications. To address this issue, the Storm Lab Modular Architecture for Capsules (SMAC), a first step towards the rapid prototyping of MCRs, was proposed by the authors in [8]. The SMAC is an open source platform that provides a set of hardware modules and associated software libraries for building MCRs. The platform aimed to consolidate the efforts and best practices of MCR researchers and to lower the barrier of entry for research groups that do not have the resources to create their own hardware. The SMAC was inspired by open source platforms (e.g., Arduino [9]) outside the medical field that enable users to rapidly prototype devices and test new algorithms.

In this work, we improve upon the SMAC by addressing some of its limitations. The main limitation of the SMAC is that it requires the user to solder thin wires for connectivity. While seeming trivial, this limitation inhibits users from easily experimenting with different composition of modules. It also prevents standardized MCU pin assignments making software portability extremely difficult. Furthermore, it increases the occurrence of wiring related failures, which are difficult to diagnose and fix. The lack of alternative means of wireless communication is another limitation that is addressed in this work. Finally, the software libraries provided by the SMAC platform do not provide a high enough level of abstraction and therefore do not accommodate researchers that are not experienced in writing in C/C++.

In this paper, we reconsider the constituents of a good rapid prototyping platform for MCRs. For a platform to enable rapid prototyping in the specific domain of MCR research, it needs to be modular and easily customizable while being energy efficient and small in size. A hardware only platform, however, does not achieve the efficiency we desire with which researchers can prototype their MCR systems. A modular operating system (OS) with a rich set of libraries and hardware drivers is required. This OS has to be energy efficient (effectively utilizing the low power states of MCU and peripheral devices) while allowing flexible composition of hardware and software modules. TinyOS,

A. Taddese, P. Völgyesi and A. Lédeczi, are with the Institute for Software Integrated Systems, Department of Electrical Engineering and Computer Science, Vanderbilt University, Nashville, TN 37212

M. Beccani and P. Valdastri are with the STORM Lab, Department of Mechanical Engineering, Vanderbilt University, Nashville, TN 37235. Email: p.valdastri@vanderbilt.edu

978-1-4799-6922-7/15/$31.00 ©2015 IEEE

a component based OS used in Wireless Sensor Networks (WSNs) research, meets all the criteria described.

TinyOS was designed to work with highly resource constrained embedded devices used in WSNs as sensor nodes. The function of these nodes is to collect sensory data and relay them to a base station. MCRs carry out the same functionality, but within a different environment, the human body. While they have different battery life requirements, with sensor nodes being expected to last weeks or months, they both strive to conserve as much power as possible. Finally, they both have limited CPU and memory resources due to their size and power constraints. The significant overlap in functionality between MCRs and sensor nodes suggests that TinyOS would be a suitable OS for MCRs.

Taking this one step further, a rapid prototyping platform for MCRs needs a comprehensive and focused design environment. This design environment would be focused in that it would be specific to MCRs and would provide additional tool support and guidance to MCR researchers in their design process. To create a focused design environment that can help users analyze and optimize their designs, a model-based approach is needed. With this in mind, a component model is built for each hardware module available in the platform. The model would contain various attributes of the module that describe its mechanical, electrical and software characteristics. It would also expose parameters that can be customized by the user. Users, then, would build their applications by composing different component models in a visual workspace. Once the application is composed, the attributes of the constituent component models would be used for answering design space questions such as power consumption, mechanical compatibility, electromagnetic interference, etc. Finally, the design environment will generate optimized code that can directly be used on the hardware platform.

From a rapid prototyping standpoint, what is required is an assortment of reliable hardware modules that can easily be pieced together to form a functioning unit. The component based programming paradigm of TinyOS complements this notion such that a one-to-one map can be created between hardware modules and TinyOS components. These components, if well designed, abstract away the details of the hardware module and allow users to think in terms of composition and functionality. Components can also encapsulate commonly used MCR behavior and logic to speed up the creation of prototypes. A collection of such component models will be made available in the design environment, which will form a repository that is readily accessible to users. Component models will be added to this repository as new hardware modules are created by various researchers. The repository will serve as a channel of collaboration fostering design reuse.

The paper is organized as follows: Section II gives an overview of the hardware platform. Section III makes a case for using TinyOS as an OS for MCRs while also discussing how it is the appropriate choice for a model-based design environment. Sections IV and V discuss experimental results and present conclusions and future works.

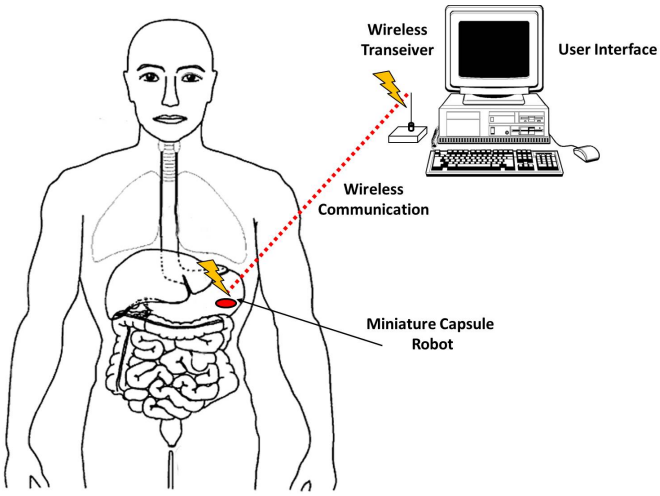

Fig. 1: A typical MCR system that is composed of the intra-body device (the MCR), an external wireless transceiver for data communication and the user interface.

II. MCR Platform Overview

As Fig. 1 indicates, a typical architecture for an MCR based system consists of the MCR (i.e., the intra-body device), an external wireless transceiver and a PC, where the user interface is implemented.

A. MCR Hardware Overview

As shown in Fig. 2, the hardware for the MCR platform can be functionally classified into different components: an MCU, modules for wireless communication, sensing and actuation, and a power source. While the MCU handles the basic arithmetic, logic, and I/O operations of the MCR, the wireless transceiver and the sensing and actuation solutions have to be chosen according to the specific task at hand. With so many variants of commercially available sensing and actuation solutions, the MCR hardware aims to provide to the developer a small package size and low power MCU with enough peripherals and General Purpose I/O (GPIO) to interface with the different sensing, actuation and communicating modules. The connectivity between the modules is achieved by flexible circuitry that forms the skeleton on which modules are mounted, which is then folded to form the body of the MCR. This connectivity mechanism reduces the time needed for soldering and assembling the MCR prototype. Modules developed thus far are: an MCU, a 433 MHz transceiver, two sensing modules (6DOFAG and 8CHADC), an actuation module (BDCC) and a power module. The modules are listed in Table I with their dimensions and power consumption.

1) MCU: The MCU module, as stated above, has to handle a variety of tasks depending on the MCR application and therefore its choice is important in the development of an MCR. In the case of the swallowable MCR, such as the commercially available Given Imaging Pillcam® [10],

Module Name	Functionality	Integrated Circuit (IC)	Diameter (mm)	Thickness (mm)	Max Consumption (mA)
MCU	Microcontroller	MSP430F5528	10.5	3.84	2.32
433 MHz Transceiver	Wireless Communication	CC1101	10.5	3.94	29.2
6DOFAG	Sensing	LIS330DLC	10	4.04	0.01
8CHADC	Sensing	AD7689	10	3.94	3.78
BDCC	Actuation	(2 $\times$) A3901	10	3.69	800 mA (Max)
Power	Power Management	NCP606, LTC4065, LTC2942	10	3.84	500 (Max)

TABLE I: Summary of currently available modules

a maximum diameter of 11 mm is needed for the shell. On the other hand when MCR are introduced into the abdomen through surgical trocars they have to fit its diameter. (e.g., the 5-12 Vesaport Plus, Covidien, USA has a diameter of 13 mm) Therefore, the MCU module has to be small enough in size to be embedded into the shell, as well as has to operate in low power modes to increase the lifetime of the device, and have hardware peripherals to interface with sensors, actuators and wireless transceivers [2].

For the rapid prototyping hardware platform, we have chosen the fifth generation MSP430 (MSP430F5528, Texas Instruments, USA). This MCU comes in a 64 pin, 3.76 mm by 3.76 mm ultra miniature Ball Grid Array (BGA) package with many peripherals and low power options. The developed miniaturized board has a diameter of 10.6 mm and a thickness of 1.6 mm. A dedicated Serial Peripheral Interface (SPI) communicates with the wireless modules through a separate connector soldered on top of the MCU module. This connector enables developers to interface multiple radios with the MCU module according to their application. Additional SPI and Inter-Integrated Circuit (I^2C) buses on the flexible circuit are used to interface with sensing and actuation modules.

2) Wireless Communication Modules: An MCR device needs wireless communication in order to transmit data from inside the human body. This communication must often be performed with high data rates for real-time applications, with transmission power bound to medically safe limits. The sub 1 GHz carriers have been considered the most suitable for medical devices because these frequencies have lower energy absorption in human tissues [11]. Given Imaging is currently using the 433 MHz carrier for the Pillcam [11], [12]. Other solutions have also been developed using this range of frequencies [13]. Currently, our platform supports a sub 1 Ghz wireless module (CC1101, Texas Instruments, USA) with a quarter wave wire antenna of 17.3 cm length ($\lambda = c/f$) folded to fit into the MCR shell. On the other hand, as shown in [11], [14], the 2.4 GHz carrier, which is used in ZigBee and Bluetooth applications, can be adopted for intra-body data communication as well. A base station is provided for MCR-Personal Computer (PC) wireless communication. It is connected to a PC via Universial Serial Bus (USB) and contains the same MCU and radio module as the MCR; therefore, it can be customized as needed. Furthermore, since most smart phones have embedded Bluetooth modules, communicating with an MCR from the phone would be feasible in the future.

3) Sensing and Actuation Modules: Sensing modules for the proposed platform consist of commercially available small package digital or analog sensors such as accelerometers, gyroscopes, pressure sensors and hall effect sensors. In general, any sensor with a digital interface (SPI or I^2C), operating at a maximum voltage of 3.3 V, can be physically connected to the MCU and be read by the software. Similarly, analog sensors can be connected to ADC channels in the MCU or to external ADC devices with digital interfaces to the MCU [2]. Regarding actuators, the type used by an MCR is determined by the target environment for which it is designed. MCRs in fluid environments may employ propeller based locomotion [15] while MCRs in dry environments may mimic crawling [16]. This implies that it is beneficial to provide multiple digital interfaces (e.g., Pulse Width Modulation (PWM), GPIO) to the user. At the time of this writing, sensing and actuation modules that have been developed consist of a 6 degree of freedom accelerometer and gyroscope (6DOFAG), an 8 channel 16 bit analog to digital converter (8CHADC) and a brushed DC motor controller (BDCC).

As part of this work, new drivers and components for some devices, including the CC1101 and MSP430F5528, have been added to TinyOS in order to support our platform. A small set of additional components for sensors and actuators have also been implemented to conduct our feasibility experiment.

The source code and the hardware design files are all open source and can be found at `https://github.com/pillforge`. This module library will continually be expanded as new modules are developed by the authors. Users also have the opportunity to contribute modules thereby helping other MCR researchers and creating a community around the platform.

III. Motivation for using TinyOS

1) Efficient use of Resources: As mentioned earlier, TinyOS was designed to work with highly resource constrained embedded devices. Therefore, the efficient use of resources is central to its design. It is lightweight, only consuming 400 bytes of memory for the core OS. Since the composition of components is static and known at compile time, optimizations can be made that significantly reduce code size as well as CPU utilization. The task based concurrency model allows for putting the CPU at a low power state when the task queue is empty saving battery power [17].

2) Composability: TinyOS uses a component based programming model, which allows hardware modules to be associated with software components. A component, in

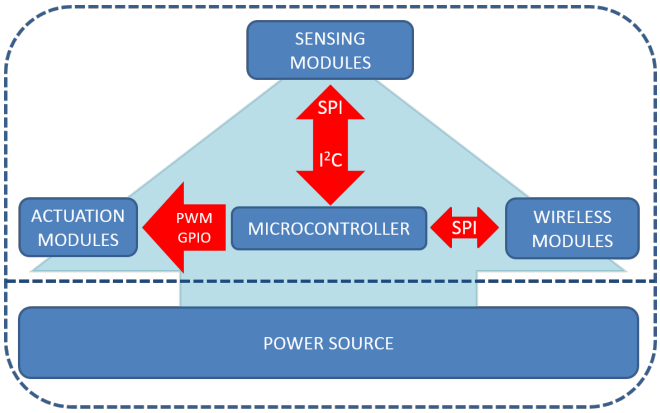

Fig. 2: Schematic diagram of the hardware modules in an MCR. The MCU handles data communication with different sensing and wireless modules with SPI/I^{2}C interfaces. Actuation is done using GPIO and Pulse Width Modulation (PWM) signals according to the driver interface for the actuator.

TinyOS, is a software abstraction that encapsulates a service or accomplishes a specific task. Components are used as building blocks for not only the application level software but also the operating system. They expose interfaces, which are a set of predefined functions inside the component that are made available externally. Interfaces are "wired" together to attach function calls made inside one component to the actual function definition in another. This concept of interfaces is the key feature of TinyOS that enables composition of component models inside the design environment.

3) Design Time Analysis Tools: TinyOS has static code analysis tools as well as runtime checks that help in designing bug free systems. Race conditions, common bugs in multitasking applications, can be detected at compile time by the TinyOS compiler NesC. This is made possible by TinyOS's event based execution model where tasks run to completion and are atomic to each other. These built-in analysis tools can be integrated with the design environment so that illegal or suboptimal design choices can be caught and presented to the user. Additionally, Safe TinyOS, an extension of TinyOS, can be employed to add type and out-of-bounds checks with modest increase in resource usage [18].

4) Communication and Data Collection tools: An important task of an MCR is collecting sensory data and forwarding them to a base station that is connected to a PC. The communication protocol among the MCR, the base station, and the PC determines the reliability and flexibility of the overall system. TinyOS offers well tested solutions for this process. Specifically, the Active Message [19] mechanism can be used for wireless as well as Universal Asynchronous Receiver/Transmitter (UART) communication. In both cases, framing and error checking of packets are provided. Additionally, the *mig* utility, part of the TinyOS toolchain, can be used to generate code for the PC in higher level languages (e.g., Java, Python). The generated code can serialize and deserialize Active Messages so that packets are presented as native objects in the respective programming languages. The design environment will expand on these tools to give users a powerful Software Development Kit for developing graphical user interfaces (GUIs) and data analysis tools.

IV. Results

The viability of an MCR is strongly correlated with its power consumption. Energy sources for MCR are usually limited to less than 100 mAh due to battery size constraints. Therefore, an MCR has to conserve its energy in order to last long enough to perform its task. Typically, MCR applications consist of a repeating pattern of sensing, wireless communication and actuation. Thus, we designed our experiment to exercise these components separately while taking current measurements. It is worth noting that, while actuators consume the most power, they are difficult to use for a comparative analysis because they are highly application specific. We have included them in our experiment in order to present a fully working MCR. The main focus of the experiment was to measure the average power consumption of the platform configured as a typical MCR prototype. Of particular focus was the wireless communication module because it is the one component that consumes the most power (other than the actuators) and is used frequently in all MCR applications. Additionally, we wanted to measure the responsiveness of TinyOS and the amount of overhead it placed on the system.

Our bench-top experiment consisted of the MCR shown in Fig. 3. It embedded the main MCU module, the 433 MHz Transceiver module, the BDCC actuation module, the 6DOFAG sensing module and the power module. Two brushed Direct Current (DC) motors (MK04 S-24, Didel, Switzerland) were then connected to the BDCC output lines. As regards to power, a 30 mAh, 3.7 V rechargeable LiPo battery (Shenzhen Hondark Electronics Co., Ltd., China, 12 mm $\times$ 10 mm $\times$ 2 mm in size) was used in the power module which supplied regulated voltage to the rest of the MCR circuitry. The modules were then mounted on the flexible circuitry, and the voltage drop across a 10-Ω resistor placed in series to the positive supply terminal of the battery was sampled with a digital oscilloscope (Tektronix, TDS12014C, USA).

The setup also included a base station that was collecting sensor data from the MCR and sending actuation commands. The MCR was running a TinyOS application which had a periodic pattern of: 1) acquiring data from the sensing module, 2) sending a radio message with the acquired data, 3) waiting for an actuation command from the base station, 4) driving the motors and finally, 5) going back to idle state until the next data acquisition. The sensor sampling rate, the duty cycle of the PWM signal for the motor, and the motor-on time were controlled in the MCR application by commands sent from the base station.

The plot in Fig. 4 shows a single instance of the periodic message transmission with the MCR operating at a transmission power P_{TX} of 10 dBm and a sampling rate of

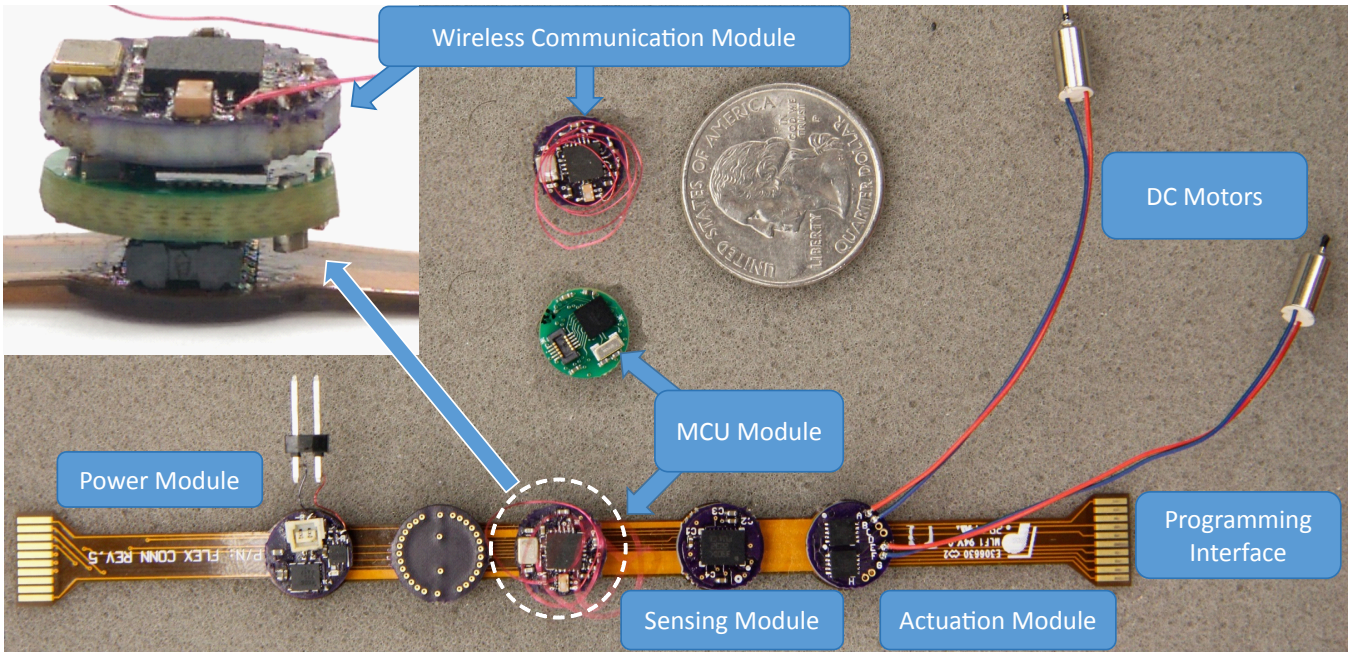

Fig. 3: The MCR mounted for the experimental setup composed of modules for the MCU, wireless communication, sensing, actuation and power.

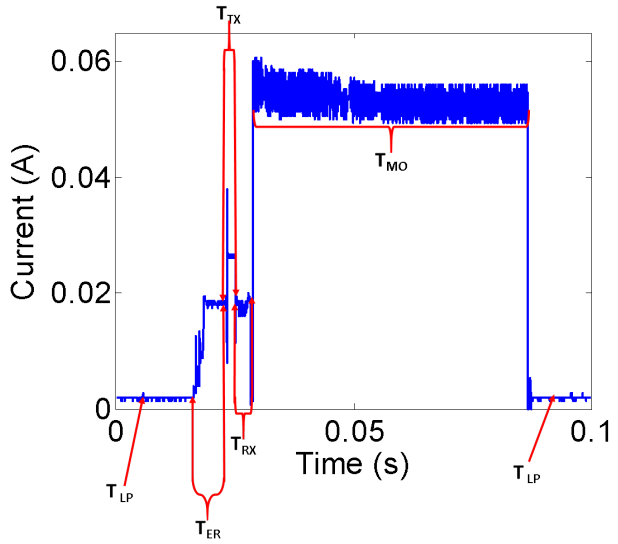

Fig. 4: Current consumption of the MCR prototype showing levels of current consumed for different system states.

0.5 Hz, with the motor being actuated for 60ms with a 50% duty cycle PWM signal. During the time interval T_{LP} the MCU was either in an idle state or acquiring data from the sensing module, while the wireless module was inactive in a low power mode. After the sensor data acquisition, the MCR turned on the wireless module in T_{ER} (2.68 ms) and transmitted the wireless message at a data rate of 125 Kbps during T_{TX} (1.8 ms). The transmitted message consisted of 18 bytes, 6 of which were the acquired sensor data while the rest were the IEEE 802.15.4 header and other overhead used by the TinyOS Media Access Control (MAC) and CC1101 physical layer. Once the radio transmission was completed, the MCU entered reception mode for T_{RX} of 2.9 ms during which it received a message from the base station that consisted of 6 bytes of payload including the actuation command, the sampling rate, the motor-on time, T_{MO}, and the PWM duty cycle. During T_{MO} the wireless module is is put in low power mode while motors are driven for the requested time interval. Subsequently the system went back to the idle state until the next data acquisition.

The average experimental values of current consumption for each specific state were I_{LP} =1.8 mA, I_{ER} = 14.5 mA, I_{TX} = 26.6 mA with a peak of 28.3 mA, I_{RX} =17 mA, and I_{MO} = 52.6 mA. Excluding the motor, the average current consumption for the sensing and communication tasks was 17.2 mA, which is 49% less than the typical current consumption exhibited by 2.4 GHz based MCR platforms as reported in [14]. The result shows that the hardware platform achieves the level of low power operation required by typical MCRs applications. It also shows that no significant overhead was accrued by using TinyOS. The responsiveness of TinyOS and the performance of the wireless communication module are on par with other MCR systems that implement real-time control using bare-bone C (no OS) or using a pseudo-kernel approach [20].

V. Conclusions

This paper presents the progresses toward the development of a comprehensive rapid prototyping platform for developing MCRs. The aim of the platform is to help the research community avoid the time and cost required to develop custom hardware and software for MCRs. The hardware

platform presented implements a modular architecture where various modules are composed together to build an MCR. The modules are all designed with specific size constraints. The platform provides uniform connectivity mechanisms to modules so that users can build their MCRs easily. The platform uses a single MCU module while providing a selection of sensor, actuator and wireless communication modules. Board-level modularity, in general, tends to cause an overall increase in the complete system size (e.g., due to the space needed on the PCB for connectors). To mitigate this problem, the platform uses thinner PCB boards and low profile connectors. Note, however, that the proposed platform is meant to be used as an early stage prototyping tool for the research community (e.g., proof of concept, test benches and animal trials but not for use on humans) that researchers can then replace by a custom engineered device once their research is validated.

In software, TinyOS is used to complement the modularity of the hardware. It offers low power operation and efficient resource utilization, which makes it suitable for MCRs. Furthermore, its component based programming model and enhanced C dialect increase code reuse, which is a key requirement for rapid prototyping in software. This also allows researchers to cooperate with each other by exchanging well tested components in an open source ecosystem. Since TinyOS was designed for wireless sensor networks, its feasibility for MCRs was tested with an experiment that involved sensing, actuation and wireless communication. The results showed that the TinyOS is a viable and appropriate OS for MCR software development.

The presented work also lays the foundation for an integrated design and simulation environment. This environment would use a model-based approach where all hardware modules and software components are represented by a component model. Users select models from a library and build their application by composing the models on a visual canvas. Preliminary work has been done in creating this design environment using the Web-based Generic Modeling Environment (WebGME) [21]. Using this web-based tool, users can create TinyOS applications by wiring together components and writing implementations (TinyOS modules) with a text editor provided by the tool. Users can then compile their application using a cloud-based build system, thus avoiding to install their own toolchain. However, more work is needed to integrate it with the hardware platform presented in this paper. More research is also needed in providing higher levels of software abstraction, higher than those available in TinyOS, so that applications can be built with minimal editing of textual code.

VI. Acknowledgment

This work was supported by the National Science Foundation under Grants No. CNS-1239355 and IIS-1453129. This material is also based upon work supported by the National Science Foundation Graduate Research Fellowship Program under Grant No. 1445197. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

[1] G. Dogangil, B. L. Davies, and F. Rodriguez y Baena, "A review of medical robotics for minimally invasive soft tissue surgery," *Proc. Inst. Mech. Eng. H.*, vol. 224, no. 5, pp. 653–79, 2010.

[2] P. Valdastri, M. Simi, and R. J. Webster III, "Advanced technologies for gastrointestinal endoscopy," *Annu. Rev. Biomed. Eng.*, vol. 14, no. 5, pp. 397–429, 2012.

[3] A. C. Lehman, J. Dumpert, N. A. Wood, L. Redden, A. Q. Visty, S. Farritor, B. Varnell, and D. Oleynikov, "Natural orifice cholecystectomy using a miniature robot," *Surg. Endosc.*, vol. 23, no. 2, pp. 260–266, 2009.

[4] G. Tortora, P. Dario, and A. Menciassi, "Array of robots augmenting the kinematics of endocavitary surgery," *IEEE/ASME Trans. Mechatronics*, vol. 19, no. 6, pp. 1821–1829, 2014.

[5] N. A. Patronik, T. Ota, M. A. Zenati, and C. N. Riviere, "A miniature mobile robot for navigation and positioning on the beating heart," *IEEE Trans. Robot.*, vol. 25, no. 5, pp. 1109–1124, 2009.

[6] M. Beccani, C. Di Natali, L. Sliker, J. Schoen, M. Rentschler, and P. Valdastri, "Wireless tissue palpation for intraoperative detection of lumps in soft tissue," *IEEE Trans. Bio-Med. Eng.*, vol. PP, no. 99, 2013.

[7] D. Canes, A. C. Lehman, S. Farritor, D. Oleynikov, and M. M. Desai, "The future of NOTES instrumentation: flexible robotics and in vivo minirobots," *J. Endourol.*, vol. 23, no. 5, pp. 787–792, 2009.

[8] M. Beccani, E. Susilo, C. Di Natali, and P. Valdastri, "SMAC A Modular Open Source Architecture for Medical Capsule Robots," *Int J Adv Robot Syst*, vol. 11, pp. 1–16, 2014.

[9] "Arduino." [Online]. Available: www.arduino.cc

[10] "Given imaging," 2014. [Online]. Available: www.givenimaging.com

[11] P. Valdastri, A. Menciassi, A. Arena, C. Caccamo, and P. Dario, "An implantable telemetry platform system for in vivo monitoring of physiological parameters," *IEEE Trans. Inf. Technol. Biomed.*, vol. 8, no. 3, pp. 271–8, 2004.

[12] P. Swain, "Wireless capsule endoscopy." *Gut*, vol. 52 Suppl 4, pp. 48–50, 2003.

[13] J. Thonè, S. Radiom, D. Turgis, R. Carta, G. Gielen, and R. Puers, "Design of a 2 mbps fsk near-field transmitter for wireless capsule endoscopy," *Sensors and Actuators A: Physical*, vol. 156, no. 1, pp. 43 – 48, 2009.

[14] P. Valdastri, A. Menciassi, and P. Dario, "Transmission power requirements for novel zigbee implants in the gastrointestinal tract," *IEEE Trans. Bio-Med. Eng.*, vol. 55, no. 6, pp. 1705–10, 2008.

[15] G. Tortora, P. Valdastri, E. Susilo, A. Menciassi, P. Dario, F. Rieber, and M. O. Schurr, "Propeller-based wireless device for active capsular endoscopy in the gastric district," *Minim. Invasive. Ther. Allied. Technol.*, vol. 18, no. 5, pp. 280–90, 2009.

[16] P. Valdastri, R. Webster, C. Quaglia, M. Quirini, A. Menciassi, and P. Dario, "A New Mechanism for Mesoscale Legged Locomotion in Compliant Tubular Environments," *IEEE Trans. Robot.*, vol. 25, no. 5, pp. 1047–1057, Oct. 2009.

[17] P. Levis, "Tinyos: An open operating system for wireless sensor networks (invited seminar)," in *Mobile Data Management, 2006. MDM 2006. 7th International Conference on*, 2006, pp. 63–63.

[18] N. Cooprider, W. Archer, E. Eide, D. Gay, and J. Regehr, "Efficient memory safety for tinyos," *Proceedings of the 5th international conference on Embedded networked sensor systems (SenSys)*, 2007.

[19] T. von Eicken, D. E. Culler, S. C. Goldstein, and K. E. Schauser, "Active messages: a mechanism for integrated communication and computation," *SIGARCH Comput. Archit. News*, vol. 20, no. 2, pp. 256–266, Apr. 1992.

[20] E. Susilo, P. Valdastri, A. Menciassi, and P. Dario, "A miniaturized wireless control platform for robotic capsular endoscopy using advanced pseudokernel approach," *Sensor. Actuat. A-Phys.*, vol. 156, no. 1, pp. 49–58, 2009.

[21] M. Maroti, R. Kereskenyi, T. Kecskes, P. Volgyesi, and A. Ledeczi, "Online Collaborative Environment for Designing Complex Computational Systems," in *The International Conference on Computational Science (ICCS 2014)*, Elsevier Procedia. Cairns, Australia: Elsevier Procedia, 2014.