$\mathbf{R}^2$: Over-the-Air **R**eprogramming on Computational **R**FIDs

Die Wu[‡][†], Muhammad Jawad Hussain[‡][*], Songfan Li[‡], Li Lu[‡]
[‡]School of Computer Science and Engineering, University of Electronic Science and Technology of China
[†]School of Information and Software Engineering, University of Electronic Science and Technology of China
[*]School of Communication and Information Engineering, University of Electronic Science and Technology of China

Abstract—**In recent years, Computational RFID tags (CRFIDs) have emerged as viable software-defined platform for academic research and experimentation. In addition to EPC protocol, these tags perform sensing and computation through power scavenging. However, existing CRFIDs are pre-programmed to execute a specific firmware whereas programming tools comprising wired interface and PC-based software are required to erase, modify or reprogram the tag. Such limitation demands an over-the-air (OTA) scheme which can wirelessly upgrade or reprogram the CRFID tags while following the EPC protocol.**

We present the design, implementation and evaluation of $\mathbf{R}^2$ - the first OTA reprogramming scheme for CRFID tags without requiring any modification to EPC protocol, or an upgrade to RFID reader or CRFID tag. We develop a User Interface and realize our scheme on three platforms which include both software-defined as well as chip-based CRFIDs, i.e., WISP5.1 and Optimized WISP (Opt-WISP), and Spider tag. It also includes both the FLASH and FRAM based micro-controller memories. We evaluate our scheme from three aspects: time and energy overhead for reprogramming operation itself; overall system delay to verify the compatibility of $\mathbf{R}^2$ with EPC protocol; success rate in line with interrogation range. We foresee our endeavour to offer viability of OTA reprogramming and upgrade for CRFID systems.

Index Terms—**Computational RFID; OTA Reprogramming; Firmware Upgrade; EPC.**

I. Introduction

Computational RFID tags (CRFIDs) can be categorized as software-defined and fully passive UHF RFID tags which can offer numerous computational and sensing facilities. The tag itself is designed from ultra-low power and discrete components in shape of a PCB circuit and follows the architecture of a backscatter radio. This typically involves dipole antenna, receiver, backscatter transmitter, power harvester, memory, power management and a micro-controller unit (MCU) as a processing module. The MCU is the core engine which performs three main functions: Executes Class-1 Generation-2 UHF RFID protocol [1]; Realizes energy supervisor to operate the system in duty cycle mode; Executes the computational or sensing tasks while polling the analog or digital sensors and accessing EEPROM for data retention. Since their inception with WISP project [2] about ten years ago, CRFIDs have emerged as an appealing platform for academic research and experimentation like sensing the body movements [3], [4], health monitoring [5], [6], bio-signal sensing [7], passive environment sensing [8]–[10], access control [11], and cardinality estimation [12], [13]. Broadly speaking, we categorize WISP-based systems as “software-defined" CRFID tags which are designed on discrete architecture. The other family, which we term as “chip-based" CRFID tags, uses commercial chips which execute EPC protocol and provide auxiliary power and communication interface for external modules. Two such chips are Andy100 [14] and SL-series [15]. Though the system is fully passive in nature, we find battery assisted version in [16] to realize an acoustic localization system.

Besides numerous advantages, the MCU in these devices is programmed with only a specific firmware during deployment time, which restricts their wider deployment and application flexibility. Though the tag itself is wirelessly controlled by RFID reader, we need specific programming tools to modify, erase or reprogram even a single firmware function. The situation becomes cumbersome once CRFIDs are deployed in hard-to-reach places or their scale grows. For example, a WISP-based CRFID is buried in concrete blocks to measure temperature [8]; and any envisaged network of CRFIDs where dozens of tags are deployed to perform some sensing task. Under such conditions, the user has to physically access each individual tag and change the firmware through wired programming adapter and PC-based software tools. This necessitates the demand for a flexible over-the-air (OTA) reprogramming scheme whereby an RFID reader should be able to wirelessly reprogram the MCU by using EPC protocol.

In recent years, the OTA programming or OTA upgrade has emerged as a hot research topic specifically in wireless sensor networks (WSNs). However, these systems savor two advantages over CRFID tags. The WSN nodes are typically battery assisted devices, and can afford the high energy budget required for reprogramming the MCU FLASH. Second, the wireless protocols like Zigbee or Bluetooth offer adequate flexibility to transfer new firmware image without strict time and size limitations. In first intuition, the OTA programming in CRFIDs seems viable as EPC protocol offers Access Round operations to transfer bulk data between the reader and the tag. However, CRFID systems are fully passive devices and it might not be possible to transfer complete firmware image in a single operation because of energy constraints. The situation is further complicated once the protocol restricts the reader to transfer the data in few byte segments, for which the tag has to calculate the CRC and acknowledge the successful receipt of each data segment within 20 *ms*. Therefore, we need an energy-aware OTA scheme that efficiently maps the firmware image in line with tag's on-board memory while executing the mandatory operations of EPC protocol. Importantly, such scheme should be compatible with EPC protocol without demanding any hardware upgrade to CRFID tags or modifications to commercial RFID reader.

We address this issue by presenting the first OTA reprogramming scheme for CRFID tags named as $\mathbf{R}^2$, which considers multiple intricacies for a system level design. For EPC protocol, as no specific commands are offered for reprogramming operation, we embed reprogramming instructions within routine *Write* command. Towards the tag side, we consider energy and time overhead for reprogramming operation. At software level, we partition the target firmware image in multiple segments in accordance with link timings and MCU memory. Each image segment is further associated with specific memory location of the tag. This leverages reprogramming flexibility in case a user wants to reprogram only a specific segment. If the CRFID has been pre-programmed with firmwares of accelerometer sensor, temperature sensor and LED control and user wishes to "swap" the temperature sensing firmware with humidity sensing, then $\mathbf{R}^2$ only erases the temperature firmware and replaces it with humidity firmware. Such memory arrangement scheme conserves harvested power as well as computational and time overhead. Finally, the firmware reprogramming is realized with assistance of bootloader that facilitates the specific placement of firmwares in memory. This offers flexibility of reprogramming a specific firmware segment instead of erasing and re-writing the whole memory itself.

We evaluate $\mathbf{R}^2$ on three tags which belong to both families of CRFIDs. As shown in Fig. 1, the tags from software-defined CRFID family include Opt-WISP (left) and WISP5.1 (middle). The former is primarily based on Intel-WISP4.1DL and uses MSP430F2132 as MCU while latter includes MSP430FR5969. Third is the chip-based Spider CRFID tag by Farsens (right) that uses Andy100 CRFID chip, and we interface MSP430F2132 as its MCU. Similarly, we also evaluate $\mathbf{R}^2$ with different embedded memory systems, i.e., FLASH memory in MSP430F2132 and Ferroelectric RAM (FRAM) in MSP430FR5969. For all three systems, we develop a User Interface which is based upon Impinj Software Development Kit (SDK) [17]. We evaluate our system from following aspects: time and energy overhead for reprogramming operation; overall system delay including EPC protocol and reprogramming operations both at reader and tag sides; and success rate within the reader's interrogation range. All said parameters are evaluated for three CRFID tags and a comparison is presented. The results show that the reprogramming operation in $\mathbf{R}^2$ introduces a delay of 1.01 μs for WISP5.1 which can directly rewrite the data to FRAM without erasure operation, and up to 14.4 *ms* both for Spider tag and Opt-WISP as they need to erase and reprogram the FLASH memory. We further evaluate energy overhead by leveraging MCU frequencies for three tags. Regarding the energy overhead, the CRFIDs with FLASH memory consume 118.64 μJ for erase and reprogram operations while one with FRAM consumes 7.264 *nJ*. For reprogramming 512 bytes, the maximum overall system delay amounts to 28116.68 *ms*, 13886.33 *ms* and 13869.11 *ms* for Spider tag, Opt-WISP and WISP5.1 while operating MCU at 1 *MHz*. Lastly, we observe a success rate of 93%, 84% and 89.5% for Spider tag, WISP5.1 and Opt-WISP at 1 meter interrogation range.

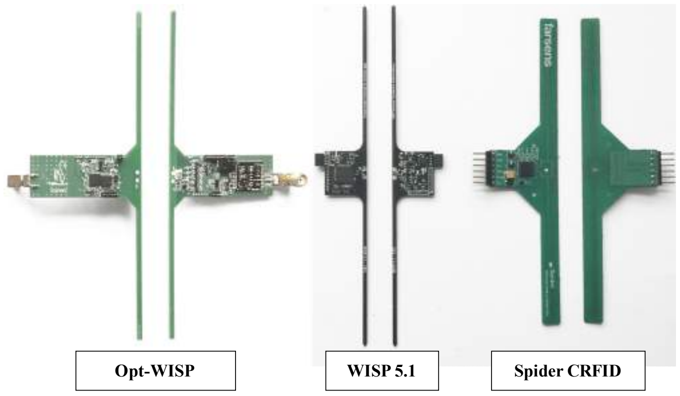


Fig. 1. Software-defined CRFIDs Opt-WISP and WISP5.1, and chip-based CRFID Spider.

Following are the chief contributions of our work:

- We present $\mathbf{R}^2$ which is the first OTA reprogramming scheme for CRFID tags.
- $\mathbf{R}^2$ is fully compatible with EPC-C1G2 protocol without demanding any modifications to RFID reader or hardware upgrade to CRFID tags.
- $\mathbf{R}^2$ efficiently maps a firmware image according to FLASH segments of MCU, which offers flexibility in reprogramming operation and also conserves harvested energy, time and computational complexity.
- We throughly evaluate our system for three CRFID tags which belong to both software-defined and chip-based CRFID families. Our selection also includes two type of MCU memories, including FRAM and FLASH.

Admittedly, if everyone can reprogram the tags, the system would be very vulnerable. This problem can, however, be resolved by authenticating the Reader which would be our future work.

The rest of this paper is organized as follows. We introduce the related work in Section II and describe the design of our system in Section III. The implementation and evaluation metrics are explained in Section IV. The evaluation results are included in Section V. The paper concludes in Section VI.

II. Related Work

To best of our knowledge, $\mathbf{R}^2$ is the first work that realizes OTA reprogramming on CRFID systems. We only find three relevant works on software-defined CRFID tags. In Firm-Switch [18], the authors remotely switch the behavior of CRFID tag by selecting amongst pre-programmed firmwares. A pre-determined look-up table approach is followed in [19] where the MCU is restrict to follow the pre-determined execution cycle flow. Also, the authors presented an energy

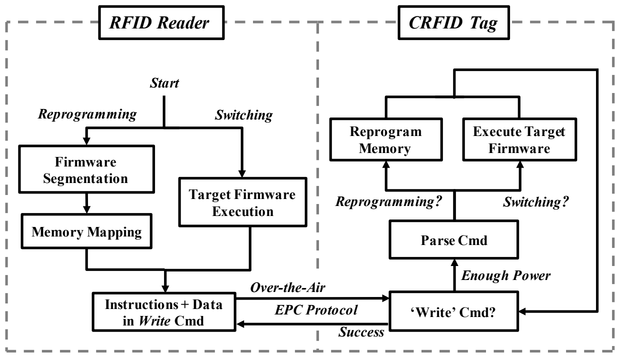


Fig. 2. Generic block diagram of $\mathbf{R}^2$.

	MemBank (2 bits)	*WordPtr* (16 bits)	*Data* (2 bits)	
⋯	00: Reprogram	Starting Address	Firmware Size	⋯
	01: New Firmware	Memory location	Firmware segment	
	10: Execute Firmware	Address of Reset Vector	—	
	11: Default operation as per EPC protocol			

Fig. 3. Instruction Encoding in *Write* Command for software-defined CRFIDs.

aware scheduling scheme [20] that maps the harvested voltage with appropriate firmware for execution.

Besides WISP-CRFID and its various hybrids, we find two commercial CRFID chips, Andy100 and SL-series. Both chips execute the EPC standard and additionally provide the power and communication interfaces (SPI or I^2C) for external modules like MCU or the digital sensors. However, no OTA operation is provided in both cases. The CRFID tags based upon Andy100 chip are used to sense data for Internet-of-Things (IoT) [21]. The SL900A chip is used for soil moisture monitoring [22] and reading tagged objects [23].

We find promising OTA programming schemes in WSNs. XNP [24] is a single hop protocol that broadcasts the intended firmware, while an energy efficient scheme is presented in [25] that only distributes the changes to currently running programs. Based on multi-hop network, Deluge [26] provides a reliable data dissemination for all sensor nodes. Aqueduct [27] and TinyCubus [28] are proposed to distribute programs to the selected nodes. As industrial products, WaspMote can be wirelessly programmed through Zigbee [29], while a user can use Bluetooth-enabled smartphone to reprogram the nRF51822 chip [30].

III. System Design

We discuss the design of $\mathbf{R}^2$ both from the reader and tag sides. At reader's end, we partition the firmware image into multiple segments in accordance with MCU memory. Next, we embed the reprogramming instructions and firmware segments in the $Write$ command. Towards the tag side, boot-loader receives the instructions along with firmware segments and executes the reprogramming operation. To make effective use of the memory and to provide reprogramming flexibility, each firmware image and corresponding memory reset vector are placed at their specific memory segment. The overall scheme is illustrated in Fig. 2 and explained below:

A. Instruction Encoding

In EPC protocol, the reader can transmit customized data to the tag through *Write* or *BlockWrite* instruction. However, *BlockWrite* is an optional command which is not supported by all CRFID tags, i.e., Spider tag. Therefore, all three devices are programmed by sending multiple *Write* commands which is the approach used during our evaluation.

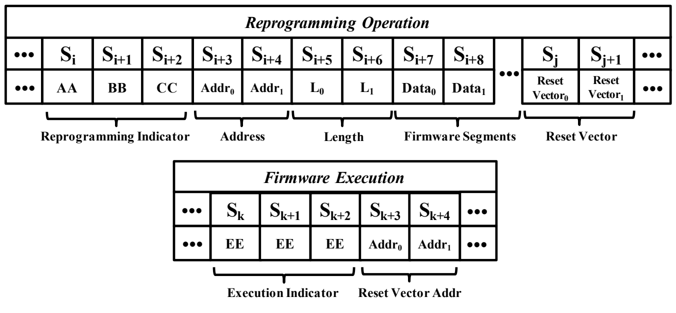


Fig. 4. Instruction Encoding for chip-based CRFIDs.

For software-defined CRFIDs, we encode the instructions in *Write* command by embedding the parameters in three fields: 2 bits of *MemBank*, 16 bits of *WordPtr* and 16 bits of *Data*. Since these tags use MCU RAM instead of reserved memory for EPC, TID and User memory, therefore, we can utilize 2 bits of *MemBank* as a flag to differentiate between start reprogramming instruction (coded as 00_b), firmware image transmission instruction (01_b), target firmware execution instruction (10_b) and routine *Write* instruction (11_b) as illustrated in Fig. 3. In our scheme, if *MemBank*=00_b, the following *WordPtr* and *Data* indicate the beginning of memory address and size of the firmware image to be reprogrammed. If *MemBank*=01_b, the *WordPtr* and *Data* indicate the target memory address and 16-bit firmware image segment. For *MemBank*=10_b, the *WordPtr* indicates the address of reset vector for target firmware to be executed. Lastly, the *MemBank*=11_b indicates the routine *Write* operation, i.e., accessing the data in User memory.

The chip-based CRFID tag (Spider) is interfaced to an external MCU through SPI interface. The internal architecture of the CRFID chip restricts the data transfer to a maximum of 8 bits at one time. To address this issue, following procedure shown in top portion of Fig. 4 is devised: we define the 8 LSBs of *Data* field as S_i in the i^{th} *Write* command for *MemBank*=11_b. To start the reprogramming operation, the instruction is defined as $S_i \| S_{i+1} \| S_{i+2} = AA_h \| BB_h \| CC_h$. This is followed by next two 8 LSBs, i.e., $S_{i+3} \| S_{i+4}$ which are combined to the memory address, while, the length of the firmware segments is contained in $S_{i+5} \| S_{i+6}$. Then following data is the firmware segment to be reprogrammed. In here, the value of reset vector is appended with the last firmware segment. To execute the firmware, the default instruction is defined as $S_k \| S_{k+1} \| S_{k+2} = EE_h \| EE_h \| EE_h$ which is followed by the reset vector address of the target firmware embedded in $S_{k+3} \| S_{k+4}$, as shown in bottom portion of Fig. 4.

B. Instruction Decoding and Execution

In software-defined CRFIDs, the MCU itself executes the EPC protocol, decodes the command and parses the parameters

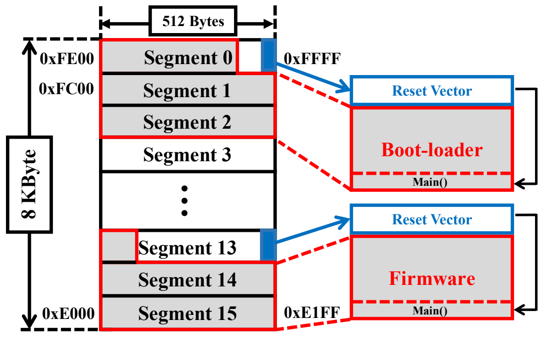


Fig. 5. Memory Arrangement in Opt-WISP and Spider CRFID tags.

embedded in three fields of *Write* command. In contrast, chip-based CRFIDs first decode the reprogramming instructions within their chip and then transfer the 8 LSBs of *Data* field to external MCU through SPI. Based on different working schemes, the instruction decoding process is executed differently for both types of CRFIDs.

In case of software-defined CRFIDs, the boot-loader receives the *Write* command, parse all three fields and checks the CRC value. Then, it executes the instruction and replies with a $Success$ to the reader. As stated earlier, the value of *MemBank* indicates the next task on the tag, i.e., for *MemBank*=10_b, the boot-loader shifts the value of *WordPtr* (reset vector address of target firmware) to the *Program Counter* that holds the address of the next instruction to be executed. For *MemBank*=00_b, the boot-loader sets the tag to reprogramming mode and erases the memory (for FLASH-based MCUs). For *MemBank*=01_b, boot-loader writes the contents of *Data* field to the address stored in the *WordPtr* field. For *MemBank*=11_b, boot-loader leaves the reprogramming mode and waits for the next command from the reader.

For chip-based CRFIDs, the boot-loader in MCU receives the data from SPI, parses the commands and firmware images, and reprograms the memory. In this case, the CRFID chip will initiate SPI communication and transfer the 8 LSBs of the *Data* field once *MemBank*=11_b. The boot-loader in MCU receives the data, stores in the RAM and labels each 8-bit data segment as S_i. Whenever the reprogramming instruction (AA_h, BB_h, CC_h) is received, the boot-loader sets the MCU to reprogramming mode and initializes the memory based upon next four bytes (S_{i+3} and S_{i+4}, S_{i+5} and S_{i+6}) that contains the starting address and the length of the firmware image. Then, the boot-loader programs the data in the memory from S_{i+7}. Once the last firmware segment is received, the reset vector is placed at the end of the current memory segment. Once the execution instruction (three consecutive EE_h) is received, the boot-loader stores the next two bytes of SPI data in *Program Counter* to execute the target firmware.

Since our work is the first proof-of-concept evaluation of wireless reprogramming of CRFID systems, we do not consider harsh multipath environment. However, in case the *Write* command fails, the Reader can, for example resend the *Write* command until a "*Success*" message is received from the tag.

C. Memory Arrangement

The memory arrangement relates to the mapping of firmware image segments in accordance with memory segments of the MCU. In our case, the Opt-WISP and Spider tags include MSP430F2132 with FLASH memory, whereas WISP5.1 includes MSP430FR5969 with FRAM which is one of the latest additions to the MSP family of ultra-low power MCUs. Broadly speaking, FRAM can be used as a universal memory for program code as well as variables. Importantly, there are no specific memory segments and it can be partitioned as desired. In our case, firmware images, constants, variables, stacks and so forth are all allocated in the FRAM. To reprogram FRAM, we simply need to write the new firmware image on a specific memory location without performing the erase operation, as in Flash memories.

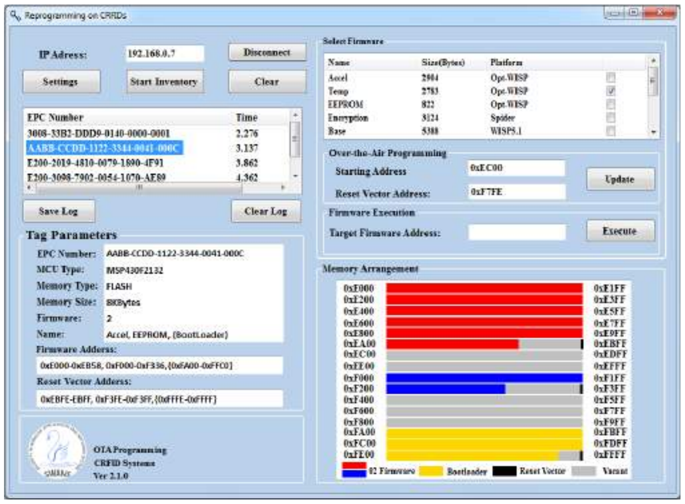
The 8 KByte Flash memory in Opt-WISP and Spider tags is partitioned in 16 segments of 512 bytes, as shown in Fig. 5. To reprogram, we need to first erase a complete memory segment followed by the write operation. Also, a reset vector is required to be placed at the end of each segment. To effectively utilize the FLASH memory and provide the flexibility of erasing individual firmware images, $\mathbf{R}^2$ places each firmware image towards the beginning of a FLASH segment.

D. Transition between Active and Power Saving Modes

CRFID tags are power constrained devices that operate in duty cycle mode to conserve power. The MSP430 series of MCUs offer one active mode and several software-selectable low power modes. In brief, all clocks are active in the active mode, while the CPU is disabled and the RAM is retained in low power mode (LPM4 mode used in our scheme). For Opt-WISP and WISP5.1, the execution starts with LPM4 mode and the tag is interrupted to work in active mode once the communication starts from the reader's side. If power is sufficient, the tag will execute the reprogramming operation as directed. Otherwise tag will set in LPM4 mode until energy supervisor generates next interrupt. For Spider CRFID tag, the micro-controller starts from LPM4 mode. In here, the SPI interrupts the MCU to active mode to perform the desired operation. Once active, the MCU keeps tracks of power in line with energy supervisor, and finally moves to LPM4 mode upon successful completion of the task.

IV. Implementation and Evaluation Metrics

We evaluate $\mathbf{R}^2$ on three platforms, i.e., WISP5.1, Opt-WISP and Spider CRFID tags. The Opt-WISP is based upon the design of WISP4.1DL but it is retrofitted with two independent antennas and power harvesters. The first antenna-harvester pair is used for realizing receiving EPC commands, power harvesting and backscattering operation similar to WISP4.1DL. The second antenna-harvester pair is used exclusively for sensing applications. Experimental results show that, such architecture results to higher output ranges and high-end sensing capabilities. Moreover, the MCU is fed with 3.3 VDC instead of 1.8 VDC (as in WISP4.1DL). The higher MCU voltage is the prime reason that we use Opt-WISP


Reprogramming on CRFIDs

IP Adress: 192.168.0.7

EPC Number	Time
3008-33B2-DDD9-0140-0000-0001	2.276
AABB-CCDD-1122-3344-0041-000C	3.137
E200-2019-4810-0079-1890-4F91	3.862
E200-3098-7902-0054-1070-AE89	4.362

Tag Parameters

EPC Number: AABB-CCDD-1122-3344-0041-000C
MCU Type: MSP430F2132
Memory Type: FLASH
Memory Size: 8KBytes
Firmware: 2
Name: Accel, EEPROM, (BootLoader)
Firmware Adderss: 0xE000-0xEB58, 0xF000-0xF336,{0xFA00-0xFFC0}
Reset Vector Adderss: 0xEBFE-EBFF, 0xF3FE-0xF3FF,{0xFFFE-0xFFFF}

OTA Programming
CRFID Systems
Ver 2.1.0

Select Firmware

Name	Size(Bytes)	Platform	
Accel	2904	Opt-WISP	☐
Temp	2783	Opt-WISP	✓
EEPROM	822	Opt-WISP	☐
Encryption	3124	Spider	☐
Base	5388	WISP5.1	☐

Over-the-Air Programming
Starting Address: 0xEC00
Reset Vector Address: 0xF7FE

Firmware Execution
Target Firmware Address:

Memory Arrangement

Fig. 6. User Interface passes reprogramming or execution instructions through commercial RFID reader, receives the tag's reply and displays the results.

in place of WISP4.1DL because the MCU (MSP430F2132) requires at least 2.2 VDC to reprogram the FLASH. In case of Spider tag, the CRFID chip itself provides 3.2 VDC to the MCU (MSP430F2132). This way, we evaluate $\mathbf{R}^2$ on MCUs with two different on-chip memories, FRAM in MSP430F2132 and FLASH in MSP430FR5969. A Graphical User Interface based on the software development kit of Impinj Speedway R420 is developed in C#. To have a comprehensive evaluation of our system, we evaluate the time delay of overall system, energy overhead and the success rate in different interrogation ranges.

A. User Interface

As shown in Fig. 6, the User Interface connects to commercial reader and displays the information regarding the tags in top left window. Once a specific tag is selected, its parameters are displayed in *Tag Parameters* panel. The memory addresses of the firmwares inside MCU memory are displayed towards the bottom of this panel in *Firmware Address* field. The memory address of boot-loader is displayed within braces to distinguish from other firmwares, e.g., *{0xFA00-0xFFC0}* which shows the starting and ending address of the boot-loader. To give a graphic illustration, we also portray the memory addresses of all existing firmwares in *Memory Arrangement* panel. As shown in bottom right of the figure, the colored area marks the firmware images, reset vector is shown at the end of the memory segment while the blank area indicates the location of empty memory. To execute the firmware, the user can input the address of intended firmware in *Firmware Execution* panel.

For reprogramming operation, the user first needs to select the desired firmware in *Select Firmware* Panel. Then, the *Over-the-Air Programming* panel provides accessibility to specify the memory location for the selected firmware that comprising of *Starting Address* and *Reset Vector* fields. This way, a new firmware can be wirelessly transmitted to a specific memory location of the tag through *Write* Command.

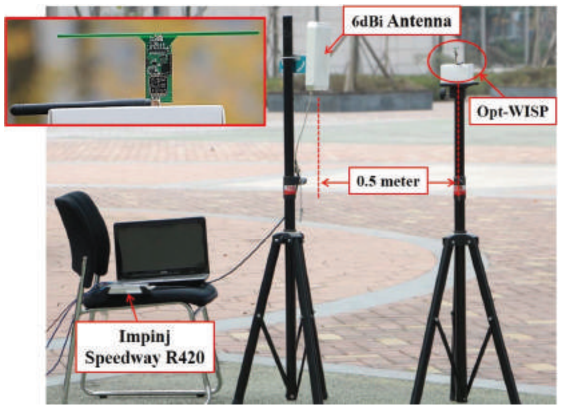


Fig. 7. Experimental Setup.

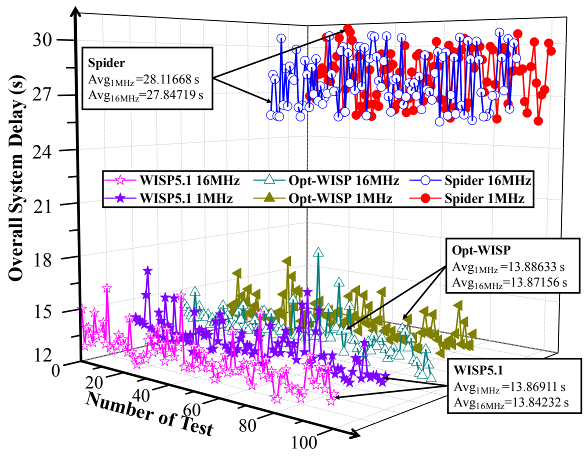


Fig. 8. Overall System Delay

B. Evaluation Metrics

The evaluation aims to validate following three aspects:

1) Overall System Delay: The overall time delay in reprogramming 512 bytes of firmware image as measured on three platforms (Opt-WISP, WISP5.1 and Spider CRFID tags). The evaluation includes the time consumption incurred by up-link and down-link communication, memory reprogramming, CRC calculation.

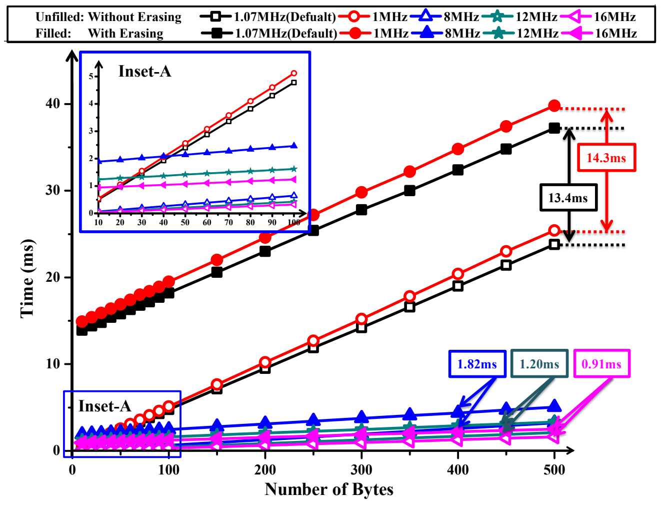
2) Time and Energy Overhead: The time delay and energy overhead once $\mathbf{R}^2$ reprograms the memory itself. The evaluation excludes the overhead caused by EPC protocol and is specific to the boot-loader operating at different MCU clock frequencies on three CRFID tags.

3) Success Rate and Interrogation Range: We calculate the success rate in line with interrogation range for all three tags. The results are compared with FirmSwitch scheme [18].

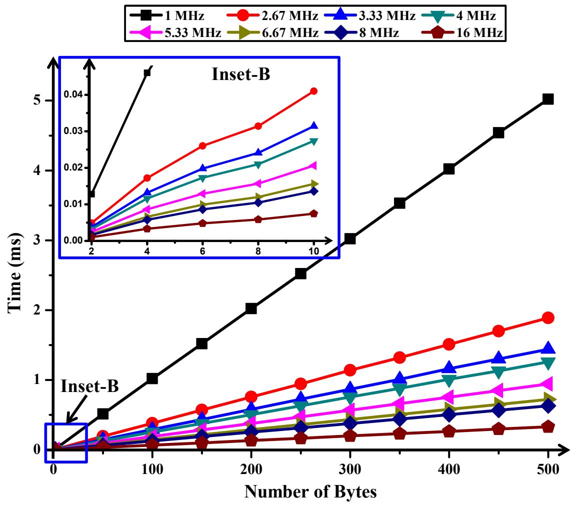
V. Evaluation Results

A. Overall System Delay

Since EPC-C1G2 strictly limits the interrogation and reply timings between the reader and the tag, it becomes important to evaluate time overhead incurred by $\mathbf{R}^2$ during reprogramming operation. In EPC standard, the interrogator maximally waits for a tag's reply for 20 *ms* after issuing the *Write*

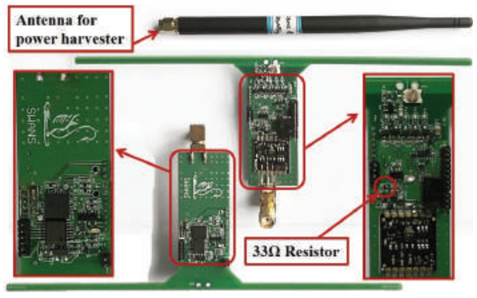


(a) MSP430F2132 with and without Memory Erasing.

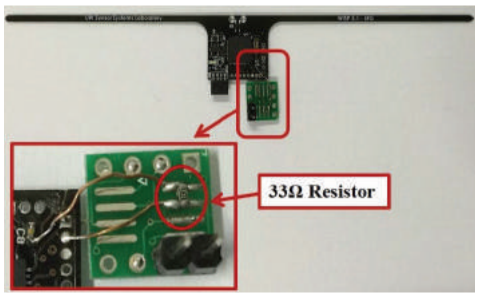


(b) Writing data on MSP430FR5969.

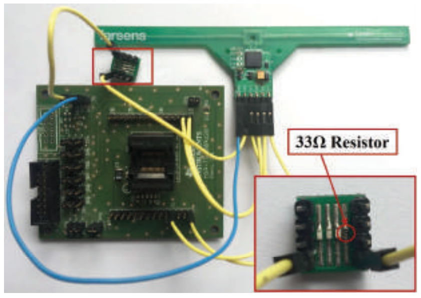
Fig. 9. Time consumption of writing data on MSP430F2132 and MSP430FR5969.



(a) Opt-WISP



(b) WISP5.1



(c) Spider

Fig. 10. Experimental setup for Opt-WISP, WISP5.1 and Spider.

command. As illustrated in Section III, the tag will backscatter the "*Success*" reply only after the reprogramming operation is executed successfully.

The overall system delay aims to ascertain the validity of $\mathbf{R}^2$ with EPC protocol. The evaluation is performed at an open place while the distance between the reader and the tag is kept as 0.5 meter, as shown in Fig. 7. The User Interface is connected with Impinj Speedway R420 RFID reader with a circularly polarized antenna with 6dBi gain. The transmission power is set to 30 dBm. The MCU clock for three CRFID tags is set to minimum and maximum clock frequencies, i.e., 1 *MHz* and 16 *MHz*. For ease of comparison, we reprogram only one segment in Opt-WISP and Spider tags, while WISP5.1 is reprogrammed with 512 bytes. The experiment is repeated for 100 times and results are shown in Fig. 8. The average response time for Spider, Opt-WISP and WISP5.1 tags at 1 *MHz* is 28116.68, 13886.33 and 13869.11 *ms* respectively, while at 16 *MHz* the values are 27847.19, 13871.56 and 13842.3 *ms*. The time delay for Spider tag is roughly double than other two tags because of the reason that only 8 bits can be transferred to MCU through SPI interface during each *Write* command. We also observe that the clock frequency and the type of memory have no serious effect on overall system delay, as we compare the results for Opt-WISP and WISP5.1.

B. Time and Energy overhead by $\boldsymbol{R}^2$

1) Time Delay Measurements: We measure the time consumption incurred by writing different Bytes of data to MCU. For MSP430F2132, the time is measured at 5 different clock frequencies: default frequency (1.07 MHz), calibrated frequencies which are 1, 8, 12 and 16 *MHz* under two conditions; write the empty FLASH memory directly without initializing; write the uninitialized FLASH memory after erasing. As shown in Fig. 9(a), the unfilled markers represent the time consumption of writing the initialized memory area, while the filled markers show the time delay introduced by writing the memory after erasing the specific memory area. However, shown results pertain to writing a single memory segment. We observe a significant linear relationship between the time consumption and the number of bytes written to the memory. In particular, the time difference between writing with and without erasing operation is the distance between two lines in Fig. 9(a). The corresponding difference at 5 clock frequencies is 13.4, 14.3, 1.82, 1.2 and 0.91 *ms* as annotated in the figure.

For FRAM, we configure the Digitally Controlled Oscillator (DCO) in MSP430FR5969 and evaluate our system on following clock frequencies: 1, 2.67, 3.33, 4, 5.33, 6.67, 8 and 16 *MHz*. The CPU clock above 8 *MHz* exceeds the FRAM access time requirements and therefore a "wait state generator" is

TABLE I
TIME AND ENERGY MEASUREMENTS OF WRITING THE BYTES ON OPT-WISP, WISP5.1 AND SPIDER

	Clock Frequency	V_{in}	V_{out}	$\triangle V$	Time (t)		Energy	
					Without Erasing	With Erasing	Without Erasing	With Erasing
Opt-WISP (2 Bytes)	Default (1.07 *MHz*)	3.346013 V	3.260247 V	0.085766 V	113 μs	13.5 ms	957.481 nJ	114.389 μJ
	1 *MHz*	3.346012 V	3.263726 V	0.082286 V	121 μs	14.4 ms	984.716 nJ	117.189 μJ
	8 *MHz*	3.346022 V	3.220431 V	0.125591 V	15.4 μs	1.84 ms	188.747 nJ	22.551 μJ
	12 *MHz*	3.346017 V	3.194128 V	0.151889 V	10.1 μs	1.21 ms	148.486 nJ	17.789 μJ
	16 *MHz*	3.346017 V	3.163778 V	0.182239 V	7.68 μs	0.92 ms	134.182 nJ	16.074 μJ
Spider (1 Byte)	Default (1.07 *MHz*)	3.163521 V	3.071369 V	0.092152 V	113 μs	13.5 ms	969.173 nJ	115.786 μJ
	1 *MHz*	3.163518 V	3.075104 V	0.088414 V	121 μs	14.4 ms	996.902 nJ	118.639 μJ
	8 *MHz*	3.163524 V	3.026743 V	0.136781 V	15.4 μs	1.84 ms	193.201 nJ	23.084 μJ
	12 *MHz*	3.163516 V	2.998503 V	0.165013 V	10.1 μs	1.21 ms	151.436 nJ	18.142 μJ
	16 *MHz*	3.163522 V	2.966066 V	0.197456 V	7.68 μs	0.92 ms	136.301 nJ	16.328 μJ
WISP5.1 (2 Bytes)	1 *MHz*	2.216182 V	2.172403 V	0.043779 V	12.8 μs		36.889 nJ	
	2.67 *MHz*	2.216181 V	2.163687 V	0.052494 V	4.88 μs		16.796 nJ	
	3.33 *MHz*	2.216179 V	2.158987 V	0.057192 V	3.76 μs		14.069 nJ	
	4 *MHz*	2.216183 V	2.155213 V	0.060970 V	3.24 μs		12.901 nJ	
	5.33 *MHz*	2.216181 V	2.147463 V	0.068718 V	2.48 μs		11.090 nJ	
	6.67 *MHz*	2.216174 V	2.131627 V	0.084547 V	1.86 μs		10.158 nJ	
	8 *MHz*	2.216188 V	2.124673 V	0.091515 V	1.62 μs		9.545 nJ	
	16 *MHz*	2.216185 V	2.103351 V	0.112834 V	1.01 μs		7.264 nJ	

required to generate the higher clocks. The resultant system clock depends upon factors like "cache hit ratio" and settings of NWAITSx register, the discussion of which is beyond the scope of our research. For evaluation at 16 *MHz*, we use the NWAITSx value of 01_b [31]. The results are shown in Fig. 9(b). We observe that the time consumption for FRAM in MSP430FR5969 is significantly lower once compared with FLASH in MSP430F2132.

2) Energy Overhead Measurements: To measure the energy overhead incurred by reprogramming operation, we use the same method as used in [32] which is used to measure the power consumption of WISP4.1DL during encryption process. We add a series resistor of 33Ω in the power path of MCU in three CRFIDs and observe the voltage drop across the resistor. The evaluation is performed for various clock frequencies. The insets in Fig. 10 highlight the small modification to Spider tag and WISP5.1, whereas we put a on-board resistor in Opt-WISP. We first measure the voltage before and after the resistor which we denote as V_{in} and V_{out} using 8846A Digit Precision Multimeter, the voltage drop caused by the resistor is ΔV. The power consumption is calculated by

$$Energy = \frac{\Delta V \cdot V_{out}}{R} \cdot t. \tag{1}$$

As the size the firmware can scale and data is transferred through multiple *Write* commands for all three platforms, we evaluate the time and energy overhead of a single *Write* command. Table I shows the results for reprogramming the memory of three CRFIDs for a single *Write* command. The maximum time and energy consumption is 14.4 ms and 118.639 μJ while reprogramming Spider CRFID at 1 *MHz*, while the minimum values are 1.01 μs and 7.264 nJ when reprogramming WISP5.1 at 16 *MHz*. From results in Table I, we observe that the higher clock frequencies will result in lower energy consumption and otherwise. Since Spider CRFID is interfaced to MCU through SPI interface and can maximally transfer 1 byte data in a single operation, the time and energy consumption of Spider CRFID tag is twofold of Opt-WISP.

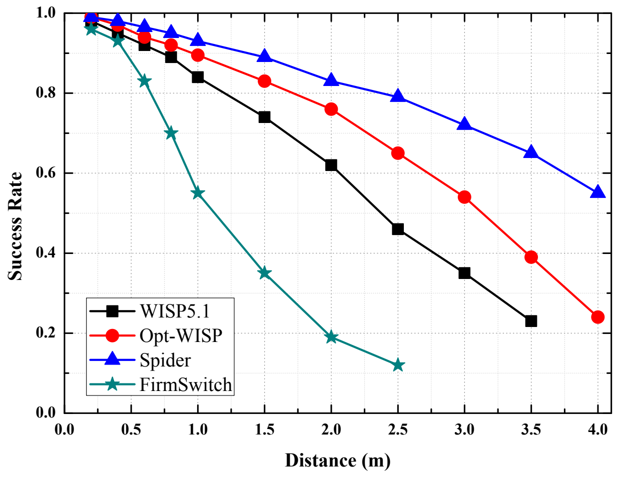


Fig. 11. Success Rate vs. Interrogation Range.

C. Success Rate vs. Interrogation Range

The success rate is evaluated for single *Write* command at an open place and evaluation setup is similar to Fig. 7 as in case of overall system delay. The distance between the reader and CRFID tag is increased from 0.2 to 4.0 meters during the course of experiment. We compare our results with FirmSwitch scheme [18] which uses WISP4.1DL and switches between multiple pre-programmed firmwares through *Write* command using the commercial RFID reader.

The evaluation results are shown in Fig. 11. We observe that Opt-WISP, WISP5.1 and Spider CRFIDs have better interrogation ranges than FirmSwitch. There is a sharp decline in the success rate of FirmSwitch after 0.5 meter because WISP4.1DL backscatters sensor data as well as pre-calculates the CRC, which consumes large amount of energy. As Opt-WISP makes use of an additional antenna-harvester pair exclusive for sensing/computational tasks, the interrogation range is increased and observed success rate is higher. In case of WISP5.1, it uses an optimized and more sensitive power harvester which increases both the range as well as the success rate. Spider CRFID tag is analogous to commercial RFID tag as far as the execution of EPC protocol is concerned, i.e., whole EPC protocol is implemented in embedded chip. The reprogramming operation is realized in the MCU which is connected through SPI and has no direct relation with the implementation of EPC protocol. This, in result, offers optimal tag operation as envisioned.

VI. Conclusion

We articulate the design, implementation and evaluation of $\mathbf{R}^2$ scheme that wirelessly reprograms the CRFID tags through commercial RFID reader. The scheme is fully compatible with EPC protocol and does not require any hardware upgrade to CRFID tags or the RFID reader. To this end, system architecture is explained with an in-depth discussion for two topologies of CRFIDs including three tags and two kinds of MCUs. The scheme is realized in the phases of instruction encoding, decoding, execution and memory arrangement. Evaluation results show that single reprogramming operation in $\mathbf{R}^2$ introduces a delay of 1.01 μs for WISP5.1 and up to 14.4 *ms* both for Spider tag and Opt-WISP. For reprogramming 512 bytes at 1 *MHz* clock, the maximum overall system delay amounts to 28116.68, 13886.33 and 13869.11 *ms* for Spider tag, Opt-WISP and WISP5.1 CRFID tags, and a success rate of 93%, 89.5% and 84% is achieved for an interrogation range of 1 meter.

Acknowledgment

This work is supported by the National Natural Science Foundation of China (Grant 61472068), and in part by the Project funded by China Postdoctoral Science Foundation (Grant 2014M550466).

References

[1] EPC C1G2 Standard, http://www.epcglobalinc.org/standards/uhfc1g2.

[2] M. Philipose et. al. Battery-free Wireless Identification and Sensing. *Pervasive Computing, IEEE*, 4(1):37–45, 2005.

[3] M. Buettner et. al. Recognizing Daily Activities with RFID-Based Sensors. In *ACM Conference on Ubiquitous computing, 2009.*

[4] Rohit Chaudhri, Jonathan Lester, and Gaetano Borriello. An RFID based system for monitoring free weight exercises. In *Proceedings of the 6th ACM conference on Embedded network sensor systems*, pages 431–432. ACM, 2008.

[5] E. Hoque, R.F. Dickerson and J.A. Stankovic. Monitoring Body Positions and Movements During Sleep Using WISPs. In *Wireless Health*, pages 44–53. ACM, 2010.

[6] Zhibin Xiao, Xi Tan, Xianliang Chen, Sizheng Chen, Zijian Zhang, Hualei Zhang, Junyu Wang, Yue Huang, Peng Zhang, Lirong Zheng, et al. An implantable rfid sensor tag toward continuous glucose monitoring. *Biomedical and Health Informatics, IEEE Journal of*, 19(3):910–919, 2015.

[7] D. Yeager, F. Zhang and A. Zarrasvand et. al. A 9 A, Addressable Gen2 sensor tag for biosignal acquisition. *Solid-State Circuits, IEEE Journal of*, 45(10):2198–2209, 2010.

[8] H. Zhang and J. Gummeson et. al. MOO: A Batteryless Computational RFID and Sensing Platform. *University of Massachusetts Computer Science Technical Report UM-CS-2011-020*, 2011.

[9] J. Gummeson, S.S. Clark and K. Fu et. al. On The Limits of Effective Hybrid Micro-Energy Harvesting on Mobile CRFID Sensors. In *ACM MobiSys, 2010*, pages 195–208. ACM, 2010.

[10] Saman Naderiparizi, Aaron N Parks, Zerina Kapetanovic, Benjamin Ransford, and Joshua R Smith. Wispcam: A battery-free RFID camera. In *RFID (RFID), 2015 IEEE International Conference on. IEEE*, 2015.

[11] Daniel Halperin, Thomas S Heydt-Benjamin, Benjamin Ransford, Shane S Clark, Benessa Defend, Will Morgan, Kevin Fu, Tadayoshi Kohno, and William H Maisel. Pacemakers and implantable cardiac defibrillators: Software radio attacks and zero-power defenses. In *Security and Privacy, 2008. SP 2008. IEEE Symposium on*, pages 129–142. IEEE, 2008.

[12] Yuanqing Zheng and Mo Li. Zoe: Fast cardinality estimation for large-scale RFID systems. In *INFOCOM, 2013 Proceedings IEEE*, pages 908–916. IEEE, 2013.

[13] Wei Gong, Kebin Liu, Xin Miao, and Haoxiang Liu. Arbitrarily accurate approximation scheme for large-scale RFID cardinality estimation. In *INFOCOM, 2014 Proceedings IEEE*, pages 477–485. IEEE, 2014.

[14] ANDY100 Evaluation Board with Integrated Start-up Circuit. Farsens, http://www.farsens.com/media/document/26/ds-spider-h254-v01.pdf [Accessed December 17, 2015].

[15] SL900A, EPC Class 3 Sensory Tag Chip - For Automatic Data Logging. AMS, http://ams.com/eng/Products/UHF-RFID/UHF-Interface-and-Sensor-Tag/SL900A.

[16] Yi Zhao and Joshua R Smith. A battery-free RFID-based indoor acoustic localization platform. In *RFID (RFID), 2013 IEEE International Conference on*, pages 110–117. IEEE, 2013.

[17] Octane SDK, https://support.impinj.com/hc/en-us/articles/202755268-Octane-SDK.

[18] Wenyu Yang, Die Wu, Muhammad Jawad Hussain, and Li Lu. Wireless firmware execution control in computational RFID systems. In *RFID (RFID), 2015 IEEE International Conference on*, pages 129–136. IEEE, 2015.

[19] Benjamin Ransford. A rudimentary bootloader for computational RFIDs. 2010.

[20] B. Ransford and S.S. Clark et. al. Getting Things Done on Computational RFIDs with Energy-Aware Checkpointing and Voltage-Aware Scheduling. In *HotPower*, 2008.

[21] Jennifer Wang, Erik Schluntz, Brian Otis, and Travis Deyle. A new vision for smart objects and the internet of things: Mobile robots and long-range UHF RFID sensor tags. *arXiv preprint arXiv:1507.02373*, 2015.

[22] C Bauer-Reich, Kay Chen Tan, F Haring, N Schneck, A Wick, L Berge, Jesse Hoey, Rudolf Sailer, and C Ulven. An investigation of the viability of UHF RFID for subsurface soil sensors. In *Electro/Information Technology (EIT), 2014 IEEE International Conference on*, pages 577–580. IEEE, 2014.

[23] Arnaud Vena, Brice Sorli, Alain Foucaran, and Yassin Belaizi. A RFID-enabled sensor platform for pervasive monitoring. In *Reconfigurable and Communication-Centric Systems-on-Chip (ReCoSoC), 2014 9th International Symposium on*, pages 1–4. IEEE, 2014.

[24] Crossbow Technology. Mote In Network Programming User Reference. TinyOS ducument, http://webs.cs.berkeley.edu/tos/tinyos-1.x/doc/Xnp.pdf.

[25] Niels Reijers and Koen Langendoen. Efficient code distribution in wireless sensor networks. In *Proceedings of the 2nd ACM international conference on Wireless sensor networks and applications*, pages 60–67. ACM, 2003.

[26] Adam Chlipala, Jonathan Hui, and Gilman Tolle. Deluge: data dissemination for network reprogramming at scale. *University of California, Berkeley, Tech. Rep*, 2004.

[27] Lane A Phillips. *Aqueduct: Robust and efficient code propagation in heterogeneous wireless sensor networks.* PhD thesis, University of Colorado, 2005.

[28] Pedro José Marrón, Andreas Lachenmann, Daniel Minder, Matthias Gauger, Olga Saukh, and Kurt Rothermel. Management and configuration issues for sensor networks. *International Journal of Network Management*, 15(4):235–253, 2005.

[29] Over the Air Programming. Libelium, http://www.libelium.com/products/waspmote/ota/ [Accessed December 17, 2015].

[30] Firmware Over the Air FOTA Updates. Mbed, https://developer.mbed.org/teams/Bluetooth-Low-Energy/wiki/Firmware-Over-the-Air-FOTA-Updates.

[31] MSP430FR59xx Mixed-Signal Microcontrollers. Texas Instruments, http://www.ti.com/lit/ds/symlink/msp430fr5969.pdf.

[32] J.R. Smith. *Wirelessly Powered Sensor Networks and Computational RFID*. Springer, 2013.