Simple Low Cost Open Source UHF RFID Reader

Nicolas Barbot, *Member, IEEE*, Raymundo de Amorim Jr., *Member, IEEE*, and Pavel Nikitin, *Senior Member, IEEE*

***Abstract*—In this paper, we present a simple low-cost SDR RFID UHF reader capable of reading a tag in real time. This reader is designed around a simple asynchronous OOK modulator in transmission and an envelope detector in reception. All tasks specific to the RFID protocol including clock recovery, data recovery and frame detection are handled in software by a Arduino Uno micro-controller. This reader is able to generate any RFID command supported by the protocol and to decode any message backscattered by the tag in real time. The details of hardware and software associated with this reader are released in open source for the community.**

***Index Terms*—EPC Gen2, software defined radio, UHF RFID reader.**

I. Introduction

RADIO Frequency Identification (RFID) is an essential technology to identify items reliably. First standard, GS1 EPC Radio-Frequency Identity Protocols Generation-2 UHF RFID Standard also known as ISO 18000-6c has been ratified in 2004 and later updated to version 2 [1]. Since then, readers and tags compliant with the standard have been developed by different companies for all world regions. At this time, typical flexible UHF tag cost is between $0.1 to $1 depending on the quantity [2]. However the cost of even a low-performance short-range reader can be on the order of $200 [3] which slows down academic experimentation with this technology. The high price of a UHF reader is mainly due to the development of complex reader chips which integrate both analog and digital circuits into a single component. Nowadays, most RFID readers are based on these RFID chips.

Also, reader chips do not offer a viable solution to study and understand the RFID protocol [1] since the API provided to the developer (or the user) is significantly different compared to the air interface. For most of the chips, an inventory round can be summarized as: 1) sending some commands to the chip over a high speed serial interface to configure the chip and start the inventory round, and 2) reading the EPC values detected by the chip and stop the inventory. Note that all the operations between these two steps are actually hidden from the user. which significantly limits the understanding of the messages exchanged during the inventory.

Manuscript received 28 September 2022; revised 6 November 2022; accepted 1 December 2022. Date of publication 12 December 2022; date of current version 20 January 2023. *(Corresponding author: Nicolas Barbot.)*

Nicolas Barbot and Raymundo de Amorim Jr. are with Univ. Grenoble Alpes, Grenoble INP, LCIS, 26000 Valence, France (e-mail: nicolas.barbot@lcis.grenoble-inp.fr).

Pavel Nikitin is with Impinj, Inc., Seattle, WA 98109 USA.

Digital Object Identifier 10.1109/JRFID.2022.3227533

Moreover, current Gen2 protocol offers a rich set of parameters to define a communication between the reader and the tags. However, parameter values are usually constrained by the chip design or simply not available from the API to limit the reader chip complexity and lower its cost. For example, the standard specifies any Tari value (which is linked to the reader data rate) between 6.25 and 25 µs [1, Sec. 6.3.1.2.4]. However, for example, the AMS 3993 chip [4] only supports Tari values of 6.25, 12.5 and 25 µs. Another example comes from timing information. The range of the parameter T_2 [1, Table 6-16] (which corresponds to the time between the end of a reader command and the beginning of the tag reply) is specified by the standard but this value is not exposed by the API of any reader chip and remains unavailable to the user. Similar remarks apply to most of the other parameters of the air interface.

Software Defined Radio (SDR) is a recent paradigm allowing to realize most of the operations of a transceiver in software. SDR designs can potentially offer a lower cost architecture and a higher flexibility since most of the modification can be realized by changing only the software. However, any UHF SDR reader has to face two main challenges in order to successfully read a UHF tag which are not present in classical wireless communication systems. The first one is to mitigate the important variation for the tag Backscatter-Link Frequency, (BLF) since EPC Gen 2 allows significant variation as high a ±22% of the BLF [1, Table 6-9]). The second one is to satisfy the timing constraints for issuing reader replies which can not be higher than $20\,T_{PRI}$ [1, Table 6-16] where $T_{PRI} = 1/\text{BLF}$. These two issues need to be carefully handled by the reader to realize a successful tag reading. Different designs can be found in the literature, but all are based on FPGA to handle the heavy processing required at the physical layer. Researchers mainly investigated two directions. The first one is based on the USRP platform developed by Ettus Research [5], [6], [7], [8]. In all these designs, the FPGA is simply used to interpolate and decimate the transmitted and received signal. All processing steps have to be done on a remote computer (including clock recovery, data recovery, frame detection, etc) and are mainly limited by the timing constraint of the RFID protocol. The other direction is to use the FPGA (or a DSP) to process the heavy tasks of the RFID protocol [9], [10], [11], [12], [13], [14], [15], these architectures allow also to use simpler micro controllers (MCU) to process the remaining tasks and are less limited by the timing constraints, but remains complex due to the FPGA development. For example, the design presented in [16], [17], which is by far the smallest FPGA design, uses more than 2000 logical cells.

SDR architecture also allows educational perspective since the associated code can easily be understood and modified to realize new functions and cover new applications. While some SDR tag designs based on simple micro-controllers are already available for the community [18], [19], [20], no simple SDR reader allowing to read a UHF tag has been reported in the literature.

In this paper, we fill the gap and present a simple low-cost SDR reader which is able to realize a complete inventory of a UHF tag. This paper is a direct continuation of the initial design presented in [21]. The proposed reader is theoretically able to generate any commands defined by the standard and to process any response from the tag in real time. In transmission, each parameter of the protocol can be set at any value (including values outside of protocol specification). In reception, this reader is able to extract the complete timing information associated to the message backscattered by the tag. The hardware architecture used by the reader is exactly the same as the one proposed in [21]. Compared to [21], this paper provides the firmware where all tasks specific to the RFID protocol (including clock recovery, data recovery, frame detection...) are entirely defined in software and can be successfully processed by any low-performance micro-controller. The proposed implementation is based on an Arduino platform which includes 8 bits/16 MHz MCU. This platform is at least two orders of magnitude less powerful than any other SDR design proposed in the literature but is sufficient to realize a fully functional RFID reader. Moreover this reader can be used to investigate the RFID protocol in a practical way by directly generating the different commands and analyzing the tag responses. It can also be used to evaluate the compliance of any UHF tag. The last point addressed by this paper is to encourage any researcher, student or person aiming to study the RFID technology to reproduce, use and improve the proposed design. For that, all hardware design files (including schematic, PCB layout, bill of material) and software design (including firmware code source) are released in open source to the community [22].

The rest of the paper is divided in three main Sections. In Section II, the architecture of the reader in both transmission and reception is presented. The partitioning between hardware and software is also described and the principle and performance of the different sub-systems are characterized. Section III presents the results associated to a complete inventory in different conditions. Finally, Section IV provides the performance and gives perspectives of the proposed design.

II. Architecture

A. Overview

The proposed RFID reader is composed of two separate boards. The first one corresponds to the RF/analog hardware part of the reader. The second one only contains the MCU which executes the firmware implementing the RFID protocol. This design allows to easily change the MCU without modifying the hardware part.

The hardware associated with this reader is extremely simple and represents less than 20 components in total.

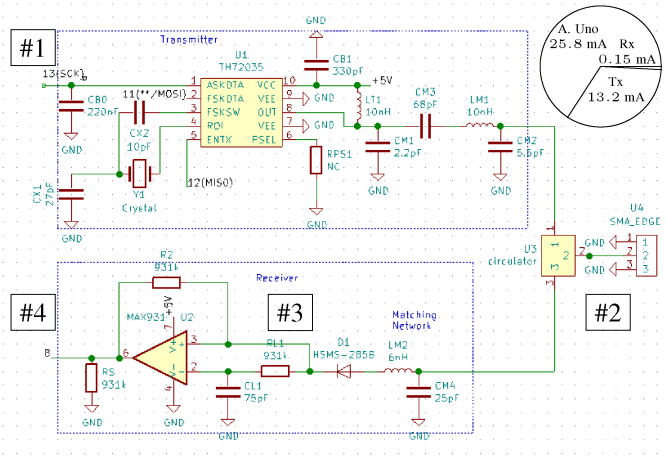

Fig. 1. Schematic of the hardware for proposed SDR RFID reader and current consumption by its various components.

Transmitter and receiver are separated (they do not share any common signal) and are only connected through a circulator. An SMA connector allows one to connect a (single) external antenna. The complete schematic is presented in Fig. 1. Transmitter is based on the Melexis TH72035 chip [23] which is a simple local oscillator that can be set to any frequency between 850 MHz and 930 MHz by proper selection of external discrete passive components. Associated circuitry is directly extracted from its evaluation board. Reception uses a simple envelope detector and a data slicer, associated circuitry is almost identical to the one presented in [21].

The chosen MCU for the UHF reader is an Arduino Uno platform [24]. This board is built around ATmega328p microcontroller and has become very popular among a large community from students to experienced engineers. The ATmega328p is however a low-performance 8 bits MCU clocked at 16 MHz but as we will see, this computing power is enough to read a UHF tag. Note also that the form factor of the hardware part allows to easily replace the micro-controller by high performance MCU such as Nucleo64 platform which embedded a 32 bits Cortex-M. Fig. 2 presents the complete UHF RFID reader with the Arduino platform and a dipole antenna.

The hardware presented in Fig. 1 is generic. All the operations specific to the EPC Gen 2 protocol are actually handled, in real time, by the firmware executed by the MCU. This firmware represents the most important part of the design. As we will see, this simple design and the associated firmware allow one to read the EPC of a UHF tag. The performance is significantly lower compared to classical industrial readers but the proposed solution also allows to realize some modes of operation that are not available in those readers. We believe that this platform represents great and unique tool to learn, understand, and evaluate the RFID technology.

B. Transmitter

The transmitter is based on the Melexis TH72035 chip [23] which is able to generate a Continuous Wave (CW) in the UHF band at a fixed frequency. The modulation used to issue the reader commands is based on On-Off Keying (OOK). Note

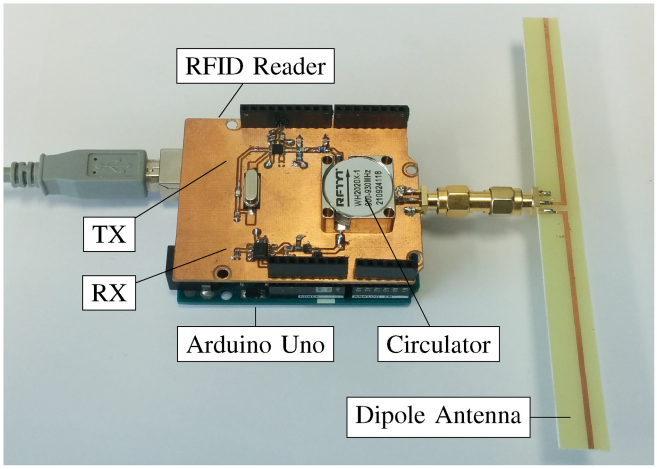

Fig. 2. Picture of the proposed SDR RFID reader. The upper board corresponds to the hardware part, the lower board is the Arduino platform.

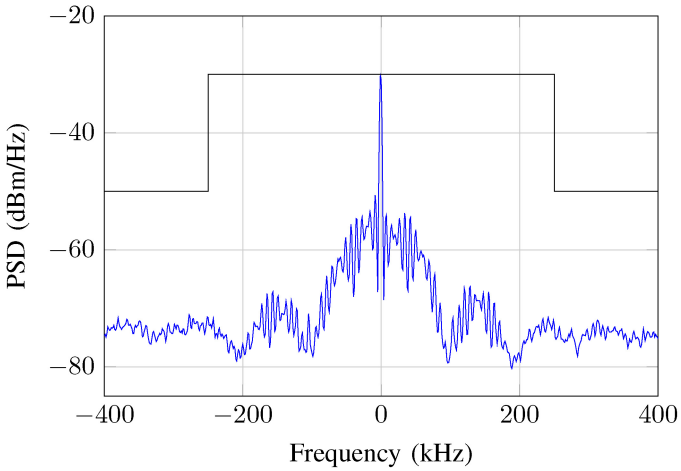

Fig. 3. Power spectral density at 915.1 MHz of the signal transmitted by the reader (at #2) using PIE data encoding (with Tari at 25 us) and mask for multiple-interrogator environments [1, Fig. 6-6].

that this modulation is still compliant with the standard even if the modulation depth is higher than the nominal value of 90% [1, Table 6-5]. This transmitter is also simply controlled by a single external signal (*i.e.,* a simple GPIO of the MCU).

The API of the firmware has mainly been chosen at the bit level of the air interface. Thus, the programmer is responsible for casting a valid binary command (including the CRC bits). The firmware simply provides a function to add the preamble (or frame-sync) [1, Sec. 6.3.1.2.8] and transmit the provided command using the PIE data encoding [1, Sec. 6.3.1.2.3] with a GPIO directly connected to the TH72035. Note that this design allows to generate any reader command and to set every parameter defined by the protocol (*e.g.,* Tari [1, Sec. 6.3.1.2.4], PW [1, Table 6-5], RTCAL [1, Sec. 6.3.1.2.8], TRCAL [1, Sec. 6.3.1.2.8]$\cdots$) in software. Also, exact timing between reader commands can be accurately controlled and non-compliant commands can be generated.

Performance of the complete transmitter has been evaluated by measuring the output signal directly at the SMA connector. Power spectral density measured at the spectrum analyzer is presented in Fig. 3 where we can check that the transmitted signal satisfies the mask for multiple-interrogator environments [1, Fig. 6-6]. Unmodulated transmitted power has been measured at $P_t = 6.5$ dBm. This value is lower than the output power of the TH72035 present in the datasheet (7.5 dBm) due to impedance mismatches and losses in the current board. Note that the maximum data rate for OOK is significantly higher than the 40 kb/s announced in the datasheet [23].

From the transmitted power P_t, the maximum distance d at which a tag can be activated is given by the Friis equation:

$$d = \frac{\lambda}{4\pi}\sqrt{\frac{P_t G_t}{P_{tag}}} \tag{1}$$

where G_t and G_r are the gain of the reader and tag antenna respectively and P_{tag} is the sensitivity of the tag (with its antenna). Considering a dipole antenna with 2 dBi gain for the reader and a tag sensitivity of $P_{tag} = -15$ dBm, the maximum distance at which the tag can be activated is $d = 40$ cm. However, we will see that the read range is lower than that due to the receiver sensitivity limitation.

C. Receiver

The receiver architecture for the proposed reader is significantly different than the classical IQ demodulator used by high-performance readers. This decision allows one to maintain the reader complexity at minimum at a price of a reduced performance. As in [21], the demodulator has been implemented in hardware and is based on a simple envelope detector and a data slicer (to dynamically set the amplitude of the decision threshold and produce a signal between 0 V and 5 V). The data slicer output is directly connected to a digital GPIO (in input) of the MCU (see Fig. 1). Fig. 4 presents the output of the envelope detector and the data slicer during the reception of the RN16 of a tag. Data slicer output is equal to 5 V when the voltage of the envelope detector is higher than the amplitude threshold and equal to 0 otherwise. The threshold amplitude is obtained by low-pass filtering the output of the envelope detector. For the envelope detector (and data slicer), the associated bandwidth have been estimated at 5 MHz which allows to receive all supported BLF values [1, Table 6-9] but also accepts a significant noise power which degrades the performance. Note that this detector is only sensitive to envelope variation and can not detect phase variation in the backscattered signal. Moreover, the sensitivity of this detector (*i.e.,* the minimum modulated power which can be detected by envelope detector and data slicer) has been estimated at -13 dBm which is, as expected, significantly higher than classical IQ demodulators. Finally, note that, contrary to classical readers, this design does not use any analog to digital converter which significantly reduces cost, complexity (and performance) but does not suffer from the transmitter leakage complication raised in [16] that typically requires adaptative carrier cancellation. Our architecture also implies that clock recovery, data recovery and frame detection have to be done entirely in software by the MCU.

TABLE I
EDGE DETECTOR AND BIT DECODER FOR A BLF OF 44 KB/S (THRESH_L = 272 AND THRESH_H = 509)

t_i	4062	4248	4438	4812	5003	5564	5942	6129	6319	6692	7071	7258	7448	7634	7824	8198
Δt_i	4062	186	190	374	191	561	378	187	190	373	379	187	190	186	190	374
$\hat{b}_j$	3		0	1		v	1		0	1	1		0		0	1

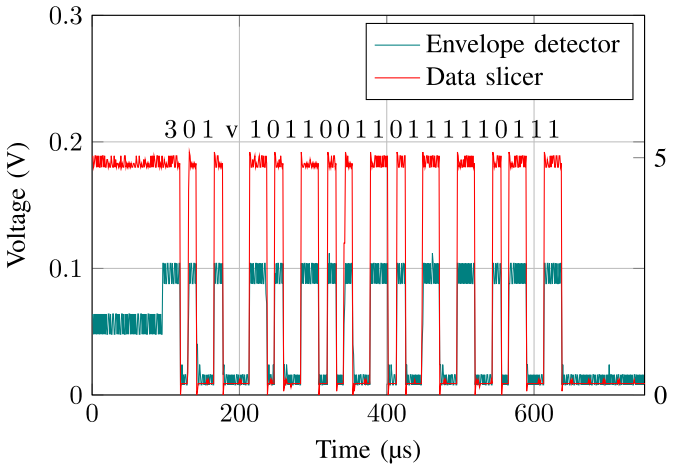

Fig. 4. Analog signals involved in the receiver (envelope detector at #3, data slicer at #4).

```
1  ISR(TIMER1_CAPT_vect) {
2    static unsigned int delta = 0, pdelta = 0xFFFF;
3
4    TCCR1B = TCCR1B ^ (1<<6); // toggle int edge
5    timing[i_glob] = ICR1;    // ti
6    delta = timing[i_glob] - timing[i_glob-1];
7    i_glob = i_glob + 1;
8    if (delta < TRESH_L && pdelta < TRESH_L) {
9      answer[j_glob++] = 0;   // 0-data
10     pdelta = 0xFFFF;
11     return;
12   }
13   else if (delta > TRESH_L && delta < TRESH_H)
14     answer[j_glob++] = 1;   // 1-data
15   else if (delta > TRESH_H && delta < 2*TRESH_H)
16     answer[j_glob++] = 2;   // violation
17   else if (time > 2*TRESH_H)
18     answer[j_glob++] = 3;   // carrier
19   pdelta = delta;
20 }
```

Fig. 5. Interrupt handler for the ATmega328p (Arduino). This code realizes, at the same time, clock recovery, data recovery and frame detection.

For the software part, the decoder has been implemented based on a simple edge detector, *i.e.,* the detector only estimates the time at which transitions occur in the (digital) received signal. Note that contrary to classical readers, no matched filter or correlator is used during the decoding. Edge detection can be accurately realized using a simple timer (already present in all micro-controllers). When the reader wants to receive a tag reply, a RX window is opened during a fixed amount of time and the timer is reset to zero. During this window, an interruption is triggered for each transition (from low to high or high to low) detected on the GPIO pin. At each interruption, the value of the timer t_i is captured and saved in the micro-controller memory.

FM0 encoding [1, Sec. 6.3.1.3.2] can be decoded using only the t_i values. To produce the binary sequence, the decoder simply computes the time interval $\Delta t_i = t_i - t_{i-1}$ between the two last transitions. A 0-data is decoded if the two last intervals are lower than the threshold value, A 1-data is decoded if the last interval is higher than the threshold. The threshold is determined statically from the nominal BLF. The complete algorithm is presented in Fig. 5.

Fig. 4, 5 and Table I can be used jointly to understand the principle of the decoder and has been generated by reading a real UHF tag. For each transition of the data slicer (see Fig. 4, the value t_i captured by the timer is saved (see Fig. 5, line 6). Table I line 1 presents the first values of t_i corresponding to the transitions in Fig. 4. The time interval Δt_i is then computed (see Fig. 5, line 6). Table I line 2 also presents the first values of Δt_i. For example Δt_1 and Δt_2 are respectively equal to $4248 - 4062 = 186$ and $4438 - 4248 = 190$. Finally, the decoded sequence is obtained from the 2 last Δt_i (see Fig. 5, line 8–14) and is presented in Table I line 3. For example, with the two previous $\Delta t_1 = 186$ and $\Delta t_2 = 190$ values, decoded bit is equal to `0`. Note that this algorithm allows one to realize, on the fly, both clock recovery and data recovery. A last point concerns the preamble detection (frame detection), since the reader has to identify the beginning of the tag reply in each RX window. This task exploits the FM0 preamble [1, Sec. 6.3.1.3.2.2] present at the beginning of the tag transmission. This preamble is equal to `1010v1` where `v` is a violation of the phase inversion. The frame detection is actually done at the same time as the data recovery by detecting the violation present inside the FM0 preamble. As for data recovery, this operation is done on the fly directly in the interrupt handler (see Fig. 5, lines 15–16). Final decoded sequence is presented in Table I line 3 and on top of the data slicer curve in Fig. 4. The interrupt presented in Fig. 5 represents the most important part of the firmware. As we will see, the execution time of this function directly determines the maximum BLF which can be supported by the reader.

Finally, note that, as seen in Section I, the two main difficulties for the RFID receiver are to handle important variation for the BLF (up to ±22%) and to satisfy the timing constraints for issuing reader replies (lower that $20\,T_{PRI}$). Surprisingly, the proposed architecture allows to solve both challenges in an original way. The first problem is handled by the edge detector, which is naturally robust to variation up to ±25% with respect to the nominal BLF (if the threshold is placed at 75% of the nominal BLF). Note that, this flexibility comes at the cost of a lower performance compared to the matched filter decoder. The second problem is solved by successfully decoding the data (which include clock, data recovery and frame detection) on the fly directly inside the interrupt handler. Thus, at the end of the RX window, if a RN16 has been detected, the binary content is directly available to generate

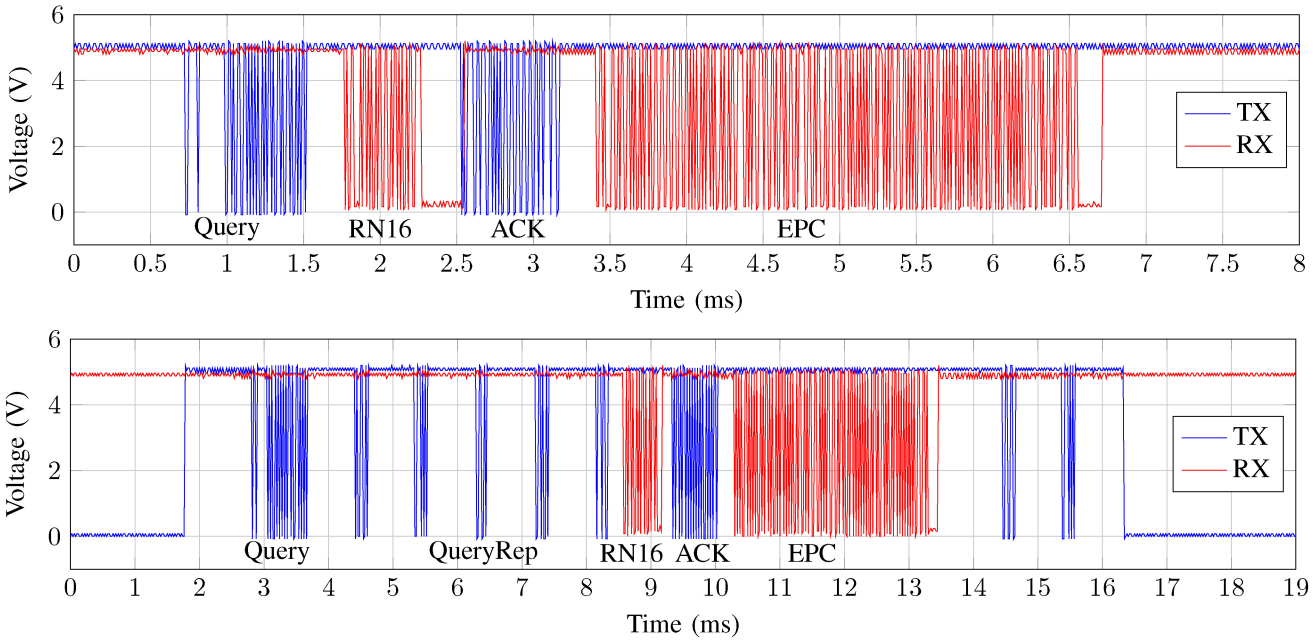

Fig. 6. (a) Single slot inventory showing the Query command, the RN16 reply, the ACK command and finally the full EPC of the tag. (b) Multiple slot inventory showing the Query and the seven QueryRep commands, RN16 and full EPC reply of the tag in slot $S = 5$. TX signal at #1, RX at #4.

the corresponding acknowledgment without almost any delay.

III. UHF Tag Inventories

This section illustrates the possibilities of this reader by reading a LAB ID UH106 which is based on an Impinj Monza R6 chip. Note that similar results have been obtained with other chips such as Impinj Monza 1, Monza 2, Alien Higgs 3 Higgs EC, NXP Ucode 8$\cdots$

A. Single Slot Inventory ($Q = 0$)

The first case considers an inventory round assuming that a single tag is present in front of the reader. This inventory requires several steps summarized in Fig. 7(a). First, the reader has to generate a Query command [1, Sec. 6.3.2.12.2.1]:

```
1000 0 00 0 00 00 0 0000 10000
```
(2)

This command allows a single slot (Q = 0) for the tag reply. This command is prepended by a preamble. After the Query command, the reader opens a RX window for the tag reply. Any tag in the ready state [1, Sec. 6.3.2.6.1]) then generates and backscatters a new RN16 which has to be received entirely during the RX window of the reader. At the end of the RX window, the reader checks if a reply is detected and then directly generates the ACK command [1, Sec. 6.3.2.12.2.4] with:

```
01 XXXXXXXXXXXXXXXX
```
(3)

where X symbols represent the 16 bits of the received RN16. Note that this command is prepended by a frame-sync [1, Sec. 6.3.1.2.8] After receiving the ACK command (before time T_2), the tag enters in the acknowledged state [1, Sec. 6.3.2.6.4] and backscatters its full EPC. This stream has to be received by the reader and decoded as any

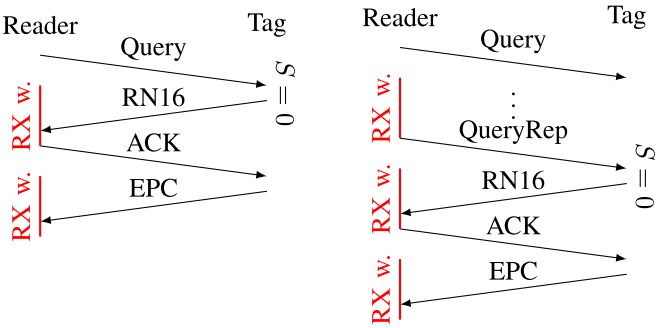

Fig. 7. (a) Single slot inventory. (b) Multiple slot inventory.

other reply. Fig. 6 presents the succession of the reader commands and tag replies described previously. Fig. 8 shows the output of the reader. Note that heavy computational tasks such as CRC checking and statistical analysis of the BLF variation are done after the inventory round. Finally, every transition in the received signal can be detected which allows to estimate all the RFID parameters including T_1, T_2, RN16, PC, L, EPC, CRC but also BLF and BLF variation.

B. Multiple Slot Inventory ($Q = 3$)

Complex inventory can also be realized by the reader. An inventory with multiple slots is now considered [see Fig.7(b)]. The query command is modified to allow 8 possible slots:

```
1000 0 00 0 00 00 0 0010 00010
```
(4)

where Q = 3. Note that CRC bits are also updated. When a tag receives this command, its slot counter is loaded with a random value in the interval [0; 7]. The tag can only enters into the reply state (and backscatter a RN16) if the slot counter is 0 otherwise, the tag remains in the arbitrate state [1, Sec. 6.3.2.6.2].

```
RFID Reader V1.0
TARI = 20, PW = 10
L0 = 20, L1 = 40
RTCAL = 60, TRCAL = 180
Tpri = 22.73 us, BLF = 44 kb/s
T1 (RN16) = 225.00 us
RN16[] = 0 1 1 0 1 1 0 0 0 0 0 0 1 0 1 0
T2 (approx) = 259.64 us
T1 (EPC) = 226.00 us
PC[] = 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0
L = 96
EPC[] = 1 1 1 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0
1 0 0 0 0 1 0 1 1 1 0 0 1 1 1 1 1 0 1 0 0 1 0 0 0 0
0 1 1 0 0 0 0 0 0 0 0 1 0 1 0 1 0 0 0 0 0 0 0 0 1 0
0 1 0 1 0 1 0 0 1 0 0 1 1 0 0 0 0 0 1 0 0 1
CRC[] = 1 1 0 0 1 1 1 0 0 0 1 1 1 0 0 1
CRC OK
Tpri0 min/avg/max = 22.62/22.76/23.44 us
Tpri1 min/avg/max = 22.44/22.78/23.19 us
Tpri min/avg/max = 22.44/22.77/23.44 us
BLF (exp) = 43.92 kb/s
```

Fig. 8. Output of the reader for a successful tag reading.

The QueryRep command [1, Sec. 6.3.2.12.2.3] decrements the slot counter by one and is simply defined as:

$$0000 \tag{5}$$

Note that this command has to be prepended by a sync-frame. When $S = 0$, the tag enters in the reply state and the remaining of the inventory is similar to the one previously described. The inventory round ends after the eighth RX window. Fig. 6(b) presents the different signals. Tag reply is visible in slot $S = 5$.

IV. Reader Characteristics and Performance

As said previously, the presented reader allows one to generate any interrogator command defined by the EPC Gen2 protocol and is able to decode any tag reply in real time. Thus, this reader can theoretically realize any inventory. Moreover, since the reader provides a full access to the timing information of the backscattered signal by the tag, a significant quantity of information is now available to the user.

BLF range supported by the reader is an important characteristic. Note first that the BLF range is defined by the standard between 40 kb/s and 640 kb/s, and that this range should supported by any tag. The Maximum BLF supported by the reader is directly proportional to the execution time of the interrupt handler presented in Fig. 5. For the Arduino platform (clocked at 16 MHz) the maximum BLF at which a tag can be successfully decoded is 55 kb/s. Note that the code has been written in C and compilation uses the default options. To increase this value, the firmware has also been written for a Nucleo64 F401RE platform (clocked at 84 MHz), the maximum BLF (with correct reading) has measured at 278 kb/s in C but improvements can still be done by carefully implementing this function to achieve a large part of the 640 kb/s. On the other side, for the minimum BLF, note that the reader can support any value, the limitation appears at the tag side. Most of the tags support BLF values lower than 40 kb/s, for example, the Monza R6-P chip can be read with a Tari of 40 µs which corresponds to a BLF of 22 kb/s.

For the memory requirements, the actual code size (which has not been optimized) represents only 26% of the available flash memory and 70% of the SRAM memory of the Arduino platform (these numbers drops at 6% and 4% respectively for the STM32F401RE) which allows significant future developments, *e.g.,* select and access commands (including write operation).

TABLE II
Bill of Materials of the Proposed Reader

Uno	Quartz	TH70085	HSMS-285B	MAX931	R/L/C	Circ.
$29	$0.57	$1.60	$0.54	$3.42	$5	$28

During the inventory, current drawn is equal to 25.8 mA for the Arduino, 13.2 mA for the transmitter and 0.15 mA for the receiver. Thus the full reader consumes 39.2 mA (195 mW) which is lower than any other UHF reader. This reader can also be easily transformed into a portable reader using a battery shield (to power the MCU and RF/analog board) and an LCD shield to display the tag EPC while keeping an important autonomy.

The performance in term of read range of this reader remains modest compared to classical readers. The main bottleneck appears in the receiver where performance is actually significantly lower than readers based on IQ demodulator and matched filter decoders. Maximum activation distance (for a Monza R6 chip) is equal to 25 cm and maximum read range for a successful decoding is 15 cm. Optimization of the receiver front-end (*e.g.,* better diodes, biased detector, multi-stages) but also adaptation of the receiver bandwidth to the BLF value can improve the read range to tend to the activation range. As shown in Section II-B, this range can be higher than 40 cm for a recent tag.

The frequency bandwidth of the reader can easily be changed to other ISM bands (2.4 GHz and 5.8 GHz) only by replacing the TH72035 chip (and the matching circuit of the envelope detector) to cover new applications and/or new protocols. The SDR architecture allows, in this case, to fully define the messages exchanged between the reader and the tag.

Finally, the total price of this reader, considering a $10 printed circuited board is $80 which is still low-cost compared to classical readers. Table II presents the detailed price information. Using a cheaper circulator or a transitioning to directional coupler should reduce the final price to about $50.

V. Conclusion

In this paper, a UHF RFID reader defined in software is presented allowing to read a UHF tag in real time. The associated hardware represents only 20 components and all tasks specific to the RFID protocol including clock recovery, data recovery and frame detection are entirely handled in software by the Arduino Uno. This reader is able to generate any RFID command supported by the protocol and to decode any message backscattered by the tag in real time. All in all, we believe that the simple low-cost open source reader presented here can become an extremely valuable research tool as well as an educational platform that can bring many talented researchers to the field of UHF RFID.

REFERENCES

[1] *EPC Radio-Frequency Identity Protocols Generation-2 UHF RFID Standard, Specification for RFID Air Interface Protocol for Communications at 860MHz–960MHz, Version 2.1*, GS1, Brussels, Belgium, 2018.

[2] "Beontag E62." 2022. [Online]. Available: www.atlasrfidstore.com/beontag-e62-rfid-paper-tag-monza-r6-p/

[3] "Yanzeo." 2022. [Online]. Available: www.yanzeo.com/rfid-write-reader.html

[4] "RAIN RFID single chip reader EPC Class1 Gen2 compatible ST25RU3993." 2022. [Online]. Available: www.st.com/en/nfc/st25ru3993.html

[5] M. Buettner and D. Wetherall, "A flexible software radio transceiver for UHF RFID experimentation," Univ. Washington, Seattle, WA, USA, Rep. UW TR UW-CSE-09-10-02, 2009.

[6] L. Catarinucci, D. De Donno, R. Colella, F. Ricciato, and L. Tarricone, "A cost-effective SDR platform for performance characterization of RFID tags," *IEEE Trans. Instrum. Meas.*, vol. 61, no. 4, pp. 903–911, Apr. 2012.

[7] G. Smietanka, S. Brato, M. Freudenberg, and J. Götze, "Implementation and extension of a GNU-radio RFID reader," *Adv. Radio Sci.*, vol. 11, pp. 107–111, Jul. 2013.

[8] F. Galler, T. Faseth, and H. Arthaber, "SDR based EPC UHF RFID reader DS-SS localization testbed," in *Proc. IEEE 16th Annu. Wireless Microw. Technol. Conf. (WAMICON)*, Cocoa Beach, FL, USA, Apr. 2015, pp. 1–4.

[9] A. J. S. Boaventura and N. B. Carvalho, "The design of a high-performance multisine RFID reader," *IEEE Trans. Microw. Theory Techn.*, vol. 65, no. 9, pp. 3389–3400, Sep. 2017.

[10] C. Huang, Y. Liu, Y. Han, and H. Min, "A new architecture of UHF RFID digital receiver for SoC implementation," in *Proc. IEEE Wireless Commun. Netw. Conf.*, Hong Kong, China, Mar. 2007, pp. 1659–1663.

[11] C. Angerer, M. Holzer, B. Knerr, and M. Rupp, "A flexible dual frequency testbed for RFID," in *Proc. 4th Int. Conf. Testbeds Res. Infrastruct. Devel. Netw. Communities*, 2008, pp. 1–6.

[12] R. Langwieser, G. Lasser, C. Angerer, and A. L. Scholtz, "A modular UHF reader frontend for a flexible RFID testbed," in *Proc. 2nd Int. EURASIP Workshop RFID Technol.*, Budapest, Hungary, Jul. 2008, pp. 1–12.

[13] M. B. A. Borisenko and M. Rostamian, "Intercepting UHF RFID signals through synchronous detection," *EURASIP J. Wireless Commun. Netw.*, vol. 214, pp. 1–10, Aug. 2013.

[14] C. Jin and S. H. Cho, "A robust baseband demodulator for ISO 18000-6C RFID reader systems," *Int. J. Distrib. Sens. Netw.*, vol. 8, no. 9, 2012, Art. no. 406710.

[15] F. Galler, T. Faseth, and H. Arthaber, "Implementation aspects of an SDR based EPC RFID reader testbed," in *Proc. Int. EURASIP Workshop RFID Technol. (EURFID)*, Rosenheim, Germany, Oct. 2015, pp. 94–97.

[16] E. A. Keehr, "A low-cost software-defined UHF RFID reader with active transmit leakage cancellation," in *Proc. IEEE Int. Conf. RFID (RFID)*, Orlando, FL, USA, 2018, pp. 1–8.

[17] E. A. Keehr and G. Lasser, "Making a low-cost software-defined UHF RFID reader," *IEEE Microw. Mag.*, vol. 22, no. 3, pp. 25–45, Mar. 2021.

[18] A. P. Sample, D. J. Yeager, P. S. Powledge, A. V. Mamishev, and J. R. Smith, "Design of an RFID-based battery-free programmable sensing platform," *IEEE Trans. Instrum. Meas.*, vol. 57, no. 11, pp. 2608–2615, Nov. 2008.

[19] S. J. Thomas, "RFID for everyone: Design of an easily-accessible, experimental UHF RFID platform," in *Proc. IEEE Int. Conf. RFID (RFID)*, Phoenix, AZ, USA, Apr. 2019, pp. 1–6.

[20] D. De Donno, L. Catarinucci, and L. Tarricone, "RAMSES: RFID augmented module for smart environmental sensing," *IEEE Trans. Instrum. Meas.*, vol. 63, no. 7, pp. 1701–1708, Jul. 2014.

[21] P. Nikitin, S. Ramamurthy, and R. Martinez, "Simple low cost UHF RFID reader," in *Proc. IEEE Int. Conf. RFID (RFID)*, Orlando, FL, USA, Apr. 2013, pp. 1–2.

[22] "SDR UHF RFID reader." 2022. [Online]. Available: https://github.com/nicolas-barbot/SDR_UHF_RFID_reader

[23] "Melexis EVB72035." Accessed: Sep. 28, 2022. [Online]. Available: www.melexis.com/-/media/files/documents/datasheets/evb72035-datasheet-melexis.pdf

[24] "Arduino UNO R3." 2022. [Online]. Available: docs.arduino.cc/hardware/uno-rev3

Nicolas Barbot (Member, IEEE) received the M.Sc. and Ph.D. degrees from the University de Limoges, France, in 2010 and 2013, respectively.

His Ph.D. work in Xlim Laboratory was focused on error-correcting codes for the optical wireless channel. He was a Postdoctoral Fellow in joint source-channel decoding with L2S Laboratory, Gif-sur-Yvette, France. Since September 2014, he has been an Assistant Professor with the Université Grenoble Alpes, Grenoble Institute of Technology, Valence, France. His scientific background at LCIS Laboratory covers wireless communications systems based on backscattering principle which include classical RFID and chipless RFID. His research interest include transponders which cannot be described by linear time-invariant systems. This gathers harmonic transponders which are based on the use of a non-linear component (Schottky diode) or linear time-variant transponders which are based on the modification of their response in the time domain. He also places special interests on antenna design and instrumentation based on these phenomenons.

Raymundo de Amorim Jr. (Member, IEEE) received the B.S. degree in electrical engineering and the M.S. degree in telecommunications and electronics from the Universidade Federal de Campina Grande, Campina Grande, Brazil, in 2016 and 2018, respectively. He is currently pursuing the Ph.D. degree with the Laboratoire de Conception et d'Intégration des Systèmes, Université Grenoble Alpes, Valence, France. From 2013 to 2018, he was a member with the Applied Electromagnetics and Microwave Laboratory (LEMA), Campina Grande. His research interests include antenna characterization, antenna array characterization, reverberant chambers, RF circuits, and chipless radio-frequency identification millimeter- and submillimeter-wave applications.

Pavel Nikitin (Senior Member, IEEE) received the Ph.D. degree in electrical engineering from Carnegie Mellon University, Pittsburgh, PA, USA, in 2002. He worked with Honeywell, Intermec, University of Washington, IBM, and Ansys. He is currently a Senior Staff Antenna Designer with Impinj, Inc., Seattle, WA, USA, where he is involved into research and development of RFID products. He is also an Affiliate Associate Professor with the Electrical Engineering Department, University of Washington. He has authored over 50 IEEE technical publications and has over 60 U.S. and European patents. He was a recipient of the 2017 and 2010 IEEE RFID Conference Best Paper Awards and the 2005 Antenna Measurement Techniques Association Symposium Best Paper Award.