RFID and Arduino: Managing RFID Events on a Real World Prototype

Nuno Correia$^{\alpha}$, Miguel L. Pardal$^{\alpha}$, Mário Romano$^{\beta}$, and José A. Marques$^{\alpha,\beta}$

$^{\alpha}$ Department of Computer Science and Engineering,
Instituto Superior Técnico, Technical University of Lisbon, Portugal
$^{\beta}$ Link Consulting - Tecnologias de Informação S.A.
Email: nuno.correia@ist.utl.pt, miguel.pardal@ist.utl.pt,
mario.romano@link.pt, jose.marques@link.pt

Abstract. Radio Frequency Identification (RFID) is a technology that is growing fast, becoming part of our daily lives more often and opening way to the Internet of Things. It can identify physical objects automatically, almost in real-time. Unfortunately, the learning curve for the technology is steep and a variety of tools are necessary just to implement a simple prototype.
This paper provides an introduction to RFID through the implementation of a prototype, covering both hardware and software. We have deployed open-source software to monitor an Arduino robot carrying tagged objects in a small “warehouse”.

Key words: RFID, EPC, Fosstrak, Software, Arduino, Robot

1 Introduction

Radio Frequency Identification (RFID) [3] can improve business processes and reduce costs in supply chains [6]. RFID tags can be attached to physical objects and then wirelessly recognized without line of sight. But, there are many difficulties in learning the technology that can be evidenced with the development of a simple prototype, as shown in previous work by Ahmed et al. [1].

The main goal of our prototype was to monitor a warehouse Arduino robot using the Fosstrak RFID middleware, providing a dashboard user interface.

A commercial RFID reader was used with two antennas, deployed in a small physical space (a table). We opted for the Arduino robot to transport the tagged objects because it is a popular open-source micro controller with several I/O pins where sensors or actuators can be plugged in, and it offers a good tradeoff between simplicity and cost [2].

We decided to use Fosstrak [4] because it is an open-source implementation of the EPCglobal Architecture [5], the most important standards available for RFID technology, providing foundations and interoperability for all the layers in the system.

2 Prototype

The prototype simulated a single warehouse door with the robot carrying the objects and passing through the door. A dashboard displays the state of the system, highlighting the object flow direction.

Figure 1 shows the *architecture* for the prototype: (i) robot, hardware readers and simulators; (ii) Fosstrak modules (Filtering & Collection Server; Capture Application; and EPC Information Services); and finally (iii) the Dashboard.

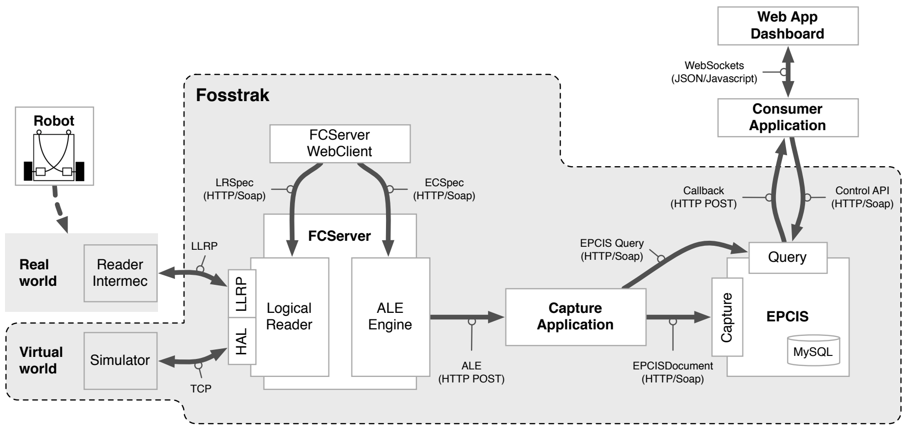
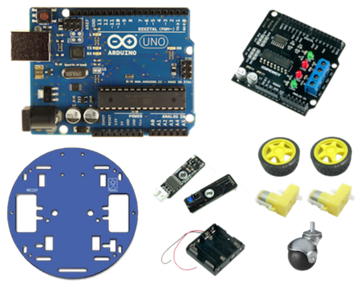


Fig. 1: Detailed prototype software architecture.

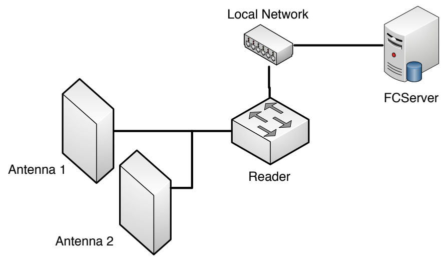
Figure 2 shows the robot *hardware kit* and how the RFID hardware parts were assembled. a) shows the hardware parts: chassis; board; motor shield; gear motors; wheels; line-tracking sensors; battery holder; batteries; and cables. b) shows all the parts and connections around the Intermec IF61[1] fixed RFID reader, connected with two IA33F Standard-Range antennas.

The prototype was developed iteratively. After the architecture overview, we installed the Fosstrak Filtering & Collection Server (fc-server), configuring logical readers and EventCycle reports with the WebClient (fc-webclient). At this stage, the system was capable of simulating tag reads and showed the EventCycle reports in the EventSink interface. The Capture Application was built to receive EventCycle reports and decide what to do with them. The initial version just dumped the captured data to the console. Next, the reports were saved into the EPCIS repository. The rules in the capture application were adjusted to match the namespace and the object flow. Discarding the simulators, the Reader and Antennas were configured and connected to the FCServer and the system started capturing actual tag reads and sending them to the FCServer. The robot was assembled and programmed to follow a black line, and was able to do it

[1] `http://www.intermec.com/products/rfidif61a/`



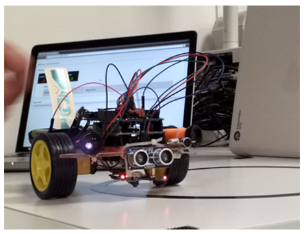
(a)



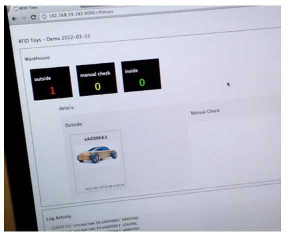
(b)

Fig. 2: (a) Prototype robot parts and (b) prototype RFID setup.

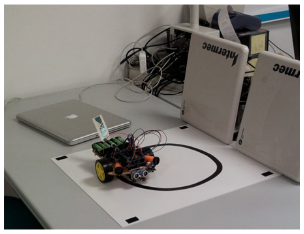
autonomously. Finally, the Dashboard was built and configured to show the relevant events. When events are received from EPCIS, the dashboard changes the state of the "warehouse" accordingly. Figure 3 shows the 3 main deliverables: the robot, the dashboard and the reader and antennas. The main objective was achieved[2].



(a)



(b)



(c)

Fig. 3: Prototype results: (a) robot; (b) dashboard; and (c) reader and antennas.

The prototype exposed several challenges that need to be addressed when building RFID applications. We found that defining the namespace of the variables in EPCIS is not easy due to the lack of conventions. Also, EPCIS uses a database for its configuration data while the FCServer configurations - LogicalReaders and EventCycle specifications - are persisted using files, making the configuration more error prone. We also found that the FCServer and the EPCIS have *stability problems* when, for example, the reader is not reachable. Also, several bugs in the EPCIS subscriptions were found[3]. EPCIS subscriptions are using *polling* instead of being triggered by asynchronous events, reducing the

[2] A video of the prototype in action is available at: `http://youtu.be/ZsG_VANtYSc`
[3] One bug was actually fixed and the patch submitted to the Fosstrak project

performance. The major problem found was the *latency* along the architecture components, because each one introduces its own delay. Managing these timer configurations holistically is hard, especially if we expect timely answers.

3 Conclusion

The real world will never be a clean stream of inputs because of all the interferences and errors. Simulators hide a lot of details like malfunctioning parts, interferences, reflections, duplicates, etc. There are also things than only happen in real world that we cannot reproduce or even anticipate. These are the main reasons we decided to implement the prototype and why real inputs were preferred over simulation inputs.

We were able to deploy an RFID system with robot carrying tags and a dashboard showing the status of the "warehouse". A better understanding of all the technologies and standards of the EPCglobal architecture was acquired.

The approach we took proved to be a good starting point for learning RFID technology. Fosstrak and Arduino are both valuable additions to the developer toolkit and provide significant benefits for prototyping applications.

3.1 Future Work

Better configuration Several Fosstrak improvements will be done, from stability problems, to improvements in the setup process to make it easier.

Better testing Data from the real world will be collected and later reproduced to help test and adapt the system to deal with RFID-specific challenges. Also, a more complete "warehouse" will be developed to test complex business rules with real world inputs.

References

1. Nova Ahmed and Umakishore Ramachandran. Reliable Framework for RFID Devices. In *the 5th Middleware doctoral symposium*, pages 1–6, New York, New York, USA, 2008. ACM Press.
2. Martin Evans, Joshua Noble, and Jordan Hochenbaum. *Arduino in Action*. Manning, June 2012.
3. K. Finkenzeller and Dorte Muller. *RFID Handbook: Fundamentals and Applications in Contactless Smart Cards, Radio Frequency Identification and Near-Field Communication*. John Wiley & Sons, 2010.
4. Christian Floerkemeier, Christof Roduner, and Matthias Lampe. RFID Application Development With the Accada Middleware Platform. *IEEE Systems Journal*, 1(2):82–94, February 2007.
5. Ken Traub, Felice Armenio, Henri Barthel, Paul Dietrich, John Duker, Christian Floerkemeier, John Garrett, Mark Harrison, Bernie Hogan, Jin Mitsugi, Josef Preishuber-Pfluegl, Oleg Ryaboy, Sanjay Sarma, KK Suen, and John Williams. The EPCglobal Architecture Framework, December 2010.
6. R Want. RFID Explained: A Primer on Radio Frequency Identification Technologies. *Synthesis Lectures on Mobile and Pervasive Computing*, 2006.