RFID: Development and Applications

Report for Final Project
ECE 493/593: Wireless Sensor Networks

Submitted by
Anthony C. Steele II
Joshua D. Tyson

Department of Electrical and Computer Engineering
The University of Alabama
December 2, 2008
Abstract

RFID technology has flourished within the last decade due to its ability to uniquely identify and track objects without line of sight. Its applications extend from supply chain management to livestock management and even to sports event timing. A MATLAB GUI is developed that integrates seamlessly with TI’s RX-MF-RNLK RFID evaluation kit. The software will provide means of demonstrating common RFID applications as well as its capabilities and limitations. Additionally, the software serves as a basis for future projects to build upon.

Research Question or Problem

Radio Frequency Identification (RFID) technology was developed over 50 years ago but has only recently received widespread attention. Its resurgence is due in large part to the requests of large retailers, such as Wal-Mart, for suppliers to use RFID tags in order to improve the efficiency of their supply chain management processes [2]. However, its applications have since expanded to include medical record and patient tracking, pet and livestock tracking, contactless payment, and many others. A demonstration illustrating RFID’s potential applications as well as its limitations is developed using TI's RX-MFR-RNLK RFID evaluation kit and MATLAB's Graphical User Interface Development Environment (GUIDE). The GUI will also provide a framework for an application programming interface (API) with the TI S4100 RFID reader.

Research Goals and Objectives

RFID technology uses radio waves to uniquely identify and track objects. Typically, RFID systems are composed of an RFID reader, multiple RFID tags, and software. An example of a common RFID architecture is shown in Figure 1 below. RFID tags can either be passive or active. Active RFID tags provide their own power, and passive tags rely on the radio frequency emitted by the RFID reader for power. The
tags also emit and respond to low frequency 134.4 KHz and high frequency 13.56 MHz signals [2]. The software, also known as middleware, gathers the hex data from the reader and interprets it into useful information for the end user.

![ RFID System Architecture Diagram ]

**Figure 1.** Most RFID systems use architectures similar to this.

**Research Design and Methods**

The MFR-RNLK RFID evaluation kit provides a suite of demo software, multiple RFID transponders, and a multi-function RFID reader. Unfortunately, the provided software is not open-source and does not yield an API for interfacing with the reader. Therefore, a graphical user interface (GUI) will be designed using MATLAB that allows the user to configure and communicate with the reader. The designed software will be modular in design, allowing for future improvements to be made. A snapshot of the GUI is shown in Figure 2 below.
Multiple passive RFID tags (transponders) are provided with the RFID evaluation kit. The tags come in many shapes and sizes. Some tags are read-only, while others are user programmable. Both low and high frequency transponders are provided. All tags adhere to the ISO 15693 standard for contactless vicinity cards. The standard defines protocols for communication between the card and its reader as well as protocols for anti-collision and data transmission [1]. Both low and high frequency tags have their advantages and disadvantages. Low frequency tags have a short read range and low data rate but work well around metal and liquid objects. On the other hand, high frequency tags are cheap and have a range of over three feet but exhibit poor performance around metal and liquid objects [2].

Accordingly, the S4100 RFID Multi-Reader is capable of detecting both low and high frequency tags. It also adheres to the ISO 15693 standard. All protocols will be strictly implemented using base protocols for the S4100 Multi-Reader [3], ISO 15693 Standard [4], Low Frequency [5], and Tag-IT™ vicinity cards [6]. The GUI will periodically check with the S4100 reader for the highest priority tag within range, and
any pertinent information detected will then be interpreted by the GUI and displayed to the user.

Results and Discussion

The S4100 reader can detect multiple RFID tags at once, but it was designed for access control applications. The reader has a programmable priority table that determines which type of tag is returned if multiple tags are detected. The possible types are ISO 14443-A, ISO 14443-B, ISO 15693, Tag-IT™, and LF listed in the default priority order. For example, if an ISO 15693 tag and a LF tag are detected, only the ISO 15693 tag will be identified and returned to the GUI. However, if two ISO 15693 tags are detected then only the tag that is detected first is returned to the GUI.

The generic packet structure for S4100 Base Application Protocol is shown in Figure 3 below. The length is equal to the number of bytes from SOF to LRC2 and the Device ID is fixed to 0x03 for the S4100 reader. The Cmd1 and Cmd2 fields select the operation to perform and the data fields are used for any necessary data for the selected operation. The final two fields LRC1 and LRC2 are used to validate the data within the packet. LRC1 is the exclusive-or of all the packet bytes beginning with the SOF, while LRC2 equals ~LRC1. The packet structure is used for communication to and from the S4100 reader. The first data byte on all packets from the S4100 is a status byte that reports errors if any occurred during the previously issued command [3].

![Figure 3. Generic Packet Structure for S4100 Base Application Protocol [3]](image)

Before instantiating the MATLAB GUI, be sure to power and connect the S4100 reader to the PC via a RS-232 cable. Once the GUI is initialized, select a COM port
using the drop-down list. Selecting a COM port opens the port and configures it for communicating with the reader. The default serial port parameters are 9600 baud rate, 8 data bits, no parity, and one stop bit [3]. The MATLAB GUI shown in Figure 2 above performs four main functions: accessing the S4100 drivers, reading RFID tags, writing RFID tags, and logging all important transactions to the user.

Among the most basic of the GUI’s features is the control over the two LEDs and a buzzer on the S4100. In most applications, the LEDs are most commonly used for error indicators, while the buzzer is generally used to indicate that a RFID tag has been found. The output devices on the S4100 are controlled using the generic packet format in Figure 3 with Cmd1 = 0x01 and Cmd2 = 0x43. The first of two data bytes controls which output using the lower 3 bits. Bits 2-0 correspond to the buzzer, LED2 and LED1, respectively. The second data byte selects what action to perform: 0x01 (On), 0x02 (Off), and 0x03 (Toggle). Figure 4 below is a snapshot of the results after LED1, LED2, and the buzzer are toggled on and off in that order.

Figure 4. Snapshot of GUI After Toggling On and Off the Drivers.
Clicking the Start button begins searching for RFID tags by turning on all RFID transmitters and issuing a FIND_TOKEN_REQUEST command every second. A FIND_TOKEN_REQUEST has Cmd1 = 0x02 and Cmd2 = 0x41 and one data field that specifies the number of times the reader loops through the priority table searching for tags [3]. If a token is found, the 64-bit identifier and type will be displayed in the RFID tag panel of the GUI as shown in Figure 5 below. If no token is found, a message is displayed indicating that an ERROR_NO_TOKEN_PRESENT status byte occurred. Clicking the Stop button will turn off all transmitters and cease sending the FIND_TOKEN_REQUEST command.

Figure 5. Snapshot of GUI Capturing A LF R/W Tag with ID=0x000000123456789.

ISO 15693 and Tag-IT™ tags are preprogrammed with a unique 64-bit identifier which cannot be changed. However, the LF tags have a programmable 64-bit identifier that can be changed as many times as desired. Consequently, once a LF tag has been detected, data can be written to it by entering a new 64-bit identifier and clicking the write button which issues a write request according to the LF protocol [4]. The LF tag must be in range of the reader during the entire write process.
**Staffing Plan**

The group consists of two students from the College of Engineering at the University of Alabama. Joshua Tyson is a graduate student in Computer Engineering. Joshua will be responsible for interfacing the wireless sensor with MATLAB and observing the results. Anthony Steele is a senior in Electrical Engineering. Anthony will be responsible for the I/O interface by way of serial communication. Group members will work together to discuss timelines, issues that arise and brainstorm on problems.

**Future Work**

Most RFID readers connect directly to the middleware via RS-232. Unfortunately, a RS-232 connection limits most systems to just one connection per computer, unless the computer has multiple serial ports. Future implementations could integrate the eZ430-RF2500 MSP430 Wireless Development Tool to wirelessly interface with the TI’s S4100 RFID Reader via the MSP430’s UART and USB port. The ability to wirelessly communicate with the middleware improves the scalability of the RFID system and allows for multiple readers to simultaneously communicate with the middleware.

**Conclusion**

RFID technology has blossomed in the last decade and a half due to its ability to identify and track objects without line of sight. However, as scientists and engineers continue to find more creative ways to apply RFID technology, issues concerning over security and privacy need to be addressed. Nevertheless, RFID technology has many non-intrusive applications such as access control and supply chain management. We have developed a simple MATLAB GUI that simulates an access control system using TI’s 4100 series of RFID readers and a multitude of RFID tags. The GUI not only demonstrates one possible application of RFID technology, but it also provides a basic API for future projects to build upon.
Reference


