International Journal of Ubiquitous Computing
Vol. 4 No. 2 (July-December, 2018)

DATA STORAGE ON A RFID TAG FOR A DISTRIBUTED SYSTEM

Sarita Pais[1] and Judith Symonds[2]

[1]Whitireia Community Polytechnic, Auckland
Sarita.pais@whitireia.ac.nz
[2]Auckland University of Technology, Auckland
Judith.symonds@aut.ac.nz

Received: 15rd April 2017 Revised: 10th May 2017 Accepted: 17th September 2017

Abstract

RFID tags can store more than just a tag ID. Data on an RFID tag can be updated through local processing. This is in contrast to the EPC global standard of data-on-network. The research study explores how much data can be stored on an RFID tag.
The scope of this study is to find a suitable data format for data stored in the tags. Two data formats viz CSV and XML along with compression techniques were discussed. The experiment conducted using the prototype examinedhow relevant data can be stored in the RFID tags and used in local processing without the need for a central database or network connectivity. The findings of the experiment results demonstrate sufficient proof of concept to suggest CSV data format. Issues encountered in the experiment are discussed, particularly related to writing data into the tag. The conclusion explores the direction for future research on improving writing data on tag, using the data-on-tag approach.

Keywords

RFID tag, data-on-tag, user memory, data format.

1 INTRODUCTION

Mark Weiser [1] envisioned a ubiquitous computing environment for the future where technology is omnipresent in the environment and 'invisible' to the user. Ubiquitous computing is also referred to as pervasive computing. Radio Frequency Identification (RFID) is one such promising technology in the field of pervasive computing [2]. RFIDs can read several tags simultaneously, store more data on the tag and data on the tag can be manipulated [3,4]. The cost of implementing RFID technology is also reducing, making realisation of Return on Investment (ROI) in commercial deployment more achievable.

RFID tags are mainly used for identification of physical objects and store an ID called an Electronic Product Code (EPC) in the tag. More details of the physical object are stored in database files that can be accessed via a centralised network. The EPC from the tag is used to retrieve more details about the physical object from the network database. This system of using data from an RFID tag in conjunction with data on the enterprise network is called as data-on-network approach.

RFID tags come with additional memory apart from that used for storing the EPC. This additional memory on the tag is of Electrically Erasable Programmable Read-Only Memory (EEPROM) type.This approach is known as a data-on-tag approach [5]. In a data-on-tag approach, data is stored on the tag, with lesser reliance on the centralised network database. Thus data-on-tag data storage systems are a type of decentralised data storage.

In such decentralised systems the stability is improved as the failure of one part of the system does not affect the system as a whole.

The application of data-on-tag approach is not popular because not enough is known about how to store data in a very small space and how to process that data with the limited capability of an RFID reader.

1.1 Research Motivation

The motivation for undertaking this research study is to determine how data can be used locally without network connectivity.

The main objective in this study is to determine a suitable data format to store data in the RFID memory for use in local processing.

Harmon [6] suggested a single string written to the RFID tag in the user memory area. The application reading this string assigned a meaning to each character position within the string. An alternate approach to storing data was undertaken by Jiang & Xiang [7] to compensate for the limited RFID memory where the application reading this data had an ontology to decode meaning from the coded data.

In order to achieve the main objective, it is important to understand how much data can be stored in a limited memory space on an RFID tag and how such data can be processed. It is also important to determine which data format is suitable to be incorporated in the limited RFID memory. Thus, the research focuses on finding a solution for the following questions:

What data format is suitable for a data-on-tag approach?
How can data be stored in the RFID tag in a data-on-tag approach?

In order to undertake this research, the literature was first considered to understand RFID technology, various data formats to store data in RFID memory to store more data in the limited memory of RFID tags. The experiment undertaken explored read and write data in the selected data format with the limited user memory of RFID tags.

This research also identifies opportunities for researchers and hardware manufacturers to design improved write procedure on an RFID tag.

2 LITERATURE REVIEW

Classification of tag by functionalities, standards and frequencies were examined from the available literature. All of these factors helped the researcher in selection of an appropriate data format in the tag to evaluate the research questions.

2.1 Tag Classification

Active tags have memory between 16 bytes and 128 KB [8]. There are passive tags which have extended memory to store additional information to a maximum of 2KB. Vendors have recently developed UHF passive RFID tags with memory of 32 KB [9]. Active tags have more memory and read from a greater distance. However the size, cost and requirement of an onboard battery are constraints to active RFID tags which is a crucial drawback in advancing their popularity. Active tags need continuous power supply to interact with the RFID reader. Active tags could be ideal for asset management in large business organisations.

The choice for the current study was for less expensive passive tags with some memory in it. UHF tags operate at 433 MHz or 865 – 956 MHz and in long ranges as many as 100 tags can be concurrently read by a reader. As these tags operate at a frequency more than 900 MHz, it improves the read rate as data from the tag is read in a shorter time interval avoiding collision with data from other tags [10].

2.2 RFID Standards

As RFID applications gained commercial popularity, it was necessary to have some standards in place. This issue became visible when Wal-Mart, one of the foremost players in RFID, required that all goods in their supply chain be tagged using RFID at pallet and case levels [11]. In the case of Wal-Mart, they took the initiative and asked all their trading partners to follow EPC Gen 2 standard. The Auto-ID Centre in MIT came up with the Electronic Product Code (EPC) to uniquely identify each RFID tag.

Two major organisations EPC Global (AutoID centre) and International Standard Organisations (ISO) have developed an international standard [12]. This research experiment has used tags that comply with the ISO 1800-6B and ISO 18000-6C standards.The existing anti-collision protocols were used without modification in this research experiment.

2.3 Data-on-tag

The concept of data-on-tag is not new. It has been used in production control and maintenance documentation in manufacturing, tracking patients and medical equipment in healthcare. In one example, additional information for garments was stored in the tag [13]. The additional information was used to convey garment details along different stages of the supply chain. As these products travel long distances, accessing network details at different distribution centres is not possible as there is no central network connection at all times. In another example, in order to avoid bottlenecks at the central database, Ford manufacturing company stored customised details of the production process in active tags. The production process is not disrupted even if there is a network failure. Another advantage is keeping an update of maintenance documentation as in Frankfurt airport's fire shutters scattered at different locations. The maintenance person reads the maintenance instructions from the tag using a portable reader and updates with the current date and time. In this way, complete maintenance records can be kept and quality assurance targets can be met even where there is no central network access.

The data-on-tag approach can have applications similar to smart card technology where extra information is stored with the physical item. Chan, Cao, Chan, & Young [14] proposed embedded processing inside a smart card. The smart card could carry patient information accessible by any hospital and did not require any sophisticated computer network to access the data from the backend database.

In order to overcome the limited memory in RFID tag, Romer, Schoch, &Mattern[15] proposed storing an internet address (URL) in the tag through which the detail of the physical item could be looked up in the database server and not stored in the tag. No access to proprietary network was needed. The application was designed to access the tag's virtual counterpart on the internet. However, it still needed an internet connection and application running on the web server in order to lookup data about the physical item.

The concept of data-on-tag can be implemented in a distributed way by having data in RFID tags explained in case of super distributed database and tuple-based distributed data which are discussed in the next section.

2.3.1 Super Distributed Database

According to Sugawara, Yamaoka & Sakai [16] a super distributed database is a network connecting many personal databases (PDBs). The data in these PDBs are not centrally controlled and can be edited locally. A super distributed database is created by RFID tags scattered in the environment as 'super-distributed' smart objects [17]. The intention of the current research study is to explore potential ways to carry data in individual RFID tags not necessarily be part of a network. Data is used in a contextual manner when required and thus locally processed.

2.3.2 Tuple-based Distributed Data

Mamei, Quaglieri, & Franco Zambonelli[18] used RFID tag memory to realise a system that worked through data distributed in the environment which they called the tuple-based distributed data approach. They used RFID tags to represent physical objects in the environment with the capability of storing digital data in them. Each data piece about the physical object was an attribute and all the attributes together were treated as a single tuple, coded as a simple data format in 512 bit memory of the RFID tag. For example a tuple with values ("flower", "red") was coded as (01, 14). The programming interface picked up the coded value and matched it with pre-defined values in the application to create more meaningful information from the data in the tag. The pre-defined values were stored in an application dependent ontology. This approach overcame the restricted memory of RFID tags and was an example of data format where a type entity could be used.

The data-on-network approach needs a secure network setup with middleware to filter data from the RFID reader as it reads the tags and before it is stored in the enterprise database. The data-on-tag approach on the other hand, is achieved locally by reading RFID tags from reader and storing data in a local database for accessing in computer applications. Furthermore, the data-on-tag approach does not depend on Savant or any other proprietary middleware. It is a decentralised way of managing data and continues to work even in case of a network failure.

With the data-on-network approach, it is only necessary to read the EPCs stored in the tag. In a data-on-tag approach the read and write process on the additional data in the tag occurs after the EPC is read.

2.4 User Memory

In addition to the EPC tag ID stored in an RFID tag, most tags are built with CMOS integrated circuits with EEPROM memory [19] which facilitates the reading and writing of additional data in the tag. This helps store additional information about an object on the object itself and does not depend on the network database for more information. The tags for UHF standard have four memory banks as shown in Table 1 [20, 21]. In order to read the user memory, it is important to understand the memory structure and how data is stored in the different banks.

Table 1: UHF Tag Memory Bank

Source: [6]

Bank	Memory Definition
Bank 00	Reserved
Bank 01	EPC
Bank 10	TID
Bank 11	User

The user bank is a memory space where the user can read and write data.

To write data on the tag, the specific memory address of the bank and the block within this bank needs to be identified. All the data to be written to the tag is considered as a single data string and written entirely within the tag's user memory location. It can then be read by an application program which retrieves the different pieces of information from that data string.
There is no standard to date to store data other than the EPC tag ID on the tag. Tribowski et al. [22] recommended *ISO 13584* standard for parts libraries (PLIB), where data attributes with description, data type and unit needs to be defined. XML schema was a possible format.

Data can be in Extensible Markup Language (XML) or Comma Separated Values (CSV) formats. The storage of data in the appropriate format and previous studies on instances where these data formats are used are discussed in next paragraph.

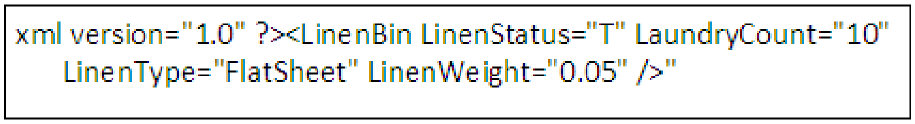
2.5 Data Format

Two possible data formats are considered – XML and CSV. The focus of the literature review is to explore whether these formats are suitable to read from and write to an RFID tag given its limited user memory.

2.5.1 XML Format

Willis and Helal[23] used HF RFID tags with a memory capacity of 2000 bits in their experiment. XML data of 869 bytes was compressed sufficiently enough to be stored in 250 bytes using Chemical Markup Language (CML) and Huffman compression techniques.

In the current study, XML can be used to store a single record of linen in a tag. A simple XML file describing four attributes needs 107 characters which can be stored as 107 bytes as shown in Figure 1.



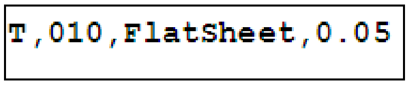
```
xml version="1.0" ?><LinenBin LinenStatus="T" LaundryCount="10"
    LinenType="FlatSheet" LinenWeight="0.05" />"
```

Figure 1: Example Linen Data in XML Format

2.5.2 CSV format

CSV is also a simple text format to store data values separated by commas. It is a very old technique from the earliest personal computer era.

The CSV data as shown in Figure2 has 20 characters. Data in different linen types (PatientGown, blanket) may vary with a possible upper limit of 30 characters, which should be sufficient in all cases to be stored in the limited RFID memory.



```
T,010,FlatSheet,0.05
```

Figure 2: Example Linen Data in CSV Format

This comparison of XML and CSV data format shows that XML takes 107 bytes whereas CSV takes 20 bytes to represent the same data. Clearly CSV format takes much fewer bytes to represent the same data when compared to XML.

3 Research Methodology

The research approach and process for the current research study was design science. The guidelines as proposed by Hevner et al. [24] were considered.

In this research study the artefact was developed as a proof of concept to read and write necessary data on tag with a suitable data format for the data-on-tag approach.

Although it is theoretically possible to store long data strings in user memory with an XML data format, practically, long strings can be written in piecemeal over several memory blocks. However, writing in pieces in the user memory of the tag is not suitable as multiple tags are involved and there is no guarantee of the reader reading and writing the same tag for the complete write procedure.

In the current study, data in the tag is stored as a single string with comma delimitation to store each of the fields necessary to include all the required data. This is to work around the limited user memory available in the tag.

4 Experiment Design

The RFID system used in the experiment were one UHF RFID reader and a SDK from Tracient Technologies, two types of tags from Skyetek and Intermec.

Skyetek RFID tag complies with EPC Gen 2 / 18000-6B standard and operates within the 860 MHz to 960 MHz frequency range. Its dimensions are 12mm x 180 mm. The user memory size is 1728 bit which translates to a storage capacity of 216 bytes.

IT67 UHF Intermec tag complies with EPC Gen 2 / 18000-6C standard and operates within the 860 MHz to 960 MHz frequency range, similar to Skyetek tags. Its dimensions are 72.3 mm x 94.5 mm. The IT67 Intermec tag has a 512 bit user memory chip.

Both Skyetek and Intermec tags operate at the UHF frequency and are compatible with the Tracient reader. However the setting in the Tracient application is different for the two tags as they are of different standards. Skyetek tags are of EPC Gen 2 18000 6B and Intermec IT67 tags are of EPC Gen 2 18000 6C standard.

The read and write functions defined in the Tracient RFID SDK were used to read and write data in the user memory as a string following the formats of CSV or XML.

5 DISCUSSION

Existing compression techniques available through SDK were explored. The two data formats were written as a string and the viability of write procedure was studied.

5.1 Compression Techniques

In order to investigate the performance of various compression techniques on small files, C# library compression techniques like Gzip and Deflate were undertaken (as the prototypes developed for the experiment were in C# SDK provided by Tracient). Gzip and Deflate compressions techniques use industry standard algorithms for lossless compression. Both internally use Huffman and LZ77 coding (Microsoft, 2008). Gzip uses the same algorithm as Deflate and has other compression formats. Thus the size of Gzip compressed data is slightly larger than data compressed by Deflate. This was evident through compression trials with data of various sizes ranging from a few bytes to 1KB as demonstrated in Table 2.

Table2: Comparison of Data Compression through Gzip and Deflate

Original Data Size	Gzip Compression	Deflate Compression
26 bytes	140 bytes	122 bytes
102 bytes	193 bytes	175 bytes
204 bytes	197 bytes	179 bytes
408 bytes	199 bytes	181 bytes
816 bytes	202 bytes	184 bytes
998 bytes	206 bytes	188 bytes
1996 bytes	222 bytes	204 bytes

From the examples in Table 2, it was evident that compression techniques available in C# successfully compressed data by 50% or more when the original data size was 408 bytes or more. It was further seen that compressing a small data file of 26 bytes actually increased the resulting file size enormously. The resultant size of the file compressed by Deflate was marginally less than that compressed by Gzip in all cases.

Further to the results from Table 2, an XML data that described four attributes with a data size of 127 bytes (as described in Figure 1for a linen data) was compressed. Deflate actually increased data size to 208 bytes. String compression techniques available in C# were not appropriate to compress a small string as Huffman coding is used internally and requires the 256 ASCII characters to build the compressed string. It was not possible to compress small strings through these techniques because an overhead was created to code each of the 256 ASCII characters regardless of the size of the string. When the string is small, the overhead makes up a larger proportion of the file overall.

5.2 Data Format Suitability

XML and CSV data formats were examined for their suitability in the current study.

5.2.1 XML Format

An XML format would be suitable for a RFID tag with user memory of 1 KB or more. However experiments with the Tracient reader and Skyetek tags did not succeed in storing 107 bytes (as required for linen data in Figure 1) by writing as a continuous string. Although Skyetek had 200 bytes, it could successfully write only 64 bytes of data at any one time. To write more bytes, the block address of the user memory had to be moved further to continue writing from where it had last written. Thus long strings were written in parts in the user memory. Compression techniques were not suitable for small sized data.

5.2.2 CSV Format

Due to limited memory in the available RFID tags, the data format was kept as a string of characters, which has simple format for the different pieces of data attributes delimited by comma. A sample CSV linen data as described in Figure 2, was below the maximum 64 bytes of data which can be written as whole string. The data string was hard coded and written onto the user memory at the starting block set through the prototype and the entire data format was written through one command and read from its memory location. The application to retrieve meaningful data attributes from the data format was done through the prototype. The comma in the CSV format was useful to separate values into different attributes.

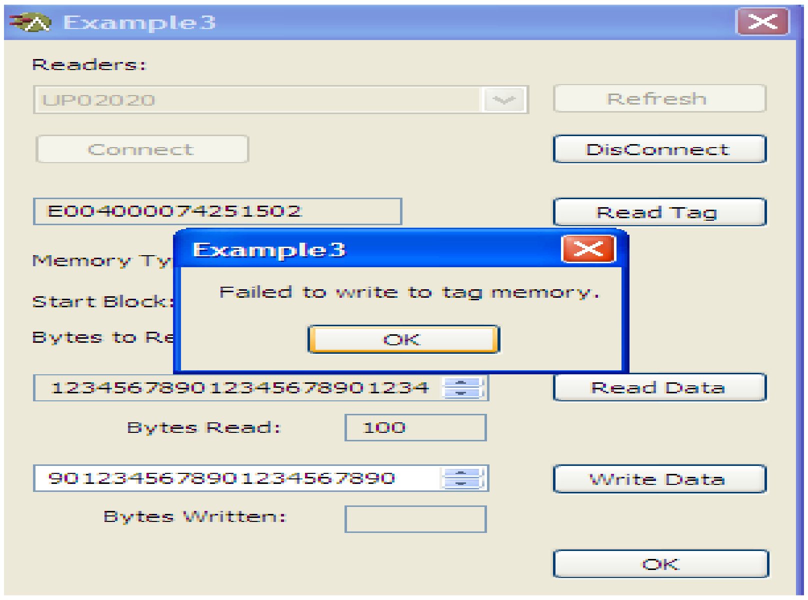
Therefore, considering XML and CSV, the only acceptable option was to store data in a CSV format.

5.2.3 Coded Data

Coded data in the RFID tag was not used in the current study since the application reading the coded data was not self explanatory and had the overhead of searching the meaning for it.

5.3 Limited Write Capability

The application provided by Tracient failed to write more than 64 bytes at a time. Figure 3 shows the error output caused by attempting to write more than 64 bytes to the tag. This application was modified in the prototype to run the experiment. To understand if this was a hardware or software problem, the following tests were done.


Example3

Readers: UP02020

E004000074251502

Example3

Failed to write to tag memory.

Bytes Read: 100

1234567890123456789012 34

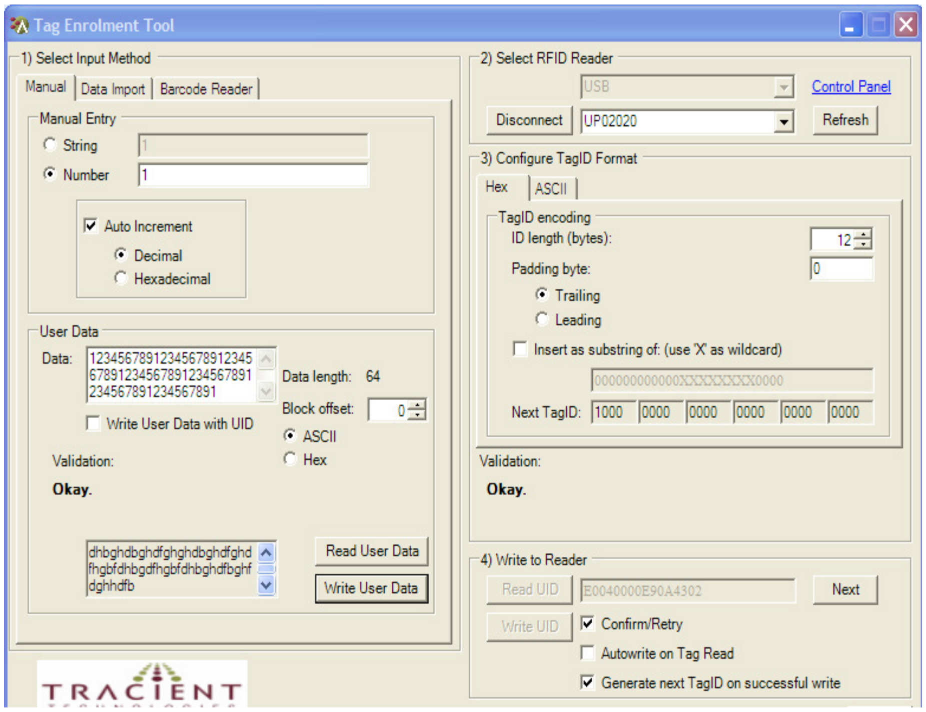
901234567890 1234567890

Bytes Written:

Figure 3: Tracient Program Example3 - Attempt to write 100 bytes

Source: Tracient Software

Another application by Tracient called Enrolment Tool was used which could write 64 bytes at a time as shown in Figure 4. However, it failed to write more than 64 bytes as evident in Figure 5. To write data more than 64 bytes, the block offset had to be increased by 64 bytes to write long strings in parts of 64 bytes at a time. Thus the application could overcome the problem of writing more than 64 bytes.


Tag Enrolment Tool

1) Select Input Method

Manual | Data Import | Barcode Reader

Manual Entry

String [ ] 1
[✓ Number] 1

[✓] Auto Increment
[✓ Decimal] Hexadecimal

User Data

Data: 1234567891234567891234567891234567891234567891234567891234567891

Data length: 64

Block offset: 0

[ ] Write User Data with UID

[✓ ASCII] Hex

Validation:

Okay.

dhbghdbghdfghghdbghdfghdfhgbfdhbgdfhgbfdhbghdfbghfdghhdfb

Read User Data | Write User Data

2) Select RFID Reader

USB | Control Panel

Disconnect | UP02020 | Refresh

3) Configure TagID Format

Hex | ASCII

TagID encoding

ID length (bytes): 12

Padding byte: 0

[✓ Trailing] Leading

[ ] Insert as substring of: (use 'X' as wildcard) 000000000000XXXXXXXX0000

Next TagID: 1000 0000 0000 0000 0000 0000

Validation:

Okay.

4) Write to Reader

Read UID | E0040000E90A4302 | Next

Write UID

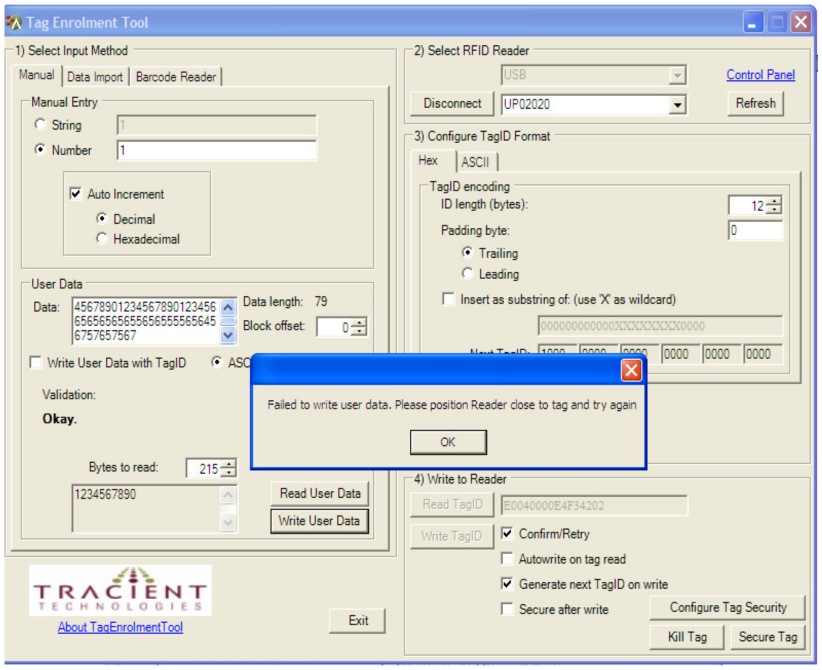
[✓] Confirm/Retry

[ ] Autowrite on Tag Read

[✓] Generate next TagID on successful write

Figure 4: Tracient Tag Enrolment Tool - Attempt to Write 64 bytes

Source: Tracient Software



Number: 1
☑ Auto Increment
● Decimal
○ Hexadecimal
User Data
Data: 4567890123456789012345665656565655656565555656456757657567
Data length: 79
Block offset: 0
☐ Write User Data with TagID
Validation: **Okay.**
Bytes to read: 215
1234567890
USB
UP02020
ID length (bytes): 12
Padding byte: 0
● Trailing
○ Leading
☐ Insert as substring of: (use 'X' as wildcard)
000000000000XXXXXXXXX0000
Next TagID: 1000 0000 0000 0000 0000 0000
Failed to write user data. Please position Reader close to tag and try again
E0040000E4F34202
☑ Confirm/Retry
☐ Autowrite on tag read
☑ Generate next TagID on write
☐ Secure after write

Figure 5: Tracient Tag Enrolment Tool - Attempt to Write 79 bytes

Source: Tracient Software

The Tracient application called Enrolment tool can control parameters to write 215 bytes. It requires data to be split and written in chunks of 64 bytes at a time in different block addresses. The energy is not sufficient enough to write all the data in the tag user memory in one go [25]. The application will have to split the data into a maximum of 64 bytes and written with the memory block offset moved to write in the appropriate position.

6 Conclusions

The current research study explored the applications using data-on-tag approach. The theoretical concept of data-on-tag approach is impressive. However the viability of data-on-tag approach is dependable on the RFID technology.

In the data-on-tag approach, CSV format appeared as the most relevant data format. CSV format stored the data in less bytes than XML. The write procedure through the COTS software was able to write only 64 bytes at a time which suited CSV format.

Experiment results using CSV format demonstrated sufficient proof of concept. Accuracy of the results was as expected for reading. However writing proved more problematic.

6.1 Direction for the Future Study

The focus of this study was to achieve a basic understanding of the data-on-tag approach and how more information can be stored in a tag. The information was useful in a distributed application by reading and writing data in a tag which can be applied where there is no access to a network database and application.

Acknowledgement

I appreciate the help provided by Tracient Technologies Ltd., Christchurch and Intermec Technologies in providing me the RFID reader, tags and SDK to carry out the experiment.

References

[1] Weiser, M. (1993). "Hot topics - Ubiquitous computing".*Computer 26*, 71-72.

[2] Matsuoka, K. ,Katou,N., Dejima, S. &Takami, K. (2010). Information selection and delivery algorithm for delivering advertisements suitable for the pedestrians present at a particular site, International Journal of UbiComp (IJU), 1(4), 13-21.

[3] Haas, L. M., & Miller, R. J. (1997). "Transforming heterogeneous data with database middleware: Beyond integration".*Bulletin of the IEEE Computer Society Technical Committee on Data engineering*, 1-6.

[4] Hardgrave, B. C., Armstrong, D. J., &Riemenschneider, C. K. (2007). "RFID assimilation hierarchy".*Proceedings of the 40th Hawaii International conference on System Sciences, Hawaii*, 1-10.

[5] Diekmann, T., Melski, A., & Schumann, M. (2007). "Data-on-network vs. Data-on-tag: Managing data in complex RFID environments". *Proceedings of the 40th Annual Hawaii International Conference on System Sciences, Hawaii*, 224-233.

[6] Harmon, C. K. (2006)."The necessity for a uniform organisation of user memory in RFID".*International Journal Radio Frequency Identification Technology and Applications, 1*(1), 41-51.

[7] Jiang, W., & Xiang, D. (2008). "A compression framework for personal image used in mobile RFID system". *9th International Conference for Young Scientists, Zhang JiaJie, China*, 769-774.

[8] Ward, M., Kraneneburg, R., & Backhouse, G. (2006). "RFID: Frequency, standards, adoption and innovation". *JISC Technology and standards Watch*, 1-36. Retrieved from http://www.rfidconsultation.eu/docs/ficheiros/TSW0602.pdf

[9] Bacheldor, B. (2009). "*Tego Launches 32-Kilobyte EPC RFID Tag"*. Retrieved 3-3-2010, from http://www.rfidjournal.com/article/view/4578

[10] Want, R. (2004). "The magic of RFID".*Queue, 2*(7), 40-48.

[11] *Wal-Mart spells out RFID vision, RFID Journal 2003*. Retrieved 8-09-2010, from http://www.rfidjournal.com/article/purchase/463

[12] Wu, N. C., Nystrom, M. A., Lin, T. R., & Yu, H. C. (2006)."Challenges to global RFID adoption".*Technovation, 26*(12), 1317-1323.

[13] Melski, A., Thoroe, L., Caus, T., & Schumann, M. (2007)." Beyond EPC – Insights from multiple RFID case studies on the storage of additional data on tag". *International Conference on wireless Algorithms, Systems and Applications, Chicago*, 281- 286.

[14] Chan, A. T. S., Cao, J., Chan, H., & Young, G. (2001). “A web-enabled framework for smart card application in health services”.*Communications of the ACM, 44*(9), 77-82.

[15] Romer, K., Schoch, T., &Mattern, F. (2004). “Smart identification frameworks for ubiquitous computing applications”.*Wireless Networks, 10*, 689-700.

[16] Sugawara, K., Yamaoka, K., & Sakai, Y. (1997). “A study on image searching method in super distributed database”.*IEEE Global Telecommunications Conference, Phoenix, AZ, USA, 2*, 736-740.

[17] Bohn,J. (2008). “Prototypical implementation of location-aware services based on a middleware architecture for super-distributed RFID tag infrastructures”, Personal and ubiquitous computing, 12 (2), 155- 166.

[18] Mamei, M., Quaglieri, R., & Franco Zambonelli, F. (2006). “Making tuple spaces physical with RFID tags”.*Proceedings of the 2006 ACM symposium on Applied computing, Dijon, France*, 434 - 439.

[19] Landt, J. (2005). “The history of RFID”.*Potentials IEEE, 24*(4), 8 – 11.

[20] Lin, D., Elmongui, H. G., Bertino, E., &Ooi, B. C. (2007). “Data management in RFID applications”.*DEXA*, 434-444.

[21] Tracient. (2007). *Tracient user manual*.from www.tracient.com

[22] Tribowski, C., Spin, K., Guenther, O., and Sielemann, O.,(2009) "Storing data on RFID tags: A standards-basedapproach". *ECIS 2009 Proceedings.*http://aisel.aisnet.org/ecis2009/146

[23] Willis, S., &Helal , S. (2005).”*RFID information grid for blind navigation and wayfinding”*. Paper presented at the Proceedings of the ninth annual IEEE International Symposium on Wearable Computers, Osaka, Japan. from http://www.icta.ufl.edu/projects/publications/willis-RFID-ISWC%20v2.pdf

[24] Hevner, A. R., March, S. T., Park, J., & Ram, S. (2004).” Design science in information systems research”. *MIS Quarterly, 28*(1), 75-105.

[25] Floerkemeier, C., & Lampe, M. (2005). “RFID middleware design - addressing application requirements and RFID constraints”.*Joint sOc-EUSAI conference, Grenoble*, 1-6.

Authors

Sarita Pais completed her Masters in Computer and Information Sciences At Auckland University of Technology, Auckland. This paper is part of The thesis study for the award of the Master's degree. The author has 20 years of experience in the IT field. She is currently teaching Bachelor of Information Technology at Whitireia Community Polytechnic, Auckland.She supervises capstone projects in web applications for undergraduate students.

Dr Judith Symonds is a Senior Lecturer at the Auckland University of Technology. She gained her PhD in Agribusiness in 2006 through the University of Queensland. In her doctoral work she studied the issues and challenges with eBusiness systems for rural SMEs and since gradu -ating, she has continued to research in areas of technology infusion of RFID, predominantly in the health industry. For the past four years Dr Symonds has served as National Committee Member of the NZ RFID Pathfinder Group, an industry steering body.She has also served on Several journal review boards and conference organising committees.Dr Symonds teaches postgraduate courses in ICT issues for SMEs, Entrepreneurship and she also supervises degree capstone projects in web development.