RFID Asset Tracking and Digital Check-Out System

Migliore, Jeff

CPE Senior Project

June 14, 2010

Abstract

An asset tracking and digital check-out system was developed using Radio Frequency Identification (RFID) and a Single Board Computer (SBC). A RFID reader was used as a check point to process transactions. The RFID reader was attached to the SBC via USB. The SBC is integrated into a network via Ethernet and hosts a web graphical user interface (GUI) frontend for users to view transactions and query availability of assets. A MYSQL database was used for the backend storage of data.

Specifically, this system was designed to track tool assets. Many modern companies have employees sharing limited resources. The common method of tracking is done with a paper form system. This process was digitized with my project. RFID allows a fast and smooth transaction for employees to get right to work. The easily accessible GUI allows employees to track down the specific tool needed for their task if it is missing.

Technologies used:
RFID, USB, Ethernet, ARM9, Linux, Apache, MYSQL, PHP, AJAX, HTML

Table of Contents

Introduction

For my senior project, I developed a radio frequency identification (RFID) asset tracking and digital check-out system with a database backend and web graphical user interface (GUI) frontend. Specifically, this system was designed to track tool assets. I am a Computer Engineering student attending California Polytechnic State University. I approached the project with combined efforts in multiple specialized knowledge areas I acquired while attending Cal Poly.

This correspondence outlines my project, its design, and implementation.

Overview

My project was accomplished by using an embedded device to host the location of check-out/check-in process. The tools are tracked by means of RFID smart tags. Users are identified by means of RFID access tags (such as an employee badge). The embedded device will maintain a log. The log will contain information such as who took the tool, when the tool was taken, and how many of each tool is available. The embedded device also has an Ethernet port located on the outside of its enclosure for web GUI access. The web GUI will be viewed by means of a web browser and made possible by software services installed on the embedded device. The software services installed are: Linux, Apache Server, MYSQL, and PHP (also known as the LAMP software bundle). The following images will help illustrate the project.

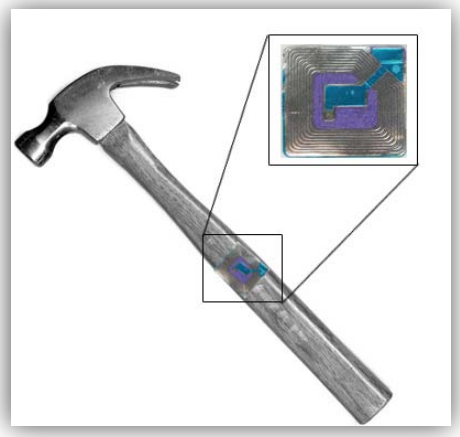


Figure 1: RFID Tag on a Hammer. With an embedded RFID tag on every tool, the system will be able to acknowledge the exact tool being taken. This will allow end users to enjoy faster, seamless asset management without filling out paper forms.

Figure 2: RFID Tag embedded within a badge. Most modern companies have their employees wear badges to identify them and for gaining access to certain areas. I adopted this paradigm for identification. Thin flexible RFID tags have been developed using film transistor technology to integrate an RFID on a flexible substrate.

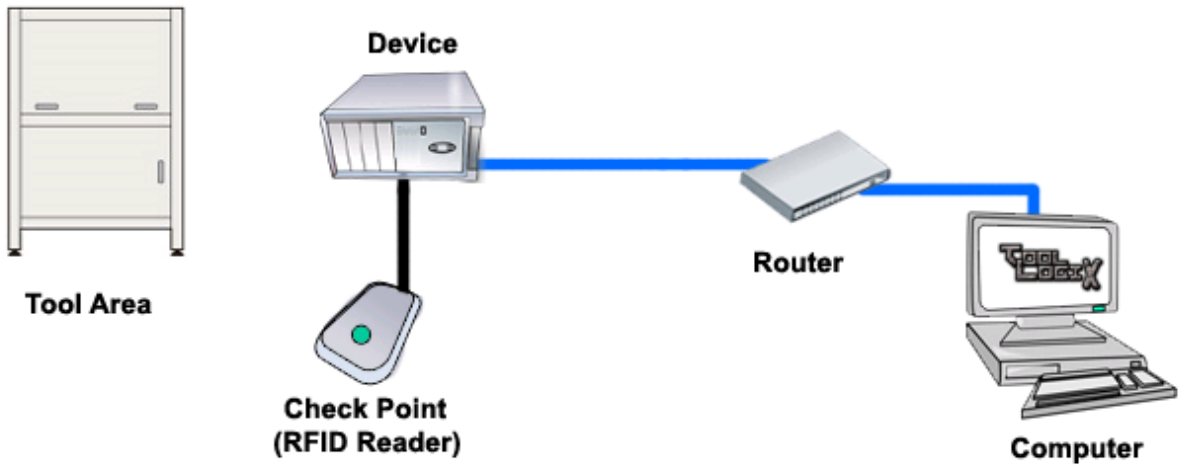


Figure 3: System Diagram. The figure above shows the system and connection setup. The device will be setup conveniently close to the tool area. It will host the database, process transactions, and host the web GUI. Employees will use the RFID reader to process their transactions. The web GUI will then be accessed anywhere in the company through a web browser.

Reasons for Development

My project is solving a problem prevalent in many companies. Namely, these companies have many employees sharing limited tool assets. Companies that use a form of logging default to a form of a sign-in/sign-out paper sheet. This method appeared outdated to me. Digitizing this process has many benefits. Firstly, the company will be more "Green" by saving paper. Secondly, they will save physical space by having years of logging information on a compact device. Time will be saved by using a simple interface rather than sifting through mounts of paper. It will be quick and easy to determine if a tool is checked-out, and if so, who has taken it.

Many companies have limited tools due to their expensive nature. If an employee were to take a tool and forget to return it, he/she may be preventing another employee from completing their task. This scenario leads to a loss of company time and money. My project will solve this issue.

Use Cases

Listed below are a couple of detailed use cases and diagrams. These are only a sample of the large amount of actual use cases possible in the workplace.

Use Case 1

1. Employee A is tasked for a job. He needs a tool in order to complete his task.
2. Employee A goes to the tool area and finds the tool.
3. Employee A scans the needed tool and his badge.
4. Employee A finishes his task and submits his work. He returns the tool by once again scanning his badge and the tool.

This is the most basic and common use case. The employee is simply using the device as designed to check out and check in the tool he had used for logging purposes. He did not have to stop to fill out a form nor did he run into any problems retrieving the tool.

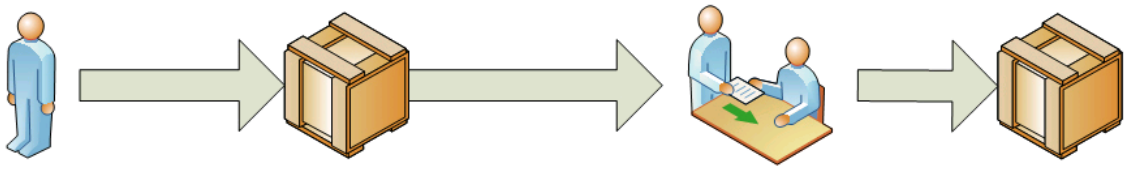


Figure 4: Use Case 1. The employee uses the system to check out the tool and complete his task.

Use Case 2

1. Employee A is tasked for a job. He needs a tool in order to complete his task.
2. Employee A goes to the tool area and finds the tool.
3. Employee A scans the needed tool and his badge.
4. Employee A finishes his task but leaves the tool at his work area.
5. Employee B is tasked for a job. He needs the same tool as Employee A to complete his task.
6. Employee B goes to the tool area. Unfortunately, Employee B does not find the tool because Employee A never returned it.
7. Employee B goes to the computer and checks the log. He sees Employee A has the tool.
8. Employee B contacts Employee A and Employee B is able to start his task.

In the above scenario, Employee A delayed Employee B from doing his job because he did not return the tool after completing his task. With the asset tracking system, Employee B was able to quickly find where the tool was located to limit his downtime. Both employees also saved time with the system by not having to fill out forms.

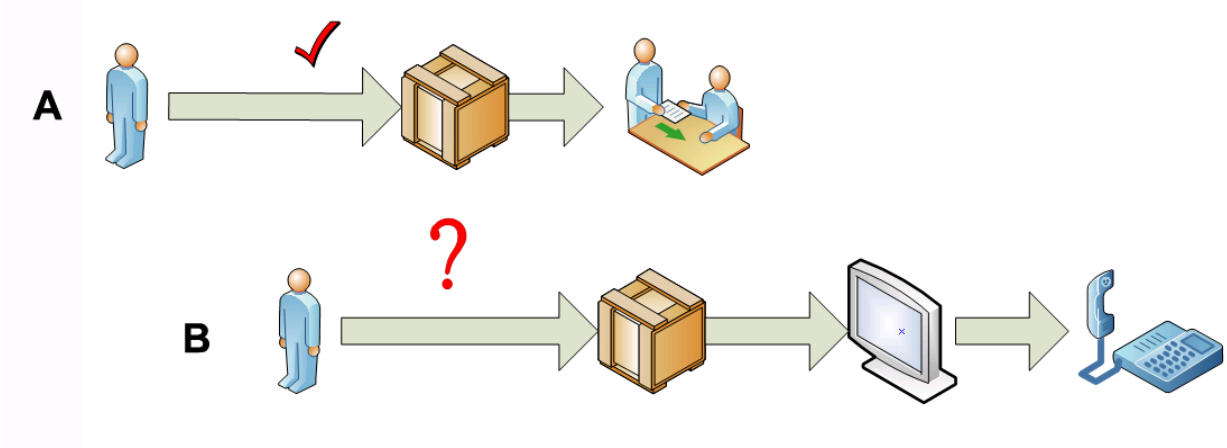


Figure 5: Use Case 2. Employee A was able to find the tool he needed for his task and submitted his work. Employee B could not find the tool in the tool area. Employee B went to the computer, noticed employee A had it last, and contacted him in order to retrieve the tool.

Use Case 3

1. Employee A and B are tasked for a job at the same time that require the same tool.
2. Employee A communicates with Employee B that he will take the tool first and then hand it to him after.
3. Employee A goes to the tool area and finds the tool.
4. Employee A scans the needed tool and his badge.
5. Employee A finishes his task and notifies Employee B and delivers the tool.
6. Employee B finishes his task and returns the tool.

In the above scenario, Employee A and B were tasked for a job at the same time. With proper communication they were able to get their jobs done. The system allows anyone to return a tool and logs who returned it. If Employee B were to forget to return it, and say Employee C needed the tool, he would contact Employee A because of the log stating Employee A checked it out. Employee A will then redirect Employee C to Employee B. Thus, the system enables the tool to be tracked down.

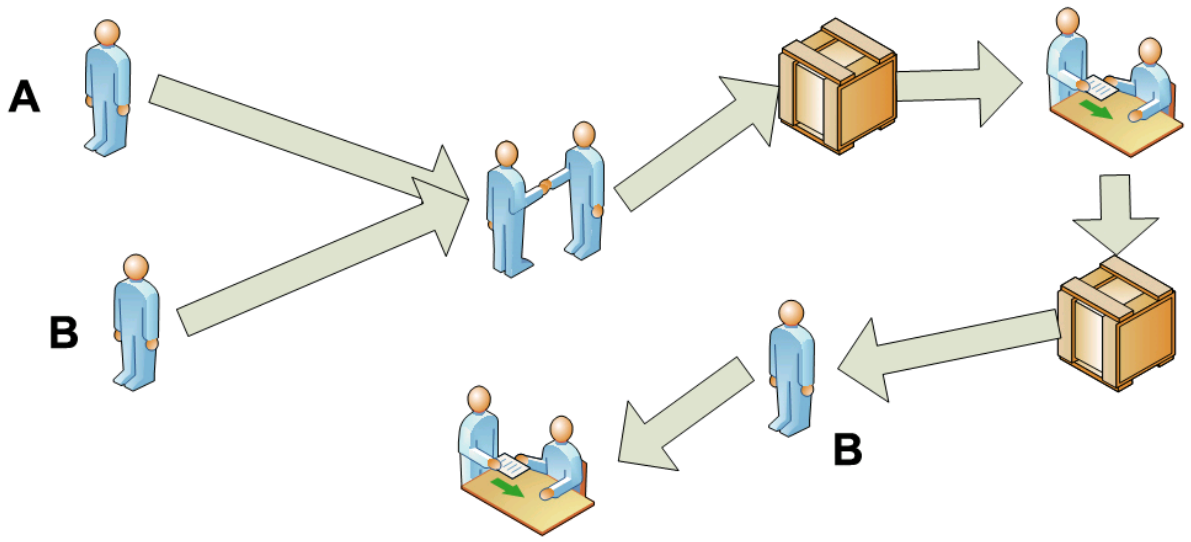


Figure 6: Use Case 3. Both employees agreed upon a work flow. Employee A completed his work first and delivered the tool directly to employee B.

Implementation

In order to implement my project, I first evaluated the requirements. With the requirements set, I could then begin looking for solutions with hardware and software combinations. Lastly, I could design the system specifically for my requirements.

Requirements

The system as a whole must digitize the check-out/check-in process currently done with paper forms. Listed below are some general requirements:

- The solution must be user-friendly and easy to deploy and use.
- The system must also be practical in terms of cost and processes.
- The system will need to be able to identify users and tools. This method should be cheap and effective.
- The check-out/check-in point must be easily identifiable and located.
- The system hosting the web services must be robust and efficient. The user GUI must also be user-friendly.
- The system should be able to sustain for long periods of time (over a month) without rebooting.

Hardware Solution

After researching through the internet and speaking to valuable resources on campus, I was able to generate a solution. The research began with hardware that could run Linux. My software choice for Linux was based on its stability, coding environment, and being free open-source software. I looked into a few potential boards including Beagle Boards™, Phidget SBC™, and the TS-7350™. These boards are classified as a Single Board computer (SBC).

SBCs are complete computers on a single board as the name describes. They have the basic necessities to be a fully operational computer (processor, RAM, I/O, memory). A SBC is perfect for my project since it will be compact and fully functional for hosting my website, database, and processing transactions. To evaluate these boards I created a table of advantages and disadvantages. This led me to pick the TS-7350 from Technologic Systems™. The tables I used are located below.

Table 1: SBC Comparison.

	Cost	CPU	Size	Linux Installation	Memory	USB	Ethernet	Serial RS-232
Beagle Board	$150	600MHz	3" x 3"	Hard	Expandable SD slot	Yes	No	Yes
Phidget SBC 1070	$265	266MHz	3" x 5"	Easy	64MB NAND	Yes	Yes	No
TS-7350	$175	200MHz	3" x 4"	Easy	Expandable SD slot	Yes	Yes	Yes

This evaluation compared three potential board solutions. The attributes I compared included general things such as price, CPU speed, and peripherals. Some specific attributes to my project included if the SBC was Linux compatible, had Ethernet ports, and was compact in size. To increase development time, I wanted to limit the amount of time it took to port Linux to the SBC. Therefore, I looked for documentation and procedures for installing Linux. I also wanted to ensure proper compiling chains for the native system was readily available.

Beagle Board

The Beagle Board is a very cheap board with great performance. It is also very compact. Unfortunately, installing Linux involved a longer, more complex process compared to the others. This product is targeted towards advanced users that are able to build, design, and program their complete system. Due to the time constraints of my project, I was against this product. Another time sink involved with this product was the lack of an on-board Ethernet. There is a large community and open-source projects related to this board – again, targeted towards advanced users.

Phidget SBC 1070

The Phidget SBC 1070 is a very capable board as well. Its downsides include its cost, small memory, and lack of an RS-232 port. A RS-232 could be installed by using the pins provided on the board. However, I was looking for a board that required the least amount of time to setup all necessary elements. The memory could also be expanded by installing proper USB modules and using a USB stick. The support for this product was well documented but an online community presence appeared bleak.

TS-7350

The TS-7350 is a very balanced board for my purposes. Its speed may be slower than the others, but it can operate in harsh environments (-40° to +70°C). Its price rivals the price of the Beagle Board and includes an Ethernet. The company, Technologic Systems, is available for support through phone calls and e-mails. As well, there is a big community available online. I was able to contact the company during the week and speak with an engineer in regards to my project. The engineer was friendly and supportive. The company also provided an online repository for a Linux distribution of Debian with full compiling chains. This gave me a solid foundation for my project.

To validate my choice, I further evaluated the boards in a decision table by rating their attributes and comparing the totals. As expected, the TS-7350 had the most points with a 71. The table is located below.

Table 2: SBC Decision Table.

	Cost	CPU	Size	Linux Installation	Memory	Ethernet	Serial RS-232	Support	Total
Beagle Board	10	10	10	1	10	0	10	8	**59**
Phidget SBC 1070	5	7	9	8	5	10	0	5	**49**
TS-7350	8	6	8	10	10	10	10	9	**71**

pcProx RFID Reader

The pcProx RFID reader is manufactured by RFIDeas. They specialize in proximity and contactless reader solutions. I choose this hardware element due to its ease of integration, cost, and fulfillment of my requirements. It can be plugged into any PC via USB. The device emulates a keyboard and will

keystroke the detected RFID tags to the PC. The device also can with a little configurable utility with some options such as typing speed and additional keystrokes.

Software Solution

As mentioned above, I chose to use Linux as my Operating System (OS) due to its stability, coding environment, and being free open-source software. As required by the specifications and description, Apache Server, MYSQL, and PHP will be installed to support the processes.

Linux

The specific distribution I used was Debian. It is a free OS and is maintained, supported, and upgraded by the public (users and enthusiasts). Linux gets its name from the creator of its original kernel: Linus Torvalds. A kernel is a piece of software that manages the application to hardware processes and communication. Kernel version 2.6 was used in my project. Excellent system stability is achieved from the 2.6 kernel.

Linux is an excellent coding environment. It gives complete control of the system through the root user. It is capable of compiling and running the C code required for my project. The programming required to complete my project will use C programs to allow the RFID reader to communicate with the OS. The information is then processed and transferred to the database.

As mentioned in the requirements section, keeping costs low is preferred. By choosing Linux, an open-source OS, some expenses were avoided.

Apache Server

Apache HTTP server is a web server software developed and maintained by an open community developers much like Linux. Apache is supported on many platforms including my particular version of Linux. Apache is by far the most popular HTTP server software in use due to its functionality, performance, and cost.

Apache is required to host the web GUI. The software allows users to connect to the web page interface through browser clients. The network and Apache software communicate to direct and handle the appropriate requests from users.

The web GUI is essentially just files (.php, .css, etc, files). Apache simply allows access to these files on the network when requested. Therefore, Apache is only a piece of the required components for the web GUI to actually function.

MYSQL

To efficiently store the log data, a database will be used. There are many advantages to using a database rather than a flat file. Database systems are faster, provide security, and maintain data integrity. Databases use indexing to improve the speed of data retrieve operations. Users can be given rights to view only specific parts of the database. Data integrity is maintained through referencing and data validation rules.

A database index data structure is created by using one or more columns in a table. Using these indexes, rapid look ups and efficient access is provided to the application. The orders of the columns are very important when choosing an index. To decide the best index, evaluation of most common operation must be determined for the best performance. For instance, my project will mostly query information about a tool and if it is checked in or out. By having an index data structure, databases are much faster than a flat file. A flat file would require loading the full file each time. Further, the flat file system would require searching through an unknown amount of data until what is requested is actually found.

A flat file has limited security depending on the OS. For example, in Linux ownership could be assigned. A much more effective system is implemented in databases. It prevents unintended activity such as deleting a log instance or user. The database security has many protective layers including access control, auditing, authentication, encryption, and integrity checks. This way, a user is allowed access to only parts of the database, not the entire database. Flat files can only give access to the entire file or none.

To be a valid database, the data must be accurate, valid, and consistent. One way this is done is by referencing. Suppose a user Jane Smith has an entry in the database. She has done many transactions with tools over the years. Next, Jane Smith gets married and changes her last name to Jones. Her name is stored in only one place in the database and referenced by many other entries (transactions). The single database entry is simply updated to Jane Jones and all the references are therefore updated. Flat files would require updating every instance of Jane Smith. This is further illustrated in the figure below.

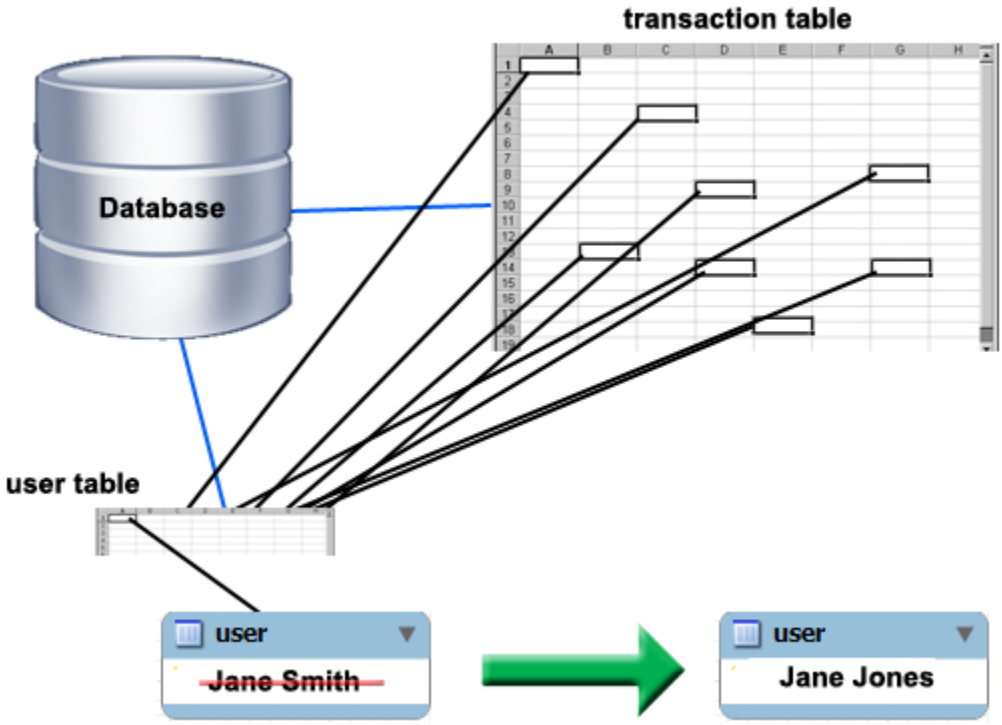


Figure 7: Jane Smith Example. Located in the database there are tables. The transaction table references the user Jane Smith. Her entry is saved under the user table.

There are a couple of other rules a database must follow:

- **Entity integrity:** allowing no two rows to have the same identity within a table.
- **Domain integrity:** restricting data to predefined data types.
- **Referential integrity:** requiring the existence of a related row in another table.

These rules further add to a level of data integrity unreachable by flat files.

MYSQL is a free open source relational database management system (RDBMS). It's a project under the GNU General Pubic License (GPL) like Apache. Therefore, the source code is openly shared for development opportunities. Meeting my project requirements, MYSQL is both inexpensive and robust. MYSQL is currently used by many large scale projects such as Facebook and WordPress.

PHP

PHP Hypertext Preprocessor (PHP) is scripting language designed for web development to produce dynamic web pages. PHP modules are installed on a web server (such as Apache) and PHP code is injected into HTML documents. The web server parses the HTML documents for PHP code and preprocesses them before delivering the web page to the client – this achieves dynamic web pages. The PHP is executed at runtime server side and therefore frees the end client's browser of resources and processing. PHP is also under the GPL and is free software.

PHP is written in C and implements communication with databases. This was the primary use for my project.

Design

With the requirements and the hardware/software combinations determined, I could begin to design the system. I'll first display the connection diagrams and then explain them.

Hardware

The figure below diagrams the hardware connections for my system. The pcProx RFID Reader was selected after online research was done. It was a quick, easily installed hardware element (via USB) that fit my solution. The software section will cover how the pcProx device works with Linux.

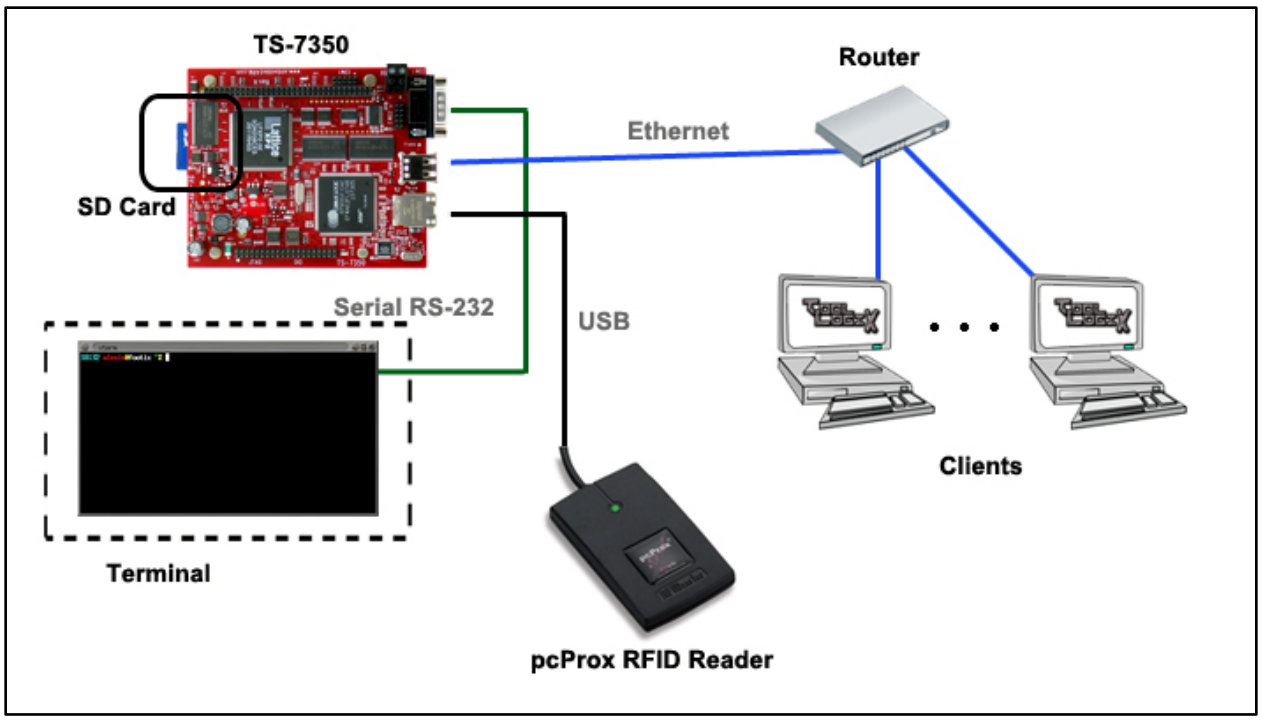


Figure 8: Hardware Setup. The hardware was setup as shown above. The TS-7350 comes with an Ethernet port, USB, and serial RS-232 inputs. With a SD card loaded with a Linux OS image, the device can boot up and recognize all devices.

The router, clients, SD card, and terminal elements are not specific and are interchangeable. The TS-7350 SBC is the central unit. The pcProx RFID Reader acts as the location where users swipe their badges and tools. The TS-7350 is hooked into a company's internal network via Ethernet and router making the web GUI accessible to all employees. This allows employees to check on a tool from their desk rather than making the trip to the tool area and returning empty handed. Employee efficiency is achieved this way. The pcProx RFID reader is connected to the TS-7350 by USB and has a 6 foot cord to allow for some placement mobility.

As indicated above, the serial RS-232 input can be hooked into a computer allowing a user to access the device via terminal. The Linux OS is located on the SD card. One can access the file system without taking the system offline by using the serial connection. The alternative method would be to turn the system off, remove the SD card, and read it on a PC. Typical company usage however will not need the serial connection at all because the system is setup to work without any setup. The system is simply given power and plugged into the network. Once it is powered on, the device auto boots into Linux and begins it required services.

The web GUI is accessed by a browser with the default internal IP of 192.168.1.122. Any employee on the active network will be able to access the interface and view the logs.

A small problem with my hardware was revealed during testing. The TS-7350 does not come with a Real Time Clock (RTC) module standard. I had switched to this model (from the TS-7200) midway during my project. I had overlooked this because the TS-7200 had the RTC standard. Essentially, this means my hardware does not keep track of time when shut off. Therefore, upon boot up, the date is reset to 1970.

Software

As discussed earlier, the software bundle I chose was Linux, Apache, MYSQL, PHP. This software bundle is commonly known as LAMP. Many web developers have used this setup making it a very common solution. A diagram below maps the software pieces together.

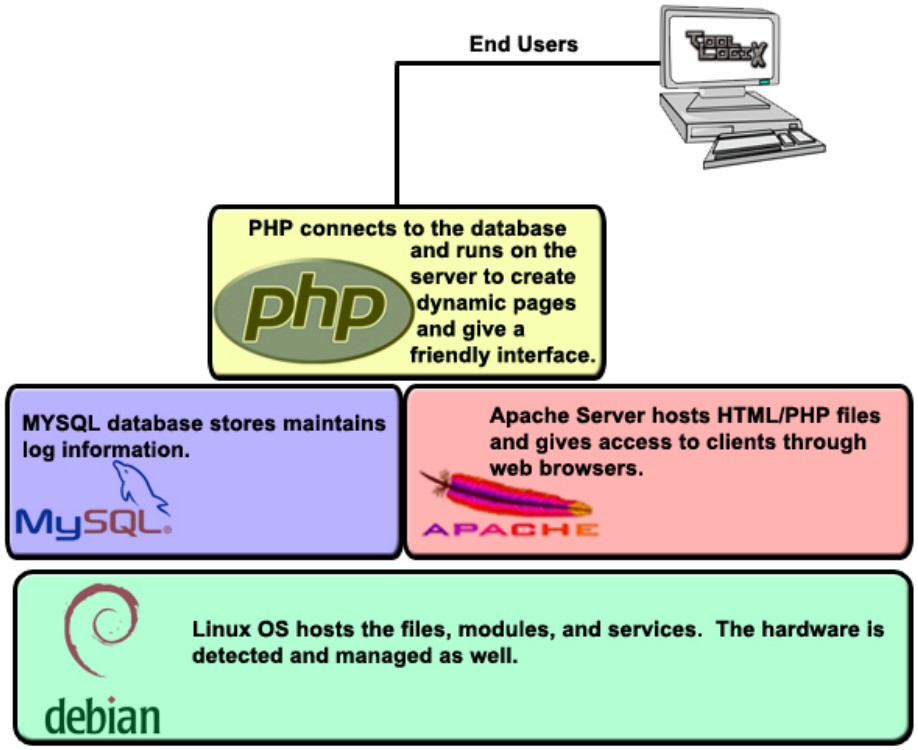


Figure 9: LAMP Software Stack. This figure depicts the communications described earlier with software. Linux runs MYSQL and Apache services. The MYSQL database stores the data and can be accessed by PHP. When a client wants to view the logs, he opens a browser and requests it through HTTP. The Apache Server runs the PHP module and processes the correct page for the client.

TS-7350 Setup

The TS-7350 has firmware residing in FPGA ROM memory that boots to the SD card on the board. The SD card must have a Linux image installed in order for it to work. Tutorials and images were available on the manufacturer's website. I however had to alter the Linux image for the purposes of my project.

Debian Linux 2.6

To achieve console access through the serial COM1 port on the board, Linux had to be setup correctly.

- Edit the Linux kernel image to display in ttyAM1.
- Edit /etc/inittab to spawn a getty on ttyAM1 at 115200 baud.
- Edit /dev/securetty and add /dev/ttyAM1 to list of root login accesses.

Auto-login Script

An auto-login script was implemented to further make the device user-friendly. This was done with the mingetty package. With Debian, packages can easily be installed using the "apt-get install" command with access to the internet. To get the auto-login script to work I put the following line in the /etc/inittab:

```
T0:23:respawn:/sbin/mingetty –L –autologin root ttyAM1 115200 vt100.
```

LAMP

To implement the end user web GUI, web services are required. The LAMP software bundle was installed using apt-get as well.

Networking

Networking is required to access the internet and for clients to connect. Apt-get relies on internet access to download packages. To properly set-up a static IP, I had to edit the /etc/network/interfaces file. I set the IP to a local address of 192.168.1.122. This will be the portal for clients to access the web GUI. The important code of the file is listed below:

```
iface eth0 inet static
      address 192.168.1.122
      network 192.168.1.0
      netmask 255.255.255.0
      broadcast 192.168.1.255
      gateway 192.168.1.1
      ### static routing ###
      up route add -net 192.168.1.0 netmask 255.255.255.0 gw
```

Database Architecture

Specifications for the database require it to be efficient and robust. Given the expected long term usage of the device, the amount of transactions will be high. Essentially, this translates to a high storage of data. Therefore, the database architecture will be crucial in securing the device's integrity in speed and usage. If the device runs out of physical space, new transactions will not be saved. If the database architecture is not efficient, it will slow down the web GUI's response for the end-user. With this in mind, the database architecture will need the following specifications: **relational** and **normalized.**

Relational

A relation is another term for a table. A relational database is a database of multiple tables that relate to another and reference each other. A flat database would contain only one table – a single file database. A flat database is the opposite of a relational database.

Normalized

A normalized database is free of potential threats to data integrity through insertion, update, or deletion operations. There are different levels of normalization but for my purposes, ensuring data integrity is enough. A database is not normalized if an insertion, update, or deletion operation would create an inconsistent data entry relative to the rest of the database. For example, if a user was placed into a table twice with different badge IDs.

Database Architecture Design

To develop database architecture, evaluation of the data to be stored must be done first. There are two types of objects: people and tools. These two objects together create a transaction. To identify our two types of objects (people and tools), RFID tags are assigned to them. This evaluation gives us four tables, namely:

1. Users
2. Tools
3. Transactions
4. RFID Tags

By following the specifications listed above, the resulting database architecture is shown below.

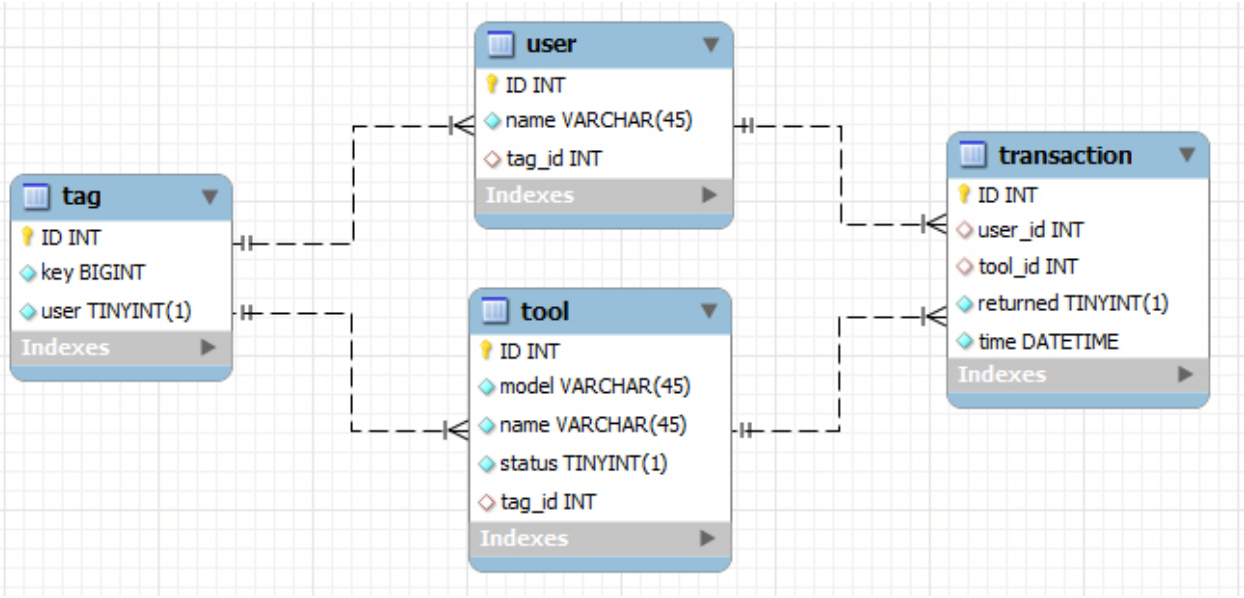


Figure 10: Enhanced Entity-Relationship Model (EER). An EER model is a high level data model used in database designs.

The diagram depicts the cardinality of the tables. Appropriately, there are many tags for users and tools, but a user or tool is assigned only one tag. Likewise, a tool or user will have many transactions, but a transaction will only contain one user and one tool. The notation above is called "crow's foot."

Tool Table

The tool table has the following attributes: ID, model, name, status, and tag_id.

Attribute	Datatype	Comment
ID	INT	Primary Key
model	VARCHAR(45)	Manufacturer identification
name	VARCHAR(45)	Tool Name
status	TINYINT	Equals '1' when checked in
tag_id	INT	Foreign Key

User Table

The user table has the following attributes: ID, name, and tag_id.

Attribute	Datatype	Comment
ID	INT	Primary Key
Name	VARCHAR(45)	User Name
tag_id	INT	Foreign Key

Tag Table

The tag table has the following attributes: ID, key, and user.

Attribute	Datatype	Comment
ID	INT	Primary Key
key	INT	RFID key
user	TINYINT	Equals '1' if assigned to a user

Transaction Table

The transaction table has the following attributes: ID, user_id, tool_id, returned, and time.

Attribute	Datatype	Comment
ID	INT	Primary Key
user_id	INT	The user involved
tool_id	INT	The tool involved
returned	TINYINT	Equals '1' when checked in
time	DATETIME	Timestamp

Web GUI

The web GUI specifications require it to be user friendly, efficient, and robust. Compatibility is also a concern given the medium. There are many different web browsers and support should be included for

as many users as possible. To accomplish compatibility, code has been written to work for Microsoft Internet Explorer (IE) and Mozilla Firefox (FF). Testing has also confirmed success in both browsers.

According to w3schools.com (May 2010), 79.1% of browsers are IE or FF.

PHP

PHP is a scripting language that enables database communication with a web browser and was the main purpose in my project. Given the database name, user, and password, SQL select statements are possible to retrieve tool and log transactions. This data is then displayed on the screen.

AJAX

To create a user friendly web GUI, Asynchronous JavaScript and XML (AJAX) has been implemented. Similar to Google-Mail, this gives the user a sense of a live application rather than a static web page. Instead of loading the page for each interaction (user click), a HTTP request is sent to the server in the background. A response is returned to the client and only particular elements on the page are updated via JavaScript. This technology also reduces bandwidth significantly due to clients not having to request static elements repeatedly. A few screen shots are displayed below.

Figure 11: Web GUI Recent. The default display gives the client the recent transactions. This is the digitized version of the paper sheet log. The table is arranged with the most recent transaction on the top. The menu on the side will change the content of the table on the right. The table element is updated through AJAX therefore the complete GUI is actually just one single page.

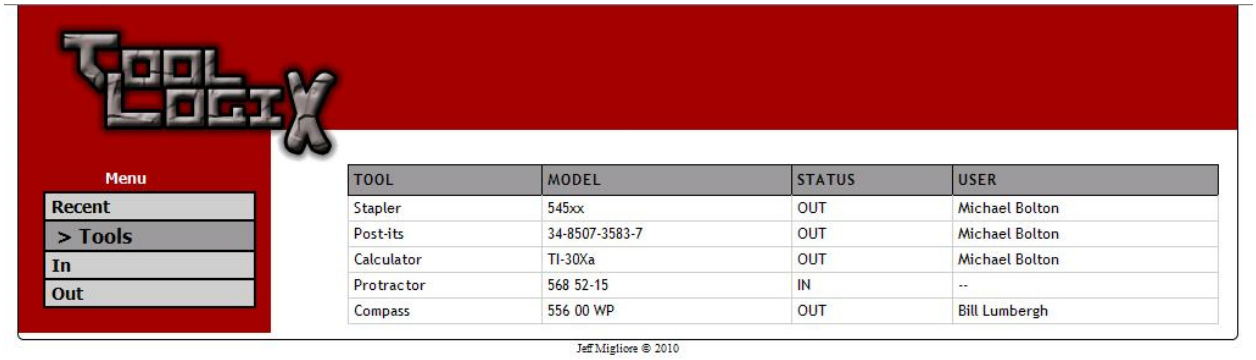


Figure 12: Web GUI Tools. This view lists all the assets that are tracked. It can give a quick overview of which tools are currently available. As indicated above, the only tool available is the protactor.

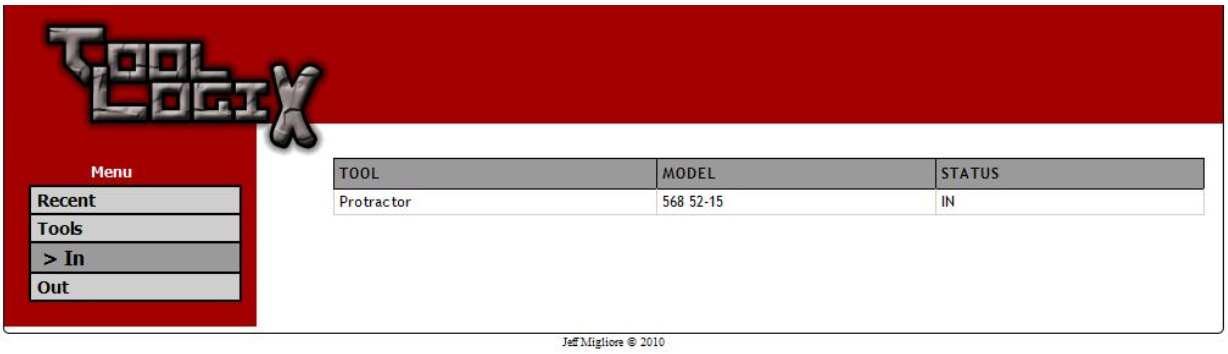


Figure 13: Web GUI Tools In. The next two views are children of the Tools menu. They further filter the display if a company has many tools. As we saw before, only the protractor is available.

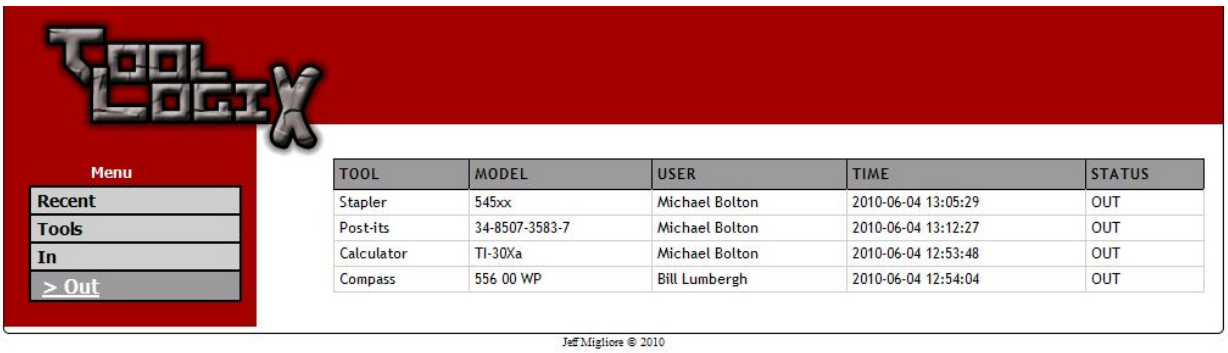


Figure 14: Web GUI Tools Out. This filter displays the tools currently checked out. As seen before, the stapler, post-its, calculator and compass have been checked out.

Miscellaneous

This section has miscellaneous information related to my project.

Future Work

As mentioned in the hardware section, my TS-7350 did not come standard with a RTC module. This prevents my OS from properly keeping track of time. A RTC keeps track of time when a computer is powered off. It usually self powered by a lithium battery and recharges when the connected computer is powered on. Due to this the battery in theory could run out of power. Most modern RTC modules however can sustain on their own for years. To fix this problem, I would simply send my TS-7350 device back to the company through RMA and have the RTC module installed for a cost of $10.00 USD.

Another implementation I'd like to add to my project is related to the web GUI. With the power of PHP available, e-mail services are made easy. I would like my GUI to allow users to sign up for e-mail alerts.

The use case would be the following:

1. Employee A is tasked for a job. He needs a tool in order to complete his task.
2. Employee A goes to the tool area and finds the tool.
3. Employee A scans the needed tool and his badge.
4. Employee A finishes his task but leaves the tool at his work area.
5. Employee B is tasked for a job. He needs the same tool as Employee A to complete his task.
6. Employee B goes to the tool area. Unfortunately, Employee B does not find the tool because Employee A never returned it.
7. Employee B goes to the computer and checks the log. He sees Employee A has the tool.
8. Employee B attempts to contact Employee A but Employee A is not responding to phone calls or e-mails.
9. Employee B signs up for an e-mail notification through the web GUI that notifies him when the tool he needs is checked back in.

Working with a Partner

Not all partnerships can be successful. This is the lesson I learned through my senior project process. I had initially partnered up with an Electronic Engineering student for this project. We had agreed upon the project's design with great expectations. The system was designed to be embedded within a toolbox. A user's badge would unlock the toolbox and allow access to the assets.

Next, my partner and I then split up the duties. I was tasked to the SBC solution including the web GUI and full LAMP software bundle. His responsibilities included the RFID reader, RFID tags, and toolbox locking mechanism. Our communication was excellent on our intended goals and duties. Unfortunately, we had failed in terms of expected completion date. My partner was planning on graduating during the summer and not spring. This ultimately meant his solution would not be completed in time for me to demo my project.

I was then forced to find a RFID solution myself and implement it. I had to also re-evaluate my project's design. This kind of event is not uncommon in the real world. Not all peers finish their work on time and it is just the harsh reality of life. If I had the chance again, I would have communicated and setup a schedule more explicitly with my partner before fully committing and beginning any work.

Bumps in the Road

As mentioned in previous sections, I had a couple of bumps in the road. Having lost a partner more than half way down the line was unfortunate. It made me scramble to find a capable solution. The pxProx RFID reader was ultimately found and integrated to my system. A proper research of RFID readers was not in the time tables for me at this point. I would have liked to evaluate solutions like I had done for my SBC.

When I initially started the project with my partner, he had a SBC lying around: TS-7200. With this, we began designing with this SBC. I agreed due to the fact we already had the hardware. I later found out the SBC only supports a 2.4 Linux kernel. This was to my distaste because the 2.4 kernel is over a decade old. I had never used a kernel that old and finding support and community would be harder as well. I did the proper analysis for other SBCs and ended up with the same company with a different model. The TS-7350 was adequate for my purposes. I unfortunately overlooked the fact that the RTC module was optional. I did not realize this was the issue at first and thought it was software. After being stumped for some time, I looked at my TS-7350 specification sheet closer and noticed the RTC module was available for $10.00 USD.

Project Management – dotProject

I decided it would be a good idea for my partner and me to track our progress for documentation (plus we could reference it later if either of us decides to do something similar in the future). I choose to use dotProject. It will also be a means of us to communicate during the project and a repository for software if we need.

This project management software is free and is managed, maintained, developed, and supported by volunteers. It has been very useful. We could dump our error messages and report on progress/new information as we work. When I completed the project, dotProject lubricated the write up process by providing all the information we used, pitfalls, lessons, dates, etc.

Appendixes

Appendix A: Analysis of Senior Project Design

Analysis of Senior Project Design

Please provide the following information regarding your Senior Project and submit to your advisor along with your final report. Attach additional sheets for your responses to the questions below.

Project Title: RFID Asset Tracking and Digital Check-Out System

Quarter / Year Submitted: Spring 2010

Student: (Print Name) Jeff Migliore

Advisor: (Print Name) Dr. Hugh Smith

• Summary of Functional Requirements

Describe the overall capabilities of functions of your project or design. Describe what your project does. (Do *not* describe how you designed it.)

For my senior project, I developed a radio frequency identification (RFID) asset tracking and digital check-out system with a database backend and web graphical user interface (GUI) frontend. Specifically, this system was designed to track tool assets.

• Primary Constraints

Describe significant challenges or difficulties associated with your project or implementation. For example, what were limiting factors or other issues that impacted your approach? What made your project difficult? What parameters or specifications limited your options or directed your approach?

My project did not go as smoothly as I wanted. My partner had dropped and some hardware/software difficulties arose. I had to change SBC models during the quarter for a proper solution. I upgraded to get a Linux 2.6 kernel from a 2.4. Some software setup took time to figure out. To enable the root login on the COM1 port took some research relative to the Linux OS. The most difficult part of my project was time management. I had though things were going smoothly because I had my duties covered. When my partner dropped out on me I had to scramble to complete his solution and redesign the project.

• Economic

o Original estimated cost of component parts (as of the start of your project)

The original estimated cost was:

TS-7200 (0) + toolbox ($20) + misc. cords/adapters ($30) = **$50**

o Actual final cost of component parts (at the end of your project)

The actual costs are a lot higher due to my partner dropping out and not having the initial adequate hardware we thought we had.

TS-7350($175) + pcProx RFID Reader ($140) + misc. cords/adapters ($30) = **$345**

- *Attach a final bill of materials for all components*
- Additional equipment costs (any equipment needed for development?)

None, software used was open-source.

- Original estimated development time (as of the start of your project)

200 man hours – shared between two people.

- Actual development time (at the end of your project)

200 man hours – this was however placed on one person instead of two.

• If manufactured on a commercial basis:

- Estimated number of devices to be sold per year

20 – Clients would usually only need one or two. Finding around 20 new clients a year would be good.

- Estimated manufacturing cost for each device

$500 – Includes cost of labor and setup.

- Estimated purchase price for each device

$800 – This sale price would include on-site technical support if needed to reset the device.

- Estimated profit per year

$6,000 – This product could be sold easily by a single person and would be supplemental income.

- Estimated cost for user to operate device, per unit time (specify time interval)

Device would normally be on all the time. However, the TS-7350 is a low power device (5 volts). It would not cost a significant amount of electricity therefore the cost to operate would be next to none.

• Environmental

Describe any environmental impact associated with manufacturing or use. The device is Restriction of Hazardous Substance (RoHS) compliment. Also, by using this device rather than paper, trees will be saved. The only consumption of the device is electricity which even then is a low power device.

• Manufacturability

Describe any issues or challenges associated with manufacturing.

Manufacturing would be simple. The components could be ordered and packaged together. The proprietary Linux image would be burned to every SD card shipped out.

• Sustainability

- Describe any issues or challenges associated with maintaining the completed device or system.

The device is plug and play. The end users would not have to do anything but ensure power is supplied.

- Describe how the project impacts the sustainable use of resources.

This device uses electrical components and consumes electricity around the clock at a low rate.

- Describe any upgrades that would improve the design of the project.

More software could be implemented for e-mail notifications and the RTC module should be installed as well.

- Describe any issues or challenges associated with upgrading the design.

With the nature of software, debugging and testing would be needed to confirm design. Once the design is stable, nothing further would need to be done. Implementing the upgrade would be simple by replacing code.

• Ethical

Describe ethical implications relating to the design, manufacture, use or misuse of the project.

The device could be used maliciously if an employee steals an identity of another employee via badge. He could then check out tools under his name and steal them.

• Health and Safety

Describe any health and safety concerns associated with design, manufacture or use.

This device is safe as long as used properly. The only element of harm would be electrical if a user were to play with the electrical cord or socket.

• Social and Political

Describe any social and political concerns associated with design, manufacture or use. This device does not have any social or political concerns. RFID use has been rising steadily in the industry.

• Development

Describe any new tools or techniques used for either development or analysis that you learned independently during the course of your project.

I learned a lot about the initialization process of Linux and Linux in general. I also became familiar with the Debian distribution. I learned about some RFID standards and many open source software packages available such as the LAMP bundle and dotProject for project management.

Appendix B: Source Code - C

insert_xaction.c

```
#include <mysql.h>
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <stdlib.h>

/*****************************
* Arg 1 is a badge or a tool rfid
* Arg 2 is a badge or a tool rfid
```

```
*
* Program determines if a tool
* is being returned or taken
* and logs the transaction in
* the database.
*************************/

MYSQL *conn;
MYSQL_RES *res;
MYSQL_ROW row;
char query[1024];

int setup(int argc)
{
      char *server = "localhost";
      char *user = "root";
      char *password = "root";
      char *database = "tool_logix";

      conn = mysql_init(NULL);

      /* Setup Check */
      if ((getuid ()) != 0)
            printf ("You are not root! This may not work...\n");

      if (argc < 3)
      {
            printf("Please give two tags\n");
            exit(0);
      }

      /* Connect to database */
      if (!mysql_real_connect(conn, server, user, password, database, 0,
NULL, 0))
      {
            fprintf(stderr, "%s\n", mysql_error(conn));
            exit(0);
      }

      return 0;

}

int do_query(int rt)
{
      int val = 0;

      printf("exe[%s]\n", query);
      if (mysql_query(conn, query))
      {
            fprintf(stderr, "%s\n", mysql_error(conn));
            return 0;
      }

      res = mysql_use_result(conn);

      if(rt)
```

```
	{
		row = mysql_fetch_row(res);
		if(row != NULL)
		{
			val = atoi(row[0]);
		}else{
			puts("Invalid RFID");
			exit(1);
		}
	}

	mysql_free_result(res);

	return val;
}

int get_tag_id(char *key)
{
	sprintf(query, "SELECT ID FROM tag WHERE tag.key = '%s'", key);

	return do_query(1);
}

int get_id(int tag_id, int user)
{
	char table[5];

	if(user == 1)
	{
		sprintf(table, "user");
	}else{
		sprintf(table, "tool");
	}

	sprintf(query, "SELECT ID FROM %s WHERE tag_id = '%d'", table, tag_id);

	return do_query(1);
}

int check_tool(int tag_id)
{
	sprintf(query, "SELECT status FROM tool WHERE tag_id = '%d'", tag_id);

	return do_query(1);
}

int check_user(char *key)
{
	sprintf(query, "SELECT user FROM tag WHERE tag.key = '%s'", key);

	return do_query(1);
}

int main(int argc, char *argv[])
{
	int tag_1, tag_2;
	int user_tag_id, tool_tag_id;
```

```
	int user_id, tool_id;
	int status, returned;

	setup(argc);

	tag_1 = check_user(argv[1]);
	tag_2 = check_user(argv[2]);

	if( (tag_1 == 1 && tag_2 == 1) || (tag_1 == 0 && tag_2 == 0) )
	{
		puts("Invalid Operation");
		exit(2);
	}else if(tag_1 == 1){
		user_tag_id = get_tag_id(argv[1]);
		tool_tag_id = get_tag_id(argv[2]);
	}else{
		user_tag_id = get_tag_id(argv[2]);
		tool_tag_id = get_tag_id(argv[1]);
	}

	user_id = get_id(user_tag_id, 1);
	tool_id = get_id(tool_tag_id, 0);

	status = check_tool(tool_tag_id);

	returned = status == 1 ? 0 : 1;

	sprintf(query, "INSERT INTO transaction (user_id, tool_id, returned,
time) VALUES ('%d', '%d', %d, NOW())", user_id, tool_id, returned);
	do_query(0);

	sprintf(query, "UPDATE tool SET status = '%d' WHERE ID = '%d'",
returned, tool_id);
	do_query(0);

	mysql_close(conn);

	return 0;
}
```

my_keyb_read.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <errno.h>
#include <fcntl.h>
#include <dirent.h>
#include <linux/input.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/select.h>
#include <sys/time.h>
#include <termios.h>
```

```
#include <signal.h>


#define VENDORID  0x0c27
#define PRODUCTID 0x3bfa
#define NUM 100000000000

/***************************
*
* Program locates the pcProx
* device and attaches itself to
* it.  Upon receiving two tags
* within 10 seconds, it will
* sent the rfid tags to the insert
* program to update the database
* and complete the transaction.
***************************/

void handler (int sig)
{
  printf ("\nexiting...(%d)\n", sig);
  exit (0);
}

void perror_exit (char *error)
{
  perror (error);
  handler (9);
}

int initDevice(){
      int count=0, scan_fd = -1;
      char path[32];
      struct dirent **files = NULL;
      struct input_id id;

      count = scandir("/dev/input",&files,0,alphasort);
      count--;

      while( count >= 0 )
      {
            if( scan_fd == -1 && strncmp(files[count]->d_name,"event",5) ==
0)
            {
                  sprintf(path,"/dev/input/%s",files[count]->d_name);
                  scan_fd = open(path,O_RDONLY);

                  if( scan_fd >= 0 )
                  {
                        if( ioctl(scan_fd,EVIOCGID,(void *)&id) < 0 )
                              perror("ioctl EVIOCGID");
                  }else{
                        if( id.vendor==VENDORID && id.product==PRODUCTID )
                        {
                              printf("Found device at %s\n", path);
                        }else{
                              close(scan_fd);
```

```
                              scan_fd = -1;
                        }
                  }
            }
            free(files[count--]);
      }

      free(files);

      if( scan_fd >= 0 )
      {
            ioctl(scan_fd,EVIOCGRAB);
      }else{
            printf("scanner not found or couldn't be opened\n");
            exit(0);
      }

      return scan_fd;
}

int main(int argc, char *argv[])
{
      struct input_event ev;
      int fd, rd, ev_size = sizeof (struct input_event);
      int key_val;
      char name[256] = "Unknown";
      char cmd[1024];
      double temp, tag_id_1, tag_id_2, k;
      time_t last_sec;
      suseconds_t last_usec;
      temp = tag_id_1 = tag_id_2 = last_sec = last_usec = 0;
      k = NUM;

      //Setup check
      if ((getuid ()) != 0)
            printf ("You are not root! This may not work...\n");

      //Device Setup
      fd = initDevice();
      ioctl (fd, EVIOCGNAME (sizeof (name)), name);
      printf ("Reading From : (%s)\n", name);

      while (1)
      {
            if ((rd = read (fd, &ev, ev_size)) < ev_size)
                  perror_exit ("read()");

            if (ev.value == 1 && ev.type == 1) // Only read the key press
event
            {
                  if(last_usec != ev.time.tv_usec) //Multi-read protection
                  {
                        if(ev.code == 28 && temp > 0) //Return key code
                        {
                              printf("tag[%.0f]\n", temp);
                              k = NUM;
```

```
                              if( (tag_id_1 == 0) || (ev.time.tv_sec -
last_sec > 10) ) //Timeout transaction after 10 sec
                              {
                                    tag_id_1 = temp;
                              }else{
                                    tag_id_2 = temp;
                                    sprintf(cmd, "./insert.o %.0f %.0f",
tag_id_1, tag_id_2);
                                    system(cmd);
                                    tag_id_1 = tag_id_2 = 0;
                              }
                              temp = 0;
                              last_sec = ev.time.tv_sec;
                              continue;
                        }

                        if(ev.code == 11) //Zero key Code
                        {
                              key_val = 0;
                        }else{
                              key_val = ev.code - 1;
                        }

                        temp += key_val * k;
                        last_usec = ev.time.tv_usec;
                        k /= 10;
                  }
            }
      }

      return 0;
}
```

Appendix C: Source Code - HTML

Index.php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

<link href="style.css"  rel="stylesheet" type="text/css"/>

<title>Tool Logix</title>

<script type="text/javascript">
```

```
var http;
var sel_table;


function GetXmlHttpObject()
{
	if (window.XMLHttpRequest)
	{
	// code for IE7+, Firefox, Chrome, Opera, Safari
	return new XMLHttpRequest();
	}
	if (window.ActiveXObject)
	{
	// code for IE6, IE5
	return new ActiveXObject("Microsoft.XMLHTTP");
	}
	return null;
}
//reload content
function reloadContent(table){
	http=GetXmlHttpObject();
	oDiv = document.getElementById("content");
	http.open('get', 'inc/display_content.php?table='+table);
	sel_table = table;
	http.onreadystatechange = reloadContent_reply;
	http.send(null);
```

```
}
function reloadContent_reply() {
	if(http.readyState == 4){
		var response = http.responseText;
		reloadContent_response(response);
	}
}
function reloadContent_response(response){
	oDiv = document.getElementById("content");
	oDiv.innerHTML = response;
	reloadMenu(sel_table);
}

//reload menu
function reloadMenu(selected){
	http=GetXmlHttpObject();
	oDiv = document.getElementById("menu");
	http.open('get', 'inc/display_menu.php?selected='+selected);
	http.onreadystatechange = reloadMenu_reply;
	http.send(null);
}
function reloadMenu_reply() {
	if(http.readyState == 4){
		var response = http.responseText;
		reloadMenu_response(response);
```

```
                }
        }
        function reloadMenu_response(response){
                oDiv = document.getElementById("menu");
                oDiv.innerHTML = response;
        }
</script>
</head>

<body>
<DIV ID="container">
  <div ID="top"><img src="img/logo.jpg" /><img src="img/red.gif" width="730" height="130"/></div>

  <div ID="wrapper">
    <div ID="side">
        <div class="header">Menu</div>
        <ul class="menu" ID="menu"><?php include 'inc/display_menu.php' ?></ul>
    </div>
    <div ID="content"><?php include 'inc/display_content.php'; ?></div>
  </div>
</DIV>

<div ID="footer">Jeff Migliore &copy; 2010</div>

</body>
```

```
</html>
```

Style.css

```
@charset "utf-8";
/* CSS Document */

body{
	margin: 0px;
	padding: 0px;
	color: #000;
	background: #FFF;
}

#container {
	width: 1000px;
	background: #FFF;
	margin: 0 auto;
	text-align: left;
	padding: 0;
	border-left: 1px solid #000;
	border-right: 1px solid #000;
	border-bottom: 1px solid #000;
	-moz-border-radius: 0 0 5px 5px;
}
```

```
#top{
        clear: both;

}

#wrapper{
        position: relative;
        overflow: hidden;
        width: 100%;
        padding-bottom: 5px;
}

#side{
        width: 270px;
        float: left;
        background-image: url(img/red2.gif);
        background-repeat: repeat-y;
}

#content{
        width: 715px;
        padding: 0 15px 0 0px;
        background-color: #FFF;
        float: left;
        position: relative;
```

```
}

#footer{
        clear: both;
        font-size: 10px;
        width: 100%;
        text-align: center;
}

#side .header {
        width: 175px;
        color:#fff;
        text-align:center;
        font-weight:bold;
        padding:4px 0 4px 0;
}
#side .menu {
        width: 175px;
        list-style-type:none;
        padding:0 0 2px 0;
        margin:0 0 20px 20px;
        background: #000;
}
#side .menu > li {
        padding:2px 2px 0 2px;
```

```
        margin:0;
}
#side .menu > li a {
        display:block;
        padding:2px;
        background-color:#CCC;
        font-weight:bold;
        text-decoration:none;
        color:#000;
}
#side .menu > li a:hover {
        text-decoration:underline;
        background-color:#999;
        color: #FFF;
}
#side .menu .selected {
        background-color:#999;
        padding-left:6px;
        font-size: 16px;
}

.header{
        font-family: Tahoma;
        font-size:12px;
}
```

```
.menu{
	background-color: #A0A0A4;
	font-family: Tahoma;
	font-size: 14px;
	font-weight: bold;
	text-indent: 3px;
}

#content table{
	width: inherit;
	border-collapse:collapse;
	border-left:1px solid #CCC;
	text-indent:3px;
	font-size-adjust:none;
	font-stretch:normal;
	font-style:normal;
	font-variant:normal;
}

#content table th {
-x-system-font:none;
background:#999;
border:1px solid #000;
color:#000;
```

```
font-family:"Trebuchet MS",Verdana,Arial,Helvetica,sans-serif;
font-size:12px;
font-weight:bold;
letter-spacing:1px;
line-height:22px;
text-align:left;
text-transform:uppercase;
vertical-align:middle;
}

#content table td {
-x-system-font:none;
border-bottom:1px solid #CCC;
border-right:1px solid #CCC;
color:#000;
font-family:"Trebuchet MS",Verdana,Arial,Helvetica,sans-serif;
font-size:12px;
font-weight:normal;
line-height:normal;
}
```

Db_connect.php

```
<?php
$dbhost = 'localhost';
$dbuser = 'root';
```

```
$dbpass = 'root';
$dbname = 'tool_logix';
$conn = mysql_connect($dbhost, $dbuser, $dbpass);
if (!$conn) {
   die('Could not connect: ' . mysql_error());
}
mysql_select_db($dbname, $conn);
?>
```

Db_close.php

```
<?php mysql_close($conn);?>
```

Display_menu.php

```
<?php
	if(!@file_exists('./db_connect.php') ) {
		include './inc/db_connect.php';
	}else{
		include('./db_connect.php');
	}
?>

<?php
	if(isset($_GET['selected'])){
		$selected = $_GET['selected'];
	}else{
		$selected = 'recent';
```

```
        }

        $s0 = $s1 = $s2 = $s3 = '>';
        $sel_str = "class='selected'> > ";

        if($selected == 'recent') $s0 = $sel_str;
        if($selected == 'tools') $s1 =  $sel_str;
        if($selected == 'in') $s2 =  $sel_str;
        if($selected == 'out') $s3 =  $sel_str;

        echo '<li><a href="#" onClick="reloadContent(\'recent\')" '.$s0.'Recent</a></li>
                <li><a href="#" onClick="reloadContent(\'tools\')" '.$s1.'Tools</a></li>
                <li><a href="#" onClick="reloadContent(\'in\')" '.$s2.'In</a></li>
                <li><a href="#" onClick="reloadContent(\'out\')" '.$s3.'Out</a></li>';
?>

<?php
        if(!@file_exists('./db_close.php') ) {
                include './inc/db_close.php';
        }else{
                include('./db_close.php');
        }
?>
```

Display_content.php

```
<?php
```

```
	if(!@file_exists('./db_connect.php') ) {
		include './inc/db_connect.php';
	}else{
		include('./db_connect.php');
	}
?>
<?php
	if(isset($_GET['table'])){
		$table = $_GET['table'];
	}else{
		$table = 'default';
	}

	echo '<table><tr>';

	if($table == 'tools'){ //Display Tool Table
		echo '<th>Tool</th>
			<th>Model</th>
			<th>Status</th>
			<th>User</th>
		       </tr>';
		$query = "SELECT model, name, status, ID as tool_id
				FROM tool";

		$result = mysql_query($query);
```

```
while($row = mysql_fetch_assoc($result))
        {
            echo "<tr>
                <td>".$row['name']."</td>
                <td>".$row['model']."</td>
                <td>";
            if($row['status'] == 1){
                echo "IN</td>
                <td>--</td>
                </tr>";
            }else{
                echo "OUT</td>";
                $query = "SELECT u.name
                    FROM user u, tool t, transaction x
                    WHERE u.ID = x.user_id AND t.ID =
x.tool_id AND t.ID = '".$row['tool_id']."'
                    ORDER BY x.time desc";
                $result_2 = mysql_query($query);
                $row_2 = mysql_fetch_assoc($result_2);
                echo "<td>".$row_2['name']."</td>
                </tr>";
            }
        }
    }else if($table == 'in'){ //Display In Table
        echo '<th>Tool</th>
```

```
                <th>Model</th>
                <th>Status</th>
              </tr>';
        $query = "SELECT model, name
                        FROM tool
                        WHERE status = '1'";

        $result = mysql_query($query);

        while($row = mysql_fetch_assoc($result))
                {
                        echo "<tr>
                                        <td>".$row['name']."</td>
                                        <td>".$row['model']."</td>
                                        <td>IN</td>
                                </tr>";
                }
}else if($table == 'out'){ //Display Out Table
        echo '<th>Tool</th>
                <th>Model</th>
                <th>User</th>
                <th>Time</th>
                <th>Status</th>
              </tr>';
        $query = "SELECT model, name, status, ID as tool_id
```

```
                                FROM tool
                                WHERE status = '0'";

                $result = mysql_query($query);

                while($row = mysql_fetch_assoc($result))
                        {
                                echo "<tr>
                                        <td>".$row['name']."</td>
                                        <td>".$row['model']."</td>";
                                $query = "SELECT u.name, x.time
                                        FROM user u, tool t, transaction x
                                        WHERE u.ID = x.user_id AND t.ID = x.tool_id AND t.ID =
'".$row['tool_id']."'
                                        ORDER BY x.time desc";
                                $result_2 = mysql_query($query);
                                $row_2 = mysql_fetch_assoc($result_2);
                                echo "<td>".$row_2['name']."</td>
                                        <td>".$row_2['time']."</td>
                                        <td>OUT</td>
                                </tr>";

                        }
        }else if($table == 'default' || $table == 'recent'){ //Display Recent Table
                echo '<th>User</th>
        <th>Tool</th>
```

```
<th>Action</th>
            <th>Time</th>
                </tr>';
        $query = "SELECT u.name, t.name as tool, x.returned, x.time
                        FROM user u, tool t, transaction x
                    WHERE u.ID = x.user_id AND t.ID = x.tool_id
                        ORDER BY x.time desc";

        $result = mysql_query($query);

        while($row = mysql_fetch_assoc($result))
                {
                    echo "<tr>
                                <td>".$row['name']."</td>
                                <td>".$row['tool']."</td>";
                    if($row['returned'] == '1'){
                        echo "<td>IN</td>";
                    }else{
                        echo "<td>OUT</td>";
                    }
                    echo "<td>".$row['time']."</td>
                        </tr>";
                }

}
```

```
        echo '</table>';
?>


<?php
        if(!@file_exists('./db_close.php') ) {
                include './inc/db_close.php';
        }else{
                include('./db_close.php');
        }
?>
```

Appendix D: Resources used

http://www.w3schools.com/browsers/browsers_stats.asp

http://www.howtoforge.com/installing-apache2-with-php5-and-mysql-support-on-debian-lenny-lamp

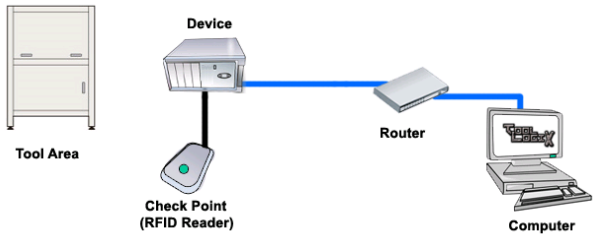
http://www.derekhildreth.com/portfolio/Documentation/TS-7350-Doc.pdf

ftp://ftp.embeddedarm.com/

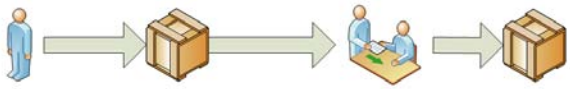
http://wikipedia.com

Appendix E: Poster Board Slides

Overview
For my senior project, I developed a radio frequency identification (RFID) asset tracking and digital check-out system with a database backend and web graphical user interface (GUI) frontend. Specifically, this system was designed to track tool assets.

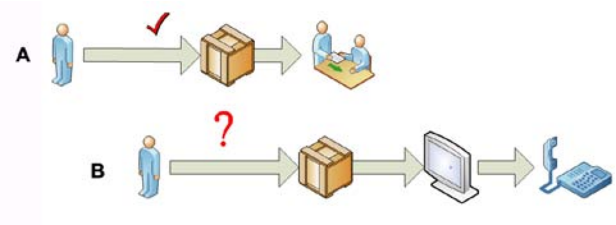


Use Case 1
1. Employee A is tasked for a job. He needs a tool in order to complete his task.
2. Employee A goes to the tool area and finds the tool.
3. Employee A scans the needed tool and his badge.
4. Employee A finishes his task and submits his work. He returns the tool by once again scanning his badge and the tool.



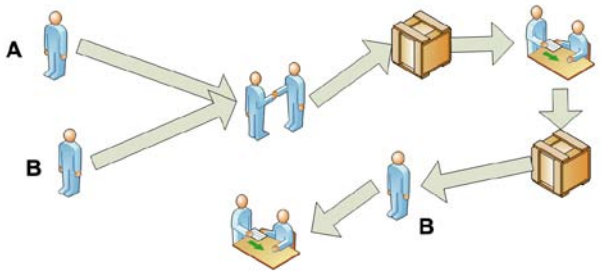
Use Case 2

1. Employee A is tasked for a job. He needs a tool in order to complete his task.
2. Employee A goes to the tool area and finds the tool.
3. Employee A scans the needed tool and his badge.
4. Employee A finishes his task but leaves the tool at his work area.
5. Employee B is tasked for a job. He needs the same tool as Employee A to complete his task.
6. Employee B goes to the tool area. Unfortunately, Employee B does not find the tool because Employee A never returned it.
7. Employee B goes to the computer and checks the log. He sees Employee A has the tool.
8. Employee B contacts Employee A and Employee B is able to start his task.

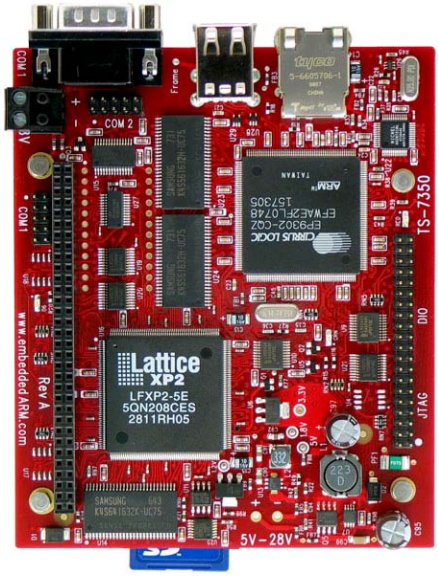


Use Case 3

1. Employee A and B are tasked for a job at the same time that require the same tool.
2. Employee A communicates with Employee B that he will take the tool first and then hand it to him after.
3. Employee A goes to the tool area and finds the tool.
4. Employee A scans the needed tool and his badge.
5. Employee A finishes his task and notifies Employee B and delivers the tool.
6. Employee B finishes his task and returns the tool.



TS-7350 Single Board Computer (SBC)
200MHz ARM9 CPU
32MB SDRAM (64-128MB opt)
5K LUT FPGA
Flexible 64-pin FPGA-PC/104 connector
8MB RAM Framebuffer, no video core
Able to drive TFT-LCDs via custom FPGA
1 10/100 ethernet port
2 USB 2.0 (12Mbit/s max)
1 SD Card slot (up to 6MB/s DMA)
3 RS-232, 2 TTL COM ports
40-pin header with ADC, SPI, I2C, DIO…
Fanless -40° to +70°C, +85°C 166Mhz 5-28VDC
Power Input Boots Linux 2.6 in about 1 second



pcProx USB RFID Reader
As a card enrollment or ID badge reader/tester, the USB model emulates a keyboard and will keystroke the card's ID and/or site code to the cursor's location on the screen. The reader can be configured to add keystrokes before and after the card's data.
It can be used as a standalone system, or be seamlessly integrated with other software applications using the optional SDK (software developer's kit). As an integrated reader you will find applications such as PC/LAN logon, single sign-on, card enrollment, visitor management, vending, point of sale, time/attendance, user authorization, seminar, fitness or training attendance, magnetic card emulation, and more.

How it Works

•The SBC runs Debian Linux with a 2.6 Kernel.
•Once it is powered on, it boots into Linux from the SD card.
•Using the mingetty package, an autologin to root follows.
•The root user has initialization scripts to start the program that detects the pcProx RFID reader. This program runs in the background.
•The program attaches to the character device input of the pcProx RFID reader and waits for two RFID tags detected within 10 seconds.
•The program then sends the two tags to another separate program that validates and handles the MYSQL database update.

debian

How it Works (cont)

•The Debian OS also hosts web services through Apache Server. The internal address is set to static. (192.168.1.122)
•The transaction log is accessed through this IP for internal tracking.
•The web GUI makes use of Asynchronous JavaScript and XML (AJAX) and PHP. With these technologies, the end user experiences more of an application type of interface.
•The GUI also refreshes itself live every 5 seconds.

How it Works (cont)

•The Debian OS also hosts web services through Apache Server. The internal address is set to static. (192.168.1.122)

•The transaction log is accessed through this IP for internal tracking.

•The web GUI makes use of Asynchronous JavaScript and XML (AJAX) and PHP. With these technologies, the end user experiences more of an application type of interface.

•The GUI also refreshes itself live every 5 seconds.

Reasons for Development

My project is solving a problem prevalent in many companies. Namely, these companies have many employees sharing limited tool assets. Companies that use a form of logging default to a form of a sign-in/sign-out paper sheet. This method appeared outdated to me. Digitizing this process has many benefits. Firstly, the company will be more "Green" by saving paper. Secondly, they will save physical space by having years of logging information on a compact device. Time will be saved by using a simple interface rather than sifting through mounts of paper. It will be quick and easy to determine if a tool is checked-out, and if so, who has taken it.

Many companies have limited tools due to their expensive nature. If an employee were to take a tool and forget to return it, he/she may be preventing another employee from completing their task. This scenario leads to a loss of company time and money. My project will solve this issue.