A significant issue in today’s world is the large amount of food being thrown out, after it has gone bad in an unconsumed state. Many consumers may end up buying products that end up in the back of the fridge, forgotten about until the fridge is cleaned and the item is discovered to be no longer safe to eat. According to the EPA, the average family of four spends approximately $1500 per year on food that goes to waste.[[1]](#footnote-2) In addition, expiration dates are often hard to find on many food items. For example, tiny writing may be stamped onto the bottom of the package, which may be difficult to read for individuals with impaired vision. Lastly, individuals must physically look at the expiration date for each item to recognize when the item is going to go bad or to realize that the item is already expired and unsafe to eat. This research project proposes an inexpensive, RFID-based system, utilizing a computer interface and an Android smartphone application, that can allow users to monitor and be notified of expiration dates on food items present in the fridge.

Radio Frequency Identification, commonly shortened to RFID, is an emerging wireless communication system that allows for identification using small “tags.” A few benefits of the tags include passive power with no batteries required, as well as the tag’s circuits being small enough to be printed on label printers or similar. An RFID reader (Figure 1) is used to read tags in a small radius, usually within a few feet. The reader sends power to the RFID tags and can retrieve the data that a tag may be programmed with. RFID systems have many applications and uses, especially in inventory management[[2]](#footnote-3). One example might be a grocery store’s freezer: within the freezer, all items have a unique tag, which can be used to track quantities, manufacture dates, and other important data. RFID is an emerging technology that provides many benefits for industry as well as consumers. This project utilizes a reader to obtain item data encoded on tags, which have been placed on some sample food items. RFID is proximity-based, so a typical reader can read multiple tags in a closed area, adding more convenience for individuals aiming to keep track of expiration dates on their newly purchased groceries.

In typical grocery stores today, a barcode, present on the packaging or an attached label, is used to scan and inventory items. In the U.S., the type of barcode used in most cases is of the UPC-A (Universal Product Code) type. However, the data that the barcode stores is referred to as the GTIN-12 (Global Trade Item Number) value. When a barcode scanner scans the UPC-A barcode, it will read the GTIN value, which is used to identify the item and its origin[[3]](#footnote-4). A GTIN value is made up of parts that contain information about the manufacturer, an item reference number, and a check digit[[4]](#footnote-5). The RFID tags used in this project store the GTIN to be able to obtain more information about a specific item, such as a detailed name and item photo.

To be able to write tags with RFID data, there are different options, ranging from a handheld data writer to dedicated label printers that are suitable for use in a commercial environment. In this case, an Avery Monarch ADTP2 label printer (Figure 2) was utilized to print labels that have barcodes printed on the paper itself, encoded with RFID data within. The specific labels used have thin, flexible RFID inlays that hold 96 bits of hexadecimal encoded data. The Avery printer used in this project can print labels and encode the inlays simultaneously, making it an invaluable tool in a grocery store or manufacturing facility. Nowadays in grocery stores, items use barcodes for identification. However, with printers becoming more commonplace such as the one used here, RFID tags could be used in more quantities of products for quick scanning and inventory both on the commercial side and the consumer side. The Avery printer can utilize batch jobs to quickly print RFID labels for hundreds of items all within a couple of minutes[[5]](#footnote-6). RFID printers such as these can help to incorporate RFID technology quickly and efficiently into manufacturer’s products.

Before delving into the details and procedures of the final implementation, it’s important to illustrate a picture of a possible scene in the not-so-distant future, in which RFID tags have become prevalent in grocery stores, and refrigerators have reader modules built within to read all items present within the appliance. A consumer goes to the grocery stores, and purchases a large load of groceries, with each item having a RFID tag with GTIN and expiration date. When at home, they begin transferring the refrigerated groceries into the fridge. As they put the groceries into the fridge, an RFID reader reads item data off each item and adds a full item name, photo, and expiration date to an account specific to that consumer. Lastly, using a smartphone app, the individual can view, in real-time, the items present in the fridge, as well as see expiration dates and generate recipes from selected items that may soon expire. This is the main design idea for this project, and it aims to reduce food waste by allowing individuals to monitor food items as they approach their expiration dates.

When encoding data into RFID hexadecimal format, there is a large variety of different formats designed by GS1, a group that is behind many barcode standards used in today’s world. The label inlays used in this project store 96 bits of data, which, when adhering to GS1 standards, cannot hold much more than a serial number and GTIN. Ideally, this project would be designed for the DSGTIN+ standard, which has provisions for a date value, serial number, and GTIN (item identifier). A manufacturer encodes the RFID tags with the expiration date, serial number based on production, and a GTIN representative of the specific item. However, due to the 96-bit limitation, the SGTIN+ standard was chosen, which can store a GTIN as well as a serial number[[6]](#footnote-7). In this project’s case, the GTIN was a padded value equivalent to the specific item’s barcode, and the serial number field was repurposed to serve as a YYMMDD format expiration date field.

Adhering to a tag data standard, such as the SGTIN+ format, requires some conversions to go between the hexadecimal value written to the tag and the readable GTIN and expiration date values. An existing tool, Avery Tag Data Standard 2.0 Demo, was used to confirm compliance with the SGTIN+ format. This tool allows the user to enter data for different RFID formats, and encode the data into hexadecimal, and vice versa. The tool also shows the bits utilized for each combination of data, which helped narrow down that the SGTIN+ format was required to be used to properly contain data within the 96-bit tags in this project[[7]](#footnote-8). In this project, a computer-based interface with a connected RFID reader is used to process detected tags. The interface is written in Java, and the reader outputs the RFID hexadecimal value as keyboard input whenever a tag is detected within the applicable proximity. It should be noted that an existing RFID-decoding repository was utilized as a foundation for this project’s decoding implementation. The existing code was pared down until it was in a form that would be able to return the GTIN, and expiration date from a hexadecimal RFID value from the reader’s keyboard input. For example, to get a GTIN from an SGTIN+ format RFID value, a binary conversion takes place, in which a set quantity of numbers is trimmed off the front to arrive at a binary representation that can be converted into a readable numeric string. Likewise, the expiration date is also extracted from the RFID tag’s data in a similar manner[[8]](#footnote-9). The SGTIN+ hexadecimal value written on a tag also contains headers preceding data, as well as check values (Figure 3).

After an SGTIN+ tag is converted into a readable GTIN and expiration date, the next step is to utilize a lookup of the item to provide a human-readable title and other information that a user can view. Initially, the item would be looked up on an item database designed for project usage only. However, it quickly becomes tedious to add many different items manually for project testing. Therefore, lookup APIs were considered, and the final implementation of this project utilizes a UPC lookup database (UPCItemDB) and the Java OkHTTP client to obtain more detailed information about a specific item from the GTIN value. When an item is scanned with the user’s phone or is recognized on the RFID scanner, the GTIN value is used to look up a detailed name of the item, as well as a matching item photo to display on the smartphone application. When looking up a general item, such as Jif Peanut Butter, the API response usually includes data from multiple shopping sources; for this project, Walmart data was selected to grab item titles and photos from[[9]](#footnote-10). Once these details are found, the item is added to a user’s unique database with GTIN, expiration date, purchase date, and other useful information.

During project testing, a laptop running Windows was utilized to run the Java-based interface. The RFID reader, as well as an Arduino Nano and DHT11 temperature/humidity sensor, were hooked up to gain relevant data on present items as well as track statistics within the fridge. Temperature and humidity are monitored continuously to allow the user to see a continuous readout, ensuring that their food products are at a safe temperature. Additionally, when an RFID item is added using the interface, a scheduled email is set up to send when the item is getting close to its expiration date. Emails are sent using an SMTP client and a Google Gmail account and are scheduled 7 days before expiration dates using the Java Timer system. Based on these reminders, users can take action to reduce household food waste. The other significant benefit is that the RFID reader can add data for multiple items automatically. In this project’s case, a task queue is utilized to ensure that when multiple items are scanned simultaneously, calls to the lookup API are being spread out. This helps to ensure that each API call works correctly without any overutilization errors. The interface reads for RFID keyboard input continuously, to capture any items that may be placed within the area. As for the Arduino temperature/humidity monitoring, serial communication over USB, using the jSerialComm library, is used to also read sensor changes continuously. Lastly, the computer interface writes to a database both the sensor changes, as well as scanned-in items using RFID. Continuous monitoring allows for item updates in near real-time.

To store user data, such as sensor changes, and owned food items, a cloud-based database was selected to share data from the computer interface and RFID reader to the smartphone application individuals can utilize to view owned items and fridge status. The database tool in question is part of Google Firebase, a cloud services platform that has real-time database functionality. This platform is also built with mobile apps in mind, which allows for easy integration into a smartphone app with value listeners that can update database changes with virtually zero delays. In an RFID environment, this is essential since many items can be added within a short period. Integrating Firebase within an Android app requires zero networking code, only a JSON configuration file, appropriate dependencies, and a few lines of code to specify the database reference to monitor/modify[[10]](#footnote-11). On the interface side, the Java Firebase Admin SDK is utilized to add food items to the database as required. The database, at its root, has a list of users, which then branches off to usernames. The next branch of the username values would be the MD5 hashed value of the password the individual chooses to sign up with. Once past that point, the database, for each user, stores a sign-up date, sensor data for the last 24 hours, as well as owned items by that user. Each owned item entry stores GTIN, name, purchase/expiration dates, image URL, and an identifier showing whether the item was added over RFID or not (Figure 4).

Most individuals in today’s world always have their phone on their person, which provides a great way for individuals to track what items are currently in their fridge, as well as the expiration dates of those items. The Android app utilizes a sign-in/sign-up account system to allow users to have their personalized item lists. Once a user has signed in with their login, the application has 3 tabs, each set up for a specific set of functions. The first tab, Owned Items, allows the user to view current items, that are either present in their fridge, or ones that have been added manually using the app (Figure 5). There is a clear synchronization indicator to set apart the RFID-monitored items. While one of the main proponents of the food-monitoring app is RFID-added items, users are also able to add items manually, that can be deleted later when they no longer have the item. Using Google’s GmsBarcodeScanner library, an external barcode scanner can be added to an Android app with just a few lines of code. Users can scan a barcode using the phone’s camera, which then utilizes the same UPC lookup API as the computer interface to get a detailed item name and photo to show on the item list (Figure 6). Compared to setting up a sample item reference database, this makes it easy to add data for thousands of items without adding each of them to a reference database manually. When adding items manually, the user can also change the purchase and expiration dates to the correct values, which are displayed on the item list tab. Using this list, individuals have a quick and convenient way to view expiration dates for their items and even add purchased items efficiently without utilizing the RFID interface.

The second tab on the Android app, Fridge Status, allows users to view real-time temperature and humidity values of the refrigerator, as well as see the change in those values over the past 5 hours (Figure 7). A toggle switch allows users to switch a graph between showing temperature change and humidity change, over the past 5 hours, up to the current hour. Users can ensure that their food items are at a safe temperature, usually around 40 degrees Fahrenheit. Additionally, in a future iteration, in which the computer interface utilizes backup power in the case of a power outage, users will be able to check the temperature of their fridge without opening the door and allowing cold air to escape, increasing the risk of spoiling food early.

The last tab on the Android app, Recipes, focuses on generating recipes that the user can make with selected ingredients (Figure 8). A list of items is shown, based on the current items that the user has stored on their account. The user selects what items they desire to make a recipe with, and enters any specific notes on what the recipe should be like. For example, a user may select peanut butter, breadcrumbs, and diced tomatoes, that they would like to make a recipe with. Users can select expiring items to help utilize these soon-to-be-wasted items together in one recipe. If the user selects the items described earlier, one suggested recipe might be spicy peanut butter noodles. The unique part about these recipes is that they are generated using AI, more specifically the same model that powers ChatGPT. Using the OpenAI API, a set prompt is used to generate 5 recipes with the selected ingredients and to include ingredients, instructions, and prep for each recipe. Additionally, each recipe is separated by a set string, which allows the application to separate each recipe in the user’s recipe list. Utilizing AI, an emerging technology, provides personalized recipes for each individual and their selected ingredients. Individuals can try new dishes by combining their expiring food items into different recipes. This encourages wasting less food, and instead utilizing it before it is destined for the garbage can.

Originally, the project was developed with NFC (near-field communication) tags instead of RFID tags. However, the opportunity to utilize RFID instead presented itself, and the technology stood out with far more ease of use. NFC requires individuals to tap an item directly on a smartphone, while RFID works within a proximity of a few feet. Additionally, RFID label printers can print barcodes with RFID data encoded on inlays, allowing for a clean appearance and easy application to existing food products. NFC, however, still has its advantages, such as the technology being incorporated into most major phones today, and being able to be used without a separate interface. Another design decision that was changed part-way through the project was the design of the temperature/humidity monitoring system. The initial idea was that a probe would be utilized, powered by a small rechargeable battery, which would monitor the sensor levels on a meat product within the refrigerator. However, using a central sensor allows for precise monitoring of the general temperature/humidity, and future iterations can allow for the sensor to be incorporated within a fridge, without batteries requiring to be charged periodically.

Most of the major challenges associated with this project were associated with designing the Android app. For example, multithreading must be utilized on resource-heavy tasks, due to the operating system terminating the app if there is no response within a few seconds. Additionally, the reason that Android notifications were not used in this project was that it was difficult to set up reliable alarm-based notifications that occur at a specific time, even when the app is not running in the background. While background notifications were able to be sent, sometimes they did not trigger until 24 hours from initiation and did not show a good degree of accuracy, making it hard to test to see if the notifications were working correctly or not. Also, as for the user account system, MD5 is not a secure hashing algorithm, and in many cases, a password can be cracked within a few minutes. In a future iteration, it would be vital to ensure that user passwords are being encrypted with a much more secure algorithm, such as SHA-512[[11]](#footnote-12). Another challenge was setting up and utilizing the Avery Monarch printer to print labels with RFID data encoded. A specific driver is required to be used for the RFID functionality to work, and the printer must be configured with the inlay specifications, as well as set up to work either as a USB printer or as a serial device. For this project, NiceLabel label design software was used to print barcode labels that also encoded the RFID data, allowing for easy printing and processing of such labels.

Adopting RFID into today’s grocery stores comes with its own set of challenges, as well. Manufacturers must find ways to add RFID tags within product packaging, which increases the cost of the item and adds new procedures to be followed when distributing items. The cost of the tags, as well as specialized printers, writers, and readers, is a large hurdle to clear before the technology comes into mainstream use. In addition, consumers must have the proper RFID interfaces to be able to take advantage of the tag’s benefits. Fridges might be built in the future with RFID hubs incorporated into them, but other individuals may not be willing to spend the money to buy a hub or new refrigerator. However, RFID is not an expensive technology to develop, and it is more than likely that prices will drop drastically over time, and as the technology improves in design. Users appreciate an integrated approach that requires little effort on top of typical routines, and RFID allows for seamlessly created lists of food items and expiration dates. This project provides a look into the future, of how food waste could be reduced significantly with a smart, inexpensive food-monitoring system that informs the user of important details and statistics anytime, anywhere from their smartphone.

Appendix

A black rectangular object with a blue circle and a white logo

Description automatically generated

A screenshot of a computer

Description automatically generated

Figure 3: SGTIN+ composition (data after the plus is not included in this project’s context)

Image Source: <https://github.com/wrightedu/Make-IT-Wright-2023/blob/main/presentations/Training-Session-2.pdf>

Figure 1: USB RFID Reader

Image Source: <https://www.amazon.com/Ajfwm-Scanner-13-56Mhz-Frequency-> Control/dp/B0CF2N2277

Figure 2: Avery-Dennison Monarch Label Printer with RFID Encoding

Image Source: <https://www.youtube.com/watch?app=desktop&v=s816i7gF1xE>

A screenshot of a computer

Description automatically generated

Figure 4: Example item entry on Firebase

A screenshot of a product

Description automatically generatedA jar of peanut butter

Description automatically generated

Figure 6: Adding an Item from a scanned barcode

Figure 5: Owned Items Tab, showing some RFID-enabled items

A screenshot of a recipe

Description automatically generatedA graph with numbers and a line

Description automatically generated

Figure 8: Recipes Tab, showing a generated recipe from some items

Figure 7: Fridge Status Tab, showing temperature and humidity

In addition to this research paper, the source code of this project is published at:

<https://github.com/tpw2033/Cost-Effective-Food-Monitoring-RFID>

Resources used are described there, in addition to within the footnotes of this document.

1. Preventing Wasted Food At Home, <https://www.epa.gov/recycle/preventing-wasted-food-home> [↑](#footnote-ref-2)
2. Radio Frequency Identification (RFID), <https://www.fda.gov/radiation-emitting-products/electromagnetic-compatibility-emc/radio-frequency-identification-rfid> [↑](#footnote-ref-3)
3. What is a GTIN?, <https://www.gs1us.org/upcs-barcodes-prefixes/what-is-a-gtin> [↑](#footnote-ref-4)
4. UPC Code GTIN-12, <https://www.barcode.graphics/gtin-12/> [↑](#footnote-ref-5)
5. Avery Dennison Monarch Tabletop Printer 2, <https://printers.averydennison.com/content/dam/averydennison/rbis/global/printer/product-information/printers/ADTP2-product-information.pdf> [↑](#footnote-ref-6)
6. GS1 US RFID Foodservice Implementation Guideline, <https://gs1ca.org/gs1ca-components/documents/GS1US-Foodservice-RFID-Guideline.pdf> [↑](#footnote-ref-7)
7. GS1 TDS 2.0 Information, <https://github.com/wrightedu/Make-IT-Wright-2023/tree/main/GS1%20TDS%202.0%20Information> [↑](#footnote-ref-8)
8. SaladMan, <https://github.com/Obscure2020/SaladMan/blob/main/Main.java> [↑](#footnote-ref-9)
9. UPCitemdb Documentation, [https://www.UPCitemdb.com/wp/docs/main/development/responses/](https://www.upcitemdb.com/wp/docs/main/development/responses/) [↑](#footnote-ref-10)
10. Firebase Realtime Database, <https://firebase.google.com/docs/database> [↑](#footnote-ref-11)
11. How long would it take to crack your password?, <https://spycloud.com/blog/how-long-would-it-take-to-crack-your-password/> [↑](#footnote-ref-12)