# Intellitory

Josephine Nghiem, Nico Shinozaki, Tamer Tamer, Barry Wang, Justin Zhang

## ABSTRACT

This capstone project describes an implementation of a shelving system that uses RFID to automatically and accurately track the number of items inside a drawer. The main goal is to replace an already-existing system that currently requires a substantial amount of user interaction. The main points of the project are counting the number of items in a drawer and updating an online database with the new information.

## INTRODUCTION

Accurately keeping track of the stock of items allows the company to efficiently reorder items when needed, allowing employees to continue their work without having to wait for new stock to come in. A system that does not require as much user interaction also allows employees to work without having a prolonged break in their work flow, which may result in ideas and plans becoming forgotten or mistaken.

This system leverages UHF RFID antennas and readers for precise item identification. A main microcontroller coordinates these inputs, communicates with a peripheral microcontroller for additional interfacing, and integrates seamlessly with an Aptitude server via the Google API. Data is stored and visualized in Google Sheets, accessible and manageable through a user-friendly web browser. Debugging and system diagnostics are facilitated through a dedicated debug terminal. This architecture minimizes manual intervention and errors, creating a robust, scalable, and user-centric inventory solution suitable for retail, warehouse, and industrial environments.

This paper focuses on the components and specifications of the implementation of this system in detail.

## SYSTEM DESIGN

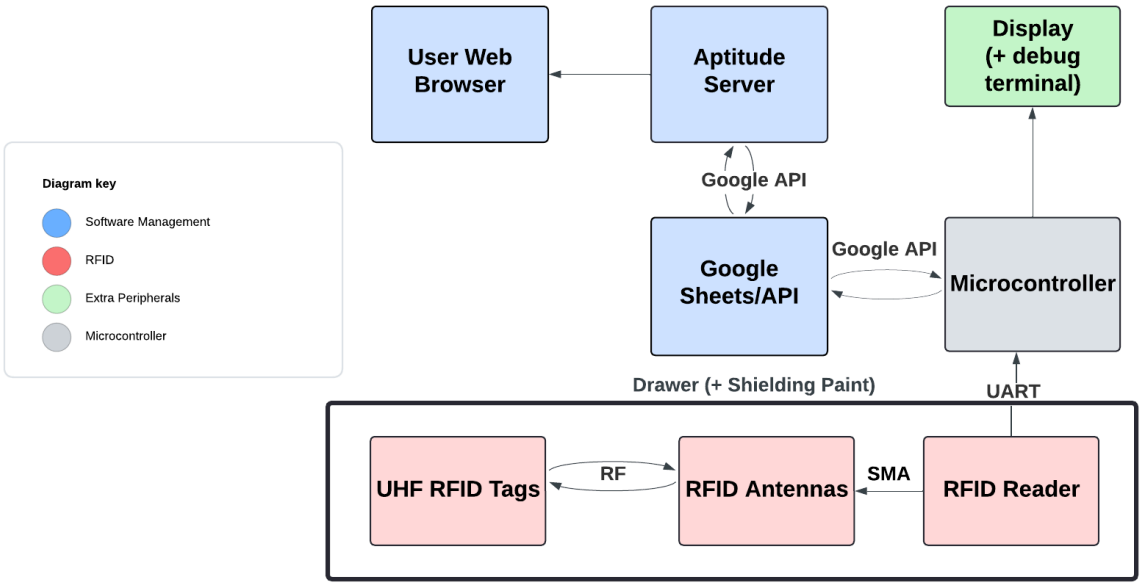


Figure 1. Full System Block Diagram

Our shelf system can be characterized into four different groups: software management, RFID, extra peripherals, and the microcontroller. As a general overview, the RFID group will detect any change in number of items in a drawer, and the microcontroller will send the updated information to a Google Sheets document through a Google API. The updated information will then be reflected on both the hardware display on the shelf as well as the website accessible through the Web.

### *HARDWARE COMPONENTS*

**UHF RFID Tags:** Radio Frequency Identification (RFID) technology operates in a number of frequency ranges with their distinct characteristics favoring their use in a number of applications. The three main frequency bands are Low Frequency (LF), High Frequency (HF), and Ultra High Frequency (UHF). LF (125–134 kHz) offers short read ranges up to 10 cm and good resistance to metal and water interference and is ideal for animal tracking and access control. HF (13.56 MHz) supports medium read ranges up to 30 cm and is also used extensively in applications like smart cards, Near Field Communication (NFC), and library systems due to its higher data rates and immunity to sensitive electronics. UHF (860–960 MHz), on the other hand, offers much longer read

ranges, typically between 1 and 12 meters, and is well-suited for fast, multi-tag reading of several hundred tags. For our smart shelving solution, UHF was the optimal choice and value option. It provides a longer-range, scalable solution that can read passive tags (no power source needed inside the tag) quickly and reliably, absolutely necessary for inventory tracking across multiple levels of shelves in a hospital or warehouse environment. UHF tags are also widely available and cheap, yet another factor supporting large-scale deployment.

**RFID Antennas:** In an RFID system, an antenna is used for both powering passive tags and receiving their data. When connected to the reader, an RFID antenna transmits a controlled electromagnetic field that powers up any UHF tags within its range. Once powered up, the tag uses a process called modulated backscatter to reflect its encoded data back to the antenna. The antenna passes the data to the reader to be processed. For our purposes, Vulcan's P11 and P12 antennae were chosen due to their versatility and performance, both of which are circularly polarized UHF antennae that provide for consistent tag reading regardless of tag orientation. This is particularly important in a dynamic shelving system where the products are not always stored evenly. The Vulcan P11 is ideal for focused, medium-range coverage such as individual shelf levels, while the higher-gain Vulcan P12 can be utilized to cover larger or more challenging spaces where greater signal penetration is needed. Together, these antennas provide stable and focused read zones within our shelving space, minimizing cross-read interference and facilitating accurate item tracking.

**RFID Reader:** Zebra FX9600 is an industrial-grade, high-performance UHF RFID reader that can be best utilized in smart shelving along with other asset tracking applications. Supporting up to 8 antenna ports, gigabit Ethernet, GPIO, and RS232 serial interfaces, it offers a scalable and flexible solution for real-time inventory management. Among the unique features utilized in our system is the Zebra FX Connect feature, which facilitates direct communication and configuration through the RS232 serial port. This is a valuable feature in applications where Ethernet connectivity is limited or where local device integration is essential. With the command-based GUI of Zebra, we can send ASCII or binary commands over the serial link to configure reader parameters such as the power levels for the antennas, region configuration, tag filtering, and read cycles. Such control implies that we enjoy excellent control over adapting the actions of the reader based on environmental conditions or operational needs. Additionally, the FX9600's ability to connect via Zebra's 123RFID configuration tool and its RESTful API enables remote monitoring, diagnostics, and software-based management of read operations.

Combined, they offer the FX9600 as a solid and reliable component of our smart shelving system for reliable data capture and seamless integration with back-end inventory systems.

| | Yanzeo SR682 | Zebra FX9600 |
|---|---|---|
| **30 tags** | **100%** accuracy, within **8s** | **100%** accuracy, within **<2.5s** |
| **40 tags** | **98%** accuracy, within **30s** | **100%** accuracy, within **<2.5s** |
| **96 tags** | **63%** accuracy, within **60s** | **98%** accuracy, within **2.5s** |

Table 1. Comparison of Yanzeo SR682 and Zebra FX9600 performance.

**Drawer:** Generic plastic drawers were purchased and coated with an EMF insulating paint on the inside to prevent leaking RF signals. Each drawer contains an RFID antenna mounted at the inner top wall facing the items inside the drawer. This antenna scans the RFID tags and sensed tags are collected by the Zebra FX9600 scanner. The scanner processes the raw signals and then reports to the raspberry pi via an UART connection. In order to accurately detect the number of tags in the drawer, tags must be positioned perpendicularly to the face of the antenna, and as such, the objects have been stored in a mold container to remain upright.

**Microcontroller:** A Raspberry Pi 4 functions as the central microcontroller, operating under the Raspbian OS distribution. Given that our application demands real-time responses ranging from seconds to several seconds, a time-sharing operating system sufficiently meets the requirements for effective resource management and offers substantial flexibility in software design. The Python runtime environment, established on Raspbian, hosts our custom software suite. This software suite features a Qt6-based graphical user interface (GUI), enabling users to view, edit, and synchronize inventory data in real-time with Google servers. Additionally, the GUI facilitates algorithm selection for data processing and includes an integrated debug console, which supports further development and internal data examination as needed.

**Display:** Our system uses a Santek touchscreen display designed to be used with a Raspberry pi. The purpose of the display itself is to allow employees at the shelf to interact with the database if needed. While the idea is to solely use RFID to accurately track the number of items, the reality is that the numbers provided by the scanner may not be entirely accurate, perhaps due to poor orientation of items within the drawer, which would result in tags interfering with each other and disturbing the RFID waves, or even

limited capabilities of the scanner itself. As such, as a failsafe, we have included the display in the situation that an employee needs to manually correct the numbers in the database. We chose to use a touchscreen display to remove the need for a mouse and keyboard at the shelf station.

*HARDWARE MANAGEMENT AND FLOW*

On a cold start, the RFID scanner must be powered on and operational before booting the Raspberry Pi. The initial handshake between the Raspberry Pi and the RFID scanner occurs during the Pi's boot sequence via Ethernet. Upon starting the Intellitory software system service, a Selenium-based web automation script accesses the scanner's management portal to configure the scanner into continuous scanning mode, outputting tag data via a UART channel. The management software collects this data at fixed polling intervals, parsing and processing the information accordingly. For instance, with a polling interval of 3 seconds, the software aggregates all reported tags within this interval as a single observation. This observation process continues uninterrupted unless the "warn on inventory change" option is enabled (requiring user confirmation to proceed after inventory changes) or the scanner driver is manually interrupted via the debug terminal.

The software provides three algorithmic options for processing tag data:

> *No Filtering:* Tags identified within the latest interval are treated as actual present states. This method offers rapid response times but reduced accuracy, suitable for non-critical or large inventories where approximation is acceptable.
>
> *LPF Window:* A discrete low-pass filtering method (sliding average window) is utilized. For example, a window size of three requires a tag to be present in at least two intervals to be confirmed as present. This method balances response speed and accuracy.
>
> *HMM-Viterbi:* Utilizing a Hidden Markov Model and the Viterbi algorithm, this method predicts actual tag states based on historical data. It includes a feedback mechanism that adjusts prior probabilities to enhance accuracy over time, providing adaptive and highly accurate tracking but requiring stabilization.

Upon detecting inventory changes, the management software notifies the GUI, prompting user confirmation if the "warn on inventory change" setting is active. Users can choose to

accept, reject, edit, undo, or redo these changes. Depending on the configured settings, the system may automatically synchronize these changes with the Google servers.

*WEB INTERFACE MANAGEMENT AND FLOW*

**Website:**

The Smart Shelving Inventory Dashboard enables employees and admins to securely view real-time stock levels from anywhere. Staff simply log into their account to access up-to-date inventory data, ensuring transparency and helping prevent stockouts

*Overview:*
Our website is deployed through Firebase, a Google Cloud deployment service. Firebase essentially serves as a version control and deployment service that makes managing frontend, backend, and databases more streamlined.

*System Design:*
Our website consists of 3 main components: Frontend, Backend, and a Database.

1. Frontend

Our frontend is created through React.js. It is one of the main libraries used in industry for creating website GUIs. The frontend is simply the user interface for our website. It handles routing between different web pages and interpreting user input. It solely talks to our backend, making API calls to it whenever it desires information of any kind, such as, login authentication and inventory data. We store authentication tokens, such as JWTs, securely after login and attach them to subsequent API requests. Additionally, CSRF tokens are included in requests that modify data to prevent cross-site request forgery attacks. All communication between the frontend and backend is conducted over HTTPS to ensure secure transmission of sensitive information.

2. Backend

Our backend is created through Javascript. It is also the industry standard for creating website backends. The backend is responsible for authenticating and handling requests for data acquisition and manipulation. If our frontend ever needs to access or change data, it must make requests through the backend. The backend will authenticate and verify these requests and react accordingly. The backend talks to both the frontend and the database.

3. Database

Our database is a PostGres database. It is one of the more popular database formats, known for being extremely flexible and powerful. In our use case, we have this database for storing user credentials. Upon a login or register request from the database, the database will return or add entries from the database accordingly.

## CONCLUSION

Overall, the system is able to detect changes in the number of items in drawers and send the information to an online database, which is referenced by the on-shelf display employee-accessible website. Our implementation shows that it is possible to automate inventory-tracking small-scale storage given a microcontroller and RFID devices.

## ACKNOWLEDGEMENTS

We would like to thank the Department of Electrical and Computer Engineering at UCSB and Sheng-Ping Liang and Qin Yang at Aptitude for providing us the opportunity to work on this project. We would also like to thank Lucas Moreira for his technical assistance.