THE UNIVERSITY OF IOWA
College of Engineering

ECE:5550 Spring 2022

IoT Final Project

JAMS Pantry
(Smart Pantry with Web Application)

Juliana Danesi Ruiz

Ashley Mathews

Matthew Hall

Samuel Murphy

Project Background and Motivation

It is extremely common to find expired, rotten or moldy food in almost everyone's pantry. Many times, it is hard to tell whether or not an item is still edible. JAMSPantry is able to alert the user of products that are about to expire and provide information about the user's pantry in real-time. Anyone can navigate to our web application, sign in, and the dashboard will display vital data to the user about the contents of their pantry.

The motivation of this product was to design a Smart IoT Pantry that is able to scan products and store them in a database to keep a record of product information and to alert the user when the temperature or humidity of their pantry threshold exceed the recommended value, or when food products are about to go bad. Whether you forget your grocery list at home or want an updated count of stock in your cupboard, JAMSPantry does just that with our web application.

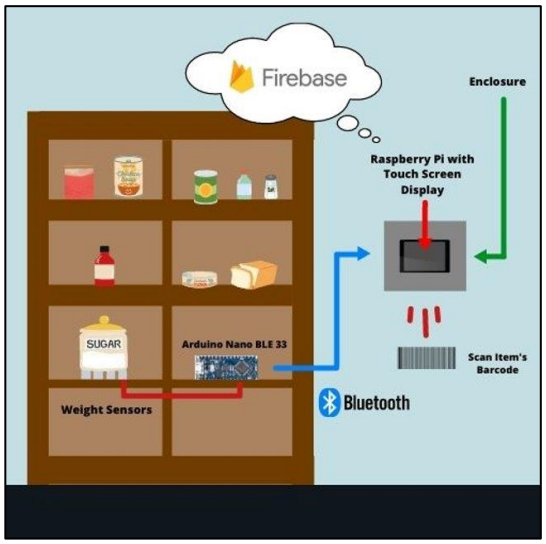

Figure 1. The idea

Overview of System Diagram

Our project is composed of two main parts, a Bluetooth connection, and a Web application. Figure 2 shows the complete overview of our system diagram. The Raspberry Pi has the "NodeJS Webapp" and the "NodeJS Local App" running simultaneously. "NodeJS Webapp" displays the Web application on the Touch screen, and the "NodeJS Local App" connects the Pi to the Arduino and sends data to the Firebase. The only wired connection to the Raspberry Pi is the Barcode scanner, which has a USB cable. After the food product is scanned, the barcode monster API is used to identify the product name and type. On the other hand, the Arduino is connected to the Load Cell, which gathers the weight of the food product on a container (Ex. Rice, Flour, Oats, Cereal…). The scale is only used for these products because they are normally placed into a secondary container such as a Mason Jar. The weight scale allows us to know how much product is left on the Jar.

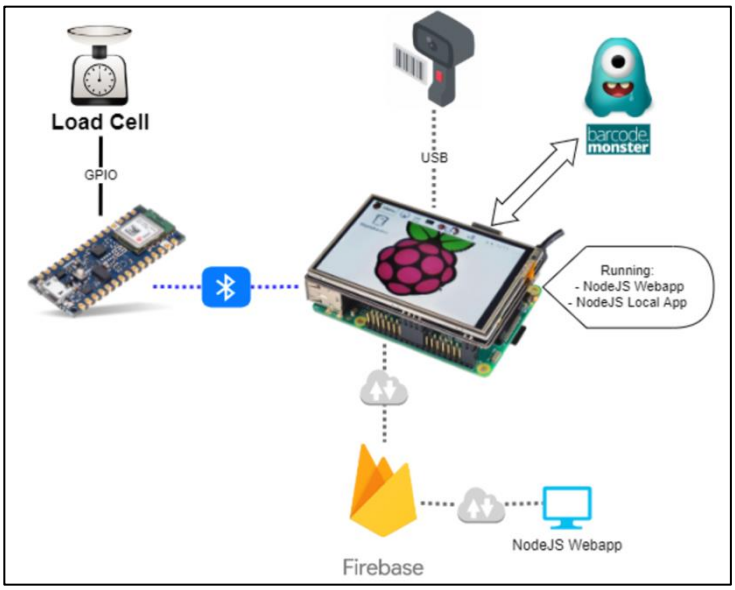

Figure 2. System Diagram

Technical Aspects

Sensors and Communications

Figure 3 demonstrates the data flow from the Arduino Nano to the Firebase. To gather Humidity and Temperature data, the built in Arduino sensors were used. Temperature was measured in Celsius, and Humidity measured in relative humidity (%). On the other hand, the weight of the container was measured using an external sensor, which was one of the group purchases. After calibrating the weight scale, we were able to get the accurate weight in kilograms. Then, temperature, humidity, and weight are sent to the Raspberry Pi via Bluetooth Low Energy (BLE). The values are sent using the ESS service and GATT specifications. The Raspberry Pi does some processing to those values, such as a mean filter and then the data is sent to the Firebase.

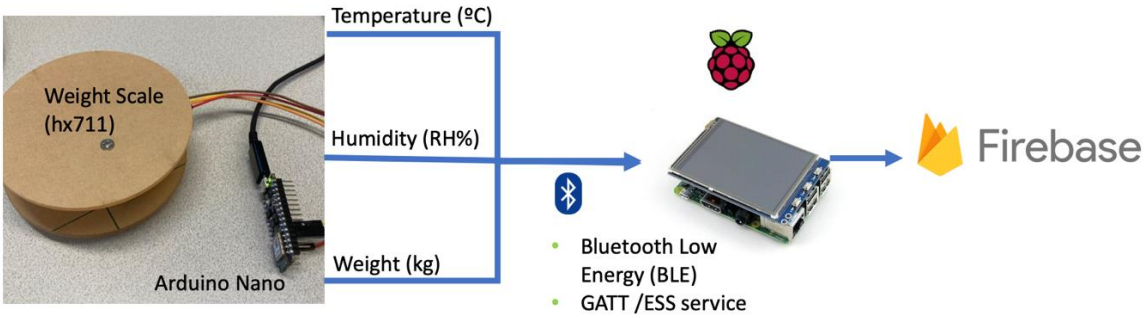

Figure 3. Data flow Arduino to Firebase

Security

When it comes to IoT devices, security and privacy become a major concern to protect sensitive user data. Our project uses Firebase's built-in authentication to secure account information pertaining to one's individual pantry. Each user has a unique username and password to login to their account. The figure below shows a systematic overview of how our Smart Pantry operates. First, one can sign in on the Kiosk by remote login which is done on any mobile device. The user can access the JAMSPantry Web App after scanning the QR code. Once at the homepage, they may sign in with their credentials or create a brand-new account where a UID (user ID) is generated and stored in the database. Firebase's built-in authentication allows us to address the security and ethics of this project entirely. As long as Google's database remains secure, we can ensure the privacy of our users.

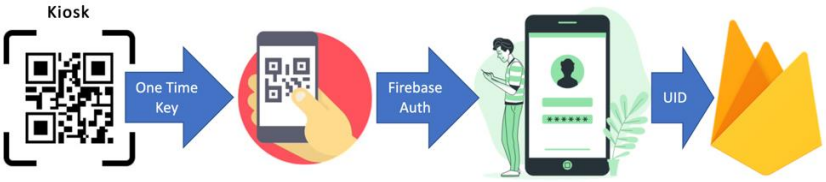

Figure 4. Systematic Overview

Cloud-based Storage

To build our IoT Smart Pantry Google's Firebase database was used to store information such as user ID, keys, the pantry environment including temperature, humidity, and weight, as well as the pantry inventory. Figure 5 shows the Firebase structure, each user has an "Environment" and an "inventory". The inventory contains all the food products the user has on their pantry. In addition, the environment section contains the "Humidity"," Temperature", and "Weight" data. The weight is gathered in 2 seconds intervals while humidity and temperature are gathered in 1 min interval. Since we wanted to have a graph analytics on humidity and temperature, those two variables have 360 values in a list, thus totaling 6-hour data. On the bottom of the Firebase structure, "keys" is the all the active one time keys for remote authentication.

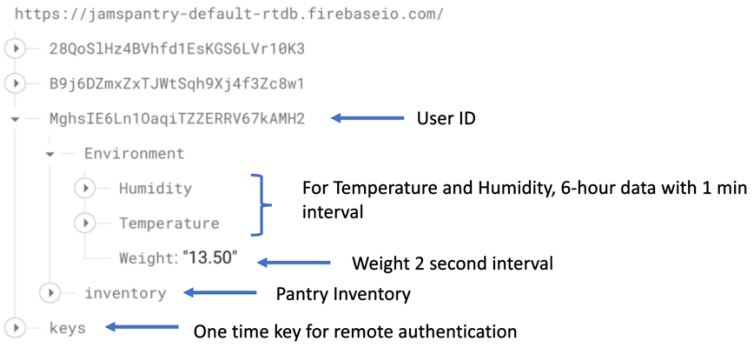

Figure 5. Displays Firebase Data for Each Individual User with Pantry Inventory and Environment

Analytics and Action

The following figure illustrates the temperature in degrees Celsius and the relative humidity of the pantry within the last six hours. As the environment changes, this data is updated in real-time on the analytics page.

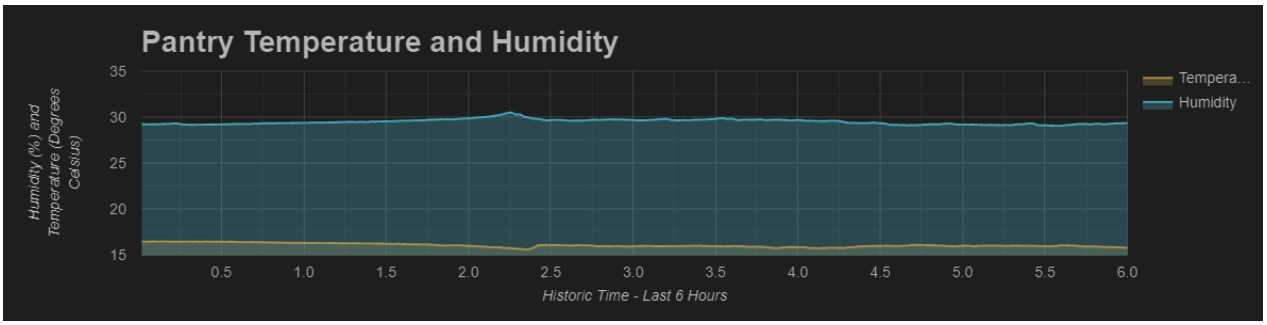

Figure 3. Temperature and Humidity Environment

The quantity of products in varying time ranges is displayed in Figure 4. The time ranges are expired, expiring in the next 1-2 months, next 3-5 months, and 6+ months. Each quantity is the total number in each range and changes automatically as time passes.

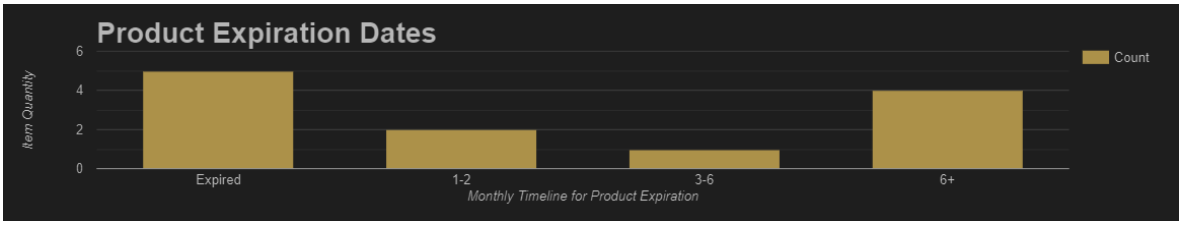

Figure 4. Monthly Timeline of Product Expiration Dates

The figure below shows the mass of the container as a percentage. However, in the image provided, one may notice the value is –1. This indicates that an object is not placed on the scale, so no weight is being sensed. The container capacity is displayed as a percentage of the total possible amount which can be stored in the container. If the sensor reads less than 1%, an empty warning is displayed on the dashboard and when less than 10%, a low quantity warning is displayed.

Figure 5. Relative Mass of Container Displayed as Percentage

The image below displays to the user the total amount of items that are scanned into their pantry. When an item is discarded or scanned out of the pantry, this number will be decremented. However, if an item is scanned into the pantry, the count is incremented. Currently, the number of items in our inventory is twelve.

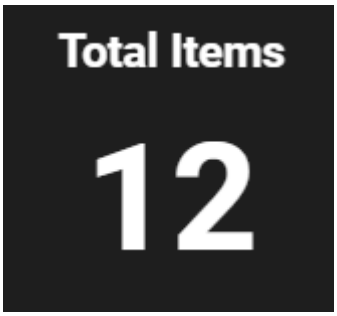

Figure 6. Total Quantity of Item in the Pantry

The figure below is the Home page which provides the user with warnings about the pantry. For the expired food and food expiring within 1-week warnings, they are based on the current day and update as food is scanned out or scanned in. The humidity and temperature warnings are threshold based and provide warnings when the average of the last couple samples passes those thresholds. The weight warning has two different messages that it can display. The first message being when the container is empty and the other being when the container is almost empty.

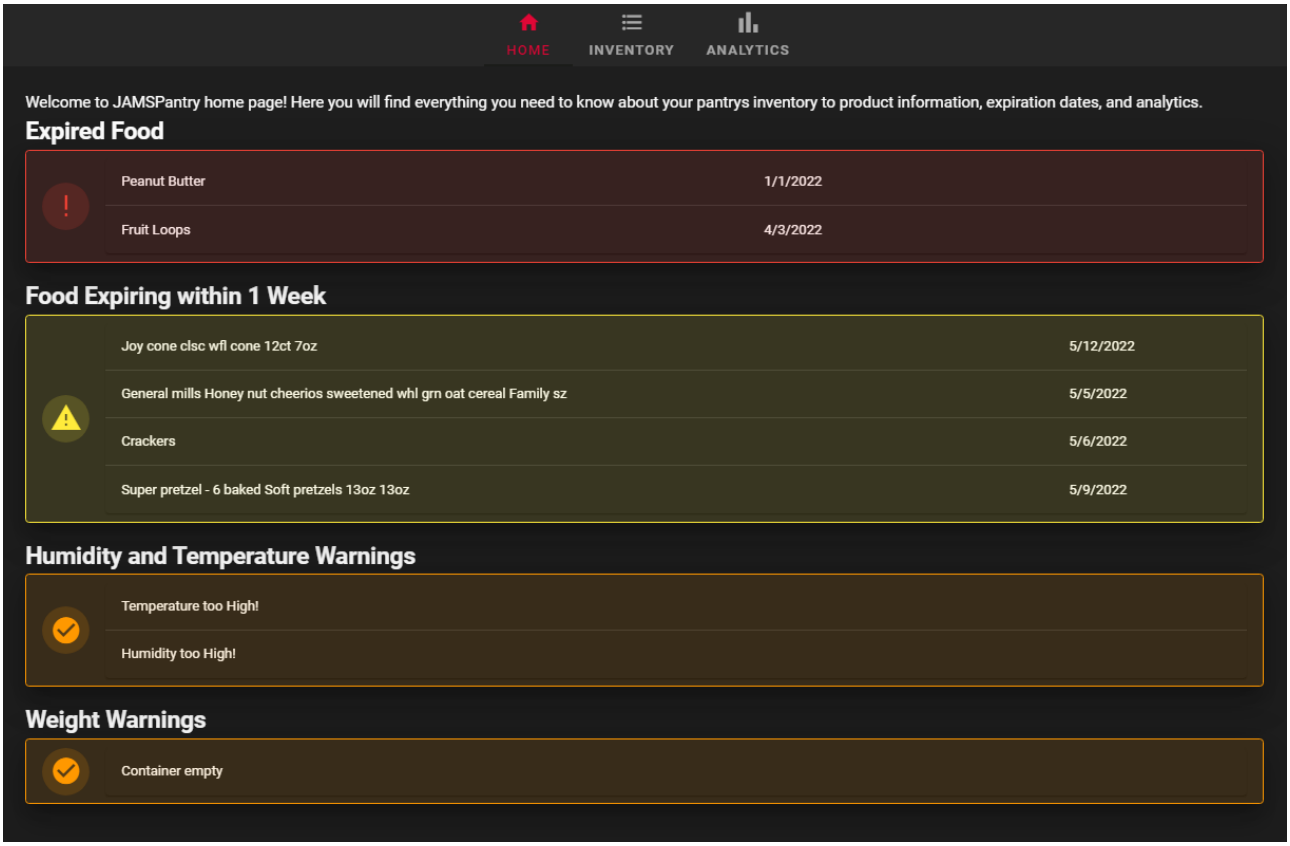

Figure 7. Home page where different warnings are displayed

Issues encountered

Along the way, the team encountered issues with the Barcode API that was implemented. Since Barcode Monster API is free, there were limited products available in their database, making scanning products tedious. Some products were undetected or not found so a description was not displayed, making our project not as user-friendly as we anticipated. To bypass this, we could have used a different API, one that is not free, allowing more products that could be scanned in. Although this roadblock made parts of the project challenging, JAMSPantry was still able to deliver.

Originally, the team wanted to use a Raspberry Pi Camera with high resolution to capture a video of the product and use object detection to acquire and scan the barcode. However, since the Pi Camera does not have the ability to autofocus, it made scanning 1D or one-dimensional barcodes impossible. Although the Pi Camera does works scanning QR codes, it was not a feasible option for 1D barcodes since it lacks autofocusing. Due to time constraints, users do not want to be waiting for an item to be scanned in, so it was necessary to troubleshoot and work around this barrier. The group decided to use a barcode scanner for scanning 1D barcodes instead. This was connected via USB to the Raspberry Pi which worked in conjunction with Barcode Monster API to query information about an item.

Discussion of Group Member Contribution

Overall, the group worked well together and was able to divide up tasks and be efficient with our time. The following is a breakdown of individual focus areas of team members:

Julie: Focused on the local node app for environmental sensing with the weight, temperature, and humidity sensors and sending that data over Bluetooth.

Ashley: Primary focus was on the analytics tab of the home page of the web app but also worked with Julie on the environmental sensing part.

Matt: Focused on the architecture of the Node JS App and on the Kiosk Page.

Sam: Focused on the inventory and home tab with alerts of the web app dashboard.

References

Google Charts was used as a reference for querying our information from firebase and displaying our real-time results to our analytics page. The code implemented for our project is on the Google Charts website under the tab Chart Types: AreaChart, Column Chart, and Guage. These various graphs were used to show product expiration data trends, comparisons between the temperature and relative humidity of our pantry, as well as show a percentage of the container's mass.

Link: Using Google Charts | Google Developers

The Barcode Monster API was used in our project to query information pertaining to a particular product that was scanned in via our USB barcode scanner. Barcode Monster is a free application programming interface that offers a limited amount of product information stored in a database.

Link: barcode.monster API: How To Use the API with Free API Key | RapidAPI

The Vue JS web application was built using the Vuetify Framework. This made the web app have a uniform and consistent look across all pages and made implementation of various components much easier.

Link: https://vuetifyjs.com/en/

JAMSPantry Links

Web Application: https://jamspantry.web.app/

GitHub Page: https://github.com/ashpmath/JAMSPantry

Figure 8. Final Product Prototype

Parts List:

Item:	**Cost:**	**Quantity:**
Raspberry Pi	Provided	1
Arduino Nano BLE 33	Provided	1
Hardware Enclosure (ABS Plastic)	$9.99	1
Internet Accessible Device (Phone, Tablet, Laptop, etc.)	Provided	1
USB Barcode Scanner	$10.79	1
Raspberry Pi Touch Screen	$40.00	1
MakerHawk Digital Load Cell Weight Sensor HX711 AD Converter Breakout Module 5KG Portable Electronic Kitchen Scale for Arduino Scale	$13.00	1