Smart Food Checkout System

Applying the Internet of Things to the Canteen

Derek Feehrer
Ubiquitous Computing Laboratory
HTWG Konstanz
Konstanz, Germany
derek.feehrer@htwg-konstanz.de

Isa Usmanov
Ubiquitous Computing Laboratory
HTWG Konstanz
Konstanz, Germany
isa.usmanov@htwg-konstanz.de

***Abstract*—This project involves creating an internet-connected assistant for canteen checkout lines. This model uses a networked scanner to scan food items and send data to a server, which processes the data and send information to a web app to be displayed on a smartphone. The app displays the net statistics for all scanned items about nutrition, budgeting and allergy warnings.**

***Keywords*—RFID; Web app; assistant; checkout; tray**

I. Introduction

Nowadays, people track their eating habits like never before. Some desire to track their nutrition to reach fitness goals, while others with dietary restrictions need to closely monitor their foods to avoid allergens or other prohibited ingredients. Many also want to track their spending habits and stick to a budget when buying food. In current checkout line scenarios, such as those found in university cafeterias, it is often difficult to quickly find and utilize the nutrition information for all the foods that make up a meal. It can also be difficult to remember the spending habits of past days and to stay on track with a budget. Furthermore, customers with dietary restrictions and severe allergies need to constantly pay attention to the ingredients lists to ensure that their meals do not contain anything they cannot eat.

In the age of the internet of things, difficulties like these can now be solved by augmenting the traditional checkout line experience with new smart devices that provide customers with insights into their meals and streamline the checkout process.

We assume that the reader knows about basic concepts of web development, embedded software and digital circuits.

II. State of the Art

Currently, many of the components to solve this problem exist as separate entities, but are not packaged in a way that allows for their separate features to interact and coexist in one simple, cohesive system. For example, food tracker apps that enable one to look up nutrition data for various foods, exist [8]. These applications often have large databases of nutrition information and diet features that allow individuals to track their eating habits, count calories and more, but are not at all integrated with the actual process of purchasing food, so they require extra effort and time on the part of the customer. Several popular allergy alert apps exist as well, but they require the user to scan the barcode of each item to search for allergy information, which again, requires extra time from the user [9]. While budgeting apps also exist, most are rather complex and go beyond the scope of simply providing the key metrics on the user's mealtime spending habits.

If these separate concepts could be combined into one system which automatically tracks a user's habits, it could make them exponentially more usable and allow users to gain valuable insights into their meals. What is needed is a new form of streamlined checkout system that both allows for quick purchases and also provides customers with an appropriate level of information on nutrition, budgeting and dietary warnings. This must be done in a totally intuitive and automated manner which does not require any extra effort on the customer's part. It should not slow down the overall process of waiting in a checkout line, but rather should increase efficiency.

III. Model

This type of checkout line assistant, has 3 basic components: a product scanner, which is connected through the internet through a microcontroller, a back-end server which is integrated with a database, and a mobile application on a smartphone, which displays information to the user.

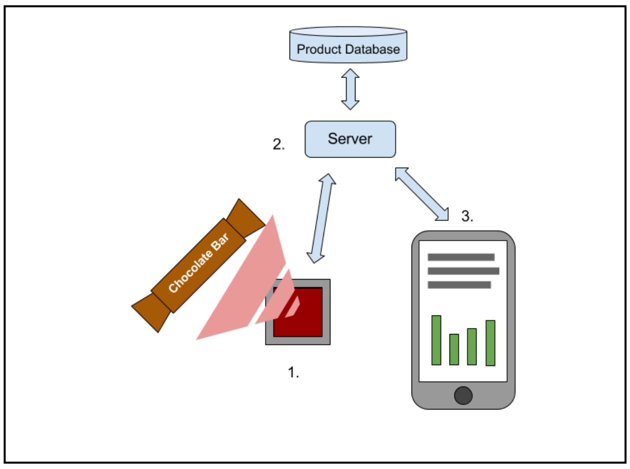


Fig. 1. X-Tray model

The user scans the items they would like to purchase and the scanner communicates this information back to the server. The server then retrieves the information about these scanned items from the database, sums up the net information on nutrition and cost, and then sends the information to client. The client displays the information to the user, along with any dietary restriction warnings, through the mobile application interface.

IV. Implementation

A. Hardware implementation

The hardware prototype which was created, named the X-Tray, was realized in the form factor of a lunch tray which scans items that are placed on its surface. This offers customers an inside look ("X-ray view") into what they are putting on their plates.

RFID, or Radio-Frequency Identification, was chosen as the means of scanning products because it is contactless, fast, and does not require the items to have any specific orientation to be scanned, in contrast with other possible methods, such as QR code or barcode scanning. This method does require, however, that all products are cataloged ahead of time and affixed with unique RFID tags. For this prototype, RFID-TP 35 model tags with 40-bit serial numbers were attached to each product [3]. After the items are tagged, an RFID reader can read the tags and send the unique identification number of products to microcontroller. The reader component chosen for this prototype was an ID20-LA - 125kHz RFID reader [2]. This component has a 180mm reading range making it sufficient for reading tags slightly above the surface of the tray.

The reader is one part of the scanning mechanism. It is connected to a Particle Photon board - a hardware development kit for the Internet of Things [1]. The Photon is a Wi-Fi-enabled microcontroller platform which transmits the data to the server. It powers the ID20-LA and waits for scanned tags over a serial connection. The Photon itself is powered by a Lithium-ion battery bank over USB. Pin connections between the Photon and ID20-LA reader are as on Fig. 2.

TABLE I. Pin Connections

ID20-LA Pins	Photon Pins
VCC	VIN
GND + FORM	GND
D0	RX

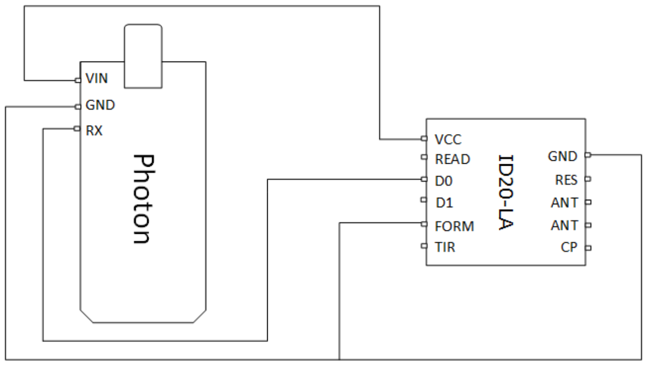


Fig. 2. Connection between the Photon and RFID reader

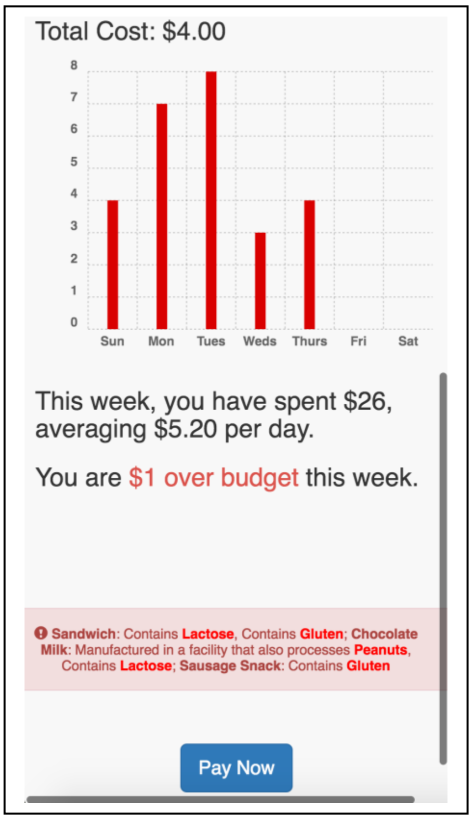
Arduino compatible C code is deployed to the Photon via the Particle Web IDE [4]. The Photon polls continuously, waiting for a tag to be scanned and then sends any scanned tag's identification number wirelessly to back-end server using Particle Webhooks. The Webhook allows the Photon to send the HTTP POST request to the server. In body of request, the identification number of the product is encoded in Javascript Object Notation (JSON), as follows, with the event value being replaced by the ID which is scanned: {"the_id": "{{PARTICLE_EVENT_VALUE}}"}.

The tray for prototype of the X-Tray is made of two standard lunch trays turned face to face. The scanner and battery are firmly fixed between them. On the surface of the tray, on the top-right corner, there is a scanning mark drawn. To add an item for purchasing, the user simply picks up an item, passes it quickly over the mark, and sets it down on the lunch tray. To remove an item, the user simply passes the item again over the mark, and places it back on the shelf. The usability of RFID scanning, when combined with the natural form factor of the lunch tray, creates a very intuitive experience, as the simple action of adding items to a lunch tray is one that all canteen customers are familiar with.

B. Software implementation

The decision was made to use a Web application for this project, rather than a native mobile app. The main benefits are compatibility across different operating systems and the ability to provide instant updates to users without the need to download a new app. This was seen as the best starting point for prototyping the software, which could later be made into a native application.

The back-end server software for this project was implemented using Node.js [7], a back-end JavaScript runtime built on Google Chrome's V8 engine. The Express framework was also used to streamline the app development process. The server in this case, acts mainly as an API to the client, but also performs some calculations, with the goal of distributing the work across the back-end and front-end, and sending only relevant data to the client.



This week, you have spent $26, averaging $5.20 per day.

You are $1 over budget this week.

Sandwich: Contains **Lactose**, Contains **Gluten**; **Chocolate Milk**: Manufactured in a facility that also processes **Peanuts**, Contains **Lactose**; **Sausage Snack**: Contains **Gluten**

Fig. 3. Screenshot of the X-Tray web app (Budgeting Chart)

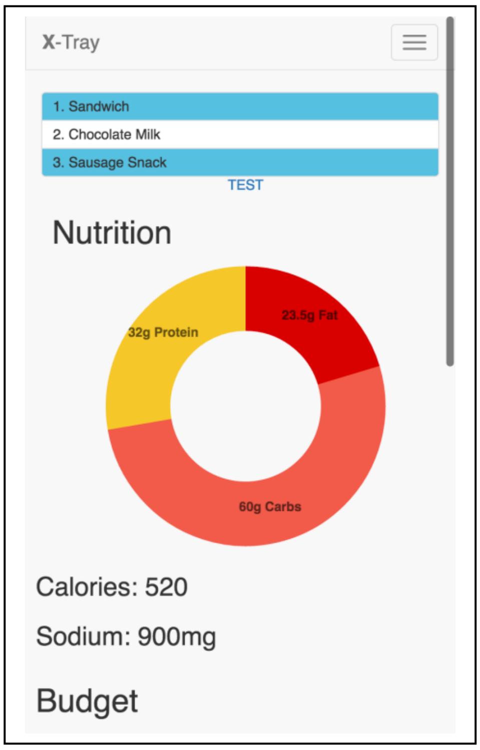
The database used is a cloud-hosted MongoDB [5] database. This database contains nutrition and cost information for all available products, as well as data on any allergens and other restricted ingredients the product contains. It also stores the user's preferences for dietary restrictions to track, their target daily budget, and data on their past spending habits.

The server contains an HTTP POST route which the microcontroller uses to send requests containing the identification numbers of products that have been scanned. When the server receives the POST request, it parses the identification number from the body of the request and uses that to look up and retrieve the corresponding product information from the database. It then uses this information to calculate the net nutrition and cost data of all the items that are active in the cart. The server also contains GET and POST routes for exchanging information on the user's account settings.

The server maintains an array which contains the data of all currently active products. When a new product is scanned, the server checks to see if the item is already in the array of active items or not. If the item was already active, then this means the user wishes to remove the item from their tray, so the product data is removed from the array. If it was not previously active, then the user is performing an addition and the product data is therefore added to the array.

The real-time communication between the server and client is done through a websocket, using Heroku's own websocket service. This enables the client to always stay up to date with the current state of the tray. The server sends the JSON objects containing all relevant data through the websocket, to the client.

On the front-end, the Javascript websocket client receives and parses the JSON objects and then passes the data to the Javascript charting library to display on the screen. The library used is Chartist.js [6], which creates responsive, animated SVG charts for displaying the data in a visually appealing way. A "donut chart" is used to display the net nutritional values of all the items in the tray, and a bar chart is used to show how the cost of the current meal compares to that of previous days in the week as shown in Fig. 3 and Fig. 4, as well as the user's progress towards meeting their budget. Thanks to the websocket communication, the graph visualizations react almost immediately after a product is scanned. This allows the user to quickly try out different combinations of products to see which ones best fit their goals. The frontend also utilizes Twitter's Bootstrap framework to provide a responsive, mobile-first interface that makes the web app look attractive on a smartphone.



Calories: 520

Sodium: 900mg

Fig. 4. Screenshot of the X-Tray web app (Nutrition Chart)

For simplicity in implementing this software prototype, the system was constructed with only one global user in mind, but if the system were to be implemented in real life scenarios it would require secure accounts for each individual user. Real purchasing of items is another feature which would be implemented in real life.

All features of this prototype were successfully implemented. The tray hardware, server, and client app all communicate together without issue. The web app reacts almost instantly when items are added and subtracted.

When testing the usability of the system on volunteers, the test subjects quickly learned how to add items to the X-Tray and make purchase decisions based on the information provided by the charts. The X-Tray was shown to provide real value without hindering the normal process of selecting meals. Some of the difficulties included separating which calculations to perform on the front-end and which to perform on the back-end. Future improvements could definitely be made to minimize the amount of data exchanged and the amount of processing power required by both the client and server.

V. Conclusion

As the internet of things grows, devices like these will continue to augment everyday processes, opening up new doors for simplifying the lives of people everywhere. All the above mentioned ideas led to creation of prototype of the Smart Food Check-out Assistant - the X-Tray. It includes the basic and most crucial features of this concept for a checkout line assistant. One of the main goals of the project was to open to audience a new development area for ubiquitous computing that can help people with budgeting, food tracking and could save a non-trivial amount of time.

VI. Future Work

The system was built with a focus on maximizing extensibility so that many more features can be added in the future. In future models of the X-Tray, the surface of the tray itself could be a high resolution touch screen which could display the visuals to the user in a more interactive and engaging way that the smartphone alone. Adding more RFID sensors, or a motorized sensor which polls the surface of the tray could eliminate the need to pass items over the specific mark on the tray. This could open up the smart tray as a platform for further developments such as suggesting new foods based on the information it learns about user's spending habits, dietary restrictions, nutrition goals, and what other customers with similar taste profiles are buying. A meal rating and feedback function could also be added so customers could provide feedback to the staff on their dining experience.

Furthermore, the future tray could employ NFC to allow users to quickly sign into their profile by just placing their phone on the tray. They could also make a quick payment via a native app on their phone, using Apple Pay, Google Wallet, and more. Having multiple of these smart trays in checkout lines in university canteens and other restaurant scenarios could be a cost effective way to cut down on long lines of customers waiting to check their products out with an attendant.

Additionally, the system could be made even more complete as RFID technology improves. It could even provide theft protection. Once a customer has paid for their items, the system would "disarm" the products. Then the customer would simply pass through an RFID detector at the exit of the line. This would look similar to a metal detector but would employ a longer range, ultra-high frequency (UHF) RFID reader, that would ensure all the products on their tray have been paid for. X-Tray is an extendable project where the only limit for additional functionalities for the trays of future is human's imagination.

References

[1] Elektronik und Technik bei reichelt elektronik günstig bestellen. (2016). RFID-TP 35 - RFIDTransponder-Schlüsselanhänger.[online] Available at:http://www.reichelt.de/RFID-TP-5/3/index.html?&ACTION=3&LA=446&ARTICLE=67319&artnr=RFID-TP+35&SEARCH=RFid [Accessed 6 Jun. 2016].

[2] Sparkfun.com. (2016). RFID Reader ID-20LA (125 kHz) - SEN-11828 - SparkFun Electronics. [online] Available at: https://www.sparkfun.com/products/11828 [Accessed 6 Jun. 2016].

[3] Particle. (2016). Particle Store. [online] Available at: https://store.particle.io/ [Accessed 6 Jun. 2016].

[4] Docs.particle.io. (2016). *Particle*. [online] Available at: https://docs.particle.io/guide/getting-started/build/photon/ [Accessed 22 Jun. 2016].

[5] MongoDB. (2016). *MongoDB for GIANT Ideas*. [online] Available at: https://www.mongodb.com/ [Accessed 23 Jun. 2016].

[6] Gionkunz.github.io. (2016). *Chartist - Simple responsive charts*. [online] Available at: https://gionkunz.github.io/chartist-js/ [Accessed 23 Jun. 2016].

[7] Nodejs.org. (2016). *Node.js*. [online] Available at: https://nodejs.org/en/ [Accessed 23 Jun. 2016].

[8] Healthline. (2016). *The Best Allergy Apps of 2016*. [online] Available at: http://www.healthline.com/health/allergies/top-iphone-android-apps#3 [Accessed 23 Jun. 2016].

[9] D'Cruze, R. (2013). *7 best diet apps for iPhone and Android*. [online] TechRadar. Available at: http://www.techradar.com/news/world-of-tech/7-best-diet-apps-for-iphone-and-android-1161222 [Accessed 23Jun.2016].