5-2022

Using Bluetooth Low Energy and E-Ink Displays for Inventory Tracking

David Whelan

Using Bluetooth Low Energy and E-Ink Displays for Inventory Tracking

An Undergraduate Honors College Thesis

in the

Department of Computer Science and Computer Engineering
College of Engineering
University of Arkansas
Fayetteville, AR

by

David Whelan

Contents

List of Figures

Abstract

The combination of Bluetooth Low energy and E-Ink displays allow for a low energy wireless display. The application of this technology is far reaching especially given how the Bluetooth Low Energy specification can be extended. This paper proposes an extension to this specification specifically for inventory tracking. This extension combined with the low energy E-Ink display results in a smart label that can keep track of additional meta data and inventory counts for physical inventory. This label helps track the physical inventory and can help mitigate any errors in the logical organization of inventory.

Chapter 1

Introduction

1.1 Problem

Inventory tracking is vital for business operations [19]. Current inventory drives purchase orders, manufacturing orders, and future planning. Without accurate inventory keeping, business will have either too many items in inventory or not enough. When the number of items are over reported, purchasing and manufacturing orders are created against inventory that doesn't exist. This can lead to dissatisfied customers and bad manufacturing planning. Under reporting the number of items is just of much of a problem. There will always be additional unneeded inventory that needs to be stored somewhere. Having to store this inventory increases the overhead cost for that inventory item decreasing the profit margins. Most business rely on forecasting to get an idea of how much inventory they need to keep for any given time. If the inventory counts are not correct then the forecast is working off bad data.

This problem is addressed with Enterprise Resource Planning(ERP) software [22]. These software packages provide inventory tracking and handles the cost of inventory as well. This software combined with inventory management practices such as cycle counting help reduce inventory errors [21]. These software packages break down spaces into inventory locations and items are tracked and moved between these locations. For example a space on a storage shelf will have a unique location in the ERP software and every time inventory is added or removed it is updated. What's important to note is that is that there is no way to tell how that inventory is actually stored. It could be a single pallet at that location, multiple pallets, cases, etc... We know that inventory was removed from the location but we are unaware what that physical process looked like.

As much as ERP software packages help with inventory management they are just a tool. Human error does occur when using these tools. They could accidentally type in the wrong number when performing an inventory move or they could forget to do the inventory move at all. These errors can and do occur making perfect inventory accuracy very difficult.

Which leads to the problem that is being addressed in this project. Take this scenario for instance. There are two pallets of items at a specific inventory location. The ERP software

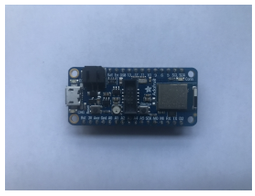

Figure 1.1: Adafruit nRF52840 Development Board

Figure 1.2: Adafruit E-Ink Featherwing

knows that there is a certain number of items in that location. As these items are sold or used in manufacturing items are removed from both pallets in unequal quantities. At the end of the day the inventory count is accurate in the ERP software but it is unknown how that inventory is distributed between two pallets. It becomes very difficult to move these two pallets to separate locations since it is unclear how much inventory is in each. Of course this can be mitigated with policies but people do make mistakes whether they are intentional or not. It can be argued that the single location needs to be split logically into two separate locations. However sometimes it does not make sense to split physical location into multiple logical locations.

1.2 Thesis Statement

A low power method to track and display inventory counts for containers using a combination of Bluetooth low energy for communication, E-Ink technology for displays, and a cross platform application for control.

1.3 Approach

This project is focused on the software running on the smart pallet display rather than the hardware implementation of the display. With that in mind a Bluetooth low energy development board was used as well as a prefabricated E-Ink display. The Adafruit Express nRF52840 development board as well as the Adafruit 2.9" E-Ink Featherwing were specifically chosen for this project. Both of these hardware components were chosen for their library support and ample documentation.

The proposed solution will implement an inventory specific Bluetooth low energy service with its accompanying characteristics. These characteristics will be used to track inventory information such as quantity, unit, and item name. These characteristics are stored locally and updated using write events. The E-Ink display will be used to show these values and is refreshed only when these values change.

Chapter 2

Background

2.1 Key Concepts

2.1.1 E-Ink Displays

E-Ink [9] displays use a combination of micro-capsules and electricity to display words and images. These displays only use power when the display is changed and refreshed. This makes them ideal for low power applications that require a display. However the refresh rate on E-Ink displays can be slow which limits their application. Refreshing the display has to be carefully managed or the display might be damaged.

2.1.2 Bluetooth Low Energy

The Bluetooth 4.0 specification introduced a low power protocol called Bluetooth low energy(BLE) [5]. The BLE protocol is intended for applications requiring low power consumption such as embedded and mobile devices. There are several layers defined by the BLE specification but for the proposed solution only the two highest level layers, the Generic Attribute Profile(GATT) and the Generic Access Profile(GAP), are used. These two profiles define how connections are made between devices and how data is organized and stored.

Generic Attribute Profile

The BLE specification organizes data into services and characteristics. Services are collections of characteristics and the BLE specification has defined services for several common uses such as battery level and heart rate monitor. A BLE server can implement several services depending on the devices requirements. Each service is distinguished by a unique UUID and when a service is read or written to the client must provide a specific UUID.

Each service has characteristics associated with it. This can be as few as one characteristic or it can contain a multitude of characteristics. Each characteristic is given a unique UUID to distinguish it from other characteristics. The individual characteristics can be used to represent numbers, strings, Boolean values, etc... These characteristics can be read and written to by clients. For example the Battery service has a battery percentage charac-

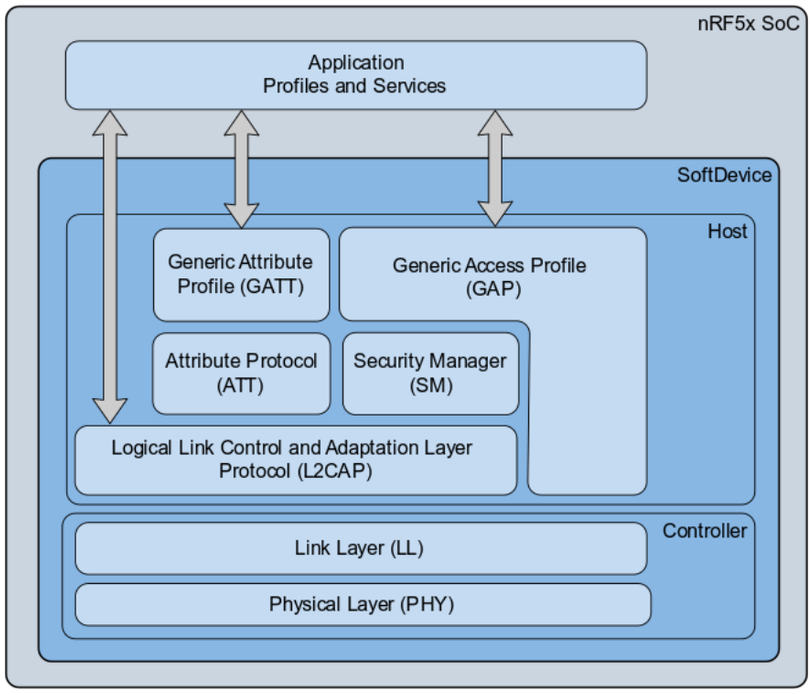

Figure 2.1: Nordic nRF52 BLE Stack

teristic. When a device implements this service a client can read that characteristic to get the current battery percentage

A characteristic also has a set of meta information that describes how that characteristic should be displayed and interacted with. A characteristic defines values for read, write, and notify. Read and write determines if a client can read the value or modify the value. If notifications are enabled then clients can subscribe to a servers characteristic. When this characteristic changes, these changes are pushed to the subscribed clients.

Generic Access Profile

The Generic Access Profile defines how devices are advertised and connect. A BLE device can either be open to connections or placed into secure mode which requires the exchange of pins and encrypted keys. The Generic Access profile can also be set up to advertise available services, the name of the device, and manufacturer information.

2.1.3 Ionic and Google Firebase

The Ionic Framework [12] was used to implement the cross platform control application. The Ionic Framework allows developers to write application using JavaScript frameworks like Angular [3] and generate native mobile applications. This allows the solution to target multiple platforms that support BLE without having to maintain several distinct code bases.

Part of this solution requires authentication before you can access devices. This authentication is provided by Google Firebase [4]. Google provides libraries that allow developers to easily integrate Firebase functionality. Google Firebase provides authentication services that can be easily added to applications.

2.2 Related Work

2.2.1 BLE Beacons

BLE devices broadcast their availability using an advertising packets that relay information such as services and device information. These advertising packets have been used for other purposes. One purpose is the BLE Beacon. These beacons re-purpose the manufacturer specific data part of the advertising packet to broadcast information [6]. There are two different types of beacons. There are non connectable beacons that broadcast internally stored information at regular intervals. Then there are connectable beacons which allow device to connect and interact with implemented BLE services.

These beacons have many different uses. For example a beacon can broadcast the current temperature outside for other devices to read. Then when a device comes within range of a beacon the user receives a notification alerting them to the weather.

There are several different specifications for BLE beacons. Some of them are platform specific while other specifications are designed to be used by a wider range of devices. Regardless these BLE beacons provide a low power way to broadcast small amounts of information. Then as devices come within range they can read that information as needed.

2.2.2 BLE Indoor Positioning

BLE beacons are not the only use of the advertising packet. There has been research into using these packets for indoor positioning [14]. The RSSI value for one meter away can be transmitted as part of the advertising packets. Using this transmitted information receivers can pick up the advertising packet and triangulate the position of the BLE tag. The perceived RSSI values do have to be run through several models in order calculate the position even more accurately [13].

Indoor positioning using BLE is a very interesting research topic. The BLE protocol allows the BLE tags to run for long periods of time on relatively low power. A concern with indoor positioning systems is how accurate can the device be placed in the space. Calculating the location of a device in a room is not very useful if that location is inaccurate. However the accuracy of BLE based indoor positioning is good enough for practical applications. One group used BLE tags and an indoor positioning system to construct a student attendance system [20].

This system shows how applicable indoor positioning with BLE is. Using a simple setup with BLE station modules placed around the classroom, the researchers were able to get enough accuracy to tell when a student was at their desk or not. The system was accurate even though there can be a lot of interference in a classroom environment. They focused on minimizing costs and complications to make it as simple as possible. They were successfully able to demonstrate that the system is usable in a real classroom at Rangsit University. This just goes to show that indoor positioning using BLE can be practical and low cost.

2.2.3 BLE Mesh Networking

A relatively recent use of BLE is in mesh networking. The originally BLE specification did not define how the protocol could be used for mesh networking. However, the newest BLE specification has defined how the BLE can be used as a mesh networking protocol.

One implementation [11] used the Nordic nRF51822 to design and demonstrate a BLE wireless mesh networking protocol. They used the soft device application for the Nordic stack to switch the device between a peripheral and central device in order to route data. The attached three different sensors to the mesh network to show the data being transferred across the mesh network. This project showed that the mesh network provided a low power way to transfer data using BLE.

In 2017 the Bluetooth working grouped announced Bluetooth Mesh which is a mesh network based on the BLE specification. Using Bluetooth Mesh a research group created a smart doorbell as a proof of concept implementation of a Bluetooth Mesh Network [16].

They wanted to test the new technology in a realistic environment to determine its strengths and weaknesses. They chose a smart doorbell because it is a well know office automation application. They use the BLE mesh network protocol to extend the reach of event messages associated with the smart doorbell.

They placed nodes at each doorway to act as doorbells with relay nodes to a central node. This central node that acts as a gateway to the internet. When the request is relayed to the central node the staff would be notified of the request.

The proof of concept was implemented using the Nordic nRF53832 SoC, Soft Device, and SDK. These near ubiquitous devices were chosen for their BLE Mesh Networking support and its very low cost.

They found that there was a relatively high power draw on the devices that relied on battery power. This could be due the hardware design or the fact the coin cell operated devices sill acted a relay nodes. They propose several solutions to help offset the power consumption. The topology of the network is dependant on the layout of the building. They used one node per 139 m^2 but this can be changed depending on the project

2.2.4 BLE and E-Ink Hardware

The solution proposed in this paper is not the first E-Ink Display that uses BLE. There are several commercial products that uses this technology for smart labels.

Pricer [10] has used E-Ink displays to create electronic shelf labels. These shelf labels display the current price of an item in real time. These tags are updated using an optical wireless communication standard. These labels can last up to ten years and really highlight how low energy this technology is. However these shelf labels do not store any kind of inventory information which limits there usefulness.

Figure 2.2: Pricer Electronic Shelf Label

Papyr [18] is an open source project that combines an E-Ink display with a nRF52832 SoC. Using the BLE SoC devices can connect to the display and change what is displayed. This combination of hardware shows that these two low energy technologies work well when combined. The solution in this paper uses this concept and extends it by defining a custom BLE service and code to interact with it.

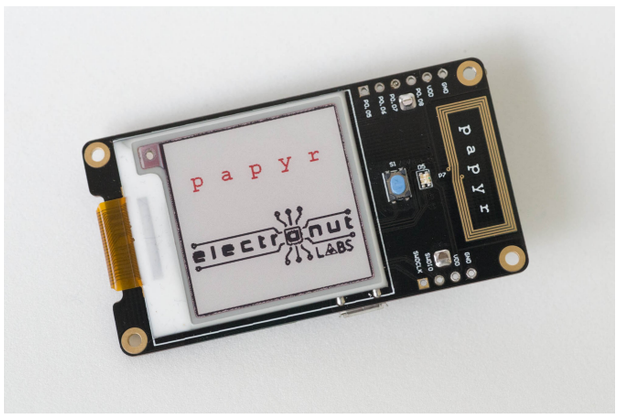

Figure 2.3: Papyr Wireless display

What makes this solution different from these related product is the implementation of an inventory specific BLE service. This extension of the concept allows the device to actually track inventory information and tie it to physical.

Chapter 3

Design and Implementation

3.1 Bluetooth Services

The BLE service was implemented using Adafruit's BLE Library [2] for the Nordic nRF52840 SoC. This library integrates with the Arduino development environment for fast prototyping and low startup time. The Nordic Connect SDK [23] was considered for this solution but setting up the development board for the SDK added additional unneeded complexity.

3.1.1 Inventory GATT

The inventory serve that the proposed solution implements contains four characteristics. The inventory service could posses multiple more characteristics but for this solution four was a sufficient number. Each characteristic is set up for both reading and writing with no additional security requirements. These four characteristics were chosen because the demonstrate how the BLE service can be used to track inventory information while at the same time remaining inside the scope of an undergraduate thesis.

CHARACTERISTIC:	UUID:
ITEM ID	ec9b72a1-9181-16a1-6541-833430804c46
QUANTITY	2e0da5f0-66d8-cfae-a049-c057cfe0ca5a
CONTAINER	2952a903-a06f-e0b1-274c-88297bba03f1
UNIT	2b1d8769-96e8-7ca7-0f4d-d7b1e34f5d24

Figure 3.1: Bluetooth Characteristics and the assigned UUID

The Item Id characteristic is used to determine what item is being referenced by the label. The values for this characteristic are represented internally as a variable length sequence of bytes. This sequence of bytes represents a string identifying the id of the item.

The Quantity characteristic is used to store how many items are physically stored in the container the label is attached to. This quantity is represented by exactly two unsigned bytes. This means that the characteristics can take on values from 0 to 65,536. This maximum value is large enough to represent large quantities of items while also limiting the amount of data that needs to be stored

The Container Type characteristic is represented by a single byte. Only the first three are assigned specific meaning while the rest of the 255 values are assigned to none. This

leaves plenty of room to grow while at the same time giving us enough data to show how the proposed solution works. This characteristic is used to help describe how the label is being used and is an example of how the inventory service can be used to store useful meta information.

VALUE:	CONTAINER
0	Pallet
1	Box
2	Shelf
3	None

Figure 3.2: Container Characteristic Values

The last implemented characteristic is the Unit Type characteristic. This characteristic can represent units such as length, weight, and pieces. Internally this characteristic is resented by an unsigned byte where each value is assigned to a unit type. Currently there are only seven units that have been implemented leaving room for future growth. Any value that is chosen that is not used is considered N/A for this proposed solution.

VALUE:	UNIT
0	pcs
1	lb
2	g
3	ft
4	in
5	m
6	cm

Figure 3.3: Unit Characteristic Values

3.1.2 Persistent Storage

The Bluetooth characteristics are not persistent. This means that on a power cycle the current data is lost and it reverts back to the default values. The proposed solution is a battery powered device so at some point power will be lost. If the current data is wiped each time then the usefulness of the solution is greatly diminished.

The first step was determining where to persist the BLE service data. Since development was being done with the nRF52840 SoC there was internal flash storage that could be used to solve this problem. One of the downsides of flash memory is that it wears out after repeated writes. For this specific device the flash memory can support at minimum 10,000 erase cycles [17]. This number is more than sufficient for this solution.

Flash access is performed by the Adafruit Little FS library [2]. This file system keeps track of stored files and performs wear leveling to ensure that the flash memory lasts as long as possible. Each BLE characteristic is stored in a separate file that is named after its specific UUID. This makes it easy to change the values in flash memory whenever the BLE characteristic is changed.

The Arduino BLE library allows write callbacks to be set for each Bluetooth characteristic. This provides an easy mechanism to update the flash memory with the new vales for the BLE service. Whenever a BLE characteristic is written to the current value is removed

```
root
|_ adafruit/
|  |_ bond_prph/
|  |_ bond_cntr/
|_ 2E0DA5F0-66D8-CFAE-A049-C057CFE0CA5A 2 Bytes
|_ 2952A903-A06F-E0B1-274C-88297BBA03F1 1 Bytes
|_ 2B1D8769-96E8-7CA7-0F4D-D7B1E34F5D24 1 Bytes
|_ EC9B72A1-9181-16A1-6541-833430804C46 11 Bytes
```

Figure 3.4: Layout of the Flash File System

from flash memory and the new value is written. This way the persistent storage always contains a copy of the current values and it is never written to more than is required. When the device first loads it reads each value stored in flash memory and sets the BLE characteristics to those initial values. If there is not a stored value then the characteristic is set to its default value.

3.1.3 GAP

For this solution security wasn't the main focus. With this in mind the connection to the smart display was left open to general connections. The access profile is set up to broadcast the name of the device and advertise the inventory service. The device also advertises RSSI values for the connection. There is a limit on the number of bytes and advertisement can contain so it was important to be selective on

It was important to advertise the service so that potential clients can recognize what services the smart display can perform. By broadcasting the service, clients can filter potential connections so only valid options remain. This makes it easier on the end user because it limits the number of connections they have to parse.

The name is advertised for a very similar reason. Having the name as part of the advertisement allows for easy parsing of devices. It also makes it easier for end users to choose which smart label to connect with.

3.2 E-Ink Display

E-Ink displays make for a great low power display method however there are limits on its refresh rates. For example that Adafruit 2.9" E-Ink Featherwing has a suggested minimum refresh rate of three minutes [1]. This makes managing the refresh rate a vital task. If not managed properly the display could be damaged or be of little use since the display never shows the current data in a reasonable amount of time.

The proposed solution keeps track of two time values and a Boolean. The first time value is when the display was last refreshed while the second time value is the time to refresh in the future. The Boolean value determines if the display needs to be refreshed at all. If none of the BLE characteristics have been changed then the display will not refresh after three minutes. This helps reduce the power consumption of the solution while extending the live

of the E-Ink display.

Whenever one of the BLE characteristics is changed the inventory callback function sets up the display for a refresh. This is where the Boolean to update the display is so important. The characteristics are written asynchronously so if multiple characteristics are changed then that callback is called multiple times. This Boolean values ensures that if multiple values are changed between refreshes the display is only refreshed once for all of those changes. Once the display is set to refresh then the time that it will refresh remain unchanged between inventory write callbacks.

After determining if the display has already been scheduled for a refresh, we determine what time we need to refresh the display. If it has been less than three minutes since the last display update then we schedule the next update three minutes after the previous one. However if it has been more than three minutes since the last update then we schedule the next screen update three seconds from the time of the request. This delay was chosen so that if multiple characteristics were changed at once, the first once wouldn't trigger a premature refresh and the display has to wait three minutes to display the other changes.

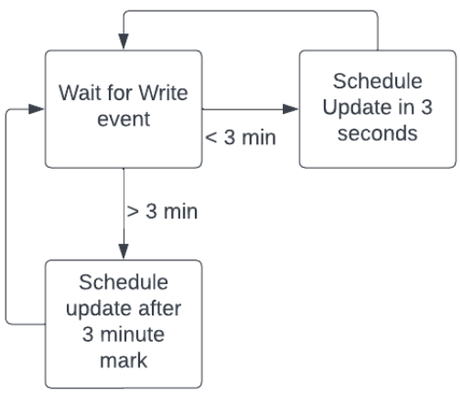

Figure 3.5: Flowchart depicting the refresh schedule process

The display shows the container type, the quantity, name, and unit of the stored inventory. Each of these values is placed on its own line with a label describing the value. This makes it easy to read the data at a glance and understand what is being stored.

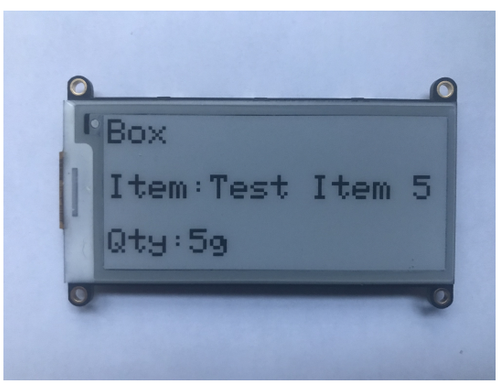

Figure 3.6: E-Ink display of the inventory information

3.3 Control Application

The Control Application is implemented using the JavaScript framework Angular and the Ionic Framework for native cross platform apps. The control application uses the Capacitor Bluetooth-LE library [7] to connect and manage Bluetooth low energy devices.

The application is split into two main views and have been designed with usability in mind. The first view is an authentication screen. Here the user is required to login using their credentials before they can access the rest of the application. Once the user is logged in the application they remained logged in until their session expires or they log out. The second view is where a user can connect and manage a Bluetooth smart label.

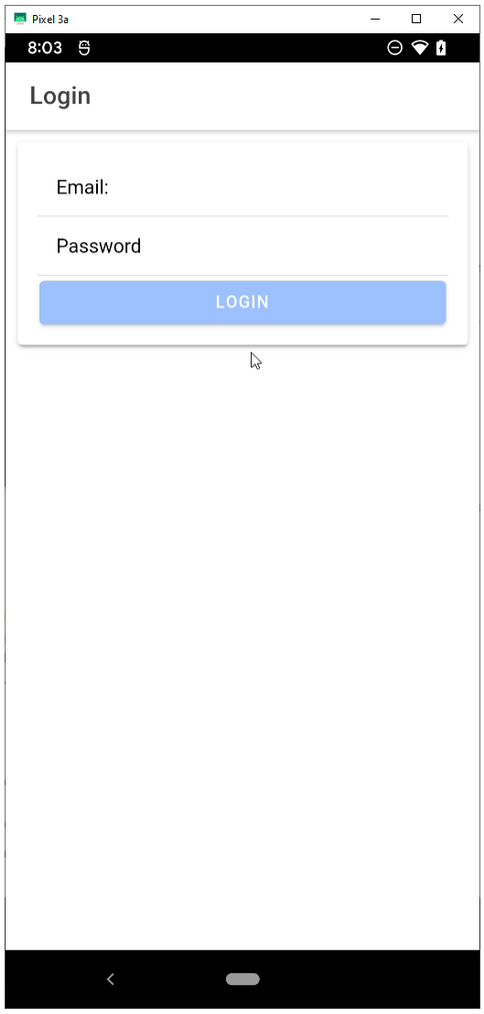

Figure 3.7: Login Screen of the Control Application

On the second view the user is shown a scan for devices button that launches a selection screen. This selection screen allows the user to select which BLE device they want to connect to. Once the mobile device has been connected to the BLE device the user is then shown what current inventory values are set on the device. They can either disconnect from the device or update these values.

The quantity and item id fields allow users to enter arbitrary values as long as they are numbers or strings respectively. However the container type and unit type are implemented

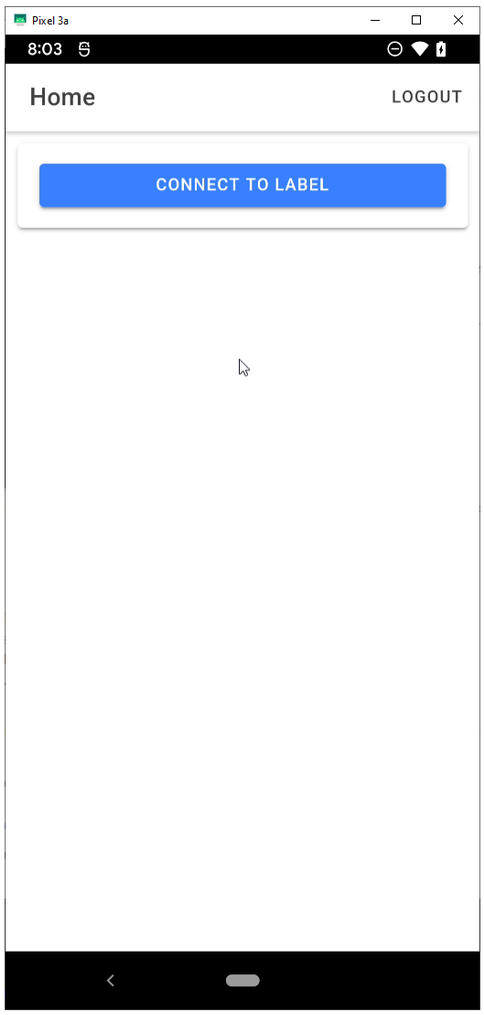

Figure 3.8: Initial view that allows the user to scan for devices

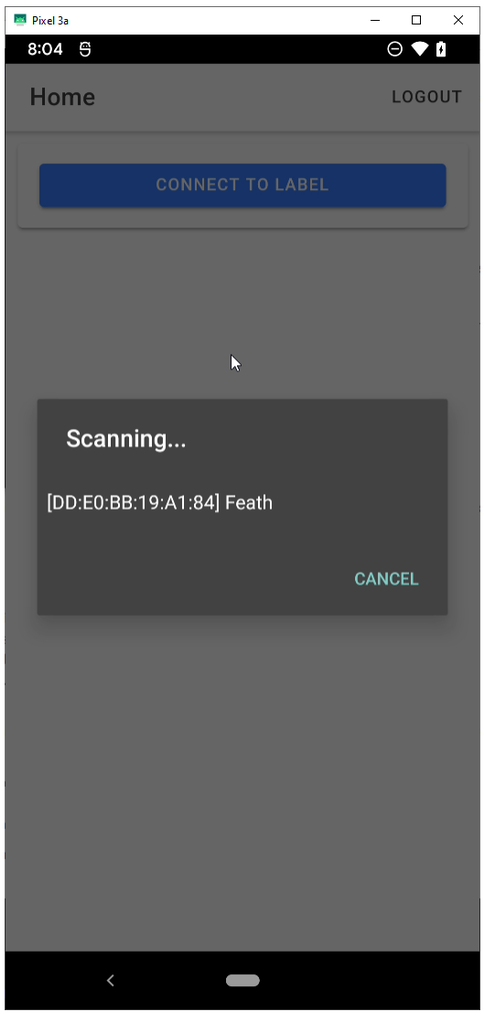

Figure 3.9: Application showing which devices are available for connection

using a drop down to restrict the user to the pre-specified options. The prevents users from selecting undefined values and causing unwanted behavior.

After filling out the values the user wants to change, they can press the update button. The application takes those values and writes them to the specific characteristics on the target device.

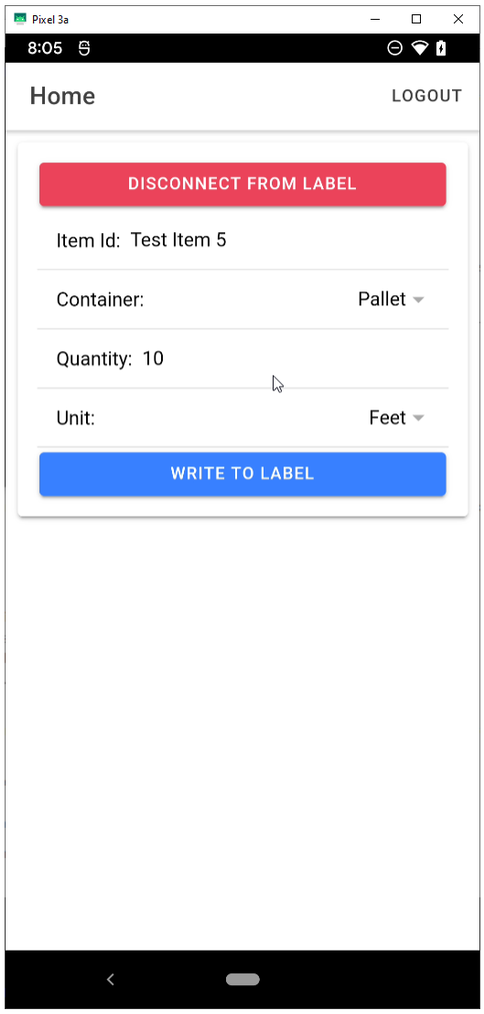

Figure 3.10: Data retrieved from the connected BLE device

The authentication screen uses Google's Firebase SDK and authentication module. The main screen is protected by an AuthGuard that redirects users back to the logon screen if they have not been authenticated yet. The login screen takes the users email and password and uses Google's SDK to contact Firebase and authenticates the user. If the right password and email are provided then the user is allowed access to the rest of the application. If the authentication fails then the user is not given access

Since the UUID's for the inventory service are not one of the standard services defined by the BLE specification, the entire 128 bit UUID has to be used instead of the much smaller 16 bit UUID for standard services. In order to organize and keep the code clean these UUID's were defined in a separate constants file. This allows easy access without having to provided the entire UUID every time it is used.

Chapter 4

Results and Evaluation

4.1 Inventory Service

The actual inventory service itself operates like its supposed to. The service is broadcast as part of the advertisement and devices can connect. The characteristics in the service can be referenced by their UUID handle allowing them to be read and written. The biggest issue while implementing the service was dealing with the order the bytes were sent. This did not affect the two characteristics that were one byte long but it did affect the Item Id and the Quantity characteristics. This was ultimately corrected for but it did differ based on what kind of client was connected.

The main problem with the given implementation is the how stable the flash file system is. It can be easy to corrupt the file system especially when interrupted on writes. Once this occurs the file system has to be removed from memory and re constructed before it will start working again. This is not ideal for a device that can lose power without very much warning.

4.2 Display

Utilizing the refresh management scheme described above the display will only update every three minutes. If multiples characteristics are written at the same time the screen will only refresh once for all of them. This could be within three seconds of the first value being changed or it could be three minutes after it was last updated. The implemented refresh code works as intended to increase the lifetime of the display.

However the restriction on large refresh times does make the display harder to use. There is no indicator that lets the end user know that the device is working on the update. This lack of immediacy and transparency is a detriment to ubiquitous devices like this. When the device does not indicate the the information has been received a user will attempt to send the data multiple times or report it as an issue. While the E-Ink display greatly reduces the power consumed by the device it is not as user friendly as it could be.

Another issue with the display is that it is a static display. Once the information is

written to the screen it will remain there regardless of power status on the device. This means it is effectively impossible to tell when the device is on or not. Without the ability to glance and tell the power status it becomes a problem when the battery dies and there is no way to update the display.

4.3 Control Application

The control application works exactly how it was intended. It uses the Google Firebase SDK to authenticate the user and ensures only valid people can use it. However the use of this SDK requires a network connection. This can lead to problems in large warehouses where internet connectivity can be spotty. If their is internet connectivity available then this works exactly as intended.

The application was tested using and Android device. The Android platform was chosen for its cross platform development and the low cost of entry. The only issue with targeting the Android platform occurred when the native application was generated. The Ionic Framework generates an application that can be run on an Android device. However there were several instances when this generation step failed. When this occurred all of the android source code had to removed and generated from scratch before it worked properly. This only occurs occasionally and once these steps are followed the native app works as intended.

Chapter 5

Conclusion

5.1 Summary

In this paper Bluetooth Low Energy was combined with and E-Ink display to create a low powered wireless display. This had been done for electronic shelf labels and as a general purpose display as well. This project took this low powered display and extended it to include an inventory specific Bluetooth Low Energy service. This service allows the tracking of inventory items and their quantities. These values are tied to the physical label and the container they are attached to. This helps shows how inventory is distributed physically which may differ then how it is organized logically.

The inventory display uses BLE which allows it to be accessed by any platform that support Bluetooth Low Energy. The labels can then be integrated into whatever system and devices that a company uses for inventory tracking.

5.2 Future Work

The proposed solution is an extension of previous work involving BLE and E-Ink displays. Instead of being a wireless display the device was specialized to for inventory information and quantity tracking. Since Bluetooth Low Energy was chosen there are several directions that this project can be extended.

Currently this project is using development boards instead of a dedicated hardware platform. In the future custom hardware can be designed so it fulfils its role as an inventory label that much better. This can include circuits to disable the display when not in use and contact points for easy charging.

Bluetooth Low Energy can be used to make beacons. These low power beacons broadcast information that other devices can pick up. If there are three receiving stations then it is possible to triangulate the location of the beacon using the differing signal strengths [15]. Since the proposed solution also makes use of the Bluetooth Low Energy it is possible to make use of beacons. With this extension it would be possible to track the location of tagged location.

Lately work has been done into Bluetooth Low Energy mesh networks [8]. These mesh networks allow devices to communicate with each other without a central authority. This technology can be used with the smart labels described in this paper. Using this technology the labels can be networked together to create a decentralized inventory tracking system. At any point the network can be queried to retrieve the current state of inventory. As inventory is added or removed the state of the network will reflect the current inventory status.

References

[1] *Adafruit eInk Display Breakouts and FeatherWings*. en-US. URL: https://learn.adafruit.com/adafruit-eink-display-breakouts/usage-expectations (visited on 04/13/2022).

[2] *Adafruit nRF52 Arduino*. URL: https://github.com/adafruit/Adafruit_nRF52_Arduino.

[3] *Angular*. URL: https://github.com/angular/angular.

[4] *Angular Fire*. URL: https://github.com/angular/angularfire.

[5] *Bluetooth Core Specification*. Dec. 2019. URL: https://www.bluetooth.org/docman/handlers/downloaddoc.ashx?doc_id=478726.

[6] *Bluetooth low energy Beacons*. Jan. 2015. URL: https://www.ti.com.cn/cn/lit/an/swra475a/swra475a.pdf.

[7] *Capacitor Bluetooth Low Energy*. URL: https://github.com/capacitor-community/bluetooth-le.

[8] Seyed Mahdi Darroudi, Carles Gomez, and Jon Crowcroft. "Bluetooth Low Energy Mesh Networks: A Standards Perspective". In: *IEEE Communications Magazine* 58.4 (Apr. 2020), pp. 95–101. ISSN: 0163-6804, 1558-1896. DOI: 10.1109/MCOM.001.1900523. URL: https://ieeexplore.ieee.org/document/9071998/ (visited on 04/13/2022).

[9] *E Ink Electronic Ink*. URL: https://www.eink.com/electronic-ink.html (visited on 04/13/2022).

[10] *Electronic Shelf Label Solutions for Retail - Digitalize Your Store with Pricer*. en-US. URL: https://www.pricer.com/products/ (visited on 04/13/2022).

[11] P. Gomathinayagam and S. Jayanthy. "Implementation of Mesh Network Using Bluetooth Low Energy Devices". In: *Intelligent and Efficient Electrical Systems*. Ed. by M.C. Bhuvaneswari and Jayashree Saxena. Singapore: Springer Singapore, 2018, pp. 205–213. ISBN: 978-981-10-4852-4.

[12] *Ionic Framework*. URL: https://github.com/ionic-team/ionic-framework.

[13] Zhu Jianyong et al. "RSSI based Bluetooth low energy indoor positioning". In: *2014 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*. 2014, pp. 526–533. DOI: 10.1109/IPIN.2014.7275525.

[14] Ankush A. Kalbandhe and Shailaja. C. Patil. “Indoor Positioning System using Bluetooth Low Energy”. In: *2016 International Conference on Computing, Analytics and Security Trends (CAST)*. 2016, pp. 451–455. DOI: 10.1109/CAST.2016.7915011.

[15] Pavel Kriz, Filip Maly, and Tomas Kozel. “Improving Indoor Localization Using Bluetooth Low Energy Beacons”. en. In: *Mobile Information Systems* 2016 (2016), pp. 1–11. ISSN: 1574-017X, 1875-905X. DOI: 10.1155/2016/2083094. URL: http://www.hindawi.com/journals/misy/2016/2083094/ (visited on 04/13/2022).

[16] Caril Martínez, Leonardo Eras, and Federico Domínguez. “The Smart Doorbell: A proof-of-concept Implementation of a Bluetooth Mesh Network”. In: *2018 IEEE Third Ecuador Technical Chapters Meeting (ETCM)*. 2018, pp. 1–5. DOI: 10.1109/ETCM.2018.8580325.

[17] *nRF52840 Product Specification*. Feb. 2019. URL: https://infocenter.nordicsemi.com/pdf/nRF52840_PS_v1.1.pdf.

[18] *Papyr nRF52840 ePaper Display*. en. URL: https://hackaday.io/project/165467-papyr-nrf52840-epaper-display (visited on 04/13/2022).

[19] Darya Plinere and Arkady Borisov. “Case Study on Inventory Management Improvement”. In: *Information Technology and Management Science* 18.1 (Jan. 2015). ISSN: 2255-9094. DOI: 10.1515/itms-2015-0014. URL: https://itms-journals.rtu.lv/article/view/itms-2015-0014 (visited on 04/13/2022).

[20] Apiruk Puckdeevongs et al. “Classroom Attendance Systems Based on Bluetooth Low Energy Indoor Positioning Technology for Smart Campus”. In: *Information* 11.6 (2020). ISSN: 2078-2489. DOI: 10.3390/info11060329. URL: https://www.mdpi.com/2078-2489/11/6/329.

[21] Manuel D. Rossetti, Terry R. Collins, and Ravi Kurgund. “Inventory Cycle Counting – A Review”. In: 2001.

[22] E.M. Shehab et al. “Enterprise resource planning: An integrative review”. en. In: *Business Process Management Journal* 10.4 (Aug. 2004), pp. 359–386. ISSN: 1463-7154. DOI: 10.1108/14637150410548056. URL: https://www.emerald.com/insight/content/doi/10.1108/14637150410548056/full/html (visited on 04/13/2022).

[23] *Welcome to the nRF Connect SDK! — nRF Connect SDK 1.9.99 documentation*. URL: https://developer.nordicsemi.com/nRF_Connect_SDK/doc/latest/nrf/index.html (visited on 04/13/2022).