Eng Int Syst (2020) 1: 61–68

Engineering
Intelligent Systems

SmartOrders: A BLE and NFC Enabled Android-Based Offline Ordering System for Restaurants

Michael Zipperle[1]*, Christian Laustsen[2†] and Anders Rikvold[2‡]

[1] *Furtwangen University*
[2] *Technical University of Denmark*

Ordering systems for restaurants offer a great way to offload work from restaurants, in taking orders, and lessens the burden of customers needing to acquire and interact with a waiter. In this paper we investigate the feasibility of an offline ordering system for restaurants that only uses commodity hardware (smartphones) and communicates entirely over Bluetooth Low Energy (BLE). This stands in contrast to current systems, which either require special hardware at the tables or that the customer uses a smartphone application that functions over the Internet. Using technology such as BLE and Near Field Communication (NFC) we propose a system to handle the interaction between the restaurant and customer, along with a prototype implementation, *SmartOrder*, that demonstrates the feasibility of the system. SmartOrder shows that common concerns such as the range and throughput of BLE are either not a concern or at least acceptable. The prototype implementation achieves a range of 71 meters with no obstacles and 12 meters around corners (6 meters on each side). Data throughput is shown to be somewhat slow, but this is acceptable since data transfers happen infrequently. Finally, both of these limitations are addressed with potential solutions.

Keywords: Mobile Application, Offline Systems, Smart Systems, Order Systems, BLE, NFC

1. INTRODUCTION

Imagine you have just arrived in a restaurant, and taken a seat at a table. In order to start eating, you first ask the waiter for the menu, which he brings to you. After thinking about what you would like, you call the waiter to your table again, and tell him what you would like to eat. The waiter writes down your order, and takes it back to the restaurant kitchen, where your meal is prepared. Then, hopefully after not waiting too long, the waiter brings you your meal.

This process involves several steps, and requires time and effort form both the customer and the restaurant personnel. Also, the process might be irritating for the customer if it is difficult to get hold of the waiter. Several systems have tried to alleviate this process by introducing a digital ordering system, where the communication between the restaurant and customer is optimized through digitally communicating devices. However, current systems have one of two drawbacks. Either they

a) require an investment in expensive hardware, or b) it requires the user to be connected to the Internet.

The former of these drawbacks usually occurs in systems which requires the restaurant to invest in a single specialized ordering device, such as a tablet, for each table. Some systems try to alleviate this drawback by allowing users to use their own smartphones to order food, through running an ordering application. To the best of our knowledge, such systems requires the user to be connected to the Internet. For a lot of customers, especially for ones with a local data plan, this is no problem. However, for certain people, especially for travellers in foreign countries or persons with a prepaid data plan, requiring Internet access

*michael@Zipperle.de
†CKL@codetalk.io
‡anders_rikvold@hotmail.com

is not desirable, as it might be inconvenient or incur extra charges.

To alleviate this problem, we introduce a system, called *SmartOrders*. The main idea of this system is that the restaurant has its own smartphone or tablet device, and that at least one customer at each table of the restaurant has the same. When users want to order, they scan an NFC chip on their table, upon which their phones will automatically connect to the restaurant's device over BLE. After this, the customer can see the menu on his or her device, and place orders from it. As the system uses BLE, it does not require an Internet connection, so that customers are not required to have their own local data plan, and restaurant owners are not required to provide free WiFi (WiFi) for its customers. Also, it does not require a high investment in hardware from the restaurant either, as it only requires each table to be equipped with a cheap NFC tag, along with a single smartphone or tablet device.

The motivation behind using BLE instead of WiFi is threefold. First, BLE, as implied by its name, has a very low energy consumption[1]. Secondly, the BLE component is often times either entirely unused or far from saturated with devices. This provides a unique opportunity to use a component that does not disrupt any existing actions or operations the customer might have going on. Finally, using WiFi introduces an additional hardware dependency. While this is usually not a problem in an indoor non-mobile restaurant, it would exclude using the application in e.g. a park or other ad-hoc instances without power outlets.

Our paper provides the following main contributions.

a) A prototype implementation, called *SmartOrders*, an Android-based offline, low-cost ordering system for restaurants, based on BLE and NFC technology.

b) An evaluation of the practical feasibility of an ordering system such as SmartOrders, which uses BLE for communication.

2. RELATED WORK

There are already some existing solutions, which are solving the same problem space, but all them need expensive additional hardware or an Internet connection for the communication between the customer and restaurant owner. In the following paragraphs, existing solutions are mentioned in more detail.

Smart Ordering System via Bluetooth [2]: This work provides an insight in sending the orders from a table to the restaurant owner. In this approach, keypads, on each table of a restaurant, allow the customers to type in the number of the menu item to order it. If the customer presses `Enter`, the order is sent to the restaurant owner via Bluetooth. The restaurant owner is using an application, running on Windows, where he can see the orders from the customers. However, this solution does not offer the customer a good user experience. First, the customer needs to know the number of the menu item he wants to order. Secondly, the customer has no overview of which orders he has already placed.

Visionect: Tabletop Ordering [3]: In this approach, there is a tablet with the installed Visionect Application on each table of a restaurant. The customers can use this application to make their orders. The restaurant owner can receive the orders via the tablet or a web interface. In addition, this system allows customers to give feedback regarding the food and drinks to the restaurant. Nevertheless, this solution requires additional hardware, which leads to high acquisition and maintenance costs for the restaurant owner.

Smart Restaurant Menu Ordering System [4]: This system is based on embedded touchscreens in each table. The customers can use this touchscreen to browse the menu and afterwards make their orders. The interface for the customer owner is a big screen in the kitchen, where the cooks and waiters can see the incoming orders. This solution has the same limitation as to the previous one, it requires additional hardware.

QikServe [5]: QikServe provides an application for the customers as well for the restaurant owner. The customers can use their own smartphone to download the menu and make an order to the restaurant. An Internet connection is necessary to enable the communication between the customers and the restaurant owner. However, many restaurants do not have good cellular coverage, or additional hardware for a WiFi network is required.

In summary, the limitations of these existing solutions are that they require an investment in expensive hardware, or that the user must be connected to the Internet.

3. SYSTEM OVERVIEW

The system consists of two applications, a *Restaurant Owner Application (ROA)*, and a *Customer Application (CA)*. The idea is the restaurant using this system is equipped with a smartphone or tablet, which is running the ROA. Each table is then equipped with an NFC tag, which contains information about the table, as well as information about how to connect to the Restaurant Owner application device. The restaurant owner will use the Restaurant Owner Application to program this information onto the tag.

When visiting the restaurant, each customer is equipped with his or her own smartphone, which has the CA installed. When a customer sits down at a table in the restaurant, and scans the NFC tag located on the table, their smartphone will automatically open the application and establish a connection to the restaurant over BLE (in other words, no Internet connection is required). After the connection is established, the customer can make orders and communicate with the restaurant's device to place orders. A graphical overview of the system can be seen in figure 1.

4. SYSTEM FUNCTIONALITY

The following sections lists the functionality of the application. The functionality has been listed for the restaurant customer and owner applications separately. It is however worth noting that the functionalities cross-cut the applications

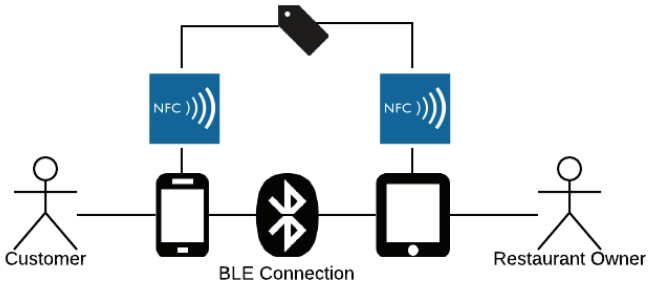

Figure 1 Overview of the SmartOrders system

in some cases, such as placing an order, or establishing a connection between the restaurant and the customer. In order to stress the advantages of our application, we have also specified certain constraints for the application.

4.1 Customer Application

Connect to Restaurant This feature will connect the user to the restaurant, so that he or she can communicate with it to use other functionalities. This should happen in the following way.

1) The user scans the NFC chip with their smartphone.
2) The smartphone automatically opens the application, reads the NFC chip contents, and uses this to connect to the restaurant via BLE.

View Menu In order to be able to place orders through the app, he or she needs to be able to see the menu, and it would be most practical to do this through the application. The customer should be able to choose to see the menu, and if he or she does, the menu should be presented.

Order from Menu This functionality is the core of the application; the customer actually ordering from the restaurant. When the user is presented with the menu, he or she should be able to build and order, and send the order to the restaurant over the already established BLE connection.

View Orders The customer can view the orders placed by him. As long as the order is not being processed, the customer can cancel or edit an order.

View Restaurant Information When the user is connected to the restaurant, the user should be able to see the information of the restaurant he or she is connected to. As well as showing basic information about the restaurant, this functionality will provide the user with a simple way of seeing which restaurant he or she is connected to, or if he or she is at all connected to one.

4.2 Restaurant Owner Application

Program NFC chip The system requires that every table of the restaurant is equipped with an NFC chip that the user can scan to connect to the restaurant. In order for this to work, the NFC chips need to contain the necessary information for establishing the connection. Therefore, the restaurant owner needs to be able to program each and every chip through the app, such that the chip contains information for a) establishing a BLE connection and b) identifying the table of the customer, for example when receiving an order from it.

Update Menu In order for the user to be able to order from the menu, the restaurant owner needs to create one through the application. This feature consists of two parts, a) adding menu items and b) deleting menu items. Restaurant customers will be sent the new menu upon establishing a new connection to the restaurant.

See Placed Orders The restaurant owner should be able to see orders placed by customers, as well as which customer (table) placed the order. Very central to the core functionality of the application, this will allow the restaurant to serve customers.

Finish Order In order to keep track of which orders and customers have been served, the restaurant owner must be able to continuously update the orders that have been served.

4.3 System Constraints

Platform Both applications must be able to run on smartphones. The restaurant owner must only need one smartphone, and each customer (table) will also need their own smartphone to order.

Communication The application must be able to function without the customer having an Internet connection. This means that the restaurant does not have to provide a WiFi connection to customers, or exclude customers without a mobile network connection. It is assumed that communication between the restaurant and customer applications are done through BLE.

5. SYSTEM ARCHITECTURE AND DESIGN

Our System Architecture exists of two separate applications, one for the restaurant owner and one for the restaurant costumers. In Fig. 2 our Software Components are shown,

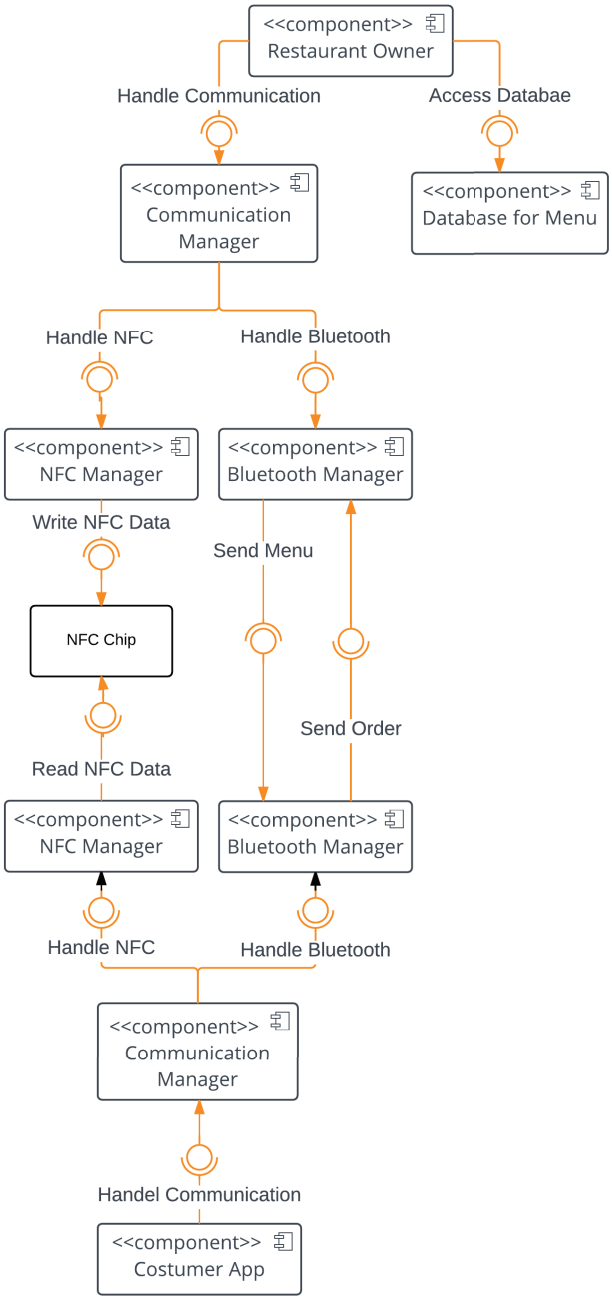

Figure 2 Software Component Diagram

both applications are using the Communication Manager to handle the communication between both applications. The Restaurant Owner Application stores the Menu, Restaurant Information, Tables and Orders in a local SQ Lite Database [6]. The Communication Manager exists of a NFC and BLE component.

The NFC and BLE components are integral parts of the architecture of the application, and as such their part will be described in detail in the following sections.

5.1 NFC

The NFC tag allows great flexibility in both the data on the tag itself and the actions performed upon scanning such a tag. Devices implement managers that handle scans, such as the Tag Dispatch System in Android[7]. This means we can register the application to open when it encounters a certain MIME type (e.g. *application/smartorders*) on the tag. From a user experience viewpoint, this is superior compared to using QR codes, which involves opening the application and then waiting on the camera to initialize and scan the code.

The data that is written on the NFC tag is rather simple, and consists of a table ID along with an identifier for the ROA in case of multiple ROA in the same proximity. Originally this was the Media Access Control (MAC) address of the device, but since later versions of the Android API removed this ability, it is for now just the name of the ROA device.

The operations that the NFC component must support is then the ability to a) acquire a tag, b) read the data on a tag

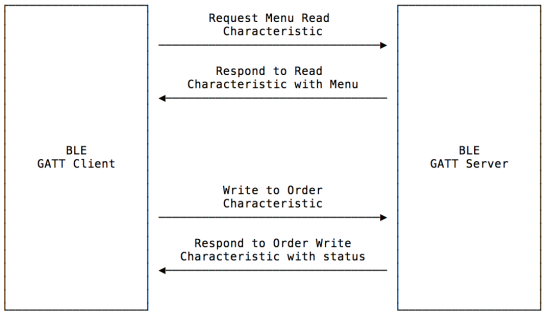

Figure 3 The BLE GATT communication between client and server

and c) write new data to a tag. Furthermore, to not let existing data interfere with the process of overwriting (e.g. by opening a related app) it takes control of the Tag Dispatch System and handles all delegations in-app.

To support the most devices, the data format written onto the tags are NDEF[8], which is specified by the NFC Forum (type 1, 2, 3 and 4). This also means that the tag must be compatible with this format, such as tag types NTag213 and NTag216.

5.2 BLE

With BLE we do encounter some limitations with regards to the devices that can run the ROA. It appears that only newer devices can act as a Bluetooth GATT server (in other words, a peripheral). That said, as of 2017 the list of compatible devices[9] is fairly long, indicating that it is reasonable to expect a restaurant owner to either have such a device or to invest in one. As for the client side, most devices support BLE, so this poses no problems here.

As such, the ROA will act as a Bluetooth GATT server (peripheral role) and the CA will act as a Bluetooth GATT client (central role). The ROA advertises its services using a universally Unique Identifier (UUID) that is known to both applications, and the client then connects to this service. To facilitate communication between the server and client, the server supports to characteristics, one the client can read from (the menu) and one the client can write to (submitting the order), as illustrated in figure 3.

In reality though, each data packet is limited to a size of 20 bytes, which means the communication is actually divided up into many subsequent requests. This is done by the client issuing read requests after each response, until it receives an `!END!` from the server. At this moment it puts together all the packets that it has received, to form the full menu. A similar approach is taken when the order is submitted, although here the packets are put together at the server-side.

6. PROTOTYPE IMPLEMENTATION

The restaurant owner and customer applications were implemented[10][11] on the Android operating system, which runs on both smartphones and tablets. More specifically, the applications target a minimum Application Programming Interface (API) version of 18. Screenshots, with explanations, from the prototype implementation can be seen in figures 4 and 5.

6.1 Connecting to the Restaurant

As mentioned in section V, the BLE and NFC components are a large underlying part of the application. In the prototype implementation this was implemented by using the NFC and BLE adapters provided by the Android Operating System (OS). Both of these adapters are wrapped up in a `CommunicationManager` object, which provides a simple and intuitive interface for the ROA and CA to use. For example, reading the NFC tag is as simple as adding a method to the `onNewIntent` Android system event.

The BLE component requires a bit more work to integrate. The ROA implements an interface `RestaurantData` that contains two methods (omitting all public keywords and void returns, they are assumed on all methods for the rest of the section unless otherwise specified):

1) `String getMenu()`
2) `String handleOrder(String order)`

The first is called from within the Bluetooth GATT server when the client requests the menu (by reading the characteristic), and returns the menu to the GATT server. The second is called from within the GATT server when the client submits an order (by writing to the characteristic), and lets the ROA handle the value contained in the argument to the function.

As for the client, the CA implements the interface `ClientData` that contains three methods:

1) `handleMenu(String menu)`
2) `handleOrderResponse(String msg)`
3) `handleConnectionResult(boolean connected)`

The first is called when the Bluetooth GATT client is done receiving the menu, the second is called after having submitted the order and the final method is called after connection status has changed, to indicate the state the GATT client is in.

The `CommunicationManager` is also implemented in a reference application that showcases the usage of it, to lessen the burden of eventual implementers.

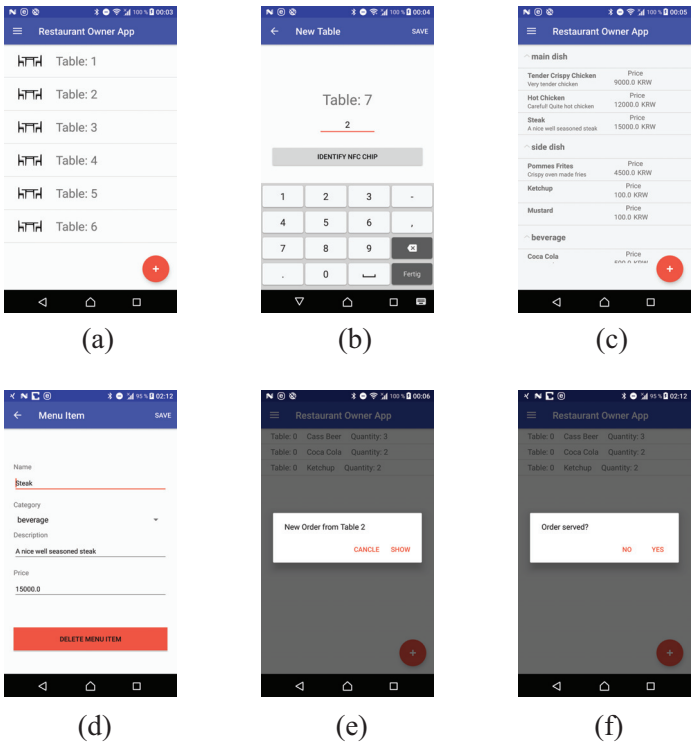

Figure 4 Screenshots of the Restaurant Owner Application Prototype. 4a shows the table overview. 4b shows table creation screen. 4c shows the menu item screen. 4d shows the menu item creation screen. 4e shows receiving an order and 4f shows serving an order

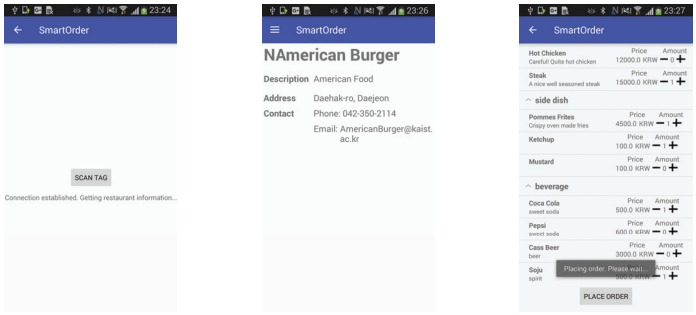

Figure 5 Screenshots of the Customer Application Prototype. 5a shows the NFC tag scanning screen. 5b shows the restaurant information screen and 5c shows the menu item screen

7. EVALUATION

To evaluate the implementation we try to address two main concerns:

1) Is the data rate of BLE enough for the transfer not to be noticeable to the user?

2) Is the range of BLE good enough to support common restaurant layouts?

These concerns will be addressed in the following sections.

Experimental Setup: For the experiment we use two Android devices to test the ROA and CA applications. The CA is run on a Samsung Galaxy S3, Android version 4.3 and API version 18 on board. The ROA is run on a Sony Z5, Android version 7.0 and API version 24 on board.

The SmartOrder applications are then tested in various environments that are similar to restaurant layouts.

7.1 Throughput

To test the throughput we tested the time it took to transfer both the menu and also the time it takes to submit an order. Both of these will vary heavily depending on their size. For the intent of testing, we defined the menu to have 10 items and the order to include three items of the menu.

Menu items: An average of 8 seconds were spent on receiving the menu items.

Submitting order: An average of 3 seconds were spent on submitting the order to the ROA.

While both of these are definitely slow and can be felt, it is arguably not a big problem since these operations occur so

infrequently. Furthermore, there are ways of addressing the throughput and increasing it, which will be discussed in the section VIII.

7.2 Range

Three scenarios were tested, all tests were performed by continuously increasing the distance until no communication were possible.

Straight line of sight: In many restaurants, there are little to no obstacles between the counter and the tables. To simulate this scenario the applications were run with each side standing as far away as possible. It was shown that the communication was still functional at a distance of 71 meters.

With tables and lower obstacles: To see if low obstacles such as tables or people would interfere, testing was performed indoor with this simulated. Results indicate that this had no effect, with easily achieving 37 meters, only limited by the size of the test building.

Around corners: Some restaurants feature a room layout that might be a bit more convoluted. To test this, the range was tested around corners by each side standing at an equal distance around a corner of a concrete wall. A maximum distance of 6 meters of each side (meaning 12 meters in total) was achieved.

As the results indicate, range seems to be of little concern, except if the counter where the ROA will be is around a corner and tables are spread far away on the other side. There are ways to address this, as will be discussed in section VIII.

8. DISCUSSION AND LIMITATIONS

Throughout the paper, several limitations have been mentioned, which will be addressed here. Furthermore, some additional thoughts on improvements and future work are also included.

1) *Increasing Throughput:* In later BLE specification it is possible to increase the Maximum Transmission Unit (MTU) size of the packets by a small amount. This can help alleviate the latency in a small way.

   Furthermore, another method of increasing the throughput, that is supported in all BLE devices is by increasing the amount of packets that are sent simultaneously in each connection window of the BLE communication. This is certainly feasible, but does add a lot of additional complexity for the client to server communication. A real-world implementation would indeed implement this. That said, this is outside the scope of this prototype implementation.

2) *Increasing Range:* To combat the problem of range dropping significantly around obstacles, such as building corners, BLE features the possibility of creating mesh networks. Such a network could be utilized to either extend the whole network by setting up fixed BLE repeaters, or by using the CA as a repeater device once it's connected, by letting it simultaneously act as a client and server.

3) *Notifications from ROA to CA:* Further enhancements to the user experience can be achieved by setting up characteristic notifications so the ROA can indicate order changes to the customer, such as delays or cancellations, effectively making two-way communication possible.

4) *Device Support:* While Android has supported passive NFC reading for quite a while, so far Apple has neither had the hardware support before iPhone 7 nor have they provided any APIs to read NFC chips. That said, as of time of writing, Apple has just announced the addition of NFC reading capabilities in iOS 11, which is currently in developer preview beta 1[12]. This means that the application of systems such as *SmartOrder*, that rely on NFC for ease-of-setup, can become a reality for a much larger audience than before.

9. FUTURE WORK

In this paper, we investigate the feasibility of an offline ordering system for restaurants that only uses commodity hardware (smartphones) and communicates entirely over BLE. The evaluation has shown, that this approach offers a great solution to overcome the drawbacks of current solutions.

In the future, we will implement a real-world application of the proposed offline ordering system for restaurants. This implementation contains the proposed methods to improve the limitation of the prototypical implementation, which were shown in Section VIII.

Furthermore, this proposed approach can be used not only as an offline ordering system for the restaurant. The use of commodity hardware and communicates completely via BLE also offers great potential in many other areas such as e-commerce and e-business. Therefore, we will evaluate the feasibility and effectiveness in these areas.

10. CONCLUSIONS

The project set out to show the feasibility of an ordering system communicating entirely over BLE using commodity devices and no Internet connection, and this goal has been achieved. By creating a prototype implementation of the concept, with an application for the restaurant owner and one for the customer, several concerns has been explored. The range of BLE has been shown to only cause minor concerns in convoluted room layouts, with potential solutions to this problem. Furthermore, the throughput of the BLE communication, while certainly slow, has been concluded to not be a problem with regards to the user experience. To this problem, there also exist potential enhancements that can be implemented in a final application. Finally, the usage of NFC has shown to great hand-in-hand with communication under the assumption of no Internet access, by allowing the NFC tags to supply the data needed to, in this instance, facilitate the BLE connection to the correct restaurant.

REFERENCES

1. M. Siekkinen, M. Hiienkari, J. K. Nurminen, and J. Nieminen, "How low energy is bluetooth low energy? comparative measurements with zigbee/802.15.4," in *2012 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, Apr. 2012, pp. 232–237. DOI: 10.1109/WCNCW.2012.6215496.
2. N. Hashim, N. Ali, A. Jaafar, N. Mohamad, L. Salahuddin, and N. Ishak, "Smart ordering system via blue-tooth," *International Journal of Computer Trends and Technology (IJCTT)-volume*, vol. 4, pp. 2253–2256, 2013.
3. Visionect. (2019). Tabletop ordering, [Online]. Available: https://www.visionect.com/blog/tabletop-ordering-the-digital-revolution-in-your-favorite-restaurant/ (visited on 06/09/2019).
4. M. A. Cotta, M. N. T. Devidas, M. A. Dias, M. S. N. Kalidas, and M. R. A. Tanaji, "Smart restaurant menu ordering system."
5. QikServe. (2019), [Online]. Available: https://www.qikserve.com (visited on 06/09/2019).
6. Android. (2019). SQLiteDatabase, [Online]. Available: https://developer.android.com/reference/android/database/sqlite/SQLiteDatabase (visited on 06/09/2019).
7. ——, (2019). NFC Basics, [Online]. Available: https://developer.android.com/guide/topics/connectivity/nfc/nfc (visited on 06/09/2019).
8. ——, (2019). Ndef, [Online]. Available: https://developer.android.com/reference/android/nfc/tech/Ndef (visited on 06/09/2019).
9. R. Networks. (2019). Android Beacon Library, [Online]. Available: https://altbeacon.github.io/android-beacon-library/beacon-transmitter-devices (visited on 06/09/2019).
10. C. Laustsen. (2019). KAIST CS442 project, [Online]. Available: https://github.com/Tehnix/SmartOrders (visited on 06/09/2019).
11. ——, (2019). SmartOrder App Demo, [Online]. Available: https://www.youtube.com/watch?v=8s4mnqKdtIA (visited on 06/09/2019).
12. Apple. (2019). Core NFC, [Online]. Available: https://developer.apple.com/documentation/corenfc (visited on 06/09/2019).