

# **UAB ECE DEPARTMENT**

## **Smart Indoor Navigation with Path Optimization**

### **An IoT Platform for Time Efficient Product Acquisition**

#### **Team 5**

Connor McGarty, cmcgarty@uab.edu

Jeremy Milam, jmilam89@uab.edu

Jake Watters, jakew21@uab.edu

Jongmoo Yim, dlawhdan@uab.edu

Advisor: Dr. Leon Jololian, leon@uab.edu

Instructors: S. Abdollah Mirbozorgi, samir@uab.edu

Abidin Yildirim, yildirim@uab.edu

**EE499 Spring 2020**

**Submitted**

**26 April 2020**

## **2. Summary**

The intention of this report is to describe the concept development and implementation of an IoT architecture that resolves deficiencies within several markets, such as big box retail shopping, warehouse management, distribution centers, and delivery services. Those deficiencies include length of time for successful product acquisition, low volume of sales, unmet revenue potential, and customer experience. The proposed solution utilizes IoT connected devices in conjunction with pathfinding algorithms, traffic optimization algorithms, machine learning, and integrated cloud computing. Several motivations exist for developing this technology. The primary reason is to satisfy the EE498/EE499 course requirements set forth by ABET and the UAB School of Engineering. Additionally, we intended to contribute to the scientific community, technological integration into relevant markets, and the professional development of individuals involved in this endeavor. The methods involved include identifying societal needs, concept development, academic research, developing system architecture diagrams, components analysis, workload management/planning, part selection, implementation, and testing.

### 3. Table of Contents

2.	Summary .....	ii
3.	Table of Contents .....	iii
4.	List of Figures .....	iv
5.	List of Tables .....	v
6.	Problem Definition .....	1
7.	Need Identification .....	1
8.	Design Concepts .....	2
	8.1 Product Requirements .....	2
	8.2 Product Constraints .....	3
	8.3 Block Diagram: Hardware .....	3
	8.4 Flow Chart .....	5
9.	Concept Evaluation and Selection .....	6
10.	Embodiment Design .....	10
	10.1 Distance Approximation .....	10
	10.2 Localization Techniques .....	13
	Fingerprinting .....	13
	Trilateration .....	15
	Multilateration .....	15
	10.7 Transmitter Beacon Layout .....	16
	10.8 Pathfinding .....	19
	10.9 Traffic Optimization .....	23
	10.10 Client/Server Operations .....	25
	10.11 Web Application .....	26
11.	Prototype Test Plan .....	27
12.	Future Work .....	29
13.	Societal, Ethical, Economic and Global Context .....	30
14.	Project Deliverables .....	31
15.	Project Management .....	35
	15.1 Gantt Chart .....	35
16.	Discussions .....	37
17.	Conclusions .....	38
18.	References .....	39
19.	Appendices .....	40
	A. Pathloss Model .....	40
	B. Datasheets .....	41
	Kontakt.io SB16-2 BLE Beacon Datasheet .....	41
	Raspberry Pi Model B Datasheet .....	42
	RAVPower 10000mAh 5V/3A Type-C Port Power Bank, Model: RP-PB078 Datasheet .....	48
	C. Parts List .....	48
20.	Closeout Memos .....	49

## **4. List of Figures**

- Fig. 1.** System Block Diagram
- Fig. 2.** Software Flowchart
- Fig. 3.** Pathloss Test
- Fig. 4.** Path loss model results
- Fig. 5.** RSSI Fingerprinting diagram
- Fig. 6.** Photographs of the test space (UAB EE Senior Design Lab)
- Fig. 7.** BLE Beacon Layout #1
- Fig. 8.** BLE Beacon Layout #2
- Fig. 9.** BLE Beacon Layout #3
- Fig. 10.** Matrix representation of walkable/unwalkable areas in Javascript server code
- Fig. 11.** Example output of pathfinding algorithm, user location (0,0)
- Fig. 12.** Example output of pathfinding algorithm, user location (0,4)
- Fig. 13.** Gantt Chart for project management

## **5. List of Tables**

**Table 1.** Decision table for beacon devices and corresponding communication protocol

**Table 2.** Decision table for server-side programming languages/frameworks

**Table 3.** Decision table for location receiver unit

**Table 4.** Decision table for product location schema

**Table 5.** Decision table for database software

**Table 6.** Components of software architecture

**Table 7.** Output Table

**Table 8.** Design Table

## **6. Problem Definition**

Often in consumer shopping situations, consumers are placed in large retail environments with little direction as to where the products they need can be found. Current solutions, like soliciting a customer service representative for assistance, or reading signage are time-consuming in the former case, or imprecise in the latter. Some retail environments allow look-up on the store's website by performing a search for a product, which may return a long list of similar products which the user would then need to take time to pick the one they are looking for, and eventually be given an aisle or bay number. This is a time-consuming process that must be repeated for every product a user needs, which could number in the hundreds. The nature of how we shop is changing every day, with delivery and pick-up services growing in market share, and so an efficient means of traversing a retail environment, grabbing needed items on the way, and ensuring no items are missed is becoming more relevant from both a user and store owner's perspective. The team's architecture intends to solve this problem.

## **7. Need Identification**

The architecture's design is abstract enough to be applied across several industries. For the consumer, it will provide a resilient indoor navigation system that allows for time efficient product procurement, product suggestions, intuitive graphical user interface, and provide a positive customer experience. Distribution services can utilize the modular technology to increase the volume of orders filled which in turn increases the rate of distribution, resulting in higher profits. Grocery and home delivery services will be able to utilize the time efficient procurement aspect to increase delivery volume, again resulting in higher revenue while retaining positive customer feedback.

## **8. Design Concepts**

The basis of the architecture features an indoor positioning system, which is made up of Bluetooth Low-Energy beacons affixed to walls, shelves, etc. that form a reference for determining the position of shopping carts within the environment. A user, through an iOS or Android mobile phone application, created using React Native, can create a list of products they wish to find and purchase at the store. Once they choose a shopping cart and pair the on-cart, battery-powered single-board computer with their mobile phone, a “turn-by-turn” navigation system is displayed to them, giving the user the most efficient path through the store to pick up all the items they placed in their list. The cart, through a client-server pattern that allows the carts’ single-board computer to communicate with a Node.js server utilizing the Express.js server framework, is then knowledgeable of the location of all other shopping carts that are traveling through the store, and thus is able to respond in real-time, providing an efficient path through a retail environment that avoids congestion. The server database also contains the list of items in the store and where they are located. As a cart passes by an item on the user’s list, the user is notified through the mobile app. The user can then pick up the item and mark it off their list, where they can then use the navigation system to navigate to the next item on the efficient path they are presented with.

### **8.1 Product Requirements**

The final product must meet several specifications to satisfy the user needs and be considered robust according to the design teams standards. An exhaustive list of requirements is presented below:

- The user location must be known
- The location of all items within the defined space must be known
- The user should have connection to a web server
- A server-side database should exist and be accessible by the user/cart
- An API with CRUD functionality must be available to the user
- A user interface should be accessible to the user
- Pathfinding must be provided
- The server must know the position of all users
- There should be an algorithm in conjunction with pathfinding to optimize traffic between all users in a defined space

## 8.2 Product Constraints

The final product also contains constraints that are outside the scope of this project. An exhaustive list is presented below as a guided effort to keep within the project scope of work and meet user needs without convolution.

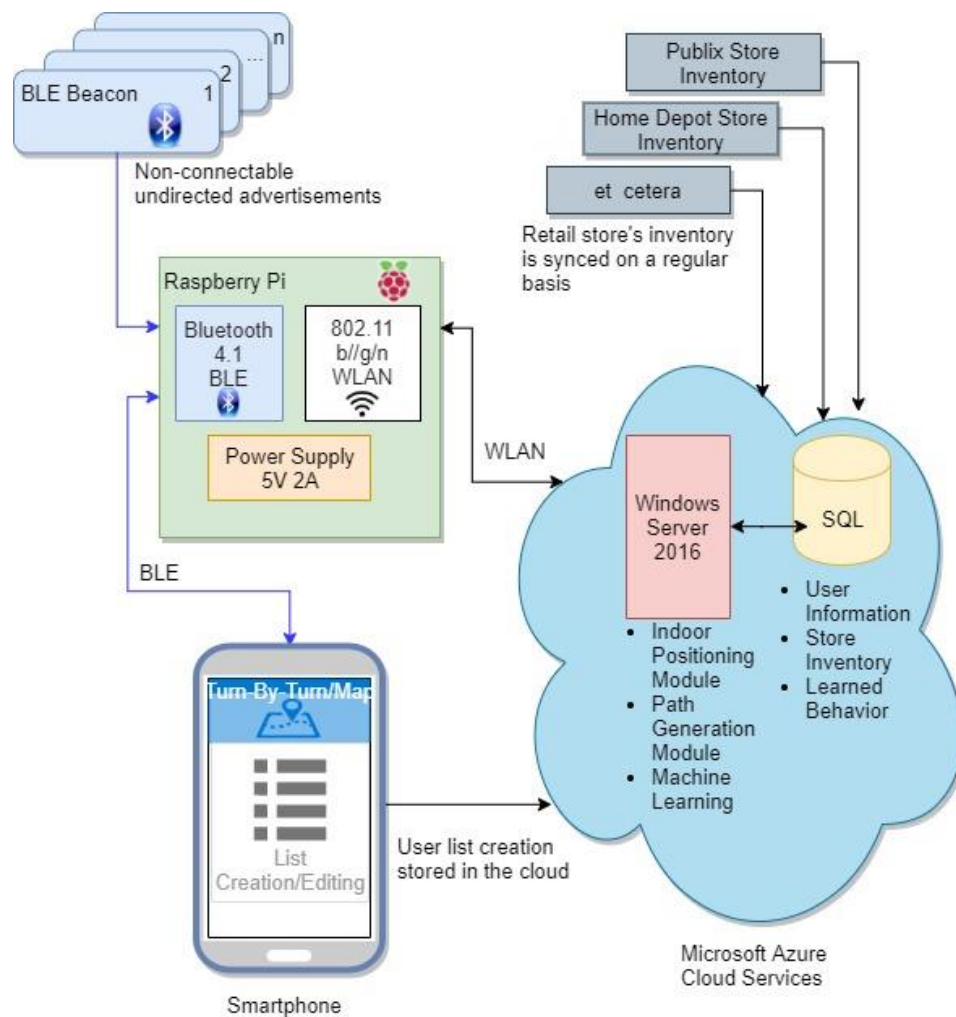
- User location shall not be known outside of a defined space
- A mesh topology among neighboring transmitter nodes shall not exist
- A mesh topology among cart receivers shall not exist
- Communication between cart receiver and user device is not necessary

## 8.3 Block Diagram: Hardware

A block diagram of a system shows the principal components and functions represented as blocks connected by arrows to components that indicate the relationship between blocks. In the case of this block diagram, the lines represent data being passed between components and web



servers. The diagram is used to represent the overall components excluding the details of the implementation. The components of the block diagram for a single user are BLE beacons, a Raspberry Pi, a smartphone and its application, a web computing server, and a database. The user will create a list of items via smart phone application, and that data will be recorded to the database. The user will then connect to the Raspberry Pi via Bluetooth, and microcontroller will detect and verify user identification through WLAN as well as the starting location. The server will return the path back to the Raspberry Pi, then back to the user via Bluetooth. The beacon will constantly ping the Raspberry Pi its location via BLE.

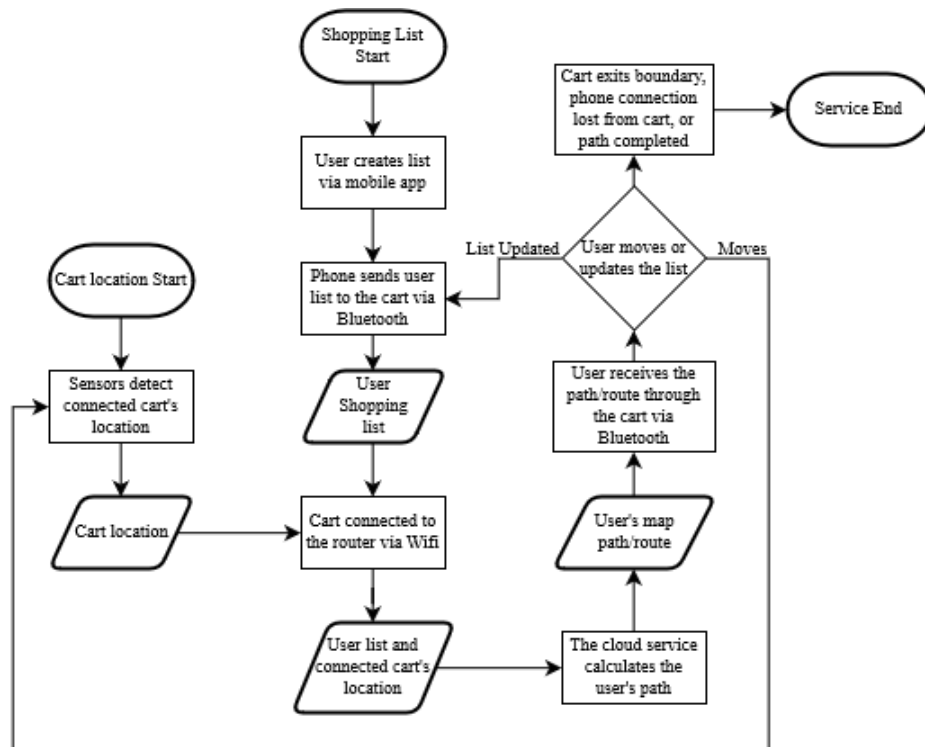


**Fig. 1.** System Block Diagram

## 8.4 Flow Chart

A flowchart is a diagrammatic representation of a system's process that illustrates a step-by-step solution to a given problem. The process at which diagram represents are indicated by different shapes of the block. The shapes and its nomenclature are as the following: arrowheads are flowlines, stadiums are terminals, rectangles are processes, diamonds are decisions, and parallelograms are inputs/outputs; flowlines indicate the process's order of operation, terminals represent the beginning or the ending of the system or a subsystem, processes show the operation performed that changes the value or data, decisions determine the conditional operation that decides different paths, and inputs/outputs indicates the subsequent entering/exiting data.

As shown in Figure 2, the system consists of two subcomponents, shopping list and cart location. The user begins the process of creating a list via mobile application that is stored in the database. The user connects to the Raspberry Pi (the cart) via Bluetooth, and the user shopping list is sent to the Raspberry Pi. Upon smartphone to cart connection, the Raspberry Pi will also read its location by receiving the pings from the BLE beacons. The data of user list and location is sent to the server via WLAN that is connected to within the establishment. The server will query the database and the received information and return the user map back to the Raspberry Pi. The user will then either move, update the list, or exit the app, and the flow repeats until service ends.



**Fig. 2.** Indoor Navigation Flowchart

## 9. Concept Evaluation and Selection

Before presenting the various sub systems it is important to reiterate the overall design approach. The primary goal, which was to assist the user in procuring items in an efficient manner would be the ultimate deciding factor in broad system design and component selection.

To satisfy the requirements and stay within the project scope meant exploring several alternate design configurations. The design team through an iterative process converged upon a final design. An attempt to quantify decision making for conceptual and component design was made using a tabulator approach presented below.

**Table 1.** Decision Table for Beacon devices and corresponding communication protocol

<b>Cart Location Devices</b>	<b>Standalone Hardware</b>	<b>Safety</b>	<b>Location Accuracy</b>	<b>Cost Effective</b>	<b>Range</b>	<b>Simplicity</b>	<b>TOTAL</b>
BLE Beacon	1	1	3	2	3	4	14
RFID	1	0	4	3	1	3	11
GPS	0	1	1	4	4	2	12
ZigBee	1	1	2	1	2	1	8

Presented in Table 2 is the decision table used to finalize which beacon devices and communication protocol to use. This was a very important decision to make, being that the choice made here consists of the fundamental layer of the indoor positioning system. Zigbee devices was the first option explored, but the complexity and price quickly ruled it out. Zigbee devices use a mesh topology, where the radio devices inter-communicate with one another, which is impractical and not necessary—the devices require more power and it would be more efficient to use the one-to-many connection architecture to reduce complexity and instead forward location information from a receiver to a server for processing. As for GPS, it lacks in accuracy when applied indoors. RFID was an economical option to obtain the precision we needed, yet to gain that precision it would require ultra-high frequency (UHF) RFID tags, which are unsafe for humans to be around for extended periods of time. BLE technology was feasible based on the simplicity of RF technology, price, and lots of literature on localization using BLE already exists.

**Table 2.** Decision table for server-side programming languages/frameworks

<b>Server-side Language and/or Frameworks</b>	<b>Speed</b>	<b>Prior Familiarity</b>	<b>Simplicity</b>	<b>Library/Module Support</b>	<b>Handles large Load/Concurrency</b>	<b>TOTAL</b>
Javascript/Node.js	4	2	4	4	4	18
C#/ASP.NET	3	3	3	3	3	15
Java	2	2	3	3	1	11

For the server-side software, the team wanted to use an asynchronous approach so that a prototype could easily scale to hundreds of simultaneous users—Node.js with the Express.js server framework is a commonly used modern solution for this kind of server programming. Additionally, the team was heavily leaning towards using Node.js on the single-board computer as well, since we were aware of pre-existing open-source libraries for receiving and parsing BLE beacon packets written for Node.js, not to mention it is a lightweight and high-level language that would perform well on a low-power single-board computer.

**Table 3.** Decision table for location receiver unit.

<b>Location Receiver</b>	<b>Standalone Hardware</b>	<b>Power Consumption</b>	<b>CPU clock speed</b>	<b>RAM</b>	<b>Cost</b>	<b>Total</b>
User Smartphone	N	N/A	N/A	N/A		N/A
RPi 4B+	Y	1	1	1	3	8
RPi Zero W	Y	3	2	2	1	6
Arduino	Y	2	3	3	2	10

The Raspberry Pi 4B+ 4GB was chosen for the prototype. The reason being was the amount of memory and processing power would be more than enough to handle receiving BLE advertising packets, doing some processing on them, and making web requests to the server. For the computing power they provide, they are a very low cost and low power option. Other computers, like the Raspberry Pi Zero W would have also provided an adequate amount of processing power, but the

team wanted to leave plenty of headroom both from a processing perspective and a peripheral perspective. This way, in the future, other components could be added on to better solve the problem and extend functionality. For instance, a UPC scanner could easily be added to allow users to scan items as they travel through, or a video camera that could assist with positioning using image recognition techniques.

**Table 4.** Decision table for product location schema.

<b>Product Location Associated To:</b>	<b>Complexity</b>	<b>Scalability</b>	<b>Processing Utilization</b>	<b>Hardware Limitation</b>	<b>Total</b>
Location	2	2	2	2	8
Server with Beacon Number	1	1	1	1	4

Another decision to be made was how to represent the product location in the database. One option was to associate items with nearby beacons' IDs. This did not appear to be the optimal choice because the beacon range proved to have a maximum 3m radius. Associating items with the beacons would render an inaccurate representation as to the actual location of the items. Instead the choice was made to tie the products to their true physical location in coordinate format. This way, there is no fixed association between a beacon and product(s), so the location of items is abstracted away from the indoor position system.

**Table 5.** Decision table for database software.

Database	Relational DB	Prior Familiarity
SQL DB	Y	Y
Mongo DB	N	N

Since the product requires storing many different types of data to make the system work, a database was required. User account credentials, user data, product and beacon locations all need to be stored and accessed programmatically. Based on the kind of data that the product requires, the team decided that it lent well to relational data storage provided by SQL-like databases. The team chose PostgreSQL since it is widely supported on all operating system and there is a large amount of documentation and tutorials on its use available online.

## **10. Embodiment Design**

### **10.1 Distance Approximation**

Received Signal Strength Indicator (RSSI) is a parameter dependent on broadcast power and distance from the RF transmitter. This parameter can be used to approximate distance between receiver and transmitter. Other factors influence radio wave propagation that include absorption, interference and diffraction, hence RSSI tends to fluctuate. For these reasons, utilizing RSSI can be very difficult. RSSI of signals broadcast by Bluetooth beacons follow the subsequent path loss model:

$$\text{RSSI} = -10n \log_{10} \frac{D}{D_0} + C_0 \quad (1)$$

where RSSI is the measured RSSI value,  $D$  is the distance from receiver to transmitter,  $n$  is a freespace constant representing the medium of transmission,  $C_0$  is the RSSI value at a reference

distance  $D_0$ , which is typically 1 meter. Rearranging to solve for a distance given an RSSI value leaves us with

$$D = 10^{\frac{RSSI+C_0}{-10n}} \quad (2)$$

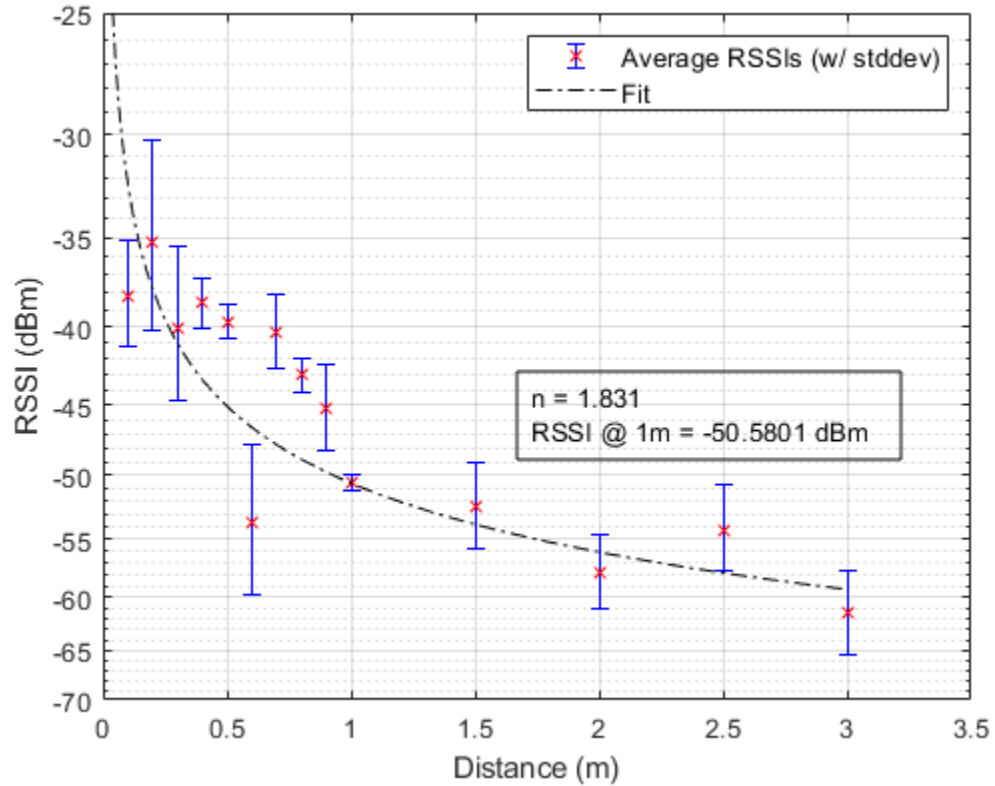
In order to obtain the best possible estimation for the distance from a Bluetooth beacon transmitter to a receiver,  $n$  needs to be optimally determined for the given environment. The best procedure for determining this is via experimentation. The team set up an experiment used in a paper by Mackey et al. titled “Improving BLE Beacon Proximity Estimation Accuracy through Bayesian Filtering”. A BLE Beacon served as the transmitter and a Raspberry Pi 4 Model B as the receiver. The beacon’s transmission parameters were set to an advertising interval of 350ms and a transmission power of +0dBm. The transmitting beacon and the Raspberry Pi receiver were stationed at equal height of about 1m from the ground and in free space on stools so that the positions between could be adjusted.





**Fig. 3.** Picture of setup for path-loss test. Raspberry Pi 4 Model B is seen on the chair to the left, with the BLE beacon on the right. The chairs were pulled further apart to the proper distance for each part of the test.

For 2 minutes at each position, the Raspberry Pi recorded the RSSI of each packet the BLE beacon transmitted, resulting in approximately 350 readings for each position. The test was paused, the distance between the receiver and transmitter was adjusted, and the test was resumed for another 2 minutes at this position. For the test at each position, the RSSI values of each packet were written to a text file. Tests were conducted at 14 positions: in 10cm intervals from 10cm to 1m, and then at 0.5m intervals from 1.5 to 3.0m. Using MATLAB, the data was imported and the mean and standard deviation at each position was calculated. Using MATLAB's Curve Fitting Toolbox, the data was fit to Eq. 2, where  $C_0$  was set to the average value of the RSSI at 1m, which was -50.58 dBm, which resulted in an  $n = 1.831$  constant.



**Fig. 4.** Results of pathloss model experiment. Results fitted to Equation 1. The

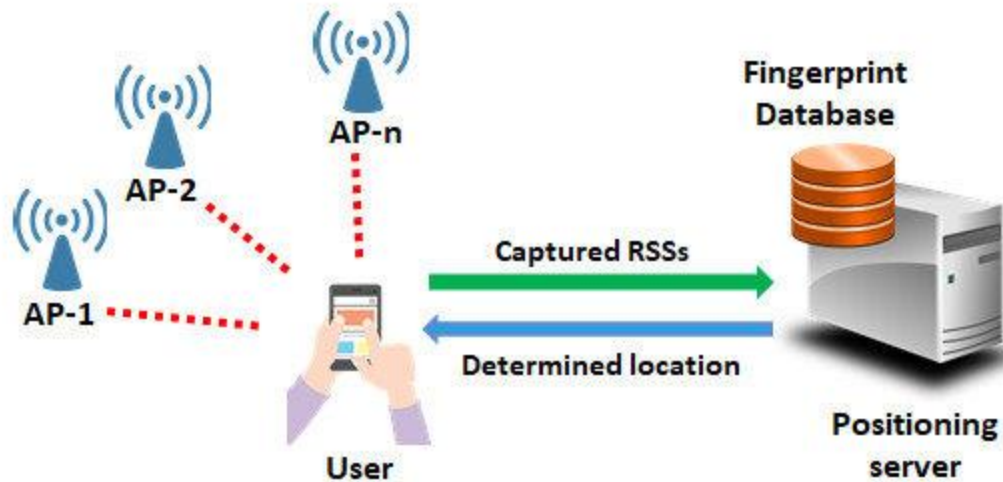
logarithmic decay behavior is clearly shown as distance from the transmitter increases.

## 10.2 Localization Techniques

Having successfully implemented a path loss model that approximates distance in terms of RSSI, it was feasible to obtain user location within a defined space. Several methods for approximating location within a defined 2-dimensional space were considered and tested. Those methods include fingerprinting, trilateration, and multilateration.

### Fingerprinting

The first method discussed was the fingerprinting method. Fingerprinting is a common method of localization used in industry.



**Fig. 5.** Fingerprinting Method for Obtaining User Location [11]

As depicted in Fig. 4, the receiver captures several RSSI values and then sends them to the web server. The server compares RSSI values to a predetermined set of RSSI values that correspond to a defined coordinate location and selects the coordinates with the best matching RSSI and sends the obtained location back to the client-side interface. To obtain this, an extensive “setup” phase is required, where the test space is meticulously walked through while RSSI and location of the receiver is recorded, so that this information can be saved to a database for later reference. The RSSI value or values at each defined location in the space is recorded and logged in a database. When the system is in use, the receiver that navigates within the spaces reports to the fingerprinting server what RSSI’s it is receiving, and the server compares them to the values in the database. The server then reports back to the receiver what its location should be based on the data obtained from the setup. The team determined that This was not implemented after discussions about scalability and labor burdens during the initial setup.

## Trilateration

Another method tested was trilateration. Trilateration is possible when the distance between a moving receiver and three stationary transmitters is known, as well as the position of the transmitters. Using the center of the transmitters as the center of a circle, and the distances calculated from the procedures in Sections 10.1 & 10.2 as the radii of those circles, a point on the perimeter of the three circles can be established as the point of the receiver.

Trilateration is a commonly used technique in localization. At first the team settled on trilateration as the main technique for determining approximate position. Using the NodeJS package node-trilateration [15], the team easily could use the distance approximation that is discussed in Section 10.1 & 10.2 to feed into the node-trilateration module to receive back an estimated position. During testing we used trilateration extensively as we tweaked other parameters of the system such as beacon advertising interval, beacon transmission power, and beacon placement. In general, the Raspberry Pi receiver would read in BLE beacon advertising packets in 2 second intervals, average the RSSI value from each beacon for the previous interval, and send this information along to the server in an HTTP POST request. The server then picks the three strongest signal averages (under the assumption that these would be the beacons that the server is the closest to) and estimates the position using distance approximations to those beacons, then trilaterating them.

## Multilateration

Eventually, the team decided to try multilateration based on the hypothesis that increasing the number of fixed anchor points that the receiver's position could be extrapolated from, the better the system could react to inaccuracies and fluctuations due to the nature of radio transmission in indoor spaces.

Multilateration follows the same principle as trilateration (Section 10.4), just using more fixed base stations (BLE beacons, in our case) to extrapolate position from. The team decided to attempt using more than just the three strongest signals, and instead considering the average RSSI from four to six of the strongest signals. The team found that while accuracy was not greatly increased, the stability of the location results increased, and the presence of erroneous, wildly inaccurate positions decreased.

The team utilized the NodeJS package multilateration [[19](#)] in the NodeJS server to perform multilateration of distance approximation to return a user's location.

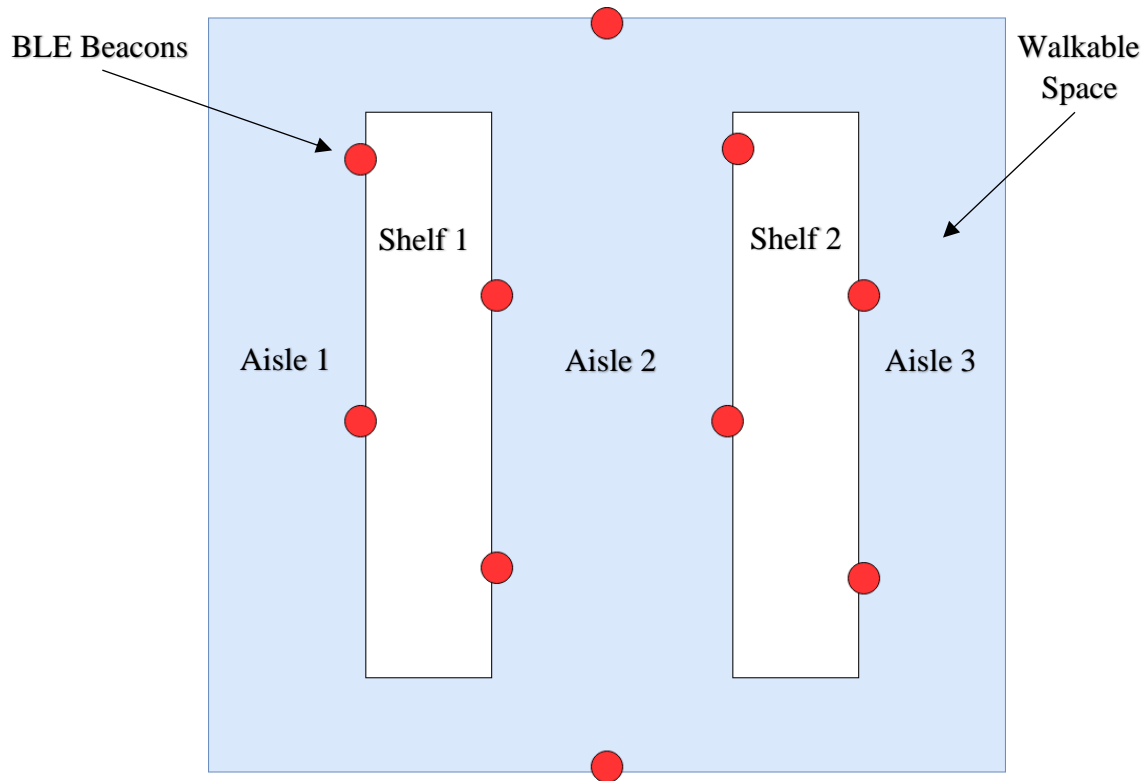
### 10.7 Transmitter Beacon Layout



**Fig. 6.** Photographs of the test space which served as the team's "grocery store", the EE499 design lab.

The team required a location to set up and use as a testing ground for the project. For the budget and resources the team had at its disposal, the UAB EE499 design lab was chosen since it was a reasonably small but sufficient space to emulate a small grocery or convenience store with two, long lab benches that could serve as "shelves" and three wide "aisles" (see Fig. 6). Based on

that location accuracy is highly dependent on beacon placement, the team evaluated multiple different beacon layouts to best utilize the 10 BLE beacons available.

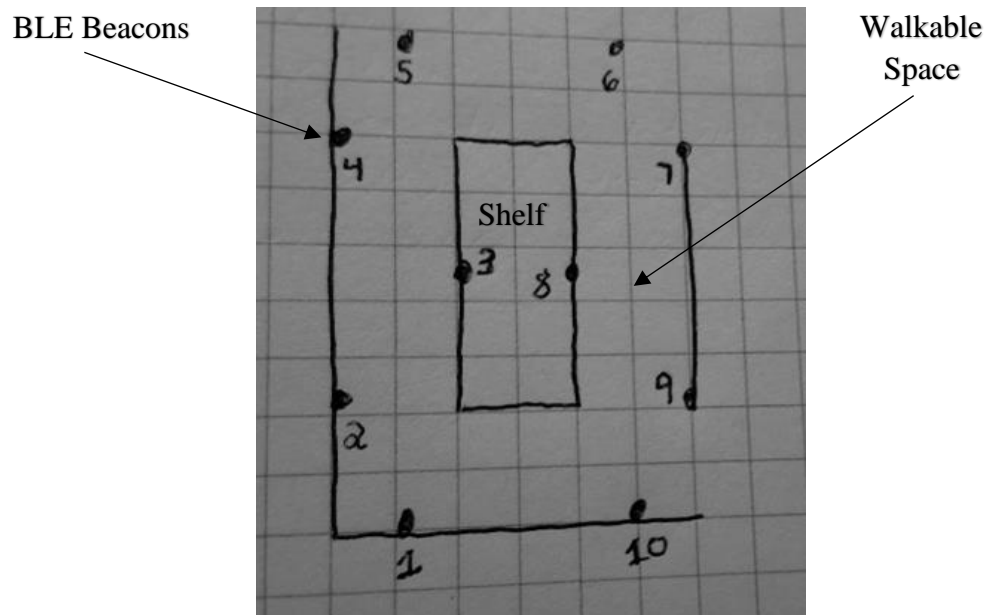


**Fig. 7.** Layout 1: The two-shelf, three-aisle layout the team experimented with for the first half of the semester.

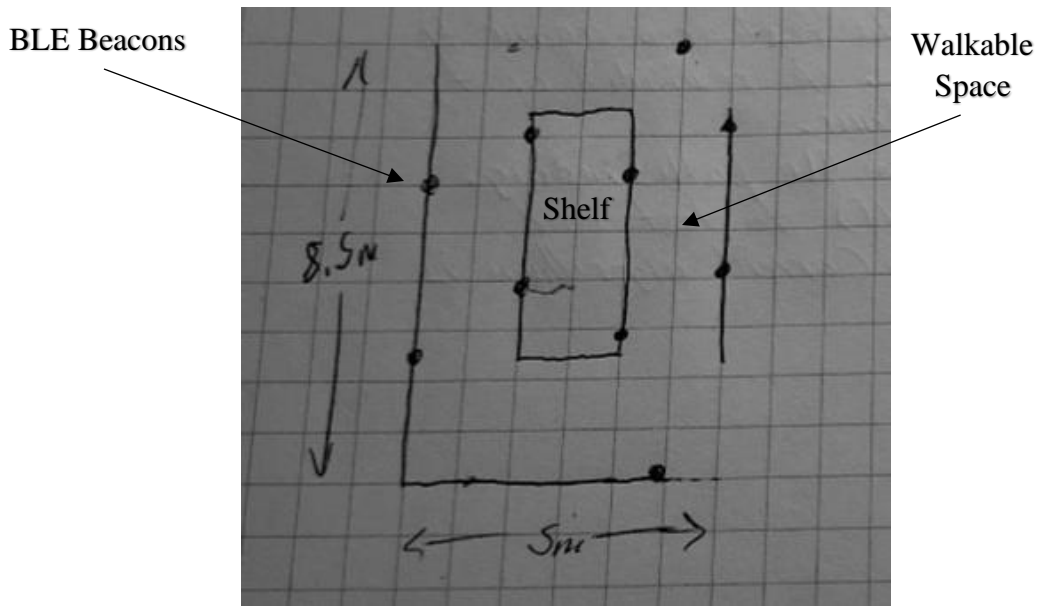
The original layout the team developed the project upon can be seen in Fig. 7. The ten BLE beacons were placed attempting to maximize coverage of the space, such that a user walking in the aisles was always within at least 3 meters of multiple beacons. The location tracking for this layout unfortunately proved difficult and unstable for parts of the space, specifically the perimeters. The team believed that the lack of being surrounded by beacons greatly diminished the accuracy of the distance approximation and multilateration procedures, resulting in erroneous location reporting. For a receiver walking between the two aisles, the location estimation was very

accurate to a sub-meter level. However, when walking around the perimeters of the space the location estimations could vary wildly. The team determined that issue results from the user not constantly being surrounded on multiple sides in geometric space. For example, when a user is walking in the leftmost aisle in Fig. 7, the Raspberry Pi receiver is only receiving advertising packets from BLE beacons located on its righthand side, which results in issues for the multilateration algorithm.

For the given space, the only solution with the given project plan would be to acquire more beacons to better cover the space, by placing an additional 2 to 4 beacons around the perimeter. Although the team had leftover budget that could have afforded enough beacons to properly cover the space, due to time constraints we were not able to procure more BLE beacons. The only other solution would be to reduce the size of the test space. As such, the team tried another layout, reducing the space to just two aisles and one shelf (see Figures 8 and 9).



**Fig. 8.** Alternative Layout 2: Reduction in area of the test space to surround the user and cart receiver to improve accuracy



**Fig. 9.** Alternative Layout 3: This concentrates more beacon density to the middle of the aisles and less on the near and far edges of the space.

In the week prior to Spring Break, the team was experimenting with these smaller test spaces in order to obtain more accurate location estimation in hopes to provide a proper demonstration of the final product. Due to the events of the 2020 COVID-19 pandemic and move to online classes and learning, testing an indoor space no longer became possible, and in consultation with team design project instructor Dr. Yildirim and team mentor Dr. Jololian the team moved forward with simulating the location of a user in order to demonstrate pathfinding and other elements of the client and server software.

## 10.8 Pathfinding

Pathfinding is the plotting of the shortest route between two or more points. In computer applications this requires the use of certain algorithms. These algorithms work on graphs in the mathematical sense as a set of vertices with edges that connect them. These graphs can be either



two-dimensional or three-dimensional. In this project we used a two-dimensional graph or matrix. This matrix is comprised of 1's and 0's. Every position in the matrix with a value of 1 is considered unwalkable and positions with a value of 0 are walkable. In the context of our test space the store aisles are walkable, and the shelves are not. The matrix that represents the layout of the test space is shown below in Figure 10.

```
matrix=[
  [0, 0, 0, 0, 0, 0, 0, 0, 0],
  [0, 0, 1, 0, 0, 0, 1, 0, 0],
  [0, 0, 1, 0, 0, 0, 1, 0, 0],
  [0, 0, 1, 0, 0, 0, 1, 0, 0],
  [0, 0, 1, 0, 0, 0, 1, 0, 0],
  [0, 0, 1, 0, 0, 0, 1, 0, 0],
  [0, 0, 1, 0, 0, 0, 1, 0, 0],
  [0, 0, 1, 0, 0, 0, 1, 0, 0],
  [0, 0, 0, 0, 0, 0, 0, 0, 0],
];|
```

**Fig. 10.** The test space represented in matrix form (JavaScript).

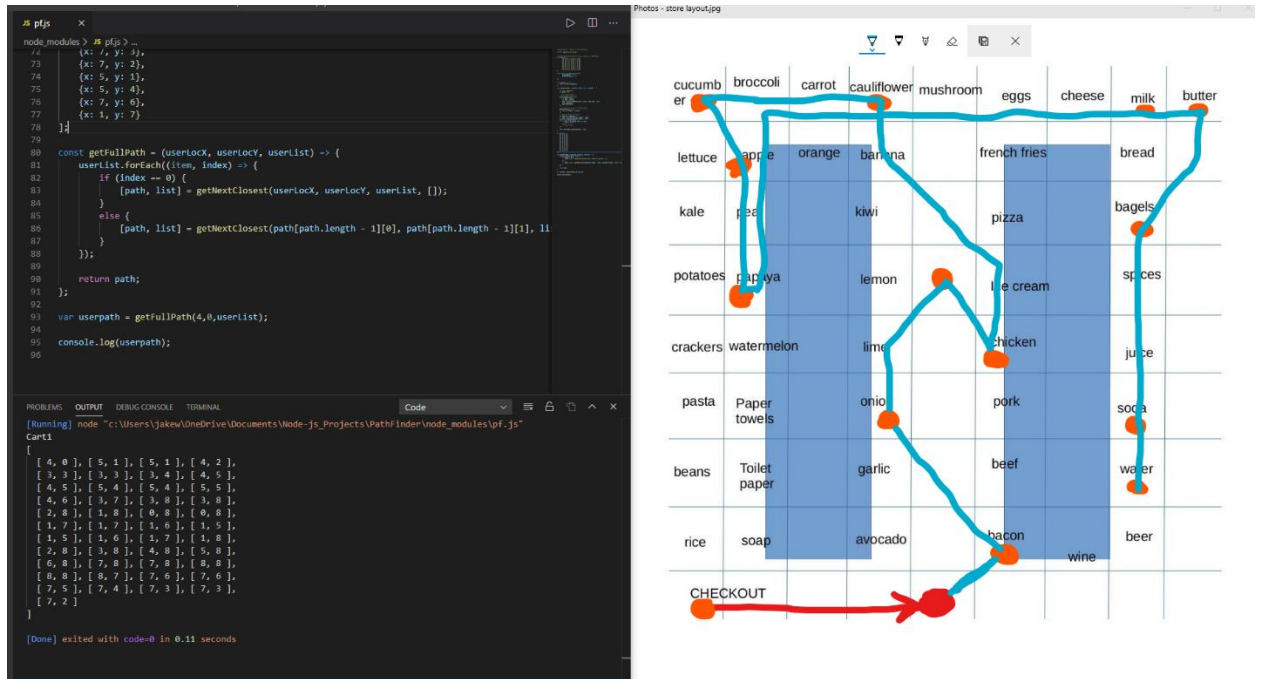
Pathfinding algorithms are required to determine the shortest path possible. There are several algorithms to find the shortest path. Some of these algorithms are Breadth First Search (BFS), Dijkstra, and A\*. The Breadth First Search algorithm visits and marks all the key nodes in a graph in a systematic breadthwise pattern. It selects a single node, initial or source point, in a graph and then inspects all the nodes bordering the selected node one at a time [14]. Dijkstra's Algorithm works by visiting vertices in the graph starting with the object's starting point. It then repeatedly examines the closest not-yet-examined vertex, adding its vertices to the set of vertices to be examined. It expands outwards from the starting point until it reaches the goal [16]. Dijkstra's algorithm is an example of a greedy algorithm because it chooses the closest perimeter vertex at every step. The A\* algorithm is built upon the Dijkstra algorithm. However, A\* also includes a

forward-looking component, which is an estimate of the length to complete the path to the destination from a specific node [17]. This forward-looking component, the heuristic value, is the estimated cost of moving from the current node to the final node. The actual cost cannot be calculated until the final node is reached; hence, it is the estimated cost. The heuristic value speeds up the shortest path search. A\* is both complete and optimal if an admissible heuristic is involved [18]. If no heuristic is stated or set to 0, using A\* by default performs as Dijkstra's. The key difference between Dijkstra and A\* is that A\* focuses to reach the goal node from the current node, not to reach every other node.

The pathfinding algorithm chosen for this design is the A\* algorithm. It was chosen due to its versatility and faster search capability and it most matches the design's need to find the shortest possible path. It's versatile because its performance can be modified based on what is used for the heuristic. In our design the heuristic is set to Octile. This allows the pathfinding algorithm to forward-look in eight directions for each node it processes. In the test space each position enables for movement in eight directions. This allows the cart to move diagonally in addition to the four cardinal directions. The main reason that A\* is used in this design is its focus to find the shortest path from starting node to end node. The starting node for this design is the current position of the user's cart. The end node is the item that user wants to find.

The design implements A\* in a program written in JavaScript. The node.js package that contains the A\* algorithm is called PathFinding.js [20]. The program receives the cart's current location and the list of items and their locations that a path needs to be generated for. The program already contains the 2D matrix or grid such as the one stated previously. This matrix can be specifically designed to represent any physical layout that would use the program. This is an important feature as it does not require any other changes to the code. The next major step is that





**Fig. 12.** Example output of the pathfinding code when the cart moves to location [0,4].

## 10.9 Traffic Optimization

A feature planned to be in this design was the aspect of optimizing traffic using the pathfinding algorithm. This would entail the location of all carts with their respective paths and the ability to optimize the paths considering all the carts. In theory the traffic optimization would work in conjunction with the pathfinding function to keep record of all the carts and based on that knowledge be able to adjust the path for the optimal route.

The first iteration of this design was to create the test space matrix as single instance or singleton. When called the pathfinding would use an instance of the matrix to calculate that cart's specific path. It would also update the matrix and specify the cart's location coordinate as '1' or unwalkable. This would allow the matrix to be dynamically updated with every cart's current location. With that location set as unwalkable the next instance of the matrix used in the

pathfinding would be able to avoid or optimize the path considering all the other carts. In example would be 'cart1' accesses the pathfinding to find the shortest path to its item list. The path would be created, and the instance of the matrix would be updated with cart1's location as unwalkable. Cart2 would then access the pathfinding for its path but this instance of the matrix would have cart1's location as unwalkable and therefore avoiding cart1 in its path output.

The major problem with this design was with the occurrence of one cart being at a location that contained an item at the same time as another cart was requesting a path to that same location. This would cause the pathfinding to error as it cannot create a path with an end point that is unwalkable. The proposed correction would be to 'catch' the error and have the cart wait a specific amount of time and then repeat the pathfinding request in hopes that the other cart had moved to another location. The expected problem with this fix action was as more carts were in use this could cause more occurrences of wait times and a possible degradation of service would be seen by the user.

The design of this system uses a Node.js package for pathfinding as to save time in the development and implementation of this project. It is feasible with more time and under different academic circumstances that a new package could be developed to correct these issues. Unfortunately, it was decided to drop the traffic optimization from the design. It should be noted though that the pathfinding program does find the shortest path and inherently it is optimized.

## 10.10 Client/Server Operations

The project's software architecture is divided into four main components:

**Table 6.** Components of software architecture.

<b><u>Subsystem</u></b>	<b><u>General Purpose</u></b>
BLE Beacons	Base stations for referencing location of the microcontroller. Broadcasts advertising packets.
Microcontroller client	Receiver for BLE advertising packets. Collects RSSI from the packets and sends to server for position computation.
User Interface Client	Interface for user to interact with the system. User creates lists of products, checks items off their list as they navigate, and is shown their determined best path.
Cloud Server	Receives RSSI values from microcontroller, computes user location. Exposes CRUD API for user interface with the database. Delivers web content for user interface.
Cloud Database	Stores all user account information, user's lists, product database.

All of the subsystems in Table 7 are used throughout the typical use case. Initially, a user will log in through a web page served by the cloud server. The user will enter in to the user interface a unique code to associate their account with a specific Raspberry Pi microcontroller, which serves as a receiver for the advertising packets broadcast by the BLE Beacons. The Raspberry Pi collects the RSSI values from the packets it receives, and on a two-second interval, averages the values from each beacon and packages it into a JSON object which is sent to the cloud server via POST request. When the server receives a POST request from a Raspberry Pi, it parses the RSSI JSON object and keeps the top four beacons that the RPi received the strongest signals from. Theoretically, these four beacons are those that the user is closest to. The server then calculates the approximate distance to each of those four beacons using the path-loss model described in

Section 10.1. Once an approximate distance to four beacons is calculated, these distances and the beacons they are computed to are passed along to the multilateration algorithm. This algorithm outputs the estimated location of the user.

In order to provide an estimation of an efficient path, two pieces of data must be known: the user's location, and the locations of the items in the user's shopping list. The user can create shopping lists in via the web application as well, and when the user's location is updated in the server, the list and the current location is passed along to the pathfinding subsystem on the server, calling the `getFullPath()` function. This returns a list of x-y coordinates that represent the most efficient path through the space to reach all the items provided. Each time the user's current location is updated (when the microcontroller has read BLE packets for another two seconds and POST requested the results to the server), this pathfinding function is called again, ensuring that the user always has a freshly calculated path in case they deviated from the one recommended to them.

### 10.11 Web Application

The web application serves as interface for user input and output where users can interact to acquire desired list of items at specified location with live navigation. The web application handles user interactions and internal operating objects to return the navigation to the user. A user may select as many desired items to create a custom items list to be picked up at a location, and upon selection, the web application will call for two key operations, user location and pathfinding algorithm.

During EE498, the team planned to use React Native to develop a smartphone application to serve as the user interface to the system. This application would allow the user to log in, edit and create shopping lists, pair themselves with a shopping cart microcontroller, and then see a top-

down layout of the store with their location, path, and list items displayed on top. React Native [21] is a JavaScript framework for writing user interfaces for native smartphone applications. React Native apps are written in JavaScript (using the React.js framework). At runtime, React Native creates the corresponding platform specific components for either Android or iOS. Android app development is done in Java or Kotlin, while iOS development requires development in Swift or Objective-C. In order to deliver a cross-platform solution with half the work, the team felt it was a good design decision to develop the application in React Native.

Due to time pressures and move to online classes from COVID-19 circumstances, the team in consultation with the team mentor felt that a proof-of-concept using a basic HTML webpage would be more achievable by the end of the EE499 semester. The web app allow for login and list creation. The user can also enter the navigation page, which shows an HTML table mimicking the store layout. As mentioned in Section 10.7, the location estimation layer of the project is being simulated for demonstration purposes. To achieve this, the HTML web app allows the user to manually input their location using navigational buttons on the web page, to move the user up, down, left, and right through the space. When the user updates their position via the web page, it calls the same code that would have previously had the microcontroller sent RSSI data to the server, updating the user's location and then calling the pathfinding code to return a user's path to its list items.

## **11. Prototype Test Plan**

To meet the requirements established by the UAB Electrical Engineering faculty and those realized by the design team, it is necessary to form a method of validation. An attempt was made to adhere to the afore mentioned requirements during implementation. Due to time constraints and



the COVID-19 circumstances, the prototype delivered by the end of the semester did not completely embody the proposed design of the previous semester. Those circumstances meant working remotely without access to vital components such as the microcontrollers and BLE beacons. Because of these hindrances, the scope was reduced to meet baseline yet fundamental project goals. Those goals were:

1. Can user position within a defined region be approximated using BLE beacons?
2. Was the user provided an efficient path through the defined space?
3. Based on the user deviating from the path or some other circumstance can the path be updated?

To show proof of concept for approximating user position required two pieces of evidence. First, the design team graphically displayed the path loss model as proof of concept that distance can be estimated using received signal strength indicator. Secondly, the team provided a simple comparison between the users known location and the approximated location provided from software. This portion of the project was demonstrated to Dr. Yildirim midway through the semester.

To validate the goal of providing an efficient path to procure items, a basic web application was developed to allow the user to input a shopping list and input current location, the application return an efficient path through the environment. This web application provides a graphical example of what the user would similarly see on the proposed smartphone application, and additionally demonstrates the pathfinding element of the project. It should be noted that although the system, for the final demonstration, is not utilizing an indoor positioning system to receive updated user location, and it is instead being controlled manually by the “user”, the components of the software that generate the paths and display them to the user are unchanged. From the

perspective of these portions of the software, there is no difference between the user location being updated by the Raspberry Pi or manually by the user.

## **12. Future Work**

One area to consider for future enhancements is the use of Bluetooth Core Specification v5.1 and higher standards. Significant to 5.1 was the addition of two new methods for direction finding. The two methods are called Angle of Arrival (AoA) and Angle of Departure (AoD). Each of the techniques requires one of the two communicating devices to have an array of multiple antennae, with the antenna array included in the receiving device when the AoA method is used and in the transmitting device when using AoD. Bluetooth Core Specification v5.1 gives the Bluetooth Low Energy (LE) controller in the receiving device the ability to generate data that can then be used to calculate the directional angle to the transmitting device [22]. Using devices implemented with the new standards could add more accuracy to the direction finding and localization aspect of this project. Unfortunately, this standard was new and very few devices utilized these enhancements at the time of this design's conception. This addition to the standards could be a huge benefit to future indoor positioning systems.

Another item to consider for future work on this project would be the number of beacons. While the range and ability to get RSSI readings was not an issue, the lack of accurate readings were. As stated in the embodiment design, the accuracy of readings greatly improved when they were taken totally enclosed by beacons. The edges of the test space lacked the extra beacons for total coverage and therefore accuracy suffered.

Another area for consideration for future work is the development and implementation of traffic optimization. In this design the use of a pathfinding library saved valuable time and

resources. The downside was that it locked the design into a specific way of executing, therefore limiting the possibilities of building upon the code to meet design concepts. In the future, and with more time, an attempt on implementing our own pathfinding algorithms that is more allows for more customization should be pursued to better fit the application.

An obvious area of future work to deliver a full prototype would be to continue to develop the React Native smartphone application that the team originally planned for in the design. Due to COVID-19 circumstances, and as is described in Section 11, the team decided to develop a simpler, proof-of-concept web application that achieved much of the same functionality as the phone application was designed to. Some of the team members plan to continue work on this after graduation for self-enrichment and self-learning.

Finally, modifications could be made to the Raspberry Pi/single-board computer portion of the project to increase functionality. As mentioned in Section 9 under Table 4, the particular model was chosen such that in the future, it would not require a different on-cart computer to extend hardware functionality to components such as image-recognition cameras, other sensors to aid in location estimation (RFID, infrared, ultrasonic, etc.), a UPC scanner to allow users to scan items as they pick them up, and more.

### **13. Societal, Ethical, Economic and Global Context**

Important to any new technology introduced is to attempt to foresee the possible implications it will have. The technology being introduced would disrupt certain markets. One major disruption would be big box retail stores. It will shift the methods of how the customer shops. Instead of time spent walking liberally through the store the shopper will be more deliberate with their movements. This could reduce “impulse buys” which the grocery stores look to for

increased revenue. However, there exists an opportunity for a shift in marketing strategies. With any new method of consuming comes the advantage to find more creative ways of encouraging spending.

To the contrary, the delivery services industry will be positively impacted by this technology. The primary purpose of the technology is to assist the end user in acquiring items in a time efficient manner. Because of this the delivery services will be able to capitalize on the increased performance and boost sales volume.

Because the technology presented is intrinsically abstract, the applications extend into the industry of sports and entertainment venues. Perhaps food and beverage delivered to the spectator's seat. Another extension of application would be distribution warehouse functions and asset tracking.

Possible societal and ethical impacts to the user could be that increased screen time inhibits human interaction and engagement, which could negatively affect the quality of life to the user, hence contributing the ever-present phenomenon of "technology addiction".

## **14. Project Deliverables**

The output table for the project is list of acceptance criteria that will verify our design expectations. The table has five main categories: Connection Speed, Data Rate, Power Consumption, Battery Life, and Software. The reason for the categories is that as the different elements of the project are completed, they can be tested individually. Under these categories are lists of specific output parameters to be tested and verified. For each specific parameter there is an expected and measured value.

**Table 7.** Output Table

<b>PARAMETERS</b>	<b>EXPECTED</b>	<b>MEASURED</b>	<b>UNIT</b>
<b>CONNECTION SPEED:</b>			
Beacon to Cart	<5		ms
Cart to Server	<10		ms
Web Computation Speed	<150		ms
Beacon Message Interval	350		ms
<b>DATA RATE:</b>			
Microcontroller-Server	20		Mbps
Phone-Microcontroller	2		Mbps
<b>POWER CONSUMPTION:</b>			
Beacon	62.3		uAh
Raspberry Pi	1200		mAh
<b>BATTERY LIFE:</b>			
Raspberry Pi Battery Output	5		V
Beacon	22		Months
Raspberry Pi 4 (2Gb)	6.66		Hr
Current Draw RPi	1500		mA
Battery Pack	10000		mAh
<b>SOFTWARE:</b>			
All 4 carts appear on Map	Yes		Y/N
User can create, read, update,	Yes		Y/N
Map updates with new	Yes		Y/N
Beacon Range	70		m
Position Accuracy	1		m

The *Input Table* or *Design Table* (Table 9) is an engineering design tool that “clarifies the design strategy and approach” for the project. The table is used to define the project’s scope, requirements, and need, from which things like the project’s specifications, features, and limits can be derived. During the initial phases of idea conception, some possible applications to the abstract idea of the project were determined and are found in the first problem. In the following column, the unifying need or problem that encapsulated these applications is recorded. After that, the most basic, abstract, defining system needs and requirements are noted, for instance we need to know a shopping cart’s location, the location of products available, user data including

their lists of desired products, etc. From these requirements of the system, the components that make it up were determined. As an example, the “cart location” requirement involves three components: the fixed-position BLE beacons, the BLE receiver on the cart’s microcontroller, and a connection between that microcontroller and the server. Then a justification for how the component satisfies the requirement is recorded, to ensure completeness and help think through the architecture. From this point, hardware and software choices are recorded, so a clear link between want individual *specific* component is tied to its reasons for being and justification. Then, important parameters like current/voltage/power requirement is tracked and defined.

**Table 8. Design Table**

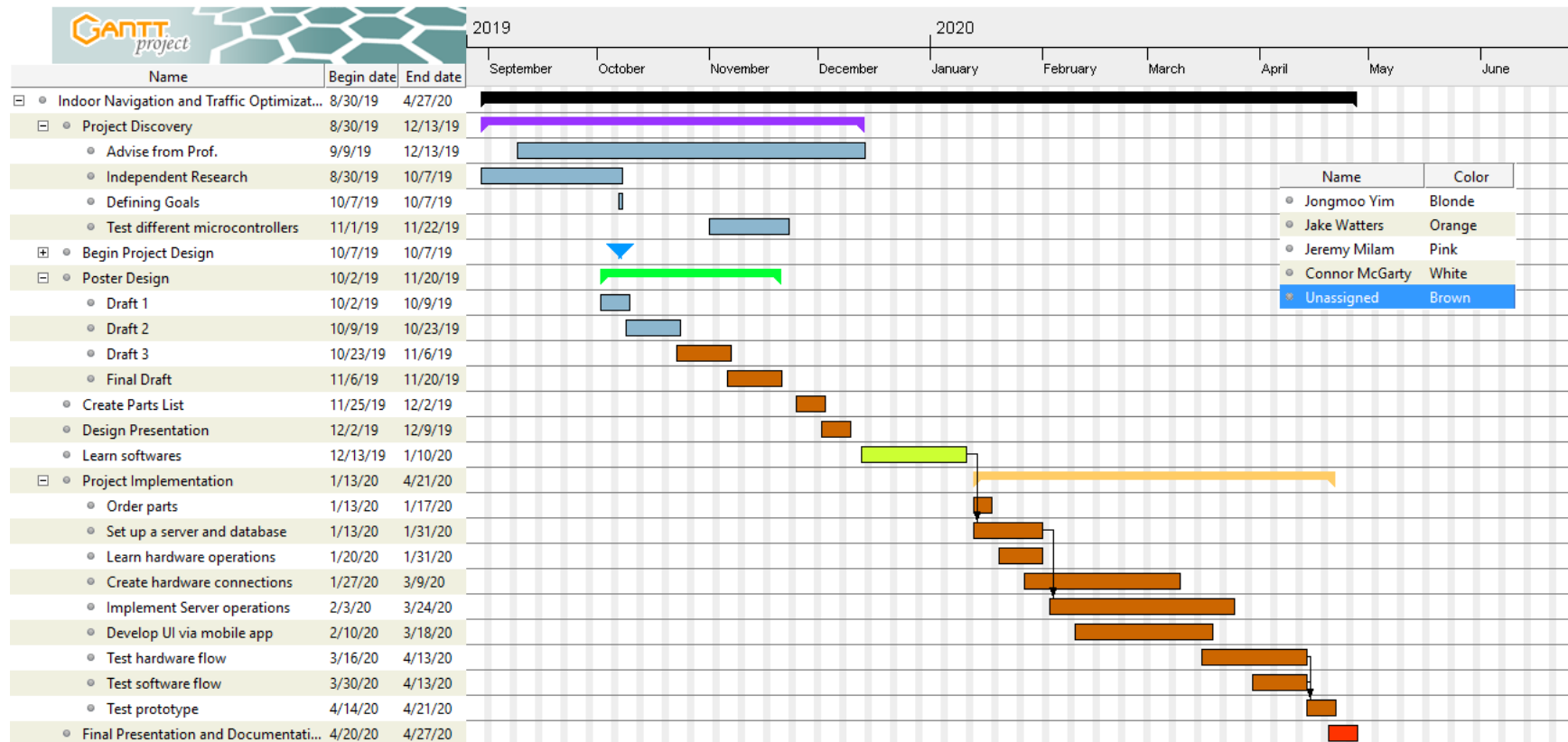
APPLICATIONS	PROBLEM	NEED/REQUIREMENT	INVOLVED SYSTEM COMPONENT(S)	HOW COMPONENT ACCOMPLISHES NEED	HARDWARE	POWER	SOFTWARE	PRICE	QUANTITY	PART NUMBER	LINKS
Workers on dock (e.g., Shipt & Walmart Pickup) to pick up list of items in the most efficient manner	Indoor Route Optimization to Save Customer Time and Provide Retail Analytics	Cart Location	Beacon (IPS Node)	Forms reference points for determining cart location	BC037-iBeacon Blue Charms	CR2032 (included)	Configurable by vendor provided mobile app	\$18	10	BC-037G	<a href="https://bluecharmbeacons.com/product/blue-charm-bc037-i-beacon">https://bluecharmbeacons.com/product/blue-charm-bc037-i-beacon</a>
					Gimbal	4 AA Batteries (included)		\$25	10	Series 21	<a href="https://store.gimbal.com/collections/beacons/products/s21">https://store.gimbal.com/collections/beacons/products/s21</a>
					Kontakt.io	CR2477 (included)		\$21	10	Smart Beacon SB16-2	<a href="https://store.kontakt.io/our-products/30-smart-beacon-sb16-2.html">https://store.kontakt.io/our-products/30-smart-beacon-sb16-2.html</a>
			Cart receiver	Receives signals from beacons to estimate distance and triangulate position	Raspberry Pi Battery	USB Li-Ion Power Bank with 2 x 5V Outputs @ 2.1A - 5000mAh	\$19	4	4288	<a href="https://www.adafruit.com/product/4288">https://www.adafruit.com/product/4288</a>	
			RPI Zero W OR 3B+	Min/Max : RPI 3b+ 450mA - 1.2A, RPI Zero 100mA - 350mA	Raspbian OS	RPI Zero W: \$15 , OR RPI 3B+: \$36	4	3400 or 3775	<a href="https://www.adafruit.com">https://www.adafruit.com</a>		
Deliver list of items indoors to multiple people (i.e., highly-populous stadium)		Cart connection to server	Cart sends received signals from beacons to server for processing	Samsung 32GB SD Card		Python/Node.js script	1			<a href="https://www.amazon.com/Samsung-Class-Micro-Adapter-MB-MC32GA/AM">https://www.amazon.com/Samsung-Class-Micro-Adapter-MB-MC32GA/AM</a>	
		Product Location	Server-side database	Connects a physical location in-store to what products are nearby							
		Store and edit users' list data	Server-side database	Stores user's login information and their shopping lists		MySQL/PostgreSQL					
CRUD API functions			Allows user to create, read, update, and delete lists		Node.js/Express.js on Azure/AWS/Firebase						
Hospital workers navigate optimal path to prioritized patients			List creation/adding/editing/deleting	User interface (mobile application)	Provides the user a way to interact with the system		React.js				
Typical consumers in retail store to get in & out, picking up their items in most efficient manner			Turn-by-turn navigation GUI								
Contractor is billing an employee time to pick up materials in a hardware store/warehouse and wants to ensure the exact items are picked up in shortest time possible			User needs most efficient path to pick up items	Server-side pathfinding algorithm	Algorithm considers user's needed products, current location, and other carts' locations to deliver best path		Node.js				
			Optimize traffic	Server simultaneously knows position of all carts	Path algorithm can consider every agent in the system and deliver new path based on this data						
		Remind user products they usually pickup but are not on their list	Machine-learning algorithm makes connections based on prior shopping data	Past user behavior can inform users on suggested products	Weka						

## **15. Project Management**

### **15.1 Gantt Chart**

A Gantt chart functions as a project scheduling tool that aids in developing project timelines and tracking progress. It was invented by Henry Gantt in the early nineteen-hundreds. The list of tasks to be performed are located on the vertical axis and the time intervals are on the horizontal axis. The tasks to be performed can be broken into blocks of tasks that house related sub tasks for further project clarity. Task dependencies can be specified by an arrow to show the need for a proceeding task to be finished before the dependent task can be started. Furthermore, color coding can be used to assign a specific task to an individual or department.





**Fig. 13.** Gantt Chart

## **16. Discussions**

The design and implementation of an IoT system aimed at helping users procure items in a time efficient manner proved feasible. Indoor positioning systems fall into the category deemed “The Location of Things (LoT)” is a growing industry with ever expanding potential for applications. This design project proved unique because it is the combination of several technologies incorporated for a host of applications.

The utilization of BLE beacons in indoor positioning systems in the scope of this project is an economical solution because they are relatively cheap and have little operation maintenance because of the long battery life. Moderate accuracy was accomplished by using received signal strength in conjunction with multilateration techniques. There is still a need to explore other methods of manipulating RSSI to determine user position.

The use of pathfinding algorithms for creating efficient paths proved satisfactory. The team chose to use the A\* algorithm, and is a sufficient approach to providing the user with efficient navigation in an indoor environment. Areas of improvement should be made by adding custom heuristics to decrease permutations during processing, and to account for random human behavior.

## **17. Conclusions**

Progress was made in the design and prototyping of an IoT system intended for providing efficient navigation in an indoor environment. Specifically, big box retail settings where consumers need to acquire many items. Several fields of study touched on in this report include localization techniques using BLE technology, utilizing RF signal strength for distance approximation, pathfinding algorithms, and communication protocols. Simplicity was sought in effort to preserve replication and scalability. Satisfying the EE499 course requirement set forth by ABET and the UAB School of Engineering was the primary motivations for contributing to this field of study. Additionally, we intended to contribute to the scientific community, technological integration into relevant markets, and the professional development of individuals involved in this endeavor. The methods involved include identifying societal needs, concept development, academic research, developing system architecture diagrams, components analysis, workload management/planning, part selection, implementation, and testing.

## 18. References

1. [Machine learning methods for predicting traffic based on traffic history](#)
2. [White paper on Indoor Positioning systems that discusses multiple technologies \(RFID, Wi-Fi, BLE\)](#)
3. [Case Study: Installing RFID Systems in Supermarkets](#)
4. [Evolution of Indoor Positioning Technologies: A Survey](#)
5. [Foglight: Visible Light-enabled Indoor Localization System for Low-power IoT Devices](#)
6. [Video: BLE Beacon Scanner using Golang w/ Raspberry Pi](#)
7. [Video: BLE Beacon Scanning using Node.js w/ Raspberry Pi](#)
8. [An indoor Bluetooth-based positioning system: concept, Implementation and experimental evaluation](#)
9. [Indoor Positioning Based on Bluetooth Low-Energy Beacons Adopting Graph Optimization](#)
10. [Comparative Analysis of Pathfinding Algorithms A \\*, Dijkstra, and BFS on Maze Runner Game](#)
11. [https://www.researchgate.net/figure/Illustration-of-Wi-Fi-fingerprint-based-indoor-positioning-system\\_fig1\\_331020073](https://www.researchgate.net/figure/Illustration-of-Wi-Fi-fingerprint-based-indoor-positioning-system_fig1_331020073)
12. [https://www.researchgate.net/figure/Indoor-location-measurement-using-trilateration\\_fig1\\_328948760](https://www.researchgate.net/figure/Indoor-location-measurement-using-trilateration_fig1_328948760)
13. [Alpha Multipliers Breadth-First Search Technique for Resource Discovery in Unstructured Peer-to-Peer Networks](#)
15. <https://www.npmjs.com/package/node-trilateration>
16. <http://theory.stanford.edu/~amitp/GameProgramming/AStarComparison.html>
17. [A Survey of Shortest-Path Algorithms](#)
18. [A Guide to Heuristic-based Path Planning](#)
19. <https://www.npmjs.com/package/multilateration>
20. [PathFinding.js](#)
21. [React.js](#)
22. [Bluetooth Core Specification v5.1](#)
23. [Kontakt.io SB16-2 BLE Beacon](#)
24. [Raspberry Pi 4 Model B 2GB](#)
25. [USB C Power Bank RAVPower 10000mAh Portable Charger](#)
26. [Samsung 32GB EVO Plus Class 10 Micro SDHC with Adapter](#)

## 19. Appendices

### A. Pathloss Model

The following MATLAB script was used to characterize the transmission of BLE Beacon packets in the test space. With this, the average RSSI at a distance of 1 meter and  $n$  describing the test space could be estimated (see Section 10.1 for details).

```
close all;
files = {'0_1meters'; '0_2meters'; '0_3meters'; '0_4meters'; '0_5meters';
        '0_6meters'; '0_7meters'; '0_8meters'; '0_9meters'; '1_0meters';
        '1_5meters'; '2_0meters'; '2_5meters'; '3_0meters'};

distance = [.1 .2 .3 .4 .5 .6 .7 .8 .9 1 1.5 2 2.5 3];
rssis = {};
for file = [1 : 1 : length(files)]
    fid = fopen(files{file, 1}, 'r');
    counter = 1;
    readings = [];
    while(~feof(fid))
        line = str2num(fgets(fid));
        readings(counter) = line;
        counter = counter + 1;
    end
    rssis{file} = readings;
end
for index = [1 : 1 : length(rssis)]
    avg(index) = mean(rssis{1,index});
    variance(index) = var(rssis{1,index});
    stddev(index) = std(rssis{1,index});
end
[fittedmodel, gof] = createFit(distance, avg); % edit the fitted model
w/ curve fitting app. then regen code from the fit session
e = errorbar(distance, avg, stddev, 'xb');
hold on;
fitted = plot(fittedmodel, '-.k');
grid on;
set(gca, 'YScale', 'log');
set(e, 'MarkerEdgeColor', 'r');
xlim([0, 3.5]);
ylim([-70, -25]);
legend(gca, 'Average RSSIs (w/ stddev)', 'Fit');
xlabel('Distance (m)');
ylabel('RSSI (dBm)');
anno_str = {'n = 1.831', 'RSSI @ 1m = -50.5801 dBm'};
annotation('textbox',[.5 .2 .3 .3], 'String', anno_str, 'FitBoxToText', 'on');
legend('FontSize',10);
```

## B. Datasheets

### Kontakt.io SB16-2 BLE Beacon Datasheet

<b>Dimensions and weight</b>	Height	15 mm (0.59 in)
	Width	55 mm (2.16 in)
	Height	56 mm (2.20 in)
	Depth	35 grams (1.23 oz)
<b>Microcontroller</b>		Nordic Semiconductor nRF51822
<b>Connectivity</b>	Bluetooth	Bluetooth 4.2 compliant
	Range	up to 70 meters
	Available transmission power levels	0 (-30dBm), 1 (-20dBm), 2 (-16dBm), 3 (-12dBm), 4 (-8dBm), 5 (-4dBm), 6 (0dBm), 7 (4dBm)
	Sensitivity	-93dBm
<b>Battery and Power</b>		Two CR2477 Lithium Manganese Dioxide Coin Battery with capacity of 1000 mAh each - battery powered device lifetime up to 60 months Batteries are replaceable
<b>Sensors</b>		None
<b>Casing</b>	Material	LUPOY GN5001RFG
	Colours	White or Black*
	Protection	IP-54
	Splashproof	Yes
	Dustproof	Yes
	Customisation	logo printed on front and casing's colour
	Flame resistance	Safe - V0 flammability class
	Antibacterial Surface	Upon request
<b>Environmental Requirements</b>	Operating Temperature	-20°C / + 60°C (-4°F / +140°F)
	Storage temperature	-5°C / +35°C (23°F / +95°F)
	Humidity (non-condensing)	from 0% up to 95%
<b>Warranty</b>		12 months
<b>Beacon standards compatibility**</b>	iBeacon	UID, URL, TLM, EID, ETLM
	Eddystone	
	Kontakt.io Secure Profile	

\* please contact sales if you want to order black beacons

\*\* All packets can be enabled and disabled at any time, except Kontakt.io Secure Profile (it's constantly broadcasted at a fixed 500ms interval)

#### Battery life\*

Tx Power / Interval (ms)	Tx=3 (-12dbm)	Tx=7
1000 ms	60 months	57 months
800 ms	59 months	47 months
500 ms	39 months	31 months
350 ms	27 months	22 months



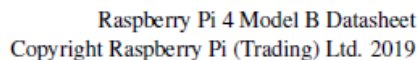
## 2 Features

### 2.1 Hardware

- Quad core 64-bit ARM-Cortex A72 running at 1.5GHz
- 1, 2 and 4 Gigabyte LPDDR4 RAM options
- H.265 (HEVC) hardware decode (up to 4Kp60)
- H.264 hardware decode (up to 1080p60)
- VideoCore VI 3D Graphics
- Supports dual HDMI display output up to 4Kp60

### 2.2 Interfaces

- 802.11 b/g/n/ac Wireless LAN
- Bluetooth 5.0 with BLE
- 1x SD Card
- 2x micro-HDMI ports supporting dual displays up to 4Kp60 resolution
- 2x USB2 ports
- 2x USB3 ports
- 1x Gigabit Ethernet port (supports PoE with add-on PoE HAT)
- 1x Raspberry Pi camera port (2-lane MIPI CSI)
- 1x Raspberry Pi display port (2-lane MIPI DSI)
- 28x user GPIO supporting various interface options:
  - Up to 6x UART
  - Up to 6x I2C
  - Up to 5x SPI
  - 1x SDIO interface
  - 1x DPI (Parallel RGB Display)
  - 1x PCM
  - Up to 2x PWM channels
  - Up to 3x GPCLK outputs



- ARMv8 Instruction Set
- Mature Linux software stack
- Actively developed and maintained
  - Recent Linux kernel support
  - Many drivers upstreamed
  - Stable and well supported userland
  - Availability of GPU functions using standard APIs

[illegible]

Figure 1: Mechanical Dimensions

**Caution!** Stresses above those listed in Table 2 may cause permanent damage to the device. This is a stress rating only; functional operation of the device under these or any other conditions above those listed in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.





Symbol	Parameter	Minimum	Maximum	Unit
VIN	5V Input Voltage	-0.5	6.0	V

Table 2: Absolute Maximum Ratings

Please note that VDD\_IO is the GPIO bank voltage which is tied to the on-board 3.3V supply rail.

Symbol	Parameter	Conditions	Minimum	Typical	Maximum	Unit
$V_{IL}$	Input low voltage <sup>a</sup>	VDD_IO = 3.3V	-	-	TBD	V
$V_{IH}$	Input high voltage <sup>a</sup>	VDD_IO = 3.3V	TBD	-	-	V
$I_{IL}$	Input leakage current	TA = +85°C	-	-	TBD	μA
$C_{IN}$	Input capacitance	-	-	TBD	-	pF
$V_{OL}$	Output low voltage <sup>b</sup>	VDD_IO = 3.3V, IOL = -2mA	-	-	TBD	V
$V_{OH}$	Output high voltage <sup>b</sup>	VDD_IO = 3.3V, IOH = 2mA	TBD	-	-	V
$I_{OL}$	Output low current <sup>c</sup>	VDD_IO = 3.3V, VO = 0.4V	TBD	-	-	mA
$I_{OH}$	Output high current <sup>c</sup>	VDD_IO = 3.3V, VO = 2.3V	TBD	-	-	mA
$R_{PU}$	Pullup resistor	-	TBD	-	TBD	kΩ
$R_{PD}$	Pulldown resistor	-	TBD	-	TBD	kΩ

<sup>a</sup> Hysteresis enabled

<sup>b</sup> Default drive strength (8mA)

<sup>c</sup> Maximum drive strength (16mA)

Table 3: DC Characteristics

Pin Name	Symbol	Parameter	Minimum	Typical	Maximum	Unit
Digital outputs	$t_{rise}$	10-90% rise time <sup>a</sup>	-	TBD	-	ns
Digital outputs	$t_{fall}$	90-10% fall time <sup>a</sup>	-	TBD	-	ns

<sup>a</sup> Default drive strength, CL = 5pF, VDD\_IO = 3.3V

Table 4: Digital I/O Pin AC Characteristics





## 4.1 Power Requirements

The Pi4B requires a good quality USB-C power supply capable of delivering 5V at 3A. If attached downstream USB devices consume less than 500mA, a 5V, 2.5A supply may be used.

# 5 Peripherals

## 5.1 GPIO Interface

The Pi4B makes 28 BCM2711 GPIOs available via a standard Raspberry Pi 40-pin header. This header is backwards compatible with all previous Raspberry Pi boards with a 40-way header.

### 5.1.1 GPIO Pin Assignments

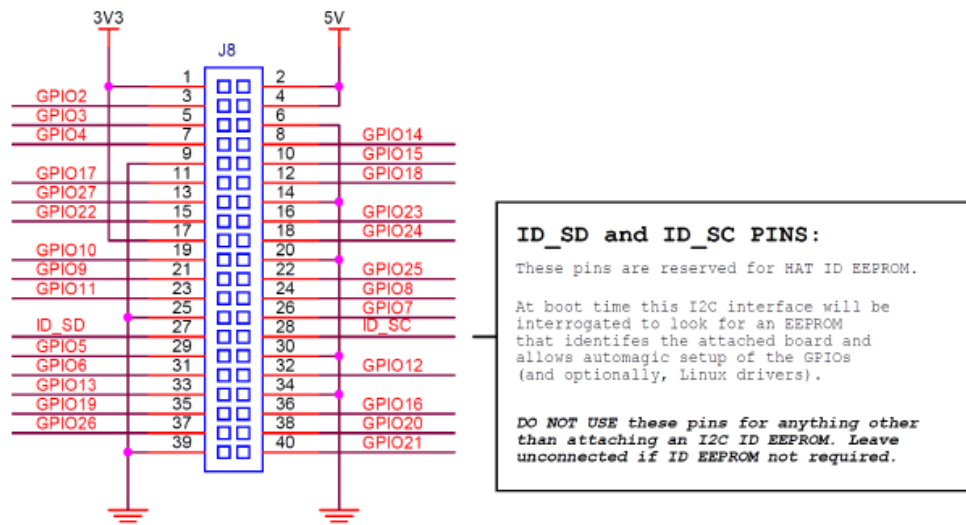


Figure 3: GPIO Connector Pinout

As well as being able to be used as straightforward software controlled input and output (with programmable pulls), GPIO pins can be switched (multiplexed) into various other modes backed by dedicated peripheral blocks such as I2C, UART and SPI.

In addition to the standard peripheral options found on legacy Pis, extra I2C, UART and SPI peripherals have been added to the BCM2711 chip and are available as further mux options on the Pi4. This gives users much more flexibility when attaching add-on hardware as compared to older models.



### 5.1.2 GPIO Alternate Functions

GPIO	Default Pull	ALT0	ALT1	ALT2	ALT3	ALT4	ALT5
0	High	SDA0	SA5	PCLK	SPI3_CE0_N	TXD2	SDA6
1	High	SCL0	SA4	DE	SPI3_MISO	RXD2	SCL6
2	High	SDA1	SA3	LCD_VSYNC	SPI3_MOSI	CTS2	SDA3
3	High	SCL1	SA2	LCD_HSYNC	SPI3_SCLK	RTS2	SCL3
4	High	GPCLK0	SA1	DPLD0	SPI4_CE0_N	TXD3	SDA3
5	High	GPCLK1	SA0	DPLD1	SPI4_MISO	RXD3	SCL3
6	High	GPCLK2	SOE_N	DPLD2	SPI4_MOSI	CTS3	SDA4
7	High	SPI0_CE1_N	SWE_N	DPLD3	SPI4_SCLK	RTS3	SCL4
8	High	SPI0_CE0_N	SD0	DPLD4	-	TXD4	SDA4
9	Low	SPI0_MISO	SD1	DPLD5	-	RXD4	SCL4
10	Low	SPI0_MOSI	SD2	DPLD6	-	CTS4	SDA5
11	Low	SPI0_SCLK	SD3	DPLD7	-	RTS4	SCL5
12	Low	PWM0	SD4	DPLD8	SPI5_CE0_N	TXD5	SDA5
13	Low	PWM1	SD5	DPLD9	SPI5_MISO	RXD5	SCL5
14	Low	TXD0	SD6	DPLD10	SPI5_MOSI	CTS5	TXD1
15	Low	RXD0	SD7	DPLD11	SPI5_SCLK	RTS5	RXD1
16	Low	FL0	SD8	DPLD12	CTS0	SPI1_CE2_N	CTS1
17	Low	FL1	SD9	DPLD13	RTS0	SPI1_CE1_N	RTS1
18	Low	PCM_CLK	SD10	DPLD14	SPI6_CE0_N	SPI1_CE0_N	PWM0
19	Low	PCM_FS	SD11	DPLD15	SPI6_MISO	SPI1_MISO	PWM1
20	Low	PCM_DIN	SD12	DPLD16	SPI6_MOSI	SPI1_MOSI	GPCLK0
21	Low	PCM_DOUT	SD13	DPLD17	SPI6_SCLK	SPI1_SCLK	GPCLK1
22	Low	SD0_CLK	SD14	DPLD18	SD1_CLK	ARM_TRST	SDA6
23	Low	SD0_CMD	SD15	DPLD19	SD1_CMD	ARM_RTCK	SCL6
24	Low	SD0_DAT0	SD16	DPLD20	SD1_DAT0	ARM_TDO	SPI3_CE1_N
25	Low	SD0_DAT1	SD17	DPLD21	SD1_DAT1	ARM_TCK	SPI4_CE1_N
26	Low	SD0_DAT2	TE0	DPLD22	SD1_DAT2	ARM_TDI	SPI5_CE1_N
27	Low	SD0_DAT3	TE1	DPLD23	SD1_DAT3	ARM_TMS	SPI6_CE1_N

Table 5: Raspberry Pi 4 GPIO Alternate Functions

Table 5 details the default pin pull state and available alternate GPIO functions. Most of these alternate peripheral functions are described in detail in the BCM2711 Peripherals Specification document which can be downloaded from the [hardware documentation](#) section of the website.



### 5.1.3 Display Parallel Interface (DPI)

A standard parallel RGB (DPI) interface is available the GPIOs. This up-to-24-bit parallel interface can support a secondary display.

### 5.1.4 SD/SDIO Interface

The Pi4B has a dedicated SD card socket which supports 1.8V, DDR50 mode (at a peak bandwidth of 50 Megabytes / sec). In addition, a legacy SDIO interface is available on the GPIO pins.

## 5.2 Camera and Display Interfaces

The Pi4B has 1x Raspberry Pi 2-lane MIPI CSI Camera and 1x Raspberry Pi 2-lane MIPI DSI Display connector. These connectors are backwards compatible with legacy Raspberry Pi boards, and support all of the available Raspberry Pi camera and display peripherals.

## 5.3 USB

The Pi4B has 2x USB2 and 2x USB3 type-A sockets. Downstream USB current is limited to approximately 1.1A in aggregate over the four sockets.

## 5.4 HDMI

The Pi4B has 2x micro-HDMI ports, both of which support CEC and HDMI 2.0 with resolutions up to 4Kp60.

## 5.5 Audio and Composite (TV Out)

The Pi4B supports near-CD-quality analogue audio output and composite TV-output via a 4-ring TRS 'A/V' jack.

The analog audio output can drive 32 Ohm headphones directly.






## 5.6 Temperature Range and Thermals

The recommended ambient operating temperature range is 0 to 50 degrees Celcius.

To reduce thermal output when idling or under light load, the Pi4B reduces the CPU clock speed and voltage. During heavier load the speed and voltage (and hence thermal output) are increased. The internal governor will throttle back both the CPU speed and voltage to make sure the CPU temperature never exceeds 85 degrees C.

The Pi4B will operate perfectly well without any extra cooling and is designed for sprint performance - expecting a light use case on average and ramping up the CPU speed when needed (e.g. when loading a webpage). If a user wishes to load the system continually or operate it at a high temperature at full performance, further cooling may be needed.

RAVPower 10000mAh 5V/3A Type-C Port Power Bank, Model: RP-PB078 Datasheet

 Capacity	10000mAh / 38Wh
 Input	Mirco USB: DC 5V / 2A Max USB-C: DC 5V / 3A Max
 Output	DC 5V / 3.1A Total iSmart 5V / 2.4A Max USB-C 5V / 3A Max
 Dimensions	14.5 x 7 x 1.4 cm / 5.7 x 2.8 x 0.6 in
 Weight	210 g / 7.4 oz

### C. Parts List

<u>Name</u>	<u>Supplier</u>	<u>Part Number</u>	<u>Description</u>	<u>Price</u>	<u>Quantity</u>
Smart Beacon SB16-2	KontaktIO	SB16-2	Bluetooth Low-Energy beacon	\$21 (@ 10 qty)	10
Raspberry Pi 4 Model B, 2GB	Newark	02AH3162	Raspberry Pi microcontroller	\$45	4
RAVPower 10000mAh Portable 5V/3A USB-C Battery Bank	RAVPower	(ASIN) B077CY4M8P	RPi power supply	\$27.00	4
Samsung 32 Gb SD Card	Samsung	MB-MC32GA/AM	Samsung 32GB EVO Plus Class 10 Micro SDHC with Adapter	\$8.00	4

\*Purchase links in References.

## 20. Closeout Memos

**Connor McGarty:** Like everyone else in the group, I had my hand in a little bit of everything. The tasks that I contributed the most to, however, was research, developing the Raspberry Pi script to read BLE packets, process and package them appropriately, and send them to the server, developing the database schema, writing the server code to receive RSSI information from the Raspberry Pi, processing it, calculating distance approximation to beacon positions, and using those distances to estimate position using trilateration/multilateration libraries. Additionally, I designed the path-loss experiment and handled the data afterwards, writing a MATLAB script (see Appendix A) to read in the data from text files, calculate averages and variance at each distance, and using the Curve-Fitting Toolbox to fit the data to the path-loss model. I helped Jake take what he had from the pathfinding submodule and modified it to sort and concatenate the paths to each individual item in order to determine a good estimate of the most efficient path from the user location throughout their entire list. I provided assistance to Jon with the web application.

This was the most complete and extensive engineering experience I have participated in throughout my academic career. I learned to never be too confident that your decision is going to turn out the way you think it will. So many times this semester, we ran into complications and unexpected results that slowed the team down and required more careful thought. Never expect that something will be as easy as it sounds! Just because something should work in theory does not mean that it will work perfectly in practice. For instance, the team only ordered 10 BLE beacons because we felt it would adequately cover the space. About halfway through the semester, we were displeased with the coverage of the room we were getting, causing less than ideal accuracy at some points in the test space due to lack of coverage. However, the team had approximately \$270 left in our budget, and we only needed a few more beacons that would push our accuracy into our desired

spec. It would have been a much better decision to take the safer bet and go ahead and buy a few more beacons than we thought we would have needed at the time. As for team dynamics, I am really satisfied with how the team worked together. There was a free flow of ideas and opinions, and I feel that everyone was committed to finding the best solution given the constraints, as opposed to pushing solely for what they think is best.

Given the entire two semesters, there are a few things that I would aim to do differently. As stated earlier, I would definitely have used more of our budget to give us some breathing room on components. The other parts we purchased (Raspberry Pi's, batteries, etc.) were locked in at a certain quantity because of our original goals for demonstration. However, the beacons were the limiting factor—too many would have been a waste while too few and we couldn't properly cover the space. In the future, I would identify what components fall into this category, and err on the side of caution.

I think one thing the team could have done differently would be to better split up tasks. We found it difficult to work in parallel and instead needed each others help often, working together on the same tasks because they were difficult to complete on their own, and because all of the pieces interlocked with one another so closely. Additionally, since the project was so software focused, I think most of us were not prepared for this. Not everyone could contribute to the software much on their own.

If I had to recommend future students how to best succeed in senior design, I have a few recommendations. No matter the project you choose, you will learn so much. I think it's a good idea to choose a project that firstly is interesting to you, but secondly involves an area of engineering that you wish to pursue in the future. You will gain valuable experience in the technologies and techniques in that area, and few classes give you that same hands on opportunity.

Be sure to choose trustworthy teammates. I am really so happy with our team and how we worked together, but the entire process could have been a nightmare had team members not contributed or were unfriendly or unhelpful. Be sure to do as much work as possible in the EE498 semester on research and learning what technologies you plan to utilize, as you want to get started on building your prototype as soon as possible—as I stated before, there will be many unforeseen problems. The more quickly you start and the more knowledge you have, the quicker you can find a workaround and continue on.

**Jeremy Milam:** I was involved in various hardware operations. Also, I aided in researching pathfinding algorithms to incorporate into the IoT system. Contributions were made to the beacon node layout – adjusting orientation, node density, geometrical layouts, room layout, and measurements. As for the role in the team, I took on the task of documenting experimental results.

The design project gave me a new perspective of what an engineer is. It is not a scientist, nor a technician. It is the discovery of needs that society isn't aware of and the relentless pursuit to solve the problem using through the application of physics, math, and creativity. Project management is a combination of communication, collaboration, approximating rate of production, compartmentalizing tasks, and tolerance of unpredictable circumstance.

If given the chance to repeat this process I would understand that time moves quickly, and research needs to be done rigorously in the beginning with an open mind, so that one does not get tunnel vision on the conventional approach to the problem. Also, I would push myself to take on tasks outside my comfort zone. I would improve my attention to the small details of design without losing vision of the final product.

I would advice my underclassmen to learn team dynamics, find a role within the group where their attributes can complement the work. Do lots of research early on to educate oneself on



the field. Do not reinvent the wheel per se but take what has been discovered and add to it. Science and engineering are a community—not isolated from one another. Finally, do not overlook simplicity. I believe that engineers have the tendency to honor complexity, yet engineering should be done in a perspective of effectiveness.

**Jake Watters:** Throughout this design process I tried to be involved in as many aspects as I could, though the main areas that I specifically worked with were the BLE beacons, Raspberry Pi, and the pathfinding code.

During the fall semester of 2019 a tremendous amount of research was done in the realm of localization. Once it was decided to move forward with BLE beacons as the source of the positioning I researched the various types of beacons from various vendors. I was in close contact with the sales representative from Kontakt.io and from that relationship and the specifications of their product, we chose them for our vendor. Moving into this semester I configured the beacons and adjusted them as needed. I also aided in the numerous beacon layout designs and testing in the senior design lab

I helped with the initial setup and configuration of the Raspberry Pi's. I personally own one and have firsthand experience with using them. Setting them up involved configuring them for SSH and getting the correct versions of software loaded on them. The initial setup also included the first round of testing using Bluetooth from the Raspberry Pi to a smartphone.

The third area that I worked on specifically was the pathfinding code. I performed substantial research in pathfinding. Jeremy talked to Dr. Patel and he suggested Dijkstra's algorithm. Using that information, I narrowed down the pathfinding algorithms and researched the A\* method. I had experience in Node.js, so using a Node.js package I built the base functions for

the code and was successful in getting it to perform as we needed. I did get stuck on the slicing and sorting functions that were needed to provide a concise output, but Connor was able to aid in its completion.

The main thing I learned from the project and about engineering, was to identify the need. The main concept is to accurately identify the need and to design a solution. The extra stuff, bells and whistles, can come later. Sound engineering practices based on math, physics, and hard work is what gets a design completed and satisfies the need.

If I had to do it all over again, I would have personally purchased a Raspberry Pi and a few beacons for testing. Not having the parts for the first six weeks of the semester hampered our progress for a more fleshed out design. When we did receive the parts, we aggressively pursued the testing and the implementation. Of course, it was no one's fault that this semester everyone had to deal with an unprecedented situation with the Covid-19 pandemic.

The second thing I would have done differently is that from the first day I would not underestimate the time it takes for a proper design. The due dates for deliverables and milestones approach quickly and if you do not use your time wisely it can affect the project. I would begin planning much earlier to not waste that valuable time.

My advice to the next class would be to stay driven and focused. Do not wait to the last moment to make a design decision. Not everything will work the first time so plan for setbacks. Keep an open mind to the opinions and suggestions of your teammates because that is the greatest asset to the project, your team.

The guidance and direction of Dr. Jololian, Dr. Mirbozorgi, and Dr. Yildirim were a great resource. The feedback we received helped us tremendously to have a better design.

By far the greatest resource for this project was my teammates. I had no issues discussing any anything with them. I believe everyone felt empowered to make decisions and voice their opinions. I knew that I could trust them with any part of the project. I knew I could ask them anytime for help without fallout. We made the project fun and that helped with the stress of the semester. I am very proud of the work we accomplished given the circumstances.

**Jongmoo Yim:** I primarily participated in the software aspect of the group project. Many of the software architecture and hardware interaction design decisions were suggested during the group discussion by me and Connor. I created the software flow of user's interaction with the product to the server operation interacting with each component of the pathfinding and list creation. I was in charge to design and create a web application that handles user interactions. The primary functions of the product, user location, pathfinding, and user services are the components of the server. The web app hosted by the server handles the user interaction where each user can create, read, update, and delete a shopping list. For the purpose of the project demonstration, I will also be adding quadridirectional buttons that simulate user navigating through the test space. The input of the directional buttons as user location and user shopping list of item's destinations will be the input of Jake's pathfinding algorithm. I have also participated in researching for distance calculation with trilateration method, which we later decided on multilateration method. The trilateration calculation works best when surrounded by beacons so each one of us suggested many ways user can be best estimated for their location.

I learned that engineering is not just problem solving, but within its core essence of problem solving, there are design, management, teamwork, and learning. The principles of engineering a senior project allowed myself to operate within a diverse team. Each with different background, we

freely discussed ideas professionally and managed project tasks to each on our own specialties. I took on a project without clear target, but a guideline of IoT platform, and we created a project that met the scope. In that process, we researched many problems and potential solutions, which felt like a surreal engineering process.

I personally would have liked to devote more time to the project if was given opportunity. The class is difficult to manage between fulltime student classes and a job. I would manage the project to be more divided between team members to more realistic goals in attempt to achieve the basics. The pandemic was certainly unhelpful, but I believe more could have been done if I spent my time more efficiently. I definitely would suggest the next class to choose a project that one is passionate about. Having a passion and time will allow for more personal and completed project rather than an assignment given by a professor that must be done to pass the class. If given the opportunity, design your own project within the scope and enjoy doing the work for two semesters.