J Supercomput (2018) 74:4315–4338
https://doi.org/10.1007/s11227-016-1875-8

Smart service relying on Internet of Things technology in parking systems

Ming-Fong Tsai[1] · **Ye Chin Kiong**[2] · **Ang Sinn**[1]

Published online: 19 September 2016

Abstract In this paper, a smart parking system formed by the integration of smart mobile device applications and Internet of Things (IoT) technology is proposed. The system is able to guide users through the process of searching for a car park until they successfully park the vehicle in a parking lot. The features of the system can be divided into four parts: recommendation and reservation, outdoor navigation to the car park, parking lot detection, and car park indoor navigation. An algorithm is designed to compute the recommendation priority of car parks based on adjustable, given conditions. Outdoor navigation provides a dynamic guide to the user during the trip to the car park. The iBeacon is implemented in indoor navigation to locate the position of users. The RSSI value of the iBeacon is filtered through an algorithm to provide an accurate result. The performance of the iBeacon in a different set-up condition is also evaluated. A field test is performed to acquire data and investigate the operational feasibility of the proposed system.

Keywords Internet of Things · Parking system

✉ Ming-Fong Tsai
mingfongtsai@gmail.com

Ye Chin Kiong
yechinkiong@gmail.com

Ang Sinn
angsinn.as@gmail.com

[1] Department of Information Engineering and Computer Science, Feng Chia University, Taichung, Taiwan

[2] Industrial Ph.D. Program of Internet of Things, Feng Chia University, Taichung, Taiwan

1 Introduction

Road traffic in urban areas is always considered a topic to be discussed. With the sharp increase in the number of vehicles, it is difficult to find a parking lot due to insufficient parking resources. Drivers confront parking-related problems especially at rush hour. The first step is searching for vacant parking lots. Drivers usually have to drive around searching for a parking lot. With luck, someone might be leaving and then a driver is able to fit into the recently vacated space. However, this method depends on luck and also wastes fuel and time. It is known that drivers and motorists who are searching for parking space partially contribute to road traffic. There has been a variety of recent research [1,2] related to on-street parking statistics, which are capable of managing and improving the traffic flow caused by the search for parking lot. Second, the drivers might not know where the nearest car parks are or where to legally park on the street in particular areas. This may cause the drivers to miss a nearby area or car parks with vacant parking spaces. Thus, an intelligent system that can manage parking problems is important to avoid unnecessary waste.

Research aimed at solving parking problems has been carried out over recent years. The development of a smart parking system is achievable with the help of Internet of Things (IoT) technology. IoT is a popular technology that claims that almost everything can be connected to the Internet. This technology has grown more mature with the emergence of LPWAN technologies such as LoRa. There is much research on smart parking system based on cloud and IoT technology [3–5]. The server communicates with node devices through the cloud to manage the car parks. A common approach to resolve parking problems is a parking lot reservation [6,7], which utilises real-time allocation algorithm. However, there is also research on auction-based [8] parking reservation systems, which analyses a user's parking behaviour influenced by parking rates. Notice-based guiding infrastructure [9], which implements IoT, is also considered by researchers.

To solve parking problems, a complete smart parking system that integrates IoT technology and a smart device, is proposed in this paper. The proposed system provides a few convenient features to users: the display and recommendation (with configurable recommendation conditions) of a nearby car park, car park reservation, outdoor navigation and indoor car park navigation. To achieve a more user-oriented recommendation, the proposed system adapts an original algorithm that enables users to configure the recommendation conditions and computes the recommendation priority based on the configuration. In outdoor navigation, the system uses Google APIs to assist outdoor navigation, and also provides a dynamic recommendation if a better parking lot is found. For indoor navigation, iBeacon Bluetooth transmitter is used to aid in indoor navigation. Indoor navigation is considered somewhat difficult because one of the problems faced in implementation of indoor navigation is the inaccuracy caused by abnormal signals.

This paper proposes an algorithm that stabilises the navigation process through elimination of abnormal Received Signal Strength Indicator (RSSI) values. The proposed system will guide users step-by-step from searching for a car park until their vehicles are successfully parked, with the aim of consuming less time and fuel compared to other existing systems. In addition, experiments with iBeacon are carried

out to determine the best settings that can be integrated with this paper's proposed algorithm when dealing with indoor navigation.

2 Related works

An intelligent smart parking system is almost unachievable without incorporating IoT technologies. However, there has been related work that comprehensively utilises the advantages of IoT technologies to form a smart parking system. In [10], Radio Frequency Identification (RFID), a Wireless Sensor Network and NFC are used. The system is able to direct users to the nearest vacant parking space through an application on smart devices, and also to pay for the occupied parking space. Another version of a software application was developed for traffic police to take action on unauthorised parking, such as parking longer than the purchased parking time. However, the system only shows an approximate route to the vacant parking space using an outdoor map, which means the system is unavailable for an indoor car park because it has no detailed navigation that is useful in a complex indoor structure.

One of the goals of developing a smart parking system is to make the car parking process automated. In [11], a miniature model of an automated parking system, which aims at minimising the dependence on human resources, is demonstrated. The technical key for this system is the communication between vehicles and a parking control unit via microcontrollers. The entry and exit of the vehicle is commanded by an Android application, and the path of the vacant parking area can be tracked by the car automatically.

There is research that proposes an algorithm for parking planning [12]. The authors aim to provide a parking guide in real time by transforming problems of parking planning on-line into off-line problems, and to build a mathematical model based on the transformed problem. In addition to resolving parking queries, studies are made based on parking reservations [6,7], which focus on ensuring parking space availability to users. In these studies, parking reservations are seen as the key feature of a smart parking system.

The systems learn real-time parking lot status from sensor networks deployed in car parks, and assign a suitable parking slot (usually the nearest parking slot) to the user. In [6], the state of a parking space is indicated by a coloured light system installed in car parks. The system assures a parking space reservation by indicating unauthorised parking with a red light, and expects the drivers violating car park system rules to be punished. However, the user is not capable of choosing which car park to reserve because the reservation spot is selected through the system allocation. There is no guarantee that a parking space will be successfully allocated after a reservation request is sent, and thus, the user might be required to wait until a space is available for allocation rather than selecting an alternative car park manually. In [7], a reservation authority is placed in each parking lot to manage their real-time situation. The parking lot information in the database is dynamically updated to provide users with the real-time parking lot information. However, in this proposed system, the identity of drivers who occupy a reserved parking space cannot be identified automatically. The user is required to visit the service provider's website through the Internet or Bluetooth

connection to verify the car owner's identity, which is considered inconvenient to users.

In [13], the authors proposed a smart parking system utilising the ZigBee wireless network protocol as the main technology for the communication between the parking monitoring system and microcontrollers in the car park. The system is inexpensive and adapts to a lower energy consumption compared to other systems. On the other hand, it can be implemented in an indoor car park including a car park with multiple floors. Yet, the parking inquiry module has to be installed on the vehicles; there have been a number of systems [6,7,11] that can be accessed through mobile devices with ease, but this limitation makes it a difficult approach for users. Moreover, ZigBee protocol shows a relatively weak performance when it comes to obstacle penetration compared to a Wi-Fi module, which is used in this paper, and also inexpensive to implement.

In [14], del Postigo estimates parking area status through background subtraction and transience map analysis. The method is capable of handling scenarios with multiple moving objects, different illumination conditions and occlusions between vehicles with a static surveillance camera. The authors argue that the use of individual sensors to monitor each parking slot is expensive and non-scalable. However, areas covered by a surveillance camera are limited. The installation, power consumption and long-term maintenance fees of a camera are much greater compared to a scenario where a low-cost, low-power consuming microcontroller sensor is used. In addition, a camera requires frequent maintenance to assure its function, for example, for dirt that interferes with the image. Moreover, although the method has been tested under a various brightness and climate conditions such as morning, evening and in rain, conditions such as heavy rain or fog which could affect the performance is left blank.

Article [15] proposed a mobile-based indoor positioning system. The system implements iBeacon technology to achieve indoor positioning. RSSI values are used to determine the location. Although the RSSI values are filtered through an equation to acquire the average value, the positioning is invalid when abnormal signals interfere with the smart device. In this paper, an improved algorithm is proposed to eliminate abnormal RSSI values before evaluating the average RSSI value of an iBeacon.

3 Proposed work

3.1 System overview

The proposed system in this paper integrates with IoT technology, providing various types of services. The proposed system can be implemented in any car park with a clear entrance and exit. The physical construction of the system includes a server, Arduino controller, RFID reader, iBeacon and ESP8266 module. Except for iBeacon, each of them contains a static IP address for their connection to the Internet. The use of Arduino controls the gate motion at the car park entrance. The RFID reader identifies users, which is important in implementing the reservation service. The RFID reader

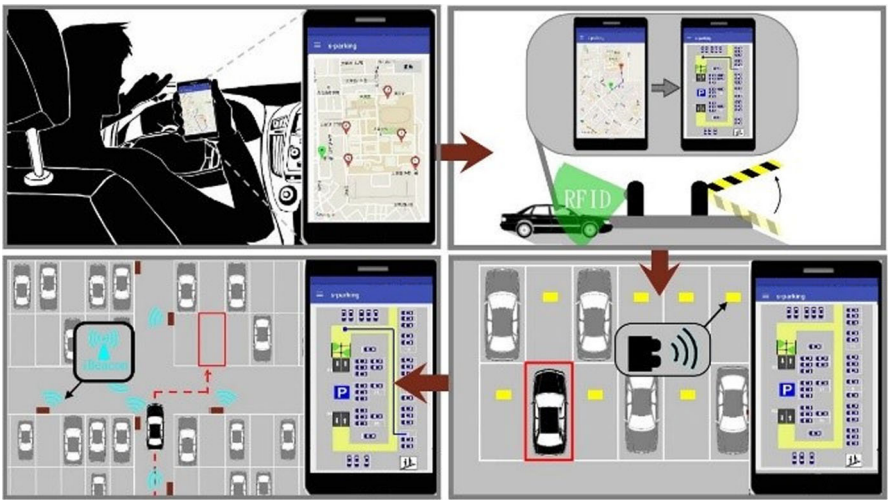


Fig. 1 System overview

also transmits valuable messages to the server to be interpreted. iBeacon is used in the implementation of the indoor navigation system, which does not usually have a good Wi-Fi signal. The use of the ESP8266 module enables low-cost parking space detection.

The entire functioning of the proposed system involves a mobile smart device, as shown in Fig. 1. The upper left section shows a recommendation for a car park—a service provided by the proposed system to notify users on available car parks. The next section indicates the cutover between outdoor GPS navigation and car park indoor navigation. The RFID reader plays a main role in the cutover process. The iBeacon signal is an alternative way of identifying the position of the user. The lower right section represents the parking slot detection, done through incorporating the ESP8266 module and server database. The last section indicates the implementation of the indoor navigation, where the iBeacon signal strength is used to assist in car park indoor navigation and detect the location of the users.

Figure 2 shows the overall system flowchart. First of all, the user is required to log in the system through a smart device application. The important information in user's account includes the tag id, which is recognised by the RFID reader. Next, the proper car parks are recommended to users, based on the user's preference. When a car park is chosen, a reservation request is sent to the central server. The server will interpret the message, and respond with information on the reserved car park if the reservation request is successful. At this stage, the system will navigate the user to the destination car park. When the tag of a user is read by RFID readers, the server will be informed of the arrival of the user and will switch into the indoor navigation mode. The smart device will receive an iBeacon signal during the indoor navigation process. When the user arrives at a parking slot, the system's task is complete. The proposed system will be described in detail in the following sections.

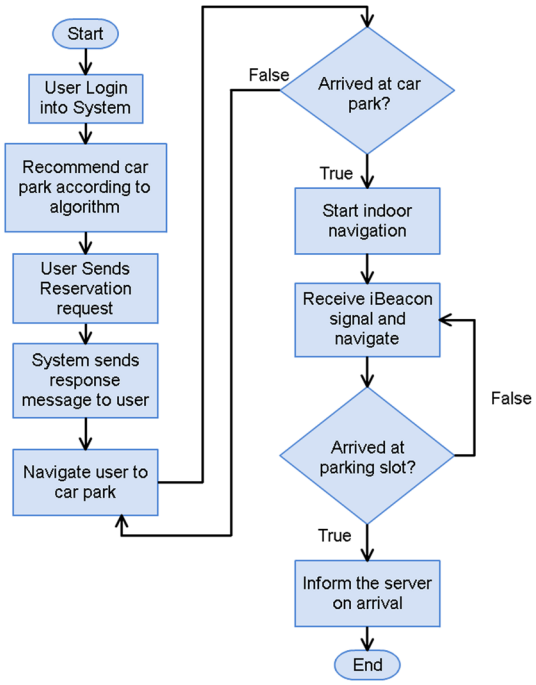


Fig. 2 System flowchart

3.2 Car park recommendation and reservation

This section will discuss the recommendation and reservation of a parking lot. The purpose of a car park recommendation is to provide information of available car parks near the user or near the destination. Meanwhile, the recommendation priority according to the user's preference will be shown to help in the selection of the car park. To make the provided information more useful, a reservation system is designed to prevent the user from getting into a full car park upon arrival. Google's API, GPS and an algorithm are used in the application of these services (Table 1).

$$F_{s\to e} = \alpha \times \frac{d_{s\to e}}{d_{\max}} + \beta \times \frac{f_e}{f_{\max}} + \gamma \times \frac{m_e}{m_{\max}} \tag{1}$$

To compute a recommendation based on a combination of different conditions, a new algorithm should be designed. Equation (1) is the algorithm used to compute the value that represents the recommendation priority. The algorithm is designed to handle calculations regardless of how the given conditions are adjusted or combined. This algorithm can be extended if more factors are taken into consideration. In this proposed system, these factors are distance, parking rate, and available parking slots. 'Distance' here refers to the distance between the user and the car park, with road traffic condition taken into account. The parking rate is also one of the conditions since price is a factor that affects a user's decision. These predefined conditions can be combined randomly and will influence the recommendation priority. To determine

Table 1 Representation of symbols in Eq. (1)

Symbol	Representation
E	Car park
$d_{s \to e}$	Distance between users and the car park
$d_{\max}$	Distant of the furthest car park
f_e	Number of the remaining parking slots
$f_{\max}$	Highest number of total parking slots
m_e	Parking rate
$m_{\max}$	Highest parking rate among car parks
α	Distance weight
β	Vacant slot weight
γ	Parking rate weight

which car parks should go through Eq. (1), the radius (km) of the distance area is configured. Depending on the configuration, the centre point of the radius might be the user's location, or the destination spot. The system will search for all car parks available in the covered distance and exclude those that have no available parking space.

Here, we will explain the implementation of Eq. (1). First, we run through the distance, parking rates and total parking space of all listed car parks to extract the values of α, β and γ. The next step is to set up the value of α, β and γ. Assume that all three conditions are selected, and α, β and γ will be set as 0.4, 0.3 and 0.3, respectively. In this case, α has a greater value since the distance values are unlikely to be repeated compared to parking rates and available parking slots. This property can prevent (1) from producing a similar value for the car parks, which would reduce the preciseness of the recommendation priority. Assume that a user takes the parking rate into consideration; then α, β and γ will be set as 0.05, 0.05 and 0.9, respectively. The unchecked selection will not be set as 0 to avoid production of similar values among car parks. When the configuration of the base values is complete, the related information of all recommended car parks will be fit into (1), respectively. The computed values will then be sorted in ascending order. The lower computed value indicates a higher recommendation priority. The recommended car parks will be represented with a location tag and priority number. Further information will be provided to the users when the tag is tapped. This information includes the car park name, address, total parking slots, available parking slots and parking rate.

Although car park locations are shown to the user in the form of a recommendation, it cannot assure that there is available space upon the user's arrival. For the reason above, a reservation service is introduced. The reservation will reserve a parking slot for the user. When the server receives a reservation request, it will attempt to reconfirm that the available slots at the desired car park is more than 0. When one available slot is reserved, it will decrease the available slots by 1, prohibiting other potential clients from entering. The user who reserved the car park can enter it when the RFID reader installed at car park entrance scans the user's tag. The user's tag is associated with the user account upon registration. When an RFID reader reads a tag, a message is

transmitted to the server. The server will then interpret the message and if the user's reservation record is found, an instruction will be sent to the Arduino gate controller to open the car park gate. Note that if the number of available slots becomes 0, other users will not be able to gain access to reserve the car park.

To avoid a situation where users abandon the reserved car park, the system implements a mechanism to filter the unused reservation record. Data (for example, the road conditions) provided by Google's API will be used to analyse the time required for the user to arrive at the reserved car park. From the moment the reservation is made, a countdown starts. If the user takes 10 minutes more than the expected time, the reservation will be cancelled and the reserved slot will be released. Since the traffic conditions are taken as a consideration when Google's API calculates the estimated arrival time, unless the user intentionally drives in the opposite direction, the probability of it taking longer than the stimulated time is relatively low. If the user really cannot make it in time, the system will cancel the reservation to protect the interest and profitability of the parking lot owner.

Figure 3 shows the flowchart of the car park recommendation and reservation. Before starting the reservation system, the system will determine whether the user has a reservation record. If there is no record, it will proceed to make a reservation.

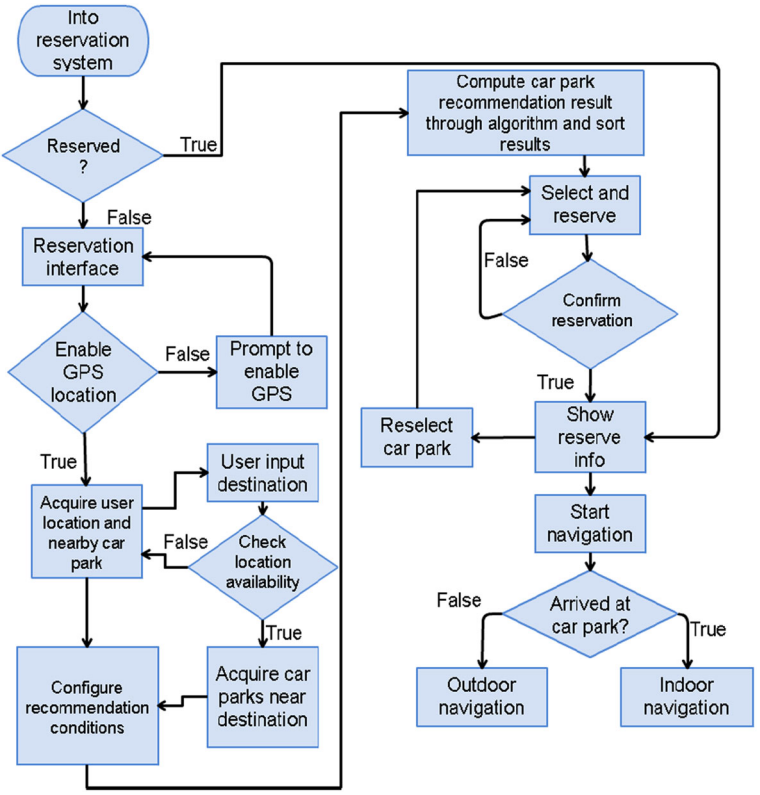


Fig. 3 Car park recommendation and reservation flowchart

If the GPS location service is disabled, the system will prompt the user to enable it. This is important because the location of the user is a critical reference for the recommendation server. If the GPS is enabled, the user's location can be obtained and the nearby car parks will be located. During this stage, if the user inputs his or her destination, the system will validate the input. If it is valid, the centre point for acquiring nearby car parks will be set as the new destination point, instead of user's current location.

The feature of recommending car parks close to the destination is useful with reference to traffic searching for parking experimental result in [7], which shows that users tend to reserve the parking lot nearest to their destination.

The next step is the configuration of recommendation conditions. The default setting for conditions is distance. The configuration step can be skipped. The system will compute the recommendation result through (1) according to the selected conditions. The result will be sorted according to the set priority and shown to user. Then, the user will be able to select a car park from these recommendations. After the reservation is confirmed, the reservation information will be shown. The user can either reselect a car park or start navigation. If the user arrives at the destination car park, the indoor navigation will be started, or it will continue its outdoor navigation.

3.3 Outdoor GPS navigation

An outdoor GPS navigation is included in the proposed system to serve as a guide and for easy reference upon driving. It also minimises the time and fuel needed to get to the car park. The system also provides a dynamic real-time car park recommendation in the navigation process. The proposed system adopts Google API whose users are familiar with its map interface. The utilisation of Google API simplifies the complex procedure of drawing maps and calculating location points. In the outdoor navigation mode, it is necessary to enable the GPS receiver so that the system can locate the user's position. The system will inquire the location point of the reserved car park and displays it on the map.

The next step is to calculate the best route. Road traffic and the distance between the user and the destination car park are considered to determine the best route of navigation. The estimated time for the user to arrive at the destination car park is also calculated based on the suggested route as well as traffic conditions. This approach can prevent the user from being stuck in traffic. The location of corners in the road along the desired route is acquired through Google's API, and then are joined together to form the navigation route. The drawn route is shown on the map, and then the navigation process can begin.

In the navigation process, the system will constantly acquire the user's position to determine whether the user has moved from the last position. If the user moves, the map will be renewed. The above procedure is repeated to form a dynamic navigation process. The system will stop acquiring the user's position when the user's tag is scanned by the RFID reader of a car park. When a user's tag is scanned by the RFID reader at entrance of car park, the server will receive the signal and change the Boolean value, which indicates whether the user is in car park.

Figure 4 shows the flowchart of the outdoor navigation. When the outdoor navigation starts, if the GPS locating service is disabled, the system will prompt the user to enable it. The GPS service is crucial because the navigation process will not function without the assistance of the GPS information. If the GPS is enabled, the location of the user and reserved car park will be shown. Next, the navigation route will be drawn by acquiring location of corners between the user and the destination. Here, the system will read information of other car parks and interpret whether a better car park is available. If so, a new navigation route will be drawn, indicated by a different colour. The user can manually select whether to change the destination car park or not. The system will then compare user's current location with previous location to determine whether the user has moved. If the user moves, it will update the user's location to show a dynamic navigation route. When the RFID reader reads the user's tag or when an iBeacon signal is received by the user's mobile device, it will switch to the indoor navigation. If the user has not arrived at the car park, the location of the user and the reserved car park will be notified, and the navigation route will be redrawn. This procedure will be repeated until the user arrives at car park.

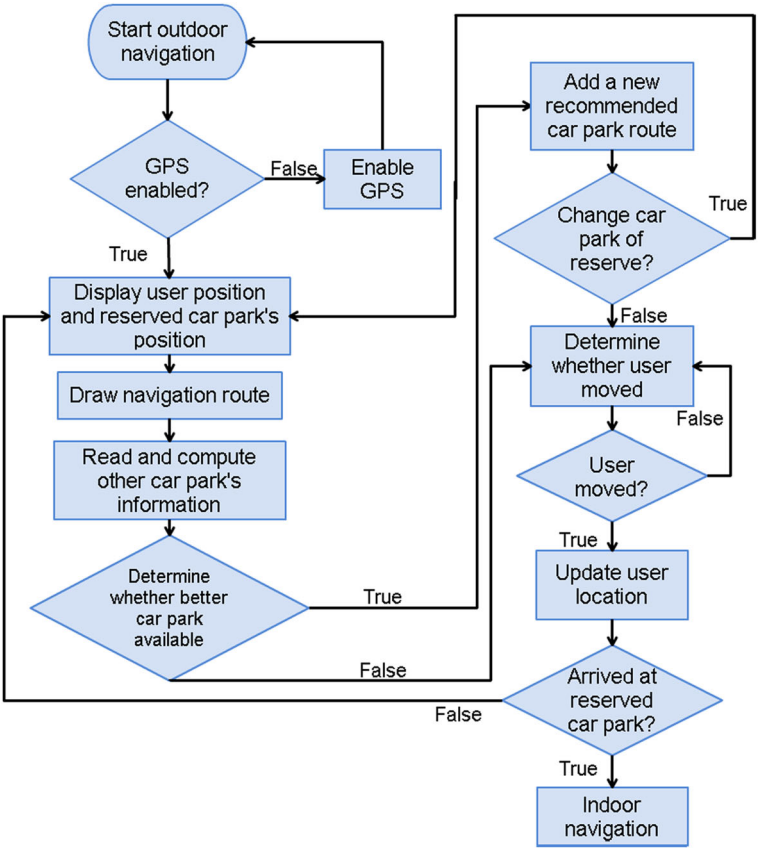


Fig. 4 Outdoor navigation flowchart

3.4 Parking slot detection

This section explains the methods of detecting parking slot status at the car park. This information is important because the server can manage the status of parking lot in real time. The parking lot information is used throughout different stages of this proposed system. In this paper, the method of parking lot detection focuses on utilising a module that is suitable for large-scale implementation and acceptably low development costs. In this section, ESP8266 module plays an important role. ESP8266 is a mini microcontroller with a Wi-Fi module, capable of integrating with various types of sensors due to its powerful computational ability and internal storage capability. ESP8266 is not only small in size, but is also inexpensive, does not consume much energy, and most importantly, it comes with a Wi-Fi function, which makes it capable of communicating with the server for transmission of parking lot unit information. The ESP8266 module is chosen because of the low cost for each single unit.

In the case of this proposed system, ESP8266 is integrated with an ultrasound sensor to detect whether or not the parking lot is occupied. Although there are different methods of vehicle sensing, an ultrasound sensor is chosen due to the cost consideration. In each parking lot, an ESP8266 module is installed. When the ultrasound signal transmitted by the ultrasound sensor is blocked by a vehicle, a message is transmitted through the ESP8266 module to the server to be interpreted and stored. Through this process, the status of parking lots is always updated. Based on the parking lot information stored in the database, all occupied parking lots will be marked on the car park indoor map.

3.5 Indoor navigation

In this proposed system, 'indoor navigation' means the navigation from the car park entrance to a parking space. The core purpose of this service is to guide users to an available parking space to park their vehicles, which is useful when a car park does not have a wide and clear view. The platform for this navigation is the smart device application. An iBeacon Bluetooth transmitter, which transmits signals constantly, facilitates the navigation process, and is distributed at the car park at every corner. The main function in this system is to assist in locating the user's position and to locate corners in the navigation route.

The operation flowchart of indoor navigation is shown in Fig. 5. When the tag of a user is scanned by the RFID reader located at the entrance of the car park, the indoor navigation will start. The system will read and display the information of the car park, such as the name and location from the server database. Then, the user's device will start to detect and receive iBeacon signals, which are installed in the car park. As long as iBeacon signals are successfully detected, the system is able to determine the user's position according to the iBeacon's RSSI value since the source of the strongest signal received is the iBeacon nearest to the user. Once the user's location is determined, a vacant parking slot nearest to the user will be assigned as the destination of the indoor navigation. When both starting and ending locations are determined, the navigation route will be generated. The system will constantly check whether the user arrives

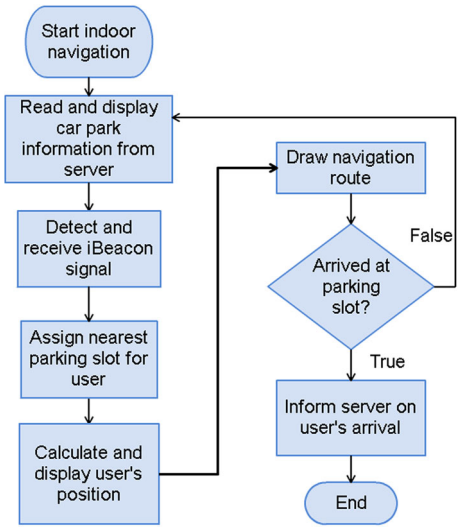


Fig. 5 Indoor navigation flowchart

at the parking slot and repeat the navigation process until the user successfully parks his or her vehicle. It will then inform the server upon the user's arrival and stop the process.

We will now explain the method of computing the indoor navigation route. Since every iBeacon is placed at a fixed spot, it is possible that when the destination parking slot is confirmed, the system will list every iBeacon that the user will pass through to arrive at destination parking slot. Location points of the listed beacons will be marked on the indoor map, and then connected to form a route. The smart device will constantly receive iBeacon signals at the car park, which are also used to determine the position of user. Therefore, a change in the user's position can be detected on time and will trigger the system to reassign the destination parking slot and redraw the whole navigation route.

The RSSI value is the key to determine the user's position. The strength of the signal received is indicated by a value (usually negative values). The strongest RSSI value is the value closest to 0. With reference to Fig. 6a, b, note that the RSSI that is nearest to the user has the greatest value. When the user moves, a change in the RSSI value can be observed. The (2) stands for the raw iBeacon RSSI data set received. If (2) is directly used for calculation, there is a high probability that the result will be affected by abnormal signals. Assume that the user stays at a fixed location. The system successfully locates the user's position. After a few seconds, an abnormal signal is received and causes the average value to differ from actual value. Thus, the location of the user is shown on the map moves from its actual position. Therefore, Eq. (3) is configured:

$$\mathrm{rssi}_{B_i}^{t} \tag{2}$$

$$\mathrm{RSSI}_{B_i}^{j} = \frac{1}{j}\sum_{k=0}^{j-1} \mathrm{rssi}_{B_i}^{k} \tag{3}$$

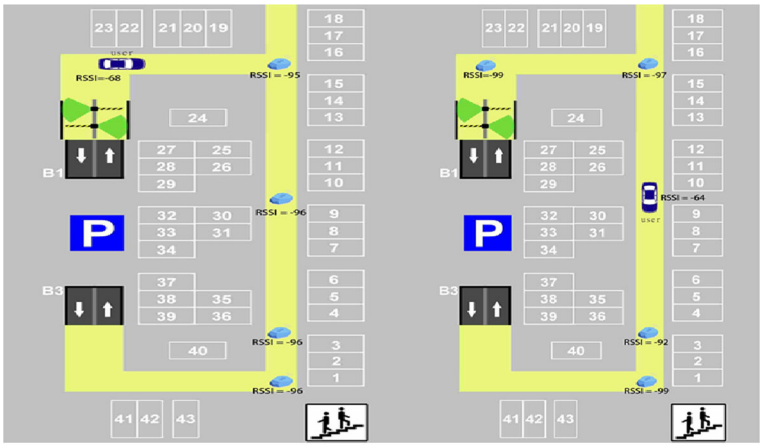


Fig. 6 RSSI values

The main purpose of (3) is to stabilise the received iBeacon signal by eliminating abnormal RSSI values. Ensuring the eligibility of a received RSSI value is important in order to ensure that user's position is correct for the indoor navigation mode, otherwise it might lead to an abnormal navigation route. In the application of Eq. (3), 10 RSSI values of the same beacon are fetched. Then, with the equation, each RSSI value is compared with its previous and next value. If the difference between the target RSSI and previous or next RSSI is greater than 20, the current RSSI value is seen as an abnormal value and thus, is eliminated.

When all abnormal RSSI values are eliminated, the mean of the remaining RSSI values will be calculated. For example, when the system receives an iBeacon's 10 RSSI values, 5 of them are -66, -64, -61, -32 and -70. The fourth value, which is compared to the third and the fifth data, has a difference greater than 20, and thus it will be eliminated from the mean calculation. Assume that the other 5 unmentioned numbers are normal, then in the calculation of the mean value, only 9 RSSI values are involved. The mean value of the RSSI values is used to compute a more accurate position of the user. Every beacon's RSSI values will go through the process mentioned above and be sorted to find the smallest value indicating the user's position.

There are a few situations that will lead to the reallocation of destination parking slot, such as when the destination slot is taken by others, or when the user passes through the destination slot. The system will recommend a new parking slot whenever this happens until the user successfully parks the car or leave the car park.

4 Result

This section will demonstrate the proposed system and the experimental results of the iBeacon. All of the provided features are successfully integrated into a smooth system.

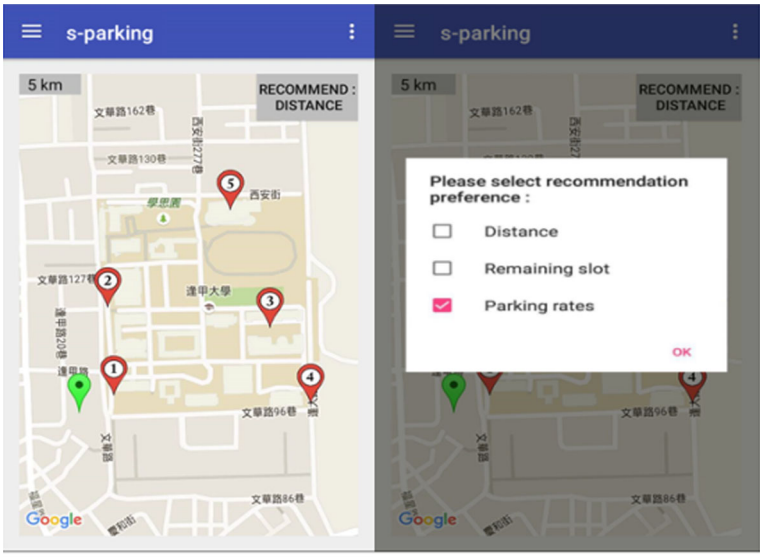
The experiments on the iBeacon are carried out to evaluate the performance of the iBeacon in different situations.

4.1 Setup

The proposed system runs on an Android platform, and is presented as a smart device application. The Android application is developed in Java using Android Studio, the official integrated development environment (IDE) of the Android platform. ESP8266, which is used for the parking slot detection, is programmed in Lua. The programme that controls the RFID reader is developed in C#, using Visual Studio IDE. The server is located in a Wireless Sensor Network laboratory of a university campus. The server is built by using XAMPP, an open source cross-platform web server solution stack package. MySQL database that comes with XAMPP, is used in the system. Five car parks on the university campus are added to the database and used as the experimental site.

4.2 Car park recommendation

Figure 7 shows the implementation of the car park recommendation feature. In the default setting, the available car parks within 5 km is shown and recommended. All five car parks included in the server database are shown since they are in the range. With reference to Fig. 7, the green tag indicates the user's current position, which is located through GPS service. The red tags indicate the recommended car parks. In



(a) (b)

Fig. 7 Recommendation of car park

Fig. 7a, the recommendation condition used is distance, which is the pre-set condition. At this moment, in the recommendation algorithm, value of α, β and γ are set as 0.9, 0.05 and 0.05, respectively. The nearer the car park is to the user, the higher the recommendation priority will be. Figure 7b shows the window interface where an alteration of the recommendation condition occurs, which is accessible by clicking the button on the upper right corner of the map. The three conditions provided can be combined. In Fig. 7b, the condition is changed to the parking rates. Clicking on the confirm button will finish the configuration. Due to the change performed in Fig. 7b, the recommendation priority of a car park is different from that shown in Fig. 7a. The lower the parking rates, the higher the priority.

4.3 Outdoor navigation

Google APIs and GPS are used in the outdoor navigation. When the user has a reservation history, and allows GPS positioning service, the outdoor navigation mode can be started. The green tag marks the user's position whereas the red tag marks the position of the destination car park. The blue line indicates the navigation route. The route drawn on the map is the shortest and smoothest route, which is calculated based on the distance and traffic situation. Figure 8a, b show that when the user's position changes, the location of the user on the map will be renewed accordingly, and the navigation route will be redrawn as well, forming a dynamic navigation. The navigation process will continue until the user's tag is scanned by the RFID reader located at entrance of a car park, which triggers the indoor navigation mode.

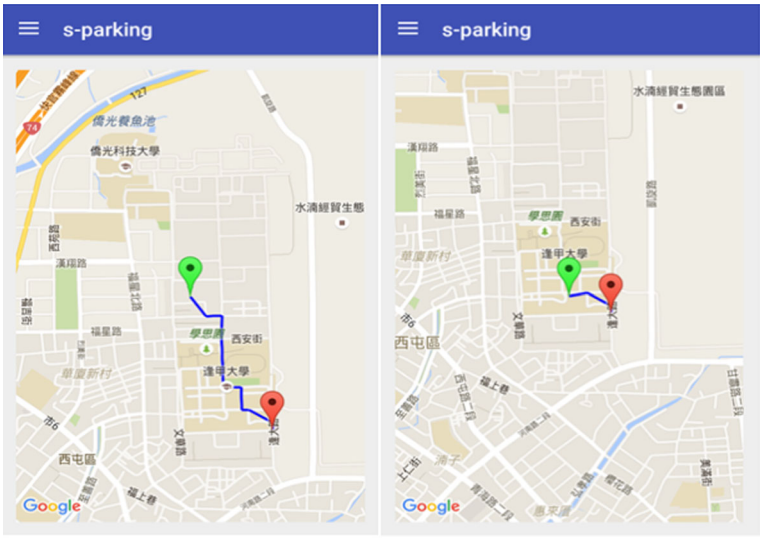
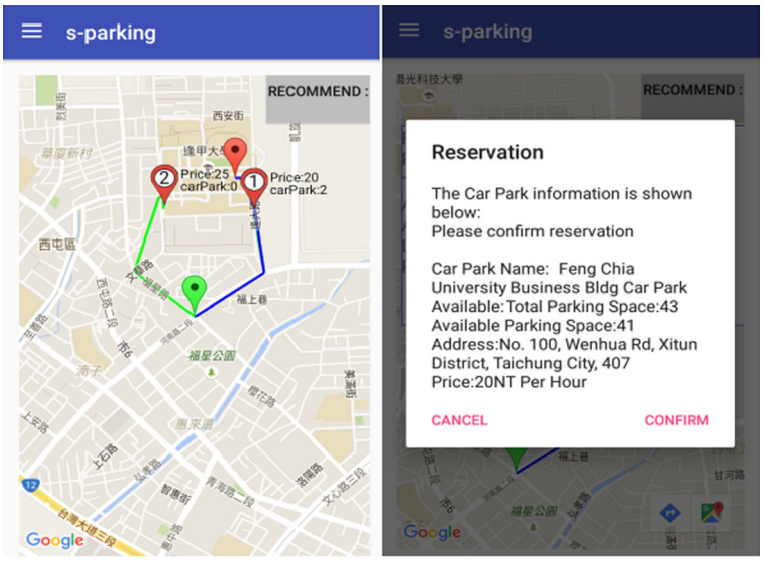


Fig. 8 Outdoor navigation to reserved car park



(a) **(b)**

Fig. 9 Dynamic recommendation and re-reservation of a car park

In the field test, the experimental results encouragingly show that real-time dynamic recommendation during the navigation process has been successfully implemented. With reference to Fig. 9a, the colour-filled red tag is the destination car park, whereas the other two red tags are the newly recommended car park. All of the recommendation conditions are checked. On the way to the previously reserved car park, the system figures out that there are two car parks that suit the checked conditions better than the current car park. The location tag of the newly recommended car park is shown and the potential navigation route is drawn in another colour. In this case, it will be green. Note that the route to tag '1' is overlapped with the main navigation route. Therefore, it will not be complicated or too late for users to alter their destination car park. They can simply click on the new tag and make a new reservation on time, as shown in Fig. 9b.

4.4 Indoor navigation

RFID readers, iBeacon and ESP8266s are used in this section. In the field test, ultrasound sensors are attached onto ESP8266s and placed on every parking slot. Five iBeacons are placed on the turning corners along the car park route. Two RFID readers are installed at the entrance of the car park, and they sense the entry and exit of vehicles, respectively. When the vehicle drives into the car park and enters the sensing scope of the RFID reader, the tag attached to it will be read by the RFID reader. The information read from the tag is sent to the server. After this verification, an instruction is sent to the gate, and the gate is controlled to open through the Arduino controller.

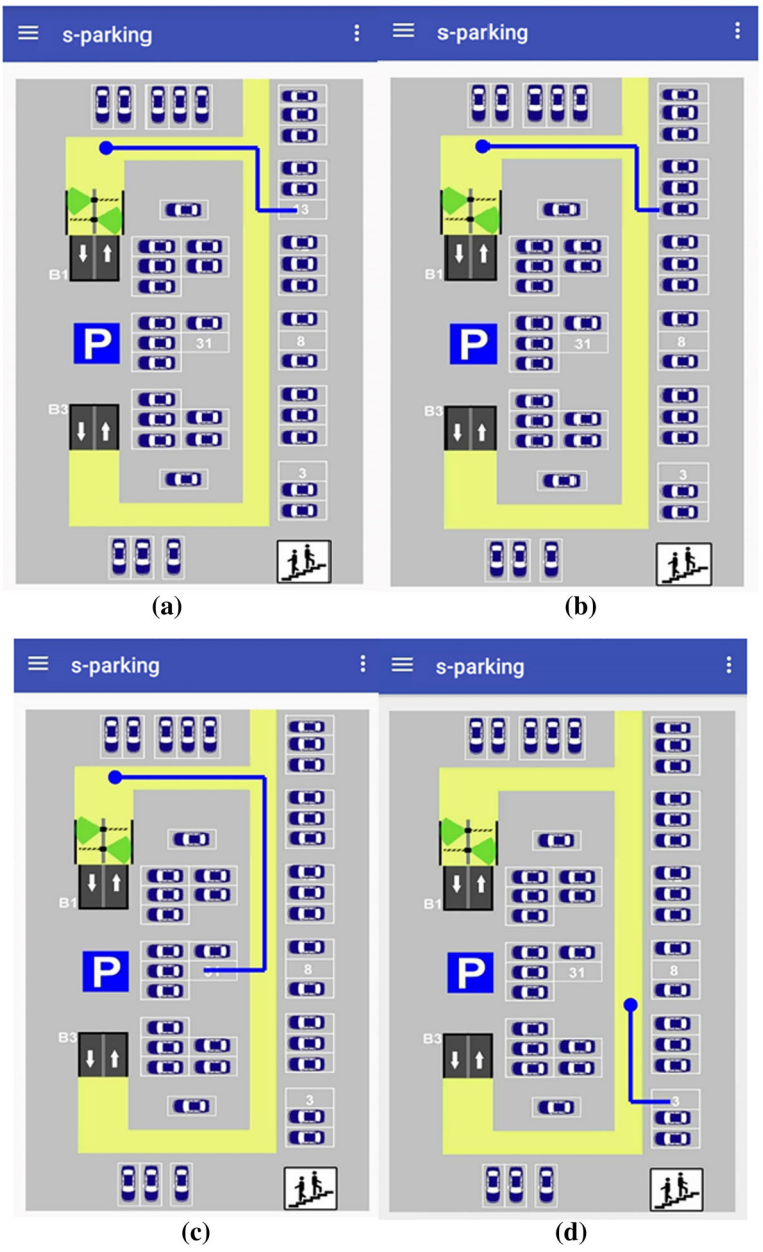


Fig. 10 Indoor navigation

When the user's device starts to receive iBeacon signals, the navigation formally starts. As shown in Fig. 10a, the location of the user is determined, which is represented by a blue circle attached to the navigation route. The system computes the location of the nearest vacant parking slot, and the navigation route leads the user to the nearest

parking slot. Figure 10b shows that the destination parking slot is occupied by another vehicle. In this situation, the system will find a new, nearby vacant slot for the user, and a new navigation route will be calculated. As shown in Fig. 10c, a new parking slot is assigned. In Fig. 10d, the user drives through the assigned slot. In this situation, another parking slot will be assigned. This means that the user does not need to follow the navigation strictly. As long as there are sufficient vacant slots in the car park, user can choose where to park. When users successfully park their vehicles, and the indoor navigation mode comes to an end.

4.5 iBeacon performance evaluation

An experiment was carried out to evaluate the most suitable distance between iBeacons to produce a better performance in indoor navigation. It is ideal for the smart device to receive and interpret iBeacon signals before actually passing through the iBeacon in front of the vehicle; in real navigation, the user is moving and a reaction in advance is required to avoid latency. In a typical situation, it is assumed that the nearest iBeacon's RSSI value is the greatest. However, in the case of a moving vehicle, the movement and the metallic materials of the vehicle interfere with the iBeacon signals. Assume that the vehicle is between two iBeacons; the iBeacon in the front actually receives less interference compared to iBeacon in the rear, which is blocked by the rear of the vehicle and driver's seat. Thus, the iBeacon in the front yields the greatest RSSI values.

Two android applications are specially developed to receive iBeacon signals and evaluate the average RSSI values, as described in Sect. 3.5. In the following experiment, 10 RSSI values are taken as a set to calculate the average value. The applications will calculate and show the information of the iBeacon with the greatest RSSI values every second. One application evaluates the RSSI without eliminating abnormal signals, whereas the other filters the values through (3). The iBeacon's transmission power is set to 0 dBm, which yields a coverage area within the radius of 30 m. The iBeacon is also configured to broadcast a signal every 100 ms. Although an iBeacon broadcasts 10 times a second, in an actual case, the smart device is only capable of receiving approximately five to eight signals from each iBeacon. The experiment is carried out on the B1 floor of the car park on the university campus. Different numbers of iBeacons are evenly distributed along a straight lane of 45 metres. The number of iBeacons used are 10, 8, 6, and 4.

To show how the evaluation process is carried out, we will now explain the whole experimental process. First, 10 iBeacons are evenly distributed in a straight line of 45 m. Two people get into a car, and the driver drives at the speed of approximately 5 km/h. One person is responsible for holding two smart devices with different experimental Android application installed. One device processes the RSSI values without filtering the values while the other device filters the RSSI values through Eq. (3). When the car passes through the lane with the iBeacon distributed every second, the iBeacon with greatest RSSI value will be shown. The other person follows the car and records the actual time when the car arrives at the iBeacon's locations. The same process is repeated using 8, 6, and 4 iBeacons, respectively. The result is shown in Fig. 11 and Table 2.

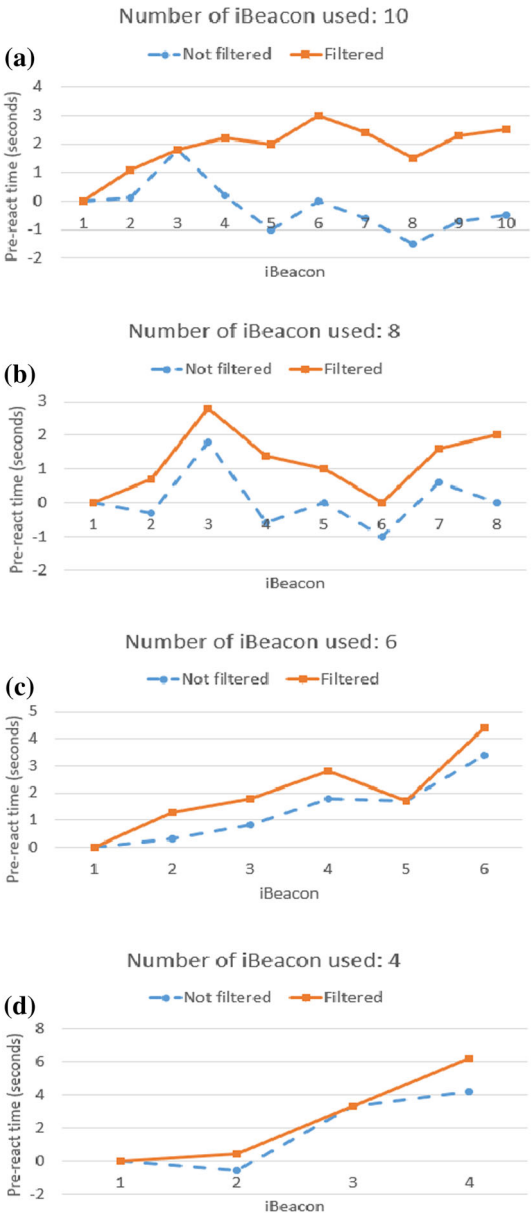


Fig. 11 Experimental results using *10, 8, 6,* and *4* iBeacons, respectively, with a speed of approximately 5 km/h

The X-axis implies the iBeacons corresponding to the actual order distributed on the straight line, while the Y-axis shows the pre-react time of the smart device. The value is acquired by subtracting the actual time of arriving at an iBeacon from the time the application determines the arrival at the corresponding iBeacon. It is shown that no matter how many iBeacons are used, the pre-react time of the smart device, when interpreting the incoming iBeacon, is faster when the RSSI values are filtered. In

Table 2 iBeacon distance evaluation, average pre-react time, under a speed of approximately 5 km/h

Number of iBeacons	Distance between iBeacons (m)	Average pre-react time (s)	Average pre-react time when filtered (s)
10	5	−0.22	1.88
8	6.4	0.06	1.19
6	9	1.33	2.00
4	15	1.73	2.48

Fig. 11a, if the abnormal signals are not filtered (with reference to the algorithm in [15]), the difference between the time when the device interprets the arrival at a particular iBeacon and the actual arrival time is 0, 0.1, 1.8, 0.2, −1, 0, −0.6, −1.5, −0.7, −0.5, respectively, resulting in a total pre-react time of -2.2 s, thus indicating a delay in responding. However, if the RSSI values are filtered, the differences are 0, 1.1, 1.8, 2.2, 2, 3, 2.4, 1.5, 2.3, and 2.5, respectively, yielding a total pre-react time of 18.8 s. Obviously, the smart device reacts faster if RSSI values are filtered. This proves that Eq. (3) is effective in enabling a smoother navigation process with less latency.

Also, Fig. 11 shows that the fewer iBeacons used, the greater the pre-react time will be. Although the filtered data performs better than unfiltered data, the performance of the unfiltered dataset improves when the number of iBeacons used decreases. In addition to the prolonged distance between iBeacons, this phenomenon is also caused by lower signal interference in that particular area. The result is summarised in Table 2. The average pre-react time is calculated by dividing the sum of the corresponding data's pre-react time by the number of iBeacons used. The average time increases when the number of iBeacons decreases. Through the experiment, it is assumed that the best distance between iBeacons is approximately between 9 and 15 m since their average pre-react time is more than 2 s.

It is shown that regardless of how many iBeacon are used, the pre-reaction of the smart device when interpreting the incoming iBeacon is faster when the RSSI values are filtered. In Fig. 11a, if the abnormal signals are not filtered, the difference between the time when the device interprets the arrival at a particular iBeacon and the actual arrival time are 0, 0.1, 1.8, 0.2, −1, 0, −0.6, −1.5, −0.7, −0.5, respectively, resulting in a total pre-react time of −2.2 s, which indicates a delay in responding. However, if the RSSI values are filtered, the differences are 0, 1.1, 1.8, 2.2, 2, 3, 2.4, 1.5, 2.3, and 2.5, respectively, yielding a total pre-react time of 18.8 s. Clearly, the smart device reacts faster if the RSSI values are filtered. This proves that Eq. (3) is effective in enabling a smoother navigation process with less latency. Another experiment was conducted to investigate the performance of iBeacons and Eq. (3) using a different speed. The procedure of the experiment is the same as the previous experiment; however, the driver now drives at a speed of 10 and 20 km/h. The experimental results are shown in Figs. 12 and 13, respectively.

With reference to Fig. 12a, if the RSSI values are not filtered, the difference between the time when the device interprets the arrival at a particular iBeacon and the actual arrival time is 0, −0.3, −0.6, −0.4, 0, −0.5, 0.1, 0.8, and 1.6, respectively, resulting

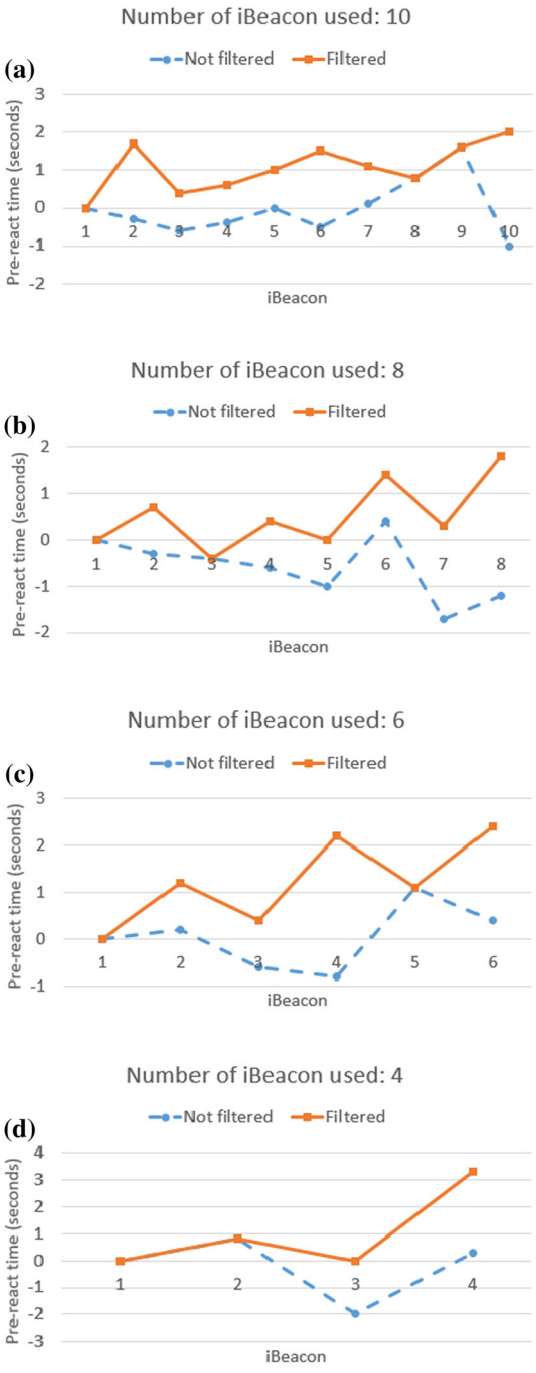


Fig. 12 Experimental results using *10, 8, 6,* and *4* iBeacons, respectively, with a speed of approximately 10 km/h

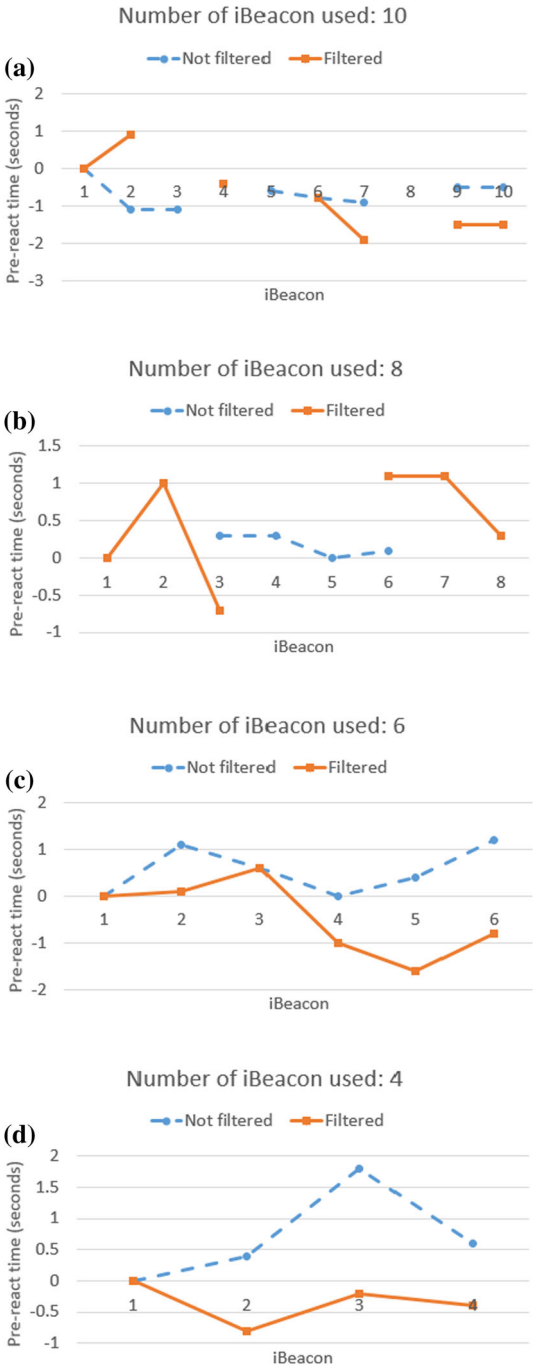


Fig. 13 Experimental results using *10, 8, 6,* and *4* iBeacons, respectively, with a speed of approximately 20 km/h

Table 3 iBeacon distance evaluation, average pre-react time, under speed of approximately 10 km/h

Number of iBeacons	Distance between iBeacons (m)	Average pre-react time (s)	Average pre-react time when filtered (s)
10	5	−0.03	1.07
8	6.4	−0.60	0.53
6	9	0.05	1.22
4	15	−0.23	1.03

Table 4 iBeacon distance evaluation, average pre-react time, under speed of approximately 20 km/h

Number of iBeacons	Distance between iBeacons (m)	Average pre-react time (s)	Average pre-react time when filtered (s)
10	5	Null	Null
8	6.4	Null	Null
6	9	0.55	−0.45
4	15	0.70	−0.35

in a total pre-react time of −0.3 s. If the RSSI values are filtered, the differences are 0, 1.7, 0.4, 0.6, 1, 1.5, 1.1, 0.8, 1.6, and 2, respectively, the total pre-react time is 10.7 s. This result proves that Eq. (3) works well when the speed is 10 km/h. However, Table 3 shows that the average pre-react time for both non-filtered and filtered results decreases when compared to 5 km/h (Table 4).

The next experiment with a speed of 20 km/h clarifies that the accuracy of the RSSI values decreases synchronously with the increment of speed. In Fig. 13a, b, the line graph for both filtered and non-filtered results are not connected, and there are approximately 10 iBeacon signals that are not interpreted. This is due to the fact that the vehicle moves too much in a given time (1 s), and the difference between RSSI values in the given time range is so large that the device application cannot interprets the exact position. However, in reality, 20 km/h is considered too fast in a car park. Thus, the efficiency and feasibility of the proposed Eq. (3) is not affected by the result.

5 Conclusion

The proposed system in this paper assumes that the parking lot is a gated area, and has a clear entrance and exit. To equip the proposed parking system, additional devices need to be installed: two RFID readers, iBeacons and ESP8266 modules. For the cars, only an RFID tag is required for each car. The RFID tag is registered with the user ID. Without the tag, the parking service is still available, however, other services, such as parking space reservation, will not available. The proposed system successfully implemented features such as car park recommendation, car park reservation, outdoor navigation and indoor navigation. For the recommendation of the car park, an original algorithm is created to enable users to adjust the recommendation condition to deter-

mine the recommend priority. The proposed system also used iBeacon to implement the indoor navigation feature. The abnormal iBeacon RSSI values are successfully filtered through a proposed algorithm, yielding a better average pre-react time by up to at least 43 %, compared to previous work.

References

1. Bock F, Eggert D, Sester M (2015) On-street parking statistics using LiDAR mobile mapping. IEEE Int Conf Intell Transp Syst:2812–2818
2. Benhassine S, Harizi R, Mraihi R (2014) Intelligent parking management system by multi-agent approach: the case of urban area of Tunis. IEEE Int Conf Adv Logist Transp:65–71
3. Suryady Z, Sinniah G, Haseeb S, Siddique M, Ezani M (2014) Rapid development of smart parking system with cloud-based platforms. IEEE Int Conf Inf Commun Technol Muslim World:1–6
4. Ji Z, Ganchev I, O'Droma M, Zhang X (2014) A cloud-based intelligent car parking services for smart cities. URSI General Assembly and Scientific Symposium of the International Union of Radio Science, pp 1–4
5. Pham T, Tsai M, Nguyen D, Dow C, Deng D (2015) A cloud-based smart-parking system based on Internet-of-Things technologies. IEEE Access 3:1581–1591
6. Geng Y, Cassandras C (2011) A new smart parking system based on optimal resource allocation and reservations. IEEE Intell Transp Syst:979–984
7. Wang H, He W (2011) A reservation-based smart parking system. IEEE Comput Commun Workshops:690–695
8. Hashimoto S, Kanamori R, Ito T (2013) Auction-based parking reservation system with electricity trading. IEEE Conf Bus Inf:33–40
9. Yan G, Yang W, Rawat D, Olariu S (2011) Smartparking: a secure and intelligent parking system. IEEE Intell Transp Syst Mag 3(1):18–30
10. Mainetti L, Palano L, Patrono L, Stefanizzi M, Vergallo R (2014) Integration of RFID and WSN technologies in a smart parking system. IEEE Int Conf Softw Telecommun Comput Netw:104–110
11. Bonde D, Shende R, Gaikwad K, Kedari A, Bhokre A (2014) Automated car parking system commanded by android application. IEEE Int Conf Comput Commun Inf:1–4
12. Zhao X, Zhao K, Hai F (2014) An algorithm of parking planning for smart parking system. World Congr Intell Control Autom:4965–4969
13. Asaduzzaman A, Chidella K, Mridha M (2015) A time and energy efficient parking system using zigbee communication protocol. In: IEEE Region South East Conference, pp 1–5
14. Postigo C, Torres J, Menendez J (2015) Vacant parking area estimation through background subtraction and transience map analysis. IET Intell Transp Syst 9(1):835–841
15. Lin X, Ho T, Fang C, Yen Z, Yang B, Lai F (2015) A mobile indoor positioning system based on ibeacon technology. IEEE Int Conf Eng Med Biol Soc:4970–4973