Implementation and Performance Analysis of Smart Attendance Checking Using BLE-Based Communications

Lorenzo Gabriel Alcantara, Alphonso Miguel Taylor Balagtas, Trixia Britania, Sean Kristian Garibay, Joshua Wyndel Uyvico, and Nestor Michael Tiglao(✉)

Ubiquitous Computing Laboratory, Electrical and Electronics Engineering Institute, University of the Philippines, Velasquez Street, Diliman, 1101 Quezon City, Philippines
{lorenzo.alcantara,alphonso.balagtas,trixia.britania, sean.kristian.garibay,joshua.uyvico,nestor}@eee.upd.edu.ph

Abstract. Current implementations of attendance checking in the University of the Philippines - Diliman (UPD) has been time consuming and easily cheated. The roll-call and pen-and-paper method cannot monitor student presence for the whole duration of the class session. Attendance checking systems using mobile technology and the Internet of Things attempt to mitigate these problems, but it also introduces new ones such as inclusivity, cost, and complex implementations. This study investigates the use of Bluetooth Low Energy (BLE) beacons, a mobile application, and a web server to create an attendance checking mechanism capable of eliminating queues and attendance cheating, monitoring student presence, and automating records. We created two procedures for sending information to our server, to determine the general advantages and disadvantages of each in terms of features, scalability, and cost-effectiveness. Procedure 1 mainly uses our Android application which was able to automate and record attendance checking in the background. Procedure 2 uses the ESP32 which was capable of scanning for information from these Android smartphones. Both of these Procedures send information to the web server to create reports based on available records. Overall, Procedure 1 served as the more scalable implementation due to its added features such as alarm systems, and ease of monitoring. However, Procedure 2 was simpler to set-up and more energy-efficient for smartphones since it relied on processing capabilities of the server.

Keywords: Smart attendance · Bluetooth low energy · Android · ESP32 · Cloud computing

1 Introduction

Recording student attendance is a common practice and is commonly mandatory in schools and universities. In a study conducted by Bekkering and Ward, there is a positive linkage between school attendance and academic performance [4].

H. Jin et al. (Eds.): CROWNCOM 2021/WiCON 2021, LNICST 427, pp. 253–268, 2022.
https://doi.org/10.1007/978-3-030-98002-3_19

The attendance rule in University of the Philippines - Diliman (UPD) stipulates that when the number of absences exceeds 20% of the total number of days for the semester, the student will be dropped from that class [10].

One method of attendance recording in the Electrical and Electronics Engineering Institute (EEEI) is by roll call. This is time consuming especially if the class lasts for only 1.5 h. Another method used is by passing an attendance sheet and encoding it in an electronic spreadsheet later on. As mentioned, the number of hours the student is present in class is important; however, both methods do not account for the students leaving the classroom earlier than scheduled.

In this study, a way to automate attendance recording was implemented. With the help of Bluetooth Low Energy (BLE) technology, two methods were implemented. A BLE beacon scans and advertises for the students' smartphones to record attendance. Depending on the Procedure, the attendance logs were sent to the server. Furthermore, the gathered information was stored in a database and was displayed both in the application and website. Smart attendance tracking also prevented students from cheating their attendance as it used their personal smartphones.

Paper Contributions: The major contributions of this paper are as follows: (1) testing the internal performance of an ESP32 as both scanner and beacon in a smart attendance checking system; (2) evaluation and comparison of the overall performance of the aforementioned Procedure 1 and Procedure 2; and (3) provides a basis for mitigating proxy attendance in hands-free attendance checking systems using ESP32 and BLE.

The rest of the paper is organized as follows. Section 2 provides a few related studies used as motivation and basis for our study. Section 3 describes the goals for our system in terms of the features it should provide. Section 4 discusses the methodology. Section 5 discusses our results and analysis. Finally, Sect. 6 presents our conclusion and recommendations for future work.

2 Related Work

Smartphone and Bluetooth Low Energy Implementations
Numerous studies have implemented different version of attendance systems through the use of BLE and smartphones. The main problems with the traditional attendance systems are cost, queues, and time. To address cost, several studies such as [3,8] used BLE beacons for a low-cost and inclusive system. While these systems are more convenient, their implementations are unable to prevent students from 'cheating' the system through proxy attendance. The works of [2,5,11] sought to prevent this through the use of the Bluetooth MAC address as unique identifiers per student. However, versions from Android 8 onward have made Bluetooth MAC address inaccessible. To solve this issue, AMAS, an attendance system made by Dankar et al. used the smartphone's IMEI address to verify the student instead of the MAC address [7]. Another issue is the possibility that students 'cheat' the system by recording attendance, then leaving the room after. To prevent this from happening, another study [5] implemented an

algorithm where they timestamped information to be able to detect student presence all throughout a given class period. However, this needs a modified beacon which would require additional costs, and IMEI became inaccessible starting from Android 10. Zoric et al. [12] suggests having the application run in the background as it obtains beacon information. This allows the application to automatically check attendance wherever the user is for a hands-free experience.

Our study adopted some of the features from these studies. The timestamp mechanism, and the background scanning suggestion from [5,12] were adopted to give the system a hands-free experience. We also want to prevent cheating so we adopted a more controlled authentication mechanism similar to [7]. Additionally, an administrative login is needed to change accounts for easier monitoring of the 'one smartphone per student' rule.

Bluetooth Low Energy

Among several wireless technologies utilized in Internet of Things applications, BLE is new and gaining popularity. We looked at the real world performance of BLE communication.

For large communication networks, network collisions and traffic become a pertinent issue. Cho et al. [6] studied the BLE discovery process and calculated the influence of parameters such as discovery latency and energy performance. They showed that while discovery latency increases as the number of BLE devices which act as receivers increase, it is not affected by the number of transmitters sending advertisements since there are few collisions among advertisers.

Bawiec and Nikodem [9] studied the effect of collisions on the communication performance of BLE when under the presence of over 200 communicating devices. Here, they explored the features of BLE connectionless mode which allows for one-way communication from end-devices to a central device. The results of this study show that by performing multiple advertisement transmissions of the same information within a set data interval, over 200 devices can simultaneously transmit data successfully with a high reception rate.

In these studies, the parameters used to quantify their systems were defined based on their respective objectives. Thus, our study adopted a similar approach and defined a set of parameters based on the functionality of our system. Furthermore, our study adopted Bawiec and Nikodem's [9] approach of using BLE's connectionless mode since the payload between our devices was relatively small and our system had no need for a Bluetooth pairing process. To observe the effects of latency and collisions on our system, we tested it using different configurations to see what broadcaster-listener setup would perform better.

3 System Features

We aimed to create an automated attendance checking system that was able to do the following.

- Eliminated queues through automated attendance checking using smartphones
- Monitored student presence throughout the class session to prevent cheating in attendance records
- Automated attendance encoding for professors, and provide a summary of reports for both students and professors based on the attendance checked
- Sent data to a web server through the application and the Bluetooth beacon

To create such a system, the following were accomplished:

- A BLE beacon capable of sending a unique identifier represented by their room assignment to the students' smartphones. This communicated with the web server for transmission of student information.
- An application capable of receiving information from the BLE beacon. It checked attendance automatically without the need for user prompts. It verified validity of attendance with the server and showed attendance progress of students.
- A web server capable of receiving information from the BLE beacon, and application. This web server automatically updated attendance information in their respective class databases, and showed such records to both student and professors.

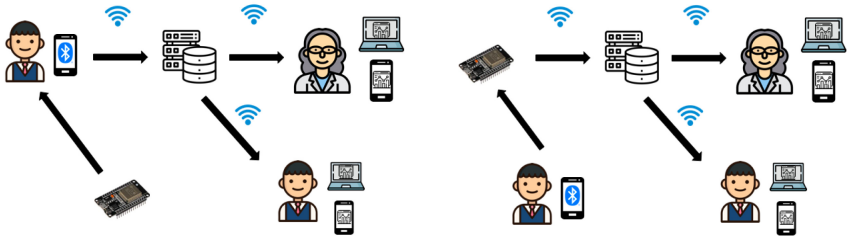
4 Methodology

This section describes the general Procedure of our implemented system. Our goal was to create a system that was capable of efficient data transmission through beacons and BLE and Wi-Fi enabled smartphones. To ensure that the students were indeed using their smartphone, they have to register in the centralized database. The database noted their basic information (name, course, year, schedule etc.). We limited the number of registered smartphones to one per student.

4.1 General System Architecture

Our study explored two procedures in data transmission to the server as seen in Fig. 1. Both architectures were implemented and tested in this study. They were evaluated based on specified performance metrics. Additionally, since we only conducted tests in a simulated classroom setting only, specified criterion were done along with the analysis of performance metrics to determine which was better for its intended use.

Figure 1a describes Procedure 1 where the system transmitted data to the server using smartphones. Students turned on the Bluetooth on their smartphone. Once they are in proximity of the beacon, an application scanned for the beacon's Universally Unique Identifier (UUID), major, and minor values. These were verified to match those values stored in the application. Also, to check for student presence throughout the class session, ping checks between



(a) Phone Transmission (b) Beacon Transmission

Fig. 1. General system architecture

the smartphone and beacon were done to record time stamps [5] depending on the duration of the class. Once internet connection was available, it sends the information to the server. Finally, the professors and students were able to view their respective attendance records through the application or website.

Figure 1b shows Procedure 2. It worked similar to Procedure 1 except that the beacon transmits the data to the server and that internet connection must always be available to do so. Relevant information from the application is fed to the beacon, and the rest of the steps are followed.

4.2 Application

The software used for the development of this application were Android Studio as the Integrated Development Environment (IDE) and DB Browser for SQLite as the medium of access for the local database.

General Logic. On initial startup of the application after its download, the user is prompted to set the username and student number for that smartphone. To safeguard it from fraudulent intentions, an additional prompt for administrative credentials (e.g., admin username and password) is required before proceeding with this process. The application will then connect to the server and get information related to that student number (e.g., classes enlisted and their schedule). Once that information is available within the smartphone, a 3-stage alarm system shown in Fig. 2 takes care of the rest of the functionality of the application in the background.

First, a daily alarm is set every day at 12 AM then application will get the user's class list for that day. A new alarm will be generated for each of the classes for the day. These class alarms will be triggered at the start of their period and will then generate another set of 10 alarms spread evenly throughout the duration of that class as 'ping checks' to determine the presence of the user.

Bluetooth Communication. The application can act either as a listener for Procedure 1 or as a broadcaster for Procedure 2, both of which function for

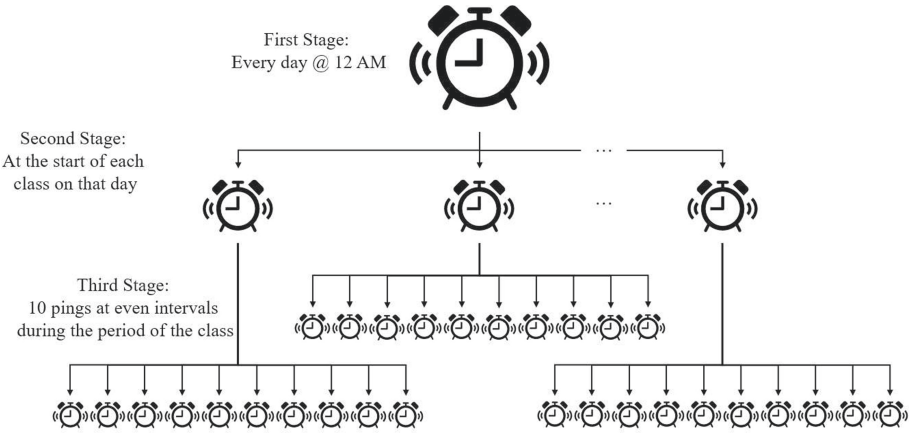


Fig. 2. 3-stage alarm system

a small period at every 'ping check' alarm. As a listener for Procedure 1, the application will scan for Bluetooth advertisements, specifically those sent by ESP32s in their broadcaster mode. The scan will be filtered by the UUID, major, and minor values to determine the correct broadcaster which corresponds to the class at that moment. As a broadcaster for Procedure 2, the application will simply broadcast Bluetooth advertisements with information containing a UUID and the student number of that smartphone for the ESP32 in its listener mode to receive.

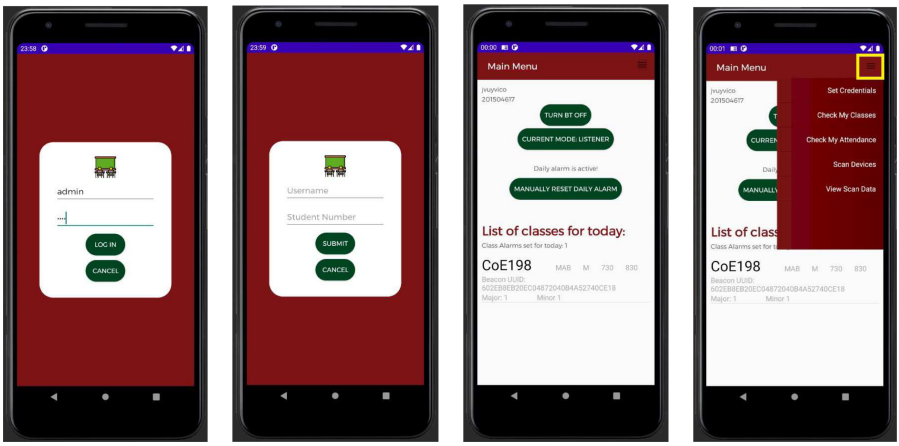
Interface. The main user interface (UI) of the application seen in Fig. 3 displays the basic functionalities the user would need, such as:

- A button to toggle the smartphone's Bluetooth on and off
- The status of the alarm system and a button to manually restart it
- A list of their classes for that day

On the top-right corner is a toolbar button which pops-up a list of a few more functionalities for the viewing of the user, such as:

- A list of all their classes
- A list of their recent attendance records
- A button to scan and list nearby Bluetooth devices

Communication with Server. The application communicates with the server through REST frameworks. All the necessary information was passed in JSON formats. The server used Django REST, while the application used Retrofit2 for their respective REST interfaces. Models were created for both the server and application for sending and receiving the datum in JSON. For the application,



(a) Admin Login (b) Set Credentials (c) Main UI (d) Toolbar

Fig. 3. User interface

these were used to get the list of JSON files in the different REST pages which were navigated using Retrofit's JSON API Interface. Majority of the communication happens whenever the user logs in, and updates the attendance. The server's attendance was updated by cross checking current server attendance with the local/application attendance list. This helps reduce redundancies when POST-ing by only sending the entries that are not yet uploaded.

4.3 ESP32

For the two Procedures defined in this study, an ESP32 is used as a broadcaster for Procedure 1 and as a listener for Procedure 2. The ESP32 also communicated with the web server in order to deliver the scanned data. The ESP32 programs were developed using C++ via the Arduino IDE. This study used the ESP32 boards package v1.0.4 by Esspressif Systems.

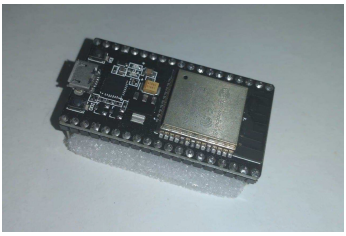


Fig. 4. An ESP32 used in this study

ESP32 as Broadcaster. In Procedure 1, the ESP32 board acted as a BLE beacon and broadcasts the necessary data to be received by the application. For this particular study, the ESP32 board is configured as an iBeacon and the program is a slight modification to the sample iBeacon program given by the ESP32 Arduino IDE Library.

The ESP32 iBeacon configuration can be divided into two parts: the advertisement parameters, and the advertisement data. The advertisement parameters involve the advertisement window, the advertisement interval, the transmit power, and other technical parameters. The advertisement window and interval are set with the common BLE convention used by currently available BLE products in the market. As such, the beacon is set to advertise for 100 ms with an interval of 10 s in between advertisements. The signal power was configured to transmit at the default high power rating such that it is able to transmit adequately within a small classroom.

The advertisement data contains the UUID, minor and major numbers. The UUID is used to filter out unwanted data from other BLE devices and is unique to all devices used in this study. It was also decided that the major and minor numbers in the beacon advertisement data would be used to identify each class's building and room number respectively. For the testing however, the study was not deployed in actual classrooms thus the major and minor number pair are used to represent classes instead. The following table shows a sample broadcast data that we used in our tests:

Table 1. ID assignments per class

Course	Room ID (Minor number)	Building ID (Major number)
CoE 198 MAB1	1	1
CoE 111	2	1
CoE 113	3	1
CoE 151	1	2
CoE 197D	2	2
EEE 100	3	2

When deployed, the ESP32 beacon is placed inside its corresponding classroom and continuously broadcasts its advertisement data to be received by participating smartphones. The low energy capability of the ESP32 allows it to enter deep sleep to conserve energy with a 10:0.1 sleep-to-broadcast ratio.

ESP32 as Listener. For Procedure 2, the ESP32 board acted as a BLE listener which listened to the periodic BLE broadcast of participating smartphones. The listener also connected to a Wi-Fi network to deliver the gathered data to the web server.

As a BLE listener, the scan window and interval must be configured to set the frequency at which the device listens for advertisements. Lower intervals and bigger scan times ensure faster detection, thus, it was set to listen for 100 ms with an interval of also 100 ms. For testing purposes, it was noted that a listening duration of 5 s is more than sufficient to detect at least 10 simulated BLE devices. The listener was also assigned integer values to act as the room and building ID. The same convention was used as in Table 1.

Every cycle of the listener must follow a specific sequence: get current time, listen for devices, then send data to server. This sequence occurs periodically as long as the device is turned on. Since the ESP32 does not have access to the current time locally, it must connect to the Network Time Protocol (NTP) server at the start of each sequence to get the actual current time. It will then switch to BLE to listen for any broadcasting smartphones. Each smartphone data is saved as a JSON object which is created as soon as that particular smartphone is scanned. The JSON object created for each smartphone contains the date and general time it was detected, the building and room ID of the listener device, and the Received Signal Strength Integer (RSSI). Each JSON object is compiled into a list which becomes the payload to be sent to the server. After the BLE scan, the device connects to the Wi-Fi and sends the JSON list via HTTP POST request. The following is a sample payload from the ESP32 listener to the web server:


```
[
{"dayStamp":"2021-05-31","timeStamp":"15:35:38","bid":1,"rid":1,"numID":201444444,"rssi":-64},
{"dayStamp":"2021-05-31","timeStamp":"15:35:38","bid":1,"rid":1,"numID":201355679,"rssi":-55},
{"dayStamp":"2021-05-31","timeStamp":"15:35:38","bid":1,"rid":1,"numID":202165432,"rssi":-64},
{"dayStamp":"2021-05-31","timeStamp":"15:35:38","bid":1,"rid":1,"numID":201222222,"rssi":-64},
{"dayStamp":"2021-05-31","timeStamp":"15:35:38","bid":1,"rid":1,"numID":201555555,"rssi":-70}
]
```


4.4 Web Server

The web server receives data coming from either the application or the ESP32 depending on the Procedure being used as specified in the General System Architecture.

Phone Side. For Procedure 1 implementation, the smartphone sends the relevant attendance information through the application to the web server. On the web server side, attendance data is simply received from the application and stored in the database. The changes made to the database are saved and are immediately available for download through the application to the relevant users. Note that the smartphone will only send a 'Present' attendance object if the smartphone satisfies the minimum number of pings. If a fewer number of pings is received, meaning the student did not stay within the classroom vicinity for the entire period, then the smartphone will send an 'Absent' attendance object to the web server.

ESP32 Side. For Procedure 2 implementation, ESP32 constantly sends data based on the smartphones it detects within its vicinity to the web server. Therefore, it can send multiple ESP32 data objects at once, depending on the number of smartphones detected. Once this data reaches the server, the server filters it out based on the room ID and building ID set by the given beacon's major and minor values respectively, as well as the data values that it receives which contains date, time, and student number. The server will only keep data that belongs to students registered in the system that have classes at the given date and time periods. The server needs to receive a certain number of ESP32 data objects (or pings) from the ESP32 to confirm that the student was there for the majority of the class. At the end of the day, additional scripts are run on the database to aggregate the received pings and update attendance accordingly.

Periodic Tasks. There are necessary periodic tasks that run daily in order to update the database. First, there is one script that will add 'Absent' attendances to the database based on the current attendance information. The script will check the current database to see if students that had classes that day do not yet have corresponding attendance data for each of the classes. This is necessary for Procedure 2 because it only records 'Present' attendances. For Procedure 1, it serves as a fail-safe if the application is unable to send the 'Absent' attendances for a given reason (e.g., smartphone runs out of battery during class period). Second, a pair of scripts that convert the ESP32 data objects (or pings) received from the ESP32 implementation and convert these pings into an actual attendance object at the end of the day. These scripts simply aggregate the number of pings that it received corresponding to each class per given user. Note that these scripts run at the end of the day in order to only update the database when all of the attendance-related information has already been received during the class hours.

5 Results and Analysis

This section tackles the internal performance of our system. Latency and Bluetooth advertisement reception were tested to quantify connectivity, and Bluetooth communication between the system components.

Procedure 1 Performance - System Connectivity. The application and beacon connection was tested by using the NRF Connect application by Nordic Semiconductors. The application was installed on a Samsung Galaxy A20s. The connection was measured by scanning the beacon multiple times while recording its RSSI value. The beacon was scanned at four distances with no physical obstruction in between. To provide a baseline, the ESP32 was configured to its default transmission power of 0 dBm at 1 m. Scanning was continuous until 500 entries were achieved. It is also important to note that alongside the ESP32, other wireless connections were present.

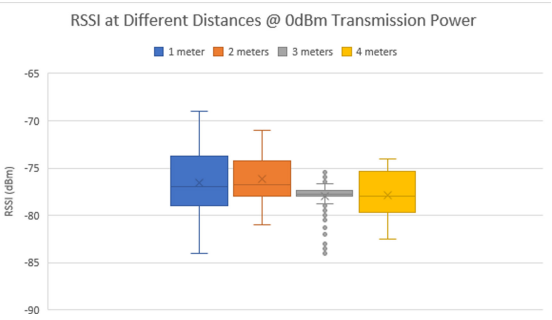


Fig. 5. RSSI at different distances @ 0 dBm transmission power

The compiled RSSI data is visualized in Fig. 5. Averaging the results of the 1 m, 2 m, 3 m, 4 m tests, the RSSI values are −76.257 dBm, −76.249 dBm, −78.0685 dBm, and −77.831 dBm, respectively. Here we see a slight decrease on the RSSI values as distance increases, which is expected. However, it is important to note that these values can differ depending on the voltage supplied to the ESP32. Although the attained RSSI values range in between the "Good" and "Low" range [1], it will not affect our system. In our case, the application does not need to connect to the beacon since they only scan for their information.

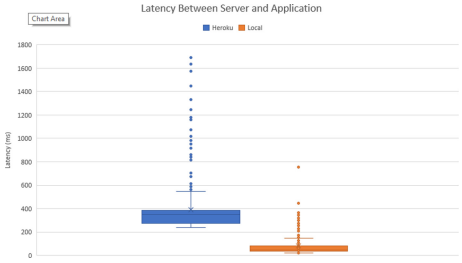


Fig. 6. Latency between the application and server

The application and server latency was measured through the application. A function was made to record the latency by subtracting the time before the HTTP request is sent, and time after the HTTP response is received. The data is visualized in Fig. 6. The deployment was done in Heroku and our local connection. The average latency is recorded at 394 ms for Heroku, and 80 ms for the local connection. If the outliers or spikes are removed, the average latency decreases to 337 ms and 65 ms respectively. Heroku gives higher latency than the local connection because Heroku is a free cloud service, and it is meant for early developments only. If lower latency is desired, the server must be ran either locally or in a better cloud service.

Procedure 1 Performance - Bluetooth Communication. To test the performance of the Bluetooth communication between the ESP32 and the application, the ESP32 was programmed to have a sleep cycle with a duration of 10.1 s where it was advertising for 100 ms and asleep for 10 s. It was plugged to a wall socket with an adapter outputting 5 V and 2.4 A and placed around 5 m away from the smartphone with minimal physical obstructions.

It should be noted that the application's Bluetooth scanning capabilities are weaker when it is ran in the background as compared to when the application is in the foreground. This is due to the limitations Android has imposed on background tasks in order to preserve battery life. The background work done by the application may take more time than the amount Android allocates for background tasks before killing them. Given this, a workaround was to disable the 'battery optimisation' feature for the application.

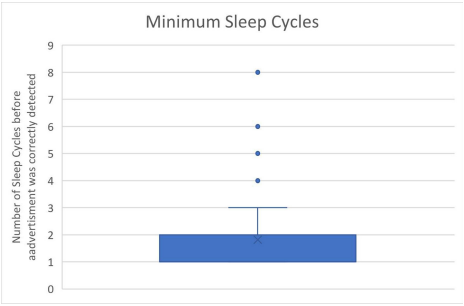


Fig. 7. Minimum sleep cycles parameter

For this test, the application's scan time was set to accommodate 12 sleep cycles of the 10.1 s sleep cycle configuration of the ESP32. Figure 7 shows 300 data points collected to analyze the minimum number of sleep cycles needed before the advertisement was correctly received. With a mean of 1.736 and a standard deviation of 1.2, it is sufficient to say that 3 sleep cycles worth of scan time is enough to account for most instances and correctly receive the advertisement within that time period. For one ping check, 30 s would be a reasonable time for the application to do some background work while keeping the smartphone's battery consumption to a minimum.

With the same setup, 200 data points were collected to analyze the ratio between the number of sleep cycles where the advertisement was correctly received over the 12 total sleep cycles. Figure 8 shows this data with a mean of 0.378 and a standard deviation of 0.154. While an average of 4 out of 12 sleep cycles where the advertisement is correctly received seems relatively low, what is important for our system is that it correctly receive the advertisement at some point and it does so during the first few sleep cycles as seen by the analysis on the minimum sleep cycles parameter.

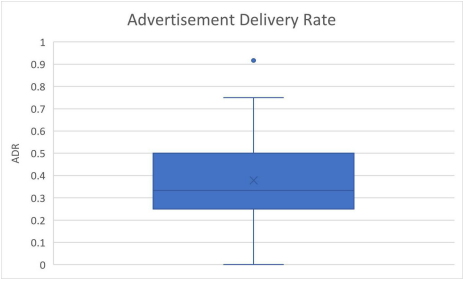


Fig. 8. Advertisement delivery rate parameter

Procedure 2 Performance - System Connectivity. The ESP32 to server latency was measured by recording the response time from sending packets to the local and Heroku servers. The packets, carrying the same payload, were sent to the server 500 times in order to collect the data points.

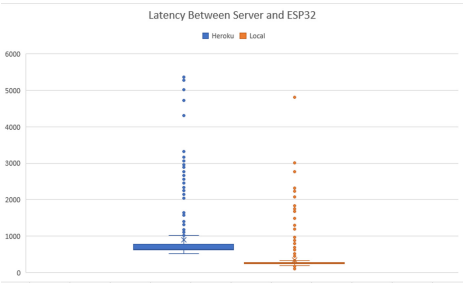


Fig. 9. Latency between ESP32 and the server

As shown in Fig. 9, the average response time of the Heroku server is 899.14 ms with a standard deviation of 748.036 ms while the average response time of the local server is 353.09 ms with a standard deviation of 391.79 ms. The higher latency of the Heroku server could be attributed to the fact that this transmission is done over the internet versus the transmission done over the local area network. A stress test was also conducted to determine the Packet Reception Rate (PRR) of the local server. Each test sent 1000 of the same packets consecutively to the server. The following table shows the PRR for each test:

With an average PRR of 99.25%, it is enough to correctly determine the presence of the students throughout each class.

Table 2. Packet Reception Rate for ESP32 to server

Test no.	PRR
1	99.30%
2	98.20%
3	100.00%
4	99.50%

Procedure 2 Performance - Bluetooth Communication. The Bluetooth communication test for Procedure 2 measures the speed and efficiency of the ESP32 listener when it comes to detecting smartphone broadcasters. For this test, we used a total of five smartphones each of different brands and models placed at various distances from the listener device. The listener scans in cycles of 100 ms with an interval of 100 ms. The smartphones advertise in bursts of one minute continuous broadcasts. The test recorded the multiple detection times for every smartphone regardless of distance from the listener. 500 data points were gathered.

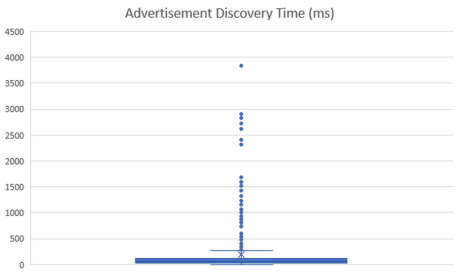


Fig. 10. Advertisement discovery time parameter

Figure 10 shows the advertisement discovery time for all smartphones during the multiple test runs. Most discovery times fall between the 20 ms to 300 ms range with the average scan time being 199.50 ms with a standard deviation of 2.24 ms. For this test, a scanning phase of 5 s was determined to be sufficient time to detect five smartphones within the area. Given the average scan time gathered from the data point, a regular classroom with about 30 students will need a scan duration of 6 s in order to detect the majority of smartphones while a larger classroom of 100 students will need a scan duration of 20 s assuming that every broadcasting smartphones is within range. We also measured the minimum scan cycles needed in order to detect any broadcaster given the 200 ms scan period defined earlier.

As shown in Figure 11, the vast majority of smartphones require only 1 scan cycle to be detected. Of the 572 data points for this test, 83.39% required only 1 scan cycle to be detected.

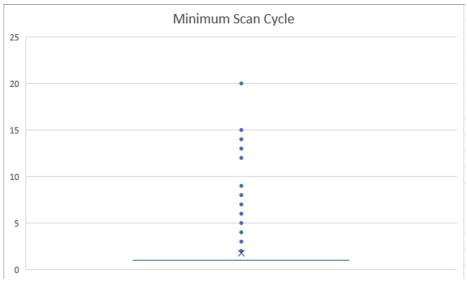


Fig. 11. Minimum scan cycle time parameter

6 Conclusion and Future Work

This study aimed to develop a smart attendance system using BLE using an Android application, an ESP32, and a web server. BLE served as an important mode of checking attendances through its advertising and scanning functionalities. Two Procedures were done to send the data to the server. The first Procedure uses the smartphone to scan beacons and update attendance. The second Procedure uses the ESP32 to do the same, except it scans for smartphones.

The application was able to successfully log attendance of the user. It was able to automatically start scanning for beacons depending on the class set in the user's schedule. The attendance log would then be stored within the application's local database after every class which would then be sent to the server to update the professors. It was also able to implement additional features such as schedule tracking, and class notifications. Lastly, the application was also able to broadcast itself to the ESP32 if the second Procedure is implemented.

The web server was able to successfully receive and process information from the smartphone in Procedure 1, and from the ESP32 in Procedure 2. Relevant attendance and class scheduling information are easily available for viewing through the web page as well. Certain user profiles (admin and teachers) were also able to access the database and make manual adjustments as needed.

Ultimately, Procedure 1 serves as the more scalable implementation between the two Procedures. It also has the added bonus of some helpful features such as an alarm system and the ability to easily check schedule information. On the other hand, Procedure 2 is simpler to set-up, and lessens the power consumption needed by the smartphones by taking advantage of the processing capabilities of the server. For future work, we recommend deploying the system in a real classroom setting. The UI can be also improved further.

References

1. RSSI level and a signal strength. https://www.netspotapp.com/what-is-rssi-level.html
2. Al-Shezawi, M., Yousif, J., Al-balushi, I.: Automatic attendance registration system based mobile cloud computing. Int. J. Comput. Appl. Sci. **2**(3), 116–122 (2017). https://doi.org/10.24842/1611/0037
3. Bae, M., Cho, D.: Design and implementation of automatic attendance check system using BLE beacon. Int. J. Multimedia Ubiquitous Eng. **10**, 177–186 (2015). https://doi.org/10.14257/IJMUE.2015.10.10.19
4. Bekkering, E., Ward, T.: Class participation and student performance: a tale of two courses. Inf. Syst. Educ. J. **18**(6), 86–98 (2020). https://files.eric.ed.gov/fulltext/EJ1258148.pdf
5. Boric, M., Fernandez, A., Redondo, R.: Automatic attendance control system based on BLE technology. In: Proceedings of the 15th International Joint Conference on e-Business and Telecommunications (ICETE), vol. 1, pp. 289–295 (2018). https://doi.org/10.5220/0006830202890295
6. Cho, K., Park, G., Cho, W., Seo, J., Han, K.: Performance analysis of device discovery of Bluetooth low energy (BLE) networks. Comput. Commun. **81**, 72–85 (2016)
7. Dankar, A., Kundapur, P.P.: Automated mobile attendance system (AMAS). In: 2019 International Conference on Advances in Computing, Communication and Control (ICAC3), pp. 1–6 (2019). https://doi.org/10.1109/ICAC347590.2019.9036787
8. Hidayat, M.A., Simalango, H.M.: Students attendance system and notification of college subject schedule based on classroom using iBeacon. In: 2018 3rd International Conference on Information Technology, Information System and Electrical Engineering (ICITISEE), pp. 253–258 (2018). https://doi.org/10.1109/ICITISEE.2018.8720948
9. Nikodem, M., Bawiec, M.: Experimental evaluation of advertisement-based Bluetooth low energy communication. Sensors **20**(1), 107 (2019). https://doi.org/10.3390/s20010107
10. University of the Philippines: Attendance (revised up code: Art. 346). https://our.upd.edu.ph/files/acadinfo/ATTENDANCE.pdf
11. Puckdeevongs, A., Tripathi, N.K., Witayangkurn, A., Saengudomlert, P.: Classroom attendance systems based on Bluetooth low energy indoor positioning technology for smart campus. Information **11**(6), 329 (2020). https://doi.org/10.3390/info11060329
12. Zorić, B., Dudjak, M., Bajer, D., Martinović, G.: Design and development of a smart attendance management system with Bluetooth low energy beacons. In: 2019 Zooming Innovation in Consumer Technologies Conference (ZINC), pp. 86–91 (2019). https://doi.org/10.1109/ZINC.2019.8769433