Determining the error distribution of BLE beacons at antenna near and far fields

Anton Novytskyi[1,†], Volodymyr Sokolov[1,*,†], Larysa Kriuchkova[1,†] and Pavlo Skladannyi[1,†]

[1]Borys Grinchenko Kyiv Metropolitan University, Bulvarno-Kudriavska Str., 18/2, Kyiv, 04053, Ukraine

Abstract

This paper investigates the error distribution and packet loss characteristics of Bluetooth Low Energy (BLE) beacons across near-field and far-field antenna zones. A custom experimental setup was developed using ESP32-based transmitters and an Ubertooth One receiver, supported by automated data collection scripts and time-synchronized measurements. The study quantifies how environmental conditions, distance, and transmission power affect BLE signal stability, particularly focusing on packet reception rates and RSSI behavior over distances from 0 to 2 meters. Cubic regression was employed to model the error distribution, highlighting a critical peak within the near-field boundary. The results emphasize the importance of adaptive beacon power management and advanced filtering for precise indoor positioning, especially in industrial and interference-prone environments.

Keywords

Bluetooth Low Energy, near-field, far-field, RSSI, packet loss, error distribution, Ubertooth One

1. Introduction

BLE beacon errors in indoor positioning systems are primarily caused by environmental factors such as multipath effects, signal fading, and interference from metallic objects, which distort the Received Signal Strength Indicator (RSSI) and lead to inaccurate distance estimations. Advanced methods like beacon weighting, Bayesian filtering, and location fingerprinting have been shown to reduce these errors. For instance, beacon weighting based on RSSI distributions can lower mean distance errors by over 66% compared to traditional three-point positioning, achieving errors as low as 0.45 meters [1]. Bayesian filtering techniques, including Kalman and particle filters, can improve proximity estimation accuracy by up to 30% when the beacon and receiver are within 3 meters, effectively mitigating environmental noise [2]. Location fingerprinting, especially with dense beacon deployment, can achieve sub-2.6-meter errors 95% of the time, outperforming WiFi-based systems [3]. In challenging environments like factories, selecting beacons based on RSSI variance helps exclude non-line-of-sight signals, reducing average positioning errors [4]. Adjusting beacon transmission power and using multiple power levels can also keep estimation errors below one meter [5, 6]. Overall, minimizing BLE beacon errors requires a combination of optimized beacon placement, advanced signal processing algorithms, and adaptive filtering to ensure reliable and precise indoor positioning across diverse environments [4].

While previous papers have addressed interference issues in Wi-Fi [7, 8, 9], Bluetooth [10], and ZigBee [11] wireless networks, the goal of this paper is to determine the impact of near-field and far-field antenna zones on packet loss.

CH&CMiGIN'25: Fourth International Conference on Cyber Hygiene & Conflict Management in Global Information Networks, June 20–22, 2025, Kyiv, Ukraine

*Corresponding author.

†These authors contributed equally.

aanovytskyi.fitm24m@kubg.edu.ua (A. Novytskyi); v.sokolov@kubg.edu.ua (V. Sokolov); l.kriuchkova@kubg.edu.ua (L. Kriuchkova); p.skladannyi@kubg.edu.ua (P. Skladannyi)

0009-0009-7766-1441 (A. Novytskyi); 0000-0002-9349-7946 (V. Sokolov); 0000-0002-8509-6659 (L. Kriuchkova); 0000-0002-7775-6039 (P. Skladannyi)

2. Sources review

Using multiple transmission power levels in BLE beacons can significantly influence error rates in industrial environments, where obstacles, metallic surfaces, and interference often challenge signal propagation. Studies show that higher transmission power (e.g., +4 dBm) generally improves positioning accuracy, as stronger signals are less susceptible to attenuation and multipath effects. Still, the improvement over the lowest power settings (e.g., −20 dBm) may be modest—sometimes only around 12%—depending on the environment and beacon density [12, 13]. Customizing transmission power for each beacon, rather than a uniform setting, can enhance localization accuracy by adapting to specific site conditions and reducing overlap or interference between beacons [14]. Employing multiple power levels at each beacon point can also help keep estimation errors below one meter, allowing the system to balance coverage and precision dynamically [5]. However, in complex industrial settings, even with optimal power settings, RSSI-based distance estimation remains vulnerable to environmental noise, and additional filtering techniques (such as Gaussian or Kalman filters) are often needed to recover or maintain accuracy [13, 15]. Moreover, while higher power increases coverage, it can also introduce more interference and reduce battery life, so a trade-off must be managed [3, 12]. Overall, using multiple and adaptive transmission power levels, advanced filtering, and careful beacon placement can minimize BLE beacon error rates and improve reliability in industrial environments [13].

Distance significantly impacts BLE packet loss, with the probability of successful packet reception generally decreasing as the distance between devices increases. Empirical models show that packet reception probability can be described as a quadratic function of distance, beacon power, and advertising frequency, with greater distances leading to higher packet loss rates and lower received signal strength (RSSI) values; for example, mean RSSI drops from about −65.5 dBm at 1 meter to −90.8 dBm at 16 meters, indicating weaker signals and more frequent packet loss at longer ranges [16, 17, 18]. In addition, environmental factors such as interference, human presence, and device orientation can further exacerbate packet loss, with interference alone reducing packet delivery rates by 13% to 40% in some scenarios [18, 19, 20]. Overcrowded environments can also cause substantial signal strength and coverage fluctuations, with distance estimation accuracy dropping by up to 41% in highly crowded settings [21, 22]. These findings highlight the importance of considering distance and environmental conditions when designing BLE-based systems, especially for applications like indoor localization, where packet loss directly affects accuracy [16].

3. Design of the experimental setup

To conduct the experiment, a BLE packet transmitter (with the required frequency, signal level, and content), a receiver and a controller were used, as shown in Figure 1.

For ease of synchronization, the transmitter and receiver (Figure 2) were connected to the controller directly, minimizing measurement errors.

However, additional software modifications were required to experiment.

4. Transmitter hardware selection

When selecting available boards for the transmitter, Software Defined Radios, Bluetooth dongles, and single-board computers were considered. However, for miniaturization and the availability of SDK and code examples, the choice fell on the family of single-board computers in the ESP32 series. Table 1 shows the characteristics of the board modifications.

According to the table above, the choice of the ESP32-S3 N16R8 Type-C board is advantageous: the dual-core LX7 processor allows you to handle several tasks in parallel (for example, receiving commands and generating packets), and a large amount of SRAM and Flash memory makes it possible to cache a sufficient amount of data, and the program code can be more functional, so make this board universal for surveillance tasks. With all the available boards on sale, the ESP32-S3 N16R8 Type-C board is the most

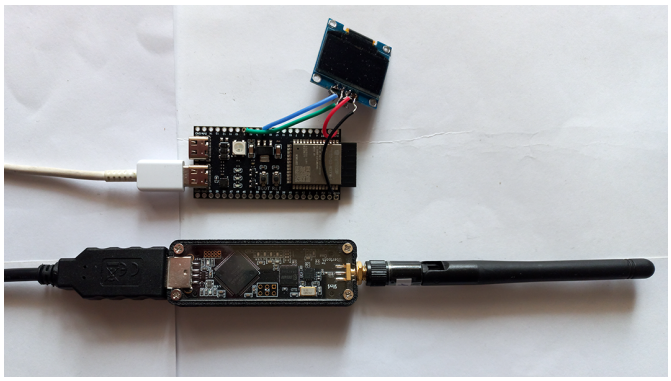

Figure 1: General view of the transmitter (ESP32) and receiver (Ubertooth One).

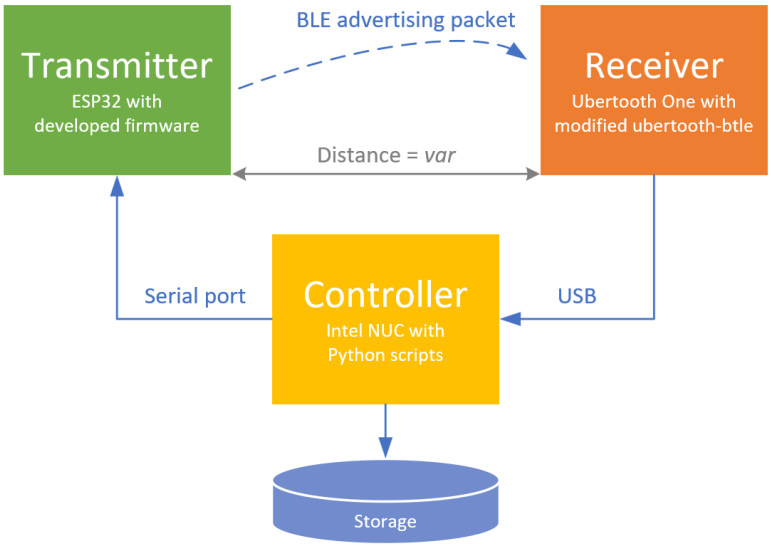

Figure 2: Experimental setup diagram.

Table 1
Comparative Characteristics of Single-board Computers of the ESP32 Series [23, 24]

Chip	ESP32	ESP32-S2	ESP32-C3	ESP32-S3	ESP32-C6	ESP32-H2
Processor	Xtensa LX6	Xtensa LX7	RISC-V	Xtensa LX7 (×2)	RISC-V (×2)	RISC-V
SRAM, KB	520	320	400	512	512	320
Flash, MB	4–16	4–16	4	4–16	4–8	1–4
Frequency, MHz	240	240	160	240	160	96
GPIO	34	43	22	45	27	20
Antenna	PCB	PCB	PCB/chip	PCB/planar	PCB/ceramic	PCB
Issue date	2016	2019	2020	2021	2022	2023
Price, USD	7	3	5	6	6	—

universal for a dual-core processor. In addition, some boards are equipped with Micro-SMA connectors, which allow the use of external antennas.

5. Transmitter software preparation

To secure work from BLE, use external libraries:

1. NimBLE-Arduino is a BLE stack based on Apache Mynewt, which provides an API for transmitting

advertising packages and securing connections [25].

2. U8g2 for OLED screen (optional) [26].

Example of packet generation:

```
#include <Arduino.h>
#include <NimBLEDevice.h>
#include <NimBLEExtAdvertising.h>
String mac;
void setup() {
Serial.begin(115200);
while(!Serial) delay(10);
NimBLEDevice::init("ESP32_Beacon");
NimBLEDevice::setPower(ESP_PWR_LVL_N24);
}
void loop() {
static NimBLEExtAdvertising* pExtAdv = NimBLEDevice::getAdvertising();
if (Serial.available()) {
String ts = Serial.readStringUntil('n');
ts.trim();
NimBLEExtAdvertisement extAdv;
extAdv.setLegacyAdvertising(true);
extAdv.setManufacturerData(ts.c_str());
if (pExtAdv->isAdvertising()) {
pExtAdv->stop();
delay(10);
}
pExtAdv->removeInstance(0);
if (!pExtAdv->setInstanceData(0, extAdv)) {
Serial.println("Bad data func setInstanceData()");
return;
}
if (pExtAdv->start(0, /*duration=*/0, /*maxEvents=*/1)) {
unsigned long start = millis();
while (pExtAdv->isAdvertising() && (millis() - start < 500)) {
delay(10);
}
if (!pExtAdv->isAdvertising()) {
Serial.println("Advertised done (confirmed via polling)");
} else {
Serial.println("Timed out waiting for adv complete");
pExtAdv->stop();
}
}
}
}
```

The strength of the signal is set to the additional NimBLEDevice::setPower(ESP_PWR_LVL_N24), which will send packets that receive the following values:

ESP_PWR_LVL_N24 = 0 (corresponding to −24 dBm)

ESP_PWR_LVL_N21 = 1 (−21 dBm)

...

ESP_PWR_LVL_N0 = 8 (0 dBm)

...

ESP_PWR_LVL_P18 = 14 (+18 dBm)

ESP_PWR_LVL_P21 = 15 (+21 dBm)

To assemble the code, PlatformIO was used with the commands:

1. pio run (build the project in “release” mode).
2. pio run -e esp_env -t debug (build the project in “debug” mode).

Basic setup of platformio.ini:

```
[env:esp32-s3-devkitm-1]
platform = espressif32
board = esp32-s3-devkitm-1
framework = arduino
lib_deps = h2zero/NimBLE-Arduino@2̂.3.2
build_type = release
build_flags =
-O3
-ffunction-sections
-fdata-sections
-Wl,–gc-sections
-DNDEBUG
-DCONFIG_BT_NIMBLE_ROLE_BROADCASTER=1
-DCONFIG_BT_NIMBLE_EXT_ADV=1
```

Also in Visual Studio Code is an extension called Platformio, which allows you to automate and visualize all configurations and commands.

6. Receiver hardware selection and software preparation

To collect packets, you can use standard BLE modules (for example, the utility BlueZ Bluemoon [27]), but they limit the controllability of the process of saving packets. Similar limitations are inherent in hardware packet sniffers (for example, nRF52833/40 [28] or TI CC2540 [29]). Therefore, the flexible open-source Ubertooth One platform was chosen [30].

Ubertooth-btle is a utility used by Ubertooth One to store and analyze BLE traffic. The utility is compiled in C and, if changes are necessary, it must be recompiled. The standard view of the software looks like this:

```
printf("systime=%u freq=%d addr=%08x delta_t=%.03f ms rssi=%dn",
systime, rx->channel + 2402, lell_get_access_address(pkt),
ts_diff / 10000.0, rx->rssi_min - 54);
```

The systime is in seconds, which does not match our world’s accuracy level. There is a need to increase the accuracy by adding the nanosystime parameter, which will represent the hour in nanoseconds.

Such changes are made to improve the accuracy. Programming through the Python library could also be done. Still, daily filtering by the device’s MAC address on the hardware level, only on the software level, would enhance the processing package and increase the loss of time.

7. Automation of the data collection process

To control the process of sending packages to software packages, a Python script that sends packages through the Serial port (via USB) to the ESP32 transmission:

```
import serial
import time
SERIAL_PORT = ’/dev/ttyACM0’
BAUDRATE = 115200
ITERATIONS = 1000
idle_timeout = 10
def measure_one(ser, num_iter):
```

```
sent_ts = time.time_ns()
ser.write(f"{sent_ts}n".encode())
last_try = time.time()
num_try = 0
while True:
num_try += 1
print(f"Try #{num_try} for {sent_ts} ## {num_iter} item")
timeout = idle_timeout - int((time.time() - last_try))
if timeout <= 0:
break
line = ser.readline()
if line and str(line).startswith("b'Advertised done"):
break
return 1
def main():
ser = serial.Serial(SERIAL_PORT, BAUDRATE, timeout=1)
try:
for i in range(ITERATIONS):
measure_one(ser, i)
finally:
ser.close()
if __name__ == '__main__':
main()
```

The function measure_one(ser, num_iter) is a whole block of code that sends just one packet and synchronously reads the ESP32 output about the capture and processing of the packet.

Here's a Bash script to make it easier to carry out the experiment:

```
#!/bin/bash
START_NUM=${1:-0}
INCREMENT=5
CURR_NUM=$START_NUM
echo "Starting from number: $CURR_NUM"
while true; do
echo "Starting Ubertooth and Python storage..."
ubertooth-btle -t 10:20:ba:4b:b1:41 -n > recieved_packets.txt &
PID_UBER=$!
python3 send_packets.py > sented_packets.txt
echo "Python has finished. Waiting for 3 seconds..."
sleep 3
echo "Completing Ubertooth (PID=$PID_UBER)..."
kill $PID_UBER
wait $PID_UBER 2>/dev/null
FOLDER_NAME="$CURR_NUM cm"
mkdir -p "$FOLDER_NAME"
mv recieved_packets.txt "$FOLDER_NAME/"
mv sented_packets.txt "$FOLDER_NAME/" 2>/dev/null || echo "sended_packets.txt not found"
echo "Files moved to $FOLDER_NAME"
read -p "Press Enter to start the cycle..."
CURR_NUM=$((CURR_NUM + INCREMENT))
done
```

Shard measurements were carried out at 5 cm intervals of 2 m, the script allows you to simplify the time required to save data, and itself:

1. START_NUM=${1:-0} is specified tracking number, which will be in cm.

Table 2
Experimental Results for the Number of Errors as a Function of Distance

i	l_i, m	E_i	i	l_i, m	E_i	i	l_i, m	E_i
1	0	175	15	0.70	214	29	1.40	221
2	0.05	181	16	0.75	212	30	1.45	219
3	0.10	187	17	0.80	211	31	1.50	223
4	0.15	209	18	0.85	210	32	1.55	225
5	0.20	222	19	0.90	208	33	1.60	222
6	0.25	229	20	0.95	209	34	1.65	229
7	0.30	233	21	1.00	207	35	1.70	241
8	0.35	246	22	1.05	200	36	1.75	238
9	0.40	239	23	1.10	203	37	1.80	242
10	0.45	237	24	1.15	201	38	1.85	247
11	0.50	233	25	1.20	205	39	1.90	250
12	0.55	227	26	1.25	210	40	1.95	254
13	0.60	224	27	1.30	208	41	2.00	256
14	0.65	219	28	1.35	216			

2. Ubertooth-btle -t 10:20:ba:4b:b1:41 -n > recieved_packets.tx – launch the modified command ubertooth-btle, which is used to catch packets sent from the ESP32.

3. Python3 send_packets.py > sented_packets.txt This runs the package sending script and redirects its output to a text file.

4. “mv recieved_packets.txt "$FOLDER_NAME/"" and "mv sented_packets.txt "$FOLDER_NAME/" 2>/dev/null || echo "sended_packets.txt not found" are automated movement of created result files of robot programs to a directory under the name that coincides with the location in cm.

5. CURR_NUM=$((CURR_NUM + INCREMENT)) increase the cycle number according to the duration in cm.

8. Features and implementation problems

During the implementation, some problems arose. For example, the issue of buffering the subprocess output and parallelism with threads when transferring data is that Python has limited true parallelism, which entails alternating code execution by different threads, which reduces the accuracy of measurements.

It should be noted that the launch order matters; if you start sending packets first and then ubertooth-btle, packet loss may occur when scanning through the utility, since the initialization of the ubertooth-btle software tool takes some time.

When using a standard BLE stack based on BLEDevice and BLEAdvertising, there is a problem of controlling the number of sent packets (events): instead of controlling the number of events, only advertising time control is available — as a result, if the advertising time is too short, no packets are sent, and if it is too long, several are sent.

9. Verification of the experimental results

An experiment was conducted using the established experimental setup. One thousand time-stamped Beacon packets were sent at different distances, and the number of lost packets was measured (see Table 2). The time delay between the received packets was also calculated.

The delay value was ignored since the first packet incorrectly returns its time. The results can be approximated using the cubic regression equation

$$E(l) = al^3 + bl^2 + cl + d, \tag{1}$$

Table 3
Coefficient Sum Calculation

	l_i, m	E_i	l_i^2, m^2	l_i^3, m^3	l_i^4, m^4	l_i^5, m^5	l_i^6, m^6	$l_i E_i, m$	$l_i^2 E_i, m^2$	$l_i^3 E_i, m^3$
Σ	41	11135	55.350	84.050	136.133	229.667	398.514	9265.200	12717.905	19616.834

where l is the distance between receiver and transmitter, a, b, c, and d are the coefficients. To calculate the coefficients, we will make a system of equations

$$\begin{cases} a\sum\limits_{i=1}^{n} l_i^3 + b\sum\limits_{i=1}^{n} l_i^2 + c\sum\limits_{i=1}^{n} l_i + dn = \sum\limits_{i=1}^{n} E_i \\ a\sum\limits_{i=1}^{n} l_i^4 + b\sum\limits_{i=1}^{n} l_i^3 + c\sum\limits_{i=1}^{n} l_i^2 + d\sum\limits_{i=1}^{n} l_i = \sum\limits_{i=1}^{n} l_i E_i \\ a\sum\limits_{i=1}^{n} l_i^5 + b\sum\limits_{i=1}^{n} l_i^4 + c\sum\limits_{i=1}^{n} l_i^3 + d\sum\limits_{i=1}^{n} l_i^2 = \sum\limits_{i=1}^{n} l_i^2 E_i \\ a\sum\limits_{i=1}^{n} l_i^6 + b\sum\limits_{i=1}^{n} l_i^5 + c\sum\limits_{i=1}^{n} l_i^4 + d\sum\limits_{i=1}^{n} l_i^3 = \sum\limits_{i=1}^{n} l_i^3 E_i \end{cases} \tag{2}$$

Then, fill out Table 3 with the summarization results.

And obtain Equation (2) with sums from Table 2, a system of linear equations

$$\begin{cases} 84.050\,a + 55.350\,b + 41\,c + 41\,d = 11135 \\ 136.133\,a + 84.050\,b + 55.350\,c + 41\,d = 9265.200 \\ 229.667\,a + 136.133\,b + 84.050\,c + 55.350\,d = 12717.905 \\ 398.514\,a + 229.667\,b + 136.133\,c + 84.050\,d = 19616.834 \end{cases} \tag{3}$$

The total sought cubic regression equation is as follows:

$$E(l) \approx 71l^3 - 196l^2 + 151l + 189. \tag{4}$$

From the experimental plot and graph, the regression equations are presented in Figure 3. The experiments were conducted on the 37 BLE channel corresponding to the frequency 2.402 GHz. Let's calculate the antenna far-field limits for this frequency, because the linear size of the antenna is less than its linear dimensions, so we use the formula for approximate calculation $r \sim 6...10, \lambda = 0.75...1.25$ m. The regression plot shows a local maximum of errors within the near-field boundary.

To assess the significance of regression and correlation parameters, we determine the average number of errors

$$\overline{E} = \frac{1}{n}\sum_{i=1}^{n} E_i = 220.5. \tag{5}$$

Let's determine the correlation index

$$R = \sqrt{1 - \frac{\sum_{i=1}^{n}\left(E_i - \widehat{E}_i\right)^2}{\sum_{i=1}^{n}\left(E_i - \overline{E}\right)^2}} \approx 0.834 \tag{6}$$

from where we obtain the determination index $R^2 > 0.5$, which means that the approximation of the function can be used to represent the law of distribution of errors depending on the range of the receiver. The average error of approximation left

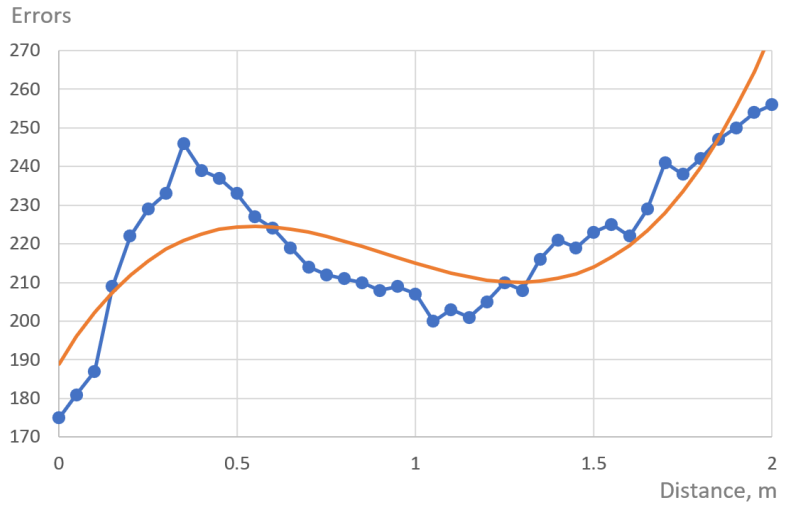

Figure 3: Measurement results and approximation by a hyperbolic function.

$$\overline{A} = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{E_i - \widehat{E}_i}{E_i}\right| \cdot 100\% \approx 4.1\%. \tag{7}$$

The actual Fisher's criterion was

$$F = \frac{R^2}{1-R^2} \cdot \frac{k_2}{k_1} \approx 28.1. \tag{8}$$

where the indices are calculated from the sampling parameters $k_1 = 3$ and $k_2 = n - k_1 - 1 = 37$. The value of Fisher's criterion indicates that for the sample, we can only observe the type of process, but it does not correspond to the approximated function.

Also when calculating the actual Durbin-Watson criterion we get:

$$d = \frac{\sum_{i=1}^{n}\left(\varepsilon_i - \varepsilon_{i-1}\right)^2}{\sum_{i=1}^{n}\varepsilon_i^2} \approx 0.27 < 2, \tag{9}$$

where ε_i is the deviation from the mean error rate. The criterion value indicates a positive autocorrelation, which indirectly confirms the conclusion about the qualitative assessment of the investigated process.

10. Conclusions

The work involved selecting the tools, preparing software for the experiment, and conducting the experiment itself in a distance range of up to 2 meters. Since the near zone for a frequency of 2.402 GHz is about 1 meter, the selected distance covers the transition from the near to the far zone of the antenna. When approximating the measurement results, it was shown that in the center of the near zone, the number of errors increases slightly, and decreases when moving to the far zone, which indicates the formation of an electromagnetic wave front at the junction of zones. The calculated coefficients confirm the reliability of the approximation results.

Further studies are planned to be carried out to check the rate of signal loss at significant distances and compare the results obtained by different packet sniffers.

Declaration on Generative AI

The authors have not employed any Generative AI tools.

References

[1] Y. Takagi, T. Kushida, Accurate indoor positioning based on beacon weighting using RSSI, 2022. doi:10.1007/978-3-031-20936-9_2.

[2] A. Mackey, et al., Improving BLE beacon proximity estimation accuracy through Bayesian filtering, IEEE Internet Things J. 7 (2020). doi:10.1109/JIOT.2020.2965583.

[3] R. Faragher, R. Harle, Location fingerprinting with Bluetooth Low Energy beacons, IEEE J. Sel. Areas Commun. 33 (2015). doi:10.1109/JSAC.2015.2430281.

[4] K. Yuzawa, T. Fujii, Indoor positioning using BLE beacons and user equipments in factory environment, in: 2024 15th Int. Conf. on Ubiquitous and Future Networks (ICUFN), 2024. doi:10.1109/ICUFN61752.2024.10625482.

[5] M. Sie, C. Kuo, Indoor location estimation using BLE beacon with multiple transmission power levels, in: 2017 IEEE Int. Conf. on Consumer Electronics—Taiwan (ICCE-TW), 2017. doi:10.1109/ICCE-CHINA.2017.7991126.

[6] O. Ivashchuk, et al., A configuration analysis of Ukrainian flight routes network, in: Experience of Designing and Application of CAD Systems in Microelectronics (CADSM), 2021, pp. 6–10. doi:10.1109/CADSM52681.2021.9385263.

[7] V. Sokolov, P. Skladannyi, V. Astapenya, Wi-Fi interference resistance to jamming attack, in: IEEE 5th Int. Conf. on Advanced Information and Communication Technologies, 2023, pp. 1–4. doi:10.1109/AICT61584.2023.10452687.

[8] V. Sokolov, P. Skladannyi, N. Mazur, Wi-Fi repeater influence on wireless access, in: IEEE 5th Int. Conf. on Advanced Information and Communication Technologies, 2023, pp. 33–36. doi:10.1109/AICT61584.2023.10452421.

[9] V. Sokolov, P. Skladannyi, A. Platonenko, Jump-stay jamming attack on Wi-Fi systems, in: IEEE 18th Int. Conf. on Computer Science and Information Technologies, 2023, pp. 1–5. doi:10.1109/CSIT61576.2023.10324031.

[10] V. Sokolov, P. Skladannyi, V. Astapenya, Bluetooth Low-Energy beacon resistance to jamming attack, in: IEEE 13th Int. Conf. on Electronics and Information Technologies, 2023, pp. 270–274. doi:10.1109/ELIT61488.2023.10310815.

[11] V. Sokolov, P. Skladannyi, N. Korshun, ZigBee network resistance to jamming attacks, in: IEEE 6th Int. Conf. on Information and Telecommunication Technologies and Radio Electronics, 2023, pp. 161–165. doi:10.1109/UkrMiCo61577.2023.10380360.

[12] G. de Blasio, et al., Beacon-related parameters of Bluetooth Low Energy: Development of a semi-automatic system to study their impact on indoor positioning systems, Sensors 19 (2019). doi:10.3390/s19143087.

[13] F. Suzanayanti, M. Alaydrus, Optimization BLE power beacon for indoor locations static smart device with gaussian filter, J. Telekom. Komput. (2021). doi:10.22441/INCOMTECH.V11I1.9811.

[14] M. Castillo-Cara, et al., An empirical study of the transmission power setting for Bluetooth-based indoor localization mechanisms, Sensors 17 (2017). doi:10.3390/s17061318.

[15] R. Ramirez, et al., A practice of BLE RSSI measurement for indoor positioning, Sensors 21 (2021). doi:10.3390/s21155181.

[16] S. De, et al., Finding by counting: A probabilistic packet count model for indoor localization in BLE environments, in: 11th Workshop on Wireless Network Testbeds, Experimental evaluation and CHaracterization, 2017. doi:10.1145/3131473.3131482.

[17] A. Strzoda, K. Grochla, K. Połys, Variability of BLE advertisement packets received signal strength and delivery probability in the presence of interferences, in: 12th ACM Int. Symposium on Design and Analysis of Intelligent Vehicular Networks and Applications, 2022. doi:10.1145/3551662.3560933.

[18] S. De, R. Kravets, Finding by counting: A packet count based indoor localization technique using BLE sensors, 2018.

[19] I. Ostroumov, et al., Relative navigation for vehicle formation movement, in: IEEE 3rd KhPI Week on Advanced Technology, 2022, pp. 1–4. doi:10.1109/KhPIWeek57572.2022.9916414.

[20] F. Yanovsky, Inferring microstructure and turbulence properties in rain through observations and simulations of signal spectra measured with doppler-polarimetric radars, in: NATO Science for Peace and Security Series C: Environmental Security, volume 117, 2011, pp. 501–542. doi:10.1007/978-94-007-1636-0_19.
[21] A. Morgan, G. Humaid, A. Moustafa, Using Bluetooth Low Energy for positioning systems within overcrowded environments: A real in-field evaluation, Comput. Electr. Eng. 93 (2021). doi:10.1016/j.compeleceng.2021.107314.
[22] N. S. Kuzmenko, I. V. Ostroumov, K. Marais, An accuracy and availability estimation of aircraft positioning by navigational aids, in: IEEE 5th International Conference on Methods and Systems of Navigation and Motion Control (MSNMC), 2018, pp. 36–40. doi:10.1109/MSNMC.2018.8576276.
[23] Espressif Systems, Chip series comparison, 2022. URL: https://docs.espressif.com/projects/esp-idf/en/v5.0/esp32s3/hw-reference/chip-series-comparison.html.
[24] Espressif Systems, Product comparison, 2025. URL: https://products.espressif.com/#/product-comparison?type=SoC&names=ESP32-C6,ESP32-H2FH4S.
[25] R. Powell, Nimble-arduino, 2025. URL: https://github.com/h2zero/NimBLE-Arduino.
[26] Olikraus, u8g2, 2025. URL: https://github.com/olikraus/U8g2.
[27] OffSec Services Limited, Packages and binaries: Bluetooth, 2025. URL: https://www.kali.org/tools/bluez/.
[28] Nordic Semiconductor, nrf sniffer for bluetooth le. development tool, 2025. URL: https://www.nordicsemi.com/Products/Development-tools/nrf-sniffer-for-bluetooth-le.
[29] Texas Instruments, Smartrf packet sniffer. user's manual, swru187g, 2014. URL: https://www.ti.com/lit/ug/swru187g/swru187g.pdf.
[30] Great Scott Gadgets, Ubertooth one, 2023. URL: https://greatscottgadgets.com/ubertoothone/.