A Consumer Product of Wi-Fi Tracker System using RSSI-based Distance for Indoor Crowd Monitoring

Syifaul Fuada[1], Trio Adiono[2], Prasetiyo[3], Harthian Widhanto[4], Shorful Islam[5], Tri Chandra Pamungkas[6]
Program Studi Sistem Telekomunikasi, Universitas Pendidikan Indonesia, Bandung, Indonesia[1]
Faculty of ITEE-Centre for Wireless Communications (CWC), University of Oulu, Oulu, Finland[1]
School of Electrical Engineering and Informatics, Institut Teknologi Bandung, Bandung, Indonesia[2]
School of Electrical Engineering, Korea Advanced Institute of Science and Technology, Daejeon, South Korea[3]
Stream Intelligence Ltd., London, United Kingdom[4, 5, 6]

***Abstract*—This study aims to design and develop Wi-Fi tracker system that utilizes RSSI-based distance parameters for crowd-monitoring applications in indoor settings. The system consists of three main components, namely 1) an embedded node that runs on Raspberry-pi Zero W, 2) a real-time localization algorithm, and 3) a server system with an online dashboard. The embedded node scans and collects relevant information from Wi-Fi-connected smartphones, such as MAC data, RSSI, timestamps, etc. These data are then transmitted to the server system, where the localization algorithm passively determines the location of devices as long as Wi-Fi is enabled. The mentioned devices are smartphones, tablets, laptops, while the algorithm used is a Non-Linear System with Lavenberg–Marquart and Unscented Kalman Filter (UKF). The server and online dashboard (web-based application) have three functions, including displaying and recording device localization results, setting parameters, and visualizing analyzed data. The node hardware was designed for minimum size and portability, resulting in a consumer electronics product outlook. The system demonstration in this study was conducted to validate its functionality and performance.**

***Keywords*—Wi-Fi tracker system; RSSI-based distance; crowd monitoring; Unscented Kalman Filter; indoor**

I. INTRODUCTION

Wi-Fi tracking technology enables the detection of Wi-Fi signals emitted by individuals or objects, allowing for tracking their locations. This technology finds applications in indoor environments such as laboratories and nursing homes, where it can be beneficial for monitoring the health of the residents. [1]. Two primary types of Wi-Fi tracking system exist, including active and passive. Active tracking relies on individuals carrying a dedicated tracking device at all times, which can be inconvenient due to the constant need for device presence [1]. In contrast, passive Wi-Fi tracking does not require individuals to carry a tracking device but can only track the behavior of detected objects within the operational area [2].

Wi-Fi tracker system has garnered significant attention and found extensive applications in many cases. It has been employed for diverse purposes, including monitoring bus passenger volume, public transit ridership, human movement analytics, tourism mobility, tracking pets or wildlife, analyzing visitor behavior, tracking indoor pedestrian movement, capturing shopper activities, detecting client positions, monitoring patients, and tracking attendance of officers, students, and teachers. [3]–[14]. System has also been successfully applied in various settings, such as museums, buildings, malls, campuses, schools, offices, airport areas, bus stations, theaters, etc. Wi-Fi tracking technology allows for the detection and tracking of smartphone locations using Wi-Fi probes [9], [15], [16]. This capability enables the analysis of device distributions, particularly in outdoor and indoor areas, providing insights into crowd levels and density[13], [17], including crowdedness levels [4].

The design and development of Wi-Fi tracking system involved conducting intensive preliminary study, which focused on studying the behavior of smartphones during connection. The evaluation was also carried out to determine the suitability of RSSI method for the tracking system [18]. The conducted experiments showed that system could collect important information such as RSSI (Wi-Fi signal strength) and MAC address of the smartphone. These two data are the required information that needs to be processed further. When Wi-Fi is enabled on a smartphone, it broadcasts packet request data containing the protocol, channel, and MAC address through Wi-Fi signal [18]. The initial step in implementing Wi-Fi tracking system using RSSI method involved sniffing these data packets and measuring the corresponding RSSI level at each node [19].

The use of RSSI method in Wi-Fi tracker system necessitates consideration of several factors. Firstly, RSSI value can exhibit instability [16]. Secondly, the interval time for broadcasting packet request data is variable and dependent on the state of the smartphones[20]. Thirdly, each smartphone emits a distinct initial power level [19]. Therefore, to address these challenges, a localization algorithm was developed to determine the location of the smartphone based on information gathered from all nodes. Study was conducted on digital filter design to enhance detection accuracy [21], [22]. Two candidate algorithms, namely the Intersection density algorithm and the Non-linear Least Square (NLS) algorithm, were evaluated in terms of their performance. The NLS algorithm was selected as the localization algorithm considering the issue of unstable RSSI values [22]. Unscented Kalman filter (UKF) algorithm was employed to reduce the noise value as well as enhance the detection accuracy. This

algorithm reduces noise and contributes to a more stable RSSI value, improving prediction accuracy [21], [23].

The ongoing study aims to develop Wi-Fi tracking system for indoor crowd-monitoring applications. This system enables the estimation and determination of the number of people in a room. It also acts as portable sensor because the nodes receive RSSI and capture Wi-Fi power emitted by Wi-Fi devices. The captured data is then transmitted using the Message Queue Telemetry Transport (MQTT) protocol and processed on the server. During the capturing process, the nodes collect data for duration of 5 seconds and store it locally in a circular file with up to five files stored in the active directory. The processor reads the third file in the active directory, parses and encapsulates the data in JSON format, and sends it to an available server.

According to the findings in study [18], when smartphones activate their Wi-Fi function and connect to an available Wi-Fi hotspot, they transmit a unique wireless signature known as MAC address. Additionally, the timestamp and RSSI values of smartphones vary depending on their types, models, and vendors. These data can be utilized as valuable references for constructing Wi-Fi tracking system. Fig. 1 shows the proposed architecture of Wi-Fi tracker system, comprising two main subsystems, including the node and server subsystem. The node subsystem is responsible for sniffing smartphone data transmitted at specific locations. Installing more than three nodes within the target area is advisable to ensure comprehensive coverage. The collected data from the nodes is then sent to the server for further processing. The server subsystem performs data computation using suitable algorithms and conducts analysis. The resulting data are summarized and presented on an online dashboard for visualization purposes.

Several steps need to be followed to operate system. Firstly, nodes are placed at desired locations within the target area. Subsequently, when smartphones in proximity have their Wi-Fi switched to on/active, they will broadcast packet request data to the surrounding environment. Alternatively, smartphones can connect to nearby Access Points (AP), and system remains active as long as broadcast packet request data is continuously processed. Each node within the location then captures and sniffs smartphone packet data. To perform this function, the nodes must be configured in monitoring mode, and once captured, the nodes transmit the data to the server. However, before sending the data, it is encrypted using Transport Layer Security/Secure Sockets Layer (TLS/SSL) with 1024-RSA encryption to ensure compliance with data security requirements. This step is crucial as the data transmitted during communication are susceptible to vulnerabilities [[24], [25]. Subsequently, the nodes establish a connection with Wi-Fi tracker AP and the collected data is finally transmitted to the server using MQTT protocol.

The server plays a crucial role in system by collecting data from all nodes. It achieves this by subscribing to MQTT broker based on the designed topic. Upon receiving the data, the server decrypts the packet data and organizes it based on MAC address information. This process results in a set of MAC address data along with their corresponding RSSI values from each node. By utilizing this data set, the server proceeds to compute the location of each device using two algorithms, including UKF and the NLS algorithm. The raw data, as well as the processed data containing the smartphones and their respective locations, are stored in the database. MongoDB was selected as the database system for Wi-Fi tracking system, while an online dashboard was developed to provide a user-friendly interface for accessing and visualizing the data. This dashboard is a web application running on the server, which also serves as a platform for basic system configuration and presents the results of the analyzed data.

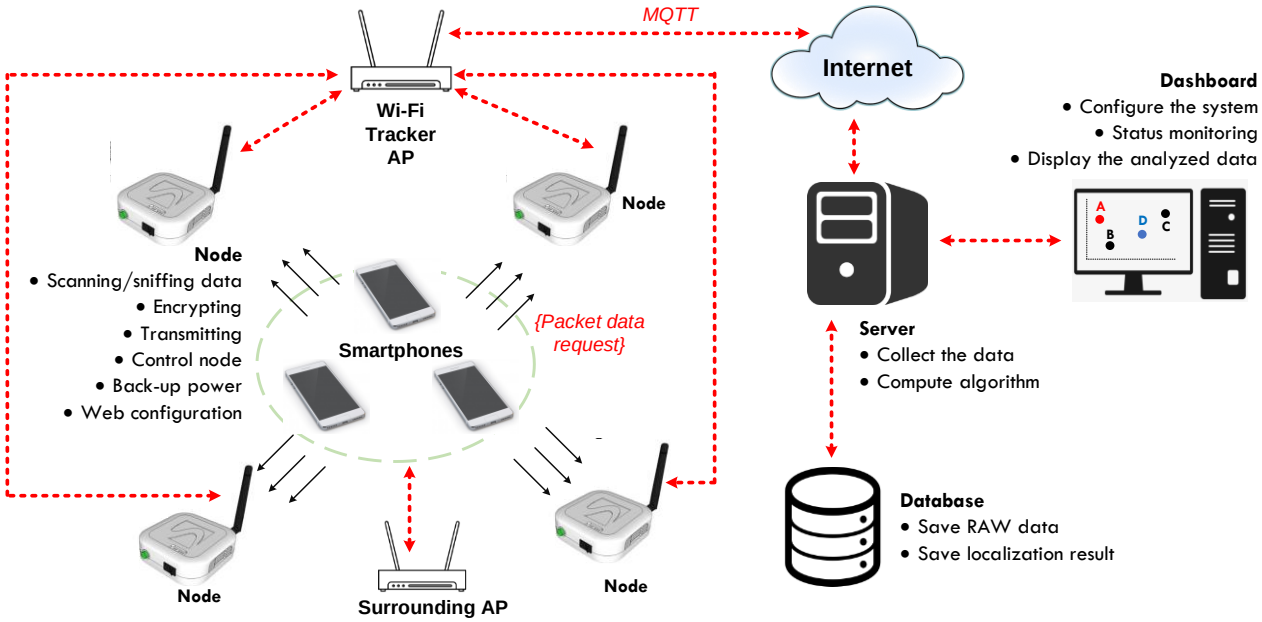


Fig. 1. The architecture of the proposed Wi-Fi tracker system.

This study consists of six sections, namely introduction, system overview, embedded node (hardware and software), localization algorithm, server system, and online dashboard. The final section encompasses a functional test conducted through system demonstration. As the focus of this study is primarily on describing the structure of system, the proposed results are limited to system demo. Wi-Fi tracker system presented is a result of extensive study in the field of Wi-Fi tracking, and it has been successfully implemented on real hardware. The viability of the proposed concept has been carefully validated. Therefore, future study efforts should aim to evaluate the proposed system comprehensively. This evaluation should include assessing accuracy in relation to the number of smartphone entities, power consumption analysis, measuring actual access time with varying user numbers, testing against multiple attack scenarios to assess vulnerability, and conducting other relevant analyses.

II. Related Works

Wi-Fi Tracker system typically comprises two primary components, including the node and the server system. The node system acts as a sensor that scans smartphone information. The scan results obtained from the node system are then transmitted to the server for further processing using various algorithms. The server system performs computations using the collected data from multiple nodes to predict the location of the smartphones. Additionally, a dashboard, which can be a web-based or smartphone-based application, may be incorporated into system to configure the algorithms and display the results.

Numerous frameworks have been introduced to facilitate the prototyping of Wi-Fi tracker system. These include CrowdProbe [26], mD-Track [27], Widar [28], IndoTrack [29], Beanstalk [30], SenseFlow [31], ARIEL [32], Probr [33], MOBYWIT [34], etc. However, many of these studies lack comprehensive discussions on all aspects of Wi-Fi tracking system, including the hardware, software, algorithms, server system, web-based application, and system demonstration. To address this gap, the focus of this study is to provide a comprehensive design of Wi-Fi tracking system that offers a clearer understanding. The study aims to delve into the different components of system in greater detail compared to previous studies [[18-26]. It is expected to make it easier for readers to understand the whole system.

In comparison to previous studies [3]–[14], [26]–[33], this Wi-Fi tracking system offers additional components such as a server system and a user-friendly web-based application. These additions enable real-time monitoring of passive smartphone positions based on RSSI values. The online dashboard included in system provides various features, including a heat map, time reports, and user identification based on MAC data. The server system plays a critical role in gathering all the scanned information from the nodes. It performs the localization algorithm and employs UKF for computation. The collected raw data, as well as the computation results, are stored in the database. Additionally, the database is used for a Graphical User Interface (GUI) to configure system, display the results, and present the analyzed data. Wi-Fi tracking system has been designed and packaged as a consumer electronics product. This aspect has a positive implication, as it ensures that system is well-suited for market deployment in the future [35].

III. Methods

A. Embedded Node

The node system of Wi-Fi tracking system is developed on the Linux platform. Specifically, the current version of the node system is designed to run on Raspberry Pi Zero W, with Raspbian (Debian 9) as its operating system. A web application is created using the NodeJS framework to simplify system configuration process. The services of the node system are managed using pm2, allowing users to configure system via a local IP browser (http://10.42.0.1). Additionally, the node system can be controlled through a dashboard deployed on the server. Users can use the dashboard reboot button to reboot a specific node. The communication between the dashboard and the node system is facilitated through MQTT protocol. The node subscribes to the command topic while the dashboard publishes commands to the same topic. This enables seamless command exchange between the dashboard and the node system. The node system can be further divided into two main components, namely hardware, and software, which will be explained in detail as follows:

1) Hardware part (Raspberry Pi): This embedded node hardware is designed to carry out the following tasks with optimal functionality: a) perform the main function, which is scanning, encrypting, storing, and sending the data, b) auto-connect to the network, c) provide node configuration system, d) supply backup power system, and e) be controlled remotely by the user. After the requirement has been defined carefully, tracker node hardware specification is decided, as shown in Table I.

TABLE I. Hardware Specifications

No.	Specification	Description
1	Power Input	Main DC 9V 1.5 A Back-up power: LI-ION bat 3.8V/ 4500mAh
2	Operating Temperature	Normal operation: -10°C ~ +60°C Storage operation: -20°C ~ +70°C
3	Feature	Remotely controlled (By internet and GSM module), it has a configuration system interface (web apps), Power back-up (up to 8 hours), LED indicator
4	Dimension	10 10 × 5 cm
5	Portability & Lightweight	11 Yes

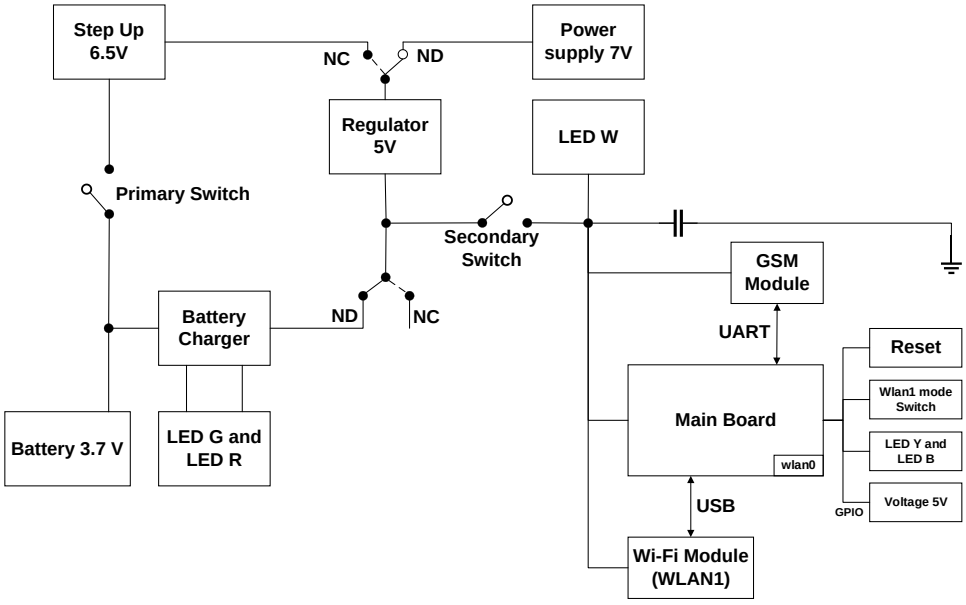


Fig. 2. Block diagram of node hardware.

The node system utilizes the Raspberry Pi Zero W, known as the mainboard, a compact and powerful development board running on the Linux platform. This mainboard provides excellent processing capability despite its small size. The functionality of the node is enhanced by connecting it to a complement board that integrates various components. These components include power management circuits, LED indicators, physics controls, GSM modules, and external antenna connectors. GSM module communicates with the mainboard using the UART Serial protocol, allowing efficient data transfer and communication. An external antenna is utilized, connected via a USB type B connector, and configured as wlan1 while the wlan0 is embedded in the mainboard.

Fig. 2 shows the structure of the node hardware, which is designed based on the requirements and functionality of the node. The main board is responsible for scanning/sniffing the packet request data through wlan1 and transmitting the data to the server through wlan0. The node hardware consists of two main power sources, including the main supply adaptor with a voltage of 7V and a backup supply in the form of a 3.7V battery. By default, when the main supply is available, the module draws power from it while the battery charge module charges the battery. Two LED indicators are provided, namely LED R and LED G indicating that the battery is charging and fully charged, respectively. When the main power is off, the relay switches from normally closed (NC) to normally open (NO), allowing the power supply to be sourced from the backup battery. The 5V regulator is necessary to ensure that the mainboard operates at its designated operating point. Additionally, a step-up module is used to ensure that the 3.7V battery voltage is appropriately boosted to drive the 5V regulator.

A capacitor bank is added to ensure a stable power supply during the switching process of the relay. It provides temporary power when the relay changes its state, and the main power becomes temporarily unavailable. Additionally, a switch is incorporated to allow the user to manually turn on and off the node, serving as both a primary and secondary switch. GSM module is responsible for receiving Short Message Service (SMS) commands from the user. The data is sent to the mainboard for processing when an SMS is received. The mainboard verifies the validity of the command, and when it is valid, the node will be restarted. There are two options for GSM module, including SIM800L version 1 and SIM800L version 2. SIM800L version 1 operates at an operating point between 3.4V and 4.4V, while SIM800L version 2 operates at 5V. When SIM800L version 1 is chosen, an additional regulator is required to provide the correct voltage. However, since the hardware only provides a 5V power line, SIM800L version 2 can be used without needing an additional regulator to supply 4V. This means that SIM800L version 2 is selected for this system configuration.

Two LED indicators (LED B and LED Y) are connected to GPIO pins on the mainboard to provide visual indicators. These LEDs can be programmed to function according to specific requirements, such as light up when the node sniffs or sends packets. A reset button is also included to allow for external hard resetting of the mainboard. System also features a wlan1 mode switch, which allows for changing the mode of wlan1 between AP mode and monitoring mode. AP mode is used when the user wants to access and configure system node through web browsing. On the other hand, monitoring mode is utilized when the node is actively sniffing packet data.

The PCBs are designed with a focus on minimalism, ensuring that the electronic components can be integrated efficiently without adding unnecessary bulk to the case of the node. This design approach aligns with market demands for electronic devices that are lightweight, portable, modern, elegant, and simple. The hardware implementation of tracker node, based on the design shown in Fig. 2, is illustrated in Fig. 3. The casing of the node is made from Polylactic Acid (PLA), the same material used in other consumer-based product developed by Pusat Mikroelektronika ITB. Examples include

IR-based remote controllers [36], humidity and temperature sensor nodes [37], electronic transaction devices [38], and generic power sockets [39]. This material is suitable for a proper case because it is light, low-cost, solid, and resistant to dust particles & water.

Fig. 3. Photographs of trackernode hardware.

2) Software part (Program node): Five programs running in the node include a) Main program (i.e., Sniffing, Encrypting, and Sending data), b) Sending node status, c) GSM module, d) Web control, and e) Web configuration. The description of each program is explained as follows:

a) Sniff the packet data: Fig. 4 illustrates the workflow of the program used to control Wi-Fi tracker and gather the required packet request data. It is crucial to consider the frequency at which devices broadcast packet request data as it directly impacts the accuracy and reliability of system. The more frequently devices send probe requests, the better the results can be achieved. During observations, several factors were identified that influence the frequency of packet requests, including the type of smartphones and the individual states. Different smartphone models may have varying time intervals for sending out packet requests, hence, to capture and analyze Wi-Fi data, Tshark, a packet sniffing tool, is utilized. The question of how Tshark analyzed the data was demonstrated in [18].

Smartphones with different chipsets and wireless adapters exhibit variations in their scanning behavior, which can be observed through the differences in scanning results. Several smartphones have a feature that enables power-saving mode, leading to slower probe request broadcasts than smartphones without this feature. Based on preliminary observations of various smartphone types, probe requests are sent at different intervals. The fastest observed interval between probe requests is approximately 3 seconds, while the slowest interval is around 60 seconds. The frequency of probe requests is higher in active scanning mode and lower during other smartphone activities such as web browsing, files downloading, or sleep mode.

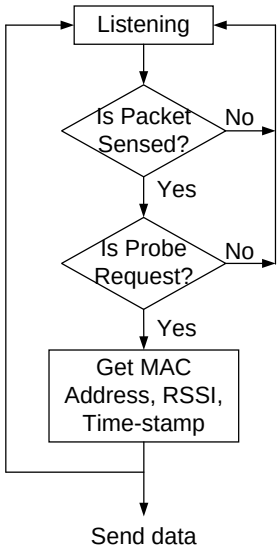


Fig. 4. Flowchart of scanning data for smartphone connected to Wi-Fi.

When considering smartphones, two scanning methods are used for active Wi-Fi AP as defined by [13], namely active and passive scanning. In passive scanning, the smartphone receiver is turned on to listen to each channel for the beacon. This scan is completely passive, and nothing is transmitted from the smartphone. While for active scanning, the smartphone will broadcast the packet on each channel. Probe requests are sent on the currently tuned channel to discover existing AP. The main focus of this system design is tracking the location of smartphones rather than AP. By setting the node to monitoring mode, no packet data is sent, thereby requiring smartphones to be in active scanning mode.

b) Encryption: As discussed in the previous chapter, security, and privacy issues are important concerns in the application due to the utilization of user information. MAC address emitted by smartphones has a considerably unique nature, allowing potential tracking of user whereabouts [11]. For example, unauthorized individuals can steal information stored in MAC address and use it to identify peoples' location and duration of stay in a building [24]. This raises significant safety and privacy concerns [25]. In this study, TLS/SSL is employed for MQTT communication to ensure secure transmission due to its ability to provide a secure communication channel between the client and server. TLS operates at the transport layer, enabling encryption without the need for application-layer encryption implementation. TLS/SSL consists of two main components, including securing the connection between the client and server using Certificate-Based Key Exchange and securing the sent messages using RSA encryption. The workflow of TLS/SSL in this system is shown in Fig. 5.

A certificate-based key exchange mechanism is employed to ensure a secure connection. The authorized user provides a certificate to both the server and node, which establish a connection assuming they possess the same certificate. The server generates private and public keys for encryption and decryption of information. The public key is then shared with the node using the certificate. The node, having the matching

certificate, can access and utilize the public key. The information is encrypted using the public key, employing the RSA method with a 1024-bit key in this study. All scanned data in the node, including the node status, will be encrypted. Similarly, the server, possessing the private key, is used to decrypt the encrypted information.

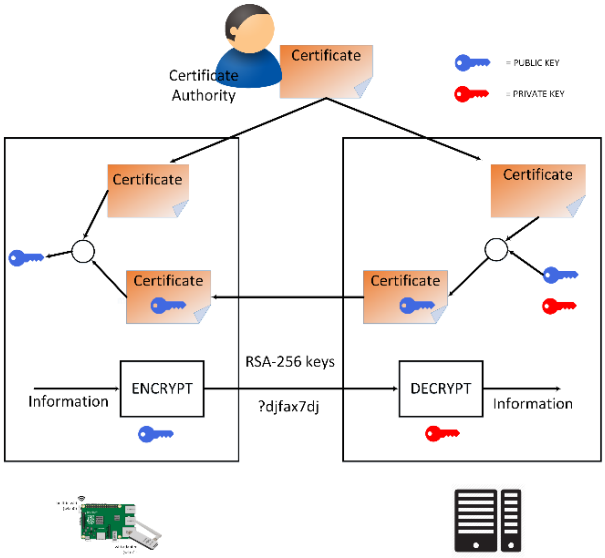


Fig. 5. TLS/SSL block diagram.

c) Sending node status: As described earlier, tracker node sends the captured data to the server using MQTT protocol, a client-server publish/subscribe messaging transport protocol. MQTT is widely known for its lightweight and efficient nature in terms of bandwidth usage. It is a popular choice for IoT platforms due to its low-power consumption and ease of implementation [40]. In system, small amounts of data are transmitted at frequent intervals, as opposed to sending large chunks of data less frequently [41]. MQTT protocol, with its support for the publish/subscribe scheme, was chosen for this purpose. Fig. 6 shows the data transmission process from the node to the server, utilizing the built-in Wi-Fi interface (wlan0) to connect to AP.

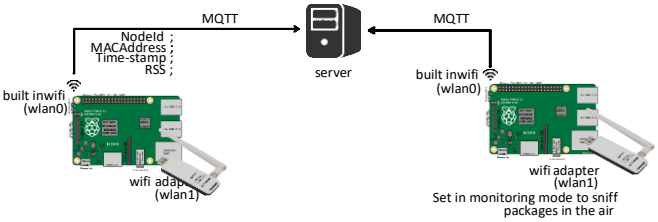


Fig. 6. Sending data from node.

MQTT protocol is utilized to send data in the form of a JSON object. The publisher (node) establishes two distinct topics, namely the main and the status topics. The main topic is responsible for transmitting the primary data whenever the device is scanned, while the status handles the status information of the node, which is sent every 10 seconds. The node status indicates the connectivity status with the server. An example of the main information JSON object, which has a data size of approximately 107 bytes, is provided below.

```
{
    "device_addr": "AA:BB:CC:DD:FF:GG,"
    "rssi_dbm": "-23",
    " time_epoch": "1493901414.065620"
}
```

An example of the node status JSON object, which has a data size of approximately 57 bytes, is provided below:

```
{
    " MAC_addr_node": "B8:27:EB:EB:D8:42"
 }
```

d) GSM module operation: GSM/SMS service enables remote node control through GSM frequency by sending a message to a specific node phone number. The available remote-control actions include allowing the configuration to state/AP and rebooting the node. The commands can be found in a default configuration with only numbers listed under authorized phone numbers capable of controlling the node. GSM module receives an SMS representing the command to restart the node. First, the command is sent to the mainboard using a serial connection. Then the mainboard processes the node and executes the command (reset the board). This functionality proves useful when a specific program, such as the scanning mode, stops working and there is no internet connection available. Table II provides a list of some AT commands, while the port for GSM module is depicted in Fig. 7. The SIM800L module is a widely used GSM/GPRS module for serial communication. It is commonly employed in remote control projects, allowing smartphones with a Microsim-type SIM card to send messages for control purposes. This module utilizes the TTL serial port and features an LED indicator. The LED blinks slowly when the module is connected to a GSM network and blinks rapidly when there is no signal. Although the module provides 13 ports, only a few of them are utilized in this study. GSM hardware is initially connected to the Raspberry Pi for development purposes, as shown in Fig. 8. The experiment involved several components, including the Raspberry Pi, a computer running Python, the SIM800 module, and various jumpers. The connection was carefully made, considering the functions of each port (especially RX, TX, GND, and VCC) to ensure the program was successfully downloaded.

The flowchart shown in Fig. 9 illustrates the control process of GSM module using the mainboard (Raspberry Pi). The main board is responsible for four tasks, including setting up GSM module to SMS mode, extracting commands, making the decision to require a restart, and executing the reboot script. On the other hand, GSM module has four tasks, including listening to incoming SMS, deciding on incoming SMS, reading the SMS, and sending messages to the mainboard. UART communication is used between the two components. In the initial step, GSM module is set up in SMS mode by sending a sequence of AT commands from the mainboard to the module. When there are no errors, GSM module enters SMS mode and listens for incoming messages. When an SMS is received, GSM module will transmit it to the mainboard via UART. The mainboard will then verify the message and extract the content to determine whether it is a restart command to execute a reboot script to restart the node.

TABLE II. COMMANDS OF THE AT

Command	Description
AT+CMGF=1	Set module to operate at SMS/Text mode
AT+CMEE=1	Check whether SMS mode is supported or not
AT+CNMI	New SMS indication
AT+CMGR	Read an SMS

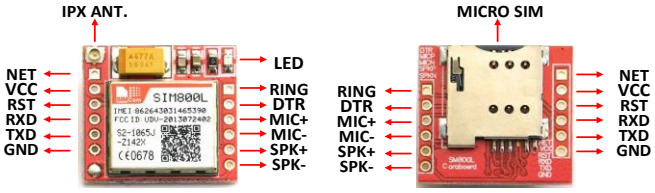


Fig. 7. GSM module SM800L port (top and bottom views).

Fig. 8. GSM hardware setup for development purposes.

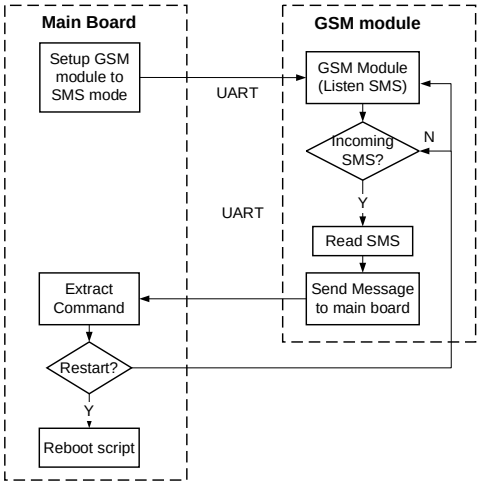


Fig. 9. Flowchart of GSM controller.

During the development process, a problem arose where the flowchart did not work correctly. The possible issue is related to the signal antenna GSM, serial communication (UART), or an incorrect AT command sequence. To resolve this problem, the code and commands were modified using the Python library. Eventually, the node was successfully tested and demonstrated the ability to receive commands from SMS and execute them.

e) Web control system: A GUI is provided in the dashboard, which includes a function to monitor the status of the nodes, whether they are sending data or not. The dashboard offers real-time data presentation with the following features: i) Basic map cartesian and dynamic configuration, the basic map will display the tracked device location in cartesian mode, ii) The trajectory of the smartphone graph, iii) Node status monitoring and iv) Calibration page. The layout can be configured from the map setting container on the right side of the map, as shown in Fig. 10. The devices map menu is used to display the algorithm result in a real-time. The map setting is used to set into desired area/location because there is also a form of MAC address filter to display only specific MAC addresses. It is useful for testing the accuracy of the algorithm purpose.

Furthermore, additional features in the dashboard were implemented, as shown in Fig. 11. These include the node name, node status, and a reset command. The reset command menu is used to reset a specific node. The node name corresponds to MAC address of Wi-Fi module connected to Wi-Fi tracker AP. Each node name is published to a different topic, and the node subscribes to the topic based on its Wi-Fi MAC address (wlan0 interface). This allows the status of each node to be displayed in the dashboard.

In system specification, the remote-control functionality of the node was defined, but the web server lacks knowledge of the address, preventing direct command transmission. MQTT protocol was employed to address this, followed by a publish and subscribe scheme. Instead of directly sending commands to the node, they are published to MQTT broker using a known address. The node then subscribes to the relevant topics to receive and execute the commands, allowing it to restart the board. The illustration of the scheme is presented in Fig. 12. An online dashboard is developed using Node.js, a popular full-stack JavaScript web application framework, to enhance the user experience. The integration with Mongo and front-end development with Angular are also implemented.

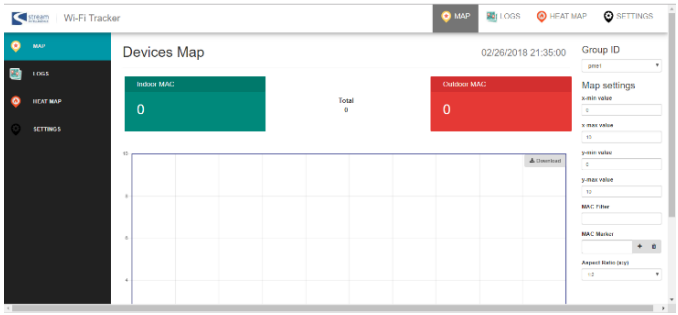


Fig. 10. Basic cartesian map, all detected devices/smartphones after the computation process is displayed in this map. Using MAC address, RSSI, and time stamp data, the location of devices can also be determined.

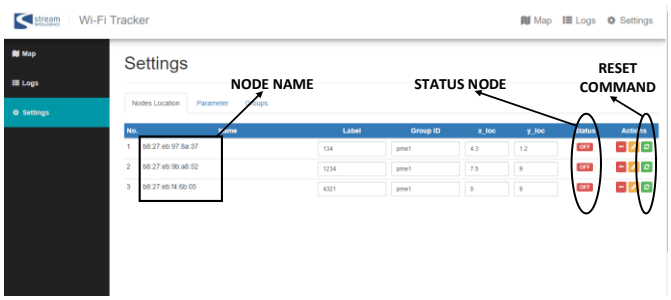


Fig. 11. Appearance of the dashboard for node status and controller.

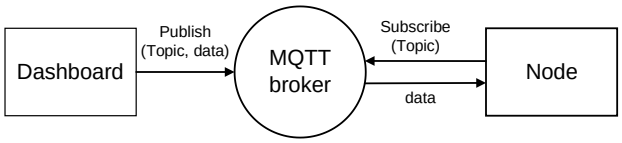


Fig. 12. MQQT broker scheme.

B. Localization Algorithm

This section describes the localization algorithm applied in system. The algorithm predicts the location of devices based on previously obtained information. For this purpose, distance-based method is implemented, which is an RSSI-based localization algorithm. Based on the set of RSSI measurement results, the location of devices is predicted using the estimated distance between tracker nodes and smartphones. In addition, UKF is used to enhance system performance, consisting of three steps, namely modeling, predicting, and correcting, as summarized in Table III. The detailed derivation and simulation under MATLAB are explained in [21], [22]. The calculations are performed in the Apache Strom-based server, and the algorithms are written in Python script. The *NumPy* library is used to perform matrix computation in Python. The code includes a function to initialize the Kalman model used in this work. The prediction and correction steps of UKF are implemented to predict the next state value from the current state and store it in memory.

C. Server System Dashboard

1) Brief description of system structure: The server is deployed on the Debian 9.1 Linux platform and has two main functions, including collecting data from all nodes and estimating the location of detected devices. The data transport from nodes to the server is facilitated through three mechanisms, namely MQTT protocol, a message broker server for MQTT, and Apache Storm. Apache Storm enables distributed real-time computation with the NoSQL platform MongoDB used to store the data. The users can set the configuration and real-time localization from web apps deployed under the Nodejs platform.

The main processor of the server utilizes Apache Storm 1.1.0, which is built on the Java platform and enables distributed computation. Apache Storm is supervised and monitored by the Supervisor and ZooKeeper components. It follows a spout-bolt architecture, where the input data is received by the spout and processed by separate bolts. Python programming language (version 2.7) was chosen for the algorithm implementation due to its extensive library support, facilitating faster development. Apache Storm is implemented using the Streamparse framework, as shown in Fig. 13. This framework allows real-time visualization on the Map dashboard, although global grouping is not currently implemented in the framework.

Serializer Spout is the data flow gate subscribing to certain topics where nodes publish. In this Spout, serializer (message parsing) is conducted from JSON format to a tuple of the device address, timestamp, node ID, and RSSI. Finally, all parsed data are sent to InsertDeviceBolt to be stored in MongoDB as devices collection and BufferMsgBolt.

TABLE III. UKF ALGORITHM FOR THE PROPOSED SYSTEM

Model

$$RSSi\,(t+1) = RSSi(t) - 10\alpha * v(t) * 10^{\frac{(Pinit(t) - RSSi(t))}{10\alpha}} * \Delta t$$

$$v\,(t+1) = v\,(t)$$
$$Pinit\,(t+1) = Pinit(t)$$
$$Z = [\,RSSI\,]$$

$$R = \left[\left(\frac{3.0 * RSSI\ +\ 340}{70}\right)^2\right]$$

$$\boldsymbol{Q} = \boldsymbol{\sigma^2}\begin{bmatrix} \boldsymbol{\Delta t^3/3} & \boldsymbol{\Delta t^2/2} & \boldsymbol{0} \\ \boldsymbol{\Delta t^2/2} & \boldsymbol{\Delta t} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{1} \end{bmatrix}$$

Prediction Step

- Calculate weight, sigma-point:

$$W_0^m = \frac{\lambda}{n+\lambda}$$
$$W_0^c = \frac{\lambda}{n+\lambda} + 1 - \alpha^2 + \beta$$
$$W_i^m, W_i^c = \frac{1}{2(n+\lambda)},\ i = 1, 2, \ldots, 2n$$
$$\lambda = \alpha^2\,(n+k) - n$$
$$\chi_0 = \mu$$

$$\chi_i = \begin{cases} \mu + \left[\sqrt{(n+\lambda)\sum}\right]_i, \text{ for } i = 1, 2, \ldots, n \\ \mu - \left[\sqrt{(n+\lambda)\sum}\right]_{i-n}, \text{ for } i = n+1, \ldots, 2n \end{cases}$$

- Unscented transform

$$\mathcal{Y}_i = f(\chi_i),\ z_i = h(\chi_i)$$

- Transition state:

$$\bar{y} = \sum_{i=0}^{2n} W_i^m\, \mathcal{Y}_i$$

$$\boldsymbol{P_{yy}} = \sum_{\boldsymbol{i=0}}^{\boldsymbol{2n}} \boldsymbol{W_i^c}(\,\bar{\boldsymbol{y}} - \boldsymbol{\mathcal{Y}_i})\,(\,\bar{\boldsymbol{y}} - \boldsymbol{\mathcal{Y}_i})^{\boldsymbol{T}} + \boldsymbol{Q}$$

Correction Step

- Kalman Gain:

$$\bar{z} = \sum_{i=0}^{2n} W_i^m\, z_i$$

$$P_{zz} = \sum_{i=0}^{2n} W_i^c(\,\bar{z} - z_i)\,(\bar{z} - z_i\,)^T + R$$

$$P_{yz} = \sum_{i=0}^{2n} W_i^c(\,\bar{y} - \mathcal{Y}_i)\,(\bar{z} - z_i\,)^T$$

$$K = P_{yz} * {P_{zz}}^{-1}$$

- Correct the prediction:

$$X_{k+1} = \bar{y} + K * (Z_k - \bar{z})$$
$$\boldsymbol{P_{k+1}} = \boldsymbol{P_{yy}} - \boldsymbol{K} * \boldsymbol{P_{zz}} * \boldsymbol{K^{-1}}$$

BufferMsgBolt receives the data from Serializer Spout and temporarily stores it until enough data from at least three nodes with the same device address are collected. In this stage, the data is filtered using UKF and stored as the maximum RSSI value. Afterward, the NLS algorithm is applied to compute the location of the device, and the result is sent to InsertDeviceLocBolt to be stored in the locations of MongoDB collection. System called heartbeat is implemented to maintain a constant stream of data in Storm, which keeps

system alive by checking the incoming stream data within a timeout range. Since the server has only one deployment, the service needs to run for a long time, which is served by a component called heartbeat_keeper.

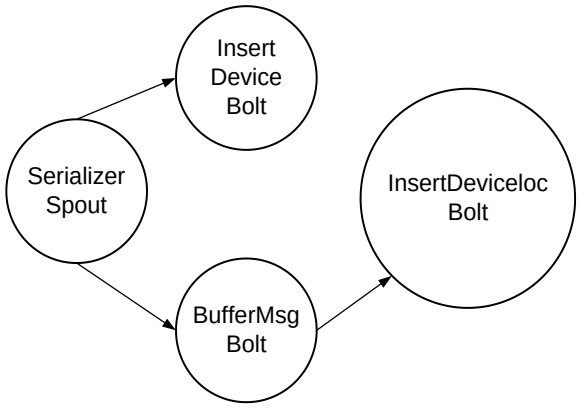


Fig. 13. Storm implementation.

2) Server system: The server system performs data calculations using Apache Storm, a parallel streaming processor. Large companies widely use Apache Storm for processing big data in near-real time. It is an open-source distributed computation system that is free to use, simple, and compatible with various programming languages. In this implementation, Python was chosen as the programming language. The server system architecture is shown in Fig. 14, with a message broker employed to receive data from the nodes. Mosquito, a lightweight and straightforward MQTT broker framework in Python, is used for this purpose.

During the first phase of server system development, Cassandra was chosen as the database solution before MongoDB usage. Cassandra is renowned for its clustered architecture and high availability. However, later in the development process, the decision was made to switch to MongoDB for storing the computation results. MongoDB was preferred due to its document-based database flexibility and ease of querying and inserting data. It is important to note that for optimal performance in a clustered environment, the tables must be structured maturely and well-designed. As a proposed solution, a scenario can be devised where, once all executions and developments are stabilized, a migration back to Cassandra can be considered for further enhancements and improvements in the server system.

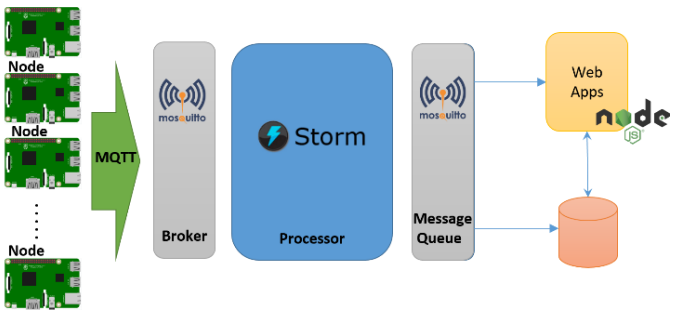


Fig. 14. Server system architecture.

3) MQQT broker: A broker is a message pool that manages where the message is delivered. When a broker receives a message on a particular topic, it broadcasts them to every subscriber who subscribes to the same topic. The broker concept for the proposed system is shown in Fig. 15. It illustrates that node 5 published a "Hi!" message in topic */hello/world/*, and MQTT broker will broadcast to wherever node is subscribing to the same topic */hello/world*. Therefore, nodes 3 and 1 receive the message because they subscribe to *the/hello/** topic. Meanwhile, node 4 will not receive the message due to different topics and node 2. This is because node 2 and 4 serves as a publisher. MQTT security is implemented by exchanging data in an SSL tunnel connection. The client, who acts as publisher or subscriber, needs to have certificates with username and password inputted before sending the message. In the server as a subscribe, the node sends data to *the broker* (publish mode) in a specific topic. Therefore, to receive data from nodes, the subscriber needs to subscribe to the same topic. For example, the server as a subscriber receives data in a topic: com. stream/track/Wi-Fi/device. The payload of this message is presented in the program as follows:

```
{
   "device_addr": "AA:BB:CC:DD:FF:GG", // uniqueID of device
   "rssi_dbm": "-23", // signal strength which read by Node
      " time_epoch": "1493901414.065620", // epoch time of data being sent
   "nodeID" : "11:22:33:44:55:66" // uniqueID of the node which sends the
data
                                                    }
```

In the server as a publisher, the result of computation needs to be pushed immediately to web apps to gain real-time experience. This feature is served by implementing a publisher which broadcasts the process to every web app. An example is in the program below, where the device is unique of the device, x_loc and y_loc denote the x and y positions of the device.

```
{
            "dev_in": 3, // amount of devices in the room
            "dev_out": 10, // amount of devices out of the room
            "devices": [ //: list of devices
                  {"device": "B3:B9","x_loc": -0.19,      "y_loc": -5.77},
                  {"device": "FD:78","x_loc": 12.17,"y_loc": 14.51}
      ]
}
```

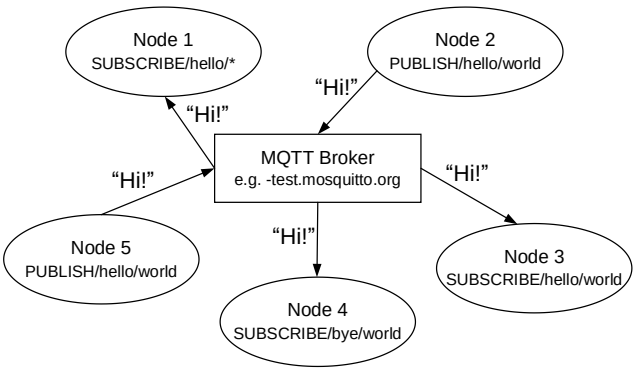


Fig. 15. MQQT broker concept.

4) Processor description in detail: As discussed earlier, Apache Storm is utilized as a processor in the server. The Storm topology consists of Spouts, which receive the data, and Bolts, as micro-computations or processes. The Storm computation is based on micro-computation, so every computation needs to be distributed in Bolts. Every incoming message from *the broker* is serialized in *SerializerSpout* and turned into a tuple. These tuples are then forwarded to the *BufferMsgBolt*, where they are merged with messages from other nodes, which is processed by the algorithm. The computed result is stored in a database through the *RecordStreamBolt* and sent to the dashboard using the *ResultPubBolt*. The workflow of the processor is illustrated in Fig. 16. Meanwhile, in Fig. 17, the flowchart depicting the implementation of UKF algorithm is shown. Firstly, the algorithm parameters are initialized, such as the path-loss exponent, node location, room location, and other relevant parameters. The user configures these parameter values through the web interface, and the database stores these configured values. Finally, when the algorithm is executed, it retrieves the configured values from the database for processing.

The Table State and Table Device serve as temporary memory to store the previous state of UKF and the previous set of smartphone data, respectively. The Table State keeps track of the previous RSSI values and covariance matrix for all existing nodes and smartphones. On the other hand, the Table Device stores the previous information of the devices detected. The long Table State and Table Device were used to determine the time to live (TTL). The program regularly checks the tables and deletes any values that have expired. This approach addresses the issue that the node may not always be able to scan RSSI values from devices, allowing us to utilize the previous values. The TTL helps determine the validity duration for using the stored data.

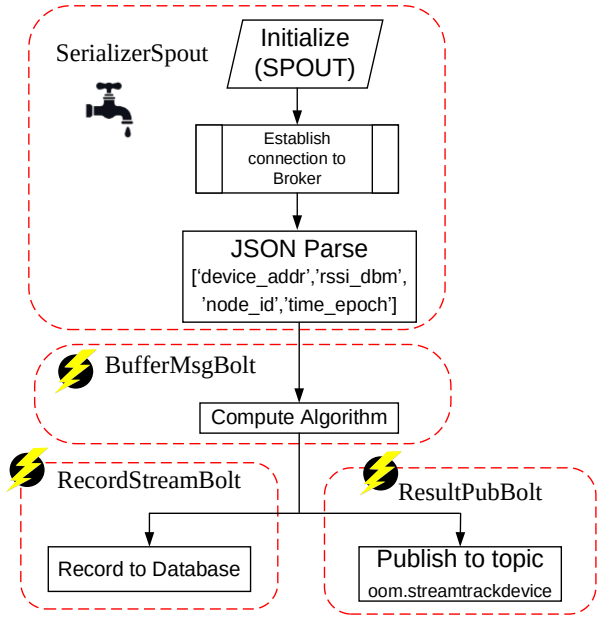


Fig. 16. Storm flowchart.

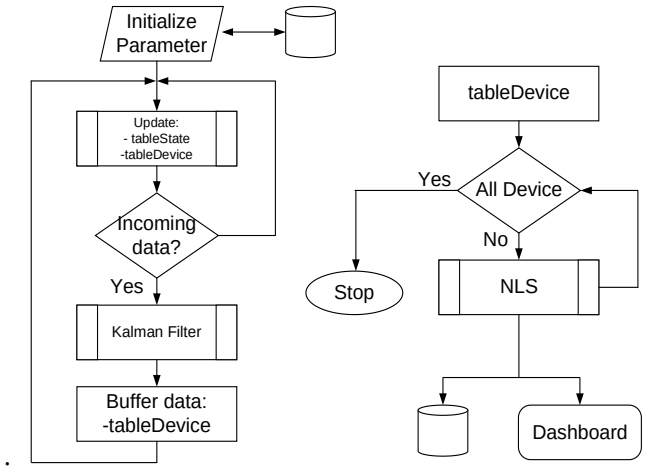


Fig. 17. Workflow of algorithm in general.

The program continuously checks the Table Device and performs the localization algorithm computation for all the available data stored. The resulting localization information is then saved in a database and sent to the dashboard for visualization. The data was modified for the algorithm by employing the NLS with initial power (NLS1). First, it will be used to decide whether the device is inside or not, followed by refining the detection using the NLS with power difference (NLS2). Fig. 18 is the flowchart of the NLS implementation.

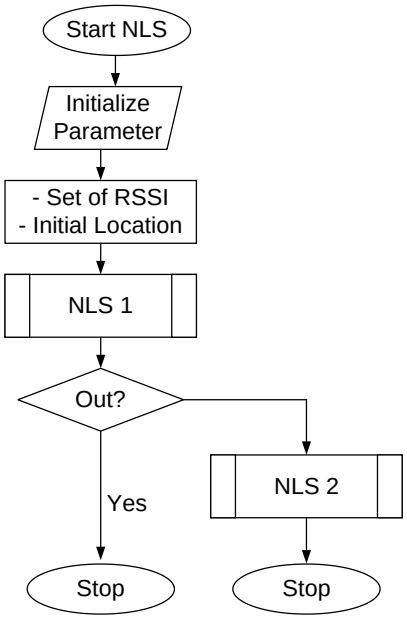


Fig. 18. The NLS algorithm flowchart.

5) Database: Data is stored in various collections with their respective data structures, which are shown in Table IV. Config collection contains display configurations and some parameters used in the algorithm. The Devices collection stores all raw data captured by the nodes. The Collections collection includes all the computation results (documents) of the devices, including their location and status. Finally, the Nodes collection contains information about all registered nodes. Database Structure.

TABLE IV. THE DATABASE STRUCTURE

Structure	Description
Configs	{ "_id" : ObjectId("5979a1c2e70bc538eba1b612"), "groupId" : "pme1", "__v" : 0, "mapConfig" : { "xmin" : 0, "xmax" : 10, "ymin" : 0, "ymax" : 10 }, "roomSize" : { "xmin" : 0, "xmax" : 10, "ymin" : 0, "ymax" : 10 }, "pathloss" : 23 }
Devices	{ "_id" : ObjectId("5976c272e28b3501e5bdef2c"), "timestamp" : "", "rrsi_dbm" : 0, "node_id" : "", "device_id" : SA:SD" }
Locations	{ "_id" : ObjectId("5979f740e28b350ff263b31a"), "status" : 0, "modified" : ISODate("2017-07-27T21:22:56.598Z"), "loc_z" : null, "loc_x" : -1.8633060346972499, "loc_y" : 2.9159220745415255, "device_id" : "DA:A1:19:99:84:A4" }
Nodes	{ "_id" : ObjectId("59788125287c9919e0295b0b"), "nodeId" : "C", "groupId" : "pme1", "locX" : 12, "locY" : 12, "__v" : 0, "modified" : ISODate("2017-07-26T03:37:29.433Z") }

Config fields:

- _id: id of document.
- GroupId: Id of a group of configurations. This will be used to group of nodes.
- mapConfig: configuration for display in cartesian
    - xmin : x min value of map
    - xmax : x max value of map
    - ymin : y min value of map
    - ymax: y max value of map.
- roomSize: the size of the room
    - Pathloss: Pathloss value of room of observation. This is used for calibration.
    - xmin: x min value of room
    - xmax: x max value of room
    - ymin: y min value of room
    - ymax: y max value of room

Devices fields:

- _id: id of document.
- timestamp: timestamp
- rssi_dbm: received signal strength in dbm
- node_id: uniqueID of the node where message comes from

Location fields:

- _id: id of document.
- status: define where the devices regarding room size config. 1 inside, 0 outside.
- modified: last updated record. Format: 2017-07-27T21:22:56.598Z
- loc_z: z location in cartesian. For current development, always null.
- loc_x: x location in cartesian
- loc_y: y location in cartesian.
- device_id: id of device

Nodes fields:

- _id: id of document.
- nodeId: uniqueID of node
- groupID: group Id of node
- loc_z: z location in cartesian. For current development, always null.
- loc_x: x location in cartesian
- loc_y: y location in cartesian.
- Modified: last updated record with the format as follows: 2017-07-27T21:22:56.598Z

6) Data exchange: This subsection provides a more detailed analysis of the data exchange mechanism in the server system. As previously stated, data are exchanged via MQTT protocol from the node to the server, where nodes publish the message to the topic (then denoted as "A"). The *SerializerSpout* subscribes to this topic, allowing it to receive messages from any nodes that send data to topic A. The received data is then processed and inserted into a database using Pymongo. The insertion is carried out by the *InsertDeviceBolt* and *InsertDeviceLocBolt*, which store the computation results in the locations and devices collections.

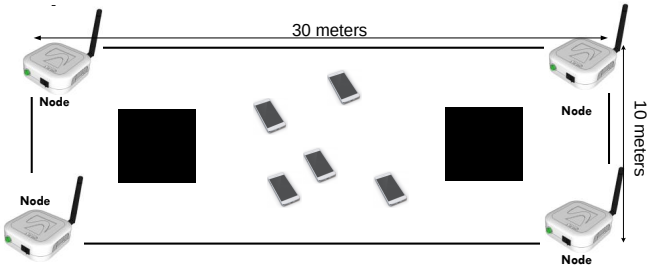
IV. RESULTS AND ANALYSIS

A. Setting at a Glance of System

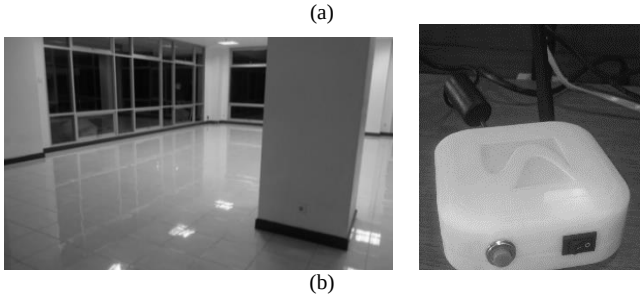
This subsection provides a comprehensive explanation of how to utilize system. The process will be described step-by-step, starting from the initial hardware node setup and leading up to displaying the scanning results on an online dashboard. In order to implement system in the field, different user access levels were defined. System consists of two types of users, namely, admin and dashboard users. Admin users are Wi-Fi tracker developers responsible for setting up and configuring the node and server system, while dashboard users can adjust algorithm parameters and view the results. For indoor localizer environment (Wireless Sensor Network), a minimum of three nodes is required, although using at least four nodes is recommended for more accurate data collection. Distance between nodes should not exceed 20 meters. Positioning the nodes at elevated points, away from obstructions, and ensuring they are connected to Wi-Fi router is advisable. The chosen location used for the live demonstration of system follows the spot-point used in [18]. A glass wall surrounds the room and has a dimension of 30m x 10m, with two static obstacles (concrete pillars) located in the middle. Four tracker nodes are used for this test, with each node placed in a corner, as illustrated in Fig. 19.

Each tracker node is powered by an Adaptor 9V 2A, which serves as the main power and is backed up with a battery of 3.7V 4500mAh, switched with a physical switch relay. The node is configured by enabling AP mode, which is done by pressing the green button for approximately 5 seconds. Subsequently, the node enters AP mode when the configuration indicator turns on (blue light) and the active

indicator turns off (orange light). The configuration process can be carried out by connecting to the default SSID and password and accessing the specific local IP of the node through a web browser. The values that can be configured include the node connection, credentials account (e.g., secret-id, secret-key, and target host), phone number, and authorized phone numbers. While in the configuration state, the node does not capture data. Once the configuration is complete, the submit button should be clicked to apply the new settings, automatically rebooting the node. Finally, the node collects data and sends it to the server.



(a)



(b)

Fig. 19. Demonstration setup: (a) Map of the room for system demonstration; (b) Photographs of the room situation and placed trackernode.

In order to transmit data to the server, the node requires an internet connection. A web client interface has been created based on the NodeJs framework, operating on a node, and this interface supports real-time visualization. Messages are sent from the server (Apache storm - *Buffermsgbolt*) to the NodeJs using MQTT, with the NodeJs acting as a subscriber. A sockets connection is implemented in NodeJs to enable updates whenever new messages arrive. The UI for the angular server and communication with the REST API is implemented using the express framework. To access the website, users need to follow these steps: i) Activate the configuration mode of the node. ii) Open a web browser and navigate to http://10.42.0.1. iii) Enter the required credentials and configurations. iv) Fill out the available forms and initiate a reboot. Detailed instructions for these steps are provided in the subsequent section.

B. *Detail Setting*

The first step involves setting Wi-Fi interface into AP mode by activating a provided button. Instead, the wlan1 interface was utilized, and specifically designated for sniffing packet data, as previously explained. Once wlan1 is in AP mode, other computers or laptops can connect to the established AP. Users can then open a browser and enter the address 192.168.0.1 to access the configuration web interface. The node was restarted after making the necessary changes. The online dashboard will display a setting summary of the node, as shown in Fig. 20(a). Users can click the edit configuration button to view the surrounding APs detected by the node. To send data to the server, the wlan0 interface of the node will establish a connection with one of the detected APs, as illustrated in Fig. 20(b).

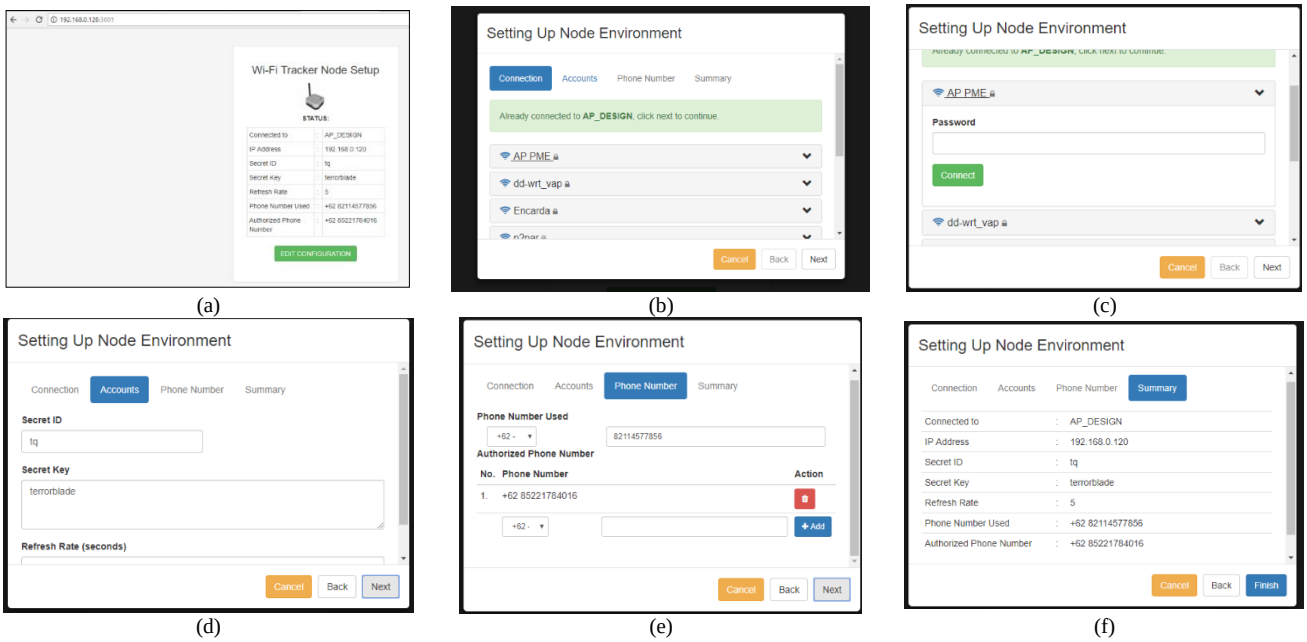


Fig. 20. Demonstration setting flow: (a) Summary of Wi-Fi tracker node setup, Informing the node status, Wi-Fi tracker AP, IP address, Secret ID, Secret key, Refresh rate, Phone number used, and Authorized phone number; (b) Setting up node environment in step I; (c) Select AP and insert password; (d) Setting up node environment in step II: user name and password setting for MQTT; (e) Setting up node environment in step III: GSM number setting used in node and list of authorized number; (f) Summary of the setting.

In the following steps, the user selects the name (SSID) of Wi-Fi network from the available connections list and clicks the connect button, as shown in Fig. 20 (c). Once the device is connected, the user proceeds by selecting the next button and enters the secret ID, secret key, and host obtained from the service provider. It is essential to keep this information private and not share it with others. Clicking the test button prompts a success message when the entered account details are correct. When the user encounters a failed message (Fig. 20d), they are encouraged to contact the provider for assistance. Upon completing a step, they can continue with the optional case by clicking the next button to activate the SMS Service. Next, the user is advised to input their phone number and authorized phone numbers. Finally, the submit button is selected, as shown in Fig. 20(e), to initiate a reboot of tracker node. The setting summary of the node is then displayed (Fig 20f). After clicking the finish button, the online dashboard is presented (Fig. 21). MAC field can be used to filter the displayed devices and leaving it empty will show all detected devices. Devices of interest can be marked in red using MAC marker. The aspect ratio setting determines the ratio of the x-axes and y-axes.

The configuration of each node location is performed in the menu section for node location settings, and all nodes have been placed in specific locations. This setting indicates the location of the node to the algorithm using the x_loc and y_loc coordinates. The menu also displays the status of the node, indicating whether it is available or not, in the status column. Additionally, any unregistered nodes are shown, allowing for the addition of their locations and saving them to the database. The nodes are monitored by being set in the settings menu (Fig. 22), and their status is updated every 1 minute. For system calibration, there is a parameter tab available in the settings. This tab allows for the adjustment of room size, map values, and path-loss of components (Fig. 23). The settings for room size and path-loss exponent can be found under the setting menu within the parameter tab. Path loss calibration is used to represent the surrounding conditions or barriers within the room for the algorithm. The room size parameter determines the observation space and helps consider whether the device is inside or outside the room.

Fig. 21. Node location setup.

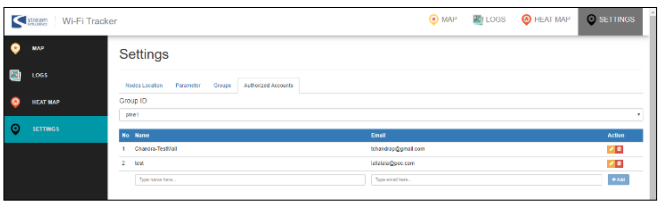


Fig. 22. Node status and setup dashboard.

Fig. 23. Calibration and room setting dashboard.

C. Displaying other Data Analysis

The dashboard includes a real-time device monitoring feature on the map, as well as the percentage of people inside versus outside the observed room to enhance analysis. A marker has been added, which turns red when devices are outside and green when inside. Fig. 24 provides an example of data analysis during the field test, showing the number of smartphones inside the room (green), outside the room (red), and in a specific location (orange) for a day.

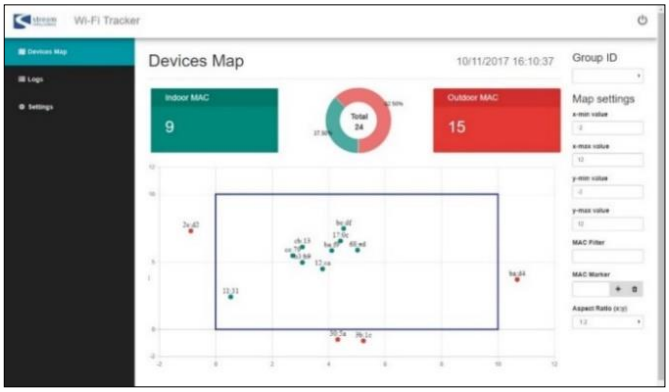


Fig. 24. Devices map on the online dashboard, red represents external devices by time, cyan represents inside, and yellow amount of device in a certain place.

The report can be accessed by daily inquiries on two pages, namely, Logs [34] and Heatmap [33]. The Logs page displays a report based on the access time parameter, while the Heatmap page displays a report based on the parameter of locations. Furthermore, the Logs page allows users to view crowd density in a room over time, with the option to select a specific date for more detailed information. The three categories in this menu are outside, inside, and certain places, with the need to define the latter before accessing system as shown in Fig. 25; it illustrates the volume of people in the test room over time, with data sampled every five seconds. The administrator can download the chart image of Logs (time reports) by clicking the calendar icon and submit button. On the other hand, CSV file containing the raw data of this day can also be downloaded smoothly by clicking CSV icon. The administrator can hide the need data to be analyzed by choosing menu as follows: outside, inside, and ranged devices. The Heatmap feature visually represents the activity density in the observed room throughout the day. By setting the desired date and submitting it, users can obtain the Heatmap, which is centered on the origin point of the room (x = 0, y = 0). Red indicates high activity, while grey represents low activity,

similar to a graph (Fig. 26). The Heatmap can be accessed daily by clicking the Heatmap page, automatically displaying the report of the day. A black square represents the room, and users can adjust the X and Y values to navigate the map. The maximum heat value can also be adjusted to define the intensity of activity. In conclusion, the dashboard enables crowd detection through the visualization of Heatmap data.

Wi-Fi tracking system should offer three important parameters, namely, device classification, user localization, and user profiling [35]. The proposed system is currently capable of defining only the location of the user. In future updates, additional services will be added to gain more benefits, such as user profiling and device classification based on RSSI signal emitted by the smartphones of users. Furthermore, system provides trajectories of devices, which can be accessed through the Logs bar. This feature displays the number of devices (smartphones) present inside or outside the room at specific times. It also allows monitoring of specific places, regardless of whether they are inside or outside the room. The Log bar is utilized to display the data analysis, allowing the user to observe the number of devices inside or outside the room at a particular time.

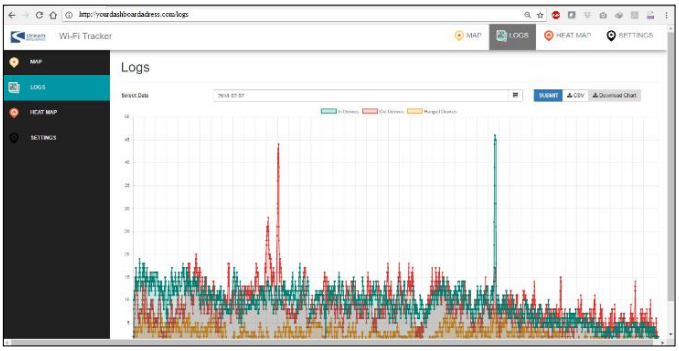


Fig. 25. Time reports.

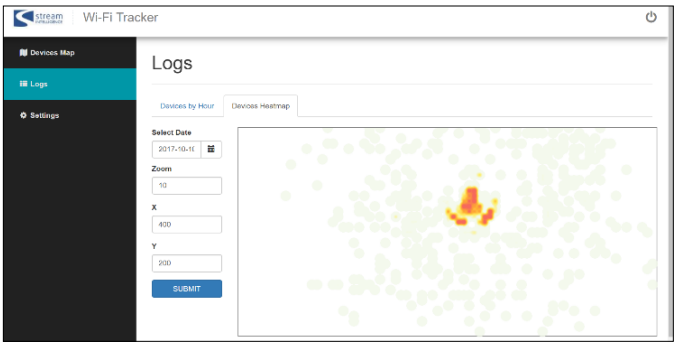


Fig. 26. Heatmap reports.

D. Discussion

The popularity of smartphones as study objects in the computing field is increasing with Wi-Fi tracker system [31]. This is because wireless human sensing system can be carried out accurately and with far coverage. Smartphones can periodically send Wi-Fi request packets which tracker can detect by capturing the passive signal information of the smartphone, including MAC address and RSSI [4], [42]. In comparison to other technologies for human sensing, such as infrared or video thermal cameras, which are less accurate in identifying indoor crowd density, Wi-Fi tracker offers greater accuracy and coverage. These alternative technologies have limitations in terms of a fixed location, angle, and limited range for detecting human presence in various indoor areas. Given the widespread use of smartphones with Wi-Fi connectivity, Wi-Fi tracker system attempts to extract information about human density in a room through smartphones. By utilizing RSSI parameter, system can determine the position of smartphones and their users in a specific area. While the current Wi-Fi tracker system is relatively simple and based on limited-scale trials, it can serve as a solution for monitoring crowd density, which is crucial in social activities. The applications of this system are diverse, such as tracking the number of daily visitors to a shopping center and identifying the most frequently visited stores. Conversely, it can also provide insights into the number of people present and working in a building. This human sensing information helps service providers understand patterns in public spaces within indoor areas. System can be implemented on campuses for various applications, including tracking occupancy in academic spaces such as offices, libraries, laboratories, lecture halls, meeting rooms, faculty rooms, and seminar auditoriums.

The proposed system has the potential to be implemented as a position-tracking and indoor crowd-monitoring solution, which is widely used in modern cell phones. This tracking methodology can be applied to monitor individuals in various indoor locations such as logistics warehouses, hospitals, or airports. Unlike GPS, which has limited accuracy, system utilizing RSSI level from Wi-Fi routers can provide higher accuracy. Wi-Fi routers are already installed in many places, resulting in a lower initial cost during system development by leveraging existing Wi-Fi infrastructure. Therefore, they offer an advantage over GPS. Wi-Fi tracking system, based on RSSI-based distance, can be implemented in indoor applications, such as campus areas, as demonstrated in this paper (i.e., ITB area). It is recommended to conduct further studies to assess the performance of system in real-world settings, as all the demonstrations were conducted in a controlled environment. The accuracy of system in tracking various elements in the real world also needs to be examined.

The demonstration results show the successful monitoring of crowdedness using the online dashboard. System effectively captures the volume of smartphones connected to Wi-Fi, as anticipated, and crowdedness data is displayed at five-second intervals. Furthermore, the online dashboard provides the ability to specify the location of smartphones, such as inside devices, outside devices, and within a specific range. This study does not include statistical analysis as it solely reports on system architecture and system demo. A previous study presented a similar framework, but it was specifically dedicated to outdoor applications in wide areas, particularly for smart city scenarios [34]. In contrast, this proposed system is intended for indoor areas. The scientific advancement highlighted by this system is the visualization purpose of the dashboard.

Future study will encompass the evaluation of other performance metrics and practical performance measurements,

as identified by Xia et al. These metrics include accuracy, precision, complexity, robustness, scalability, and costs [43]. Several existing RSSI-based localization methods can be incorporated to enhance the proposed system and achieve an optimal balance. These methods include fingerprinting, distance-based approaches, statistical techniques, machine learning, deep learning, etc. Exploring these algorithms will allow for the identification of the most suitable approach within system.

V. Conclusion and Future Works

A significant area of study in recent years has been the development of Wi-Fi tracking system driven by the increasing number of smartphone users. People tend to be constantly connected to the internet through various sources, including free Wi-Fi spots. Therefore, there is a growing interest in tracking individuals using the signals emitted by their smartphones. This study focuses on designing, implementing, and demonstrating Wi-Fi Tracker system specifically for indoor crowd monitoring. System comprises two main subsystems, namely, tracker node and the server. To utilize system, users must place a minimum of three tracker nodes in a specific location to establish wireless networks. These tracker nodes serve as wireless sensors that scan smartphone packet request data. Simultaneously, the server and an online dashboard must be prepared to display real-time data. System captures three crucial pieces of information, namely, Received Signal Strength Indicator (RSSI), Media Access Control (MAC) address, and timestamp of the smartphones. These data are then transmitted from tracker node to the server. Once received, the server performs computations on the data, including processing MAC address, timestamp, and RSSI. By implementing a localization algorithm configured through the web-based dashboard, system can predict the location of smartphones and analyze their distribution. The dashboard is accessible through a specific PC using a unique web address.

The primary significance of the described system lies in its ability to capture and analyze MAC addresses and RSSI instances. However, the challenge posed by MAC randomization is an important consideration for future work. Nowadays, most smartphones employ MAC randomization techniques to prevent Wi-Fi tracking. This behavior involves broadcasting a randomized MAC address instead of the actual one, making it challenging to track and identify devices accurately. In practice, this could result in an influx of fake users generated by the same smartphone, particularly in scenarios where two users are close. Addressing this issue and developing strategies to differentiate between genuine and fake users in real-world situations is essential.

Acknowledgment

The authors are grateful to Universitas Pendidikan Indonesia for their support and assistance throughout this study. They are also grateful to Program Peningkatan Global Competitiveness Perguruan Tinggi Indonesia for handling the publication fee and proof-editing process through the Universitas Pendidikan Indonesia 2021 Batch II with No SK 1370/UN40/PT.01.02/2021. This article is based on the project report in collaboration between Pusat Miktroelektronika ITB the Stream Intelligence, Inc.

References

[1] L.-P. Tian, L.-Q. Chen, Z.-M. Xu, and Z. (David) Chen, “Wits: An Efficient Wi-Fi Based Indoor Positioning and Tracking System,” Remote Sensing, vol. 14, no. 1, p. 19, Jan. 2022, doi: 10.3390/rs14010019.

[2] M. Ribeiro, D. Teixeira, P. Barbosa, and N. J. Nunes, “Using passive Wi-Fi for community crowd sensing during the COVID-19 pandemic,” Journal of Big Data, vol. 10, no. 1, p. 7, Jan. 2023, doi: 10.1186/s40537-022-00675-3.

[3] N. Jarvis, J. Hata, N. Wayne, V. Raychoudhury, and M. O. Gani, “MiamiMapper: Crowd Analysis using Active and Passive Indoor Localization through Wi-Fi Probe Monitoring,” in Proceedings of the 15th ACM International Symposium on QoS and Security for Wireless and Mobile Networks, in Q2SWinet’19. New York, NY, USA: Association for Computing Machinery, Nov. 2019, pp. 1–10. doi: 10.1145/3345837.3355959.

[4] X. Tang, B. Xiao, and K. Li, “Indoor Crowd Density Estimation Through Mobile Smartphone Wi-Fi Probes,” IEEE Transactions on Systems, Man, and Cybernetics: Systems, vol. 50, no. 7, pp. 2638–2649, Jul. 2020, doi: 10.1109/TSMC.2018.2824903.

[5] T. D. Vy, T. L. N. Nguyen, and Y. Shin, “Pedestrian Indoor Localization and Tracking Using Hybrid Wi-Fi/PDR for iPhones,” in 2021 IEEE 93rd Vehicular Technology Conference (VTC2021-Spring), Helsinki, Finland: IEEE, Apr. 2021, pp. 1–7. doi: 10.1109/VTC2021-Spring51267.2021.9448859.

[6] A. Hidayat, S. Terabe, and H. Yaginuma, “Bus Passenger Volume and Origin-Destination Based on Field Surveys Using a Wi-Fi Scanner,” Transportation Research Procedia, vol. 48, pp. 1376–1389, Jan. 2020, doi: 10.1016/j.trpro.2020.08.169.

[7] A. Hidayat, S. Terabe, and H. Yaginuma, “Estimating bus passenger volume based on a Wi-Fi scanner survey,” Transportation Research Interdisciplinary Perspectives, vol. 6, p. 100142, Jul. 2020, doi: 10.1016/j.trip.2020.100142.

[8] D. B. Paradeda, W. K. Junior, and R. C. Carlson, “Bus passenger counts using Wi-Fi signals: some cautionary findings,” TRANSPORTES, vol. 27, no. 3, pp. 115–130, Nov. 2019, doi: 10.14295/transportes.v27i3.2039.

[9] S. Ryu, B. B. Park, and S. El-Tawab, "Wi-Fi Sensing System for Monitoring Public Transportation Ridership: A Case Study," KSCE J Civ Eng, vol. 24, no. 10, pp. 3092–3104, Oct. 2020, doi: 10.1007/s12205-020-0316-7.

[10] G. Pipelidis, N. Tsiamitros, M. Kessner, and C. Prehofer, "HuMAn: Human Movement Analytics via Wi-Fi Probes," in 2019 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops), Kyoto, Japan: IEEE, Mar. 2019, pp. 370–372. doi: 10.1109/PERCOMW.2019.8730703.

[11] D. N. Fernández, “Implementation of a WiFi-based indoor location system on a mobile device for a university area,” in 2019 IEEE XXVI International Conference on Electronics, Electrical Engineering and Computing (INTERCON), Lima, Peru, Aug. 2019, pp. 1–4. doi: 10.1109/INTERCON.2019.8853556.

[12] K. S. Chaithra and P. Parimala, “Client Position Detection using Wi-Fi Technology,” in 2020 Second International Conference on Inventive Research in Computing Applications (ICIRCA), Coimbatore, India: IEEE, Jul. 2020, pp. 620–624. doi: 10.1109/ICIRCA48905.2020.9182890.

[13] J. Andión, J. M. Navarro, G. López, M. Álvarez-Campana, and J. C. Dueñas, “Smart Behavioral Analytics over a Low-Cost IoT Wi-Fi Tracking Real Deployment,” Wireless Communications and Mobile Computing, vol. 2018, p. e3136471, Dec. 2018, doi: 10.1155/2018/3136471.

[14] Y. Zeng, P. H. Pathak, and P. Mohapatra, "Analyzing Shopper's Behavior through Wi-Fi Signals," in Proceedings of the 2nd Workshop on Workshop on Physical Analytics, in WPA ’15. New York, NY, USA: Association for Computing Machinery, May 2015, pp. 13–18. doi: 10.1145/2753497.2753508.

[15] A. Hidayat, S. Terabe, and H. Yaginuma, "Mapping of MAC Address with Moving Wi-Fi Scanner," International Journal of Artificial Intelligence Research, vol. 1, no. 2, pp. 34–40, Oct. 2017, doi: 10.29099/ijair.v1i2.27.

[16] Y. Wang, B. Zhao, and Z. Jiang, "RSSI-Based Smooth Localization for Indoor Environment," The Scientific World Journal, vol. 2014, p. e639142, Jun. 2014, doi: 10.1155/2014/639142.

[17] F. Potortì, A. Crivello, M. Girolami, P. Barsocchi, and E. Traficante, "Localising crowds through Wi-Fi probes," Ad Hoc Networks, vol. 75–76, pp. 87–97, Jun. 2018, doi: 10.1016/j.adhoc.2018.03.011.

[18] S. Fuada et al., "Your MAC Address Can be Detected Easily When Your Smartphone Connected to the Wi-Fi," International Journal of Interactive Mobile Technologies (iJIM), vol. 15, no. 07, pp. 176–184, Apr. 2021.

[19] Y. Li, J. Barthelemy, S. Sun, P. Perez, and B. Moran, "A Case Study of Wi-Fi Sniffing Performance Evaluation," IEEE Access, vol. 8, pp. 129224–129235, 2020, doi: 10.1109/ACCESS.2020.3008533.

[20] M. Cunche, "I know your MAC address: targeted tracking of individual using Wi-Fi," J Comput Virol Hack Tech, vol. 10, no. 4, pp. 219–227, Nov. 2014, doi: 10.1007/s11416-013-0196-1.

[21] S. Fuada, T. Adiono, and P. Prasetiyo, "Accuracy Improvement of RSSI-based Distance Localization using Unscented Kalman Filter (UKF) Algorithm for Wi-Fi Tracking Application," International Journal of Interactive Mobile Technologies (iJIM), vol. 14, no. 16, pp. 225–233, Sep. 2020.

[22] S. Fuada, T. Adiono, P. -, and H. Widhanto, "Modelling an Indoor Crowd Monitoring System based on RSSI-based Distance," IJACSA, vol. 11, no. 1, 2020, doi: 10.14569/IJACSA.2020.0110181.

[23] Y. Sung, "RSSI-Based Distance Estimation Framework Using a Kalman Filter for Sustainable Indoor Computing Environments," Sustainability, vol. 8, no. 11, p. 1136, Nov. 2016, doi: 10.3390/su8111136.

[24] C. Matte and M. Cunche, "DEMO: Panoptiphone: How Unique is Your Wi-Fi Device?," in Proceedings of the 9th ACM Conference on Security & Privacy in Wireless and Mobile Networks, in WiSec '16. New York, NY, USA: Association for Computing Machinery, Jul. 2016, pp. 209–211. doi: 10.1145/2939918.2942417.

[25] S. Jain, E. Bensaid, and Y.-A. de Montjoye, "UNVEIL: Capture and Visualise Wi-Fi Data Leakages," in The World Wide Web Conference, in WWW '19. New York, NY, USA: Association for Computing Machinery, May 2019, pp. 3550–3554. doi: 10.1145/3308558.3314143.

[26] H. Hong, G. D. De Silva, and M. C. Chan, "CrowdProbe: Non-invasive Crowd Monitoring with Wi-Fi Probe," Proc. ACM Interact. Mob. Wearable Ubiquitous Technol., vol. 2, no. 3, p. 115:1-115:23, Sep. 2018, doi: 10.1145/3264925.

[27] Y. Xie, J. Xiong, M. Li, and K. Jamieson, "mD-Track: Leveraging Multi-Dimensionality for Passive Indoor Wi-Fi Tracking," in The 25th Annual International Conference on Mobile Computing and Networking, in MobiCom '19. New York, NY, USA: Association for Computing Machinery, Aug. 2019, pp. 1–16. doi: 10.1145/3300061.3300133.

[28] K. Qian, C. Wu, Y. Zhang, G. Zhang, Z. Yang, and Y. Liu, "Widar2.0: Passive Human Tracking with a Single Wi-Fi Link," in Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services, in MobiSys '18. New York, NY, USA: Association for Computing Machinery, Jun. 2018, pp. 350–361. doi: 10.1145/3210240.3210314.

[29] X. Li et al., "IndoTrack: Device-Free Indoor Human Tracking with Commodity Wi-Fi," Proc. ACM Interact. Mob. Wearable Ubiquitous Technol., vol. 1, no. 3, p. 72:1-72:22, Sep. 2017, doi: 10.1145/3130940.

[30] N. Nunes, M. Ribeiro, C. Prandi, and V. Nisi, "Beanstalk: a community based passive Wi-Fi tracking system for analysing tourism dynamics," in Proceedings of the ACM SIGCHI Symposium on Engineering Interactive Computing Systems, in EICS '17. New York, NY, USA: Association for Computing Machinery, Jun. 2017, pp. 93–98. doi: 10.1145/3102113.3102142.

[31] K. Li, C. Yuen, and S. Kanhere, "SenseFlow: An Experimental Study of People Tracking," in Proceedings of the 6th ACM Workshop on Real World Wireless Sensor Networks, in RealWSN '15. New York, NY, USA: Association for Computing Machinery, Nov. 2015, pp. 31–34. doi: 10.1145/2820990.2820994.

[32] Y. Jiang et al., "ARIEL: automatic Wi-Fi based room fingerprinting for indoor localization," in Proceedings of the 2012 ACM Conference on Ubiquitous Computing, in UbiComp '12. New York, NY, USA: Association for Computing Machinery, Sep. 2012, pp. 441–450. doi: 10.1145/2370216.2370282.

[33] J. Scheuner et al., "Probr - A Generic and Passive Wi-Fi Tracking System," in 2016 IEEE 41st Conference on Local Computer Networks (LCN), Dubai, United Arab Emirates, Nov. 2016, pp. 495–502. doi: 10.1109/LCN.2016.30.

[34] A. Fernández-Ares et al., "Studying real traffic and mobility scenarios for a Smart City using a new monitoring and tracking system," Future Generation Computer Systems, vol. 76, pp. 163–179, Nov. 2017, doi: 10.1016/j.future.2016.11.021.

[35] A. E. C. Redondi and M. Cesana, "Building up knowledge through passive Wi-Fi probes," Computer Communications, vol. 117, pp. 1–12, Feb. 2018, doi: 10.1016/j.comcom.2017.12.012.

[36] T. Adiono et al., "Prototyping design of IR remote controller for smart home applications," in TENCON 2017 - 2017 IEEE Region 10 Conference, Penang, Malaysia: IEEE, Nov. 2017, pp. 1304–1308. doi: 10.1109/TENCON.2017.8228059.

[37] T. Adiono, M. Y. Fathany, S. Fuada, I. G. Purwanda, and S. F. Anindya, "A portable node of humidity and temperature sensor for indoor environment monitoring," in 2018 3rd International Conference on Intelligent Green Building and Smart Grid (IGBSG), Yi-Lan: IEEE, Apr. 2018, pp. 1–5. doi: 10.1109/IGBSG.2018.8393575.

[38] S. Fuada, A. Alfaruq, and T. Adiono, "A Portable Electronic Transaction Device Based on Dual Interface Smart Card," International Journal of Online and Biomedical Engineering (iJOE), vol. 16, no. 03, pp. 27–45, Mar. 2020.

[39] T. Adiono, M. Y. Fathany, S. Feranti Anindya, S. Fuada, and I. G. Purwanda, "Using A Smart Plug based on Consumer Electronics to Support Low Power Smart Home," in 2019 4th International Conference on Intelligent Green Building and Smart Grid (IGBSG), Hubei, Yi-chang, China: IEEE, Sep. 2019, pp. 376–379. doi: 10.1109/IGBSG.2019.8886272.

[40] M. B. Yassein, M. Q. Shatnawi, S. Aljwarneh, and R. Al-Hatmi, "Internet of Things: Survey and open issues of MQTT protocol," in 2017 International Conference on Engineering MIS (ICEMIS), Monastir, Tunisia: IEEE, May 2017, pp. 1–6. doi: 10.1109/ICEMIS.2017.8273112.

[41] S. Kraijak and P. Tuwanut, "A survey on internet of things architecture, protocols, possible applications, security, privacy, real-world implementation and future trends," in 2015 IEEE 16th International Conference on Communication Technology (ICCT), Hangzhou, China: IEEE, Oct. 2015, pp. 26–31. doi: 10.1109/ICCT.2015.7399787.

[42] R. S. Campos, L. Lovisolo, and M. L. R. de Campos, "Wi-Fi multi-floor indoor positioning considering architectural aspects and controlled computational complexity," Expert Systems with Applications, vol. 41, no. 14, pp. 6211–6223, Oct. 2014, doi: 10.1016/j.eswa.2014.04.011.

[43] S. Xia, Y. Liu, G. Yuan, M. Zhu, and Z. Wang, "Indoor Fingerprint Positioning Based on Wi-Fi: An Overview," ISPRS International Journal of Geo-Information, vol. 6, no. 5, p. 135, May 2017, doi: 10.3390/ijgi6050135.