Smart Insoles for Multi-User Monitoring: A Case Study on Received Signal Strength Indicator-Based Distance Measurement

Víctor Huilca Cabay, Alexandra Flores, Paúl Hernán Machado Herrera, Byron Paul Huera Paltan
Department of Informatics and Electronics, Escuela Superior Politécnica De Chimborazo, Riobamba, Ecuador 060150

***Abstract*—In the current context of high adoption of wearables and Internet of Things (IoT) devices, this work develops a smart insole system to measure the distance between users using the RSSI signal (Received Signal Strength Indicator). ESP32 WROOM microcontrollers with Bluetooth Low Energy, Wi-Fi, and multiple functionalities were used. The prototype includes sensors to count steps, detect activity (walking/running) and a configurable alarm to alert when the distance is less than a threshold. Collected data are sent directly and in real-time to a database using the ThingSpeak web platform, which allows to visualize the data acquired from the insole sensors. Using the RSSI signal provided by the Bluetooth LE module, a significant response was interpreted and modeled using a multilayer perceptron (MLP) neural network, achieving an average distance estimation accuracy of 90.89% using data measured in real time.**

***Keywords*—Internet of Things; RSSI; smart insole; distance; wearables; neural network**

I. Introduction

Modern electronics have achieved the miniaturization of devices and rapid processing speeds, reaching a level of integration that has made it possible to include computational capabilities in everyday objects. In addition, garments can advantage over the potential of digital electronics, incorporating sensors and actuators. Two strong areas, industry and scientific research, are promoting this type of device, commonly called wearables [1]. Typically, wearables gather user data related to specific activities or physiological parameters. Usability, discretion, and reliability are the key points involved in the design of these devices. Furthermore, measurement precision and reliability play an important role, particularly in professional sports [2] and health applications [3], where portable devices are intended to replace or at least act as closely as possible with high-quality laboratory and hospital devices. Activity trackers, which incorporate inertia detection [4], are one of the most popular types of wearable devices, taking advantage of the consumer's need to maintain healthy behaviors, stay active, and take care of their physical condition. Currently, there are wearable devices with several sensors that are of great help in collecting body data in general [5], [6], but none include cell phones and devices from the Global Positioning System (GPS) that determine the distance between devices or users. These wearable devices are very useful when it comes to monitoring the behavior and movement of people; An example of this is human walking [7], which is one of the most common activities that humans perform from an early age. This activity provides us with several useful data to determine aspects related to the health of the individual and also considerations in the sports area. Plantar pressures in a walk provide important information on the duration and symmetry of gait cycles [8]; With this we can determine whether an individual walks in a normal way or suffers from pathologies and abnormal plantar conformations, as they often alter the functionality of the foot with consequences throughout the muscular system of the lower extremities and in the spine. Lately, intelligent insole systems have come onto the market for individual use and are mainly aimed at the sports area, allowing monitoring data related to walking or running.

In [9], the authors propose the use of Smart Insole to obtain an accurate step count directed to the real world. The step counting method is based on the threshold differential value of the mean plantar pressure. The results obtained with this prototype indicate an accuracy of almost 100% in measuring the count of steps. The Smart Insole FreeWalker detailed in [10] works with a custom designed acquisition, transmission, and reception unit. It is composed of an MPU-6050 inertial unit that has an accelerometer and gyroscope, fed 3.3 Vdc, a robust and sufficient microcontroller to carry out the objective of the project. This device is capable of identifying the steps during a gait cycle, both the acquisition and the transmission of information being carried out in real time. In [11], the comparison of three methods is proposed to estimate the distance by means of the received RSSI signal, with which they obtain the indoor positioning and navigation (IPIN). The proposed methodology is that of a novel multi-step approach that combines the flat-earth model, the Friis free space model, and the linear approximation model to measure the distance from RSSI for smart devices with Bluetooth low-energy connectivity (BLE). In addition, the authors propose an improved RSSI averaging and smoothing algorithm to obtain a better result, with which they claim a reduction of 13.4% in the error of the measured distance.

This work presents the development of a smart insole with user-friendly and reliable features, designed to monitor human gait. Its primary function is to determine the distance between two individuals, each equipped with a smart insole, offering innovative applications in both sports and medical fields. In sports, this technology facilitates training in pairs or teams by ensuring that users maintain an optimal distance to improve coordination and prevent accidents during physical activity. In the medical field, smart insoles serve as a valuable tool for physical rehabilitation, allowing the monitoring and evaluation of patient progress in gait therapy. Additionally, their implementation in sports competitions could help ensure

safe and efficient performance. To achieve this goal, a system was developed using smart insoles capable of quantifying the distance between two devices through the RSSI signal. The technological solution uses ESP32 WROOM microcontrollers with low-energy Bluetooth (BLE) modules, complemented by sensors for step detection, activity analysis, and scheduled notifications. The data collected are transmitted in real time to a cloud database via the ThingSpeak platform [12], allowing precise and continuous data analysis.

The paper is structured as follows. Section II describes the system architecture. Section III provides a detailed explanation of the system implementation along with each of the processes involved. Section IV presents the numerical results obtained. Finally, the conclusions are presented in Section V.

II. System Architecture

This section details the complete architecture of the smart insoles system through electrical diagrams. Fig. 1 shows a high-level schematic in its entirety. The system is composed of the ESP-32 WROOM processing unit or microcontroller, the FSR (Force sensing resistors) sensors connected by analog pins, the MPU6050 accelerometer to identify the stroke phase, the BLE module built into the ESP32 WROOM, to determine the RSSI parameter, the Wi-Fi module to communicate between the smart insole and the ThingSpeak Web Server and finally the 3.7 Volt battery used as the power supply of the entire system.

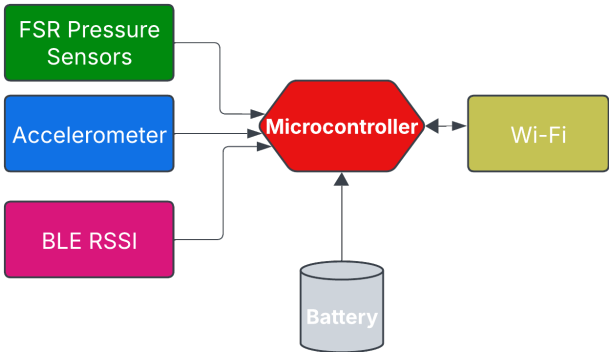

Fig. 1. High-level scheme of the smart insole system.

The diagram in Fig. 2 details the connection of all the electronic elements used for the smart insoles, these are:

- The ESP32 WROOM microcontroller;
- A BLE module included in the microcontroller;
- Three FSR sensors (heel, right forefoot and left forefoot);
- The inertial module MPU6050;
- Three 220 ohm resistors;
- A 3.7 volt and 1000mAh battery.

Due to the versatility of the microcontroller used, it is possible to choose between a 5Vdc supply through the Vin pin, or also a 3.3 Vdc supply through the 3V3 pin; for this work the 3.3Vdc option has been used since with this voltage is also fed directly to the MPU6050 inertial module. In the case of FSR sensors, to properly condition their signals before entering the reading into the analog inputs of the microcontroller, it is necessary to make a voltage divider for each of these. To measure RSSI, it is useful to determine the distance between the two smart insoles using the Bluetooth BLE module built into ESP32. As mentioned above, the 3.7Vdc battery is also shown, which provides the adequate voltage for the operation of the entire system, it should be noted that the above detailed is considered for both the MASTER and SLAVE prototypes.

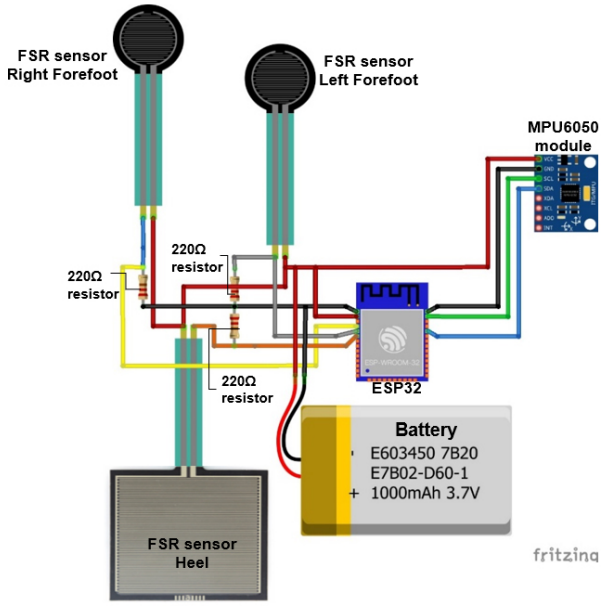

Fig. 2. Smart Insole system architecture.

Fig. 3 details the steps carried out by the master and slave systems: in which the procedure is exactly the same, with the difference that the slave will be the one that processes the RSSI signal of the system. The raw signals are acquired by the three pressure sensors, the accelerometer, and the RSSI, after which all this data is processed by the microcontroller and determine the phases of movement (stop, walk, and running), distance traveled, and distance between the two individuals through the algorithms. Using the Wi-Fi module, this information is transmitted to the ThingSpeak platform for visualization and can later be exported in $.xls$ format for further analysis of the collected data.

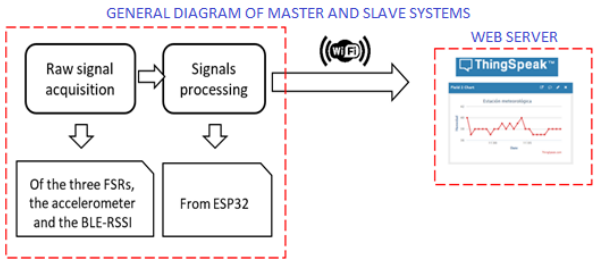

Fig. 3. Conceptual diagram of the MASTER/SLAVE smart insole system.

III. Implementation

The elements integrated into the smart insole system are described in detail below:

A. Processes for Reading Data from FSR Sensors

The procedure of obtaining the step count begins with the analysis of the FSR 402 and FSR 406 sensors. The initial stage involves collecting the signals from each of these sensors. Since these sensors operate by changing their resistance in reaction to applied force, it is crucial to condition their signals utilizing a voltage divider circuit, as depicted in Fig. 4. This voltage divider converts resistance changes into an analog signal for processing. To analyze analog signals more precisely and reliably, a mapping is needed to discretize the output of the voltage divider to a value between 0 and 5000, which is more manageable for the electronic field.

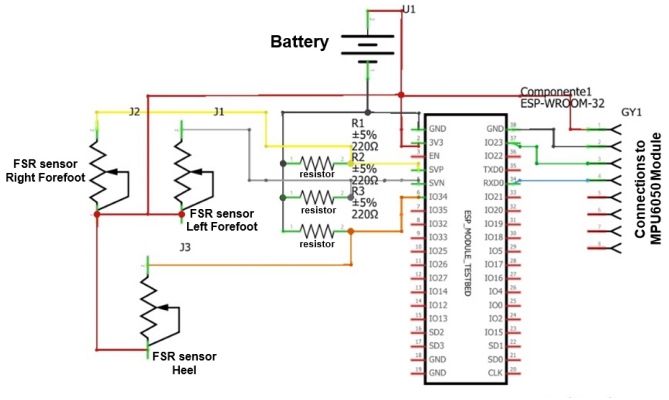

Fig. 4. Smart insole system wiring diagram.

B. Processes for Counting Steps and Identify Walk

The system initiates the step counting phase by pressing the heel sensor, then applies pressure to the right forefoot sensor, and completes the sensing process by applying pressure to the left forefoot sensor, which corresponds to the heel strike, mid-stance, and toe-off phases. To ensure accurate step detection, a state variable is implemented. This variable helps the system recognize when the heel sensor has already been pressed, preparing it to read subsequent inputs from the forefoot sensors. This mechanism ensures that only valid step sequences are counted. In addition, a control mechanism is integrated to distinguish between isolated steps and continuous walking. To achieve this, a fine-step variable is used to determine whether the user has taken more than two consecutive steps, which is considered the threshold to detect the beginning of a walk. To further refine the identification of steps, a function was used to store time stamps, allowing the system to measure the time difference between successive steps. The system then compares this difference to a predefined update rate value, enabling it to distinguish between a walking state and a stationary state. Algorithm 1 details this process.

C. Process for Determining the Run

This section analyzes the method for determining when the individual utilizing the smart insole starts a race. We have selected the IMU MPU6050 sensor for this purpose because it provides acceleration data across three critical axes. X, Y, and Z, essential for analysis and comparison in this context. Due to the placement of the sensor, it is adequate to perform the analysis solely on the X-axis, as it exhibits the most significant variation throughout the running movement. The Algorithm 2 determines whether a user is running based on acceleration measurements from the IMU sensor on $X-axis$. First, it reads the absolute acceleration and scales it by multiplying it by a factor of 5. If the acceleration exceeds a predefined threshold, it indicates the start of movement, storing the timestamp. If acceleration remains above the threshold with control $= 1$, it checks whether the time difference between detections is less than the running frequency threshold. If successive peaks are detected within this time window, a hit counter is incremented. When more than three peaks are detected within the running frequency, the system confirms that the user is running. If too much time passes without detecting new acceleration peaks, the algorithm assumes that the user has stopped running, resetting the variables. Finally, if hit_coun is reset to 0, the function ensures that the running is also set to false, preventing false running detections.

Algorithm 1 Step Counting and Walk Detection

```
Initialize: step_count, motion_state, start_steps, completed_steps, walking ← false
Set Thresholds: heel_activation, forefoot_low
Detect Step
if heel_pressure > heel_activation and forefoot_pressure < forefoot_low then
    motion_state ← 1, last_time ← get_current_time() (if first step)
end if
Confirm Step
if motion_state and heel_pressure > heel_release and forefoot_pressure < forefoot_release then
    step_count++, motion_state ← 0, start_steps++, completed_steps++
end if
Check Walking
if completed_steps > step_threshold and time since last_time < time_window and not running then
    walking ← true, last_time ← get_current_time()
end if
Reset If Inactive
if time since last_time > time_window then
    start_steps, completed_steps ← 0, walking ← false
end if
```

D. Process for Determining the Distance Traveled

In this section of the study, the distance traveled should be measured while keeping in mind whether the walking and running stages have already started. The logic followed is detailed in Algorithm 3.

The Algorithm 3 determines the distance traveled. It begins by establishing parameters including the step length and update interval, and initializing the required variables. The method gets the current time, computes the distance by multiplying the step count by the average step length using (1), and displays

Algorithm 2 Running Detection Algorithm

```
Initialize: motion_state, hit_count, running ← false
Set Thresholds: acceleration_limit, time_window, hit_threshold
Measure Acceleration: acceleration ← abs(sensor_data) × scale
Detect Motion
if acceleration > acceleration_limit then
    motion_state ← 1, last_time ← current_time()
    if time since last_time < time_window then
        hit_count++
    else
        hit_count ← 0, motion_state ← 0
    end if
end if
Determine Running
if hit_count > hit_threshold and time since last_time < time_window then
    running ← true
else
    running ← false, hit_count ← 0
end if
return running
```

Algorithm 3 Distance Traveled Calculation

```
Initialize: previous_time, current_time, steps, state
Set Parameters: step_length, update_interval
if time since previous_time > update_interval then
    current_time ← get_current_time()
    distance ← steps × Average_Steps
    Print "Steps:", steps, "Distance Traveled:", distance
    if walking and not running then
        Print "Walking"
        state ← 1
    else if not walking and not running then
        Print "Stopped"
        state ← 0
    else if running then
        Print "Running"
        state ← 2
    end if
    previous_time ← current_time
end if
Delay 100 ms
```

the number of steps and the distance traveled every time the designated update interval has passed. It then checks the user's activity state: if the user is walking (but not running), it prints $Walking$ and sets the state to 1; if the user is stopped, it prints $Stopped$ and sets the state to 0; if running, it prints $Running$ and sets the state to 2. After updating the state, it records the current time as $previous\_time$ for the next interval and introduces a 100 ms delay to control the update frequency.

$$d = (s * \Delta a) \tag{1}$$

where: d is the distance traveled, s is the number of steps and Δa is the average step length.

E. Mechanism to Evaluate the Distance between Two Individuals

The interpersonal distance between two smart insoles (Master and Slave) is determined using RSSI (Received Signal Strength Indicator) values through BLE communication. The ESP32 modules transmit signals, and RSSI readings are translated into distances measured in meters. A multilayer perceptron (MLP) neural network was used due to the non-linear relationship between RSSI values and distance, which is influenced by elements such as interference, reflections, and signal attenuation in the surroundings. Although linear models such as regression could serve as an initial approximation, environmental variability induces data fluctuations that require a more adaptable model. Initially, 200 samples were used; however, to enhance the model's generalization, data augmentation techniques were implemented, expanding the dataset to 1000 samples, hence facilitating improved learning and adaption to data variances.

1) Data processing: The dataset comprises 1000 RSSI values along with their associated real distances. To enhance the learning efficiency of the model, the normalization of the data was performed using Min-Max Scaling [13], which ensured that the input values were maintained within a standardized range. The application of this transformation was executed using (2).

$$X' = \frac{X - X_{\text{min}}}{X_{\text{max}} - X_{\text{min}}} \tag{2}$$

where X is the original value and X' is the normalized value.

After normalization, the dataset was divided into three subsets: 80% designated for training, 10% for validation, and 10% for testing. This division enables the model to train effectively while ensuring robust generalization to new inputs.

2) MLP Neural network model: A deep multilayer perceptron (MLP) was constructed with three hidden layers comprising 32, 16, and 8 neurons, respectively, as shown in Fig. 5. We employed the ReLU activation function after each hidden layer to incorporate non-linearity, thereby enhancing the ability of the model to discern intricate correlations between RSSI and distance. The Adam optimizer [14] was selected for its efficiency in weight adjustments, facilitating accelerated convergence and stability. The model was trained with Mean Squared Error (MSE) [15] as the loss function to reduce the discrepancy between the predicted and actual distances.

3) Training process: The model was trained for 500 epochs, during which training loss and validation loss were continuously monitored to ensure proper learning and avoid overfitting. A training vs. validation loss plot (Fig. 6) was generated to visualize the learning process of the model over time. The plot shows that the model learned the RSSI-to-distance mapping correctly because the loss continued to decrease with each epoch.

After training, the model produced the learned (3) to predict the distance from the RSSI values.

$$Distance = 0.0386 \times \text{RSSI} - 1.3271 \tag{3}$$

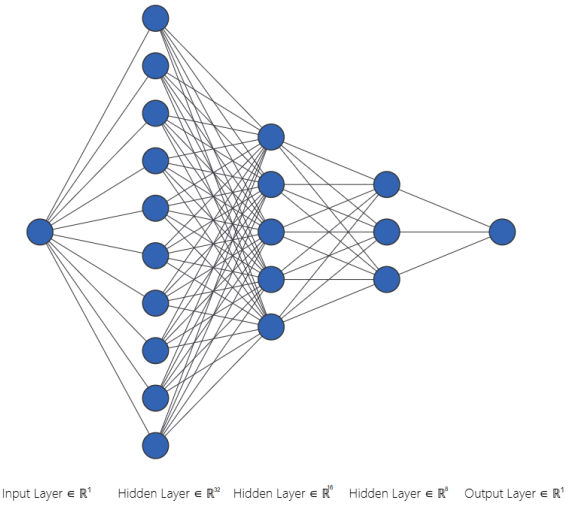

Fig. 5. Deep multilayer perceptron (MLP).

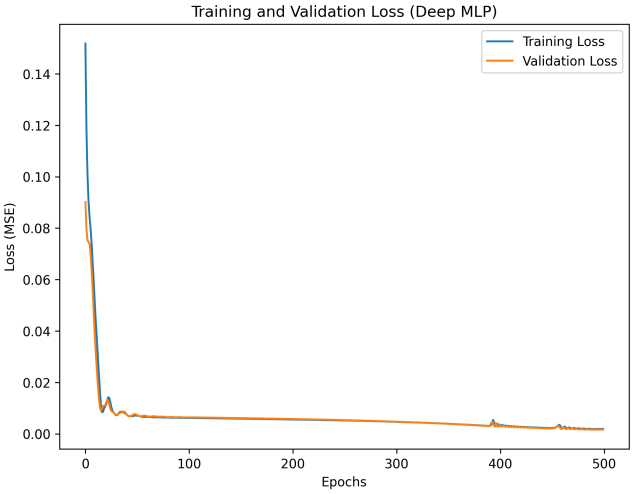

Fig. 6. Training and validation loss plot.

The learned weight is represented by 0.0386, and the learned bias from the first layer of the deep MLP is represented by -1.3271. This equation gives a number value to the relationship between RSSI and distance, which makes it easier to figure out distances in the future without having to retrain the model. To evaluate the prediction of the model, a scatter plot was generated (Fig. 7). In it, the blue dots represent the predicted distances, while the red dashed line indicates the ideal predictions, corresponding to a perfect correlation. The predictions closely align with the ideal line, indicating their high predictive accuracy.

In the Arduino GUI code, (3) was used to calculate the power values in decibels that correspond to changes in distance, as shown in Fig. 8, which shows these changes through the Arduino serial interface. It is now feasible to achieve an automatic variation of the distance values.

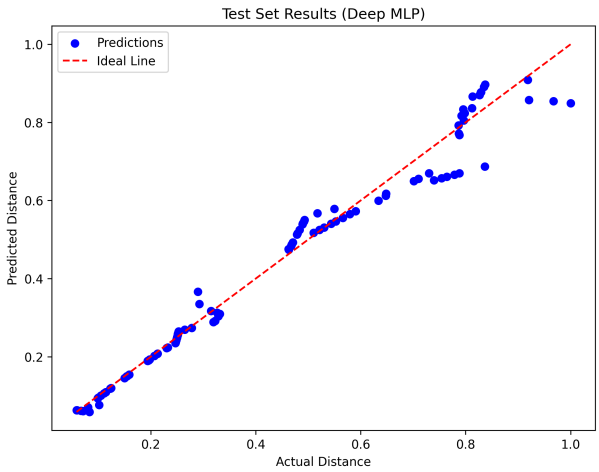

Fig. 7. Predictions vs Actual distances plot.

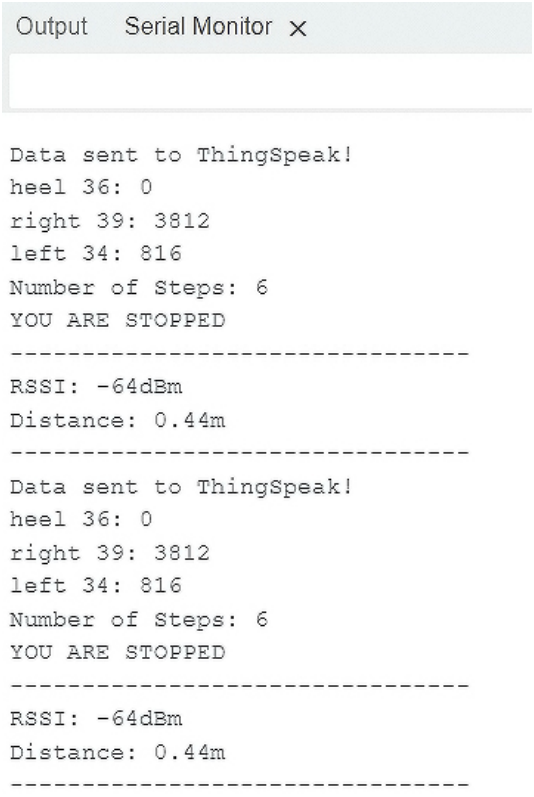

Fig. 8. Reading the arduino IDE serial port.

F. Process to Transmit Data to ThingSpeak

The platform used for real-time visualization of the data acquired by the intelligent insole is ThingSpeak [16]; this allows us to collect and store data from sensors in the cloud and develop IoT applications. Described as an open source platform with an API to store and retrieve data from objects using the HTTP protocol over the Internet or through LAN (local area network) [17]. All data, user state, step count, traveled distance, RSSI, and interpersonal distance are sent to a ThingSpeak database. The system is configured to update

every 15 seconds with an alert mechanism for distances below the set threshold. This comprehensive monitoring supports multi-user activity tracking and ensures accurate validation of the functionality of the smart insole. Fig. 9 illustrates the representation of the parameters on the ThingSpeak platform.

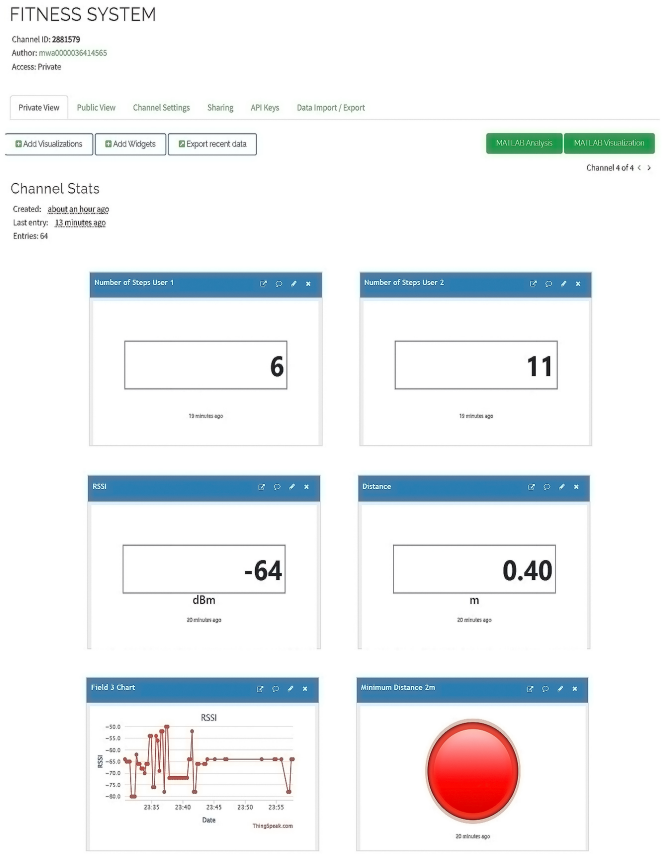

Fig. 9. Parameters on the ThingSpeak platform.

IV. Numerical Results

This section analyzes the test data from the smart insole system across various scenarios and planned routes. The system comprises two prototypes: a MASTER that transmits the RSSI signal via Bluetooth and a SLAVE that processes these RSSI data before transmitting them to the ThingSpeak platform, which functions as an IoT database, enabling real-time data visualization and export to Excel for subsequent statistical analysis. Each prototype has an ESP32-WROOM microcontroller, an MPU6050 module, and is powered by a rechargeable lithium battery of type NCR18650B, as an option, a Lipo battery of lower volume and higher performance or characteristics is recommended for an optimal wearable design. Several tests were performed to make sure that the RSSI-based step-counting and distance measurement system worked. These tests also found the system's accuracy and error range, which helped with the comparative analysis. Following the methodology described in [18], routines for data collection were designed, including two patterns per scenario: frontal crossing and cross-crossing within a defined area of 5×5 m^2. For this work, the results of a single scenario called "Crossing between smart insole prototypes in parallel opposite directions" are shown. The configuration for this scenario is illustrated in Fig. 10. The calculation of the real distance traveled by the participants was performed using a pedometer-based system. To obtain the real distance, we relied on the step count obtained from the system and multiplied it by an average step length that was calibrated for the individual. The speed of travel can be derived by calculating the time between updates and dividing the distance by the time elapsed.

Furthermore, to reduce interference, we executed the experiment in open and unobstructed environments, avoiding areas with walls or reflections that can influence sensor results. Outdoor testing was carried out on clear, dry days with minimal wind to mitigate weather influences. We verified that all equipment, including sensors, was fully charged before testing and performed periodic checks during extended tests to prevent battery-related data loss. In addition, test routes were meticulously chosen to avoid significant obstructions, such as trees or other objects, that could disrupt sensor readings, thus ensuring accurate data collection.

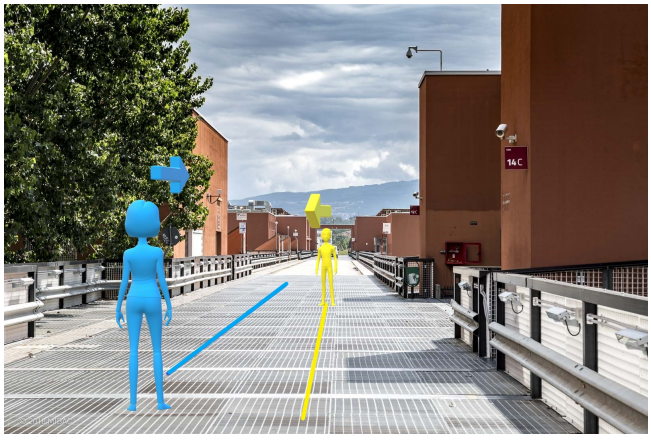

Fig. 10. Example of crossing between smart insole prototypes in parallel opposite direction.

A. Results of Crossing between Smart Insole Prototypes in Parallel Opposite Direction

Table I shows the data obtained for this case of analysis, in which it can be observed that the individual using the SLAVE prototype has taken 7 steps during the 5 meters of established area, for the user who uses the MASTER prototype visualizes that it has taken 8 steps during the established area. The average distance traveled by each prototype is 5.012 meters. It can also be visualized that in the fourth value, the distance measured by the system reflects 1.0404 meters; this distance will be compared with the real distance in Table II, and we will also obtain a percentage of precision and realative error. The relative error Re and the accuracy percentages were calculated using Eq. (4) and (5):

$$Re\ \% = \left(\frac{\text{Calc. Distance} - \text{Real Distance}}{\text{Real Distance}} \right) \times 100 \quad (4)$$

$$\text{Accuracy } \% = 100\% - \text{Error } \% \quad (5)$$

Furthermore, to compare our work with that proposed by [11], we used the root mean square error (RMSE), as detailed in Eq. (6). For the data obtained in Table II, the calculated RMSE was 0.313. To compute the RMSE percentage as in [11], we follow the procedure described in Eq. (7). The result obtained is 8. 79%, which is less than the 13. 4% reported in the reference work. This indicates that our proposed model achieves superior performance.

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{6}$$

Where, $\hat{y}_i$ is the distance obtained, y_i is the real distance, and n is the total number of observations.

$$\text{RMSE}_{\%} = \left(\frac{\text{RMSE}}{\text{mean}(y_i)}\right) \times 100 \tag{7}$$

TABLE I. CROSS BETWEEN SMART INSOLE PROTOTYPES IN PARALLEL OPPOSITE DIRECTION

Samples	RSSI (dBm)	User1 Steps	User2 Steps	Distance Obtained (m)	Distance Traveled (m)
1	-90	0	0	5.785	0
2	-92	0	1	4.335	0
3	-81	1	1	2.601	0.716
4	-72	2	3	1.040	1.432
5	-80	2	4	2.081	1.432
6	-92	3	5	4.029	2.148
7	-88	5	7	5.027	3.58
8	-91	7	8	5.297	5.012

TABLE II. INDIVIDUAL ERROR PERCENTAGE: CROSS BETWEEN SMART INSOLE PROTOTYPES IN PARALLEL OPPOSITE DIRECTION

Sample	Distance Obtained (m)	Real Distance (m)	*Re* (%)
1	5.28	5.1	3.62
2	4.34	3.7	17.16
3	2.60	2.3	13.09
4	1.04	1.2	13.30
5	2.08	2.3	9.53
6	4.03	3.7	8.89
7	5.27	5.1	3.39
8	5.30	5.1	3.86

According to the findings presented in Table II, the mean Re rate is 9.11% and the accuracy rate is 90.89%, as there are no impediments influencing the RSSI signal. Furthermore, it is evident that eight samples were collected during the test, with one instance of proximity between prototypes where the measurement fell below the stipulated minimum allowable distance of 2 meters.

Finally, in Fig. 11 we can see the system of multi-user intelligent insoles, mounted or assembled on the shoe of each person. We can also see graphically how the system works by collecting the data from the FSR force sensors, and IMU inertial, the interaction between Master and Slave prototypes, the connection of each prototype to the Wi-Fi network of a cell phone and the sending of the data to the ThingSpeak cloud platform.

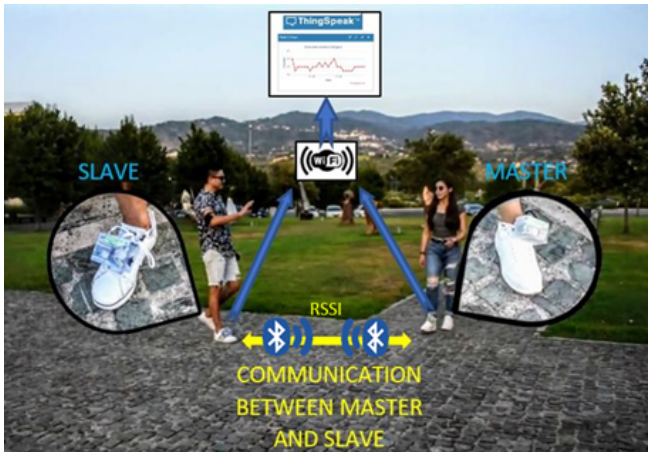

Fig. 11. Multi-user intelligent insole system assembled in each person's shoe.

V. CONCLUSIONS

In this work, a smart insole-based system has been proposed that allows the main objective to measure the distance between two users who interact with each other, be it in a daily activity, sports, or even at a medical level, in which the measurement and real information of the distance between individuals plays an important role. The prototype also includes sensors to count steps, detect activity (walking/running), and a configurable alarm to alert when the distance is less than a threshold. This system allows the storage, visualization, and monitoring of the data on the ThingSpeak Web platform, which allows quick and timely access to the data obtained thanks to the fact that the information is sent remotely to the web through protocols of wireless communication based on the internet of things. Distance estimation relies on the (RSSI), a cost-effective but unstable method due to low signal power and environmental obstructions. To address this, the work applied an MLP neural network, achieving an average accuracy of 90.89% in real data. The easy use and easy insertion in a common shoe make this system one of the best options as a wearable system, in addition to having a cost well below existing systems and with versatile features, which due to the microprocessor features can be varied or added according to the need of the end user.

In future work, the accuracy of the system could be improved by optimizing the RSSI method, integrating multiple sensors, and using advanced localization algorithms. In addition, its functionality could be expanded by creating a complementary mobile application and integrating it with wearable devices. Furthermore, new applications could be explored in the healthcare sector, such as monitoring patients or elderly individuals, and in high-density scenarios like mass events, enhancing safety and accident prevention.

ACKNOWLEDGMENT

The authors express their sincere gratitude to the University of Calabria, Italy, for the support provided during the development of this work. In particular, we extend our

appreciation to the Department of Computer Engineering, Modeling, Electronics, and Systems, DIMES, where the testing and implementation of the system were carried out. We also extend our heartfelt thanks to the Escuela Superior Politécnica de Chimborazo, ESPOCH, Ecuador, for its valuable support and collaboration in this research.

REFERENCES

[1] J. J. Rutherford, "Wearable Technology," in IEEE Engineering in Medicine and Biology Magazine, vol. 29, no. 3, pp. 19-24, May-June 2010, doi: 10.1109/MEMB.2010.936550.

[2] A. Ç. Seçkin, B. Ates, and M. Seçkin, "Review on wearable technology in sports: Concepts, challenges and opportunities," Appl. Sci., vol. 13, no. 18, p. 10399, Sep. 2023, doi: 10.3390/app131810399.

[3] A. K. Yetisen, J. L. Martinez-Hurtado, B. Ünal, A. Khademhosseini, and H. Butt, "Wearables in Medicine," Adv. Mater., vol. 30, no. 33, 2018, Art. no. 1706910.

[4] A. Wang, G. Chen, J. Yang, S. Zhao and C. -Y. Chang, "A Comparative Study on Human Activity Recognition Using Inertial Sensors in a Smartphone," in IEEE Sensors Journal, vol. 16, no. 11, pp. 4566-4578, June1, 2016, doi: 10.1109/JSEN.2016.2545708.

[5] M. Chan, D. Estève, J.-Y. Fourniols, C. Escriba, and E. Campo, "Smart wearable systems: Current status and future challenges," Artif. Intell. Med., vol. 56, no. 3, pp. 137–156, 2012.

[6] S. M. A. Iqbal, I. Mahgoub, E. Du, M. A. Leavitt, and W. Asghar, "Advances in healthcare wearable devices," npj Flexible Electron., vol. 5, no. 1, pp. 1–14, Apr. 2021.

[7] G. Zizzo and L. Ren, "Position tracking during human walking using an integrated wearable sensing system," Sensors, vol. 17, no. 12, p. 2866, Dec. 2017.

[8] M. N. Orlin and T. G. McPoil, "Plantar pressure assessment," Phys. Therapy, vol. 80, no. 4, pp. 399–409, Apr. 2000, doi: 10.1093/ptj/80.4.399.

[9] F. Lin, A. Wang, C. Song, W. Xu, Z. Li y Q. Li, "A comparative study of smart insole on real-world step count.," IEEE Signal Processing in Medicine and Biology Symposium (SPMB), vol. 1, no. 1, pp. 1-6, 2015.

[10] B. Wang, K. Rajput, W. Tam, A. Tung y Z. Yang, "FreeWalker: a smart insole for longitudinal gait analysis," 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), vol. 37, pp. 3723-3726, 2015.

[11] T. I. Chowdhury et al., "A multi-step approach for RSSi-based distance estimation using smartphones," 2015 International Conference on Networking Systems and Security (NSysS), Dhaka, Bangladesh, 2015, pp. 1-5, doi: 10.1109/NSysS.2015.7042942.

[12] S. Pasha, "Thingspeak based sensing and monitoring system for IoT with MATLAB analysis," Int. J. New Technol. Res., vol. 2, pp. 19–23, Jun. 2016.

[13] S. G. K. Patro and K. K. Sahu, "Normalization: A preprocessing stage," Mar. 2015, arXiv:1503.06462.

[14] I. K. M. Jais, A. R. Ismail, and S. Q. Nisa, "Adam optimization algorithm for wide and deep neural network," Knowl. Eng. Data Sci., vol. 2, no. 1, pp. 41–46, 2019

[15] Harville, D. A., and Jeske, D. R. (1992). "Mean squared error of estimation or prediction under a general linear model," Journal of the American Statistical Association, vol. 87, no. 419, pp. 724-731, 1992.

[16] M. A. A. Razali, M. Kassim, N. A. Sulaiman, and S. Saaidin, "A ThingSpeak IoT on real time room condition monitoring system," in Proc. IEEE Int. Conf. Autom. Control Intell. Syst. (I2CACIS), Jun. 2020, pp. 206–211.

[17] M. Artiyasa et al., "Comparative study of internet of things (IOT) platform for smarthome lighting control using NODEMCU with Thingspeak and Blynk Web Applications," FIDELITY : Journal of Electrical Engineering, vol. 2, no. 1, pp. 1-6, 2020. doi:10.52005/fidelity.v2i1.10.

[18] I. P. I. Pappas, T. Keller, S. Mangold, M. R. Popovic, V. Dietz and M. Morari, "A reliable gyroscope-based gait-phase detection sensor embedded in a shoe insole," in IEEE Sensors Journal, vol. 4, no. 2, pp. 268-274, April 2004, doi: 10.1109/JSEN.2004.823671.