TinyML to Predict Occupancy Levels Using Microcontrollers

Nicholas DeBaise
Union College, ECBE Department
Email: debaisen@union.edu

Abstract—**The ability to track occupancy levels of buildings such as campus attractions like the gym or dining hall is crucial to both users and administrators. An occupancy tracking system will enable users to effectively plan visits at optimal times to minimize wait times and equip administrators to allocate resources better. This work proposes a microcontroller-based system that captures WiFi probe requests at a location and uses a machine-learning model to determine the level of occupancy from the probe requests. The system runs on an ESP32-S3 and works by capturing WiFi probe requests, which WiFi-enabled devices such as phones and laptops emit, and uses a Random Forest regression model to predict the levels of occupancy. Occupancy data and WiFi probe requests in a gym area of 30m by 80m were collected and the data was used to train a Random Forest regression model. The system performance was evaluated when the regression model was deployed on the microcontroller and an off-device server. The system predicts the number of people in the gym with a mean absolute error of 1.743 off-device and 4.36 on-device. These results demonstrate that low-cost microcontrollers like the ESP32 are viable devices to run machine learning on microcontrollers (TinyML) and predict levels of occupancy using WiFi signals.**

1. Introduction

Modern day WiFi-enabled devices consistently emit probe requests as part of their network management routines, seeking to discover and connect with nearby wireless networks. Microcontrollers with WiFi capabilities such as the ESP32 can collect these probe requests, which offers a rich dataset where levels of occupancy in buildings can be determined. Occupancy information can be used for numerous operations, such as resource allocation or crowd management. For instance, accurate occupancy data in a building can allow for more efficient energy use in terms of heating, cooling, and lighting, and helps in deploying staff where they are needed most. In a college or university setting, it can be useful to know when campus attractions such as the gym or dining hall are overcrowded or busy as it can aid in planning schedules accordingly. Despite the potential, there is currently a lack of cost-effective, energy-efficient, non-invasive methods for real-time occupancy level prediction. This research proposes a low-cost method which can be easily deployed in non-invasive manners in different buildings.

This paper proposes a system which uses an energy-efficient WiFi-based microcontroller to capture probe request data and predict levels of occupancy in given rooms on campus. In a university setting, most students and faculty move about with WiFi-based devices such as smartphones, laptops, or smartwatches. Since these devices constantly emit probe requests, even while connected to networks, they can be used to predict the levels of occupancy. Furthermore, these requests contain important data about the sending device, such as the MAC address, which also allows our device to deduce the manufacturer and collect counts of distinct devices in the microcontroller's vicinity, and the Received Signal Strength (RSS) value of the received packet. Using this data, we will train a regression model on a server and deploy it to the microcontroller using TinyML. After deploying the model, the microcontroller will upload occupancy estimates every 15 minutes to a server and a frontend interface will allow the end user to view the current number of people as well as filter for the last 1, 6, and 24 hours of occupancy data.

This experimental data was collected in the university gym. The gym allowed a large open area which made it easy to collect ground truth occupancy levels for training the TinyML model. Furthermore, it is a popular attraction on campus with busy and non-busy peaks which provides a wide range of data points. While this current experimentation is limited to one location, the process of data collection, training, and deployment is the same for other attractions on campus, underscoring the system's adaptability and versatility.

2. Related Work

The exploration of occupancy detection using WiFi signals has been a subject of research interest across various studies and numerous methodologies and technologies have been discovered. This section reviews relevant literature pertaining to occupancy detection using WiFi signals in novel ways.

The authors of the work in [6] presented methods of filtering wifi signals so that accurate wifi probe request sniffing can be conducted in a given area. Their goal was to distinguish between bystanders of a demonstration and actual participants. In their study, they positioned two Raspberry Pi devices at two ends of an entrance and used these devices to conduct the distance and time based filtering. They found that the two filters did not accurately reflect the actual number of participants.They concluded that wifi-based probe request counting only represented a fraction of the actual number of participants in a demonstration.

In contrast, the authors in [4] explored the use of BLE beacons for predicting if a room is occupied or not and the count of people in a given room. They applied several different machine learning algorithms to find this relationship. They found that occupancy detection using this method was just as accurate as WiFi-based occupancy detection algorithms. In fact, they found that their detection algorithm was over 97% effective, suggesting that BLE is a viable alternative to WiFi sensing.

The authors in [12] extended occupancy detection and counting to both indoor and outdoor settings. They used WiFi pineapple devices and drones to capture WiFi signals and then employed a KNN algorithm for occupancy counting.

Researchers in [11] further demonstrated the efficacy of BLE beacons in tracking occupancy levels in both indoor and outdoor spaces, emphasizing BLE's advantages in terms of cost and energy efficiency. Their work highlighted the evolution of occupancy detection technologies and presented a practical application in a university context.

Newer methods, such as one proposed in [13], use WiFi Channel State Information (CSI) and Percentage of nonzero Elements (PEM) analysis for crowd counting. This study presented a departure from traditional RSSI filtering to underscore the potential of CSI for providing a more accurate representation of occupancy levels by considering the propagation characteristics of wireless signals.

Researchers of the work in [3] explored device-free crowd counting and localization using ESP32-S3 microcontrollers to collect CSI data. Their research indicates the utility of inexpensive microcontrollers in gathering accurate occupancy data. However, their study did show that there are limitations with using CSI data and device-free counting such as a lack of line-of-sight areas, suggesting that passive WiFi radar is better in areas that do not have line-of-sight between microcontrollers. They also constructed a machine-learning algorithm to test different feature sets that best predict occupancy levels.

The authors of [10] presented a novel approach to derandomizing MAC addresses which addresses any privacy-related data when capturing WiFi probe requests. They found that they were able to detect 96% of ungrouped devices and 70% of grouped devices (i.e. a cluster of devices). However, the paper noted that it lacked feature/parameter fine-tuning. The main contribution of this paper came from the implementation of a low-cost system for capturing WiFi probe requests and the development of a novel algorithm for MAC address derandomization on-device.

To further enhance occupancy estimation accuracy, researchers in [8] combined both WiFi and Bluetooth packet capture. They combined data from both efforts to address the device-to-people ratio discrepancy. They found that there is strong potential for improved occupancy predictions through multi-signal analysis.

The authors of [2] and [9] further contributed to occupancy detection research by presenting studies on wireless infrared sensors and machine learning algorithms. Their studies emphasize the diversity of approaches in the field, from direct sensor-based methods to sophisticated algorithmic analysis for occupancy prediction.

Lastly, the integration of TinyML for on-device processing, as discussed in [7] and [1], represents a significant advancement in privacy-preserving and low-latency occupancy detection. These studies highlight the landscape of occupancy sensing by leveraging machine learning within constrained hardware environments.

In summary, the related work demonstrates a broad spectrum of technologies and methods for occupancy detection, from traditional WiFi and emerging BLE signals to sophisticated machine learning algorithms. This research contributes to this growing field by proposing a novel system that harnesses WiFi probe requests for accurate occupancy prediction and uses a TinyML model to preserve privacy and keep the device low-cost.

3. Proposed System

This section describes the comprehensive system that was developed, including the WiFi sniffing, occupancy estimation, microcontroller design, backend server, and user-interface via a frontend. An ESP32-S3 from Seeed Studio was chosen as the microcontroller device due to its external antenna support, dual-core Xtensa processor, and ample RAM and storage capacity. It collects WiFi probe request data, including MAC address and RSSI data, and uses the deployed TinyML model to predict the level of occupancy. Then, it sends this estimate to a server along with a timestamp. The server stores this information and provides ways to query for it. Finally, there is a static frontend which queries the server for estimates and updates the graphs accordingly.

3.1. WiFi Sniffing

The system initiates with the ESP32-S3 syncing its clock via a Network Time Protocol (NTP) pool to ensure accurate time-keeping. Subsequently, the ESP32-S3 enters into promiscuous mode, a state designed for the passive collection of WiFi packets at a low energy. Every 15 minutes on the hour (e.g., 10:00, 10:15, 10:30), the ESP32-S3 will enter a sniffing mode where WiFi packet information is collected for a scan duration of 9 minutes. Every 3 minutes, the ESP32 will switch to a different wifi channel. It will cycle between channels 1, 6, and 11 as those are frequented channels for smartphones to emit probe requests on.

To ensure that the WiFi packet is a probe request, the system extracts the frame control header and checks to ensure that it is a probe request. It then extracts the Received Signal Strength (RSS) values and Organizational Unique Identifier (OUI) from the MAC address. The OUI is compared to a list of IEEE registered manufacturer OUIs. If the OUI is found to be in this list, it is considered a valid

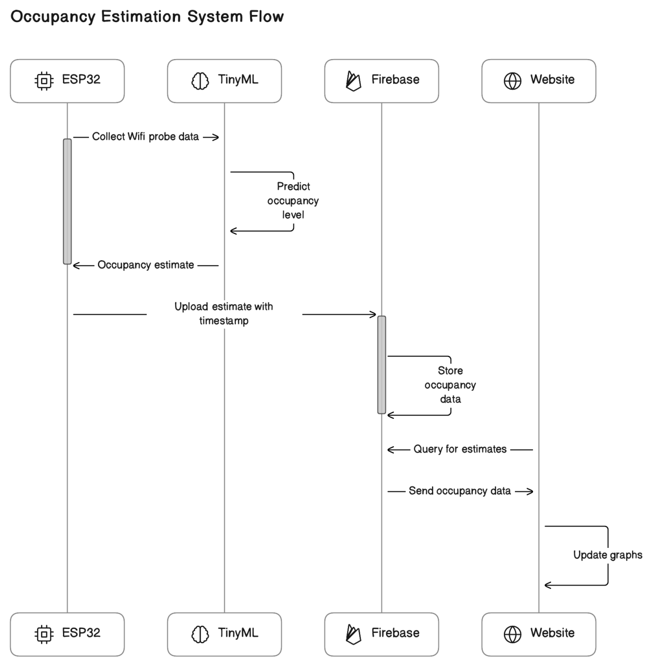

Figure 1. System diagram of the proposed system

MAC address. If it is not found in this list, it is considered to be a random MAC address. This helps distinguish between the counts of random and valid devices.

To further enhance data quality, RSSI values are organized into a structured bucket system, as proposed in [5]. We allocate 40 buckets for both valid and random MAC addresses, each corresponding to an RSSI range between -120 and -40 dBm, thereby enabling precise signal strength analysis according to the following rules:

1) For each device type (random or valid), create a bucket of m=40 counters. Each index in the bucket corresponds to a certain RSSI range. Break up the 40 counters into equal ranges between -120 and -40dBm. Thus, each counter spans 2dBm. For example, index 0 in the bucket contains a counter which increments for any RSSI values between -40dBm and -42dBm.
2) For each received RSSI value, map the RSSI value to the given index and increase all counters such that the $counter_i < RSSI_i$. For example, given an RSSI value of -78dBm which maps to index 19, all counters up to and including $counter_{19}$ should be incremented.

Concurrently, the system tracks 2 sets of data. One set tracks the unique valid MAC addresses, while the other set tracks the unique random MAC addresses. By the end of the scan, the sets should contain the number of unique random and valid MAC addresses discovered during the scan.

3.2. Occupancy Estimation

In order to be able to compress this model to usable C code, a Random Forest regression model was chosen as it can be easily ported to C code. Data was collected with n=150 samples and used to train a Random Forest regression model with k = 164 estimators. The data was split into training and testing data with a 70/30 split. After training this model, it was converted to C code and uploaded to the ESP32.

After data from the 9 minute scan was collected, it was fed into the TinyML model on-device and an estimate of the number of occupants in the given space was calculated.

3.3. Server and Frontend Interface

Occupancy estimates, accompanied by timestamps, are transmitted to a central server designed to store and manage this data efficiently. A static frontend interfaces queries the server, retrieving the latest occupancy information and dynamically updating visual representations such as graphs to reflect current and historical occupancy trends. This user-friendly dashboard enables end-users to interact with the data, facilitating informed decisions based on occupancy patterns.

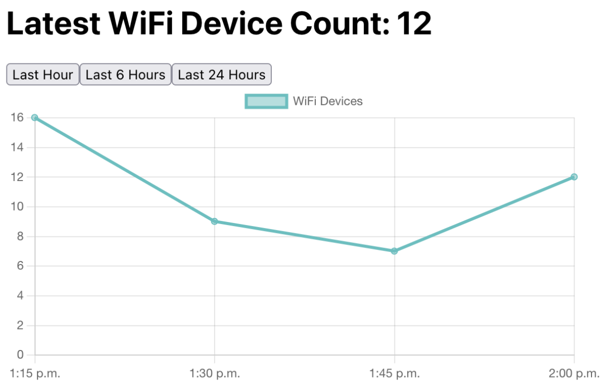
Latest WiFi Device Count: 12

Figure 2. Frontend interface of the graphs and current count

4. Results

To evaluate the system, data was collected in the campus gym. We measured the ground truth number of occupants on the first floor of the gym by counting each individual person every 9 minutes. Occupants were only counted if they were in the first floor of the gym for the duration of a scan in order to prevent the capture of all probe requests packets, which happen at up to 10 minute intervals. After 9 minutes, the ground truth number of occupants is entered into the ESP32 and all of the data is uploaded to the backend server for training purposes. This process was repeated 150 times for a total of 150 data points across different time periods and periods of occupancy to ensure holistic data collection.

After collecting this data, the data was split into a 70/30 train/test split. A scaler was not applied to the data before being trained using the Random Forest Regression model because it would have required a scaler with appropriate median and standard deviation values for each feature to also be embedded into the ESP32. After training the Random Forest Regression model, the Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE) were calculated to determine how well the data actually fits the model. The model takes into account the number of unique valid devices, the number of unique random devices, and the two RSS threshold buckets as features and outputs a single number which estimates the ground truth occupancy.

When running the model on a server, the MAE and RMSE are 1.743 and 2.523, respectively. When running the model on device and applying the testing data to get the RMSE and MAE values, there was a stark contrast in the RMSE and MAE of running the regression on-device versus on a server. Our RMSE and MAE on-device was 5.14 and 4.36, respectively, as reported in Table 1.

The discrepancy in performance metrics between the on-device and off-device model evaluations highlights a critical aspect of deploying machine learning models on low-power microcontrollers. Running the model on-device ensures privacy and reduces data transmission requirements but introduces constraints related to computational resources and model complexity. The increased MAE on-device compared to off-device can be attributed to these constraints, which may necessitate model simplification. Such simplifications can affect the model's accuracy, as seen in the reported MAE values.

Device type	**RMSE**	**MAE**
Server	2.523	1.743
Microcontroller	5.14	4.36

Table 1. RESULTS OF RUNNING RANDOM FOREST REGRESSION ON VS OFF DEVICE

5. Conclusion

In this paper, we introduced a system that utilizes WiFi probe requests to estimate occupancy levels in settings such as gyms or dining halls on campus. Understanding occupancy levels is crucial for both students, who can plan their activities to avoid crowded times, and administrators, who can optimize resource allocation based on actual usage. This proposed system employs an ESP32-S3 microcontroller to collect WiFi probe requests emitted by devices like smartphones and laptops. The data gathered is then analyzed using a Random Forest regression model running directly on the ESP32, enabling the prediction of occupancy levels in a gym environment. The system demonstrated its capability to accurately estimate the number of people present, achieving a Mean Absolute Error (MAE) of 1.743 when the model was evaluated off-device and 4.36 on-device.

Future work will focus on addressing the discrepancy between on and off-device predictions by exploring advanced model compression techniques, such as optimizing a neural network architecture for the ESP32 and fine-tuning the model to better accommodate the hardware constraints without significantly impacting accuracy.

References

[1] Maria Francesca Alati et al. "Time series analysis for temperature forecasting using TinyML". In: *2022 IEEE 19th Annual Consumer Communications and Networking Conference (CCNC)*. 2022, pp. 691–694. DOI: 10.1109/CCNC49033.2022.9700573.

[2] Claudia Chiţu et al. "Wireless system for occupancy modelling and prediction in smart buildings". In: *2017 25th Mediterranean Conference on Control and Automation (MED)*. 2017, pp. 1094–1099. DOI: 10.1109/MED.2017.7984264.

[3] Hyuckjin Choi et al. "Wi-CaL: WiFi Sensing and Machine Learning Based Device-Free Crowd Counting and Localization". In: *IEEE Access* 10 (2022), pp. 24395–24410. DOI: 10.1109/ACCESS.2022.3155812.

[4] Florenc Demrozi et al. "Estimating Indoor Occupancy Through Low-Cost BLE Devices". In: *IEEE Sensors Journal* 21.15 (2021). DOI: 10.1109/jsen.2021.3080632. URL: http://dx.doi.org/10.1109/JSEN.2021.3080632.

[5] Paolo Galluzzi et al. "Occupancy Estimation Using Low-Cost Wi-Fi Sniffers". In: *CoRR* abs/1905.06809 (2019). arXiv: 1905.06809. URL: http://arxiv.org/abs/1905.06809.

[6] C. Groba. "Demonstrations and people-counting based on Wifi probe requests". In: *2019 IEEE 5th World Forum on Internet of Things (WF-IoT)*. ID: 1. 2019, pp. 596–600. ISBN: null. DOI: 10.1109/WF-IoT.2019.8767208.

[7] Piyapat Leeraksakiat and Wanchalerm Pora. "Occupancy Forecasting using LSTM Neural Network and Transfer Learning". In: *2020 17th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)*. 2020, pp. 470–473. DOI: 10.1109/ECTI-CON49241.2020.9158103.

[8] Edoardo Longo, Alessandro E. C. Redondi, and Matteo Cesana. "Accurate occupancy estimation with WiFi and bluetooth/BLE packet capture". In: *Computer Networks* 163 (2019), p. 106876. DOI: 10.1016/j.comnet.2019.106876. URL: https://www.sciencedirect.com/science/article/pii/S1389128618313045.

[9] Jesica Elizabeth Marchelina et al. "Two-Stages Occupancy Number Detection Based on Indoor Environment Attributes By Utilizing Machine Learning Algorithm". In: *2019 International Conference on Fuzzy Theory and Its Applications (iFUZZY)*. 2019, pp. 38–43. DOI: 10.1109/iFUZZY46984.2019.9066241.

[10] A. Simončič et al. “Non-Intrusive Privacy-Preserving Approach for Presence Monitoring Based on WiFi Probe Requests”. In: *Sensors* 23.5 (2023), p. 2588. DOI: 10.3390/s23052588. URL: https://doi.org/10.3390/s23052588.

[11] Montserrat Mateos Sánchez et al. “A Tool to Calculate the Level of Occupancy in Indoor and Outdoor Spaces Using BLE and Open Data to Be Published in Real-Time”. In: *Sensors* 20.14 (2020). 32674470. DOI: 10.3390/s20143916. URL: https://www.mdpi.com/1424-8220/20/14/3916.

[12] Edwin George Vattapparamban. “People Counting and occupancy Monitoring using WiFi Probe Requests and Unmanned Aerial Vehicles”. In: Florida International University (2016). DOI: 10.25148/etd.FIDC000246.

[13] W. Xi et al. “Electronic frog eye: Counting crowd using WiFi”. In: *IEEE INFOCOM 2014 - IEEE Conference on Computer Communications*. Toronto, ON, Canada, 2014, pp. 361–369. DOI: 10.1109/INFOCOM.2014.6847958.