BLE-based Room Occupancy Detection with Raspberry Pi Pico 2W

Ainsley Jong Zhe
School of Electronics and Computer Science
University of Southampton, United Kingdom
azj1n21@soton.ac.uk

***Abstract*—This project showcases a low-cost, wireless presence detection system using three Raspberry Pi Pico 2W boards (RP2350 model). It consists of two "scanner" nodes, one in the kitchen and another in the bedroom, where it continuously scans for a BLE beacon. When the badge's RSSI exceeds -50dBm, the scanner publishes a retained MQTT over Wi-Fi message ('occupied' or 'vacant') to distinct topics for the kitchen and living room leveraging the built-in Wi-Fi capability of the Pico 2W. This proof-of-concept demonstrates a scalable, BLE-and-MQTT approach to real-time home automation occupancy monitoring without additional hardware required.**

Keywords—Bluetooth Low Energy (BLE), Received Signal Strength Indicator (RSSI), Message Queuing Telemetry Transport (MQTT), Raspberry Pi Pico 2W, occupancy detection, wireless sensor network

I. Introduction

Smart home applications increasingly rely on detecting whether a person is present in a room to optimise comfort, security, and energy use. In this project, the inexpensive Raspberry Pi Pico 2W boards (RP2350) are used as both a BLE beacon detector and presence scanners, combined with an MQTT broker to aggregate and distribute occupancy state of a certain room.

The system uses a single beacon that advertises its ID over Bluetooth Low Energy (BLE) at 100ms intervals. Then, two Pico 2W scanners, one in the kitchen and the other in the bedroom, will measure the beacon's Received Signal Strength Indicator (RSSI). Each scanner applies a threshold (-50dBm) to detect whether the person, holding the beacon is present in the room. Both the scanners continuously publish a Message Queuing Telemetry Transport (MQTT) message to home/<room>/occupancy at every 5 second intervals and publish that the person is present once the RSSI strength is greater than the threshold.

This project mainly displays the RP2350's dual-core performance, integrated with Wi-Fi and BLE radio on top of its rich I/O makes it ideal for distributed presence-based systems.

II. Related Work

Various techniques of indoor occupancy detection such as passive infrared (PIR) sensors and ultrasonic sensors to camera-based systems and many more have been explored over the recent years. However, these approaches continues to prove to involve higher costs, privacy concerns, or increased complexity.

Therefore, this is where Bluetooth Low Energy (BLE) – based presence detection using RSSI has been inspired and gained popularity as a lightweight and cost-effective alternative as shown in [1]. Even though in [2], it shows that the accuracy of RSSI may vary due to environmental noise and signal reflections, it also provides some solutions such as Deep-BLE.

Unlike most prior implementations, this project demonstrates a simplified, MQTT-based architecture using Raspberry Pi Pico 2W boards for both beaconing and scanning. It eliminates the need for additional hardware or any external sensors and demonstrates real-time occupancy detection through a distributed network of scanner nodes.

III. Methodology

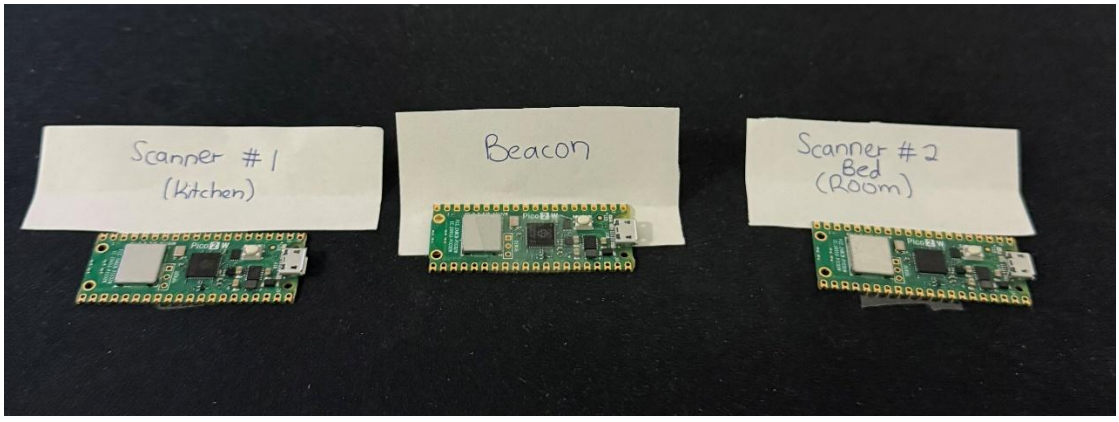

Fig. 1. Setup of naming convention for each RP2350 board

This section describes the hardware setup, networking configuration, and occupancy-detection logic that combines to implement the BLE-and-MQTT presence system. *Figure 1* shows the naming conventions for each RP2350 boards to properly identify one another.

A. Experimental Setup

Hardware:

- **Board:** Three Raspberry Pi Pico 2W boards (RP2350)

Software:

- **OS:** Windows 11
- **Firmware:** MicroPython v.1.25.0
- **IDE:** Thonny IDE v4.1.7
- **Libraries used:**
  - `ubluetooth` (BLE interface)
  - `umqtt.simple` (MQTT communication)
  - `micropython` (declare immutable constants)
  - `machine`, `time`, `network` (hardware control and timing)

B. BLE Beacon Firmware

The beacon node broadcasts a 100ms interval advertisement containing a complete local name ("Beacon-1"). A simple custom advertising payload builder `ble_helper.advertising_payload( )` is used to encode the device name.

The implementation can be seen under the scripts `ble_helper.py` and `beacon.py`. On the device itself, rename the main script of `beacon.py` to `main.py` so that it runs automatically at startup.

C. Scanner Firmware and RSSI Filtering

Each scanner executes a continuous BLE scan (30ms window + 30ms interval) using Interrupt Request (IRQ) callbacks. On every advertisement result, the `bt_irq` handler decodes the name and measures the RSSI value. An in-memory of 'present' beacons are maintained and the RSSI threshold value of -50dBm is applied to determine whether the person is present in the room. If a person is present in the room, the LED will light up and vice versa where the corresponding message will be published via MQTT. The state is published at every 5 second interval to prevent rapid MQTT message publishing.

The implementation can be seen under the scripts `ble_helper.py` and `scanner.py`. On the device itself, rename the main script of `scanner.py` to `main.py` so that it runs automatically at startup.

D. Wi-Fi and MQTT Integration

Upon reboot, each scanner connects to the Wi-Fi using the `network.WLAN()` function with its network credentials. Once connected, it incorporates a MQTT client using HiveMQ's public broker at `broker.hivemq.com:1883` with a unique `client_id` of `rp2350_scanner_1` and `rp2350_scanner_2` respectively. Subsequent occupancy updates – for example, "occupied" or "vacant" – are published to their topics at `home/kitchen/occupancy` and `home/bedroom/occupancy` respectively. By setting the retain flag True, any new subscriber immediately receives the latest room state without waiting for the next publish interval and can act accordingly.

IV. RESULTS

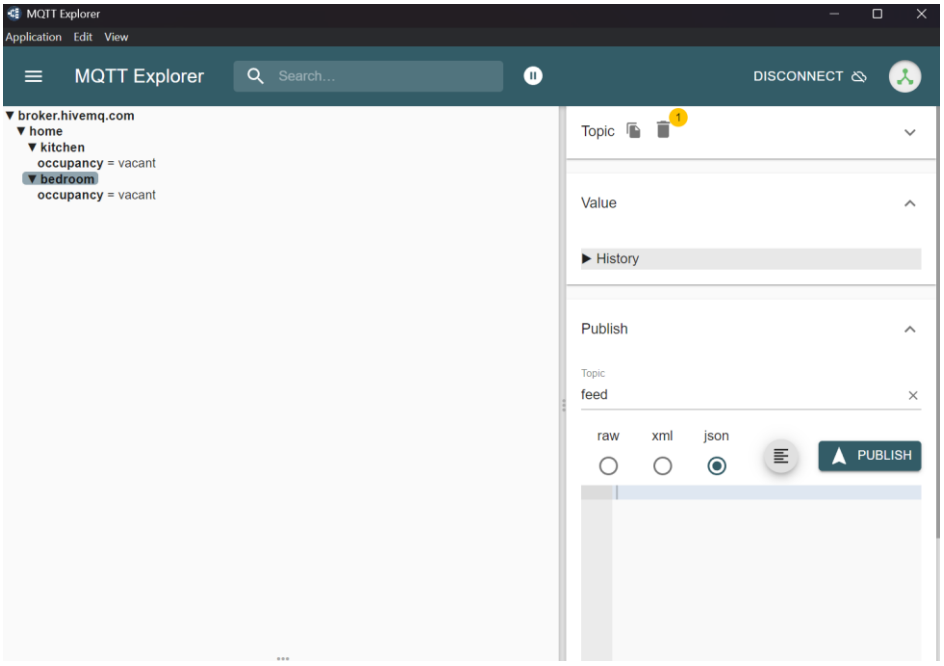

Fig. 2. Figure of MQTT Explorer Output

A. *MQTT Explorer Output*

MQTT Explorer was used to subscribe to home/#/occupancy states to check the room occupancy detection. MQTT messages were successfully published at 5-second intervals with no message loss as shown in *Figure 2*. New subscribers to the broker immediately received the latest occupancy status due to the retained message flag.

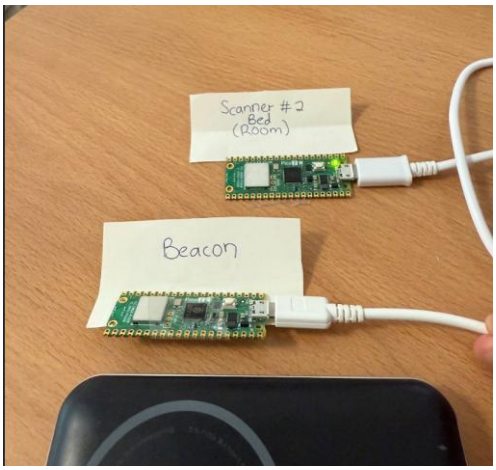

Fig. 3. Scanner node #2 lights up when the beacon is detected

B. *Visual Output*

Figure 3 illustrates the hardware setup during a test scenario where the BLE beacon is placed near Scanner #2, which is located in the bedroom. The LED of the scanner node lights up, visually confirming that the RSSI threshold condition (-50dBm) was met and the beacon was detected within range. This LED activation serves as a real-time, physical indicator of room occupancy detection which aligns with the corresponding MQTT message being published to its topic home/bedroom/occupancy as "occupied". The LED lights on or off according to the beacon movement, which proves that the visual feedback system responded accurately to presence changes.

C. *Occupancy Detection Accuracy*

Out of 15 room transitions, the system correctly detected the beacon's presence in the intended room 13 times which gives an accuracy of 86.67%. The false positives occurred when the user was in one room but both scanner nodes detected the BLE beacon. This proves that using the RSSI strength proves to not be fairly reliable and accurate.

V. Discussion and Future Works

A. *Limitations Faced*

1. Network Constraints

The Wi-Fi being used is an enterprise network where these networks often enforce client isolation. This prevents peer-to-peer TCP connections, where devices connecting to the same SSID – such as the HomeAssistant server running on Linux and the mobile app – are unable to communicate directly, preventing successful pairing and local control.

2. RSSI Variability and Thresholding

The accuracy of using BLE beacons is severely limited due to the large variance in RSSI as shown in [2] where it may cause fluctuations due to multiple reasons. A fixed -50dBm threshold could possibly still yield false results (e.g. where the beacon is detected present when it is in the adjacent room or detected absent when it is not detected in a bigger room).

3. Security of Public MQTT Broker

Leveraging HiveMQ's public broker enables rapid prototyping and testing without any credentials or usage of TLS. This leaves the messages open to interception where in a real home-automation deployment, an adversary on the same network could eavesdrop on presence data or publish false messages.

4. ESPHome / Home Assistant Integration

Integration of ESPHome is currently supported on the RP2040 (Pico W), but not yet available for the RP2350 (Pico 2W) chip as shown in [3]. Therefore, this proof of concept resulted in not being able to "plugged into" Home Assistant via its ESPHome add-on, which would have provided automatic device discovery and notification support.

B. *Potential Extensions*

1. Future ESPHome and Home Assistant Integration

Once ESPHome fully supports the RP2350 chip in the near future, full integration with Home Assistant can finally be implemented. This would enable seamless device discovery, enhanced automation capabilities, and real-time notifications, making the system more functional in real-life scenarios.

2. Multi-modal Sensor Integration

In addition to BLE-based occupancy, additional sensors such as DHT22 or BME280 for temperature and humidity data alongside the Home Assistant could trigger automations accordingly to provide comfort and energy-use optimization.

3. Secure MQTT Deployment

Migrate from a public broker to a cloud-hosted MQTT service with TLS such as HiveMQ Cloud to ensure secure and encrypted communication between devices. This would enhance data privacy and align the system with the best practices for IoT security.

VI. Conclusion

A cost-effective, wireless room-occupancy detection system has been implemented using three Raspberry Pi Pico 2W (RP2350) boards where one acts as a BLE beacon detector and the other two are scanner nodes that infer presence by applying an RSSI threshold of -50dBm and publish retained occupancy states via MQTT to enable remote monitoring.

While the current implementation demonstrates the feasibility of low-cost BLE-presence detection system with minimal hardware, there exists several enhancements that could be implemented in the near future for system completion. These include the integration of additional environmental sensors to enable context-aware automation, transitioning to a secure MQTT broker that provides TLS support, and full ESPHome integration once RP2350 support becomes available.

Acknowledgment

I would like to express my sincere gratitude to the lecturers of the COMP3210 module for their continuous guidance and support throughout this project, making this an enjoyable learning experience.

References

[1] F. Demrozi, C. Turetta, F. Chiarani, P. H. Kindt and G. Pravadelli, "Estimating Indoor Occupancy Through Low-Cost BLE Devices," IEEE Explore, 2021. Available: http://dx.doi.org/10.1109/JSEN.2021.3080632

[2] H. Agarwal, N. Sanghvi, V. Roy and K. Kitani, "DeepBLE: Generalizing RSSI-based Localization Across Different Devices," arXiv, 2021. Available: https://arxiv.org/abs/2103.00252

[3] ESPHome, "ESPHome Docs – ESPHome," [Online] Available: https://esphome.io/components/