Designing a Testbed for Human Activity Recognition Using Multiple ESP32 Microcontrollers with Wi-Fi and Serial Data Transfer

Jelle Veldmaat, j.p.veldmaat@student.utwente.nl, University of Twente, The Netherlands

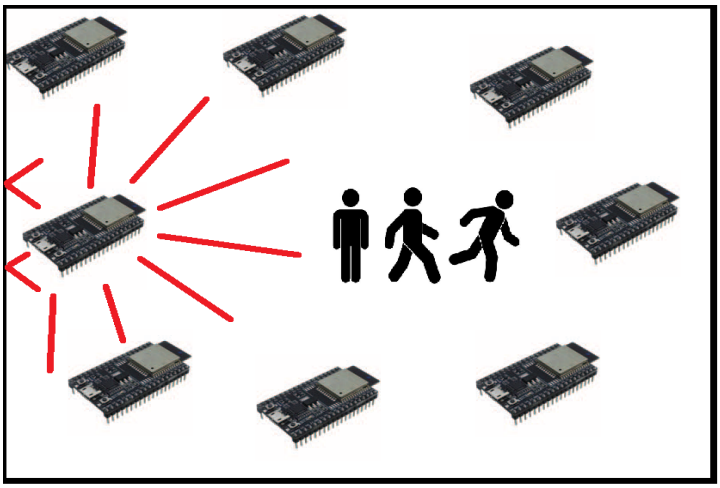

Figure 1. Multiple ESP32 microcontrollers used for HAR.

ABSTRACT

Significant advancements have been made in leveraging Wi-Fi signals for Human Activity Recognition (HAR). Most notably with the utilization of Channel State Information (CSI). Nevertheless, the current landscape lacks a straightforward approach to conduct HAR experiments involving multiple ESP32 microcontrollers. This research paper presents a notable contribution by introducing a well-designed testbed for HAR, employing multiple ESP32 microcontrollers. The study demonstrates the feasibility of transmitting CSI data over Wi-Fi without detrimental consequences, thus offering enhanced convenience and scalability compared to conventional serial cable connections.

1. INTRODUCTION

1.1 Context and motivation

Human Activity Recognition (HAR) is an area of research that focuses on the automatic identification and classification of human activities using data from various sensors [1]. By utilizing advanced machine learning techniques and algorithms, HAR can offer valuable insights into understanding human behavior and support various applications such as health monitoring, smart devices, surveillance systems and augmented reality [2].

The goal of HAR is to categorize individuals' activities based on collected sensor data, such as accelerometers, gyroscopes, heart rate monitors, GPS, and even environmental data [3]. By analyzing and interpreting patterns in this data, HAR systems can identify various activities including walking, running, stair climbing, sitting, standing and others [4].

Wi-Fi-based sensors for HAR have demonstrated significant promise, particularly those that emphasize the utilization of Channel State Information (CSI) rather than solely relying on Received Signal Strength Indicator (RSSI) measurements [4]. Devices and routers that make use of Wi-Fi are numerous in today's homes, it offers advantages in terms of accessibility and convenience [5]. HAR utilizing Wi-Fi technology provides the advantage of contactless monitoring, as illustrated in Figure 1. This approach eliminates the requirement for participants to wear dedicated devices [6], which is particularly beneficial in scenarios involving elderly individuals who may occasionally forget to wear such devices. Moreover, it enhances privacy by relying on CSI to predict activities, avoiding invasive techniques such as camera surveillance. This technology finds valuable applications in various domains, including elderly homes where HAR can be utilized for fall detection [7], as well as in security systems [7].

Numerous research studies have indicated the feasibility of HAR through the utilization of CSI [8]. Historically, these experiments

have heavily relied on the utilization of an Intel 5300 Network Interface Card (NIC), an outdated device that is known for its unreliability and being over a decade old [9]. However, recent advancements have demonstrated the capability to capture CSI data using ESP32 microcontrollers [10]. The results obtained for HAR using ESP32 have been comparable to those achieved with the NIC. The compact size and affordability makes it an ideal choice for large-scale deployment [10]. Given that the position of these devices significantly impacts HAR outcomes, a greater number of devices can yield improved results [11]. Nevertheless, establishing a testbed for multiple ESP32s has posed challenges due to the management of CSI data Coming from these devices.

1.2 Specific problem

The challenge resides in the consolidation of data from multiple devices into a central point for processing. Presently, there is a lack of an optimal solution, especially when the devices are dispersed throughout a room. The conventional approach of establishing serial cable connections to a central point becomes impractical in such scenarios. Additionally, conducting experiments with these devices poses difficulties in terms of managing labeling, as well as controlling start and stop commands from a centralized location. Effectively managing the data becomes paramount to ensure that the respective CSI data corresponds accurately to the designated experiment.

1.3 Research questions

Considering the problem statement, the following main research question is formulated:

“What is an efficient and scalable approach to construct a testbed for HAR using multiple ESP32 microcontrollers?”

With the following sub questions:

1. “How can the testbed be designed to enable seamless experimentation, including labeling, synchronization and control of multiple ESP32 devices?”
2. “How can the ESP32 microcontrollers be interconnected to create a scalable network for data collection?”
3. “What are the potential challenges and limitations associated with building and using a testbed for HAR with multiple ESP32 microcontrollers and how can they be mitigated?”

1.4 approach and structure

In terms of the approach, the first step involves addressing research question 1. In order to facilitate the management of multiple serial connections, a USB hub will be implemented. Subsequently, efforts will be directed towards optimizing the experimentation process. This will involve the establishment of dedicated folders for each experiment, as well as the development of methods for initiating, terminating and labeling experiments. The collected CSI data will be stored in CSV format, with file names that correspond to the MAC addresses of the respective ESP32 devices.

Using serial cable connections present limitations. Consequently, the focus will be shifted towards investigating transmitting CSI data over Wi-Fi networks. However, to validate the effectiveness of the initial approach involving serial cable connections and the subsequent approach utilizing Wi-Fi, a series of experiments will be conducted. These experiments aim to amplify the packet request frequency, thereby augmenting the amount of collected Channel State Information (CSI) data. Additionally, the number of ESP32 nodes will be increased to assess the limitations associated with employing multiple nodes. In conclusion, the viability of the new method will be assessed and a list of the necessary components will be provided.

2. REQUIREMENTS

Considering the research questions the following features are required:

1. Usable for HAR
2. Scalable with multiple nodes
3. Controlled data collection

3. EXISTING SOLUTIONS

3.1 Current tool overview

The “ESP32 CSI Toolkit” [10], an established tool, provides the means to extract CSI data from ESP32 microcontrollers by leveraging various modes of operation. These modes encompass an active access point, active station, and passive mode. In the active station mode, the ESP32 initiates pings to the access point node, generating the necessary CSI data. Conversely, the passive mode operates by monitoring network traffic and collecting CSI data from it. It is important to note that the tool functions using Wi-Fi technology within the 2.4 GHz frequency band, which is known to be more congested compared to the 5 GHz frequency band. During typical collection not only CSI data is stored, but also the following:

- Type
- Role
- MAC Address
- RSSI
- Rate
- Signal Mode
- MCS
- Bandwidth
- Smoothing
- Not Sounding
- Aggregation
- STBC
- FEC Coding
- SGI
- Noise Floor
- A-MPDU Count
- Channel
- Secondary Channel
- Local Timestamp
- Antenna
- Signal Length
- Receiver State
- Real-Time Set
- Real Timestamp
- Length
- CSI Data

Two attributes are of particular interest for this research: The first attribute is the "role" which represents the current mode of the ESP32 microcontroller. This attribute provides information about the operational state of the device during data collection. The second attribute is the MAC (Media Access Control) address,

which indicates the original sender of the Wi-Fi data. However, it should be noted that the MAC address of the ESP32 responsible for collecting the data is not included in the dataset, which is an important consideration. Also note that the size of this data is typically about 454 bytes.

This tool enables the transfer of CSI data from multiple devices to a centralized collection point via a serial cable. It leverages the timestamps generated by the data-collecting device to ensure accurate synchronization. Additionally, the tool provides the capability to store the collected data on an SD card, if present. While experiments utilizing multiple ESP32 microcontrollers have demonstrated positive outcomes [10], these studies did not address the collection and management of incoming CSI data, particularly when dealing with data from multiple devices simultaneously.

3.2 Multiple nodes

Currently, the focus lies on utilizing one ESP32 as an access point and only one as a station [11], overlooking the challenges of managing data from multiple devices. Furthermore, the passive mode has shown inconsistency in this research, with a notable failure to achieve the 34.5% data collected compared to an active mode [10]. The limitation of the passive mode poses a challenge when utilizing multiple nodes, as they cannot effectively contribute useful CSI data in this mode. Consequently, the network becomes more congested due to the increased frequency of pings from the nodes in active mode.

Although the unmodified tool can be used with serial connections to all ESP32 devices, this approach is not scalable as it involves the inconvenience of running serial cables everywhere. Additionally, managing the substantial amount of data generated for HAR purposes becomes impractical without a more efficient solution.

4. NEW ARCHITECTURE

4.1 ESP32 limitations

Since the ESP32 is a fairly cheap and small device certain significant limitations exist:

1. Limited processing power: The ESP32 has a relatively modest processing power compared to more powerful microcontrollers or processors [12].
2. Limited memory: The ESP32 has limited internal memory available for program storage and data handling [13]. This can pose constraints when developing complex applications or handling large datasets.
3. Limited compatibility with certain software or libraries: Since the original “ESP32 CSI Toolkit” [14] that is built on the ESP-IDF development framework [15] instead of the more popular arduino framework [16]. Many of the existing libraries are not compatible.

4.2 Facilitating HAR experiments

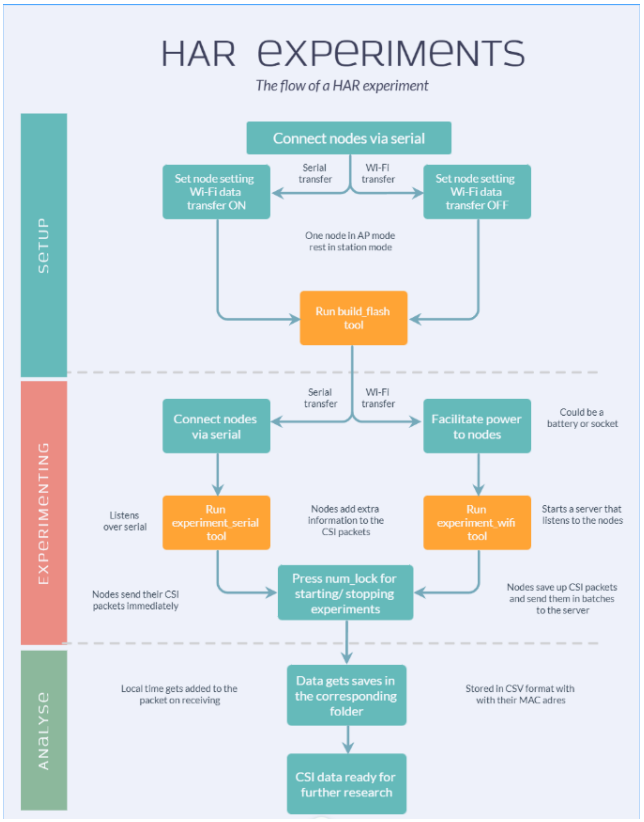

Figure 2. Flow diagram for conducting experiments

In order to optimize HAR experiments utilizing multiple ESP32 microcontrollers, a flowchart (see Figure 2) was designed. Supporting this, a custom Python tool [17] was developed to enhance the management of these experiments. This tool facilitates the initiation and termination of each experiment, ensuring precise data collection from each ESP32 device while effectively associating the collected data with its corresponding experiment.

4.3 Wi-Fi data transfer

To facilitate the transfer of CSI data from ESP32 devices to a central laptop, the use of Wi-Fi connectivity was implemented as an alternative to relying on serial cables. This necessitated modifications to be implemented to the original "ESP32 CSI Toolkit" tool [14] enabling this functionality.

In addition to the attributes mentioned in Section 3.1, further attributes were incorporated to support the Wi-Fi transfer process. While the existing CSI data packet only included the MAC address of the sender, typically the ESP32 acting as an access point, a modification was made to include the MAC address of the receiving node as well. This was necessary due to the absence of a serial connection and the inability to identify the source of each signal accurately. Incorporating the MAC of the receiving node allowed for proper handling and management of the data on the laptop.

To ensure data integrity, a numbering system was implemented within each CSI data packet on the ESP32 devices. This numerical identifier incrementally increases with each collected CSI data, enabling the detection of any potential data loss during the transfer process.

The size of the newly introduced CSI data packet is typically 551 bytes, representing an increase of 97 bytes compared to the previous version, which had a size of 454 bytes. This increase is due to the inclusion of additional information in the packets. Considering the larger packet size, it is not desirable to send each CSI data packet individually to the laptop due to the potential for significant network congestion. Instead, the CSI data is accumulated in the ESP32 and transmitted in larger batches.

Through experimentation, it has been observed that the ESP32 can store approximately 100 of these data packets, resulting in a total data size of around 55,100 bytes (100 * 551). However, despite the observation that the ESP32 still has available memory, the ESP32 crashes at that point. This behavior could be attributed to the ESP32 detecting a steady increase in memory usage and interpreting it as a memory leak. While this issue has not been addressed in this research, a predetermined threshold has been set to trigger the transmission of data when the batch size reaches 40, with the maximum batch size limited to 80.

It is worth noting when increasing the frequency issues may arise where the ESP32 encounters challenges in transmitting previously collected CSI data, resulting in the loss of all data within the batch. A more sophisticated approach would involve overwriting the most recently collected data instead of discarding the entire batch. However, in the current design, this consideration has been omitted since, in the context of HAR with low frequencies, it is unlikely to reach this frequency limit in practice.

However, it's important to note that the timing in the CSI data is not synchronized across all nodes, resulting in variations. To address this, the local time of the laptop is added to the data. Although the timing may not be perfect due to the data being sent in batches, the incorporation of the node's time helps improve the accuracy and provides a sense of the actual recording time of the data.

Consequently, a laptop needs to be connected to the ESP32 acting as an access point, establishing a server that listens to the POST requests sent by the nodes containing their respective batches of CSI data.

4.5 Required components

The introduction of Wi-Fi CSI data transfer eliminates the requirement for a serial cable and port for each node. A serial cable is only needed to set up the testbed for the initial flashing of the ESP32's only the following components are needed:

Hardware:

- 2 or more ESP32 microcontrollers
- 1 USB A male to micro USB male cable
- 1 Laptop with Wi-Fi and a serial port

Software:

- Custom Python tool for HAR experiments [17]
- ESP32 CSI toolkit with Wi-Fi CSI data transfer [18]
- ESP-IDF version 4.3 [15]

5. VERIFICATION

To validate the effectiveness of the new architecture that is utilizing Wi-Fi data transfer, a series of experiments were conducted (see Appendix A). The primary objective was to assess whether the Wi-Fi-based data transfer method performed comparably to the conventional serial data transfer approach. Additionally, the experiments aimed to determine the feasibility of multiple nodes simultaneously requesting packets from the access point and the maximum achievable frequency at which this is possible.

The validation experiments for the proposed model were conducted within an office room (see Appendix B), where the ESP32 nodes were distributed throughout the space. The experiments included both serial and Wi-Fi CSI data transfer methods, involving configurations with 1, 2 and 9 station nodes and 1 access point node. Also a laptop was present for data collection and experiment management with the use of the Python tool [17]. It is important to note that for HAR purposes, a frequency of CSI information as low as 10 Hz is typically sufficient [19].

5.1 CSI data frequency

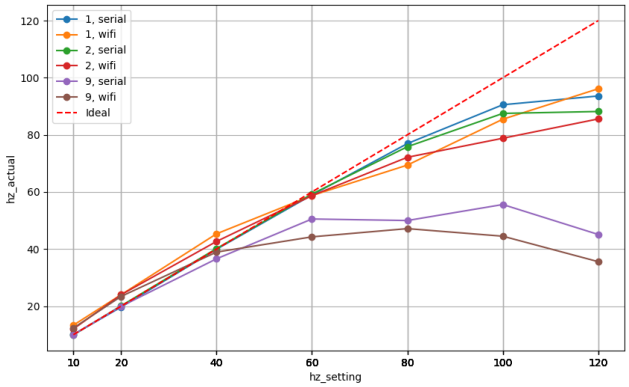

Figure 3. Comparing the actual frequency with the set frequency.

The frequency at which CSI data is collected serves as a critical parameter for achieving optimal results in HAR [7]. As illustrated in Figure 3, an interesting observation reveals that the actual frequency starts to decline as the frequency setting increases. This phenomenon can be attributed to network congestion within the Wi-Fi network. Notably, the impact is most pronounced when there are 9 station nodes involved in the experiments. Another noteworthy finding is that experiments employing Wi-Fi for data transfer exhibit a higher actual frequency than the set frequency. This occurrence can be attributed to the additional POST requests transmitted to the laptop via the ESP32 acting as the access point. Consequently, the access point's response to the POST request allows for the collection of that CSI data as well. It is important to mention that each POST request contains a minimum batch of 40 CSI data. Thus, the ESP32 might already have some CSI data

stored before the experiment commences. Currently, no action is taken with regard to this fact.

Nevertheless, the ability of the new Wi-Fi-based solution to handle 9 nodes at a frequency of 10 Hz which meets the requirements for HAR [19] without major deviations even if the frequency is doubled confirms its validity.

5.2 Time between CSI data collected

Similar to the impact of frequency, maintaining a consistent time interval between CSI data collections contributes to the effectiveness of HAR [19]. It is important to note that this consideration differs from the frequency comparison discussed in section 5.1 but has similarities.

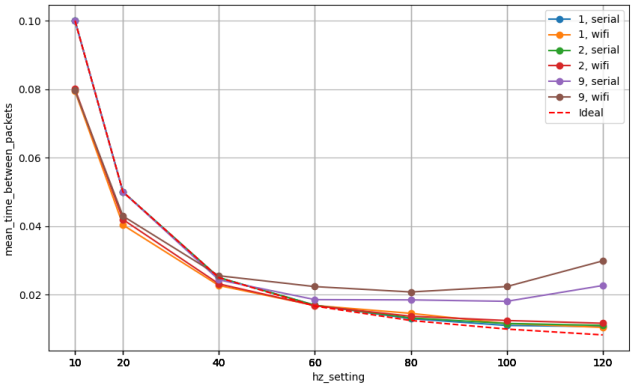

Figure 4. The mean time between CSI collections.

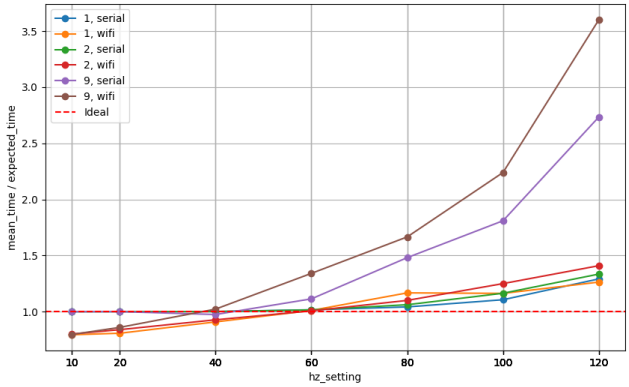

Figure 5. Ratio of mean time and expected time between CSI collection

Figure 4 illustrates that when utilizing higher frequency settings with 9 nodes, both the Wi-Fi and serial-based solutions exhibit an increase in the mean time between data packets, resulting in a lower sampling rate than the frequency set. To provide a more comprehensive representation of the deviation in Figure 5 The ratio of the expected time to the actual time is represented. Notably, significant deviations become noticeable at approximately 40 Hz, which, considering its higher than the required 10 Hz for HAR, makes it well-suited for the intended application [19].

5.3 Variation in time of CSI collection

Consistent time intervals between CSI packet collections in the proposed solution contribute to its predictability, thereby assisting the HAR model.

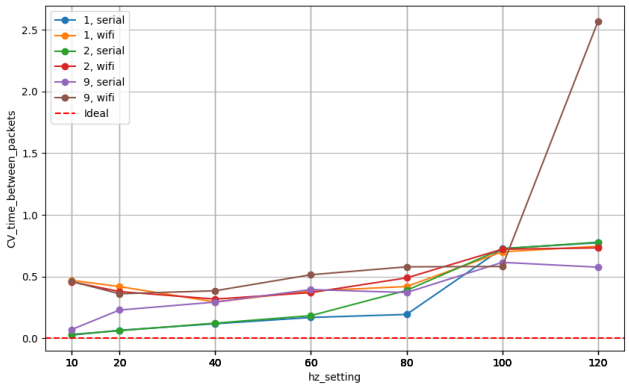

Figure 6. Coefficient of variation of the time between CSI collections

The coefficient of variation (CV) serves as a metric for evaluating the consistency of time intervals between CSI data collection. This metric plays a crucial role in relation to timing data, as it indicates the stability of the network and, consequently, the consistency of the results obtained from the HAR experiments. A more consistent timing between packets leads to an evenly distributed amount of data throughout the duration of the experiment. Figure 6 illustrates that the 3 experiments using serial connections show a more predictable time between CSI data collections. This finding aligns with expectations, as these nodes do not rely on Wi-Fi for transmitting their CSI data. A CV value of 0.5 indicates that the standard deviation is half as large as the mean. Given that the mean time is either lower or equal to the expected time until 40 Hz (Figure 5), this is not a significant issue since we have more CSI data than we need in these situations.

5.4 CSI data lost

By incorporating incremental numbering in the CSI data packet, we have created a mechanism to identify missing numbers, thereby detecting instances where CSI data generated by the ESP32 is absent at the data collection point.

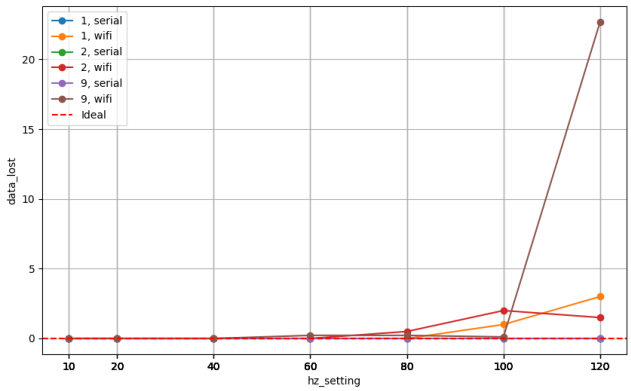

Figure 7. Amount of data lost during transmission in a 15 seconds experiment

As seen in Figure 7 there is only data loss during transmission over Wi-Fi. This is likely because of network congestion and that the nodes are unable to get rid of their CSI data batches with POST requests. Because of the limited memory of the ESP32 CSI data batches will be dropped.

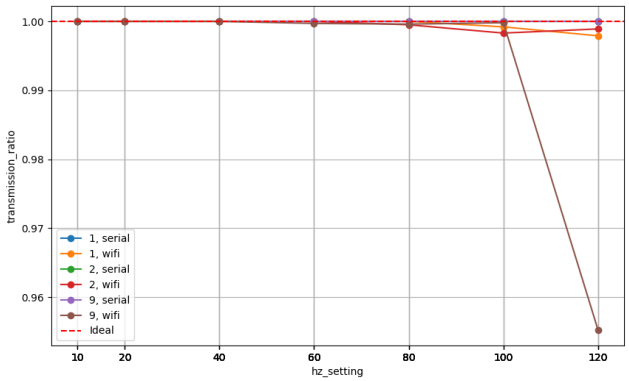

Figure 8. Successful transmission ratio

To provide a contextual understanding of the data loss, Figure 8 presents the ratio of transmitted data. This highlights that even with relatively high frequency settings, the ratio of data lost remains minimal. This observation underscores the robustness of the system, as the impact of data loss on the overall data collection process with ordinary frequencies is negligible.

5.4 Comparison to existing solutions

In the previous subsections, we have showcased the successful integration of Wi-Fi and serial data transfer methods utilizing multiple ESP32 microcontrollers for HAR experiments. The management and facilitation of data from multiple nodes were made possible by the Python tool. Nevertheless, it is important to note that the Wi-Fi solution stands out for its exceptional scalability and user-friendly nature, eliminating the requirement for additional cables when incorporating additional nodes. Moreover, the ESP32's low power consumption enables node connectivity to a battery, further improving its convenience.

6. CONCLUSION

This paper presents an effective solution on how to build a testbed using multiple ESP32 microcontrollers using Wi-Fi for CSI data transfer.

6.1 Research question 1

“How can the testbed be designed to enable seamless experimentation, including labeling, synchronization and control of multiple ESP32 devices?”

The proposed solution incorporates several key components. Firstly, the nodes connected to the access point synchronize their time and include this information in their CSI data. Additionally, a central location is designated for data collection, where the received data from multiple nodes is labeled based on its origin and stored for the ongoing HAR experiment. Importantly, this solution is compatible with both serial and Wi-Fi CSI data transfer methods, ensuring flexibility with different connectivity options.

6.2 Research question 2

“How can the ESP32 microcontrollers be interconnected to create a scalable network for data collection?”

Firstly, it has been demonstrated that CSI data transfer is feasible using serial cable connections in conjunction with the Python tool. However, this approach faces scalability limitations due to the reliance on physical cables. To address this constraint, a more viable solution is proposed, involving the transfer of CSI data over Wi-Fi using POST requests to a central laptop for data collection. This alternative method offers improved scalability..

6.3 Research question 3

“What are the potential challenges and limitations associated with building and using a testbed for HAR with multiple ESP32 microcontrollers and how can they be mitigated?”

Section 5 highlights several limitations that arise when increasing the frequency beyond 40 Hz with 9 nodes operating in station mode. Additionally, the current implementation lacks the ability to switch a node from station mode to access point mode without the requirement of a serial cable. Another drawback of the original tool is the inconsistency observed in its passive mode, where it occasionally fails to detect traffic for extended periods, despite the presence of active signals. The cause of this inconsistency remains unknown, but it is speculated that the passive mode struggles to capture the relatively weak signals emitted by an ESP32 functioning as an access point and another ESP32 functioning as a station.

6.4 Main research question

“What is an efficient and scalable approach to construct a testbed for HAR using multiple ESP32 microcontrollers?”

In order to establish an efficient and scalable testbed for HAR using multiple ESP32 microcontrollers, the utilization of Wi-Fi for CSI data transfer, along with a dedicated tool for data management and labeling, has been adopted. This design has proven to be both effective and scalable.

6.5 Future work

While the current verification has indicated satisfactory results regarding the amount, variation and data loss of CSI data in my proposed design, future work can further validate its effectiveness. This can be achieved by conducting an in-depth analysis of the CSI data, training a model using serial data and subsequently testing it on data collected through Wi-Fi transfer. Such experimentation would provide a more robust assessment of the proposed design. Also, it is important to investigate potential interferences from other nodes, which may introduce changes in the CSI data and potentially impact the model's performance, particularly when multiple station nodes are involved.

Additional future work can be done by dynamically changing the access point node and the station nodes changing this dynamically based on where the human activity takes place further improves performance [10].

An interesting application of the proposed design is the real-time use of the collected CSI data for activity recognition. By feeding the CSI data directly into a trained model, it should become possible to analyze ongoing activities in real-time.

7. REFERENCES

[1] E. Kim, S. Helal, and D. Cook, "Human Activity Recognition and Pattern Discovery," *IEEE Pervasive Comput.*, vol. 9, no. 1, pp. 48–53, Jan. 2010, doi: 10.1109/MPRV.2010.7.

[2] C. Jobanputra, J. Bavishi, and N. Doshi, "Human Activity Recognition: A Survey," *Procedia Comput. Sci.*, vol. 155, pp. 698–703, Jan. 2019, doi: 10.1016/j.procs.2019.08.100.

[3] M. Vrigkas, C. Nikou, and I. A. Kakadiaris, "A Review of Human Activity Recognition Methods," *Front. Robot. AI*, vol. 2, 2015, Accessed: Jun. 24, 2023. [Online]. Available: https://www.frontiersin.org/articles/10.3389/frobt.2015.00028

[4] J. Schäfer, B. R. Barrsiwal, M. Kokhkharova, H. Adil, and J. Liebehenschel, "Human Activity Recognition Using CSI Information with Nexmon," *Appl. Sci.*, vol. 11, no. 19, p. 8860, Sep. 2021, doi: 10.3390/app11198860.

[5] A. K. Sahoo, V. Kompally, and S. K. Udgata, "Wi-Fi Sensing based Real-Time Activity Detection in Smart Home Environment," in *2023 IEEE Applied Sensing Conference (APSCON)*, Bengaluru, India: IEEE, Jan. 2023, pp. 1–3. doi: 10.1109/APSCON56343.2023.10101249.

[6] S. M. Hernandez and E. Bulut, "WiFi Sensing on the Edge: Signal Processing Techniques and Challenges for Real-World Systems," *IEEE Commun. Surv. Tutor.*, vol. 25, no. 1, pp. 46–76, 2023, doi: 10.1109/COMST.2022.3209144.

[7] Yongsen Ma, Gang Zhou, and Shuangquan Wang, "WiFi Sensing with Channel State Information: A Survey," *ACM Comput. Surv.*, vol. 52, no. 3, pp. 1–36, May 2020, doi: 10.1145/3310194.

[8] H. Jiang, C. Cai, X. Ma, Y. Yang, and J. Liu, "Smart Home Based on WiFi Sensing: A Survey," *IEEE Access*, vol. 6, pp. 13317–13325, 2018, doi: 10.1109/ACCESS.2018.2812887.

[9] D. Halperin, W. Hu, A. Sheth, and D. Wetherall, "Tool release: gathering 802.11n traces with channel state information," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 1, pp. 53–53, Jan. 2011, doi: 10.1145/1925861.1925870.

[10] S. M. Hernandez and E. Bulut, "Lightweight and Standalone IoT Based WiFi Sensing for Active Repositioning and Mobility," in *2020 IEEE 21st International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM)*, Cork, Ireland: IEEE, Aug. 2020, pp. 277–286. doi: 10.1109/WoWMoM49955.2020.00056.

[11] S. M. Hernandez and E. Bulut, "WiFederated: Scalable WiFi Sensing Using Edge-Based Federated Learning," *IEEE Internet Things J.*, vol. 9, no. 14, pp. 12628–12640, Jul. 2022, doi: 10.1109/JIOT.2021.3137793.

[12] "Overview of ESP32 features. What do they practically mean?," *Tutorials*. https://exploreembedded.com/wiki/Overview_of_ESP32_features._What_do_they_practically_mean%3F (accessed Jun. 24, 2023).

[13] R. Teja, "Getting Started with ESP32 | Introduction to ESP32," *ElectronicsHub*, Feb. 17, 2021. https://www.electronicshub.org/getting-started-with-esp32/ (accessed Jun. 24, 2023).

[14] S. M. Hernandez, "ESP32 CSI Tool." Jun. 07, 2023. Accessed: Jun. 22, 2023. [Online]. Available: https://github.com/StevenMHernandez/ESP32-CSI-Tool

[15] "espressif/esp-idf at release/v4.3." https://github.com/espressif/esp-idf/tree/release/v4.3 (accessed Jun. 22, 2023).

[16] "Arduino IDE 2.x." Arduino, Jun. 24, 2023. Accessed: Jun. 24, 2023. [Online]. Available: https://github.com/arduino/arduino-ide

[17] "Jelle3345/ESP32_testbed." https://github.com/Jelle3345/ESP32_testbed (accessed Jun. 22, 2023).

[18] "General · Jelle3345/MY_ESP32_CSI_TOOL," *GitHub*. https://github.com/Jelle3345/MY_ESP32_CSI_TOOL (accessed Jun. 22, 2023).

[19] A. Khan, N. Hammerla, S. Mellor, and T. Plötz, "Optimising sampling rates for accelerometer-based human activity recognition," *Pattern Recognit. Lett.*, vol. 73, pp. 33–40, Apr. 2016, doi: 10.1016/j.patrec.2016.01.001.

8. APPENDICES

Appendix A. Wi-Fi vs serial data transfer experiment results

Table Legend

The duration of each experiment was 15 seconds.

- Station Amount: The number of ESP32 station nodes used in the experiment.
- Transfer Type: The type of data transfer method used (e.g., Wi-Fi, serial).
- Hz Setting: The intended frequency setting for data transmission.
- Hz Actual: The actual frequency achieved during data transmission.
- Expected Time Between Packets: The expected time interval between consecutive packets of data.
- Mean Time Between Packets: The average time interval achieved between consecutive packets of data.
- CV Time Between Packets: The coefficient of variation (CV) of the time intervals between packets, indicating the variability in timing.
- Data Lost: The number of data packets lost in the 15 second experiment during transmission.
- Transmission Ratio: The ratio of data packets received relative to the total transmitted data.

station amount	transfer type	hz setting	hz actual	expected time between packets	mean time between packets	CV time between packets	data lost	transmission ratio
9	serial	10	10.019	0.1	0.1	0.0713	0	1
2	wifi	10	12	0.1	0.08	0.4578	0	1
1	serial	10	10.0667	0.1	0.1	0.0263	0	1
2	serial	10	9.9333	0.1	0.1	0.0309	0	1
9	wifi	10	12.3333	0.1	0.0796	0.46	0	1
1	wifi	10	13.3333	0.1	0.0794	0.4693	0	1
2	serial	20	20.1	0.05	0.05	0.0618	0	1
1	serial	20	19.6667	0.05	0.05	0.0648	0	1
9	wifi	20	23.4074	0.05	0.043	0.3606	0	1
9	serial	20	19.9333	0.05	0.05	0.229	0	1
2	wifi	20	24	0.05	0.0419	0.3787	0	1
1	wifi	20	24	0.05	0.0404	0.419	0	1
9	wifi	40	38.9333	0.025	0.0256	0.385	0	1
9	serial	40	36.5778	0.025	0.0244	0.2933	0	1
1	wifi	40	45.3333	0.025	0.0227	0.2943	0	1
2	wifi	40	42.6667	0.025	0.0232	0.3174	0	1
1	serial	40	39.9333	0.025	0.025	0.1179	0	1
2	serial	40	40.1667	0.025	0.025	0.1226	0	1
2	serial	60	59.2667	0.0167	0.017	0.1834	0	1
1	wifi	60	58.7333	0.0167	0.0169	0.3833	0	1
2	wifi	60	58.7	0.0167	0.0168	0.3702	0	1
9	wifi	60	44.2667	0.0167	0.0224	0.5149	0.2222	0.9997
9	serial	60	50.5259	0.0167	0.0186	0.3947	0	1
1	serial	60	58.9333	0.0167	0.017	0.1686	0	1
2	wifi	80	72.1333	0.0125	0.0137	0.4893	0.5	0.9995
1	wifi	80	69.4	0.0125	0.0146	0.4195	0	1
2	serial	80	75.8333	0.0125	0.0133	0.3884	0	1
9	wifi	80	47.1481	0.0125	0.0208	0.5789	0.2222	0.9996
9	serial	80	49.9852	0.0125	0.0185	0.3709	0	1

1	serial	80	76.9333	0.0125	0.013	0.1935	0	1
1	serial	100	90.5333	0.01	0.0111	0.7275	0	1
9	serial	100	55.5889	0.01	0.0181	0.6151	0	1
2	serial	100	87.5	0.01	0.0116	0.7248	0	1
9	wifi	100	44.4963	0.01	0.0224	0.5817	0.1111	0.9998
2	wifi	100	78.8	0.01	0.0125	0.7218	2	0.9983
1	wifi	100	85.4667	0.01	0.0116	0.7005	1	0.9992
2	serial	120	88.1667	0.0083	0.0111	0.7786	0	1
1	serial	120	93.6	0.0083	0.0107	0.774	0	1
9	wifi	120	35.5778	0.0083	0.0299	2.5673	22.6667	0.9552
9	serial	120	45.1037	0.0083	0.0227	0.576	0	1
2	wifi	120	85.5667	0.0083	0.0117	0.7324	1.5	0.9989
1	wifi	120	96.1333	0.0083	0.0105	0.7456	3	0.9979

Appendix B. The office used for verification of Wi-Fi vs serial data transfer

The image depicts a configuration consisting of a USB hub connected to ten ESP32 microcontrollers through their respective serial ports. The node positioned on the table operates as an access point, while the remaining nodes are configured in station mode as per the experimental requirements.