Human Activity Monitoring with a Wi-Fi Embedded Network

Gregor Watt

H00341198

Dissertation submitted for the Degree of MEng Robotics and Autonomous Systems

Heriot-Watt University Department of Computing and Electrical Engineering

Table of Contents

Abstract

This paper represents a 3 month research placement at Heriot-Watt University as part of the penultimate year of a MEng degree in Robotics and Autonomous Systems. This work involved extensive software development, data processing and academic research. The eventual goal of the project was to develop a Wi-Fi embedded sensor network capable of monitoring human activity in a home environment. This system was designed to be operational 24/7 and to be as non-intrusive as possible in everyday life.

Project Motivation - Fall Detection

Falls and trips are the leading cause of death and injury for over 65s[1]. Every year in the UK, at least 5 million elderly people will experience at least one fall [2]. Non-fatal falls are especially common and cost the NHS £1B annually on treating hip fractures alone. As the UK has an aging population this issue will only become more prevalent in the coming years. ”Long lies”, or falls with a subsequent hour spent on the floor are especially dangerous. Half of elderly people who have a long lie will die within six months due to related complications [3]. Elderly people who live alone are especially at risk of such falls and may be unable to notify carers of a fall, leading to a long lie.

Fall detection systems have been developed and are typically wearable devices with a separate base station that notifies emergency services or carers in the event of a fall. Most of these systems require regular charging and must be worn at all times. They also must be able to withstand everyday wear and tear, which can add cost and make devices more bulky.

Fig 1: Careline fall detector and base station

The above example of an autonomous fall detector is a subscription service priced at £22.67 per month and requires a setup fee of £45.
There are 1 million people living with dementia in the UK and the vast majority are elderly. The impact of memory loss and other conditions caused by dementia mean that a wearable fall alarm may not be suitable for a significant portion of people at risk of a fall.

Project Proposal - A sensorless Wi-Fi embedded network

A sensorless system would potentially be more suitable for dementia sufferers as it eliminates the need for regular charging and is less intrusive than a wearable device. Such a system should be designed to be cheaper than alternatives, low-power, non-intrusive and with negligible installation costs. Wi-Fi is ubiquitous across the modern world, and 94% of UK residents have a wireless router in their homes. In recent years, a lot of research has focused on optimising signal propagation in homes. This has resulted in commercial routers becoming more advanced pieces of hardware, capable of accounting for multipath reflections and interference in a complex and dynamic environment. Movement in a room changes how signals propagate in a measurable way, meaning a wireless system could be developed to detect movement in a room. Domestic spaces are flooded with such signals and this dense RF field can be taken advantage of. A human monitoring system could consist of Wi-Fi embedded chips placed around a room and a commercial router. Multiple devices would be used to open up the possibility of data synchronisation and more widespread coverage, allowing for more reliable fall detection than a single device solution.

Background Theory - Wi-Fi

Most modern Wi-Fi signals use a 2.4-5Ghz carrier frequency and use multiple subcarriers to transmit data reliably at very high rates. A subcarrier is a sideband of a carrier frequency which is a frequency adjacent to the carrier frequency.

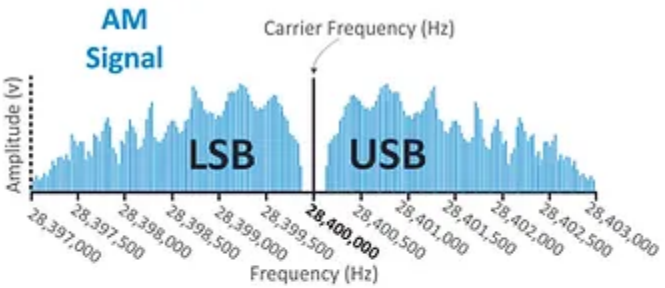


Fig 2 : The lower and upper sidebands of a RF Signal

In the IEEE Wi-Fi standard 802.11ax or “Wi-Fi 6”, there can be up to 2048 subcarriers, equivalent to a total channel width of 160Mhz. Wi-Fi 6 takes advantage of the multipath effect, allowing for multiple paths to a receiver, meaning the rate and reliability of data transmission is increased significantly.

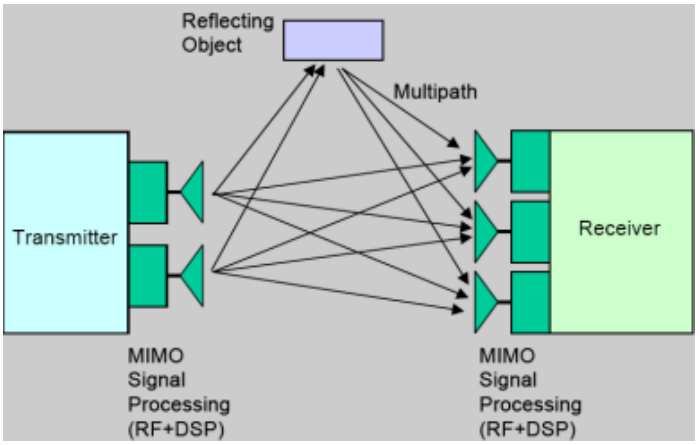


Fig 3: Multipath propagation

Background Theory - Channel State Information

Radio frequency (RF) signals are affected by phenomena such as scattering, reflection and refraction. These change the amplitude and shift the phase of a signal in a measurable way, allowing for the detection of changes in the environment, purely based on the difference between a transmitted and received signal. Channel State Information (CSI) represents the combined effect of scattering, reflection and refraction on a signal's propagation through a communication channel. It can be used to adapt to adverse channel conditions, which is especially important in home environments with unpredictable characteristics. CSI is most often used in multiantenna systems, as signals can take multiple paths from a transmitter to a receiver. A MIMO (multiple input, multiple output) system can be represented as the following.

$$V = \sum_{i=1}^{N} ||V_i|| e^{-j\theta i}$$

Where V_i and θ_i are the amplitude and phase of path i, N is the amount of paths, and V is the complex voltage measured at the receiver. In a multipath scenario, the system can be represented as:

$$Y = Hx + n$$

Where Y and x are the received and transmitted signals, n is noise and H is a complex matrix representing the channel state information. The channel impulse response is useful to identify the characteristics of individual path and can be derived from the above formula as a linear filter:

$$H(\tau) = \sum_{i=1}^{N} \left|a_i\right| e^{-j\theta_i} \delta(\tau - \tau_i)$$

Where a_i, θ_i, and τ_i represent the amplitude, phase, and time delay of each multipath signal. The channel frequency response can then be determined as the fourier transform of the impulse response:

$$H(f) = FFT(H(\tau)) = [H(f_1), \ldots H(f_k)]$$

Channel state information at subcarrier frequencies k can then be represented as a complex number which represents amplitude and phase shift.

Hardware Selection

The devices chosen for collecting CSI were subject to a number of requirements. The principal requirement was extensive Wi-Fi capability, not just for CSI collection but also for connecting to a local area network (LAN) in order to communicate with other devices. A small form factor was also a significant requirement, as non-intrusiveness in everyday life is a key benefit of a sensorless system. There are multiple devices capable of collecting CSI, however most rely on an OS such as Linux to be installed in order to communicate with embedded network interface cards.

Device	Advantages	Disadvantages
ESP32 built in CSI tool	Very low cost Low power requirements Small form factor	Limited local storage Limited processing power Closed source Wi-Fi API
Atheros-CSI-Tool [12] on a Linux device with embedded NIC	Adequate processing power for on-board data analysis Adequate local storage Fully open source	Higher power requirements Larger form factor Has to be attached to a computer with OS installed

The ESP32 development board was selected for its exceptionally low cost and power requirements.

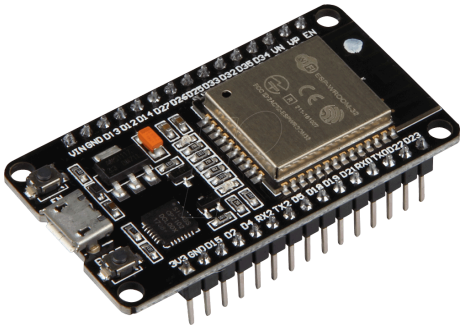


Fig 4: ESP32

The issues of the ESP32, specifically the lack of sufficient local storage for CSI data were remedied by the addition of a local database server hosted on a Raspberry Pi 3B+.

Cost analysis:

Hardware Components	Estimated cost (£)
3x ESP32-DevKitM-1	16.00
Raspberry Pi 3 Model B+	33.50
Standard Wi-Fi Router	10
Total cost:	59.50

System Overview

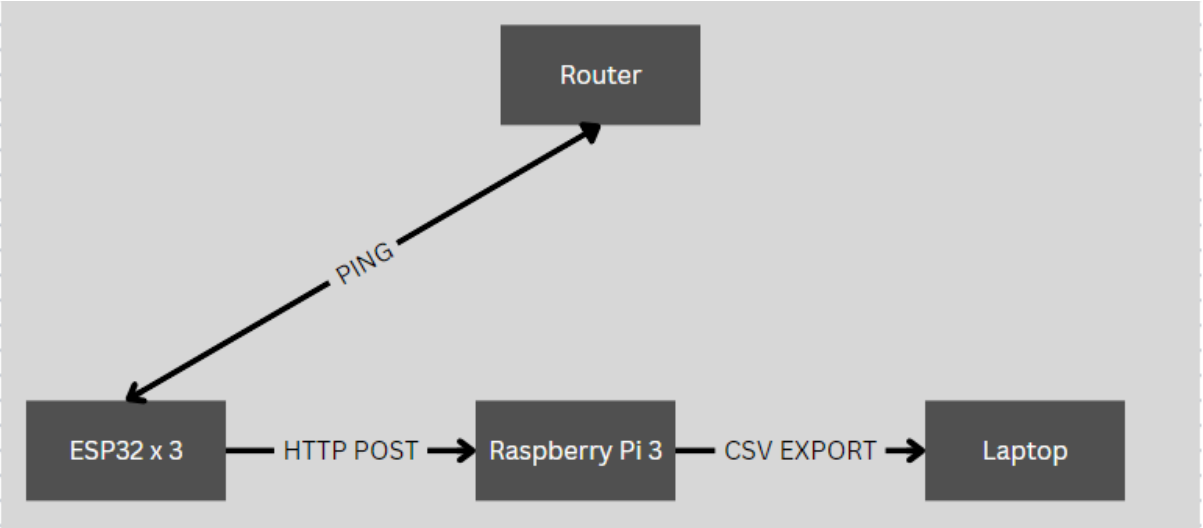


Fig 5: System Block Diagram

The system was implemented on a dedicated WLAN network with only the devices shown connected. The router used was a second-hand NOW TV Hub Two Dual Band Wireless Broadband Wi-Fi Router (Model NR801). The ESP32 would collect CSI by sending ICMP ping packets to the router, waiting for a reply and then capturing one frame of CSI data. This data would then be sent via a HTTP POST request to the Raspberry Pi Server which would store it in a MySQL database. Data was then processed by downloading the database as a .CSV file onto a laptop with sufficient processing power..

Fig 6: Commercial Router

ESP32 CSI Collection

CSI was obtained by repeatedly sending ICMP (Internet Control Message Protocol) ping packets to a wireless router. ICMP pings involve sending an ICMP echo request packet to a host and then waiting for an ICMP echo reply packet.

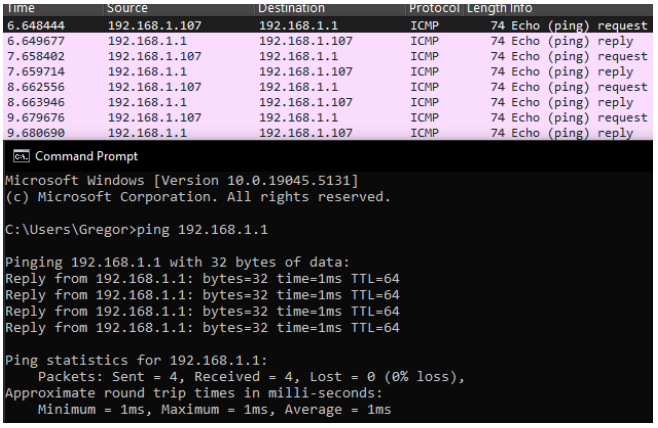


Fig 7: an example of ICMP ping request and replies (captured by Wireshark)

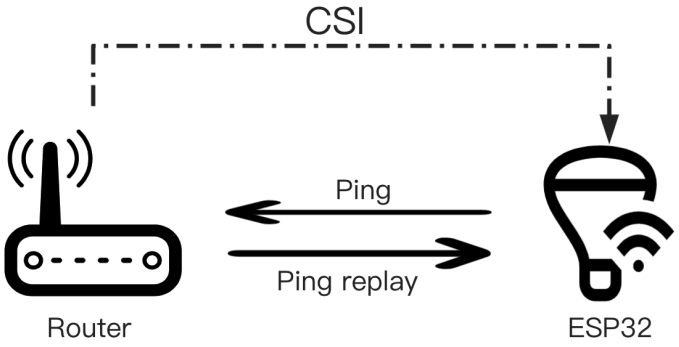


Fig 8: CSI collection via ICMP ping

Pings were sent and received at a rate of ~10Hz and a built-in callback function (*esp_wifi_set_csi_rx_cb*)[7] in the ESP32 networking API returned CSI every time a ping reply was received. CSI data returned by the ESP contained headers, such as the RSSI and timestamp that the data was received, followed by a buffer containing the amplitude and phase data of each subcarrier. Data was sent via POST request from the ESP32 to the Raspberry Pi database using the HTTP API. The devices would send the CSI headers, the CSI buffer, and an ID unique to each device.

Raspberry Pi LAMP Server and Database

A LAMP server utilises Linux (OS), Apache (web server), MySQL (database) and PHP to create a server capable of storing large amounts of data and then serving it to clients. It is a standard software stack which is widely used for similar projects. PHP was used to process POST requests, a simple script was written to parse the request and insert it into a MySQL database. Each time a request was received, the script would also insert the current UNIX time in nanoseconds, this was done to synchronise readings from many devices simultaneously. Scripts were written to export the data to a .CSV file, which could then be downloaded and analysed by a python script on a more powerful computer.

Data Processing and Analysis in Python

A significant amount of data was generated during testing. As each device had 128 subcarriers and collected data at a rate of ~10Hz, ~1.3KB of data was generated every second. As large amounts of data were required to be parsed, analysed and plotted, appropriate python libraries were required. Numpy was used for all the mathematical analysis as it has exceptional performance when dealing with large arrays and matrices. There is also a large selection of built-in functions such as linear filters and vector math functions. Matplotlib was then used to plot data due to its relative simplicity and speed when compared to other graphing libraries. As all data was time series data, a dedicated *TimeSeries* class was created with methods for plotting and data processing. Each subcarrier contained time series data for the amplitude and phase, so a *SubCarrier* class was also implemented, as was an overarching *CSI* class which contained all *SubCarrier*s for a specific device. The class diagram of the whole program is shown below.

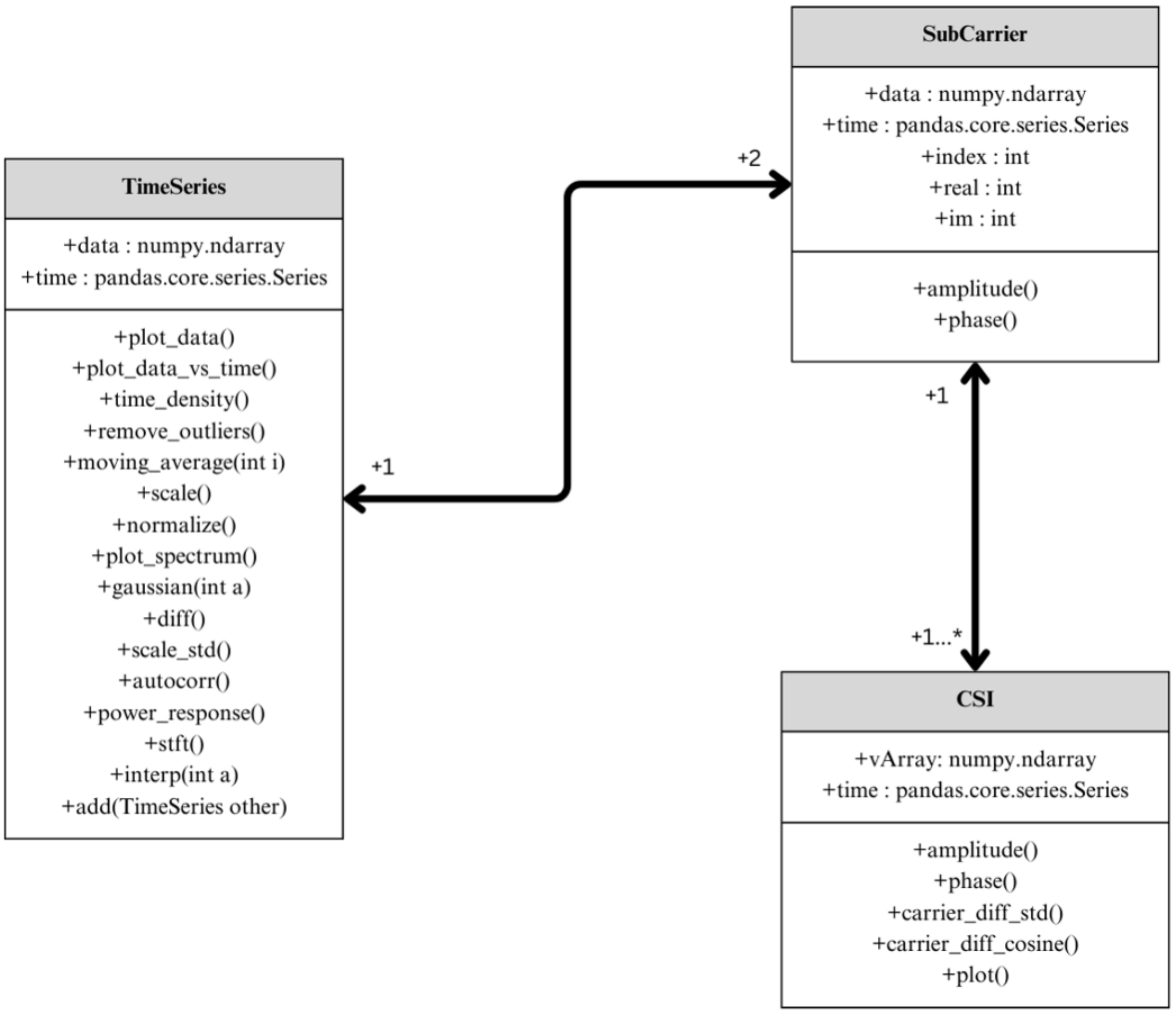


Fig 9: Python Class Diagram

Initial Findings

Both phase and amplitude were found to be noisy but highly correlated between subcarriers on the same device. Raw phase was especially noisy and unlike amplitude, was not correlated between devices.

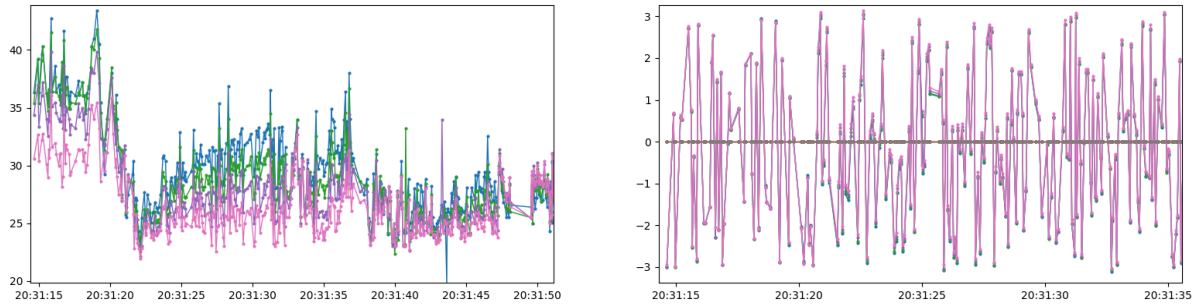


Fig 10: Raw phase and amplitude measurements from 4 subcarriers on one ESP32 (amplitude in dB)

Time delays between measurements of up to 4000 ms were also apparent on some occasions. The source of these time delays was uncertain but was related to ESP32s not receiving CSI frames from the router for short periods of time. This was believed to be an issue with the router being unable to or rejecting pings, as periods of long delay were sometimes correlated between multiple devices. Outside of these regions an average delay of ~70ms was observed. Raw phase data was significantly more noisy when compared to amplitude due to phase offsets being affected by unsynchronised sampling clocks of the receiver and transmitter. Methods exist that can be used to unwrap the phase from the raw phase data, however these methods have limited accuracy and are beyond the scope of the project.

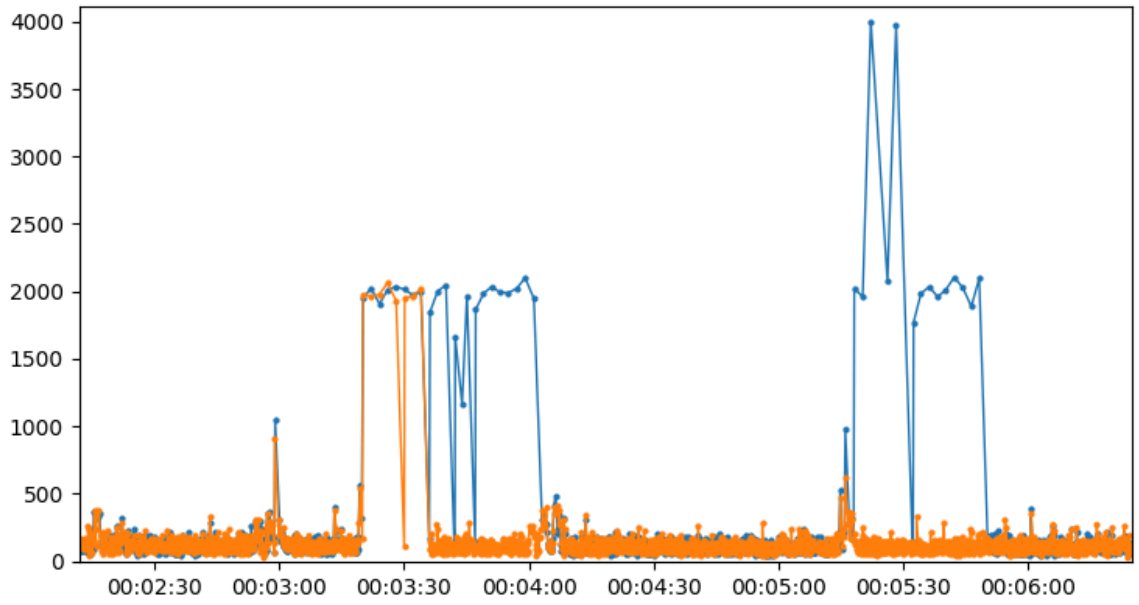


Fig 11: Raw time delay in milliseconds between measurements for 2 devices

Amplitude data was primarily focused on, as the effect of TX-RX desynchronisation was insignificant and it did not require phase unwrapping. Data points not within a certain amount of standard deviations of the mean were removed and gaussian smoothing was performed to reduce noise. Outlier

removal used a z-score threshold of 2 and resulted in ~4% of data points being replaced by the moving average of the last few points. Experimentation was also done involving band pass filtering to reduce high frequency noise instead of gaussian smoothing as it also removed the DC component of the signal.

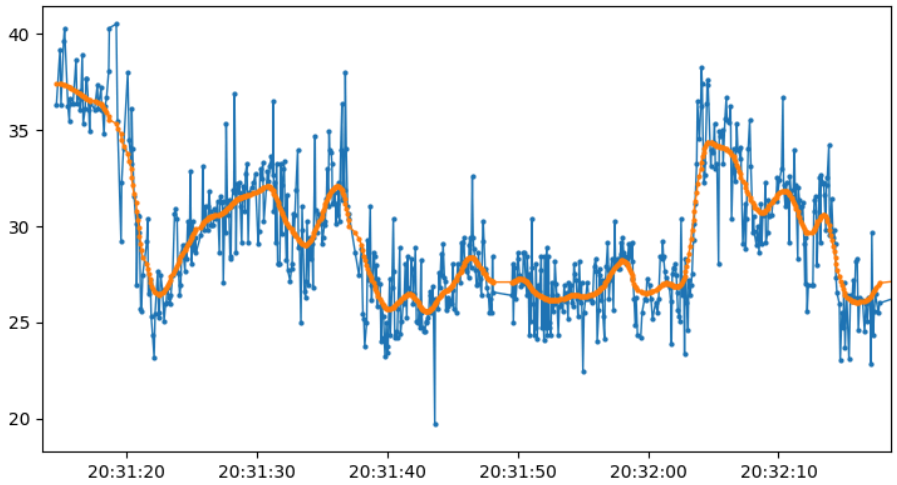


Fig 12: Gaussian smoothed subcarrier amplitude ($\sigma = 7$)

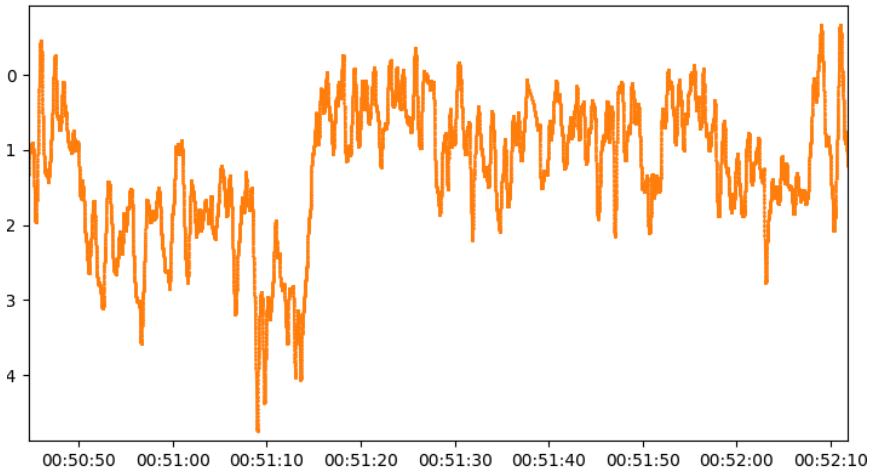


Fig 13: Band-Pass filter from 0-5Hz (DC component removed)

The amplitude values of individual subcarriers were not always indicative of movement in a room, as each subcarrier path was affected differently. However, it was observed that subcarriers tended to deviate from one another during movement.

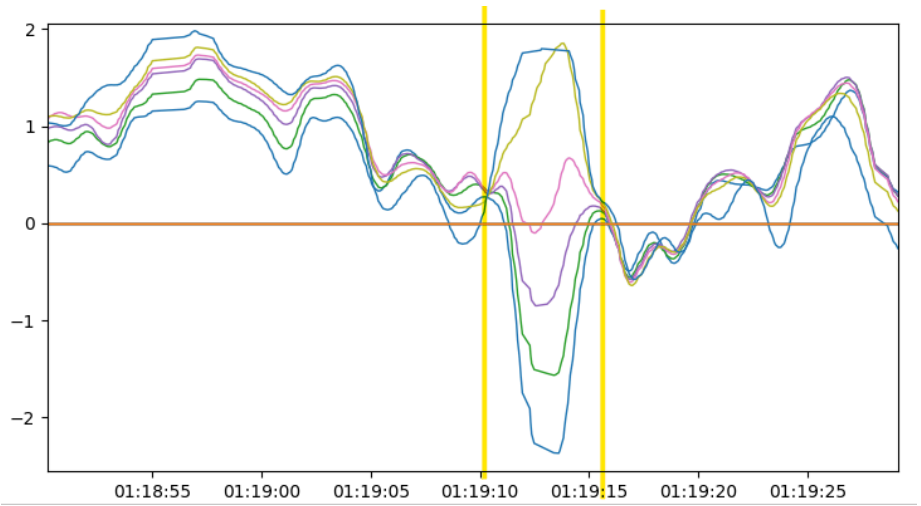


Fig :14 Subcarrier amplitude deviation during a fall(normalized)

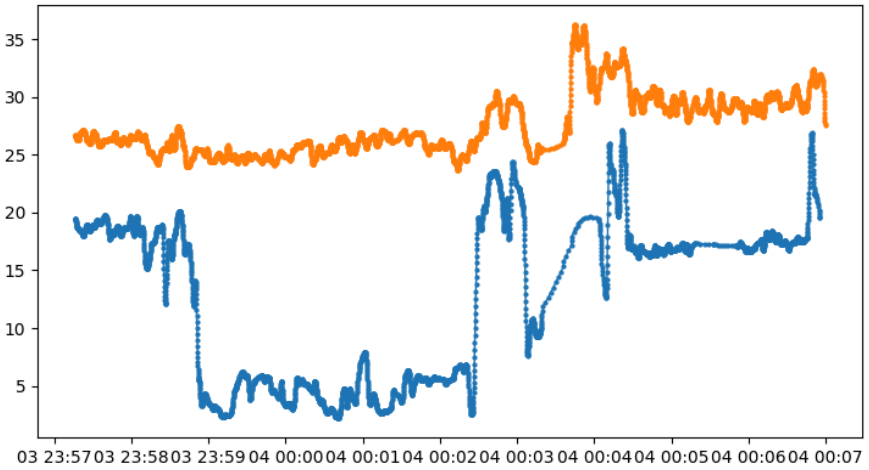


Fig 15: Correlation of subcarriers of the same frequency on two different devices

Further Data Analysis

It is apparent that deviation between subcarriers is highly correlated with movement in a room. Subcarriers of the same or similar frequencies are also highly correlated between devices. This can be exploited to highlight periods of movement when subcarrier amplitudes deviate from each other. The cosine similarity between subcarriers was used for individual devices.

$$S(t_n) = \frac{\langle \hat{H}(t_n), \hat{H}(t_{n-1}) \rangle}{|\hat{H}(t_n)||\hat{H}(t_{n-1})|}$$

Fig 13: Cosine similarity formula

Where $\widehat{H}(t_n)$ is a vector containing all subcarrier amplitudes at time t_n. If vectors are orthogonal, the cosine similarity is 0, when they are parallel and opposite the similarity is 1 and -1 respectively. The cosine difference can then be defined as $1 - S(t_n)$.

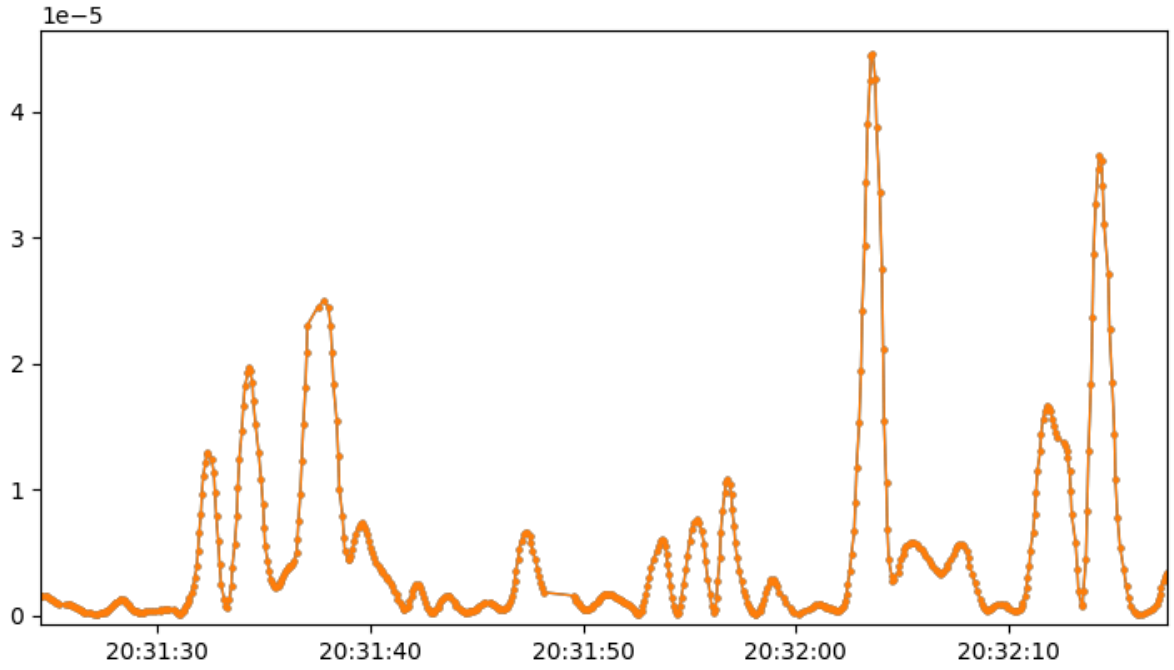


Fig 16: Cosine difference between 92 smoothed subcarriers (fall at 20:32:03)

In the above example, the system was started and a person walked slowly to a location in the centre of the room. The walking motion is visible from 20:31:33 to 20:31:38. The person then fell onto a mattress in the centre of the room and lay there until 20:32:11 when they slowly got back up. This approach successfully detects most movements in a room but it is challenging to identify a fall versus everyday movement. Doors opening in a room created a cosine difference significantly higher than a fall, therefore a simple threshold value is not sufficient for accurate fall detection. Proximity to a device also had a significant effect on the magnitude of readings. A fall that was closer to a device would have an exponentially larger effect. This issue was difficult to account for as different areas of the room also had higher peaks and troughs.

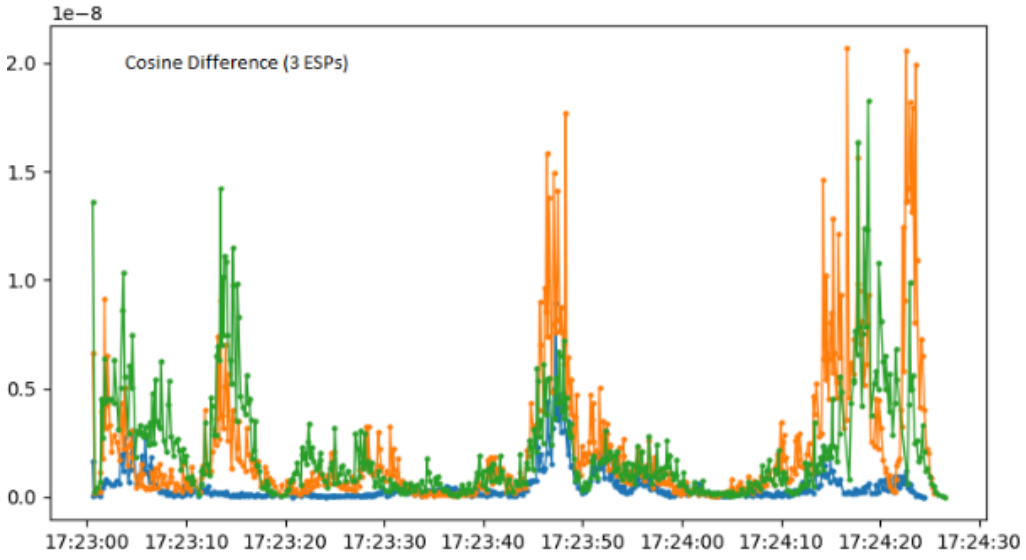


Fig 17: Cosine difference from 3 ESP32s

In the above example, the ESP32 coloured in blue does not detect movement spikes that are otherwise visible in the other two devices. The movement at 17:23:15 was closest to the green device and the movement at 17:23:47 was closest to the orange device.

Despite a lack of detail on the types of movements performed, the system was readily able to detect movements of any kind, and had extremely low noise levels in static environments.

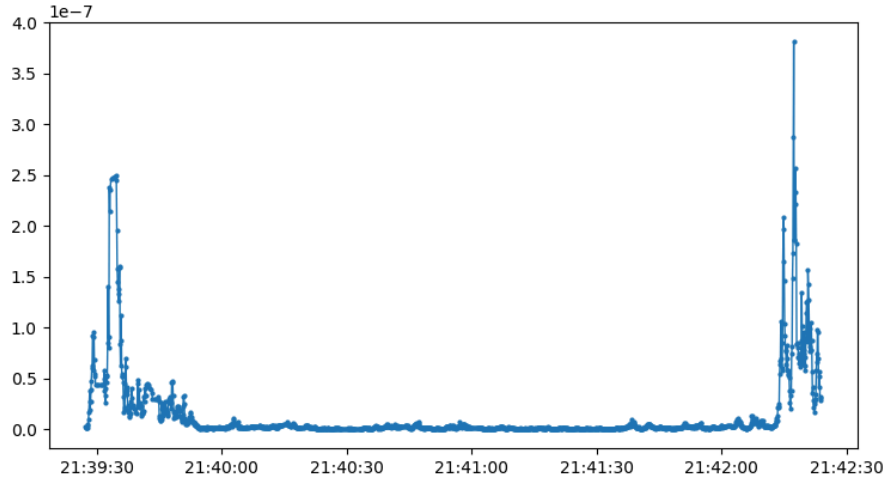


Fig 18: Cosine difference of an ESP32 in an empty room (person left (21:39) and returned (21:42))

However, there was significant difficulty in determining the specifics of a movement. Small gestures such as a hand moving would sometimes produce signal magnitudes comparable to that of a fall or a door opening. Due to this issue, simple thresholding of the cosine difference was not enough to detect a fall reliably. This issue was exacerbated by the fact that movements which blocked line of sight had a disproportionately large effect on readings. A higher sample rate could be highly beneficial, as it would allow for more data on shorter movements and has been used for speed estimation [10]. The limited CSI data rate was believed to be a combination of limited ESP32 processing power and limited commercial router capabilities. The code flashed onto the ESP32s could also potentially have been optimised for faster data collection rates, however the speed of HTTP requests would also likely become a factor.

Hardware modifications

The ESP32 board is equipped with a PCB antenna for picking up Wi-Fi Signals. The built-in antenna was relatively short, at ~5cm and it was determined that a longer antenna could pick up signals more readily and at a higher amplitude. PCB covering was scraped off so that an external antenna could be soldered onto the board.

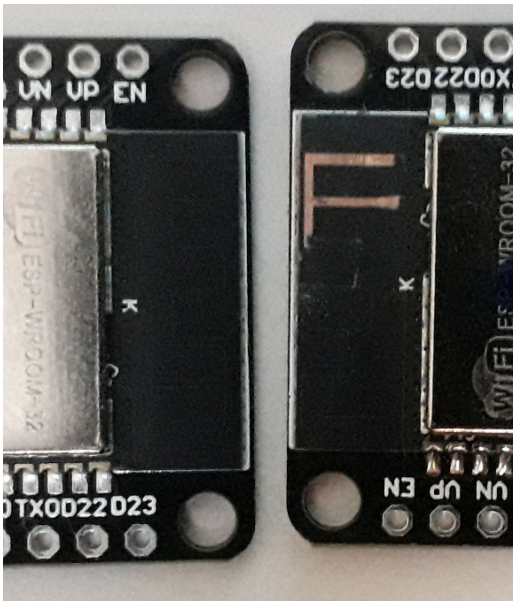
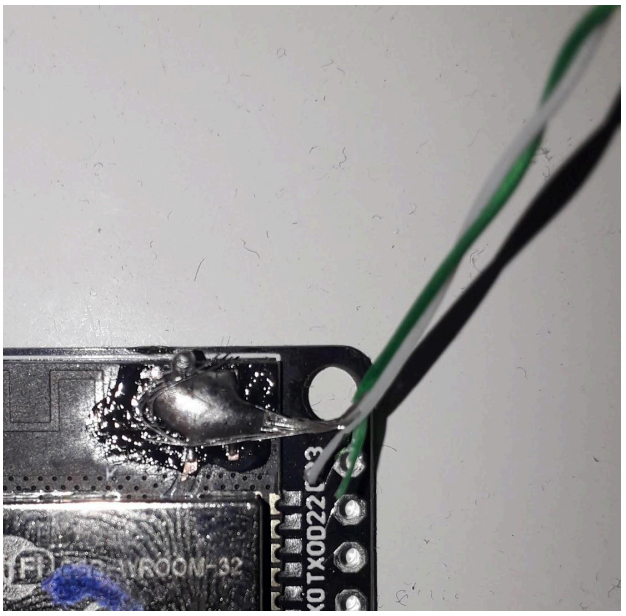


Fig 19: Unmodified ESP32 and one with antenna covering partially removed



]

Fig 20: ESP32 with custom antenna soldered

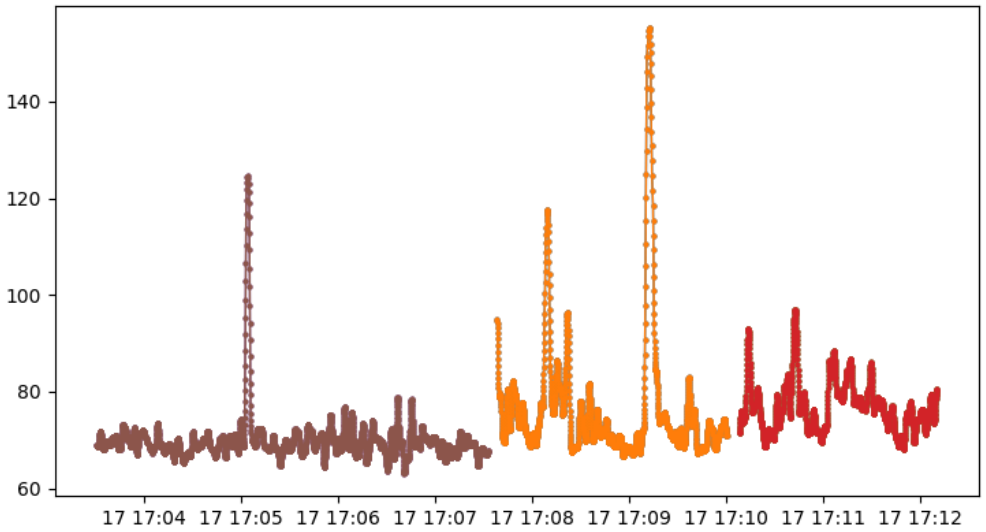


Fig 21: Modified ESP32 latency (modified device in brown)

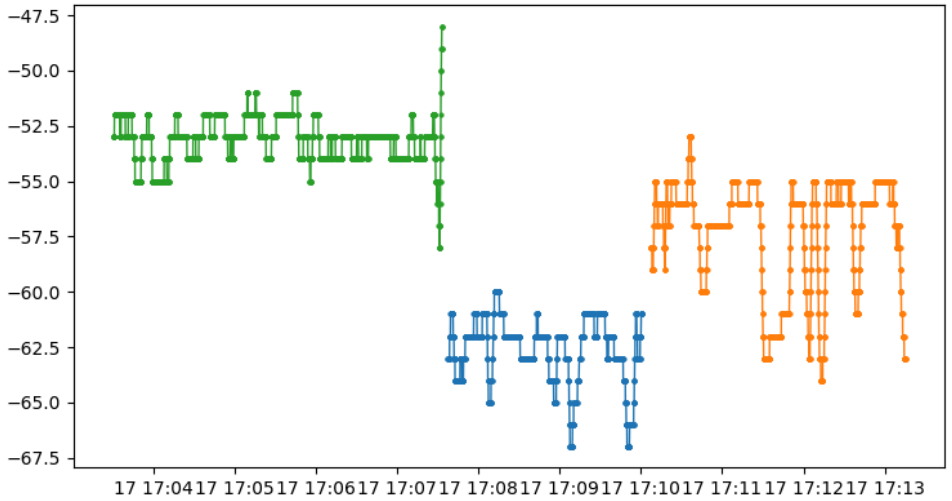


Fig 22: RSSI of each device (modified device in green)

ESP	Average latency over 1 minute (ms)	Average RSSI over 1 minute (dBm)
1	74.9	-62.6
2	75.0	-57.3
3 (modified)	69.6	-53.1

The addition of an external antenna reduced latency between HTTP packets by ~5ms and increased the Received Signal Strength Index (RSSI) by 10dBm when compared to an unmodified device at the same distance to the router.

System Feasibility and Conclusions

After testing and research of related works, it was clear that a higher rate of data collection was required to identify falls as unique from other movements. However, the low rate of data collection was sufficient to identify movement in a room very accurately.

Simple thresholding of the cosine difference resulted in a robust system for basic movement detection. The best algorithm tested used three ESP32s and all subcarriers available. The low sample rate was also accounted for by linearly interpolating between data points, allowing for a synthesised rate of ~100Hz.
The amplitude signals were then subjected to a band pass filter from 0.1-0.5Hz to reduce high frequency noise and also to remove DC components. Outlying data points were also removed during pre-processing. Due to varying peak levels between devices depending on room position, the minimum cosine difference of all three devices was taken as a form of consensus between the three devices.

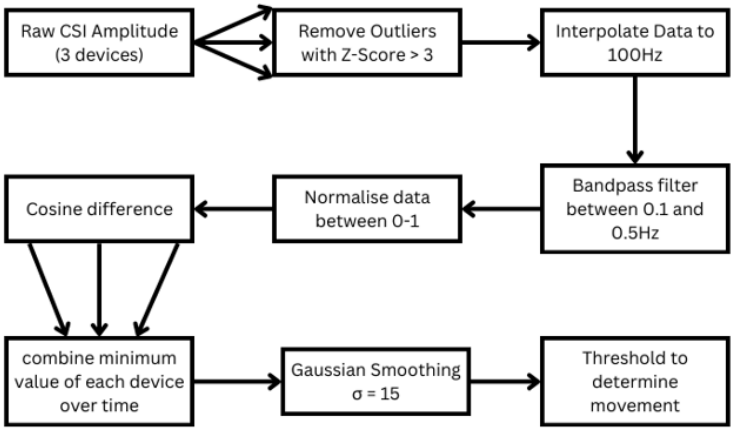


Fig 23: Data processing diagram for motion detection

During testing the threshold was never exceeded unless a significant motion (limb moving, door opening) occurred. Over a two hour testing period in a static environment the threshold was not exceeded until a person entered the room.

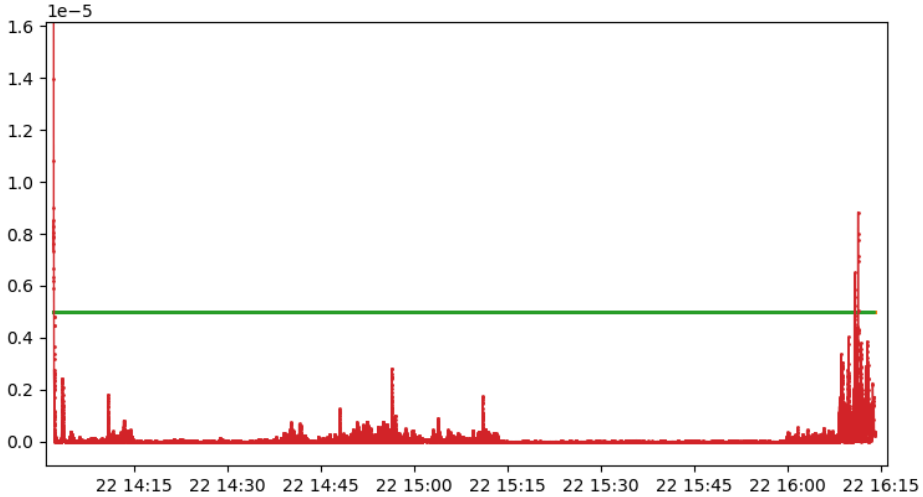


Fig 24: two-hour test of cosine difference thresholding in an empty room (room was entered at 16:10)

In the above test, the system was enabled, then the test subject went into another room in the building. From 14:40 until 15:10, the test subject was cooking in a room that shared a wall with the room testing was being performed in. It was noted that activity in adjacent rooms could sometimes register on the system, especially when devices were placed next to thin walls. From 15:10 to 16:00 there was no test subject present in the room or in the adjacent room. The test subject returned and sat down at a desk in the testing room before stopping data collection.

The system was found to be accurate at detecting movements in a room. The graph below is an example where a test subject sat on the floor for 20 seconds, then stood up and walked around for 20 seconds. This process was repeated 6 times. The movement indicator detected the start of all movements and was accurate within a few seconds of movement starting. However it was noted that when the person stopped there was sometimes a delay of up to 10 seconds, this is especially clear in movements 3, 5 and 6.

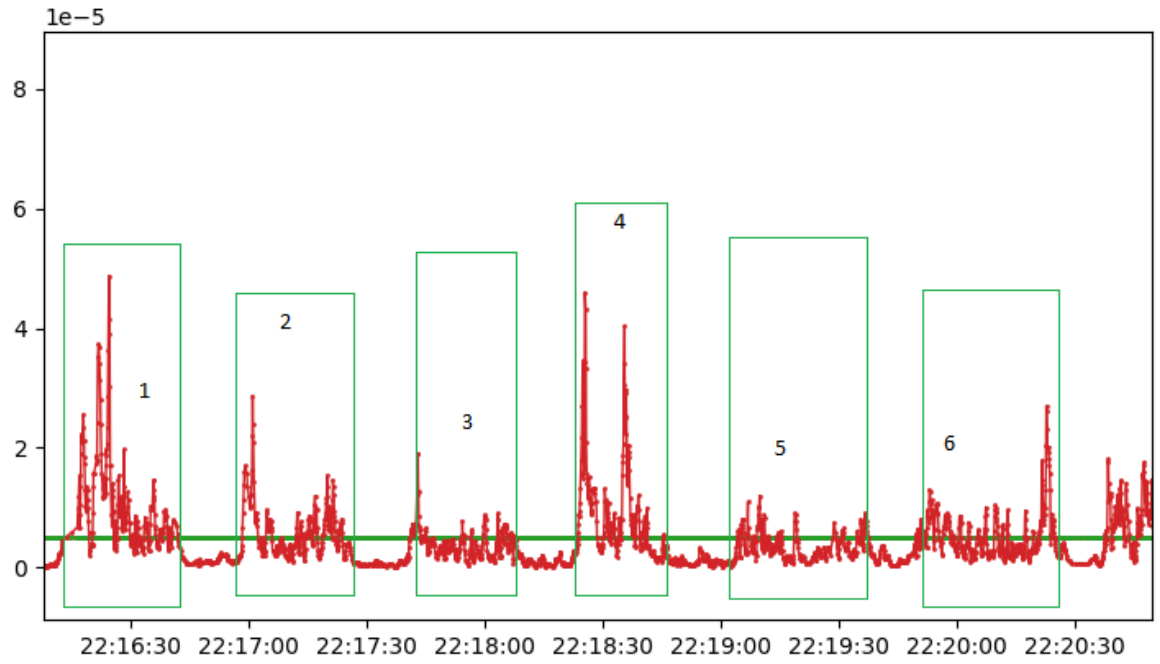


Fig 25: 6 movement testing run

The system achieved a 100% detection rate over a limited testing run with no false alarms, however more testing is required to gain confidence that the system is versatile in other environments. The location that the test subject sat down varied across the room and this did not seem to affect detection rate.

Related Works

SiFall: Practical Online Fall Detection with RF Sensing [8]

This paper used the cosine difference between subcarriers as an indicator of movement in a room, however it also highlighted the difficulty of highlighting falls from other movement. To achieve this, a neural network model was trained on the short time fourier transform (STFT) of the cosine difference between subcarrier amplitudes.

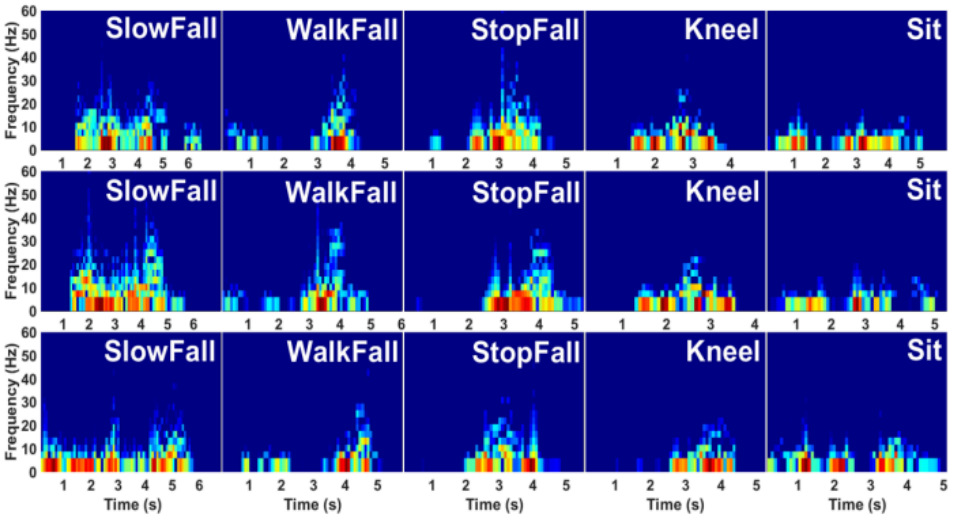


Fig 26: STFT segments from various movements [8]

This network was trained to identify movements and achieved a very high accuracy and low false positive rate. An auto-encoder network structure was used to reduce an STFT segment to a set of parameters, effectively compressing the data.

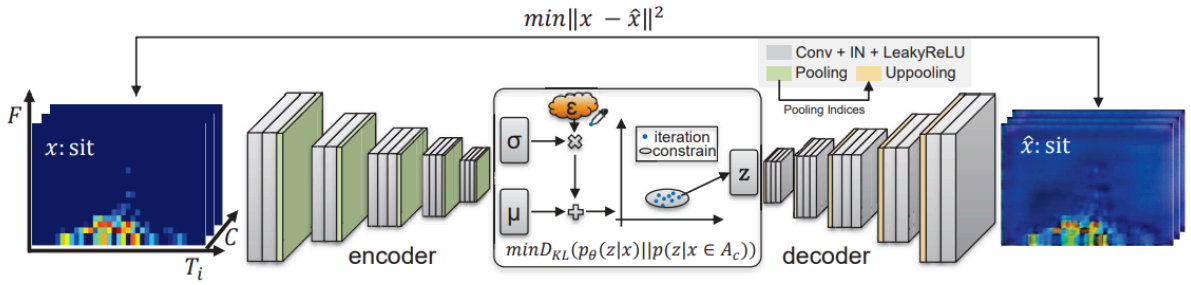


Fig 27: neural network architecture used to identify falls [8]

The above network was trained on very powerful hardware (an NVIDIA 2080Ti GPU) for a total of 19.3 hours before a fall detection rate of 100% was achieved. The data set used to train the network contained 1447 various segments of human activity, involving falling, walking and kneeling in different environments with different test subjects. This system used the Atheros SoC QCA9558 to collect CSI data at a very high rate (200 packets per second) which allowed for superior time resolution. Accuracy of fall detection was highly dependent on distance to the sensor, and would drop to 40% at a distance of 7m. The paper used a single device to receive CSI instead of a sensor network.

DeFall: Environment-Independent Passive Fall Detection using WiFi [9]

This paper proposed a method of a speed estimation algorithm which could be used for fall detection. It used two devices with Intel 5300 network interface cards to detect falls at an accuracy of 98% with a false alarm rate of 1.5%. Accurate speed estimation and fall detection required a high rate of CSI capture at 1500 Hz, whereas a simple motion detection system was possible at ~30Hz. A statistical approach was used instead of a learning based approach, which allowed for a system that was more robust in different environments. Speed estimation was achieved by analysing the autocorrelation function of the CSI power response. Due to the fact that the noise in the amplitude of subcarriers followed a normal distribution in a static environment, movement could be identified.

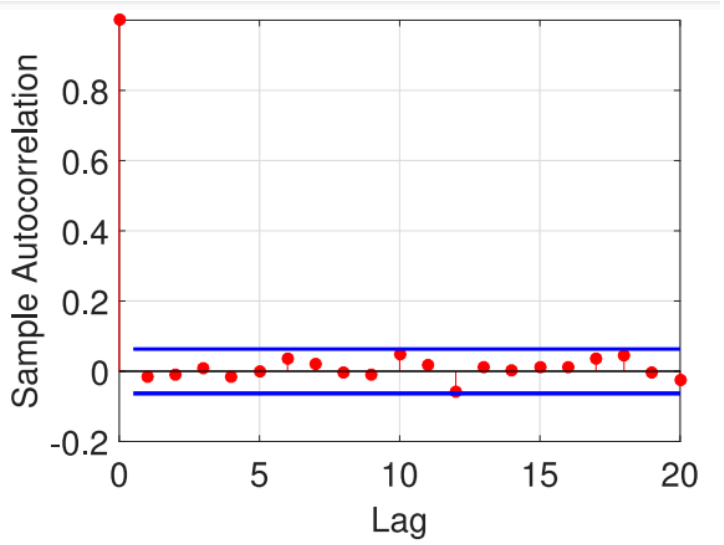


Fig 28: Autocorrelation function of white noise [10]

Once speed detection was achieved, the identification of falls was achieved using a set of template falls, which were determined from a dataset of falls. A DTW-based pattern matching algorithm was then used to identify falls.

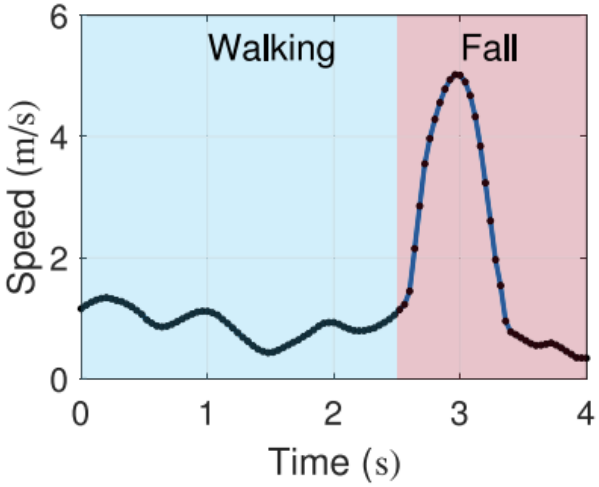


Fig 29: Speed vs Time detected for a walk then fall [9]

Future Work

A factor currently limiting the system is that the practical rate of data collection from the hardware is not sufficient to identify falls from other movements. Code optimisation and overclocking of the ESP32s could help mitigate this issue. The firmware for the ESP32 MAC layer is also closed source, meaning there is limited information about the specifics of the Wi-Fi SDK. There are currently projects which are reverse-engineering the hardware and software of the ESP32 [11] and it is likely that further optimisation or increased CSI collection capabilities could be possible when more is publicly known about the board. Most existing research [8,9,10] uses open source hardware and software is able to achieve a significantly higher rate of CSI collection, albeit at a higher cost and power consumption. Despite these limitations, the system was capable of accurate motion detection and could be still applied to care homes or domestic homes in the form of an autonomous sleep monitor. The system was not developed to perform in real time, however with further development a real time solution could be achieved. The Raspberry Pi server is also capable enough to process small amounts of data, so a simple motion detection algorithm could be implemented. A home monitoring system could then be implemented by attaching a physical alarm to the Raspberry Pi data server, or a system could be implemented to send a notification to a mobile phone if movement was detected.

References

[1] Elizabeth R. Burns, Judy A. Stevens, Robin Lee “The direct costs of fatal and non-fatal falls among older adults — United States,,” https://doi.org/10.1016/j.jsr.2016.05.001

[2] R. Bush, “Falls in the over 65s: facts, numbers and trends,” Felgains, Aug. 03, 2021. https://www.felgains.com/blog/just-how-likely-am-i-to-experience-a-fall-falls-in-the-elderly-facts-numbers-and-trends/

[3] B. J. Vellas et al., “One-leg standing balance and functional status in a population of 512 community-living elderly persons,” Aging (Milan, Italy), vol. 9, no. 1–2, pp. 95–98, Feb. 1997, doi: https://doi.org/10.1007/BF03340133.

[4] I. Molyneaux, “How to buy the best personal alarm,” *Which?*, Jan. 03, 2024. https://www.which.co.uk/reviews/assistive-technology/article/how-to-buy-the-best-personal-alarm-aiDEp0U1Tdr2

[5] “Wireless router penetration in the UK 2007-2021,” Statista. https://www.statista.com/statistics/272754/wireless-router-take-up-in-the-united-kingdom-uk/

[6] Z. Yang, K. Qian, C. Wu, and Y. Zhang, “Understanding of Channel State Information,” *Springer eBooks*, pp. 11–21, Jan. 2021, doi: https://doi.org/10.1007/978-981-16-5658-3_2.

[7]“Wi-Fi Driver - ESP32 - — ESP-IDF Programming Guide latest documentation,” *docs.espressif.com*. https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-guides/wifi.html#wi-fi-channel-state-information

[8] S. Ji, Y. Xie, and M. Li, “SiFall,*Practical Online Fall Detection with RF Sensing*, Nov. 2022, doi: https://doi.org/10.1145/3560905.3568517

[9]Y. Hu, F. Zhang, C. Wu, B. Wang, and K. J. R. Liu, “DeFall: Environment-Independent Passive Fall Detection using WiFi,” *IEEE Internet of Things Journal*, pp. 1–1, 2021, doi: https://doi.org/10.1109/jiot.2021.3116136

[10]F. Zhang, C. Chen, B. Wang, and K. J. R. Liu, “WiSpeed: A Statistical Electromagnetic Approach for Device-Free Indoor Speed Estimation,” *IEEE Internet of Things Journal*, vol. 5, no. 3, pp. 2163–2177, Jun. 2018, doi: https://doi.org/10.1109/jiot.2018.2826227

[11] J. Devreker, “ESP32 open MAC,” *ESP32 open MAC*, 2024. https://esp32-open-mac.be (accessed Dec. 21, 2024).

[12]“Atheros CSI tool,” *Wands.sg*, 2023. https://www.wands.sg/research/wifi/AtherosCSI/ (accessed Dec. 22, 2024).

Appendix

Github Repository with code: https://github.com/riceboat/CSI-FI