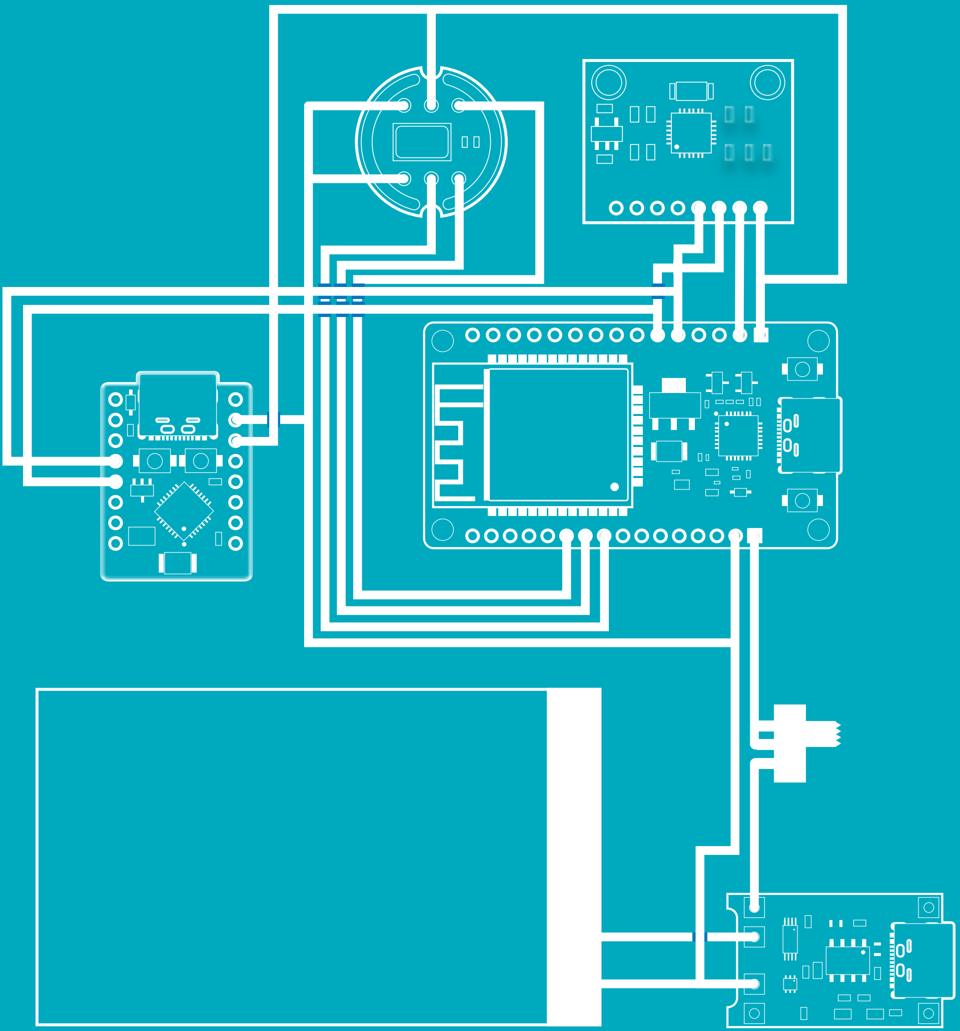
Privacy-oriented Wearable Data Acquisition for MMLA

Sensor & Modalities

BAP 2024 Q1/Q2

Micha Hoogendoorn
Lub Ras

Delft University of Technology



TU Delft

Privacy-oriented Wearable Data Acquisition for MMLA

Sensor & Modalities

by

Micha Hoogendoorn
Lub Ras

Project Duration: October, 2024 - December, 2024
Faculty: Electrical Engineering, Mathematics and Computer Science, Delft

TUDelft

Abstract

This project addresses the challenge of monitoring large, dynamic classrooms by proposing a privacy-oriented multimodal data acquisition system tailored for MMLA. Traditional learning analytics rely on unimodal data and fail to capture complex classroom interactions. In contrast, MMLA leverages multiple data sources to better understand learning behaviors. Current systems lack adaptability, user-friendliness, and privacy considerations, impeding their integration into classrooms. The proposed system comprises static and dynamic nodes, with dynamic nodes worn by individuals and static nodes strategically placed in classrooms. Data features, selected on MMLA relevance, are transmitted wirelessly to the static node for storage and analysis. Privacy is prioritized by avoiding sensitive data collection and adhering to GDPR guidelines. The design ensures adaptability, supporting additional sensors and seamless integration into various educational settings. This foundational system enables future research while addressing ethical and technical challenges in large-scale classrooms.

Preface

This thesis is part of the Bachelor Graduation Project at TU Delft, aiming to fulfill the requirements to obtain the Bachelor of Science in Electrical Engineering. The project was proposed by Dr. Abdikivanani and Dr. Dauwels, who have shown a growing interest in the field of Multimodal Learning Analytics (MMLA). Its objective is to develop a data acquisition system that can serve as a foundation for future research in MMLA. Given their expertise, the development of Machine Learning algorithms falls outside the project's scope. The ultimate goal is to improve learning outcomes in large, dynamic classrooms like Tellegen Hall.

Due to the nature of the BAP-project for which this thesis is written, we recommend to also read the thesis of the other sub group –our loyal fellow students– to improve contextual understanding of this thesis.

We would like to extend our gratitude to Dr. Abdikivanani and Dr. Dauwels for their supervision throughout the whole project, dedicating their time to offer support and guidance when needed.

We also wish to thank Mr. Lager for his exceptional effort in coordinating the project.

Micha Hoogendoorn
Lub Ras
Delft, January 2025

Contents

1 Introduction

Teachers often encounter difficulties in monitoring large, dynamic classrooms, particularly when students are mobile or when the number of students increases. The limited vision makes it difficult to maintain a broad overview of students and their behaviour, which is essential for providing personalised, targeted support. This issue mainly occurs in spaces such as Tellegen Hall at TU Delft (Figure 1.1), as well as in other educational settings that involve practical tasks.

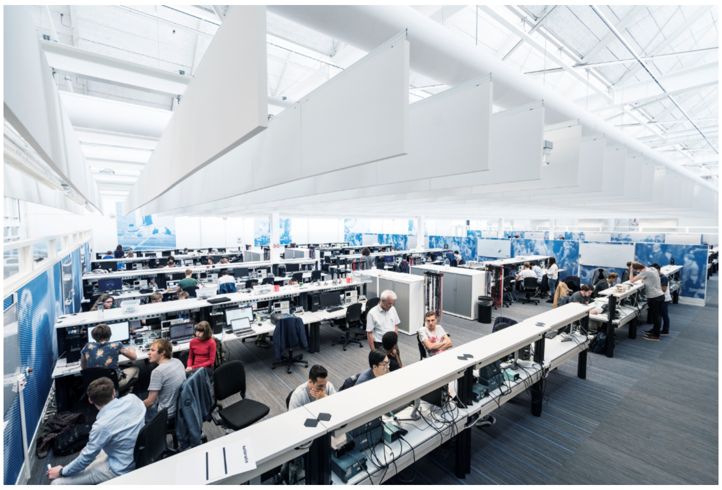


Figure 1.1: Overview of Tellegen Hall [1]

Despite the increase in educational tools, current technology has not yet been able to solve this problem. Interactive tools such as Kahoot! may improve engagement and learning experience, but they are too simplistic to automatically monitor students and accurately model their behaviour, as it is not their intended purpose.

Recent advancements in machine learning (ML) and sensing technologies have generated significant interest in their application within Multimodal Learning Analytics (MMLA). This domain involves the analysis of data collected from sensors using ML algorithms to monitor, understand and improve learning.

The field of MMLA introduces a new dimension to the educational setting by automating data collection and using ML for analysis. This approach proactively assists educators in identifying challenges, allowing for quicker and more accurate intervention. The goal is to enhance learning outcomes by identifying students who may require additional support, promoting higher level of engagement, and enhancing the overall educational experience.

These advancements present great potential, but they also raise important ethical and privacy concerns, related to the collection and use of sensitive data like video, audio, and physiological signals. Addressing these concerns is crucial for ensuring that the technology is both effective and responsible.

Therefore, **a privacy-oriented multimodal data acquisition system will be designed, tailored for large, dynamic learning environments**. The system will adhere to the ethical guidelines set by the Human Research Ethics Committee (HREC) at TU Delft, which is responsible for safeguarding participants' rights and privacy. This includes minimising the sensitivity of the collected data, in compliance with GDPR regulations.

The system does not currently employ Machine Learning. Instead, it functions as a foundational research product, designated to be adaptable for future modifications, allowing for adjustments or the integration of additional features to support research and development in this area.

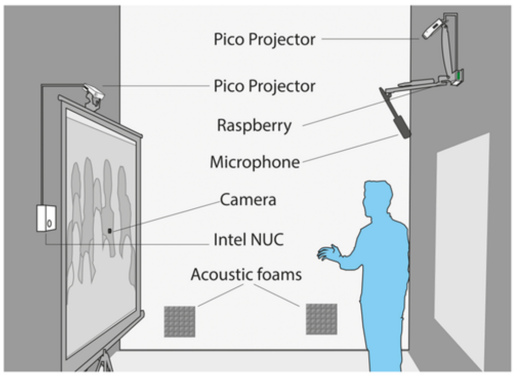
This chapter provides an overview of the project, beginning with a brief introduction to the field of MMLA (section 1.1). It addresses the current challenges and concerns in the field (section 1.2), followed by an analysis of existing systems (section 1.3), highlighting their shortcomings. This leads to the problem definition (section 1.4), followed by our proposed design as a solution to the problem(section 1.5). The final section (section 1.6) outlines the structure of the report.

1.1. Multimodal Learning Analytics

Learning analytics is an emerging field focused on understanding and improving education through analysing and visualising data. Traditional learning analytics tools have primarily concentrated on collecting data as a series of actions performed by an individual student, such as clicks or keystrokes on a computer [2]. This approach simplifies the presentation of data in charts or graphs, thereby assisting teachers in refining the learning process. These tools are usually integrated into existing digital systems, such as a computer. However, they are limited due to their reliance on unimodal data-data from a single source, such as video, audio, motion, or physiological signals [3]. As a result, these systems fail to capture the broader context and deeper interactions in classroom settings.

Conversely, Multimodal Learning Analytics (MMLA) addresses the shortcoming of traditional methods by integrating various complementary data sources. This integration provides a more robust foundation for identifying specific learning indicators and offers a thorough understanding of learning dynamics.

An example of MMLA in practice is the Automatic Presentation Feedback System (RAP), which illustrates how MMLA can improve oral presentation skills. In this scenario, participants present to a virtual audience while a camera, microphone, and slide-tracking software capture their performance. By extracting and analysing certain features, such as posture, vocal tone and slide text size, the system provides real-time feedback [4]. Similarly, another study [5] used motion sensors and video recordings to examine the impact of classroom surroundings on student collaboration. Features including participation levels, head movements, and interpersonal distances were extracted to analyse how table shapes affected group dynamics.



(a) Setup of the RAP system [4]

(b) Experimental setup of student collaboration study [5]

Figure 1.2: Examples of MMLA systems

To achieve its goal, an MMLA system needs to establish the learning indicator and its resulting behaviours. While indicators, such as student collaboration, are not measurable, related behaviours, such as speaking turns or eye contact are, which can be captured through data [6] Once these behaviours and their data traces are identified, the next step is to select modalities measure these behaviours effectively. For instance, audio can measure total speech duration [7], while video can capture gestures [8]. Selecting the right modalities ensures the collection accurate and meaningful data.

1.2. Challenges and concerns

Despite recent growth in MMLA research and advancements in sensing technologies and computational methods, MMLA has yet to reach its full potential in classroom settings. Two main challenges contribute to this gap.

- Firstly, teachers often lack experience with modern data collection and visualisation techniques, hindering their ability to use these tools in teaching [9],
- Secondly, existing MMLA systems may not align with teachers' instructional needs, as the educational field lacks an understanding on how to support teachers interpret and apply MMLA insights effectively [9]. As a result, systems are often designed without considering teachers' levels of technical knowledge, making them difficult to use for educators without technical training.

At present, not many MMLA systems effectively overcome these challenges by being both user-friendly and seamlessly integrable into classrooms without requiring technical expertise.

There is also added concern about the validity of results, as individuals change their behaviour when monitored. More obtrusive systems make natural behaviour less likely. Minimising obtrusiveness is desirable to obtain natural learning behaviour. However, the term ”unobtrusive” is interpreted in various ways across the literature, resulting in five distinct definitions [10]:

1. Interaction without main attention: Causes minimal disturbance to the user's routine.
2. Interaction using hard-to-notice hardware placement or design: Placing the hardware in subtle locations.
3. Interaction based on non-distracting data collection: Hardware might be visible, but data collection is non-distracting.
4. Interaction that is socially accepted: Designed to be socially comfortable and non-disruptive to others.
5. Interaction naturally integrated with the task: Does not alter original, natural interactions or experiences.

There is no universally accepted definition, leaving designers to prioritise aspects of unobtrusiveness. Restricting hardware design may limit the data collection. For example, subtle hardware placement might impede capturing full facial and body data. *Balancing unobtrusiveness with effective data-gathering is necessary.*

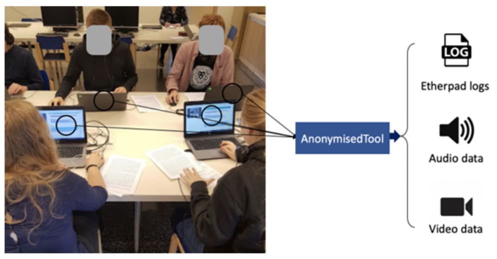
Last but not least, privacy concerns pose a significant challenge in MMLA research, particularly with data sensitivity and storage [11]. Researchers need ethics committees' approvals such as the HREC at TU Delft, to address potential risks and, comply with data protection laws such as GDPR. Despite this, there has been limited research on privacy and ethical considerations in MMLA [12], and few systematic frameworks exist to mitigate these risks [13].

While it may be challenging to create a universal approach to ethical concerns, existing MMLA systems have limited measures to minimise data sensitivity or prioritise anonymisation. *This gap indicates a need for systems that explicitly address these privacy risks, which could simplify the process of obtaining ethical approval and support broader adoption in educational settings.*

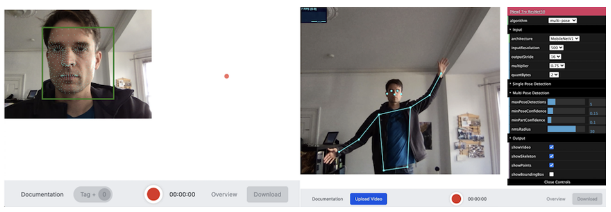
1.3. Available systems and tools

At present, there are several systems and research initiatives in the field of MMLA, though many of them are still in the research or prototype phase rather than being commercially available. These systems are still undergoing trials in various educational settings to assess their effectiveness. However, none of them fully address the issues highlighted in the previous section. By evaluating their design objectives and limitations, especially those encountered during data acquisition, we can gain useful insights. Since adaptability is an important requirement of the project, it will also be taken into account. The systems under consideration are as follows:

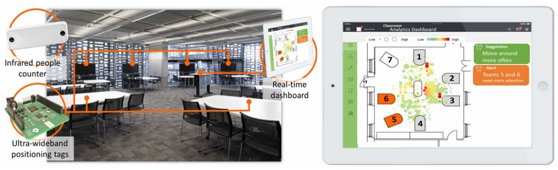
1. CoTrack - system or software that involves tracking or monitoring multiple entities and their interactions simultaneously through audio, video, and written logs. It features a graphical user interface for easy navigation. However, it lacks privacy considerations of the students during data collection [14];
2. EZ-MMLA toolkit – is a web-based application that uses video and audio recordings to generate data within the browser without transmitting sensitive information over the network. Data is collected and stored securely. While it has some privacy considerations, the collected data is sensitive. Additionally, integrating the toolkit into large, dynamic classrooms is complicated due to its reliance on computers for functionality[15];
3. Classroom prototype - system that provides a teacher with a heat map using localisation, proximity, and motion sensors. This allows the teacher to have an image of their interactions and shift attention to students who have not been visited recently. Even though privacy is not a primary concern, the collected data is less sensitive and excludes elements like video or audio. However, the system is not adaptable for broader research applications and operates similarly to an embedded system, dedicated to a single task [16];
4. Empatica E3 - a wearable data acquisition wristband that integrates 4 sensors (PPG, EDA, motion and temperature) to obtain valuable information regarding the individual's health. This device is lightweight and compact, making it minimally intrusive. However, the collected data is highly sensitive and may not be suitable for learning applications. Additionally, while learning applications might require different sensors, the system does not permit modifications [17].



(a) CoTrack [14]



(b) The EZ-MMLA toolkit [15]



(c) The classroom prototype with heat map [16]

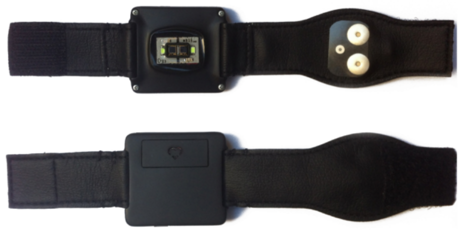

(d) Empatica E3 [17]

Figure 1.3: Available tools and systems

The assessment of current MMLA systems highlights a need for a more efficient solution that addresses privacy concerns, adaptability, and integrability in large, dynamic classroom settings. Existing tools often rely heavily on static components like computers, collect sensitive data that may not comply with GDPR guidelines , or lack adaptability for diverse research purposes. The proposed system should address these limitations to ensure that the same data acquisition platform can evolve to meet various research needs while maintaining reliability and robustness.

1.4. Problem Definition

MMLA aims to improve teacher visibility in large, dynamic classrooms. An adaptable data acquisition system is needed that stores data for future use in MMLA applications. The system should be adaptable to allow for later adjustments such as the addition of extra sensors and users. Additionally, it should not limit itself to measuring specific learning indicators, but instead utilise sensors that can capture various data traces.

Furthermore, addressing the challenges and concerns associated with MMLA requires developing a system that focuses on privacy and integrates easily into large, dynamic classrooms. A privacy-oriented approach relates to mitigating privacy risks, mainly regarding anonymisation and collection of sensitive data. This aligns closely with HREC guidelines. For effective integration, the system should not solely rely on static components.

Therefore, the problem definition can be stated as follows:

How can a privacy-oriented data acquisition system for MMLA be designed that is adaptable and integrates easily into large, dynamic classrooms?

The system is not intended to provide a standardised framework but rather to offer an adaptable solution that researchers can adjust and build upon. It does not guarantee HREC approval, nor does it claim that the specific selection of sensors is universally suitable for all types of learning indicators. Such decisions should be made by educators and future researchers.

The main objective of the system is to provide reliable and robust data acquisition, with the capability to store data for future analysis. By focusing on privacy considerations, adaptability and integrability, the system can assist researchers in achieving their goals in large, dynamic classrooms, regardless of what those may be. As unobtrusiveness is an added concern, it will be defined here as ”data collection that occurs in a non-distracting way”, allowing some flexibility in data collection methods.

Considering the system’s adaptability, it is essential to provide clear performance specifications, such as data output rate, the total amount of data it can handle and the student support capacity. This will ensure that researchers are aware of its limitations and can tailor the system to meet their specific needs.

1.5. Design

The proposed design addresses all requirements outlined in section 1.4, with its structure illustrated in Figure 1.4. To effectively monitor students in large, dynamic classrooms, students and teachers will be equipped with small, wearable devices containing sensors that record data. These wearable devices are referred to as the system’s dynamic nodes. Since these devices handle all the data collection, it reduces the need for significant classroom modifications.

In large classrooms with many participants, all data will converge at a central point, known as the root node. The root node, which remains in a fixed location, communicates directly with a server to anonymously store the data for future analysis. Dynamic nodes transmit their data wirelessly to the root node when within range. To handle scenarios where direct communication is not possible, the classroom will also include relay devices called static nodes. These static nodes forward data from the dynamic nodes to the root node and record data to capture additional interactions. The placement of static and root nodes should be strategic provide wide network coverage and can be adjusted to the teacher’s needs.

This communication and storage infrastructure forms the system’s backbone, facilitating reliable data transmission and robust collection. The system must support the addition or removal of nodes without impacting its overall functionality, and maintaining ease of setup. The system requires a minimum configuration of 2 static and 2 dynamic nodes for demonstration purposes. A Graphical User Interface (GUI) will be developed to display system status information, including active nodes and incoming messages.

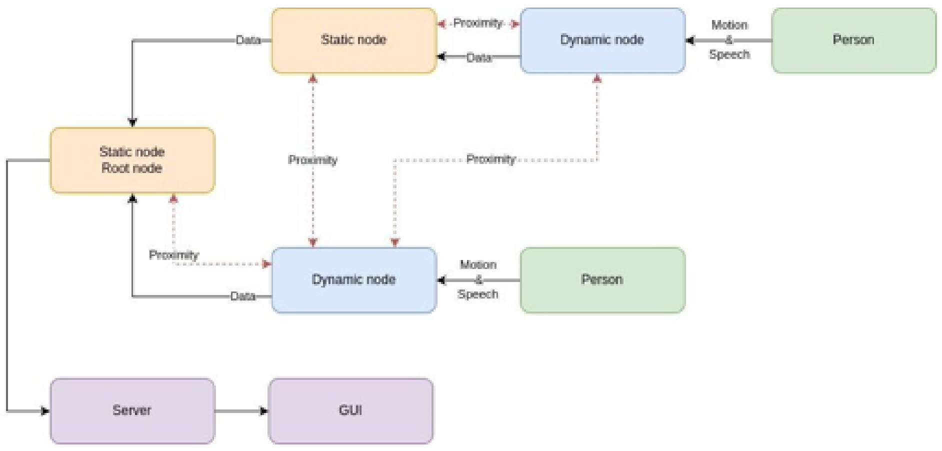


Figure 1.4: Black box diagram of the system

Each node in the system is essentially a microcontroller equipped with various sensors, capable of wireless communication and initial processing of sensor data. Once processed, the data is packaged into packets for transmission. Figure 3 illustrates this design in more detail. The selection of sensors should be in line with GDPR guidelines, avoiding collection of sensitive data. This includes pre-processed audio (where no identifiable information is stored), as well as non-intrusive sensors like proximity and motion sensors. A more detailed review of the sensor selection strategy will follow later. The static nodes only use proximity sensors, since they are not attached to talking individuals. To ensure adaptability, the system and data packages should accommodate the integration of additional sensors through allowing for a high data throughput.

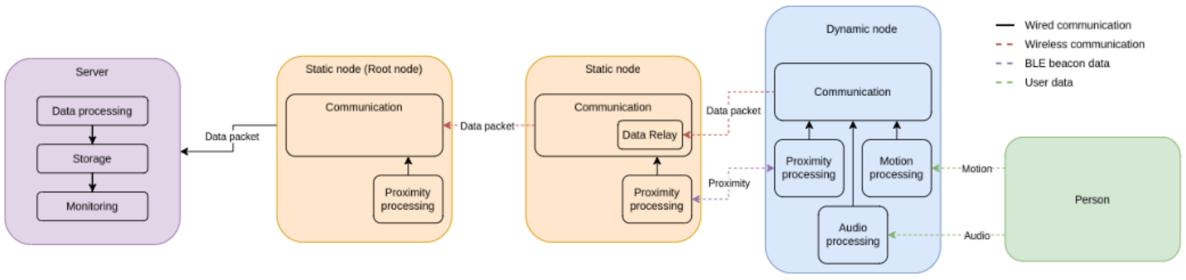


Figure 1.5: More detailed block diagram of system

The system is divided into two main modules: **Wireless Communication & Data Management (WCDM) and Sensors & Modalities (SM)**, each managed by a dedicated subgroup.

1.5.1. Wireless Communication & Data Management (WCDM)

The WCDM subgroup focuses on the communication between the nodes and the management of the data that is being transferred. The communication will be done wirelessly to keep the system unobtrusive. The subgroup must find suitable wireless technology that meets the requirements. At server level, the incoming data should be processed, stored, and monitored to allow for simple data extraction.

1.5.2. Sensors & Modalities

The SM subgroup is responsible for selecting sensors and modalities that can capture learning indicators. This selection must consider the privacy of the participants as it should not include sensitive data. The implementation of sensors and modalities should be kept as unobtrusive as possible. This will be a challenge for the dynamic nodes. The subgroup is also responsible for processing the sensor data and packaging it into data packets suitable for transmission.

1.6. Thesis overview

This report addresses the wireless communication and data management aspects of the project.

Chapter 1 provides an introduction to the project, including some background information, a state-of-the-art analysis, the problem definition and proposed solution.

The subsequent chapter outlines the specific requirements that the system must meet.

Chapter 3 details the design choices made during the project. Chapter 4 addresses the tangible implementation of these design choices, supported by testing results of various design elements. Chapter 5 covers the overall implementation of the system and the results that validate its performance.

The discussion segment presents the significance of the findings and future work. The conclusion summarises the key points of the project.

2 Program of Requirements

The specific requirements are set to develop a data acquisition system contributing to MMLA purposes according to the problem definition stated in Chapter 1.

The main takeaways and Program of Requirements (henceforth 'PoR') have labels to describe the different requirements. **TA** is for take-aways, **MR** is for mandatory requirements, **TR** for trade-off requirements, **BC** for boundary conditions.

2.1. Main Take-aways PoR

- **TA1** The end-goal of the system is to do data acquisition and storage for MMLA purposes.
- **TA2** The system should collect data in a non-distractive way.
- **TA3** The system should store data ready to be further analysed.
- **TA4** The system should be easily integrable for users in dynamic classroom settings.
- **TA5** The system should implement a privacy-sensitive sensor selection strategy that minimises data sensitivity while obtaining useful data.
- **TA6** The cost should be minimized when designing a data acquisition and storage system.
- **TA7** The system should be designed as a foundational research product, intended to serve as basis for further research.

2.2. General PoR

To be able to accomplish the needs of this product a few requirements should be adhered to. Both hard, mandatory and soft, trade-off requirements are listed below. Notice how the TA-codes recorded at the end of the line are referring to the corresponding take-aways in section 2.1.

2.2.1. Mandatory Requirements

The product must

- **MR1** Extract data features from multiple sensors on the nodes. [TA1]
- **MR2** Communicate the data features wirelessly between the nodes. [TA2]
- **MR3** Receive and store the data features in the database. [TA1, TA3]
- **MR4** Enable the extraction of the data in the database for use in MMLA. [TA1, TA7]
- **MR5** Be extendible in terms of sensors and nodes. [TA7]
- **MR6** Not have name-related user ID's. [TA5]
- **MR7** Cost less than €500. [TA7]
- **MR8** Operate shortly after being powered on, with no manual configuration required. [TA4]

- **MR9** Should not require human intervention after setup. [TA4]
- **MR10** Not collect sensitive data defined by the GDPR guidelines. [TA5]

2.2.2. Trade-off Requirements

Also, the product should preferably

- **TR1** Have little visibility when collecting data [TA2]
- **TR2** Be light and small. [TA2, TA6]
- **TR3** Be scalable in range. [TA7]
- **TR4** Be usable for individuals without extensive technical knowledge. [TA4]
- **TR5** Cost as low as possible. [TA6]
- **TR6** Have low power consumption. [TA2, TA6, TA7]
- **TR7** Take less than 20 minutes to setup. [TA4]
- **TR8** Collect data in a non-distractive way. [TA2]

2.2.3. Boundary Conditions

Looking at main take-away #1 (section 2.1); the focus of this project is to do data acquisition and storage for MMLA purposes. This entails the following:

- **BC1** The system will be tested using 2 static and 2 dynamic nodes. [TA1]
- **BC2** Analysing data and reasoning using Machine Learning is out of scope. [TA3]
- **BC3** Data security will be out of scope. [TA5]

2.3. Sensors & Modalities PoR

For the sensors and modalities of the project, there are some requirements in addition to the general PoR, these are listed below, labeled with an extra **S** to distinguish from the general PoR. The choices for some ”random” specific requirements will be further clarified in the upcoming chapters.

2.3.1. Mandatory Requirements

The sub-product must

- **SMR1** Be connected to the whole system setup. [TA1]
- **SMR2** Output at least 4 sensor features that contribute to MMLA. [TA1]
- **SMR3** Read and process data from the microphone and 6-axis accelerometer simultaneously. [TA1]
- **SMR4** Broadcast Bluetooth® for proximity purposes to nearby nodes.
- **SMR5** Measure at least once a second (1 Hz polling rate).
- **SMR6** Have a data measurement window of at least 100 milliseconds long.
- **SMR7** Combine the microphone and 6-axis accelerometer data streams into one output data package stream.
- **SMR8** Have data packages of at most 200 Bytes.
- **SMR9** Communicate wirelessly with the data management (WCDM) module. [TA2]
- **SMR10** Be extendable in terms of value-adding sensors. [TA1, TA3]
- **SMR11** Have an internal powering system. [TA2]

2.3.2. Trade-off Requirements

Also, the sub-product should preferably

- **STR1** Be easily set up by non-experts. [TA4, MR8, MR9]
- **STR2** Be scalable in configuration for larger group setups. [TA7, MR5]

- **STR3** Be light weight and small sized to be minimally invasive to wear. [TA2, TR1, TR2]
- **STR4** Have low costs to allow many sensor modules. [TA6, TR5]
- **STR5** Make only costs that are adding to the mandatory requirements. [TA6, TR5]
- **STR6** Have a rapid and solid system start-up procedure. [TA4, TR7]
- **STR7** Be power efficient when in an inactive state. [TA6, TR6]

2.3.3. Boundary Conditions

- **SBC1** The outputted sensor data will not be further analysed by this system. [TA3]
- **SBC2** The sub-product will be tested based on 2 students/users. [TA1, BC1]

3 Design

3.1. Introduction

Considering the overarching project structure, the main target of the Sensors & Modalities (SM) subgroup is to design and implement a wearable (dynamic node) to measure useful data. To acquire the useful data, different sensors will be used and their important features will be extracted. These features, relevant for MMLA, need to be packaged and sent wirelessly to the data server. In order to reach these targets, the electronic components have to be chosen based on the PoR (chapter 2).

3.2. Micro controller

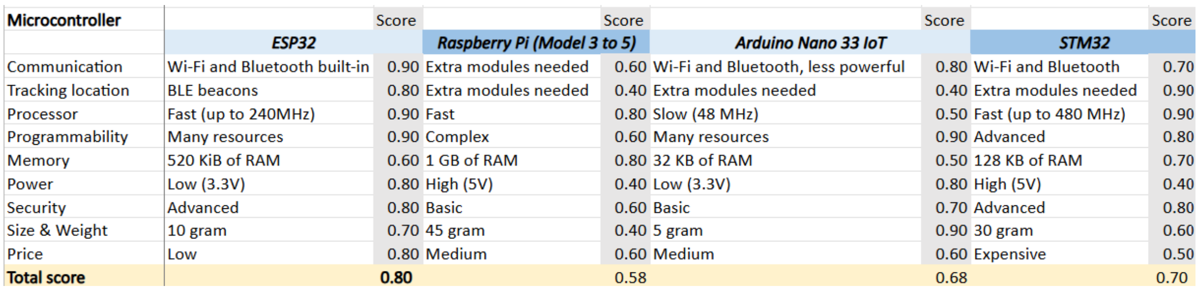
All of the subgroup's requirements can not be fulfilled without using some sort of micro processor to do feature extraction, data packaging, and package transmission. A micro controller is needed to control the signal transfer between input and output. Hence, a micro controller unit (MCU) is selected based on the PoR of chapter 2 and is preferably chosen in cooperation with the WCDM subgroup to allow integration flexibility; adhering to requirement SMR1 (subsection 2.3.1).

STM32 "The STM32 family of 32-bit micro controllers based on the Arm Cortex®-M processor is designed to offer new degrees of freedom to MCU users. It offers products combining very high performance, real-time capabilities, digital signal processing, low-power / low-voltage operation, and connectivity, while maintaining full integration and ease of development" [18].

Arduino Nano 33 IoT The Arduino Nano is a small sized controller that is working with I^2S protocol. This model can feature either a Wi-Fi® module or Bluetooth® Low Energy (BLE) and several environment sensors, dependent on the considered extension version. The Arduino Nano 33 IoT model specifically [19] is very small and light weight (5 gram), but is equipped with a relatively slow processor (48 MHz).

Raspberry Pi The Raspberry Pi model 3 to 5 [20] is also considered as a MCU. Since this unit can be classified as a full-fledged computer system, it is too powerful for the task. As a result, simple tasks are executed less efficiently and much power is unnecessary consumed. Even though the unit is already relatively large, it needs module extensions to do wireless communication. This makes the Raspberry Pi inapt for pre-processing acquired data on a wearable.

ESP32 The ESP32 [21] has a relative powerful chip (up to 240 MHz), so analysis can be done on the processor itself. It supports both BLE and WiFi connectivity; for which it even has its own ESP-NOW protocol to allow for easy implementation. In addition, location tracking –elaborated more in section 3.3.3– can be implemented without the need for extra parts, contrary to the other MCU options. Low power consumption (3.3V) and the compact character (10 gram) of the module are also advantageous. Compared to other micro controllers (see Figure 3.1), ESP32 scores the highest and is most suited to use for a prototype.



Microcontroller		Score		Score		Score		Score
	ESP32		*Raspberry Pi (Model 3 to 5)*		*Arduino Nano 33 IoT*		*STM32*	
Communication	Wi-Fi and Bluetooth built-in	0.90	Extra modules needed	0.60	Wi-Fi and Bluetooth, less powerful	0.80	Wi-Fi and Bluetooth	0.70
Tracking location	BLE beacons	0.80	Extra modules needed	0.40	Extra modules needed	0.40	Extra modules needed	0.90
Processor	Fast (up to 240MHz)	0.90	Fast	0.80	Slow (48 MHz)	0.50	Fast (up to 480 MHz)	0.90
Programmability	Many resources	0.90	Complex	0.60	Many resources	0.90	Advanced	0.80
Memory	520 KiB of RAM	0.60	1 GB of RAM	0.80	32 KB of RAM	0.50	128 KB of RAM	0.70
Power	Low (3.3V)	0.80	High (5V)	0.40	Low (3.3V)	0.80	High (5V)	0.40
Security	Advanced	0.80	Basic	0.60	Basic	0.70	Advanced	0.80
Size & Weight	10 gram	0.70	45 gram	0.40	5 gram	0.90	30 gram	0.60
Price	Low	0.80	Medium	0.60	Medium	0.60	Expensive	0.50
Total score		**0.80**		0.58		0.68		0.70

Figure 3.1: Comparison table of considered MCUs

3.3. Sensor selection

A privacy-sensitive sensor selection strategy has been used to minimize data sensitivity while obtaining useful data; as intentionally planned in the main project takeaways (TA5; section 2.1).

3.3.1. Voice recording

Based on research [22] emotions and mood can be monitored well using speech analysis. Although the combination of "both audio and visual modalities contribute to express emotions" [22, p. 53], raw video data is not privacy oriented and will therefore not be utilized.

The research of Viswanathan and Vanlehn [23] shows that the degree of collaboration, cooperation and asymmetric contribution can be measured with accuracies between 85% and 96%. For example, trans-activity (i.e., the extent to which learners build on each other's reasoning) is a property of joint problem solving; this cooperation-related characteristic can be monitored by speech recording.

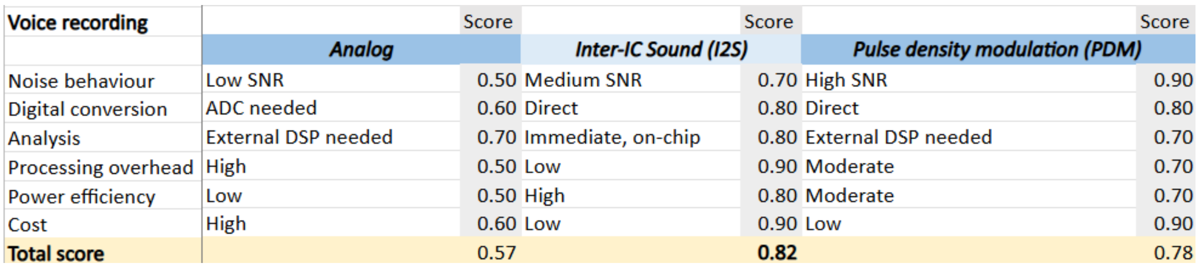
Earlier work of Zhou et al. [24] has shown that collaboration quality and creative fluency can be measured by means of speech analysis. They underline the importance of examining the relationship between social behavior and reaching the full potential of collaborative designing and -learning. "Considering that researchers have identified strong associations between interaction patterns and collaborative engineering design outcomes ..., there is a critical need to examine the relationships between social interaction processes and collaborative design outcomes in different spatial and material contexts" [24, p. 2]. In fact, conducting analysis on speech time series data has often been used in context of social interaction research [25] [26].

Concluding from the research presented above, monitoring learning behavior using speech has a high data density. This means that a lot of information can be drawn from only a handful of data. Sound characteristics like pitch, spectral energy distribution, noise levels, speech duration, jitter and shimmer are some examples of helpful indicators for measuring learning behavior of the user. Feature selection and extraction will be discussed in section 3.4.2.

In the search after a competent microphone, several options have been considered and compared (see Figure 3.2):

1. **Analog microphone**, which leaves the incoming analog sound signal–that has a continuously and smoothly varying amplitude or frequency–unmodified.
2. **Inter-IC Sound (I^2S) microphone**, a digital microphone using a protocol to enable a serial bus interface specially designed for communicating digital audio data between integrated circuits (IC's).
3. **Pulse density modulation (PDM) microphone**, a digital microphone which has a simple PDM interface that generates digital audio signals; directly correlated with the original analog versions.

An I^2S microphone seems to be best for the wearable. Although PDM is much more resilient to electrically noisy environments, I^2S has a high signal to noise ratio (SNR) to provide relative noise free audio at the output. Because of internal processing, a digital signal is send out, so there is no need for analog to digital conversion afterwards, as is required for an analog microphone. While PDM signals require further processing by an external digital signal processor (DSP) or micro-controller, the I^2S protocol provides immediate analysis of the digital audio signal by the micro-controller. Therefore, the data rate of the audio signal is already at an acceptable level when it arrives at the DSP. This eliminates the need for additional components within the design for processing or conditioning the captured audio data. Al together, I^{2}S is likely to be the best way to reach the project target in relation to price-sensitive products that are wholly self-contained and where energy efficient battery-powered operation and compact integration are a prerequisite (TA2, TA6, TR6; section 2.1 & 2.2.2).



Voice recording		Score		Score		Score
	Analog		*Inter-IC Sound (I2S)*		*Pulse density modulation (PDM)*	
Noise behaviour	Low SNR	0.50	Medium SNR	0.70	High SNR	0.90
Digital conversion	ADC needed	0.60	Direct	0.80	Direct	0.80
Analysis	External DSP needed	0.70	Immediate, on-chip	0.80	External DSP needed	0.70
Processing overhead	High	0.50	Low	0.90	Moderate	0.70
Power efficiency	Low	0.50	High	0.80	Moderate	0.70
Cost	High	0.60	Low	0.90	Low	0.90
Total score		0.57		**0.82**		0.78

Figure 3.2: Comparison table of considered voice recording means

3.3.2. Motion tracking

Monitoring a student's posture and linear motion has been found insightful for measuring the extent of boredom and engagement in group learning cases [27], as engagement is closely related to common purpose, which manifests itself in the interaction of students "through repeated patterns and repetition of posture and through proxemics, the physical closeness and synchronic, aligned postural shifts" [28, p. 91].

Tracking posture helps map the user's learning behavior since good posture increases learning performance [29] [30] [31]. Research also shows that motion and posture monitoring supports the validation of speech measurements, since both verbal and non-verbal communication are correlated in case of consistent social learning behavior [32]. In order to draw correct and consistent conclusions with MMLA, both speech and motion needs to be measured simultaneously.

Several motion tracking options have been considered and compared:

1. **3-axis Accelerometer**, measuring total acceleration, including the static acceleration from gravity that would be present even when a student is not moving.
2. **3-axis Gyroscope**, measuring instantaneous angular velocity on each axis. Due to noise, it ensures high quality signals in the short term. However, since angular position has to be integrated off of the gyroscope, it will build up error over time.
3. **6-axis Sensor**, combining both the 3-axis accelerometer and gyroscope to allow for more accurate and context related measurements.

While the accelerometer can measure approximate posture and tilt–enabling step counting and activity tracking–these features are not enough to track the orientation of the sensor, which can be used to measure user orientation specifically. Due to angular velocity measurements, the gyroscope can add metrics to the feature list such as momentum to allow for rotation tracking and gesture recognition. It also gives a reference point for the accelerometer, allowing to measure the yaw rotation. The sensors complement each other well and provide additional information. Although this will infer an overall higher power consumption, it will not have significant impact on the costs. Also, the 6-axis sensor will not introduce any significant complexity for implementation. This leaves the 6-axis sensor to be the best match.

3.3.3. Proximity mapping

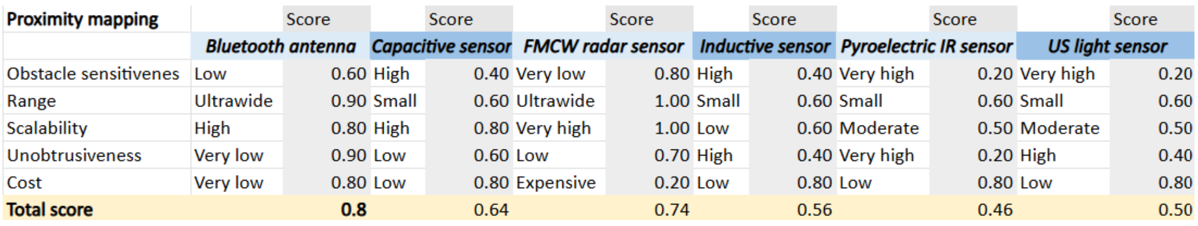
Measuring the distance between students and their relative proximity to educational equipment, such as a lab-desk with an oscilloscope on top, is found to be a good quantitative indicator of social aspects of teamwork. Hoegl et al. state that "team member proximity is positively associated to all three more socially oriented aspects of the teamwork quality construct, i.e., mutual support, work norms of high effort, and cohesion" [33, p. 1160]. Moreover, "team members in proximity are more likely to develop a stronger sense of group identity ... leading to increased effort on the common task" [33, p. 1157].

Reason enough to monitor and analyze proximity for MMLA objectives, making it a product requirement (SMR4, section 2.3.1).

Implementing this into this project's related terms: proximity has to be measured between dynamic nodes reciprocally and between dynamic- and static nodes. To implement this metric, several means have been considered and compared (see Figure 3.3):

1. **Bluetooth® (BT) antenna**, uses detection on signal strength to measure distance from received signals. Key parameters like Advertising Time, Advertising Interval, Scan Interval and Scan Window are determining how quickly a specific BT device can detect other BT devices in the vicinity.
2. **Capacitive sensor**, emitting an oscillating electric field between target and sensor that changes with distance .
3. **Frequency Modulated Continuous Wave (FMCW) radar sensor**, a panoramic localization and mapping tool, designed to measure distance and speed of objects within range. This is ideal for mapping proximity of high-speed moving objects or individuals [34].
4. **Inductive sensor**, emitting an oscillating electromagnetic field that is draining electric energy when a metal target approaches.
5. **Pyroelectric infrared (IR) sensor**, designed to sense heat exchange, considering only radiation in a certain domain, specific to human body thermal radiation waves (0.75 - 15 μm).
6. **Ultrasonic (US) sensor**, designed to emit US bursts to a target and receive the reflected wave. Traveling time is the key parameter to measure the distance.

Elimination of these six options is done considering the requirements in chapter 2. In a dynamic classroom, there will be many obstacles to be accounted for while mapping proximity (e.g. tables, lab equipment, walls, or individuals). Now, for the US- and the IR sensor, the target to which the distance has to be measured has to be in a direct line of sight of the sensor, introducing errors and non-complete measurements. Therefore, these two options are not found competent to do the job. This characteristic applies on capacitive- and inductive sensors too. In addition, inductive- and capacitive sensors can only measure distance within a short range. Due to this, it was deemed to be insufficient. Making these two options a mismatch. In case of the inductive sensor, an additional disadvantage is that only metallic objects can be observed, limiting the scalability and non-invasive character of the product. Meanwhile, two potential proximity sensing means are left considering: FMCW radar and Bluetooth® antenna. Both of these technologies do have a wide range of measuring distance. These allow for minimally invasive sensing too, since (radar) antennas can be placed out of sight. Although the FMCW radar can be placed centrally–not requiring extra equipment while scaling the local deployment of the product–its significant higher market retail price is the most outstanding property that is differentiating itself from the Bluetooth® antenna application; making the latter option the best match for proximity mapping, with a total score of 8/10 (see Figure 3.3).



Proximity mapping		Score		Score		Score		Score		Score		Score
	Bluetooth antenna		*Capacitive sensor*		*FMCW radar sensor*		*Inductive sensor*		*Pyroelectric IR sensor*		*US light sensor*	
Obstacle sensitivenes	Low	0.60	High	0.40	Very low	0.80	High	0.40	Very high	0.20	Very high	0.20
Range	Ultrawide	0.90	Small	0.60	Ultrawide	1.00	Small	0.60	Small	0.60	Small	0.60
Scalability	High	0.80	High	0.80	Very high	1.00	Low	0.60	Moderate	0.50	Moderate	0.50
Unobtrusiveness	Very low	0.90	Low	0.60	Low	0.70	High	0.40	Very high	0.20	High	0.40
Cost	Very low	0.80	Low	0.80	Expensive	0.20	Low	0.80	Low	0.80	Low	0.80
Total score		**0.8**		0.64		0.74		0.56		0.46		0.50

Figure 3.3: Comparison table with considered means of proximity mapping

3.4. Sensor implementation

3.4.1. 6-axis sensor

Sensor characteristics

The best 6-axis sensor on the market, that is both affordable and qualitative reliable, is the *MPU-6050*. To allow for direct sensor data readouts by the Integrated Development Environment (IDE) for programming purposes, this sensor model features three 16-bit analog-to-digital converters (ADCs) for digitizing the gyroscope outputs and three 16-bit ADCs for digitizing the accelerometer outputs. Communication with all registers of the device is performed using the I^2C protocol at 400 kHz.

For precision tracking of both fast and slow motions, the parts feature a user-programmable gyroscope full-scale range of ±250, ±500, ±1000, and ±2000°/sec (dps) and a user-programmable accelerometer full-scale range of ±2g, ±4g, ±8g, and ±16g. Where 'g' stands for gravitational acceleration, related to Earth's gravitational force. For example, an object at rest on Earth's surface is subject to 1g, equaling the conventional value of gravitational acceleration on Earth, about 9.81 m/s^2.

For power supply flexibility, the *MPU-6050* operates from VDD power supply voltage range of 2.375V-3.46V, as stated in the datasheet [35]. This means that direct powering from the micro controller 3V3 output is possible. Integrating with time, this allows for relative low power consumption; helping to meet requirement STR7 (section 2.3.2).

Feature selection

A 6-axis sensor was chosen to be able to monitor learning behavior by measuring the movements and body orientation of the user. Raw data that can be directly extracted from the sensor's registers entails a) orientation of the sensor and b) angular velocity. Multiple features can be drawn from these metrics, contributing to monitor user's movements;

- Momentum
- Pitch-, Roll-, and Yaw angle
- Approximate posture & tilt

This short list indicates the possibilities to uncover someone's level of activeness as movement behavior can be translated into someone's level of engagement in learning. Whether it is learning individually or within a group. However, only the Pitch-, Roll-, and Yaw angle (degrees) are calculated, packaged and transmitted. Based on these angles, the tilt of the device, and thus the approximate posture of the user wearing the device can be determined [31]. Leaving the data unprocessed allows for compact data packages that can be analyzed more efficiently afterwards (MR4, SMR8; section 2.2.1,2.3.1).

3.4.2. I^2S Microphone

Sensor characteristics

The sensor model *INMP411* found to be a good fit for a I^2S Microphone. It is a fully integrated, high performance, high SPL (i.e. Maximum sound pressure level without distortion), low noise, low power, analog output bottom-ported, omnidirectional MEMS microphone. "The *INMP411* consists of a MEMS microphone element and an impedance converter amplifier for audio signal processing. The *INMP411* sensitivity specification makes it an excellent choice for both near field and far field applications" [36, p. 1]. However, this means unwanted noise distortion because of far field recording, introducing a challenge that has to be dealt with during the design process. This will be touched upon in section 5.4.1. The low current consumption of the *INMP411* enables long battery life, ideal for the portability nature of the dynamic nodes (TR2, SMR11; section 2.2.2,2.3.1).

The datasheet also indicates that "the *INMP411* has a linear response up to 131 dB SPL. It offers high SNR and extended wideband frequency response resulting in natural sound with high intelligibility" [36, p. 1]. These statements will be checked by testing (see chapter 4.4) to guarantee the quality and usability of the product, stated in requirement MR4 (section 2.2.1).

Feature selection

A microphone was chosen in order to gain insights into learning behavior by analyzing speech data. In the previous section (3.3), the INMP441 I^2S Microphone has been selected to do recordings that can be allow for further feature extraction. Raw audio is coming into the micro controller. However, to prevent data trace-backs to individual users, some specific features need to be extracted before storage and further analysis. Therefore, no voice recording will be done in the strict sense of the word 'recording'. In other words, complete conversations of the users will not be stored.
This means that features, helpful for MMLA purposes, need to be extracted. Due to limited time to implement all the features, only the most usable ones will be extracted and sent (SMR2, section 2.3.1).

According to The Multimodal Learning Analytics Handbook [2], the top five speech features used in OPAFs –Multimodal systems for Oral Presentation Automated Feedback– are:

1) Filled pauses, 2) Volume, 3) Pitch, 4) Cadence, and 5) Empty pauses.

With filled pauses, ”the use of sounds or words that are spoken to fill up gaps in speech utterances (e.g. “ahm”, “uhm”)” [2, p. 63] are meant. Since the audio recording should be privacy oriented, this feature will not be extracted. Volume and Pitch are easily implementable features. Cadence; the velocity of speech, does not add more value to MMLA compared to empty pauses (moments of silence employed by the person speaking) since the rate of speech is personality specific and does not change that much over time. Empty pauses, however, are more valuable for MMLA and can be extracted by looking at noise levels and zero crossings [37]. To conclude this comparison, the following three features are selected to be extracted from raw audio data and subsequently sent to be stored and analyzed:

- Volume level of strongest frequency (dB)
- Pitch of strongest frequency (Hz)
- Zero crossings

3.4.3. Bluetooth® Antenna

Proximity mapping by use of Bluetooth® antenna requires one antenna per user such that mutual distance can be measured. Since every user is eventually equipped with a wearable that is controlled by a wireless communicative MCU, an on-board BLE antenna is automatically implemented. Hence, there is no need for extra modules since the ESP32 module covers BLE broadcasting, featuring an average typical adjacent channel transmission power of -56.7 dBm, according to the datasheet of the ESP32-WROOM-32 [21]. This feature of BLE beacon contributes to power efficient use while in an inactive state (STR7, section 2.3.2).

3.5. Design choices

To enable flawless measuring of motion, speech and proximity of the user, while keeping power consumption rates moderate, a compromise for transmitted signal power has been chosen. This means that data requests will be made by the server every second: a polling rate of 1 Hz. While the sensors are read out continuously, the features are extracted only once a second. In practice, this translates to a refreshment rate of incoming motion, speech and proximity data once a second (SMR5, section 2.3.1).

According to the requirements (SMR6, section 2.3.1), the measurement window has to be at least 100 ms long. This minimum window size is chosen to ensure the optimization of the sensor readings while maintaining data quality. To measure the microphone features, 1/8th of a second (125 ms) is used. 1 second (1000 ms) is used for reading out features of the 6-axis sensor and broadcasting BLE for proximity mapping.

Dependent on the use case, the load of connected devices can be extended tremendously (MR5, STR2; section 2.2.1, 2.3.2). To allow for many separate dynamic nodes in parallel, the size of the transmitted data packages should be limited. When sending out only required MMLA data, the package size should not exceed 200 Bytes. This restriction would help the other sub group with scaling to a high number of connected dynamic nodes. The total size of the data package –including motion, speech and proximity features– should end up way smaller than 200 Bytes. This would help the scalability as there would be plenty of room to add multiple extra sensors, as is a mandatory requirement (SMR10; section 2.3.1); at least 5 sensors for MMLA purposes can be appended.

4 Feature Extraction and Data Packaging

4.1. Introduction

In this chapter the chosen features from section 3.4 will be extracted from the data of the sensors. After the extraction of the features they will be packed into a unified structure ready to be send wirelessly. For this to deliver correct data first the sensors have to be tested to be working correctly. This can be read in Appendix A. All components will be gone through with their respective chosen features.

4.2. Micro controller

The ’beating heart’ of the dynamic node is the micro controller. It needs to work properly to ensure that all data is read and analyzed correctly. To verify whether the micro controller functions correctly, a few main functionalities of the micro controller –which are needed in order to extract the data and eventually send this in packages– are tested. These functionalities are:

- Reading inputs from pins (check if a pin has high voltage (3.3V).
- Setting a pin to high voltage (3.3V).
- Receiving data through serial connection.
- Being able to send and receive through Bluetooth®.

If all of these functionalities work correctly, the sensors can be read out and their data can be send wirelessly; hereby meeting requirements SMR3 and SMR9 (section 2.3.1). The testing of the functionalities of these points can be found in appendix A.1

4.3. 6-axis sensor

The first component that will be tested and validated is the MPU6050, a 6-axis sensor. Before different features can be extracted, this sensor itself must be tested to be found operational. Before the sensor can be used, the sensor working needs to be verified. This verification is done in appendix A.2.

4.3.1. Feature extraction

As decided upon in section 3.4.1, three features are extracted and send: The Pitch-, Roll-, and Yaw angle. These need to be tested to validate the working of the sensor. These three angles are related to the three coordinate axes –X, Y, and Z– which are fixed on the sensor (see Figure 4.1). The Roll angle is rotating around the X-axis and is thus pointing in the YZ-plane (see Figure 4.2). The Pitch angle is rotating around the Y-axis and is thus pointing in the XZ-plane. The Yaw angle is rotating around the Z-axis and is pointing in the XY-plane. Two of these angles could be calculated by using a polar coordinate system if there is a consistent known vector. This can be seen in [38]. The two important formulas are

$$\text{Pitch} = \arctan\left(\frac{a_x}{\sqrt{a_y^2 + a_z^2}}\right)$$

$$\text{Roll} = \arctan\left(\frac{-a_y}{a_z}\right)$$

These calculations for roll and pitch do have a downside. When moving they will not be as accurate as when stationary since the calculation are done based on stationary accelerometer data. These inaccuracies however should not be that big unless accelerating at high speeds. And the inaccuracies should never take that long as acceleration is usually only done in shorter bursts. Now only a way to measure the yaw is needed, and it was decided to do this by integrating the rotation speed. since integrating over time gives location. this will turn into the following formula

$$\text{yaw}(t) = \theta_0 + \int_0^t G_z(t)\, dt$$

Here θ_0 stands for the starting angle of the yaw and G_z stand for the rotation speed in z-axis. This can be simplified if yaw is measured in discrete increments of time. This will result in

$$\text{yaw}(t) = \text{yaw}(t-1) + G_z(t) * dt$$

Here yaw$(t-1)$ is the previous yaw angle and dt is just a discrete time interval. With this formula the yaw can go above 360 so this has to be accounted for. To keep it consistent with pitch and roll the yaw will be set from $-180°$ to $+180°$. The three features can now be calculated, allowing it to be tested. The code that was used to test them can be found in appendix C.2. To test the angles, the roll, pitch and yaw are measured for when each (x,y and z) axis is measured to have an angle of $90°$. The measurements according to the MPU6050 are then written down. So either the roll, pitch or yaw angles should be $\pm 90°$ in a perfect test. The results of the test are shown in the following Table 4.1.

Rotation used	measured Roll (°)	measured Pitch(°)	measured Yaw(°)
Roll angle +90°	91.3	0.5	2.0
Roll angle -90°	-89.3	3	0.2
Pitch angle +90°	-2.3	89.1	0.5
Pitch angle +90°	0.4	-89.8	-1.9
Yaw angle +90°	1.8	3	88.4
Yaw angle -90°	0.4	2.66	-87.88

Table 4.1: measurements of roll, pitch and yaw in different orientations

The results are not exact $90°$. But this can be explained by not angling the sensor exactly $90°$. The biggest difference is $2.12°$. This is still fairly accurate. The roll, pitch and yaw are working within acceptable ranges.

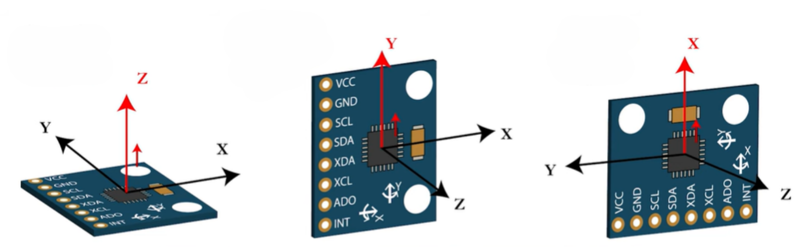


Figure 4.1: Coordinate system of the MPU6050; 6-axis sensor

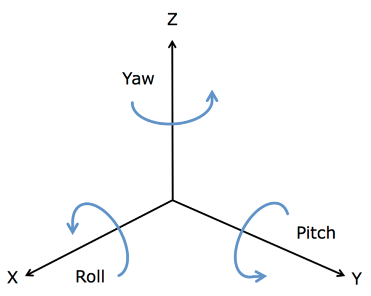


Figure 4.2: Orientation of Roll-, Pitch-, and Yaw angles

4.3.2. Data packaging

The roll, pitch and yaw are all three stored as a normal float. A float has a total size of 4 bytes. This will make the 3 features together a size of 12 bytes. It is possible to decrease the number of bytes however. This is because the floats are only ever within -180.00 to 180.00 degrees with a precision of 2 digits after decimal. It could even be stored as a int16 where last 2 digits are the 2 decimals. This would give the range of -327.68 to 327.67. Doing this would bring down the total byte size of the packet to 6 bytes since a int16 is only 2 bytes. Due to agreements with the other subgroup of the packets size this was not implemented as a 12 byte packets was already expect.

4.4. Microphone

The second component of which some features will be extracted is the microphone. But before this the functionality of the microphone needs to be certain and thus tested. This is done in appendix A.3

4.4.1. Feature extraction

As can be read earlier, three features from audio have been decided to be measured and extracted from the audio data.

- Volume; measured in dB
- Pitch or strongest frequency; measured in Hz
- Zero crossings.

The following subsections will be more in depth into how the different features are tested and validated.

Volume

The first metric that will be tested for and measured is volume. To measure the volume a measurement window needs to be chosen. To get the volume in this measurement window multiple paths can be taken. for example doing a root mean square measurement or make use of an Fourier transform [39], this will change the data from time domain to frequency domain. From the frequency domain you can get the average volume by getting the average magnitude of the frequencies. Since frequency detection will be needed later for the pitch detection, a Fourier transform will already need to be done. Because of this it was decided to use this second method to obtain the volume. So to get the volume first a measurement window is chosen for the Fourier transform. This measurement window needs to be big to be more precise and give an average over a longer period. But not too big as to eat up too much of the resources of the ESP32 as it needs to do other tasks as well. So the measurement window is a trade off that must be made. To limit the calculations needed for the Fourier transform. It was decided to make use of a Fast Fourier Transform (FFT) [39]. This reduces the complexity of the calculations from $\mathcal{O}(n^2)$ to $\mathcal{O}(n \log n)$. It was decided on a measurement window of 2048 samples. This gives no zero padding to the FFT, since it is a power of 2, and thus no redundant calculations need to be performed. This equates to a measurement time of 0.128 seconds.

$$\frac{samples \quad per \quad measurement}{sample \quad rate \quad (Hz)} = \frac{2048}{16000} = 0.128s$$

This means the average volume over a period of 0.128 seconds will be measured. As mentioned before, to get the Volume from a FFT, the average amplitude of all the frequencies needs to be rated. This can be done by summing up all the magnitudes of each frequency ($|f_n|$) and dividing this by the amount of samples (N).

$$\frac{\sum_{n=0}^{N/2} |f_n|}{N/2} = volume$$

The dividing by two happens because a FFT will mirror at the halfway point and thus the second half of the samples is redundant. The full code for testing for volume can be found in appendix C.1.2. The following code snippet is just how the volume calculation is done

```
        // Calculate average dB
        double totalVolume = 0.0;
        int volumeCount = 0;
        for (uint16_t i = 1; i < (samples / 2); i++) {
            totalVolume += vReal[i];
            volumeCount++;
        }
        double averageLinearVolume = totalVolume / volumeCount;
        double localAverageDb = 20.0 * log10(averageLinearVolume);
```

The amplitude of the frequencies obtained from the FFT corresponds to the volume because it represents the strength of the audio signal at each frequency. These magnitudes are unitless and indicate the relative intensity of the signal. While they provide a measure of the signal's power, the values are only relative to the microphone's sensitivity and do not correspond to absolute physical units like pressure or sound level unless calibrated. So in the end it will only give a relative volume for INMP441 microphones.

To put this all into practice and see if it works, a small test was conducted. The microphone was connected to the ESP32 and volume measurement started. the test starts with no sounds and from there slowly increase the volume from a constant audio source, for this a laptop was used. So the graph starts with the level of the noise and increases from there as the audio source volume increases. The laptop output audio will go from 0% to 100% in increments of 10%. The results can be seen in the Table 4.2.

Volume	0%	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
dB	34	40	44	55	61	64	67	69	71	72	73

Table 4.2: Audio volume VS dB-level

This was only a test to see if the volume metric can measure relative volume and it seems to do this well as the volume increases so does the decibel rate. Since the distribution of the output audio of the laptop is not known, the accuracy can not be obtained. More tests will be seen later.

Pitch

Second metric that will be tested and measured is pitch. This means the frequency with the highest magnitude. The already obtained FFT (section 4.4.1) can be used to extract the pitch. To do this, all frequencies must be passed by to check which frequency has the highest amplitude. How this is done can be seen in the following code snippet. The full code for testing of pitch can be seen in appendix C.1.2

```
        for (uint16_t i = 1; i < (samples / 2); i++) {  // Only positive frequencies
            double frequency = (i * SAMPLE_RATE) / samples;
            double magnitude = vReal[i];

            // Filter: Only consider frequencies between 20 Hz and 4000 Hz
            if (frequency >= 20 && frequency <= 4000) {
                if (magnitude > maxMagnitude) {
                    maxMagnitude = magnitude;
                    peakFrequencyLocal = frequency;
                }
            }
        }
```

Like for the volume, only the first half of the samples is tested since the second half is just a mirror from first half. A filter is put in place to only get frequencies from the human speech range since human speech frequencies is the most important for the wearable. Now a quick test can be done to see if it can measure the frequency well. This test will be done by a tone generator from a laptop. A few different frequencies are tested.

As can be seen in Table 4.3, the measured frequency is always close to the generated frequency but often has a slight difference. This slight difference can be explained by our sample size and the

generated frequency (Hz)	116	192	244	441	805	1318	2489	3971
measured frequency (Hz)	117	195	242	437	805	1318	2492	3968

Table 4.3: Detected frequency accuracy

limitations of a FFT. This is because a FFT has a frequency resolution based on the sample. This can be easily calculated by the following formula.

$$\Delta f = \frac{f_s}{N}$$

If the chosen values are put in this formula a resolution frequency of $\Delta f = 1600/2048 = 7.8125Hz$ is attained. Since it it will go to the closest frequency, which can be either just above or under the actual frequency, this can be divided by 2. This results in a maximum difference of $\approx \pm 4Hz$. This corresponds with the test where the biggest difference turned out to be 4Hz.

Zero Crossings
Final feature that needs to be extracted from the audio data is zero crossings. This can be done very easily by checking how often the audio data switches from positive to negative and vice versa. The following code snippet shows how this is done on the ESP32. Full code can be found in appendix C.1.2.

```
        // Zero-crossing count
        zeroCrossings = 0;
        for (uint16_t i = 1; i < samples; i++) {
            if ((vReal[i - 1] > 0 && vReal[i] < 0) || (vReal[i - 1] < 0 && vReal[i] > 0)) {
                zeroCrossings++;
            }
        }
```

Now a test is needed to see if it works. Since zero crossings can be highly variable due to background noise and even electrical noise on the signal it is hard to come up with a definite test. The test that was decided upon was to to test a very loud sound with a set zero crossings that will overpower all other sources. This can be done by generating a known frequency and just making it very loud. A frequency of 350Hz was decided upon. This frequency should have $f * t = frequency * measurement \quad time = 350 * 0.128 \approx 45$. It will probably not bring it this low but it should bring it down in comparison to white noise, which should have more zero crossings. The results of this test can be seen in Table 4.4.

	average Zero crossings
White noise	730
350 Hz	620

Table 4.4: Zero crossing test

A drop in zero crossings can be observed. This drop however does seem small, this could have many causes like the effect of electrical noise or just background noise is too big. To see if this metric will be effective, it has to be tested. This will be done in chapter 5.

4.4.2. Data packaging

The 3 different features of the microphone are tested. Now they need to be packaged together to be ready to send to the server. This needs to be done in a consistent and small package size. The current size of the features are listed in Table 4.5.

Volume and frequency are stored as doubles. But the precision of volume is not that important. This can be changed to an integer without much loss of accuracy. Since the volume will never be higher than the maximum for a uint16 (65535), it can be stored as a uint16. The same goes for frequency,

Feature	Data type	Data size
Volume dB	double	8 bytes
Frequency	double	8 bytes
Zero crossings	uint16	2 bytes

Table 4.5: Initial data sizes

so frequency can be stored as a uint16 as well. This does not cause any loss in accuracy since the resolution of the FFT itself is already bigger than 1 Hz. This means all three features can be stored as a uint16. The volume could be stored as a uint8, but the choice was made to keep it consistent with the other 2 features. This results in the following struct of microphone data seen in Table 4.6.

Feature	Data type	Data size
Volume dB	uint16	2 bytes
Frequency	uint16	2 bytes
Zero crossings	uint16	2 bytes

Table 4.6: Final data sizes

This saves a total of 12 bytes overall. This will be the data send to the server.

4.5. Bluetooth® antenna

The ESP32 micro controller is equipped with a Bluetooth® antenna that is found apt for proximity measuring (see section 3.3.3). To guarantee proper working and reliable data outcomes, this antenna needs to be tested and validated. This is done in appendix A.4

4.5.1. Feature extraction

Only sensing one possible other device is not that useful. To make it useful, all nearby ESP32's should be sensed. This, however, can already be done by the code that was used for testing the component. This is because it is searching continuously for Bluetooth® signals. This means, however, that it searches for all Bluetooth® signals. Since only the signals of other nodes are of interest, a simple filter is put in place. This filter will check if the received Bluetooth® device starts with a certain name. So this way, it will only send rssi data of named nodes. The full code for the feature extraction and testing can be seen in appendix C.3.2, this will include a part that will be explained in section 4.5.2. To do the final testing of the feature extraction, the code is uploaded on 5 ESP32s which were placed in close proximity. The results can be seen in the following Table 4.7. This concludes that multiple nodes can be measured at once; meeting requirement SMR4 (section 2.3.1)

ESP32 ID	Rssi in dBm
Dynamic_Node003	-63
Dynamic_Node002	-65
Dynamic_Node004	-70
Dynamic_Node005	-72

Table 4.7: Proximity test

4.5.2. Data packaging

Now the collected rssi data has to be packaged and send to the server. To do this it was decided to keep the amount of possible received devices limited to 10. This was decided to not inflate the data too much by sending too much proximity data. For the rssi data to be useful it needs to be send together with the node ID for that rssi. To do this two arrays are created one with node ID's and one with the rssi values. This needs to be done in a data efficient way. The nodes have consistent ID's starting with either ”Dynamic” or ”Static” followed by a ID number. To limit the size of the ID only the ID number will be used, the rest of the ID will be truncated. The ID is in the form of a string, so after truncating it is needed to be turned into an int to save the ID number more efficiently. if a uint8 is used for this ID number, a maximum of 255 nodes can be saved. If it is done like this the information of wether a static or dynamic node is will be lost. This is why it was decided to limit all device ID numbers starting with 2 for Static devices, and ID numbers starting with 0 or 1 for Dynamic devices. The following code snippet shows how the ID is turned into a ID number

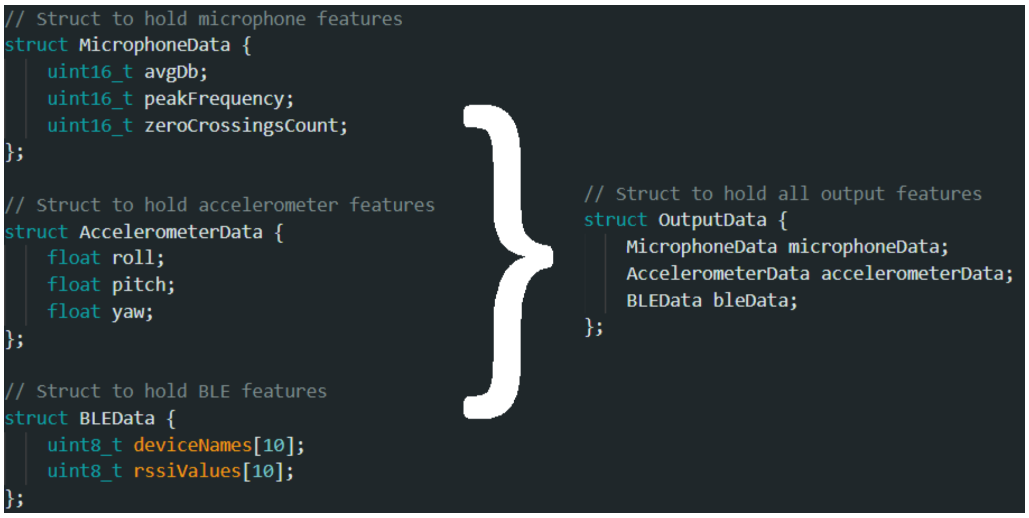
```
        if(name.startsWith("Dynamic") || name.startsWith("Static")){
        // Get the last 3 characters of the name (the digits)
        int lastThreeDigits = (name[length - 3] - '0') * 100 + (name[length - 2] - '0') *
            10 + (name[length - 1] - '0');

        bledata.deviceNames[i] = static_cast<uint8_t>(lastThreeDigits);
```

This way the device ID's can be saved as a uint8 which is only a single byte. Now the rssi still has to be send. Since the rssi will have values from at maximum around -30 and at minimum -120 dBm [21]. This could be send through a int8, but to be consistent with the data it was decided to take the absolute of the rssi instead and save it as a uint8 like the ID numbers. So in the end the data from the Bluetooth® antenna will be saved as 2 arrays of a lenght 10 with uint8 values, which gives a total size of 20 bytes.

4.6. Final data package

All the required features from 3.4 can be extracted. Now these need to be stored in a consistent format to meet requirement SMR7 (section 2.3.1). To do this, a new struct is created which contains the 3 previous structs. The following figure 4.3 shows how the final data package struct is created in code; resulting in a final package size of 38, which is way smaller than 200 Bytes as required (SMR8, section 2.3.1).



```
// Struct to hold microphone features
struct MicrophoneData {
    uint16_t avgDb;
    uint16_t peakFrequency;
    uint16_t zeroCrossingsCount;
};

// Struct to hold accelerometer features
struct AccelerometerData {
    float roll;
    float pitch;
    float yaw;
};

// Struct to hold BLE features
struct BLEData {
    uint8_t deviceNames[10];
    uint8_t rssiValues[10];
};
```

```
// Struct to hold all output features
struct OutputData {
    MicrophoneData microphoneData;
    AccelerometerData accelerometerData;
    BLEData bleData;
};
```

Figure 4.3: Final data package creation

Now that all the extracted feature data is packaged, it is ready to be sent to the server. The code to combine the codes and get the final package can be seen in appendix C.4. This will be done through the use of Wi-Fi® communication; documented by the other sub group.

5 System Integration

5.1. Introduction

In this chapter, the integration of the SM sub product will be discussed. Also the integration into the whole system is elaborated; the merging of the WCDM- and SM sub product into a fully functional product: A ”Privacy-oriented Wearable Data Acquisition Product for MMLA”; as was required (SMR1, section 2.3.1)

5.2. Design integration

All the sensors and components, as discussed in chapter 3 (Design), are combined, allowing to construct a setup that is able to independently send its acquired data packages to the WCDM sub product via a wireless communication protocol (SMR9, section 2.3.1). The total setup of the dynamic node can be seen in Figure 5.1; showing its ability to process both motion- and audio data from the user, and combine this with proximity rates of nearby user devices. These data points are continuously and separately measured, but regularly and simultaneously released by use of data packages for further processing.

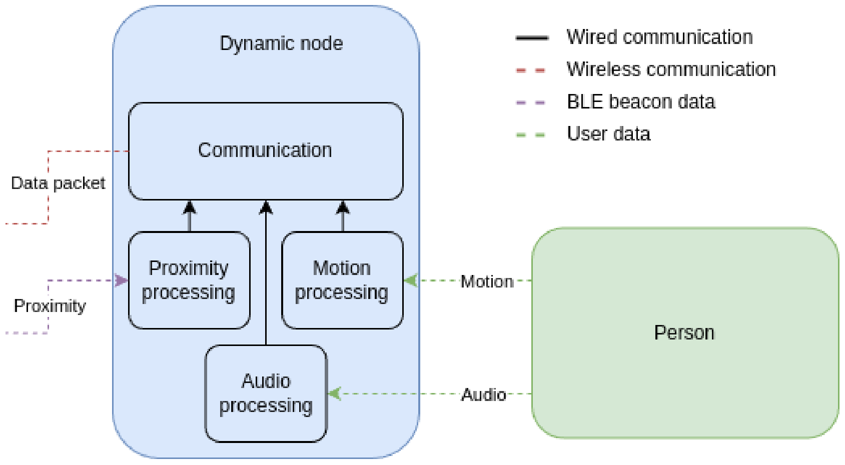


Figure 5.1: Block diagram of the dynamic node

5.3. Prototype building

Assembling a working prototype that can be worn on the chest by the user needs careful component placement. The costs of the total arsenal of components needed for building the prototype stayed within the agreed budget (under €200,-) as stated in the PoR (MR7, STR5; section 2.2.1, 2.3.2). After buying all the parts needed for building, the total design was soldered together unto a blue PCB; step by step.

1. **Female headers**; to easily disconnect both ESP32 controllers and develop the system.
2. **ESP32-DEV-38P**; main microcontroller for processing and communication.
3. **ESP32-DEV-16P**; for BT broadcasting, via ESP32-DEV-38P.
4. **MPU6050**; for 6-axis sensor readouts, via ESP32-DEV-38P.
5. **INMP441**; for audio recording, via ESP32-DEV-38P.
6. **TP4056**; charge circuit for 3.7V Li-Po battery cell.
7. **Switch**; for easily toggling between ’battery mode’ & ’uploading mode’.

All above mentioned components were wired together and then checked to be properly connected.

Since both wireless communication can not be done simultaneously on the same processing chip, a ESP32-DEV-16P is implemented to allow for broadcasting BLE® whilst at the same time transmitting data via WiFi on the ’main’ ESP32-DEV-38P controller. This additional ESP32 will act as a Bluetooth® module and will only communicate the received Bluetooth® signals. Since this is a ESP32 board too, the testing in section 4.5 is still relevant.

A powering circuit–3.7V Li-Po battery cell + TP4056 charge circuit–makes the dynamic node internally-powered and wearable; hereby meeting SMR11 (section 2.3.1).

A switch is used to prevent overloading the circuit with 5V or more. ’Battery mode’ allows the dynamic node to be powered internally and function substantive, without direct linkage with the computer. ’Uploading mode’ is used when data features are readout through USB-linking with the computer, or when new code is being uploaded to the microprocessor.

A schematic of the fully integrated sub product can be seen in Figure 5.2.

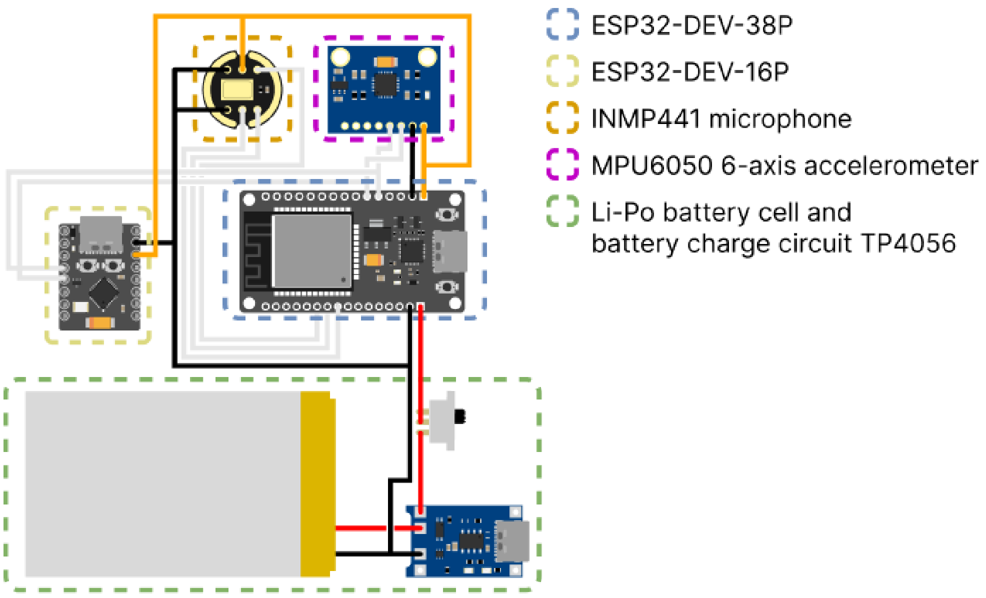


Figure 5.2: Wiring diagram of the dynamic node

5.4. Merging of both sub products

Now that the prototype is assembled, it can be integrated into the whole system and tested. To achieve successful data storage, the acquired data features need to be send first to a static node, and eventually to the server. This means that the dynamic node prototype has to be connected to the system of static nodes. When considering the WCDM sub product as a blackbox model (see Figure 5.3), the data features that are send in package form by the SM sub product are entering the blackbox at the input. At the output, the data features are stored at the server and can be displayed directly by the graphical user interface (GUI). Therefore, correct data feature transmission can be tested by reading out real-time measurements at the end of the blackbox model.

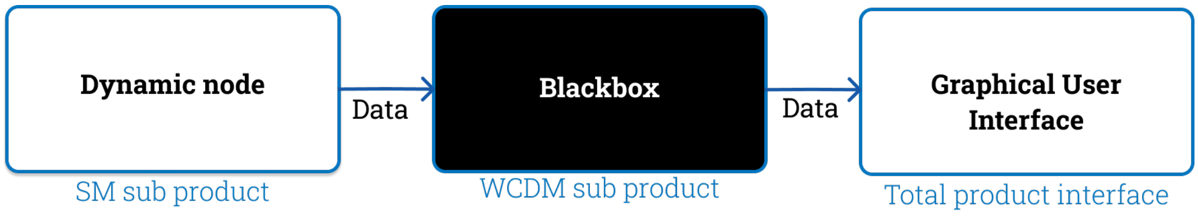


Figure 5.3: Blackbox model of the integrated system

5.4.1. On-body testing

Multiple tests have been done to see if the feature extracting, data packaging and sending this to the server, works. To do this, the dynamaic node is wore on the chest and the incoming data send to the server. All following tests have been read out from the gui made by the other sub group.

Proximity readouts

Mutual distance between two devices is measured by noting the incoming dBm signal. Three tests have been done in three different environments. The data has been read out from the GUI. The dBm data is an average over a longer time. As can be seen in table 5.1 the proximity data is send and

Distance (m)	0.5	1.0	2.0	5.0	10.0
Proximity (dBm)					
Tellegen hall	-75	-81	-93	-	-
Classroom	-72	-78	-82	-90	-94
Room in house	-71	-75	-78	-85	-

Table 5.1: Proximity readouts

received. It can also be noted that each session and location gives different results. This means that results should only be compared in comparison to different results in the same session. This test was done with only measuring 1 dynamic node. The performed test on larger scale, for a maximum of 10 dynamic nodes, can be found in appendix B.3.

Microphone readouts from GUI

Microphone tests have been carried out in different circumstances. Figures 5.4, 5.5 and 5.6 show combined snippets of the time series response when reading out the INMP441 at the GUI in three different cases:

- Silence; No conversations, only environmental noise.
- Talking; Close conversations, with environmental noise.
- Inactive; Background conversations, with environmental noise.

As is to be expected, sound amplitude differs a lot between silence and conversations of the users nearby (see Figure 5.4. A surprising result is that ambient conversations are easily separated from direct conversations. This facilitates future analysis by Machine Learning.

During silence, tilting forwards just a little bit can already introduce noise in high frequencies. It is not proven, but the red peak in Figure 5.5 is most likely due to scratching sounds from clothing or peeping

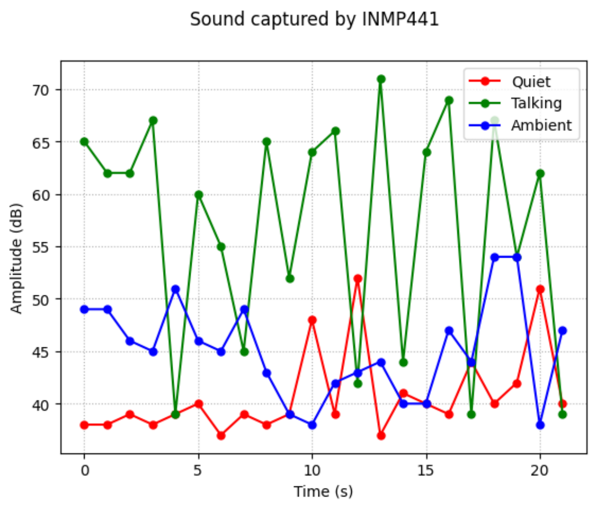


Figure 5.4: Amplitude (dB) measured during a) silence, b) close conversation, and c) background conversations

of a tilted chair. Little distortions like these are impacting the frequency spectrum and will eventually influence the 'communication status' of the speech recorded user. More filtering and analysis should be done in order to guarantee reliable audio measurements.

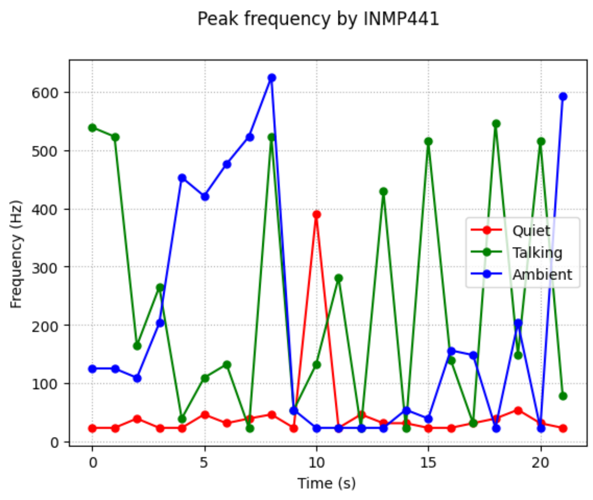


Figure 5.5: Highest pitch measured during a) silence, b) close conversation, and c) background conversations

Figure 5.6 shows the number of zero crossings per second during a 20 second period. No specific content-related conclusion can be made as the number of zero crossing is on average similar, regardless of the situation. The outlier can be attributed to a connection failure. When suddenly no audio is detected, or when the device is turned off, the measurements show deviant values.

More testing for the microphone can be seen in appendix B.1. It can be concluded from these tests that the microphone features are sent successfully and that the features are received by the server as intended; meeting MR1, MR2 and MR3 (section 2.2.1).

Accelerometer readouts from GUI

The last features that need to be tested and be able to be read out from the GUI are from the accelerometer. The data is sent to the server and read out from the GUI in Figure 5.7. Tests with the prototype and the quality of the accelerometer data can be seen in appendix B.2.

This concludes that the prototype can extract all selected motion features and that these are sent to

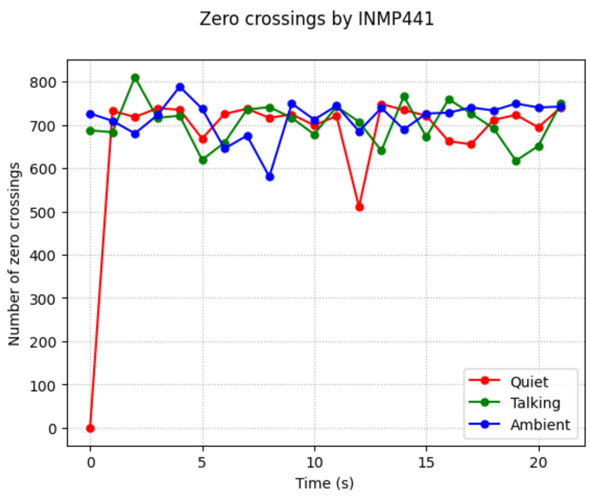


Figure 5.6: Number of zero crossings measured during a) silence, b) close conversation, and c) background conversations

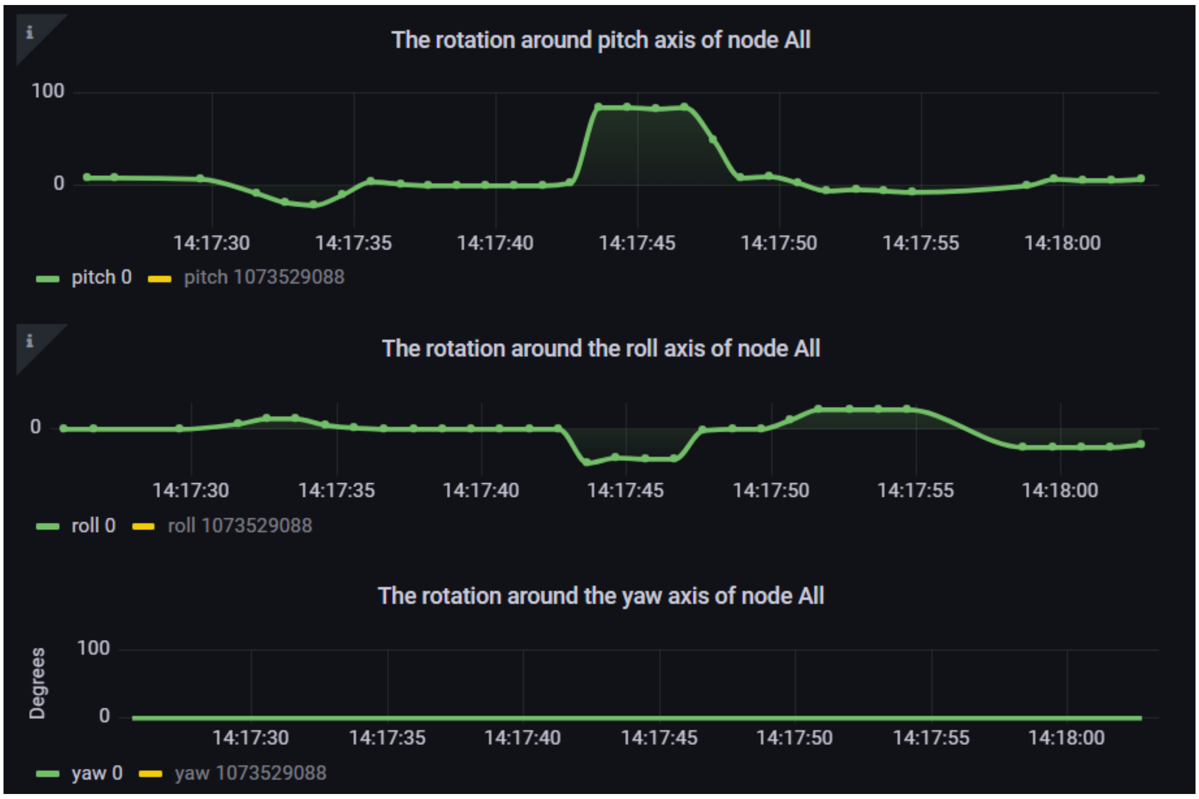

The rotation around pitch axis of node All
100
0
14:17:30
14:17:35
14:17:40
14:17:45
14:17:50
14:17:55
14:18:00
pitch 0
pitch 1073529088
The rotation around the roll axis of node All
0
roll 0
roll 1073529088
The rotation around the yaw axis of node All
Degrees
yaw 0
yaw 1073529088

Figure 5.7: Accelerometer data read out from the GUI

the server and can be read from the GUI; for direct and convenient monitoring by the product user. Requirements MR1, MR2 and MR3 are met successfully (section 2.2.1).

6 Discussion & Future Work

The development of wearable technology with privacy concerns for Multimodal Learning Analytics (MMLA) data collecting in dynamic classroom environments can be advanced by this study. By emphasizing scalability, unobtrusiveness, and GDPR compliance, the system establishes a basis for further study in educational environments. There are, nevertheless, a number of areas that could use improvement and additional research.

Difficulties in Feature Selection
Early in the project, the features that was thought to be useful from intuition and research for MMLA were chosen. However zero crossings, for instance, might not be the most instructive feature in this situation. Additional audio metrics can be added; for example a better defined metric for speech time or spectral energy distribution. These might offer more information. For motion tracking, Pitch-, Roll-, and Yaw angles have been selected. Based on literature [40], some more technical features have been considered: Mean Absolute Value & Root Mean Square, Standard Deviation & Variance, Maximum & Minimum, Simple Squared Integral, Wavelet Entropy, Skewness & Kurtosis, Static- & Dynamic Acceleration Change, and Log Energy Entropy. Extracting these features might support future research by magnifying the significance of the extracted data. Future study could concentrate on finding features that are both technically possible to assess and extremely useful for MMLA.

Limitations of Wearable Design
Although the current prototype works, which was a rough prototype on a breakout board, it could be made even smaller and less noticeable to make it less intrusive. First of all by using a special made PCB. The total volume of the wearable could also be decreased by using a smaller micro controller and more compact sensor integration. It would also save power and space if a specialized Bluetooth® module were used in place of the second ESP32 micro controller. In order to prevent problems like component damage during soldering, which presented difficulties during this project, cautious assembly techniques should also be stressed.

Prospects & Applications for the Future
As a first step, this experiment shows that wearable technology may be used to collect data in educational contexts while maintaining privacy. The system's scalability makes it simple to add more sensors and functionalities; at least five extra. Future iterations could investigate use cases that loosen these restrictions, even if this prototype places a higher priority on privacy and unobtrusiveness. Although there are more privacy concerns, incorporating physiological sensors or permitting full audio recordings may yield richer datasets for research.

In conclusion, this project's effort demonstrates how wearable technology can be used to advance MMLA research. Future versions can fully realize the potential of such systems by resolving existing constraints and investigating novel approaches, allowing for a deeper understanding of educational results and collaborative learning.

7 Conclusion

In the end, the project achieved the goal; designing a wearable with different sensors that can extract multiple features. These features got efficiently data packaged and can be sent wirelessly to a central server where they were stored. The sensors and data extraction worked reliably over long periods of time. By having focused on being unobtrusive and privacy oriented, the wearable should be easy to use for future applications. All 11 mandatory requirements for the sub-product are met. Whereas not all trade-off requirements can be confirmed to be met due to the immeasurable nature of these preferences, all of them have been taken into account while designing. Most of them are met with high enough satisfactory.

The key results from the project are:

- Data collection from different sensors.
- Design of easily expandable wearable.
- Extraction of different features from the sensor data streams.
- Having private oriented extracted features to help with GDPR compliance.
- Efficient packaging of the features which can be sent wirelessly.

Even though the initial requirements are met with satisfactory, this project offers opportunities for future research. Because of the expandability, a lot more can be done. This would include adding more sensors, modalities and features. Lastly, together with the other sub group, this product could have a lot more significance when combining it with machine learning driven analyzing methods. The acquired data could be used to teach machine learning and eventually do real time monitoring. This real time monitoring could help students who are sometimes struggling with working in groups or on projects. So in the end, there is still a lot that can be done. This project offers support with innovating in the field of Multimodal Learning Analytics.

A
Component testing

This appendix will go on about how the components have been verified to be working as intended.

A.1. The micro controller (ESP32-DEV-38P)

The functionalities of the micro controller that will be tested can be read in section 4.2. The first four functionalities can easily be tested by means of a simple program. This program sets one of the pins high, this pin will be connected to another pin. That other pin will be readout if it is high and if it is through the serial monitor a message will be send. The following code was used to verify the workings.

```
#define OUTPUT_PIN 5  // Replace with the pin number you want to set HIGH
#define INPUT_PIN 18  // Replace with the pin number you want to read

void setup() {
  Serial.begin(115200);       // Initialize serial communication
  pinMode(OUTPUT_PIN, OUTPUT); // Set OUTPUT_PIN as an output
  pinMode(INPUT_PIN, INPUT);   // Set INPUT_PIN as an input
  digitalWrite(OUTPUT_PIN, HIGH); // Set OUTPUT_PIN HIGH
}

void loop() {
  if (digitalRead(INPUT_PIN) == HIGH) { // Check if INPUT_PIN is HIGH
    Serial.println("pin is high");
  }
  delay(1000); // 1 sec pause
}
```

This was programmed unto the ESP32. Once the chosen pins were connected, the serial monitor reads out the ESP32. So the first four points are tested to be working successfully. Testing the wireless capabilities is a bit more involved. The testing of the Bluetooth® part can be seen in appendix A.4. The Wi-Fi® functionalities will be tested by the other sub group since they will be using this.

A.2. 6-axis accelerometer (MPU6050)

The 6-axis sensor is capable of measuring acceleration (m/s^2) in all 3 directions (X, Y, Z). Also, the gyroscope allows to do direct readouts from the internal register of the angular rotation ($^\circ/s$) around all three axes. So to test the component, the acceleration and rotation of every axis has to be tested. To do this a MPU6050 was connected to an ESP32 (see Figure A.1). Some code was made to read out all data from the MPU6050. Before the tests can be done, a calibration is recommended, since the MPU6050 does not know what the zero values should be. A calibration is done by doing a lot stationary measurements, averaging it and equaling the output to zero. There is, however, gravity. This means the z-axis should be set to the gravity which is around -9.81. This code which includes calibration can be found in appendix C.2 and will be used for the following tests

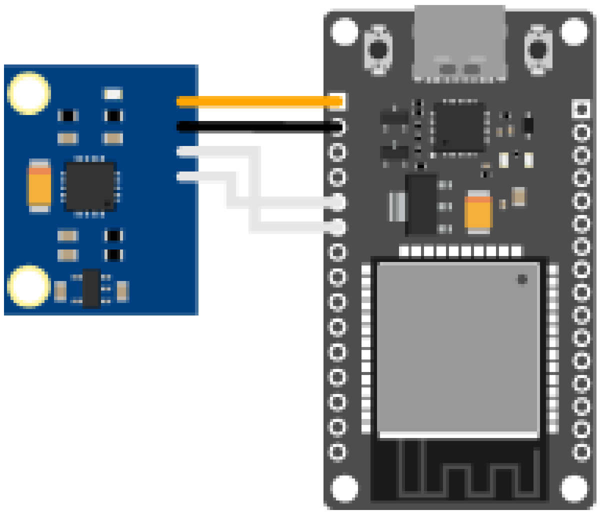


Figure A.1: Testing configuration of the 6-axis sensor

A.2.1. Accelerometer testing

First the functionality of the accelerometer part will be tested. This will be done by moving and accelerating the accelerometer in different directions, and visually verifying the results. The directions that will be tested are each axis.

Acceleration direction	X-acceleration (m/s^2)	Y-acceleration (m/s^2)	Z-acceleration (m/s^2)
stationary	0 ± 0.01	0 ± 0.01	-9.81 ± 0.01
moving x-axis	0 ± 10	0 ± 1	0 ± 1
moving y-axis	0 ± 1	0 ± 10	0 ± 1
moving z-axis	0 ± 1	0 ± 1	-9.81 ± 10

Table A.1: Acceleration testing results

As can be seen from the table A.1, the accelerometer seems to have very slight inaccuracies when in stationary position. This should not give a big problem. The inaccuracies when moving can mostly be explained by that sensor is not moved perfectly in the axis direction and thus the other axis get a slight movement too. The accelerometer seems to be functioning.

A.2.2. Gyroscope testing

The same tests will be done for the gyroscope but instead of moving it in the axis direction. It will instead rotate around the axis. The results of the gyroscope testing are in the following table A.2.

rotation direction	range of angular velocity around x-axis in rad/s	range of angular velocity around y-axis in rad/s	range of angular velocity around z-axis in rad/s
stationary	0	0	0
rotating around x-axis	0 ± 3	0 ± 0.2	0 ± 0.2
rotating around y-axis	0 ± 0.2	0 ± 3	0 ± 0.2
rotating around z-axis	0 ± 0.2	0 ± 0.2	0 ± 3

Table A.2: rotating testing results

Following the results, the gyroscope seems to be functioning well in stationary position. When rotating there seems to be inaccuracies in the other axis but these can easily be explained like the accelerometer that the rotation direction was not perfect. So the gyroscope is functioning as well.

A.3. Microphone testing (INMP441)

To find out if the microphone is working there needs to be an input stream of data, and also this data needs to be actual comprehensible audio. So to test for this, the microphone will be connected to the ESP32 in the following configuration, this configuration will be used for all microphone tests (see Figure A.2). To first test the output stream of the microphone, and if this output stream corresponds to comprehensible audio that should have been recorded, a simple test was made. First, the output stream of the microphone will be output through the serial connection, the ESP32 will be programmed to do this. Second, a simple python script will read the serial connection and make a short '.wav' file that can be listened to on a computer. Both of these programs can be seen in appendix C.1.1. The used sample rate for this test is 8 kHz. This gives a Nyquist frequency of 4kHz. This means that not the whole hearing range of humans, 20-20 kHz, can be recorded. However, a frequency band of 0-4 kHz is normal for recording speech and should be sufficient [41]. The recorded audio from the microphone corresponded to what was recorded. The quality of the recorded audio was similar to using a phone microphone, although a bit worse. This quality is sufficient since no actual raw audio will be recorded; only some features will be extracted. The following sections will test and validate the extraction of features from the microphone.

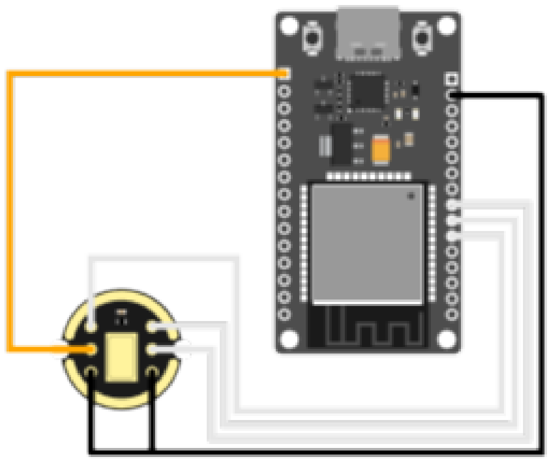
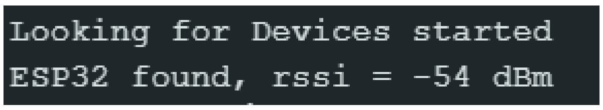


Figure A.2: Testing configuration of the microphone

A.4. Bluetooth® functionality testing (Antenna of the ESP32)

The working of the Bluetooth® antenna has to be tested. To do this, two ESP32's are used. Both ESP32's are both advertising and trying to receive advertised signals. This way, they should see each other. When they receive each others their signal they will output the received signal strength indicator (rssi) of the signal. This can be used for approximate proximity. The code that was used to test this function can be seen in appendix C.3.1. The following image shows the serial monitor after testing



```
Looking for Devices started
ESP32 found, rssi = -54 dBm
```

Figure A.3: Result of Bluetooth® testing

This concludes that the Bluetooth® antenna is working and can be used for the proximity measurement.

B
Prototype testing

This appendix will show some more tests done for the quality of the extracted features from the prototype.

B.1. Microphone

Here the extra tests for the quality of the microphone features will be shown.

B.1.1. Volume

Some volume measurements were conducted to test the quality of the volume feature. Also it was tested if a speaker at certain distances would be able to be picked up. For this first test 3 different noise levels were taken. And a speaker (phone output) was tested at different background noise levels (no (n), medium (m), and high (h)) and distances. The results can be seen in table B.1

Distance (m)	0.3	0.3	.3	1	1	1	3	3	3
background noise level	(n)	(m)	(h)	(n)	(m)	(h)	(n)	(m)	(h)
Measured Volume (dB)									
0% volume (noise)	30	40	43	30	40	43	30	40	43
75% volume	53	55	55	48	50	51	41	43	45
100% volume	64	65	65	56	57	58	50	51	53

Table B.1: Volume tests

From this table B.1 it can be seen that up to three meter a specific person can be measured when talking, however it gets really close and probably will not be consistent. Up to 1 meter however the difference in volume is still significant. Even the difference betweeen 1m and 30 cm is quite high. This can mean if the person wearing the dynamic node or is being talked to can be distinguished. This will be tested upon as well.

Measuring over time

Here three scenarios will be tested and the volume will be graphed over time, with timestamps of what happened. The three scenarios are:

- Lots of moving around, no background noise
- Moving around with background noise
- Lots of noise and moving

These were tested to see if moving, and thus the dynamic node moving, would have significant impact of the measured volume. In the figures with noise some speaking is done as well, both by the person wearing the dynamic node and towards the person wearing the dynamic node. All the tests were done over a period of 50 seconds.

Moving but no noise
In figure B.1 the results for only moving around with the dynamic node. It does show some peaks but these peaks are never even higher then 40 dB which is tested to be generated by background noise already. So moving does not add significant noise.

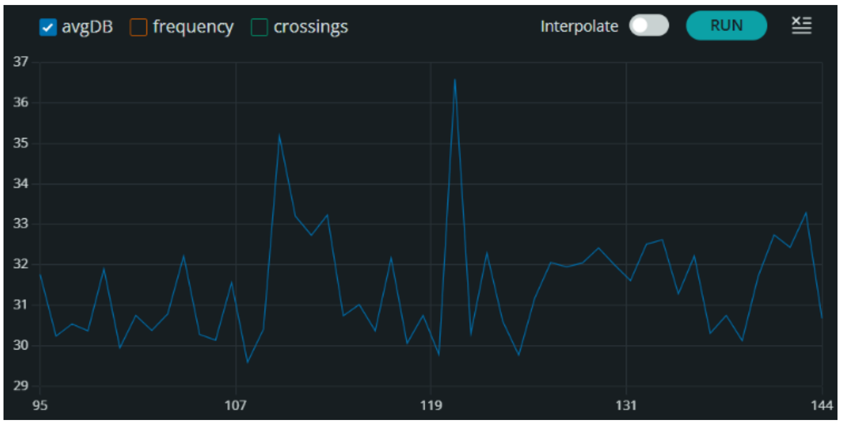


Figure B.1: Moving but no background noise

Moving, some background noise and some talking
In figure B.2 the results can be seen when there is moving and background noise. The peak at the beginning and the second peak just over halfway mark are done by someone talking to the person wearing the dynamic node. The last 2 peaks are made by talking by the person wearing the dynamic node. The difference can be seen in amplitude of dB. And the talking can easily be picked out from the background noise

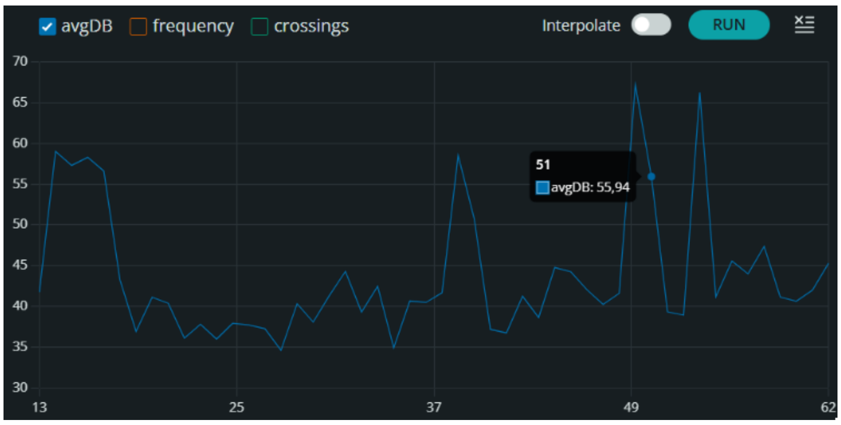


Figure B.2: Moving with some background noise and talking

Moving, some background noise and some talking
In figure B.3 the results can be seen when there is moving and high background noise. In this figure 5 peaks can be seen. the three highest that span multiple seconds are the person wearing the dynamic node talking. the peak around 25 seconds is someone very close by talking Lastly the peak at around 30 seconds mark is someone a bit further talking. It can be concluded that even in high background noise it can be picked out if people close by are talking and especially if the wearer is talking.

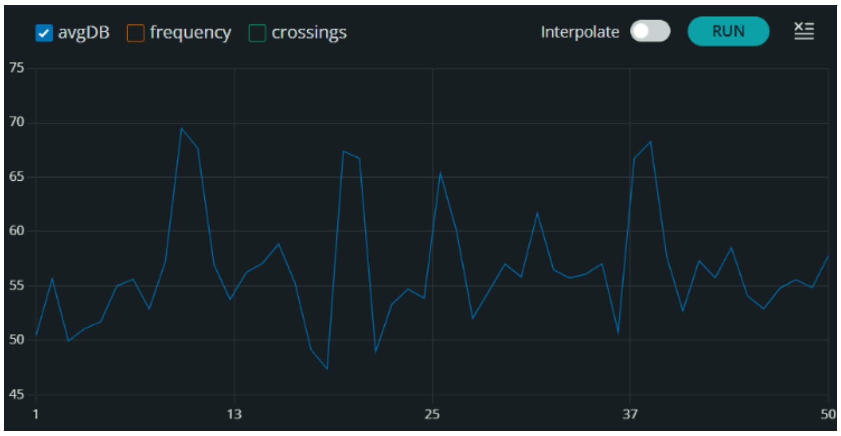


Figure B.3: Moving with high background noise and talking

B.1.2. Frequency

Some measurements were conducted using a frequency tone generator to test the spectral frequency response of the INMP441 microphone, placed at a distance of 10cm from the tone generator that was generally playing with a volume level of 100%. The microphone is programmed to filter and detect only frequencies in the range of speech (20 to 4000 Hz). As can be seen in Table B.2, frequencies outside this range are filtered, leaving only the background noise ($\sim$ 30Hz) to be detected as highest pitch present. The detected frequency has an overall accuracy of at most 4 Hz (see Figure B.5). This accuracy is due to the chosen sampling rate 4.4.1. Generated frequencies between 403 and 410 Hz are detected as 406 Hz (see Figure B.6). In the end, the detected frequencies are divided into bins with a bin size of around 7.8 Hz. Although this means relative less accuracy, it will still guarantee quality and also contribute to a more compact data package.

Frequency response in range of speech

Also, the amplitude response can be seen in Figure B.4. The exact peak SPL numbers are not determined. However, as can be seen in the graph, the response seemingly promises sensitivity reliability over the full frequency range, used for further analysis. As is standing out after testing, higher frequencies have greater amplification rates, which is indicated by the datasheet of the INMP441 [36]. This means that the sensitivity of the microphone is higher as frequency increases. Thus, high pitches in speech will be more strongly present; making the data presumably slightly less accurate for users with a low speaking pitch.

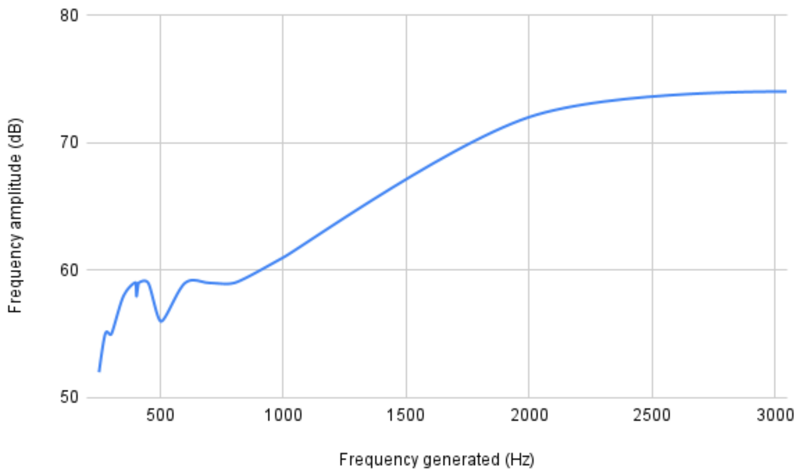


Figure B.4: Measured amplitude (dB) over a generated frequency spectrum

Freq. generated (Hz)	Freq. detected (Hz)	Deviation (Hz)	Amplitude (dB)	Volume (%)
250	250	0	52	100
275	273	2	55	100
300	296	4	55	100
350	351	1	58	100
400	398	2	59	100
402	398	4	58	100
403	406	3	58	100
410	406	4	59	100
411	414	3	59	100
450	453	3	59	100
500	500	0	56	100
600	601	1	59	100
700	703	3	59	100
800	796	4	59	100
1000	1000	0	61	100
2000	2000	0	72	100
3050	3046	4	74	100
4000	4000	0	50	50
5000	23	4977	39	50
5000	31	4969	42	100

Table B.2: Frequency accuracy and amplitude testing with the INMP441

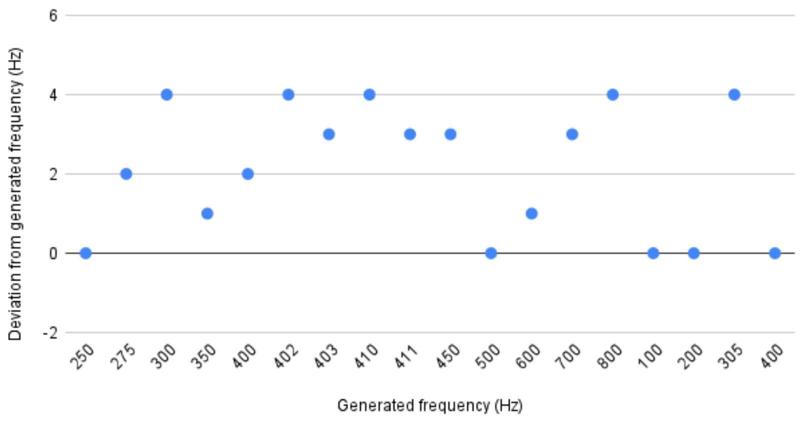


Figure B.5: Accuracy of detected frequency (Hz)

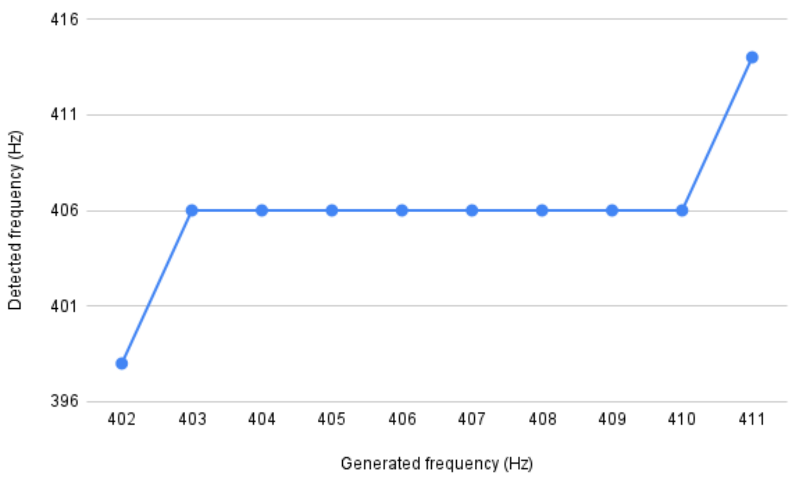


Figure B.6: Bin size of detected frequency (Hz)

B.1.3. Zero Crossings

To test the Zero Crossings feature, 3 test recordings of 50 seconds have been done. These were:

- No speech, seen in figure B.7
- Only speech, seen in figure B.8
- Switching from speech to no speech, seen in figure B.9

These were chosen because this feature is meant to give an idea if sound is speech or not 3.4.2. It does seem speech is way more volatile with the zero crossings, but a human can not really put an end conclusion to this. Machine learning would be needed to actually see if it makes a difference or not.

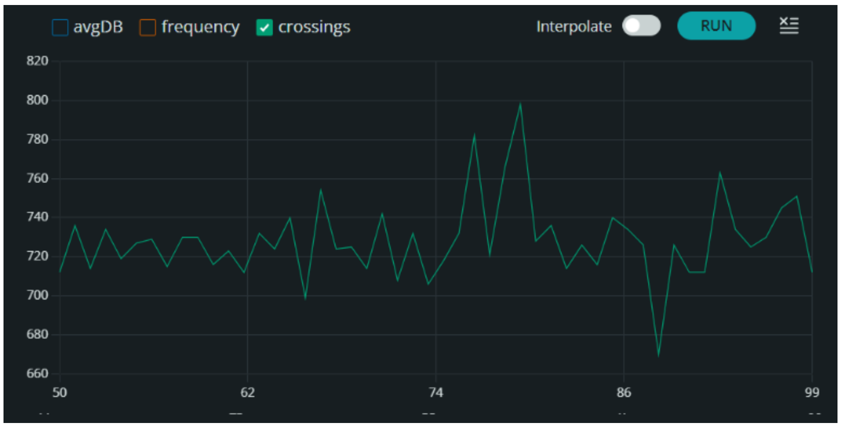


Figure B.7: No speech

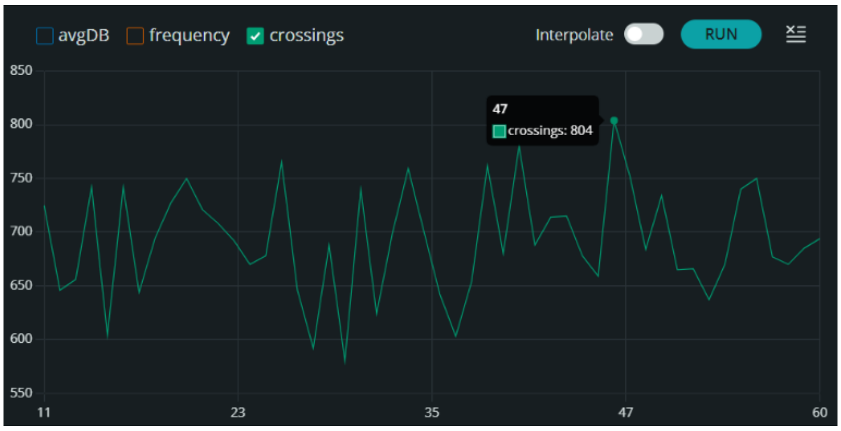


Figure B.8: Only speech

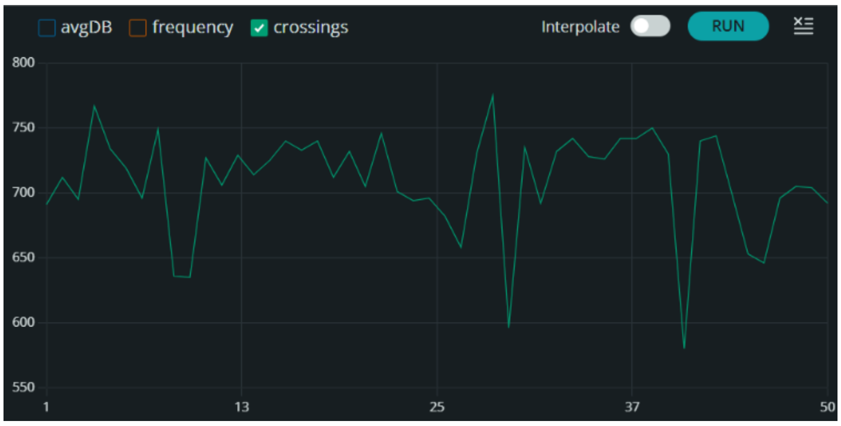


Figure B.9: Mixed. 0-10: speech, 10-20: no speech, 20-30: speech, 30-40: no speech, 40-50: speech

B.2. 6-axis Accelerometer

In this section more on body tests for the 6-axis accelerometer will be shown. First four tests were done to see if different circumstances can be seen and measured by the accelerometer. These tests were done over a period of 50 seconds (but not always starting at 0 seconds). The person wearing the dynamic node will be called subject to keep it short. The subject was also monitored and some stuff was written down to see if it could be seen in the graph. The notes are called observations. The conducted tests were

- Just doing work on computer. Seen in figure B.10. Observations: At 30 seconds mark subject looked left. At 50 seconds mark subject looked right. At 60 seconds mark subject leaned forward.
- Being restless and moving on chair. Seen in figure B.11. Observations. The fluctuations of the yaw are the subject turning in their chair. Readjustments of the subjects sitting position were performed at the 13, 20 and 30 seconds mark.
- While having a conversation with a group. Seen in figure B.12. Observations: The peaks and changes of the yaw correspond to when the subject faced towards a different person. the peaks and dips of roll and pitch are moments the subject adjusts their posture.
- While moving to different places. Seen in figure B.13. Observations: The big changes in the yaw are when the subject is changing moving direction. The subject started with lower speed and acceleration of movement and increased this later. This corresponds to lower fluctuations in the beginning of the graph and the higher fluctuations later in the graph.

The results show that different actions result in very different graphs. This could lead to identifying different actions by machine learning.

For another test of the accelerometer a very long time measurement was done. This was done to check if there would be significant drifting of any feature over a lenghty period of time. The start of the measurement can be seen in figure B.14, and the end of more than 30 minute of active measurements in figure B.15. From these it can be seen that the pitch and roll do not really have any drifting. This is what was expected with how these are calculated from earth's gravity as written in 3.4.1. The yaw however does have a drift from $\pm 5°$ to $\pm 35°$ this is a drift of ± 30 degrees. This drift was accumulated over a total of more than 30 minutes. So a drift of $1°$ per minute. This is not that significant since mostly big changes in orientation are important to measure and small drifts over longer periods do not influence these moments by a significant margin.

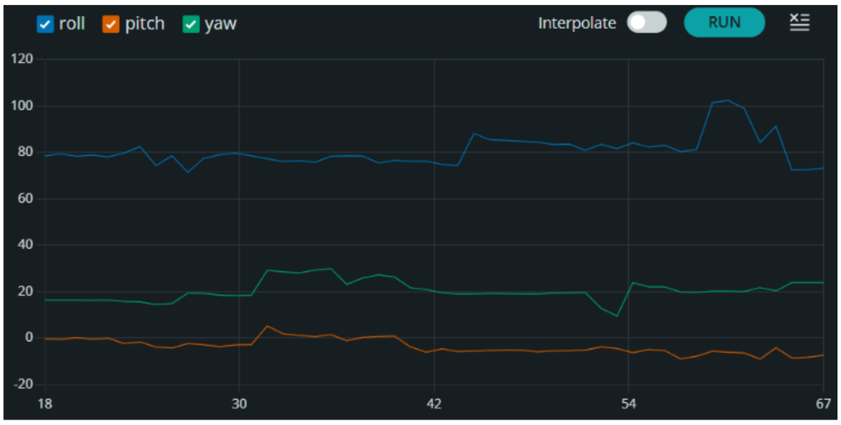


Figure B.10: Stationary, working on computer

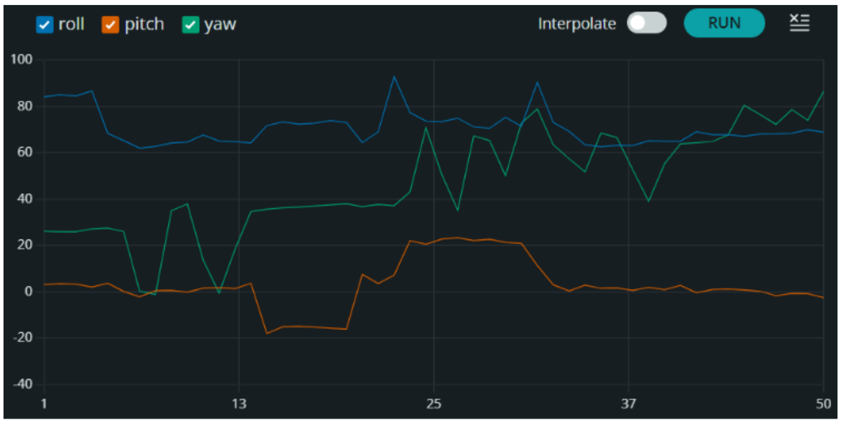


Figure B.11: Lounging and moving around on chair restlessly

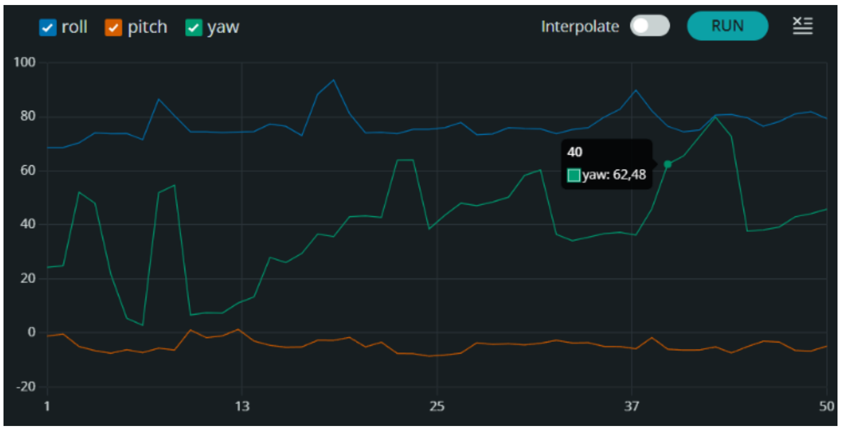


Figure B.12: In conversation with a group of people

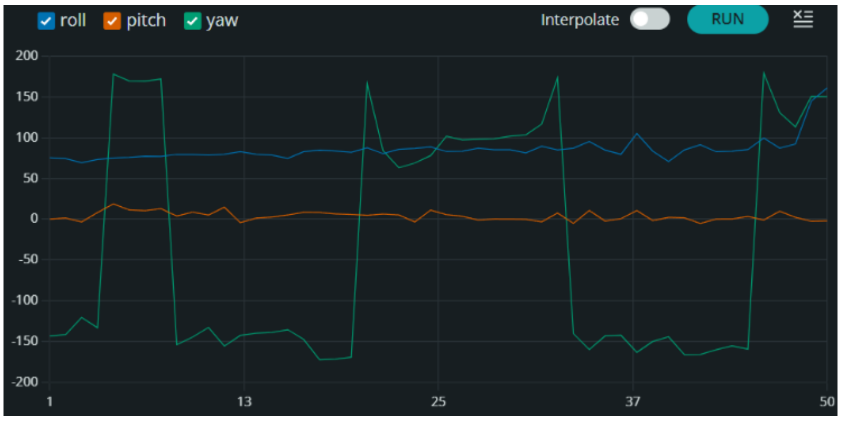


Figure B.13: While on the move

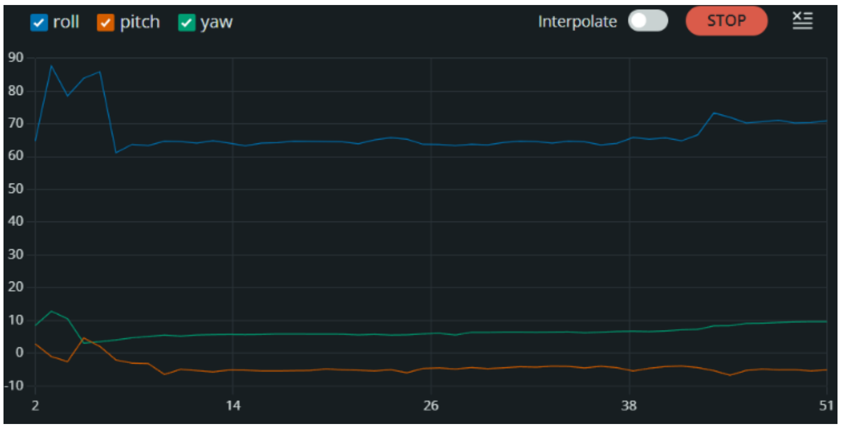


Figure B.14: Start of a long measurement

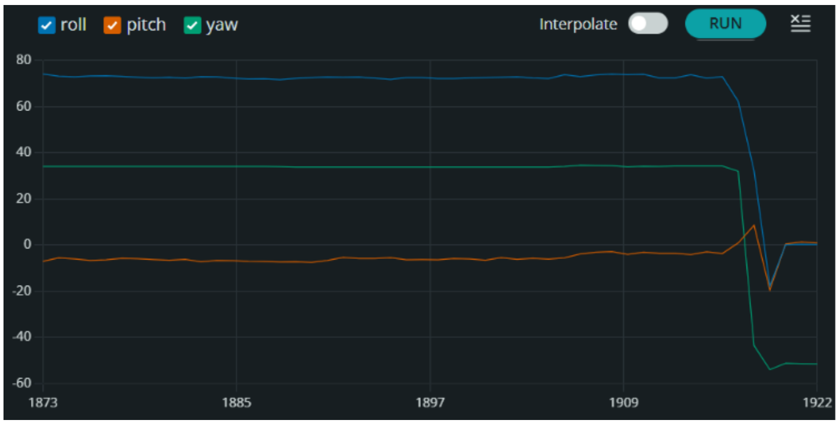


Figure B.15: End of a long measurement

B.3. Proximity

Since there was only an availability of at maximum 6 ESP32's. Testing with a total of 10 nodes was impossible. But to test if the dynamic node was able to see 10 different Bluetooth® devise and sent this information. The filtering of devices was turned off, this mean it would measure all Bluetooth® devices in the vicinity. The results of this can be seen in Figure B.3.

ID	Rssi in -dBm
48	50
1	51
48	79
227	86
48	90
48	95
48	96
48	104
48	105
48	105

Table B.3: rssi of 10 Bluetooth® devices

From this it can be noted that the Bluetooth® sensor works even with the max of 10 Bluetooth® devices. All the 48 ID's and the 227 are because of how truncating the ID works with devices that don't have a name or a different name then static or dynamic. no name gives 48 A random name gives a random ID.

C
Code for feature extracting and tests

This appendix will contain all of the code used for testing the sensors & modalities and for extracting the features.

C.1. Microphone

C.1.1. Microphone functionality test code

The following code will send the microphone data through serial communication

```
#include <Arduino.h>
#include <driver/i2s.h>

// I2S configuration
#define I2S_WS_PIN     22         // Word Select (L/R Select)
#define I2S_SD_PIN     21         // Data In
#define I2S_SCK_PIN    26         // Serial Clock
#define I2S_PORT       I2S_NUM_0  // is unused

// Sampling settings
#define SAMPLE_RATE    16000      // 16 kHz sample rate
#define SAMPLE_BUFFER_SIZE 1024   // Buffer size for each read

// Set up I2S
void setupI2S() {
    i2s_config_t i2s_config = {
        .mode = i2s_mode_t(I2S_MODE_MASTER | I2S_MODE_RX),
        .sample_rate = SAMPLE_RATE,
        .bits_per_sample = I2S_BITS_PER_SAMPLE_16BIT,
        .channel_format = I2S_CHANNEL_FMT_ONLY_LEFT,
        .communication_format = I2S_COMM_FORMAT_I2S_MSB,
        .intr_alloc_flags = ESP_INTR_FLAG_LEVEL1,
        .dma_buf_count = 4,
        .dma_buf_len = SAMPLE_BUFFER_SIZE,
        .use_apll = false,
        .tx_desc_auto_clear = false,
        .fixed_mclk = 0
    };

    // config the I2s With used pins on esp32
    i2s_pin_config_t pin_config = {
        .bck_io_num = I2S_SCK_PIN,
        .ws_io_num = I2S_WS_PIN,
        .data_out_num = I2S_PIN_NO_CHANGE,
        .data_in_num = I2S_SD_PIN
    };

    // Install and start I2S driver
    i2s_driver_install(I2S_PORT, &i2s_config, 0, NULL);
    i2s_set_pin(I2S_PORT, &pin_config);
    i2s_set_clk(I2S_PORT, SAMPLE_RATE, I2S_BITS_PER_SAMPLE_16BIT, I2S_CHANNEL_MONO);
}

void setup() {
    Serial.begin(1000000);  // Initialize serial communication baud rate needs to be high
        enough for the full bit rate of the audio
    setupI2S();            // Initialize I2S microphone
    Serial.println("Streaming␣audio␣data...");  // Optional, for debugging
}

// read the microphone and outpat the data to serial monitor
void loop() {
    int16_t sampleBuffer[SAMPLE_BUFFER_SIZE];
    size_t bytesRead;

    // Read audio data from I2S
    i2s_read(I2S_PORT, sampleBuffer, SAMPLE_BUFFER_SIZE * sizeof(int16_t), &bytesRead,
        portMAX_DELAY);

    // Send raw audio data directly over Serial
    Serial.write((uint8_t*)sampleBuffer, bytesRead);
}
```

The following python code will read the serial communication and record 5 seconds of audio

```
    import serial
import wave
import time

# Configure the serial port with a higher baud rate
ser = serial.Serial('COM3', 1000000)  # Replace 'COM3' with your actual port

# Audio parameters
sample_rate = 16000  # Ensure this matches the sample rate used in the ESP32 code
channels = 1
sample_width = 2  # 16-bit samples (2 bytes)

# File to save audio data
output_filename = 'recorded_audio_5s.wav'

# Duration of recording in seconds
record_duration = 5

print("Recording␣audio␣for␣5␣seconds...")
start_time = time.time()

try:
    with wave.open(output_filename, 'wb') as wav_file:
        wav_file.setnchannels(channels)
        wav_file.setsampwidth(sample_width)
        wav_file.setframerate(sample_rate)

        # Record for the specified duration
        while time.time() - start_time < record_duration:
            if ser.in_waiting > 0:  # Check if data is available to read
                data = ser.read(1024)  # Adjust the size as needed
                wav_file.writeframes(data)

except Exception as e:
    print(f"An␣error␣occurred:␣{e}")
finally:
    ser.close()
    print(f"Audio␣saved␣to␣{output_filename}")
```

C.1.2. Feature testing code for microphone

```
#include <Arduino.h>
#include <driver/i2s.h>
#include <arduinoFFT.h>

```

```
// Pins for I2S microphone
#define I2S_NUM I2S_NUM_0
#define I2S_WS 25  // Word select (L/R clock) pin
#define I2S_SD 33  // Data input pin
#define I2S_SCK 26 // Bit clock pin

// FFT parameters
const uint16_t samples = 2048; // Number of samples for FFT (must be a power of 2)
double vReal[samples];
double vImag[samples];
const double SAMPLE_RATE = 16000.0;

//creating function to do FFT
ArduinoFFT FFT = ArduinoFFT(vReal, vImag, samples, SAMPLE_RATE);

// Variables for storing results
double averageDb = 0.0;
double peakFrequency = 0.0;  // To store peak frequency
int zeroCrossings = 0; // To store the count of zero-crossings
bool dataReady = false; // Flag to signal new data is available

// Struct to hold microphone data
struct MicrophoneData {
    uint16_t avgDb;
    uint16_t peakFrequency;
    uint16_t zeroCrossingsCount;
};

// Mutex to protect shared data
portMUX_TYPE dataMutex = portMUX_INITIALIZER_UNLOCKED;

// Task handle
TaskHandle_t microphoneTaskHandle;

//setup microphone
void setupMicrophone() {
    // Configure I2S
    i2s_config_t i2s_config = {
        .mode = (i2s_mode_t)(I2S_MODE_MASTER | I2S_MODE_RX),
        .sample_rate = SAMPLE_RATE,
        .bits_per_sample = I2S_BITS_PER_SAMPLE_16BIT,
        .channel_format = I2S_CHANNEL_FMT_ONLY_LEFT,
        .communication_format = I2S_COMM_FORMAT_I2S_MSB,
        .intr_alloc_flags = ESP_INTR_FLAG_LEVEL1,
        .dma_buf_count = 8,
        .dma_buf_len = 1024,
        .use_apll = false,
        .tx_desc_auto_clear = false,
        .fixed_mclk = 0
    };

    i2s_pin_config_t pin_config = {
        .bck_io_num = I2S_SCK,
        .ws_io_num = I2S_WS,
        .data_out_num = I2S_PIN_NO_CHANGE,
        .data_in_num = I2S_SD
    };

    i2s_driver_install(I2S_NUM, &i2s_config, 0, NULL);
    i2s_set_pin(I2S_NUM, &pin_config);
    i2s_start(I2S_NUM);

    // Start microphone processing task
    xTaskCreatePinnedToCore(
        microphoneTask,      // Task function
        "Microphone␣Task",   // Task name
        8192,                // Stack size
        NULL,                // Parameter
        1,                   // Priority
        &microphoneTaskHandle, // Task handle
        0                    // Core
    );
}

// Microphone processing task
void microphoneTask(void *param) {
    while (true) {
        int16_t sampleBuffer[samples];
        size_t bytesRead;

        // Read data from I2S
        i2s_read(I2S_NUM, sampleBuffer, samples * sizeof(int16_t), &bytesRead, portMAX_DELAY)
            ;

        // Prepare data for FFT
        for (uint16_t i = 0; i < samples; i++) {
            vReal[i] = (double)sampleBuffer[i];  // Copy real part
            vImag[i] = 0.0;  // Imaginary part set to 0
        }

        // Perform FFT
        FFT.windowing(FFT_WIN_TYP_HAMMING, FFT_FORWARD);
        FFT.compute(FFT_FORWARD);
        FFT.complexToMagnitude();

        // Find the peak frequency in the range of interest (20Hz to 4000Hz)
        double peakFrequencyLocal = 0.0;
        double maxMagnitude = 0.0;

        for (uint16_t i = 1; i < (samples / 2); i++) {  // Only positive frequencies
            double frequency = (i * SAMPLE_RATE) / samples;
            double magnitude = vReal[i];

            // Filter: Only consider frequencies between 20 Hz and 4000 Hz
            if (frequency >= 20 && frequency <= 4000) {
                if (magnitude > maxMagnitude) {
                    maxMagnitude = magnitude;
                    peakFrequencyLocal = frequency;
                }
            }
        }

        // Zero-crossing count
        zeroCrossings = 0;
        for (uint16_t i = 1; i < samples; i++) {
            if ((vReal[i - 1] > 0 && vReal[i] < 0) || (vReal[i - 1] < 0 && vReal[i] > 0)) {
                zeroCrossings++;
            }
        }

        // Calculate average dB
        double totalVolume = 0.0;
        int volumeCount = 0;
        for (uint16_t i = 1; i < (samples / 2); i++) {
            totalVolume += vReal[i];
            volumeCount++;
        }
        double averageLinearVolume = totalVolume / volumeCount;
        double localAverageDb = 20.0 * log10(averageLinearVolume);

        // Update shared data with mutex
        portENTER_CRITICAL(&dataMutex);
        averageDb = localAverageDb;
        peakFrequency = peakFrequencyLocal;  // Store the peak frequency
        dataReady = true; // Signal new data is ready
        portEXIT_CRITICAL(&dataMutex);

        vTaskDelay(200 / portTICK_PERIOD_MS);  // Process every second
    }
}

// Function to get microphone data
```

```
bool getMicrophoneData(double &avgDb, double &peakestFrequency, int &zeroCrossingsCount) {
    bool ready = false;

    // Access shared data with mutex
    portENTER_CRITICAL(&dataMutex);
    if (dataReady) {
        avgDb = averageDb;

        // The peak frequency should already be calculated in your microphone task.
        peakestFrequency = peakFrequency; // `peakFrequency` holds the peak frequency

        zeroCrossingsCount = zeroCrossings; // Get the zero-crossing count

        dataReady = false; // Reset flag
        ready = true;
    }
    portEXIT_CRITICAL(&dataMutex);

    return ready;
}

//start up of the esp and microphone
void setup() {
    Serial.begin(115200);

    // Initialize microphone
    setupMicrophone();

}

void loop() {
    static unsigned long lastRunTime = 0; // Store the last time the loop ran

    //run only when new microphonedata available
    if (millis() - lastRunTime >= 400) {
        lastRunTime = millis(); // Update the last run time

        // Fetch microphone data
        MicrophoneData microphonedata;

        //initialise features
        double avgDb;
        double peakestFrequency;
        int zeroCrossingsCount;

        if (getMicrophoneData(avgDb, peakestFrequency, zeroCrossingsCount)) {

            //put microphone data in microphone struct
            microphonedata.avgDb = static_cast<uint16_t>(avgDb);
            microphonedata.peakFrequency = static_cast<uint16_t>(peakestFrequency);
            microphonedata.zeroCrossingsCount = static_cast<uint16_t>(zeroCrossingsCount);

        } else {
            Serial.println("Microphone data not ready.");
        }
        // Print MicrophoneData
        Serial.print(microphonedata.avgDb); Serial.print(", ");
        Serial.print(microphonedata.peakFrequency); Serial.print(", ");
        Serial.println(microphonedata.zeroCrossingsCount);

    }

}
```

C.2. 6-axis accelerometer

```
#include <Wire.h>
#include <Adafruit_MPU6050.h>
#include <Adafruit_Sensor.h>

```

```
Adafruit_MPU6050 mpu;

// Calibration offsets

// Variables to store the accelerometer offsets
float accelOffsetX = 0.0;
float accelOffsetY = 0.0;
float accelOffsetZ = 0.0;

// Variables to store the gyroscope offsets
float gyroOffsetX = 0.0;
float gyroOffsetY = 0.0;
float gyroOffsetZ = 0.0;

// Constants for gravitational acceleration
const float GRAVITY = 9.81; // m/s^2

// measuring difference in time
int timestamp = 0;
int diff = 0;
int counter = 0;

// roll pitch yaw
float roll = 0, pitch = 0, yaw = 0;

//calculating offsets to calibrate sensor
void calibrateAccelerometer() {
  float gyroX = 0, gyroY = 0, gyroZ = 0;
  float accelX = 0, accelY = 0, accelZ = 0;
  int sampleCount = 200; // Number of samples for averaging

  Serial.println("Calibrating␣sensor,␣keep␣the␣sensor␣still...");

  // Collect sampleCount readings of sensor data
  for (int i = 0; i < sampleCount; i++) {
    sensors_event_t a, g, temp;
    mpu.getEvent(&a, &g, &temp);

    // Sum up accelerometer readings
    accelX += a.acceleration.x;
    accelY += a.acceleration.y;
    accelZ += a.acceleration.z;

    // Sum up gyroscope readings
    gyroX += g.gyro.x;
    gyroY += g.gyro.y;
    gyroZ += g.gyro.z;

    delay(50); // Small delay between readings
  }

  // Calculate the average offset for accelerometer and gyroscope
  accelOffsetX = accelX / sampleCount;
  accelOffsetY = accelY / sampleCount;
  accelOffsetZ = (accelZ / sampleCount) - GRAVITY;

  // Gyroscope calibration is already covered in the previous example
  gyroOffsetX = gyroX / sampleCount;
  gyroOffsetY = gyroY / sampleCount;
  gyroOffsetZ = gyroZ / sampleCount;

    Serial.println("Calibration␣complete!");
}

//setup the accelerometer
void setup() {
  Serial.begin(115200);
  while (!Serial) {
    delay(10); // Wait for Serial to initialize
  }

  // Initialize I2C communication
  if (!mpu.begin()) {
    Serial.println("Failed␣to␣find␣MPU6050␣chip");
    while (1) {
      delay(10);
    }
  }

  Serial.println("MPU6050␣Found!");

  // Set accelerometer and gyroscope ranges
    mpu.setAccelerometerRange(MPU6050_RANGE_8_G);
    mpu.setGyroRange(MPU6050_RANGE_500_DEG);
    mpu.setFilterBandwidth(MPU6050_BAND_21_HZ);

  delay(100);

  calibrateAccelerometer();

  delay(100);

  timestamp = millis();


}



void loop() {
  // Read accelerometer and gyroscope data
  sensors_event_t a, g, temp;
  mpu.getEvent(&a, &g, &temp);

  // Calculate roll and pitch
  roll = atan2((a.acceleration.y - accelOffsetY), (a.acceleration.z - accelOffsetZ)) * 180 /
      PI;
  pitch = atan2(-(a.acceleration.x - accelOffsetX), sqrt((a.acceleration.y - accelOffsetY) *
      (a.acceleration.y - accelOffsetY) + (a.acceleration.z - accelOffsetZ) * (a.acceleration
      .z - accelOffsetZ))) * 180 / PI;

  diff = millis() - timestamp;
  counter = counter + diff;
  timestamp = millis();
  if(abs(diff*(g.gyro.z - gyroOffsetZ)) > 0.4){

    yaw = yaw + diff*(g.gyro.z - gyroOffsetZ)/1000 * 180 / PI;

    //keep yaw in range of -180 to 180
    if(yaw > 180){
      yaw = yaw - 360;
    }
    if(yaw < -180){
      yaw = yaw + 360;
    }
    }

  //print roll pitch yaw (counter is used to not print all the time)
  if(counter > 500){
  Serial.print("roll:␣");
  Serial.print(roll);
  Serial.print("pitch:␣");
  Serial.print(pitch);
  Serial.print("␣yaw:␣");
  Serial.print(yaw);
  Serial.println("␣degrees");

  //reset counter
  counter = 0;

    // // Print accelerometer values
  Serial.print("Accelerometer␣X:␣");
  Serial.print(a.acceleration.x - accelOffsetX);
  Serial.print("␣m/s^2,␣Y:␣");
  Serial.print(a.acceleration.y - accelOffsetY);
  Serial.print("␣m/s^2,␣Z:␣");
  Serial.print(a.acceleration.z - accelOffsetZ);
  Serial.println("␣m/s^2");

  // Print gyroscope values
  Serial.print("Gyroscope␣X:␣");
  Serial.print(g.gyro.x - gyroOffsetX);
  Serial.print("␣rad/s,␣Y:␣");
  Serial.print(g.gyro.y - gyroOffsetY);
  Serial.print("␣rad/s,␣Z:␣");
  Serial.print(g.gyro.z - gyroOffsetZ);
  Serial.println("␣rad/s");
  }

  delay(20); // Adjust delay as needed
}
```

C.3. Bluetooth® antenna

C.3.1. Test for functionality of Bluetooth® antenna

```
    #include <NimBLEDevice.h>
#include <Arduino.h>

BLEScan *pBLEScan;

String targetName = "ESP32_Device1";    // Name to search for

//make callback so it finds rssi and name when signal is received
class MyAdvertisedDeviceCallbacks : public BLEAdvertisedDeviceCallbacks {
  void onResult(BLEAdvertisedDevice *advertisedDevice) {
    int rssi = advertisedDevice->getRSSI();
    String deviceName = advertisedDevice->haveName() ? advertisedDevice->getName().c_str() :
        "";
    if (targetName == deviceName){
      Serial.print("ESP32␣found,␣rssi␣=␣");
      Serial.print(advertisedDevice->getRSSI());
      Serial.println("␣dBm");
    }

  }
};

//continuously scan for devices
void bleScanTask(void *pv) {
  pBLEScan = BLEDevice::getScan();
  pBLEScan->setAdvertisedDeviceCallbacks(new MyAdvertisedDeviceCallbacks());
  pBLEScan->setActiveScan(true); // Active scan for more detailed results

  while (true) {
    pBLEScan->start(1);

    pBLEScan->clearResults(); // Free up memory
  }
}

// Initialize BLE and start the scanner task
void init_ble() {
  const char *customName = "ESP32_Device2"; // Replace with your desired device name
  BLEDevice::init(customName); // Initialize the BLE device with the custom name

  // Start advertising with the set name
  BLEAdvertising *pAdvertising = BLEDevice::getAdvertising();
  pAdvertising->start();

  xTaskCreate(
      bleScanTask,    // Function to run
      "bleScanTask",  // Name of the task
      3000,           // Stack size
      NULL,           // Task input parameter
      1,              // Priority
      NULL            // Task handle
  );
}

void setup() {
    Serial.begin(115200);
    // Initialize BLE functionality (and start proximity sensing)
    Serial.println("Looking␣for␣Devices␣started");
    init_ble();
}

void loop(){

}
```

C.3.2. BLE® feature test

```
    #include <Wire.h>
#include "NimBLEDevice.h"

#define I2C_SLAVE_ADDR 0x08  // Address of the slave ESP32

BLEScan *pBLEScan;

// Structure to hold device name and RSSI
struct DeviceRSSI {
  uint8_t deviceName;  // Using int to store the last 3 digits of the device name (as before)
  uint8_t rssi;
};

// Array to store the top 10 strongest RSSI values
DeviceRSSI topDevices[10]; // To store the 10 strongest devices

// Helper function to check if a string starts with a prefix
bool startsWith(const String &str, const String &prefix) {
  return str.indexOf(prefix) == 0; // Returns true if `prefix` is found at the start of `str`
}

class MyAdvertisedDeviceCallbacks : public BLEAdvertisedDeviceCallbacks {
  void onResult(BLEAdvertisedDevice *advertisedDevice) {
    int rssi = advertisedDevice->getRSSI();
    String deviceName = advertisedDevice->haveName() ? advertisedDevice->getName().c_str() :
        "";

    if (deviceName.startsWith("Dynamic") || deviceName.startsWith("Static")) {
      int length = deviceName.length();
      // Get the last 3 characters of the name (the digits)
      int lastThreeDigits = (deviceName[length - 3] - '0') * 100 + (deviceName[length - 2] -
          '0') * 10 + (deviceName[length - 1] - '0');

      // Insert the device into the list based on RSSI
      for (int i = 0; i < 10; i++) {
        if (abs(rssi) < topDevices[i].rssi) {
          // Shift down the other entries to make room for the new device
          for (int j = 9; j > i; j--) {
            topDevices[j] = topDevices[j - 1];
          }
          // Insert the new device
          topDevices[i].rssi = static_cast<uint8_t>(abs(rssi));
          topDevices[i].deviceName = static_cast<uint8_t>(lastThreeDigits);
          break;
        }
      }
    }
  }
};
```

```

void bleScanTask(void *pv) {
  pBLEScan = BLEDevice::getScan();
  pBLEScan->setAdvertisedDeviceCallbacks(new MyAdvertisedDeviceCallbacks());
  pBLEScan->setActiveScan(true); // Active scan for more detailed results

  while (true) {
    // Start scanning for 2 seconds
    pBLEScan->start(1);

    Serial.println("--------------------------------");

    for (int i = 0; i < 10; i++){
    Serial.print("|");
    Serial.print(topDevices[i].deviceName);
    Serial.print("|");
    Serial.print(topDevices[i].rssi);
    Serial.println("|");
    Serial.println("------------");
    }

    Serial.println("--------------------------------");

    // Reset the list of top 10 devices
    for (int i = 0; i < 10; i++) {
      topDevices[i].rssi = 255; // Initialize RSSI to a very low value
      topDevices[i].deviceName = 0; // Clear the device name
    }

    pBLEScan->clearResults(); // Free up memory
    //delay(1000); // Delay between scans
  }
}

void init_ble() {
  const char *customName = "Dynamic_Node_019"; // Replace with your desired device name
  BLEDevice::init(customName);                  // Initialize the BLE device with the custom
      name

  // Start advertising with the set name
  BLEAdvertising *pAdvertising = BLEDevice::getAdvertising();
  pAdvertising->start();

  xTaskCreate(
      bleScanTask,   // Function to run
      "bleScanTask", // Name of the task
      3000,          // Stack size
      NULL,          // Task input parameter
      1,             // Priority
      NULL           // Task handle
  );
}

void setup() {
  // Initialize serial for output
  Serial.begin(115200);
  // Initialize BLE
  init_ble();
}

void loop() {
  // Main loop can be left empty as BLE scanning and I2C handling are done in tasks
}
```

C.4. Final data package

The following code combines the three previous feature extraction codes. Small adjustments must be made to those for this to work. their setup and main loop must be deleted as this code will do that for them. all 4 codes should be included in the same map.

```
    #include <Arduino.h>

// Struct to hold microphone features
struct MicrophoneData {
    uint16_t avgDb;
    uint16_t peakFrequency;
    uint16_t zeroCrossingsCount;
};

// Struct to hold accelerometer features
struct AccelerometerData {
    float roll;
    float pitch;
    float yaw;
};

// Struct to hold BLE features
struct BLEData {
    uint8_t deviceNames[10];
    uint8_t rssiValues[10];
};

// Struct to hold all output features
struct OutputData {
    MicrophoneData microphoneData;
    AccelerometerData accelerometerData;
    BLEData bleData;
};

// starting everything
void setup() {
    Serial.begin(115200);

    // Initialize accelerometer and microphone (and start microphone)
    setupAccelerometer();
    setupMicrophone();

    // Calibrate accelerometer
    calibrateAccelerometer();

    // Start the accelerometer task
    startAccelerometerTask();  // Start the accelerometer task from accelerometer.ino

    // Initialize BLE functionality (and start proximity sensing)
    init_ble();
}

void loop() {
    static unsigned long lastRunTime = 0; // Store the last time the loop ran

    // Check if 1000ms (1 second) has passed since the last execution
    if (millis() - lastRunTime >= 1000) {
        lastRunTime = millis(); // Update the last run time

        // Fetch microphone data (including zero-crossings)
        MicrophoneData microphonedata;

        double avgDb;
        double peakestFrequency;
        int zeroCrossingsCount;
        if (getMicrophoneData(avgDb, peakestFrequency, zeroCrossingsCount)) {

            //put microphone data in microphone struct
            microphonedata.avgDb = static_cast<uint16_t>(avgDb);
            microphonedata.peakFrequency = static_cast<uint16_t>(peakestFrequency);
            microphonedata.zeroCrossingsCount = static_cast<uint16_t>(zeroCrossingsCount);

        } else {
            Serial.println("Microphone␣data␣not␣ready.");
        }

        // Fetch accelerometer data
        float roll, pitch, yaw;
        getAccelerometerData(roll, pitch, yaw);

        // put accelerometer data in acceleerometer struct
        AccelerometerData accelerometerdata;
        accelerometerdata.roll    = roll;
        accelerometerdata.pitch   = pitch;
        accelerometerdata.yaw     = yaw;

        // Print timestamp
        Serial.println("-----------------------");
        Serial.printf("Timestamp:␣%lu␣ms\n", millis());
        // Serial.println("-----------------------");

        // Create arrays to store device names and RSSI values
        String deviceNames[10];
        int rssiValues[10];

        // Get the top 10 devices and their RSSI values
        getTopDevices(deviceNames, rssiValues);

        BLEData bledata;

        // encode and put ble data in blestruct
        for(int i = 0; i < 10; i++){
          String name = deviceNames[i];
          int length = name.length();

          if(name.startsWith("Dynamic") || name.startsWith("Static")){
          // Get the last 3 characters of the name (the digits)
          int lastThreeDigits = (name[length - 3] - '0') * 100 + (name[length - 2] - '0') * 10 + (name[length - 1] - '0');

          bledata.deviceNames[i] = static_cast<uint8_t>(lastThreeDigits);
          bledata.rssiValues[i] = static_cast<uint8_t>(abs(rssiValues[i]));
          }
          else{
          bledata.deviceNames[i] = static_cast<uint8_t>(0);
          bledata.rssiValues[i] = static_cast<uint8_t>(0);
          }

        }

        //make final output data struct
        OutputData data;
        data.microphoneData = microphonedata;
        data.accelerometerData = accelerometerdata;
        data.bleData = bledata;

        // Print MicrophoneData
        Serial.print(data.microphoneData.avgDb); Serial.print(",␣");
        Serial.print(data.microphoneData.peakFrequency); Serial.print(",␣");
        Serial.println(data.microphoneData.zeroCrossingsCount);

        // Print AccelerometerData
        Serial.print(data.accelerometerData.roll, 2); Serial.print(",␣");
        Serial.print(data.accelerometerData.pitch, 2); Serial.print(",␣");
        Serial.println(data.accelerometerData.yaw, 2);

        // Print BLEData
        Serial.print("[");
        for (int i = 0; i < 10; ++i) {
          Serial.print(data.bleData.deviceNames[i]);
        if (i < 9) Serial.print(",␣");
        }
        Serial.print("],␣[");
        for (int i = 0; i < 10; ++i) {
            Serial.print(data.bleData.rssiValues[i]);
        if (i < 9) Serial.print(",␣");
        }
```

```
        Serial.println("]");
        Serial.println("------------------------");

    }

}
```

Bibliography

[1] [Online]. Available: https://microelectronics.tudelft.nl/eee/tellegen_hall/.

[2] M. Giannakos, D. Spikol, D. Di Mitri, K. Sharma, X. Ochoa, and R. Hammad, "Introduction to multimodal learning analytics," in *The Multimodal Learning Analytics Handbook*, M. Giannakos, D. Spikol, D. Di Mitri, K. Sharma, X. Ochoa, and R. Hammad, Eds. Cham: Springer International Publishing, 2022, pp. 3–28, ISBN: 978-3-031-08076-0. DOI: 10.1007/978-3-031-08076-0_1. [Online]. Available: https://doi.org/10.1007/978-3-031-08076-0_1.

[3] H. Ouhaichi, D. Spikol, and B. Vogel, "Rethinking mmla: Design considerations for multimodal learning analytics systems," in *Proceedings of the Tenth ACM Conference on Learning @ Scale*, ser. L@S '23, Copenhagen, Denmark: Association for Computing Machinery, 2023, pp. 354–359, ISBN: 9798400700255. DOI: 10.1145/3573051.3596186. [Online]. Available: https://doi.org/10.1145/3573051.3596186.

[4] X. Ochoa and F. Dominguez, "Controlled evaluation of a multimodal system to improve oral presentation skills in a real learning setting," *British Journal of Educational Technology*, vol. 51, no. 5, pp. 1615–1630, 2020, ISSN: 0007-1013. DOI: 10.1111/bjet.12987. [Online]. Available: https://berajournals.onlinelibrary.wiley.com/doi/10.1111/bjet.12987.

[5] M. Vujovic, D. Hernández-Leo, S. Tassani, and D. Spikol, "Round or rectangular tables for collaborative problem solving? a multimodal learning analytics study," *British Journal of Educational Technology*, vol. 51, no. 5, pp. 1597–1614, 2020, © 2020 British Educational Research Association. DOI: 10.1111/bjet.12988.

[6] X. Ochoa, "Multimodal learning analytics – rationale, process, examples, and direction," in *Handbook of Learning Analytics*, C. Lang, G. Siemens, A. F. Wise, D. Gašević, and A. Merceron, Eds., 2nd, Vancouver, BC: SoLAR, 2022, pp. 54–65. DOI: 10.18608/hla22.

[7] K. Sharma and M. Giannakos, "Multimodal data capabilities for learning: What can multimodal data tell us about learning?" *British Journal of Educational Technology*, vol. 51, no. 5, pp. 1450–1484, 2020. DOI: 10.1111/bjet.13013.

[8] M. Worsley and P. Blikstein, "A multimodal analysis of making," *International Journal of Artificial Intelligence in Education*, vol. 28, pp. 385–419, 2018. DOI: 10.1007/s40593-017-0160-1. [Online]. Available: https://doi.org/10.1007/s40593-017-0160-1.

[9] R. Hammad, M. Bahja, and M. A. Kuhail, "Bridging the gap between informal learning pedagogy and multimodal learning analytics," in *The Multimodal Learning Analytics Handbook*, M. Giannakos, D. Spikol, D. Di Mitri, K. Sharma, X. Ochoa, and R. Hammad, Eds. Cham: Springer International Publishing, 2022, ISBN: 978-3-031-08076-0. DOI: 10.1007/978-3-031-08076-0_7. [Online]. Available: https://doi.org/10.1007/978-3-031-08076-0_7.

[10] P. K. Tiffany C.K. Kwok and M. Raubal, "Unobtrusive interaction: A systematic literature review and expert survey," *Human–Computer Interaction*, vol. 39, 2024. DOI: 10.1080/07370024.2022.2162404. eprint: https://doi.org/10.1080/07370024.2022.2162404. [Online]. Available: https://doi.org/10.1080/07370024.2022.2162404.

[11] H. Alwahaby and M. Cukurova, "Chapter 2 - navigating the ethical landscape of multimodal learning analytics: A guiding framework," in *Ethics in Online AI-based Systems*, ser. Intelligent Data-Centric Systems, S. Caballé, J. Casas-Roma, and J. Conesa, Eds., Academic Press, 2024, ISBN: 978-0-443-18851-0. DOI: https://doi.org/10.1016/B978-0-443-18851-0.00014-7. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780443188510000147.

[12] H. Alwahaby, M. Cukurova, Z. Papamitsiou, and M. Giannakos, “The evidence of impact and ethical considerations of multimodal learning analytics: A systematic literature review,” in *The Multimodal Learning Analytics Handbook*, M. Giannakos, D. Spikol, D. Di Mitri, K. Sharma, X. Ochoa, and R. Hammad, Eds. Cham: Springer International Publishing, 2022, ISBN: 978-3-031-08076-0. DOI: `10.1007/978-3-031-08076-0_12`. [Online]. Available: `https://doi.org/10.1007/978-3-031-08076-0_12`.

[13] M. Cukurova, M. Giannakos, and R. Martinez-Maldonado, “The promise and challenges of multimodal learning analytics,” *British Journal of Educational Technology*, vol. 51, no. 5, 2020. DOI: `https://doi.org/10.1111/bjet.13015`. eprint: `https://bera-journals.onlinelibrary.wiley.com/doi/pdf/10.1111/bjet.13015`. [Online]. Available: `https://bera-journals.onlinelibrary.wiley.com/doi/abs/10.1111/bjet.13015`.

[14] P. Chejara, R. Kasepalu, L. Prieto, M. J. Rodríguez-Triana, and A. Ruiz-Calleja, “Bringing collaborative analytics using multimodal data to masses: Evaluation and design guidelines for developing a mmla system for research and teaching practices in cscl,” in *Proceedings of the 14th Learning Analytics and Knowledge Conference*, ser. LAK ’24, Kyoto, Japan: Association for Computing Machinery, 2024, ISBN: 9798400716188. DOI: `10.1145/3636555.3636877`. [Online]. Available: `https://doi.org/10.1145/3636555.3636877`.

[15] J. Hassan, J. Leong, and B. Schneider, “Multimodal data collection made easy: The ez-mmla toolkit: A data collection website that provides educators and researchers with easy access to multimodal data streams.,” in *LAK21: 11th International Learning Analytics and Knowledge Conference*, ser. LAK21, Irvine, CA, USA: Association for Computing Machinery, 2021, ISBN: 9781450389358. DOI: `10.1145/3448139.3448201`. [Online]. Available: `https://doi.org/10.1145/3448139.3448201`.

[16] R. Martinez-Maldonado, V. Echeverria, O. C. Santos, A. D. P. D. Santos, and K. Yacef, “Physical learning analytics: A multimodal perspective,” in *Proceedings of the 8th International Conference on Learning Analytics and Knowledge*, ser. LAK ’18, Sydney, New South Wales, Australia: Association for Computing Machinery, 2018, ISBN: 9781450364003. DOI: `10.1145/3170358.3170379`. [Online]. Available: `https://doi.org/10.1145/3170358.3170379`.

[17] M. Garbarino, M. Lai, D. Bender, R. W. Picard, and S. Tognetti, “Empatica e3 - a wearable wireless multi-sensor device for real-time computerized biofeedback and data acquisition,” in *Empatica, Inc. and Massachusetts Institute of Technology*, Cambridge, MA, USA and Milan, Italy, 2020.

[18] STMicroelectronics, *St.com*, `https://www.st.com/en/microcontrollers-microprocessors/stm32-32-bit-arm-cortex-mcus.html`, [Accessed 10-12-2024].

[19] Arduino, “Arduino® Nano 33 IoT Product Reference Manual,” Tech. Rep., Dec. 2024. [Online]. Available: `https://docs.arduino.cc/resources/datasheets/ABX00027-datasheet.pdf`.

[20] R. P. Ltd, *Raspberry Pi 5*, `https://datasheets.raspberrypi.com/rpi5/raspberry-pi-5-product-brief.pdf`, [Accessed 10-12-2024], Aug. 2024.

[21] Espressif, *Espressif.com*, `https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf`, [Accessed 10-12-2024].

[22] S. Haq and P. Jackson, “Speaker-dependent audio-visual emotion recognition,” Sep. 2009. [Online]. Available: `https://www.isca-archive.org/avsp_2009/haq09_avsp.pdf`.

[23] S. A. Viswanathan and K. Vanlehn, “High Accuracy Detection of Collaboration From Log Data and Superficial Speech Features,” *Proceedings*, pp. 335–342, Jan. 2017. DOI: `10.22318/cscl2017.46`. [Online]. Available: `https://asu.pure.elsevier.com/en/publications/high-accuracy-detection-of-collaboration-from-log-data-and-superf`.

[24] N. Zhou, L. Kisselburgh, S. Chandrasegaran, S. K. Badam, N. Elmqvist, and K. Ramani, “Using social interaction trace data and context to predict collaboration quality and creative fluency in collaborative design learning environments,” *International Journal of Human-Computer Studies*, vol. 136, p. 102 378, Nov. 2019. DOI: `10.1016/j.ijhcs.2019.102378`. [Online]. Available: `https://doi.org/10.1016/j.ijhcs.2019.102378`.

[25] A. R. McGarva and R. M. Warner, “Attraction and social coordination: Mutual entrainment of vocal activity rhythms,” *Journal of Psycholinguistic Research*, vol. 32, pp. 335–354, 2003. [Online]. Available: `https://api.semanticscholar.org/CorpusID:6408212`.

[26] R. M. Warner, “Cyclicity of vocal activity increases during conversation: Support for a nonlinear systems model of dyadic social interaction,” *Systems Research and Behavioral Science*, vol. 37, no. 2, pp. 128–138, Apr. 1992. DOI: 10.1002/bs.3830370204. [Online]. Available: https://doi.org/10.1002/bs.3830370204.

[27] S. S. D’Mello, P. Chipman, A. Graesser, and U. Merced, “Posture as a predictor of learner’s affective engagement,” *Proceedings of the Annual Meeting of the Cognitive Science Society*, vol. 29, p. 29, 2007. [Online]. Available: https://escholarship.org/content/qt7hs9v2hr/qt7hs9v2hr.pdf.

[28] R. Taylor, “The multimodal texture of engagement: Prosodic language, gaze and posture in engaged, creative classroom interaction,” *Thinking Skills and Creativity*, vol. 20, pp. 83–96, Apr. 2016. DOI: 10.1016/j.tsc.2016.04.001. [Online]. Available: https://doi.org/10.1016/j.tsc.2016.04.001.

[29] J. E. Sasaki, K. S. Da Silva, B. G. G. Da Costa, and D. John, *Measurement of physical activity using accelerometers*. Jan. 2016, pp. 33–60. DOI: 10.1016/b978-0-12-802075-3.00002-4. [Online]. Available: https://doi.org/10.1016/b978-0-12-802075-3.00002-4.

[30] K. R. Doane, “Good posture and school achievement,” *The Clearing House A Journal of Educational Strategies Issues and Ideas*, vol. 31, no. 6, pp. 329–331, Feb. 1957. DOI: 10.1080/00098655.1957.11475591. [Online]. Available: https://doi.org/10.1080/00098655.1957.11475591.

[31] R. Sacchetti, T. Teixeira, B. Barbosa, *et al.*, “Human body posture detection in context: the case of teaching and learning environments,” Tech. Rep., May 2018. [Online]. Available: https://personales.upv.es/thinkmind/dl/conferences/signal/signal_2018/signal_2018_5_20_68004.pdf.

[32] H. Wojtaszek, A. Wojcik-Czerniawska, M. Mastalerz, and P. Stepien, “The role of consistency in verbal and nonverbal communication: enhancing trust and team effectiveness in management,” *EUROPEAN RESEARCH STUDIES JOURNAL*, vol. XXVII, no. Issue 3, pp. 621–636, Jul. 2024. DOI: 10.35808/ersj/3456. [Online]. Available: https://doi.org/10.35808/ersj/3456.

[33] M. Hoegl and L. Proserpio, “Team member proximity and teamwork in innovative projects,” *Research Policy*, vol. 33, no. 8, pp. 1153–1165, Sep. 2004. DOI: 10.1016/j.respol.2004.06.005. [Online]. Available: https://doi.org/10.1016/j.respol.2004.06.005.

[34] D. Vivet, P. Checchin, and R. Chapuis, “Localization and mapping using only a rotating FMCW radar sensor,” *Sensors*, vol. 13, no. 4, pp. 4527–4552, Apr. 2013. DOI: 10.3390/s130404527. [Online]. Available: https://www.mdpi.com/1424-8220/13/4/4527.

[35] I. Inc., *MPU-6000 and MPU-6050 product specification*, Aug. 2013. [Online]. Available: https://invensense.tdk.com/wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf.

[36] I. Inc., *Inmp411*, https://invensense.tdk.com/wp-content/uploads/2015/02/INMP411.pdf, [Accessed 10-12-2024].

[37] L. R. Rabiner and M. R. Sambur, “An Algorithm for Determining the Endpoints of Isolated Utterances,” *Bell System Technical Journal*, vol. 54, no. 2, pp. 297–315, Feb. 1975. DOI: 10.1002/j.1538-7305.1975.tb02840.x. [Online]. Available: https://doi.org/10.1002/j.1538-7305.1975.tb02840.x.

[38] C. Fisher, *Using An Accelerometer for Inclination Sensing*, May 2011. [Online]. Available: https://www.digikey.com/en/articles/using-an-accelerometer-for-inclination-sensing.

[39] A. Anand, *A brief study of discrete and fast fourier transforms*. [Online]. Available: https://math.uchicago.edu/~may/VIGRE/VIGRE2010/REUPapers/Anand.pdf.

[40] A. Leone, G. Rescio, A. Caroppo, P. Siciliano, and A. Manni, “Human postures recognition by accelerometer sensor and ML architecture integrated in embedded platforms: benchmarking and performance evaluation,” *Sensors*, vol. 23, no. 2, p. 1039, Jan. 2023. DOI: 10.3390/s23021039. [Online]. Available: https://www.mdpi.com/1424-8220/23/2/1039.

[41] R. Cox, S. De Campos Neto, C. Lamblin, and M. Sherif, “ITU-T coders for wideband, superwideband, and fullband speech communication [Series Editorial,” *IEEE Communications Magazine*, vol. 47, no. 10, pp. 106–109, Oct. 2009. DOI: `10.1109/mcom.2009.5273816`. [Online]. Available: `https://doi.org/10.1109/mcom.2009.5273816`.