UNIVERSITY OF TWENTE.

Faculty of Electrical Engineering,
Mathematics & Computer Science

Wearable device for interactive and collaborative sound making for autistic people

Christopher Benjamin Leonard

M.Sc. Thesis
March 2022

Acknowledgement

The entire master's journey was filled with ups and downs. The last few months were very challenging, and I would like to thank Gijs for being my supervisor and supporting me. I'm grateful for your patience and guidance. Also, I want to thank Jelle for this exciting assignment and your advice on approaching the problem.

Tremendous thanks go to all the participants who joined my research, and their feedback was invaluable for the development.

Lastly, I could not be more grateful to my friends and family for their immense love and support during the study period.

Abstract

Stimming is a repetitive action found among autistic people, like bouncing, jumping, flapping hands, etc. In the past it was suppressed as it was considered as odd actions. Suppressing it will create adverse psychological effects on autistic people. Recent research states that stimming creates positive feedback for them to relieve from stress. Even then, stimming was not encouraged and autistic people were judged often.

This research aims to develop a prototype based on DivComp's sense-making concept. It is a framework that connects people from diverse backgrounds (autistic and neurotypical individuals) to understand each other better. For this problem statement, it is necessary to make a neurotypical (NT) person understand stimming and encourage autistic people to stim freely without negative feedback.

A hardware and communication protocol to develop the prototype was selected based on the literature survey. It is necessary to create a prototype that consists of an audio recorder (microphone INMP441, storage, transfer), movement sensor (inertial measurement unit, IMU, Xsens Dot), microcontroller (ESP32) and soundscaper (Max/MSP) to involve in a sound collaboration activity. This prototype can record sounds on a button press and manipulate the recorded audio according to body movements using the soundscaper.

A series of workshops with NT persons was conducted to find the preferred functionalities for developing and improving the prototype. Finally, the prototype was user-tested and evaluated for network latency. This research demonstrates that it is possible to engage in a sense-making activity using a low-cost prototype.

Contents

List of acronyms

ADDM	Autism and Developmental Disabilities Monitoring
ASCII	American Standard Code for Information Interchange
ASD	Autism Spectrum disorder
CDC	Centres for Disease Control and Prevention
CD	Compact Disk
DMA	Direct Memory Access
DivComp	Diversity Computing
DOF	Degrees of Freedom
EMG	Electromyogram
FETS	Flexible Epidermal Tactile Sensor
FTP	File Transfer Protocol
FSR	Force Sensing Resistors
GPIO	General Purpose Input Output
I/O	Input/ Output
I2C	Inter-Integrated Circuit
I2S	I-squared-S
IMF	Interactive Musical Fruit
IMU	Inertial Measurement Unit
IMMU	Inertial and Magnetic Measurement Unit
IoT	Internet of Things
LDO	Low-dropout regulator
LED	Light Emitting Diode

MCU	Micro Controller Unit
MEMS	Micro-electromechanical Systems
MIDI	Musical Instrument Digital Interface
MMG	Mechanomyogram
NT	Neurotypical
OSC	Open Sound Control
PC	Personal Computer
Pd	Pure Data
SCK	Serial Clock
SD	Secure Digital
SPI	Serial Peripheral Interface
TCP	Transmission Control Protocol
UART	Universal Asynchronous Receiver/Transmitter
UDP	User Datagram Protocol
UIC	User Interface Control
WAV	Waveform Audio File Format

Chapter 1

Introduction

Around 1-2% of the world population is diagnosed with autism, a gene-based human neurological variant. Autistic brains are characterised by high levels of synaptic connectivity and responsiveness. Few autistic people practice “odd, unusual, or repetitive behaviours” such as hand flapping and body rocking and these actions are termed as ‘Stimming’. Not only people who are diagnosed with autism stims but also neurotypical people tend to stim to an extent (not frequent or socially acceptable). Many forms of fidgeting, such as twisting hair or tapping fingers, are also stimming [1]. These forms of stimming are so common that they often go unnoticed. However, odd movements associated with stimming like flapping hands, repeating words, rocking front and back, repetitive blinking are not accepted as normal behaviour by most populations, i.e. socially not accepted.

In earlier days, stimming was considered to be a result of trauma or emotional deprivation. Psychologist Ole Ivar Lovaas, an autism specialist, reportedly referred to them as “garbage behaviour.” Suppression of stimming was a priority because it was considered an abnormal action that creates discomfort in a social environment. In the process of suppressing stimming, inhumane acts such as electric shock, physical abuse, etc., were carried out as punishments or suppressant drugs were prescribed [2].

However, recent studies show that stimming is a self-regulatory mechanism that helps people calm down, cope with frustration and sometimes even with boredom. Yet, when a person stims oddly, they attract negative judgments in a social environment and to overcome this negativity, autistic people try to suppress their stimming, which induces stress or strain. Trying to hide a biological function can be torturous and can psychologically create a negative impact. The autistic person should be allowed to stim freely without any negative comments passed on them. We can achieve this by making neurotypical people understand why stimming is crucial for autistic people [3].

This research aims to develop a prototype device and system where a neurotypical person can stim along with an autistic person to understand stimming better. The wearable device prototype will allow the neurotypical person and autistic person to make sounds and music together and pave the way for interactive sense-making as users explore stimming with one another. Furthermore, it provides a common ground for two different individuals, i.e. a new communication method via music. Additionally, it encourages users to stim freely without prejudice and increase the sense of social interaction in an autistic person.

The thesis is organized as follows Chapter 2 covers the literature survey background of the problem statement, Diversity computing framework, sensors and wearability for the development of the prototype. Chapter 3 discusses the development of the prototype in detail, from the selection of hardware to the setting up. Chapter 4 is focused on the product flow and use case of the prototype. Further, Chapter 5 deals with the conducted workshops and prototype evaluation. Finally, the thesis ends with Chapter 6 conclusion and recommendations.

1.1 Research questions

For the literature survey, analysis on existing systems that suits the prototype. The main focus of the thesis is answering the below research questions

1. How to design a set of two wearable devices that allows users to record and manipulate sounds while also allowing them to connect with one another?

To answer the research question following sub questions have to be answered

1. How to measure the body movements (e.g. hands, legs or wrists) for sound manipulation?
2. What could the architecture and implementation for a prototype of the proposed system look like?
3. What technique(s) can be used to manipulate the audio signals?

Chapter 2

Literature Survey

This chapter consists of various literature studies that aims to the sub-question "How to measure the body movements (e.g. hands, legs or wrists) for sound manipulation?".

2.1 Autism and Stimming

2.1.1 Autism

Autism is a neurodevelopmental disorder characterised by deficits in social communication and the presence of restricted interests and repetitive behaviours [4]. Autism is a spectrum disorder, meaning it has a wide variety of symptoms and types. Autism can occur in people of any race or origin. It is formally known as ASD (Autism Spectrum disorder). Autism is said to be heterogeneous where each individuals can show wide spectrum of symptoms with different root causes. Irrespective of whether autistic people have different types of symptoms or impairments most of them show abnormalities in social reciprocity [5]. For the diagnosis of autism, along with at least 2 types of repetitive movements unusual to the environment, the individuals are assessed based on the problems they encounter with developing and maintaining relationships, social-emotional reciprocity and non-verbal communications [6].

During early days, autism was considered to be a severe psychiatric illness and was treated with electroconvulsive therapy (passing small electric currents) or shock therapy. Aversive punishments were carried out even during childhood to restrain their unusual behaviours. Later, in the 1970s, autism was branded as a behavioural and mental disorder. Autistic people were offered less opportunity in companies and no admission at schools [7] and researchers believed that behaviour modification was the appropriate treatment for autism [8].

The Autism and Developmental Disabilities Monitoring (ADDM) network, a part of the Centres for Disease Control and Prevention (CDC), after their research concluded that in recent days the diagnosis rate of autism is increasing. According to the data collected by ADDM, in the year 2000, 1 in 150 children were diagnosed with autism in the USA and by 2018 this number increased to 1 in 44 children [9]. This surge in the number of autistic people led to more research and to a deeper understanding that Autism is rather a social and emotional disorder than a mental illness. Nowadays, Autism can be diagnosed right at birth or at most by the age of 2 or 3 [10] [11]. Researchers, through participatory research, are trying to communicate and understand the needs and wants of autistic people. International communities have accepted that the situation of autistic people needs to be changed and have agreed to openly discuss the issues [12].

2.1.2 Stimming

Stimming behaviours are observed more commonly among autistic children during their daily activities [13]. In the early days, autism accompanied by stimming was considered to be caused by poor parenting. There were negative views on the self-stimulatory behaviour in autism since the behaviours were not socially acceptable [14]. Stimming can be categorised as visual, auditory, tactile and vestibular based on the repetitive action observed

Few researchers say that stimming causes adverse effects on an autistic person, like weakening their social interaction and slow learning. They believed suppression of stimming would reduce such effects. To suppress stimming, they incorporated punishment as primary reinforcement [15]. As a punishment, electric shocks were applied to eliminate "self-destructive" behaviours for the "well being". Implementing shock therapies created a sense of fear in autistic people, which was overlooked because of the temporary suppression of stimming [16]. In the 2010s, differential reinforcement was practised, not including physical harm. For example, the hand clasping reinforcement method was used to suppress hand flapping [15].

As mentioned previously in the introduction, stimming is a mechanism that offers an individual to focus their concentration, calm down during overwhelming emotions and deal with huge sensory overload. As per applied behaviour analysis, a person stims because it feels good. Many people fail to understand that not only autistic people stim but also non-autistic people stim to cope with a stressful situation or with boredom. Sometimes the consequences of uncontrollable emotions, either negative or positive, can trigger stimming. Stimming is also mainly the output of overwhelming stimuli [17]. Suppressing stimming would lead to worse repeated behaviour or adverse psychological effects. People can develop feelings of insecurity, anxiety, anger and exhaustion when they try to suppress stimming. The adverse negative reactions people have towards stimming are because they fail to understand its benefits [3]. Autistic people sometimes replace their stimming with other socially accepted activities like dancing and sailing. However, practising it takes more effort and is less effective [18]. They also agree that intervention of stimming behaviours should be discouraged and all non-harmful forms of stimming must be accepted. Now researchers are working on studying the beneficial effects of stimming and how to make non-autistic people understand stimming better. As per [18], most autistic people suppress stimming because of social pressure and thus the acceptance and understanding of stimming will create a positive effect on these individuals.

2.2 Diversity Computing

Diversity computing or DivComp is a non-normative framework proposed by Fletcher et al. [19] for technology mediated human-to-human interaction. DivComp provides scenarios that allow diverse individuals or groups to participate in active and reflective processes of meaning-making with the help of one another.

One of the approaches under DivComp is participatory sense-making, an interactive environment for people from different backgrounds to share their views and meaning. In the early '00s, various researchers came up with the concept of participatory sense-making but most of it lacked in the process of interaction between individuals. De Jaegher et al. [20] wanted to emphasise the importance of interaction in sense-making between individuals and moving away from social norms and towards understanding of an individual. Their work mentions cognitive engagement as a part of sense-making where one can interact with their whole body, actions and responding to each other's bodily actions.

De Jaegher in [21] believes that sensorimotor interactional coordination forms the basis of connection for interaction but autistic people show differences in sense-making and the way of interaction and perception varies. Due to the differences, there arises a difficulty in coordination which hampers participatory sense-making. These difficulties create a lack of social interaction and result in fewer opportunities to explore interactive sense-making for autistic individuals.

De Jaegher proposes exploring different interaction domains of sense-making if one or the other standard method fails. Erik Rynell, based on this concept of De Jaegher, suggested acting as another way to interact during participatory sense-making [22].

His theory states that acting makes meaning similar to a real-life conversation. On the other hand, Nicolas Davis et al. [23] proposed a creative environment, ”casual creators” for abstract drawing. In which the user can make sense with another fellow user in a virtual drawing environment. Schiavio et al., supports the previously mentioned work of De Jaegher by stating that the cognitive process does not happen in the head alone but also integrates with the body in different ways [24].

In this research, a new perspective that involves musical performers as interactive agents is explored. Studies have shown that music usually leads to spontaneous body movement, even when people consciously are standing still [25]. There is a strong link between body movement and music. Performers produce music using body movements while the listener dances (body movement) according to the music [26] and thus creating or playing music can act as a way to develop skilled coordination and can be a mode for sense making for both the musicians and audience.

Sense making through music can be especially beneficial with autistic participants as their stimming behaviours like flapping hands, jumping, moving to and fro are sound producing and sound accompanying movements. Sound producing movements are movements that are closely related to the stimming such as tapping, knocking, hitting, scrubbing, etc that can create music from everyday objects while sound accompanying movements are the movements done mostly by listeners like dancing by moving the whole body, adjusting body movement to the tempo of the music mostly by tapping, nodding the head [27]. Additionally, many autistic people say that listening to music, singing or playing an instrument, reduces their need for stimming.

In [28], the author discusses Divcomp framework to establish a new meaning for stimming in the form of a sound collaboration activity. Similarly, this research also proposes a sound collaboration activity based participatory sense-making between two diverse individuals. For participatory sense-making, 2 individuals should engage in reciprocatable joint activity to understand each other better [21]. The concept developed by [28] allows Neurotypical (NT) and autistic people to engage by expressing themselves through body interactions. It also allows the users to experience a rule-free exploration environment while providing guidelines for the development of a stimming device that is based on DivComp.

With music having the ability to capture the attention of both autistic and NT individuals, this research aims at providing a music based diversity computing model device. The device is based on the concept developed by [28] for a NT person and an autistic person to understand one another, undoing the existing normative order.

2.3 Sensing

As per [28] a device suitable for sound collaboration activity, should allow the users to control the soundscapes. Since the music (sounds) and body movements are closely related like in the case of dancing, orchestra, playing an instrument, etc, the repetitive movements ”Stimming” can also be used as a controller of the soundscape. This section discusses the various types of sensors that can be used for the implementation of a prototype that measure body movements and use it to manipulate the soundscapes. Ultimately, appropriate sensors are for the proposed concept are selected.

The design idea is to manipulate audio (music) signals based on body movements. Most of the current gesture recognition prototypes use sensors such as accelerometers, gyroscopes, inertial measurement units (IMU), force pressure sensors, flex sensors, etc., to measure body movements [29] [30]. These sensors can be divided into two categories - contact sensors and non-contact sensors. Each sensor has its merits and demerits depending on the area of application and are discussed in the following sections.

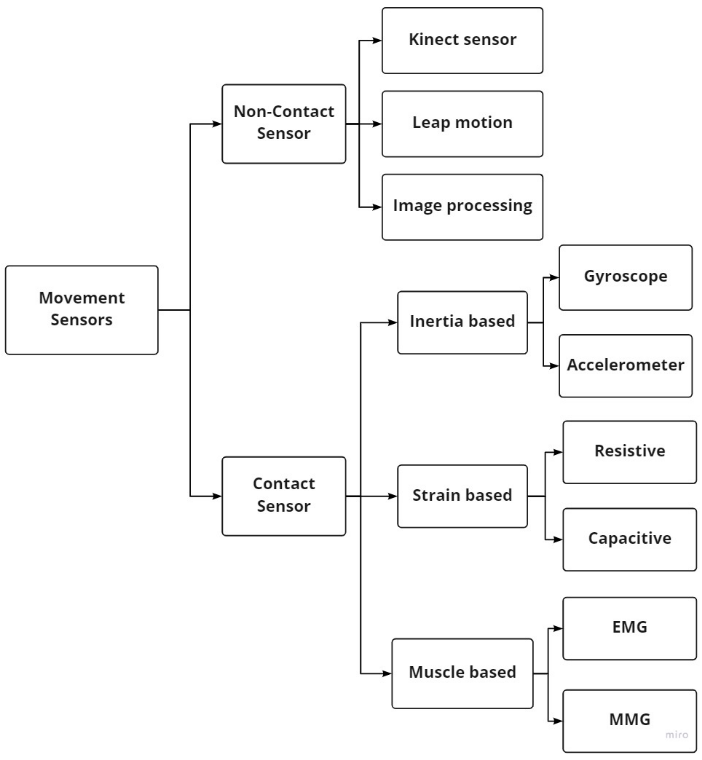


Figure 2.1: Classification of sensor technologies to measure body movements [31] [32]

2.3.1 Non contact sensors

Camera-based systems where the body movements of an individual are captured with the help of video processing techniques are most often described as non-contact systems.

Vision-based systems are reliable only with proper lighting conditions and camera positioning [33] [34]. Kinect (Microsoft) [35] uses a colour camera, infrared projector and depth camera to detect body movements. However, Kinect needs specific hardware requirements to function properly and the data needs to be collected and analysed separately to extract the palm, finger and wrist movements. Another vision-based gesture recognition system is the Leap motion controller (Ultraleap) [36] which utilises optical infrared sensors. Unlike Kinect, the Leap motion controller internally calculates the finger's position, hand centre and orientation and its output can directly be used for actualisation. Even though it has better performance than Kinect, there were delays and difficulties while mapping the music to the action when fingers are close to one another [36].

2.3.2 Contact sensors

Sensors that are body-worn or attached to the garments come under this category. Liu et al., 2020 designed a wearable device using a mechanomyogram (MMG) based sensor, along with a 3 axis accelerometer. Their prototype is designed and programmed in such a way that it can identify 8 different hand gestures with an accuracy of 94% by taking samples of muscle vibration frequencies [29]. Byun and Lee, 2019 suggested a new way of recognising hand gestures using Flexible Epidermal Tactile Sensor (FETS) by monitoring the muscle activity while having the sensor connected to the wrist. They used 4 arrays of FETS to detect 5 different wrist movements with an accuracy of 97%. In addition to this, several wearable sensors detect muscle stress when there are slight movements in the body like strain sensors, EMG, force-sensitive resistors, piezoresistive sensors, etc. [30]. [37] developed 3D printed soft EMG 8-electrode band to recognize hand gesture using pattern recognition 80-90%.

In [38] EMG and MMG sensor bands are used to detect muscle movements. Even though muscle movements are closely related to music creation, it fails to acquire the speed of the movements or rotation, characteristic of the music's tempo. To determine or to track these movements, IMU sensors are widely used. They are small and highly power-efficient and contain accelerometers that measure the acceleration in a specific direction and gyroscopes that measure the orientation. Filippeschi et al., 2017 [39] proposed a system with IMU sensors to track the motion of limbs. They implemented 5 different methods for tracking motion in the lower limbs and validated them against optical motion capture systems. Similarly, Tajadura-Jiménez et al., 2018 [40] developed a gesture-sound wearable system using force sensing resistors (FSR) connected with a Bitalino R-IoT sensor module containing IMU sensors. This setup is placed on the user's legs and wirelessly connected to Max/MSP (a programming language for music).

They also proposed different sensor mappings for sound generation. Visi et al., 2017 [41] also worked on motion capturing using IMU sensors. The IMU sensor returns a 3-dimensional vector representing acceleration, rotational velocity and orientation. The prototype was equipped with Myo sensors to detect the movement of the fingers. From the data gathered from the sensors, an algorithm in Max/MSP was implemented for real-time audio processing.As a proof of concept, T. Nguyen, in 2021 [28], implemented a Stim4sound basic prototype to prove the Divcomp concept of sound collaboration activity. The author implemented a hand gesture recognition system using an IMU sensor and Neural Networks to identify three gestures. The identified gestures are used to manipulate sound properties. Based on the references, most of the sound related applications used IMU in the prototype. It could be concluded that IMU sensors are most suitable for motion-based music manipulation systems.

2.4 Wearable design and wearability

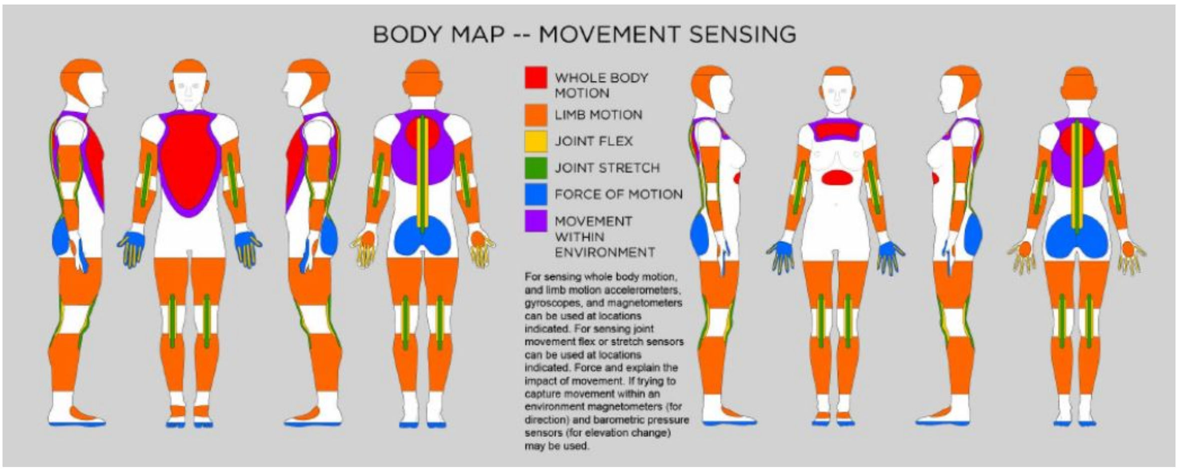
Wearable sensors have become more common in recent years, one of the rising concerns of wearable devices is appropriate placement of the sensors. Designing wearable devices should also take the area of application into consideration for the better placement of the sensor for efficient output. Wearable devices should be designed in such a way that it is comfortable for the wearer with the ability to move freely with no strain or pressure [42].

With technological advancements, wearable devices are starting to be a part of fashion and daily use. Experimentation in the field of wearable devices in terms of electronics and design has led to a set of principles for wearable technology. In [43] the authors discuss the goal of wearable design as it being more natural and in close contact with the body most of the time for easy monitoring. They also provide a set of technological characteristics for a wearable device - energy efficient, compact, effective wireless and secure transmission.

In mid 2010's, Motti and Caine reviewed different wearable device scientific articles and framed 20 human-centred design principles for wearable development which mainly focuses on the physical, cognitive and emotional aspects of design development [44]. This allows developers to focus on user-centric development and narrow down the requirements depending on the target group while also simplifying the whole process.

When the wearable development is entirely focused on social interaction it is important to make sure that there are no distractions during user interaction. A design framework is proposed by Dagan et al., which is used to evaluate a wearable design, making the developer question each and every design choice right from sensing/input to the output and social acceptability. Examining the development's main requirements helps us stay on track with the development [45].

Another main aspect of the wearable device is the placement of the sensor; in [46] and [47] the placement of the sensor/ wearable device on the body and how it should be placed on proper locations to capture the required movements for the application, are discussed. Figure 2.2 shows the map for sensor placements for various movement detection requirements.



For sensing whole body motion, and limb motion accelerometers, gyroscopes, and magnetometers can be used at locations indicated. For sensing joint movement flex or stretch sensors can be used at locations indicated. Force and explain the impact of movement. If trying to capture movement within an environment magnetometers (for direction) and barometric pressure sensors (for elevation change) may be used.

Figure 2.2: Movements sensor placement body map [47]

Stim4Sound mostly focuses on stimming with the body which is most closely related to movement with limbs, wrists or head movement. On that aspect, placing the sensor on the wrist allows efficient information collection on the hand movements if the user uses the hands. On the forehead as a band if the user desires to use head movements and can also be strapped to the leg to detect the leg movements. This placement also makes sure that it is socially acceptable and non-obstructive in normal behaviour.

2.5 State of the art

2.5.1 Sound Bikes

Maes et al developed a musical installation which is an example of embodied and collaborative interaction. The main focus of the SoundBikes is to develop and implement participatory sense making in practice which is a part of [20]. This musical installation allows the users to dynamically control the tempo and filter pre recorded music using sensors. They have installed stationary bikes equipped with sensors which can detect rotations per minute, weight balance and phase consistency. All actions are mapped to a musical feedback. They achieve participatory sense making possible by involving co-creation of music with synchronisation of tempo, phase and balance. Figure 2.3 shows the SoundBikes used at a museum [48]

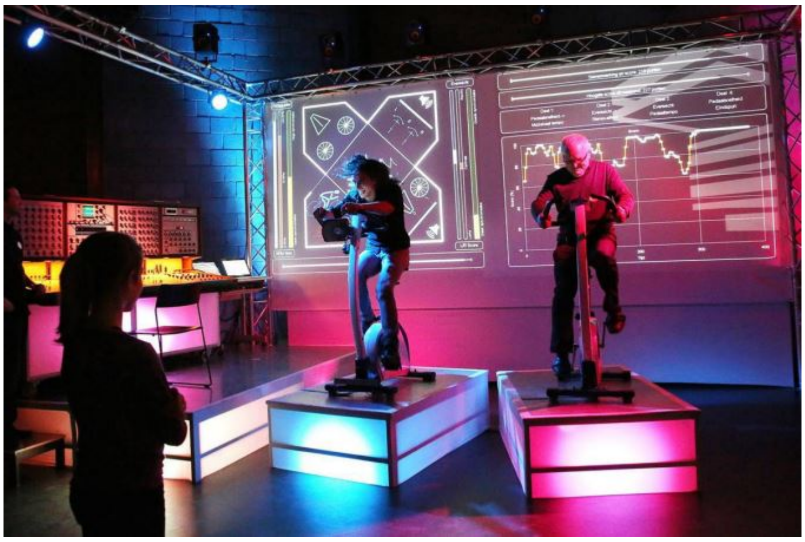


Figure 2.3: Impression of SoundBikes, in action [48]

2.5.2 Telematic Wearable Music

Telematic wearable music is a low cost IoT based wearable device which can be used to compute and add sounds with respect to the IMU information, see Figure 2.4. The primary focus of this project is to enhance embodied participation in remote learning environments. Using M5 Stick-C, a device which sends IMU data via User Datagram Protocol to a media processing software Max/MSP, students can develop their own end product like musical wings, earrings, gloves and more. This wearable device embodies sense making by allowing students to play music together. The act of group interaction involuntarily elevates the emotions in a positive way and keeps the students engaged. Telematic wearable music is for people to creatively explore and involve in participatory sense making with music. [49]

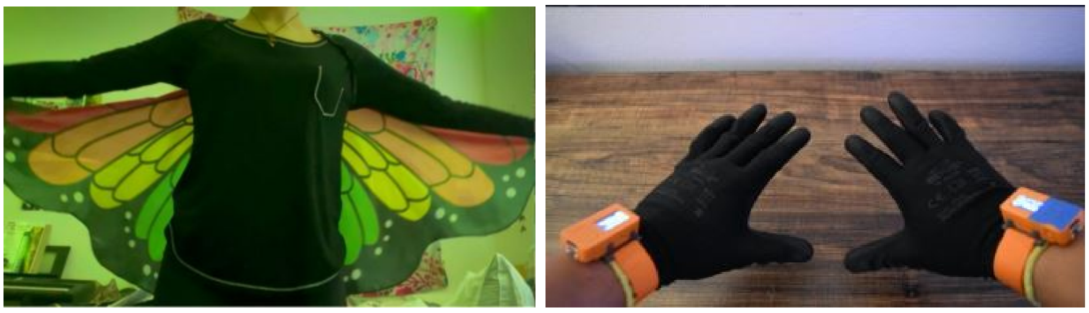


Figure 2.4: Final projects developed by students using Telematic wearable music (butterfly wings and gloves) [49]

2.5.3 Vrengt: A Shared Body–Machine Instrument for Music–Dance Performance

Vrengt is a device that creates musical performances based on the bodily interactions between the musician and the dancer [50]. In this paper the authors discuss creating a single instrument that can provide room for interaction between the musicians and the dancers based on human centred design. For the hardware implementation, 2 myo bands and a MIDI controller are connected with Max/MSP patches. They have mapped micro, macro and meso movements (EMG data from myo) of the dancers to a user interface where the resultant sounds are controlled by the musician. EMG data are transmitted to Max/MSP via OSC (communication protocol between computers, sound synthesizers, and other multimedia devices). Along with the movements, breathing is converted into audio signals which is also played along with the resultant sound.

2.5.4 Interactive Musical Fruit

Interactive musical fruits (IMF) is an interactive installation developed by Erkut et al. [51], which uses music as its end product, see Figure 2.5. This paper discusses embodied interaction and action-sound correlation, mainly focusing on children. The implementation of IMF was carried out with a Raspberry Pi - core system, MPU9150 - IMU sensors to detect orientation changes and a LED chain for visual feedback. They mapped action to sounds for the movements to left/right and up/down with different filters. They used pure data and OSC to transmit the sensor data. They tested IMF with the focus group which showed that more than one child can play with the prototype creating a way for a collaborative system.

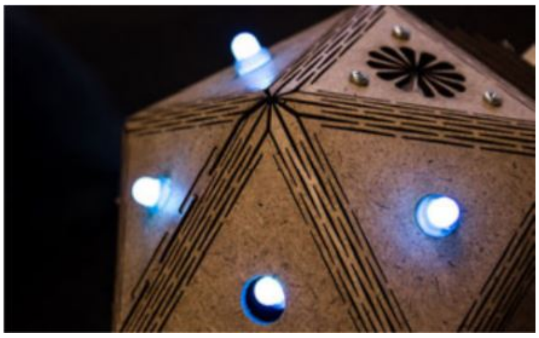


Figure 2.5: IMF 3d prototype with LED chain feedback [51]

2.6 Conclusion

The literature survey focused on the topics that would make an NT person understand stimming better and autistic people to stim freely. This background research started with autism and stimming. It focused more on the views from earlier to recent days and the negative impacts on autistic people because of prejudice. Following that, the literature survey focused on the DivComp framework model and their concept of sense-making, which proved efficient for the diverse individuals to understand each other. Choosing sound collaboration activity laid out a set of requirements followed for the development of the prototype. The requirement was to measure the body movements while stimming. A detailed survey on sensors was carried out to select the type of sensor to be used. In addition to the sensor, the sensor's wearability was discussed for the placement of the sensors. Similar prototypes were explored using sounds and movements for an engaging activity.

In this chapter, one of the sub-questions were answered partially "How to measure the body movements (e.g. hands, legs or wrists) for sound manipulation?" IMU sensors were decided to be used because of their accuracy and flexible usage for measuring body movements. The sensor's placement also plays a vital role in measuring body movements with high accuracy. The selection of hardware will be in the next chapter, and with the sensor selected, this will be answered.

Chapter 3

Development of Prototype

This chapter deals with developing the primary prototype of the wearable device for recording and a body movements-based music controller. The research question of "What could the architecture and implementation for a prototype of the proposed system look like" will be answered in this chapter.

There is a set of requirements/functionalities to be fulfilled to implement the concept successfully. In the next section, first the concept of the wearable device will be discussed.

3.1 Concept of the Wearable device

The concept of Stim4Sound [28] provides a new way for understanding stimming and expressing through body interaction. The concept based prototype should pave the way for the users to engage in activities that involve sound-making and help explore the soundscapes. The process of the Stim4Sound is given below in a concise manner.

1. The entire collaboration process starts with two people (NT and autistic) exploring and making music with the objects available and finding interesting sounds.
2. As a subsequent stage of the interaction, the users can record the sounds by taking turns.
3. Following it, people can stim freely, to manipulate the sound properties of the recordings in order to derive fun and exciting sounds from the original recordings.
4. Understanding one another during the process, the NT person experiences the necessity of stimming and the autistic person experiences the freedom of expressing themselves in a social interaction without being judged.

To implement the concept of Stim4Sound, a set of features and requirements for the surroundings are to be fulfilled.

i. The activity should take place in a surrounding that is comfortable for both users

ii. The system should allow the users to communicate between each other,

iii. to record sounds,

iv. to measure body movements,

v. provide a sound platform to collaborate

vi. through a user interface control.

vii. Little human intervention with the end system is desired

viii. Collaboration times of up to ≈1hr are desired

The system and functional requirements to implement the features are listed in the below section 3.2. Following the requirements section, the system architecture is discussed, which helps with the choice of hardware for implementation.

3.2 Requirements

In this section, the essential requirements needed for the prototype are listed to help with the implementation. Usually a wearable system comprises different functions like sensing, feedback, transmission, controller, etc.,. As per the Stim4Sound concept requirements, there should be sensing, recording, communication, controller and computation modules. The prototype system is divided into modules as shown in the block diagram in Figure 3.1. It also shows whether each module is a wearable or a stationary part of the system.

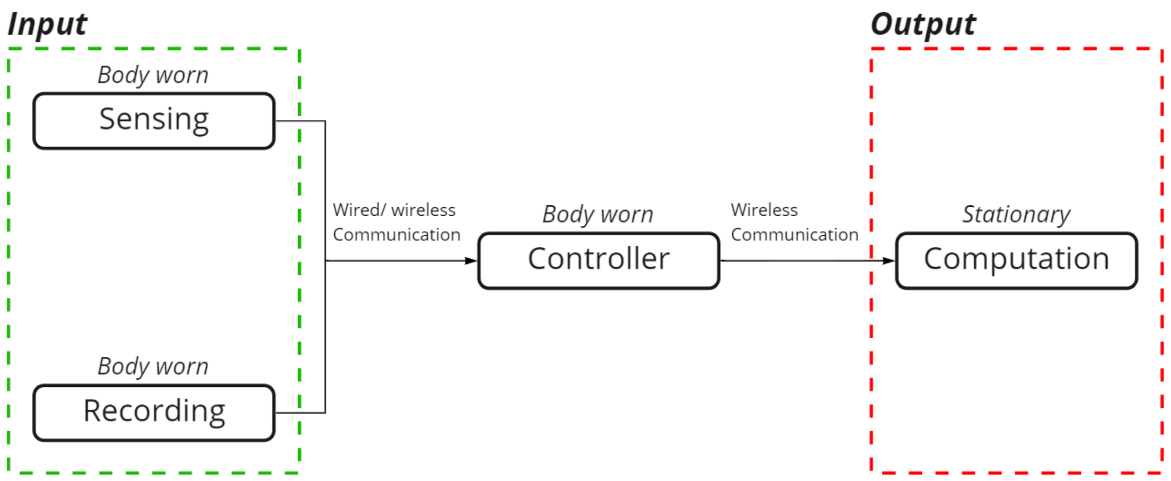


Figure 3.1: System requirements to implement Stim4Sound

3.2.1 System requirements

- Input: It comprises the sensing and recording modules that receive the data from the surroundings and send it to the communication module.
  - Body movement sensor: It is a part of the input module that consists of body movement measuring sensors that are essential to measure body movements (Stimming). It is a "body-worn" module to give accurate results.
  - Recorder: This module comprises an audio sensor that gathers surrounding audio signals. The system should allow the user to explore different soundscapes in the surrounding area; hence this module is also "body-worn" to achieve free movement within the user's surroundings.
- Communication: it is a part of the system that transmits the data from the sensor and the recorder module to the Soundscaper module, which manipulates the sound properties of the recorded sounds. Since the system's main aim is to allow free body movement to support it, the communication system should be wireless.
- Controller: it is the user interface controller (UIC) of the system where users can control the functionalities of the system.
- Output: it is the end of the process flow and it consists of a soundscaper module. The required system output is obtained from this part of the system.
  - Soundscaper: It is the final module of the system where the processed data from the controller module is transferred. This module uses the processed data to manipulate the music as per the concept requirement. This computation part is a software development environment available on a desktop/ laptop, and it is the stationary part of the system.

3.2.2 Functional requirements

- Requirement 1: the prototype should record music on a button press. Since the concept of Stim4Sound is to allow users to explore different soundscapes and allow them to record, having a record button allows the user to take control and record when they find intriguing sounds. (Feature: allowing the users to record. Module: recorder. System: input).
- Requirement 2: the prototype should store the recorded music to the desktop for future use and this should be automated to minimize human intervention. (Feature: little human intervention with the end system. System: controller).

- Requirement 3: the prototype should have a button interface to control the tracks. The prototype allows the user to record sounds, and a user might find different intriguing sounds to be recorded separately. Users should not be limited to one recording at a time or control only one track at a time. Therefore it is essential to create a user interface to control as many tracks as possible with easy use. (Feature: user interface control. System: controller).

- Requirement 4: the prototype should send data from the sensor to the computation module. Since the concept behind Stim4Sound is free body movement, the prototype should send raw sensor data wirelessly for further sound processing. (Feature: measure body movements. little human intervention with the end system. Module: sensing. System: input).

- Requirement 5: the computation module should allow the user to manipulate the music through body movements and collaborate easily. (Feature: sound platform to collaborate. Module: computation. System: output).

- Requirement 6: the prototype should work in real-time and be responsive. It is essential to ensure that the prototype is responsive to the body movements and process the sounds recorded based on it without any significant delay. (Feature: quality of the prototype).

- Requirement 7: the prototype should have enough battery backup for the system to run for more than an hour. (Feature: sufficient long collaboration time 1hr. System: controller).

3.2.3 System architecture

There are four modules along with a set of requirements to implement the complete system. In this section the functional and performance of each module will be discussed.

Body movement sensor

As discussed in subsection 3.2.1, the body movement sensor module consists of sensors to measure the movements. As per the literature survey, IMU sensors were the choice of sensor that pick up the body movements. An IMU combines multiple sensors, namely accelerometers, gyroscopes, and compass, to measure angular rate, force, and heading. In addition, the IMMU sensor consists of a magnetometer.

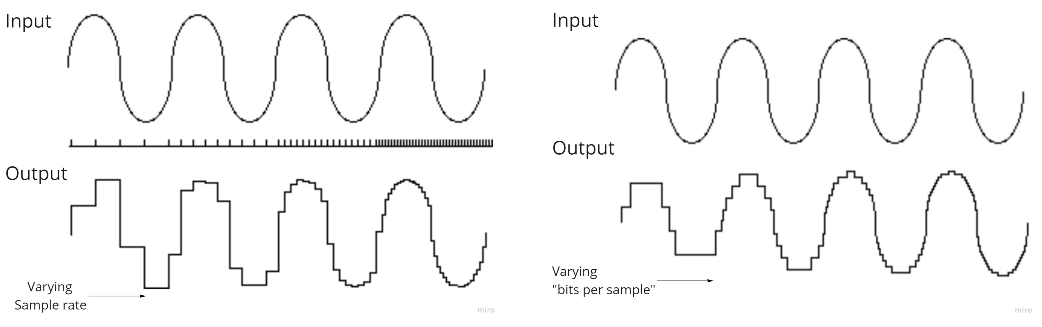
- Accelerometers: the accelerometers measure the acceleration ($\mathrm{m\,s^{-2}}$) in x, y and z directions and are also used to determine pitch and roll.

- Gyroscopes: the gyroscopes measure angular velocity ($\mathrm{rad\,s^{-1}}$) and help to determine orientation. Also, measuring angular velocity and integrating over a period of time gives the change in orientation (rad) in that period.
- Magnetometer: the magnetometer measures earth's magnetic field (or artificially generated magnetic fields) and acts as a compass (points to the north).

The IMU is expected to give the orientation and the acceleration when there is movement in the body part to which the IMU is attached. The IMU is expected to perform better with high output rates. In addition to the performance of the IMU, the shape and size should also be considered; the IMU needs to be compact and it should be possible to place it anywhere on the body as per the requirements. In addition, using a low powered sensor reduces energy consumption.

Recorder

The recorder is a module where audio signals are acquired by an audio sensor and further handled in the controller unit. For this application, recording is a process of storing the audio signals in a playable audio format. To give an overview of the digital recording system, it consists of an analog-to-digital converter (ADC) that converts the audio signals into digital information. The ADC converts the audio signal into discrete numbers known as "samples", and the number of samples present in a second is known as "sample rate". These samples are presented in the form of bits (binary), e.g., number 2 is represented as "0000000000000010" in a 16 bits binary form, it is known as "bits per sample". These are the few critical parameters of a recording system as it directly affects the quality of the end product. The image below can give a clear understanding.



(a) Sample rate (b) Bits per sample

Figure 3.2: Impact of the critical parameters on the output signal [52]

When sampled at a lower rate, the input analog signal loses information and fails to recreate the same signal. However, the input information is increasingly well preserved at the output when the sampling rate increases as can be seen in the 3.2(a).

When an input signal is sampled with a lower number of “bits per sample”, there is a loss of information. As the “bits per sample” increases, the quality of the signal increases, as can be seen in the 3.2(b).

The recorder module has a set of requirements for parameters to achieve an acceptable audio output and they are followed below.

- Sampling rate: 16 - 48 kHz
- Bits per sample: 8 - 32 bits

Recording system requirements:

- Microphone (Analog/Digital) to receive the audio data.
- Low powered
- Data storage

Communication

Communication here means the communication protocols used between hardware or devices connected to the same network. In this prototype, there is a need for sending and receiving data between the controller and the peripherals.

- Sensor data must be streamed continuously from the sensor module to measure body movement.
- The recorded data must be sent to the controller for further processing from the recorder module.

Meanwhile, the controller also communicates with input and output modules to control actions.

To achieve the communication between hardware, there are a few commonly used hardware protocols

- Serial Peripheral Interface (SPI),
- Inter-Integrated Circuit (I2C),
- Universal Asynchronous Receiver/Transmitter (UART),
- I-squared-S (I^2S, or I2S).

Each communication protocol has its application area in hardware interfacing and is selected based on the speed of data transfer or compatibility with the hardware.

Controller

The controller is at the core of the system. It receives data from the input modules to process and transmit it to the output module. The controller should have the ability to support the communication interface with the sensor and recorder modules to receive the data without any interference. The crucial requirements for choosing the controller are as follows:

- Wireless: Bluetooth/ WiFi
- Low power consumption
- Storage capacity
- Available inputs and outputs
- Hardware interfaces
- Compact
- Cost

Choosing a controller device with this set of features will make the prototype suitable for implementation.

Soundscaper

It is necessary to choose a suitable audio manipulation to achieve the required outcome. It receives the data from the input module through the controller. A software application or sound platform is required to receive the sensor data and to manipulate the sound properties of the recording. Many audio synthesis software environments are based on programming languages like python, graphical, C/C++, etc. For comparing the available software few attributes are looked into, that are mainly

- Ease of use
- Audio quality
- System integration
- System Capacity
- Support forums

3.2.4 Overview of available protocols

Before selecting hardware, it is necessary to have a basic understanding of the protocols used for interfacing. In this section, a brief introduction to the protocols is given.

SPI is the most used serial communication protocol to send and receive a continuous stream of data. Peripherals such as SD card modules, LCD displays or Radio-frequency identification (RFID) modules use the SPI protocol to interface with a microcontroller. SPI communicates using the master-slave concept, where the master mostly will be the controller and the slave will be peripheral. Moreover, it is possible to send and receive data simultaneously (full-duplex). SPI allows only one master, but many slaves can be connected to the master. SPI requires four signals to establish an interface for one slave. [53]

- Master Output/Slave Input (MOSI) - Master to send data to the slave.
- Master Input/Slave Output (MISO) - Slave to send data to the master.
- Clock (SCLK) - Clock signal.
- Slave Select(SS) - Select a particular slave to send data

UART is an IC in a microcontroller that can transfer and receive data with another UART IC. UART transmits the data serially and only requires two pins to communicate between 2 UARTs, namely Tx (Transmitting) and Rx (Receiving) pins. UART communications are sent in packets that can be used to identify any data loss if a bit is missing.

I2C is a communication protocol that uses only lines similar to UART and can also communicate using the master-slave concept as SPI. I2C can have multiple masters and multiple slaves connected to them. Before transferring data, I2C tries to establish a handshake between master and slave. Then messages are transferred in data frames, which consist of the address of the slave device along with a start, stop and acknowledge bits.

- Serial Data (SDA) - Master and slave to send and receive data.
- Serial Clock (SCL) - Clock signal.

In communicating digital audio signals, the I2S protocol is widely used. I2S is similar to I2C but introduces one extra pin called “Word Select” and avoids handshake with the audio peripheral (slave). I2S is mainly designed for transferring data for stereo sounds. I2S has three pins for communication.

- Serial Data (SD) - Send data to the master
- Word Select (WS) - Represents audio channel (left/right)
- Clock(SCK) - Clock signal

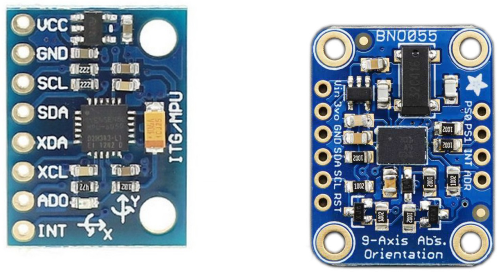
The choice of protocols was taken according to the hardware compatibility in the next section.

3.3 Hardware

This section discusses and justifies the choice of hardware to implement and fulfil the system requirements. Each subsection will include available hardware and a selection of hardware.

3.3.1 Body movement sensor

The criteria to choose an IMU for the implementation are known from the sensor module section 3.2.3. There are many available IMU sensors in the market that would fit our requirements. The two most common types of IMU sensors are 6DOF IMU and 9DOF IMU. A 6DOF or 6-axis IMUs comprises a 3-axis accelerometer and a 3-axis gyroscope. Meanwhile, 9DOF or 9-axis IMUs are composed of a 3-axis magnetometer and a 6-axis IMU.



(a) MPU6050 [54] (b) BNO055 [55]

Figure 3.3: IMU sensors

A small implementation was done using MPU6050 (Figure 3.3(a)), a 6DOF IMU and the same implementation was also done using BNO055 (Figure 3.3(b)), a 9DOF IMU. In this implementation, sensors are connected to an Arduino UNO, through which sensor data are collected. Both the sensors worked as expected, as only the linear acceleration and orientation data were used. These are wired sensors using I2C communication subsection 3.2.4 for communicating data. There are also wireless sensors available that work with Bluetooth for transmitting sensor data like Xsens Dot.

MPU6050	BNO055	Xsens Dot
I2C - Wired communication	I2C - Wired communication	Bluetooth - Wireless communication
Small and compact	Small and compact	Small and compact
Output data rate for Gyroscope - 4 Hz to 8 kHz Accelerometer - 4 Hz to 1 kHz	Output data rate for Orientation - 100 Hz Acceleration - 100 Hz	Output data rate for Real time - 1 Hz, 4 hertz, 10 Hz, 12 Hz, 15 Hz, 20 Hz, 30 Hz and 60 Hz Recording - Real time output rates and 120 Hz.
Current consumed 3.8 mA	Current consumed 12.3 mA	Battery stand by - 9 hours if used continuously

Table 3.1: Comparison of sensors - MPU6050, BNO055 and Xsens Dot

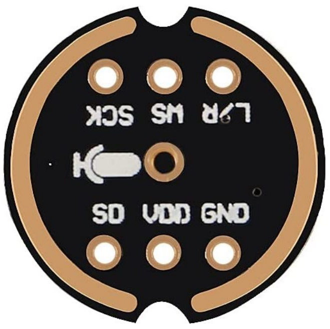
The Table 3.1 shows that the MPU6050 has a far larger output data rate than the other choices. However, it will be difficult to process the streams of data at a high data rate. The output data rates can be in the range of 100 Hz for this application. Taking wireless and wired communication into consideration, body movements can be measured at multiple locations on our bodies. In the case of measuring at multiple locations, wired communication between the sensor and the controller can affect the free movements or require more than one controller for each sensor. Xsens Dot has a battery capacity of 70 mA h which can continuously supply power to the sensor for 9 hours. The perks of selecting Xsens Dot is that it can work as a stand-alone device, unlike MPU6050 or BNO055 and also, that is has better accuracy. Xsens Dot comes with five sensors that can be used to measure body movements at multiple locations. Based on the requirement of a wireless prototype and to measure body movements at multiple locations choosing Xsens Dot (Figure 3.4) is a better fit.

Figure 3.4: Xsens Dot [56]

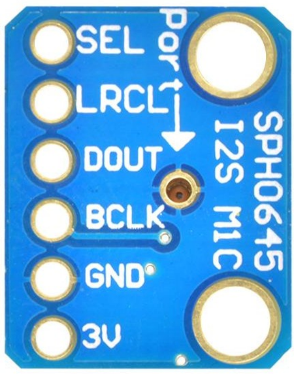
3.3.2 Recorder

In this section, the implementation of the recording function of the audio signals is discussed. An audio signal can be recorded using a headphone's microphone (via audio jack) with the help of a controller with inbuilt/external audio codec to convert analog to digital signals and digital to analog signals or by using a MEMS digital microphones.

When looking into the available audio sensors for the implementation, two different I2S MEMS microphones were shortlisted - Adafruit I2S microphone SPH0645 and InvenSense I2S INMP441. Compared to the recording system requirements, both sensors can be connected with a microcontroller without an audio codec. Both use the I2S hardware interface, that is widely used for digital audio communication, and are only compatible with microcontrollers that support I2S communication subsection 3.2.4. In addition, both sensors allow a sampling rate range of 16 kHz - 48 kHz with a maximum of 16 bits per sample.



(a) InvenSense INMP441 [57]



(b) Adafruit SPH0645 [58]

Figure 3.5: I2S microphones

All these above properties match the system requirements. So, the current consumption and shape are taken into consideration. The Adafruit has lower energy consumption than the InvenSense I2S microphone but in terms of shape and size, the InvenSense is more compact than the Adafruit Figure 3.5. After looking into all the pros and cons, both sensors are very similar with the features they provide. INMP441 has better availability and a lower price compared to the other. Ultimately, cost was a deciding factor to select the INMP441 as the choice of audio sensor.

Storage

The storage is also part of the recording system requirements as per section 3.2.3. It is essential to store the recordings for reusing, and always good to have an extra storage element to reduce the controller's internal memory usage. A microcontroller has an internal flash memory that stores data that will be preserved even after the system is restarted. It is also known as non-volatile memory. It is mostly used for storing the software code to run the microcontroller. It is best to not use the internal memory to store other data, if the code space is larger. So it would be a better system design choice to add a storage device to the prototype.

Among the available external storage elements, the most often used are SD cards, which are convenient, small, compact and easy to use. There are a few prerequisites to use an SD card in our prototype. A controller must

- have an internal or external card reader,
- be able to communicate with the controller for data transfer,
- have the ability to create, read, update, and delete files.

A micro SD card is used in this prototype since it is widely available. An SD card reader is always selected based on the SD card used, so a suitable micro SD card reader should be selected for the prototype. Two different SD card readers are available, one with 3.3 V and another with 5 V operation voltage. Both the SD card readers interface with the microcontroller using the SPI protocol subsection 3.2.4. The significant differences between the two are the shape and energy consumption, where the 3.3 V device is smaller, compact and consume less energy when compared to the 5 V device. So, a 3.3 V micro SD card reader will be used for further implementation of the prototype.

3.3.3 Controller

This subsection discusses the selection of a microcontroller based on the critical requirements listed under the section 3.2.3. There are lots of available wireless microcontrollers that can serve the purpose of Stim4Sound. In this prototype, the microcontroller needs to process the input data from the input module Figure 3.1 and act as a User Interface control (UIC) of the prototype. So the microcontroller selected for this prototype must fulfil these requirements at most for better performance.

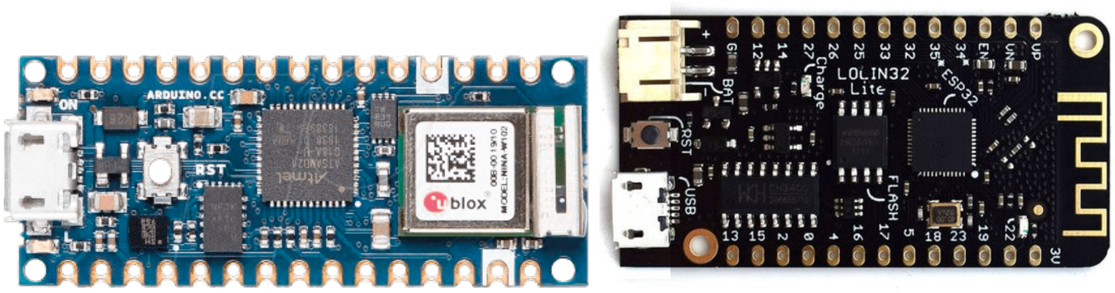
Based on the requirements, two microcontrollers with Bluetooth and WiFi components - ESP32 and Arduino nano 33 IoT- are analysed. Each device has a slightly different configuration in terms of memory, the available number of I/O pins and energy consumption, which are tabulated in Table 3.2.

	WeMos LOLIN32 Lite (ESP32)	Arduino Nano 33 IoT
Flash Memory	4 MB	256 KB
Digital I/O pins	34	14
Deep sleep mode current	10 µA	10 µA
Bluetooth current	130 mA	130 mA
WiFi Tx packet current	190 mA - 240 mA	190 mA
WiFi/BT Rx and listening current	80 mA - 90 mA	95 mA
PCB Antenna	MIFA(31.4x18x3.2) mm	PIFA(10.0x14.0x3.8) mm
Communication protocols	I2S, SPI, I2C	I2S, SPI, I2C

Table 3.2: Comparison between ESP32 and Arduino Nano33 IoT

When looking into the table, it is clearly seen that both devices are more or less similar, but significant differences are flash memory, number of I/O pins and current consumption.

- Flash memory (Non-volatile memory) is essential for a controller to store the code and few other essential small data to retain after restarting. As mentioned in section storage section 3.3.2, it is necessary to use internal memory for larger code space and storing small data, so larger memory is more suitable for implementation. ESP32 has larger memory availability than the Arduino Nano33.
- In terms of I/O pins, for implementing the user interface with several button controls along with the recording, storage and sensing module, the available number of pins in the microcontroller should be sufficient for implementation. Even in this requirement aspect, ESP32 has more pins than Nano 33.
- A wireless system needs to have minimal energy consumption since all wireless devices run on a battery section 3.3.3. The implementation should be efficient to have an extended stand by time while delivering sufficient bandwidth at the same time. Comparing both devices shows them to be neck to neck except for WiFi energy consumption where more energy is used by ESP32.



(a) Nano 33 IoT [59] (b) WeMos LOLIN32 lite [60]

Figure 3.6: Microcontrollers

Both the devices have the needed hardware interfaces to connect with SD card reader, sensor and microphone. The shape and size look similar for both devices Figure 3.6. After looking at the critical requirements for the implementation, section 3.2.3, ESP32 is the better choice, as it has more I/O pins, see Figure 3.7, which will be needed for setting up a suitable and flexible UIC.

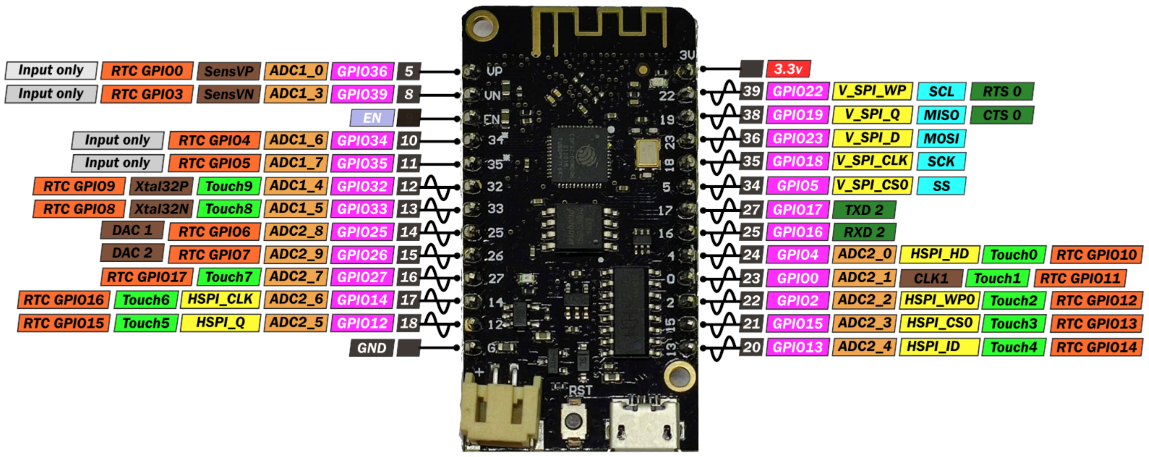

Input only RTC GPIO0 SensVP ADC1_0 GPIO36 5 UP
Input only RTC GPIO3 SensVN ADC1_3 GPIO39 8 UN
EN
Input only RTC GPIO4 ADC1_6 GPIO34 10 34
Input only RTC GPIO5 ADC1_7 GPIO35 11 35
RTC GPIO9 Xtal32P Touch9 ADC1_4 GPIO32 12 32
RTC GPIO8 Xtal32N Touch8 ADC1_5 GPIO33 13 33
DAC 1 RTC GPIO6 ADC2_8 GPIO25 14 25
DAC 2 RTC GPIO7 ADC2_9 GPIO26 15 26
RTC GPIO17 Touch7 ADC2_7 GPIO27 16 27
RTC GPIO16 Touch6 HSPI_CLK ADC2_6 GPIO14 17 14
RTC GPIO15 Touch5 HSPI_Q ADC2_5 GPIO12 18 12
GND
3U 3.3v
22 39 GPIO22 V_SPI_WP SCL RTS 0
19 38 GPIO19 V_SPI_Q MISO CTS 0
23 36 GPIO23 V_SPI_D MOSI
18 35 GPIO18 V_SPI_CLK SCK
5 34 GPIO5 V_SPI_CS0 SS
17 27 GPIO17 TXD 2
16 25 GPIO16 RXD 2
4 24 GPIO4 ADC2_0 HSPI_HD Touch0 RTC GPIO10
0 23 GPIO0 ADC2_1 CLK1 Touch1 RTC GPIO11
2 22 GPIO2 ADC2_2 HSPI_WP0 Touch2 RTC GPIO12
15 21 GPIO15 ADC2_3 HSPI_CS0 Touch3 RTC GPIO13
13 20 GPIO13 ADC2_4 HSPI_ID Touch4 RTC GPIO14
RST

Figure 3.7: WeMos LOLIN32 Lite pinout [61]

Battery

It is essential to use a battery to make the device wireless. There is a wide range of available microcontroller batteries. Batteries are either rechargeable or non-rechargeable. Using a rechargeable battery will ensure the device can be used frequently and is safer for the environment. So, using a rechargeable battery would be a better option than non-rechargeable. There are different batteries available for ESP32 microcontrollers. LiFePO4, Li-ion, Li-Po and 9 V block battery are the most frequently used batteries.

The selected battery should be sufficient to power the prototype for an hour or so for an uninterrupted usage. For this purpose, the Lithium Polymer (Li-Po) battery was selected because of its size and availability. The Li-Po's are also available with different battery capacities with a maximum of 10 000 mA h.

For the ESP32 to operate it requires an operating voltage of 3.3 V and ∼200 mA current for wireless activities. Meanwhile, the voltage output of the Li-Po battery is in the range of 3.7 V and 4.2 V, with current capacity ranging from 500 mA h to 10 000 mA h. The battery selected for this prototype has a capacity of 500 mA h, i.e it can supply power to the microcontroller for maximum of 2 h.

Since the operating voltage of ESP32 and the battery supply voltage are different it is necessary to lower the supply voltage. For this purpose ESP32 has a low dropout regulator (LDO). It is voltage regulator to which the battery is connected that regulates the output voltage of 3.3 V from the higher input voltage (Li-Po battery, 3.7 V - 4.2 V).

3.3.4 Soundscaper

As discussed in section 3.2.3, the audio synthesis software creates musical prototypes, synthesisers, audio mixes, etc. These audio synthesisers use the processed input data from the controller to manipulate the recorded sound using body movements data. At this part of the system, the software application aims to achieve the main objective of the prototype.

Audio synthesis is possible using a wide range of available hardware and software configurations. Two visual programming languages are chosen from the whole range as the literature survey they used - Max/MSP and Pure Data (Pd). These two software environments are created to provide a medium for interactive computing. The same person created both initially, but Max/MSP taken over by Cycling '74 is paid, and Pd is free. Both have their pros and cons which can be seen in Table 3.3.

Max/MSP	Pure Data
Has a vast community and available libraries with proper documentation.	Requires more research from the user with a lack of documentation, and it is a growing community.
Is more accessible for beginners to begin and has a nicer look and is user friendly	Is for beginners too but takes time to learn, and it is not user friendly
Max/MSP is licenced software and paid to use	Is open source, free to use and can be incorporated into applications at no cost.
Is available only in Windows and Mac OS.	Is available in Linux, Windows, Pi and Mac.
Has constant stable updates	Has no frequent updates (rare)
Is suitable for embedded boards or mobile OS.	Can be used in mobile applications using a library and works well with Raspberry Pi boards.

Table 3.3: Comparison between Max/MSP and Pure Data

Max/MSP is easy to use with all the available documentation and examples, but the licence must be paid. PD is vice versa of Max/MSP in these aspects. Max/MSP and Pd allow the user to install externals (developed by users) for development purposes, but Max/MSP has a larger number of contributions to creating externals. Developers using Pd have been increased to create an interactive environment but Max/MSP has a better support forum that is responsive. Max/MSP is chosen for this implementation as it is suitable for beginners to start with, has a 30-day free trial version, and also has plenty of documentation to rely on.

3.4 Setting up the prototype

This section will discuss setting up the interfaces between microcontroller and other peripherals. Each functionality was divided into different subsections. From the section requirements section 3.2, all essential functionalities were collected and from the hardware section all the required hardware components were chosen. Now, how they are connected to one another and how they are used in the development of the prototype, is discussed.

The essential functionalities that were gathered based on the requirements were implemented in the order of the functional flow of the prototype:

1. recording,
2. save recording to storage,
3. upload stored recording via FTP,
4. download stored recordings to PC via FTP,

5. wireless communication of UIC between microcontroller and Max/MSP,
6. wireless communication of sensor data between sensor and Max/MSP,
7. setting up Max/MSP.

3.4.1 Recording

As mentioned in the subsection 3.3.2, the INMP441- MEMS microphone is selected for recording. The INMP441 uses I2S communication. As a first step, all the microcontroller's pins are connected to the INMP441 and should be defined. The pins, as seen in Figure 3.8 - SCK, SD and WS are the signals used for sending audio data. SCK is a serial clock, SD is serial data and WS - 'Word Strobe' is set to 0 or 1 indicating whether the signal is from the left or right channel respectively. These pins are connected to ESP32 pins `2, 13, 15` respectively. There are libraries available for implementing I2S drivers for ESP32, making the recording process simpler.

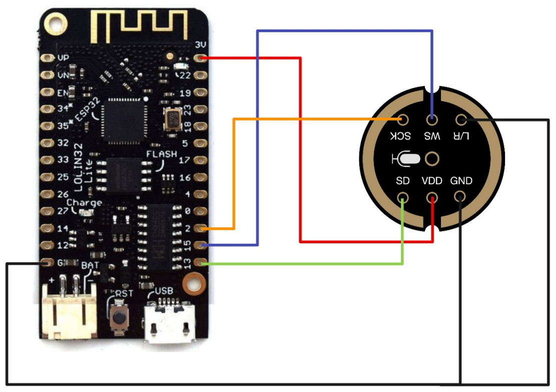


Figure 3.8: ESP32 pin connection with INMP441

A few essential parameters need to be defined for recording audio. The following decisions were taken to start a recording: sampling rate, bits per sample, recording time and file storage format. As per section 3.2.3 for a good quality recording, a sampling frequency of 44.1 kHz with 16 bits per sample (CD quality) should be used. The most used file formats for storing music are MP3 and WAV. Between the two, WAV has better quality when compared to MP3 because MP3 looses quality when the files are lossly compressed and stored as MP3. Selecting the file format comes with a trade-off where the uncompressed WAV format will occupy larger storage space than the MP3 format. The priority is given to the quality of the recording, and the sound is recorded with a sampling rate of 44.1 kHz and 16 bits per sample value and stored in WAV file format.

All WAV files will have a header at the beginning that provides specifications on the file type, sample rate, sample size and overall length. A WAV header is utilised hard-coded when a file is saved to the SD card. The first 44 bytes of the WAV file are allocated for the header of the WAV file. Table 3.4 shows the WAV header format.

Name	offset	Size	Value
ChunkID	0	4	“`RIFF`”
ChunkSize	4	4	405040
Format	8	4	“`WAVE`”
Subchunk1 ID	12	4	“`fmt`”
Subchunk1 Size	16	4	16
Audio Format	20	2	1
Num Channels	22	2	2
Sample Rate	24	4	44100
Byte Rate	28	4	88200
Block Align	32	2	4
Bits per Sample	34	2	16
Subchunk2 ID	36	4	“`data`”
Subchunk 2 Size	40	4	405004

Table 3.4: The header of a WAV file

To initiate the process of recording, a series of steps has to be followed as per [62]. The I2S driver must be installed with the configuration setup required for recording. As a second step, GPIO pins are defined for I2S communication, which helps the I2S driver to route the clock, WS and data signals. The recording is a process of reading and saving the data from the digital microphone. The data from the microphone is stored in a direct memory access (DMA) buffer. To retrieve the data from the direct memory allocation buffer (internal to I2S driver) and to store it in the flash memory (buffer), the “read” function is used. It is necessary to uninstall the I2S driver as a final step to free the resource allocation once the recording process is completed.

3.4.2 Save recording to storage device

A micro SD card is the choice of storage element from section 3.3.2. The selected micro SD card reader uses an SPI interface to connect with the microcontroller. SPI pins are defined at the beginning of the code, and the physical connections are made between the microcontroller and the SD card reader, Figure 3.9. The different functionalities for the SD card can be availed through the SD card library. The SD card module is initialized using the "`begin`" command at the `SD_CS` pin (chip select), which should return ”1” when the initialization is successful. Initialization must be done even before the recording process starts. Once the recording process starts data stored in the buffer (known from subsection 3.4.1) is written into the created WAV file. The data from the buffer is written to the file until the user stops the recording or reaches the maximum time, after which the file is closed and saved into the SD card.

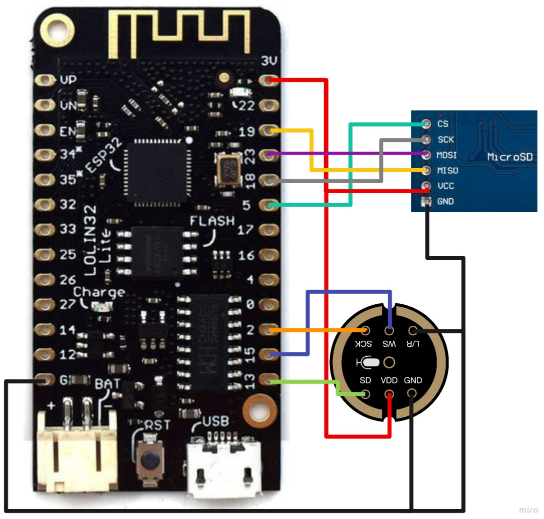


Figure 3.9: ESP32 pin connection with INMP441 and micro SD card reader

3.4.3 Upload file from MCU to PC

MCU to FTP server

Even though the recording is saved in the SD card, wireless data transfer needs to be implemented to make it available on the PC without human intervention. The most used protocol for file transfer is FTP, and there are also other methods available like HTTP post, database transfer, etc.

As ESP32 contains WiFi components, it supports the use of FTP. FTP allows users to access, manipulate and transfer data between computers. Users can connect through FTP to a server securely with a password. They can either set the MCU as a server and access it remotely from the client device or vice versa, and each has its pros and cons.

When an ESP32 is used as a “server”, the files in the SD card can be accessed e.g. by using an FTP application such as Filezilla, or by using Python scripts. The FileZilla client does not support any automation [63], and thus a Python script was implemented to automate the process. Despite the successful data transfer using this Python script, the session kept expiring when the system was idle. Due to this issue with the sessions, using ESP32 as a “server” does not suit the requirements.

Meanwhile, using ESP32 as a “client” to be connected with an FTP server when required proved reasonable. The FTP server in the system/PC can be set up using Python libraries, Filezilla, or a website with an FTP server. Providing a backup for the recording and allowing the user to use it anytime by making it available online was also a choice. Hence a website that will be available all the time with an option of uploading files to the website via FTP is chosen to be a part of the implementation.

The ESP32 uses an FTP client library, making the FTP connection and file transfer easier. Once the connection between the server and the client is set up, the file to be transferred is read from the SD card and stored in a buffer. The data stored in the buffer is written to the FTP server present in the free hosted website, and the connection is closed. However, the drawback is that the recorded data will not be directly available on the PC.

FTP server to PC

An automation script was developed using Python to make the recorded file available to Max/MSP on the PC’s local folder. This script checks for the latest file uploaded to the FTP server and compares it with the previously downloaded file’s date and time. If the date and/or time vary, it downloads the latest files from the FTP server to make them available for Max/MSP.

3.4.4 Wireless communication of UIC

The prototype’s user interface controls the functionalities set up at the microcontroller using buttons. The set of functionalities that are required to control the prototype is

- Start recording - Start button initiates the recording process
- Stop recording - Stop button to stop the recording and initiate file transfer
- Mute/unmute a track and delete a track - Track button to control the track (multiple functions).

The mute and unmute control data is the only control data that has to be sent to Max/MSP from the microcontroller. Since the prototype is wireless, the control data must be sent to Max/MSP via Bluetooth or other network protocols. Max/MSP does not support Bluetooth, so the only available option is data transmission through available wireless network protocols over WiFi.

Max/MSP supports Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) communication protocols for transmitting data. Both of the communication protocols use IP to send and receive data. To send data via UDP, only the IP address and destination must be specified. However, it does not give feedback on packet status, like whether the data reached the destination or failed due to network issues. On the other hand, TCP sends an acknowledgement signal to the sender on receiving. Unlike UDP, it requires a connection between sender and receiver before sending data. Both UDP and TCP transfer data as packets. UDP packets contain a destination IP-address and the data in a packet, while TCP includes extra information to track packets. The significant difference between these two protocols is that TCP is reliable, but UDP is fast. UDP is simpler than TCP in implementation, and missing some data from an entire data stream will not disastrously affect the prototype's function. Hence, UDP is implemented for transmitting data.

ESP32 uses a ”`WiFiUDP`” library for setting up UDP connections by providing the IP address and the destination port. For UDP to work, both sender and receiver should be connected to the WiFi network. A UDP connection is opened with a destination port when a track button is pressed. Using the function ”`beginPacket`”, the IP address, port and data are added to a packet, and by using ”`endPacket`”, the packet is closed and sent to the destination.

Max/MSP has a function ”`udpReceiver`” to receive messages from UDP by connecting to the same local IP address and port as the sender. These values are then processed as per the requirements

3.4.5 Wireless connection between sensors and Max/MSP

Xsens Dot has a user interface (UI) for mobile and desktop applications, allowing users to connect to the sensors. This UI provides multiple measurement modes under ”`Start Logging`”, allowing users to log the acceleration, Euler angle data together, or the raw sensor data from the accelerometer and gyroscope. The UI also provides an option for syncing the sensor devices for synchronised output.

However, the data from the sensor are stored by default in an excel file instead of being transferred to Max/MSP. The prototype aims to transmit the sensor data to Max/MSP, and thus for the Max/MSP to read the data, changes in the existing algorithm of the Xsens Dot application are required.

The Xsens Dot desktop application for the sensor was developed using JavaScript, and the code is accessible for the users for development as per the requirements. The data should be wirelessly transmitted to the Max/MSP as per the needs. From subsection 3.4.4, UDP transfer would be a perfect fit for transferring the data between the sensor and Max/MSP .

UDP communication between the sensor and Max/MSP is achieved using the Node js library "`dgram`". This library allows to "create a socket" for sending UDP packets. Xsens Dot algorithm when set in "`measurement`" mode, an excel file with the device id, timestamp and timestamp is created. Creating a port for each sensor to transmit the respective sensor data to Max/MSP is possible with the stored device id. Once the data is entered in an excel sheet based on the device id a socket is created with a local IP address and port. Each device id is bound with individual ports, when data arrives at a port a UDP connection is established to transfer the data. This UI server is set up at a desktop local server, and the UDP socket binds to the local IP address.

Max/MSP uses the function "`udpReceiver`" to receive messages as mentioned in subsection 3.4.4.

3.4.6 Setting up Max/MSP to receive data

Max/MSP is a programming platform for music and multimedia, utilised for a plethora of musical prototyping, as stated in section 3.2.3. For the development, Max/MSP is programmed to receive sensor values from the sensor and read the recorded audio from the ESP32. Max/MSP is the end stage of the project flow.

The recorded WAV file that is available on the PC is read using Max/MSP's "`read`" function. The recording will be saved in the buffer and used for audio manipulation Figure 3.10.

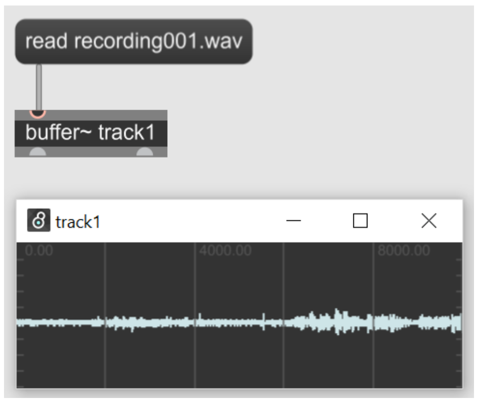


Figure 3.10: Max/MSP patch to read a recording

Musical data is essential at this point of the project where the whole manipulation of audio depends on it. The usual way of transferring musical data in real-time is through MIDI (Musical Instrument Digital Interface) which communicates pitch, velocity, and notes when pressed. The idea of the prototype is to use recorded music rather than digital instruments, so MIDI was not preferred. However, the pitch, tempo, loudness, etc., need to be changed to control or manipulate the recorded music. Max/MSPreceives the sensor values based on the body movements through UDP(subsection 3.4.5). Those values are used to manipulate the music's pitch, volume and tempo. As per the subsection 3.4.5, Max/MSP uses "`udpReceiver`" to receive UDP data, and the data is received in ASCII format. But for the prototype the original sensor values are required to manipulate the sound properties. Therefore the "`itoa`" function of Max/MSP is used to convert ASCII values to characters and are separated into device id, timestamps, angles (x,y,z), acceleration (x,y,z) Figure 3.11.

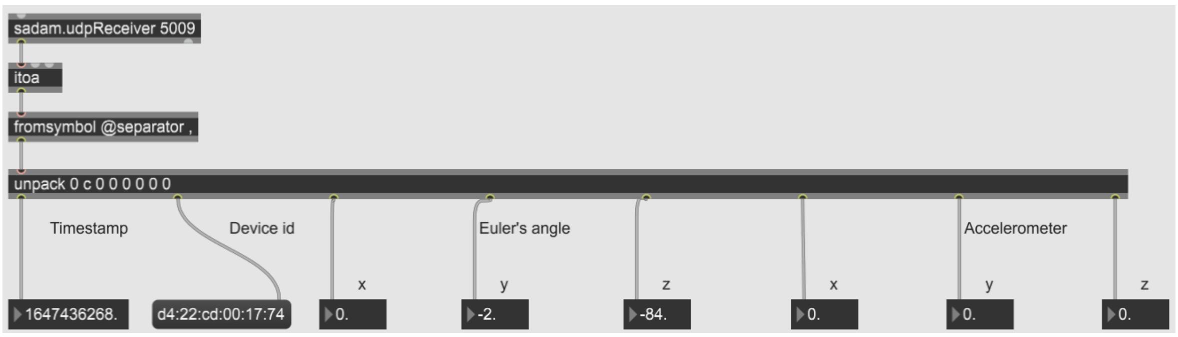


Figure 3.11: Max/MSP patch to read sensor data

3.5 System architecture - Revision

The system architecture seen in Figure 3.1 is changed according to the selection of hardware and communication protocols. There are significant changes from the existing architecture, such as

- the sensor module communicates directly with the soundscaper module (Max/MSP) instead of the microcontroller.
- the recorder module also has a storage device connected to the microcontroller.
- the microcontroller has a battery connected to it.

The updated architecture with the above mentioned changes is as shown in Figure 3.12.

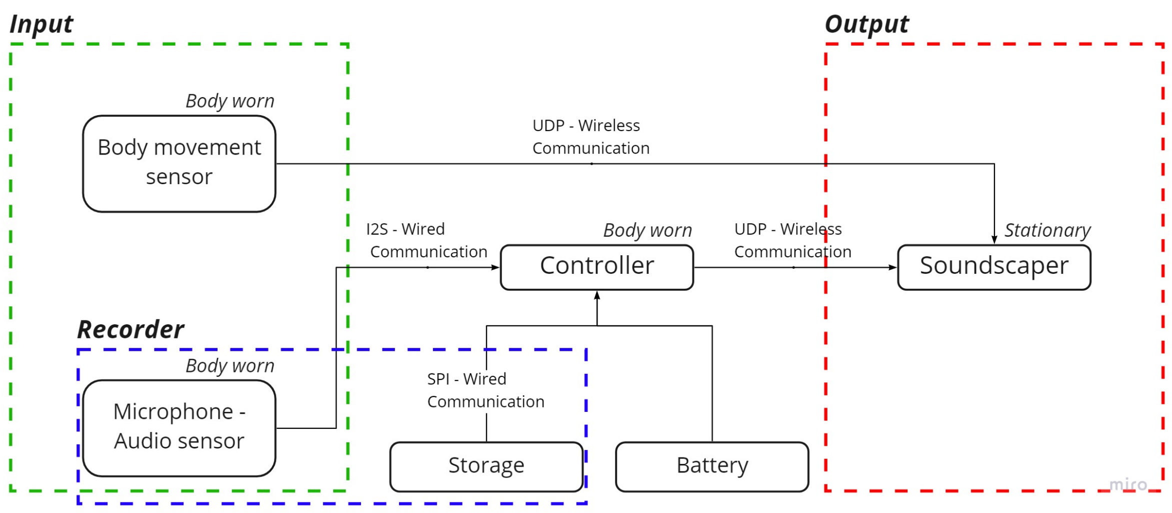
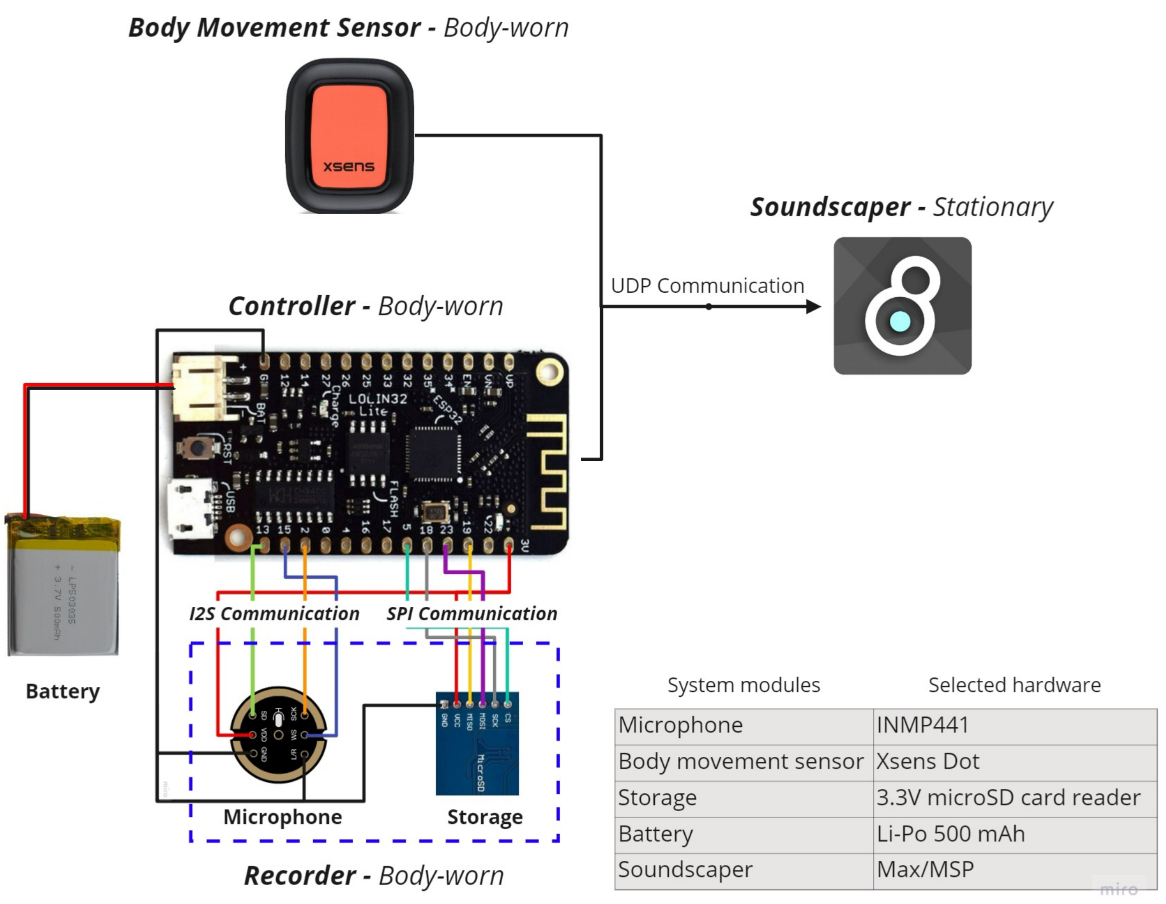

Input
Output
Body worn
Body movement sensor
UDP - Wireless Communication
I2S - Wired Communication
Body worn
Controller
UDP - Wireless Communication
Stationary
Soundscaper
Recorder
Body worn
Microphone - Audio sensor
SPI - Wired Communication
Storage
Battery

Figure 3.12: Updated system architecture module

For a better visualization, the system architecture modules are replaced by the chosen components Figure 3.13



System modules	Selected hardware
Microphone	INMP441
Body movement sensor	Xsens Dot
Storage	3.3V microSD card reader
Battery	Li-Po 500 mAh
Soundscaper	Max/MSP

Figure 3.13: Updated system architecture with selected hardware

3.6 Conclusion

The steps involved in the development process of the prototype were discussed in this chapter. The process started with figuring out the functional and system requirements, which formed the basis of the prototype development. A high-level design for the end product's system architecture was defined using five modules: the body movement sensor, recorder, controller, communication, and soundscaper. Each module is analysed individually to determine the hardware feature requirements of the prototype. The hardware selection was made by comparing the suitable shortlisted devices to achieve the system goals. The major requirements for the system were to be wireless and low powered, so there was a need for suitable hardware selections that supported wireless communication and low power consumption. The comparison resulted in hardware choices Xsens Dot as the Body movement sensor, INMP441 digital microphone and 3.3V microSD card reader, and the recorder to store the recordings, Wemos Lolin32 lite (ESP32) as controller and Max/MSP as a soundscaper to manipulate audio signals.

All hardware choices were also based on compatibility and to communicate the data to the Max/MSP. The communication between hardware begins from the recording module. The microphone is connected to the microcontroller using I2S communication, and the micro SD card reader is connected to the microcontroller through the SPI communication protocol. This communication helps the prototype to gather the audio signals using a microphone and store them in the micro SD card in a WAV format. The recording is then sent to the PC's local folder via FTP for the Max/MSP to read the recording to manipulate the audio. Parallel to recording, a UDP connection was implemented in the sensor to send sensor data to the Max/MSP. A Max/MSP patch was implemented to test whether Max/MSP reads the sensor data and the recording file. The test result shows that the hardware selected and the prototype setup works as expected. This chapter answers the research question, "What could the architecture and implementation for a prototype of the proposed system look like" and "How to measure the body movements (e.g. hands, legs or wrists) for sound manipulation". In the next chapter, the functional flow of the prototype will be discussed along with the use case.

Chapter 4

Product flow and Use case

This chapter discusses the flow of the prototype and the manipulation of audio signals based on different body movements. Now that a functional prototype is developed, it is necessary to discuss the flow of the prototype, including the use case of the prototype.

4.1 Prototype design

chapter 3 discussed hardware and communication choices, and this section is about the additional functional design choices for the hardware to work as expected. Few additional functional design choices have to be considered while programming, like

- how many tracks the user should be allowed to record.
- how to notify the system is busy.
- how to communicate with the fellow user if they like the recording.
- how the user can take control of the recording (mute, unmute, delete).

A minor hardware design change needs to be done to implement the new, more detailed, design choices. The following changes to the system will be sufficient for the prototype to incorporate these changes.

1. The number of recordings a user can record should not be limited to one or two, which would restrict the user to explore the soundscape or make sense with fellow users when having to few tracks. So the design choice of allowing the user to record six tracks was decided based on availability of I/O pins and complexity of the prototype. Having more than six recordings would be complicated and will distract the user from the main goal of participatory sense-making.
2. Since it was decided to use six recordings, it was necessary to allow the user to control the six tracks. Having a button assigned to each track would be

the less distracting option. The button has multiple functionality that should communicate with the Max/MSP and with the SD card. Functionalities of the button are to mute, unmute and delete the track.

3. An LED light was used to notify the user that the system is busy recording or transferring the recorded data.

4. One more button was added to the prototype to send information between the users, which will notify the other user whether the recording is likeable or comfortable.

Figure 4.1 is the updated connection of the prototype. The additional design choices were implemented so they did not affect the existing connections of the microcontroller.

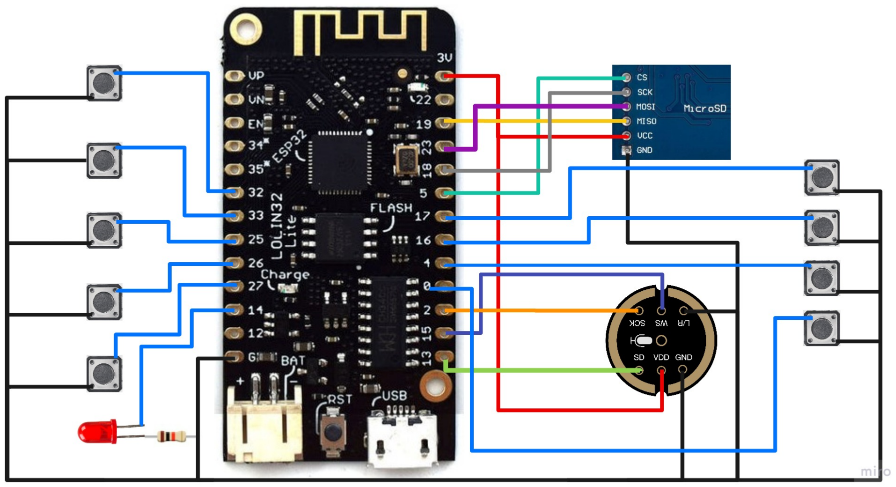


Figure 4.1: Revised connections with the addition of buttons

4.2 Product flow

A flowchart is nothing but a visual representation of the process or the system. Flowcharts are handy to represent the code more straightforwardly and easier to be understood. Flowcharts consist of a few standard shapes that represent steps, and it is essential to know about the shapes and their representation, see Figure 4.2.

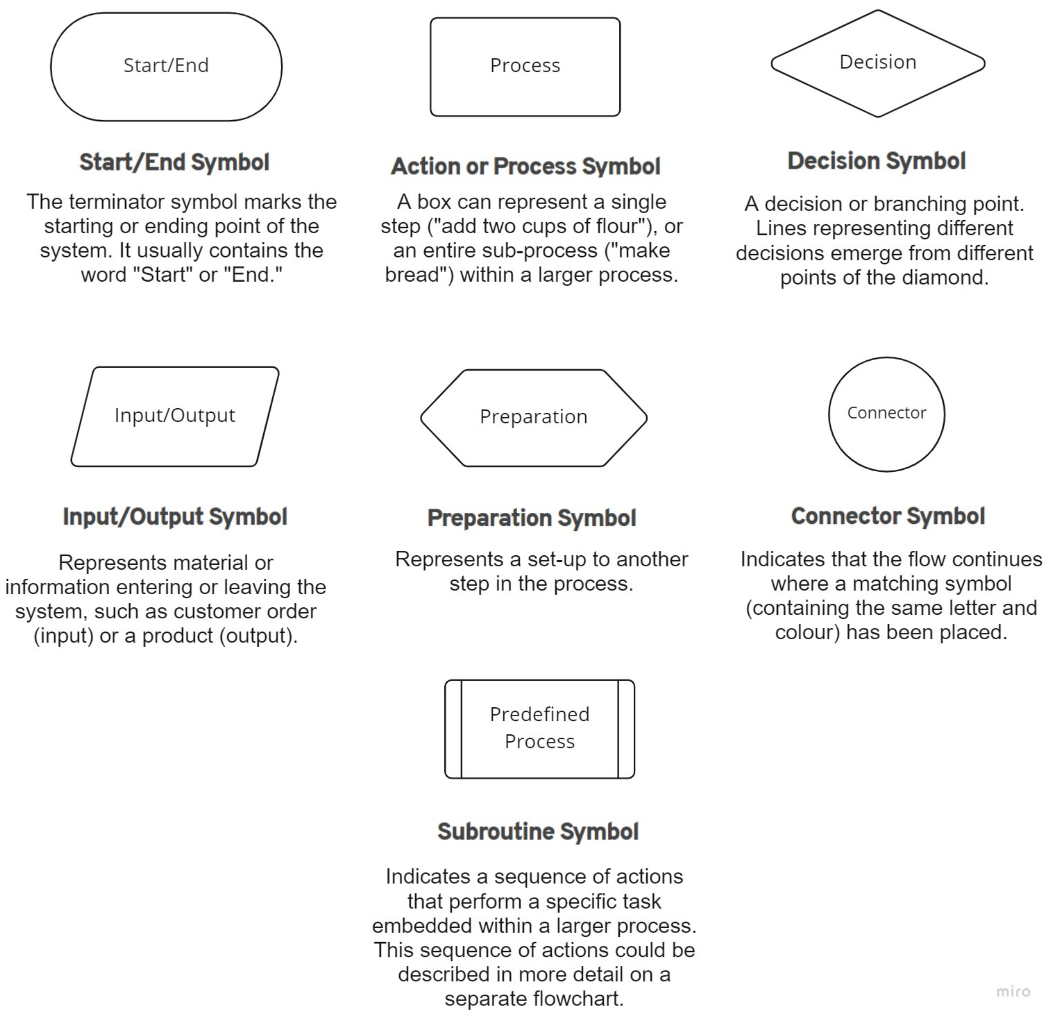

Start/End
Start/End Symbol
The terminator symbol marks the starting or ending point of the system. It usually contains the word "Start" or "End."
Process
Action or Process Symbol
A box can represent a single step ("add two cups of flour"), or an entire sub-process ("make bread") within a larger process.
Decision
Decision Symbol
A decision or branching point. Lines representing different decisions emerge from different points of the diamond.
Input/Output
Input/Output Symbol
Represents material or information entering or leaving the system, such as customer order (input) or a product (output).
Preparation
Preparation Symbol
Represents a set-up to another step in the process.
Connector
Connector Symbol
Indicates that the flow continues where a matching symbol (containing the same letter and colour) has been placed.
Predefined Process
Subroutine Symbol
Indicates a sequence of actions that perform a specific task embedded within a larger process. This sequence of actions could be described in more detail on a separate flowchart.
miro

Figure 4.2: Flowchart shapes and their representations

The flow of Stim4Sound can be represented using multiple processes that are put together to form the whole system. Based on the functional flow of the prototype in section 3.4 flowcharts are formulated for each function and are explained in the upcoming subsections.

Before diving into each function, a high-level system flow is given in this section. There are two major processes running in parallel, one process pertains to functionalities of the microcontroller and another is the process of sending sensor data to the Max/MSP.

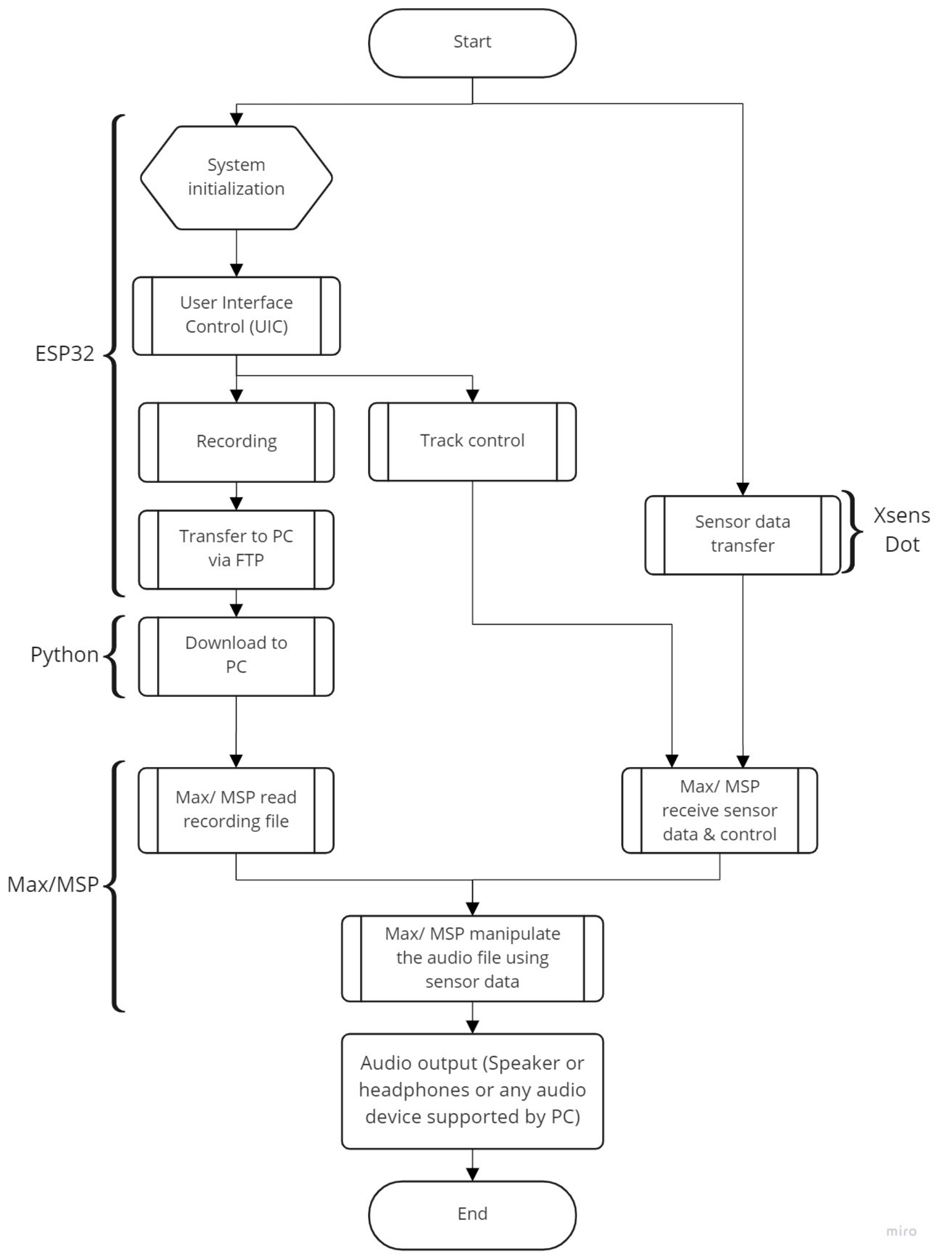

Start
System initialization
User Interface Control (UIC)
Recording
Track control
Transfer to PC via FTP
Sensor data transfer
Xsens Dot
ESP32
Python
Download to PC
Max/ MSP read recording file
Max/ MSP receive sensor data & control
Max/MSP
Max/ MSP manipulate the audio file using sensor data
Audio output (Speaker or headphones or any audio device supported by PC)
End
miro

Figure 4.3: High-level flowchart of the system

4.2.1 Recording

The main process of the prototype is to record audio on a button press, and there are prerequisites to start the recording function like initializing the variables. The firmware for Stim4Sound is developed using the Arduino IDE, and the entire flow of the recording program is given as a flowchart in Figure 4.4. The connectors in the flow chart are color coded for better understanding of the connector's beginning and end in a flowchart. Connector flows are explained in the upcoming subsections with separate flowcharts.

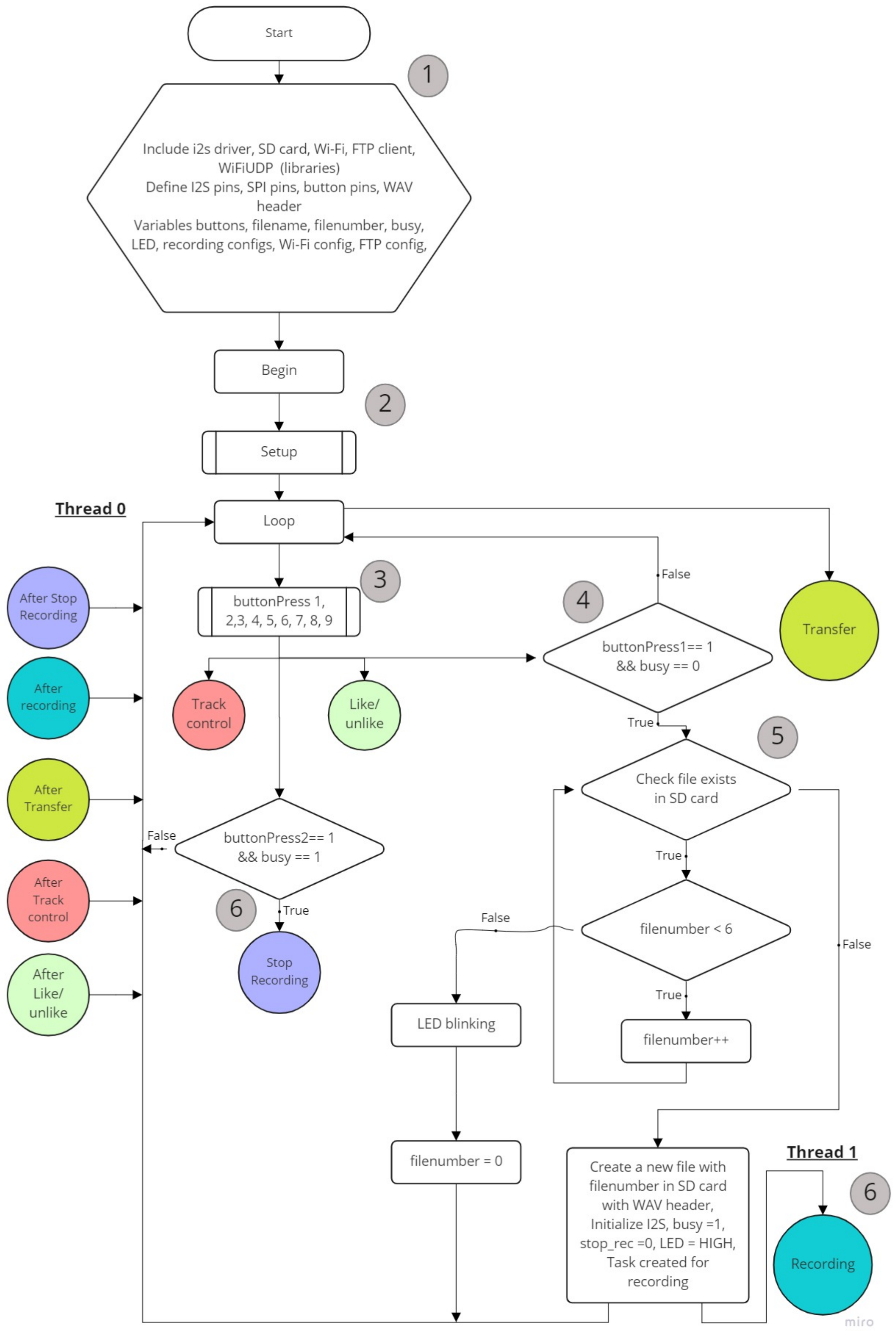

Start
1
Include i2s driver, SD card, Wi-Fi, FTP client, WiFiUDP (libraries)
Define I2S pins, SPI pins, button pins, WAV header
Variables buttons, filename, filenumber, busy, LED, recording configs, Wi-Fi config, FTP config,
Begin
2
Setup
Thread 0
Loop
3
buttonPress 1, 2,3, 4, 5, 6, 7, 8, 9
After Stop Recording
After recording
After Transfer
After Track control
After Like/ unlike
Track control
Like/ unlike
4
buttonPress1== 1 && busy == 0
False
True
Transfer
5
Check file exists in SD card
True
False
filenumber < 6
True
False
filenumber++
LED blinking
filenumber = 0
buttonPress2== 1 && busy == 1
False
True
6
Stop Recording
Create a new file with filenumber in SD card with WAV header, Initialize I2S, busy =1, stop_rec =0, LED = HIGH, Task created for recording
Thread 1
6
Recording
miro

Figure 4.4: The system flow of recording process (1-6 represents to the corresponding point below).

1. Arduino has libraries for Wi-Fi connection, I2S driver, SD card, FTP, UDP, etc.,

that are initialized along with the pins of the peripherals, constants and the variables required for the program.

2. `Setup` is the predefined process where the Wi-Fi connection, the availability of the SD card and the button pins of `button1, button2` are set up to be an input. These are the two buttons used to start and stop the recording process. After the process setup, a loop function starts with a series of conditions to check to initiate various processes (`buttonPress`, `Like/ unlike`, `Track control` and `Transfer`).

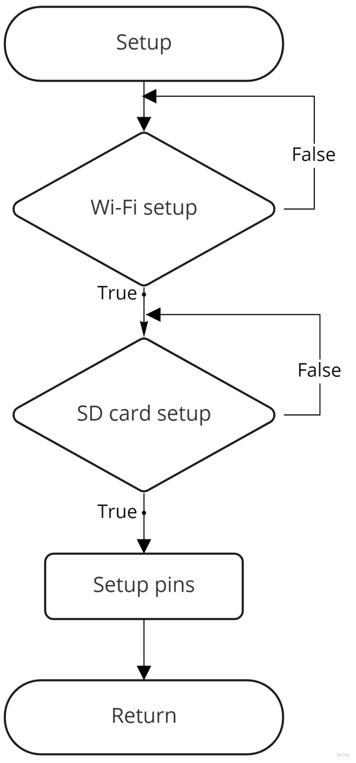


Figure 4.5: The `Setup` process happens once before recording

3. `buttonPress` is a predefined process where it reads `buttonState` continuously, and depending on how long the button is pressed, it updates `buttonPress` values `0, 1, 2`. `0` - no activity, `1` - short press, `2` - long press. `button1` and `button2` are the two buttons required for recording.

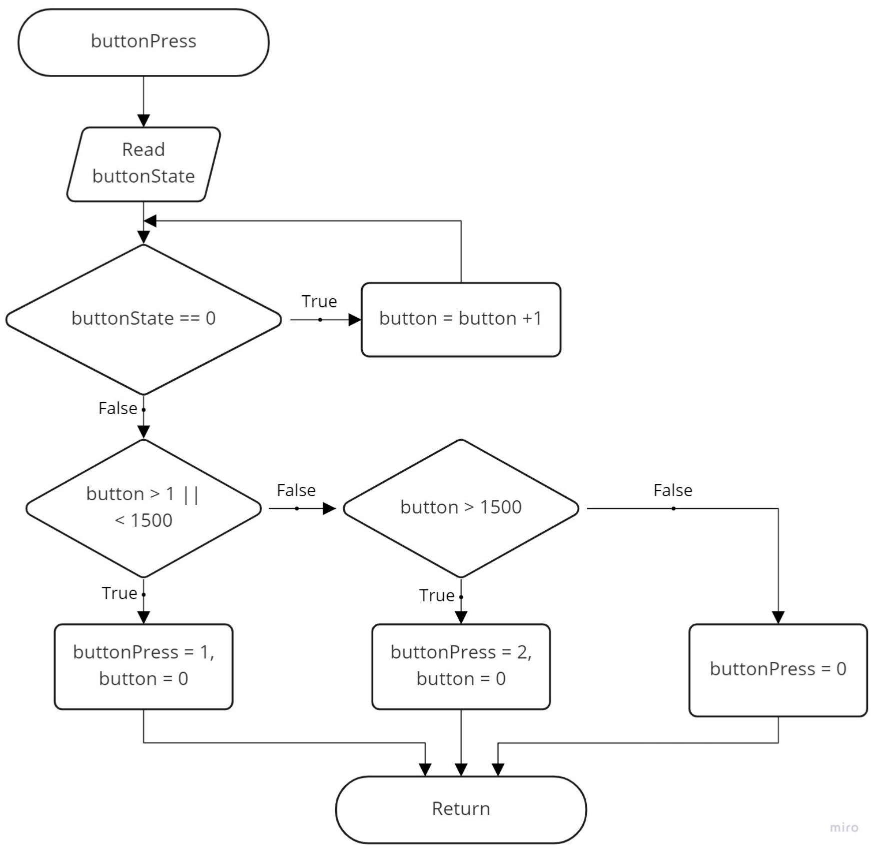


Figure 4.6: The `buttonPress` process of recording

4. Recording process starts only when there is a short press of `button1` (`buttonPress1 = 1`), and the system is free (`busy = 0`), see Figure 4.4.

5. When the condition is true, the program checks whether the file exists on the SD card, see Figure 4.4.

   - If the file does not exist, a set of initialization and preparation is done for the recording. A new WAV format file "recording000.wav" (or as per the `filename`) is created with a WAV header. The variables `busy` and `stop_rec` are updated to `1` and `0`, respectively, indicating the system has taken up a task and is waiting for it to be stopped. The `LED` is turned `HIGH`, indicating to the user that the process has started. A task is created for the recording process to run in `core1` (`Thread1`) such that it will not affect the button check that runs in `core0` (`Thread0`).

   - If the file already exists on the SD card, the `filename` is updated by incrementing the file number but only if the `filenumber` count is less than `6` (total number of recordings are 6 numbered 0 - 5). The code changes the filename from "recording000.wav" to "recording001.wav" and can have maximum up to "recording005.wav".

- If the `filenumber` is greater than or equal to `6` then the user has recorded all six tracks and the program through LED blinking notifies the user.

6. Two processes happen simultaneously at `Thread0` and `Thread1`.

- `Thread0`: the `buttonPress` process continuously checks for the `buttonPress` update to stop the audio acquisition process (`Stop Recording`) or other available processes, which runs parallel to the `Thread1`.

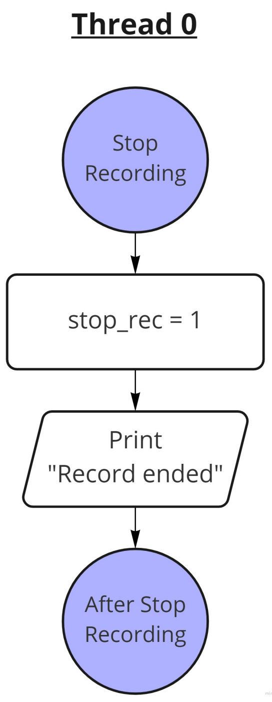


Figure 4.7: The `Thread0` task

- `Thread1`: the process of audio acquisition and storing it, is explained in the next subsection. Once the process ends, then `Thread1` stops and all processes will run in `Thread0` serially.

`Thread1`

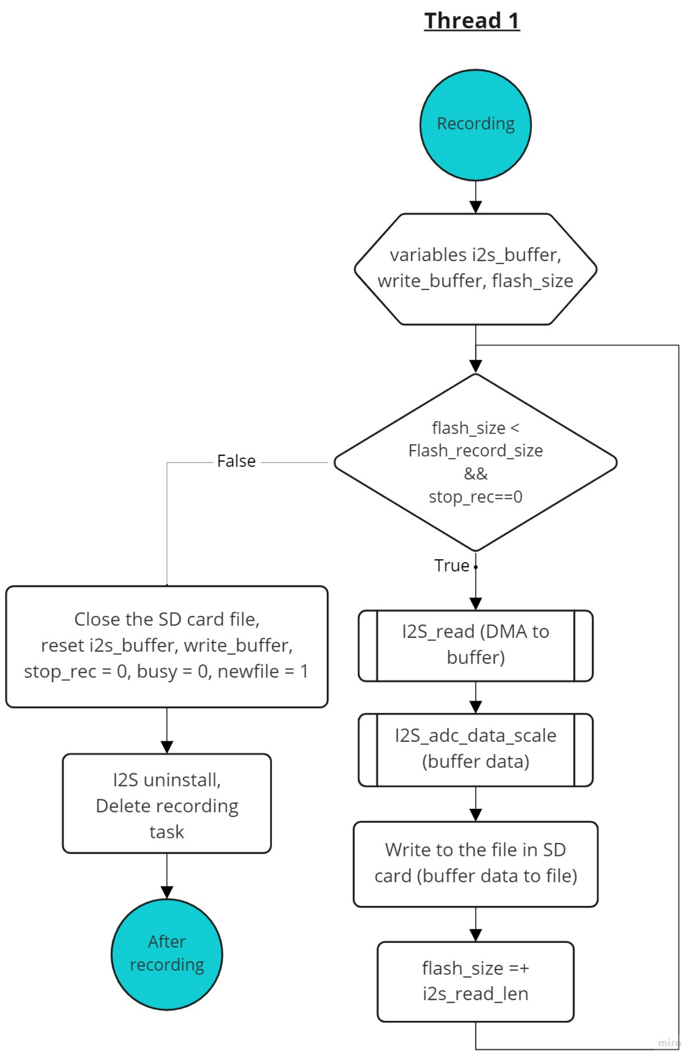

Thread 1
Recording
variables i2s_buffer, write_buffer, flash_size
flash_size < Flash_record_size && stop_rec==0
False
True
Close the SD card file, reset i2s_buffer, write_buffer, stop_rec = 0, busy = 0, newfile = 1
I2S uninstall, Delete recording task
After recording
I2S_read (DMA to buffer)
I2S_adc_data_scale (buffer data)
Write to the file in SD card (buffer data to file)
flash_size =+ i2s_read_len

Figure 4.8: The `Thread1` task

1. `Thread1` intiates recording process by initalizing the two 16 kB size buffers and a counter variable `flash_size`.

2. Acquisition starts if the condition of `flash_size` is less than the size of the recording file 10 s and the control signal `stop_rec` from `Thread0` is `0`.

3. `I2S_read` is a predefined library function which receives audio data from the INMP441 via I2S communication and stores it in the defined 16 kB `i2s_buffer` buffer.

4. `I2S_adc_data_scale` is a library function used to covert the digital data to an analog data before saving it as WAV format.

5. The converted data is written to the WAV file created in the SD card and increments the `flash_size` by 16 once an iteration is completed.

6. Once the `flash_size` is greater than the recording size of an audio file for 10 s or if the `stop_rec` is updated `1` on a button press in `Thread0` then the recording will stop.

7. Once the recording is stopped, the SD card file will be closed, all the buffers will be cleared, the I2S driver is uninstalled, the variables are reset, meanwhile updating `newfile = 1`, and the parallel task `Thread1` is deleted.

4.2.2 Transfer to website via FTP (`Transfer`)

Once the recording process is done with saving the recorded file to the SD card, the program runs so that the transfer process begins in `Thread0`. The recorded file is transferred via FTP to the website, and the following process is involved while transferring the file, see Figure 4.9.

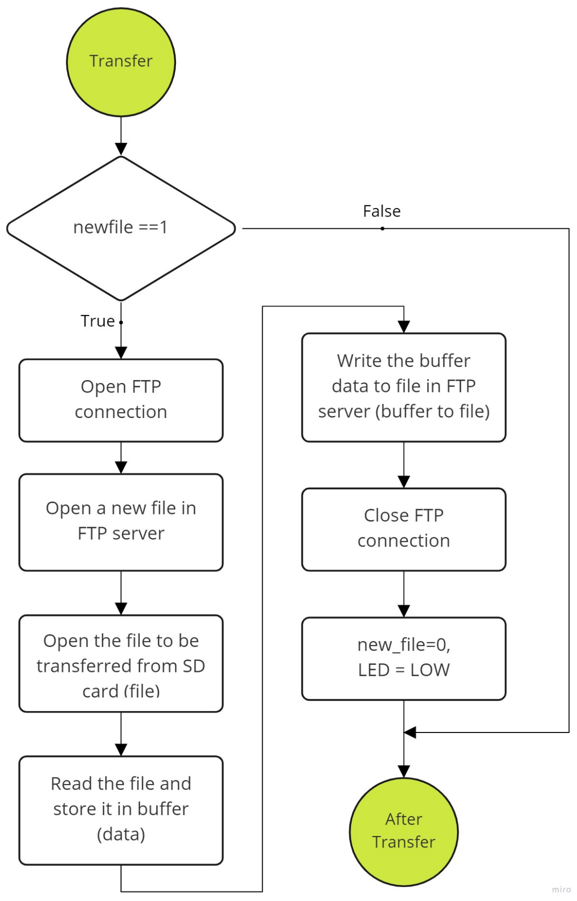

Transfer
newfile ==1
False
True
Open FTP connection
Open a new file in FTP server
Open the file to be transferred from SD card (file)
Read the file and store it in buffer (data)
Write the buffer data to file in FTP server (buffer to file)
Close FTP connection
new_file=0, LED = LOW
After Transfer
miro

Figure 4.9: The `Transfer` task

1. If the `newfile` is updated to `1`, then a new FTP connection is established using an FTP library function.
2. A new file is created in the FTP server with the same name as it is recorded in the SD card.
3. The recorded file to be transferred is read from the SD card and stored in a buffer.
4. Using the FTP library function, the data stored in the buffer is written to the file created
5. Once the data is completely transferred the FTP connection is closed, the variables are reset, and `LED` is turned `LOW`, indicating the recording and transfer process is completed.

The system takes up memory during the transfer so that no other tasks in `core0` were run because it will disrupt the transfer process.

4.2.3 Download to PC

A Python automation script was developed to download the recorded audio from the server via FTP to PC automatically. For this to happen, an empty file was created in the FTP server manually (only for the first use) and named as `time,recording000.wav,31 01 2022 21 42 44` with date, time and recording name separated with a comma. This empty file was used in the program to automate the download. The script checks for the recent update of the file in the server to download the recording.The flow of the program is given in Figure 4.10.

1. The program should initialize the necessary libraries, variables and declarations required for the Python script to get the job done.
2. A new connection is made with the FTP server using login credentials using the FTP library.
3. The available files in the FTP server are listed and stored into a list variable "line" and also an empty list `file_list` is created in Python to separate the files.
4. A loop is implemented to check for the file that starts with "time" to add those it to the empty list `file_list`
5. A loop is implemented to get the timestamp for the file that starts with "record" (uploaded recorded file) and compares it with the time in the `file_list`. If the uploaded file's timestamp is more recent than that of time present in the file

name time, recording000.wav, `31 01 2022 21 42 44`, then the recording file is downloaded.

6. For download, a filename (same as the filename in the SD card) is created in the local folder, and with the help of the FTP library, download is completed.
7. Once the file is downloaded, the empty file which starts with `time` and has the same name as the downloaded file is deleted. Furthermore, a new file is created with the same name except for the date and time, which will be updated to the downloaded time, which will be used for the next update.

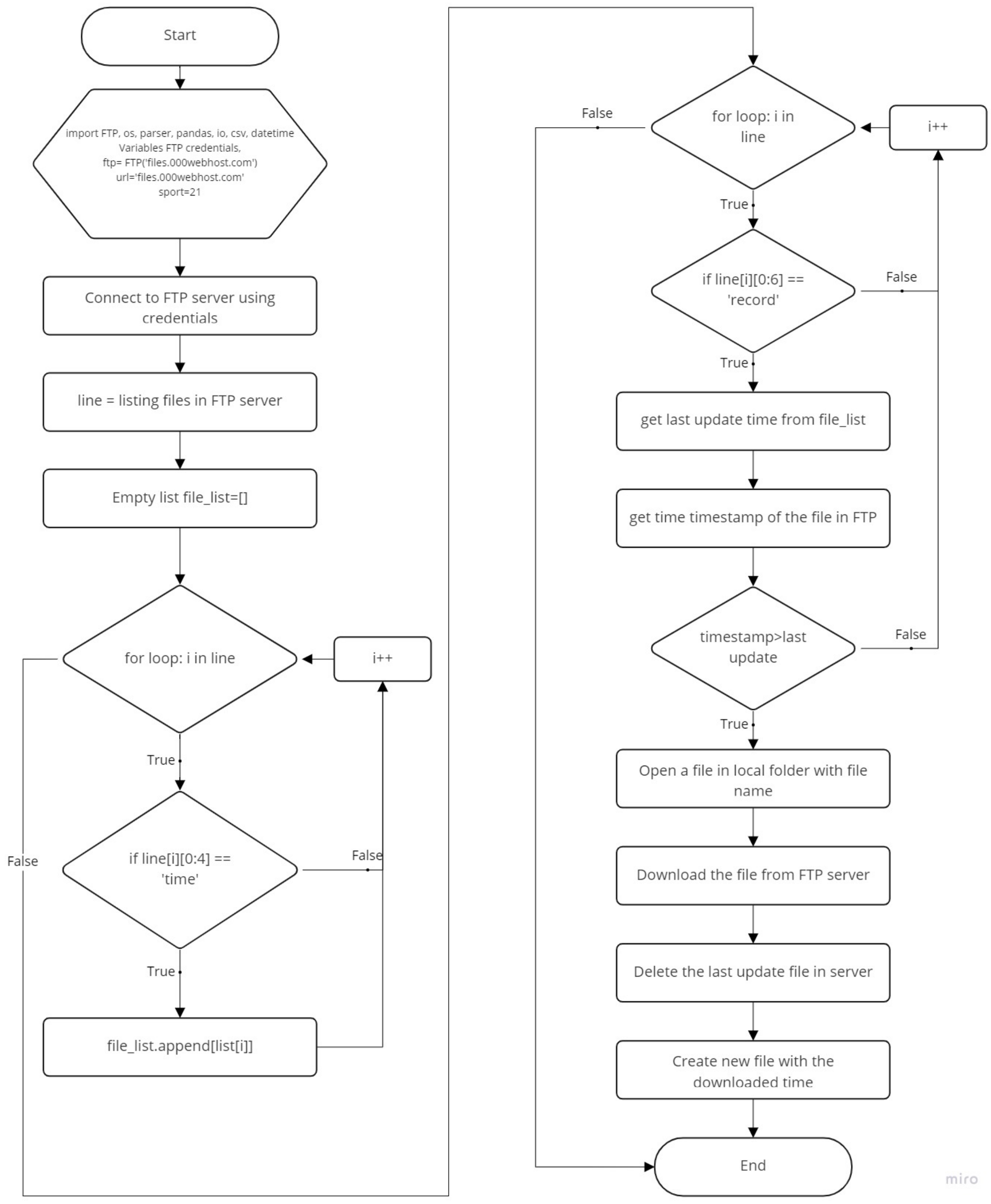

Start
import FTP, os, parser, pandas, io, csv, datetime
Variables FTP credentials,
ftp= FTP('files.000webhost.com')
url='files.000webhost.com'
sport=21
Connect to FTP server using credentials
line = listing files in FTP server
Empty list file_list=[]
for loop: i in line
i++
True
False
if line[i][0:4] == 'time'
False
True
file_list.append[list[i]]
for loop: i in line
i++
False
True
if line[i][0:6] == 'record'
False
True
get last update time from file_list
get time timestamp of the file in FTP
timestamp>last update
False
True
Open a file in local folder with file name
Download the file from FTP server
Delete the last update file in server
Create new file with the downloaded time
End
miro

Figure 4.10: Flowchart of the code for downloading file to the PC

4.2.4 Track Control

Track control is a part of the prototype's UIC, that should communicate with Max/MSP or trigger a process in the microcontroller. A UDP communication library is used in Arduino IDE to establish a UDP transfer between the UIC and Max/MSP. There are 6 buttons used for this purpose and each button was assigned to the recorded audio file. The process flow of the track control using the buttons are given as a flow chart in the Figure 4.11.

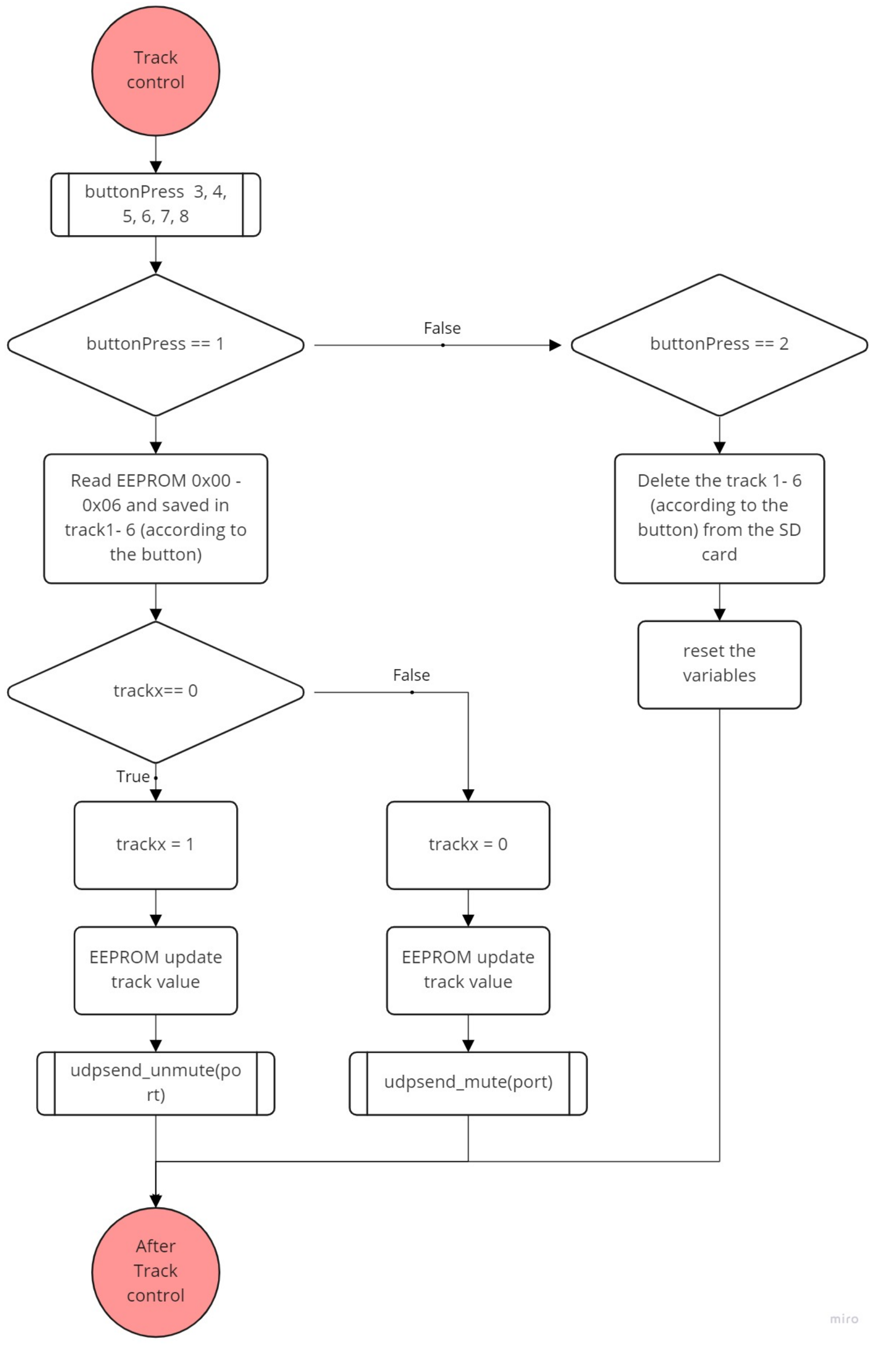

Track control
buttonPress 3, 4, 5, 6, 7, 8
buttonPress == 1
False
buttonPress == 2
Read EEPROM 0x00 - 0x06 and saved in track1- 6 (according to the button)
Delete the track 1- 6 (according to the button) from the SD card
reset the variables
trackx== 0
False
True
trackx = 1
trackx = 0
EEPROM update track value
EEPROM update track value
udpsend_unmute(port)
udpsend_mute(port)
After Track control
miro

Figure 4.11: Flowchart of the code for track control

1. The `buttonPress` works the same way as item 3 of subsection 4.2.1.
2. The condition for the `buttonPress` is checked whether it is a short press - `1` or a long press - `2`.
3. If the condition is true for the short press (`buttonPress = 1`), the data stored in EEPROM during the initialization of the system is read to know whether the track is muted or unmuted. If the track value is 0, it is updated to 1 or from 1 to 0.
   - The updated values are then written into the same EEPROM address. Each track has its EEPROM address to read and update.
   - As per the update from 0 to 1 or 1 to 0, the corresponding UDP function is called to mute or unmute. From 0 to 1 call, `udpsend_unmute` function and from 1 to 0 call, `udpsend_mute` function.
4. If the condition is true for the long press (`buttonPress = 2`), the corresponding track assigned to that button is deleted from the SD card using the SD card library function and variables are reset.

`udpsend_mute` and `udpsend_unmute` functions

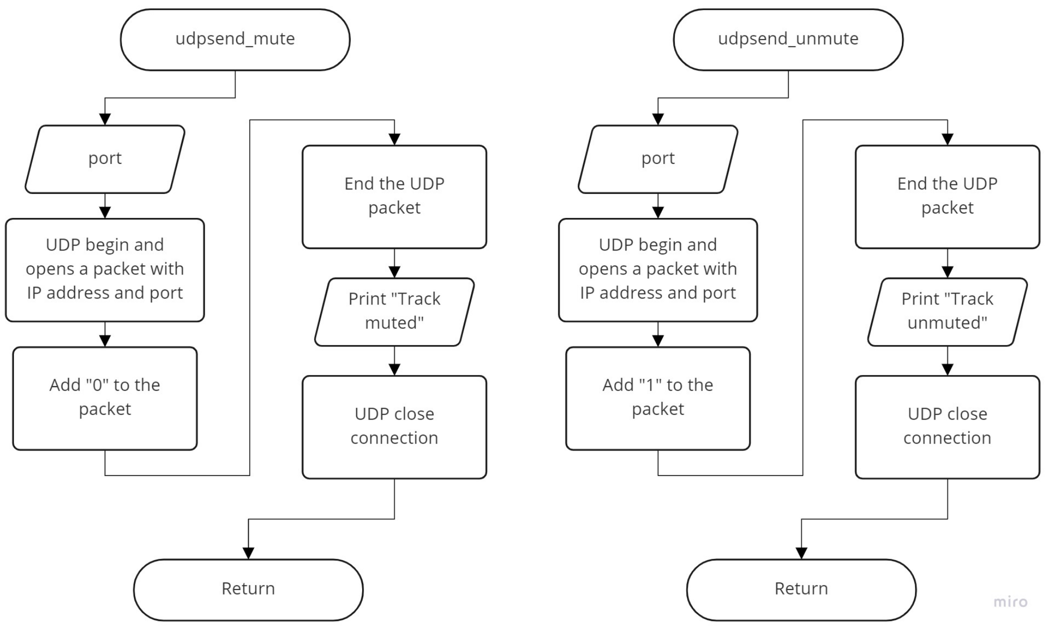


Figure 4.12: Flowchart of the code for muting and unmuting the track

1. The function `udpsend_mute` and `udpsend_unmute` works similarly except for sending values as Figure 4.12.

2. When the button is pressed to mute/ unmute the track, the `udpsend_mute/unmute` function is triggered with the port as an input to the function.

3. A UDP communication port is set up with the port and the broadcasting IP address (255.255.255.255).

4. A UDP message packet is created and value (`0` for mute and `1` for unmute) needs to be sent is written to it.

5. Once the data is sent, a message is printed "Track muted" or "Track unmuted", variables are reset and then the UDP communication is closed.

4.2.5 Like/Unlike

This function is implemented to allow the users to notify each other about their likes and dislikes of the current recording they are playing. This function is also part of UIC, where the control signal is sent to Max/MSP to turn on and off the visual LEDs (green and red). The process flow Figure 4.13 for this function varies slightly from subsection 4.2.4 because a new variant of the very long press is implemented.

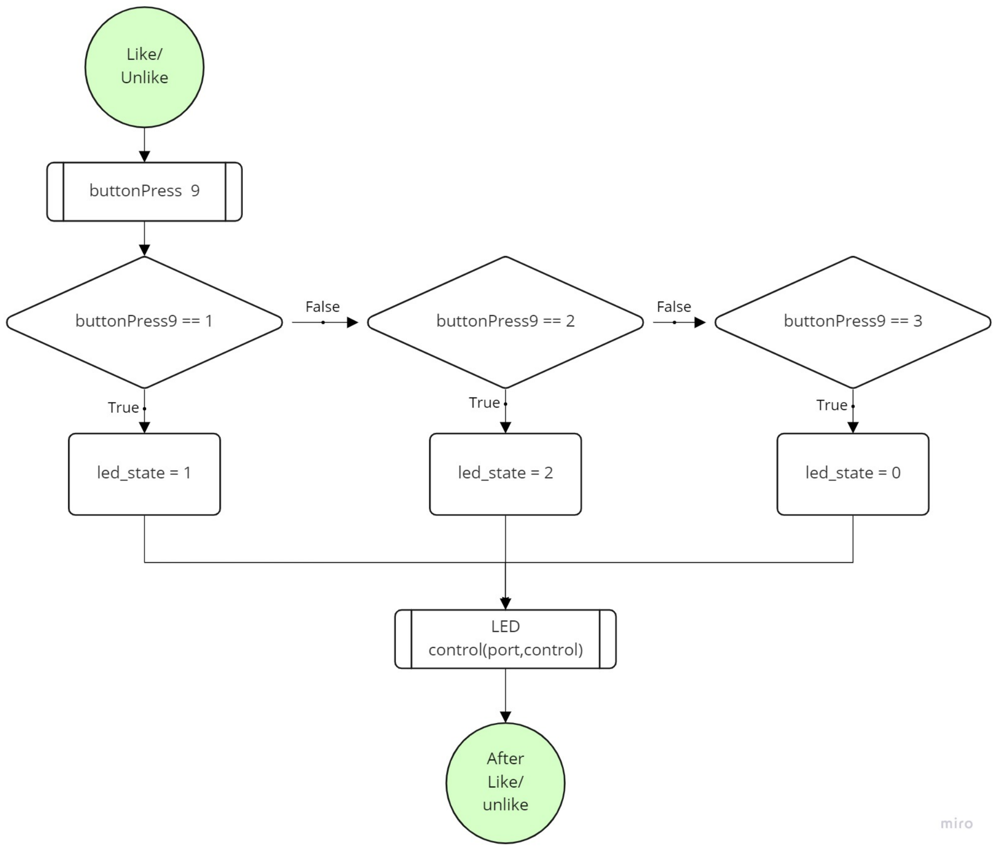


Figure 4.13: Flowchart of the code for like and unlike

1. This code runs in `Thread0`, the main program checks for the button press for triggering the like/unlike function.

2. `buttonPress` is a process where it reads `buttonState` continuously same as item 3, but in addtion `0, 1, 2` one more value of `3` is added. `3` means very long press. `button9` is the button used to send control data for like or unlike. The `buttonPress` code flow is shown in the Figure 4.14.

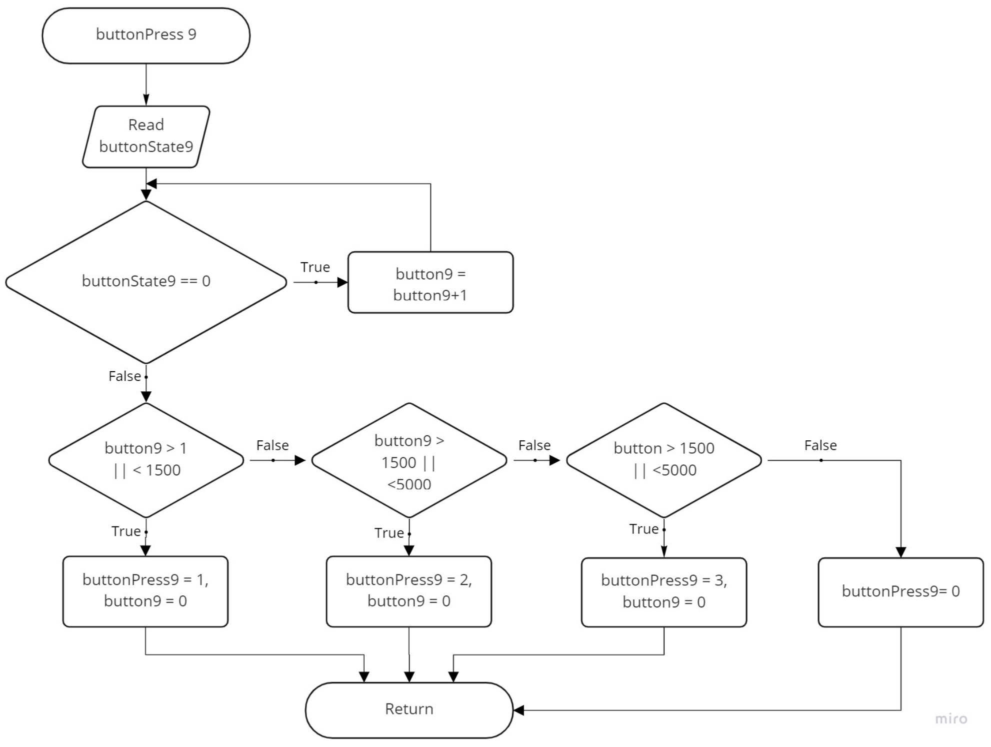


Figure 4.14: Flowchart of the code for the `buttonPress` like and dislike

3. Based on the `buttonPress` output, the program assigns a value to the `led_state` as `0,1,2` accordingly. The `led_state` is send to `LED control` for UDP communication as shown in the Figure 4.13.

4. `LED control` is the same as the section 4.2.4 for communicating the data through UDP. But in `LED control` in addditon to the port, `led_state` is sent as a control data. Where `1` is `like` and turns on green LED, `2` is `unlike` and turn on red LED, `0` turns of both the LEDs means neutral (depends on the users).

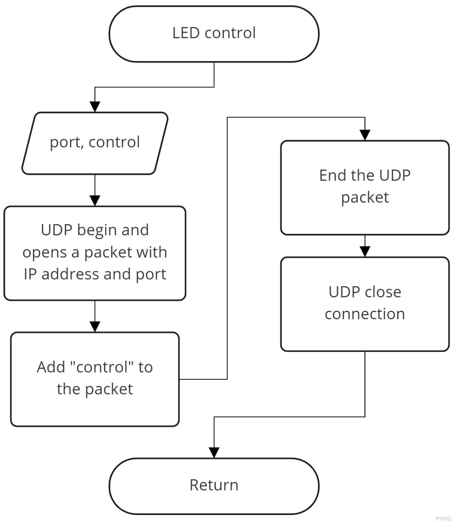
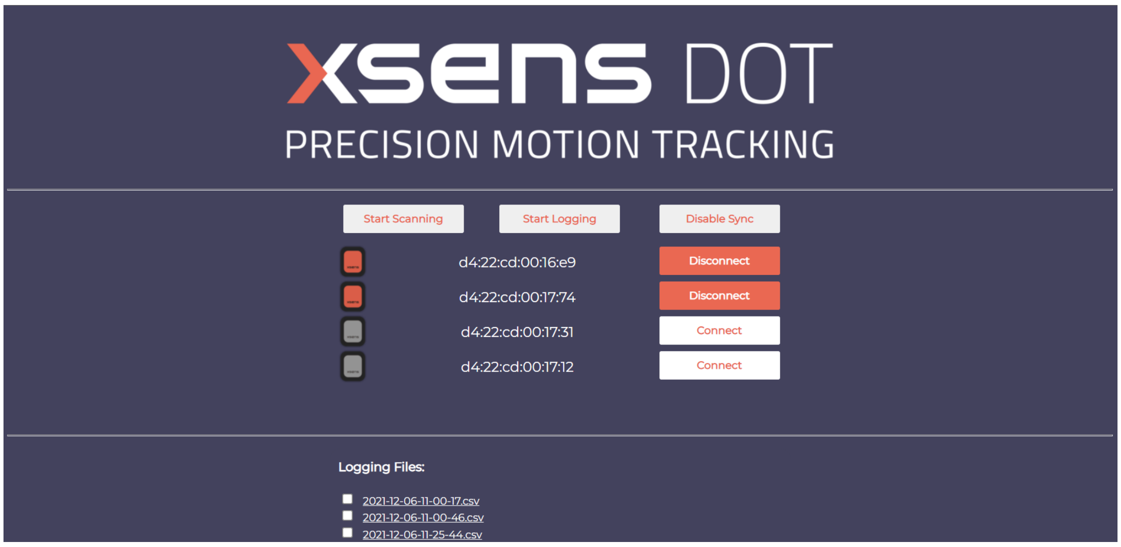

LED control
port, control
UDP begin and opens a packet with IP address and port
Add "control" to the packet
End the UDP packet
UDP close connection
Return

Figure 4.15: Flowchart of the code for the UDP communication of like and unlike

4.2.6 Sensor data transfer

Xsens Dot has its own windows application for connecting with Xsens Dot sensors. Xsens application is based on JavaScript subsection 3.4.5. The application runs on the local server with a set of UI options Figure 4.16. The flow of the sensor from sensor connection till transmitting data to Max/MSP is given in Figure 4.17.



d4:22:cd:00:16:e9
d4:22:cd:00:17:74
d4:22:cd:00:17:31
d4:22:cd:00:17:12

Logging Files:

- 2021-12-06-11-00-17.csv
- 2021-12-06-11-00-46.csv
- 2021-12-06-11-25-44.csv

Figure 4.16: Xsens Dot UI

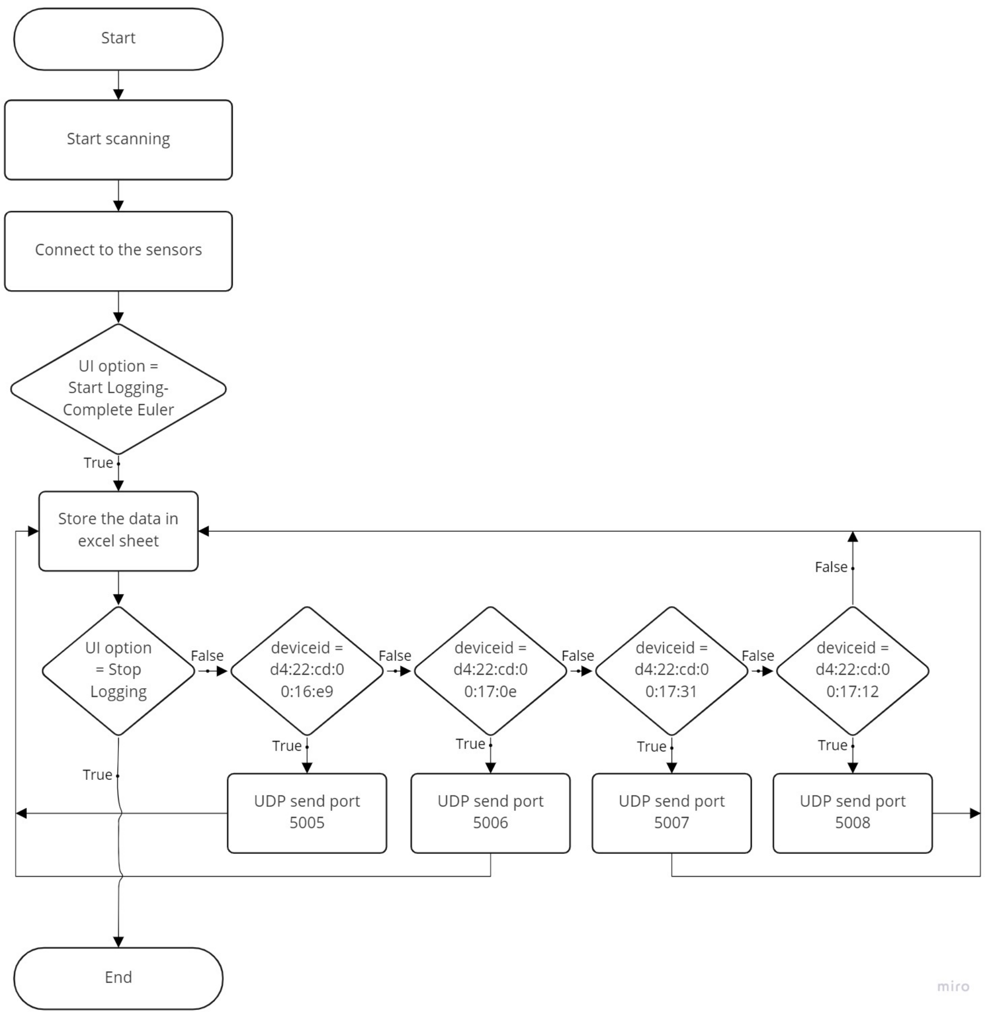

Start
Start scanning
Connect to the sensors
UI option = Start Logging-Complete Euler
True
Store the data in excel sheet
UI option = Stop Logging
False
True
deviceid = d4:22:cd:00:16:e9
deviceid = d4:22:cd:00:17:0e
deviceid = d4:22:cd:00:17:31
deviceid = d4:22:cd:00:17:12
UDP send port 5005
UDP send port 5006
UDP send port 5007
UDP send port 5008
End
miro

Figure 4.17: Flowchart of the sensor from the sensor connection to the UDP transmission

1. The UI of Xsens Dot has a button `Start Scanning`, which checks for the available sensors for connection.
2. After establishing a connection with the sensor, the UI has an option `Start Logging` with options to start recording the sensor data. The option chosen for our implementation was `Complete Euler` to get angle and acceleration data.
3. Xsens Dot stores the data in an excel sheet and parallely checks whether the `Stop Logging` option is pressed to stop recording data.
4. When the `Stop Logging` is not pressed, the data entering the excel sheet also transmitted through the UDP port created the device id. Max/MSP will receive the transmitted data.

4.3 Max/MSP

This session discusses how the sensor data is used to manipulate the sound properties, and how the data are preprocessed for sound manipulation.

4.3.1 Opening a track and receiving the sensor data

A first step of the flow in Max/MSP is to read the file from the local folder and to receive the sensor data from the sensor and the controller data from the microcontroller. Max/MSP is setup for receiving the data and for opening the audio WAV file subsection 3.4.6. The Max/MSP patch in subsection 3.4.6 is implemented to receive data only from one sensor, from here on the same sensor is used as an example in upcoming sections.

4.3.2 Preprocessing of sensor data

The acceleration data received by the Max/MSP is the acceleration in x, y, and z directions. Acceleration data varies according to the speed of the movement, and if the movement is fast, the acceleration data will be large at that point of time. Xsens Dot accelerometers can pick up tiny movements and give output in decimals. Instead of having decimal values (0.000 - 0.999), it is better to round the values to a whole number that will be easier for calculations. Acceleration data are taken as an absolute value and compared among the 3 axes and the maximum is routed to manipulate the sound properties.

The angle outputs from the sensors are angles around the x, y and z axes. The angle outputs from the y and z axes create a 2-D pointer or cursor by scaling. The angle outputs are scaled to form a 127 x 127 rectangle. The y and z angles are scaled to 0 - 127 and used as the vertical and horizontal sides of the square. Figure 4.18 shows that the sensor data from the sensor was received by the UDP receiver and are manipulated to translate body movements into a cursor. The values under the square in Figure 4.18 are the x and y coordinates of the square.

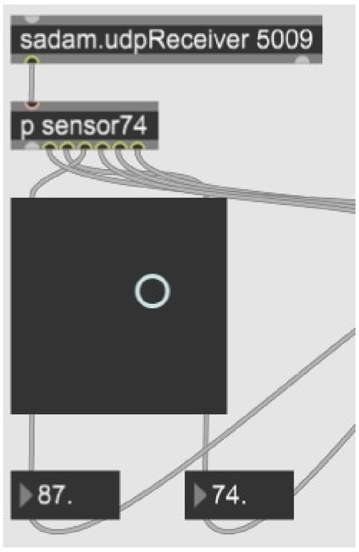


Figure 4.18: Program for using y and z axes as a cursor, blue circle is the cursor

4.3.3 Mapping of acceleration data

Max/MSP uses the maximum absolute values and a counter to check for the body tempo. A counter with a threshold and a timer were used to detect how fast the user is moving. If the accelerometer value is greater than the threshold, it triggers a bang (is like a push button) and subsequently counts the number of bangs in 1 s, see Figure 4.19. A function that plays the recorded file uses the output of the counter to set the playback speed.

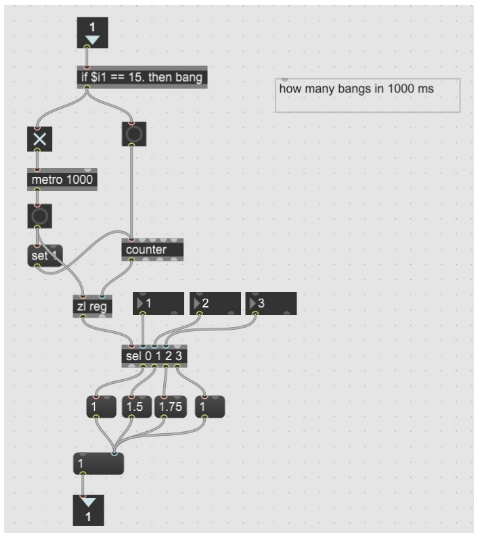


Figure 4.19: Program for determining the playback speed from acceleration

The preprocessed accelerometer data controls the audio file's playing volume. When the maximum accelerometer value reaches the function, according to the value there will be a moment of increase in the volume. Since all the direction values are combined, and the maximum of the combination is taken, user movements in any direction will have the same effect on the sounds.

4.3.4 Mapping of angular data

As per the preprocessing subsection 4.3.2, the angle outputs are used to create a 2-D pointer. These values are used as the scaled inputs to the `filtergraph`, which creates different sounds for different filters. Also by using the values x values of 2-D pointer the volume of the audio is controlled, when the pointer is at the top then the volume will be maximum and vice versa.

The angular data from the x axes is used for echoing effects, the absolute data is passed to a function delay, which controls the echoing effect to the recordings. The values sent are scaled to a range of 0 - 300 for better echo effects. Figure 4.20 is the graphical program used in the Max/MSP to control the volume and create an echo effect.

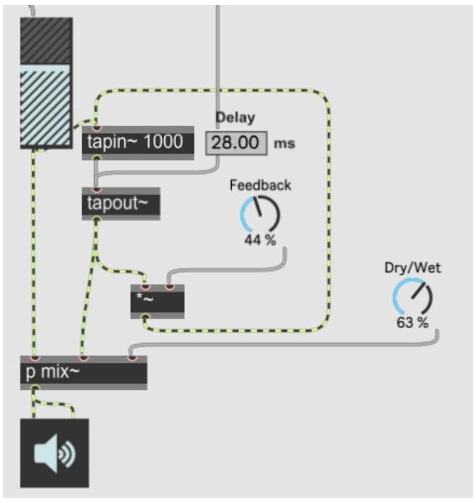


Figure 4.20: Program for controlling echo and volume

4.3.5 Control data

The control data from the microcontroller will be received by the Max/MSP using the UDP receiver function. The received values from the track control are connected to the audio output of the Max/MSP, which will mute/unmute the audio track.

LED control data will be received by the same UDP receiver function. The data received will be checked to turn `On` or `Off`, see Figure 4.21

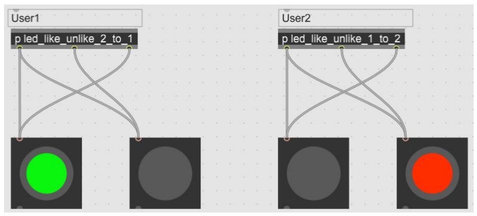


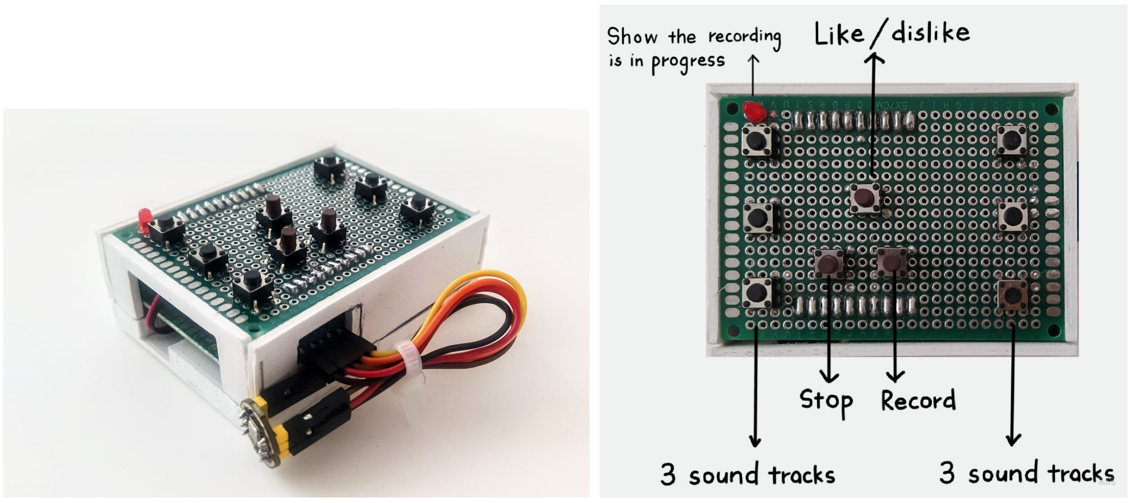
Figure 4.21: Program for LED notification for likeable or comfortable

4.3.6 Mapping of the tracks

The user was allowed to record six tracks and was attached with two sensors. However, having six tracks mapped to both sensors can limit the user's exploration. If the user wore the sensor on their hands for stimming, the upward left-hand movement would have the same effect on the tracks as the right-hand movement. If the six tracks were separated into three tracks and mapped to each sensor, there would be a possibility for the user to explore more.

4.4 How to use the prototype

This section discusses how the system is used for participatory sense-making along with the guides on how to operate the prototype. The prototype's updated connection (Figure 4.1) is used for finishing up the prototype and are encased using a simple 3D model, see Figure 4.22.



(a) Encased complete prototype (b) Top view of the prototype with the functionalities

Figure 4.22: Developed Stim4sound prototype

4.4.1 How to operate the prototype

The user should know the availability of different functions of the buttons and how to operate them. There are nine buttons in the prototype Figure 4.22 (b). Few buttons were programmed for multiple functions and were triggered by the different duration of button press which will be discussed in this section. Different button presses used in the prototype were short-press - when the button is pressed less than a second, long-press - When the button is pressed for 3 seconds, and very long press - when the button is pressed for more than 6 seconds.

1. Record (short press): the record button does not have multiple functions and triggers recording process.
2. Stop (short press): it is a function to stop the recording process.
3. Button tracks (short press, long press): each button has tagged to particular tracks and multiple functions (mute, unmute and delete).
   - Short press - this toggles between muting and unmuting of the tracks.
   - Long press - is used to delete a track from the SD card.
4. Like/ unlike (short press, long press, very long press): this button is to notify the other user whether their sound is likeable.
   - short press - if the sound is likeable. (turns on green visual LED)
   - long press - if the sound is not likeable. (turns on red visual LED)
   - very long press - if the sound is neutral. (turns off the visual LED)

The Stim4Sound's concept is an interactive process (section 3.1) between two users from diverse backgrounds. Below, questions were answered for use case of the prototype and also a short summary of the assignment.

1. Who is going to use the product?
   The aim of the product is to make an NT person to understand stimming better, and so the use case is primarily for NT and an autistic person.
2. What will the user do in participatory sense-making?
   The user will be exploring soundscapes and stimming with the fellow user to engage in an interactive activity using the prototype.
3. How will the user use the prototype for the interaction activity?
   The user will be using the prototype to record sounds created by them, and to control the tracks while stimming to manipulate the sounds.

4. How does the user manipulate the sounds? The stimming of the user will manipulate the sound properties with the help of the sensors and soundscaper.

5. How are the sensors attached to the user?
   The sensor attachment is flexible with the use of an elastic band and also can be attached to the gloves (Figure 4.23). Users can stim freely with no disturbance.

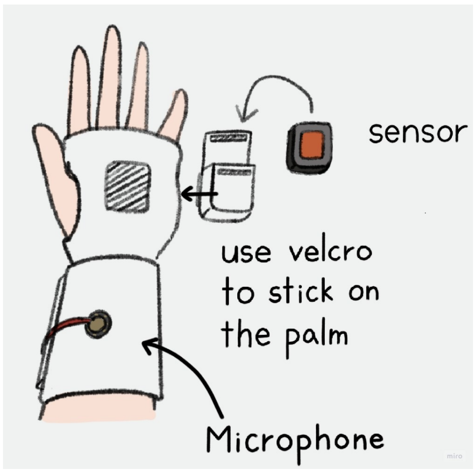


Figure 4.23: Concept design of the gloves for the prototype

6. How will the user use the soundscaper?
   The user communicates to the soundscaper Max/MSP through the prototype and the sensor. They do not have to do anything with it manually. Max/MSP does everything from playing the recordings in a loop to manipulating the sound properties with no intervention.

7. How can user listen to the manipulated sounds from the soundscaper?
   Users can use their speaker or headphones attached to the PC and they can listen to it in real-time.

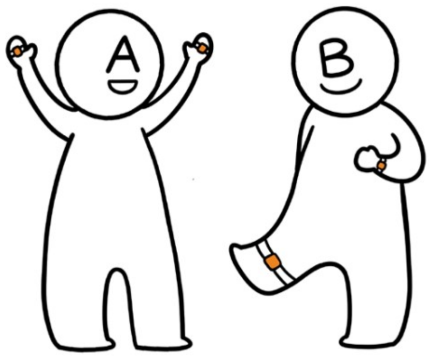


Figure 4.24: Users can stim by wearing the sensors

4.5 Conclusion

This chapter discussed the product flow of the prototype from the system initialization to the output from the soundscaper. Each microcontroller functionality's flow was explained briefly with flowcharts. Revised prototype design with the new functions was also explained in here. Finally, the use case of the prototype was given in a question and answer format for easier understanding.

This chapter answers the sub-questions "2. What could the architecture and implementation for a prototype of the proposed system look like" and "3. What technique(s) can be used to manipulate the audio signals". The sub-question two was almost answered in the previous chapter with the hardware selection, and this chapter answered it entirely by giving the final connection and the implementation of the prototype. The sub-question three were answered by the Max/MSPpatches and the mapping of the data to audio manipulation.

Chapter 5

Workshops and Evaluation

5.1 Workshop execution

The workshops were carried out after receiving approval from the ethics committee. It started after the consent from the participants in indoor rooms with social distancing. The aim of the workshops were to introduce the concept of sense making and to get user feedback for the developing the prototype. Each workshop used a version of the prototype which was revised based on the previous workshops. It was conducted between an NT and another NT person by briefly introducing the workshop and the prototype. Once the setup was explained, participants engaged in an interactive activity with guidance whenever necessary. Each session with a participant was limited to 30 min due to COVID-19.

5.2 Workshop 1

This workshop was conducted with seven participants with the help of a fellow student Yi Zhang. Due to COVID regulations, participants were invited individually. However, this workshop aimed to introduce the concept of participatory sense-making and establish a mutual connection, which requires two participants to be involved. Thus we, the workshop coordinators, took turns and engaged with the participants.

The application we used for this workshop was Loopify beta version. The app allowed the user to record a maximum of four tracks. It ran the recording in an indefinite loop, and they were manipulated using different options manually. The Stim4sound prototype was at an early development phase with only a recording function, and only its recording capabilities were tested.

1. A set of guiding music classified into happy, exciting, soft, natural, etc. were played to each participant. This music was used to get the participants to ease

into the workshop and help them start exploring sounds.

2. Different rhythms were also played to the participants to showcase the diverse possibilities available and enable them to choose a favourite rhythm.

3. After selecting the rhythm, the participants were told to explore freely to identify the objects they would like to make sounds with. The available items in the room for them to explore were glasses, ceramic mugs, chopsticks, tape, balloons, metal keys, etc., see Figure 5.1.

4. We had a basic setup of the prototype where the users could record a sound, after which we manually adjusted sound properties according to the participant's movements.

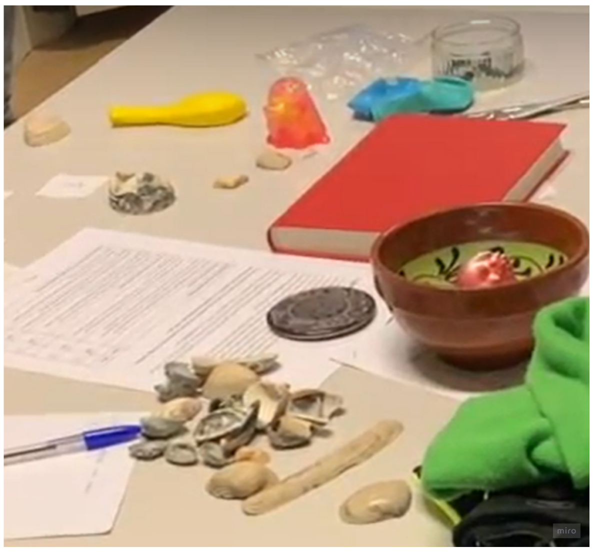


Figure 5.1: Sample objects to create music

5.2.1 Observations and results

When the guiding music and rhythms were played to the participants, they liked at least one music track. After listening to the introductory music and rhythms, they were given time to explore the sounds with their pair. Three out of seven participants tried to recreate the rhythm or the guiding music they liked using the available objects. However, they were hesitant to explore, but when we (organizers) began exploring the sounds, they followed and created new sounds eventually. Then they explored freely and created new sounds with us using multiple objects. The whole music-making using the daily objects triggered their creativity and engaged with one another for different sounds.

The participants liked multiple sounds, but they had to choose two sounds to record when it came to the recording. With the organiser's help and Loopify, each participant recorded two sounds they created. We were able to see the participants take turns while recording through verbal communication. At the same time, the developed prototype was turned on for testing. The recorded output was not good as it had lots of noises due to the wire connection.

Once the recording process was completed, participants were asked to stim along with us but were uncomfortable and did not make any significant movements. However, when the participants listened to the manipulated sounds from the movements, they were intrigued and joined the activity. While stimming, they utilized an option to mute and unmute the tracks in the application to find exciting ways to merge tracks.

A few takeaways from this workshop were that participants needed the push to make sound and record. They had trouble with making sounds, and they needed suggestions. The participants who knew each other did not hesitate to explore the sounds or record. They did not have any expectations with the sound manipulation. Even though participants hesitated initially, a mutual engagement eventually was established when creating music together and sharing the same experience.

Changes to the prototype

This workshop helped with the development of the prototype, from the observation it was required to improve the quality of the recording and add functionalities. The added functionalities were record multiple tracks and control the recorded tracks (mute and unmute).

5.3 Workshop 2

It was conducted with two participants per session when COVID regulations were relaxed and there were totally eight participants. The main aim of this workshop was to allow the participants to work with the developed prototype with the requirements from workshop 1. In the previous workshop, we were involved in sense-making activities along with the participants. Therefore they had less self-exploration by mimicking us as we initiated the process. In this workshop, we let the participants explore by themselves with minimal intervention from us.

The workshop was divided into three sessions similar to section 5.2,

1. Choosing the preferred guiding music and rhythm

Guiding music was played to the participants. They were asked to choose the preferred guiding music as a background if they needed it for the next section.

2. Using actions to make sounds
   Different actions to create sounds were introduced to the participants to explore more soundscapes using the objects available to them (Participants struggled initially to create music in workshop 1).

3. Stim freely to change the music
   The participants were provided with the sensors to have a real-time experience of the sound manipulation through their free body movements.

The prototype had the functionality of recording a maximum of six tracks per participant and the ability to control their tracks and their pairs too. Sensors were attached to an elastic band and made available. Max/MSP patch was implemented with basic sound modulations for the participant's feedback.

5.3.1 Observation and results

During the first session, two pairs out of four did not find the guiding music interesting. Instead, they preferred to explore on their own. All participants tried different ways of making music along with the ways we suggested as an example, see Figure 5.2. However, everyone was nervous about exploring sounds with strangers, but they got comfortable as their fellow participants created interesting sounds. At the same time, a pair of participants from the orchestra (orchestra pair) did not hesitate while exploring different music, unlike the participants with no music background. They used almost everything available in the room to create multiple sounds, beats that go with their pair's sounds, and actively interacted when exploring together. Other participants at times followed their pair's behaviour in creating music, like using the same objects and similar methods to create sounds, eventually interacting, adding sounds to their pair's, and taking turns.

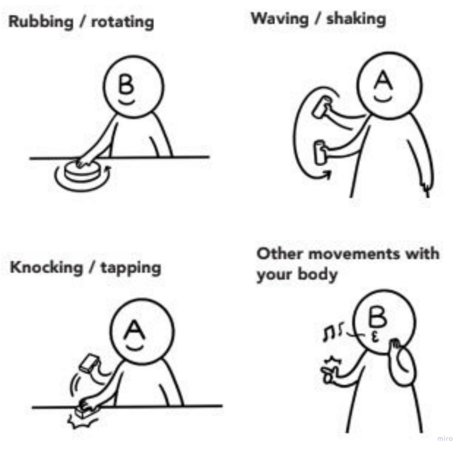


Figure 5.2: Sample actions to create music

As a part of the second session, participants were asked to record the sounds using the prototype. They found it comfortable recording with the prototype and were happy with the number of tracks they could record. On the other hand, they were unsure whether the recording process was completed because there was no notification from the prototype found to be a drawback. During the session, the recorded audio files were not transferred to the soundscape and created an issue.

Once the recording process was done, participants started unmuting their tracks for the third session. All the participants preferred wearing the sensor to their hands. Three out of four pairs tried to figure out how the sound properties were changed due to their movements. While the other pair was not interested in exploring it, they enjoyed the stimming process with sound manipulation. One sound manipulation technique was to adjust the playback speed as per the participant's movements. Everyone found it interesting, but they felt their recorded sound completely changed. Another sound manipulation technique was a sudden increase in the volume during a strong movement. One pair tried sword fighting to explore more on this sound manipulation technique. They liked one other functionality of controlling the tracks, but they were not comfortable controlling their pair's track as it may feel rude. Most participants expected a volume change when moving the hands simply up and down. Nevertheless, they had significant involvement in the activity as their movements changed sounds in real-time. The participants unconsciously mimicked each other movements enjoyed the interaction activity.

Takeaways from this workshop were that one pair asked for smaller instruments to create sounds with different tunes. In the feedback, there were different opinions regarding guiding music requirements. The participants who chose not to use it interacted more with their pairs. In terms of hesitation, except for the orchestra pair, all others had hesitation while exploring tracks and stimming. Everyone was able to involve themselves in sense-making as they dived into the sessions.

Changes to the prototype

This workshop helped update the prototype by adding the requirements and improving the prototype's quality. The changes were implemented to the prototype after the observations were made. The significant changes made were,

1. the function to control the pair's track was removed,
2. an LED was added to the prototype to notify the user of the recording process,
3. volume change to the track during up and down movements,
4. reduce the playback speed modification to retain the original sound.

5.4 User testing - revised prototype

Prototype testing was done with two participants, due to the COVID-19 regulations, using the updated prototype Figure 4.22 based on the previous workshop's feedback. Max/MSP patch was also updated with different sound manipulation techniques section 4.3.

The participants were explained about the use case of the prototype and the concept of sense-making. After the briefing, they were allowed to record sounds they liked, and feedback was collected for the prototype. Feedback for each stage of sense-making is listed down.

1. Recording stage: They found the prototype easier to record on a button push and move around with it. LED notification helped them to track the completion of the process. Nevertheless, they were frustrated to wait for 10-30 seconds for the recording file to reach the PC. Their recorded tracks were transferred to PC with no issues which was rectified from the previous workshop.
2. Stimming stage: Sensors were attached to them and they comfortably stimmed with it. The participants were excited and danced together to manipulate the sounds. There was a bug with the sensor's update identified during the testing. One of the sensors had a one-second delay which also induced a delay in sound manipulation. However, they enjoyed the whole stimming process and overlooked the delay.

3. Track controls: While stimming, they tried muting and unmuting functions. Because of the delays they faced in the previous function, they also expected the muting and unmuting will time, but there was no delay in the communication. They were happy with the track control options.

4. Like/unlike: This option was introduced to communicate through the prototype. Participants were comfortable using this function as they were roommates. They mentioned they would not have used it if they had interacted with the stranger as they found it rude.

Takeaways from the user testing were the issues with the sensor delay and the waiting time during the transfer process. The sensor problem is occurred because of the sensor's existing program. The transfer waiting time was used as a time to record sounds by another user so that both users could help each other while recording.

5.5 Effectiveness of the network Communication

Latency in communicating the data will affect the system's quality. In this prototype, it is a one way communication where the control signals from the microcontroller and the sensor data from the Xsens Dot communicate to Max/MSP. UDP communication is chosen for its low latency and little loss of packets subsection 3.4.4, but it is recommended to evaluate the prototype in terms of latency and packet losses to have a better system.

Wireshark, a packet analyzer application, was used to evaluate the communication issues for this prototype. It captures the packets in a network connection and decodes them into data. The decoded data consists of the sender's and receiver's IP address, the port used to send the packets, the time when the packets were received by the receiver and the data embedded in the packet. These data were used to check the latency and packet loss.

Latency can be calculated using the received timestamp from the Wireshark and sent timestamp from the sensor. The difference between these timestamps is the time taken for the packets to travel from the sender to the receiver port, which gives the communication latency. Figure 5.3 shows the distribution of latency across the time in milliseconds for the sensor data to reach the Max/MSP. It is observed that the average latency was between 3-4 ms which is negligible and does not affect the system's working.

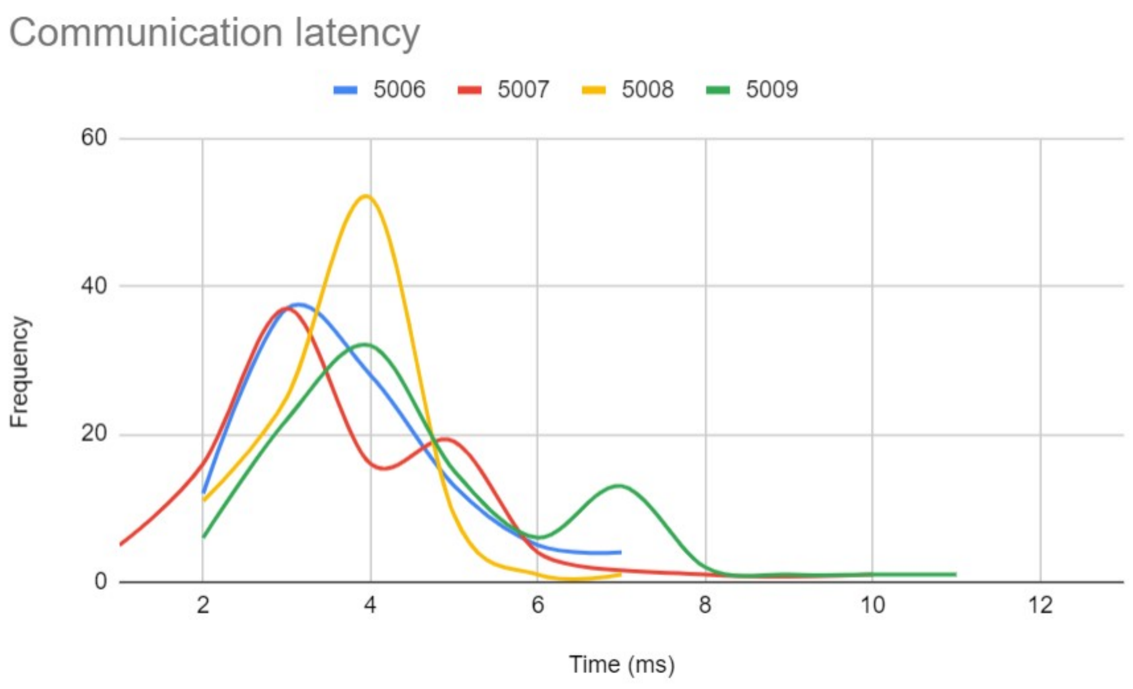


Figure 5.3: Distribution of Communication Latency

Packet losses were calculated by comparing the counter values from the sensor to the total values received by the Wireshark. It was checked for the sensors individually and together. The Table 5.1 shows the comparison of sent and received data for the sensors. There was no loss in packets during communication when the sensors were connected individually.

When the sensor data was grouped into three sensors, there were discrepancies in the packets, as seen from Table 5.1 the sensor with port 5007 sent and received 256 packets of data while the other two sensors sent and received 141 and 142 packets. This discrepancy did not affect the performance of the prototype. After looking into the sensor data, it was because of the sensor's delayed connection to the application.

A major issue was found after the sensors firmware was updated; when sensors were grouped into four sensors, there were no packets from one of the sensors. This created a problem with the prototypes working. This issue did not happen with the earlier version. There are two reasons for this issue to occur, it occurred because of the sensor's firmware and the laptop used for establishing connections.

Sensor id	UDP port	Packets sent	Packets received
d4:22:cd:00:17:0e	5006	140	140
d4:22:cd:00:17:31	5007	188	188
d4:22:cd:00:17:12	5008	177	177
d4:22:cd:00:17:74	5009	221	221
d4:22:cd:00:17:31	5007	256	256
d4:22:cd:00:17:12	5008	142	142
d4:22:cd:00:17:74	5009	141	141
d4:22:cd:00:17:0e	5006	0	0
d4:22:cd:00:17:31	5007	430	430
d4:22:cd:00:17:12	5008	443	443
d4:22:cd:00:17:74	5009	756	756

Table 5.1:

The UDP communication between the sensor and Max/MSP have issues with latency but with the number of sensors being used together. This can be resolved with the new update of the sensor's firmware or laptop with different specifications.

Chapter 6

Conclusions and recommendations

6.1 Conclusions

This thesis explored the ways to implement a wearable device to enable participatory sense-making. The aim of the study is to answer the research question"How to design a set of two wearable devices that allows users to record and manipulate sounds while also allowing them to connect with one another". In order to answer this question, sub questions were answered in different chapters of the thesis.

How to measure the body movements (e.g. hands, legs or wrists) for sound manipulation?

Based on the literature survey, the type of sensor needed for the development was an IMU that can measure body movements with high accuracy. For the sensor to gather accurate data, it should be appropriately placed, decided from the wearability of the design. Using Xsens Dot from the hardware selection and proper sensor placement delivers accurate outputs that are then sent to Max/MSP for sound manipulation.

What could the architecture and implementation for a prototype of the proposed system look like?

The requirements and features were identified from the concept of sound collaboration activity. The different applications with the music and interaction system used the same high-level architecture. The system was divided into the motion sensor, communication, sound recorder, controller and soundscaper. Based on the architecture, hardware was selected and a prototype implemented. After a series of workshops, the architecture was updated, and the functionalities were added to the prototype. The final prototype had user interface buttons, a microphone, storage device, motion sensor, microcontroller and soundscaper.

What technique(s) can be used to manipulate the audio signals?

The sensor data gathered from body movements are transmitted to Max/MSP for sound manipulation. Each body movement was mapped as a manipulator of a particular sound property. Chapter 4 answered this sub-question by implementing the Max/MSP patch in the soundscaper to have arbitrary sound effects. These are the techniques as used in the audio effects to manipulate the audio signals.

All the sub-questions put together forms the answer to the main research question. The literature survey of the sensors to the techniques for the audio manipulation helped develop a working prototype that engages the user to interact with the fellow user. The prototype serves the purpose of participatory sense-making but has not yet been tested with autistic people. However, with the help of this prototype, NT persons will likely understand the benefits of stimming, and they will stop controlling or judging autistic people. There is scope for further development of the prototype and recommendations will be given in the next section.

6.2 Recommendations

The following recommendations are suggested for future work on the Stim4Sound prototype.

This prototype was not tested with autistic people due to COVID. It is important to test with autistic people for their feedback on possible improvement of the prototype. The prototype should be comfortable for both type of users.

In my recent interaction with my nephew, a five-year-old autistic kid, he stims with his fingers and does not interact with anyone. His stimming made me realize that many more stimming behaviours were not included in this prototype. This prototype can be extended to measure finger movements using flex resistors or EMG sensors as future work.

This prototype uses Max/MSP as the soundscaper, which is efficient, but it was challenging to make a standalone device for Stim4Sound because of Max/MSP's unavailability in Raspberry pi or such systems. In the future, to make the Stim4Sound as a standalone device, the whole setup can be implemented with Raspberry PI as a speaker and soundscaper. This standalone device can have a whole package with two controllers, four sensors and a speaker.

It is possible to print the circuit on a flexible PCB board, making the controller smaller than it is now. The prototype should be simple as possible because having multiple functions or any other distracting additions to the prototype will likely hamper interaction activity.

In this development of Stim4Sound, the main concern was the choice of an Xsens Dot as the sensor. However, Xsens Dot worked as per the expectations, but they did not have a proper application to read or stream the data. The instability of the sensor updates and compatibility with the laptops made it challenging to work with. In the future, if the Xsens Dot gets a stable release, it can be continuously used along with other sensors, or any other wireless sensors with the proper application interface can be selected.

Although the battery capacity of the prototype supplies current for 1.5 - 2 h, which is sufficient, if a battery type with a larger capacity is selected the interaction process can be prolonged and one does not need to worry about the battery being exhausted during the interaction activity. This development was not co-designed with the user. The functionalities and sound manipulation can be improved a lot with the user's involvement during the further development process. A person with a musical background can make sound manipulation more enjoyable and more diverse.

Bibliography

[1] V. Zawn, "Foot-tapping and hand-flapping: Why do people stim?" 1 2019. [Online]. Available: https://www.goodtherapy.org/blog/foot-tapping-hand-flapping-why-do-people-stim-0104194

[2] O. I. Lovaas, L. Schreibman, and R. L. Koegel, "A behavior modification approach to the treatment of autistic children 1,2," 1974.

[3] S. K. Kapp, R. Steward, L. Crane, D. Elliott, C. Elphick, E. Pellicano, and G. Russell, "'people should be allowed to do what they like': Autistic adults' views and experiences of stimming," Autism, vol. 23, pp. 1782–1792, 10 2019.

[4] H. Hodges, C. Fealko, and N. Soares, "Autism spectrum disorder: definition, epidemiology, causes, and clinical evaluation," Translational Pediatrics, vol. 9, pp. S55–S65, 2 2020.

[5] C. Lord, E. H. Cook, B. L. Leventhal, and D. G. Amaral, "Review autism spectrum disorders coordinating vocalizations with their intentions, and com," pp. 355–363, 2000.

[6] K. Maich, "Autism spectrum disorders in popular media: Storied reflections of societal views," Brock Education Journal, vol. 23, no. 2, 2014.

[7] R. R. Grinker, "Autism, "stigma," disability a shifting historical terrain," Current Anthropology, vol. 61, pp. S55–S67, 2 2020.

[8] O. I. Lovaas, "Behavioral treatment and normal educational and intellectual functioning in young autistic children." Journal of Consulting and Clinical Psychology, vol. 55, pp. 3–9, 1987.

[9] N. C. on Birth Defects and D. D. (NCBDDD), "Data statistics on autism spectrum disorder," 12 2021. [Online]. Available: https://www.cdc.gov/ncbddd/autism/data.html

[10] C. O. Osborn, "Signs of autism in a 3-year-old: Symptoms and diagnosis," Nov 2021. [Online]. Available: https://www.healthline.com/health/signs-of-autism-in-3-year-old#symptoms

[11]

[12] S. Fletcher-Watson, J. Adams, K. Brook, T. Charman, L. Crane, J. Cusack, S. Leekam, D. Milton, J. R. Parr, and E. Pellicano, “Making the future together: Shaping autism research through meaningful participation,” Autism, vol. 23, pp. 943–953, 5 2019.

[13] S. S. Rajagopalan, A. Dhall, and R. Goecke, “Self-stimulatory behaviours in the wild for autism diagnosis,” pp. 755–761, 2013.

[14] R. Amundson, “Against normal function,” Studies in History and Philosophy of Science Part C: Studies in History and Philosophy of Biological and Biomedical Sciences, vol. 31, no. 1, pp. 33–53, 2000.

[15] R. Lilley, “What’s in a flap? the curious history of autism and hand stereotypies,” Manuscript submitted for publication, 2018.

[16] K. L. Lichstein and L. Schreibman, “Employing electric shock with autistic children,” Journal of autism and childhood schizophrenia, vol. 6, no. 2, pp. 163–173, 1976.

[17] L. J. Miller, M. E. Anzalone, S. J. Lane, S. A. Cermak, and E. T. Osten, “Concept evolution in sensory integration: A proposed nosology for diagnosis,” American Journal of occupational therapy, vol. 61, no. 2, pp. 135–140, 2007.

[18] R. A. Charlton, T. Entecott, E. Belova, and G. Nwaordu, ““it feels like holding back something you need to say”: Autistic and non-autistic adults accounts of sensory experiences and stimming,” Research in Autism Spectrum Disorders, vol. 89, p. 101864, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1750946721001392

[19] S. Fletcher-Watson, H. D. Jaegher, J. V. Dijk, C. Frauenberger, M. Magnée, and J. Ye, “Diversity computing,” Interactions, vol. 25, pp. 28–33, 9 2018.

[20] H. D. Jaegher and E. D. Paolo, “Participatory sense-making: An enactive approach to social cognition,” Phenomenology and the Cognitive Sciences, vol. 6, pp. 485–507, 12 2007.

[21] H. D. Jaegher, “Embodiment and sense-making in autism,” Frontiers in Integrative Neuroscience, 2 2013.

[22] E. Rynell, “Uc irvine 2016 conference proceedings title acting as participatory sense-making,” 2016. [Online]. Available: https://escholarship.org/uc/item/182756v1

[23] N. Davis, C. P. Hsiao, K. Y. Singh, L. Li, and B. Magerko, "Empirically studying participatory sense-making in abstract drawing with a co-creative cognitive agent," vol. 07-10-March-2016. Association for Computing Machinery, 3 2016, pp. 196–207.

[24] A. Schiavio and H. D. Jaegher, "Participatory sense-making in joint musical practice," pp. 31–39, 4 2017.

[25] A. Zelechowska, V. E. Gonzalez-Sanchez, B. Laeng, and A. R. Jensenius, "Headphones or speakers? an exploratory study of their effects on spontaneous body movement to rhythmic music," Frontiers in Psychology, vol. 11, 4 2020.

[26] R. I. Godøy and A. R. Jensenius, "Body movement in music information retrieval," 2009.

[27] B. Burger, M. R. Thompson, G. Luck, S. Saarikallio, and P. Toiviainen, "Influences of rhythm- and timbre-related musical features on characteristics of music-induced movement," Frontiers in Psychology, vol. 4, 2013.

[28] T. Nguyen, "Stim4sound : a diversity computing device helps to alleviate the double empathy problem," May 2021. [Online]. Available: http://essay.utwente.nl/86193/

[29] M. K. Liu, Y. T. Lin, Z. W. Qiu, C. K. Kuo, and C. K. Wu, "Hand gesture recognition by a mmg-based wearable device," IEEE Sensors Journal, vol. 20, pp. 14 703–14 712, 12 2020.

[30] Byun and Lee, "Implementation of hand gesture recognition device applicable to smart watch based on flexible epidermal tactile sensor array," Micromachines, vol. 10, p. 692, 10 2019.

[31] H. Zhou and H. Hu, "Human motion tracking for rehabilitation—a survey," Biomedical Signal Processing and Control, vol. 3, pp. 1–18, 01 2008.

[32] J.-F. Wu, C. Qiu, Y. Wang, R. Zhao, Z.-P. Cai, X. Zhao, S.-S. He, F. Wang, Q. Wang, and J.-Q. Li, "Human limb motion detection with novel flexible capacitive angle sensor based on conductive textile," Electronics, vol. 7, p. 192, 09 2018.

[33] H. Lahamy and D. Lichti, "Real-time hand gesture recognition using range cameras," 05 2012.

[34] J. K. Westlund, S. K. D'Mello, and A. M. Olney, "Motion tracker: Camera-based monitoring of bodily movements using motion silhouettes," PLOS ONE, vol. 10, p. e0130293, 6 2015.

[35] F. Cordella, F. D. Corato, L. Zollo, B. Siciliano, and P. V. D. Smagt, "Patient performance evaluation using kinect and monte carlo-based finger tracking," 2012, pp. 1967–1972.

[36] T. Guzsvinecz, V. Szucs, and C. Sik-Lanyi, "Suitability of the kinect sensor and leap motion controller—a literature review," Sensors, vol. 19, p. 1072, 3 2019.

[37] G. Wolterink, P. Dias, R. G. P. Sanders, F. Muijzer, B.-J. v. Beijnum, P. Veltink, and G. Krijnen, "Development of soft semg sensing structures using 3d-printing technologies," Sensors, vol. 20, no. 15, 2020. [Online]. Available: https://www.mdpi.com/1424-8220/20/15/4292

[38] M. Donnarumma, B. Caramiaux, and A. Tanaka, "Muscular interactions combining emg and mmg sensing for musical practice." [Online]. Available: http://res.marcodonnarumma.com/projects/xth-sense/

[39] A. Filippeschi, N. Schmitz, M. Miezal, G. Bleser, E. Ruffaldi, and D. Stricker, "Survey of motion tracking methods based on inertial sensors: A focus on upper limb human motion," 6 2017.

[40] A. Tajadura-Jiménez, A. Singh, F. Cuadrado, P. Rick, A. Väljamäe, N. Bianchi-Berthouze, and F. Bevilacqua, "Designing a gesture-sound wearable system to motivate physical activity by altering body perception." Association for Computing Machinery, 6 2018.

[41] F. Visi, E. Coorevits, R. Schramm, and E. R. Miranda, "Musical instruments, body movement, space, and motion data: Music as an emergent multimodal choreography," Human Technology, vol. 13, pp. 58–81, 2017.

[42] L. E. Dunne, H. Profita, C. Zeagler, J. Clawson, S. Gilliland, E. Y.-L. Do, and J. Budd, "The social comfort of wearable technology and gestural interaction." IEEE, 8 2014, pp. 4159–4162.

[43] M. Canina and V. Ferraro, "The biodesign approach to wearable devices." IEEE, 2008, pp. 264–267.

[44] V. G. Motti and K. Caine, "Human factors considerations in the design of wearable devices," Proceedings of the Human Factors and Ergonomics Society Annual Meeting, vol. 58, pp. 1820–1824, 9 2014.

[45] E. Dagan, E. M. Segura, F. A. Bertran, M. Flores, R. Mitchell, and K. Isbister, "Design framework for social wearables." ACM, 6 2019, pp. 1001–1015.

[46] C.-C. Yang and Y.-L. Hsu, "A review of accelerometry-based wearable motion detectors for physical activity monitoring," Sensors, vol. 10, pp. 7772–7788, 8 2010.

[47] C. Zeagler, "Where to wear it: Functional, technical, and social considerations in on-body location for wearable technology 20 years of designing for wearability," New York, NY, USA, p. 150–157, 2017. [Online]. Available: https://doi.org/10.1145/3123021.3123042

[48] P.-J. Maes, V. Lorenzoni, B. Moens, J. Six, F. Bressan, I. Schepers, and M. Leman, "Embodied, participatory sense-making in digitally-augmented music practices: Theoretical principles and the artistic case "soundbikes"," Critical Arts, vol. 32, pp. 77–94, 5 2018.

[49] S. Thorn, "Telematic wearable music." ACM, 9 2021, pp. 188–195.

[50] C. Erdem, K. H. Schia, and A. R. Jensenius, "Vrengt: A shared body-machine instrument for music-dance performance," 10 2020. [Online]. Available: http://arxiv.org/abs/2010.03779http://dx.doi.org/10.5281/zenodo.3672917

[51] C. Erkut, S. Serafin, J. Fehr, H. M. F. Figueira, T. B. Hansen, N. J. Kirwan, and M. R. Zakarian, "Design and evaluation of interactive musical fruit," in Proceedings of the 2014 conference on Interaction design and children, 2014, pp. 197–200.

[52] P. Elsea, "Basics of digital recording: Converting sound into numbers," Mar 2020. [Online]. Available: https://www.prosoundweb.com/basics-of-digital-recording-converting-sound-into-numbers/

[53] "Basics of the spi communication protocol," Nov 2021. [Online]. Available: https://www.circuitbasics.com/basics-of-the-spi-communication-protocol/

[54] "Mpu-6050 accelerometer en gyroscope 3-axis module 3.3v-5v." [Online]. Available: https://www.tinytronics.nl/shop/nl/sensoren/acceleratie-rotatie/mpu-6050-accelerometer-en-gyroscope-3-axis-module-3.3v-5v

[55] "9-dof absolute orientation imu fusion integrated card - bno055." [Online]. Available: https://www.direnc.net/9-dof-absolute-orientation-imu-fusion-breakout-bno055-adafruit-en

[56] X. knowledge base, "Xsens dot knowledge base." [Online]. Available: https://xsenstechnologies.force.com/knowledgebase/s/article/DOT-Reference-frame-and-Data-types-1605873990674?language=en_US

[57] C. Gruber, "I2s mikrophone," Apr 2017. [Online]. Available: https://community.hiveeyes.org/t/i2s-mikrophone/266

[58] A. Industries, “Adafruit i2s mems microphone breakout - sph0645lm4h.” [Online]. Available: https://www.adafruit.com/product/3421

[59] reichelt elektronik GmbH amp; Co. KG Internet Team (webmaster@reichelt.de), “Ard nano 33iot.” [Online]. Available: https://www.reichelt.nl/nl/nl/arduino-nano-33-iot-samd21g18a-zonder-koptekst--ard-nano-33iot-p261302.html?PROVID=2809&amp;gclid=CjwKCAjwloCSBhAeEiwA3hVo_RR8COhPKXdiv24LKjcss5ubY-KIwroHrjUq8yt6_0nMZNtdihcrghoC2oQQAvD_BwE

[60] “Esp32.” [Online]. Available: https://nl.aliexpress.com/item/32831394824.html?gatewayAdapt=glo2nld

[61] R. Mischianti, “Esp32 wemos lolin32 lite high resolution pinout and specs,” Renzo Mischianti, Dec 2021. [Online]. Available: https://www.mischianti.org/2021/07/30/esp32-wemos-lolin32-lite-high-resolution-pinout-and-specs/

[62] L. Espressif Systems (Shanghai) Co., “Inter-ic sound (i2s),” ESP, Feb 2022. [Online]. Available: https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/peripherals/i2s.html

[63] “Automating filezilla,” WinSCP, Jul 2016. [Online]. Available: https://winscp.net/eng/docs/guide_filezilla_automation