Bi-modal Stimulator

Simultaneous electrical and acoustical stimulation to test a hypothesis for treating tinnitus.

Stef Zwartveld

Delft University of Technology

TU Delft

Bi-modal Stimulator

Simultaneous electrical and acoustical stimulation to test a hypothesis for treating tinnitus.

by

Stef Zwartveld

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Friday April 19, 2024 at 14:00 PM.

Student number:	4495683	
Project duration:	February 1, 2023 – April 19, 2024	
Thesis committee:	Prof. dr. ir. W. Serdijn,	TU Delft, supervisor
	Dr. ir. C. Strydis,	TU Delft, Erasmus MC

This thesis is confidential and cannot be made public until April 19, 2026.

Cover: Acoustical stimulation through headphones and electrical neuron-stimulation on the tragus, generated by Copilot (Artificial Intelligence, https://learn.microsoft.com/en-us/copilot/terms-of-use)

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

TU Delft

Preface

Dear reader,

This master thesis is a document I would, at age 18, never have dreamed of finishing. A bicycle accident at age 14 resulted in a basilar skull fracture and subdural hematoma. After surgery and a short stay at the hospital, everything seemed normal. It was a great relief for me and my family.
My life turned upside down when I started my Bachelor of Applied Sciences in Aerospace Engineering. My concentration dropped, my sensory input was unfiltered, and I became too anxious to go into a grocery store. I did not trust my abilities anymore. The only joy in life was playing drums and hanging out with my closest friends at "the cab". My psychologist referred me to rehabilitation for an acquired brain injury, where I learned tools to balance my energy and sensory inputs.
2009, I started the Bachelor of Applied Sciences in Healthcare Technologies in Rotterdam. I intended to focus only on my abilities to study again without the social aspects of studying. The study was a success, I gained a new group of friends, and met my girlfriend.
Mirna, I am unbelievably thankful for your never-ending love and support. Your curiosity to discover new experiences and passion for food challenged me, for example, to enter restaurants even if I could not order my own food and we had to eat without having a conversation. You helped me to rehabilitate completely. You have had a lot of patience to support me in building a business and gaining a master's degree. Thank you so much for everything.
While building the business, I felt that I did not use my full intellectual potential. I gained a lot of experience but never tested what a competence and interest test had pointed out: Medicine and technique at a university level. When I reread this result at my parent's dinner table, I knew this would be my next goal. To be able to apply, I had to complete an English and mathematics test. So I started at the very beginning again: additions and subtraction up to essential calculus. It has been a long way from there, 5 years ago, until now, finishing this master thesis.
My parents have always been great supporters of fulfilling my dreams, and they supported my decision to go to university at age 26. So once more, thank you again for your support during the pre-masters and masters.
Next, I want to thank Jasper for his support in building and maintaining the business during the master's program and our strange, turbulent times. Without you, I could not have continued and finished the master's degree while running the business.
The pre-masters took place during the COVID-19 pandemic. One of the most challenging courses was electromagnetism, which I enjoyed studying with Jefta. Jefta, thank you for your support and joy during the pre-master's and master's. I hope you can succeed in your master's thesis soon.
Wouter, I remember that at the beginning of my master's, I struggled to choose a course program. I wanted a broad package, and after explaining my reasoning, you agreed and said, "For the thesis, we will find you a nice assignment and a combination of supervisors to support the thesis." When I started looking for assignments, I wanted to discuss this again with you, but I came across this assignment: A multimodal stimulator for tinnitus using a microcontroller. I always wanted to build a device with a microcontroller, so this was the perfect fit. Luckily, I was not the only one interested in this project. Jonathan, you joined after a few weeks, which made diving deeper into the literature possible. We divided the work into software and hardware, which are still very intertwined. It was delightful and easy to work together. Wouter asked the technician Brian to join the team and design the PCB. Together, we made a great team and produced a great result. Thank you all for the experience, support, and joy during this master thesis.
Finally, thank you, Dirk, for the exciting topic and your time helping us with the requirements. Also, the people from Tinnitus House, thank you for your support and the lovely dinners at the hangout. Hopefully, we can all meet in the future and celebrate great results from the clinical trials.

Stef Zwartveld
Den Haag, April 2024

Summary

Subjective tinnitus, also known as ringing ears or phantom sound, is a sensation of sound without an external source [1]. The prevalence is estimated at 740 million people and 120 million people with tinnitus and severe tinnitus worldwide, respectively [2]. Cognitive behavioral therapy is often used to treat tinnitus's psychological comorbidities, but the cause of tinnitus is not cured [3].

There is no clear understanding of the cause and pathology of tinnitus. Literature suggests a combination of deafferentation of the tonotopic map [4], the Bayesian brain model [5], and the triple mode network [6], creating one combined hypothesis for tinnitus pathology. Earlier, invasive vagus nerve stimulation paired with tones has been indicated to be a possible solution for tinnitus in one patient [7]. However, the invasiveness of this treatment does not outweigh the severity for most tinnitus sufferers. Based on the combined hypothesis, transcutaneous vagus nerve stimulation paired with tinnitus-matched sound stimulation should train the brain to reduce the tinnitus sound's importance, thereby eliminating it from conscious awareness.

To investigate this hypothesis in clinical studies, a device that combines transcutaneous electrical nerve stimulation on the tragus with tinnitus-matched sounds in the ears is needed. As it will be used in clinical trials, there should be an option for a control group. Multiple patients will be investigated thus every patient should have their personal settings and sounds in the device. The device should be easy to use and electrically safe.

A device has been designed with a user interface that interacts with a microcontroller, following the requirements. The user interface comprises an LCD screen with a rotary encoder to adjust the digital menu. The microcontroller controls the audio stimulation shield and the custom-made stimulation circuit designed by Jonathan Kneepkens. The audio is played from the MicroSD card through an audio player shield. An internal battery powers the device to prevent direct contact between the patient and the mains. An error message is displayed on the LCD screen in case of a device error. During the error messages, the stimulation is always deactivated for safety.

Electrical stimulation of the tragus provides the right effect. It activates the parasympathetic nervous system. The device's efficacy as a treatment must await clinical trials with patient-specific tinnitus-matched acoustical stimulation.

Contents

Nomenclature

Abbreviations

Abbreviation	Definition
AC	Auditory Cortex
AC1	Primary auditory cortex
ADHD	Attention deficit hyperactivity disorder
AI	Anterior insula
BBN	Broadband Noise
BFR	Burst firing rate
BOLD	Blood oxygen level dependent
C2	Cervical nerve 2
C5	Cervical nerve 5
CN	Cochlear nucleus
CN V	Trigeminal Nerve
CN VIII	Vestibulocochlear Nerve
CN X	Vagus nerve
CEN	Central Executive Network
CNS	Central nervous system
DAC	Digital Analog Converter
dACC	Dorsal anterior cingulate cortex
DBS	Deep brain stimulation
DC	Direct current
DCN	Dorsal cochlear nucleus
DMN	Default mode network
EEG	Electroencephalography
FPGA	Field programmable gate array
FDA	Food & Drug Administration
fMRI	Functional Magnetic Resonance Imaging
FUS	Focused ultrasound stimulation
HF	High frequency
iACS	Intracranial alternating current stimulation
IC	Inferior collicullus
IDE	Integrated development environment
ISR	Interrupt service routine
iVNS	Invasive vagus nerve stimulation
I2C	Inter-Integrated Circuit
LC	Locus coeruleus
LTA	Left temporoparietal area
µC	Micro controller
MGB	Medial geniculate body
MISO	Master in slave out
MOSI	Master out slave in
MVP	Minimal viable product
nLL	Lateral lemniscus
NTS	Nucleus tractus solitarius
OCD	Obsessive-compulsive disorder
P1	The first pulse of the burst waveform
P2	The second pulse of the burst waveform

Abbreviation	Definition
P3	The third pulse of the burst waveform
P4	The fourth pulse of the burst waveform
P5	The fifth pulse of the burst waveform
P6	The sixth pulse of the burst waveform or the charge-balancing pulse
PCB	Printed circuit board
PD	Pulse delay
pgACC	Pregenual anterior cingulate cortex
PWM	Pulse width modulation
rACC	Rostral anterior cingulate cortex
rTMS	Repetitive transcranial magnetic stimulation
RNS	Random noise stimulation
SCL	Serial Clock
SCLK	Serial Clock
SDA	Serial Data
SFR	Spontaneous firing rate
SN	Salience network
sgACC	Subgenual anterior cingulate cortex
SPI	Serial Peripheral Interface
SS/CS	Slave Select / Chip Select
SOC	Superior olivary complex
tACS	Transcranial alternating current stimulation
taVNS	Transcutaneous auricular branch vagus nerve stimulation
tDCS	Transcranial direct current stimulation
TENS	Transcutaneous electrical nerve stimulation
tRNS	Transcranial random noise stimulation
TMS	Transcranial magnetic stimulation
TNM	Triple network model
TFI	Tinnitus function index
THI	Tinnitus Handicap Index
TMNMT	Tailor-made notched music therapy
tVNS	Transcutaneous vagus nerve stimulation
U-HF	Ultra high frequency
UART	Serial Universal Asynchronous Reciever Transmitter
µC	Microcontroller
UI	User interface
VCN	Ventral cochlear nucleus
VNS	Vagus Nerve Stimulation

Symbols

Symbol	Definition	Unit
I	Current	[A]
r	Distance	[m]
T	Time period	[s]
v_{sound}	Velocity of sound	[m/s]
V_e	Extracellular potential	[V]
σ	Conductivity	[S/m]
λ	Wavelength	[m]

1 Introduction

Tinnitus, also known as ringing ears or phantom sound, is a sensation of sound without an external source [1]. Tinnitus is subdivided into subjective and objective tinnitus. Objective tinnitus has a source inside the body, such as the heart rate, which other people can perceive. Objective tinnitus has no source, inside or outside of the body. Only subjective tinnitus is considered in this master's thesis, and it will be referred to as tinnitus.

Until 1990, tinnitus was frequently experienced by 35% of the population, and 15% as continuous [8]. A review study by McCormack, et al. considered 12 studies out of 500 papers between 1980 and 2015. In the study, only papers using a similar definition for tinnitus were used. These papers report a prevalence between 11.9% and 30.3% of the population [9]. Jarach, et al. estimate a worldwide prevalence of 740 million and 120 million people with tinnitus and severe tinnitus, respectively [2].

Tinnitus occurs in association with other hearing problems [8]. Tinnitus is often related to psychological discomforts such as stress, anxiety, and depression. Cognitive behavioral therapy is frequently used to treat the psychological comorbidities of tinnitus, but the cause of tinnitus is not cured [3]. Besides psychological comorbidities, there are physical comorbidities, such as hearing loss and hyperacusis.

To this date, there is no clear understanding of the cause and pathology of tinnitus. Research thus far points in the direction of a combination of deafferentation of the tonotopic map [4], the Bayesian brain model [5], and the triple mode network [6], creating one combined hypothesis for tinnitus pathology: Under stress, the brain considers all sensory input as salient. In the case of deafferentation of the tonotopic map, the brain is missing auditory information, which is necessary to make a prediction of the world around it. This missing information is so important that it is retrieved from memory, resulting in the perception of tinnitus sound.

Thus far, a technical approach to the research and treatment of tinnitus makes use of a variety of neuromodulation techniques, such as Neurofeedback (NFB) [10], transcranial Electrical Stimulation (tES), transcranial Repetitive Magnetic stimulation (tRMS), Deep Brain Stimulation (DBS), vagus nerve stimulation (VNS) [11], and VNS paired with tones [7].

According to Krog, et al., people suffering from tinnitus have higher stress levels and lower self-esteem. Over a larger population, the effects are present but weak [12]. However, 1% of the population is suffering from tinnitus disorder with psychological comorbidities such as stress, anxiety, insomnia, and depression [13]. Finding a cure for tinnitus would release a lot of stress on a large amount of the world population and even take away a factor of depression for some people.

Prof. Dr. Dirk de Ridder has been a forward-thinking neurosurgeon and researcher who has investigated tinnitus. In 2015, he studied the possibility of vagus nerve stimulation paired with tones. Over the last eight years, his understanding of tinnitus pathology grew, leaving him with several hypotheses. To investigate his hypotheses, he wishes to have a Multimodal Stimulator device, in which acoustic stimulation, transcutaneous electrical stimulation, and transcranial electrical stimulation can be given with a controlled time delay. Due to the feasibility of the master thesis project, only the acoustical and transcutaneous electrical stimulation are considered, resulting in a Bi-modal Stimulator. This master thesis reports such device's design, manufacture, and test process. If testing the hypothesis with the Bi-modal Stimulator brings satisfactory results, the theory behind the hypothesis, the design and manufacturing of the device could not only cure tinnitus but possibly be translated to other modalities and

brain disorders.

The main question for this thesis: *How can a user-friendly, portable, flexible, extendable, and safe bi-modal stimulator for tinnitus be designed as an experimental platform for translational research to effectively stimulate multiple modalities simultaneously, allowing for the testing of a wide range of hypotheses?*

In Chapter 2, a literature study will explain the theoretical framework for why a multimodal stimulator will contribute to the investigation of tinnitus pathology. The auditory system, neurological pathways related and unrelated to hearing, brain networks, tinnitus, neuromodulation, and neuromodulation for tinnitus treatment will be discussed. Chapter 3 will explain the requirements and methods used to design and manufacture the device, as well as the functions, software, safety, and user interface of the device. Chapter 4 will show the results, such as the device, the effects on the outputs of user interface changes, the user interface test, and the demonstration at the BRAI3N clinic. In Chapter 5, the conclusion, contributions, and recommendations are presented.

2

Literature study

2.1. The auditory apparatus and the neurological pathways related to hearing

To better understand mechanisms by which tinnitus manifests itself, it is essential to get some basic understanding of the anatomy and physiology of the auditory system, including the ear and neurological auditory pathways.

2.1.1. Anatomy and physiology of the ear

Hearing is one of the five major human sensors. The ears are responsible for transducing sound into neural action potentials, comprehensible to the human brain.

Sound, as a physical phenomenon, is the propagation of pressure waves through a medium, often a liquid or gas. The waves consist of alternating areas of high and low pressure, which is a high or low concentration of particles. Every wave is defined by its wavelength (λ), defined as the distance between two points of high pressure, and its propagation speed (v_{sound}), defined by the medium it travels through. The frequency (f) is defined as the number of waves passing a point per second and is inversely proportional to the time period (T) it takes for one wave to pass that same point.

$$f = \frac{v_{sound}}{\lambda} = \frac{1}{T} \tag{2.1}$$

Sound denotes exclusively the audible spectrum of this phenomenon when waves alternate with a frequency between 20Hz (very low-frequency sound) and 20kHz (very high-frequency sound). The pressure difference between high- and low-pressure areas defines the sound intensity or sound amplitude, with a higher difference meaning a higher intensity. In reality, audible sound consist of multitudes of overlapping waves, with different frequencies and amplitudes, all reaching our ears at the same time. It is a natural wonder that the ear and brain are able to distinguish all those frequencies at the same time, encoding and decoding the sound into comprehensible information, such as music or speech.

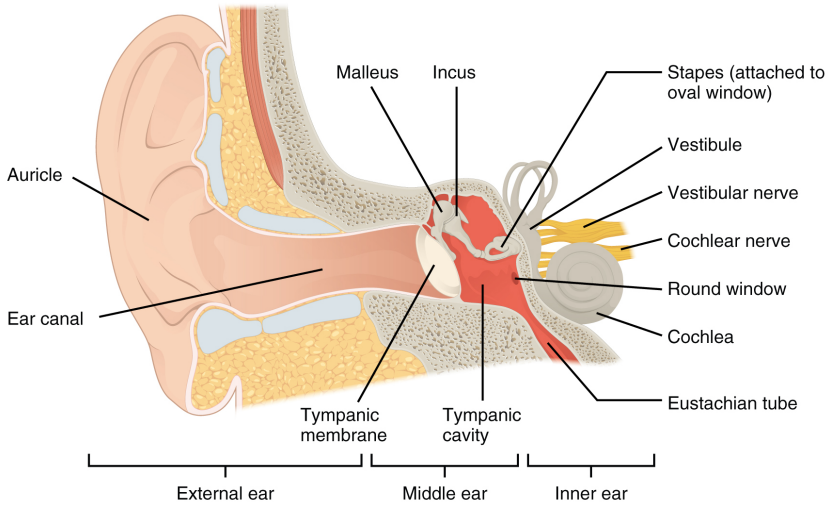


Figure 2.1: General anatomy of the ear, highlighting the outer ear, middle ear, and inner ear with the cochlea and the vestibulocochlear nerve (CN VIII) connected. [14]

The ear consists of three different parts: the outer ear, middle ear, and inner ear, see Figure 2.1. Sound waves hit and reflect on the pinna, eventually entering the ear canal, both part of the outer ear. In the middle ear, the pressure wave hit the tympanic membrane, vibrating the membrane and propagating the vibrations to the ossicles (or ear bones): the malleus, incus and stapes. A muscle, the tensor tympani, is connected to the malleus, dampening loud sounds if contracted, but not protecting the ear from damage of unexpected sounds. The stapes is connected to the cochlea at the round window. The round window guides the sound wave into the perilymph, the liquid of the cochlea.

The cochlea is spiraling structure with three tubes, the scala vestibuli, scala media and scala tympani, see Figure 2.2. Sound waves propagate all the way through of the scala vestibuli up to the apex of the cochlea where it is connected to the scala tympani. From there it propagates back to the base of the cochlea where pressure is released into the middle ear at the round window.

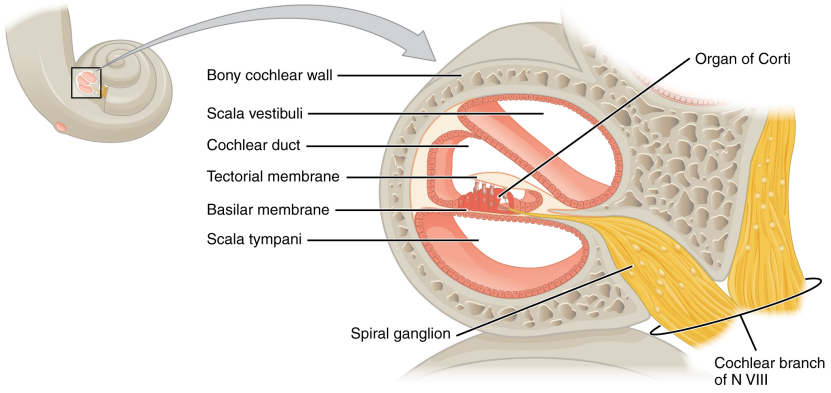


Figure 2.2: A cross-section of the cochlea [15]

Depending on the frequency, the pressure waves can also directly cross over from scala vestibuli, and scala tympani by passing the scala vestibuli and organ of Corti. The scala vestibuli and scala media are divided by Reissner's membrane, similarly, the scala media and scala tympani are divided by the basilar membrane and organ of Corti.

The organ of Corti is responsible for transducing pressure waves into neural signals, which are propagated through the cochlear nerve further into the brain. It consists of efferent outer hair cells and afferent inner hair cells. Only the outer hair cells are connected to the tectorial membrane which is attached to Reissner's membrane. They amplify the vibrations in the tectorial membrane. The outer hair cells also receive efferent signals from the superior olivary nucleus, which makes it possible for the

CNS to modulate auditory impulses at the receptor level [16]. Solely, the inner hair cells are responsible for transducing the vibration of the hairs into action potentials. The spiraling shape of the cochlea works as a tonotopic map, transducing low-frequency sounds at the base, and high-frequency sounds at the apex of the cochlea [17].

2.1.2. Neuroanatomy of neurological pathways exclusively related to hearing

Following the transduction from sound waves to neural action potentials, neural information is transmitted from the cochlea to the auditory cortex, through a series of nerves and neural nuclei. This series of auditory nerves and nuclei is called the ascending auditory pathway.

The auditory nerve enters the brainstem at the Cochlear Nucleus (CN). From there, multiple parallel pathways ascend through the brainstem, some of which crossover to the contralateral (opposite) side and others that stay on the ipsilateral (same) side from where the auditory nerve enters the brain stem, see Figure 2.3. The crossing over of information is used to compare stimuli from both ears. This way, the location of the audio source can be decoded, relying on small time delays and minute differences in the frequency spectrum. Similar to the cochlea, the CN, auditory pathway, and Auditory Cortex (AC) are all tonotopically organized.

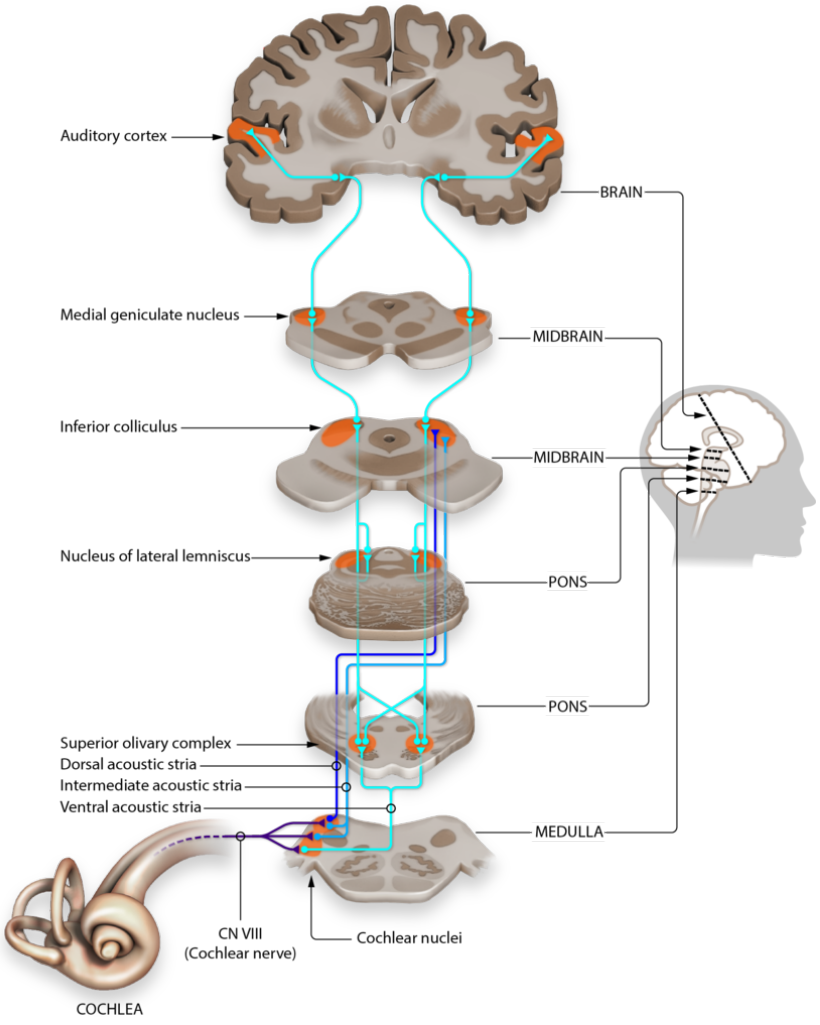


Figure 2.3: The ascending auditory pathway. Schematic representation of the different parallel pathways that originate from one side and terminate at the primary AC (A1). [18]

The primary auditory pathway runs contralateral to the AC. From the CN, nerves cross over to the contralateral Superior Olivary Complex (SOC) in the brain stem. From there the pathway continues to the Medial Geniculate Nucleus (MGN) in the thalamus, whereafter it enters the AC. The secondary

pathway goes from the CN directly to the ipsilateral SOC. Through the lateral lemniscus (nLL), the inferior colliculus (IC) in the midbrain is reached. There are also other pathways crossing back to the ipsilateral side from the IC in the midbrain to the MGN in the thalamus. The primary pathway continues to the AC in the lateral cortex. [19].

In addition to the ascending pathway, there is also a descending auditory pathway. Which have cortico-thalamic and cortico-collicular connections. Additionally, some pathways also descend from the brain stem to the outer hair cells, in order to modulate the resonance frequency of the tectorial membrane, as was mentioned in Section 2.1.1 [19].

2.2. The basic principles of neurophysiology, neuroplasticity and brain networks

The body makes use of multiple nervous systems to process and transmit information, including the Central Nervous System (CNS) and peripheral nervous system.

The CNS consists of the brain and spinal cord, covered by the meninges. The spinal cord is responsible for transmitting information from the body to the brain and vice versa. The brain processes information such as movement, sensation, thought, memory, and emotion. The CNS is connected through the cranial nerves and spinal nerves to the peripheral nervous system.

The peripheral nervous system is divided into the somatic nervous system and the autonomic nervous system. The somatic nervous system is responsible for voluntary movements and sensory information, including the motor and sensory neurons. The autonomic nervous system is responsible for involuntary functions, such as the heart rate and digestion. The autonomic nervous system is subdivided into the sympathetic and parasympathetic nervous system, responsible for the 'fight or flight' response in case of danger and the 'rest and digest' response in case of relaxation, respectively [20].

2.2.1. Neurophysiology

To better understand some of the subjects in the subsequent sections, we will go into a little more detail on neurophysiology. On a micro level, these nervous systems consist of neurons. Neurons can be divided into different neural modalities. The propagation direction is indicated by afferent (sensory) or efferent (motor) neurons. The anatomical distribution can be visceral (to or from organs) or somatic (to or from nonvisceral body parts). The last division is based on the neuron's embryological origin, general or special neurons.

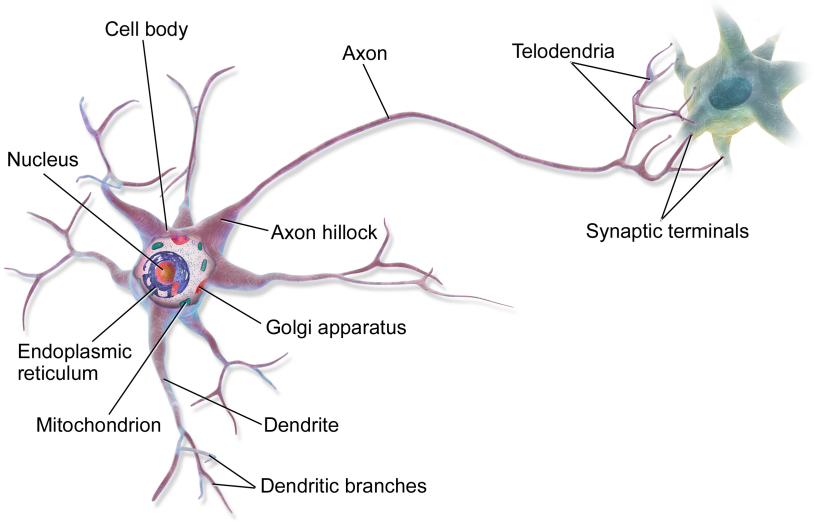


Figure 2.4: Schematic of a neuron, including synapse [21]

Neurons consist of a cell body, called the soma. Originating from the soma, every neuron also has dendrites and one (relatively) large axon, see Figure 2.4. The area where the axon of one neuron interacts with the dendrite of another neuron is called the synapse. If an axon is unmyelinated, the

action potential propagates in one motion from the soma to the synapse. If the axon is myelinated, the action potential propagates from the node of Ranvier to the next node of Ranvier. Myeline increases the propagation speed along the axon. At the end of the dendrites and the axon, a synapse connects one neuron to another neuron, or a target cell, such as a muscle. The action potential is transduced at the synapse into an electrochemical messenger and transduced back again to an action potential at the next neuron.

An action potential consists of four phases: depolarization, repolarization, refractory period, and resting state. In resting state, the neuron has a potential of around -70 mV, which can fluctuate to a maximum of -55 mV. -55mV is the threshold for an action potential to depolarize. At depolarization, the action potential crosses the threshold and peaks at +40 mV after 1 ms. After the peak, the repolarization phase brings the polarity down to the refractory period, which peaks is under -70 mV, and ends in the resting state at -70 mV after 5 ms. The intensity of a neurological signal is encoded in either the frequency of successive action potentials or the summation of action potentials arriving from multiple other neurons. An action potential has a temporal length of 5 ms. The arrival of successive action potentials has a higher duration to indicate its intensity, than the arrival of a summation of multiple action potentials from different neurons [20].

2.2.2. Brain networks

Functions of the brain are a result of an interplay of brain networks consisting of a multitude of neurons. The organization and behavior of brain networks are not regular, nor completely random, and are changing over time. Over decades, research has been conducted with the aim to understand both the micro-, meso-, and macro-levels of the physiology of the brain. Those working principles are often imitated by mathematical models and theories, such as biophysical models, network models, and information theory. Lynn and Basset wrote a great review about the development of the understanding of brain networks from the perspective of statistical mechanics and mathematics [22]. They explore brain network structure, brain network function, and recording of brain networks.

It is not sufficient to understand the working principle of a single neuron, the physical structure of the brain, or the individual locations where groups of neurons fire alone to understand the working principle of the brain. This section will start with defining various brain structures based on network theory. After the structures, the function of brain networks comes into play, combining brain recording techniques with mathematical models. This will give a better understanding of the interplay of brain networks and their functions.

Brain structures

Graph theory is used to model the structure of brain networks. Graph theory uses nodes as neurons or brain regions and edges as the physical connection between them. The edges are representations of synapses for communication between neurons [19] or white matter for communication between brain regions [23]. Network models are differentiating topologies, such as random structure, community structure, small-world, a hub- and spatially embedded structure. Different topologies in brain networks have been linked to various behaviors [24].

As the name already hints, a random topology is characterized by a random chance of one node connecting to another node. Over a multitude of nodes, this will result in an unstructured network of connections between nodes. This can be mathematically modeled by the Erdös-Rényi model [25] which uses a fixed probability to create connections between nodes.

The complete opposite is a spatially embedded structure, in which all nodes are equally distributed and connected to each other. This is constrained by the physical barriers of the brain [22].

In between lies the small world topology, characterized by a large amount of clusters and a short average path length. Most nodes are connected to their direct neighbor, but some are connected to a neighboring cluster. This results in a balance between local and global connectivity, and efficient information integration and segregation at the same time [26]. The efficiency of the brain is a constant trade-off in cost-value, in which the cost is the wire and the value is the information propagation [27]. To match this trade-off as well as possible, the brain is considered to be a small-world topology [28].

These network models only take into account the connections between nodes in terms of information but do not take into account the functional, spatial, and temporal representation of the brain. Brain structures differ from human to human, and even change over time, as the brain develops itself during a lifetime.

Brain recordings

Complementary to the understanding of structural brain networks, network science allows neuroscientists to investigate functional brain networks. Functional brain networks take the dynamics of brain regions into account. To create functional brain network models, the recording of brain activity is inevitable, as the relationship between brain structure and function is highly nonlinear [29]. Over time, multiple brain recording techniques are used, of which electroencephalography (EEG) and blood oxygen level-dependent (BOLD) functional magnetic resonance imaging (fMRI) are the most used by neuroscientists. The choice of using one over the other is always a trade-off between a high temporal resolution in EEG and a high spatial resolution in BOLD fMRI.

In EEG recordings, surface potentials of the scalp can be recorded by placing electrodes on the scalp. The potentials are very low, in the range of micro voltages. The recording of these potentials is called EEG recording. A standardized 10-20 system uses 21 electrodes, distributed over the scalp. The recorded potentials are generated by synaptic firing inside the scalp. Due to an orchestra of neurons firing, the potentials are considered stochastic signals. From these stochastic signals, different low-frequency bands can be extracted by correlation techniques, filtering the noise from the signals. These frequencies are often called brain waves [30]. The brain waves in the alpha band (8-13 Hz) are recorded at the occipital region and linked to the resting state with eyes closed. The beta band (13-30 Hz) is linked to an active brain state, recorded over the parietal and frontal lobes. Delta waves are in the frequency band of 0.5-4 Hz and theta waves have a frequency band of 4-8 Hz. Both are linked to sleeping adults [31]. EEG is a low-cost non-invasive technique, which is used to deviate between a normal and pathological functioning brain.

BOLD fMRI is a technique that takes into account certain physiological phenomena related to neural activity and especially certain blood qualities. Cerebral blood flow has been associated with neuronal activity. For pre-synaptic firing, energy is needed in the form of oxidative glucose consumption. The metabolic rate of glucose and blood flow are closely coupled. The metabolic rate of glucose is the first parameter to detect neural activity using fMRI. In a resting period of neural activity, the neurons are in need of energy provided by a blood flow through the widening of capillaries. In a period of rest of neural activity, the capillaries are filled with blood. Due to a larger inflow of blood than the consumption of oxygen, the oxygen levels in the blood increase. In general, oxygen levels in the blood are thought to be related to the metabolic rate of glucose. This is not the case in cerebral blood flow. The difference in the oxygen levels in the cerebral blood flow on one hand, and the capillary and venous oxygenation levels on the other, create a second parameter for the detection of neural activity using fMRI. Due to the high oxygen levels, oxygen bounds to deoxygenated hemoglobin resulting in oxygenated hemoglobin. The ferrous iron in hemoglobin becomes more diamagnetic when more oxygen is bound to the hemoglobin. Thus oxygenated hemoglobin is more diamagnetic than deoxygenated hemoglobin, which results in a contrast difference on fMRI images, creating markers of neural activity. This technique is called BOLD fMRI [32]

Brain functions

To understand brain functions as one integrated model, it is inevitable to first understand the dynamics of individual components such as neurons and brain regions, before investigating the connections between individual components and in the end creating models that give rise to the understanding of brain functions.

In an attempt to understand brain networks, some of the mathematical models are used as artificial neuron models, mimicking brain functions, while others are purely biophysical models, used for the understanding of physiology.

An artificial model for propagation of action potentials from pre- to postsynaptic neurons is designed by Warren S. McCulloch and Walter Pitts in 1943, taking into account multiple binary inputs, and a binary output based on linear weights, implementing the ”all-or-none” principle. A fixed amount of presynaptic neurons should fire within a certain amount of time for the postsynaptic neuron to fire. Only the synaptic delay is taken into account throughout the whole nervous system. The neuron can be inhibited by at least one inhibitory presynaptic neuron firing. The last prerequisite is that no neuroplasticity (see Section 2.2.3) takes place in the network [33]. This model gives insight into the propagation of action potentials through multiple neurons and was improved on by Frank Rosenblatt, which showed that the weight of the neuron could be altered based on prior information. This algorithm is known as the perception, which lays a basis for support vector machines, neural networks and machine learning.

On the microscale, a biophysical model for the generation and propagation of an action potential in a neuron, including the opening and closing of ion channels, is mathematically modeled by the Huxley-Hodgkin model, developed in 1952 [34]. It describes the dynamics of an action potential based on the calcium-sodium, potassium, calcium, and chloride ion channels. The model has been a very useful basis in the understanding of the communication between neurons and created a basis for the understanding of how the brain processes information Additionally it has been used to investigate other aspects, such as oscillations in activity and stochastic dynamics [22]. The FitzHugh-Nagumo model is a concurrent model that is useful for simulating the dynamics of large groups of neurons. It models the neuron using an electrical circuit as a transmission line using tunnel diodes. The model characteristics in which it has a threshold value, line shapes signal waveforms, and bi-directional transmission where colliding signals vanish [35][36][22].

The activity of a single, or a small number of neurons, does not represent functions on the level of the whole brain. The models in between the micro and macro levels are called neural mass models, based on the Wilson-Cowan model of population dynamics [22]. The Wilson-Cowan model adds inhibitory actions to the excitatory actions, enabling the investigation of neural oscillations, which are a limit cycle activity of spatially localized populations of neuron models. These limit cycles can emerge by different stimuli, showing a stable state for the whole neuron population [37].

The investigation of macro-scale regional networks is done by combining multiple neural mass models into a network, representing the whole brain. This approach gave insight into large frequency band oscillations recorded with EEG. The coupling of neural mass networks results in phase-locked oscillations between different neural masses [38], which seems to represent synchronous neural firing of brain regions. The rhythms produced by the bidirectional coupling of the neural masses depend on the coupling strength and phase delay [38].

Synchronization of oscillatory chemical and biological systems is described by the Kuramoto model [39]. The Kuramoto model has a multitude of phase oscillators that oscillate at different natural frequencies and are independent of each other. It takes into account the grade of coupling between oscillators, which at a certain threshold, results in synchronization of the oscillators [40]. Investigating the Kuramoto model as a simplification of a network of neural masses gave insight into the spontaneous synchronization of brain regions. Also, local synchronization between clusters of neurons occurs, while no synchronization is found on the level of brain regions. These phenomena are also found in fMRI recordings [41].

2.2.3. Neuroplasticity

Neuroplasticity or brain plasticity is an umbrella term for the ability of neurons or brain structures to build or adapt to changing intrinsic or extrinsic factors resulting in structural or functional changes. The reorganization of brain functions depends on the plasticity of both individual neurons and clusters of neurons [42].

The brain, as a self-organizing system, uses modularity, redundancy, and degeneracy to make sure it maintains the same or similar functionalities. Modality represents separate building blocks of neuron clusters. Redundancy means that a function can be executed in a multitude of ways, by identical elements. Degeneracy, on the other hand, means that distinct elements can execute the same functionalities.

These principles together create a robustness of the system, in which the functions are executed with small perturbations. To maintain this robustness, the brain is dependent on neuroplasticity. Plasticity can be divided into structural- and functional reorganization on a micro (neuron) and macro (brain region) level [43].

Functional plasticity

On a micro-scale, synaptic plasticity is a form of functional reorganization of the neuron. In general, synaptic plasticity is a mechanism in which a postsynaptic neuron's effective susceptibility to action potentials from the presynaptic neuron changes. The change in synaptic strength is commonly described by Hebbian and homeostatic plasticity.

Hebbian plasticity is the strengthening or weakening of synaptic connectivity due to the activity of the pre- and postsynaptic neurons. If the pre- and postsynaptic neuron fire at the same time, it increases synaptic strength, and can even induce another Hebbian plastic event in a positive feedback loop. Hebbian plasticity can occur in a matter of seconds.

Homeostatic plasticity works as the counterbalance of Hebbian plasticity. Opposite to Hebbian plasticity, it aims to stabilize neural networks. It is a much slower process, in which the amount of ion channels in the synapse changes, and thus the responsiveness to incoming action potentials decreases [43].
In the case of Hebbian plasticity, the firing of the pre-synaptic neuron followed by the postsynaptic neuron increases the synaptic strength. On the other hand, experiments indicate that the neuronal firing of the postsynaptic neuron followed by the presynaptic neuron decreases the synaptic strength. The combination of these two principles is called spike time dependent plasticity, while still considered "Hebbian plasticity" [44].

Structural plasticity

Structural reorganization on a microscale is related to the structural changes of neurons, such as synapse remodeling and neurogenesis. Synapses can be remodeled spontaneously or reactively. Dendritic spines and axonal varicosities are considered potential synapses, of which a few are able to grow into a synapse. Presynaptic terminals can also degrade due to dysfunctionality. This is reversible if the axon terminal is not degraded. Synapses can also be rerouted to other targets as far as 300 µm away. Reactive synapse remodeling can occur after a lesion. The intact fiber tracts can sprout towards postsynaptic targets. These methods are not well understood in the CNS. However, in the peripheral nervous system, target cells release neurotrophic factors to attract axons. Also, sensory deafferentation can lead to remapping of the somatosensory representations. In the lesion area, deafferented neurons seem to change their susceptibility by synaptic rewiring, and increase the unstable spines [45]. Another form of structural plasticity at the micro level is neurogenesis. New granule cells emerge for example the dorsal cochlear nucleus (DCN). The granule cell needs to form new synapses to connect to an existing network of neurons. Depolarization of young neurons seems to increase the chance of survival. For structural plasticity and network rewiring in the hippocampus, neurogenesis is considered as an important factor [45].

Cortical reorganization is a form of structural reorganization on a macroscale in which the connectivity, excitability, and activity of motor- or somatosensory cortices change in structure. In the case of the somatosensory cortices, cortical reorganization often happens due to a change in sensory input being increased or decreased. This can result in larger regions for specific brain parts [43]. Structural connections between brain regions are evolving over a matter of days to years. These connections change due to development, aging, and experience-dependent plasticity [46].

Developmental reorganization, a form of structural reorganization, appeals to the imagination as every child goes through it. In the neocortical development of mice, neurons in different local clusters fire spontaneously and synchronously during the first four days after birth. After two weeks, the neurons start firing desynchronized, suggesting that the plasticity between the neuron clusters is done, and functional connections are made [47]. This spontaneous firing is also found in humans in resting state, after visual training. The brain networks used for the visual task were also firing spontaneously when the person is at rest. The function of spontaneous firing at rest is to consolidate the newly learned task [48]. This is related to the learning mechanisms of the brain, which are discussed in section 2.3.1.

Learning mechanisms

Cortical map expansion is present in highly trained humans. The sensory or motor cortex can expand in size due to a lot of training on processing sensory input or motor output. This is believed to be an indicator of learning ability but is not necessary to optimize specific sensory inputs or motor commands [49]. During the training itself, the cortical map expands and normalizes in size again after training. Kilgard proposed a model to describe this phenomenon, based on a Darwinian perspective [50].

A well-known behavioral reconditioning technique is Pavlovian Reconditioning, where an unconditioned stimulus resulting in a response is paired with a conditioned stimulus. If this is repeated, the response to the conditioned stimulus becomes part of the behavior of an animal or human [51].

2.3. Neurological pathways non-exclusively related to hearing

In Section 2.1.2, the neurological pathways directly related to hearing are briefly discussed to give an overview of the processing of sound to the auditory cortex. However, there are more neurological pathways and brain regions indirectly related to hearing, which will create a basis for tinnitus pathology. These are introduced in Section 2.4. This section will introduce the concept of the Bayesian brain as

a prediction model, the ascending, descending, and medial pathways related to the Bayesian brain concept, and the Triple Network Model (TNM) as an all-encompassing theory describing why some auditory inputs are important, ignored, stored, or used.

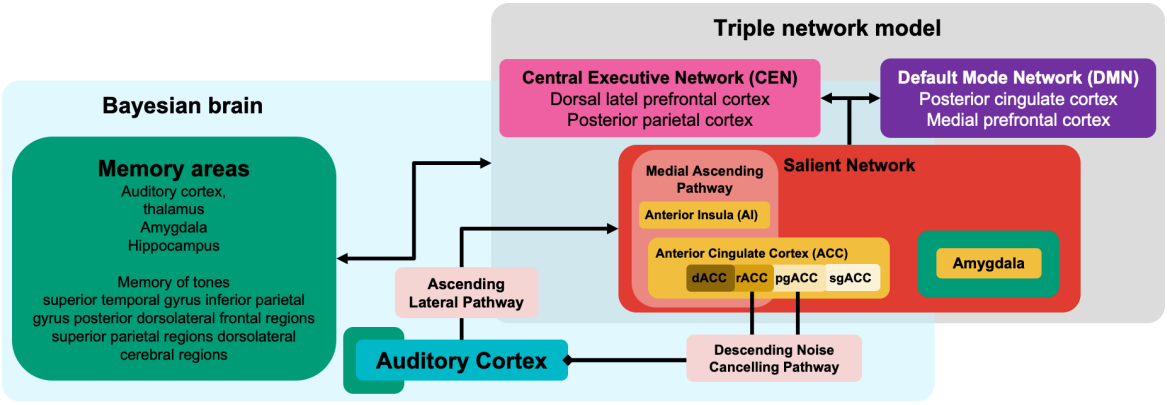


Figure 2.5: An overview of the neurological pathways, principles, and networks non-exclusively related to hearing

2.3.1. Bayesian brain

The Bayesian theory is a probability and prediction theory originating in mathematics. However, the theory can be and has been applied in many scientific fields, among which neurology, leading to the Bayesian brain model. The Bayesian brain model is used to explain how the brain processes information and how it uses this information to act accordingly. As such, it has also been an important theory for explaining the processing of auditory information and more specifically in explaining tinnitus, as will be discussed in Section 2.4.5.

According to this theory, the brain is a prediction model, that predicts how the world works based on all prior information. The better these predictions are, the better the brain is able to respond to things that happen in the environment. Therefore, new information in the form of sensory inputs, such as auditory stimuli, is used to update the prediction model and make better predictions in the future [52]. However, not all sensory inputs are equally useful for updating this prediction model. Logically, only unexpected or unpredictable inputs contain new information and can therefore be used to update this model. Consequently, the brain needs to be able to make a distinction between predictable and unpredictable stimuli.

The EEG measurements of the Brodmann areas have identified the functions of the Bayesian brain theory across different frequency bands. The Brodmann areas are the cytoarchitectonic subdivisions of the human cerebral cortex, used in anatomical brain mapping [53]. The accuracy of auditory predictions is represented by delta-beta coupled oscillations, which fall within the lower frequency range. On the other hand, prediction updates are represented by alpha-band oscillations, which fall within the higher frequency range. The low-frequency delta and theta oscillations are responsible for the time prediction of an auditory stimulus, while the gamma and beta oscillations, which fall within the higher frequency range, represent the type of auditory stimulus. By segregating the functions based on frequency bands, the Bayesian brain theory can be better understood and studied. [54].

2.3.2. Pathways

Three different pathways are associated with the Bayesian brain theory. The lateral pathway, the medial pathway, and the descending pathway. All three are related to tinnitus, but not exclusively to tinnitus. The (ascending) lateral pathway is a pathway that is anatomically related to the somatosensory and auditory system. It is responsible for transporting sensory and auditory inputs to the sensory and auditory cortices respectively. It encodes the intensity and characteristics of the sensory inputs, and the intensity and the affective component of the auditory inputs. The medial pathway is a non-specific pathway responsible for the motivational or salience of the stimuli. It is connected to the dorsal Anterior Cingulate Cortex (dACC) and anterior insula. It encodes how relevant a stimulus is. The descending pathway is an inhibitory pathway, which is connected to the rostral ACC (rACC), subgenual ACC

(sgACC), and pregenual ACC (pgACC). This pathway balances both the lateral and medial pathways, maintaining the correct reference in the Bayesian brain concept [5].

2.3.3. Triple network model

The TNM is proposed by Vinod Menon [55]. The aim of the model is to investigate cognitive and affective dysfunction in psychiatric and neurological disorders in large-scale brain networks. The network approach of neuroscience, as discussed in Section 2.2.2, gave significant insights into psychopathology. The investigation of brain architecture, neurocognitive networks, and several neurological disorders gave rise to a unifying 'triple network' model. It is a framework that helps understand various disorders, such as autism, schizophrenia, depression, and anxiety. The three networks are the central executive network (CEN), the salience network (SN), and the default mode network (DMN) [55].

The CEN is anchored in the dorsal lateral prefrontal cortex and posterior parietal cortex [56]. As the name suggests, the CEN is used for processing information, problem-solving, and decision-making. It controls goal-oriented behavior [57]. The opposite of the CEN is the DMN, which is switched off during goal-oriented processes and activates during mind wandering. The activation of the DMN occurs at rest and is associated with the self. It is anchored in the posterior cingulate cortex and medial prefrontal cortex, with some nodes in the medial temporal lobe and angular gyrus [58]. The SN is a network responsible for the filtering of salient stimuli, including external stimuli and emotional stimuli. It works closely together with the CEN, as the SN decides if the CEN should become active after the presentation of certain stimuli. The SN includes the dorsal anterior cingulate cortex and frontoinsular cortex. It also includes the amygdala and substantia nigra or ventral tegmental area, which is responsible for the processing salience of emotional and reward [56]. The SN controls the activity of the CEN and DMN and activates the CEN after the presentation of a salient stimulus. Subsequently, activity between SN and CEN is always correlated, and the activity of the DMN is always anti-correlated with the other two networks [59].

2.3.4. Memory areas

The brain can reproduce audio from memory as if images are visualized. Auditory memory is distributed over multiple locations in the brain, such as the auditory cortex, thalamus, amygdala, and hippocampus. The amygdala is the region responsible for the emotional reaction to presented audio, determining the emotion labeled to an auditory memory. The hippocampus both stores auditory memories and compares new auditory information to the earlier stored auditory information [60]. The parahippocampus is responsible for the retrieval of auditory memories into consciousness. If this is not functioning correctly, it can result in auditory hallucinations [61]. In auditory memory, pitch is memorized in the superior temporal gyrus, inferior parietal gyrus posterior dorsolateral frontal regions, superior parietal regions, and dorsolateral cerebral regions [62].

2.3.5. The vagus nerve

The vagus nerve, often denoted by cranial nerve X (CN X), is a nerve connecting both the parasympathetic and sympathetic systems. Cranial nerves are peripheral nerves which can be motor, sensory, or innervating nerves emerging from the spinal cord and exit the CNS from the brain stem. CN X has two motor components and three sensory components. The general motor components originate from the dorsal motor nucleus, connecting to the parasympathetic system and activating the organs such as the trachea, lungs, heart kidneys, and stomach. Special motor components activate several muscles in the throat and the superior cardiac nerve. These motor components are activated for speech, the breath, the pumping of the heart, and spreading up peristalsis [63]. In particular, the right vagus nerve ending in the tragus has efferent components to the heart [64]. The sensory components are connected to the throat, aorta, and ear. Most sensory inputs of CN X do not reach consciousness except for some, which give positive feelings. Also, the respiratory functions are activated by the information about the blood in the aortic body. Some fibers are connected to the throat, and used as tastebuds. Some sensory components are positioned in the external auditory ear canal and outer tympanic membrane. The cell bodies are positioned in the superior ganglion of the vagus nerve. Inside the CNS, these fibers connect to the spinal tract. The spinal tract is also connected to the trigeminal nerve (CN V). The spinal nucleus is an area right next to the spinal nucleus and also connected to CN V, thus the CN X becomes part of the trigeminal system, propagating through the ventral posteromedial nucleus of the thalamus to the cerebral cortex [63].

2.4. Tinnitus

As mentioned in Chapter 1, tinnitus could be divided into subjective tinnitus and objective tinnitus. Objective tinnitus is sound produced by the body and received by the ears via internal vibrations. However, subjective tinnitus does not have a physical origin and is perceived in the brain. This thesis only considers subjective tinnitus. The term tinnitus has been proposed to represent solely the phantom sound perceived by the brain, which can be a side effect of for example trauma. If tinnitus is associated with suffering, it would be considered as a tinnitus disorder. The proposed distinction is formulated as follows: *"Tinnitus is the conscious awareness of a tonal or composite noise for which there is no identifiable corresponding external acoustic source,* with tinnitus disorder component *"when associated with emotional distress, cognitive dysfunction, and/or autonomic arousal, leading to behavioral changes and functional disability."* [13].

Historically it was thought that the ear was the sole cause of the generation of tinnitus [65, 66]. This hypothesis initially makes sense since tinnitus correlates with hearing loss and damage to the ear. However, dissection of the auditory nerve most often leads to tinnitus and does not remove it when already present [67, 68]. Thereby indicating that neither the cochlea nor the auditory nerve are in itself the place of origin for the generation of the tinnitus percept. In addition, early neurological studies indicate that tinnitus patients show neurological changes in both cortical and subcortical regions [69, 70].

Indeed, currently, the generation and progression of tinnitus is thought to be a complex multifactorial process [71]. Although research is ongoing and a consensus is not been reached yet, causes for tinnitus generally consist of one or more of the following factors: (1) lack of auditory input, triggering (2) overexpression in auditory neurons, where after (3) the tinnitus signal becomes wrongly labeled as important and processed into the conscious parts of the brain (4) and generation is unsuccessfully suppressed in due course [5, 72]. Most recent theories on the generation and persistence of tinnitus integrate these theories. The integration combines three pathways plus some additional brain areas and networks that interact with these three pathways and influence the course of progression of tinnitus [5, 6, 73]. We will take the time to explain the most important theories and findings using the three previously mentioned pathways as a framework.

2.4.1. Types of phantom sounds

Tinnitus or phantom sounds can come in a large variety. Tinnitus is mostly described as a constant beep referring to tonal tinnitus. Tonal tinnitus is the perception of single or multiple pure tones. Most people would recognize this in the non-chronic variant of tinnitus, which often occurs after visiting a club or concert with loud music. This non-chronic variant attenuates over time. Other tinnitus sounds are described as hissing or cricket-like sounds [13]. Tinnitus can also be a buzzing, clicking, or pulsing sound. The sound can be unilateral or bilateral, with an intermittent or pulsatile characteristic. The loudness can vary between a subtle sound just above the hearing threshold or a high intensity [71].

2.4.2. Causes for tinnitus related to ascending auditory pathway

The ascending auditory pathway is responsible for the conversion of sound into comprehensible auditory information for the brain as explained in Section 2.3.2. Along this pathway, multiple factors are suspected to be able to cause tinnitus, such as sensory deafferentation, overexpression in the DCN, and the influence of somatosensory nerves.

Sensory deafferentation due to cochlear damage

Sensory deprivation is the loss of sensory input to the brain. A specific form of sensory deprivation is sensory deafferentation, which occurs when the sensory nerves are deprived of stimuli. Therefore sensory deafferentation inhibits the propagation of sensory information from the periphery to the CNS.

Hearing loss, also known as (detectable) auditory deprivation, is most often a consequence of sensory deafferentation. Cochlear dead regions, which are regions of the cochlea that are not able to transmit auditory information due to inner hair cell damage or non-functioning neurons are one of the main causes of auditory deafferentation [74].

There is a strong relationship between hearing loss and tinnitus sufferers being subjected to hearing loss [4, 75]. Not everyone with tinnitus suffers from hearing loss; however, cochlear dead regions are often detected in people with tinnitus who do not suffer hearing loss [76–78]. Small amounts of cochlear dead regions and small changes in hearing thresholds cannot be reliably detected. Therefore,

it has been hypothesized that tinnitus is always accompanied by at least some degree of sensory deafferentation, wherein sensory deafferentation is an essential trigger for the development of tinnitus. This relationship between tinnitus and hearing loss is much less obvious in the other direction since most people who suffer from hearing loss do not suffer from tinnitus [75]. Therefore, although sensory deafferentation might be essential for the development of tinnitus, it does not lead to tinnitus most of the time.

Maladaptive plasticity causing overexpression in brain stem

If sensory deafferentation is the initial cause or trigger for tinnitus, it suggests that the lack of sensory inputs leads to over-activity or at least to neurological changes further into the auditory pathway. As mentioned in the introduction of this section (section 2.4), over-activity does not emerge from the cochlea itself or the auditory nerve, since removing the auditory nerve often induces tinnitus and tinnitus persists if it had been present before the removal [67, 68]. In addition, human audiograms, as well as animal studies, have shown that tinnitus is often present at the fall of frequency of hearing, which marks the boundary between frequencies that are still audible and non-audible for the specific individual [71, 79]. This also suggests that changes in the auditory nervous system are at play, while further cementing the relationship between hearing loss and tinnitus [79]. This leads us to the CNS as a possible origin for increased over-activity due to sensory deprivation in the case of tinnitus.

Indeed, tinnitus has been associated with the over-activity of neurons in the DCN and other subcortical structures in the brainstem. Several animal studies have been conducted that tested the effects of hearing loss and tinnitus on a cell level [79]. These studies would expose animals to loud sounds of a specific frequency, thereby inducing hearing loss. By using the so-called gap startle reflex, they were able to differentiate whether the animal in question had hearing loss, tinnitus, both, or neither [80]. These studies found that the Spontaneous Firing Rate (SFR) of fusiform cells in the DCN was significantly increased after noise exposure [4, 79]. Specifically, the neurons associated with the determined tinnitus frequency and the noise-exposure frequency showed increased SFR, even after normal hearing was restored in the case of temporary hearing loss. Similar increases may be present in the Ventral CN (VCN), but these are less well-studied. Similar results are found for increased SFR in the IC, the MGN, and the primary auditory cortex (A1) [79]. However, most of these studies show that activity in these subsequent areas is strongly related to activity in the DCN. This makes sense since these nuclei directly or indirectly receive their efferent input from the DCN thus marking the DCN as the origin of hyperactivity. Furthermore, tinnitus is also associated with an increase in the Burst Firing Rate (BFR) and synchronicity of fusiform cells in the DCN and subsequent structures up to the A1. Specifically, synchronicity is important because this could indicate the grouping of auditory inputs, thereby over-expressing the associated auditory stimulus, which could be a cause for the perception of a phantom sound [79].

These changes in neuronal activity are likely caused by neural plasticity explained in Section 2.2.3, which is in turn caused by the sensory deafferentation of the auditory inputs as explained in Section 2.4.2. More specifically, the changes are explained by homeostatic plasticity along the auditory pathway, most notably of course in the CN, but probably also in other subcortical and cortical regions, among which the AC [71, 81, 82].

In addition to the increase in SFR, BFR, and synchronicity in the A1, it is found that the tonotopical map in the A1 shifts due to hearing loss. When neurons are deprived of sensory inputs, for example, due to cochlear damage, the neurons will start to respond to neighboring neurons that still receive sensory inputs [73], see Figure 2.6. This in due course results in a shift in the tonotopic map, where especially frequencies near the edge frequency of normal hearing are overrepresented [83, 84]. This mechanism is most likely at play to combat the inevitable effects of hearing loss as aging progresses. Especially at higher frequencies, the shifting of frequencies at both the cochlea and the AC are less selective [73]. Thereby neurons can still receive auditory information from neighboring cochlear regions, compensating for the hearing loss.

The shift in the tonotopic map also results in overexpression of the edge frequencies and increases the synchronicity of the associated neurons. This has been argued that the shift in the tonotopic map is a major cause for the development of tinnitus [73]. In fact, this forms the basis for the treatment of tinnitus with Vagus Nerve Stimulation (VNS) in combination with auditory stimulation [85]. However, it has to be stated that tonotopic map reorganization might just be related to hearing loss and not per se to tinnitus [73, 86].

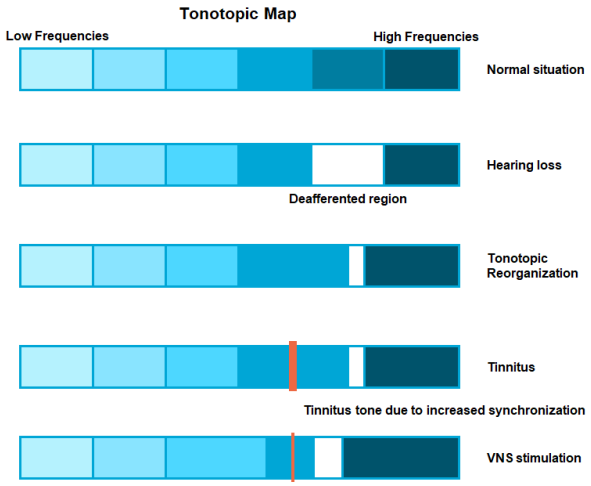


Figure 2.6: Schematic representation of the tonotopic map shift in the primary auditory cortex (AC1) due to sensory deafferentation. Each blue-shaded block represents the frequency range to which that specific set of neurons is tuned. (1) in a normal situation where no hearing loss is present the tonotopic map is equally spaced out. (2) deafferentation of one frequency range prevents sensory information from reaching the associated region (in white) in the AC1. (3) To solve the lack of incoming information, the neurons will tune themselves to sensory input from the neighboring regions. (4) However, this also leads to the over-representation of the neighboring frequencies (hyper-activity in the AC), which can give rise to a tinnitus percept (in red). (5) VNS stimulation in combination with audio stimulation is hypothesized to amplify the tonotopic representation of the frequencies used in the audio stimulation. Audio stimulation with all frequencies excluding the tinnitus frequency, is thus hypothesized to reduce the tinnitus-associated over-activity in the AC and thus the presence of the tinnitus percept.

Influence of somatosensory nerves

Temporomandibular joint disorder and neck injuries are both associated with tinnitus [87]. This implies the influence of the somatosensory system on tinnitus. Unsurprisingly, different nerves of the somatosensory system have nerve endings on different auditory structures in the brainstem. The CN V synapses indirectly on fusiform cells in the DCN [88]. Also, the dorsal column ganglia have projections on the DCN [88]. The reason why this is the case is still debated, but one of the reasons might be to compensate for body-own sounds.

The auditory nerves also have synapses on the fusiform cells, which thereby in turn integrates both auditory and somatosensory stimuli [89]. It has been shown in animal models that excitation of the CN V indeed influences excitation patterns in the fusiform cells [89–91] and other places up the auditory pathway up to at least the AC [92]. The influence of the somatosensory system increases in case of hearing loss. When the input of the auditory nerve to the CN reduces due to cochlear damage, the input of the somatosensory nerves increases [79, 93]. In the case of tinnitus, this integration was found to cause long-term potentiation of neurons in the CN, while in animals without tinnitus, this caused long-term depression [90].

It is hypothesized that stimulation of these somatosensory nerves, most importantly the CN V, might positively influence tinnitus in humans. However, this sensory integration in humans has thus far not been extensively studied. The LENIRE device, which will be covered in more detail in Section 2.6, uses this principle to excite the CN V on the tongue.

2.4.3. Medial pathway as a cause for tinnitus related distress

As explained in Section 2.3.3, the medial pathway for tinnitus (and sound processing) is related to the affective component of tinnitus [5]. Where the affective component is the person's emotional response or involvement to tinnitus, including both positive and negative reactions. The general purpose of the medial pathway in a healthy situation is still debated, but it is most likely a salient network, which labels sensory inputs as relevant or irrelevant and directs these stimuli to the conscious parts of the brain accordingly as was also described in Section 2.3.2 [5].

As with the SN, the medial pathway is not specific to tinnitus or auditory stimuli. It was first described in detail for pain. But is also described in relation to social exclusion, hunger, and thirst [5]. The existence of an auditory medial pathway was first suggested in 2011 [72, 94] and has been largely incorporated into the theorem on tinnitus just recently [5]. Based on earlier studies, the affective com-

ponent of sound processing, in general, has been related to activity in the amygdala, dACC, and insula [72].

The medial pathway for tinnitus is generally agreed to include at least the pgACC and rACC and the Anterior Insula (AI), which are also part of the SN. Sound processing in the ascending and medial pathways does not necessarily follow each other in sequence but may happen at least partly in parallel, as is the case with pain [5]. For sound, the distinction between the ascending and medial pathways is not fully clear and possibly overlaps partly. For example, the AC is also involved in affective processing [95] and the ACC and insula are also involved in loudness perception [96].

2.4.4. A noise-canceling descending auditory pathway

The possible involvement of a noise-canceling pathway responsible for suppressing the tinnitus percept was first coined by Laever and Rauschecker [97, 98]. The descending auditory pathway is suggested to be a noise-canceling pathway that suppresses activity in the lower brain areas such as the DCN. The pathway is suggested to originate in the pgACC, sgACC, and nucleus accumbens area and most likely projects onto the tectal longitudinal column, which connects to the olivary complex. From here the olivocochlear bundle transmits the inhibitory information to the cochlea [5]. From the perspective of the Bayesian brain, see Section 2.3.1, the pgACC of the descending auditory pathway confirms the tinnitus state as reliable, which is confirmed by the nucleus accumbens [5]. Inhibiting this connection can mark the tinnitus state as unreliable, reducing its conscious perception.

2.4.5. A Bayesian viewpoint of tinnitus

The Bayesian brain theory as described in Section 2.3.1 applies to tinnitus. The prior model of the world is updated by the lateral ascending auditory pathway, the auditory input. Sensory deafferentation results in sensory deprivation, and thereby an incomplete model of the world. Under the influence of the medial affective pathway, the importance of the auditory input is defined. The missing information due to sensory deprivation becomes even more important to create a complete model of the world. From the prior model of the world, a prediction of the expected stimulus is created. This happens due to a switch of the prior predictions under the influence of the rdACC. The reference state is then updated with a new reference in which the tinnitus is embedded. This new reference state is then confirmed by the nucleus accumbens, which is part of the reward system. The change in reference is then maintained by the connectivity between the nucleus accumbens and the pgACC [5]. It is suspected that this process is not only passive but also active. To limit the uncertainty of the model, the brain is also actively looking for sensory information. Sensory deprivation limits the amount of sensory input and consequently increases the unpredictability or the prediction error, of the world around us. According to De Ridder *et al.* (2014) [72], the brain is actively trying to fill in this lack of information by utilizing some of the aforementioned processes. This involves hyperactivity in the brainstem nuclei and auditory cortex, and tonotopic map reorganization to retain sensory input.

In a study to graph theoretical analysis of EEG data in tinnitus patients, an increased gamma activity would represent deafferentation, related to the prediction error. This reflects a change in the auditory environment, thus a constant change of auditive percept [54].

2.4.6. Involvement of memory areas

The involvement of deafferentation as explained in Section 2.4.2 can result in over-activity of specific neurons in the tonotopic map of the auditory cortex. However, it is also suggested that the tinnitus as a phantom sound could be pulled from memory [65]. Different areas in the brain are involved in tinnitus with and without hearing loss. Tinnitus with hearing loss involves the parahippocampus where the auditory memory is located [5]. As explained in Section 2.3.4, auditory input is stored in the brain in various areas, such as the auditory cortex, thalamus, amygdala, and hippocampus. The specific area depends on which component of the auditory input is stored. In Section 2.4.3, it is explained that stimuli can be flagged as relevant for the conscious brain, and thus be memorized. Tinnitus awareness persists due to memory mechanisms. If these memories are associated with negative emotions, the memory can become much stronger. These memory mechanisms also reinforce distress associated with the phantom percept [94].

2.4.7. The Triple Network Model related to tinnitus

While the pathways describe the loudness and distress of tinnitus, they only partially explain the chronification of tinnitus or the persistence of tinnitus over an extended period. The TNM, which is already explained in a global sense in Section 2.3.3 is proposed to play an important role in the chronification of the tinnitus percept [6].

When the ascending lateral pathway, medial pathway, and descending noise-canceling pathway become correlated to the TNM, chronification, and functional disability can appear. Comorbidities of tinnitus can therefore be clarified with the TNM. In non-chronic tinnitus, the pathways are uncorrelated to the DMN. The chronification of tinnitus appears when the pathways become correlated with the DMN. The DMN then represents tinnitus as part of the self. The correlation of the DMN with the CEN can induce functional impairment [6].

2.4.8. Tinnitus distress networks

As mentioned in the introduction of this section, there is a proposed difference in the definitions of tinnitus, and tinnitus disorder. Tinnitus disorder is associated with tinnitus-induced distress [13]. In combination with the Bayesian brain theory of Section 2.4.5, the tinnitus percept becomes correlated to distress by co-activation of distress networks, such as the ACC, AI, and amygdala [94]. In an EEG study of the Brodmann areas, a difference between healthy patients and patients with tinnitus disorder was observed. Healthy subjects tend to have a small world topology, as explained in Section 2.2.2, corresponding to a $1/f$, pink noise EEG frequency spectrum. Patients with tinnitus disorder tend to have a more white noise characteristic [54]. A classification study of EEG data in healthy subjects, subjects with tinnitus and distress, and tinnitus without distress was conducted. Linear vector machines were used to distinguish these different groups [99]. This classification study is compliant with an earlier conducted study, distinguishing tinnitus from tinnitus distress. Distress of tinnitus is recorded by differences in the β-band [100], specifically originating in the dACC [101].

2.4.9. Vagus Nerve as a potential gateway to the tinnitus networks

Next to the CN V and C2 fibers, the vagus nerve is also shown to influence brain regions associated with hearing and tinnitus. That is why the vagus nerve might also be particularly useful in the treatment of tinnitus as a gateway to influence some of the earlier-mentioned pathways and networks associated with tinnitus.

VNS has already been used for the treatment of epilepsy and depression [102, 103]. Mainly the cervical part of the vagus nerve has afferent nerve endings on the Locus Coeruleus (LC), Nucleus Tractus Solitarius (NTS), and the cholinergic nucleus basalis [104–106]. It is shown that the LC directly influences activity in the DCN [107]. However, these nuclei also release neuromodulators which can influence the activity and behavior of neurons [105]. These neuromodulators influence activity in various areas in the brain, such as the reticular and autonomic areas in the brain. Stimulation of the vagus nerve has been shown to modulate activity in parts of the auditory system (superior temporal gyrus, Heschl's gyrus, planum porale, and planum temporale) and parts of the limbic system such as the amygdala [106]. All these areas also play a role in tinnitus [73].

2.5. Methods for neuromodulation

In Section 2.4, it is shown that tinnitus is much more a disease in the brain, than a disease of the ear. Although the initial trigger might be caused by sensory deprivation, the persistence of tinnitus is implicated by neurological processes. To treat tinnitus at its core, it makes more sense to modulate the neurological behavior involved in tinnitus, e.g. neuromodulation.

This chapter will discuss the common and less common methods for neuromodulation. Most of them will be focused on electromagnetic stimulation methods. However, there are a lot of alternatives for electromagnetic stimulation that will be discussed in this section.

2.5.1. Influencing neural networks

To investigate, understand, and possibly treat tinnitus in the brain networks introduced in Section 2.4, it is necessary to interact with these brain networks. There are various methods to do so, such as surgical interventions and neuromodulation, e.g. controlling neuronal activity.

Surgical interventions

Surgical interventions such as ablation or dissection in the brain can be used to disrupt brain functions. Strictly speaking, this is not a neuromodulation technique as the consequences are not irreversible. Ablation is a surgical technique by which tissue is damaged in a controlled manner. In neurological interventions, often a lesion is created to interrupt malfunctioning cerebral networks. This can be done by electrocoagulation, radiosurgery, or magnetic-resonance-guided focused ultrasound stimulation (FUS). These techniques have been used for several movement disorders, chronic pain, and psychiatric disorders such as depression, eating disorders, aggression, and epilepsy [108]. For example, through ablation of the DCN in rats, hypotheses of the DCN as a source of tinnitus are ruled out [109]. Of course, some of these techniques are highly invasive and often can have strong side effects. The most important factor is the irreversible damage that is done to the brain tissue.

Excitation vs. inhibition

If we target specific neurons or specific groups of neurons we can do so by influencing the activity of these neurons. Simply speaking, we can inhibit activity (e.g. neurons are prevented from generating action potentials) or we can excite activity (e.g. action potentials are actively induced in the neurons). Depending on the desired effects, one might be more useful than the other. For example, since neuronal activity in the DCN is increased in case of tinnitus, one might try to actively inhibit activity using neuromodulation, hoping to reduce the effects of tinnitus.

2.5.2. Target location and placement of stimulation source

Besides the stimulation method, the placement of the stimulation source and the targeted area can have a major impact on the outcome of the treatment.

The target location is an important factor in the selection of the device and its placement. Stimulation of the brain can be done cortical or subcortical. Cortical neuromodulation can be done by non-invasive devices, such as transcranial magnetic stimulation, transcutaneous VNS (tVNS), transcranial electric stimulation, or focused ultrasound. Subcortical structures can be modulated by implantable devices such as a deep-brain stimulator using an electrode probe into the subcortical tissue, but also with focused ultrasound which is placed on the skull and has higher penetration depth than e.g. transcranial electrical stimulation. The stimulation of peripheral nerves can be done sub-, trans- or percutaneous. Besides the chosen stimulation method, two important factors for deciding the appropriate placement are; (1) required spatial resolution (e.g. the size of the stimulation area) and (2) Depth beneath the skin of the targeted area [110].

The required spatial resolution is strongly dependent on two things. First, the size of the target area, and second, the proximity of the target area to nearby neuronal receptive tissue (e.g. tissue that is receptive to neuronal stimuli and thus also artificial stimulation). For example, a peripheral nerve might have a small target area, but since no receptive tissue is located nearby, the spatial resolution can be much larger. On the contrary, when targeting the parahippocampus, the target area is bigger. However, since a lot of neurons are nearby (= neuronal receptive tissue), the spatial resolution cannot exceed the target area. The spatial resolution is determined by a combination of the distance between the stimulation source and the target area, and the stimulation method used. The further away the stimulation source is from the target area, the more it diverges. It follows from this that the required spatial resolution and depth beneath the skin directly influence whether the stimulation must be subcutaneous or transcutaneous [110].

2.5.3. Modalities

Different modalities have been developed (or are being developed) to influence the nervous system. Arguably, the most promising neuromodulation methods for tinnitus are electrical and to some extent magnetic stimulation. Therefore, those will also be the biggest focus for this literature study. However, we will take our time to review some of the other modalities that can be used.

Electrical stimulation principles

As explained in Section 2.2.1, an action potential happens when the membrane potential of a neuron surpasses the membrane threshold. The membrane potential can be directly affected by an electric field, which is subsequently induced by a current flowing through the tissue [31], see Figure 2.7. In the

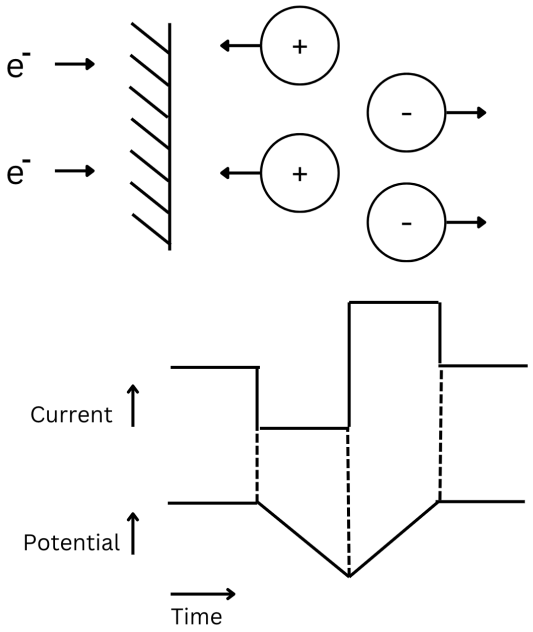


Figure 2.7: Schematic representation of current influence on membrane potential.

simplest scenario of a segmented myelinated neuron, the extracellular potential (V_e) at the tissue is:

$$V_e = I/(4\pi\sigma r)$$

Where I (A) is a point current source, σ (S/m) is the conductivity in the fluid, and r is the distance between the segment and the current source [31, 111]. As can be seen, the relation between the membrane potential and the current is independent of the impedance of the tissue or the Electrode Tissue Interface (ETI). Thus, by controlling the current that is flowing through the tissue, one can directly control the membrane potential and thus the excitability of the neurons. Therefore, most electrical stimulation methods use a current-controlled stimulation

Chemical neuromodulation

The use of drugs or other chemical compounds to modulate neuronal activity and/or function is called chemical neuromodulation. Cells (and thus also neurons) are made of dozens of different proteins and other small and large molecules. By using drugs to target and influence the behavior of some of these molecules, cascading effects on the nervous system can be initiated, thereby altering neuron activity, excitability, cell generation, and much more. Although the focus is shifting towards other neuromodulation methods, such as electrical stimulation, the use of drugs is arguably the most widely used neuromodulation method to date.

Oral and intravenous drugs are the most common types of neuromodulators. The United States Food & Drug Administration (FDA) approved over 1200 different drugs for the treatment of neurological disorders [112], such as depression, Alzheimer's disease, ADHD, and many more. The biggest challenge for these oral drugs is the blood-brain barrier which prevents most drugs from entering the CNS [113]. (Temporarily) increasing permeability, modulating transport activity, and utilizing natural signaling pathways are used as approaches to circumvent the blood-brain barrier [113]. However, these methods are potentially damaging to the brain, and only a small amount of drugs are allowed for these types of blood-brain barrier modulation. Besides, since drugs diffuse throughout the body, the affected regions are large and thus the spatial resolution is small, see Figure 2.8, thereby increasing the possible side effects.

Another method of administering these drugs is by direct delivery into the targeted brain area either by injection or the use of implants [110, 114]. Although the spatial resolution is most often greatly reduced, it is also much more invasive than some of the other methods.

Chemical neuromodulation for tinnitus
Also in the treatment of tinnitus, drugs might be a useful method of neuromodulation. Some antidepressants have been shown to cause a significant reduction in tinnitus-related symptoms [115–117]. However, no follow-up studies have been done since and results might be primarily related to the stress-reducing effects of antidepressants [118]. Another example is the glutamergic acamprosate, which showed improvements in tinnitus in two trials [119, 120]. However, up until now, no drug has been approved for the treatment of tinnitus. The main difficulties are often the large amount of side effects and overcoming the earlier mentioned blood-brain barrier in the case of oral drugs. Chemical neuromodulation is not the focus of this literature study, however, Langguth and colleagues (2019) wrote a comprehensive review article on many of the drugs that have been tested for tinnitus so far [118].

Magnetic stimulation
An alternative to electrical stimulation is magnetic stimulation. Magnetic stimulation works similarly to electrical stimulation (which will be discussed in more detail hereafter), in that it induces an electric field by which the membrane potential of neurons is changed. Subsequently, it excites or inhibits the neuron in question.

By far the most common method for magnetic stimulation is Transcranial Magnetic Stimulation (TMS), where an electromagnetic coil is placed on the scalp [114]. TMS often uses quickly discharging capacitors to generate large currents through an electromagnet, which in turn creates a pulsating magnetic field [114, 121]. As mentioned, magnetism induces an electric field in the target area, which can generate action potentials [121]. To induce a sufficient response, large magnetic fields are required (> 1 Tesla) [114, 121].

The magnetic pulse frequency and application area generally determine the excitatory or inhibitory effects of TMS. Therefore TMS is often subdivided into slow TMS (< 1 Hz) which is inhibitory and fast TMS (> 1 Hz) which is excitatory [121].

A subvariant is repetitive TMS (rTMS) where pulses are administered in series which is believed to have the most therapeutic effects. The main disadvantages of TMS are the large size, high power consumption, and relatively poor spatial resolution.

Focused Ultra Sound
Focused Ultra Sound (FUS) is a neuromodulation technique in which ultrasound waves interact with mechanosensitive ion channels of neurons in the brain. Since sound is a pressure wave, this interaction can invoke action potentials in the neuron. FUS provides a high spatial resolution and deep penetration depth compared to other non-invasive neuromodulation techniques. This resolution increases with aperture size and frequency [122]. Ultrasound can be used in low or high intensity for reversible inhibition or exhibition, or for irreversible tissue ablation respectively [123].

Optogenetics
One special upcoming type of neuromodulation is optogenetics. It uses light-sensitive proteins (opsins) that are genetically inserted into the cell [110, 114]. Opsins are a group of light-sensitive molecules, ranging from light-sensitive ion channels, ion pumps, and enzymes [124]. Opsins are also used in animals as part of photoreceptors for vision. By genetic modification, these opsins can be inserted into neurons, so that exposure to light will initiate either the excitation or the inhibition of the specific neuron [125, 126]. Either directly, by using light-sensitive ion channels or ion pumps, or indirectly by using cascading effects leading to (the inhibition of) an action potential. The subsequent activation of the opsins by a controlled light source can be highly specific, both temporally and spatially [110]. Because of this, optogenetics is a tool in neuroscience of increasing importance. However, the need for genetic modification makes this method unlikely to be used in humans soon, both due to safety concerns and ethical reasons [114].

2.5.4. Electrical stimulation waveforms
The electrical stimulation pattern with which a neuron is stimulated can influence its subsequent response. These patterns can be categorized into two main groups. The first group of electrical stimulation methods aims to directly induce an action potential in the targeted neurons. This includes tonic and burst stimulation. The second group does not directly induce an action potential but increases (or decreases) the susceptibility of a group of neurons to incoming action potentials.

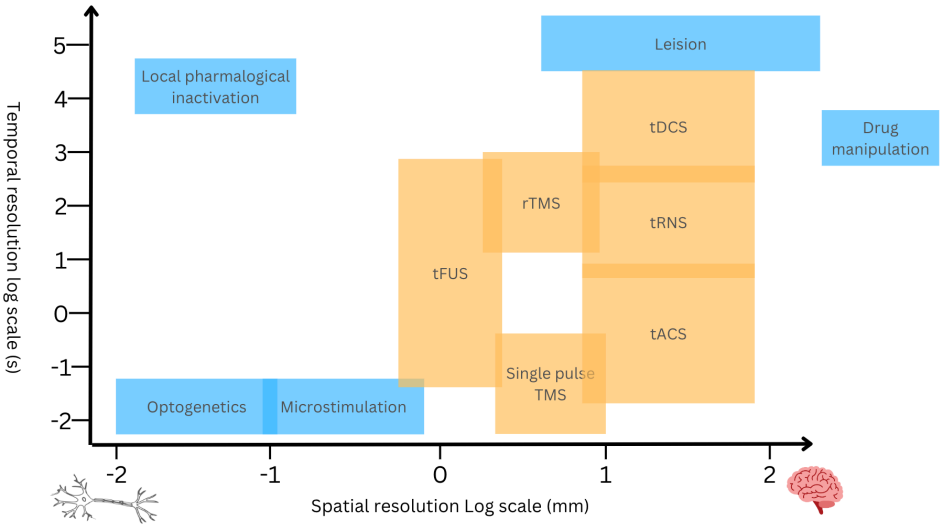


Figure 2.8: The spatial and temporal resolution of different neuromodulation techniques. [127]

Tissue damage and charge balancing

Electrical stimulation can harm and damage the tissue. Mechanisms for tissue damage due to electrical stimulation are still debated, but according to Merrill and colleagues (2005), two types of mechanisms have been proposed [111]. First, excessive stimulation and subsequent firing of neurons cause tissue damage in itself. For example, by reducing oxygen or an excess release of neurotransmitters. Second, the exchange of electrons causes chemical reactions at the Electrode Tissue Interface (ETI), which causes tissue damage.

The stimulation pattern or waveform can influence not only the effectiveness but also the damage done to the tissue. It has been suggested that both charge per phase and charge density per phase affect the amount of tissue damage [128]. Charge per phase is the integral of the current, and charge density per phase is the integral of the current density, thus the current per square meter. When using biphasic pulses, where both the positive and negative pulses have equal charge, the charge left in the tissue should be zero. It is shown that these types of charge-balanced biphasic pulses cause less to no tissue damage compared to monophasic pulses while remaining as effective in stimulating neurons [129, 130]. Charge balancing is thought to be less critical in transcutaneous electrical neurostimulation since electrochemical reactions at the ETI are at the skin, and not inside the body releasing toxic chemicals. Nonetheless, it has been the convention to charge balance pulses to prevent any tissue damage. A notable exception for charge balancing is direct current stimulation, which will be discussed in greater detail. An extensive analytic study in 2016 found no reported cases of irreversible injury [131].

Tonic stimulation

The most common form of stimulation is called tonic stimulation. With tonic stimulation, a pulse of a certain pulse width and pulse amplitude is fired at a certain pulse frequency. Tonic stimulation can be either monophasic or biphasic (charge-balanced), see Figure 2.9. As mentioned before, charge-balanced biphasic stimulation is viewed as a safer alternative. As can be seen, charge-balancing can be done in different manners.

In tonic stimulation, a pulse width of 100-200 us, pulse amplitudes below 3.5 mA, and pulse frequencies of 20-30 Hz are most common [132]. However, for example in VNS, the pulse width (100-500 us), pulse amplitude (0.13-6 mA), and pulse frequency (1.5-120 Hz) can vary greatly [132]. Predominantly, stimulation effects are increased by an increasing pulse charge. The pulse charge can be increased by increasing the amplitude or temporal length of a pulse. [133, 134].

Burst stimulation

A special form of stimulation is burst stimulation. Burst stimulation is different from tonic stimulation in that multiple pulses follow each other in rapid succession ($\sim$500Hz), then followed by a larger inter-burst period [135, 136]. The pulse amplitudes are generally lower than in tonic stimulation.

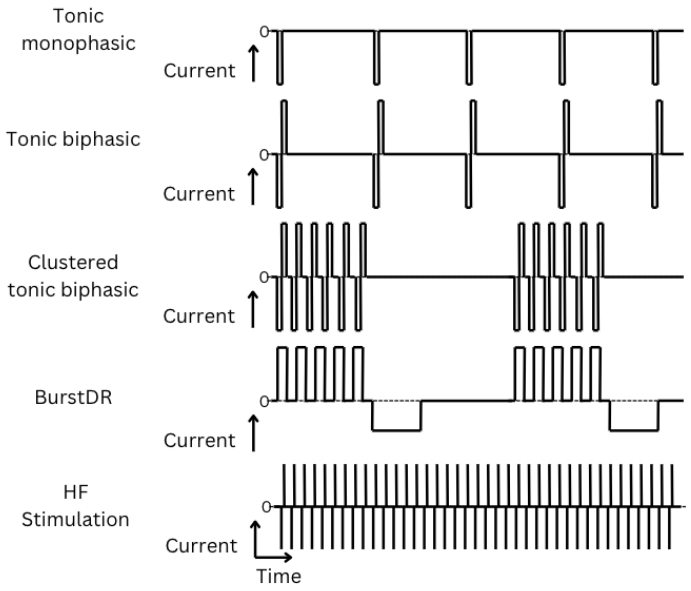


Figure 2.9: Schematic representation of stimulation waveforms

Neurons in the thalamus can fire in tonic and burst patterns [137]. Burst stimulation was designed to mimic one of these two neuronal firing patterns in the thalamus [138]. Thereby it is thought that the thalamus responds differently to tonic and burst stimulation patterns, subsequently activating different cortical areas [138]. It is argued that burst stimulation has a higher chance to induce a response in the cortex.

The first version of burst stimulation, called burstDR (DR are the initials of one of its developers), was used in 2010 for the treatment of noise-like tinnitus [139] and chronic pain [140]. Since then burstDR has been applied to the ACC for the treatment of tinnitus, alcohol addiction, and OCD, and to the somatosensory cortex for the treatment of pain, amongst other applications [141]. Furthermore, it has been shown that burstDR stimulation was superior to tonic stimulation in most of its applications [141]. The burst pattern in burstDR consists of 5 monophasic pulses of increasing amplitude, which is only passively charge-balanced. Average pulse widths are around 300-1000 μs and pulse amplitudes are between 2-4 mA [135, 138].

Other burst stimulation patterns are developed as well, most notably burst stimulation by Boston Scientific [141]. However, it is contested whether Boston's burst has the same neurological effects [141, 142]. Opposed to burstDR, Boston's burst pulses are individually charge-balanced and were therefore argued by De Ridder and colleagues (2020) to be better described as clustered tonic firing [141]. And indeed, supporting this claim, both methods exert different clinical effects [143].

High-frequency stimulation

A special variant of tonic stimulation is high-frequency stimulation. Here, charge-balanced biphasic pulses are created at high frequencies (HF) (100-1000 Hz) and ultra-high frequencies (U-HF) (up to 10 kHz) [135, 144], see Figure 2.9. Opposed to normal tonic and burst stimulation, HF is used to block axonal conduction in nerves. It was shown that for frequencies over 50 Hz, neurons were not able to continuously follow firing patterns. For even higher frequencies (> 150 Hz), HF can completely block the conduction of neural signals through the stimulated region [145].

Deep-brain stimulation using HF was successfully used as a treatment of Parkinson's disease [146, 147], although the underlying mechanisms are unclear. In addition, HF and U-HF were successfully used in spinal cord stimulation for the treatment of neuropathic pain [135].

Direct-Current Stimulation

With direct current (DC) stimulation, a current is passed through the tissue from one electrode to the other, see Figure 2.10. The aim is to influence neuronal excitability by polarizing the membrane [127]. The most common example of DC stimulation is transcranial DC stimulation (tDCS), where two or more

electrodes are placed on the scalp, one being the cathode, and the other being the anode. Commonly, a DC of 1-2 mA would be administered for around 10-30 min [127, 148].

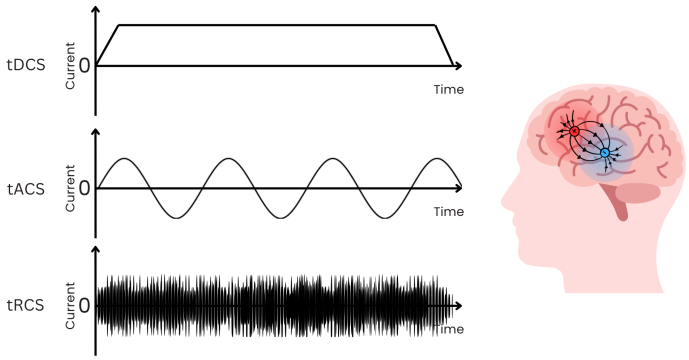


Figure 2.10: Schematic representation of different waveforms for transcranial electric stimulation.

Alternating-current stimulation

Alternating current stimulation uses an alternating current in specific frequencies to modulate intrinsic cortical oscillations. The brain functions in rhythmic patterns, such as discussed in Section 2.2.2. Transcranial alternating current stimulation (tACS) modulates these natural rhythms. Common current amplitudes are between 1.5-2 mA at frequencies of 6-13 Hz and stimulation periods of 20 min for tACS [148].

Noise stimulation

Lastly, a noise signal can be used for stimulation, also called Random Noise Stimulation (RNS), where a random noise is generated with a certain frequency spectrum.

The physiological effects of RNS are still debated. The general aim is to randomly increase the membrane potential of neurons closer to the threshold, either to increase the chance to fire randomly by utilizing the stochastic resonance [149] or to make them more susceptible to incoming stimuli [150]. It has also been suggested that RNS is able to desynchronize ”hyper”-synchronized brain networks by pushing neurons randomly over or under the membrane threshold [151, 152].

RNS is commonly divided into low-frequency RNS (lf-RNS; < 100 Hz) and high-frequency RNS (hf-RNS; > 100 Hz) [149, 150]. The magnitudes commonly follow a normal distribution and the net charge is in principle always zero. Common current magnitudes are between -2 and 2 mA and stimulation periods of 20 min for both lf-tRNS and hf-tRNS [148].

2.6. Tinnitus and neuromodulation

Different neuromodulation treatments have been tried for tinnitus. As mentioned in Section 2.5.2, neuromodulation can both be targeted at the central parts of the nervous system (e.g. the brain) and the peripheral nerves. Similarly, a distinction can be made between invasive and non-invasive stimulation methods, with non-invasive methods being the most preferable.

Using these two categorizations, we will discuss some of the neuromodulation treatments that have been tried for tinnitus, including their success and the possible prospects of these methods. First, we will go into the treatments where a single modality (unimodal) is applied. Second, we will go into some of the treatments where multiple modalities (multimodal) are used to treat tinnitus.

Tinnitus characterization

To compare the efficacy of neuromodulation techniques, tinnitus should be measurable. However, comparing the subjective tinnitus loudness and distress between patients is difficult. Therefore different protocols and questionnaires have been developed to assess the severity of someone’s tinnitus symptoms. Tinnitus-related distress is often assessed using questionnaires, such as the Tinnitus Handicap Index (THI) and Tinnitus Function Index (TFI) [153]. In addition, tinnitus loudness and pitch might be assessed by audiometric tests, often presenting a generated tone to the patient and matching its loudness and/or pitch to the tinnitus tone/sound [153].

2.6.1. Unimodal stimulation for tinnitus

Invasive stimulation of the brain

There have not been many tinnitus treatments using invasive stimulation of the brain, also known as intracranial stimulation. However, Deep Brain stimulation (DBS) and intracranial alternating current stimulation (iACS) have been used.

DBS DBS has been used to treat a variety of disorders other than tinnitus. It is thought that DBS could interfere with tinnitus by either inhibiting or exciting certain neurons, or by disrupting pathological oscillations in functional connectivity. The placement of the electrodes differs depending on whether the tinnitus is unilateral or bilateral, with the electrode being placed in the contralateral and non-dominant hemisphere, respectively [154]. In 2015, Smit et al. conducted a comprehensive review of DBS in tinnitus, noting several potential DBS locations that have only been tested in animal models, such as the DCN. The MGN was also identified as a potential target, as it is thought to be involved in the multisensory integration of tinnitus [155]. In 2019, Zwieten et al. proposed that the MGN is spontaneously firing in tinnitus patients, as well as in those with Parkinson's disease, for which DBS is already used. Stimulation of the MGN is believed to reduce tinnitus distress [156]. However, as of 2023, this is still a research proposal for use in humans [157].

iACS iACS have been explored less extensively for the treatment of tinnitus. One study was conducted, where patients benefitting from TMS took iACS with burst waveform which better results for noise-like tinnitus compared to tonal tinnitus. In tonal tinnitus, there was no difference between a tonic or burst waveform [158].

Non-invasive stimulation of the brain

The most common non-invasive brain stimulation methods that have been used in the treatment of tinnitus are TMS, tACS, tDCS, and transcranial Random Noise Stimulation (tRNS).

TMS In repetitive TMS, pulses are repeated in a rhythmic pattern during a session. The frequency is one of the most important factors in the effectiveness of the stimulation technique, which can induce an increased excitability or an inhibitory effect. In tinnitus treatment, multiple applications of rTMS have been conducted. Application on the temporal and frontal cortices was aimed to suppress tinnitus perception, concluding those brain areas as important for tinnitus perception [159]. Low-frequency stimulation is used to increase neuronal activity in the AC by targeting the temporal cortex. Outcomes differ over multiple studies, and clinical outcomes are partial and temporary [160]. The effects of treatment protocols, tailored to the patient have more effects than standardized treatment protocols, as was shown in a study comparing the targeting of the dorsolateral prefrontal or temporoparietal cortex [161].

tACS tACS modulates the natural rhythms of the brain, such as the alpha and gamma activity in the AC for tinnitus [99]. A study using tACS at the alpha frequency applied to the temporal cortices did not suppress the tinnitus [162].

tDCS Anodal tDCS typically increases the excitability of the cortical areas, whereas cathodal tDCS decreases the excitability. Thereby, the plasticity of the cortical areas changes [163]. The targets of tDCS for tinnitus are the auditory, temporoparietal, or dorsolateral prefrontal cortex. Anodal tDCS on the AC using repeated sessions did not have the expected effect on tinnitus. Bifrontal tDCS was investigated with repeated sessions and gave preliminary effects. [160] The effect of tDCS on tinnitus does not seem to present a treatment solution.

tRNS tRNS uses the principle of stochastic resonance. Stochastic resonance applies noise to a system to improve the signal-to-noise ratio. Applying noise to the temporal cortices did suppress tinnitus [162]. It has also been shown to reduce tinnitus loudness and distress by applying stochastic resonance on the AC [11].

2.6.2. Multi-modal stimulation for tinnitus

Following the expression; neurons that ”fire together, wire together” [160, 164], it has been shown that bimodal stimulation can induce long-lasting neuroplastic changes in animal models [79, 89–91]. Because of neuroplastic mechanisms, see Section 2.2.3, such as Hebbian and spike-time dependent plasticity, long-lasting changes in neural activity can be induced when pairing stimulation methods. While unimodal stimulation may induce long-term potentiation and other longer-lasting effects, multimodal stimulation might be more effective in doing so [160].

Various bimodal stimulation methods have been tried for tinnitus so far, some of which have shown promising results. We will discuss some of these methods down below.

Pairing of transcranial stimulation methods

tDCS and tRNS A study using multimodal stimulation with bifrontal tDCS before bilateral auditory cortex tRNS was compared to unimodal bifrontal tDCS. Both are examples of multisite stimulation, while only the first is an example of multimodal stimulation (tDCS + tRNS). First tDCS was applied and second tRNS was applied. This study showed a more pronounced effect by using multimodal stimulation over unimodal stimulation [165]. However, another study using tDCS to stimulate the frontal cortex and simultaneously inhibit the auditory cortex did not have positive effects [166]. There was a difference in using only tDCS for both locations instead of tRNS, compared to using tDCS and tRNS sequentially.

tDCS and TMS A combination of tDCS and TMS was used in a study where bi-frontal tDCS was combined with TMS on the contra single side of the temporoparietal cortex. Four groups were used: one for each modality, one with both modalities, and a control group with sham TMS. All groups showed improvements compared to the control group, and the bimodal group showed the largest mean difference in tinnitus intensity and distress [167].

Auditory stimulation in tinnitus

In the past, a variety of auditory stimulation methods have been used to either mask or decrease tinnitus symptoms. Many of them have been initially designed as a unimodal method to treat or relieve tinnitus. However, since then some of them have also been used in combination with other methods of stimulation.

Auditory training Auditory training is a therapy in which a tinnitus patient is either actively trained requiring behavioral responses, or passively trained listening to sounds in the background. The audio presented can be in the tinnitus frequency range or outside of the tinnitus frequency range [168].

In active training in the tinnitus frequency range, two groups were trained with either a tone matched with their tinnitus frequency or one distant from their frequency. After training, they were questioned with tones identical to or different from their tinnitus frequency, and received feedback about the correctness of their answer [169]. In most of these and similar procedures, small to no differences were measured. In active training outside of the tinnitus frequency range, the goal is to train attention to sounds outside of the tinnitus frequency, by identifying and localizing sounds. These sounds are daily sounds, presented over background noise. The sounds are not specifically outside or excluding the tinnitus frequency [170] . Another approach is to use a hearing aid that amplifies sounds around the tinnitus frequencies [171].

Passive auditory training within the frequency is for example done by tailor-made music which enriches the music with tones at the tinnitus frequencies. Patients start with a volume that masks the tinnitus, but after several months, gradually decrease the volume. This was combined with relaxation therapy. The outcomes were a substantial reduction in tinnitus distress [172–174].

An approach for passive auditory training outside the frequency range is tailor-made notched music training (TMNMT). A study used TMNMT with a notched octave around the tinnitus frequency. This was controlled by a group that had a changing octave notched throughout the training. Tinnitus loudness was reduced in the treatment group and less than the control group [175].

Sound masking Sound masking is a technique in which a simple sound generator adds environmental sounds to the room the tinnitus patient is in. For most patients, tinnitus is chronic which is the reason they need a small, portable device, a kind of similar to a hearing aid. This device adds sound to the ear

while not blocking the ear canal for environmental sounds. In the case of a tinnitus patient with hearing loss, a tailor-made hearing aid including sound generation can be used [176].

Transcranial stimulation with auditory stimuli

Shekhawat *et al.* (2014) combined the use of tDCS on the left temporoparietal area (LTA) and hearing aids [177]. Hearing aids can provide long-term improvement of tinnitus [178, 179] and tDCS is able to lead to transient suppression of tinnitus as was mentioned before [160]. Therefore it was hypothesized that, due to neuroplastic mechanisms, combining both treatments would mutually benefit each other. The results showed that the hearing aids were beneficial for tinnitus reduction. However, tDCS would only improve tinnitus loudness in the short term (3 months) while in the long term (6 months) both hearing aids with and without tDCS improved tinnitus loudness.

The same researchers paired tDCS on the LTA with auditory stimulation using broadband noise (BBN) in a subsequent study [180]. With similar reasoning, they hypothesized that auditory stimulation would extend the effects of tDCS. However, also in this study, no beneficial effects were found by pairing both methods.

Two other studies paired tDCS with TMNMT. A first study targeting the AC did find a significant effect, however, irrespective of the use of tDCS [181]. On the contrary, a second study that targeted the dorsolateral prefrontal cortex did find a significant reduction of tinnitus-related symptoms [182]. No other studies combining tDCS and TMNMT have been performed since.

Peripheral nerve stimulation with auditory stimuli

Many new tinnitus treatments under development use peripheral nerve stimulation. Although non-invasive, many of the transcranial methods are bulky and expensive and are not easily applicable for tinnitus patients with mild to medium symptoms. Peripheral nerve stimulation methods are often smaller and can potentially also made non-invasive. Most nerve stimulation is performed using electrical stimulation using one of the electrical pulsing patterns (tonic, burst, or high frequency) described in Section 2.5.4.

Although some studies have been performed where only one peripheral nerve is stimulated, most of them pair electrical stimulation with auditory stimulation. The bimodal methods targeting peripheral nerves for the treatment of tinnitus can be largely subdivided into two (groups of) targets: stimulation of somatosensory nerves and VNS.

Somatosensory nerve stimulation The first is utilizing the somatosensory integration into the auditory nerve system [4, 79], see Section 2.4.2. Stimulation of the somatosensory nerves aims to reduce activity in the DCN. The targeted nerves include the CN V and cervical nerves 2 and 5 (C2 and C5).

Non-paired stimulation

Koning *et al.* (2021) suppressed activity in C5 by means of an injected drug (bupivacaine 0.5%) [183]. They found that tinnitus was suppressed in 30% of patients after 7 weeks. For patients with hearing loss at 1 kHz and osteophytes (”bone spurs”) around the 5th cranial nerve, this could be improved to 50%. However, none of the results were significant.

Unpaired transcutaneous electrical nerve stimulation (TENS) in the neck, targeting the C2 fibers, gave a significant response in only 17.9% of patients [184]. Although the response was strong (on average 43% reduction), the effects only lasted till right after the treatment. More recently, animal studies [185] and human trials [185, 186] have been performed to test the efficacy of TENS of the C2 fibers paired with sound. Both used bursts of tones with frequencies matched to the tinnitus frequency. Opposed to unpaired TENS of C5, they found a significant reduction in tinnitus-related symptoms that lasted long after treatment. However, Spencer *et al.* (2022) only had a response rate of 23% similar to unpaired TENS, as opposed to a 50% response rate by Marks *et al.* (2018). This might indicate that this treatment might only be effective in somatic tinnitus with pure tones which Marks selected for, although further research is required to make those claims.

Trigeminal nerve stimulation

Another approach that tries to make use of the somatosensory integration in the auditory system, uses stimulation of the CN V paired with auditory stimulation. Similar to stimulation of the cervical nerve, trigeminal stimulation is based on its innervation of the DCN [187]. That somatosensory stimulation

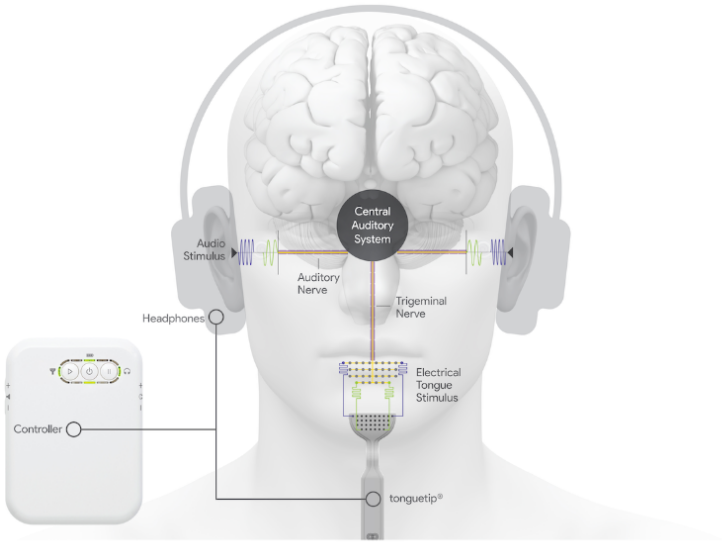


Figure 2.11: Schematic representation of the LENIRE device by Conlon *et al.* (2019) [193]. over-ear headphones are used for audio stimulation. This is paired with electrical stimulation of the trigeminal nerve through the tongue, with a *tongue tip* that uses 32 microelectrodes. A controller is used to control both audio and electrical stimulation.

affects activity in the DCN due to auditory stimuli [188]. This has resulted in the development of a device (LENIRE) by Neuromod Devices Limited, which has recently received FDA approval in the United States (see Figure 2.11). With this method, auditory stimulation consists of a combination of music and noise-like sounds whereby the frequencies surrounding the fall-off frequency are amplified [189]. The fall-off frequency is not equal to the tinnitus frequency but is often closely related [190]. Furthermore, they used an array of 32 stimulators on the tongue, the amplitudes of each frequency band in the auditory stimulus were paired with each stimulator, thereby suggesting that the CN V endings on the tongue are tonotopically mapped. Although this is the case for the AC [19], there seems to be no scientific evidence that this also happens in the tongue nor that bimodal stimulation can induce such a tonotopic map. Hamilton *et al.* (2016) first showed the efficacy of this method [189]. Subsequent large randomized trials showed a large response rate (70-80%) with significant reductions after 12 weeks of treatment [191, 192]. Although response rates are high, the efficacy is less profound than in some of the other bimodal treatments (10-20 points in THI). Furthermore, all the studies up to date lack a control group, which makes these results less reliable.

Vagus Nerve Stimulation The second is stimulation of the vagus nerve. As described in Section 2.4.9, the vagus nerve activates the LC and NTS which subsequently influences activity in areas implicated in tinnitus, including areas related to the medial pathway and the auditory cortex.

Non-paired stimulation
Although unimodal VNS is a common treatment for epilepsy [102, 103], most VNS for the treatment of tinnitus has been paired with auditory stimulation [105, 148, 160]. However, some studies have been performed on the effectiveness of tVNS for tinnitus treatment. A pilot study using tVNS for 6 months did conclude safety and tolerance to tVNS, but positive effects on tinnitus did not occur [194]. Subsequent studies on tVNS were able to find significant reductions in tinnitus-related symptoms. However, the effects remained small [195, 196]. Tutar *et al.* (2020) also found a large response in the control, although that was less profound than in the active group.

Paired stimulation
Combining VNS and auditory stimuli was first tested on rats by Engineer et al. (2011) [85]. The underlying idea is that the tonotopic map in the AC is distorted due to noise-induced tinnitus, see Section 2.4.2. VNS paired with tones excluding the tinnitus frequency was expected to induce cortical map reorganization in the auditory cortex. Indeed, it was found that VNS and sound completely removed

tinnitus-related symptoms in rats [85].

Since then, other studies have found the same results in humans, although the outcome was less profound [7, 197–199]. In all studies, invasive VNS (iVNS) was performed and tonic stimulation with the same parameters was used. All studies found a significant reduction in tinnitus-related symptoms. Tyler *et al.* (2017) in particular found a reduction of symptoms in around 50% of the patients against 28% of the controls after 6 weeks [198]. Controls also received VNS, the relatively high reduction rate in controls might therefore also be related to the stress-lowering effects of VNS. As was expected, Vanneste *et al.* (2017) also found desynchronization at the gamma frequency band in the left AC which correlated with the amount of loudness reduction [199]. tVNS has also been used as a non-

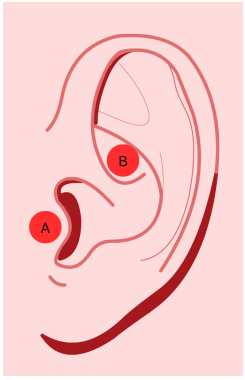


Figure 2.12: Nerve endings of the auricular branch of the vagus nerve: A. Tragus, B. Concha. [105]

invasive alternative to VNS. tVNS is most often performed at the auricular branch of the vagus nerve which has nerve endings around the ear, particularly the concha and the tragus. It has been shown that tVNS at the auricular branch (taVNS) influences the same pathways and brain networks as VNS [106, 194]. taVNS paired with TMNMT has therefore also been tested with stimulation of the concha [200] and tragus [201]. These studies were designed with the same underlying idea of cortical map reorganization. Similar positive results were found in both studies. It has to be said, however, that the effectiveness of VNS paired with sound needs further research since sample sizes of all these studies often have been small $(n < 30)$ [11, 105]. Furthermore, as also discussed in Section 2.2.3, cortical map reorganization in tinnitus might be more related to the associated hearing loss [73, 86].

Already in 2016, Dr. de Ridder proposed a theory to pair VNS with tinnitus-matched sound. The goal of VNS paired with tinnitus-matched sound is to remove the salience of the tinnitus sound, by stimulating the habenula and AC simultaneously. To make this as effective as possible, the most unrewarding stimulus should be given to the habenula [202].

2.7. Concluding remarks

Over time, 80% of people suffer from hearing damage, but only a part also get tinnitus. Tinnitus is not purely dependent on hearing damage but can occur under high stress. The Bayesian brain principle (Section 2.4.5) predicts that audio frequencies are entering the ear, where the absence of specific frequencies due to deafferentation (Section 2.4.2) introduces a prediction error. The prediction error, which is a constant after deafferentation, can be marked as salient by the SN (Section 2.4.7). If the brain flags this missing information as salient, which can occur under stress (Section 2.4.8), it can be reproduced from memory 2.4.6. VNS paired with the tinnitus-matched sound should remove the salience of the tinnitus sound thereby reducing the presence of tinnitus [202].

3 Methods

The device's prototyping is divided into three separate parts. First, the device's requirements are defined using the literature study and an interview with Prof. Dr. de Ridder. Second, the design approach involves specifying the device's block diagram with inputs and outputs at the highest level. This is then iterated over as one delves deeper into each block diagram, eventually replacing the blocks with individual components. This approach was used for the electrical circuit and the software design, which came third.

3.1. Device Requirements

The concluding observations of the literature study are used as a guideline for device requirements. The device will be used to test Prof. Dr. de Ridder's hypothesis. The first session with Prof. Dr. de Ridder defined the functional device requirements and parameters. Prof. Dr. de Ridder was well aware of what type of parameters and ranges should be available for the user. After a large part of the electrical and acoustic stimulation circuits were designed, the user interface (UI) requirements were set in a second session with Prof. Dr. de Ridder. In preparation for the UI session, a list of possible parameters was set up, asking for a range for the parameter and a possible UI component, such as a button or rotational knob. The electrical and safety requirements were set based on conversations with Prof. Dr. Ir. Serdijn. Prof. Dr. de Ridder had a lot of wishes for the device. He wants to be able to test as many hypotheses as possible by combining multiple stimulation methods, such as transcutaneous burst stimulation, transcranial noise stimulation, and acoustical stimulation. To keep the requirements feasible for a master thesis project, the requirements were restricted to create a minimal viable product (MVP). Other demands were classified as nice to have and postponed to future implementations. The restrictions on the device requirements were also checked and confirmed by Prof. Dr. de Ridder. The final requirements for the device are presented in this subsection. The complete list of requirements is added to Appendix B.

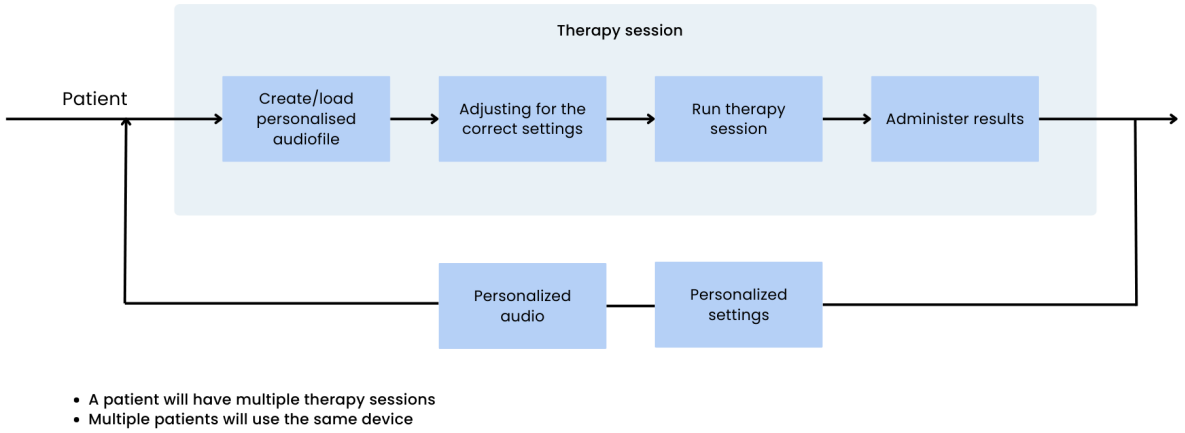


Figure 3.1: The process around using the device

To ensure the device is feasible, functional, and user-friendly, knowing who, where, and how it will be used is essential. Figure 3.1 shows a schematic representation of this process. The device will be used by a clinician who will research multiple patients. Every patient experiences a unique tinnitus sound. This specific tinnitus sound should be matched as an acoustical stimulation signal. When a patient comes in for therapy, an audio file will be created that simulates the tinnitus sound for that person as much as possible. Personalized audio needs to be used with the device for stimulation. The audio will be paired with electrical stimulation on the tragus. Thus, it is plausible that the device will need a headset connection for acoustical stimulation and an ear clip for electrical stimulation connection. The intensity of electrical stimulation varies from person to person. For example, the skin is a highly variable medium to work with in terms of electrode placement and contact. The load impedance of the electrode will vary from patient to patient. The clinician will need tools to find the correct therapy parameters. After these are found, the treatment will start. After the therapy, the clinician must register the settings and results. The same parameter settings and audio file can be used as a starting point for the next session. Each therapy session will take approximately 30 minutes, and each patient will have three weekly sessions for three weeks. The functional and technical device requirements are elaborate and detailed. A schematic representation is shown in Figure 3.2. The requirements are divided into general requirements, requirements for synchronicity of stimulation, electrical stimulation, auditory stimulation, safety, and UI requirements.

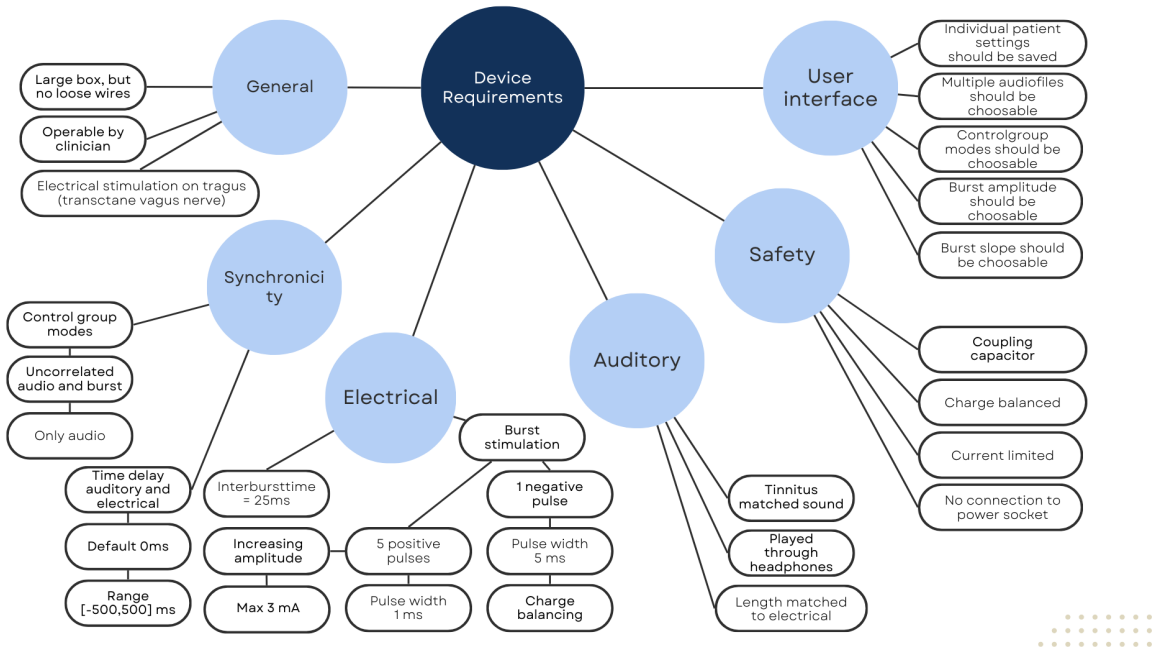


Figure 3.2: All device requirements specified in a diagram

3.1.1. General requirements

As explained in the introduction of this section, the device will be used by a clinician in a clinic. During a stimulation session, the device simultaneously simulates the tragus transcutaneously using the burst waveform and plays tinnitus-matched audio through headphones. The device could be a large box, although it should not have loose wires, which could deter the patient. However, a very appealing device could bias a patient with a placebo effect.

3.1.2. Synchronicity

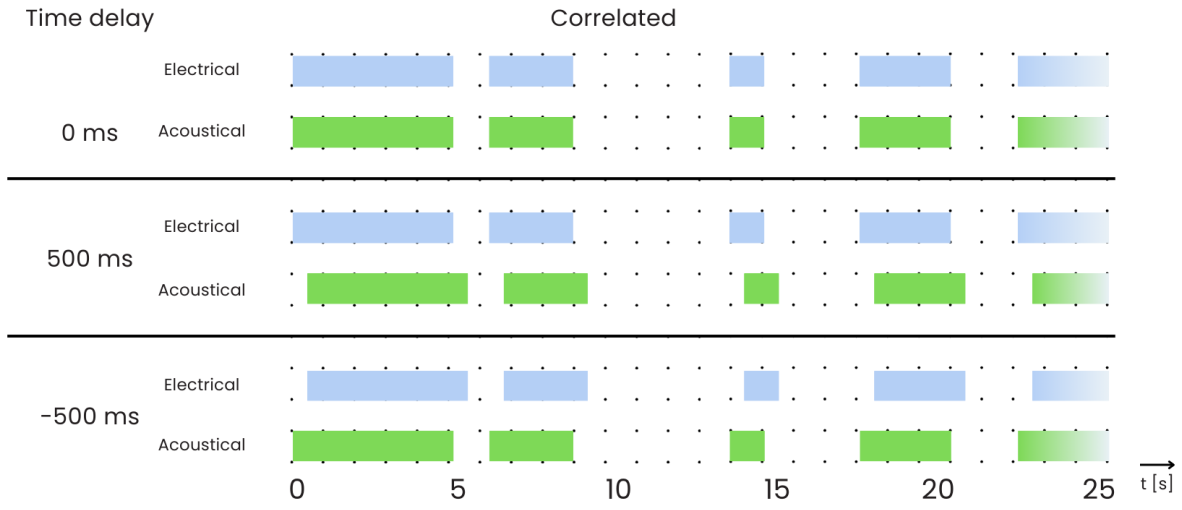


Figure 3.3: The synchronization of electrical and acoustical stimulation

A stimulation session should last a maximum of 30 minutes. During the session, a multitude of electrical and acoustic stimulations should be given, with a stimulation length randomly between 1 and 5 seconds. The pause length between two stimulations should also be random, between 1 and 5 seconds. The delay, Δt, can be adjusted between -500 and 500 ms by the user. A positive Δt indicates the

electrical stimulation leading the acoustical stimulation and vice versa. These parameters are visually represented in Figure 3.3.

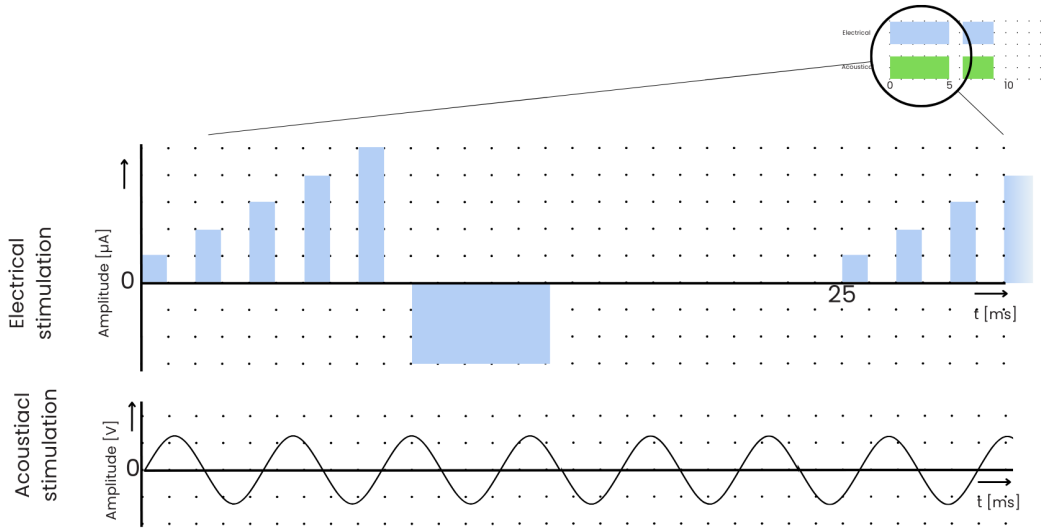


Figure 3.4: The waveforms of electrical and acoustical stimulation

Multiple electrical burst stimulations should be given during one acoustical stimulation, as shown in Figure 3.4.

3.1.3. Electrical stimulation

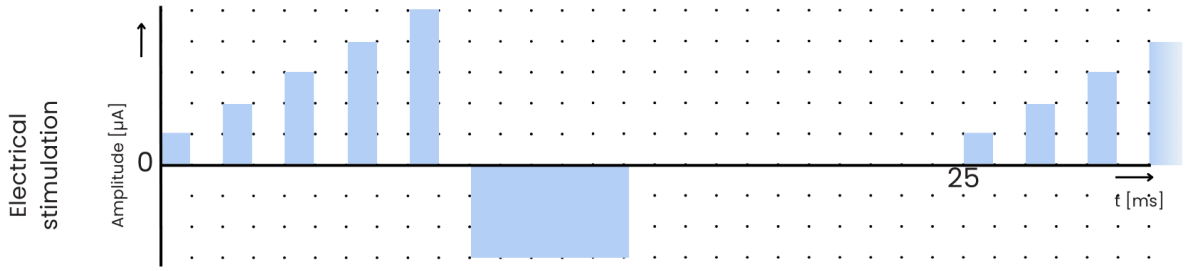


Figure 3.5: Burst stimulation

The electrical stimulation uses the burst waveform, displayed in Figure 3.5. The burst waveform should have five positive monophasic pulses equal to or increasing amplitude. The burst amplitude represents the current through the tissue, which should be a maximum of 3 mA. The pulse width and the delay between pulses should be 1 ms. This results in a spike frequency of 500 Hz, which should be fixed. The reasoning behind this frequency is that neurons do not repolarize between the pulses, resulting in a nonlinear build-up. This invokes a postsynaptic action potential and, thus, more effective stimulation. The number of pulses should not exceed 5, as the chance of evoking a postsynaptic action potential is not increasing with more pulses. The amount of charge provided to the tissue from the positive pole should equal the charge supplied to the tissue from the negative pole. The five positive pulses should be followed by one negative charge-balancing pulse. Charge balancing means that the temporal length times amplitude of the five positive pulses together should equal one negative charge-balancing pulse. By default, the time between the starting point of each consecutive burst should be 25 ms, which corresponds to a burst frequency of 40 Hz. The clinician should adjust the burst frequency through the MicroSD card to 0.1-40 Hz.

3.1.4. Acoustical stimulation

An audio source that plays a tinnitus-matched sound should provide the acoustic stimulation. The wish was that the audio could also be created with the device or that the patient could tweak the audio frequencies. This degree of flexibility is considered to be beyond this project's scope. The tinnitus-matched sound could also be made and even played with a computer. To ensure the entire audible frequency range of 20 Hz to 20kHz can be created, the audio should have a sampling rate up to 44,1kHz, which is CD quality. Most people have tinnitus below 8kHz, but some have it at higher frequencies, such as 12kHz. Another requirement is that the audio should be in stereo, as uni- or bilateral tinnitus exists. The temporal length of acoustical stimulation should equal that of electrical stimulation. The audio should be played through headphones with sufficient volume to drown out the patient's tinnitus sound.

3.1.5. User interface requirements

The User Interface (UI) is designed after an interview with Prof. Dr. D. de Ridder. All parameters with their possible range, default value, and adjustment method were discussed. Figure 3.6 shows the conclusions of both the input parameters and feedback the UI should give. This section will go over each of the UI components.

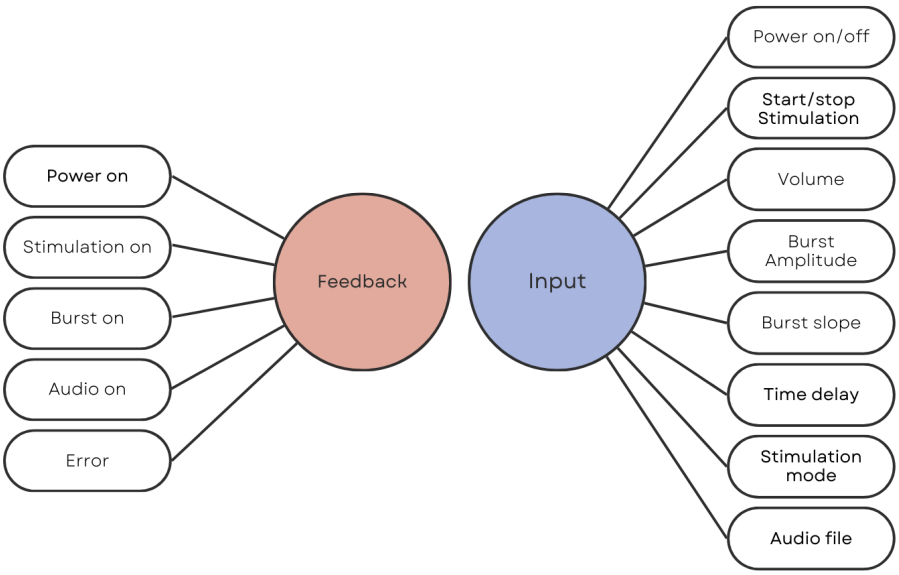


Figure 3.6: The input and feedback parameters of the user interface

The chances are that the demands of the device will change over time, such that some parameters are not necessary to adjust, and maybe new parameters are introduced to the device. There should be a certain level of flexibility in the adjustment of parameters.

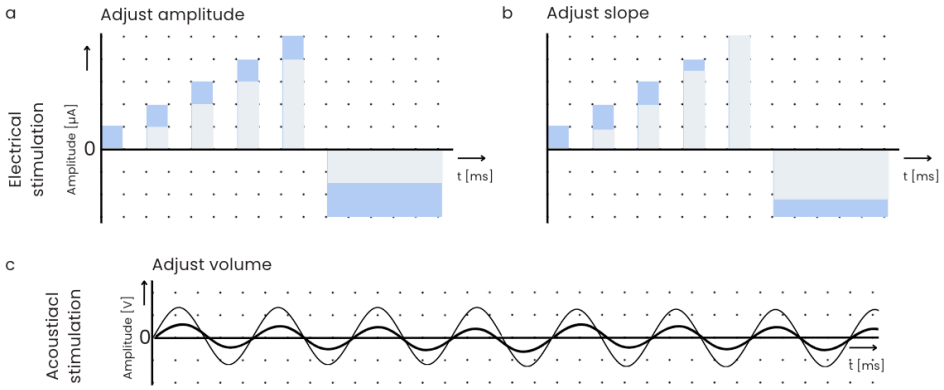


Figure 3.7: Schematic representation of parameter adjustments in the user interface for: a. The burst amplitude changes are represented by the light and dark blue, b. The burst slope changes are represented by the light and dark blue, c. The acoustical amplitude changes are represented by a thick and thin line

Figure 3.7 shows the effect of the parameter adjustments. If the burst amplitude is reduced, the slope angle should depend on the fifth pulse's amplitude. If the slope is adjusted, the amplitude of the fifth pulse remains the same while the others are changed. All pulses still have a range of 0 to 3 mA. The volume of the audio should be adjustable through the menu.

The time delay should be adjustable from -500 ms to 500 ms with steps of 100 ms. By default, the time delay is 0 ms.

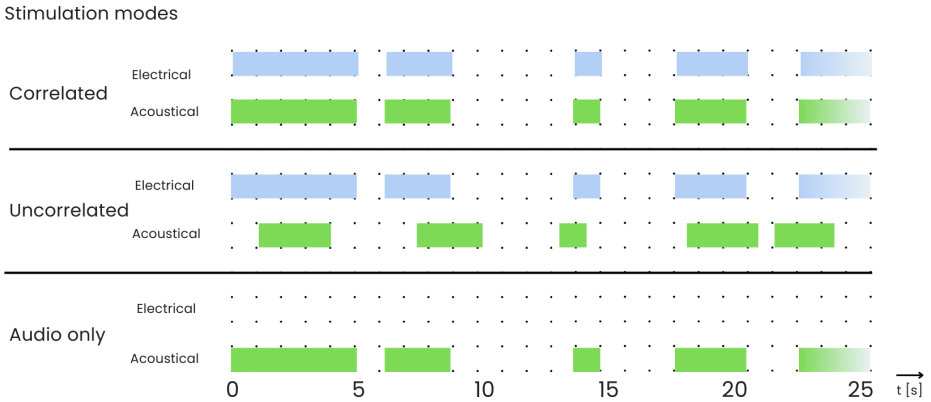


Figure 3.8: The timing of a stimulation session

3.1.6. Safety requirements

The safety requirements for a device such as this should comply with the Directive 93/42/EEC on Medical Devices before being sold on the market. [203]. Although the device will not be designed for the market but for clinical studies, it is still very important to comply with the Medical Device Directive. The device will be a stimulation, non-invasive device, classifying it as a class I medical device. Article 12 of ANNEX I states the requirements for medical devices connected to or equipped with an energy source. Programmable systems should be designed to ensure repeatability, reliability, and performance. In case of a single fault condition, appropriate means should be adopted to eliminate risks. During regular operation and single-fault situations, the device should be designed to avoid the risk of accidental electric shocks. In the case of an internal power source, the device should indicate the state of the power source.

3.2. Device design

The design process started by defining the device's inputs and outputs as a system block diagram. The next stage was to identify subsystems and define their inputs and outputs repeatedly until all hardware

components and connections were defined. From there, the implementation of the UI and the software was developed. At last, the device's boxing was designed.

Figure 3.9: The inputs and outputs of the device

A first simplified functional top-level block scheme of the inputs and outputs consists of command data as an input and electrical and acoustical stimulation as outputs, shown in Figure 3.9. The absence of a power supply as input suggests an internal power source. This choice was made to avoid the possibility of dangerous situations where the device is connected to the mains.

On the subsystem level, command data should be given through a UI and need to be decoded by a command decoder to make the device run the commands. The command decoder controls an electrical stimulation circuit and an acoustical stimulation circuit, which output their electrical and acoustical stimulation signals, respectively. All these circuits need electrical energy to work, which should be provided by a battery during operation. A circuit is required to convert the battery voltage into a feasible reference and supply voltage for the circuits. From the battery, a DC voltage regulator will provide supply and reference voltage for the other subsystems according to their needs.

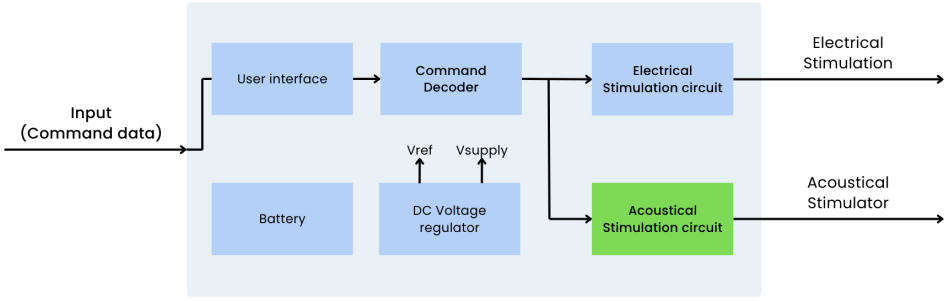


Figure 3.10: The subsystems of the device

3.2.1. Command decoder

Considerations

The control circuit could be implemented using various technologies, such as a field-programmable gate array (FPGA), a microprocessor such as a Raspberry Pi, or a microcontroller (µC) such as an ATmega328 used in an Arduino UNO.

FPGAs and microprocessors can compute functions in parallel, which means that electronic and acoustical stimulation can run simultaneously. µCs are sequential, which means that they can run one line of code at a time. Both options can process inputs and outputs, which will be necessary to direct the stimulation circuits. The FPGAs are programmed by the composition of building blocks, which can be done using a hardware description language such as VHDL. A µC, on the other hand, uses a high-level programming language, such as C or C++, where C is a procedural programming language, and C++ is also an object-oriented programming language.

The colleague student and the author have experience in Arduino programming and prototyping. Arduino also has a large community, forums, and third parties building shields for it, which will accelerate the prototyping process.

Microcontroller (µC)

Arduino is a user-friendly open-source platform for hardware and software. The Arduino can read digital and analog input values and write digital and pulse-width-modulation outputs. These inputs and outputs are connected to the sockets on the Arduino. They can be programmed using the Arduino programming language, which is based on C++. These programs can be uploaded to the Arduino using the Arduino software integrated development environment (IDE) via a USB serial connection. This IDE works on all computer operation systems, which is essential for the team. Compared to other platforms, the Arduino is relatively inexpensive, simple, and easy to work with for hardware and software.

Initially, an Arduino UNO was used to develop the device. In a later stage, the flash memory and SRAM of the Arduino UNO were a limiting factor for the requirements of the device. The switch was made to an Arduino MEGA 2560 Rev3 to implement all requirements. The structure of the MEGA is the same, although it increases the number of inputs, outputs, and SPI outputs and changes the I2C pin numbers. The processor of the Arduino Mega is an ATmega2560, which runs at 16MHz, has 8KB SRAM, 256KB Flash, and 4KB EEPROM memory [204].

The Arduino Mega has 54 digital input/output pins and 16 analog input/output pins, of which 12 pulse width modulation outputs (PWM). PWM is a modulation technique where a timer is used as a clock with a specific frequency, and a duty cycle percentage determines the average voltage output. The output is a square wave which the duty cycle determines the width of the pulse. 100% duty cycle is a constant high voltage, and 50% corresponds to half the voltage. If a low-pass filter is applied with a cut-off frequency below the frequency of the clock, the signal will become smoother. The PWM signal and SPI ports use the different timers available in the Arduino Mega as shown in Figure 3.11. Digital inputs register a voltage as HIGH when it is above 2.6V and LOW if it is below 2.2V. Digital outputs set the voltage to 5V in case of a HIGH output. This is useful for controlling switches or transistors, for example. Analog input pins are 8-bit ADCs, which transform a voltage range between 0V and 5V with steps of ±0.2V into an integer value between 0 and 255.

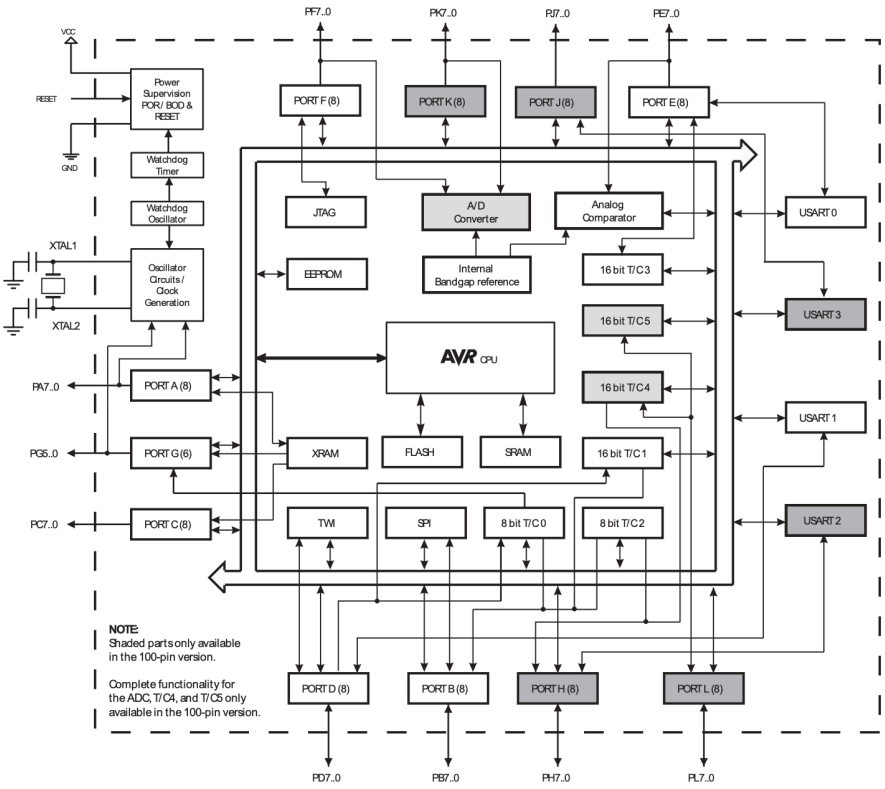


Figure 3.11: ATmega2560 [205]

The selection of the Adafruit Music Maker in conjunction with Arduino was primarily driven by the compatibility of the DAC with Arduino's SPI communication protocol, as outlined in Sections 3.2.1 and 3.2.2. The Adafruit Music Maker buffers audio files from the SD card through the Arduino via SPI before transmitting them to the audio decoder during playback. The DAC also receives every change in voltage output over SPI. Thus, the DAC and Adafruit Music Maker are concurrently communicating over SPI, leading to interruptions in their communication processes. This interruptions are illustrated in Figure 3.12. Attempts to regulate buffering time intervals within the Adafruit Music Maker proved challenging. Ultimately, transitioning to an ESP32 microcontroller provided a viable solution to address these communication conflicts and enhance system performance.

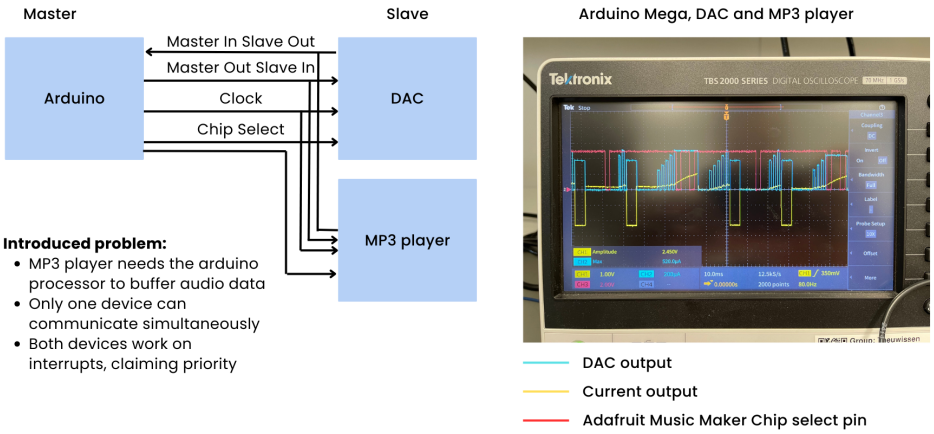


Figure 3.12: On the left hand, a schematic representation of the SPI connections between the Arduino, DAC, and Adafruit Music Maker is shown. On the right side, the measurements of the oscilloscope are shown. The blue line is the output of the DAC, displaying the burst stimulation. The yellow line shows the output after the H-bridge, and the red line shows the chip select connection between the Arduino and Adafruit Music Maker. Each time the chip select pin is pulled low, the Arduino communicates to the Adafruit Music Maker, interrupting the burst stimulation output of the DAC.

The ESP32 is a µC with a dual-core processor and three SPI channels, which could be operated separately. It can also run the Arduino environment, meaning all the code written thus far is still useful. It has 12-bit ADCs, with 4096 steps between 0V and 3.3V, and two 8-bit DACs, with 256 steps between 0V and 3.3V.

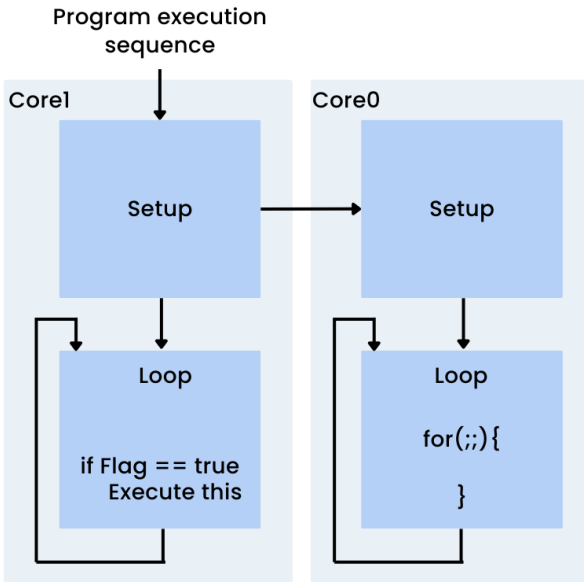


Figure 3.13: ESP32 dualcore

The second core can be initiated in `setup()` of the first core, shown in Figure 3.13. The second

core function will only be run once. If a `for(;;)` is placed, the part inside the bracket will function as the `loop()` function of the original core. This makes it possible to run a high-priority task on its core without being interrupted by any other process [206].

Clock and Timers

All computations and processes in a µC are dependent on the clock. The clock outputs a square wave at a specific frequency. The processor runs at the speed of the clock. The clock of the ESP32 is adjustable from 80 to 240 MHz. Timers are, for example, used for the PWM output signal but can also be used for custom functions. The timer can run on the clock frequency divided by an integer prescaler. The speed of the clock is the fastest a timer can run.

Interrupt service routines (ISR)

Interrupt Service Routines (ISR) are small parts of code that can interrupt the main program execution sequence. The interrupt can occur based on an interrupt pin, which changes, or based on a timer.

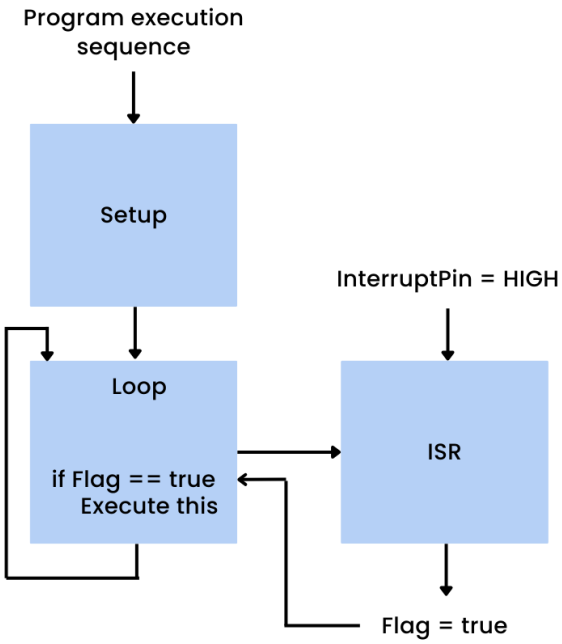


Figure 3.14: Pin interrupt service routine

The pin interrupt can be helpful when a second device needs the µC to stop the regular program routine and temporarily switch to a different program, shown in Figure 3.14 [207]. For example, the Adafruit Music Maker uses a pin interrupt to inform the µC that the buffer is empty and a new part of an audio file can be loaded.

In case of a timer interrupt, a predefined time interval can be used to interrupt the regular routine and, for example, flag a boolean true, which then activates a specific part of the regular loop code, shown in Figure 3.15. ISR should be kept short to prevent the watchdog timer from overflowing, resulting in a device reset [208]. The watchdog is a mechanism that ensures the ISR is not blocked from running for a prolonged time.

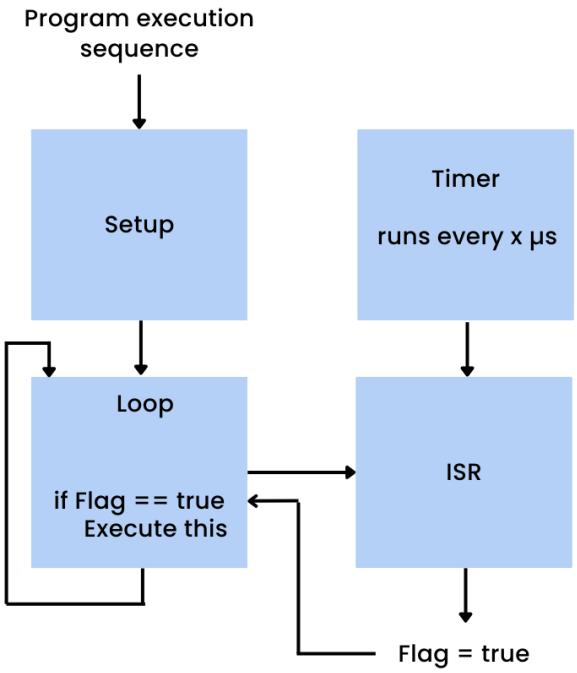


Figure 3.15: Timer-based interrupt service routine

A delay function can be used inside the main loop as an alternative to the timer-based ISR. The downside is that the delay function then occupies all processing opportunities; thus, nothing else could be processed during the delay.

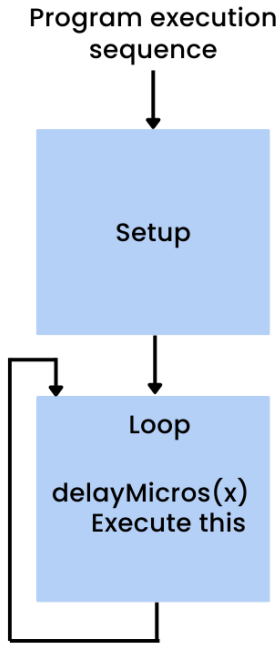


Figure 3.16: Delay in the loop for a timed functionality

Serial Universal Asynchronous Receiver Transmitter (UART)

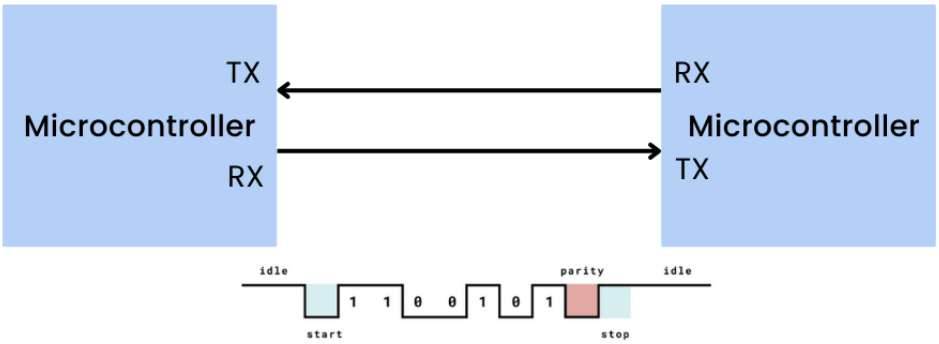


Figure 3.17: Serial UART communication protocol

Serial UART is a relatively slow protocol that can communicate one bit at a time, as shown in Figure 3.17. The bit train is opened with a start bit and ended with a stop bit. In between is one byte of communication. The second last bit is a parity bit that is used to detect errors during transmission. No clock is needed for this communication, as the receiver always expects the same message structure. The serial UART is also used to upload software from the computer to the µC [209].

Inter Integrated circuit (I2C)

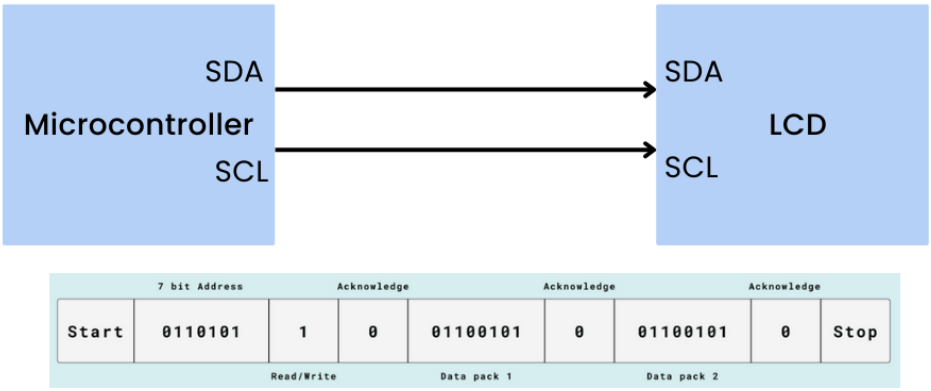


Figure 3.18: I2C communication protocol

Inter-Integrated Circuit (I2C) is a bidirectional serial communication protocol between a master and one or multiple slaves. It uses two connections, serial data (SDA) and serial clock (SCL) connections, as shown in Figure 3.18. The communication speed of I2C is between 100 kbps and 5 Mbps. An I2C message consists of a start condition, an address frame of 7 or 10 bits, and the read/write bit. This is followed by an ACK/NACK bit, an 8-bit dataframe, another ACK/NACK bit, a second 8-bit dataframe, and an ACK/NACK bit again, closing with a stop condition.

At the start condition, the data line drops in voltage before the clock drops. Each slave has an 8-bit address, which should be matched for the slave to accept the message. The read/write bit is high or low when the master requests or sends data, respectively. The ACK/NACK functions as a receiving confirmation to the master. The stop condition is the reverse of the start condition; the data line rises after the clock rises. The bits of communication are read while the clock is at a high level [210].

Serial Peripheral Interface (SPI)

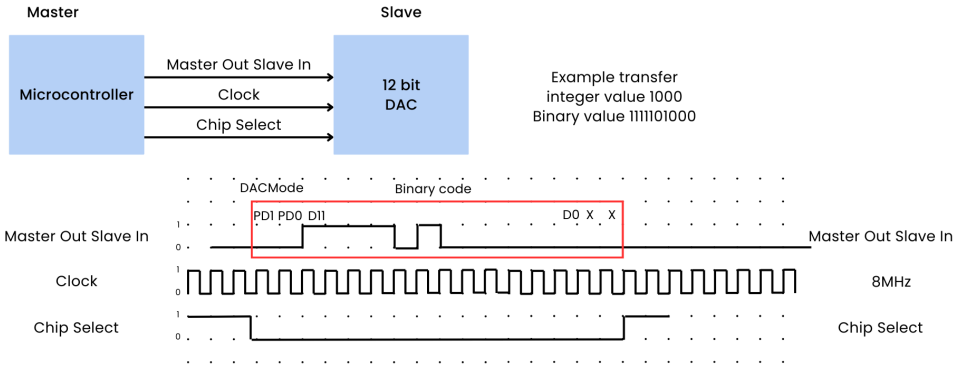


Figure 3.19: SPI communication protocol

Serial peripheral interface (SPI) communication needs four connections between a master and slave: the Master Output/Slave Input (MOSI), the Master Input/Slave Output (MISO), the clock (SCLK), and the Slave Select/Chip Select (SS/CS), as shown in Figure 3.19. The benefit of SPI over I2C is that data can be sent or received continuously without interruption. SPI is also a serial communication with a speed of up to 10 Mbps. The master sends the SCLK signal to synchronize the slave. In the case of one master and multiple slaves, the SS/CS is pulled low to select a specific chip. The MOSI connection transfers a byte with the most significant bit first, whereas the MISO connection transfers the least significant bit first. The data bits are synchronized with the rising edge of the clock [211].

3.2.2. Electrical stimulation

The electrical stimulation circuit was completely designed by the colleague student, and the technician designed the DualSTIMs printed circuit board (PCB). Knowing the design and layout of the electrical stimulation circuit is important to knowing how to interact with it. For this reason, this chapter only gives an overview of the components of the DualSTIM and explains the components and interactions with the Arduino in greater detail.

The DualSTIM consists of a stimulation circuit and a power management circuit. The stimulation circuit consists of a 12-bit Digital-Analog Converter (DAC), which the Arduino directs over SPI. The digital-to-analog converter (DAC) output is linked to the current source, where the voltage is converted into a stable current output independent of the load impedance of the stimulator. The current output uses a reference voltage of 38V, facilitated by a 38V boost converter within the power management circuit. The current output is connected to the H-bridge, which provides the option to charge balance the five pulses of the burst. The digital outputs of the Arduino direct the switch of the H-bridge. As a safety measure, a single fault safety is connected in parallel to the electrode outputs. As the last stage before the electrode outputs, a coupling capacitor is added to isolate the tissue from direct contact with the power source for safety reasons. The power management circuit starts with a 9V battery. One output of the PCB is used as a battery status indicator. In parallel, a 5V buck converter, a 5V reference, and a 38V boost converter are connected to the 9V battery. The 5V buck converter powers the ESP32 and the Music Maker shield. The 5V reference only provides the DAC with its reference voltage, which should be stable and reliable, as the stimulation intensity is directly related to this output. The 38V boost converter is connected to the current source, H bridge, and single fault safety. The 38V boost converter stimulates the tissue with sufficient charge. For practical purposes, the rotary encoder, LCD screen, and start/stop button are also connected to the PCB, where some components also need minor circuitry to work.

DAC

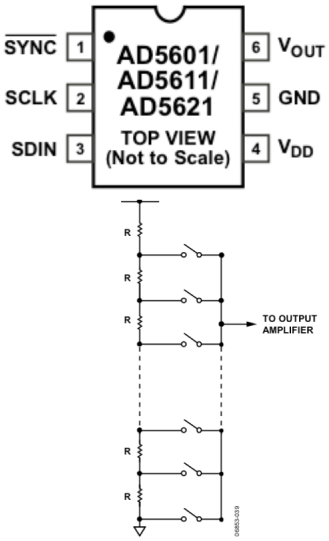


Figure 3.20: AD5621 string DAC

The 12-bit AD5621 of Analog Devices is used for the DAC, which is a resistor string DAC [212]. A resistor string DAC is a series of resistors with switches parallel to the resistors, as shown in Figure 3.20. The specific output voltage can be obtained by closing specific combinations of switches based on the binary input value. The binary input value is to the DAC through SPI, as shown in Figure 3.21. The switch is closed when a bit is high, allowing current to flow through that resistor. All switches open result in 0 Voltage, and all switches closed to give the maximum supply voltage. This DAC operates with a 5V supply voltage provided by the 5V reference converter. The stability of the 5V reference voltage is essential because this is the maximum output voltage of the DAC. The ESP32 communicates with the DAC through SPI (see Section 3.2.1), using outputs 15 (HSPI-CS), 13 (HSPI-MOSI), and 14 (HSPI-CLK) to inputs 1 (SYNC), 3 (SDIN) and 2 (SCLK) of the DAC, respectively. The DAC is connected to the ground on pin 5 and outputs the voltage through pin 6. The DAC registers communication when the SYNC voltage drops from high to low. The first two bits in the communication represent the power-down modes, followed by MSB first. At the end of the communication, the SYNC voltage should rise again. This DAC outputs a maximum of 100µA at 5V and a maximum error of ±0.0244% [213].

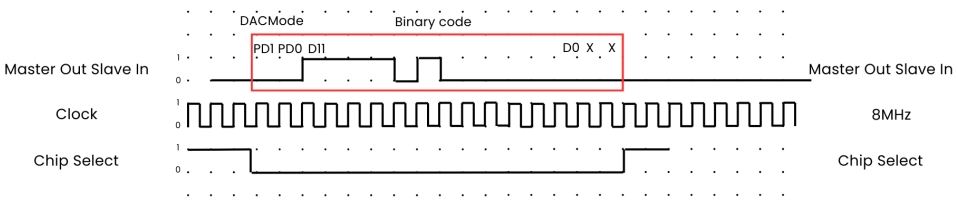


Figure 3.21: SPI communication towards the DAC

Current source

In the electrical stimulation of tissue, in this case transcutaneous, there is a high variability in the impedance of the electrode-tissue interface. Enough charge should be transferred through the electrode-tissue interface to the neurons to induce an action potential in the target neurons. By using current-controlled stimulation, there is more control over the amount of charge provided to the tissue. This is not only important for the efficacy but also for the safety of the stimulation. The downside is a lower power efficiency compared to voltage-controlled stimulation, which is not limited by the requirements [214]. The DAC's output is used as a reference voltage for the current source. The current source takes in the reference voltage and supply voltage. The reference voltage is converted into a current

LOGIC		B to A	
		LOW	HIGH
A to B	LOW	Ground	Positive
	HIGH	Negative	Zero

Table 3.1: The effects of the H-bridge logic switching

output through an op-amp and current mirror. The current will be kept stable independent of the load impedance, which is the tragus in this case.

H-bridge

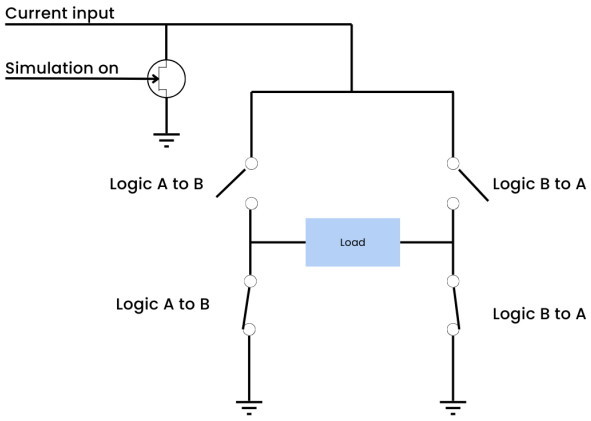


Figure 3.22: The H-bridge configuration to switch the current from a positive direction through the load to a negative direction. Stimulation On, Logic A to B, and Logic B to A are all connected to a separate digital output of the ESP32 to control the H-bridge.

An H-bridge is used to switch the direction of the current through the load from one direction to the other. An N-channel Mosfet connected to ESP32 pin 25 is used to switch the H-bridge on or off, represented by ”stimulation on” in Figure 3.22. Four switches are placed arranged in the configuration comparable to the letter ”H”. The upper left switch connects the current source to one side of the load, while the right upper switch connects the current source to the other side of the load. The bottom switches connect the load to the ground on both sides of the load. The upper switches are normally open, and the bottom switches are normally closed to ensure the load, in this case, the tissue, is normally connected to the ground. The switches on the same side of each other are paired to ESP32 outputs 26 and 2, representing logic A to B and logic B to A, respectively, as shown in Figure 3.22. If the stimulation is active, the ESP32 output 25 will be set to LOW to ensure the current is not going to ground but directed towards the H-bridge. When the logic from A to B is LOW and B to A is HIGH, the current will run to the ground in the ”positive direction” through the tissue. If logic B to A is LOW and logic A to B is HIGH, the current will run in the ”negative direction” through the tissue to the ground, representing the charge-balancing pulse of the burst. The timing of switching the H-bridge is of great importance. By default, both A to B and B to A are LOW to guarantee that the load is connected to the ground. When switching from one current direction to another, first, the logic that is HIGH should be pulled LOW to guarantee that both logics are LOW and the load is connected to the ground.

Single fault safety

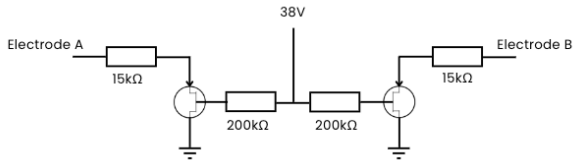


Figure 3.23: Single fault safety

Single-fault safety is a safety measure in case of an unexpected system shutdown. When the system shuts down while stimulation is on, there is a high chance of charge being left in the tissue. Two P-channel MOSFETs are connected to either side of the electrode and ground, as shown in Figure 3.23. The 38V boost converter provides the gate voltage. When the boost converter shuts off, the electrodes are pulled to the ground, dissipating any leftover charge from the tissue to the ground.

Coupling capacitor

A coupling capacitor filters the DC offset in the stimulation signal. A constant DC signal harms the tissue as a continuous non-zero charge builds up. In the case of passive charge balancing, a coupling capacitor would have adverse effects [215]. In this case, active charge balancing will be used, and the stimulation will be transcutaneous, meaning that the electrode-tissue interface is at the skin. These factors reduce the adverse effects of passive charge balancing using a coupling capacitor.

3.2.3. Acoustical stimulation

Considering the acoustic stimulation, it is required to be able to play audio with a sampling frequency of 44.1 kHz, preferably in stereo, because of possible bilateral tinnitus. Alessandro, a previous master's student, used an Arduino UNO to play audio with a sample frequency of 16kHz [216]. To get 44.1 kHz, he recommended using an Arduino MKR Zero. The MKR Zero has an internal 10-bit DAC. Using an amplification circuit, such as shown in Figure 3.24, the analog 44.1 kHz output of the DAC output can be amplified to drive an 8Ω headphone speaker. An 88.2kHz 8-bit unsigned PCM mono audio file can be played from the MicroSD card slot of the Arduino MKR Zero [217]. This could be a solid option to play audio in mono, but it will be a problem for Stereo, as the MKR Zero has only one output. As Arduino is a sequential platform, creating electrical stimulation with a resolution of the burst spikes in milliseconds while also playing an audio file at this frequency could collide and cause problems.

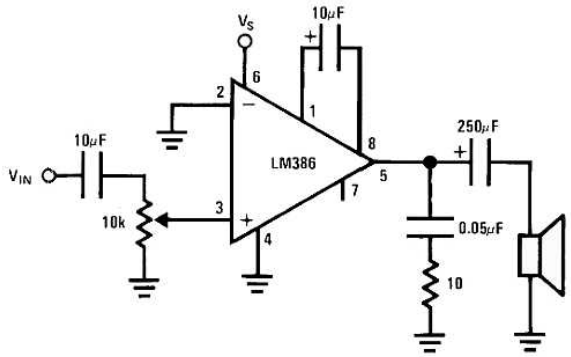


Figure 3.24: Arduino MKR Zero audio amplifier circuit.

The sequential working principle of the Arduino blocks the possibility of simultaneous stimulation. Another option would be to connect an Arduino to a computer. The Arduino could command the laptop to play audio, and the Arduino would be responsible for outputting the burst stimulation [218]. The downside of this option is the chance that the user will plug the laptop into the power socket, introducing safety risks for the patient.

For a Raspberry PI, it will be able to extend it with a high-quality DAC playing 192kHz/24bit audio quality [219]. This would even be considered overqualified. Due to Jonathan and Stef's programming experience with Arduino, the Raspberry PI option has been dropped.

So-called shields can also extend Arduino UNO. One company that creates a lot of shields for Arduino is Adafruit. Adafruit provides the Music Maker mp3 shield for Arduino [220]. It is compatible with Arduino Uno, Mega, and Leonardo. The Music Maker works as a standalone chip with its own decoder, which makes it possible to play audio files from a MicroSD card. Most audio codecs, such as MP3, MP4, WAV and AAC, are supported. The shield is provided with a stereo 3.5 mm jack for 32Ω headphones, also connected to the ground to safely plug in headphones. The shield communicates over the Serial Peripheral Interface (SPI) with the Arduino (see Section 3.2.1). This makes it possible to use an interrupt pin of the Arduino, such that the Arduino can run code while the shield is playing audio. This is precisely what the device needs. The Music Maker also provides seven general-purpose input output pins, which can be used for buttons or LEDs.

Later in the process, it was clear that the Adafruit Music Maker had to buffer the audio file every 40 ms through the processor of the Arduino over SPI. At that point, the switch was made to the ESP32 (see Section 3.2.1, which is also able to run the Adafruit Music Maker. A tweaked version of the library for the ESP32 was found on Git Hub [221]. This version used the second core to play the audio, while the electrical stimulation should run on the second core. Both stimulation modalities run on the same core again and are thus colliding again. The library is tweaked again, so the electrical stimulation runs on core0, and the acoustic stimulation runs on core1. The final library is saved as a custom library in the project folder and referred to from the main code with "import". This was explicitly done to ensure the library will not be changed or overwritten by library updates if someone wants to tweak the device's software. The entire library is in Appendix D. The specific change to play audio from core1 instead of core0 is at line 104 in section D.3.

3.2.4. User interface

The user interface is designed after a second interview with Prof. Dr. de Ridder. All parameters with their possible range-, default value-, and adjustment method were discussed. At the time of implementation, the aim was to work with an Arduino UNO with limited inputs and outputs. The efficient use of inputs and outputs was an important criterion. Also, the flexibility and extendability of the user interface design were accounted for. Changing the parameters later with the same hardware should be possible when the device is already in use. This gives the user flexibility and autonomy over the necessary settings without consulting a software engineer.

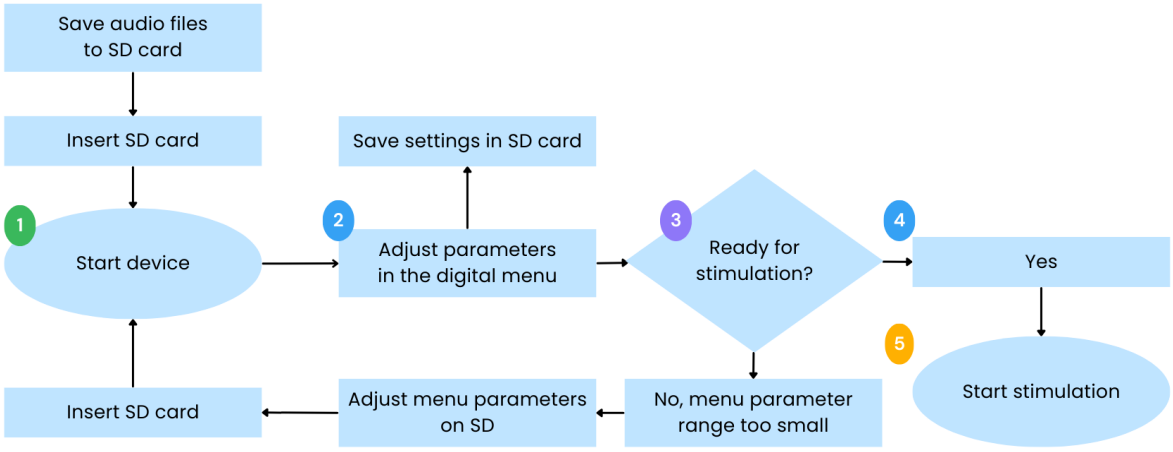


Figure 3.25: User interaction flow

Figure 3.25 shows the intended user flow. The user uploads the tinnitus-matched audio files to the MicroSD card and inserts it into the device. The correct parameters for stimulation are set and, when confirmed, saved to the MicroSD card. If the values are in the range of the digital menu, the stimulation session can start. If not, the user can turn the device off, take the MicroSD card out, put it back into the computer, and adjust the range of the digital menu parameters. After the changes are made, the

MicroSD card is inserted into the device again and powered on. The correct settings are set, and the stimulation session can start.

This user flow results in the UI interface components as shown in Figure 3.26.

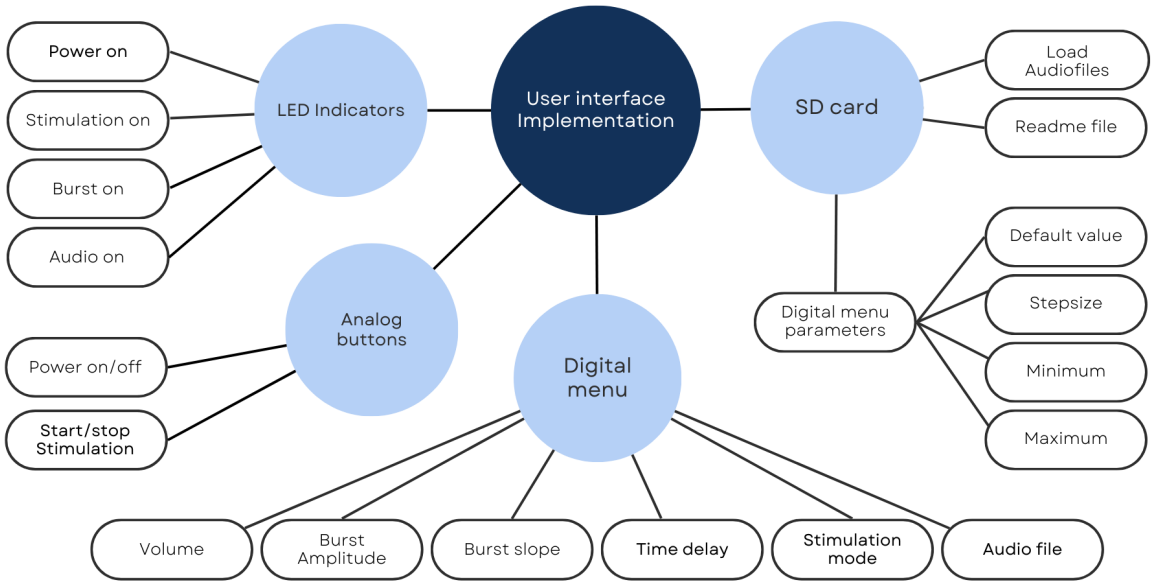


Figure 3.26: User interface components

Working principle

The device uses analog buttons and a digital menu. The digital menu makes the device flexible and extendable if more parameters are added. The MicroSD card is used as external and personal storage for each patient. Still, it can also be used to adjust the device's basic parameters and save information between sessions.

LED indicators

It is also beneficial to know the device's state, such as if the stimulation is on or off. This is indicated with a red LED. Also, the clinician can monitor the operation mode through the audio and burst LED's. The green one indicates if the audio stimulation is active, and the blue one indicates if the electrical stimulation is active.

Analog buttons

Only two analog buttons are used on the device: a power-on rocker switch and a start/stop button. The rocker switch includes an LED inside to indicate the power on. The start/stop button is a momentary push button that only temporarily connects the reference voltage to the digital input of the µC.

Digital menu

The digital menu is visible through a 20x4-character LCD screen. A rotary encoder is used to control the digital menu. The menu includes six parameters. For each parameter, a default value is set. By clicking the rotary encoder, the parameter value can be adjusted. This value is incremented or decremented with the step value. If the maximum is reached, the value is not incremented. The same holds for the minimum value when decrementing. The new value is confirmed by pressing the rotary encoder again, as shown in Figure 3.27. At the same time, all menu values are logged in datalog.txt on the MicroSD card, as explained in Section 3.2.4.

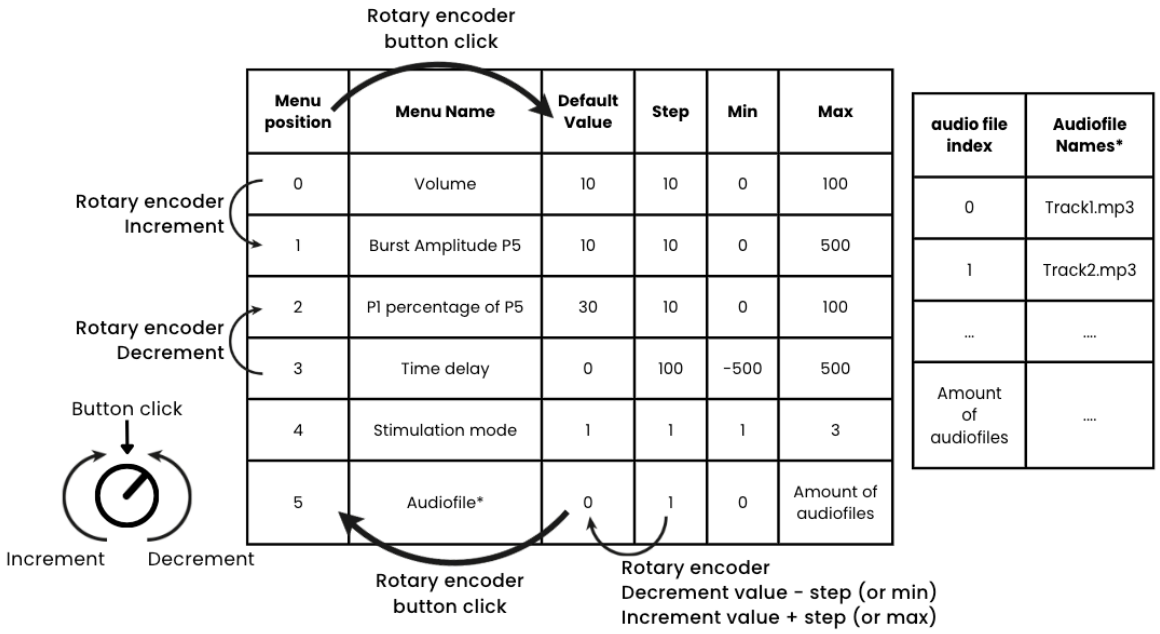


Figure 3.27: Working principle of the digital menu represented by a table

Choosing a digital menu over analog buttons for every setting makes the device very flexible. The menu can be expanded with any value or text used in the code. The only thing needed is a software version update to make this possible.

A function of the Music Maker library adjusts the volume. This works inverse, as 0 is the highest possible volume, and 100 is taken as the lowest volume. The burst amplitude and slope are calculated using the computation shown in Listing 3.1, for which the credits go to the colleague student. .

```
1 for (int i = 0; i < 5; i++) {
2   PulseAmplitudeArray[i] = (((P1+(1-P1)*(static_cast<float>(i)/4))*P5     /
        3) * 4096) -1;
3 }
4 PulseAmplitudeArray[5] = (((P5+P5*P1)/2      / 3) * 4096) -1;
```

Listing 3.1: Burst pulse amplitude calculation

The time delay between electrical and acoustical stimulation is taken from the menu value, which can be positive or negative. A separate function converts the signed value into two unsigned values, one for each delay of the acoustical and electrical stimulation. The synchronicity function then uses these delay values to run the stimulation with the correct delay.

The stimulation modes are selected as 1, 2, or 3, where 1 is correlated, 2 is uncorrelated, and 3 is audio only, as shown in Figure 3.28. Modes 2 and 3 are created as a control group function for clinical research. Mode 1 should be the one to be used as a therapy. Only in stimulation mode 1 is the selected time delay taken into account. In stimulation mode 2, the time delays are created randomly every time a stimulation cycle ends. In stimulation mode 3, burst stimulation is never activated.

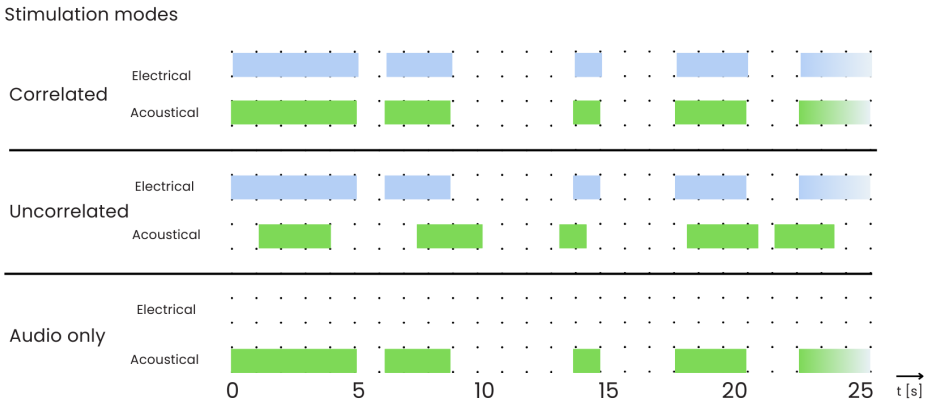


Figure 3.28: Stimulation modes

User interface circuits

The rotary encoder, LCD screen, and start/stop stimulation button are connected to the DualSTIM PCB for practical connectivity. These user interface components are all fixated on the lid of the device's box. To quickly open the box, the components are only connected to the PCB by two connectors. A schematic representation is shown in Figure 3.29.

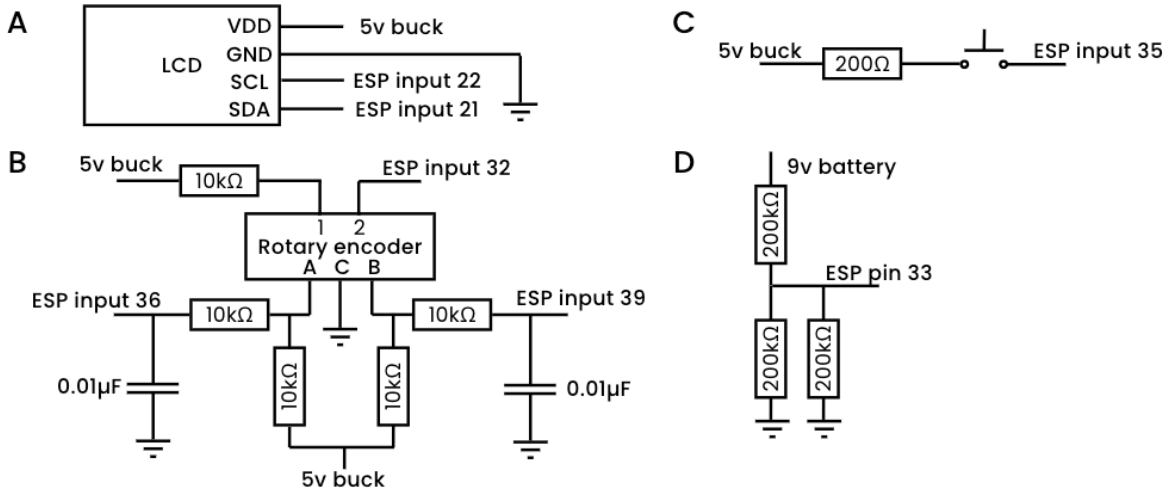


Figure 3.29: Electronic connections of user interface components to the ESP. A. The connections of the LCD screen with the ESP32. B. The connections of the rotary encoder to the ESP. Terminals 1 and 2 detect the rotary encoder's press button action. Terminals A, B, and C are used to detect the rotation of the rotary encoder. C. The start/stop button connection. D. The connections for the battery indicator.

The LCD screen uses I2C communication. It is connected through the PCB to the ESP32 pins 21 and 22 and powered by the 5V buck converter. A rotary encoder walks through the menu and adjusts and confirms menu settings.

The rotary encoder is powered by the 5v buck converter and surrounded by a circuit of two low-pass filters on the A and B terminals. These are 10kΩ resistors connected in parallel to a 0.01µF ceramic capacitors. The rotary button is connected via the PCB to the ESP32. Terminals A and B are connected to ESP32 pins 36 and 39, respectively. The push button pin is connected to ESP32 pin 32.

The 5v buck converter is connected in series to a 200kΩ resistor, the start/stop button, and the ESP32 pin 35. This pin works as a digital input pin.

There is also a battery level indicator on the LCD screen. The battery level is transformed from 9V to 3V by a 200kΩ resistor connected to the battery input pin 33 of the ESP 32. Another two 200kΩ resistors are connected to the ground in parallel from the input pin. Voltage levels up to 7.9V are indicated as 100%, 7.3V as 75%, 6.7V as 50%, 6.1V as 25%, 5.8V as 10%, and 5.5V as 0%. At 0%

battery level. The device cannot provide 3 mA electrical stimulation anymore. The maximum amplitude of the burst stimulation's current drops with a dropping battery level.

MicroSD card

Every patient will have their own unique MicroSD card. At least one tinnitus-matched audio file should be uploaded to the MicroSD card. The first time the MicroSD card is added to the device and the device is powered, a datalog.txt file will be written on the MicroSD card. The datalog.txt file contains the list of default menu parameters. A schematic representation of the datalog.txt file is shown in Figure 3.30. Each time a setting changes in the digital menu, the datalog.txt file will be updated with a new row of parameters. If the device is powered off, and the same MicroSD card is inserted while powering it on, it will load the last line of menu parameters into the device. The maximum burst amplitude of 3 mA is especially restricted. Any value above 3 mA will be blocked, and the LCD will display an error.

		Volume				Burst Amplitude P5				Burst amplitude P1				Time Delay				Mode				Audiofile			
Max stimulation time	Burst frequency	value	step	min	max	"	"	"	"	"	"	"	"	"	"	"	"	"	"	"	"				

Figure 3.30: Datalog file parameters

Connectors

To provide electrical and acoustical stimulation, there must be some connection between the device and the headphone, and the device and the ear clip. Therefore, a 3.5 mm stereo audio jack socket is included in the Adafruit Music Maker and available on the outside of the device. A 2.5 mm mono jack socket is added to connect to the ear clip for electrical stimulation. The ear clip will have two poles connected by the tip and sleeve of the 2.5 mm plug. The H-bridge is responsible for switching the current directions. The electrodes on the clip change their positive and ground poles. The choice of two different plug sizes was explicitly made to ensure the user could not confuse the two sockets and switch them around.

Box design

The first concept design of the device's casing was made for Arduino Uno, which did not suffice at a later stage. The decision was made to use a pre-made, closed box of polycarbonate with a seal quality of IP65. This choice was made in favor of designing a 3D-printed box. The positioning of components in a 3D-designed box would have to be very precise to fit, whereas customizing an existing box would be more accessible and more straightforward for this project. The custom PCB, µC, Adafruit music maker, and battery packs should be mounted in the device. Figure 3.31 shows the sketches of the inside, outside, and placement of components of the device.

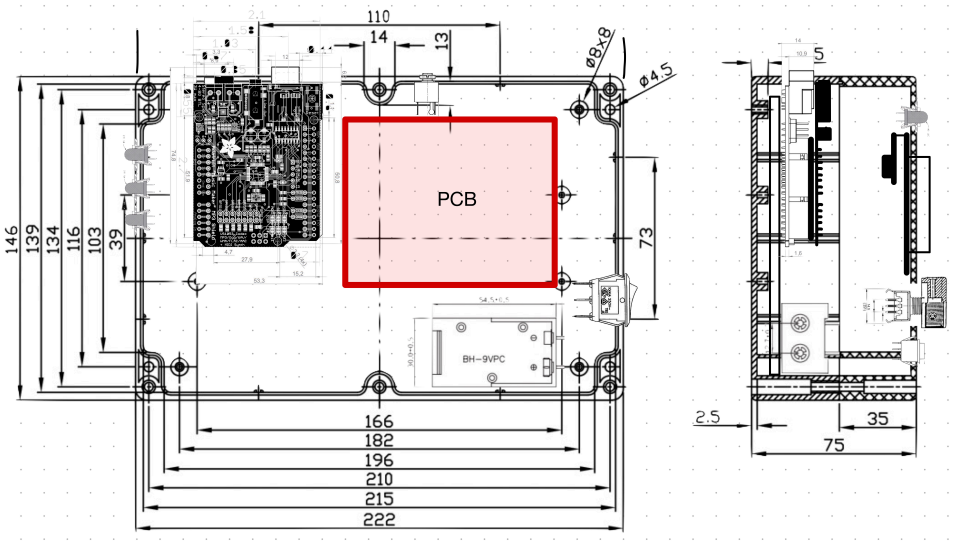


Figure 3.31: First version concept design of the physical casing

The components are placed with the following situation in mind: A clinician and a patient are in the same room. The patient sits in a chair and will receive therapy from that position. The clinician must be able to operate the device and see the acoustical and electrical stimulation on LEDs so that it is possible to check which stimulation mode is active. The patient should not see those LEDs but be closest to the acoustical and electrical stimulation sockets. Jonathan Kneepkens continued with a second design for the device, which included the ESP32. The result of his final design is shown in Figures 3.32 and 3.33

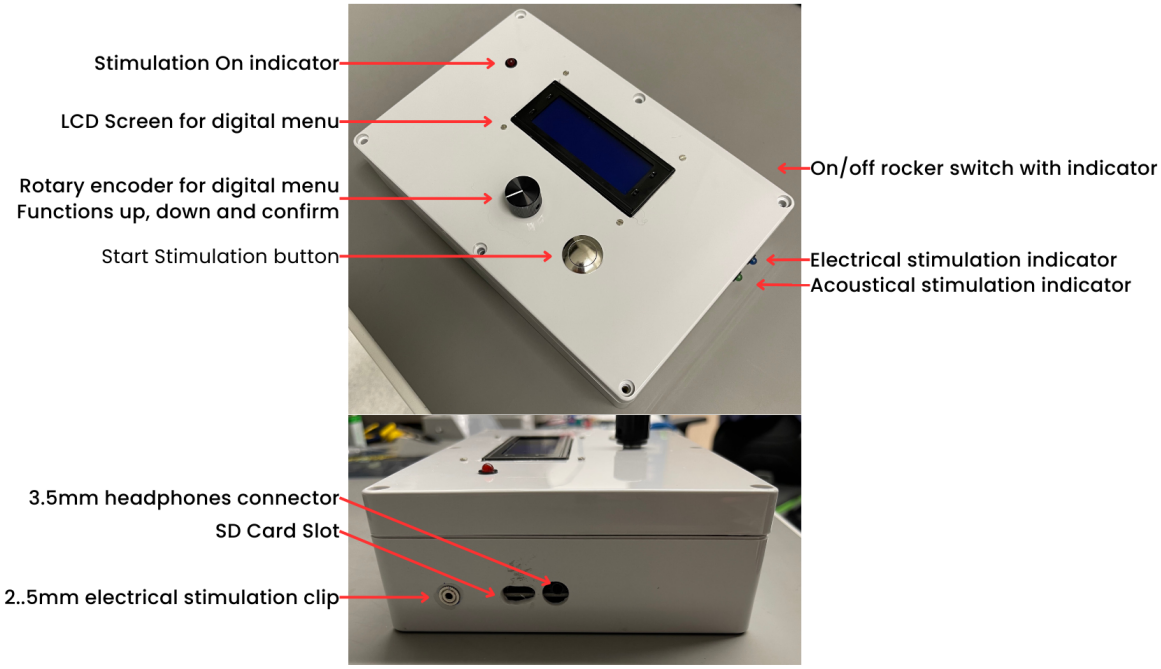


Figure 3.32: User interface components and their descriptions

Figure 3.33: Final device including the ear clip and headphones

Earclip

The vagus-nerve ear clip can be any arbitrary, bipolar transcutaneous electrical nerve stimulation clip, as long as it can be connected with a cable with a 2.5 mm jack. The chosen vagus nerve clip is a bipolar clip with rubber-like contact points for comfort. The exact material is unknown, but it conducts electricity well enough for transcutaneous electrical nerve stimulation [222].

Headphones

The choice of headphones depends on the load impedance of the mp3 player, the frequency range of the headphones, and the comfort of the headphones. The load impedance of the mp3 player is 32Ω. The frequency range should be between at least 20-20kHz. The ear clip should be able to fit under the left headphone or be able to stick under it. The ear clip is only applied to the left tragus, as the right auricular vagus nerve ending is connected to the heart, as explained in Section 2.3.5. The headphones should be comfortable to wear in combination with the ear clip for half an hour. Also, the price should be in a suitable range for the needs. Two headphones were chosen that could meet these criteria.

The Alesis DRP 100 is a headphone with big over-ear ear shells, so the ear clip fits under it. This headphone is designed to play along with music while playing drums. The headphones should be comfortable enough to perform live music on stage, which exceeds 30 minutes easily as required for the device. The impedance ranges between 30-39Ω. The frequency range is approximately 10-14 Hz up to 29-31.9 kHz. The cable length is 1.8 m. The headphone can be connected with a 3.5 mm jack [223].

The Devine PRO 900 DJ/Studio headphones are also over-ear headphones with smaller earshells. The benefit of this headphone is that the shells can rotate so that the pressure on the ear clip can be removed. In that case, the headphones would not close off on the ear but would at least provide comfort to the patient. It is designed for studio/DJ purposes, where the time the headphones are worn is also far-reaching half an hour. The headphones have a 40Ω impedance, providing a high enough volume level to overrule the tinnitus sound. The frequency range is from 10-14 Hz to 25-26.9 kHz. The cable length is 3 m, enough to keep the distance between the device and the patient. The headphone is connected to the device with a 3.5 mm jack [224].

3.3. Software design

3.3.1. Software design process

Software is written with the idea of reliability, extendability, performance, and safety. For object-oriented programming, software can be divided into services on a high level and subservices on a low level. The design patterns used to solve problems are embedded in numerous different topologies. However, because the Arduino programming language is not particularly object-oriented, Dr.Ir.C. Strydis advised starting with a small functionality and building up from there. This opposes defining all services and sub-services and creating an entire architecture before writing any software. The architecture will be created in a professional setting using Unified Modeling Language. A mixture of these methods is used, combining block schemes with an iterative process of designing the software.

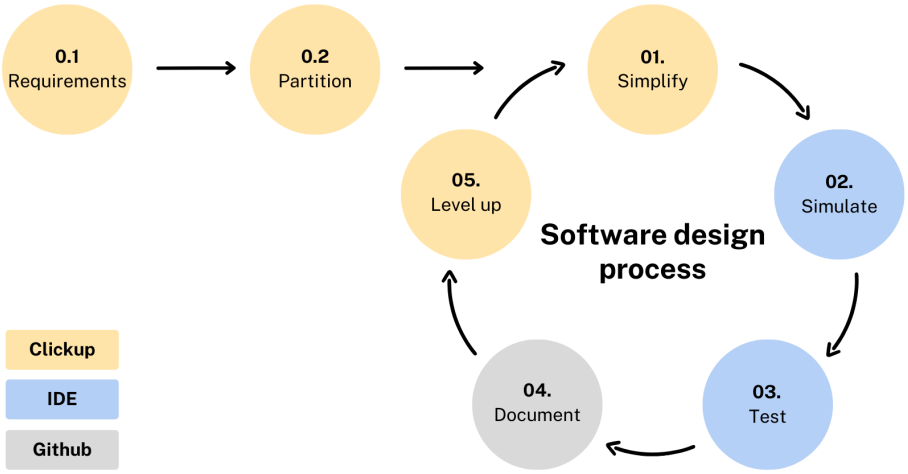


Figure 3.34: The design process of the software is done by separating, simplifying, and simulating functionalities in an iterative process.

Figure 3.34 shows a schematic representation of the software design cycle. First, block schemes of software parts are defined from the device requirements using Clickup as a ticket board and documentation software. Second, each part is simplified and implemented using Tinkercad, an Arduino simulation program, or an Arduino with a breadboard. Third, the functionality is then tested and compared with the requirements. Fourth, the iteration (code) result is registered using Github. These steps are then repeated for the next iteration, adding a slight functionality improvement to the code. After a certain number of iterations, the parts of the code are finalized, tested, and compared with the requirements. This partial code is then added to the main code using Github's merge functionalities.

3.3.2. Software overview

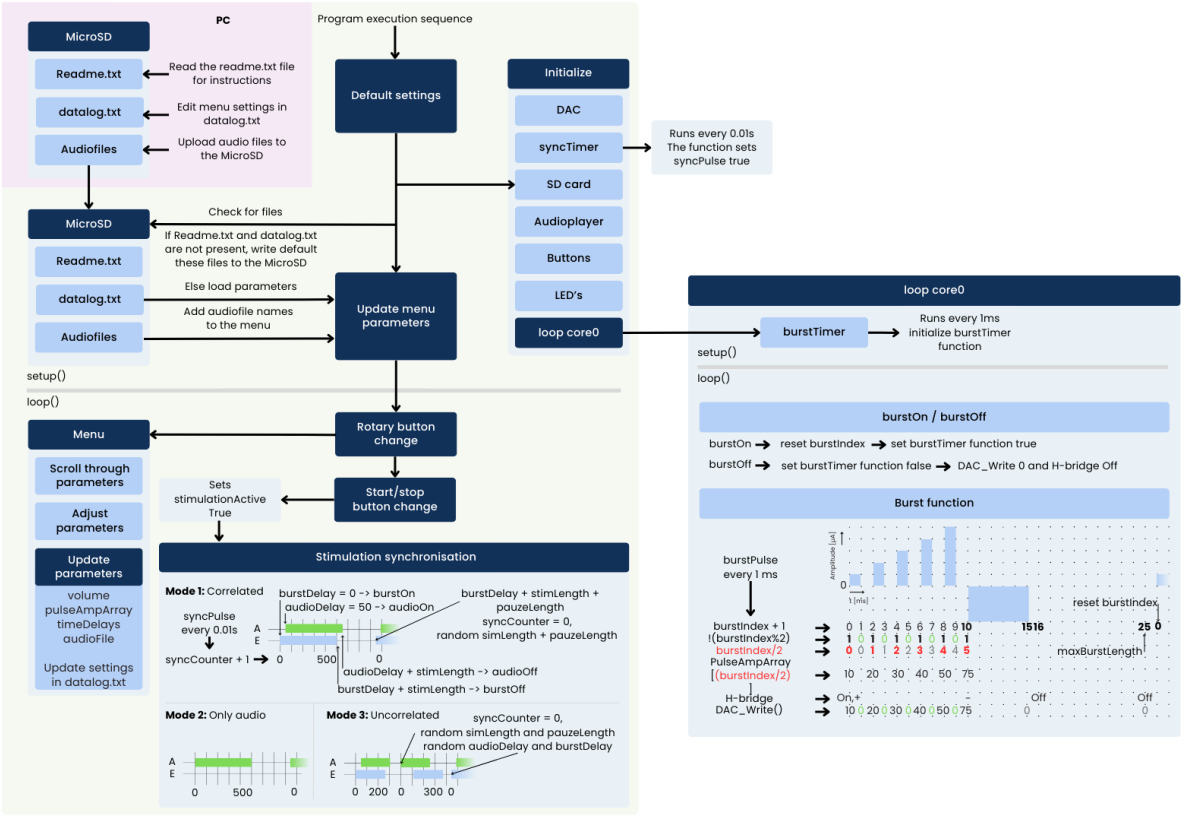


Figure 3.35: Schematic overview of the software execution flow

Figure 3.35 shows the final result of all iterations. Two critical elements of Arduino programming are the setup and loop functions. The setup function is run once, while the loop function is repeated over and over again. Still, the order of tasks in the loop function is respected. For the Arduino, this runs only on one core. Using the Arduino environment, the ESP32 can run two different setups and loops in two different cores.

In this software program, three main functionalities are essential: the user interface, the burst stimulation function on the second core, and the synchronous start and stop criteria for the electrical and acoustic stimulation. The program uses two cores and two timers. The burst function is very time-specific. Therefore, it is specifically run on the second core to prevent interrupting time-non-specific functions such as the user interface. No delay functions are used in the main loop to ensure the program runs continuously, meaning that the user interface is still available to change settings while stimulating burst and audio. In this section, the order of the program will be discussed per part. The complete code is available in Appendix D.

3.3.3. Initializing components

Many variables, libraries, and pin definitions are written in the memory each time the device is powered on. The initialization process is shown in Figure 3.36

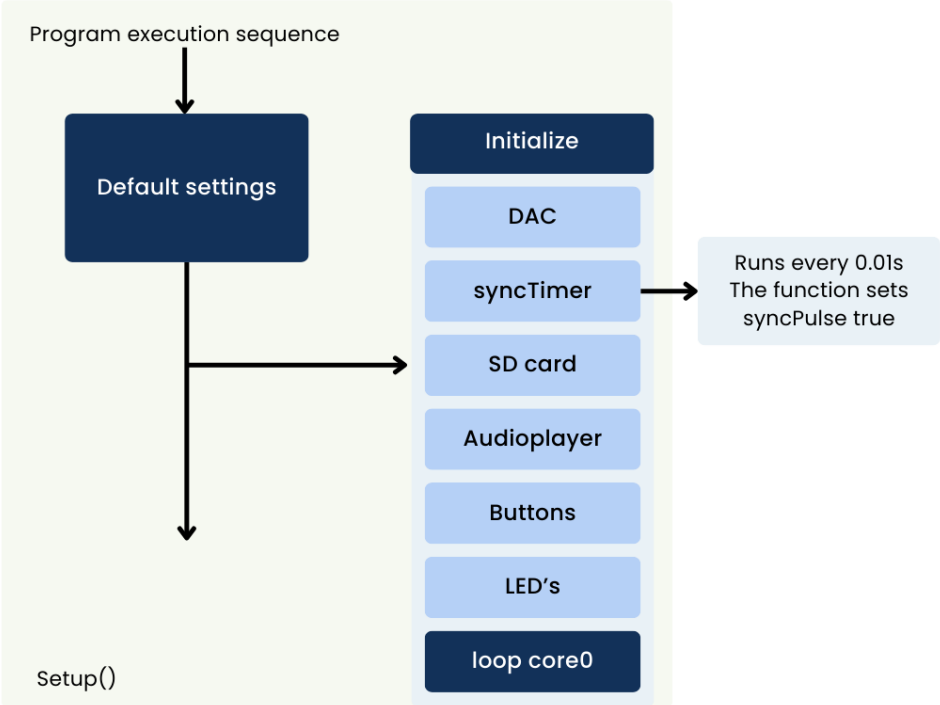


Figure 3.36: Initialization of all components in the setup function

The menu positions, length, default values, battery level, audio files, synchronization variables, and burst parameters are allocated and assigned in memory. Libraries are included for SPI communication, the Adafruit Music Maker, the MicroSD card reading and writing, the LCD screen, rotary encoder for the user interface, and the button click functions. Pins are also defined for these components, including the LED indicators and battery level. The LCD is initialized by its class. The Adafruit Music Maker is also initialized using a class and the pin definitions. The timers are already allocated and later used for the burst function and the synchronization of the start and stop commands of the burst and audio. The synchronization timer runs every 0.01 ms. It's corresponding interrupt service routine function, shown in Listing 3.2 is also created before the setup starts. This is a necessity to let the processor find the particular function. If the function were placed after the setup function, it would not work. The same holds for the burst interrupt service routine function. Both only flag a boolean syncPulse and burstPulse to true for the synchronization and burst functions, respectively. These booleans are directly set to false after the main loops runs the specific function.

```
1   /*! ISR Synchroniser
2     * @brief Set syncPulse true, timer based interrut service routine
3     * @param syncPulse is a flagged true every time the function runs
4     */
5     void IRAM_ATTR synchroniser() {
6       syncPulse = true;
7     }
8
9   /*! ISR Burstsequence
10    * @brief set burstPulse true, timer based interrut service routine
11    * @param burstPulse is a flagged true every time the function runs
12    */
13    void IRAM_ATTR burstSequence() {
14      burstPulse = true;
15    }
```

Listing 3.2: Interrupt service routine functions for both the synchonisation of stimulation and the burst waveform

As one of the last functions, the loop for core0 is initialized, which is responsible for the burst function. This is shown in Listing 3.3 .

```
1       // Core0 loop initiation
2       TaskHandle_t LoopCore0;                         // Task object to run
            electrical stimulation on core0
```

Listing 3.3: Initialization of the taskhandle of loop core0

3.3.4. Burst stimulation through Core0

The second core, core0, can be initialized using the task handle in the `void setup()` function of core 1. The initialized function called `void loopCore0()`, comparable to `void loop()`, is placed under the original `loop()` function. With the initialization shown in Listing 3.4, stack size and priority can be given to the core. If the stack is too small, there is insufficient temporary memory to calculate functions, and the watchdog will be alerted and reset the ESP32. The initialization and loop processes of core0 are shown in Figure 3.37. .

```
1       /* init core0 */
2     delay(1000);                                      // Wait for a
          second before initializing the second core
3     xTaskCreatePinnedToCore(                          // Add the task to
          core0, used for burst stimulation
4     loopCore0,          /* Function to implement the task */
5     "LoopCore0",        /* Name of the task */
6     4096,               /* Stack size in words */
7     NULL,               /* Task input parameter */
8     3,                  /* Priority of the task (higher is higher prio) */
9     &LoopCore0,         /* Task handle. */
10    0);                 /* Core where the task should run */
11    delay(1000);
```

Listing 3.4: Pin the created task to core0

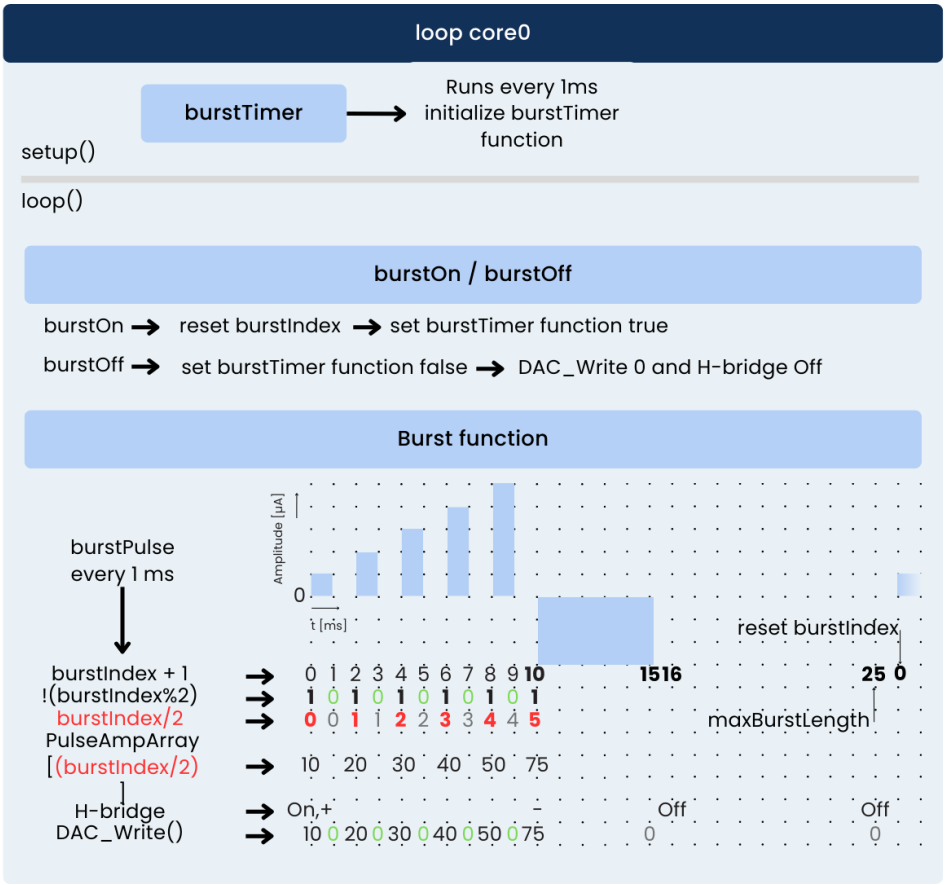


Figure 3.37: Burst function running on core0

Inside `loopCore0()`, first, the second timer for the burst is initialized but not activated yet. This happens outside of `for(;;)` such that the function only runs once. The `burstTimer` runs every 1 ms and flags `volatile bool burstPulse` to `true`. After the program enters the `for(;;)` loop, which keeps running, it checks the `volatile bool burstOn` flag, which will be activated by core1 if the burst stimulation is activated. If this happens, the `if` statement is entered where ISR `IRAM_ATTR burstSequence()` is attached to the timer. From this moment, the `burstPulse` will be set `true` every millisecond, the `volatile uint8_t burstIndex` resets to 0, and the list of if statements can determine which action should be run.

The burst function, or list of if statements, is entered every time `burstPulse` is `true` and sets `burstPulse` to `false` immediately after the `if` statement is entered. The goal is to write the correct value of the `uint16_t* PulseAmplitudeArray` to the DAC through SPI, such that a burst sequence will be provided to the electrodes on the patient's tragus.

The amplitude values of the burst are calculated by the values the user added to the digital menu. The `PulseAmplitudeArray` is created and filled with default values at power, overwritten based on datafile.txt on the MicroSD card, and updated every time a parameter is adjusted in the digital menu.

If `burstIndex` is 0, the H-bridge is switched on with a positive current direction through the load. The first value of the `PulseAmplitudeArray` will be written to the DAC. The `burstIndex` will be incremented by one. Every time the remainder of `!(burstIndex%2)` is zero and the `burstIndex` is lower than 10, a zero will be written to the DAC. When the `burstIndex` is 2, the `burstIndex/2` equals 1, and `!(burstIndex%2)` equals 1, the second value in `PulseAmplitudeArray` will be written to the DAC. This pattern will be repeated until the `burstIndex` reaches 10.

When the `burstIndex` is 10, the H-bridge is switched from one current direction through the load to the other, and the sixth value in `PulseAmplitudeArray` is written to the DAC. At a `burstIndex` of 15, the DAC output voltage is zero, and at 16, the H-bridge is turned off, and both electrodes are connected to the ground for passive charge balancing. For safety reasons, the H-bridge is switched a millisecond later. This is due to the charge still present at the electrodes. Immediately switching the H-bridge such

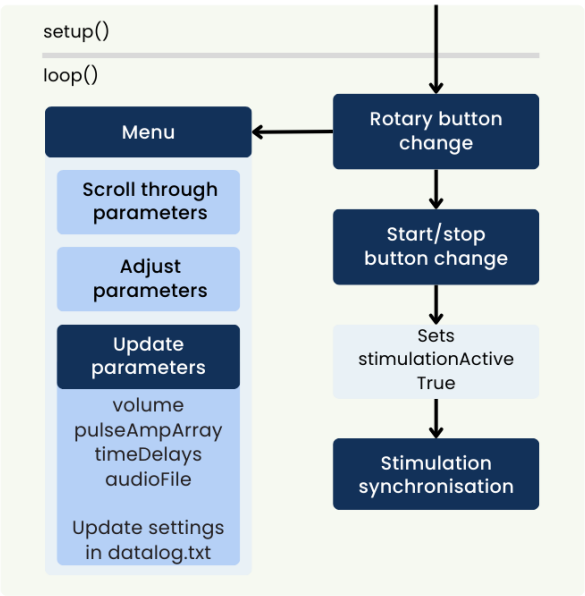


Figure 3.38: Overview of functions running in the loop of core1

that the load is connected to the ground on both sides while the DAC output voltage is changed to zero could cause a positive peak current that dissipates fast. The positive peak current is uncontrolled and undesirable.

When `void IRAM_ATTR synchronizer()` flags the `volatile bool burstOff` to `true`, the timer interrupt is detached, the DAC outputs voltage is zero, and the H-bridge is closed and connected to ground.

At the end of the `for(;;)`, but just inside, it is necessary to call the function `vTaskDelay(1)`, to make sure the core can do its housekeeping and clear its stack, such that it is not overflowing and alarming the watchdog.

3.3.5. Functions on core1

Core1 is the processing core where the default Arduino program runs, shown in Figure 3.38. This core is responsible for the user interface, the acoustic stimulation through the Adafruit Music Maker, and the electrical and acoustic stimulation synchronization.

The rotary encoder initiates a change in the digital menu. Once in the loop, the rotary encoder checks if it is being rotated or pressed. When the rotary encoder is turned clockwise, the `uint8_t menuposition` is incremented by 1. By clicking the rotary encoder, `bool adjustSettings` is set to `true`, and the parameter value can be adjusted, indicated by a blinking cursor on the screen. By rotating clockwise or anticlockwise, the `int16_t value[menuPosition]` is increased or decreased by the `int16_t step[menuPosition]`. If the rotary encoder is pressed again, `adjustSettings` is set to false again, and `value[menuPosition]` is updated. The value array is signed because the time delay can be both positive and negative. The difference between a signed and unsigned integer is that the signed integer can be both positive and negative, while the unsigned integer is only positive. The number of bits of the integer determines the number of values it can represent. The maximum value of a signed integer is lower than that of an unsigned integer, because the signed integer can represent the same amount of values and the negative ones.

Changing `value[menuPosition]` alone is not enough to push the changes through to the correct functions. Additional computations must be done for the volume, `PulseAmplitudeArray` (burst amplitude and slope), time delays, and audio file to change correctly. For example, the `PulseAmplitudeArray` should be calculated again using the function `COMPUTE_PulseAmplitudeArray(float P1, float P5)`. At the same time, the datalog.txt file should be updated with a new line of menu parameters. This is done by the function `dataLog()`. At every change of the rotary encoder, the LCD screen is updated with the new values, and also checking the battery level and printing on the screen.

The start-stop button is also checked once in the loop. If the start/stop button is clicked, the function

`startStopButtonHandler(Button2& startStopButton)` checks if the device is changing any settings at that moment. If that is not the case, the stimulation is activated. `boolean stimulationActive` is set `true`, `stimulationTime` and `volatile uint16_t syncCounter` are set to 0, and `synchroniser()` is attached to the `syncTimer`. `stimulationTime` keeps track of the temporal length of the stimulation session. By default, the stimulation is stopped after 30 minutes. Suppose the start/stop button is pressed again, and the stimulation is active. In that case, the stimulation is stopped, the DAC output is zero voltage, the H-bridge is disconnected from the source, and both sides of the load are connected to the ground and stop the music player.

3.3.6. Synchronisation and modes of stimulation

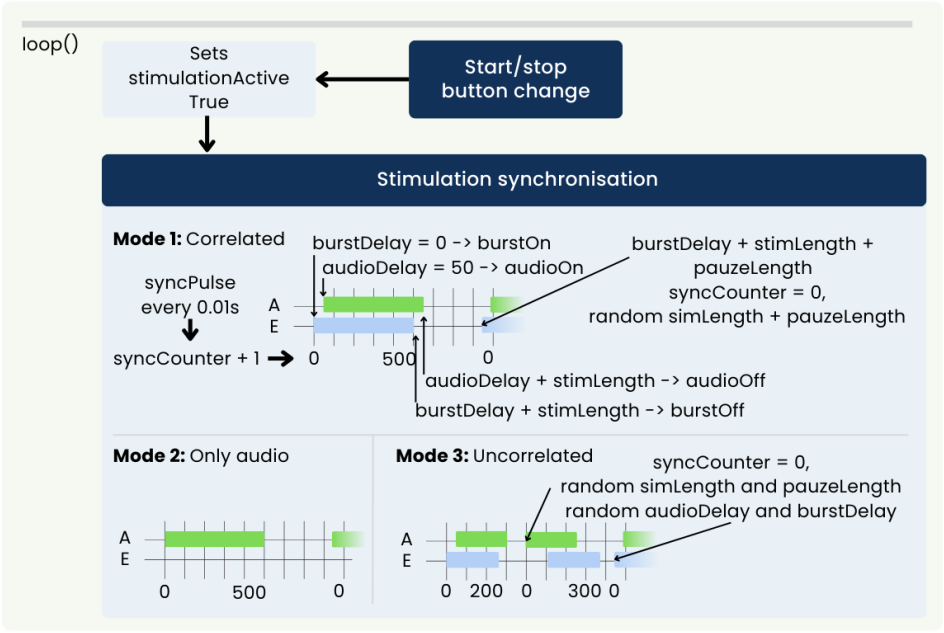


Figure 3.39: Synchronisation and modes of stimulation

During stimulation, `synchronizer()` flags `volatile bool syncPulse` true every 0.01s. The synchronization function is a collection of if statements looped through after the button loops, as shown in Figure 3.39. The if statements are entered when `stimulationActive` is `true` and `syncPulse` is `true` after which the `syncPulse` is directly set `false` again to make sure the if statement is not entered every iteration of `loop()`. `volatile uint16_t syncCounter` is used to keep track of the time for the synchronization of the stimulation.

To generalize the starting point of the audio and burst with the time delay in mind, `volatile bool burstOn` and `volatile bool audioOn` are set `true` if the `syncCounter` equals the previously computed `uint16_t burstDelay` or `uint16_t audioDelay`. These are calculated from the time delay setting in the digital menu and can only be a positive value. The stimulation length and pause length vary every stimulation cycle between 1 and 5 seconds using `random(100,500)`. `volatile bool burstOff` and `volatile bool audioOff` are flagged true if `syncCounter` equals `burstDelay + stimLength` or `audioDelay + stimLength`. When also the `pauseLength` is reached, the last of audio or burst to finish will be the if statement that determines the new `stimLength` and `pauseLength`.

In the case of stimulation mode 2, `burstOn` and `burstOff` are excluded and thus never set true, resulting in no electrical stimulation.

In the case of stimulation mode 3, called uncorrelated stimulation, at the end of `pauselength`, also the `burstDelay` and `audioDelay` are randomized with `random(0,200)`. This results in a different start and endpoint for the burst and audio stimulations. At the end of the burst and audio, there is a check which delay was the greatest, and based on that, the new `stimLength` and `pauseLength` are calculated at the end of the latest stopping stimulation modality.

3.3.7. Safety and errors

Safety measures are taken not only in the design of the stimulation circuit but also in the software. One of the consequences of these safety measures can be that the device displays an error and stops the stimulation. All errors can be found in the Appendix C, section F.

Errors are continuously shown in the LCD by using `while(1);` in the if statement of the error. The user needs to consult the user manual for possible solutions. The device should be restarted to resolve the error if the solution is found.

If the stimulation is active, it will be turned off by any error using `void stimulationOff()`. This function detaches `syncTimer()` from the interrupt timer, sets `burstOff` to `true` if the burst LED is on, and stops the audio if it is playing. When `burstOff is true`, the `burstTimer` interrupt is detached from the timer, the DAC is written to zero, the load is connected to ground on both sides of the H-bridge, the H-bridge is turned off, and the burst LED is turned off.

On device startup, all components are initialized. First, the LCD is initialized to ensure error messages can be written to the LCD screen. Second, the SPI protocol and the DAC are also initialized for the electrical stimulation. A zero value is written to the DAC directly after initialization to ensure no current running towards the current source. Then, the H-bridge is turned off, and the load is connected to the ground on both sides of the H-bridge. This will not be changed until the burst stimulation is activated on core0. If the electrical stimulation is started without reviewing the device settings, the default maximum amplitude of the burst stimulation is set to 10 µA.

One of the essentials is the Adafruit Music Maker. If there is a technical problem, an error is displayed on the LCD. The Adafruit Music Maker works with an interrupt pin. An error will be displayed on the LCD screen if the interrupt pin is incorrect or not connected. The Adafruit Music Maker will play a 440Hz tone through the headphones to indicate that it works.

The audio files are loaded from the MicroSD card and filtered by various file extensions. The names are copied into an array of strings. An error message is displayed on the LCD screen if no MicroSD card or audio files or more than ten audio files are present.

If the datalog.txt file is not present, it is written on the MicroSD card. The menu parameters are updated from the datalog.txt file if it is present. If the device cannot read the datalog.txt file, it will throw an error on the LCD screen. The same holds if the device can not write readme.txt to the MicroSD card. An error will be shown if the max amplitude of 3 mA is exceeded in the datalog.txt file. The error will also be displayed if a decimal point is misused. Datalog.txt allows you to adjust the burst frequency, which ranges from 0.1 Hz to 40 Hz. If the value exceeds this frequency range, an error will be displayed.

If the MicroSD card is ejected or a missing audio file is selected while the stimulation is on, the stimulation will be turned off, and an error message will be displayed.

If the start/stop button is pressed while a parameter is adjusted in the digital menu, the user is asked to confirm the setting for 2 seconds before returning to the menu.

Every time the menu changes position or value, the battery is checked for its level. If the battery level is low, the stimulation is turned off, and an error is displayed on the LCD screen.

3.4. Measurements

3.4.1. Device output measurements

The device's measurements were done using a Tektronix TBS 2000 Series oscilloscope. In place of the headphone connection, the acoustical stimulation is connected to the oscilloscope at the end of a double-wired cable without a load impedance. One wire of the cable is connected to the probe of the oscilloscope, and the other is connected to the probe's ground clip.

Electrical stimulation is measured using different methods. The first method is used when only electrical stimulation is measured. The second method is used when electrical and acoustical stimulation are measured simultaneously.

When measuring only electrical stimulation, a 1 kΩ resistor serves as a load and is connected in series with the current amplifier. The positive connector of the Keithley 428 current amplifier is connected to the resistor. The negative clamp is connected to a ground pin on the PCB. A BNC cable connects the current amplifier to the oscilloscope.

When measuring acoustical and electrical stimulation simultaneously, the acoustical stimulation measurement connections with the oscilloscope stay the same. The electrical stimulation is measured using a second channel on the oscilloscope. The probe is connected to the output pin of the DAC and

grounded on a ground pin of the PCB.

3.4.2. User interface test

To test the user interface, an instruction was written based on the intended user interaction flow, see Figure 3.25. Each flow step is done at least once and could be rated on a scale of 1 (not user-friendly) to 5 (very user-friendly). MSc F. Varkevisser conducted the test.

During the demonstration, Prof. Dr. de Ridder was shortly instructed on the operation of the device and presented the user manual, which can be found in Appendix C. After the short instruction, Prof Dr. de Ridder operated the device's digital menu independently.

4

Results

In Chapter 3, the requirements for the device are discussed in detail. From the requirements, the device is based on a command decoder, an electrical stimulation circuit, an acoustical stimulation circuit, and a user interface. Based on the hardware, the software was written to fulfill the device's requirements. Finally, the measurement setup and user interface test for validating the device are described in Section 3.4.

In this chapter, the physical device is presented in pictures, followed by measurement of the device outputs based on different user inputs using the user interface. An adjustment for each menu setting is added to the results. The user interface is also tested separately. A description of the results is added to this chapter. Finally, the device is also demonstrated to Prof. Dr. de Ridder in the BRAI3N clinic. A summary of the demonstration is also added to this chapter.

4.1. The device

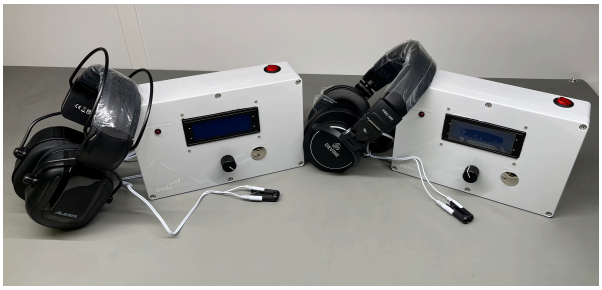


Figure 4.1: Two devices are constructed for the clinical research

In a joint effort with Jonathan Kneepkens and Brian Nanhekhan, two devices were constructed for use in clinical studies, see Figure 4.1. The devices are robust, and all components from the user interface requirements are added. There are no loose wires visible on the outside. The device includes a power on, start/stop button, digital menu operated with a rotary encoder, a stimulation indication LED, a burst and audio stimulation LED, connections for the ear clip and headphones, and a MicroSD card slot.

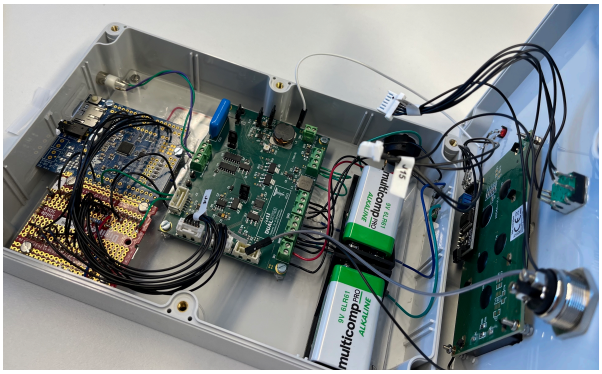


Figure 4.2: The inside of the device.

The device consists of an ESP32 microcontroller, a custom stimulation circuit, an Adafruit Music Maker shield, and a user interface, see Figure 4.2. The box lid can be detached using a screwdriver and disconnecting four connectors. The batteries can be exchanged without detaching the four connectors.

4.2. Device outputs and user interface adjustments

4.2.1. Burst amplitude and slope

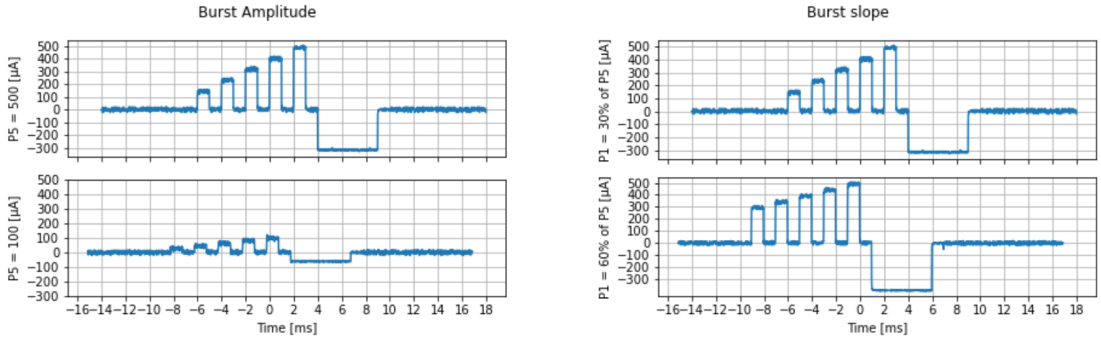


Figure 4.3: Burst amplitude and burst slope at different parameter settings

The burst stimulation has a pulse width of 1 ms and a pulse delay of 1 ms. The five pulses are increasing in amplitude by default, and the charge is balanced by one pulse of 5 ms and an amplitude of the average of the five pulses.

The electrical stimulation can be modified in two ways, see Figure 4.3. The amplitude of the burst waveform's fifth pulse (P5) can be adjusted. While adjusting this pulse, the other pulses will be adjusted too. The second parameter that can be adjusted is the percentage of P5 that results in the amplitude of the first pulse (P1), as explained in Section 3.2.4, Listing 3.1. The second (P2), third (P3), and fourth pulse (P4) will also be adjusted according to this percentage. So, if the amplitude of P5 is adjusted, the amplitude of P1, a percentage of P5, will adjust according to the percentage. If the percentage is changed, the amplitudes of P1 to P4 are adjusted according to this percentage, which could be seen as the slope of the burst. The percentage explicitly does not change the amplitude of P5, and the amplitudes of P1 to P4 can not be higher than P5. This gives the user flexibility in the intensity of the burst with two different parameters. For both the amplitude of P5 and the percentage, the charge balancing pulse is adjusted according to the sum of the five pulses.

4.2.2. Simultaneous stimulation

The electrical and acoustical stimulation is provided simultaneously, as shown in Figure 4.4. Sidenote: The electrical stimulation in Figures 4.4 to 4.6c are shown in voltage, the output of the DAC. The oscilloscope measurements on the electrical and acoustical outputs simultaneously resulted in distorted graphs. Section 5.3.1 will further discuss these distorted graphs.

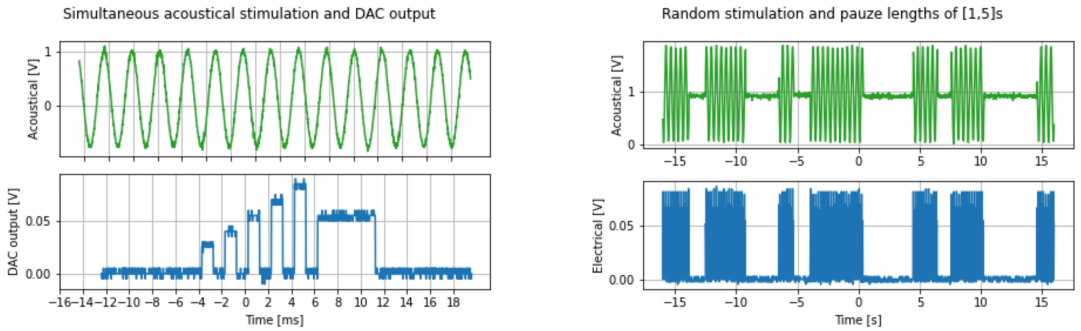
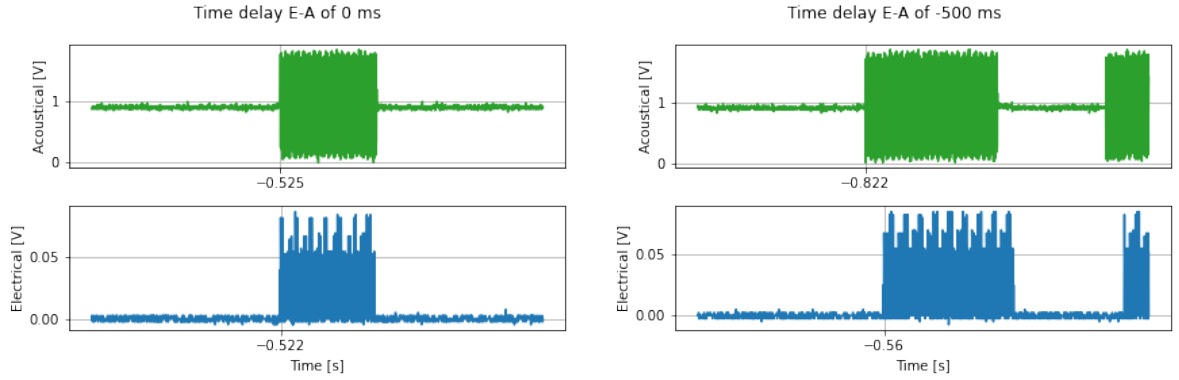


Figure 4.4: Simultaneous and synchronized stimulation

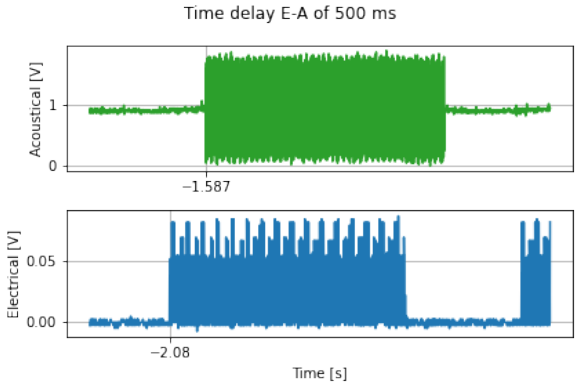
The electrical and acoustical stimulation start and stop simultaneously, thus the stimulation lengths are equal. The temporal lengths of stimulation and pauses are randomized between 1 and 5 seconds, as required.

4.2.3. Time delay



(a) Time delay setting of 0 ms results in a delay of 3 ms

(b) Time delay setting of -500 ms results in a delay of -262 ms



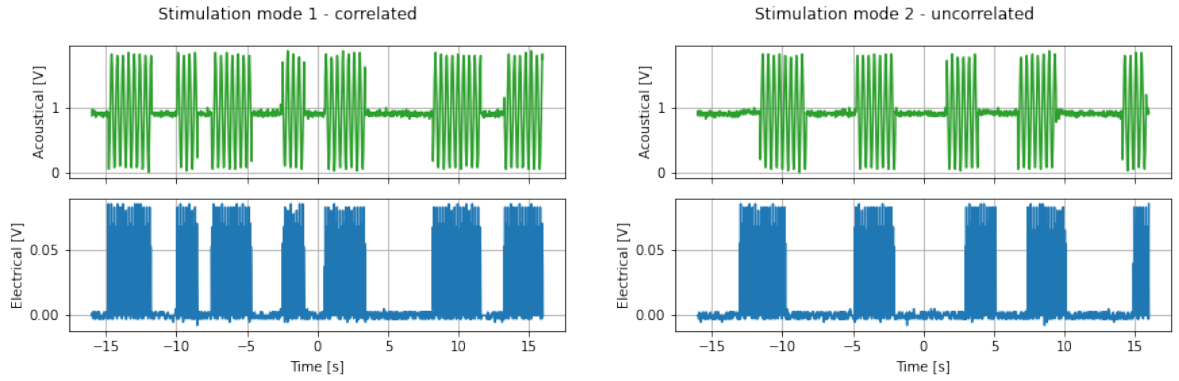
(c) Time delay setting of 500 ms results in a delay of 507 ms

Figure 4.5: Time delay electrical and acoustical stimulation

By default, the time delay between the electrical and acoustical stimulation is 3 ms, as shown in Figure 4.5a. If the time delay value in the digital menu is set to -500 ms, the acoustical stimulation leads the electrical stimulation by 262 ms, as shown in Figure 4.5b. Unfortunately, this is less than the intended 500 ms delay. This will be further discussed in Section 5.3.2 If the time delay value in the digital menu is set to 500 ms, the electrical stimulation leads the acoustical stimulation by 507 ms, as shown in Figure 4.5c.

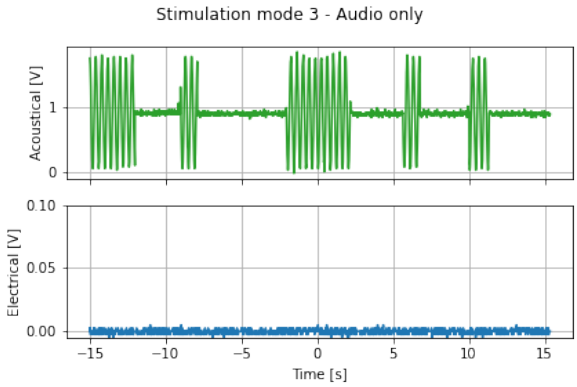
4.2.4. Stimulation mode

In stimulation mode 1, the electrical and acoustical stimulation is provided simultaneously, with randomized lengths and pauses of the same lengths, as shown in Figure 4.6a. In stimulation mode 2, the electrical and acoustical stimulation is provided, but not at the same moments and lengths, as shown in Figure 4.6b. The stimulation lengths are equal, but the pause and starting points differ, providing uncorrelated stimulation. In stimulation mode 3, only acoustical stimulation is provided, and electrical stimulation is turned off, as shown in Figure 4.6c.



(a) Stimulation with correlated stimulation patterns

(b) Stimulation with uncorrelated stimulation patterns



(c) Stimulation without electrical stimulation patterns

Figure 4.6: Stimulation modes

4.3. User interface test

The results of the user interface test are shown in Appendix E. The use of the Micro SD card, uploading audio files to the Micro SD card and inserting them into the device is very intuitive for F. Varkevisser. Also, starting/stopping and changing parameters are found to be intuitive, although the question of whether it is desirable to change settings during stimulation is raised. The indication for the battery at the left top corner of the LCD screen is not intuitive, as it was said to be ”confusing”, see Figure 4.7.

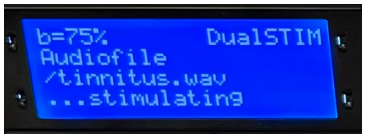


Figure 4.7: Battery level indication at the left top of the LCD screen.

Adjusting parameters in the datalog.txt file on the MicroSD card was found to be not intuitive. The comment was made that it is hard to read which parameter has been changed, and thus, it is prone to errors. The maximum allowed ranges for the values are unclear, which could break the device’s

functionalities. A suggestion was made to create a software user interface on the computer to change the menu parameters and upload the audio files, such that there would be another safety check to do this.

During the demonstration, Prof. Dr. de Ridder operated the digital menu after a short instruction. He called it ”very intuitive” to change settings in the digital menu.

4.4. Demonstration at the BRAI3N clinic

On January 19th, 2024, the device was demonstrated to Prof. Dr. de Ridder at the BRAI3N clinic in Gent, Belgium, as shown in Figure 4.8. During this visit, the device was connected to a portable oscilloscope. The burst stimulation wave and the acoustical stimulation of a 440 Hz sine wave were shown, similar to Figure 4.4. After a short introduction to the operation of the device, Prof Dr. de Ridder tested the electrical stimulation efficacy. The first time the electrical stimulation was applied to Prof Dr. de Ridder, the conclusion was drawn that the maximum amplitude of 3 mA and a step of 100 µA was too big. This was based on the fact that the intensity of the electrical stimulation should be just below the threshold of feeling it tingling, which was impossible at 100 µA. The sympathetic nervous system will be activated if the electrical stimulation is painful. This was also investigated during the demonstration. The device settings were quickly adjusted through the datalog.txt file on the MicroSD card. The settings were changed to a step of 10 µA and a maximum amplitude of 500 µA.

The parasympathetic nervous system will be activated if the burst stimulation on the vagus nerve through the tragus is effective. The heart rate variability can indicate the activation of the parasympathetic nervous system [225]. These tests came out positive, where the heart rate variability changed in the right direction, indicating the parasympathetic nervous system was activated. A conclusion on the efficacy of the device as a treatment option for tinnitus has to await the clinical studies.

Figure 4.8: Prof. Dr. de Ridder receives vagal nerve stimulation and checks the heart rate variability

5

Conclusion, contributions and recommendations

5.1. Conclusion

After extensive literature research, interviews on the requirements, the design process, and testing of the bimodal stimulator, the main question can be answered:

How can a user-friendly, portable, flexible, extendable, and safe bi-modal stimulator for tinnitus be designed as an experimental platform for translational research to effectively stimulate multiple modalities simultaneously, allowing for the testing of a wide range of hypotheses?

By combining the functional requirements, user interface requirements, safety requirements, and the demand for flexibility, the design process was pushed toward a direction where the device provides a flexible and easy-to-use UI with small components, such as the µC, which could be extended with stimulation circuits for other modalities and is portable and safe at the same time.

The design of this bi-modal stimulator shows a method to design a device capable of stimulating multiple modalities simultaneously or with a controllable delay between the modalities. It contributes to translational research as the device will be used in clinical trials to find out if and which parameter combinations of burst stimulation on the vagus nerve through the tragus paired with tinnitus-matched sound can rewire the brain to detach salience from the internally generated tinnitus sounds.

5.2. Contributions

This master thesis is a contribution to the latest insights in tinnitus research. It is not the first bi-modal stimulation device ever made. Still, it distinguishes itself from, for example, the Lenire by the flexibility to use any audio file in combination with burst stimulation on the tragus. The device offers a large parameter space to use for research, such as the possibility to adjust the burst amplitude, burst slope, control group stimulation modes, and time delays between the acoustical and electrical stimulation. The device gives the clinical researcher an easy-to-use device to search for and optimize the correct settings for tinnitus treatment.

5.3. Recommendations

5.3.1. Results

Figures 4.4 to 4.6c in section 4.2 are displayed as the DAC voltage output. When both the ear clip and headphone outputs were connected to the same oscilloscope, the burst stimulation showed a distorted signal, see Figure 5.1.

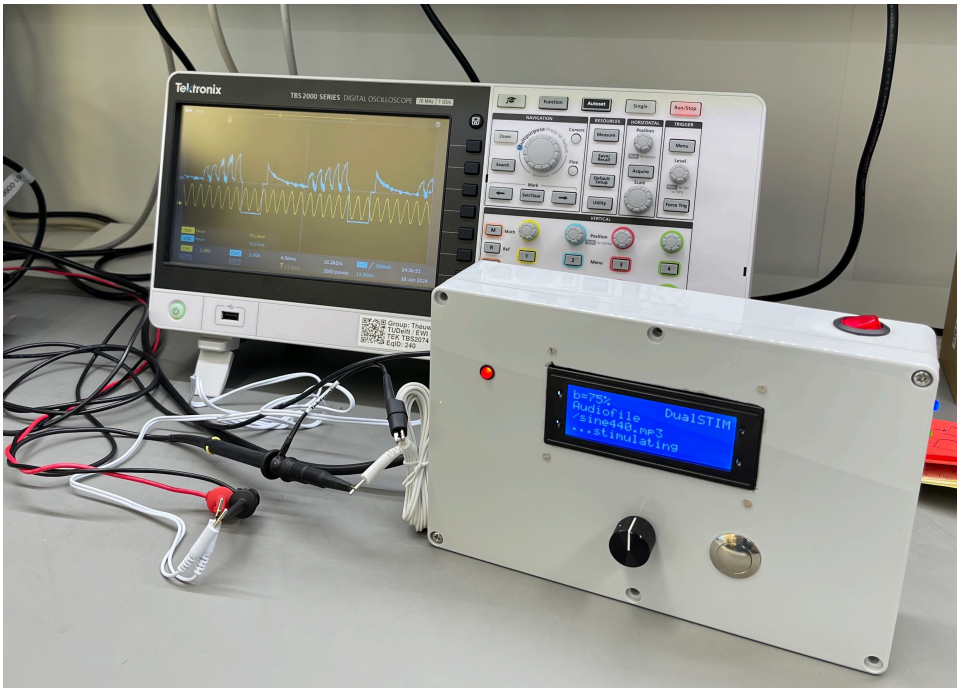


Figure 5.1: Distorted burst: Electrical stimulation [mA] in blue, acoustical stimulation [V] in yellow.

The oscilloscope is a Tektronix TBS 2000 Series. The acoustical stimulation is connected to an oscilloscope at the end of a double-wired cable. One side is connected to the positive, and the other is connected to the ground connector of the probe. There is no headphone or load impedance connected in series to the oscilloscope. A similar cable connects the electrical stimulation output to a current amplifier without a load connected in series. A BNC cable connects the Keithley 428 current amplifier to the same oscilloscope. The current amplifier and oscilloscope are connected to the power network by the same extension cord, sharing the same ground.

The origin of the measurements has not been investigated thoroughly. The fact that no load impedance is added to the measurement could be significant. The burst signal shows a slow settling time, where something in the circuit could work as a low-pass filter. This is only true for the positive stimulation direction.

When `burstindex == 15`, the DAC is writing 0, but the H-bridge is still switched to a negative current direction. When `burstindex == 16`, the H-bridge closes, and both sides of the load are connected to the ground. A positive peak current occurs at this moment, which dissipates pretty slowly. The absence of the load impedance could imply that during the switching of the H-bridge, the leftover charge at the negative side of the H-bridge is dissipating towards the positive side, resulting in a positive current peak. The current amplifier counts as a load impedance in this case.

These distorted figures did not occur when the electrical or acoustical stimulation output was physically connected to the oscilloscope. The distortions also occurred when both stimulation outputs were physically connected, but one of the two inputs was selected to show on the oscilloscope's display. Based on this information, the conclusion can be drawn that the shared ground connections of the oscilloscope and the current amplifier are the source of the distorted figures.

5.3.2. Stimulation

As was discussed during the demonstration at the BRAI3N clinic, the maximum stimulation amplitude of 3 mA is too high. The stimulation should be under the threshold of the patient feeling the stimulation. The maximum amplitude is changed in the software to a maximum of 500 µA. The power could have been lower in the first place, reducing not only the intensity but also the amount of energy consumed by the device. For future device designs, the stimulation intensity, the 38V boost converter, and the batteries could be adjusted to the proper range of stimulation intensity.

The problem introduced in Section 5.3.1 was not investigated when stimulating a phantom or patient. For safety, it would be essential to check the behavior of the burst signal.

In case the time delay of stimulation is set to -500 ms, the actual time delay in the measurements is -262 ms, as shown in Figure 4.5b. This could be due to a delay in the starting moment of the Adafruit Music Maker, although it should also appear if the time delay is 0 ms or 500 ms. Not every time delay setting was checked for its temporal precision. If this had been done, the software could have compensated by adjusting time delay values according to each setting. From Figures 4.5b, 4.5a and 4.5c, it is clear that the time delay is not monotonic. It has not been tested if all time delay values in the menu will translate to that specific time delay in the actual time delay on the electrical and acoustical stimulation outputs.

5.3.3. Energy

The battery indication function needs to be optimized. The steps between the percentages can be changed as they are not thoroughly tested. There was never a test where one battery charge was fully discharged during a stimulation session. The minimum battery life and the discharge curve are, therefore, unknown. The voltage level according to the 0% battery level can be lowered according to a lower maximum stimulation value of 500µA instead of 3000µA.

Energy efficiency was not a requirement and, therefore, not a priority during the device design. A simple step towards reducing power consumption is to turn the LCD screen light off while stimulating and after changing a parameter. Using a rechargeable battery can improve energy efficiency because the battery's leftover charge is never thrown away. It is also better for the environment to use a rechargeable battery. In that case, a safety check should be added to ensure that stimulation is not possible while the device is connected to the power socket.

5.3.4. User-friendliness

Unfortunately, the requirement to provide two menus for the patient and clinician was not implemented. The requirement was overlooked after a long time was spent designing and building more essential functionalities. Overall, the device is meant for clinical trials, where the clinician could stay in the same room as the patient.

Another user interface requirement not implemented in the device is the ability to adjust the spike frequency of the burst pulses through the datalog.txt file. The requirement was long overlooked and turned out to be such a fundamental change to the software that it was decided to leave the option out. A start was made to investigate if this was possible with the current implementation, but it seemed that the spike frequency, dependent on the timer, could not run significantly faster than 1 ms per tick. Either the timer frequency should be much faster, at a predetermined speed, or it should be adjustable to make the spike frequency adjustable. If the frequency is predetermined, the burst function on the second core should have been adjusted. The function now depends heavily on the `burstIndex`, where it uses `burstIndex` as a reference to select the correct stimulation amplitudes from the `PulseAmplitudeArray`. If the frequency of the timer is adjustable, the `uint_8 maxBurstLength`, which is inversely proportional to the `float burstFrequency`, should be adjusted according to the frequency of the timer. The adjustability of the spike frequency also influences the amount of charge provided to the tissue with each burst. To ensure the safety of the stimulation, the maximum amplitude could also be scaled with the change in spike frequency.

Although the user interface was designed to meet the requirements of an end user, no specified method is used for creating it, such as a master's student of industrial design would use. For the next version of the device, it would be nice if the user interface was designed by a master's student or alumnus of Industrial Design to make the device user-friendly and attractive.

The same holds for the user test. Based on the intended use, a user test was designed, but the test was not implemented using a specific methodology. It indicates user-friendliness and especially points out aspects to improve user-friendliness and safety. The user test was not done by a clinician but by an engineer. This provides a bias in technical knowledge of devices in general. The engineer can also not put functionalities in the context of a clinical setting. This was also noted down during the user test. Clinicians should test the user interface design in the clinic to optimize its design. During the user test, it was pointed out that the "B=...%", shown in Figure 4.7, is not a very good indicator as a battery life indicator. This could be changed to a battery icon. The parameters could be adjusted in the digital menu during stimulation. The question of whether this functionality is desired was raised. It partially is, as it is helpful to adjust the volume or burst amplitude and slope while the stimulation is on. The effects of the changes can be noticed directly. On the other hand, it is probably not practical to

adjust the stimulation mode, time delay, or audio file while the stimulation is active. These parameters should stay the same during a stimulation session with the device.

The use of the MicroSD card was said to be prone to errors, which is considered correct. The suggestion to make a software user interface on the computer is a good idea to keep control over the parameters and settings. This software user interface can be expanded with a login prompt, so only authorized persons can update the device. Another option would be to transfer the settings over WIFI or Bluetooth. However, additional safety measures should be taken to connect the computer and µC inside the device to ensure patient safety and privacy.

Using WIFI in the device introduces new possibilities regarding efficiency and data analysis. The perfect settings for each patient, including the THI and TFI, could be added to a database. This data could then be analyzed and optimized to find the perfect settings. These outcomes can even be added to the Recommender System of the non-profit Tinnitushouse.

5.3.5. Scalability

Scalability can be interpreted in three ways: in size, functionality, and amounts.

To start with the amount, the design is not built with mass production in mind. One box construction takes four days of work with three persons. The device is custom-built to fit the requirements, and it was essential to be able to adjust a large number of parameters. After the clinical trials, if the device provides a cure for tinnitus with specific settings, the device can be scaled down in size and functionality. At least the microcontroller, user interface circuits, audio stimulation, and power circuits could be integrated into one PCB with a rechargeable battery. Also, the LCD could be changed to a smaller version, such as an OLED touchscreen.

If the device works very well for research, the opposite of integration into one PCB can be done. Making the hardware and software modular would be helpful if the functionality is scaled up. For example, Prof. Dr. de Ridder initially asked for transcranial stimulation with waveforms such as alternating current or random noise. It would be beneficial to add these functionalities through another PCB. The lesson was learned that every stimulation modality could best use its own processing core and communication channels to ensure the stimulation is reliable. Therefore, the user interface, including the menu and synchronization functions, could be one PCB with its own power management circuits and integrated µC. The same holds for the acoustical stimulation circuit, TENS and tACS, tRNS, and TMS. This can be expanded with visual, haptic, and ultrasound stimulation. One µC is responsible for the synchronization of the various stimulation modalities. The other µCs receive the start/stop commands and are accountable for running their specific stimulation modality. PCBs can be connected to their own particular rechargeable battery.

5.3.6. Translatability

The translatability in the definition of translatable research is that this research to design and create a bi-modal stimulator is practical and applicable in research. The fact that the device will be used in clinical trials makes it translatable research.

On the other hand, the device's functionality and the base of the hypothesis by which tinnitus could be cured are also translatable. One disorder closest to phantom sound is phantom pain. If the theory on tinnitus works, it could be translated into a cure for phantom pain. The brain must re-learn that the pain receptors in phantom pain are no longer there. This could be done by pairing local nerve stimulation in the sensory cortex with burst stimulation on the vagus nerve. When the working mechanism is investigated thoroughly, it could be possible that the salience of thoughts in depression or brain activity in epilepsy could also be dropped.

5.3.7. Portability

The device is now portable because it is small and has no connection to a power socket. In later versions, it could be a device as big as a smartphone or even an extension to a smartphone.

5.3.8. Safety

Battery life is only checked when the menu is changed. This could be replaced with a periodic check of the battery level. It is not checked while a patient is stimulated for 30 minutes, which is not ideal. If the stimulation session is halfway through and the battery is empty, the session is not useful for the

research.

The idea of the datalog.txt file has been helpful for flexibility but is not found to be user-friendly and could also be unsafe. It could be hazardous if the parameters are edited wrongly. This part is considered the most dangerous aspect of the device, as there are only two checks in the software on the user inputs. It is crucial to update either the software with more safety checks or build a UI on the computer to check the user inputs as discussed in Section 5.3.4.

5.3.9. Efficacy

The efficacy of the device is to be determined in the clinical studies. During the demonstration, see Section 4.4, Prof. Dr. de Ridder tested the efficacy of the electrical stimulation using the heart rate variability to indicate the activity of the parasympathetic nervous system. The determination of the heart rate variability takes time due to the ECG recording. He mentioned that the clinic requires a faster way to detect if the parasympathetic nervous system is activated. If that were the case in the future, it could be possible to create a closed-loop system where the stimulation parameters are adjusted according to the activation of the parasympathetic nervous system.

This thesis has purposely not discussed the process of creating tinnitus-matched audio. It was excluded from the functional requirements of Prof. Dr. de Ridder. The device's efficacy could strongly depend on the measure of the matching of the tinnitus-matched audio file to the tinnitus sound the patient perceives. People without musical backgrounds find it hard to compare two tones and tell which one is higher in pitch. A solution should be found to create these perfectly matched audio files. Possibly, this could be done using some machine learning approach, where a system learns to approach the perfect tone by mimicking the tinnitus sound of the patient.

Bibliography

(1) Krug, E. et al., *Hearing loss due to recreational exposure to loud sounds A review*; World Health Organisation: 2015, ISBN: 978 92 4 150851 3.

(2) Jarach, C. M.; Lugo, A.; Scala, M.; Van Den Brandt, P. A.; Cederroth, C. R.; Odone, A.; Garavello, W.; Schlee, W.; Langguth, B.; Gallus, S. Global Prevalence and Incidence of Tinnitus: A Systematic Review and Meta-analysis. *JAMA Neurology* **2022**, *79*, 888–900, ISSN: 21686157, DOI: 10.1001/jamaneurol.2022.2189.

(3) Hébert, S., Psychological Comorbidities of Tinnitus In *Current Topics in Behavioral Neurosciences*; Springer Science and Business Media Deutschland GmbH: 2021, pp 349–359, DOI: 10.1007/7854{_}2021{_}218.

(4) Roberts, L. E.; Eggermont, J. J.; Caspary, D. M.; Shore, S. E.; Melcher, J. R.; Kaltenbach, J. A., Ringing ears: The neuroscience of tinnitus In *Journal of Neuroscience*; 45, 2010, pp 14972–14979, DOI: 10.1523/JNEUROSCI.4028-10.2010.

(5) De Ridder, D.; Vanneste, S., The Bayesian brain in imbalance: Medial, lateral and descending pathways in tinnitus and pain: A perspective In *Tinnitus - An Interdisciplinary Approach Towards Individualized Treatment: Towards understanding the complexity of tinnitus*; Progress in Brain Research, Vol. 262; Elsevier: 2021, pp 309–334, DOI: https://doi.org/10.1016/bs.pbr.2020.07.012, https://www.sciencedirect.com/science/article/pii/S0079612320301102.

(6) De Ridder, D.; Vanneste, S.; Song, J. J.; Adhia, D., Tinnitus and the Triple Network Model: A Perspective In *Clinical and Experimental Otorhinolaryngology*; 3; Korean Society of Otolaryngology: 2022, pp 205–212, DOI: 10.21053/ceo.2022.00815.

(7) De Ridder, D.; Kilgard, M.; Engineer, N.; Vanneste, S. Placebo-controlled vagus nerve stimulation paired with tones in a patient with refractory tinnitus: a case report. *Otology & Neurotology* **2015**, *36*, 575–580, DOI: 10.1097/MAO.0000000000000704.

(8) Jastreboff, P. J. Phantom auditory perception (tinnitus): mechanisms of generation and perception. *Neuroscience Research* **1990**, *8*, 221–254, ISSN: 0168-0102, DOI: 10.1016/0168-0102(90)90031-9.

(9) McCormack, A.; Edmondson-Jones, M.; Somerset, S.; Hall, D., A systematic review of the reporting of tinnitus prevalence and severity In *Hearing Research*; Elsevier B.V.: 2016, pp 70–79, DOI: 10.1016/j.heares.2016.05.009.

(10) Guillard, R.; Fraysse, M.-J.; Simeon, R.; Cervoni, T.; Schmutz, J.; Piedfort, B.; Ferat, V.; Congedo, M.; Londero, A., A portable neurofeedback device for treating chronic subjective tinnitus: Feasibility and results of a pilot study In *Tinnitus - An Interdisciplinary Approach Towards Individualized Treatment: From Heterogeneity to Personalized Medicine*; Progress in Brain Research, Vol. 260; Elsevier: 2021, pp 167–185, DOI: https://doi.org/10.1016/bs.pbr.2020.08.001, https://www.sciencedirect.com/science/article/pii/S0079612320301278.

(11) Peter, N.; Kleinjung, T., Neuromodulation for tinnitus treatment: an overview of invasive and non-invasive techniques In *Journal of Zhejiang University: Science B*; 2; Zhejiang University Press: 2019, pp 116–130, DOI: 10.1631/jzus.B1700117.

(12) Krog, N. H.; Engdahl, B.; Tambs, K. The association between tinnitus and mental health in a general population sample: Results from the HUNT Study. *Journal of Psychosomatic Research* **2010**, *69*, 289–298, ISSN: 00223999, DOI: 10.1016/j.jpsychores.2010.03.008.

(13) De Ridder, D.; Schlee, W.; Vanneste, S.; Londero, A.; Weisz, N.; Kleinjung, T.; Shekhawat, G. S.; Elgoyhen, A. B.; Song, J.-J.; Andersson, G., et al. Tinnitus and tinnitus disorder: Theoretical and operational definitions (an international multidisciplinary proposal). *Progress in brain research* **2021**, *260*, 1–25, DOI: 10.1016/bs.pbr.2020.12.002.

(14) *"OpenStax AnatPhys fig.14.5 - The Structures of the Ear - English labels" by OpenStax, license: CC BY. Source: book 'Anatomy and Physiology',* https://anatomytool.org/content/openstax-anatphys-fig145-structures-ear-english-labels (accessed 02/28/2024).

(15) *"OpenStax AnatPhys fig.14.7 - Cochlea - English labels" by OpenStax, license: CC BY. Source: book 'Anatomy and Physiology',* https://anatomytool.org/content/openstax-anatphys-fig147-cochlea-english-labels (accessed 02/28/2024).

(16) *Textbook of Neuromodulation*; Knotkova, H., Rasche, D., Eds.; SpringerLink: 2019, pp 3–6, ISBN: 978-1-4939-1407-4, DOI: 10.1007/978-1-4939-1408-1.

(17) White, H. J.; Helwany, M.; Biknevicius, A. R.; Peterson, D. C., *Anatomy, Head and Neck, Ear Organ of Corti*; StatPearls Publishing: 2023, https://www.ncbi.nlm.nih.gov/books/NBK538335/ (accessed 08/14/2023).

(18) Peelle, J. E., *Schematic figure of the auditory neural pathway. This file is licensed under the Creative Commons Attribution-Share Alike 4.0 International license.* 2016, https://commons.wikimedia.org/wiki/File:Auditory_Pathway.png (accessed 02/28/2024).

(19) Silbernagl, S; Despopoulos, A, *Atlas van de fysiologie*; ThiemeMeulenhoff: 2005, ISBN: 90-5574-303-2.

(20) Boron, W. F.; Boulpaep, E. L., *Medical physiology*, 2 updated; Elsevier Saunders: 2012, ISBN: 978-1-4377-1753-2.

(21) *"Blausen 0657 - Multipolar neuron - English labels " by Bruce Blaus, license: CC BY. Source: "Wikimedia Commons: category: Images from Blausen Medical Communications"*, https://anatomytool.org/content/blausen-0657-multipolar-neuron-english-labels (accessed 03/01/2024).

(22) Lynn, C. W.; Bassett, D. S. The physics of brain network structure, function, and control. **2018**, DOI: 10.1038/s42254-019-0040-8, http://arxiv.org/abs/1809.06441 (accessed 08/16/2023).

(23) Avena-Koenigsberger, A.; Misic, B.; Sporns, O., Communication dynamics in complex brain networks In *Nature Reviews Neuroscience*; 1; Nature Publishing Group: 2018, pp 17–33, DOI: 10.1038/nrn.2017.149.

(24) Kanai, R.; Rees, G., The structural basis of inter-individual differences in human behaviour and cognition In *Nature Reviews Neuroscience*; 4, 2011, pp 231–242, DOI: 10.1038/nrn3000.

(25) Erdős, P.; Rényi, A., et al. On the evolution of random graphs. *Publ. math. inst. hung. acad. sci* **1960**, *5*, 17–60, https://static.renyi.hu/~p_erdos/1960-10.pdf (accessed 08/22/2023).

(26) Latora, V.; Marchiori, M. Efficient Behavior of Small-World Networks. *Physical Review Letters* **2001**, *87*, 198701, ISSN: 0031-9007, DOI: 10.1103/PhysRevLett.87.198701.

(27) Bullmore, E.; Sporns, O., The economy of brain network organization In *Nature Reviews Neuroscience*; 5, 2012, pp 336–349, DOI: 10.1038/nrn3214.

(28) N Amaral, L. A.; Díaz-Guilera, A.; Moreira, A. A.; Goldberger, A. L.; Lipsitz, L. A., Emergence of complex dynamics in a simple model of signaling networks APPLIED MATHEMATICS In *PNAS November*, 2004, pp 15551–15555, www.pnas.orgcgidoi10.1073pnas.0404843101.

(29) Medaglia, J. D.; Huang, W.; Karuza, E. A.; Kelkar, A.; Thompson-Schill, S. L.; Ribeiro, A.; Bassett, D. S. Functional alignment with anatomical networks is associated with cognitive flexibility. *Nature Human Behaviour* **2018**, *2*, 156–164, ISSN: 23973374, DOI: 10.1038/s41562-017-0260-9.

(30) Plonsey, R.; Fleimg, D. G., *Bioelectric Phenomena*; McGraw-Hill book company: 1969, ISBN: 9780070503427.

(31) Malmivuo, J.; Plonsey, R., *Bioelectromagnetism: Principles and Applications of Bioelectric and Biomagnetic Fields*; Oxford University Press: 1995, ISBN: 9780195058239, DOI: 10.1093/acprof:oso/9780195058239.001.0001.

(32) Faro, S. H.; Mohamed, F. B., *BOLD fMRI: A guide to functional imaging for neuroscientists*; Springer Science & Business Media: 2010, ISBN: 978-1-4419-1328-9, DOI: 10.1007/978-4419-1329-6.

(33) McCulloch, W. S.; Pitts, W. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics* **1943**, *5*, 115–133, DOI: 10.1007/BF02478259.

(34) Hodgkin, A. L.; Huxley, A. F. A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of Physiology* **1952**, *117*, 500–544, ISSN: 0022-3751, DOI: 10.1113/jphysiol.1952.sp004764.

(35) FitzHugh, R. Impulses and Physiological States in Theoretical Models of Nerve Membrane. *Biophysical Journal* **1961**, *1*, 445–466, ISSN: 00063495, DOI: 10.1016/S0006-3495(61)86902-6.

(36) Nagumo, J.; Arimoto, S.; Yoshizawa, S. An active pulse transmission line simulating nerve axon. *Proceedings of the IRE* **1962**, *50*, 2061–2070, DOI: 10.1109/JRPROC.1962.288235.

(37) Wilson, H. R.; Cowan, J. D. Excitatory and Inhibitory Interactions in Localized Populations of Model Neurons. *Biophysical Journal* **1972**, *12*, 1–24, ISSN: 00063495, DOI: 10.1016/S0006-3495(72)86068-5.

(38) David, O.; Friston, K. J. A neural mass model for MEG/EEG: Coupling and neuronal dynamics. *NeuroImage* **2003**, *20*, 1743–1755, ISSN: 10538119, DOI: 10.1016/j.neuroimage.2003.07.015.

(39) Kuramoto, Y, *Chemical oscillations, waves, and turbulence*; Springer–Verlag: 1984, ISBN: 978-3-642-69691-6.

(40) Acebrón, J. A.; Bonilla, L. L.; Vicente, C. J. P.; Ritort, F.; Spigler, R. The Kuramoto model: A simple paradigm for synchronization phenomena. *Reviews of modern physics* **2005**, *77*, 137, DOI: 10.1103/RevModPhys.77.137.

(41) Cabral, J.; Hugues, E.; Sporns, O.; Deco, G. Role of local network oscillations in resting-state functional connectivity. *NeuroImage* **2011**, *57*, 130–139, ISSN: 10538119, DOI: 10.1016/j.neuroimage.2011.04.010.

(42) Puderbaugh, M.; Emmady, P. D., Neuroplasticity In *StatPearls*, 2023, https://www.ncbi.nlm.nih.gov/books/NBK557811/ (accessed 08/28/2023).

(43) The Plastic Brain In *Advances in Experimental Medicine and Biology*, von Bernhardi, R., Eugenín, J., Muller, K. J., Eds.; Springer International Publishing: Cham, 2017, ISBN: 978-3-319-62815-8, DOI: 10.1007/978-3-319-62817-2, http://link.springer.com/10.1007/978-3-319-62817-2.

(44) Brzosko, Z.; Mierau, S. B.; Paulsen, O., Neuromodulation of Spike-Timing-Dependent Plasticity: Past, Present, and Future In *Neuron*; 4; Cell Press: 2019, pp 563–581, DOI: 10.1016/j.neuron.2019.05.041.

(45) Butz, M.; Worgotter, F.; van Ooyen, A., Activity-dependent structural plasticity In *Brain Research Reviews*; 2, 2009, pp 287–305, DOI: 10.1016/j.brainresrev.2008.12.023.

(46) Fornito, A.; Bullmore, E. T. Connectomics: A new paradigm for understanding brain disease. *European Neuropsychopharmacology* **2015**, *25*, 733–748, ISSN: 18737862, DOI: 10.1016/j.euroneuro.2014.02.011.

(47) Golshani, P.; Gonçalves, J. T.; Khoshkhoo, S.; Mostany, R.; Smirnakis, S.; Portera-Cailliau, C. Internally mediated developmental desynchronization of neocortical network activity. *Journal of Neuroscience* **2009**, *29*, 10890–10899, ISSN: 02706474, DOI: 10.1523/JNEUROSCI.2012-09.2009.

(48) Lewis, C. M.; Baldassarre, A.; Committeri, G.; Romani, G. L.; Corbetta, M. Learning sculpts the spontaneous activity of the resting human brain. *Proceedings of the National Academy of Sciences* **2009**, *106*, 17558–17563, DOI: 10.1073/pnas.0902455106.

(49) Reed, A.; Riley, J.; Carraway, R.; Carrasco, A.; Perez, C.; Jakkamsetti, V.; Kilgard, M. P. Cortical Map Plasticity Improves Learning but Is Not Necessary for Improved Performance. *Neuron* **2011**, *70*, 121–131, ISSN: 08966273, DOI: 10.1016/j.neuron.2011.02.038.

(50) Kilgard, M. P., Harnessing plasticity to understand learning and treat disease In *Trends in Neurosciences*; 12, 2012, pp 715–722, DOI: 10.1016/j.tins.2012.09.002.

(51) Pavlov, P. I. Conditioned reflexes: An investigation of the physiological activity of the cerebral cortex. *Annals of neurosciences* **2010**, *17*, 136–41, ISSN: 0972-7531, DOI: 10.5214/ans.0972-7531.1017309.

(52) Manjaly, Z. M.; Iglesias, S. A Computational Theory of Mindfulness Based Cognitive Therapy from the Bayesian Brain Perspective. *Frontiers in Psychiatry* **2020**, *11*, DOI: 10.3389/fpsyt.2020.00404, ISSN: 16640640.

(53) Zilles, K. Brodmann: a pioneer of human brain mapping—his impact on concepts of cortical organization. *Brain* **2018**, *141*, 3262–3278, ISSN: 0006-8950, DOI: 10.1093/brain/awy273.

(54) Mohan, A.; De Ridder, D.; Vanneste, S. Graph theoretical analysis of brain connectivity in phantom sound perception. *Scientific Reports* **2016**, *6*, DOI: 10.1038/srep19683, ISSN: 20452322.

(55) Menon, V. Large-scale brain networks and psychopathology: a unifying triple network model. *Trends in Cognitive Sciences* **2011**, *15*, 483–506, ISSN: 13646613, DOI: 10.1016/j.tics.2011.08.003.

(56) Seeley, W. W.; Menon, V.; Schatzberg, A. F.; Keller, J.; Glover, G. H.; Kenna, H.; Reiss, A. L.; Greicius, M. D. Dissociable intrinsic connectivity networks for salience processing and executive control. *Journal of Neuroscience* **2007**, *27*, 2349–2356, ISSN: 02706474, DOI: 10.1523/JNEUROSCI.5587-06.2007.

(57) Koechlin, E.; Summerfield, C. An information theoretical approach to prefrontal executive function. *Trends in Cognitive Sciences* **2007**, *11*, 229–235, ISSN: 13646613, DOI: 10.1016/j.tics.2007.04.005.

(58) Qin, P.; Northoff, G., How is our self related to midline regions and the default-mode network? In *NeuroImage*; 3, 2011, pp 1221–1233, DOI: 10.1016/j.neuroimage.2011.05.028.

(59) Fox, M. D.; Snyder, A. Z.; Vincent, J. L.; Corbetta, M.; Van Essen, D. C.; Raichle, M. E. The human brain is intrinsically organized into dynamic, anticorrelated functional networks. *Proceedings of the National Academy of Sciences of the United States of America* **2005**, *102*, 9673–9678, ISSN: 0027-8424, DOI: 10.1073/pnas.0504136102, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1157105/ (accessed 09/13/2023).

(60) De Ridder, D.; Fransen, H.; Francois, O.; Sunaert, S.; Kovacs, S.; Van De Heyning, P., Amygdalohippocampal involvement in tinnitus and auditory memory In *Acta Oto-Laryngologica*; SUPPL. 556, 2006, pp 50–53, DOI: 10.1080/03655230600895580.

(61) Diederen, K. M.; Neggers, S. F.; Daalman, K.; Blom, J. D.; Goekoop, R.; Kahn, R. S.; Sommer, I. E. Deactivation of the Parahippocampal Gyrus Preceding Auditory Hallucinations in Schizophrenia. *American Journal of Psychiatry* **2010**, *167*, 427–435, ISSN: 0002-953X, DOI: 10.1176/appi.ajp.2009.09040456.

(62) Gaab, N.; Gaser, C.; Zaehle, T.; Jancke, L.; Schlaug, G. Functional anatomy of pitch memory - An fMRI study with sparse temporal sampling. *NeuroImage* **2003**, *19*, 1417–1426, ISSN: 10538119, DOI: 10.1016/S1053-8119(03)00224-6.

(63) Siegel, A.; Sapru, H.; Siegel, H., *Essential Neuroscience*; Wolters Kluwer Health: 2015, ISBN: 9781451189681.

(64) Colzato, L. S.; Vonck, K., Transcutaneous Vagus and Trigeminal Nerve Stimulation In *Theory-Driven Approaches to Cognitive Enhancement*; Springer International Publishing: Cham, 2017, pp 115–126, DOI: 10.1007/978-3-319-57505-6{_}9.

(65) De Ridder, D.; Vanneste, S.; Weisz, N.; Londero, A.; Schlee, W.; Elgoyhen, A. B.; Langguth, B., An integrative model of auditory phantom perception: Tinnitus as a unified percept of interacting separable subnetworks In *Neuroscience and Biobehavioral Reviews*; Elsevier Ltd: 2014, pp 16–32, DOI: 10.1016/j.neubiorev.2013.03.021.

(66) Jastreboff, P. J. Phantom auditory perception (tinnitus): mechanisms of generation and perception. *Neuroscience Research* **1990**, *8*, 221–254, ISSN: 0168-0102, DOI: 10.1016/0168-0102(90)90031-9.

(67) Jackson, P. A comparison of the effects of eighth nerve section with lidocaine on tinnitus. *The Journal of Laryngology and Otology* **1985**, *99*, 663–666, ISSN: 0022-2151, DOI: 10.1017/s0022215100097449.

(68) House, J. W.; Brackmann, D. E. Tinnitus: surgical treatment. *Ciba Foundation Symposium* **1981**, *85*, 204–216, ISSN: 0300-5208, DOI: 10.1002/9780470720677.ch12.

(69) Schneider, P.; Andermann, M.; Wengenroth, M.; Goebel, R.; Flor, H.; Rupp, A.; Diesch, E. Reduced volume of Heschl's gyrus in tinnitus. *NeuroImage* **2009**, *45*, 927–939, ISSN: 1053-8119, DOI: 10.1016/j.neuroimage.2008.12.045, https://www.sciencedirect.com/science/article/pii/S1053811908013049 (accessed 10/13/2023).

(70) Mühlnickel, W.; Elbert, T.; Taub, E.; Flor, H. Reorganization of auditory cortex in tinnitus. *Proceedings of the National Academy of Sciences of the United States of America* **1998**, *95*, 10340–10343, ISSN: 0027-8424, DOI: 10.1073/pnas.95.17.10340.

(71) Langguth, B.; Kreuzer, P. M.; Kleinjung, T.; De Ridder, D. Tinnitus: causes and clinical management. *The Lancet Neurology* **2013**, *12*, 920–930, ISSN: 1474-4422, DOI: 10.1016/S1474-4422(13)70160-1, https://www.sciencedirect.com/science/article/pii/S1474442213701601 (accessed 04/06/2023).

(72) De Ridder, D.; Vanneste, S.; Freeman, W. The Bayesian brain: Phantom percepts resolve sensory uncertainty. *Neuroscience & Biobehavioral Reviews* **2014**, *44*, 4–15, ISSN: 0149-7634, DOI: 10.1016/j.neubiorev.2012.04.001, https://www.sciencedirect.com/science/article/pii/S0149763412000607 (accessed 04/06/2023).

(73) De Ridder, D.; Langguth, B.; Vanneste, S., Chapter 20 - Vagus nerve stimulation for tinnitus: A review and perspective In *Tinnitus - An Interdisciplinary Approach Towards Individualized Treatment: Towards understanding the complexity of tinnitus*, Langguth, B., Kleinjung, T., De Ridder, D., Schlee, W., Vanneste, S., Eds.; Progress in Brain Research, Vol. 262; Elsevier: 2021, pp 451–467, DOI: https://doi.org/10.1016/bs.pbr.2020.08.011, https://www.sciencedirect.com/science/article/pii/S0079612320301485.

(74) Trevino, M.; Lobarinas, E. Current topics in hearing research: Deafferentation and threshold independent hearing loss. *Hearing Research* **2022**, *419*, 108408, ISSN: 03785955, DOI: 10.1016/j.heares.2021.108408.

(75) Roberts, L. E.; Moffat, G.; Baumann, M.; Ward, L. M.; Bosnyak, D. J. Residual Inhibition Functions Overlap Tinnitus Spectra and the Region of Auditory Threshold Shift. *JARO: Journal of the Association for Research in Otolaryngology* **2008**, *9*, 417–435, ISSN: 1525-3961, DOI: 10.1007/s10162-008-0136-9, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2580805/ (accessed 10/13/2023).

(76) Weisz, N.; Hartmann, T.; Dohrmann, K.; Schlee, W.; Norena, A. High-frequency tinnitus without hearing loss does not mean absence of deafferentation. *Hearing Research* **2006**, *222*, 108–114, ISSN: 0378-5955, DOI: 10.1016/j.heares.2006.09.003, https://www.sciencedirect.com/science/article/pii/S0378595506002528 (accessed 10/13/2023).

(77) Job, A.; Raynal, M.; Kossowski, M. Susceptibility to tinnitus revealed at 2 kHz range by bilateral lower DPOAEs in normal hearing subjects with noise exposure. *Audiology & Neuro-Otology* **2007**, *12*, 137–144, ISSN: 1421-9700, DOI: 10.1159/000099025.

(78) Kara, E.; Aydın, K.; Akbulut, A. A.; Karakol, S. N.; Durmaz, S.; Yener, H. M.; Gözen, E. D.; Kara, H. Assessment of Hidden Hearing Loss in Normal Hearing Individuals with and Without Tinnitus. *The Journal of International Advanced Otology* **2020**, *16*, 87–92, ISSN: 1308-7649, DOI: 10.5152/iao.2020.7062, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7224424/ (accessed 04/06/2023).

(79) Shore, S. E.; Roberts, L. E.; Langguth, B. Maladaptive plasticity in tinnitus-triggers, mechanisms and treatment. *Nature reviews. Neurology* **2016**, *12*, 150–160, ISSN: 1759-4758, DOI: 10.1038/nrneurol.2016.12, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4895692/ (accessed 04/06/2023).

(80) Dehmel, S.; Eisinger, D.; Shore, S. Gap prepulse inhibition and auditory brainstem-evoked potentials as objective measures for tinnitus in guinea pigs. *Frontiers in Systems Neuroscience* **2012**, *6*, ISSN: 1662-5137, https://www.frontiersin.org/articles/10.3389/fnsys.2012.00042 (accessed 04/06/2023).

(81) Yang, S.; Weiner, B. D.; Zhang, L. S.; Cho, S.-J.; Bao, S. Homeostatic plasticity drives tinnitus perception in an animal model. *Proceedings of the National Academy of Sciences of the United States of America* **2011**, *108*, 14974–14979, ISSN: 0027-8424, DOI: 10.1073/pnas.1107998108, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3169130/ (accessed 10/13/2023).

(82) Schaette, R.; Kempter, R. Development of tinnitus-related neuronal hyperactivity through homeostatic plasticity after hearing loss: a computational model. *European Journal of Neuroscience* **2006**, *23*, 3124–3138, ISSN: 1460-9568, DOI: 10.1111/j.1460-9568.2006.04774.x, (accessed 10/13/2023).

(83) Eggermont, J. J.; Roberts, L. E. The neuroscience of tinnitus. *Trends in Neurosciences* **2004**, *27*, 676–682, ISSN: 0166-2236, DOI: 10.1016/j.tins.2004.08.010, https://www.sciencedirect.com/science/article/pii/S0166223604002760 (accessed 10/13/2023).

(84) Møller, A. R., Neural plasticity in tinnitus In *Progress in Brain Research*, Møller, A. R., Ed.; Reprogramming of the Brain, Vol. 157; Elsevier: 2006, pp 365–372, DOI: 10.1016/S0079-6123(06)57022-0, https://www.sciencedirect.com/science/article/pii/S0079612306570220 (accessed 10/13/2023).

(85) Engineer, N. D.; Riley, J. R.; Seale, J. D.; Vrana, W. A.; Shetake, J. A.; Sudanagunta, S. P.; Borland, M. S.; Kilgard, M. P. Reversing pathological neural activity using targeted plasticity. *Nature* **2011**, *470*, 101–104, ISSN: 1476-4687, DOI: 10.1038/nature09656, https://www.nature.com/articles/nature09656 (accessed 04/06/2023).

(86) Langers, D. R. M.; de Kleine, E.; van Dijk, P. Tinnitus does not require macroscopic tonotopic map reorganization. *Frontiers in Systems Neuroscience* **2012**, *6*, 2, ISSN: 1662-5137, DOI: 10.3389/fnsys.2012.00002, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3269775/ (accessed 10/12/2023).

(87) Levine, R. A. Somatic (craniocervical) tinnitus and the dorsal cochlear nucleus hypothesis. *American Journal of Otolaryngology* **1999**, *20*, 351–362, ISSN: 0196-0709, DOI: 10.1016/S0196-0709(99)90074-1, https://www.sciencedirect.com/science/article/pii/S0196070999900741 (accessed 10/13/2023).

(88) Haenggeli, C.-A.; Pongstaporn, T.; Doucet, J. R.; Ryugo, D. K. Projections from the spinal trigeminal nucleus to the cochlear nucleus in the rat. *The Journal of Comparative Neurology* **2005**, *484*, 191–205, ISSN: 0021-9967, DOI: 10.1002/cne.20466.

(89) Koehler, S. D.; Shore, S. E. Stimulus-Timing Dependent Multisensory Plasticity in the Guinea Pig Dorsal Cochlear Nucleus. *PLoS ONE* **2013**, *8*, e59828, ISSN: 1932-6203, DOI: 10.1371/journal.pone.0059828, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3603886/ (accessed 04/06/2023).

(90) Koehler, S. D.; Shore, S. E. Stimulus Timing-Dependent Plasticity in Dorsal Cochlear Nucleus Is Altered in Tinnitus. *The Journal of Neuroscience* **2013**, *33*, 19647–19656, ISSN: 0270-6474, DOI: 10.1523/JNEUROSCI.2788-13.2013, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3858633/ (accessed 04/06/2023).

(91) Wu, C.; Martel, D. T.; Shore, S. E. Transcutaneous induction of stimulus-timing-dependent plasticity in dorsal cochlear nucleus. *Frontiers in Systems Neuroscience* **2015**, *9*, 116, ISSN: 1662-5137, DOI: 10.3389/fnsys.2015.00116, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4536405/ (accessed 10/12/2023).

(92) Basura, G. J.; Koehler, S. D.; Shore, S. E. Multi-sensory integration in brainstem and auditory cortex. *Brain research* **2012**, *1485*, 95–107, ISSN: 0006-8993, DOI: 10.1016/j.brainres.2012.08.037, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3489970/ (accessed 10/13/2023).

(93) Zeng, C.; Yang, Z.; Shreve, L.; Bledsoe, S.; Shore, S. Somatosensory Projections to Cochlear Nucleus Are Upregulated after Unilateral Deafness. *The Journal of Neuroscience* **2012**, *32*, 15791–15801, ISSN: 0270-6474, DOI: 10.1523/JNEUROSCI.2598-12.2012, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3501653/ (accessed 10/13/2023).

(94) De Ridder, D.; Elgoyhen, A. B.; Romo, R.; Langguth, B. Phantom percepts: Tinnitus and pain as persisting aversive memory networks. *Proceedings of the National Academy of Sciences* **2011**, *108*, 8075–8080, ISSN: 0027-8424, DOI: 10.1073/pnas.1018466108.

(95) Schecklmann, M.; Lehner, A.; Poeppl, T. B.; Kreuzer, P. M.; Rupprecht, R.; Rackl, J.; Burger, J.; Frank, E.; Hajak, G.; Langguth, B.; Landgrebe, M. Auditory cortex is implicated in tinnitus distress: a voxel-based morphometry study. *Brain Structure and Function* **2013**, *218*, 1061–1070, ISSN: 1863-2661, DOI: 10.1007/s00429-013-0520-z, https://doi.org/10.1007/s00429-013-0520-z (accessed 10/13/2023).

(96) De Ridder, D.; Congedo, M.; Vanneste, S. The neural correlates of subjectively perceived and passively matched loudness perception in auditory phantom perception. *Brain and Behavior* **2015**, *5*, DOI: 10.1002/brb3.331, ISSN: 2162-3279.

(97) Leaver, A. M.; Renier, L.; Chevillet, M. A.; Morgan, S.; Kim, H. J.; Rauschecker, J. P. Dysregulation of Limbic and Auditory Networks in Tinnitus. *Neuron* **2011**, *69*, 33–43, ISSN: 0896-6273, DOI: 10.1016/j.neuron.2010.12.002, https://www.sciencedirect.com/science/article/pii/S0896627310009876 (accessed 09/03/2023).

(98) Rauschecker, J. P.; Leaver, A. M.; Mühlau, M. Tuning Out the Noise: Limbic-Auditory Interactions in Tinnitus. *Neuron* **2010**, *66*, 819–826, ISSN: 0896-6273, DOI: 10.1016/j.neuron.2010.04.032, https://www.sciencedirect.com/science/article/pii/S0896627310003259 (accessed 09/03/2023).

(99) Piarulli, A.; Vanneste, S.; Nemirovsky, I. E.; Kandeepan, S.; Maudoux, A.; Gemignani, A.; De Ridder, D.; Soddu, A. Tinnitus and distress: an electroencephalography classification study. *Brain Communications* **2022**, *5*, DOI: 10.1093/braincomms/fcad018.

(100) Meyer, M.; Luethi, M. S.; Neff, P.; Langer, N.; Büchi, S. Disentangling tinnitus distress and tinnitus presence by means of EEG power analysis. *Neural Plasticity* **2014**, *2014*, DOI: 10.1155/2014/468546, ISSN: 16875443.

(101) Vanneste, S.; Plazier, M.; der Loo, E. v.; de Heyning, P. V.; Congedo, M.; De Ridder, D. The neural correlates of tinnitus-related distress. *NeuroImage* **2010**, *52*, 470–480, ISSN: 10538119, DOI: 10.1016/j.neuroimage.2010.04.029.

(102) González, H. F.; Yengo-Kahn, A.; Englot, D. J. Vagus Nerve Stimulation for the Treatment of Epilepsy. *Neurosurgery clinics of North America* **2019**, *30*, 219–230, ISSN: 1042-3680, DOI: 10.1016/j.nec.2018.12.005, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6432928/ (accessed 10/12/2023).

(103) Grimm, S.; Bajbouj, M. Efficacy of vagus nerve stimulation in the treatment of depression. *Expert Review of Neurotherapeutics* **2010**, *10*, 87–92, ISSN: 1473-7175, DOI: 10.1586/ern.09.138, https://doi.org/10.1586/ern.09.138 (accessed 10/12/2023).

(104) Berthoud, H.-R.; Neuhuber, W. L. Functional and chemical anatomy of the afferent vagal system. *Autonomic Neuroscience* **2000**, *85*, 1–17, ISSN: 1566-0702, DOI: 10.1016/S1566-0702(00)00215-0, https://www.sciencedirect.com/science/article/pii/S1566070200002150 (accessed 09/28/2023).

(105) Yakunina, N.; Nam, E.-C. Direct and Transcutaneous Vagus Nerve Stimulation for Treatment of Tinnitus: A Scoping Review. *Frontiers in Neuroscience* **2021**, *15*, 680590, ISSN: 1662-4548, DOI: 10.3389/fnins.2021.680590, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8193498/ (accessed 04/06/2023).

(106) Yakunina, N.; Kim, S. S.; Nam, E.-C. BOLD fMRI effects of transcutaneous vagus nerve stimulation in patients with chronic tinnitus. *PLoS ONE* **2018**, *13*, e0207281, ISSN: 1932-6203, DOI: 10.1371/journal.pone.0207281, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6261575/ (accessed 09/22/2023).

(107) Kaltenbach, J. A. The dorsal cochlear nucleus as a participant in the auditory, attentional and emotional components of tinnitus. *Hearing Research* **2006**, *216-217*, 224–234, ISSN: 0378-5955, DOI: 10.1016/j.heares.2006.01.002, https://www.sciencedirect.com/science/article/pii/S0378595506000037 (accessed 09/28/2023).

(108) Franzini, A.; Moosa, S.; Servello, D.; Small, I.; DiMeco, F.; Xu, Z.; Elias, W. J.; Franzini, A.; Prada, F., Ablative brain surgery: an overview In *International Journal of Hyperthermia*; 2; Taylor and Francis Ltd: 2019, pp 64–80, DOI: 10.1080/02656736.2019.1616833.

(109) Brozoski, T. J.; Bauer, C. A. The effect of dorsal cochlear nucleus ablation on tinnitus in rats. *Hearing Research* **2005**, *206*, 227–236, ISSN: 03785955, DOI: 10.1016/j.heares.2004.12.013.

(110) Henry, R.; Deckert, M.; Guruviah, V.; Schmidt, B. Review of Neuromodulation Techniques and Technological Limitations. *IETE Technical Review* **2016**, *33*, 368–377, ISSN: 0256-4602, DOI: 10.1080/02564602.2015.1106926, https://doi.org/10.1080/02564602.2015.1106926 (accessed 10/06/2023).

(111) Merrill, D. R.; Bikson, M.; Jefferys, J. G. Electrical stimulation of excitable tissue: design of efficacious and safe protocols. *Journal of Neuroscience Methods* **2005**, *141*, 171–198, ISSN: 01650270, DOI: 10.1016/j.jneumeth.2004.10.020, https://linkinghub.elsevier.com/retrieve/pii/S0165027004003826 (accessed 05/11/2023).

(112) Centerwatch, *FDA Approved Drugs | CenterWatch*, 2023, https://www.centerwatch.com/directories/1067-fda-approved-drugs?page=62 (accessed 10/06/2023).

(113) Weiss, N.; Miller, F.; Cazaubon, S.; Couraud, P.-O. The blood-brain barrier in brain homeostasis and neurological diseases. *Biochimica et Biophysica Acta (BBA) - Biomembranes* **2009**, *1788*, 842–857, ISSN: 0005-2736, DOI: 10.1016/j.bbamem.2008.10.022, https://www.sciencedirect.com/science/article/pii/S0005273608003489 (accessed 10/06/2023).

(114) Luan, S.; Williams, I.; Nikolic, K.; Constandinou, T. G. Neuromodulation: present and emerging methods. *Frontiers in Neuroengineering* **2014**, *7*, ISSN: 1662-6443, https://www.frontiersin.org/articles/10.3389/fneng.2014.00027 (accessed 10/06/2023).

(115) Zöger, S.; Svedlund, J.; Holgers, K.-M. The Effects of Sertraline on Severe Tinnitus Suffering-A Randomized, Double-blind, Placebo-Controlled Study. *Journal of Clinical Psychopharmacology* **2006**, *26*, 32, ISSN: 0271-0749, DOI: 10.1097/01.jcp.0000195111.86650.19, https://journals.lww.com/psychopharmacology/fulltext/2006/02000/the_effects_of_sertraline_on_severe_tinnitus.8.aspx (accessed 10/10/2023).

(116) Bayar, N.; Boke, B.; Turan, E.; Belgin, E. Efficacy of amitriptyline in the treatment of subjective tinnitus. *The Journal of Otolaryngology* **2001**, *30*, 300–3, ISSN: 03816605, https://www.proquest.com/docview/218309582/abstract/3B743C84B83A47FDPQ/1 (accessed 10/10/2023).

(117) Sullivan, M.; Katon, W.; Russo, J.; Dobie, R.; Sakai, C. A Randomized Trial of Nortriptyline for Severe Chronic Tinnitus: Effects on Depression, Disability, and Tinnitus Symptoms. *Archives of Internal Medicine* **1993**, *153*, 2251–2259, ISSN: 0003-9926, DOI: 10.1001/archinte.1993.00410190091011, https://doi.org/10.1001/archinte.1993.00410190091011 (accessed 10/10/2023).

(118) Langguth, B.; Elgoyhen, A. B.; Cederroth, C. R. Therapeutic Approaches to the Treatment of Tinnitus. *Annual Review of Pharmacology and Toxicology* **2019**, *59*, 291–313, DOI: 10.1146/annurev-pharmtox-010818-021556, https://doi.org/10.1146/annurev-pharmtox-010818-021556 (accessed 10/10/2023).

(119) Azevedo, A. A. d.; Figueiredo, R. R., Treatment of tinnitus with acamprosate In *Progress in Brain Research*, Langguth, B., Hajak, G., Kleinjung, T., Cacace, A., Møller, A. R., Eds.; Tinnitus: Pathophysiology and Treatment, Vol. 166; Elsevier: 2007, pp 273–277, DOI: 10.1016/S0079-6123(07)66025-7, https://www.sciencedirect.com/science/article/pii/S0079612307660257 (accessed 10/10/2023).

(120) Sharma, D. K.; Kaur, S.; Singh, J.; Kaur, I. Role of acamprosate in sensorineural tinnitus. *Indian Journal of Pharmacology* **2012**, *44*, 93–96, ISSN: 0253-7613, DOI: 10.4103/0253-7613.91876, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3271548/ (accessed 10/10/2023).

(121) O'Reardon, J. P.; Peshek, A. D.; Romero, R.; Cristancho, P. Neuromodulation and Transcranial Mag Netic Stimulation (TMS). *Psychiatry (Edgmont)* **2006**, *3*, 30–40, ISSN: 1550-5952, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2990546/ (accessed 10/06/2023).

(122) Darmani, G. et al. Non-invasive transcranial ultrasound stimulation for neuromodulation. *Clinical Neurophysiology* **2022**, *135*, 51–73, ISSN: 13882457, DOI: 10.1016/j.clinph.2021.12.010.

(123) Di Biase, L.; Falato, E.; Caminiti, M. L.; Pecoraro, P. M.; Narducci, F.; Di Lazzaro, V., Focused Ultrasound (FUS) for Chronic Pain Management: Approved and Potential Applications In *Neurology Research International*; Hindawi Limited: 2021, DOI: 10.1155/2021/8438498.

(124) Yizhar, O.; Fenno, L. E.; Davidson, T. J.; Mogri, M.; Deisseroth, K. Optogenetics in Neural Systems. *Neuron* **2011**, *71*, 9–34, ISSN: 0896-6273, DOI: 10.1016/j.neuron.2011.06.004, https://www.sciencedirect.com/science/article/pii/S0896627311005046 (accessed 03/04/2024).

(125) Chow, B. Y.; Han, X.; Dobry, A. S.; Qian, X.; Chuong, A. S.; Li, M.; Henninger, M. A.; Belfort, G. M.; Lin, Y.; Monahan, P. E.; Boyden, E. S. High-performance genetically targetable optical neural silencing by light-driven proton pumps. *Nature* **2010**, *463*, 98–102, ISSN: 1476-4687, DOI: 10.1038/nature08652.

(126) Nagel, G.; Szellas, T.; Huhn, W.; Kateriya, S.; Adeishvili, N.; Berthold, P.; Ollig, D.; Hegemann, P.; Bamberg, E. Channelrhodopsin-2, a directly light-gated cation-selective membrane channel. *Proceedings of the National Academy of Sciences of the United States of America* **2003**, *100*, 13940–13945, ISSN: 0027-8424, DOI: 10.1073/pnas.1936192100, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC283525/ (accessed 10/06/2023).

(127) Yang, D.; Shin, Y.-I.; Hong, K.-S. Systemic Review on Transcranial Electrical Stimulation Parameters and EEG/fNIRS Features for Brain Diseases. *Frontiers in Neuroscience* **2021**, *15*, 629323, ISSN: 1662-4548, DOI: 10.3389/fnins.2021.629323, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8032955/ (accessed 04/19/2023).

(128) McCreery, D.; Agnew, W.; Yuen, T.; Bullara, L. Charge density and charge per phase as cofactors in neural injury induced by electrical stimulation. *IEEE Transactions on Biomedical Engineering* **1990**, *37*, 996–1001, ISSN: 00189294, DOI: 10.1109/10.102812, http://ieeexplore.ieee.org/document/102812/ (accessed 05/11/2023).

(129) Lilly, J.; Hughes, J.; Alvord, E.; Galkin, T. Brief, Noninjurious Electric Waveform for Stimulation of the Brain. *Science* **1955**, *121*, 468–469, DOI: 10.1126/science.121.3144.468, https://www.science.org/doi/10.1126/science.121.3144.468 (accessed 10/03/2023).

(130) Mortimer, J. T.; Shealy, C. N.; Wheeler, C. Experimental Nondestructive Electrical Stimulation of the Brain and Spinal Cord. *Journal of Neurosurgery* **1970**, *32*, 553–559, DOI: 10.3171/jns.1970.32.5.0553, https://thejns.org/view/journals/j-neurosurg/32/5/article-p553.xml (accessed 10/04/2023).

(131) Bikson, M. et al. Safety of Transcranial Direct Current Stimulation: Evidence Based Update 2016. *Brain Stimulation* **2016**, *9*, 641–661, ISSN: 1935-861X, DOI: 10.1016/j.brs.2016.06.004, https://www.sciencedirect.com/science/article/pii/S1935861X16301401 (accessed 10/05/2023).

(132) Thompson, S. L.; O'Leary, G. H.; Austelle, C. W.; Gruber, E.; Kahn, A. T.; Manett, A. J.; Short, B.; Badran, B. W. A Review of Parameter Settings for Invasive and Non-invasive Vagus Nerve Stimulation (VNS) Applied in Neurological and Psychiatric Disorders. *Frontiers in Neuroscience* **2021**, *15*, 709436, ISSN: 1662-4548, DOI: 10.3389/fnins.2021.709436, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8313807/ (accessed 07/31/2023).

(133) Roosevelt, R. W.; Smith, D. C.; Clough, R. W.; Jensen, R. A.; Browning, R. A. Increased extracellular concentrations of norepinephrine in cortex and hippocampus following vagus nerve stimulation in the rat. *Brain Research* **2006**, *1119*, 124–132, ISSN: 0006-8993, DOI: 10.1016/j.brainres.2006.08.048, https://www.sciencedirect.com/science/article/pii/S0006899306024474 (accessed 10/08/2023).

(134) Hulsey, D. R.; Riley, J. R.; Loerwald, K. W.; Rennaker, R. L.; Kilgard, M. P.; Hays, S. A. Parametric characterization of neural activity in the locus coeruleus in response to vagus nerve stimulation. *Experimental Neurology* **2017**, *289*, 21–30, ISSN: 1090-2430, DOI: 10.1016/j.expneurol.2016.12.005.

(135) Ahmed, S.; Yearwood, T.; De Ridder, D.; Vanneste, S. Burst and high frequency stimulation: underlying mechanism of action. *Expert Review of Medical Devices* **2018**, *15*, 61–70, ISSN: 1743-4440, DOI: 10.1080/17434440.2018.1418662, https://doi.org/10.1080/17434440.2018.1418662 (accessed 04/18/2023).

(136) Slavin, K. V.; North, R. B.; Deer, T. R.; Staats, P.; Davis, K.; Diaz, R. Tonic and burst spinal cord stimulation waveforms for the treatment of chronic, intractable pain: study protocol for a randomized controlled trial. *Trials* **2016**, *17*, 569, ISSN: 1745-6215, DOI: 10.1186/s13063-016-1706-5, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5131423/ (accessed 10/04/2023).

(137) Sherman, S. M. Tonic and burst firing: dual modes of thalamocortical relay. *Trends in Neurosciences* **2001**, *24*, 122–126, ISSN: 0166-2236, DOI: 10.1016/S0166-2236(00)01714-8, https://www.sciencedirect.com/science/article/pii/S0166223600017148 (accessed 10/04/2023).

(138) De Ridder, D.; Vanneste, S.; Plazier, M.; Vancamp, T. Mimicking the brain: evaluation of St Jude Medical's Prodigy Chronic Pain System with Burst Technology. *Expert Review of Medical Devices* **2015**, *12*, 143–150, ISSN: 1743-4440, DOI: 10.1586/17434440.2015.985652, https://doi.org/10.1586/17434440.2015.985652 (accessed 10/09/2023).

(139) De Ridder, D.; Vanneste, S.; van der Loo, E.; Plazier, M.; Menovsky, T.; van de Heyning, P. Burst stimulation of the auditory cortex: a new form of neurostimulation for noise-like tinnitus suppression. *Journal of Neurosurgery* **2010**, *112*, 1289–1294, ISSN: 1933-0693, DOI: 10.3171/2009.10.JNS09298.

(140) De Ridder, D.; Vanneste, S.; Plazier, M.; van der Loo, E.; Menovsky, T. Burst spinal cord stimulation: toward paresthesia-free pain suppression. *Neurosurgery* **2010**, *66*, 986–990, ISSN: 1524-4040, DOI: 10.1227/01.NEU.0000368153.44883.B3.

(141) De Ridder, D.; Vancamp, T.; Falowski, S. M.; Vanneste, S. All bursts are equal, but some are more equal (to burst firing): burstDR stimulation versus Boston burst stimulation. *Expert Rev Med Devices* **2020**, DOI: 10.1080/17434440.2020.1736560, ISSN: 1743-4440, https://www.tandfonline.com/doi/epdf/10.1080/17434440.2020.1736560?needAccess=true&role=button (accessed 04/18/2023).

(142) Meuwissen, K. P. V.; Gu, J. W.; Zhang, T. C.; Joosten, E. A. J. Response to: Fundamental Differences in Burst Stimulation Waveform Design: Eliminating Confusion in the Marketplace. *Neuromodulation: Technology at the Neural Interface* **2018**, *21*, 721–722, ISSN: 1094-7159, DOI: 10.1111/ner.12857, https://www.sciencedirect.com/science/article/pii/S1094715921018201 (accessed 10/09/2023).

(143) Falowski, S. M. An Observational Case Series of Spinal Cord Stimulation Waveforms Visualized on Intraoperative Neuromonitoring. *Neuromodulation: Technology at the Neural Interface* **2019**, *22*, 219–228, ISSN: 1094-7159, DOI: 10.1111/ner.12781, https://www.sciencedirect.com/science/article/pii/S1094715921018766 (accessed 10/09/2023).

(144) Jamali, M.; Jamali, Y.; Golshani, M. Theory of cyborg: a new approach to fish locomotion control. **2019**, DOI: 10.48550/arXiv.1904.12460.

(145) Jensen, A. L.; Durand, D. M. High Frequency Stimulation Can Block Axonal Conduction. *Experimental neurology* **2009**, *220*, 57, DOI: 10.1016/j.expneurol.2009.07.023, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2761511/ (accessed 10/09/2023).

(146) Santaniello, S.; McCarthy, M. M.; Montgomery, E. B.; Gale, J. T.; Kopell, N.; Sarma, S. V. Therapeutic mechanisms of high-frequency stimulation in Parkinson's disease and neural restoration via loop-based reinforcement. *Proceedings of the National Academy of Sciences* **2015**, *112*, E586–E595, DOI: 10.1073/pnas.1406549111, https://www.pnas.org/doi/10.1073/pnas.1406549111 (accessed 10/09/2023).

(147) Kühn, A. A.; Kempf, F.; Brücke, C.; Gaynor Doyle, L.; Martinez-Torres, I.; Pogosyan, A.; Trottenberg, T.; Kupsch, A.; Schneider, G.-H.; Hariz, M. I.; Vandenberghe, W.; Nuttin, B.; Brown, P. High-Frequency Stimulation of the Subthalamic Nucleus Suppresses Oscillatory β Activity in Patients with Parkinson's Disease in Parallel with Improvement in Motor Performance. *The Journal of Neuroscience* **2008**, *28*, 6165–6173, ISSN: 0270-6474, DOI: 10.1523/JNEUROSCI.0282-08.2008, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6670522/ (accessed 10/09/2023).

(148) Chen, S.; Du, M.; Wang, Y.; Li, Y.; Tong, B.; Qiu, J.; Wu, F.; Liu, Y. State of the art: non-invasive electrical stimulation for the treatment of chronic tinnitus. *Therapeutic Advances in Chronic Disease* **2023**, *14*, 20406223221148061, ISSN: 2040-6223, DOI: 10.1177/20406223221148061, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9969452/ (accessed 04/19/2023).

(149) Van der Groen, O.; Potok, W.; Wenderoth, N.; Edwards, G.; Mattingley, J. B.; Edwards, D. Using noise for the better: The effects of transcranial random noise stimulation on the brain and behavior. *Neuroscience and Biobehavioral Reviews* **2022**, *138*, 104702, ISSN: 1873-7528, DOI: 10.1016/j.neubiorev.2022.104702, https://www.sciencedirect.com/science/article/pii/S0149763422001919?via%3Dihub.

(150) Potok, W.; van der Groen, O.; Bächinger, M.; Edwards, D.; Wenderoth, N. Transcranial Random Noise Stimulation Modulates Neural Processing of Sensory and Motor Circuits, from Potential Cellular Mechanisms to Behavior: A Scoping Review. *eNeuro* **2022**, *9*, ENEURO.0248–21.2021, ISSN: 2373-2822, DOI: 10.1523/ENEURO.0248-21.2021, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8751854/.

(151) Herrera-Murillo, M. A.; Treviño, M.; Manjarrez, E. Random noise stimulation in the treatment of patients with neurological disorders. *Neural Regeneration Research* **2022**, *17*, 2557–2562, ISSN: 1673-5374, DOI: 10.4103/1673-5374.339474, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9165386/ (accessed 04/19/2023).

(152) Joos, K.; De Ridder, D.; Vanneste, S. The differential effect of low- versus high-frequency random noise stimulation in the treatment of tinnitus. *Experimental Brain Research* **2015**, *233*, 1433–1440, ISSN: 1432-1106, DOI: 10.1007/s00221-015-4217-9, https://doi.org/10.1007/s00221-015-4217-9 (accessed 10/05/2023).

(153) Henry, J. A. "Measurement" of Tinnitus. *Otology & Neurotology* **2016**, *37*, e276, ISSN: 1531-7129, DOI: 10.1097/MAO.0000000000001070.

(154) Deklerck, A. N.; Marechal, C.; Perez Fernandez, A. M.; Keppler, H.; Van Roost, D.; Dhooge, I. J., Invasive Neuromodulation as a Treatment for Tinnitus: A Systematic Review In *Neuromodulation*; 4; Blackwell Publishing Inc.: 2020, pp 451–462, DOI: 10.1111/ner.13042.

(155) Smit, J.; Janssen, M.; Schulze, H.; Jahanshahi, A.; Van Overbeeke, J.; Temel, Y.; Stokroos, R. Deep brain stimulation in tinnitus: Current and future perspectives. *Brain Research* **2015**, *1608*, 51–65, ISSN: 00068993, DOI: 10.1016/j.brainres.2015.02.050.

(156) Van Zwieten, G.; Devos, J. V. P.; Ackermans, L.; Brinkmann, P.; Dauven, L.; George, E. L. J.; Miranda, A; Janssen, L; Kremer, B.; Leue, C.; Schwartze, M.; Smit, J. V.; Janssen, M. L. F. Deep Brain Stimulation for Refractory Tinnitus: Pilot Study Protocol for a Randomized Double-Blinded Crossover Trial. **2019**, DOI: 10.21203/rs.3.rs-692616/v1, https://doi.org/10.21203/rs.3.rs-692616/v1.

(157) Van Zwieten, G.; Devos, J. V.; Kotz, S. A.; Ackermans, L.; Brinkmann, P.; Dauven, L.; George, E. L.; Janssen, A. M. L.; Kremer, B.; Leue, C.; Schwartze, M.; Temel, Y.; Smit, J. V.; Janssen, M. L. A Protocol to Investigate Deep Brain Stimulation for Refractory Tinnitus: From Rat Model to the Set-Up of a Human Pilot Study. *Audiology Research* **2023**, *13*, 49–63, ISSN: 20394349, DOI: 10.3390/audiolres13010005.

(158) De Ridder, D.; Vanneste, S.; Kovacs, S.; Sunaert, S.; Menovsky, T.; van de Heyning, P.; Moller, A. Transcranial magnetic stimulation and extradural electrodes implanted on secondary auditory cortex for tinnitus suppression. *Journal of Neurosurgery* **2011**, *114*, 903–911, ISSN: 0022-3085, DOI: 10.3171/2010.11.JNS10197.

(159) Kreuzer, P. M.; Poeppl, T. B.; Rupprecht, R.; Vielsmeier, V.; Lehner, A.; Langguth, B.; Schecklmann, M. Daily high-frequency transcranial random noise stimulation of bilateral temporal cortex in chronic tinnitus – a pilot study. *Scientific Reports* **2019**, *9*, 12274, ISSN: 2045-2322, DOI: 10.1038/s41598-019-48686-0, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6706578/ (accessed 10/10/2023).

(160) Langguth, B. Non-Invasive Neuromodulation for Tinnitus. *Journal of Audiology and Otology* **2020**, *24*, 113–118, ISSN: 2384-1621, DOI: 10.7874/jao.2020.00052.

(161) Kreuzer, P. M.; Poeppl, T. B.; Rupprecht, R.; Vielsmeier, V.; Lehner, A.; Langguth, B.; Schecklmann, M. Individualized repetitive transcranial magnetic stimulation treatment in chronic tinnitus? *Frontiers in Neurology* **2017**, *8*, DOI: 10.3389/fneur.2017.00126, ISSN: 16642295.

(162) Claes, L.; Stamberger, H.; Van de Heyning, P.; De Ridder, D.; Vanneste, S. Auditory Cortex tACS and tRNS for Tinnitus: Single versus Multiple Sessions. *Neural Plasticity* **2014**, *2014*, 436713, ISSN: 2090-5904, DOI: 10.1155/2014/436713, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4283418/ (accessed 10/10/2023).

(163) Moossavi, A.; Najafi, S., Transcranial direct current stimulation in treatment of tinnitus In *Auditory and Vestibular Research*; 1; Tehran University of Medical Sciences: 2021, pp 1–8, DOI: 10.18502/avr.v30i1.5305.

(164) Hebb, D., *The organization of behavior: a neurophysiological theory*, 1949, https://pure.mpg.de/rest/items/item_2346268_3/component/file_2346267/content (accessed 10/12/2023).

(165) To, W. T.; Ost, J.; Hart, J.; De Ridder, D.; Vanneste, S. The added value of auditory cortex transcranial random noise stimulation (tRNS) after bifrontal transcranial direct current stimulation (tDCS) for tinnitus. *Journal of Neural Transmission* **2017**, *124*, 79–88, ISSN: 1435-1463, DOI: 10.1007/s00702-016-1634-2, https://doi.org/10.1007/s00702-016-1634-2 (accessed 10/10/2023).

(166) Pal, N.; Maire, R.; Stephan, M. A.; Herrmann, F. R.; Benninger, D. H. Transcranial Direct Current Stimulation for the Treatment of Chronic Tinnitus: A Randomized Controlled Study. *Brain Stimulation* **2015**, *8*, 1101–1107, ISSN: 1935861X, DOI: 10.1016/j.brs.2015.06.014.

(167) Bae, E. B.; Lee, J. H.; Song, J. J. Single-Session of Combined tDCS-TMS May Increase Therapeutic Effects in Subjects With Tinnitus. *Frontiers in Neurology* **2020**, *11*, DOI: 10.3389/fneur.2020.00160, ISSN: 16642295.

(168) Roberts, L. E.; Bosnyak, D. J., Auditory Training in Tinnitus In *Textbook of Tinnitus*; Springer New York: New York, NY, 2011, pp 563–573, DOI: 10.1007/978-1-60761-145-5{_}72.

(169) Flor, H.; Hoffmann, D.; Struve, M.; Diesch, E., Auditory Discrimination Training for the Treatment of Tinnitus In *Applied Psychophysiology and Biofeedback*; 2, 2004, DOI: 10.1023/B:APBI.0000026637.77671.f4.

(170) Searchfield, G. D.; Morrison-Low, J.; Wise, K., Object identification and attention training for treating tinnitus In 2007, pp 441–460, DOI: 10.1016/S0079-6123(07)66043-9.

(171) Baguley, D. M.; Atlas, M. D., *Cochlear implants and tinnitus*, 2007, pp 347–355, DOI: 10.1016/S0079-6123(07)66033-6.

(172) Davis, P. B.; Paki, B.; Hanley, P. J. Neuromonics Tinnitus Treatment: Third Clinical Trial. *Ear & Hearing* **2007**, *28*, 242–259, ISSN: 0196-0202, DOI: 10.1097/AUD.0b013e3180312619.

(173) Hanley, P. J.; Davis, P. B. Treatment of Tinnitus With a Customized, Dynamic Acoustic Neural Stimulus: Underlying Principles and Clinical Efficacy. *Trends in Amplification* **2008**, *12*, 210–222, ISSN: 1084-7138, DOI: 10.1177/1084713808319942.

(174) Hanley, P. J.; Davis, P. B.; Paki, B.; Quinn, S. A.; Bellekom, S. R. Treatment of Tinnitus with a Customized, Dynamic Acoustic Neural Stimulus: Clinical Outcomes in General Private Practice. *Annals of Otology, Rhinology & Laryngology* **2008**, *117*, 791–799, ISSN: 0003-4894, DOI: 10.1177/000348940811701101.

(175) Okamoto, H.; Stracke, H.; Stoll, W.; Pantev, C. Listening to tailor-made notched music reduces tinnitus loudness and tinnitus-related auditory cortex activity. *Proceedings of the National Academy of Sciences* **2010**, *107*, 1207–1210, ISSN: 0027-8424, DOI: 10.1073/pnas.0911268107.

(176) Del Bo, L.; Baracca, G.; Forti, S.; Norena, A., Sound Stimulation In *Textbook of Tinnitus*; Springer New York: New York, NY, 2011, pp 597–604, DOI: 10.1007/978-1-60761-145-5{_}74.

(177) Shekhawat, G. S.; Searchfield, G. D.; Stinear, C. M. Randomized Trial of Transcranial Direct Current Stimulation and Hearing Aids for Tinnitus Management. *Neurorehabilitation and Neural Repair* **2014**, *28*, 410–419, ISSN: 1552-6844, DOI: 10.1177/1545968313508655.

(178) Kikidis, D.; Vassou, E.; Markatos, N.; Schlee, W.; Iliadou, E. Hearing Aid Fitting in Tinnitus: A Scoping Review of Methodological Aspects and Effect on Tinnitus Distress and Perception. *Journal of Clinical Medicine* **2021**, *10*, 2896, ISSN: 2077-0383, DOI: 10.3390/jcm10132896, https://www.mdpi.com/2077-0383/10/13/2896 (accessed 10/13/2023).

(179) Jacquemin, L.; Gilles, A.; Shekhawat, G. S. Hearing more to hear less: a scoping review of hearing aids for tinnitus relief. *International Journal of Audiology* **2022**, *61*, 887–895, ISSN: 1499-2027, DOI: 10.1080/14992027.2021.2007423, https://doi.org/10.1080/14992027.2021.2007423 (accessed 10/13/2023).

(180) Shekhawat, G. S.; Kobayashi, K.; Searchfield, G. D. Methodology for studying the transient effects of transcranial direct current stimulation combined with auditory residual inhibition on tinnitus. *Journal of Neuroscience Methods* **2015**, *239*, 28–33, ISSN: 0165-0270, DOI: 10.1016/j.jneumeth.2014.09.025, https://www.sciencedirect.com/science/article/pii/S0165027014003501 (accessed 10/10/2023).

(181) Teismann, H.; Wollbrink, A.; Okamoto, H.; Schlaug, G.; Rudack, C.; Pantev, C. Combining Transcranial Direct Current Stimulation and Tailor-Made Notched Music Training to Decrease Tinnitus-Related Distress – A Pilot Study. *PLoS ONE* **2014**, *9*, e89904, ISSN: 1932-6203, DOI: 10.1371/journal.pone.0089904, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3934956/ (accessed 10/13/2023).

(182) Lee, H. Y.; Choi, M. S.; Chang, D. S.; Cho, C.-S. Combined Bifrontal Transcranial Direct Current Stimulation and Tailor-Made Notched Music Training in Chronic Tinnitus. *Journal of Audiology & Otology* **2017**, *21*, 22–27, ISSN: 2384-1621, DOI: 10.7874/jao.2017.21.1.22, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5392009/ (accessed 10/13/2023).

(183) Koning, H. M. Somatosensory-Auditory Processing of the Fifth Cervical Nerve in Tinnitus. *The International Tinnitus Journal* **2021**, *24*, 49–53, ISSN: 0946-5448, DOI: 10.5935/0946-5448.20200009.

(184) Vanneste, S.; Plazier, M.; Van de Heyning, P.; De Ridder, D. Transcutaneous electrical nerve stimulation (TENS) of upper cervical nerve (C2) for the treatment of somatic tinnitus. *Experimental Brain Research* **2010**, *204*, 283–287, ISSN: 1432-1106, DOI: 10.1007/s00221-010-2304-5.

(185) Marks, K. L.; Martel, D. T.; Wu, C.; Basura, G. J.; Roberts, L. E.; Schvartz-Leyzac, K. C.; Shore, S. E. Auditory-somatosensory bimodal stimulation desynchronizes brain circuitry to reduce tinnitus in guinea pigs and humans. *Science Translational Medicine* **2018**, *10*, eaal3175, DOI: 10.1126/scitranslmed.aal3175, https://www.science.org/doi/10.1126/scitranslmed.aal3175 (accessed 04/06/2023).

(186) Spencer, S.; Mielczarek, M.; Olszewski, J.; Sereda, M.; Joossen, I.; Vermeersch, H.; Gilles, A.; Michiels, S. Effectiveness of bimodal auditory and electrical stimulation in patients with tinnitus: A feasibility study. *Frontiers in Neuroscience* **2022**, *16*, ISSN: 1662-453X, https://www.frontiersin.org/articles/10.3389/fnins.2022.971633 (accessed 04/06/2023).

(187) Wu, C.; Stefanescu, R. A.; Martel, D. T.; Shore, S. E. Listening to Another Sense: Somatosensory Integration in the Auditory System. *Cell and tissue research* **2015**, *361*, 233–250, ISSN: 0302-766X, DOI: 10.1007/s00441-014-2074-7, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4475675/ (accessed 04/06/2023).

(188) Shore, S. E.; Koehler, S.; Oldakowski, M.; Hughes, L. F.; Syed, S. Dorsal cochlear nucleus responses to somatosensory stimulation are enhanced after noise-induced hearing loss. *The European Journal of Neuroscience* **2008**, *27*, 155–168, ISSN: 1460-9568, DOI: 10.1111/j.1460-9568.2007.05983.x.

(189) Hamilton, C.; D'Arcy, S.; Pearlmutter, B. A.; Crispino, G.; Lalor, E. C.; Conlon, B. J. An Investigation of Feasibility and Safety of Bi-Modal Stimulation for the Treatment of Tinnitus: An Open-Label Pilot Study. *Neuromodulation: Technology at the Neural Interface* **2016**, *19*, 832–837, ISSN: 1094-7159, DOI: 10.1111/ner.12452, https://www.sciencedirect.com/science/article/pii/S109471592105412X (accessed 04/06/2023).

(190) Moller A, R; Langguth, B; de Ridder, D; Kleinjung, T, *Textbook of Tinnitus*; SpringerLink: 2011, ISBN: 978-1-60761-144-8, DOI: 10.1007/978-1-60761-145-5.

(191) Conlon, B.; Langguth, B.; Hamilton, C.; Hughes, S.; Meade, E.; Connor, C. O.; Schecklmann, M.; Hall, D. A.; Vanneste, S.; Leong, S. L.; Subramaniam, T.; D'Arcy, S.; Lim, H. H. Bimodal neuromodulation combining sound and tongue stimulation reduces tinnitus symptoms in a large randomized clinical study. *Science Translational Medicine* **2020**, *12*, eabb2830, DOI: 10.1126/scitranslmed.abb2830, https://www.science.org/doi/10.1126/scitranslmed.abb2830 (accessed 10/12/2023).

(192) Conlon, B.; Hamilton, C.; Meade, E.; Leong, S. L.; O Connor, C.; Langguth, B.; Vanneste, S.; Hall, D. A.; Hughes, S.; Lim, H. H. Different bimodal neuromodulation settings reduce tinnitus symptoms in a large randomized trial. *Scientific Reports* **2022**, *12*, 10845, ISSN: 2045-2322, DOI: 10.1038/s41598-022-13875-x, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9246951/ (accessed 04/06/2023).

(193) Conlon, B.; Hamilton, C.; Hughes, S.; Meade, E.; Hall, D. A.; Vanneste, S.; Langguth, B.; Lim, H. H. Noninvasive Bimodal Neuromodulation for the Treatment of Tinnitus: Protocol for a Second Large-Scale Double-Blind Randomized Clinical Trial to Optimize Stimulation Parameters. *JMIR Research Protocols* **2019**, *8*, e13176, DOI: 10.2196/13176.

(194) Kreuzer, P. M.; Landgrebe, M.; Resch, M.; Husser, O.; Schecklmann, M.; Geisreiter, F.; Poeppl, T. B.; Prasser, S. J.; Hajak, G.; Rupprecht, R.; Langguth, B. Feasibility, Safety and Efficacy of Transcutaneous Vagus Nerve Stimulation in Chronic Tinnitus: An Open Pilot Study. *Brain Stimulation* **2014**, *7*, 740–747, ISSN: 1935-861X, DOI: 10.1016/j.brs.2014.05.003, https://www.sciencedirect.com/science/article/pii/S1935861X14001739 (accessed 04/19/2023).

(195) Suk, W. C.; Kim, S. J.; Chang, D. S.; Lee, H. Y. Characteristics of Stimulus Intensity in Transcutaneous Vagus Nerve Stimulation for Chronic Tinnitus. *The Journal of International Advanced Otology* **2018**, *14*, 267–272, ISSN: 1308-7649, DOI: 10.5152/iao.2018.3977, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6354472/ (accessed 10/10/2023).

(196) Tutar, B.; Atar, S.; Berkiten, G.; Üstün, O.; Kumral, T. L.; Uyar, Y. The effect of transcutaneous electrical nerve stimulation (TENS) on chronic subjective tinnitus. *American Journal of Otolaryngology* **2020**, *41*, 102326, ISSN: 0196-0709, DOI: 10.1016/j.amjoto.2019.102326, https://www.sciencedirect.com/science/article/pii/S0196070919308737 (accessed 04/06/2023).

(197) Ridder, D. D.; Vanneste, S.; Engineer, N. D.; Kilgard, M. P. Safety and Efficacy of Vagus Nerve Stimulation Paired With Tones for the Treatment of Tinnitus: A Case Series. *Neuromodulation: Technology at the Neural Interface* **2014**, *17*, 170–179, ISSN: 1094-7159, DOI: 10.1111/ner.12127, https://www.sciencedirect.com/science/article/pii/S1094715914600544 (accessed 04/06/2023).

(198) Tyler, R.; Cacace, A.; Stocking, C.; Tarver, B.; Engineer, N.; Martin, J.; Deshpande, A.; Stecker, N.; Pereira, M.; Kilgard, M.; Burress, C.; Pierce, D.; Rennaker, R.; Vanneste, S. Vagus Nerve Stimulation Paired with Tones for the Treatment of Tinnitus: A Prospective Randomized Double-blind Controlled Pilot Study in Humans. *Scientific Reports* **2017**, *7*, 11960, ISSN: 2045-2322, DOI: 10.1038/s41598-017-12178-w.

(199) Vanneste, S.; Martin, J.; Rennaker, R. L.; Kilgard, M. P. Pairing sound with vagus nerve stimulation modulates cortical synchrony and phase coherence in tinnitus: An exploratory retrospective study. *Scientific Reports* **2017**, *7*, 17345, ISSN: 2045-2322, DOI: 10.1038/s41598-017-17750-y, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5725594/ (accessed 10/11/2023).

(200) Shim, H. J.; Kwak, M. Y.; An, Y.-H.; Kim, D. H.; Kim, Y. J.; Kim, H. J. Feasibility and Safety of Transcutaneous Vagus Nerve Stimulation Paired with Notched Music Therapy for the Treatment of Chronic Tinnitus. *Journal of Audiology & Otology* **2015**, *19*, 159–167, ISSN: 2384-1621, DOI: 10.7874/jao.2015.19.3.159, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4704553/ (accessed 04/19/2023).

(201) Lehtimäki, J.; Hyvärinen, P.; Ylikoski, M.; Bergholm, M.; Mäkelä, J. P.; Aarnisalo, A.; Pirvola, U.; Mäkitie, A.; Ylikoski, J. Transcutaneous vagus nerve stimulation in tinnitus: a pilot study. *Acta Oto-Laryngologica* **2013**, *133*, 378–382, ISSN: 0001-6489, DOI: 10.3109/00016489.2012.750736, https://doi.org/10.3109/00016489.2012.750736 (accessed 04/19/2023).

(202) De Ridder, D.; Vanneste, S., Visions on the future of medical devices in spinal cord stimulation: What medical device is needed? In *Expert Review of Medical Devices*; 3; Taylor and Francis Ltd: 2016, pp 233–242, DOI: 10.1586/17434440.2016.1136560.

(203) THE COUNCIL OF THE EUROPEAN COMMUNITIES, *Official Journal of the European Communities: COUNCIL DIRECTIVE 93/42/EEC,* 1993, https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX:01993L0042-20071011 (accessed 11/17/2023).

(204) Arduino, *Arduino Mega 2560 Rev3*, 2024, https://store.arduino.cc/products/arduino-mega-2560-rev3 (accessed 02/16/2024).

(205) Microchip, *ATmega640/V-1280/V-1281/V-2560/V-2561/V*, 2020, https://ww1.microchip.com/downloads/en/DeviceDoc/ATmega640-1280-1281-2560-2561-Datasheet-DS40002211A.pdf (accessed 02/16/2024).

(206) randomnerdtutorials, How to use ESP32 Dual Core with Arduino IDE In 2024, https://www.randomnerdtutorials.com/esp32-dual-core-arduino-ide/ (accessed 02/16/2024).

(207) Arduino reference, *attachInterrupt*, 2023, https://www.arduino.cc/reference/en/language/functions/external-interrupts/attachinterrupt/ (accessed 02/16/2024).

(208) upesy, *Using ESP32 timers in Arduino code*, 2023, https://www.upesy.com/blogs/tutorials/timer-esp32-with-arduino-code (accessed 02/16/2024).

(209) Siebeneicher, H., *Universal Asynchronous Receiver-Transmitter (UART)*, 2024, https://docs.arduino.cc/learn/communication/uart/ (accessed 02/16/2024).

(210) Scott Campbell, *BASICS OF THE I2C COMMUNICATION PROTOCOL*, 2023, https://www.circuitbasics.com/basics-of-the-i2c-communication-protocol/ (accessed 11/22/2023).

(211) Scott Campbell, *BASICS OF THE SPI COMMUNICATION PROTOCOL*, 2023, https://www.circuitbasics.com/basics-of-the-spi-communication-protocol/ (accessed 11/22/2023).

(212) Analog devices, *Data Sheet AD5601/AD5611/AD5621*, 2005, https://www.analog.com/media/en/technical-documentation/data-sheets/AD5601_5611_5621.pdf (accessed 02/16/2024).

(213) Reeder, Rob, *An Inside Look at High Speed Analog-to-Digital Converter Accuracy*, 2023, https://www.analog.com/en/technical-books/an-inside-look-at-high-speed-analog-to-digital-converter-accuracy.html#:~:text=A%201%20LSB%20in%20a,4096%2C%20or%20approximately%200.024%25. (accessed 11/21/2023).

(214) Engelen, M. C. W.; Van Den Heuvel, P. C. Control & Interface module for a High Frequency Arbitrary Waveform Neural Stimulator. **2018**, http://resolver.tudelft.nl/uuid:eb06f920-5fa4-4fee-980c-6d759bb15c9a (accessed 02/19/2024).

(215) Van Dongen, M.; Serdijn, W.; van Dongen, M.; Serdijn, W. Electrode–Tissue Interface During a Stimulation Cycle. *Design of Efficient and Safe Neural Stimulators: A Multidisciplinary Approach* **2016**, 25–47, ISSN: 2197-1854, DOI: 10.1007/978-3-319-28131-5.

(216) Barbon-Pedrina, A., *Design of a novel multi-modal stimulation device for the treatment of tinnitus*; Delft University of Technology: 2020, http://resolver.tudelft.nl/uuid:79331b7a-2e67-412d-affd-a22e18ed6547 (accessed 02/28/2023).

(217) Arduino, *Simple audio player*, 2023, https://docs.arduino.cc/tutorials/generic/simple-audio-player (accessed 11/20/2023).

(218) Carelton, *Getting your Arduino to pay music on your laptop*, 2023, https://www.cs.carleton.edu/faculty/dmusicant/cs102s18/pages/page380839.html (accessed 11/20/2023).

(219) HIFIBERRY, *HIFIBERRY DAC*, 2023, https://www.hifiberry.com/dacs (accessed 11/20/2023).

(220) Adafruit, *Adafruit "Music Maker" MP3 Shield for Arduino w/3W Stereo Amp - v1.0*, 2023, https://www.adafruit.com/product/1788 (accessed 11/20/2023).

(221) Adafruit; Clark, D., *danclarke / Adafruit_VS1053_Library*, 2018, https://github.com/danclarke/Adafruit_VS1053_Library (accessed 02/20/2024).

(222) vagusnerveclip.com, *Vagal TENS Ear Clips*, 2024, https://vagusnerveclip.com/products/vagal-tens-ear-clips (accessed 02/22/2024).

(223) Bax shop, *Alesis DRP 100*, 2024, https://www.bax-shop.nl/drum-koptelefoons/alesis-drp-100 (accessed 02/22/2024).

(224) Bax shop, *Devine PRO 900 DJ/Studio koptelefoon*, 2024, https://www.bax-shop.nl/koptelefoons/devine-pro-900-dj-studio-koptelefoon (accessed 02/22/2024).

(225) Duong, H. T. H. et al. Heart Rate Variability as an Indicator of Autonomic Nervous System Disturbance in Tetanus. *The American Journal of Tropical Medicine and Hygiene* **2020**, *102*, 403–407, ISSN: 0002-9637, DOI: 10.4269/ajtmh.19-0720.

List of Figures

List of Tables

A

Task Division

	Task	Student Name(s)
	Summary	Stef Zwartveld
Chapter 1	Introduction	Stef Zwartveld
Chapter 2	Literature study	Jonathan Kneepkens & Stef Zwartveld
Chapter 3	Methods	Stef Zwartveld
Chapter 4	Results	Stef Zwartveld
Chapter 5	Conclusion	Stef Zwartveld

Table A.1: Distribution of the workload

B

Requirements

Requirements MVP multimodal stimulator for tinnitus

Commented [JK1]: Misschien nog benoemen dat noise stimulation dan geen onderdeel is van het MVP

General

- The apparatus could be a large box, as long as no loose wires are visible.
- It must be operable by a hospital technician/clinician or De Ridder.
- Electrical stimulation will be done by burst stimulation on the tragus (transcutanous on the Vagus Nerve).
- Auditive stimulation will be done by a tinnitus matched auditory stimulus in the ear.
    - Sound amplitude SA is random in a session, varying from SA ± 6 dB
- The time delay Δt between the auditory and tragus stimulus:
    - Default: Zero time delay (Δt = 0)
    - Parameter: Set the time delay for a stimulation session (Δt = userinput)
- The inter burst time T_{ib} is random in a session, varying from T_{ib} = 25 ms-10 s
- The burst time T_b and sound time T_s are equal $T_b = T_s$

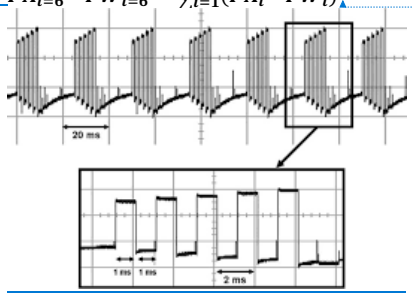
Deleted: 1

Auditory stimulus

- Tinnitus matched sound sample (Created using computer software, audacity)
- Played from an audio interface on the apparatus or a laptop on headphones.
- The length of the sound is **matched/proportional** to the electrical stimulus

Commented [JK2]: Or using headphones? Earplugs?

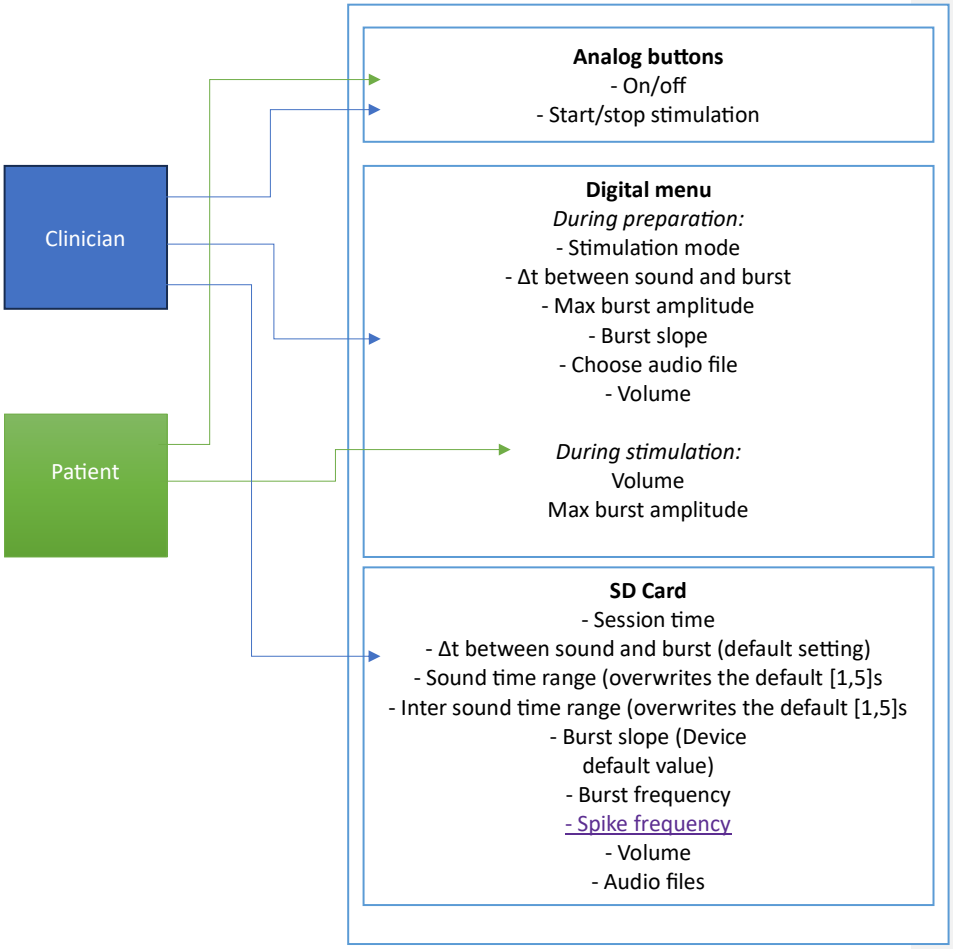
Electrical stimulus

- Burst stimulatie parameters;
    - 5 positieve mono-phasic pulses
        - Pulse width $PW_{i=\{1,5\}}$ = 1 ms
        - Pulse delay $PD_{i=\{1,5\}}$ = 1 ms
        - Pulse amplitude $PA_{i=\{1,5\}} = PA_0 + a*(i-1)$, where a and PA_0 = userinput
            - PA_0 is set in the range (to be defined)
            - a is set in the range (to be defined)
    - 1 negatieve phasebalancing pulse
        - **Pulse width $PW_{i=6}$ < 10 ms**
        - $PA_{i=6} * PW_{i=6} = \sum_{i=1}^{5}(PA_i * PW_i)$
        - 

Formatted: Font: Not Bold, English (UK)

Multimodal Device operation overview:

The SD card is personal per patient. All adjusted settings during device operation will be saved in a new file to the SD card and loaded when the device is turned on again.



General

- Power on/off: operated with a rocker switch and indicated with a LED.
- Start/stop session: operated with a physical button, indicated with a LED.
- Session time: operated for max 30 minutes, unless overwritten on a SD card.
- Error: Displayed on an LCD screen.
- Reset: Operated by a button. This is in case of an error. It will load the last saved settings from the SD card.
- The SD card can be added into the device through a push-push slot. The SD is personal for each patient.

Synchronization of burst and audio

- Δt, the delay between the audio and sound stimulation is adjusted in a digital menu by using an LCD screen and rotary knob.
    - If Δt is positive than the sound leads the burst.
    - The range is [–500,500] milliseconds with steps of more than 10 milliseconds.
- Sound time (length of sound and burststimulation sequence): Random lengths in the range of [1-5] seconds, with steps of 100 milliseconds.
- Intersound time (length between sound): Random lengths in the range of [1-5] seconds, with steps of 100 milliseconds.

Burst stimulation

- The following formula will be used for burst stimulation: https://www.desmos.com/calculator/ivzmz3zqzw
    - The P5 value is the max amplitude, max 3 mA.
    - The P1 defines the slope, which is default 0.4 (P1 = 40% of P5)
- Maximum amplitude: Adjustable through a digital menu with a rotary knob.
    - The amount will be displayed on the LCD screen.
    - This setting will be saved to the SD card.
- Burst slope: Hidden in the digital menu.
    - The first and last pulse amplitudes will be shown on the LCD screen.
    - This setting will be saved to the SD card.
- Burst frequency: Adjustable through the SD card.
    - The range is [0.1-40] Hz, default 40Hz.
- Spike frequency; adjustable through SD card
    -
- Burst active: A LED will be visible for the clinician while the burst sequence is active.

Deleted: Range: 50-800 Hz

Audio stimulation

- Multiple audio files are loaded on the SD card.
- Through the digital menu, the audio file can be chosen.
- Volume: Adjustable through the rotary knob
    - This setting will be saved to the SD card.
- Audio active: A LED will be visible for the clinician while the audio stimulation is active.

Control group functions:

- These and normal modes can be chosen through the digital menu
- Audio and no burst (see the appendix)
- Unsynced audio (see appendix)
    - Is this “unsynced audio” correct?. Both the audio, burst, and time in between are random between [1,5] seconds

Are we on the same track in terms of terminology and meaning?

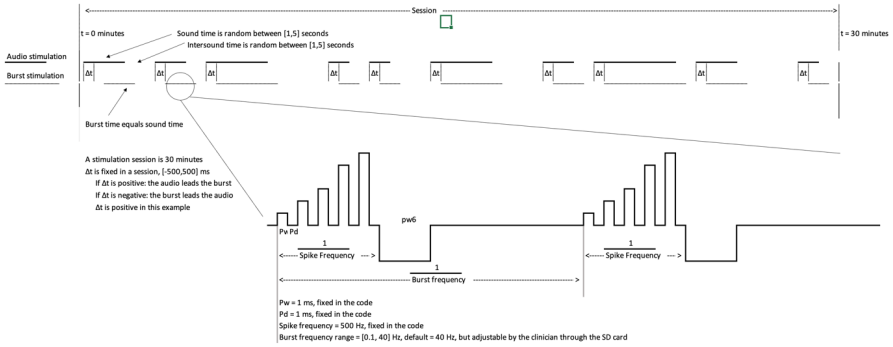


Figure 1: (upper graph) An example session with audiostimulus (black) followed by burst stimulus (grey) with a Δt delay. (lower graph) The burst stimulus time is equal to the soundtime (1-5 s) and consists of a sequence of bursts defined by its spike frequency and burst frequency.

Burst time equals sound time is not essential, but practical for programming.

Are these two optionss the correct options for the control group settings?

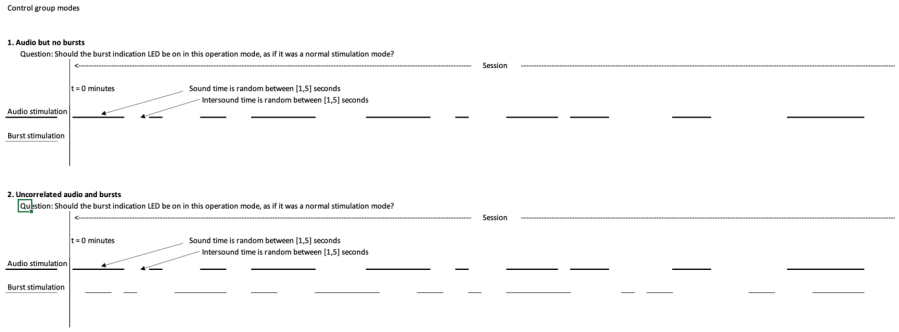


Figure 2: The Two placebo operating modes: (1) With audio stimulation and without burst stimulation and (2) both audio stimulation and burst stimulation, where both are administered randomly and uncorrelated.

For the control settings, it would be good if the LED is active, but if possible in another color?

C

Usermanual

USER MANUAL

DUALSTIM

A. MicroSD Card

1. Format the MicroSD card to FAT16/FAT32.
2. Put the MicroSD into the DualSTIM.
3. The DualSTIM writes “datalog.txt” file containting default parameters and Readme.txt file to the MicroSD.
4. Read the Readme.txt file carefully. Here you will find information about adjusitng “datalog.txt”.
5. Adjust the parameters in datalog.txt file if nessecary.
6. Put the MircoSD card back into the computer.
7. Upload at least one tinnitus matched audiofile with the following requirements:
   a. At least 6 seconds long
   b. A name with 8.3 characters (e.g. track001.mp3)
   c. Formats MP3, WAV, WMA, OGG.
8. Put the MicroSD card back into the DualSTIM.

B. Startup

1. Turn on the DualSTIM using the red switch on the side of the DualSTIM
2. The display also indicates "Loading... " while updating the MicroSD cards data and loading settings.
3. If anything is wrong, an error will be displayed. Always restart the DualSTIM after fixing an error. Refer to the “common errors” of the instruction for more information.
4. After loading the display shows "Ready to stimulate!" and continues to the menu to adjust parameters.
5. The display will indicate the battery life in the left top corner with "b=...%"

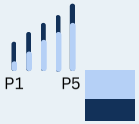
C.Adjust parameters

1. Rotate the rotational button to select the parameter to adjust.
2. Press the rotational button to adjust the parameter. The cursor at the end of the line will blink.
3. Rotate clockwise or anticlockwise to increase or decrease the parameter value, respectively.
4. Press the rotational button to confirm the new parameter. The cursor stops blinking.

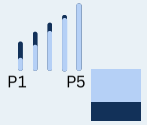
D. Menu parameters

I. Volume (Audio)

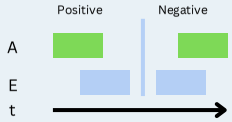
II. Max amplitude



III. P1 percentage of P5*



IV. Delay E-A**



V. Stimulation mode

VI. Audio file

> Track001.MP3
Track002.WAV
Track003.WMA
Track004.OGG

** P1 is the amplitude of the first pulse of the burst
P5 is the amplitude of the fifth pulse of the burst
** E stands for Electrical stimulation
A stands for Acoustical stimulation

E. Stimulation

I. Start

1. When the parameters are correct, press the silver "Start/stop" button.

II. During stimulation

1. The red LED indicator next to the screen will turn on while stimulation is active.
2. Depending on the stimulation mode, the LED indicators on the side will turn on:
   a. The blue LED indicates acoustic stimulation.
   b. The green LED indicates electrical stimulation.
3. Parameters can be adjusted during stimulating, see "C. Adjust sparameters".

III.Stop

1. To stop stimulation, press the silver “Start/stop” button again.
2. The DualSTIM stops stimulating automatically after 30 minutes, or the amount of minutes added to the first index of the last line in the datalog.txt file.

F. Common errors and possible solutions

1. **ER:Musicplayer not working**: There is a problem with the connection between the microcontroller and the mp3 player inside the DualSTIM. Technical support is needed to fix this error.
2. **ER:insert SD&restart**: The DualSTIM does not recognize the MicroSD card. The MicroSD could be missing or formatted in an other format than FAT32 or FAT16. The MicroSD could also have been ejected unsafe from the computer, in that case restart the DualSTIM a few times.
3. **DREQ pin is not an interrupt pin**: There is a problem with the connection between the microcontroller and the mp3 player inside the DualSTIM. Technical support is needed to fix this error.
4. **ER:opening audiofile**: The audiofile could be missing or corrupted. The MicroSD card could have been ejected during stimulation. Insert it and restart the DualSTIM.
5. **ER:Max 10 audiofiles exceeded on SD card**: For storage reasons, the DualSTIM is restricted to a maximum of 10 audiofiles on the MicroSD card.
6. **ER:No audio on SD**: There are no audiofiles on the MicroSD card. Add at least one audiofile.
7. **ER:Battery level low Change the battery**: Turn the DualSTIM off. CAREFULLY open up the lit by using a star screwdriver. The user interface components are still connected to it. Exchange the battery with a new 9V battery. Before closing the lit, reconnect the red LED to the left pin and the start/stop button to the right pin on the red electronic board.
8. **ER:Datalog not open**: The DualSTIM can not open the “datalog.txt” file on the MicroSD card. Put the MicroSD card in the computer. If datalog.txt exists, copy it to the computer and delete it from the MicroSD. Also empty the bin for the MicroSD card. Eject the MicroSD card from the computer and put it back in the DualSTIM.
9. **ER:readme.txt not open**: The DualSTIM is not able to open the “readme.txt” file. Try the same method as F.8
10. **ER:SD max amplitude change to max 3000**: The max amplitude value in “datalog.txt” exceeds the maximum of 3000 microAmpères. Change the value to a maximum of 3000 or lower. This measure is taken for safety reasons.
11. **ER:SD datalog exceed max params. Use . as decimal point eg 0.9**: in datalog.txt file, there are to many parameters. It could be that the burst frequency is written with a comma in stead of a dot as decimal seperator. Write the burst frequency with a dot.

G. Adjusting datalog.txt

A datalog.txt file will be written to the card with a list of default values. Every time a parameter is changed in the menu, a new line of settings is written to datalog.txt. These values can be edited on the card, corresponding to the following order

1. Max stimulation time in minutes
2. Burst Frequency in Hz, ranging from 0.1 to 40 Hz. IMPORTANT: Use a dot as decimal seperator, e.g. 0.1
3. Volume value
4. Volume stepsize
5. Volume minimum
6. Volume maximum (Does not higher the volume but creates a bigger range (softer audio possible)
7. Burst amplitude P5 value
8. Burst amplitude P5 stepsize
9. Burst amplitude P5 minimum
10. Burst amplitude P5 maximum (MAX 3000 for safety!)
11. Burst amplitude P1 value
12. Burst amplitude P1 stepsize
13. Burst amplitude P1 minimum
14. Burst amplitude P1 maximum
15. Time delay E-A value
16. Time delay E-A stepsize
17. Time delay E-A minimum
18. Time delay E-A maximum
19. Stimulation mode value (Don't edit)
20. Stimulation mode stepsize (Don't edit)
21. Stimulation mode minimum (Don't edit)
22. Stimulation mode maximum (Don't edit)
23. Audio file value
24. Audio file stepsize (Don't edit)
25. Audio file minimum (Will be overwritten)
26. Audio file maximum (Will be overwritten)

More detailed information can be found on https://bit.ly/MultiModalStimulator

D

Software

D.1. MultiModalStimulator.ino

```
/**
  *     Project:                    Multimodal stimulator for tinnitus
  * Version:        12.0.0
  * Author:                 Stef Zwartveld, Jonathan Kneepkens
  * Summary:        The Multimodal stimulator pairs electrical stimulation with audio sounds
  *                 It has a ESP32 dev kit with 18 pins, connected to the custom DualSTIM PCB and the Adafruit Music Maker.
  *                 The ESP32 has two cores. Arduino runs on core1, where the user interface, menu and Music Maker run their processes
  *                 On core0, the electrical stimulation is run, which is dependent on timer2, running every 1ms.
  *                 The synchronisation of the start/stop moments of electrical and audio stimulation is dependent on timer3
  * Clickup task:       CU-86bx1dh0r_Code---Elaborate-comments_Stef-Zwartveld
  * GitHub:         https://github.com/StefZw/MultiModalStimulator, https://bit.ly/MultiModalStimulator
  */

/** Tips for understanding this big onepager document:
  * - The text indents can be collapsed or expanded. When first going through the code, close all indents,
  *   which gives some overview of all functions.
  * - When searching for the origin of variables/functions, hover over a variable to see its,
  *   or "ctrl/cmd + click" the variable to jump to its origin.
  * - Searching is also fast with selecting a variable and pressing "ctrl/cmd + F".
  * - Functions are (mostly) not taking or returning parameters, to make sure they are globally available
  *   and adjusted on the go. This could be changed if there would be a security reason for it (e.g. connection to wifi)
  */

/* Libraries */
```

```
25  #include <SPI.h> // For communication between the Adafruit music maker
        and the DAC
26  #import "Adafruit_VS1053_Library_ESP32/Adafruit_VS1053_ESP32.h" //
        Edited version for the ESP32 to use the Adafruit Music Maker
27  #import "Adafruit_VS1053_Library_ESP32/Adafruit_VS1053_ESP32.cpp" //
        https://www.adafruit.com/product/1790
28  #include <SD.h>   // For reading and writing the SD card
29  #include <Wire.h> // For the connection of the SD card
30  #include <LiquidCrystal_I2C.h> // For the LCD screen
31  #include <ESPRotary.h> // For the rotational menu button knob
32  #include <Button2.h> // For the press actions of the menu and start/stop
        knobs
33
34 /* Initialization part */
35  /* -------------Pin definitions------------- */
36    //Stimulation circuit pin definition
37      // ESP32 HSPI connection pins to the DAC:
38        // HSPI-CLK pin 14; Taken care of by the SPI library
39        // HSPI-MOSI is pin 13; Taken care of by the SPI library
40        // HSPI-MISO is pin 12; Taken care of by the SPI library
41      const byte DAC_CS = 15;     // HSPI-CS Chip select pin for DAC
42      // H-bridge connection pins
43      const byte StimOff = 25;    // If stimoff == low, than the hbridge
            is active, thus stimulation is possible. Controls the NMOS in
            front of H-brdige, if this pin is high, current leaks to GND,
            thus stimulation is off
44      const byte LogicAtoB = 26;  // Turn logic gates so that current goes
            from A to B (or positive) when high
45      const byte LogicBtoA = 27;  // Turn logic gates so that current goes
            from B to A (or negative) when high
46      // Both Logics low is stimulation off and connection to ground,
            which is preferable when no current needs to flow.
47
48    // User Interface
49      // LCD uses I2C connection
50        // SDA is pin 21; Taken care of by the LCD library
51        // SCL is pin 22; Taken care of by the LCD library
52        LiquidCrystal_I2C lcd(0x27,20,4);  // set the LCD address to 0x27
              for a 16 chars and 2 line display
53      // Rotary encoder
54        #define rotaryButtonPin 32
55        #define ROTARY_PIN1     39 //VP
56        #define ROTARY_PIN2     36 //VN
57        #define CLICKS_PER_STEP 4 // this number depends on your rotary
              encoder
58        ESPRotary encoderRotate;  // Encoder button rotation object
59        Button2 encoderButton;    // Encoder button click object
60      // startStopButton
61        const byte startStopPin = 35;  // Start stop button pin
62        Button2 startStopButton;       // Start/stop button click object
63      // LED Indicators
64        const byte stimLED = 0;         // LED on while stimulating
65        const byte burstLED = 2;        // LED on while burst stimulation
              is on
66        const byte audioLED = 12;       // LED on while audio stimulation
              is on
```

```
67        // Batterylevel
68          const byte batteryPin = 33;
69
70      // Adafruit Music Maker
71        #define SHIELD_RESET 3     // VS1053 reset pin (unused!)
72        #define SHIELD_CS 5        // VS1053 chip select pin (output)
73        #define SHIELD_DCS 17      // VS1053 Data/command select pin (output)
74        #define CARDCS 16          // 14 Card chip select pin
75        #define DREQ 4             // 3 on adafruit VS1053 Data request,
              ideally an Interrupt pin
76        Adafruit_VS1053_ESP32_FilePlayer musicPlayer =
              Adafruit_VS1053_ESP32_FilePlayer(SHIELD_RESET, SHIELD_CS,
              SHIELD_DCS, DREQ, CARDCS);  // create shield-example object
77
78    /* -------------Parameter definitions------------- */
79      // startStopButton
80        bool stimulationActive = false;
81      // Menu
82        // Menu
83        File dataFile; // Object used for reading and writing the SD card
84        const byte menuLength = 6;                    // The amount of
              parameters in the menu
85        const byte menuLengthY = 2;                   // Only used for the
              menu names and units
86        byte menuPosition = 0;                        // Menu position,
              adjusted by onEncoderRotateRight() and onEncoderRotateLeft()
87        const byte menuPositionMax = menuLength - 1;  // Define the max menu
               position
88        bool adjustSettings = 0;                      // Checks if a
              parameter is currently being edited
89        char* menu[menuLength][menuLengthY] = {{"Volume","%"},        //
              IMPORTANT: Keep this order of settings
90                              {"Burstamplitude P5","uA"},             // OR
                                  update updateSettings() when changing the
                                  order or adding settings
91                              {"Burstamplitude P1","% of P5"},
92                              {"Time delay E-A","miliseconds"},
93                              {"Stimulation mode","mode"},
94                              // mode 1 is audio+burst
95                              // mode 2 is uncorrelated audio and bursts
96                              // mode 3 is audio but no bursts
97                              {"Audiofile"," "}};// KEEP AUDIOFILE IN LAST
                                  INDEX (important for dataLog() and dataRead
                                  ())
98        int16_t value[menuLength] = {10,10,30,0, 1, 0};          //
              Important: this should be a SIGNED integer type to not break the
              menu
99        int16_t step[menuLength] = {10,10,10,100, 1, 1};         //
              Important: this should be a SIGNED integer type to not break the
              menu
100       int16_t minimum[menuLength] = {0,0,0,-500, 1, 0};         //
              Important: this should be a SIGNED integer type to not break the
              menu
101       int16_t maximum[menuLength] = {100,500,100,500, 3, 0};   //
              Important: this should be a SIGNED integer type to not break the
              menu
```

```
102       char* audioFileList[10] = {"0000000000000","0000000000000","
              0000000000000","0000000000000","0000000000000","0000000000000","
              0000000000000","0000000000000","0000000000000","0000000000000"};
              // Audio filename pointers
103       char name[10][14] = {"","","","","","","","","",""}; // Audio
              filename variables
104       char* audioFile;  // Current chosen audio file
105       uint8_t maxAudioFiles = 0; // Max audiofiles is used to reserve
              space in createAudioFileList()
106       uint16_t lastBatteryLevel = 4095; // ESP32 uses a 12-bit ADC, so
              this is the max value of the battery
107     // Synchronization
108       hw_timer_t *syncTimer = NULL;                // Timer object used to
              synchronise burst and audio, running on timer3, and updates every
               0.01s
109       volatile bool syncPulse = false;             // Is the pulse used to
              update the syncCounter every 0.01s
110       volatile uint16_t syncCounter = 0;            // Keeps time, counts
              the amount of ticks which the burst and audio start/stop moments
              depend on
111       int16_t timeDelay = 0;                        // Variable filled by
              the menu setting for the time delay, used in calculateTimeDelays
              ()
112       uint16_t burstDelay = 0;                      // Used to determine
              the start and stop moment of burst stimulation
113       uint16_t burstDelayOffset = 0;               //
114       uint16_t audioDelay = 0;                      // Used to determine
              the start and stop moment of audio stimulation
115       bool burstIsDelayed = 0;                     // Used to determine
              when the next stimLength and pauzeLength should be calculated
116       uint16_t stimLength = 100;                    // Determines the
              stimulation length, is updated by a random number between 1s and
              5s after one stimulation. Default 100 = 1s
117       uint16_t pauzeLength = 100;                   // Determines the
              pauze length, is updated by a random number between 1s and 5s
              after one stimulation. Default 100 = 1s
118     // Burst parameters
119       hw_timer_t *burstTimer = NULL;               // Burst timer object,
              running timer2, and updates burstPulse every 1ms
120       volatile bool burstPulse = false;            // Is the pulse used to
              update the burstIndex every 1ms
121       volatile uint8_t burstIndex = 0;             // Used to loop through
              PulseAmplitudeArray to send values to the DAC
122       volatile bool burstOn = false;               // Updated in loop()
              when the burst stimulation should be activated. This is
              recognized in loopCore0()
123       volatile bool burstOff = false;              // Updated in loop()
              when the burst stimulation should be deactivated. This is
              recognized in loopCore0()
124       SPIClass* burstHspi = NULL;                  // Uninitalised pointer
              to SPI object connecting HSPI pins to the DAC
125       const byte DAC_MODE = 0;                     // Determines the mode
              of the DAC
126       float P1_ElectricalStim = value[2]*0.01;     // PulseAmplitude of 1th
               Pulse, value between 0 and 1, percentage of P5
```

```
127       float P5_ElectricalStim = value[1]*0.001;   // PulseAmplitude of 5th
              pulse, value between 0 and 3 mA
128       //burst amplitude
129       uint16_t* PulseAmplitudeArray;              // Burst values per 1ms,
              filled by COMPUTE_PulseAmplitudeArray based on P1_ElectricalStim
              and P5_ElectricalStim
130       uint16_t maxBurstLength = 25;                // Determines the
             length before the next burst is started (burst+interval) in ms,
             edited via SD card
131     // time keeping parameters
132       uint32_t stimulationTime;                                        //
             stimulationTime is the current time from the start of stimulation
133       uint32_t maxStimulationTimeSD = 30;                               //
             default stimulation time in minutes
134       uint32_t maxStimulationTime = maxStimulationTimeSD * 6000;   //
             Convert from 0.01s to minutes
135     // Core0 loop initiation
136       TaskHandle_t LoopCore0;                     // Task object to run
             electrical stimulation on core0
137     // Interrupt Service Routine functions
138       //These should be before the setup to make sure they are findable by
              the timer initiations in setup()
139
140       /*! ISR Synchroniser
141         * @brief Set syncPulse true, timer based interrut service routine
142         * @param syncPulse is a flagged true every time the function runs
143         */
144         void IRAM_ATTR synchroniser() {
145           syncPulse = true;
146         }
147
148       /*! ISR Burstsequence
149         * @brief set burstPulse true, timer based interrut service routine
150         * @param burstPulse is a flagged true every time the function runs
151         */
152         void IRAM_ATTR burstSequence() {
153           burstPulse = true;
154         }
155
156 /*! setup
157   * @brief Setup is runned once at startup, initializing the lcd, DAC, H-
        bridge, timer, buttons, audioplayer, SD card, menu and core0
158   */
159 void setup() {
160   // Serial.begin(115200); // Serial connection to a computer is usefull
        when debugging
161     pinMode(stimLED,OUTPUT);
162     digitalWrite(stimLED, LOW);
163   /* LCD */
164     lcd.init();                                 // initialize the lcd screen
165     lcd.backlight();
166     lcd.print(F("Welcome to DualSTIM"));        // Print a welcome message
167     lcd.setCursor(0,2);                         // Select the third line
168     lcd.print(F("...LOADING"));                 // Print a loading message
169   /* Burststimulation - DAC and H-bridge*/
170     // Hbridge
```

```
171       pinMode(StimOff, OUTPUT);                // Define the StimOff pin as
              an output
172       pinMode(LogicAtoB, OUTPUT);              // Define logicAtoB as
              output
173       pinMode(LogicBtoA, OUTPUT);              // Define logicBtoA as
              output
174     // Dac initialization                       // And startup safety
          measrures
175       burstHspi = new SPIClass(HSPI);          // Create a SPI class for
              the connection from the HSPI pins to the dac
176       burstHspi->begin();                      // Initialize the HSPI
              connection
177       pinMode(burstHspi->pinSS(), OUTPUT);     // Set the chipselect pin
              for the DAC as an output
178       DAC_Write(0, 1);                         // Point at which device is
              started, this is the reference point
179       digitalWrite(StimOff, HIGH);             // If stimoff == low, than
              the hbridge is active,thus stimulation is possible
180       delay(1);                                // Delay 1 ms
181       digitalWrite(LogicBtoA, LOW);            // Turn logic gates so that
              current goes from A to B (or positive) when high
182       digitalWrite(LogicBtoA, LOW);            // Turn logic gates so that
              current goes from B to A (or negative) when high
183   /* Time synchronisation */
184     syncTimer = timerBegin(3, 80, true);       // Set timer 3 with 80 MHz
          devided by 80 = 1MHz clockspeed, counting up
185     timerAlarmWrite(syncTimer, 10000, true); // Fire the synctimer
          function  every 0.01 seconds (10.000/1MHz), and autoreload
186     timerAlarmEnable(syncTimer);               // Enable the alarm such
          that the function runs
187     stimLength = random(100,500);                // Set a first stimulation
            length between 1 and 5 seconds
188     pauzeLength = random(100,500);               // Set a first stimulation
            pauze between 1 and 5 seconds
189     pinMode(burstLED,OUTPUT);                  // Set the burst LED
          indicator pin as an output
190     pinMode(audioLED,OUTPUT);                  // Set the audio LED
          indicator pin as an output
191   /* Buttons and indicators*/
192     // LEDss
193     pinMode(stimLED,OUTPUT);                   // Set the stimulation
          active LED indicator pin as an output
194     pinMode(burstLED,OUTPUT);                  // Set the burst LED
          indicator pin as an output
195     pinMode(audioLED,OUTPUT);                  // Set the audio LED
          indicator pin as an output
196     //startStop
197     startStopButton.begin(startStopPin);                                  //
            Initialize start/stop button
198     startStopButton.setTapHandler(startStopButtonHandler);                //
            Link function startStopButtonHandler() to the setTapHandler (fires
            at letting go of the button)
199     //menu rotate
200     encoderRotate.begin(ROTARY_PIN1, ROTARY_PIN2, CLICKS_PER_STEP);   //
          Initialize rotaryencoder rotation
```

```
201     encoderRotate.setLeftRotationHandler(onEncoderRotateLeft);        //
            Link function onEncoderRotateLeft() to the left rotation
202     encoderRotate.setRightRotationHandler(onEncoderRotateRight);      //
            Link function onEncoderRotateLeft() to the right rotation
203     //menu button
204     encoderButton.begin(rotaryButtonPin);                             //
            Initialize rotaryencoder button
205     encoderButton.setTapHandler(onEncoderButtonClick);                //
            Link function onEncoderButtonClick() to the setTapHandler (fires at
             letting go of the button)
206   /* audio player */
207     pinMode(CARDCS, OUTPUT);                          // Set the card chip
            select pin as output
208     digitalWrite(CARDCS, HIGH);                       // Set the card chip
            select pin HIGH (inactive)
209     // initialise the music player
210     if (!musicPlayer.begin()) {                       // Initialize the
            audioplayer
211       lcd.setCursor(0,3);
212       lcd.print(F("ER:Musicplayer not working"));     // Throw an error if
              the connection to the music player is not working
213       while (1);                                      // Don't do anything
              more
214     }
215     // Initialize SD card
216     if (!SD.begin(CARDCS)) {                          // Initialize the SD
            card
217       lcd.setCursor(0,3);
218       lcd.print(F("ER:insert SD&restart"));           // Throw an error if
              the connection to the music player is not working
219       while (1);                                      // don't do anything
              more
220     }
221     musicPlayer.sineTest(0x44, 300);                  // Make a tone to
            indicate music player is working
222     // The music player uses interrupts
223     if (! musicPlayer.useInterrupt(VS1053_FILEPLAYER_PIN_INT)) {    // Use
             an interrupt pin for communication
224       lcd.setCursor(0,3);
225       lcd.println(F("DREQ pin is not an interrupt pin"));
226       while (1);                                        // don't do anything
              more
227     }
228   /* Read data from the SD card */
229     addReadmeToSD();                                  // Add a readme file
             to the SD card
230     dataRead();                                       // Read the dataLog.
            txt file from the SD card and put information in the menu
231     delay(1);
232     P1_ElectricalStim = value[2]*0.01;                // PulseAmplitude of
             1th Pulse, value between 0 and 1, percentage of P5
233     P5_ElectricalStim = value[1]*0.001;               // PulseAmplitude of
             5th pulse, value between 0 and 3 mA
234     PulseAmplitudeArray = COMPUTE_PulseAmplitudeArray(P1_ElectricalStim,
            P5_ElectricalStim);     // Calculate the pulses for the burst
            stimulation
```

```
235     calculateTimeDelays();                                  // Calculate the
            time delays which are updated from the SD card
236     maxStimulationTime = maxStimulationTimeSD * 6000;   // Convert from
            0.01s to minutes
237     musicPlayer.setVolume(maximum[0]-value[0],maximum[0]-value[0]); // Set
             volume for left, right channels. lower numbers == louder volume!
238   /* run audio file check */
239     createAudioFileList();                                  // Check the SD
            card on audiofiles and add the names to an array such that they are
             available in the menu
240     audioFile = audioFileList[value[menuPositionMax]];  // Set the audio
            file to the one which was last selected on the SD card
241   /* init core0 */
242     delay(1000);                                            // Wait for a
            second before initializing the second core
243     xTaskCreatePinnedToCore(                                // Add the task to
             core0, used for burst stimulation
244     loopCore0,          /* Function to implement the task */
245     "LoopCore0",        /* Name of the task */
246     4096,               /* Stack size in words */
247     NULL,               /* Task input parameter */
248     3,                  /* Priority of the task (higher is higher prio) */
249     &LoopCore0,         /* Task handle. */
250     0);                 /* Core where the task should run */
251     delay(1000);                                            // wait again to
            give
252   /* LCD ready message */
253     lcd.clear();                                            // Clear the LCD
            screen before printing a new message
254     lcd.print(F("Welcome DualSTIM"));                       // Print a welcome
             message
255     lcd.setCursor(0,2);
256     lcd.print(F("Ready to stimulate!"));                    // Give a notice
            that the stimulation can be started
257     delay(2000);                                            // Keep the
            message in the screen for a while
258     lcdPrintSettings();                                     // Open up the
            menu to adjust parameters
259 }
260
261 /*! loop User interface, synchronisation and audio
262   * @brief loop is running continuously on core1. It checks changes in the
        buttons of the interface.
263   * In the case stimulation is active, it runs the start/stop functions
         for burst and audio, and determines the next start/stop moments
264   */
265 void loop() {
266   /* Button checks */
267     encoderButton.loop();                                       // Check
            encoderButton changes every loop, if so run onEncoderButtonClick()
268     startStopButton.loop();                                     // Check
            startStopButton changes every loop, if so run
            startStopButtonHandler()
269     encoderRotate.loop();                                       // Check
            encoderRotate changes every loop, if so run onEncoderRotateLeft()
            or onEncoderRotateRight()
```

```
270   /* Stimulation start/stop functions*/
271     if (stimulationActive && syncPulse) {                    //
          StimulationActive = true after the startStopButton has been pressed
          , and syncpulse = true by synchroniser()
272       syncPulse = !syncPulse;                                // Set
            syncPulse directly false to avoid a second iteration through this
             loop
273       if (stimulationTime < maxStimulationTime) {            // Check if
            the maximum stimulation time is not exceeded
274       /* audio burst start stop functions*/
275         // start Burst
276         if (syncCounter == burstDelay) {                       // Check if
              burst stimulation should start, run by loopCore0() and
              burstSequence()
277           burstIndex = 0;                                      // Reset the
                index of PulseAmplitudeArray to the first index
278           if (!(value[4] == 3)){                               //
                Stimulation mode 3 is NOT including burst, the others will
                start burst stimulation
279             burstOn = !burstOn;                                // Activate
                  timer2 interrupt for the burst in loopCore0()
280           }
281         }
282         // start audio
283         if (syncCounter == audioDelay) {                       // Check if
              audio stimulation should start, run by loop() on a pin
              interrupt service routine
284           if (! musicPlayer.startPlayingFile(audioFile)) {     // If audio
                is not playing yet, play the selected audiofile
285             lcd.setCursor(0,3);
286             lcd.print(F("ER:opening audiofile"));                // Error
                  message, audiofile is not opening
287             stimulationOff();
288             while (1);                                           // Stay
                  here, restart is nessecary
289           }
290           digitalWrite(audioLED, HIGH);                        // Activate
                audioLED to show audio stimulation is active
291         }
292         // stop burst
293         if (syncCounter == (burstDelay + stimLength)) {        // Check if
              the stimLength for the burst has expired
294           if (!(value[4] == 3)){                               //
                Stimulation mode 3 is NOT including burst, the others will
                stop burst stimulation
295             burstOff = !burstOff;                              //
                  Deactivate timer2 interrupt for the burst in loopCore0()
296           }
297         }
298         // stop audio
299         if (syncCounter == (audioDelay + stimLength)) {        // Check if
              the stimLength for the audio has expired
300           musicPlayer.stopPlaying();                           // Stop the
                audio
301           digitalWrite(audioLED, LOW);                         //
                Deactivate audioLED to indicate audio stimulation has stopped
```

```
302         }
303
304       /* Simulation timing functions */
305         if (syncCounter == (burstDelay + stimLength + pauzeLength) &&
              burstIsDelayed) { // If burst stimulation is the last one to
              end
306           syncCounter = 0;                                    // Reset the
                syncCounter
307           stimLength = random(100,500);                        //
                Determine a new random stimLength
308           pauzeLength = random(100,500);                       //
                Determine a new random pauzeLength
309           if (value[4] == 2){                                 // In case
                of stimulation mode 2, uncorrelated audio and burst
310             burstDelay = random(0,200);                        // Set the
                  burstDelay random
311             audioDelay = random(0,200);                        // Set the
                  audioDelay random
312           }
313         } else if (syncCounter == (audioDelay + stimLength + pauzeLength)
              && !burstIsDelayed) { // If audio stimulation is the last one
              to end
314           syncCounter = 0;                                    // Reset the
                syncCounter
315           stimLength = random(100,500);                        //
                Determine a new random stimLength
316           pauzeLength = random(100,500);                       //
                Determine a new random pauzeLength
317           if (value[4] == 2){                                 // In case
                of stimulation mode 2, uncorrelated audio and burst
318             burstDelay = random(0,200);                        // Set the
                  burstDelay random
319             audioDelay = random(0,200);                        // Set the
                  audioDelay random
320           }
321         } else {
322           syncCounter++;                                      // In a
                normal routine, increase the syncCoutner by 1, corresponding
                to 0.01s
323         }
324         stimulationTime++;                                    // increment
              the stimulation time
325       } else if (stimulationTime == maxStimulationTime) {     // If the
            stimulation time is exceeded
326         stimulationTime = 0;                                  // Reset the
              stimulation time
327         stimulationOff();                                     // Turn all
              stimulation off
328         lcdPrintSettings();                                   // Print the
              settings
329       }
330     }
331 }
332 /*! loopCore0 burst functions
333   * @brief Outside the for(;;){} can be seen as the arduino native setup()
        function.
```

```
334   * The first time this core runs, it initiates the burst timer at timer 2
          and runs it every milisecond.
335   * The timerAlarm is always while the device is powered.
336   * The burstOn function attaches the burstSequence() to the timer,
         setting burstPulse true.
337   * @param parameter Nessecary for assigning this function to core0
338   */
339 void loopCore0(void *parameter) {
340   // Inialize the timer for the burst function
341     burstTimer = timerBegin(2, 80, true);  // Burst timer begin
342     timerAlarmWrite(burstTimer, 1000, true); // Determine the frequency at
            which it fires every milisecond
343     timerAlarmEnable(burstTimer); // Enable the alarm such that the
           function runs
344   for(;;) {
345     if (burstPulse) {                                        // Check the
          state of burstPulse
346       // positive stimulation 5 pulses of 1 ms
347       if (burstIndex < 10) {                                 // A
            burstIndex lower than 10 indicates positive current stimulation
348         if (burstIndex == 0 && !(burstIndex%2)) {            // At the
            start of the burst the Hbridge should be activated
349         digitalWrite(StimOff, LOW);                          // if stimoff
            == low, than the hbridge is active,thus stimulation is possible
350         digitalWrite(LogicAtoB, HIGH);                       // Switch
            LogicAtoB HIGH to initiate positive current stimulation
351         digitalWrite(LogicBtoA, LOW);                        // Switch
            logicBtoA LOW to initiate positive current stimulation
352         }
353         if (!(burstIndex%2)) {                               // At every
            even burstIndex
354           DAC_Write(PulseAmplitudeArray[burstIndex/2],0);    // Write the
              value of PulseAmplitudeArray to the DAC
355         } else if (burstIndex%2) {                           // Every
            uneven burstIndex
356           DAC_Write(0, 0);                                   // Write 0 to
              the DAC
357         }
358         burstIndex++;                                        // Increase
            the burstindex for the next burstPulse to be true
359       // negative stimulation 1 chargebalancing 5ms pulse
360       } else if(burstIndex == 10 && !(burstIndex%2)) {       // At burst
            index 10, the H-bridge should be switched to negative current
            stimulation
361         digitalWrite(LogicAtoB, LOW);                        // Switch
            LogicAtoB LOW to initiate negative current stimulation
362         digitalWrite(LogicBtoA, HIGH);                       // Switch
            logicBtoA HIGH to initiate negative current stimulation
363         DAC_Write(PulseAmplitudeArray[burstIndex/2],0);      // Write the
            value of PulseAmplitudeArray to the DAC
364         burstIndex++;                                        // Increase
            the burstindex for the next burstPulse to be true
365       // End the chargebalancing pulse
366       } else if (burstIndex == 15) {                         // Stay in the
            negative chargebalancing pulse until 5 ms has passed
```

```
367         DAC_Write(0, 0);                                    // Write 0 to
                the DAC, until the next burst sequence starts
368         burstIndex++;                                       // Increase
                the burstindex for the next burstPulse to be true
369       // Switch the Hbridge
370       } else if (burstIndex == 16) {                        // Switch the
             H-bridge 1ms later to prevent a positive current spike from
             appearing
371         digitalWrite(LogicAtoB, LOW);                       // Detach the
                electrodes from the stimulation circuit, and connect them to
                ground
372         digitalWrite(LogicBtoA, LOW);                       // Detach the
                electrodes from the stimulation circuit, and connect them to
                ground
373         digitalWrite(StimOff, HIGH);                        // if stimoff
                == low, than the hbridge is active,thus stimulation is possible
374         burstIndex++;                                       // Increase
                the burstindex for the next burstPulse to be true
375       // End of burst sequence
376       } else if (burstIndex == maxBurstLength) {            // At the end
             of one burst interval, shut of the stimulation circuit again for
             safety
377         DAC_Write(0, 0);
378         digitalWrite(LogicAtoB, LOW);
379         digitalWrite(LogicBtoA, LOW);
380         digitalWrite(StimOff, HIGH);
381         burstIndex = 0;
382       } else {
383         burstIndex++;                                       // Increase
                the burstindex for the next burstPulse to be true
384       }
385       burstPulse = !burstPulse;                             // Set the
              burstPulse false again to prevent a direct iteration through this
              loop again
386     }
387
388     if (burstOn) {                                          // Check if
            the burst stimulation should be activated
389       burstOn = !burstOn;                                   // If so,
              directly set burstOn to false to prevent entering this loop again
390       burstIndex = 0;                                       // Reset the
              burst index when started again
391       timerAttachInterrupt(burstTimer, &burstSequence, true); // Attach
              burstTimer function to the timer, which sets burstPulse true
              every 1 ms
392       digitalWrite(burstLED, HIGH);                         // Activate
              the burstLED to show burst stimulation is active
393     } else if (burstOff) {                                  // Check if
            the burst stimulation should be deactivated
394       // delay(120);
395       burstOff = !burstOff;                                 // If so,
              directly set burstOff to false to prevent entering this loop
              again
396       timerDetachInterrupt(burstTimer);                     // Disable
              the alarm such that the function stops running
397       // Safety measure to shut down the stimulation circuit once more
```

```
398       DAC_Write(0, 0);                                    // Write the DAC
              to 0 so no current is running through the stimulation circuit
399       digitalWrite(LogicAtoB, LOW);                       // Detach the
              electrodes from the stimulation circuit, and connect them to
              ground
400       digitalWrite(LogicBtoA, LOW);                       // Detach the
              electrodes from the stimulation circuit, and connect them to
              ground
401       digitalWrite(StimOff, HIGH);                        // if stimoff ==
              low, than the hbridge is active,thus stimulation is possible
402       digitalWrite(burstLED, LOW);                        // Deactivate
              burstLED to indicate burst stimulation has stopped
403     }
404   vTaskDelay(1);                                          // IMPORTANT Gives
          FreeRTOS space to run their housekeeping code.
405                                                           // If not here,
                                                                 the storage
                                                                 will overload
                                                                 and the ESP32
                                                                 will reboot
                                                                 over and over
406   }
407   vTaskDelete(NULL);                                      // IMPORTANT to
          prevent the ESP32 from rebooting
408 }
409 /* ------------- Functions -------------*/
410   /*! Stimulation off
411     * @brief This function turns all stimulation off
412     */
413   void stimulationOff() {
414     timerDetachInterrupt(syncTimer);                          // Detach the
            synchronisation timer
415     digitalWrite(stimLED, LOW);                               // Turn of the
            stimLED
416     if (digitalRead(burstLED)) {                              // Check if
            burst stimulation is on
417       burstOff = true;
418     }
419     if (musicPlayer.playingMusic) {                           // Check if
            the music is playing
420       musicPlayer.stopPlaying();                              // Stop
              playing music
421       digitalWrite(audioLED,LOW);                             // Turn off
              the audioLED
422     }
423
424   }
425   /*! Create a list of audio files
426     * @brief Create a list of audio files to be used in the menu
427     */
428   void createAudioFileList() {
429     File root = SD.open("/");                       // Open the SD card folder
430     uint8_t i = 0;                                  // Index to count the amount
            of files
431     while (true) {                                  // Stay inside this loop
```

```
432       File entry =  root.openNextFile();      // Check for the next file
              on the SD card
433       if (entry && i == 10) {                 // Max 10 audio files can be
               loaded from the SD card (descision for a robust program,
              preventing exceeding the storage)
434         lcd.setCursor(0,2);                   // Error message in LCD
              screen
435         lcd.print(F("ER:Max 10 audiofiles"));
436         lcd.setCursor(0,3);
437         lcd.print(F("exceeded on SD card"));
438         while (1);                            // Stay here, a reboot is
              needed
439       } else if (! entry && i == 0) {         // If there is no audio file
             at all
440         lcd.setCursor(0,3);                   // Error message in LCD
              screen
441         lcd.print(F("ER:No audio on SD"));
442         while (1);                            // Stay here, a reboot is
              needed
443       } else if (! entry && i > 0) {          // No more files left
444         break;                                // break out of the qhile
              loop and continue
445       }
446       char* singleName = (char*)entry.name(); // Save the file name
447       if (isFnMusic(singleName)) {            // Check if the file is an
              audio file
448         strcpy(name[i], "/");                 // Copy a / to the index of
              the name array containing string literals (NO POINTER)
449         // "/" is nessecary for the music player to find the audio file on
               the SD card
450         strcat(name[i], singleName);          // Copy the file name to
              that same index of the name array
451         audioFileList[i] = (char*)name[i];    // Save a pointer to the
              string literal name in audioFileList, used for the menu
452         i++;
453       }
454       entry.close();                          // Close the file
455     }
456     root.close();                             // Close the SD card root
          folder
457     maxAudioFiles = i;                        // Count the maximum audio
          files on the card, used for the menu
458     if (value[menuPositionMax] > maxAudioFiles-1) { // Check if the
          settings loaded from the SD card by dataRead() are not exceeding
          the current amount of audio files
459       value[menuPositionMax] = 0; // adjust the audio file if the amount
            of audio files is less than before
460     }
461     maximum[menuPositionMax] = i-1;           // Set the menu maximum
          position to the amount of audio files
462   }
463   /*! Check for audio file names
464     * @brief This function checks if character arrays end on audio file
          types
465     * @param filename a file name which should be checked
466     * @return It returns true if the file is an audio file
```

```
467     */
468   bool isFnMusic(char* filename) {
469     int8_t len = strlen(filename);                    // Determine the
          length of the character array
470     if (  (strstr(strlwr(filename + (len - 4)), ".mp3") // Check the last
          four characters
471       // || strstr(strlwr(filename + (len - 4)), ".aac") //Char* copy file
            name does not work in the audioFileList array
472       || strstr(strlwr(filename + (len - 4)), ".wma")   // Check the last
            four characters
473       || strstr(strlwr(filename + (len - 4)), ".wav")   // Check the last
            four characters
474     || strstr(strlwr(filename + (len - 4)), ".ogg")     // Check the last
          four characters
475       // || strstr(strlwr(filename + (len - 4)), ".fla") //Char* copy file
            name does not work in the audioFileList array
476       )
477       && !(filename[0] == '.' || filename[0] == '_')    // Some SD cards
            add a copy of files as hidden files
478       ) {
479       return true;                                     // The file is an
            audio file
480     } else {
481       return false;                                     // The file is
            not an audio file
482     }
483   }
484   /*! Starts or stops the stimulation sequence
485     * @brief Check the state of the start stop button, and starts or stops
          the stimulation sequence
486     * @param startStopButton the startStopButton object which should be
          checked
487     */
488   void startStopButtonHandler(Button2&  startStopButton) {
489     if (!adjustSettings){                                // If no
          parameter is being edited in the menu
490       stimulationActive = !stimulationActive;            // Set the
            stimulation active
491     } else if (adjustSettings) {                         // If a
          parameter is being edited in the menu
492       lcd.setCursor(0,3);                                // Throw an
            error message
493       lcd.print(F("Confirm the setting"));
494       delay(2000);                                       // Display
            it for 2s
495       lcdPrintSettings();                                // Show the
            menu again without the error message
496     }
497
498     if (stimulationActive){                              // If the
          stimulation is started
499       syncCounter = 0;                                   // Reset the
            syncCounter, aka time increaing with 0.01s
500       stimulationTime = 0;                               // Reset the
            stimulation time to test the max stimulation time allowed
```

```
501       timerAttachInterrupt(syncTimer, &synchroniser, true);   // Attach
              syncTimer() and start setting syncPulse true every 0.01s
502       digitalWrite(stimLED, HIGH);                            // Turn on
              the stimulation LED
503       lcdPrintSettings();                                     // Print the
              menu and show a stimulation message in the LCD screen
504     } else if (!adjustSettings) {                             // If the
          stimulation is toggled to inactive, and the settings are not
          adjusted
505       stimulationOff();
506       lcdPrintSettings();                                     // Print the
              menu
507     }
508   }
509   /*! Print the menu
510     * @brief print the menu, also while stimulating
511     */
512   void lcdPrintSettings() {
513     lcd.clear();
514     checkBattery();                                         // Run a battery
          status check and print it in the screen
515     lcd.setCursor(0,1);
516     lcd.print(menu[menuPosition][0]);                       // Print the name of
          the current setting
517     lcd.setCursor(0,2);
518     if (menuPosition == menuPositionMax) {                  // Print the name of
          the audiofile if it is the last stting in the menu array
519       lcd.print(audioFileList[value[menuPosition]]);
520     } else {
521       lcd.print(value[menuPosition]);                       // Print the value
            of the current menu setting
522     }
523     lcd.print(" ");
524     lcd.print(menu[menuPosition][1]);                       // Print the unit of
          the current menu setting
525     if (stimulationActive) {                                // Settings can be
          adjusted while stimulating, show a stimulating notice when
          stimulating is active
526       lcd.setCursor(0,3);
527       lcd.print("...stimulating");
528     }
529     lcd.setCursor(17,2);                                    // Set the cursor to
          the end of the third line. In case of adjusting a parameter, this
          wil blink.
530   }
531   /*! Check the battery level
532     * @brief Check the battery level and print a notice in the LCD screen.
533     * If the battery is at 0% (±5.5V), the maximum stimulation current of
          3mA can not be guaranteed
534     * This will display an error message to switch the battery
535     */
536   void checkBattery() {
537     uint16_t batteryLevel = analogRead(batteryPin);         // Read the battery
          level from the 12-bit ADC input (max value of 4095 corresponding to
          ±9V)
```

```
538     if (lastBatteryLevel > batteryLevel) {            // Check if the
            battery level has droped compared to the previous state
539       lastBatteryLevel = batteryLevel;                // Update the last
              battery level with the current one
540     } else {                                          // If the battery
            level did not drop
541       batteryLevel = lastBatteryLevel;                // Keep the last
              battery level, preventing a constant switching of battery values
542     }
543     if (batteryLevel > 3200 ) {                       // Corresponding to
            ± 7.9v
544       lcd.setCursor(0,0);
545       lcd.print(F("b=100%"));
546     } else if (batteryLevel > 2800 ) {                // Corresponding to
            ± 7.3v
547       lcd.setCursor(0,0);
548       lcd.print(F("b=75%"));
549     } else if (batteryLevel > 2600 ) {                // Corresponding to
            ± 6.7v
550       lcd.setCursor(0,0);
551       lcd.print(F("b=50%"));
552     } else if (batteryLevel > 2400) {                 // Corresponding to
            ± 6.1v
553       lcd.setCursor(0,0);
554       lcd.print(F("b=25%"));
555     } else if (batteryLevel > 2300 ) {                // Corresponding to
            ± 5.8v
556       lcd.setCursor(0,0);
557       lcd.print(F("b=10%"));
558     } else if (batteryLevel > 2200 ) {                // Corresponding to
            ± 5.5v
559       lcd.setCursor(0,0);
560       lcd.print(F("b=0%"));
561       lcd.setCursor(12,0);
562       lcd.print(F("DualSTIM"));
563       lcd.setCursor(0,2);
564       lcd.print(F("ER:Battery level low"));
565       lcd.setCursor(0,3);
566       lcd.print(F("Change the battery"));
567       stimulationOff();
568       while(1);                                       // Stay with this
              message, battery change and reboot is needed
569     }
570     lcd.setCursor(12,0);
571     lcd.print(F("DualSTIM"));
572   }
573   /*! Rotary encoder button click function
574     * @brief The rotary encoder button is used to adjust the menu and
          settings.
575     * The click function is used to adjust, or confirm parameters in the
          menu settings.
576     * @param encoderButton the encoderButton object which should be
          checked
577     */
578   void onEncoderButtonClick(Button2& encoderButton) {
```

```
579     if (adjustSettings) {                // If the button is clicked while
            a parameter is adjusted
580       updateSettings();                  // Run updateSettings() to update
              the current value
581       lcd.noBlink();                     // Stop blinking the cursor
582     } else {
583       lcd.setCursor(17,2);               // Set the position of the cursor
              at the end of the third line
584       lcd.blink();                       // Blink the cursor to indicate a
              parameter could be changed with onEncoderRotateLeft() and
              onEncoderRotateRight()
585     }
586     adjustSettings = !adjustSettings;    // Invert the value of adjust
            settings
587   }
588   /*! Rotary encoder rotate left
589     * @brief Rotating the rotary encoder left sets the menu position down.
590     * When the menu is adjusted, the parameter value will go down by the
            defined step size
591     * @param encoderRotate the encoderRotate object which should be
            checked
592     */
593   void onEncoderRotateLeft(ESPRotary& encoderRotate) {
594     if (adjustSettings && value[menuPosition] > minimum[menuPosition]) {
          // A parameter is adjusted and the value of the parameter is bigger
            than the minimum of that parameter
595       value[menuPosition] = value[menuPosition] + -1*step[menuPosition];
            // Decrease the value of the parameter with the stepsize of the
            parameter
596     } else if (!adjustSettings) {
          // If the parameter is not adjusted
597       if (menuPosition > 0) {
            // And the menu position is bigger than 0
598         menuPosition--;
              // The menu position can be decreased by 1
599       } else {
            // If the menu position is 0
600         menuPosition = menuPositionMax;
              // The menu position is updated to the max value, the last
              parameter setting
601       }
602     }
603     lcdPrintSettings();
          // Print the menu again with the updated menu position
604   }
605   /*! Rotary encoder rotate right
606     * @brief Rotating the rotary encoder right sets the menu position up.
607     * When the menu is adjusted, the parameter value will go up by the
            defined step size
608     * @param encoderRotate the encoderRotate object which should be
            checked
609     */
610   void onEncoderRotateRight(ESPRotary& encoderRotate) {
611     if (adjustSettings && value[menuPosition] < maximum[menuPosition]) {
          // A parameter is adjusted and the value of the parameter is
          smaller than the maximum of that parameter
```

```
612       value[menuPosition] = value[menuPosition] + step[menuPosition];   //
              Increase the value of the parameter with the stepsize of the
             parameter
613     } else if (!adjustSettings) {                                        //
            If the parameter is not adjusted
614       if (menuPosition < menuPositionMax) {                              //
              And the menu position is smaller than the maximum menu position
615         menuPosition++;                                                  //
                The menu position can be increased by 1
616       } else {                                                           //
            If the menu position is the maximum menu position
617         menuPosition = 0;                                                //
                The menu position is set to 0, the first parameter in the menu
618       }
619     }
620     lcdPrintSettings();                                                  //
          Print the menu again with the updated menu position
621   }
622   /*! Update parameter settings
623     * @brief Update the settings after the user has confirmed their
          settings
624     */
625   void updateSettings() {
626     switch (menuPosition) {
627       case 0:                                                 // Change the
            audio volume
628         if (musicPlayer.playingMusic) {                       // While the
            music is playing
629           musicPlayer.stopPlaying();                          // First stop
              playing the music to prevent an error
630           musicPlayer.setVolume(maximum[0]-value[0],maximum[0]-value[0]);
              // Update the volume
631           musicPlayer.startPlayingFile(audioFile);            // Continue
              playing the file
632         } else {
633           musicPlayer.setVolume(maximum[0]-value[0],maximum[0]-value[0]);
              // Update the volume (100 is low volume, 0 is highest volume)
634         }
635         break;
636       case 1:
            //burst amplitude
637         P5_ElectricalStim = value[menuPosition]*0.001;                  //
              PulseAmplitude of 5th pulse, value between 0 and 3 mA
638         PulseAmplitudeArray = COMPUTE_PulseAmplitudeArray(
              P1_ElectricalStim, P5_ElectricalStim); // List containing the
              pulseamplitudes per ms
639         break;
640       case 2:                                                        //
            burst slope
641         P1_ElectricalStim = value[menuPosition]*0.01;                //
              PulseAmplitude of 1th Pulse, value between 0 and 1, percentage
              of P5
642         PulseAmplitudeArray = COMPUTE_PulseAmplitudeArray(
              P1_ElectricalStim, P5_ElectricalStim); // List containing the
              pulseamplitudes per ms
643         break;
```

```
644       case 3:                                                    //
              time delays
645         calculateTimeDelays();
646         break;
647       case menuPositionMax:                                      //
              Audio file
648         audioFile = audioFileList[value[menuPosition]];          //
                Update the chosen audio file name using the menu position,
               value and audioFileList which includes pointers to the names
649         break;
650       default:
651         break;
652     }
653     dataLog();                                                   //
          Save the new settings to the SD card in datalog.txt
654   }
655   /*! Datalog to the SD card
656     * @brief Datalog the parameter settings from the menu to the SD card.
657     * If there is no datalog.txt file available, the function will
          generate a new one with the default parameter settings from this
          script
658     */
659   void dataLog() {
660     String dataString;                      // Create a string to write
          the settings to
661     for(byte i = 0; i < menuLength; i++) {  // Loop through the menu
662       if (i == 0) {
663         dataString += String(maxStimulationTimeSD); // add stimulation
              time as last variable in datalog.txt
664         dataString += ",";
665         float burstFrequency = 0; // Uodate the maxStimulationTime
666         burstFrequency = 1/(0.001*maxBurstLength);
667         dataString += String(burstFrequency); // add stimulation time as
              last variable in datalog.txt
668         dataString += ",";
669       }
670       dataString += String(value[i]);       // Add the value to the string
671       dataString += ",";                    // Separate by a comma
672       dataString += String(step[i]);        // Add the step to the string
673       dataString += ",";                    // Separate by a comma
674       dataString += String(minimum[i]);     // Add the minimum to the
              string
675       dataString += ",";                    // Separate by a comma
676       dataString += String(maximum[i]);     // Add the maximum to the
              string
677       if (i < menuPositionMax) {            // At the last one skip the
              comma
678         dataString += ",";                  // Separate by a comma
679       }
680     }
681     // open the file. note that only one file can be open at a time,
682     // so you have to close this one before opening another.
683     dataFile = SD.open("/datalog.txt", FILE_APPEND);        // open the
          file to append text
684
685     // if the file is available, write to it:
```

```
686     if (dataFile) {
687       dataFile.println(dataString);                          // Print the
              data to the datalog.txt file
688       dataFile.close();                                      // Close the
              file
689     }
690     // if the file isn't open, pop up an error:
691     else {
692       lcd.setCursor(0,3);
693       lcd.println(F("ER:Datalog not open"));                 // Throw an
              error if the file can not be opened
694     }
695   }
696   /*! readme.txt to the SD card
697     * @brief Create a readme file if it does not exist
698     */
699   void addReadmeToSD() {
700     String dataString;                        // Create a string to write
          the settings to
701     dataString = "The DualSTIM is used to provide auditive and electrical
          stimulation simultaneously\n";
702     dataString += "Instructions are only for clinicians, not for patients
          !\n";
703     dataString += "At first use, add a SD card (FAT16/32 format) to the
          device\n";
704     dataString += "Add audio files to the SD card for audio stimulation\n"
          ;
705     dataString += "Audiofile names should be max 8 characters, followed by
           the file extension .mp3/.wav/.ogg/.wma\n";
706     dataString += "For example: 'track001.mp3'\n";
707     dataString += "A datalog.txt file will be written to the card with a
          list of default values\n";
708     dataString += "Every time a parameter is changed in the menu, a new
          line of settings is written to datalog.txt\n";
709     dataString += "These values can be edited on the card, corresponding
          to the following order\n";
710     dataString += "- Max stimulation time in minutes\n";
711     dataString += "- Burst Frequency in Hz, ranging from 0.1 to 40 Hz.
          IMPORTANT: Use a dot as decimal seperator, e.g. 0.1\n";
712     dataString += "- Volume value\n";
713     dataString += "- Volume stepsize\n";
714     dataString += "- Volume minimum\n";
715     dataString += "- Volume maximum (Does not higher the volume but
          creates a bigger range (softer audio possible)\n";
716     dataString += "- Burst amplitude P5 value\n";
717     dataString += "- Burst amplitude P5 stepsize\n";
718     dataString += "- Burst amplitude P5 minimum\n";
719     dataString += "- Burst amplitude P5 maximum (MAX 3000 for safety!)\n";
720     dataString += "- Burst amplitude P1 value\n";
721     dataString += "- Burst amplitude P1 stepsize\n";
722     dataString += "- Burst amplitude P1 minimum\n";
723     dataString += "- Burst amplitude P1 maximum\n";
724     dataString += "- Time delay E-A value\n";
725     dataString += "- Time delay E-A stepsize\n";
726     dataString += "- Time delay E-A minimum\n";
727     dataString += "- Time delay E-A maximum\n";
```

```
728     dataString += "- Stimulation mode value (Don't edit)\n";
729     dataString += "- Stimulation mode stepsize (Don't edit)\n";
730     dataString += "- Stimulation mode minimum (Don't edit)\n";
731     dataString += "- Stimulation mode maximum (Don't edit)\n";
732     dataString += "- Audio file value\n";
733     dataString += "- Audio file stepsize (Don't edit)\n";
734     dataString += "- Audio file minimum (Will be overwritten)\n";
735     dataString += "- Audio file maximum (Will be overwritten)\n";
736     dataString += "More detailed information can be found on https://bit.
            ly/MultiModalStimulator";
737
738     // open the file. note that only one file can be open at a time,
739     // so you have to close this one before opening another.
740     dataFile = SD.open("/readme.txt", FILE_WRITE);        // open the file
             to append text
741
742     if (dataFile) {                                          // if the
            file is available, write to it:
743       dataFile.println(dataString);                          // Print the
              data to the datalog.txt file
744       dataFile.close();                                      // Close the
              file
745     } else {                                                 // if the
            file isn't open, pop up an error:
746       lcd.setCursor(0,3);
747       lcd.println(F("ER:readme.txt not open"));              // Throw an
            error if the file can not be opened
748     }
749   }
750   /*! Read information from the SD card
751     * @brief If the datalog.txt file exists, the menu parameters are
            updated.
752     * If the file does not exists, a new file with the default settings
            from the script is created
753     */
754   void dataRead() {
755     dataFile = SD.open("/datalog.txt", FILE_READ);          // Open the
            datalog file
756     if (!dataFile) {                                         // If the
            datafile does not exist
757       dataFile.close();
758       dataLog();                                             // Create a
            new file from the default settings
759       dataFile = SD.open("/datalog.txt", FILE_READ);        // Open the
            file again because it was closed
760     }
761     if (dataFile) {                                          // If the file
             opens
762       uint8_t maxLines = 0;
763       while (dataFile.available()) {
764         if (dataFile.readStringUntil('\n')) {                // Count the
              amount of lines there are in the datalog file
765           maxLines++;
766         }
767       }
```

```
768       dataFile.close();
                                                                  // Close
          the file
769       dataFile = SD.open("/datalog.txt", FILE_READ);
                                    // Reopen the file to read the last line
770       uint8_t countLines = 0;
                                                          // Count the
          lines
771       if (dataFile) {
                                                                  // If the
           file is open
772         bool lastLine = false;
                                                           // Be aware of
            the last line
773         uint8_t i = 0;
                                                                   // Index
             used to loop through the menu parameters
774         while (dataFile.available()) {
                                                      // While there is new
            information in datalog.txt
775           if (lastLine) {
                                                                  // Only
              run this part in the last line
776             if (i == 0) {
777               maxStimulationTimeSD = (uint32_t)dataFile.readStringUntil(',
                    ').toInt(); // Uodate the maxStimulationTime
778               float burstFrequency = dataFile.readStringUntil(',').toFloat
                    (); // Uodate the maxStimulationTime
779               maxBurstLength = (uint16_t)((1/burstFrequency)*1000);
780               if (maxBurstLength < 25 || maxBurstLength > 10000) {
781                 lcd.setCursor(0,2);
782                 lcd.print(F("ER:SD maxBurstLength"));
783                 lcd.setCursor(0,3);
784                 lcd.print(F("outOfRange 0.1-40Hz"));
785                 while(1);
786               }
787             }
788             value[i] = (uint16_t)dataFile.readStringUntil(',').toInt();
                          // update the value to the menu
789             step[i] = (uint16_t)dataFile.readStringUntil(',').toInt();
                           // update the step to the menu
790             minimum[i] = (uint16_t)dataFile.readStringUntil(',').toInt();
                         // update the step to the menu
791             if (i < menuPositionMax) {
792               maximum[i] = (uint16_t)dataFile.readStringUntil(',').toInt()
                    ;   // update the maximum to the menu
793               i++;
794             } else {
795               String x = dataFile.readStringUntil('\n');
796               if (x.indexOf(',') != -1) {
797                 lcd.setCursor(0,1);
798                 lcd.print("ER:SD datalog exceed");
799                 lcd.setCursor(0,2);
800                 lcd.print("max params. Use . as");
801                 lcd.setCursor(0,3);
802                 lcd.print("decimal point eg 0.9");
```

```
803                 while(1);
804               } else {
805                 maximum[i] = (uint16_t)x.toInt();  // update the maximum
                        from the end of the line to the menu
806               }
807               break;
808             }
809           } else if (dataFile.readStringUntil('\n')) {
                                        // Count the amount of lines in
                 datalog.txt
810             if (countLines == maxLines-2) {
                                                    // At the second to last
                   line (\n indicating the end of the line)
811               lastLine = true;
                                                                  // flag
                      that we are at the last line
812             } else {
813               countLines++;
                                                                     //
                      Increment the countLines by 1
814             }
815           }
816         }
817         dataFile.close();
                                                                 // Close
              the dataline
818         if (maximum[1] > 3000) {
                                                              // Make sure the
              maximum amplitude is not exceeding 3000 mA for safety
819           lcd.setCursor(0,2);
                                                                 // Else throw
                 an error message
820           lcd.print(F("ER:SD max amplitude"));
821           lcd.setCursor(0,3);
822           lcd.print(F("change to max 3000"));
823           stimulationOff();
824           while(1);
                                                                       //
                 Stay here
825         }
826       }
827     }
828   }
829   /*! Calculate the time delays for electrical and auditory stimulation
830     * @brief Calculate the time delays from the menu parameters,
          considering a positive or negative value
831     * If the timedelay is positive, the audio is delayed, thus E leads A.
832     * If the timedelay is negative, the burst is delayed, thus A leads E.
833     * The burstIsDelayed bool is updated acording to one of the two cases,
          nessecary to determine the new stimulation length and pauze is
          calculated.
834     * In case of stimulation mode 2, the burst and audio is uncorrelated,
          thus the delayes are calculated random.
835     */
836   void calculateTimeDelays(){
```

```
837     if (!(value[4] == 2)) {             // Stimulation mode is not 2 (
            uncorrelated)
838       timeDelay = value[3]/10;          // Determine the time delay per
              0.01 second (syncTimer)
839       if (timeDelay > 0) {              // If the time delay is positive
840         burstDelay = burstDelayOffset;
841         audioDelay = timeDelay;         // E leads A
842         burstIsDelayed = 0;             // StimLength and pauzeLength
              updates at the end of the audio
843       } else if (timeDelay < 0) {       // If the time delay is negative
844         burstDelay = abs(timeDelay+burstDelayOffset);    // A leads E
845         audioDelay = 0;
846         burstIsDelayed = 1;             // StimLength and pauzeLength
              updates at the end of the burst
847       } else {                          // Else no delays
848         burstDelay = burstDelayOffset;
849         audioDelay = 0;
850         burstIsDelayed = 0;
851       }
852     } else if (value[4] == 2) {                             // In case of
          stimulation mode 2, uncorrelated audio and burst
853       burstDelay = random(0,200);                         // Set the
            burstDelay random
854       audioDelay = random(0,200);                         // Set the
            audioDelay random
855     }
856   }
857   /* WRITING PROTOCOL DAC*/
858     /* Writing protocol for the DAC5601 -> datasheet: https://www.analog.
          com/media/en/technical-documentation/data-sheets/AD5601_5611_5621.
          pdf
859     * communicates in 16 bit: | PD1 | PD0 | D11 | D10 | D9 | D8 | D7 | D6
          | D5 | D4 | D3 | D2 | D1 | D0 | x | x |
860     * D11 is Most Significant Bit  (MSB) and D0 is Least Significant Bit (
          LSB), thus MSB first
861     * PD1 and PD0 are for the POWER MODES
862         * PD1 =  0 | PD0 = 0, Normal Operation
863         * PD1 =  0 | PD0 = 1, 1kohm to GND
864         * PD1 =  1 | PD0 = 0, 100kohm to GND
865         * PD1 =  1 | PD0 = 1, THREE-STATE
866     * With this function you should be able to send any input value (D
          AC_VALUE) (between 0 and 3) to the DAC through SPI.
867     * Written by Jonathan Kneepkens
868     */
869   void DAC_Write(uint16_t DAC_VALUE, uint8_t DAC_MODE) {
870     uint16_t DAC_TRANSFER = DAC_MODE << 14 | DAC_VALUE << 2;  // makes 16
          bit value with DAC mode spanning the first two bits, then the
          potentiovalue the 12 bits thereafter and the last two bits are 0
871     // SPI.begin(DAC_SCK, DAC_MISO, DAC_MOSI, DAC_CS);
872     burstHspi->beginTransaction(SPISettings(8000000, MSBFIRST, SPI_MODE1))
          ;  // check MHz of SPI_MODE
873     digitalWrite(burstHspi->pinSS(), LOW);
                                              // select DAC -> chip select LOW
874     burstHspi->transfer16(DAC_TRANSFER);
875     digitalWrite(burstHspi->pinSS(), HIGH);  // unselect DAC -> chip
          select HIGH
```

```
876     burstHspi->endTransaction();
877   }
878   /* CALCULATE BURST FIRE VALUES */
879     /**
880     * Creates an array with six values, representing the six pulse
            amplitudes. These can be used independently of the PulseWidth and
            PulseDelay.
881     * Binary Values: All pulse amplitudes are represented as a 12 bit
            value (4 MSB can be discarded).
882     * 1st to 5th Value: Normal pulses of increasing amplitudes
883     * 6th Value: Chargebalancing pulse represented as the sixth pulse and
            is a positive value
884     * Written by Jonathan Kneepkens
885     */
886   uint16_t* COMPUTE_PulseAmplitudeArray(float P1, float P5) {
887     static uint16_t PulseAmplitudeArray[6];
888     for (int i = 0; i < 5; i++) {
889       PulseAmplitudeArray[i] = (((P1+(1-P1)*(static_cast<float>(i)/4))*P5
                / 3) * 4096) -1;
890     }
891     PulseAmplitudeArray[5] = (((P5+P5*P1)/2     / 3) * 4096) -1;
892     return PulseAmplitudeArray;
893   }
```

D.2. Adafruit_VS1053_Library_ESP32/Adafruit_VS1053_ESP32.h

```
1  /*!
2   * @file Adafruit_VS1053_ESP32.h
3   */
4
5  #ifndef Adafruit_VS1053_ESP32_H
6  #define Adafruit_VS1053_ESP32_H
7
8  #if (ARDUINO >= 100)
9  #include <Arduino.h>
10 #else
11 #include <WProgram.h>
12 #include <pins_arduino.h>
13 #endif
14
15 #if !defined(ARDUINO_STM32_FEATHER)
16 #include "pins_arduino.h"
17 #include "wiring_private.h"
18 #endif
19
20 #include <Adafruit_SPIDevice.h>
21
22 #if defined(PREFER_SDFAT_LIBRARY)
23 #include <SdFat.h>
24 extern SdFat SD;
25 #else
26 #include <SD.h>
27 #endif
28
29 // define here the size of a register!
30 #if defined(ARDUINO_STM32_FEATHER)
31 typedef volatile uint32 RwReg;
32 typedef uint32_t PortMask;
33 #elif defined(ARDUINO_ARCH_AVR)
34 typedef volatile uint8_t RwReg;
35 typedef uint8_t PortMask;
36 #elif defined(__arm__)
37 #if defined(TEENSYDUINO)
38 typedef volatile uint8_t RwReg;
39 typedef uint8_t PortMask;
40 #else
41 typedef volatile uint32_t RwReg;
42 typedef uint32_t PortMask;
43 #endif
44 #elif defined(ESP8266) || defined(ESP32)
45 typedef volatile uint32_t RwReg;
46 typedef uint32_t PortMask;
47 #elif defined(__ARDUINO_ARC__)
48 typedef volatile uint32_t RwReg;
49 typedef uint32_t PortMask;
50 #else
51 typedef volatile uint8_t RwReg; //!< 1-byte read-write register
52 typedef uint8_t PortMask; //!< Type definition for a bitmask that is used
       to
```

```
53                          //!< specify the bit width
54 #endif
55
56 typedef volatile RwReg PortReg; //!< Type definition/alias used to specify
       the
57                                  //!< port register that a pin is in
58
59 #define VS1053_FILEPLAYER_TIMER0_INT
                                                  \
60   255 //!< Allows useInterrupt to accept pins 0 to 254
61 #define VS1053_FILEPLAYER_PIN_INT
                                                    \
62   5 //!< Allows useInterrupt to accept pins 0 to 4
63
64 #define VS1053_SCI_READ 0x03  //!< Serial read address
65 #define VS1053_SCI_WRITE 0x02 //!< Serial write address
66
67 #define VS1053_REG_MODE 0x00       //!< Mode control
68 #define VS1053_REG_STATUS 0x01     //!< Status of VS1053b
69 #define VS1053_REG_BASS 0x02       //!< Built-in bass/treble control
70 #define VS1053_REG_CLOCKF 0x03     //!< Clock frequency + multiplier
71 #define VS1053_REG_DECODETIME 0x04 //!< Decode time in seconds
72 #define VS1053_REG_AUDATA 0x05     //!< Misc. audio data
73 #define VS1053_REG_WRAM 0x06       //!< RAM write/read
74 #define VS1053_REG_WRAMADDR 0x07   //!< Base address for RAM write/read
75 #define VS1053_REG_HDAT0 0x08      //!< Stream header data 0
76 #define VS1053_REG_HDAT1 0x09      //!< Stream header data 1
77 #define VS1053_REG_VOLUME 0x0B     //!< Volume control
78
79 #define VS1053_GPIO_DDR 0xC017   //!< Direction
80 #define VS1053_GPIO_IDATA 0xC018 //!< Values read from pins
81 #define VS1053_GPIO_ODATA 0xC019 //!< Values set to the pins
82
83 #define VS1053_INT_ENABLE 0xC01A //!< Interrupt enable
84
85 #define VS1053_MODE_SM_DIFF
                                                        \
86   0x0001 //!< Differential, 0: normal in-phase audio, 1: left channel
       inverted
87 #define VS1053_MODE_SM_LAYER12 0x0002  //!< Allow MPEG layers I & II
88 #define VS1053_MODE_SM_RESET 0x0004    //!< Soft reset
89 #define VS1053_MODE_SM_CANCEL 0x0008   //!< Cancel decoding current file
90 #define VS1053_MODE_SM_EARSPKLO 0x0010 //!< EarSpeaker low setting
91 #define VS1053_MODE_SM_TESTS 0x0020    //!< Allow SDI tests
92 #define VS1053_MODE_SM_STREAM 0x0040   //!< Stream mode
93 #define VS1053_MODE_SM_SDINEW 0x0800   //!< VS1002 native SPI modes
94 #define VS1053_MODE_SM_ADPCM 0x1000    //!< PCM/ADPCM recording active
95 #define VS1053_MODE_SM_LINE1 0x4000 //!< MIC/LINE1 selector, 0: MICP, 1:
     LINE1
96 #define VS1053_MODE_SM_CLKRANGE
                                                      \
97   0x8000 //!< Input clock range, 0: 12..13 MHz, 1: 24..26 MHz
98
99 #define VS1053_SCI_AIADDR
                                                          \
```

```
100   0x0A //!< Indicates the start address of the application code written
          earlier
101         //!< with SCI_WRAMADDR and SCI_WRAM registers.
102 #define VS1053_SCI_AICTRL0
                                                                        \
103   0x0C //!< SCI_AICTRL register 0. Used to access the user's application
          program
104 #define VS1053_SCI_AICTRL1
                                                                        \
105   0x0D //!< SCI_AICTRL register 1. Used to access the user's application
          program
106 #define VS1053_SCI_AICTRL2
                                                                        \
107   0x0E //!< SCI_AICTRL register 2. Used to access the user's application
          program
108 #define VS1053_SCI_AICTRL3
                                                                        \
109   0x0F //!< SCI_AICTRL register 3. Used to access the user's application
          program
110 #define VS1053_SCI_WRAM 0x06     //!< RAM write/read
111 #define VS1053_SCI_WRAMADDR 0x07 //!< Base address for RAM write/read
112
113 #define VS1053_PARA_PLAYSPEED 0x1E04 //!< 0,1 = normal speed, 2 = 2x, 3 =
       3x etc
114
115 #define VS1053_DATABUFFERLEN 32 //!< Length of the data buffer
116
117 /*!
118  * Driver for the Adafruit VS1053
119  */
120 class Adafruit_VS1053_ESP32 {
121 public:
122   /*!
123    * @brief Software SPI constructor - must specify all pins
124    * @param mosi MOSI (Microcontroller Out Serial In) pin
125    * @param miso MISO (Microcontroller In Serial Out) pin
126    * @param clk Clock pin
127    * @param rst Reset pin
128    * @param cs SCI Chip Select pin
129    * @param dcs SDI Chip Select pin
130    * @param dreq Data Request pin
131    */
132   Adafruit_VS1053_ESP32(int8_t mosi, int8_t miso, int8_t clk, int8_t rst,
          int8_t cs,
133                         int8_t dcs, int8_t dreq);
134   /*!
135    * @brief Hardware SPI constructor - assumes hardware SPI pins
136    * @param rst Reset pin
137    * @param cs SCI Chip Select pin
138    * @param dcs SDI Chip Select pin
139    * @param dreq Data Request pin
140    */
141   Adafruit_VS1053_ESP32(int8_t rst, int8_t cs, int8_t dcs, int8_t dreq);
142   /*!
143    * @brief Initialize communication and (hard) reset the chip.
144    * @return Returns true if a VS1053 is found
```

```
145     */
146   uint8_t begin(void);
147   /*!
148    * @brief Performs a hard reset of the chip
149    */
150   void reset(void);
151   /*!
152    * @brief Attempts a soft reset of the chip
153    */
154   void softReset(void);
155   /*!
156    * @brief Reads from the specified register on the chip
157    * @param addr Register address to read from
158    * @return Retuns the 16-bit data corresponding to the received address
159    */
160   uint16_t sciRead(uint8_t addr);
161   /*!
162    * @brief Writes to the specified register on the chip
163    * @param addr Register address to write to
164    * @param data Data to write
165    */
166   void sciWrite(uint8_t addr, uint16_t data);
167   /*!
168    * @brief Generate a sine-wave test signal
169    * @param n Defines the sine test to use
170    * @param ms Delay (in ms)
171    */
172   void sineTest(uint8_t n, uint16_t ms);
173   /*!
174    * @brief Reads the DECODETIME register from the chip
175    * @return Returns the decode time as an unsigned 16-bit integer
176    */
177   uint16_t decodeTime(void);
178   /*!
179    * @brief Set the output volume for the chip
180    * @param left Desired left channel volume
181    * @param right Desired right channel volume
182    */
183   void setVolume(uint8_t left, uint8_t right);
184   /*!
185    * @brief Prints the contents of the MODE, STATUS, CLOCKF and VOLUME
           registers
186    */
187   void dumpRegs(void);
188
189   /*!
190    * @brief Decode and play the contents of the supplied buffer
191    * @param buffer Buffer to decode and play
192    * @param buffsiz Size to decode and play
193    */
194   void playData(uint8_t *buffer, uint8_t buffsiz);
195   /*!
196    * @brief Test if ready for more data
197    * @return Returns true if it is ready for data
198    */
199   boolean readyForData(void);
```

```
200   /*!
201    * @brief Apply a code patch
202    * @param patch Patch to apply
203    * @param patchsize Patch size
204    */
205   void applyPatch(const uint16_t *patch, uint16_t patchsize);
206   /*!
207    * @brief Load the specified plug-in
208    * @param fn Plug-in to load
209    * @return Either returns 0xFFFF if there is an error, or the address of
           the
210    * plugin that was loaded
211    */
212   uint16_t loadPlugin(char *fn);
213
214   /*!
215    * @brief Write to a GPIO pin
216    * @param i GPIO pin to write to
217    * @param val Value to write
218    */
219   void GPIO_digitalWrite(uint8_t i, uint8_t val);
220   /*!
221    * @brief Write to all 8 GPIO pins at once
222    * @param i Value to write
223    */
224   void GPIO_digitalWrite(uint8_t i);
225   /*!
226    * @brief Read all 8 GPIO pins at once
227    * @return Returns a 2 byte value with the reads from the 8 pins
228    */
229   uint16_t GPIO_digitalRead(void);
230   /*!
231    * @brief Read a single GPIO pin
232    * @param i pin to read
233    * @return Returns the state of the specified GPIO pin
234    */
235   boolean GPIO_digitalRead(uint8_t i);
236   /*!
237    * @brief Set the Pin Mode (INPUT/OUTPUT) for a GPIO pin.
238    * @param i Pin to set the mode for
239    * @param dir Mode to set
240    */
241   void GPIO_pinMode(uint8_t i, uint8_t dir);
242
243   /*!
244    * @brief Initialize chip for OGG recording
245    * @param plugin Binary file of the plugin to use
246    * @return Returns true if the device is ready to record
247    */
248   boolean prepareRecordOgg(char *plugin);
249   /*!
250    * @brief Start recording
251    * @param mic mic=true for microphone input
252    */
253   void startRecordOgg(boolean mic);
254   /*!
```

```
255     * @brief Stop the recording
256     */
257   void stopRecordOgg(void);
258   /*!
259     * @brief Returns the number of words recorded
260     * @return 2-byte unsigned int with the number of words
261     */
262   uint16_t recordedWordsWaiting(void);
263   /*!
264     * @brief Reads the next word from the buffer of recorded words
265     * @return Returns the 16-bit data corresponding to the received address
266     */
267   uint16_t recordedReadWord(void);
268
269   uint8_t mp3buffer[VS1053_DATABUFFERLEN]; //!< mp3 buffer that gets sent
        to the
270                                            //!< device
271
272 #ifdef ARDUINO_ARCH_SAMD
273 protected:
274   uint32_t _dreq;
275
276 private:
277   Adafruit_SPIDevice *spi_dev_ctrl = NULL; ///< Pointer to SPI dev for
        control
278   Adafruit_SPIDevice *spi_dev_data = NULL; ///< Pointer to SPI dev for
        data
279   int32_t _mosi, _miso, _clk, _reset, _cs, _dcs;
280   boolean useHardwareSPI;
281 #else
282 protected:
283   uint8_t _dreq; //!< Data request pin
284 private:
285   Adafruit_SPIDevice *spi_dev_ctrl = NULL; ///< Pointer to SPI dev for
        control
286   Adafruit_SPIDevice *spi_dev_data = NULL; ///< Pointer to SPI dev for
        data
287   int8_t _mosi, _miso, _clk, _reset, _cs, _dcs;
288   boolean useHardwareSPI;
289 #endif
290 };
291
292 /*!
293  * @brief File player for the Adafruit VS1053
294  */
295 class Adafruit_VS1053_ESP32_FilePlayer : public Adafruit_VS1053_ESP32 {
296 public:
297   /*!
298     * @brief Software SPI constructor. Uses Software SPI, so you must
          specify all
299     * SPI pins
300     * @param mosi MOSI (Microcontroller Out Serial In) pin
301     * @param miso MISO (Microcontroller In Serial Out) pin
302     * @param clk Clock pin
303     * @param rst Reset pin
304     * @param cs SCI Chip Select pin
```

```
305     * @param dcs SDI Chip Select pin
306     * @param dreq Data Request pin
307     * @param cardCS CS pin for the SD card on the SPI bus
308     */
309    Adafruit_VS1053_ESP32_FilePlayer(int8_t mosi, int8_t miso, int8_t clk,
           int8_t rst,
310                                     int8_t cs, int8_t dcs, int8_t dreq, int8_t
                                           cardCS);
311    /*!
312     * @brief Hardware SPI constructor. Uses Hardware SPI and assumes the
            default
313     * SPI pins
314     * @param rst Reset pin
315     * @param cs SCI Chip Select pin
316     * @param dcs SDI Chip Select pin
317     * @param dreq Data Request pin
318     * @param cardCS CS pin for the SD card on the SPI bus
319     */
320    Adafruit_VS1053_ESP32_FilePlayer(int8_t rst, int8_t cs, int8_t dcs,
           int8_t dreq,
321                                     int8_t cardCS);
322
323    /*!
324     * @brief Hardware SPI constructor. Uses Hardware SPI and assumes the
            default
325     * SPI pins
326     * @param cs SCI Chip Select pin
327     * @param dcs SDI Chip Select pin
328     * @param dreq Data Request pin
329     * @param cardCS CS pin for the SD card on the SPI bus
330     */
331    Adafruit_VS1053_ESP32_FilePlayer(int8_t cs, int8_t dcs, int8_t dreq,
           int8_t cardCS);
332
333    /*!
334     * @brief Initialize communication and reset the chip.
335     * @return Returns true if a VS1053 is found
336     */
337    boolean begin(void);
338    /*!
339     * @brief Specifies the argument to use for interrupt-driven playback
340     * @param type interrupt to use. Valid arguments are
341     * VS1053_FILEPLAYER_TIMER0_INT and VS1053_FILEPLAYER_PIN_INT
342     * @return Returs true/false for success/failure
343     */
344    boolean useInterrupt(uint8_t type);
345    File currentTrack;              //!< File that is currently playing
346    volatile boolean playingMusic; //!< Whether or not music is playing
347    /*!
348     * @brief Feeds the buffer. Reads mp3 file data from the SD card and
            file and
349     * puts it into the buffer that the decoder reads from to play a file
350     */
351    void feedBuffer(void);
352    /*!
353     * @brief Checks if the inputted filename is an mp3
```

```
354     * @param fileName File to check
355     * @return Returns true or false
356     */
357    static boolean isMP3File(const char *fileName);
358    /*!
359     * @brief Checks for an ID3 tag at the beginning of the file.
360     * @param mp3 File to read
361     * @return returns the seek position within the file where the mp3 data
            starts
362     */
363    unsigned long mp3_ID3Jumper(File mp3);
364    /*!
365     * @brief Begin playing the specified file from the SD card using
366     * interrupt-drive playback.
367     * @param *trackname File to play
368     * @return Returns true when file starts playing
369     */
370    boolean startPlayingFile(const char *trackname);
371    /*!
372     * @brief Play the complete file. This function will not return until
            the
373     * playback is complete
374     * @param *trackname File to play
375     * @return Returns true when file starts playing
376     */
377    boolean playFullFile(const char *trackname);
378    void stopPlaying(void); //!< Stop playback
379    /*!
380     * @brief If playback is paused
381     * @return Returns true if playback is paused
382     */
383    boolean paused(void);
384    /*!
385     * @brief If playback is stopped
386     * @return Returns true if playback is stopped
387     */
388    boolean stopped(void);
389    /*!
390     * @brief Pause playback
391     * @param pause whether or not to pause playback
392     */
393    void pausePlaying(boolean pause);
394    /*!
395     * @brief Set state for playback looping
396     * @param loopState Sets playback loop state (Note: Only use with
397     * startPlayingFile, if used with playFullFile no subsequent code in the
398     * sketch executes)
399     */
400    void playbackLoop(boolean loopState);
401    /*!
402     * @brief Retrieve playback loop state
403     * @return Returns true when looped playback is enabled
404     */
405    boolean playbackLooped();
406    /*!
407     * @brief Determine current playback speed
```

```
408     * @return Returns playback speed, i.e. 1 for 1x, 2 for 2x, 3 for 3x
409     */
410   uint16_t getPlaySpeed();
411   /*!
412     * @brief Set playback speed
413     * @param speed Set playback speed, i.e. 1 for 1x, 2 for 2x, 3 for 3x
414     */
415   void setPlaySpeed(uint16_t speed);
416
417 private:
418   void feedBuffer_noLock(void);
419
420   uint8_t _cardCS;
421 };
422
423 #endif // Adafruit_VS1053_ESP32_H
```

D.3. Adafruit_VS1053_Library_ESP32/Adafruit_VS1053_ESP32.cpp

```
1  /*!
2   * @file Adafruit_VS1053_ESP32.cpp
3   *
4   * @mainpage Adafruit VS1053 Library
5   *
6   * @section intro_sec Introduction
7   *
8   * This is a library for the Adafruit VS1053 Codec Breakout
9   *
10  * Designed specifically to work with the Adafruit VS1053 Codec Breakout
11  * ----> https://www.adafruit.com/products/1381
12  *
13  * Adafruit invests time and resources providing this open source code,
14  * please support Adafruit and open-source hardware by purchasing
15  * products from Adafruit!
16  *
17  * @section author Author
18  *
19  * Written by Limor Fried/Ladyada for Adafruit Industries.
20  *
21  * @section license License
22  *
23  * BSD license, all text above must be included in any redistribution
24  */
25
26 // #include <Adafruit_VS1053_ESP32.h>
27
28 #if defined(ARDUINO_STM32_FEATHER)
29 #define digitalPinToInterrupt(x) x
30 #endif
31
32 static Adafruit_VS1053_ESP32_FilePlayer *myself;
33
34 #ifndef _BV
35 #define _BV(x) (1 << (x)) //!< Macro that returns the "value" of a bit
36 #endif
37
38 #if defined(ARDUINO_ARCH_AVR)
39 SIGNAL(TIMER0_COMPA_vect) { myself->feedBuffer(); }
40 #endif
41
42 volatile boolean feedBufferLock = false;
43 boolean _loopPlayback; //!< internal variable, used to control playback
       looping
44
45 #ifdef ESP32
46 SemaphoreHandle_t bufferSemaphore;
47
48 static void feedTask(void *param) {
49         while (true) {
50                 xSemaphoreTake(bufferSemaphore, portMAX_DELAY);
51                 myself->feedBuffer();
52         }
```

```
53 }
54
55 static void IRAM_ATTR feeder(void) {
56         xSemaphoreGiveFromISR(bufferSemaphore, NULL);
57 }
58 #elif defined(ESP8266)
59 ICACHE_RAM_ATTR
60 #else
61 static void feeder(void) {
62   myself->feedBuffer();
63 }
64 #endif
65
66
67 boolean Adafruit_VS1053_ESP32_FilePlayer::useInterrupt(uint8_t type) {
68   myself = this; // oy vey
69
70   if (type == VS1053_FILEPLAYER_TIMER0_INT) {
71 #if defined(ARDUINO_ARCH_AVR)
72     OCR0A = 0xAF;
73     TIMSK0 |= _BV(OCIE0A);
74     return true;
75 #elif defined(__arm__) && defined(CORE_TEENSY)
76     IntervalTimer *t = new IntervalTimer();
77     return (t && t->begin(feeder, 1024)) ? true : false;
78 #elif defined(ARDUINO_STM32_FEATHER)
79     HardwareTimer timer(3);
80     // Pause the timer while we're configuring it
81     timer.pause();
82
83     // Set up period
84     timer.setPeriod(25000); // in microseconds
85
86     // Set up an interrupt on channel 1
87     timer.setChannel1Mode(TIMER_OUTPUT_COMPARE);
88     timer.setCompare(TIMER_CH1, 1); // Interrupt 1 count after each update
89     timer.attachCompare1Interrupt(feeder);
90
91     // Refresh the timer's count, prescale, and overflow
92     timer.refresh();
93
94     // Start the timer counting
95     timer.resume();
96
97 #else
98     return false;
99 #endif
100   }
101   if (type == VS1053_FILEPLAYER_PIN_INT) {
102 #if defined(ESP32)
103         bufferSemaphore = xSemaphoreCreateBinary();
104         xTaskCreatePinnedToCore(feedTask, "VS1053Feeder", 1536, NULL, 10,
              NULL, 1); // Arduino main loop is on core 1, so process on the
              'other' core
105 #endif
106
```

```
107     int8_t irq = digitalPinToInterrupt(_dreq);
108     // Serial.print("Using IRQ "); Serial.println(irq);
109     if (irq == -1)
110       return false;
111 #if defined(SPI_HAS_TRANSACTION) && !defined(ESP8266) && !defined(ESP32)
        &&    \
112     !defined(ARDUINO_STM32_FEATHER)
113     SPI.usingInterrupt(irq);
114 #endif
115     attachInterrupt(irq, feeder, CHANGE);
116     return true;
117   }
118   return false;
119 }
120
121 Adafruit_VS1053_ESP32_FilePlayer::Adafruit_VS1053_ESP32_FilePlayer(int8_t
        rst, int8_t cs,
122                                                                   int8_t dcs, int8_t
                                                                      dreq,
123                                                                   int8_t cardcs)
124     : Adafruit_VS1053_ESP32(rst, cs, dcs, dreq) {
125
126   playingMusic = false;
127   _cardCS = cardcs;
128   _loopPlayback = false;
129 }
130
131 Adafruit_VS1053_ESP32_FilePlayer::Adafruit_VS1053_ESP32_FilePlayer(int8_t
        cs, int8_t dcs,
132                                                                   int8_t dreq,
133                                                                   int8_t cardcs)
134     : Adafruit_VS1053_ESP32(-1, cs, dcs, dreq) {
135
136   playingMusic = false;
137   _cardCS = cardcs;
138   _loopPlayback = false;
139 }
140
141 Adafruit_VS1053_ESP32_FilePlayer::Adafruit_VS1053_ESP32_FilePlayer(int8_t
        mosi, int8_t miso,
142                                                                   int8_t clk, int8_t
                                                                      rst,
143                                                                   int8_t cs, int8_t
                                                                      dcs,
144                                                                   int8_t dreq,
145                                                                   int8_t cardcs)
146     : Adafruit_VS1053_ESP32(mosi, miso, clk, rst, cs, dcs, dreq) {
147
148   playingMusic = false;
149   _cardCS = cardcs;
150   _loopPlayback = false;
151 }
152
153 boolean Adafruit_VS1053_ESP32_FilePlayer::begin(void) {
154   // Set the card to be disabled while we get the VS1053 up
155   pinMode(_cardCS, OUTPUT);
```

```
156   digitalWrite(_cardCS, HIGH);
157
158   uint8_t v = Adafruit_VS1053_ESP32::begin();
159
160   // dumpRegs();
161   // Serial.print("Version = "); Serial.println(v);
162   return (v == 4);
163 }
164
165 boolean Adafruit_VS1053_ESP32_FilePlayer::playFullFile(const char *
      trackname) {
166   if (!startPlayingFile(trackname))
167     return false;
168
169   while (playingMusic) {
170     // twiddle thumbs
171     feedBuffer();
172     delay(5); // give IRQs a chance
173   }
174   // music file finished!
175   return true;
176 }
177
178 void Adafruit_VS1053_ESP32_FilePlayer::stopPlaying(void) {
179   // cancel all playback
180   sciWrite(VS1053_REG_MODE, VS1053_MODE_SM_LINE1 | VS1053_MODE_SM_SDINEW |
181                                 VS1053_MODE_SM_CANCEL);
182
183   // wrap it up!
184   playingMusic = false;
185   currentTrack.close();
186 }
187
188 void Adafruit_VS1053_ESP32_FilePlayer::pausePlaying(boolean pause) {
189   playingMusic = (!pause && currentTrack);
190   if (playingMusic) {
191     feedBuffer();
192   }
193 }
194
195 boolean Adafruit_VS1053_ESP32_FilePlayer::paused(void) {
196   return (!playingMusic && currentTrack);
197 }
198
199 boolean Adafruit_VS1053_ESP32_FilePlayer::stopped(void) {
200   return (!playingMusic && !currentTrack);
201 }
202
203 void Adafruit_VS1053_ESP32_FilePlayer::playbackLoop(boolean loopState) {
204   _loopPlayback = loopState;
205 }
206
207 boolean Adafruit_VS1053_ESP32_FilePlayer::playbackLooped() { return
      _loopPlayback; }
208
209 // Just checks to see if the name ends in ".mp3"
```

```
210 boolean Adafruit_VS1053_ESP32_FilePlayer::isMP3File(const char *fileName)
    {
211   return (strlen(fileName) > 4) &&
212          !strcasecmp(fileName + strlen(fileName) - 4, ".mp3");
213 }
214
215 unsigned long Adafruit_VS1053_ESP32_FilePlayer::mp3_ID3Jumper(File mp3) {
216
217   char tag[4];
218   uint32_t start;
219   unsigned long current;
220
221   start = 0;
222   if (mp3) {
223     current = mp3.position();
224     if (mp3.seek(0)) {
225       if (mp3.read((uint8_t *)tag, 3)) {
226         tag[3] = '\0';
227         if (!strcmp(tag, "ID3")) {
228           if (mp3.seek(6)) {
229             start = 0ul;
230             for (byte i = 0; i < 4; i++) {
231               start <<= 7;
232               start |= (0x7F & mp3.read());
233             }
234           } else {
235             // Serial.println("Second seek failed?");
236           }
237         } else {
238           // Serial.println("It wasn't the damn TAG.");
239         }
240       } else {
241         // Serial.println("Read for the tag failed");
242       }
243     } else {
244       // Serial.println("Seek failed? How can seek fail?");
245     }
246     mp3.seek(current); // Put you things away like you found 'em.
247   } else {
248     // Serial.println("They handed us a NULL file!");
249   }
250   // Serial.print("Jumper returning: "); Serial.println(start);
251   return start;
252 }
253
254 boolean Adafruit_VS1053_ESP32_FilePlayer::startPlayingFile(const char *
      trackname) {
255   // reset playback
256   sciWrite(VS1053_REG_MODE, VS1053_MODE_SM_LINE1 | VS1053_MODE_SM_SDINEW |
257                                 VS1053_MODE_SM_LAYER12);
258   // resync
259   sciWrite(VS1053_REG_WRAMADDR, 0x1e29);
260   sciWrite(VS1053_REG_WRAM, 0);
261
262   currentTrack = SD.open(trackname);
263   if (!currentTrack) {
```

```
264     return false;
265   }
266
267   // We know we have a valid file. Check if .mp3
268   // If so, check for ID3 tag and jump it if present.
269   if (isMP3File(trackname)) {
270     currentTrack.seek(mp3_ID3Jumper(currentTrack));
271   }
272
273   // don't let the IRQ get triggered by accident here
274   noInterrupts();
275
276   // As explained in datasheet, set twice 0 in REG_DECODETIME to set time
        back
277   // to 0
278   sciWrite(VS1053_REG_DECODETIME, 0x00);
279   sciWrite(VS1053_REG_DECODETIME, 0x00);
280
281   playingMusic = true;
282
283   // wait till its ready for data
284   while (!readyForData()) {
285 #if defined(ESP8266)
286     ESP.wdtFeed();
287 #endif
288   }
289
290   // fill it up!
291   while (playingMusic && readyForData()) {
292     feedBuffer();
293   }
294
295   // ok going forward, we can use the IRQ
296   interrupts();
297
298   return true;
299 }
300
301 void Adafruit_VS1053_ESP32_FilePlayer::feedBuffer(void) {
302   noInterrupts();
303   // dont run twice in case interrupts collided
304   // This isn't a perfect lock as it may lose one feedBuffer request if
305   // an interrupt occurs before feedBufferLock is reset to false. This
306   // may cause a glitch in the audio but at least it will not corrupt
307   // state.
308   if (feedBufferLock) {
309     interrupts();
310     return;
311   }
312   feedBufferLock = true;
313   interrupts();
314
315   feedBuffer_noLock();
316
317   feedBufferLock = false;
318 }
```

```
319
320 void Adafruit_VS1053_ESP32_FilePlayer::feedBuffer_noLock(void) {
321   if ((!playingMusic) // paused or stopped
322       || (!currentTrack) || (!readyForData())) {
323     return; // paused or stopped
324   }
325
326   // Feed the hungry buffer! :)
327   while (readyForData()) {
328     // Read some audio data from the SD card file
329     int bytesread = currentTrack.read(mp3buffer, VS1053_DATABUFFERLEN);
330
331     if (bytesread == 0) {
332       // must be at the end of the file
333       if (_loopPlayback) {
334         // play in loop
335         if (isMP3File(currentTrack.name())) {
336           currentTrack.seek(mp3_ID3Jumper(currentTrack));
337         } else {
338           currentTrack.seek(0);
339         }
340       } else {
341         // wrap it up!
342         playingMusic = false;
343         currentTrack.close();
344         break;
345       }
346     }
347
348     playData(mp3buffer, bytesread);
349   }
350 }
351
352 // get current playback speed. 0 or 1 indicates normal speed
353 uint16_t Adafruit_VS1053_ESP32_FilePlayer::getPlaySpeed() {
354   noInterrupts();
355   sciWrite(VS1053_SCI_WRAMADDR, VS1053_PARA_PLAYSPEED);
356   uint16_t speed = sciRead(VS1053_SCI_WRAM);
357   interrupts();
358   return speed;
359 }
360
361 // set playback speed: 0 or 1 for normal speed, 2 for 2x, 3 for 3x, etc.
362 void Adafruit_VS1053_ESP32_FilePlayer::setPlaySpeed(uint16_t speed) {
363   noInterrupts();
364   sciWrite(VS1053_SCI_WRAMADDR, VS1053_PARA_PLAYSPEED);
365   sciWrite(VS1053_SCI_WRAM, speed);
366   interrupts();
367 }
368
369 /*****************************************************************/
370
371 /* VS1053 'low level' interface */
372 static volatile PortReg *clkportreg, *misoportreg, *mosiportreg;
373 static PortMask clkpin, misopin, mosipin;
374
```

```
375 Adafruit_VS1053_ESP32::Adafruit_VS1053_ESP32(int8_t mosi, int8_t miso,
        int8_t clk,
376                                  int8_t rst, int8_t cs, int8_t dcs,
377                                  int8_t dreq) {
378   _mosi = mosi;
379   _miso = miso;
380   _clk = clk;
381   _reset = rst;
382   _cs = cs;
383   _dcs = dcs;
384   _dreq = dreq;
385
386   useHardwareSPI = false;
387
388   clkportreg = portOutputRegister(digitalPinToPort(_clk));
389   clkpin = digitalPinToBitMask(_clk);
390   misoportreg = portInputRegister(digitalPinToPort(_miso));
391   misopin = digitalPinToBitMask(_miso);
392   mosiportreg = portOutputRegister(digitalPinToPort(_mosi));
393   mosipin = digitalPinToBitMask(_mosi);
394 }
395
396 Adafruit_VS1053_ESP32::Adafruit_VS1053_ESP32(int8_t rst, int8_t cs, int8_t
        dcs,
397                                  int8_t dreq) {
398   _mosi = 0;
399   _miso = 0;
400   _clk = 0;
401   useHardwareSPI = true;
402   _reset = rst;
403   _cs = cs;
404   _dcs = dcs;
405   _dreq = dreq;
406 }
407
408 void Adafruit_VS1053_ESP32::applyPatch(const uint16_t *patch, uint16_t
        patchsize) {
409   uint16_t i = 0;
410
411   // Serial.print("Patch size: "); Serial.println(patchsize);
412   while (i < patchsize) {
413     uint16_t addr, n, val;
414
415     addr = pgm_read_word(patch++);
416     n = pgm_read_word(patch++);
417     i += 2;
418
419     // Serial.println(addr, HEX);
420     if (n & 0x8000U) { // RLE run, replicate n samples
421       n &= 0x7FFF;
422       val = pgm_read_word(patch++);
423       i++;
424       while (n--) {
425         sciWrite(addr, val);
426       }
427     } else { // Copy run, copy n samples
```

```
428       while (n--) {
429         val = pgm_read_word(patch++);
430         i++;
431         sciWrite(addr, val);
432       }
433     }
434   }
435 }
436
437 uint16_t Adafruit_VS1053_ESP32::loadPlugin(char *plugname) {
438
439   File plugin = SD.open(plugname);
440   if (!plugin) {
441     Serial.println("Couldn't open the plugin file");
442     Serial.println(plugin);
443     return 0xFFFF;
444   }
445
446   if ((plugin.read() != 'P') || (plugin.read() != '&') ||
447       (plugin.read() != 'H'))
448     return 0xFFFF;
449
450   uint16_t type;
451
452   // Serial.print("Patch size: "); Serial.println(patchsize);
453   while ((type = plugin.read()) >= 0) {
454     uint16_t offsets[] = {0x8000UL, 0x0, 0x4000UL};
455     uint16_t addr, len;
456
457     // Serial.print("type: "); Serial.println(type, HEX);
458
459     if (type >= 4) {
460       plugin.close();
461       return 0xFFFF;
462     }
463
464     len = plugin.read();
465     len <<= 8;
466     len |= plugin.read() & ~1;
467     addr = plugin.read();
468     addr <<= 8;
469     addr |= plugin.read();
470     // Serial.print("len: "); Serial.print(len);
471     // Serial.print(" addr: $"); Serial.println(addr, HEX);
472
473     if (type == 3) {
474       // execute rec!
475       plugin.close();
476       return addr;
477     }
478
479     // set address
480     sciWrite(VS1053_REG_WRAMADDR, addr + offsets[type]);
481     // write data
482     do {
483       uint16_t data;
```

```
484       data = plugin.read();
485       data <<= 8;
486       data |= plugin.read();
487       sciWrite(VS1053_REG_WRAM, data);
488     } while ((len -= 2));
489   }
490
491   plugin.close();
492   return 0xFFFF;
493 }
494
495 boolean Adafruit_VS1053_ESP32::readyForData(void) { return digitalRead(
      _dreq); }
496
497 void Adafruit_VS1053_ESP32::playData(uint8_t *buffer, uint8_t buffsiz) {
498   spi_dev_data->write(buffer, buffsiz);
499 }
500
501 void Adafruit_VS1053_ESP32::setVolume(uint8_t left, uint8_t right) {
502   // accepts values between 0 and 255 for left and right.
503   uint16_t v;
504   v = left;
505   v <<= 8;
506   v |= right;
507
508   noInterrupts(); // cli();
509   sciWrite(VS1053_REG_VOLUME, v);
510   interrupts(); // sei();
511 }
512
513 uint16_t Adafruit_VS1053_ESP32::decodeTime() {
514   noInterrupts(); // cli();
515   uint16_t t = sciRead(VS1053_REG_DECODETIME);
516   interrupts(); // sei();
517   return t;
518 }
519
520 void Adafruit_VS1053_ESP32::softReset(void) {
521   sciWrite(VS1053_REG_MODE, VS1053_MODE_SM_SDINEW | VS1053_MODE_SM_RESET);
522   delay(100);
523 }
524
525 void Adafruit_VS1053_ESP32::reset() {
526   // TODO:
527   // http://www.vlsi.fi/player_vs1011_1002_1003/modularplayer/vs10xx_8c.
       html#a3
528   // hardware reset
529   if (_reset >= 0) {
530     digitalWrite(_reset, LOW);
531     delay(100);
532     digitalWrite(_reset, HIGH);
533   }
534
535   delay(100);
536   softReset();
537   delay(100);
```

```
538
539   sciWrite(VS1053_REG_CLOCKF, 0x6000);
540
541   setVolume(40, 40);
542 }
543
544 uint8_t Adafruit_VS1053_ESP32::begin(void) {
545   if (_reset >= 0) {
546     pinMode(_reset, OUTPUT);
547     digitalWrite(_reset, LOW);
548   }
549
550   pinMode(_dreq, INPUT);
551
552   if (useHardwareSPI) {
553     spi_dev_ctrl = new Adafruit_SPIDevice(_cs, 250000,
          SPI_BITORDER_MSBFIRST,
554                                           SPI_MODE0, &SPI);
555     spi_dev_data = new Adafruit_SPIDevice(_dcs, 8000000,
          SPI_BITORDER_MSBFIRST,
556                                           SPI_MODE0, &SPI);
557   } else {
558     spi_dev_ctrl = new Adafruit_SPIDevice(_cs, _clk, _miso, _mosi, 250000,
559                                           SPI_BITORDER_MSBFIRST, SPI_MODE0
                                                );
560     spi_dev_data = new Adafruit_SPIDevice(_dcs, _clk, _miso, _mosi,
          8000000,
561                                           SPI_BITORDER_MSBFIRST, SPI_MODE0
                                                );
562   }
563   spi_dev_ctrl->begin();
564   spi_dev_data->begin();
565
566   reset();
567
568   return (sciRead(VS1053_REG_STATUS) >> 4) & 0x0F;
569 }
570
571 void Adafruit_VS1053_ESP32::dumpRegs(void) {
572   Serial.print("Mode = 0x");
573   Serial.println(sciRead(VS1053_REG_MODE), HEX);
574   Serial.print("Stat = 0x");
575   Serial.println(sciRead(VS1053_REG_STATUS), HEX);
576   Serial.print("ClkF = 0x");
577   Serial.println(sciRead(VS1053_REG_CLOCKF), HEX);
578   Serial.print("Vol. = 0x");
579   Serial.println(sciRead(VS1053_REG_VOLUME), HEX);
580 }
581
582 uint16_t Adafruit_VS1053_ESP32::recordedWordsWaiting(void) {
583   return sciRead(VS1053_REG_HDAT1);
584 }
585
586 uint16_t Adafruit_VS1053_ESP32::recordedReadWord(void) {
587   return sciRead(VS1053_REG_HDAT0);
588 }
```

```
589
590 boolean Adafruit_VS1053_ESP32::prepareRecordOgg(char *plugname) {
591   sciWrite(VS1053_REG_CLOCKF, 0xC000); // set max clock
592   delay(1);
593   while (!readyForData())
594     ;
595
596   sciWrite(VS1053_REG_BASS, 0); // clear Bass
597
598   softReset();
599   delay(1);
600   while (!readyForData())
601     ;
602
603   sciWrite(VS1053_SCI_AIADDR, 0);
604   // disable all interrupts except SCI
605   sciWrite(VS1053_REG_WRAMADDR, VS1053_INT_ENABLE);
606   sciWrite(VS1053_REG_WRAM, 0x02);
607
608   int pluginStartAddr = loadPlugin(plugname);
609   if (pluginStartAddr == 0xFFFF)
610     return false;
611   Serial.print("Plugin at $");
612   Serial.println(pluginStartAddr, HEX);
613   if (pluginStartAddr != 0x34)
614     return false;
615
616   return true;
617 }
618
619 void Adafruit_VS1053_ESP32::stopRecordOgg(void) { sciWrite(
      VS1053_SCI_AICTRL3, 1); }
620
621 void Adafruit_VS1053_ESP32::startRecordOgg(boolean mic) {
622   /* Set VS1053 mode bits as instructed in the VS1053b Ogg Vorbis Encoder
623      manual. Note: for microphone input, leave SMF_LINE1 unset! */
624   if (mic) {
625     sciWrite(VS1053_REG_MODE, VS1053_MODE_SM_ADPCM | VS1053_MODE_SM_SDINEW
          );
626   } else {
627     sciWrite(VS1053_REG_MODE, VS1053_MODE_SM_LINE1 | VS1053_MODE_SM_ADPCM
          |
628                                   VS1053_MODE_SM_SDINEW);
629   }
630   sciWrite(VS1053_SCI_AICTRL0, 1024);
631   /* Rec level: 1024 = 1. If 0, use AGC */
632   sciWrite(VS1053_SCI_AICTRL1, 1024);
633   /* Maximum AGC level: 1024 = 1. Only used if SCI_AICTRL1 is set to 0. */
634   sciWrite(VS1053_SCI_AICTRL2, 0);
635   /* Miscellaneous bits that also must be set before recording. */
636   sciWrite(VS1053_SCI_AICTRL3, 0);
637
638   sciWrite(VS1053_SCI_AIADDR, 0x34);
639   delay(1);
640   while (!readyForData())
641     ;
```

```
642 }
643
644 void Adafruit_VS1053_ESP32::GPIO_pinMode(uint8_t i, uint8_t dir) {
645   if (i > 7)
646     return;
647
648   sciWrite(VS1053_REG_WRAMADDR, VS1053_GPIO_DDR);
649   uint16_t ddr = sciRead(VS1053_REG_WRAM);
650
651   if (dir == INPUT)
652     ddr &= ~_BV(i);
653   if (dir == OUTPUT)
654     ddr |= _BV(i);
655
656   sciWrite(VS1053_REG_WRAMADDR, VS1053_GPIO_DDR);
657   sciWrite(VS1053_REG_WRAM, ddr);
658 }
659
660 void Adafruit_VS1053_ESP32::GPIO_digitalWrite(uint8_t val) {
661   sciWrite(VS1053_REG_WRAMADDR, VS1053_GPIO_ODATA);
662   sciWrite(VS1053_REG_WRAM, val);
663 }
664
665 void Adafruit_VS1053_ESP32::GPIO_digitalWrite(uint8_t i, uint8_t val) {
666   if (i > 7)
667     return;
668
669   sciWrite(VS1053_REG_WRAMADDR, VS1053_GPIO_ODATA);
670   uint16_t pins = sciRead(VS1053_REG_WRAM);
671
672   if (val == LOW)
673     pins &= ~_BV(i);
674   if (val == HIGH)
675     pins |= _BV(i);
676
677   sciWrite(VS1053_REG_WRAMADDR, VS1053_GPIO_ODATA);
678   sciWrite(VS1053_REG_WRAM, pins);
679 }
680
681 uint16_t Adafruit_VS1053_ESP32::GPIO_digitalRead(void) {
682   sciWrite(VS1053_REG_WRAMADDR, VS1053_GPIO_IDATA);
683   return sciRead(VS1053_REG_WRAM) & 0xFF;
684 }
685
686 boolean Adafruit_VS1053_ESP32::GPIO_digitalRead(uint8_t i) {
687   if (i > 7)
688     return 0;
689
690   sciWrite(VS1053_REG_WRAMADDR, VS1053_GPIO_IDATA);
691   uint16_t val = sciRead(VS1053_REG_WRAM);
692   if (val & _BV(i))
693     return true;
694   return false;
695 }
696
697 uint16_t Adafruit_VS1053_ESP32::sciRead(uint8_t addr) {
```

```
698   uint8_t buffer[2] = {VS1053_SCI_READ, addr};
699   spi_dev_ctrl->write_then_read(buffer, 2, buffer, 2);
700   return (uint16_t(buffer[0]) << 8) | uint16_t(buffer[1]);
701 }
702
703 void Adafruit_VS1053_ESP32::sciWrite(uint8_t addr, uint16_t data) {
704   uint8_t buffer[4] = {VS1053_SCI_WRITE, addr, uint8_t(data >> 8),
705                        uint8_t(data & 0xFF)};
706   spi_dev_ctrl->write(buffer, 4);
707 }
708
709 void Adafruit_VS1053_ESP32::sineTest(uint8_t n, uint16_t ms) {
710   reset();
711
712   uint16_t mode = sciRead(VS1053_REG_MODE);
713   mode |= 0x0020;
714   sciWrite(VS1053_REG_MODE, mode);
715
716   while (!digitalRead(_dreq))
717     ;
718   //  delay(10);
719
720   uint8_t sine_start[8] = {0x53, 0xEF, 0x6E, n, 0x00, 0x00, 0x00, 0x00};
721   uint8_t sine_stop[8] = {0x45, 0x78, 0x69, 0x74, 0x00, 0x00, 0x00, 0x00};
722
723   spi_dev_data->write(sine_start, 8);
724   delay(ms);
725   spi_dev_data->write(sine_stop, 8);
726 }
```

E

User test

Name: Cesc Date: 26-01-'24

User interface test DualStim

In front of you is an empty MicroSD card, the DualStim, a stimulation clip, a headphone, and laptop with three audio tracks. The goal of this test is to check the user friendliness of the device. In this document, three steps have to be conducted, which can be scored on a scale of 1 being not user friendly to 5 very user friendly.

The device is used by a clinician/researcher to test if the device can treat patients with tinnitus. Multiple patients will be treated in three sessions per week, for three weeks. The device pairs transcutaneous electrical nerve stimulation with auditive stimulation. The clinician will adjust parameters in both the digital menu, as well as the SD card. The combination should give the clinician flexibility.

1. **Without any explanation, try to start a stimulation program with the following settings:**
   - **Volume: 20%**
   - **Max amplitude P5: 100 µA**
   - **P1 percentage of P5: 50%**
   - **Time delay E-A: -300 ms**
   - **Stimulation mode: 1**
   - **Audio track: track003.mp3**

Aspect	**Score (1-5)**
Use of the SD card	5
Navigating the menu	4
Selecting and changing a parameter	4
Starting/stopping the stimulation	5
Flexibility of the settings	4
Amount of times a step had to be redone:	

If you have any comments on the scores, please leave them here:

- P5 Amp doesn't say 'max' on the device
- As I'm not familiar with the stim protocol, I can't say much about the desired flexibility.
- I'm distracted by "B=75%". As I don't know what it means.
  - Would be good to have an option that shows all parameters at once (or auto scroll)

In the following steps, you may consult the user manual.

2. **The patient does not feel any electrical stimulation, even if the maximum amount of stimulation is given. Also the increments of the time delay is accurate enough. Adjust the menu options to the following settings:**
   - **Max stimulation amplitude: 2000µA**
   - **Step size in the menu: 50µA**
   - **Time delay step size: 50ms**

   Save the datalog.txt file also to the computer
   Afterwards, start the device and check the results of the adjustments.

Name: Cesc. Date: 26-01-'24

Aspect	Score (1-5)
Use of the SD card	3
Adjusting parameters in the datalog.txt file	1
Flexibility of the settings	3
Efficiency of the process of treatment of patients (e.g. if a treatment is 30 minutes, how much time do you need to install a new patient compared to the treatment duration?)	3
Amount of times a step had to be redone:	

If you have any comments on the scores, please leave them here:
- It's hard (and thus error prone to errors) to read which parameter you are changing in the datalog
- The efficiency is a matter of getting used to the device
- For most parameters it is not clear to me what the allowed values are. I'm pretty sure I could break some things if I wanted to

3. **A new patient comes in, which brings an empty SD card. You want to have the same menu settings as the previous patient loaded onto the device. Afterwards, start the device and check the results of the adjustments.**

Aspect	Score (1-5)
Use of the SD card	5
Adjusting parameters in the datalog.txt file	3
Flexibility of the settings	3
Efficiency of the process of treatment of patients (e.g. if a treatment is 30 minutes, how much time do you need to install a new patient compared to the treatment duration?)	3
Amount of times a step had to be redone:	

If you have any comments on the scores, please leave them here:
- For this step I didn't need to change datalog, and it was clear to me that I could just copy the file.
- Same comments as before for flex. & efficiency

Overall scores:

Aspect	Score (1-5)
Use of the SD card	5
Navigating the menu	4
Selecting and changing a parameter	5
Starting/stopping the stimulation	5
Adjusting parameters in the datalog.txt file	1
Flexibility of the settings	3
Efficiency of the process of treatment of patients (e.g. if a treatment is 30 minutes, how much time do you need to install a new patient compared to the treatment duration?)	3
Amount of times a step had to be redone:	

If you have any comments on the scores, please leave them here:
- I could change the stim. parameters during stimulation, is that desired?
- Datalog is probably way to complex for the average clinician
- I wonder why you want to limit parameters on the device if they're not safety based.
- I assume that the first couple of times they might spend some time in configuring the datalog, but after that they will only use the device which would make the process very efficient.
- I would mak all the interaction on the laptop with an UI.