

Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Measurement



Voice assistant for controlling the reclining bed

Master's thesis

Bc. Anel Poluektova

Master's programme: Intelligent buildings
Supervisor: Ing. Vít Janovský

Prague, May 2024

Declaration

I hereby declare I have written this doctoral thesis independently and quoted all the sources of information used in accordance with methodological instructions on ethical principles for writing an academic thesis. Moreover, I state that this thesis has neither been submitted nor accepted for any other degree.

In Prague, May 2024

.....
Bc. Anel Poluektova

I. Personal and study details

Student's name: **Poluektova Anel** Personal ID number: **516433**
Faculty / Institute: **Faculty of Electrical Engineering**
Department / Institute: **Department of Measurement**
Study program: **Intelligent Buildings**

II. Master's thesis details

Master's thesis title in English:

Voice assistant for controlling the reclining bed

Master's thesis title in Czech:

Hlasový asistent pro ovládání polohovatelné postele

Guidelines:

The aim of the work is to design and implement a module for voice control of an electrically adjustable bed. The users will be people with specific needs, for example wheelchair users.

Select an appropriate bed according to the results in previous projects in which a consultation with a target group of wheelchair users was carried out, the functional requirements for the bed and the voice assistant were defined, suitable components selected and a speech recognition module implemented.

Connect the voice control module to the bed and implement testing with users. Voice controlled bed:

- recognizes the trigger word,
- recognises at least 10 commands,
- requires command confirmation,
- gives feedback on the instruction,
- allows personalised recording of the command (specific pronunciation for people with disabilities),
- has a web interface for setup.

Test the reliability of the voice module on user commands. Create documentation for the solution and code to allow further expansion and modification of the solution. The bed installation and implementation are at the UCEEB research centre.

Bibliography / sources:

- [1] Nedvěd, Jakub. *Evaluace hlasových dialogových systémů*. Plzeň, 2013. Bakalářská práce. Západočeská univerzita v Plzni. Dostupné na: <http://hdl.handle.net/11025/10426>
- [2] Nurul Fadillah and Ahmad Ihsan, *Smart Bed Using Voice Recognition for Paralyzed Patient*, IOP Conf. Series: Materials Science and Engineering, 2020, DOI 10.1088/1757-899X/854/1/012045
- [3] Kenichiro Noda, *Google Home: smart speaker as environmental control unit*, Disability and Rehabilitation: Assistive Technology, 13:7, 674-675, 2017, DOI: 10.1080/17483107.2017.1369589
- [4] D. Wang and H. Yu, "Development of the control system of a voice-operated wheelchair with multi-posture characteristics," 2017 2nd Asia-Pacific Conference on Intelligent Robot Systems (ACIRS), Wuhan, China, 2017, pp. 151-155, doi: 10.1109/ACIRS.2017.7986083.

Name and workplace of master's thesis supervisor:

Ing. Vít Janovský Quality of Inner environment UCEEB

Name and workplace of second master's thesis supervisor or consultant:

Date of master's thesis assignment: **12.02.2024** Deadline for master's thesis submission: **24.05.2024**

Assignment valid until:
by the end of summer semester 2024/2025

Ing. Vít Janovský
Supervisor's signature

Head of department's signature

prof. Mgr. Petr Páta, Ph.D.
Dean's signature

III. Assignment receipt

The student acknowledges that the master's thesis is an individual work. The student must produce her thesis without the assistance of others, with the exception of provided consultations. Within the master's thesis, the author must state the names of consultants and include a list of references.

Date of assignment receipt

Student's signature

Abstract

This thesis presents the development and implementation of a voice-controlled bed positioning system, designed to enhance accessibility for individuals with limited mobility. The system utilizes Edge Impulse platform with TensorFlow Lite, to create a speaker-dependent tiny model speech recognition library based on machine learning. The core functionality is achieved through continuous audio sampling and inferencing, ensuring real-time responsiveness. The system is capable of recognizing a wake-up word, followed by eleven distinct commands to adjust the bed's position or a degree of adjustment. It seeks user permission before executing any adjustments and provides feedback on each command executed for testing purposes using the BLE technology.

Additionally, the system supports the customization of voice commands through a web application and platform API, allowing users to record their own commands tailored to their speech patterns. The hardware implementation is based on an Arduino Nano 33 BLE Sense microcontroller, a small yet powerful device for autonomous command classification. This project demonstrates the feasibility of voice-activated assistive technology on the edge device, offering improvements in independence and quality of life for users with mobility challenges.

Keywords: Medical bed, tiny machine learning, Internet of Things, microcontrollers.

Abstrakt

Tato diplomová práce představuje vývoj a implementaci systému pro polohování lůžka ovládaného hlasem, který je navržen pro osoby se sníženou soběstačností. Systém využívá platformu Edge Impulse s TensorFlow Lite k vytvoření knihovny pro rozpoznávání řeči, založené na strojovém učení v malém modelu. Základní funkčnost je dosažena prostřednictvím kontinuálního vzorkování zvuku a inferencí, což zajišťuje okamžitou odezvu v reálném čase. Systém je schopen rozpoznat aktivační slovo, po kterém následuje jedenáct různých příkazů pro nastavení stupně polohy lůžka. Před provedením jakýchkoli úprav systém vyžaduje povolení od uživatele a poskytuje zpětnou vazbu o každém provedeném příkazu za účelem testování pomocí technologie BLE ze zařízení uživatele.

Systém navíc podporuje přizpůsobení hlasových příkazů prostřednictvím webové aplikace a API platformy, což umožňuje uživatelům nahrávat své vlastní příkazy přizpůsobené jejich řečovým vzorcům. Hardwarová implementace je založena na mikrokontroléru Arduino Nano 33 BLE Sense, malém, ale výkonném zařízení pro autonomní klasifikaci příkazů. Tento projekt demonstruje proveditelnost asistivní technologie ovládané hlasem na koncovém zařízení a nabízí zlepšení nezávislosti a kvality života pro uživatele s pohybovými obtížemi.

Klíčová slova: Polohovací lůžko, malé strojové učení, Internet věcí, mikrokontroléry.

Acknowledgements

I would like to sincerely thank my supervisor, Ing. Vít Janovský, for his unwavering support throughout the project, his guidance, and assistance in helping me find the right solutions and maintain a steady workflow. I am also deeply grateful to Ing. Martin Faltus for sharing his expertise in electronics. His help with component selection and troubleshooting hardware issues was essential to the project's success. Additionally, I would like to acknowledge Ing. Ales Vodička and Ing. Jakub Vaněk for their assistance in identifying and resolving electronic problems with the bed. Their contributions were crucial in overcoming these challenges. Lastly, I extend my gratitude to my supervisor for granting me free access to the UCEEB facilities and laboratory of telemedicine, which provided the necessary resources and environment to conduct my research and complete this project.

Contents

Abstract	i
Abstrakt	ii
1 Introduction	1
1.1 Significance	1
1.2 Objectives	2
1.3 Literature review	2
2 TinyML overview	4
3 Component research	6
3.1 Language support	6
3.1.1 PocketSphinx	6
3.1.2 Open AI Whisper	7
3.1.3 Vosk API	7
3.1.4 Google Cloud	7
3.1.5 Smart voice assitants	8
3.1.6 Edge Impulse and Tensor flow lite	10
3.1.7 Electronic modules for speech recognition	10
3.2 Component comparison	11
4 Project's requirements	12
4.1 Results of the survey	13
5 Technology overview	15
5.1 Medical bed research	15
5.2 Selected medical bed parameters	16
5.3 Micro-controller comparison	17
5.4 Hardware components	17
5.4.1 BLE technology	18
5.4.2 OTA update	19
6 Implementation	21
6.1 Language model approaches	21
6.1.1 MFE parameters	22

6.1.2	MFCC	23
6.2	Model training and library creation	23
6.2.1	Model optimization	23
6.2.2	Neural network architecture	24
6.2.3	Neural network settings	25
6.2.4	Commands	26
6.2.5	Training the model	27
6.3	Algorithm description	28
6.4	Code examples	31
6.5	Web application	31
6.5.1	Audio recording	32
6.5.2	Uploading to Edge Impulse API	33
6.6	Hardware connection	33
7	Testing	37
7.1	Performance requirements	37
7.2	Test conditions	37
7.3	Testing with the user	39
7.4	Testing results	40
8	Discussion	42
9	Conclusion	44
9.1	Future work	44
A	Attachments	46
	List of Tables	55
	List of Figures	56
	List of Acronyms	57

Chapter 1

Introduction

This Master's thesis is focused on the creation of a working prototype of an adjustable medical bed with voice control. The intended group of users is defined as people with disabilities, for example, wheelchair users. The project is carried out at UCEEB facilities in the laboratory of telemedicine. In the project, both software and hardware parts are engineered and discussed. The tests and optimisations are conducted on various parts of the project using simulations and hands-on realisation. The thesis is structured as follows: in Chapter 1, the importance of the issue is discussed along with the thesis's targets and solutions provided by other researchers regarding a similar problem. In Chapter 2, a place of Tiny Machine learning for IoT technologies is briefly discussed. Chapter 3 provides an overview of the components for both language models and hardware solutions. In Chapter 4, the requirements for the prototype are discussed based on the survey and literature. Chapter 5 provides a technology overview with a more detailed description of the selected parts and their functionality. Chapter 6 focuses on the actual implementation and ways in which different parts are connected. It includes the language model, the physical implementation, and the web application. Testing procedure and results are depicted in Chapter 7, followed by a thorough overall discussion in Chapter 8. Chapter 9 concludes the thesis along with suggestions for future work.

1.1 Significance

According to the World Health Organization, more than 1.6 billion people worldwide suffer from disabilities, among them people with sensory, mental, intellectual, and developmental types of impairments. [1] With the development of technology, smart houses have become more comfortable, safe, and functional. People with severe disabilities now have an opportunity to maintain high living standards outside of hospitals. Voice-controlled bed is a technology that allows people with mobile disabilities to feel comfort and independence at home. Using this system, the person can control the bed without the help of a caregiver. The module has several preset modes of operation based on the user's habits. This technology could benefit patients with visual impairment and limited mobility. Such a bed aims to create more opportunities and make everyday activities more accessible for people with low self-sufficiency, wheelchair users in particular. Automatic control also

helps caregivers with manipulating the weight of the person when necessary.

Telemedicine and smart healthcare are trends in the technological world. The demand for efficient healthcare solutions for preventive, monitoring, treating, and rehabilitating purposes has increased in the last few years, especially after the COVID-19 pandemic. [2] The market has been expanding at a rapid rate, which resulted in a smart healthcare market value of USD 221.8 billion in 2021. The forecast from 360 Research Reports indicates a compound and annual growth rate of 14.79 %, reaching over 507 billion by 2027. [3] The latest area of interest lies in the segment of patient monitoring and integration of AI and data analytics. While the concept of machine learning is known for half a century, interest in deep machine learning for speech recognition of various techniques and methods has grown tremendously since 2009. [4] The convergence of new available technologies and healthcare paves the way to innovative assistive solutions that have potential to transform approach to patient care and rehabilitation.

1.2 Objectives

There are several conditions to the projects that have to be fulfilled. The system has to be designed to recognize a wake-up word, after which it becomes ready to accept further instructions. It shall interpret and respond to at least ten different commands for bed positioning. Before executing any position adjustment, the system seeks permission. Users receive feedback about each executed command. Additionally, the system allows users, particularly those with limited accessibility, to record their own voice and commands based on their speaking capabilities. There also shall be a web application available for setting up the system. Testing of the model with a user also has to be fulfilled.

1.3 Literature review

This section is aimed at familiarising the reader with existing approaches to creating an adjustable bed or a wheelchair commanded by voice. These solutions follow similar ways to addressing the issue, where speech recognition is implemented in the cloud and commands are then sent to the microcontroller.

In ‘Smart Bed Using Voice Recognition for Paralyzed Patient’, authors have proposed a model that uses Arduino UNO, Elechouse v3 as a voice recognition module, linear actuator to control the bed in five positions. [5] The system used a microphone for the invoice input after which the comparison between the stored voice sample and speech command in the voice recognition module. The recognized command is then processed by the Arduino microprocessor. The signal is then sent to the relay module to execute the command. The linear actuator performs the command by changing the position of the grid up or down. The authors have designed and constructed the bed frame by themselves. The voice recognition module supports up to 7 commands at given settings and is speaker-dependent.

In the ‘Next Generation of Medical Care Bed with Internet of Things Solutions’, the focus of the work was on creation of the technologically advanced medical bedding with

sensors for monitoring patient's condition. [6] The transmitter module is also based on Arduino Uno. Easy VR Shield 2.0 is the speech recognition module. The transmitter includes an LED for visual representation during the process of command acquisition. In their consecutive conference paper, authors have added vitality monitoring sensors such as temperature, humidity, and pulse measurements with data available to follow on the display. [7] The system has 4 relays for operating different modes.

Wongson et al., have developed a system that uses voice commands to control a hospital bed. [8] Authors have used an Arduino module produced by Wemos that operates three stepper motors to regulate the top, middle, and bottom parts. The Arduino device includes an Arduino Mega 2560 microcontroller and a Wi-Fi communication module. The speech recognition and processing algorithms are based on identifying the Mel Frequency Cepstral Coefficients (MFCC) to create feature vectors from the received sound, neural network with consequent application of fuzzy logic to identify the command. The accuracy of the model achieved during testing was 71.5 %. In their implementation speech recognition is performed on the smartphone and a request is sent via Wi-Fi to the Arduino module. The application recognizes 8 commands and utilizes a back-propagation neural network (BPNN) as a speech recognition model.

Similar research has been conducted for modeling of the multi-posture electrical wheelchair. [9] The wheelchair is designed for sitting, standing, and lying positions. The voice module is based on an RSC4128 speech recognition processor that is connected to the AT89S51 MCU (micro control unit) of linear actuators. Software is divided into two modes, training and recognition parts.

Chapter 2

TinyML overview

TinyML (tiny machine learning) is a newly emerging field in machine learning that focuses on edge devices such as watches, and small sensors. Instead of sending data to the cloud services and receiving responses, embedded devices use optimized models for more efficient and fast processing of the data from the real world.

The interest in TinyML is increasing at a rapid rate. In a survey by Han [10], in 2019 there was only one publication about the topic, while in 2021 21 articles and papers were published. Since the beginning of year 2024, there are 24 papers with 'TinyML' included in the name on research database IEEE Xplore alone. [11]

TinyML's flexibility and adaptability empower its integration into a wide array of domains, from healthcare and industrial automation to consumer electronics. For instance, in the context of a voice-controlled bed positioning system, TinyML facilitates the development of compact and efficient models capable of running on microcontrollers, catering to the specific needs of the application domain. Applications of TinyML include personalized health care and monitoring, smart home devices, human-machine interface, smart vehicle, and transportation, agriculture, and anomaly detection. [12] Edge devices have an advantage in power consumption being reduced by 23 % while maintaining an accuracy of 90 % in both training and testing. This is due to the new approach of approximate computing. [13] According to [14] the power consumption of such devices shall not exceed a few milliwatts. Approximate computing proves to be a promising strategy for curtailing power consumption, as it allows for the execution of imprecise computations.

Edge devices are capable of performing complex algorithms in portable, cheap, and frugal devices, which makes them attractive for many purposes. Large, powerful, and accurate neural networks consume so much energy, that for some industries it does not appeal in certain cases. Recent market trends from [15] indicate, that the industry opts out for less power-consuming devices.

Microcontrollers, the backbone of TinyML, are resource-constrained devices with limited processing power, memory, and energy. TinyML algorithms are tailored to operate efficiently within these constraints, making them well-suited for edge devices. This efficiency not only optimizes resource utilization but also extends the lifespan of battery-powered devices, contributing to sustainable and eco-friendly technology. [16]

Another game-changing feature of edge devices with machine learning is low latency. [17] By minimizing the distance between the endpoint and the processor, a much higher

speed can be achieved contrary to the cloud-based architectures. The scope of TinyML applications is extensive and evolving as the field gains momentum. Its distinct advantage lies in its ability to bring machine learning capabilities directly to the sensor level, where data originates. Consequently, TinyML enables a diverse array of new applications that conventional machine-learning methods cannot support due to constraints related to bandwidth, latency, cost-effectiveness, reliability, and privacy considerations. Nevertheless, new technologies and advances in wireless technologies allow embedded systems to be online and work with cloud services at a limited scale. [18]

Low price and high accessibility is another benefit of this approach. Inference in these low-cost microcontrollers makes scalability and autonomous work possible even at distant locations without the internet or power grid. [17] The number of edge devices exceeds that of more conventional cloud-based mobile systems. [19] The prevalence of small embedded devices makes TinyML suitable for localized machine learning tasks that were previously impractical due to cost constraints, including distributed sensor networks and predictive maintenance systems in industrial manufacturing environments.

TinyML finds its application in a variety of areas: keyword spotting and audio event recognition, image classification, anomaly detection, motor control, gesture recognition, forecasting, face recognition, activity detection, and many others. [15]

TinyML usually uses deep learning for building models. A network in deep learning consists of so-called neurons to create a relationship between input and output. These neurons are represented by arrays of numbers. Different neural network architectures may accommodate a particular task better, than another one. By choosing the correct architecture, the accuracy of an embedded system may be as high as that of a larger and more powerful machine. [17]

While the importance of TinyML is evident, challenges such as model size, accuracy, and compatibility persist. Ongoing research and advancements in algorithmic efficiency, model optimization, and hardware capabilities are actively addressing these challenges, positioning TinyML as a dynamic field with promising prospects.

Chapter 3

Component research

3.1 Language support

3.1.1 PocketSphinx

PocketSphinx, a compact version of CMU Sphinx made by Carnegie Mellon University, presents an offline-capable, language-independent, and flexible solution for speech recognition. Utilizing the Mel-frequency cepstrum coefficients (MFCC) algorithm for audio feature extraction, PocketSphinx aligns with the criteria for speech recognition model selection. [20]

One of its key strengths lies in its offline capability, making it suitable for applications where internet connectivity is unreliable or unavailable. This feature enhances privacy by processing audio data locally without the need for cloud services, addressing potential concerns about data security and privacy, which applies to healthcare usage.

Despite its compact size and offline capabilities, PocketSphinx does have limitations. It may struggle with noisy environments or accents outside its trained dataset, impacting its accuracy in such scenarios. [21] Additionally, its inability to recognize non-speech sounds may limit its applicability in certain contexts where robust audio analysis is required. Thus, making it difficult potentially for people with disabilities to use. The model requires adaptations in such cases to increase its accuracy.

PocketSphinx offers some degree of customization and adaptability, allowing users to fine-tune parameters to better suit specific use cases or dialects. However, compared to cloud-based solutions, it requires more technical expertise for setup and optimization.

Integration with existing systems requires familiarity with Python, GitHub and potentially Android OS. PocketSphinx provides libraries and APIs for various programming languages, facilitating integration into different platforms and devices.

In terms of maintenance, while PocketSphinx does not receive regular updates from the community, the frequency and extent of these updates may vary compared to commercially supported models. The tutorials for model integration have also not been tested for some time, according to the developers. [22] The reliability and accuracy are questionable as stated in the website. [23]

3.1.2 Open AI Whisper

OpenAI Whisper was released in September 2022 and emerged as a compelling language recognition model, offering support for 57 languages, including Czech. [24] Whisper offers state-of-the-art open-source models for audio transcription. Application of this language model requires knowledge of Python programming language. Another possibility is the usage of Whisper API. With a pricing structure of 0.006 USD/minute, it introduces a cost-effective solution. However, its dependence on an internet connection may limit its applicability in certain scenarios.

Whisper AI has built a large library of data through weakly supervised pre-training. [25] It has five models of different sizes and capabilities. The accuracy is not widely known, especially for languages other than English. Whisper might require a fairly large computing power to run the model, thus making it a costlier model compared to other ones.

3.1.3 Vosk API

Vosk API, a speech recognition solution, offers an array of features and functionalities tailored to meet diverse linguistic and computational requirements. Operating as a versatile tool, Vosk API supports several Slavic languages, including Czech, thus catering to a wide user base. It was created for scientific research and now offers commercial solutions for enterprise. [26] Noteworthy features of Vosk API include speaker identification, low latency response, and the flexibility for vocabulary modifications, rendering it adaptable to various contexts and user needs. [27]

One of its distinguishing characteristics is its scalability, enabling deployment on small devices such as Raspberry Pi. Despite its compact size and suitability for resource-constrained environments, the model's bulk, weighing in at a considerable 50 Mb, may limit its application on microcontrollers. [28] Such models could find better implementation on Android and iOS smartphones.

Vosk API provides bindings with many widely used programming languages: Python, C#, JavaScript. Its adaptability requires adding new vocabulary letter by letter in a transcription of sounds, which might be quite accurate for some common accents. On the other hand, not all sounds can be transcribed precisely, especially with patients with disabilities. Despite this limitation, Vosk API remains a formidable choice for speech recognition tasks, offering a balance between efficiency, scalability, and adaptability.

3.1.4 Google Cloud

Google Cloud Speech-to-Text stands out as a powerful and versatile language recognition model, offering support for a vast array of languages and dialects, totaling 125 languages. [29] Leveraging Google's advanced machine learning technologies, including deep neural networks, the model achieves high accuracy in transcribing speech into text across diverse linguistic contexts.

In terms of accuracy and performance, Google Cloud Speech-to-Text consistently delivers reliable results, especially in clear and well-articulated speech. Its robustness in

handling various accents, languages, and environmental noise makes it suitable for a wide range of applications, from transcription services to voice-activated assistants. However, its application for medical purposes is yet to be verified.

One of the notable advantages of Google Cloud Speech-to-Text is its seamless integration with other Google Cloud services, allowing for easy deployment and scalability. Its RESTful API enables straightforward integration into existing applications and workflows, while also providing Software Development Kits SDKs for popular programming languages, simplifying development across different platforms.

Privacy and security considerations are paramount when utilizing cloud-based services, and Google Cloud Speech-to-Text addresses these concerns through comprehensive data protection measures, including encryption in transit and at rest, identity and access management controls, and adherence to international compliance standards such as GDPR and HIPAA. [30]

Google Cloud Speech-to-Text benefits from regular updates and maintenance by Google's engineering team, ensuring that the model remains state-of-the-art and adaptable to evolving language patterns and technological advancements. Additionally, Google Cloud's extensive documentation, tutorials, and dedicated support channels offer users the resources they need to maximize the model's capabilities and troubleshoot any issues that may arise.

While Google Cloud Speech-to-Text offers exceptional performance and scalability, its reliance on an internet connection may pose constraints in scenarios with limited or intermittent connectivity. Furthermore, users should be mindful of the associated costs, which are based on usage, with a free tier available for up to 60 minutes of audio transcription per month and additional usage incurring a fee of USD 0.024 per minute. [31]

The only issues are internet dependency, quality of connection, and speed of speech recognition. Also, while Google Cloud services come at an affordable price, it may add extra constraints to the project. Audio files have to be uploaded into Google Cloud Provider Buckets, a tool for storing and managing data, which might result in greater latency. [32]

3.1.5 Smart voice assistants

Smartphone assistants have become integral to modern mobile devices, providing users with voice-activated help for various tasks. These assistants are powered by advanced AI and natural language processing algorithms, allowing them to understand and respond to user queries in multiple languages. Here is a short overview of the most prominent smartphone assistants: Google Assistant, Siri, Bixby, Alexa, and Cortana. Voice assistants are usually cloud-based solutions that require a connection to the internet. Otherwise, their functionality is limited. In the last few years, they have been widely used for home automation algorithms. [33]

Google Assistant

Google Assistant is one of the most widely used smartphone assistants. It supports over 30 languages, making it accessible to a broad range of users. However, Czech is not one of them at the moment of writing this thesis. [34] Google Assistant can perform

tasks such as setting reminders, sending messages, controlling smart home devices, and providing weather updates. For developers, Google offers the Actions on Google platform, which allows the creation of custom applications that can interact with Google Assistant. These actions can be used to extend the functionality of the assistant and integrate it with other services and devices, including Arduino projects via webhooks and APIs.

Siri

Siri, Apple's virtual assistant, is integrated into all Apple devices, including iPhones, iPads, and Macs. [35] Siri supports over 20 languages, including Czech, and offers a range of functionalities such as sending messages, making phone calls, setting reminders, and controlling HomeKit-enabled smart home devices. Developers can create custom Siri Shortcuts using the SiriKit framework, which allows for the automation of tasks and integration with third-party apps. While direct integration with Arduino is not straightforward, Siri Shortcuts can be used to trigger webhooks or other intermediary services that interact with Arduino devices.

Bixby

Bixby is Samsung's intelligent assistant, available on its range of smartphones and other smart devices. [36] Bixby supports several languages, but its language support is not as extensive as Google Assistant. As of now, Czech is not one of the supported languages. Bixby can perform tasks like controlling Samsung's smart home ecosystem, setting reminders, and more. Developers can use the Bixby Developer Studio to create custom capsules (Bixby's version of apps) that extend the assistant's functionality. Integration with Arduino is possible but typically requires intermediate steps, such as using Bixby to trigger webhooks or cloud services that then interact with Arduino devices.

Alexa

Amazon Alexa is well-known for its presence in smart speakers but is also available on smartphones. [37] Alexa supports multiple languages, but its support of Czech language is yet to be released. Alexa can control smart home devices, play music, provide news updates, and more. Developers can create custom skills using the Alexa Skills Kit (ASK), which allows for extensive customization and interaction with other services. For Arduino integration, Alexa can be connected via cloud services like Amazon Web Services IoT or by using smart home skills to control Arduino-based projects through intermediaries such as smart hubs. Another recent addition is Arduino Cloud and Alexa Skill integration. This allows discovery of smart home devices and enables control over their functionality using voice commands. [38]

Cortana

Cortana is Microsoft's digital assistant, integrated into Windows devices and available on smartphones through the Cortana app. [39] Cortana's language support includes several languages, but Czech is not one of them. Cortana's capabilities include setting reminders, sending emails, managing calendars, and providing information from the web. Microsoft provides the Cortana Skills Kit for developers to create custom skills. While Cortana's direct integration with Arduino projects is less common, it can be achieved by leveraging Microsoft's Azure cloud services to facilitate communication between Cortana and Arduino devices.

3.1.6 Edge Impulse and Tensor flow lite

Edge Impulse, in collaboration with TensorFlow, offers a purpose-built platform for edge devices. This solution stands out as easy to use and tailored for microcontrollers. [40] It addresses the limitations of other models by providing an integrated environment for training, optimizing, and deploying models, making it well-suited for the specific requirements of the bed positioning microcontroller project.

Edge Impulse provides a comprehensive platform tailored for developing and deploying machine learning models directly onto edge devices. Leveraging its user-friendly environment, developers can collect data, train models, and deploy them onto microcontrollers and similar edge devices. The platform's ease of use, coupled with pre-built signal processing blocks and machine learning algorithms, accelerates the development process, making it accessible to a wide range of users. [41]

TensorFlow, an open-source machine learning framework developed by Google, serves as a foundational tool for building and training machine learning models. [42] Its versatility and scalability make it a preferred choice for a diverse range of applications, including edge computing. TensorFlow offers extensive support for deep learning tasks, such as natural language processing and computer vision, providing developers with the flexibility to implement complex machine learning algorithms efficiently.

A platform that complements TensorFlow is Keras, a high-level neural networks API designed for ease of use and rapid prototyping. [43] Integrated with TensorFlow, Keras simplifies the process of building neural network architectures, allowing developers to focus on model design rather than low-level implementation details. Its modular and intuitive API, along with support for convolutional and recurrent neural networks, makes it an ideal choice for both beginners and experienced practitioners in the field of deep learning.

Together, Edge Impulse, TensorFlow, and Keras form a cohesive ecosystem that empowers developers to create and deploy machine learning models on edge devices effectively. By leveraging the capabilities of these tools, developers can address the challenges of edge computing and unlock new opportunities for innovation in various domains, ranging from IoT to industrial automation.

Deployment options of the platform include Arduino IDE library, C++ library, binaries for MacOS, Linux, Raspberry Pi, Nordic Semiconductor, and others.

3.1.7 Electronic modules for speech recognition

There are several options available on the market that are built as an additional module for a microcontroller. One of the commercial modules supports up to 80 commands, however, only 7 of them are available at a given time. **module** The supported languages include Chinese and English. Also, the programming and command setting requires the usage of Arduino IDE software and other programming skills that are considered as out of the scope of the target user. [44]

3.2 Component comparison

The selected approach allows people to create their own commands based on speaking capabilities. Larger models may perform better on people of different backgrounds, accents, ages, and gender, however, they are not intended for any speech disorders or any diversities from normal commands.

	Supported commands	Requires internet connection	Price	Speaker dependent	Size of the smallest model
Google Speech-To-Text API	unlimited	Yes	0.024 USD/minute	No	-
PocketSphinx	unlimited	No	Free	No	70 Mb
Vosk API	unlimited	No	Free	No	50 Mb
Open AI Whisper	unlimited	Yes	0.006 USD/minute	No	-
Electronic module	80 max 7 at a time	No	990 CZK	Yes	-
Edge Impulse	tens	No	Free	May be	up to 1 Mb

Table 3.1: Comparison of main features of the language models.

In the table, the comparison between available speech recognition solutions that support Czech is presented. Criteria of interest are the number of supported commands, the need for internet connectivity, the price for the service (without a microcontroller), dependency on a particular speaker and size of the model to be installed on-device. Google API and Whisper, being presumably the most accurate models, require an internet connection to the internet to send out requests for speech transcription. Vosk API and PocketSphinx were created as research projects, are free and work offline. The memory space taken up by these two models is small enough for smartphone deployment, however too large for microcontrollers with a maximum of 1 MB of storage available. Electronic module requires extensive training with resulting 7 commands, which might be too few for certain purposes. Edge Impulse on the other hand is a light-weight speaker-adapted solution available for free.

In conclusion, comparing language speech recognition models highlights the trade-offs between language support, cost, offline capability, and model size. Considering the project's focus on Czech language support, offline functionality, and deployment on microcontrollers, Edge Impulse with TensorFlow emerges as the most suitable solution, offering a balance of features, usability, and adaptability for the envisioned speech-controlled bed positioning system.

Chapter 4

Project's requirements

The criteria for evaluation of the complete control system are usability, comfort, stability, and comprehensibility. [45] The bed control can be either by voice or by remote control. The choice of mechanical properties includes available positioning regulations, maximum weight of the user, expected lifespan of the construction without compromising functionality and stability over time, remote control type, and aesthetics for everyday use. Positioning control may include the head, hips, legs, inclination angle, or height of the bed, shown as follows:





			
Inclination angle	Knee elevation	Head support	Bed height

Table 4.1: Positioning control of a medical bed.

Head elevation is intended for reading, watching TV, or relieving pressure. Adjusting the knee section promotes blood circulation and reduces strain on the lower back. Raising or lowering the entire bed to make getting in and out easier and to accommodate caregivers. Changing the inclination angle control may further assist the user, but is not a requirement.

The typical positioning bed for hospitals looks very practical, sterile, and clinical, and may not fully reflect the user's desire to create a pleasant and welcoming atmosphere for everyday use. Possible options include an affordable medical bed and another one that avoids the hospital look and promotes a sense of comfort, tranquility, and well-being.

Other considerations include a user-friendly interface, especially for people with special needs. Both the voice command system and the remote control should be intuitive and easy to use, accommodating users with varying levels of technical proficiency. Safety features must incorporate safety mechanisms, such as automatic stops at certain angles to prevent accidental injury. Personalized settings and adjustments are important, thus a certain degree of adaptability and variability has to be ensured.

By addressing these requirements, the project aims to develop a bed control system

that is not only functional and safe but also enhances the user's quality of life through thoughtful design and usability considerations.

4.1 Results of the survey

The survey has been conducted to understand the needs of the target group, account for several possible situations, and discover the most universal solution possible. The questionnaire consisted of questions regarding the adjustability of the height of the bed; inclusion of accessories such as a railing, a trapeze with a handle, and a table; daily usage of the bed; common positionings; interest in the addition of new features and preset functions by the users using the application interface; the possible benefit of the positioning grid that can be installed in any bed, which allows more choices of bed designs. The participants provided answers online through a Google form.

Overall, the requirement for height adjustment of the bed was quite high and resulted in 85 %. 75 % of the respondents have shown interest in choosing the design of the bed by themselves. 14 respondents would like to have access to the new basic functions according to their habits and routines. 7 participants out of 16 have responded against any type of accessories, while others favored the possibility of adding the night table or assisting railings. A quarter of the participants would agree to use the standard bedding, while others would appreciate the development of more flexible and universal solutions. The accessories are very individual and are to be added or removed based on the particular case. Their integration and drive can be beneficial but are not required in the final product. Height adjustment is beneficial. However, the realization is more complicated as it comes in conflict with other criteria, the design, cost, and complexity of the frame of the bed. Participants are mostly willing to be active product consumers and would like to engage in the programming of the preset functions of their bed. This will allow the end users to feel more control over the technologies and their surroundings.

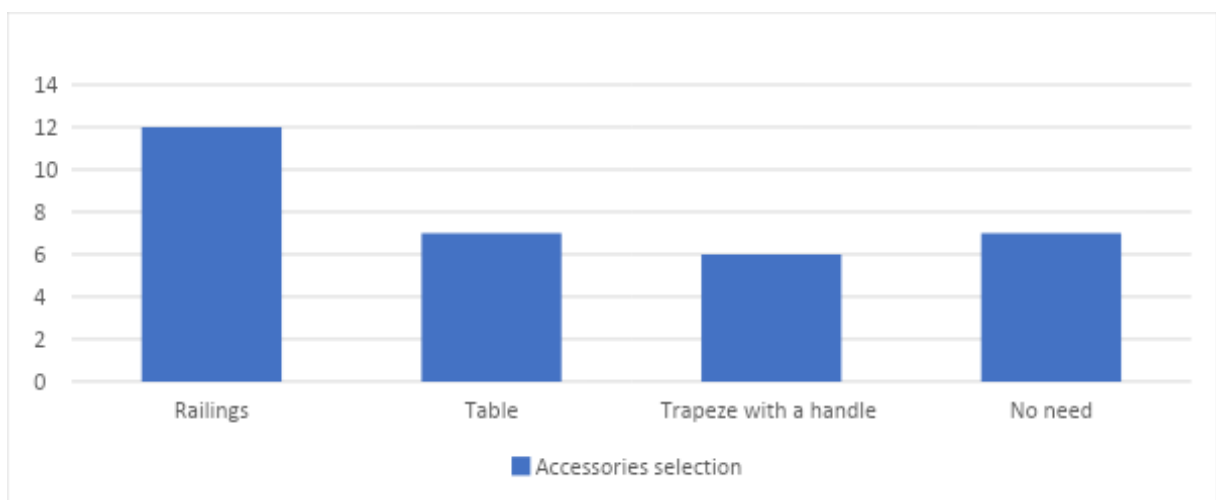


Figure 4.1: Results of the survey regarding bed accessories.

According to the respondents, the most frequent bed usage is watching TV, eating,

sleeping, changing clothes, resting, working, stretching, and rehabilitation exercises. Positions are with lifted head, and spine, moving and bending legs against swelling, readjustments for back pain prevention, and sitting positions for watching TV, working, reading, changing, and other hygienic activities. Respondents have mentioned positioning of the back is used most frequently. Bending the legs slightly is not a common option on the positioning grid market according to one of the participants of the survey. Positioning the grid with the widest possible degree of inclination is a desirable feature. Participants mentioned that bed design is very individual and each user's requirements can differ from that of another one. It can be concluded, that bed is used frequently for a variety of purposes during the day. To provide comfort, reliability, and safety, the bed system is to be modeled in a way to create opportunities for easier access to everyday needs without overcomplicating it. Based on the responses, the universality of the product is the main feature of the bed systems. By choosing a general solution that would satisfy the basic needs of the majority of the target group, this project can be expanded with additional features and accessories depending on the case of each user.

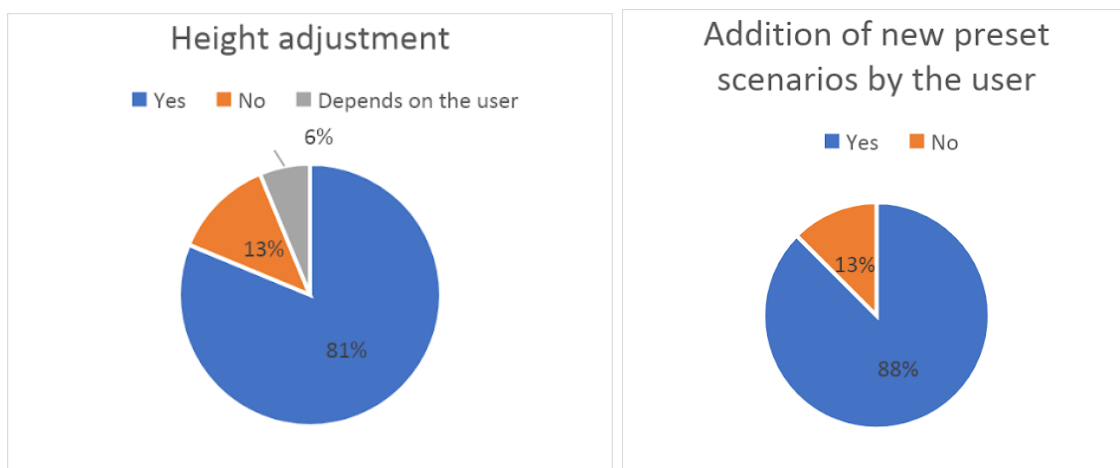


Figure 4.2: Results of the survey regarding bed functionality.

Chapter 5

Technology overview

5.1 Medical bed research

The beds available on the market differ based on the number of positioning options. Primarily, the beds are constructed as a one-piece item suitable for hospitals. Another option is a positioning grid that can be integrated into the bed. Mattresses are not included with the bed frame. All the beds support the weight of the patient up to 140-145 kg. Bed motors use a 24 V DC or 29 V DC power supply. [46] [47] [48] [49] [50]

The positioning grid may be a more versatile solution for beds to suit the needs of the user. However, according to the results of the research, the positioning grid's main and crucial disadvantage is the impossibility of regulation of the height of the bed. Other commercially available medical beds are bulky and may be unsophisticated for everyday use.



Figure 5.1: Selected medical adjustable bed.

Name	Control	Price	Advantages	Disadvantages
AKS 2fx	Wired remote control	9900 CZK	Highly affordable; Height adjustment;	Short warranty (6 months); unsophisticated design;
Burmeier Dali IV	Bluetooth remote control	22990 CZK	Higher reliability (warranty 2 years); wireless remote control; change of the bed incline angle; adjustable height;	Unsophisticated design;
SILVERTON 42 MOT	Radio remote control (+900 CZK) or wired control	8490 CZK	Affordable; Many variations of sizes; Flexibility with bed construction; 7 position regulation zones; 5 year warranty	No bed height adjustment;
Blissful nights	Alexa and google home voice control	500 USD + 70 USD (12000 CZK)	Complete product for a reasonable price	Unavailable on the Czech market; does not support Czech language; unknown suitability for people with impairments
Volker	Wired remote control	11900 CZK	Affordable; Height adjustment; 3 independent motors; Sturdy	No change of the inclination angle; Warranty 12 months; Unsophisticated design;

Table 5.1: Medical bed market overview[46] [47] [48] [49] [50]

5.2 Selected medical bed parameters

The bed is an electrically adjustable bed of the Volker brand. It has all the necessary functionality and does look pleasant despite being a medical assistance appliance. The parameters of the selected bed are presented as following:

Technical parameters height	
Bedding area (cm ²)	90 x 200
Minimum adjustable height (cm)	47
Maximum adjustable height (cm)	75
Weight (kg)	90
Maximum load capacity (kg)	130

Table 5.2: Bed parameters.

5.3 Micro-controller comparison

At the early stages of the project, some of the more common Arduino micro-controllers were taken into account Arduino UNO, and Arduino Mega. [5] [6] However, despite being popular and functional for their purposes, their main drawback for this particular project was small SRAM and onboard flash memory.

Model	Advantages	Disadvantages
Arduino Mega		8KB SRAM + 256KB Flash
Arduino Nano 33 BLE	264KB SRAM + 1MB on-board Flash Inbuilt Bluetooth Low Energy	
Raspberry Pi Pico	264KB SRAM + 2MB on-board Flash	
Arduino Nano 33 BLE Sense rev2	264KB SRAM + 1MB on-board Flash Inbuilt Bluetooth Low Energy Inbuilt sensors (incl. microphone)	not compatible with Arduino Cloud
Arduino Nano RP2040 Connect	448 KB ROM, 520KB SRAM, 16MB Flash Inbuilt Wi-Fi and Arduino Cloud compatibility Inbuilt sensors (incl. microphone)	Not available on demand on Czech market

Table 5.3: Microcontrollers comparison.[51] [52] [53] [54] [55]

Arduino Nano 33 BLE Sense rev2 has been selected as a microcontroller with optimal data, an inbuilt microphone, large memory storage, availability on the Czech market, and personal familiarity with Arduino products rather than Raspberry Pi ones. One drawback is that the selected microcontroller is not compatible with Arduino Cloud, a service for IoT devices, that provides extended functionality for building, controlling, updating, and monitoring IoT projects using the cloud service. [56]

5.4 Hardware components

The main component of this project is an Arduino Nano 33 BLE Sense rev2. The nRF52840 microcontroller features an ARM Cortex-M4 processor. [53]

nRF52840 characteristics

- Operating Voltage: 3.3V
- Input Voltage limit: 21V
- Clock Speed: 64MHz

- CPU Flash Memory: 1MB
- SRAM: 256KB
- Digital Input / Output Pins: 14
- PWM Pins: all digital pins
- USB: Native in the nRF52840 Processor
- Size: 45 mm x 18 mm
- Weight 5 gr (with headers)
- builtin LEDs, microphone, gesture, light, proximity, and color sensor; barometric pressure sensor; temperature and humidity sensor

5.4.1 BLE technology

Bluetooth Low Energy (BLE), also known as Bluetooth Smart, is a wireless communication technology designed for short-range data exchange between devices with low power consumption. [57]

In BLE technologies, two types of devices are assigned a particular role and responsibilities: the central and the peripheral devices. The central device acts as a client and scans for broadcasting devices. Once the central device detects advertising, a connection is to be established information from connected devices, while the peripheral advertises the Bluetooth service and broadcasts parameters, thus acting like a server.

The services can either be documentation defined 16-bit UUID or a user may create their own using 128-bit UUID. Along with the services come characteristics. They represent the data as readable characteristics from sensors that can be either read by the client or their values could be rewritten to control actuators. The four actions that the central can do on a characteristic are read, write, indicate and notify. These actions are specified during service declaration.

The **read** function simply provides the value when the central node asks for it. It is usually used on values that do not change often.

The **write** allows the central to modify the value and thus commands the peripheral, for instance, to change the state of an actuator.

The **indicate** and **notify** urge the server to send the values once they were updated without the central explicitly asking for it.

GATT server is a database where data that needs to be transmitted from one node to another is placed. On the receiving end, the data is then moved to GATT client. Both GATT databases are independent of the central or peripheral role and are determined by the particular application. [57]

The Attribute Protocol (ATT) manages the communication of data between devices by defining how data is structured and accessed. The Generic Access Profile (GAP) handles device discovery, connection establishment, and the roles and behaviors of BLE devices, such as advertising and scanning. [57]

BLE uses Frequency-Hopping Spread Spectrum (FHSS) like the classical version. It operates in the 2.4 GHz ISM band and is divided into 40 channels with a channel spacing of 2 MHz. The data rate of BLE varies depending on the version and configuration, with typical rates ranging from 1 Mbps to 2 Mbps. BLE utilizes Gaussian Frequency Shift Keying (GFSK) modulation to transmit data, allowing for efficient communication while minimizing power consumption.

BLE technology is widely used in various applications such as smart home devices, wearable technology, healthcare sensors, and asset tracking due to its low power consumption and short-range communication capabilities. BLE enables efficient data transmission over short distances, making it ideal for small devices or sensors and intermittent communication.

Possible attainable throughput in an error-free scenario using BLE technology is 237.712 kbit/s, as achieved in [58]. For a better understanding of the BLE technology for troubleshooting during project implementation, the layer hierarchy inside the stack as well as the role of characteristics and services of the GATT have been studied more carefully: [57]

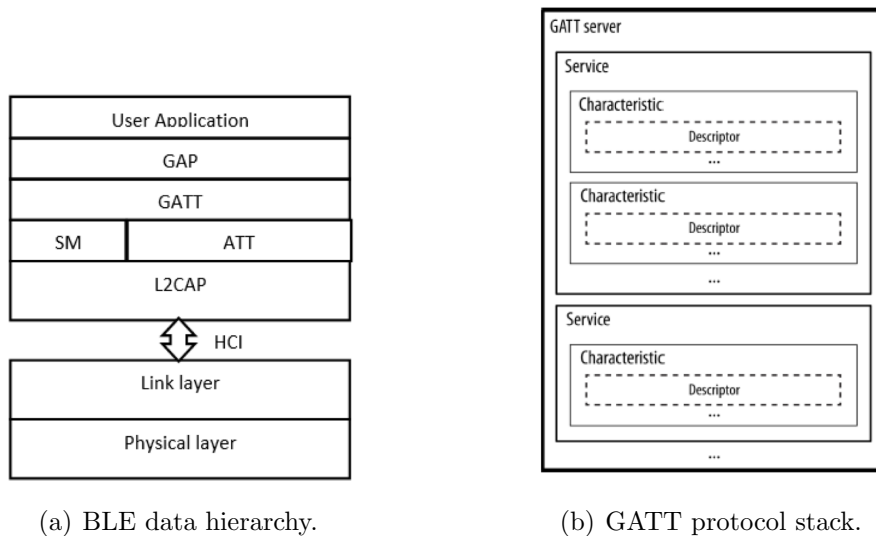


Figure 5.2: BLE technology.

5.4.2 OTA update

Over-The-Air (OTA) provisioning allows edge devices to download updates despite their limited capabilities. Most of the wearable products and IoT devices are not programmable which creates difficulties for carrying out updates and changing their services with time. [59]

For further training of the device, new data is to be collected with the correct label from devices on-field. The data is then transmitted into the Edge Impulse API, where the administrator indicates the components that require updating and uses new data for that. This process could potentially be automated to train on all new data. The

quality of the data has to be ensured. After that, the model undergoes retraining, testing, and validation on the server endpoint. Then, a new change is detected by the device. Edge Impulse API allows the device to send a request about the availability of a newer version. An indicator could be the improved performance metrics or a more recent last modified date. More metrics can be checked to check the validity of the version update. Next, the updated components are deployed onto the device in the binary firmware image form using available OTA technologies. As the last step, the device is monitored to assess its performance with the new updates to ensure that the desired functionality is achieved. [60] Nordic Semiconductors provides Device Firmware Update (DFU) Service for its nRF5X microcontrollers, which is a solution that requires a .zip file with the update and a compatible device for an easy update transfer. [61]

Edge Impulse's integration with OTA streamlines the process, enabling users to remotely deliver updates to their devices, ensuring the microcontroller is continuously upgraded with the latest features and improvements.

Chapter 6

Implementation

The project consists of three main parts. First is a user interface in the form of a web application that can be accessed from any device using the local Wi-Fi network. There user can record commands with a particular label to differentiate between the intended function on the Edge impulse platform. The second part is the language model that converts the speech into executable commands. In the case of this project, it is a platform for edge devices machine learning that is remarkable with its flexibility and ease of approach for a user or developer, who does not specialize in deep learning algorithms. The third part is the hardware implementation. Without this part, this thesis would be theoretical and be of little value in real-world applications.

Some of the key performance markers on the device are latency, RAM, and flash memory. From the user's perspective reliability, accuracy rate, speed of the command implementation and overall robustness are key factors to assess the experience with the bed.

The measurement of accuracy is calculated as the following: [62]

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

6.1 Language model approaches

Automatic Speech Recognition is a general term for an extensive field of research and technologies. [63] A system with ASR takes as an input the audio recording or direct speech stored in buffers from the microphone and converts it into recognized text or commands. Human speech however comes in a variety of patterns, pitches, and accents, which is more so applicable to people with disabilities. According to [64], speech recognition can be divided by utterance approach, utterance style, type of speaker, vocabulary, and channel variability. Thus, the approach in this case can be classified as speaker-dependent continuous speech in isolated words with mild channel variability with small or tiny vocabulary size. As for classification techniques, an artificial intelligence approach is used. It is a hybrid of the two predeceasing approaches: acoustic-phonetic and pattern recognition. [65]

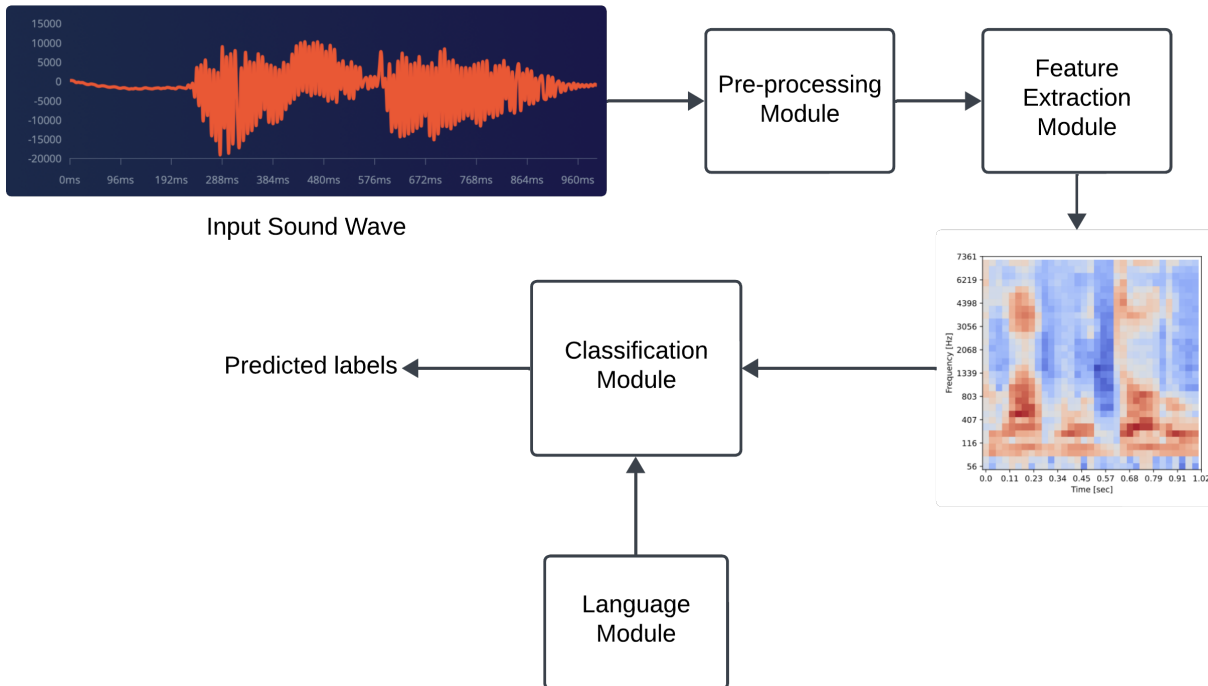


Figure 6.1: ASR diagram.[63].

Keyword spotting can be approached by two methods: Mel Frequency Cepstral Coefficients (MFCC) and Mel-filterbank energy features (MFE). MFE extracts features from the audio file in the Mel-scale frequency and time domain. Mel-scale frequency is a non-linear scale that is frequently used for audio data. In this scale, listeners perceive pitches to be equally spaced apart. It's anchored at 1000 mels for a 1000 Hz tone, 40 dB above the listener's hearing threshold. [66] Listeners perceive larger pitch intervals as the frequency goes beyond 500 Hz. It performs well with non-voice sounds and is designed to distinguish and classify sounds like the human ear does. Our goal is to extract more features, or so-called filter banks, in the lower frequencies and fewer in the higher frequencies, as it aligns with the sensitivity of the human ear.

After the spectrogram is computed, frequency bands are extracted with the help of triangular filters that are applied on the Mel scale. Its parameters and their meaning are shown below:

6.1.1 MFE parameters

Mel-filterbank energy features [67]

- Number of coefficients
- Frame length (s): the length of each frame
- Frame stride (s): the step between two frames
- Filter number: indicates the number of triangular filters that are applied to the spectrogram

- FFT length: the size of the FFT
- Low and high frequency: lowest and highest band edge within the Mel-scale filterbanks

Normalization

- Noise floor (dB): signals lower than this level are not processed

In the last step, the noise reduction occurs.

6.1.2 MFCC

MFCC similarly to MFE extracts time and frequency features in the Mel-scale and preserves all the parameters for extraction and filtering. After obtaining the features, an extra step of compressing the filterbanks is taken using the application of the Discrete Cosine Transform. Thus, cepstral coefficients are obtained. Usually, 13 coefficients are used, while others are dropped.

Mel Frequency Cepstral Coefficients [68]

- Number of coefficients: number of cepstral coefficients that are kept after the DCT
- Frame length (s): the length of each frame
- Frame stride (s): the step between frames
- Filter number: indicates the number of triangular filters that are applied to the spectrogram
- FFT length: the size of the FFT
- Low and high frequency: lowest and highest band edge within the Mel-scale filterbanks
- Window size:

Pre-emphasis

- Coefficient: the pre-emphasizing coefficient that defines a filter applied to the input signal (0 means no filtering)

6.2 Model training and library creation

6.2.1 Model optimization

Both MFCC and MFE models were tested. A common notion is that MFCC is better suited for human voice recognition. [68] However, developers of the platform advise to try out each model and see which one works best. Settings and parameters also affect the resultant performance of each model. The two trials shared the same set of recorded files.

Testing of the device can be done without connection to the bed, thus making it easier to optimize the model. The parameters in both future generation and neural network architecture were varied to discover the best model. Some parameters did not show any positive change on the accuracy and were not considered in the analysis. In feature generation frame length and frame stride were varied. FFT length is dependent on the frequency of the audio and frame length and has the following relationship: [69]

$$FFT \geq f \times frame_{stride}$$

Thus, for 16000 Hz audios, an FFT of 512 and frame stride of 0.032 is considered optimal. Indeed, it shows the best performance based on several variations.

6.2.2 Neural network architecture

The neural network architecture created is using Keras API. [43] 992 features are input into the neural network, which is presented as follows:

- Reshape layer: the input data is reshaped to fit the convolutional layers. This reshaping operation organizes the input into a 2D format with dimensions determined by the input length, number of columns, and number of channels. This step prepares the data for processing by the convolutional layers.
- Convolutional layers: three convolutional layers are applied successively. Each convolutional layer performs a series of operations on the input data using filters (also known as kernels). These filters extract features from the input data through convolution operations. In this architecture, each convolutional layer has 16 filters, and each filter has a kernel size of 3x3. The activation function used after each convolution operation is ReLU (Rectified Linear Unit), which introduces non-linearity into the model, allowing it to learn complex patterns in the data.
- Max Pooling layers: after each convolutional layer, a max pooling layer is added. Max pooling reduces the spatial dimensions of the feature maps produced by the convolutional layers, effectively downsampling the data. In this architecture, max pooling is performed with a pool size of 2x2 and a stride of 2, which means that the maximum value within each 2x2 window is selected, and the window moves by 2 pixels at a time. This helps in reducing the computational complexity of the model and making it more robust to variations in input.
- Dropout layer: to prevent overfitting, a dropout layer is added after the max pooling layers. Dropout randomly sets a fraction of input units to zero during training, which helps prevent the model from relying too heavily on any specific set of features.
- Flatten layer: after the dropout layer, the feature maps are flattened into a 1D vector. This flattening operation converts the spatial dimensions of the feature maps into a single vector, which serves as the input to the subsequent fully connected layers.

- Dense layer: finally, a dense (fully connected) layer is added. This layer consists of neurons that are connected to every neuron in the previous layer. The number of neurons in this layer is determined by the number of output classes. The activation function used in this layer is softmax, which converts the raw output scores into probabilities corresponding to each class.

6.2.3 Neural network settings

Here are presented the settings of the neural network that a user can change. [70] Along with the simplified user-friendly parameter setting, an expert version with code can be accessed.

- Use learned optimizer: VeLO (Versatile Learned Optimizer) that calculates gradients and learning rates
- Learning rate: a rate that indicates how much the model updates its parameters with each epoch. May help in case the model overfits during training (performs well on a known dataset, but struggles with the new data). This optimizer is available with an Enterprise plan
- Validation set size: a share taken from a training set for validation of the data
- Split train/validation set on metadata key: excludes the data leakage between training and validation datasets based on metadata. This parameter ensures that the model deals only with unseen data during validation.
- Batch size: Represents a number of samples processed in one epoch before updating the model's parameters
- Auto-weight classes: In case of an imbalanced dataset, where some classes have significantly fewer samples than others, assigns higher weights to these under-represented classes during training. The loss contributed by these samples has a larger impact on the overall training process.
- Profile int8 model: Enabling this option initiates a process called profiling, which analyzes the dataset to optimize quantization parameters. However, profiling can be time-consuming, particularly on large datasets.

Processing block type	Neural network layers	Window increase (ms)	Frame length/ frame stride (ms)	Accuracy	Conclusion
MFCC	2x1D	1000		80.6%	
	2x1D	200		88.1%	
	2x2D	200		84.3%	
	dropout rate=0.15	200		86.6%	Accuracy above 90% was not achieved, poor on-device performance; MFE is selected as a processing block
MFE	2x2D	200		88.3%	MFE is a more suitable model for the needs of the project
		500		89.1%	
		700		90.3%	
		800		90.3%	window size = 700 ms is optimal
	inclusion of VeLO layer			83.5%	VeLO is not improving accuracy
	2x2D, dropout rate =0.45	700		88.4%	drop rate = 0,5 is optimal
	2x2D, dropout rate = 0.5	700	0.03/ 0.003	85.6%	
	2x2D, dropout rate = 0.5	700	0.01/ 0.01	87.1%	
	2x2D, dropout rate = 0.5	700	0.032/ 0.032	88.2%	Frame length and frame stride of 0.032 ms are optimal
	1x2D with 32 filters and 1x2D with 8 filters with 1 dropout layer	700		81.5%	32 filters in one of the convolutional layers decrease accuracy
	2x2D with 8 filters	700	0,032/ 0,032	85.4%	
	2x2D with 16 filters with 2 dropout layers	700		88.8%	16 filters in both convolutional layers increase accuracy
	3x2D with 16 filters each	700		90.3%	three convolutional layers are optimal

Table 6.1: Model optimisation steps.

6.2.4 Commands

In this project, two approaches to the language model were pursued. One was the whole command uttered by the speaker, while the other one included separate words, a combination of which is to be processed by the microcontroller and a corresponding function to be executed. Thus, two versions of the project were created and specific commands were recorded. The performance of the two approaches can be assessed based on the accuracy of the keyword spotting as well as the actual performance on the Arduino. The advantage

of the approach with separate words is a smaller size window for the word and thus easier auto sampling of larger audio recordings. The command consisting of two words might not always fit into the one-second window and important parts of the sample would be cropped out.

There are two ways to sample the audio: continuous and non-continuous. In the conventional non-continuous procedure of executing the model, the data collection and inference processes are executed sequentially. Initially, audio is sampled, filling a block of the model's size. This block is then forwarded to the inference module, where the features are extracted and inference is performed. Subsequently, the classified output is utilized within the application, typically displayed via the serial connection.

On the other hand, in the continuous sampling approach, audio sampling is conducted concurrently with the inference and output procedures. This means that while inference is ongoing, audio sampling persists as a background process. In the continuous mode, the device listens to commands and divides the window into several slices while at the same time analyzing another chunk of audio file. The disadvantage is that the commands with the same beginning might get mixed up depending on the moment the window is sliced.

Model Slicing: Utilizing continuous inferencing, the input data undergoes segmentation into smaller slices, which are sequentially processed over time. These slices are organized in a FIFO (First In First Out) buffer to accommodate the model's size requirements. Following each iteration, the oldest slice is purged from the buffer, while a fresh slice is inserted at the forefront. Each slice undergoes multiple inference runs, determined by the designated number of slices for the model. For instance, in a 1-second keyword model with 4 slices (each lasting 250 ms), each slice is subject to four inference passes. Consequently, if the keyword spans across two edges of the slice buffers, they are reconstituted within the FIFO buffer, facilitating precise keyword classification.

Averaging: Another notable advantage of this approach lies in its ability to mitigate false positives. Consider a scenario with a keyword detection model for "hlavudolu" and "hlavunahoru". It's imperative that "hlavudolu" not be misclassified as "hlavunahoru" or vice versa. However, if the "hlavudolu" portion is captured in one buffer and "nahoru" in the next, there's a risk of erroneously classifying the first buffer as "hlavudolu". By subjecting the slices to multiple inference iterations, continuous inferencing effectively filters out such false positives. As the "hlavudolu" buffer enters the FIFO, it confidently registers as a positive classification. Subsequently, as the remainder of the word is processed, the classification for "hlavudolu" rapidly diminishes. To ensure accurate classification, a moving average filter smooths the classified output, attenuating spikes. Thus, a valid "hlavudolu" requires consistent high-rated classifications across multiple iterations.

6.2.5 Training the model

As can be seen above, many parameters take part in the model's overall performance and accuracy. The complexity and dependence on coefficients are not straightforward, thus, many combinations and variations had to take place. Several markers measure the accuracy and functionality, such as false positive, false negative, true positive, and true negative. In the neural network design, your extracted features serve as inputs, passing

through each layer of the architecture. In classification tasks, the final layer typically consists of a softmax layer, which calculates the probabilities of belonging to different classes.

In the table below the number of utterances for training and testing sets are presented. Some commands required less training to be recognized, while others needed more audio recording time. Besides the speaker's voice recordings, two other classes were included in the training: 'background noise' and 'unknown'. The 'background noise' helps the model ignore audible events that take place around, such as working TV, faucet running, humming, or buzzing. The 'Unknown' category includes random words uttered by different people to reduce the likelihood of the model reacting positively to an unknown command or conversation happening around it. The number of utterances is extremely small in our case. The intent was to find out the minimum number of voice recordings to provide decent results. Normally, models are trained on hours of data. [71]

Command	Translation	Training Set	Testing set
AhojZoro	Hey, Zoro	51	3
HlavuNahoru	Head segment up	71	14
HlavuDolu	Head segment down	93	12
NohyNahoru	Leg segment up	56	13
NohyDolu	Leg segment up	60	15
PostelNahoru	Bed up	69	27
PostelDolu	Bed up	33	6
SlozSe	Fold	40	5
RozlozSe	Unfold	29	8
Trochu	A bit	40	8
Uplne	Fully	49	10
Stop	Stop	100	27

Table 6.2: Commands and number of utterances.

6.3 Algorithm description

Arduino Integrated Development Environment (IDE) is the interface for writing the code for the microcontroller Nano BLE 33 Sense Rev2. It uses a simplified version of C++ programming language to create so-called Arduino sketches. Arduino IDE provides an interface for writing, compiling, verifying, debugging, and uploading code to Arduino boards. The libraries and functions are added upon necessity.

Firstly, the libraries used are imported: ArduinoBLE.h, PDM.h and EdgeImpulse.h. They provide functions and variables for the BLE service, inbuilt microphone manipulations, and language model respectively. The main library is generated using the Edge Impulse platform and includes the neuron network and DSP digital-signal processing. The device starts BLE service advertising and listens to the wake-up word. DSP takes care of filtering and feature extraction, while MFCC turns audio data into a series of spikes based on frequency. After the trigger word is recognized, other commands to adjust the

bed become available. The model goes through the loop where the audio sample in the buffer and its recognized label with a probability above a certain value is compared to the labels to perform distinctive functions.

To manage incoming sample data, a double buffering mechanism is employed, utilizing two buffers: one for sampling and another for inference classification. Initially, the sampling process fills one buffer while the inference process awaits its completion. Upon buffer completion, the sampling process passes it to inference and begins filling the second buffer. This cycle ensures uninterrupted operation. Timing and memory are critical considerations, necessitating careful buffer management. On Arduino Nano 33 BLE Sense Rev2, maintaining timing is crucial to avoid buffer overruns and data loss. Adjusting parameters such as slice size impacts both memory usage and sampling cycle duration. The buffer switching process is implemented through callback functions, ensuring seamless data transfer between sampling and inference processes. This meticulous process guarantees reliable performance in real-time applications.

Upon positive classification, the user receives the information on their central device, a smartphone. In case the classification is recognized as the one intended, the user sends the confirmation to the device, which activates the necessary pins and subsequently the relays and the motors. At the end of each loop, the program compares the time that the motors have been activated to the required time and turns them off once the time limit is overcome. In the earlier stages of the project, instead of this condition checking, a delay was used, however, this led to buffer overrun as it added time to the processing of the sample and the system was not accepting further commands such as 'stop' before the delay was over. The loop usually lasts for a few hundred milliseconds, which gives enough precision for firing and turning off the motors.

The degree to which the head or the leg sections are adjusted can be varied by timings when the pins are activated. Commands 'uplne', or 'fully', and 'trochu', meaning 'a bit', change the time that motors are activated. These commands as well as the 'stop' do not require a confirmation from the user.

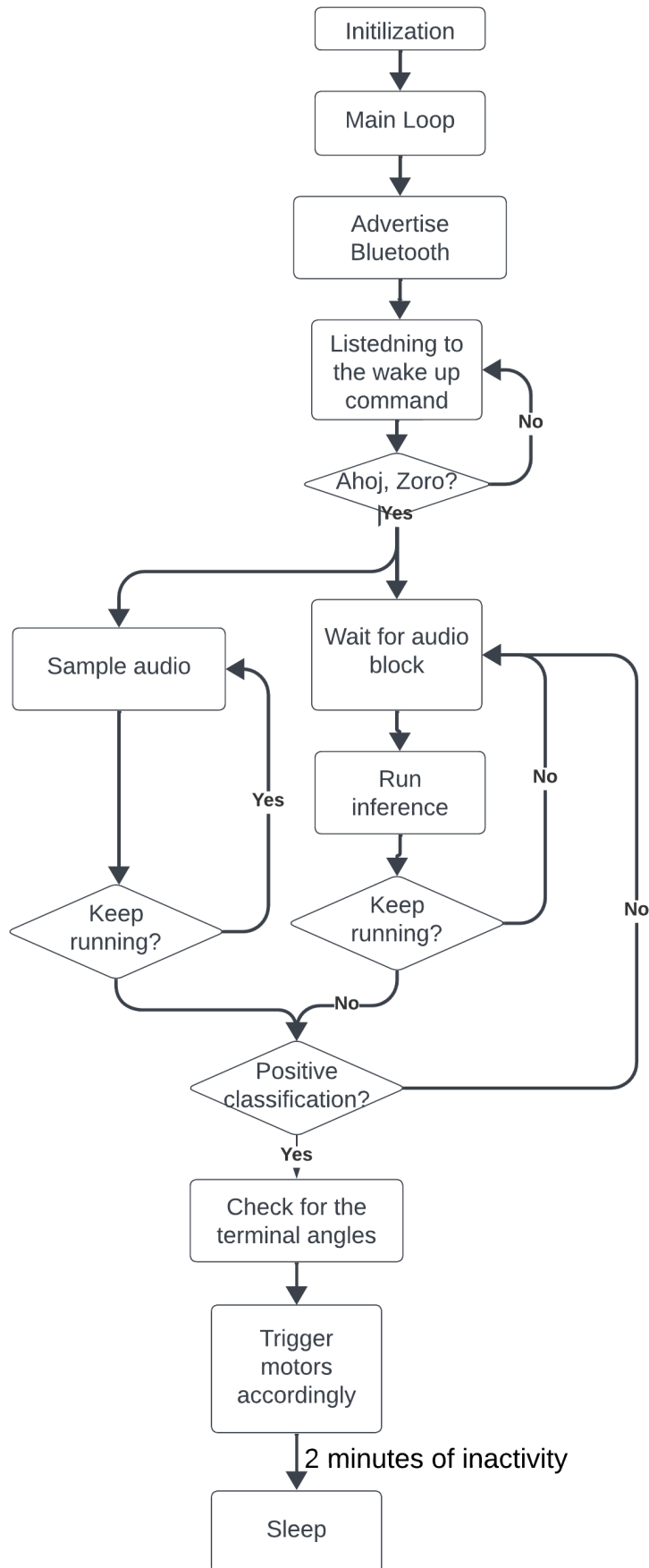


Figure 6.2: Algorithm with double buffering.

6.4 Code examples

Here, the logic of one of the commands. Others are made similarly, except for folding and unfolding commands triggering two motors instead of one.

```

if (result.classification[ix].value >= 0.9) {
    if (result.classification[ix].label == "hlavunahoru") {
        txChar.writeValue("hlavu_nahoru"); //send command to the master
        status = 1; //each command
        ei_printf("hlavu_nahoru_%d\n", hlava);
        positiveClassification(); //LED blinking
    }
}

// Execute action based on status
if (status == 1 && rxChar.value() == "ano") { //requires confirmation of
    // Move head up
    if (hlava >= 1 && hlava <= 24) { // 1 is for the rest
        //position and 24 is the maximum possible position on top
        startTime1 = millis(); //indicate the time when the motor
        //was triggered
        digitalWrite(RELAY1_PIN, LOW); // Activate the motor
        hlava = hlava + DELAY_TIME / 1000; // updates the
        //position of the segment
    }
    rxChar.writeValue("ne"); //condition for user confirmation
    status = 0; // resets the command
}

// Deactivate the motor after the specified delay
elapsedTime1 = millis();
if (elapsedTime1 >= startTime1 + DELAY_TIME) {
    digitalWrite(RELAY1_PIN, HIGH); // Deactivate the motor
    elapsedTime1 = 0; //reset of the elapsed time
    startTime1 = 0; //reset of the start time
    Delay_time(3000); // reset of the DELAY_TIME
}

```

6.5 Web application

A web application is a tool for an end user to record their commands, apply labels to them, and send them to the project repository for future training and model modifications. For simplicity and cross-platform availability, a web application was created. It can be accessed from all devices without the need to download any additional software. Edge

Impulse platform provides a powerful API for these purposes. The mobile client is open-source and can be accessed easily from any device by scanning the QR code or following the link to the project from any mobile device.

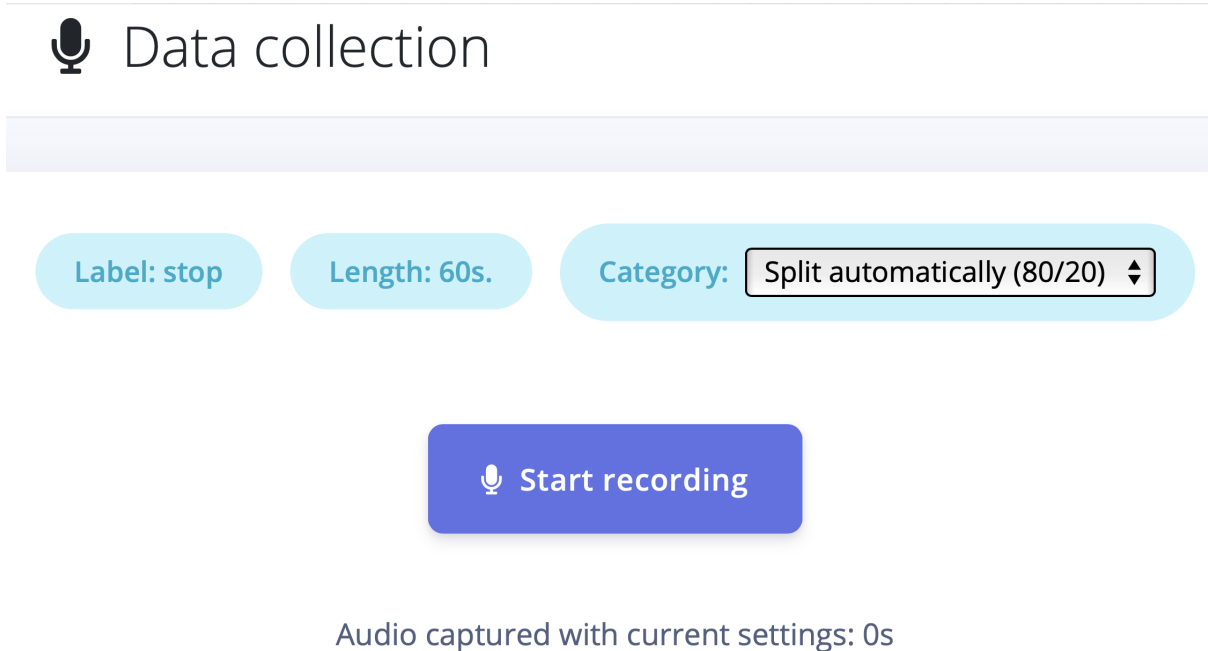


Figure 6.3: Data collection window.

6.5.1 Audio recording

Edge Impulse has a special format for audio files that are being uploaded. Not all audio recording libraries support such a format and thus are not suitable for this application. Audio recordings have to be in WAVE file format with a particular structure. Each file starts with a RIFF sub-chunk followed by information about the audio such as sample rate, and raw audio data. Without this structure Edge Impulse rejects the audio file. Widely used MediaRecorder web API does not follow this structure regardless of recording a .wav file, thus was not suitable for this application. Another library that was created 9 years ago, does fulfill the requirements and was subsequently used in the web application. [72] Audio files are advised to be recorded for about 20 to 60 seconds and then split into one-second-long utterances. This could be performed either on the Edge Impulse platform itself or in a general audio editing desktop application like Audacity. [73]

Field name	Field size	
Chunk ID	4	The "RIFF" chunk descriptor, determines the "WAVE" format
Chunk size	4	
Format	4	
Sub-chunk1 ID	4	The "fmt" sub-chunk describes the format of the sound information in data sub-chunk
Sub-chunk1 size	4	
Audio format	2	
Num Channels	2	
Sample rate	4	
Byte rate	4	
Block align	2	
Bits per sample	2	
Sub-chunk 2 ID	4	
Sub-chunk2 size	4	
data	subchunk2 size	

Table 6.3: WAVE file data structure.

6.5.2 Uploading to Edge Impulse API

For audio uploading, Ingestion API is used. After recording the audio, the uploading function takes care of the file upload. A binary large object (BLOB) is formed with HMAC encoding, label information, frequency, device name, type, sensor used attached to the payload. An API key and an HMAC key are required for a given project to securely upload the data.

Edge Impulse API provides a large functionality besides uploading the data. It also handles login tasks, changing labels, training, and deploying models using HTTP requests. This functionality is to be used in the later development of the project to automate tasks.

6.6 Hardware connection

The bed is as described in the previous section. Regarding its notable features, two motors supporting head and leg adjustment are powered by 24 V, whereas the height-controlling motor is powered by 40 V. Thus, the remote control includes some more complex circuitry

and circuit protection that can be seen with an unarmed eye. The medical bed is normally operated with a wired remote control. There are 6 buttons for operating the bed. By taking a closer look at the circuitry of the remote control, the common ground and 7 wires were observed. By short-circuiting the ground and one particular wire, one of the three motors is turned into motion, and position is adjusted as long as the button is pressed, or in other words, the ground and position wire are connected. Another advantage of using the bed- is three motors that can be simultaneously used for folding up and unfolding commands. Wiring includes six relays that send signals from the Arduino microcontroller to the control unit.

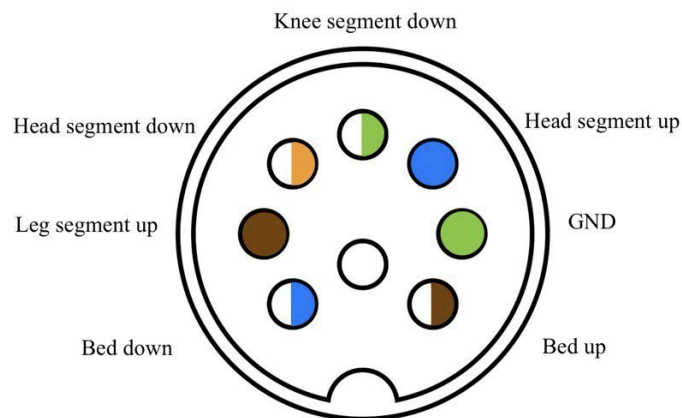


Figure 6.4: 8-pin connector.

- Arduino is powered by the power outlet
- The digital pins D2, D3, D4, D5, D6 and D9 of Arduino are connected to inputs on the low-voltage side of the three logic converters
- Outputs on the high-voltage terminal are connected to In1, In2, In3, In4, In5 and In6 of the relay module to control different motors respectively
- 12V DC power source is connected to relay
- 5V and 3.3V DC are connected to the appropriate terminals of the converters. In this project, another Arduino UNO is used as a power source
- All the parts of the circuit are sharing GND (ground)

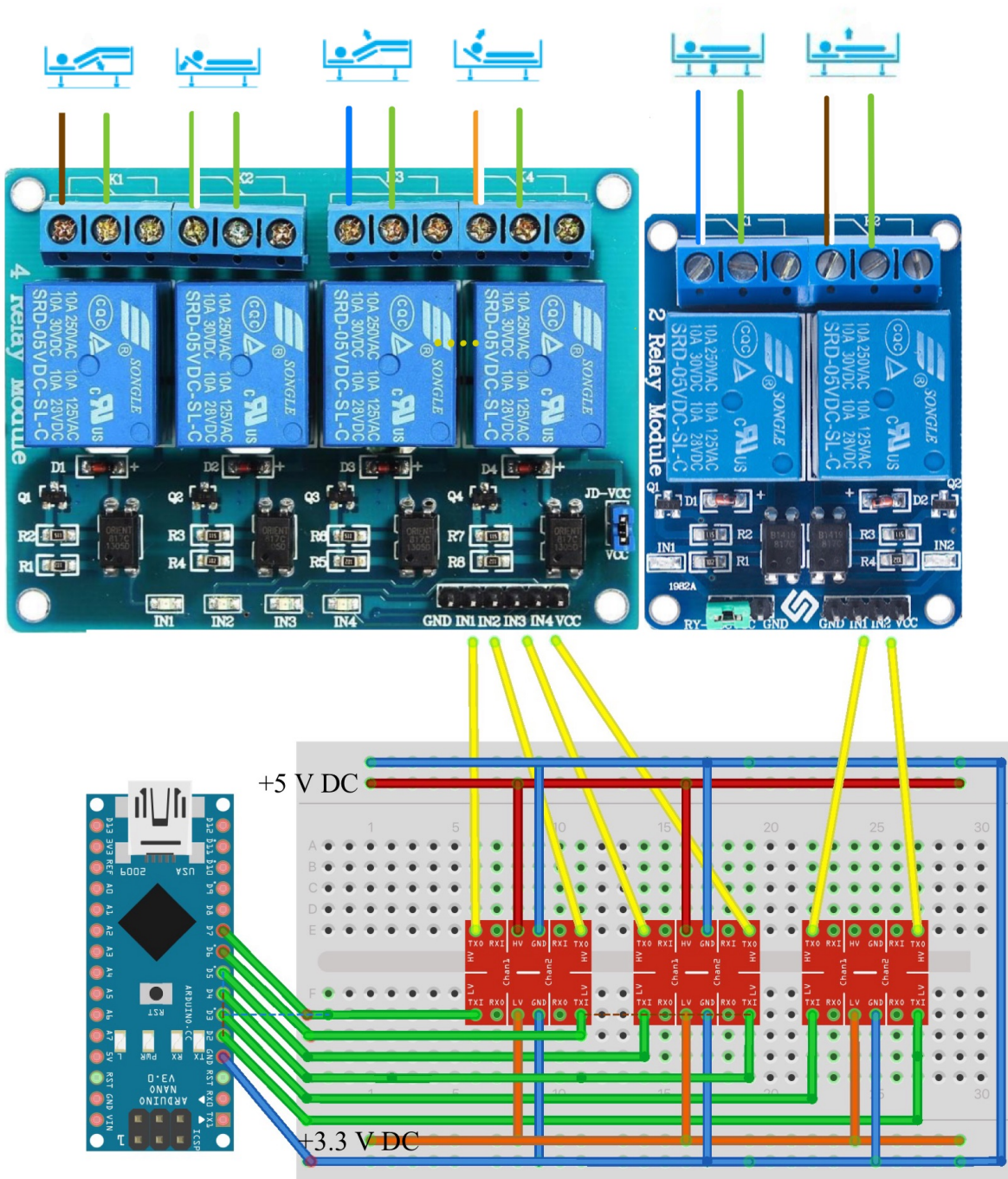


Figure 6.5: Hardware connection schematics.

Several approaches to the hardware implementation have been made. In the original design, the control of the motors was based on motor drivers. However, the bed configuration proved to be controlled more easily through the remote control cable. Since the bed was commercially available and was not provided with a detailed manual of the electrical circuit structure, some reverse engineering techniques had to be applied. From the uncovering of the remote control, several points were revealed. To activate one of the functions,

Unlike some other Arduino microcontrollers, Nano Sense has an output signal voltage of 3.3 V, thus not enough to activate the relay. To overcome this issue, two 2-channel logic converters were utilized. Each such converter has one higher voltage (HV) side that is powered by 5 V DC and a lower voltage (LV) side that is connected to 3.3 V DC. Input signal is wired to the TX1 terminal and the converted output signal switches the relay ON and OFF. The relay used in the project is a 4-channel reversed-logic relay operated at a 5 V DC input signal. Each relay is optically separated from the rest. Besides the relay, the module is equipped with a flyback diode for circuit protection, resistors, a transistor, and LEDs that signal and input for each relay.

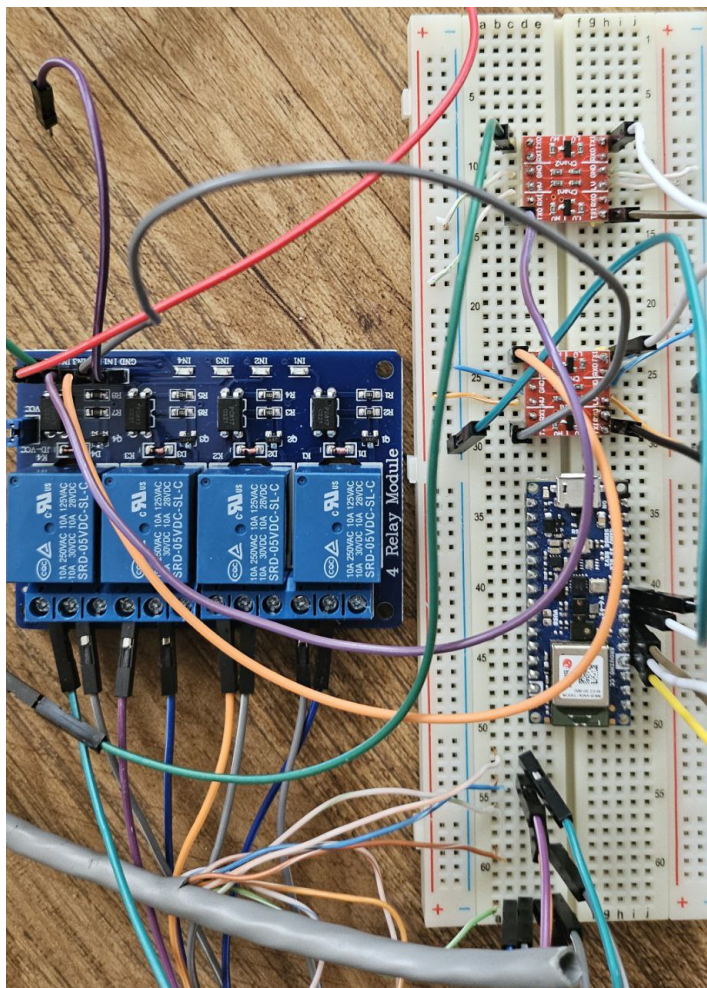


Figure 6.6: On-device wiring.

Chapter 7

Testing

The testing phase is crucial to ensuring the success of this project. This chapter delves into the detailed evaluation of the language model and microcontroller system, focusing on key performance metrics such as accuracy, responsiveness, timing precision, safety compliance, and command execution smoothness. By thoroughly testing these aspects, the aim is to confirm the system's reliability and effectiveness. This chapter presents the testing conditions, methodologies, and results, highlighting the steps taken to optimize performance and achieve the project's goals.

7.1 Performance requirements

The main parameters for testing are the language model accuracy, responsiveness of the microcontroller, adequate timings, prevention of exceeding a certain terminal angle for safe operation, and smooth command execution. This includes an ongoing bed motors operation for several cycles without error in the logic of the program. All the variables have to be updated as intended to ensure that timing and terminal angles of inclination requirements are met. To deem the project as a successful one, a certain degree of reliability and accuracy has to be achieved. From a user's perspective, an accuracy of at least 70 % would be considered as satisfying. Timing of the application in terms of audio buffering and classification execution has to be small enough for the user not to notice any delays.

7.2 Test conditions

Since this model is a speaker-dependent one, only one person whose voice is recorded can perform the testing. The device got the model uploaded. Using a Bluetooth application installed on a smartphone, the connection was established. For such purposes, any Bluetooth application would suffice. The one that was used in this project was nRF Connect, developed by Nordic semiconductors. [61] The normal usage conditions are not extremely noisy environments with relative proximity of the user's mouth to the microphone for capturing audio.

When the device has an uploaded code and is turned on, the initiation of the microphone and BLE services is completed. Both true positives and false positives are subject to judging the accuracy of the model. The testing also ensured, that the maximum angles of inclination for head and leg segments are controlled by the variables in the code are not exceeded as a part of safety measures for both the user and bed structure.

The resulting parameters for the model were as follows:

Mel-filterbank energy features

- Number of coefficients
- Frame length (s): 0.032
- Frame stride (s): 0.032
- Filter number: 32
- FFT length: 512
- Low and high frequency: 0

Normalization

- Noise floor (dB): -72

The resulting NN architecture is presented below:

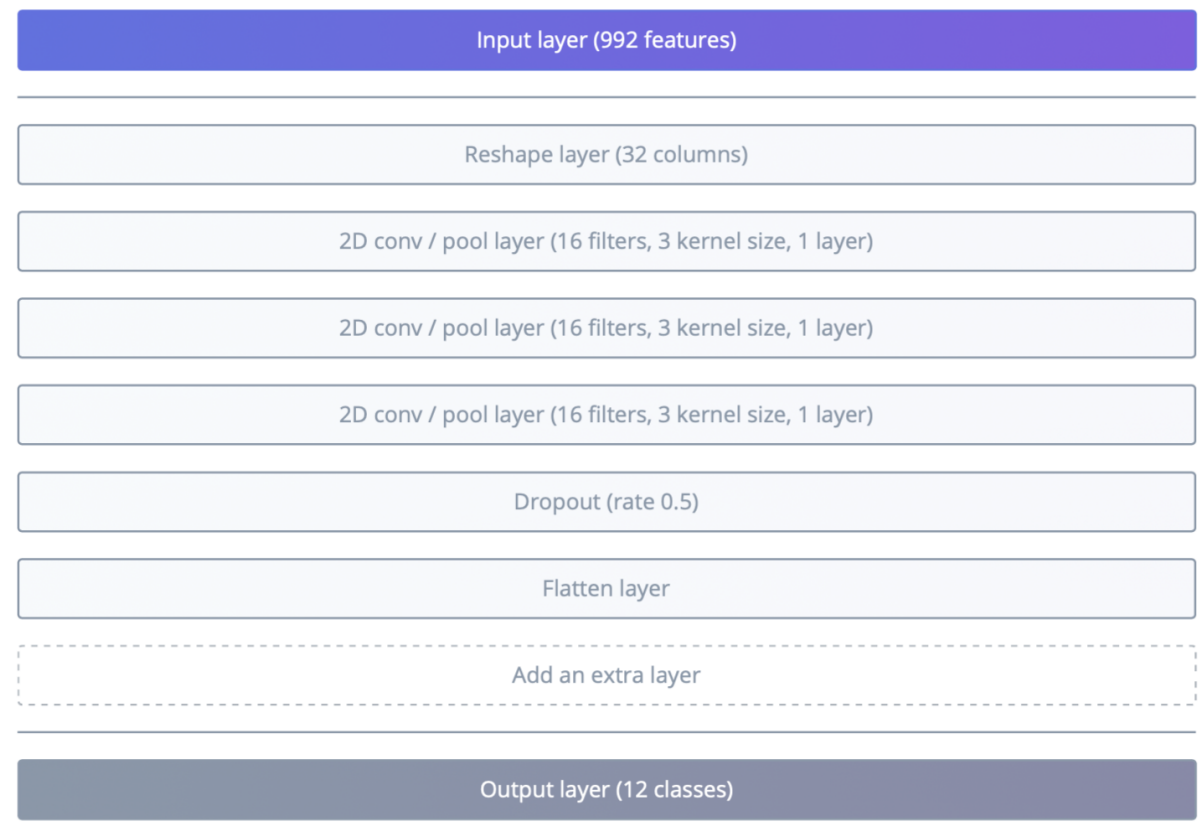


Figure 7.1: Neural network architecture.

7.3 Testing with the user

The testing took place in the laboratory of telemedicine at UCEEB research facilities. Before testing with the user, the responsiveness of the bed to the commands was ensured. The language model showed decent results initially. Subsequently, a person of the same gender and similar age has been selected to test the model, hoping to match the pitch and voice characteristics of the training recordings. As a result, the model proved to be highly speaker-dependent, treating the user's voice as background noise. Variations in accent, speaking speed, and word emphasis resulted in a poor model response.

In an attempt to improve the accuracy, an additional 10 seconds with the voice of the new user for the command "Hlavu nahoru" ("Leg segment up") have been added to the project, new library created and deployed onto the device. Despite these efforts, the model failed to extract the necessary features for the new user, and the results did not improve significantly. The model began to recognize the commands, however incorrectly (false positives). Nevertheless, this signifies the ability of the model to train and it is safe to assume, that if all the commands were recorded at least for 20 seconds, the model would have a higher probability of distinguishing between them. Notably, the commands with more distinctive and unique sounds such as "Slož se" ("Fold") and "Rozlož se" ("Unfold") had a greater success of being recognized correctly.

Another important takeaway was the need for an external microphone. Arduino Nano 33 BLE Sense rev2 was a choice for this project partially because of the inbuilt microphone, which proved to be useful and convenient during development and testing, but not at all suitable for real usage.

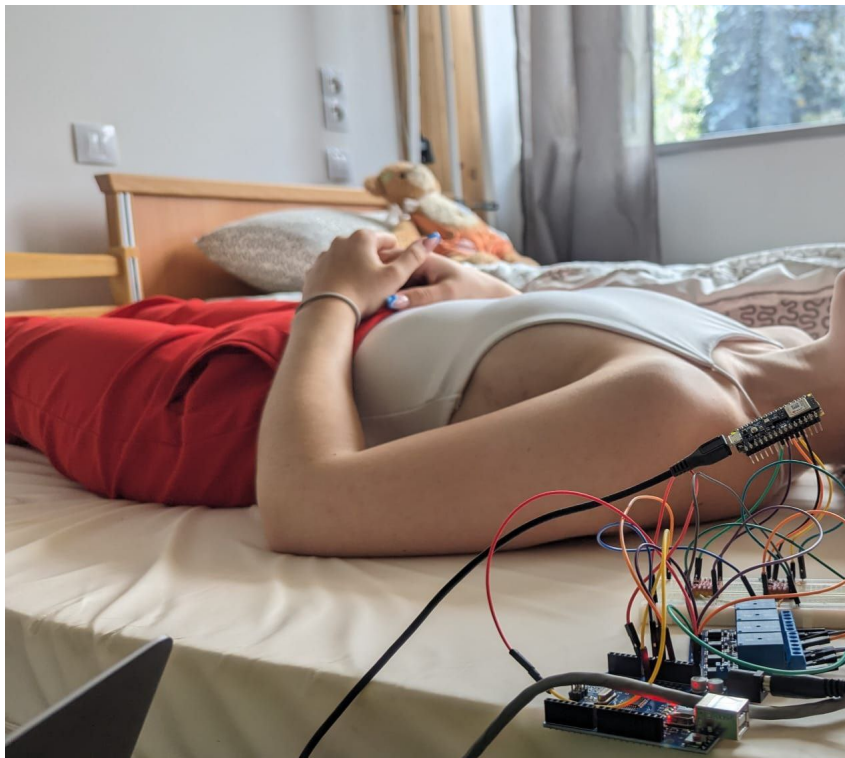


Figure 7.2: Testing with the user.

7.4 Testing results

The model takes about 18 seconds to upload onto the device. The sketch uses 429016 bytes (43 %) of program storage space. Maximum is 983040 bytes. Global variables use 72816 bytes (27 %) of dynamic memory, leaving 189328 bytes for local variables. The maximum is 262144 bytes. The size of one page is 4096 bytes, which results in 105 pages. 429760 bytes are written on the device in 16.7 seconds. Due to the time constraints and unexpected issues with the hardware, the testing has been conducted on one test user. The testing has confirmed the dependency of the model on the speaker, whose voice was recorded for training of the models. Without prior voice recording, the speech recognition model shows poor performance and does not recognize most of the commands or recognize them as faulty.

PL \ AL	Ahoj Zoro	Hlavu Nahoru	Hlavu Dolů	Nohy Nahoru	Nohy Dolů	Postel Nahoru	Postel Dolů	Slož Se	Rozlož Se	Trochu	Úplně	Stop
Ahoj Zoro	71,4%	0%	0%		28,6%	0%	0%	0%	0%	0%	0%	0%
Hlavu Nahoru	0%	100%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
Hlavu Dolů	0%	5,3%	89,5%	0%	5,3%	0%	0%	0%	0%	0%	0%	0%
Nohy Nahoru	7,7%	0%	0%	84,6%	7,7%	0%	0%	0%	0%	0%	0%	0%
Nohy Dolů	8,3%	0%	0%	8,3%	83,3%	0%	0%	0%	0%	0%	0%	0%
Postel Nahoru	0%	0%	0%	0%		90%	0%	0%	0%	0%	0%	10%
Postel Dolů	0%	0%	0%	0%	0%	0%	100%	0%	0%	0%	0%	0%
SložSe	0%	0%	0%	0%	0%	0%	0%	100%	0%	0%	0%	0%
Rozlož Se	0%	0%	0%	0%	0%	0%	0%	0%	100%	0%	0%	0%
Trochu	0%	0%	0%	0%	16,7%	0%	0%	0%	0%	83,3%	0%	0%
Úplně	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	100%	0%
Stop	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	100%

Table 7.1: Accuracy results after training the model.

PL \ AL	Ahoj Zoro	Hlavu Nahoru	Hlavu Dolů	Nohy Nahoru	Nohy Dolů	Postel Nahoru	Postel Dolů	Slož Se	Rozlož Se	Trochu	Úplně	Stop
Ahoj Zoro	71.4%	0%	7.1%	0%	7.1%	0%	0%	0%	0%	0%	0%	14.29%
Hlavu Nahoru	12.5%	75%	0%	12.5%	0%	0%	0%	0%	0%	0%	0%	0%
Hlavu Dolů	0%	20%	80%	0%	0%	0%	0%	0%	0%	0%	0%	0%
Nohy Nahoru	0%	0%	0%	85.7%	0%	0%	0%	0%	0%	14.3%	0%	0%
Nohy Dolů	0%	0%	0%	0%	75%	0%	0%	0%	0%	0%	0%	25%
Postel Nahoru	0%	33.3%	0%	0%	0%	66.7%	33.3%	0%	0%	0%	0%	0%
Postel Dolů	0%	0%	0%	0%	0%	33.3%	66.7%	0%	0%	0%	0%	0%
Slož Se	0%	0%	0%	0%	0%	0%	0%	55.6%	0%	0%	0%	33.3%
Rozlož Se	0%	16.7%	0%	0%	0%	0%	0%	0%	100%	0%	0%	0%
Trochu	20%	0%	0%	0%	0%	0%	0%	0%	0%	60%	20%	0%
Úplně	0%	0%	33.3%	0%	0%	0%	0%	0%	0%	0%	66.7%	0%
Stop	0%	25%	0%	0%	0%	0%	0%	0%	0%	0%	0%	100%

Table 7.2: Accuracy results after testing the model.

The tables represent true positives and false positives of the language model for every command. The first column 'AL' is the intended utterance, or actual label, while the first row 'PL' is a predicated label classified by the model. The diagonal boxes are true positives, then the command was recognized correctly, and all the other boxes reveal incorrect predictions. The Accuracy of the trained set is 90.3 %, with model verification accuracy being 85.2 %. However, this is a simulated test made on the platform and does not necessarily depict real-world on-device performance. The on-board accuracy resulted in 75.2 % while testing with the speaker whose voice was used for the training of the model and under 50 % accuracy for an invited tester.

On-board classification takes under 300 ms to complete, where DSP takes 63 ms on average and classification 175 ms. It took some additional time to confirm the command execution from a smartphone. Without the confirmation condition, the microcontroller starts bed positioning almost immediately after the classification is made.

Chapter 8

Discussion

During the model training the focus was on the amount of data required to make the model perform to a degree of accuracy from scratch, e.i. if the user wants to create their own commands. The library of commands shall be expanded to become more general and speaker-independent to accommodate the needs of more potential users.

BLE technology embedded in the microcontroller allows for data communication between a smartphone and the device for both reading and writing data. Thus, the BLE library for Arduino devices has been studied and successfully applied to the requirements of the project. Thanks to the practical approach to this project, many issues regarding the hardware part were uncovered and consequently solved. The solution has changed several times and in the end, a working prototype has been realized. If I were to start this project anew, I would change the implementation of the language model slightly. Nevertheless, TinyML is a new and vigorously developing field, thus it was not possible to know how well its capabilities would suit this particular task.

Speech recognition itself can be approached in many ways. Bigger models do exist and have naturally higher accuracy than a tiny one, The main outcome of this project is finding a working solution to operate the bed with a microcontroller and connecting to it with a smartphone via BLE technology. Then, the commands can be wirelessly sent to the device to perform the required tasks. This is a traditional way that is described in the literature review, though it does bring additional links into the chain, which potentially could cause complexities, increase latency during command execution, and strip the system of its autonomy.

The Model evaluation relied heavily on the speech recognition model performance. Metrics such as accuracy and latency were considered the most important ones. Power consumption has not been taken into consideration at this stage. The issue of audio sampling buffer overrun, which did not allow the user to stop the bed once it initiated its motion, has been mitigated through careful studying of the documentation of the model and code adjustments.

The training of the model lasted a long time because of all the parameters taken into account. So far, a common rule of thumb for training tiny models does not exist and developers must use a trial-and-error method to create the best system possible. Furthermore, the model's estimated performance might be different in real life despite all the efforts to simulate it off-device. It was a meticulous process and resulted in an overall

accuracy of 75 %. Taking into account the size of the processor and the memory it takes up, the performance is promising.

In this project, the number of commands might be an issue for its performance since some of them are quite similar, e.g. have similar beginnings and endings. Another potential problem could be an insufficient library of various audio inputs. Enhancing the data with many human voices would increase the accuracy. However, it is important to take into consideration the microphone on the device and the frequency at which it records audio. During the project, several devices were used for recording data files, and sometimes more data would actually decrease the overall performance. This might be due to the fact that the microphone on the device creates more distortions and noise. It was evident from the audio recordings and their quality. In that case, the model might not generate the features that the device would recognize with its microphone. Furthermore, for a better user experience, an external microphone would be beneficial as the built-in one does not provide enough quality and is not flexible in its position.

The main challenge would be adapting this service for the target user group. Each person with disabilities has their own conditions and requirements that are impossible to predict. Nevertheless, this model presents the highest degree of adaptability and flexibility, which is difficult to attain with mass-market products. Together, a patient and a model engineer can create a system that would assist the user at the highest level possible.

Chapter 9

Conclusion

Speech recognition in the Czech language is unfortunately not yet available on popular mobile voice assistants. Other models that support the Czech language might be bulky, exclude offline usage, or increase time processing due to connection to the cloud services. Thus, a novel approach of creating its own language model based on the speaker's voice and speaking capabilities to deploy on a microcontroller with very limited memory storage has been taken.

This model is versatile, autonomous from the internet connection, fast, adaptive, and reliable. It includes the microcontroller, an 8-pin cable connected to the bed with all the required circuitry in between, a language model for speech recognition, and a web application for audio recording. The model waits for the wake-up word before accepting any further commands, which prevents the bed adjustment randomly when someone is talking. The model recognizes 12 commands including the trigger word. For testing purposes, the microcontroller requires permission to execute the command, which is provided by BLE service, and change its characteristics from a user's smartphone. The feedback is also provided in the form of a message and LED indicators. Any user may record their own utterances of the command, hence increasing the accuracy for a particular user. A web application is available for acquiring the data.

Overall, the accuracy of the model is 75 %, which is satisfactory for a prototype. Besides physical implementation, theoretical knowledge about state-of-the-art technologies in the field of speech recognition, tiny machine learning, deep learning, programming in Arduino IDE, troubleshooting hardware issues and many other valuable knowledge has been achieved.

Tiny machine learning that is being transformed into deep learning for various devices in both healthcare and smart homes is a field worth researching and developing.

9.1 Future work

The possible connectivity to other sensors might be favorable for the user to give an overview of their condition and lead to possible adjustments of the position. Such sensors could be medical ones, such as heart rate, blood pressure, body temperature, or capacity sensors to determine the position of a user on the mattress and adjust it based on the

time and condition of a laying person.

The position of the bed is implemented in the code only using global variables and is not verified with the physical angle of inclination. The medical bed does not have a stopping mechanism for the adjustments, which creates a danger for the user and the mechanism itself. Additional sensors that follow the position of both head and leg sectors would be a necessary addition for testing with actual users.

Another function would be the Over-the-Air update described in Chapter 5. It will allow the edge device to be up-to-date, optimized, and cut for a particular user. Another microcontroller that can be connected to the Arduino Cloud service might be a better option since it is supported by the developers themselves.

From the hardware side of the project, a similar but different microcontroller might be considered, as it provides the above-mentioned OTA update without compromising the technical parameters of the device and thus retains performance. It is required to order the parts in advance from stores abroad, as Arduino Nano RP2040 Connect was not found in Czech Republic electronics stores at the moment of writing this thesis. An external microphone of higher quality would also be necessary to implement in code as well as connection to the board. Additional design adjustments will follow.

The prototype is wired into the breadboard for simplicity and ease of the process of testing. Nonetheless, the wires are not tightly attached to corresponding connectors, are bulky, and are not intended for the end user to operate with. A PCB design with optimal size and DC power source is required for further testing. An attachment to the bed itself has to be functional and allow the user a seamless experience without hindering any other activities.

Further optimization of the language model is possible. More audio data will increase the accuracy of the model without increasing its RAM usage. With recordings from people of different genders, ages, speaking capabilities, and accents, it is possible to create a speaker-independent model that can be adjusted according to the user requirements if necessary.

Appendix A

Attachments

Web_application.zip	Source code for web application
Arduino_code.ino	Source code with bed controlling logic for the microcontroller
Language_model_library.zip	Library with language model
Zadani.pdf	Scanned Master's thesis assignment
demonstration.mp4	Video demonstration of the bed control
audio_recordings.zip	All the recorded audio files

Bibliography

- [1] *Global survey report on persons with disabilities and disasters*, <https://www.undrr.org/report/2023-gobal-survey-report-on-persons-with-disabilities-and-disasters>, 2023.
- [2] F. Alshehri and G. Muhammad, “A comprehensive survey of the internet of things (iot) and ai-based smart healthcare”, *IEEE Access*, vol. 9, pp. 3660–3678, 2021. DOI: 10.1109/ACCESS.2020.3047960.
- [3] *Smart healthcare market size research report [2023-2030]*, <https://www.linkedin.com/pulse/smart-healthcare-market-size-research-report-2023-2030-qelkf/>, 2024.
- [4] L. Deng, “Deep learning: From speech recognition to language and multimodal processing”, *APSIPA Transactions on Signal and Information Processing*, vol. 5, e1, 2016.
- [5] A. Ihsan, “Smart bed using voice recognition for paralyzed patient”, *IOP Conference Series: Materials Science and Engineering*, vol. 854, p. 012045, Jul. 2020. DOI: 10.1088/1757-899X/854/1/012045.
- [6] M. M. Elsokah and A. R. Zerek, “Next generation of medical care bed with internet of things solutions”, in *2019 19th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering (STA)*, 2019, pp. 84–89. DOI: 10.1109/STA.2019.8717204.
- [7] M. M. Elsokah and A. R. Zerek, “Design and development intelligent medical care bed using voice recognition”, in *2022 IEEE 2nd International Maghreb Meeting of*

- the Conference on Sciences and Techniques of Automatic Control and Computer Engineering (MI-STA)*, 2022, pp. 299–304. DOI: 10.1109/MI-STA54861.2022.9837521.
- [8] A. Wongsorn, K. Srijiranon, and N. Eiamkanitchat, “Application of iot using neuro-fuzzy based on thai speech classification to control model hospital bed with arduino”, in *2019 IEEE 8th Global Conference on Consumer Electronics (GCCE)*, 2019, pp. 702–706. DOI: 10.1109/GCCE46687.2019.9015560.
- [9] D. Wang and H. Yu, “Development of the control system of a voice-operated wheelchair with multi-posture characteristics”, in *2017 2nd Asia-Pacific Conference on Intelligent Robot Systems (ACIRS)*, 2017, pp. 151–155. DOI: 10.1109/ACIRS.2017.7986083.
- [10] H. Han and J. Siebert, “Tinyml: A systematic review and synthesis of existing research”, in *2022 International Conference on Artificial Intelligence in Information and Communication (ICAIIIC)*, 2022, pp. 269–274. DOI: 10.1109/ICAIIIC54071.2022.9722636.
- [11] *Ieee xplore search*, <https://ieeexplore.ieee.org/>, Accessed: 2024-22-04.
- [12] J. Lin, L. Zhu, W.-M. Chen, W.-C. Wang, and S. Han, “Tiny machine learning: Progress and futures [feature]”, *IEEE Circuits and Systems Magazine*, vol. 23, no. 3, pp. 8–34, 2023. DOI: 10.1109/MCAS.2023.3302182.
- [13] I. Taştan, M. Karaca, and A. Yurdakul, “Approximate cpu design for iot end-devices with learning capabilities”, *IEEE Circuits and Systems Magazine*, 2020. DOI: 10.3390/electronics9010125.
- [14] P. Warden and D. Situnayake, *TinyML: Machine Learning with TensorFlow Lite on Arduino and Ultra-low-power Microcontrollers*. O’Reilly, 2020, ISBN: 9781492052043. [Online]. Available: <https://books.google.cz/books?id=sB3mxQEACAAJ>.
- [15] L. Capogrosso, F. Cunico, D. S. Cheng, F. Fummi, and M. Cristani, “A machine learning-oriented survey on tiny machine learning”, *IEEE Access*, vol. PP, pp. 1–1, Jan. 2024. DOI: 10.1109/ACCESS.2024.3365349.

- [16] J. M. Phillips and J. M. Conrad, “Robotic system control using embedded machine learning and speech recognition”, pp. 214–218, 2022. DOI: 10.1109/HONET56683.2022.10019106.
- [17] V. Janapa Reddi, B. Plancher, S. Kennedy, *et al.*, “Widening access to applied machine learning with tinyml”, *Harvard Data Science Review*, Jan. 2022. DOI: 10.1162/99608f92.762d171a.
- [18] S. A. R. Zaidi, A. M. Hayajneh, M. Hafeez, and Q. Z. Ahmed, “Unlocking edge intelligence through tiny machine learning (tinyml)”, *IEEE Access*, vol. 10, pp. 100 867–100 877, 2022. DOI: 10.1109/ACCESS.2022.3207200.
- [19] *Mcus expected to make modest comeback after 2020 drop*, <https://www.linkedin.com/pulse/smart-healthcare-market-size-research-report-2023-2030-qelkf/>, 2020.
- [20] *Cmusphinx*. <https://cmusphinx.github.io>, Accessed: 2024-23-04.
- [21] *Pocketsphinx documentation for python*. <https://pocketsphinx.readthedocs.io/en/latest/>, Accessed: 2024-10-04.
- [22] *Pocketsphinx on github*. <https://github.com/cmusphinx/pocketsphinx>, Accessed: 2024-15-04.
- [23] *Cmusphinx tutorial*. <https://cmusphinx.github.io/wiki/tutorialam/>, Accessed: 2024-23-04.
- [24] *Whisper ai github*. <https://github.com/openai/whisper>, Accessed: 2024-23-04.
- [25] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, and I. Sutskever, “Robust speech recognition via large-scale weak supervision”, 2022. arXiv: 2212.04356 [eess.AS].
- [26] *Alpha cephei vosk api*. <https://alphacephei.com/en/#about-page>, Accessed: 2024-15-04.
- [27] *Vosk offline speech recognition api*. <https://alphacephei.com/vosk/>, Accessed: 2024-04-04.

- [28] *Vosk api github*. <https://github.com/alphacep/vosk-api#>, Accessed: 2024-11-04.
- [29] *Gcp speech-to-text documentation*. <https://cloud.google.com/speech-to-text/docs>, Accessed: 2024-15-04.
- [30] *Google cloud compliance*. <https://cloud.google.com/compliance?hl=en>, Accessed: 2024-23-04.
- [31] *Google cloud speech-to-text pricing*. <https://cloud.google.com/speech-to-text?hl=en#pricing>, Accessed: 2024-23-04.
- [32] *Google cloud speech-to-text pricing*. <https://cloud.google.com/storage/docs/buckets>, Accessed: 2024-23-04.
- [33] C. Soni, M. Saklani, G. Mokhariwale, A. Thorat, and K. Shejul, “Multi-language voice control iot home automation using google assistant and raspberry pi”, in *2022 International Conference on Advances in Computing, Communication and Applied Informatics (ACCAI)*, 2022, pp. 1–6. DOI: 10.1109/ACCAI53970.2022.9752606.
- [34] *Google assistant documentation*. <https://assistant.google.com/?sjid=7745679240305134362-EU>, Accessed: 2024-21-04.
- [35] *Siri documentation*. <https://developer.apple.com/design/human-interface-guidelines/siri/>, Accessed: 2024-17-04.
- [36] *Bixby developer*. <https://bixbydevelopers.com>, Accessed: 2024-21-04.
- [37] *Amazon alexa documentation*. <https://developer.amazon.com/en-US/docs/alexa/documentation-home.html>, Accessed: 2024-17-04.
- [38] *Arduino alexa skill*. <https://www.amazon.com/Arduino-LLC/dp/B07ZT2PK2H>, Accessed: 2024-21-04.
- [39] *Cortana documentation*. <https://support.microsoft.com/en-us/cortana>, Accessed: 2024-21-04.
- [40] *Edge impulse fdocumentation*. <https://docs.edgeimpulse.com/docs>, Accessed: 2024-15-04.

- [41] *Edge impulse tutorials*. <https://docs.edgeimpulse.com/docs/readme/for-beginners>, Accessed: 2024-10-04.
- [42] *Tensorflow lite*. <https://www.tensorflow.org/lite>, Accessed: 2024-23-04.
- [43] *Keras api documentation*. <https://keras.io/api/>, Accessed: 2024-15-04.
- [44] *Elechouse v3 manual*. https://dratek.cz/docs/produkty/1/1037/vr3_manual.pdf, Accessed: 2024-10-04.
- [45] X. Lin and Z. Zhang, “Designing remote control of medical bed based on human factors”, in *2020 IEEE 7th International Conference on Industrial Engineering and Applications (ICIEA)*, 2020, pp. 838–841. DOI: 10.1109/ICIEA49774.2020.9101944.
- [46] *Elektrické polohovací lůžko - aks 2fx*. <https://www.invira.cz/p/elektricke-polohovaci-luzko-aks-2fx>, Accessed: 2023-10-11.
- [47] *Elektrické polohovací lůžko - burmeier dali iv*. <https://www.invira.cz/p/elektricka-polohovaci-postel-burmeier-Dali-iv-standard>, Accessed: 2023-10-11.
- [48] *Polohovací rošť silverton 42 mot*. https://www.fajnspanek.cz/motorovy-lamelovy-polohovaci-rost-silverton-42-mot-elektricky/?gad_source=1&gclid=Cj0KCQjw0rUYBhDuARIsANSZ3wo9JsbIFbjePlicZS0re_uqvQI1bo6pGtFhiUi2_BN698jtocrsM1IaArsQEALw_wcB, Accessed: 2023-10-11.
- [49] *Blissful nights adjustable beds*. <https://www.blissfulnights.com>, Accessed: 2023-10-11.
- [50] *Elektrická polohovací postel völker 2080*. <https://www.kv-postele.cz/produkt/e14-volker/>, Accessed: 2023-10-11.
- [51] *Arduino mega*. <https://store.arduino.cc/products/arduino-mega-2560-rev3>, Accessed: 2024-15-04.
- [52] *Arduino nano 33 ble*. <https://store.arduino.cc/products/arduino-nano-33-ble>, Accessed: 2024-15-04.

- [53] *Arduino nano 33 ble sense rev2*. <https://docs.arduino.cc/hardware/nano-33-ble-sense-rev2/>, Accessed: 2024-15-04.
- [54] *Arduino nano rp2040 connect*. <https://docs.arduino.cc/hardware/nano-rp2040-connect/>, Accessed: 2024-15-04.
- [55] *Raspberry pi pico*. <https://www.raspberrypi.com/products/raspberry-pi-pico/>, Accessed: 2024-15-04.
- [56] *Arduino cloud*. <https://cloud.arduino.cc>, Accessed: 2024-15-04.
- [57] F. J. Dian, A. Yousefi, and S. Lim, “A practical study on bluetooth low energy (ble) throughput”, in *2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, 2018, pp. 768–771. DOI: 10.1109/IEMCON.2018.8614763.
- [58] C. Gomez, I. Demirkol, and J. Paradells, “Modeling the maximum throughput of bluetooth low energy in an error-prone link”, *IEEE Communications Letters*, vol. 15, no. 11, pp. 1187–1189, 2011. DOI: 10.1109/LCOMM.2011.092011.111314.
- [59] W.-H. Chen and F. J. Lin, “Over-the-air provisioning for iot wearable devices via ble and onem2m”, in *2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC)*, vol. 2, 2019, pp. 1–6. DOI: 10.1109/COMPSAC.2019.10174.
- [60] *Ota model updates*. <https://edge-impulse.gitbook.io/docs/tutorials/lifecycle-management/ota-model-updates>, Accessed: 2024-15-04.
- [61] *Nordic semiconductors nrf connect desktop app*. <https://www.nordicsemi.com/Products/Development-tools/nRF-Connect-for-Desktop>, Accessed: 2024-21-04.
- [62] W. Pratumthong, N. Phinyosab, P. Saiyut, and S. Prongnuch, “Mobile application for basic computer troubleshooting using tensorflow lite”, in *2021 7th International Conference on Engineering, Applied Sciences and Technology (ICEAST)*, 2021, pp. 226–229. DOI: 10.1109/ICEAST52143.2021.9426292.

- [63] M. Malik, M. Malik, K. Mehmood, and I. Makhdoom, “Automatic speech recognition: A survey”, *Multimedia Tools and Applications*, vol. 80, pp. 1–47, Mar. 2021. DOI: 10.1007/s11042-020-10073-7.
- [64] A. Y. Vadwala, K. A. Suthar, Y. A. Karmakar, and N. Pandya, “Survey paper on different speech recognition algorithm: Challenges and techniques”, *International Journal of Computer Applications*, vol. 175, pp. 31–36, 2017. [Online]. Available: <https://api.semanticscholar.org/CorpusID:56262167>.
- [65] M. R. Gamit, K. Dhameliya, and N. S. Bhatt, “Classification techniques for speech recognition: A review”, 2015. [Online]. Available: <https://api.semanticscholar.org/CorpusID:17451400>.
- [66] L. Tóth, “A hierarchical, context-dependent neural network architecture for improved phone recognition”, in *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2011, pp. 5040–5043. DOI: 10.1109/ICASSP.2011.5947489.
- [67] *Audio mfe*, <https://edge-impulse.gitbook.io/docs/edge-impulse-studio/processing-blocks/audio-mfe>, Accessed: 2024-22-04.
- [68] *Audio mfcc*, <https://edge-impulse.gitbook.io/docs/edge-impulse-studio/processing-blocks/audio-mfcc>, Accessed: 2024-25-04.
- [69] M. Cerna and A. F. Harvey, “The fundamentals of fft-based signal analysis and measurement”, Citeseer, Tech. Rep., 2000.
- [70] *Learning blocks*, <https://edge-impulse.gitbook.io/docs/edge-impulse-studio/learning-blocks#neural-network-architecture>, Accessed: 2024-26-04.
- [71] M. Kirola, M. Memoria, M. Shuaib, K. Joshi, S. Alam, and F. Alshanketi, “A referenced framework on new challenges and cutting-edge research trends for big-data processing using machine learning approaches”, in *2023 International Conference on Smart Computing and Application (ICSCA)*, 2023, pp. 1–5. DOI: 10.1109/ICSCA57840.2023.10087686.

- [72] *Simple recorder demo*. <https://github.com/addpipe/simple-recorderjs-demo/tree/master>, Accessed: 2024-15-04.
- [73] *Audacity*. <https://www.audacityteam.org>, Accessed: 2024-20-04.

List of Tables

3.1	Comparison of main features of the language models.	11
4.1	Positioning control of a medical bed.	12
5.1	Medical bed market overview.	16
5.2	Bed parameters.	16
5.3	Microcontrollers comparison.	17
6.1	Model optimisation steps.	26
6.2	Commands and number of utterances.	28
6.3	WAVE file data structure.	33
7.1	Accuracy results after training the model.	40
7.2	Accuracy results after testing the model.	41

List of Figures

4.1	Results of the survey regarding bed accessories.	13
4.2	Results of the survey regarding bed functionality.	14
5.1	Selected medical adjustable bed.	15
5.2	BLE technology.	19
6.1	ASR diagram.	22
6.2	Algorithm with double buffering.	30
6.3	Data collection window.	32
6.4	8-pin connector.	34
6.5	Hardware connection schematics.	35
6.6	On-device wiring.	36
7.1	Neural network architecture.	38
7.2	Testing with the user.	39

List of Acronyms

AI Artificial intelligence.

ATT Attribute Protocol.

BLE Bluetooth Low Energy.

BPNN Back-propagation neural network.

DFU Device firmware update.

DSP Digital signal processing.

FHSS Frequency-Hopping Spread Spectrum.

GAP Generic Access Profile.

GATT Generic Attribute Profile.

GDPR General Data Protection Regulation.

GFSK Gaussian frequency shift keying.

HIPAA The Health Insurance Portability and Accountability Act.

IDE Integrated Development Environment.

IoT Internet of Things.

OTA Over-The-Air.

PWM Pulse Width Modulation.

SDK software development kit.

TinyML Tiny machine learning.

UCEEB University centrum of energetically efficient buildings.

UUID Universally Unique ID.