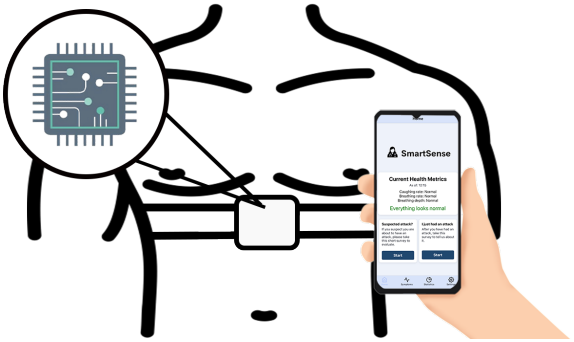
CHALMERS
UNIVERSITY OF TECHNOLOGY



SmartSense: AI-Driven Low-Cost Wearable for Real-Time Symptom Detection and Risk Estimation

Bachelor Thesis *EENX16-VT25-35*

LUKAS DAHLBERG
JAMAL EL-HAJ
ELSA ERKFELDT
RAMI GHALIB
ASHKAN IZADI
CASPER RANSTRÖM

DEPARTMENT OF ELECTRICAL ENGINEERING
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2025
www.chalmers.se

Bachelor Thesis 2025

SmartSense

AI-Driven Low-Cost Wearable
for Real-Time Symptom Detection
and Risk Estimation

LUKAS DAHLBERG
JAMAL EL-HAJ
ELSA ERKFELDT
RAMI GHALIB ASHKAN IZADI
CASPER RANSTRÖM

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Chalmers University of Technology
Gothenburg, Sweden 2025

SmartSense

AI-Driven Low-Cost Wearable for Real-Time Symptom Detection and Risk Estimation

LUKAS DAHLBERG
JAMAL EL-HAJ
ELSA ERKFELDT
RAMI GHALIB
ASHKAN IZADI
CASPER RANSTRÖM

Supervisor: Kann Okumas, Department of Electrical Engineering
Examiner: Giuseppe Durisi, Department of Electrical Engineering

Bachelor Thesis 2025
Department of Electrical Engineering
Chalmers University of Technology
SE-412 96 Gothenburg
Sweden
Telephone +46 31 772 1000

Cover: This illustration shows the SmartSense app displaying real time health metrics on a smartphone held in a user's hand[1]. A zoomed in microchip highlights the AI-driven sensor technology enabling the app's functionality[2]. The image visually connects wearable hardware, smart data processing, and user interaction in a seamless health monitoring system.

Typeset in LaTeX
Gothenburg, Sweden 2025

SmartSense
AI-Driven Low-Cost Wearable for Real-Time
Symptom Detection and Risk Estimation
LUKAS DAHLBERG
JAMAL EL-HAJ
ELSA ERKFELDT
RAMI GHALIB
ASHKAN IZADI
CASPER RANSTRÖM
Department of Electrical Engineering
Chalmers University of Technology

Abstract

Asthma is a chronic respiratory condition that demands continuous symptom monitoring to prevent severe health complications. This project aims to create SmartSense, a low-cost, wearable prototype designed to detect early symptoms of asthma using real-time physiological monitoring. The device integrates a conductive rubber band to track chest expansion and a digital microphone for cough detection. A microcontroller samples the sensor data, performs onboard signal processing using FFT for breathing analysis, and applies a quantized neural network model for AI-based cough classification. Processed data is transmitted to a companion mobile application via BLE, enabling real-time visualization, alerts, and symptom tracking. Emphasis is placed on local data processing and energy efficiency to support extended autonomous operation. Testing showed promising results in detecting respiratory patterns and classifying cough events, although challenges such as background noise sensitivity and sensor nonlinearity remain. SmartSense demonstrates the feasibility of affordable, embedded AI in wearable health technologies and lays the groundwork for future clinical validation and commercial development.

Keywords: wearable technology, asthma management, symptom detection, artificial intelligence, bluetooth low energy, mobile application, real time monitoring, health risk

Sammanfattning

Astma är ett kroniskt respiratoriskt tillstånd som kräver kontinuerlig symtomövervakning för att förebygga allvarliga hälsokomplikationer. Det här projektet presenterar SmartSense, en kostnadseffektiv, bärbar prototyp utformad för att upptäcka tidiga astmasymtom genom realtidsövervakning av fysiologiska signaler. Enheten integrerar ett ledande gummiband för att mäta bröstkorgens expansion samt en digital mikrofon för hostdetektion. En mikrokontroller samplar sensordatan, utför signalbehandling ombord med hjälp av FFT för andningsanalys och tillämpar en kvantiserad neuronnätsmodell för AI-baserad klassificering av hostningar. Den bearbetade datan överförs via BLE till en mobilapplikation, vilket möjliggör realtidsvisualisering, varningar och symtomuppföljning. Fokus har lagts på lokal databehandling och energieffektivitet för att stödja längre autonom drift. Testresultat visade lovande förmåga att upptäcka andningsmönster och klassificera hosthändelser, även om utmaningar såsom känslighet för bakgrundsljud och sensorns icke-linjära respons kvarstår. SmartSense visar på genomförbarheten av prisvärd, inbäddad AI inom bärbar hälsoteknik och lägger grunden för framtida klinisk validering och kommersiell utveckling.

Acknowledgements

We would like to thank our supervisor, Kaan Okumas, and our examiner, Giuseppe Durisi. We also wish to thank CASE and TRACKS for allowing us to build our prototype in their workshops. Special thanks to Maria Dahlberg for her help with the sewing. We are also grateful to our respective departments for their support. Finally, we extend our heartfelt thanks to our families and friends for their encouragement throughout our bachelor thesis.

List of Acronyms

Below is the list of acronyms that have been used throughout this report listed in alphabetical order:

ADC	Analog-to-Digital Converter
AI	Artificial Intelligence
BLE	Bluetooth Low Energy
FFT	Fast Fourier Transform
FN	False Negative
FP	False Positive
I2S	Inter-Integrated Circuit Sound
JSON	JavaScript Object Notation
Li-Po	Lithium Polymer
MB	Megabyte
PSRAM	Pseudo Static Random-Access Memory
PWM	Pulse Width Modulation
RAM	Random-Access Memory
SNR	Signal-to-Noise Ratio
STFT	Short-Time Fourier Transform
TN	True Negative
TP	True Positive
TPU	Thermoplastic Polyurethane
UUID	Universally Unique Identifier

Contents

List of Figures

List of Tables

1
Introduction

1.1 Background

Asthma is a chronic respiratory condition that causes inflammation and narrowing of the airways, leading to symptoms such as coughing, wheezing, shortness of breath, and chest tightness [3]. These symptoms often escalate into asthma attacks, which are episodes of intense breathing difficulty that may require medical intervention [4, 5]. Asthma is a serious affliction that affects approximately 339 million people worldwide [5], with an estimated 455,000 deaths each year, often as a result of asthma attacks [3].

Effective asthma management relies on the recognition of worsening symptoms, allowing individuals to take action to prevent attacks or minimize the effects of an attack. Some such early signs could include frequent coughing episodes, more shallow breaths, faster breathing, or the person struggling to breathe [6]. The actions to take if these symptoms are recognized include using a reliever inhaler or seeking medical attention [7, 8]. However, people may not always notice subtle changes in symptoms or realize how severe they are, increasing the risk of delayed intervention. Such delays can lead to severe consequences, including hospitalization or even death, underscoring the critical importance of continuous symptom monitoring to provide real-time insights and early warnings.

One promising solution for continuous health monitoring, which requires minimal user intervention, is wearable technology. Wearable technology is any piece of technology that is worn on the user, such as smart watches, smart glasses, or hearing aids. In recent years, these types of solutions have been widely adopted for tracking respiratory patterns, heart rate, and other vital signs, with the global market projected to grow from $91.21 billion in 2024 to $324.73 billion by 2032 [9]. By enabling real-time physiological monitoring, these devices can assist in detecting changes that might otherwise go unnoticed, supporting proactive asthma management [10].

Alongside wearable technology, AI-driven health applications are becoming increasingly common for automated symptom detection and risk assessment [11]. When integrated with mobile applications, AI-based analysis enhances accessibility and usability in healthcare, providing personalized alerts and real-time insights for users. By combining wearable technology with AI, a novel product can be developed to help individuals with asthma monitor their symptoms effectively.

1.2 Aim and Design Overview

This project aims to develop SmartSense, a lightweight, low-cost, AI-powered wearable device for real-time detection of asthma symptoms and early risk estimation. The device is intended to provide an affordable, accessible solution for individuals living with asthma, minimizing user burden while maximizing health benefits. The device is designed to continuously monitor respiratory patterns, detect potential asthma symptoms, and transmit relevant data via Bluetooth Low Energy (BLE) to a companion. SmartSense will center on three key hardware elements:

- A microcontroller that samples the sensors, runs the on-board AI model, and handles BLE.
- A microphone for capturing sound.
- A stretch-sensitive conductive rubber band, worn around the chest, to measure expansions.

The mobile application will analyze the received data and provide actionable recommendations based on the user's current symptoms, enabling preventative actions before symptoms escalate into severe episodes. In addition to real-time monitoring, the device will store historical symptom data for later diagnostics, helping to identify potential triggers and contributing to long-term asthma management. The mobile application will feature a user-friendly interface that allows users to manually log symptoms alongside sensor data, further improving the accuracy of recommendations.

1.3 Delimitations

To clearly define the scope of this project, several delimitations have been established. These clarify what is included in the project and what falls outside its boundaries. The scope is defined by the following limitations:

- **Medical validation:** While SmartSense aims to support asthma management by detecting symptoms such as coughing and irregular breathing, the prototype developed in this project will not undergo clinical trials or medical validation. Instead, the prototype serves as a proof-of-concept demonstrating technical feasibility rather than medical reliability.
- **Scope of symptom detection:** The prototype focuses on detecting asthma symptoms based on coughing, breathing frequency, and breathing depth, supplemented by user input. It will not incorporate additional physiological markers such as blood oxygen levels or heart rate. Breathing will be monitored using chest movement only, with abdominal breathing detection excluded. Furthermore, the system will be designed solely for asthma symptom monitoring and cannot diagnose or monitor other medical conditions.
- **Limited cross-platform support:** The mobile application developed in this project will be implemented for a single operating system only, without cross-platform support.
- **Security considerations:** Ensuring robust security measures against potential hacking, particularly related to BLE communication, is beyond the scope

of this project. Instead, the focus will primarily be on reliability, functionality, and user experience.
- **Battery life:** This project aims to create a device capable of running for at least 2 hours on a single charge.
- **Budget:** The project budget is set as $200, covering all hardware and potential software expenses.
- **Proof-of-concept prototype:** This project focuses on creating a functional prototype of SmartSense to validate the concept and demonstrate essential features, including a lightweight design and easy to use interface. The final stages of product development, such as commercial packaging, extensive user testing, and manufacturing, falls outside the scope of this project.

1.4 Report Outline

The remaining chapters of the report are organized as follows. Chapter 2 defines the functional and technical requirements of SmartSense, highlighting the needs that guided the hardware and software design. Chapter 3 provides a detailed explanation of the system architecture, including hardware and software components, as well as the methodologies used for data processing and testing. Chapter 4 describes the integration of individual components into a working prototype, including the AI model, microcontroller code, mobile application, and physical assembly. Chapter 5 presents the results from various tests, such as AI performance, cough detection accuracy, and breathing rate analysis, followed by a discussion of technical challenges and limitations. Chapter 6 discusses the environmental impact, power consumption, and ethical considerations of SmartSense. Finally, Chapter 7 summarizes the main contributions and outcomes of the project, reinforcing the feasibility and potential impact of SmartSense.

2
Needs Analysis and Requirements

This chapter outlines the functional and technical requirements necessary for the development of SmartSense. The functional requirements presented in this chapter describe the core functionalities of SmartSense and why they are essential. In turn, all technical requirements and hardware constraints are derived from these functional requirements, ensuring that each component directly supports its intended purpose.

2.1 Functional Requirements

SmartSense aims to continuously monitor vital respiratory symptoms, enabling early detection of asthma attacks. The functional requirements define the core capabilities that SmartSense must provide. The identified functional requirements are as follows:

- **Ability to monitor respiratory patterns:** The system must continuously and accurately track breathing patterns to detect irregularities that may signal the onset of an asthma attack [12].
- **Cough detection:** The system must reliably detect coughing events, as increased coughing often serves as an early warning sign of respiratory disease exacerbation [13].
- **Wireless data transmission through BLE:** Reliable and energy-efficient wireless communication ensures seamless transmission of respiratory data from the SmartSense wearable to the mobile application.
- **Mobile application with real-time data:** To effectively communicate health information and recommendations to the user, the mobile application must offer an intuitive and user-friendly interface. Additionally, the application must have the ability to store and access historical data for further analysis of trends.
- **Manual symptom input by user:** Certain symptoms, such as increased heart rate, feelings of confusion or difficulty speaking in full sentences, can also contribute to making predictions about future attacks [12]. These symptoms will not be tracked automatically by SmartSense. Instead, the mobile application must allow the user to manually input these symptoms.
- **Compact, lightweight, and wearable form factor:** The device must be designed to be comfortably wearable throughout the day without causing inconvenience or discomfort.

2.2 Needs and Technical Requirements

The functional requirements presented in Section 2.1 necessitate specific technical abilities and hardware choices. Each technical requirement listed below corresponds to the device functionalities and explains the need for their implementation.

- **Microphone with acceptable SNR:** To effectively detect coughing against typical background noise, the chosen microphone must have an SNR (Signal-to-Noise Ratio) ratio of over 65 dB [14]. High SNR ensures clarity and accuracy in the captured audio signals.
- **Stretch sensitive conductive rubber band:** To accurately measure breathing depth and rate, the conductive rubber must reliably detect chest expansions and contractions. The chosen conductive band must offer high sensitivity and a consistent response to chest movements.
- **AI-compatible microcontroller with sufficient memory:** The selected microcontroller must be capable of executing AI algorithms. Although certain optimized models occupy less than a few hundred kilobytes and need only a similar amount of peak RAM (Random-Access Memory) [15], the chosen microcontroller must still provide at least 2 MB of RAM and 8 MB of flash storage. These constraints ensure all necessary software can be stored and executed reliably.
- **Minimum 2-hour battery life:** To ensure practical usability, the device battery must support continuous operation for at least two hours. Additionally, software must be implemented to ensure energy efficiency, including optimized BLE communication intervals.
- **BLE functionality:** BLE consumes about 90% less power than classic Bluetooth [16], making it the better choice of the two in the case of this project. Some BLE microcontrollers draw around 1 µA in idle and about 9 mA during transmission or reception [17]. The microcontroller must support BLE to enable low-power, continuous data transmission to the mobile application.
- **Cost-effective components within a total budget of $200:** To ensure accessibility and adoption, the selected components must balance performance with cost constraints, providing a practical and affordable solution for widespread use.
- **Device weight limit of 120 g:** Commercial devices made to be worn around the chest tend to weigh about 70 g [18]. For this prototype, the combined mass of the microcontroller, microphone, battery, conductive band, and casing must stay under 120 g to remain comfortable.

2.3 Design Challenges

The defined functional and technical requirements form the foundation for the development of SmartSense. However, translating these requirements into a practical prototype introduces several design challenges. The following questions highlight critical aspects that require consideration to ensure SmartSense fulfills its intended purpose and user needs:

- How accurately can the wearable detect respiratory patterns and symptoms such as coughing and irregular breathing?
- What level of actionable recommendations are achievable given the available sensor data?
- How can BLE communication be optimized for energy-efficient data transfer?
- What is the most effective way to present symptom insights and alerts to users via the mobile application?
- How can the system be designed to ensure usability, reliability, and affordability while maintaining a compact and lightweight form?

These questions set the agenda for the project. The next chapter details the system design and methodology that translates each challenge into a practical decision.

3
System Design and Methodology

This chapter will give an overview of the general setup of the system and detail how the choice of components was made in order to fulfill the requirements. It will also include the method for the tests done on the different components and the integrated prototype to ensure everything worked as well as possible.

3.1 System Architechture

The SmartSense system consists of hardware components, including sensors and a microcontroller, and software components, including an AI-based detection algorithm and a mobile application. The development process is structured into the following categories:

Hardware Design:

- **Microcontroller setup and programming:** Configuring the processing unit to handle sensor inputs and transmit data via Bluetooth.
- **Sensor selection and testing:** Evaluating and integrating a microphone for cough detection and conductive rubber for respiratory monitoring.
- **Power management:** Ensuring the system operates within power constraints to achieve at least 2 hours of battery life.

Software Design:

- **AI model development:** Implementing and optimizing a cough detection algorithm to run efficiently on the microcontroller.
- **Mobile application development:** Designing a user-friendly iOS application for real-time monitoring, alerts, and historical data visualization.
- **Bluetooth communication:** Establishing seamless and energy-efficient wireless data transmission between the wearable device and the mobile application.

Each component is designed to work in an integrated system, where the hardware collects and transmits data, and the software processes the information to generate insights for the user. An overview of the system's architecture and subcomponents is illustrated in Figure 3.1.

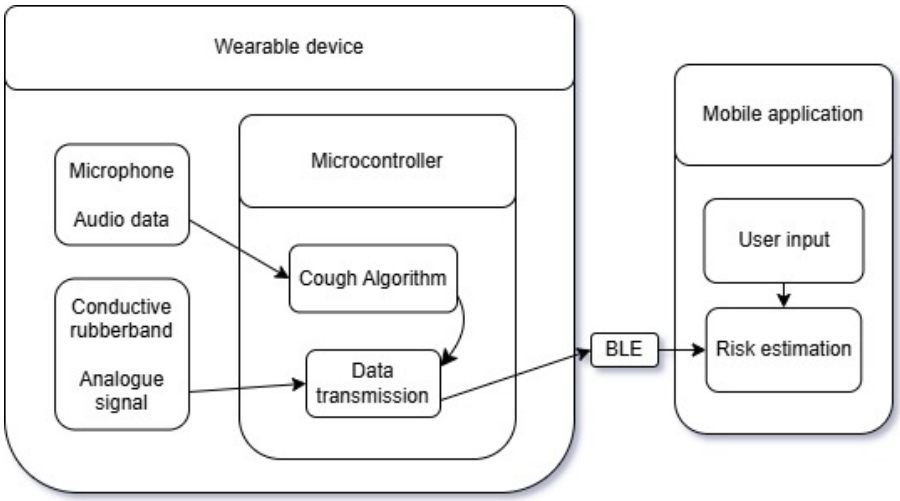


Figure 3.1: Overview of the system architecture and components.

3.2 Hardware Design

The hardware components of SmartSense include a microphone, a conductive rubber sensor, and a microcontroller, which are integrated into a compact and wearable prototype. The hardware is responsible for detecting respiratory symptoms, processing sensor signals, and transmitting data to the mobile application.

3.2.1 Sensor Selection and Integration

The system employs two primary sensors:

- **Microphone:** A microphone captures variations in air pressure (sound waves) and converts them into electrical signals that can be processed and recorded. In the case of this system, it will be used to record coughs and other sounds.
- **Conductive rubber sensor:** This sensor detects changes in electrical resistance caused by stretching. It can measure breathing depth by monitoring the expansion and contraction of the chest or other areas.

The microphone is selected based on its SNR, ensuring clear cough detection. To enhance accuracy, a noise filtering algorithm is applied to the raw audio signal before further processing. The placement of the microphone is tested to determine the optimal position for reliable symptom detection.
The conductive rubber sensor is tested and calibrated to ensure accurate measurement of breathing depth. If initial results indicate weak or inconsistent signals, different placement configurations and sensor settings are explored. Additional signal filtering techniques may be applied to improve accuracy.

3.2.2 Microcontroller Setup

The microcontroller serves as the central processing unit, responsible for receiving sensor data from the microphone and conductive rubber sensor, preprocessing and formatting data before transmission, and managing Bluetooth communication

to transfer data to the mobile application. The microcontroller is programmed to efficiently handle real-time sensor data, ensuring minimal latency and power consumption.

3.2.3 Power Management

To ensure sufficient battery life, power consumption is optimized by using low-power components whenever possible, minimizing BLE transmission intervals to reduce energy usage, and implementing sleep modes when no critical data is being processed.
Battery performance is evaluated through continuous testing, with the goal of maintaining at least 2 hours of operation before recharging.

3.2.4 Selection of Conductive Rubber Band

To measure respiratory movement, a conductive rubber band was chosen. After researching several options, the product Conductive Rubber Cord, 1 m, 2 mm Diameter, 350 Ω per inch was selected [19]. This band has the unique property of varying its electrical resistance when stretched. In its relaxed state, the resistance is approximately 350 Ω per inch. When the band is stretched, the conductive particles embedded within the rubber become more spaced out, leading to an increase in resistance.
For example, a 6-inch segment of the cord has an approximate resistance of 2.1 kΩ [19]. If stretched to 10 inches, the resistance increases to about $10/6 \times 2.1\,\text{k}\Omega = 3.5\,\text{k}\Omega$. The band can be stretched up to 50-70% beyond its original length without permanent deformation [19]. After release, it gradually returns to its initial length, although this process is relatively slow. Due to its non-linear and somewhat inconsistent resistance characteristics (which may vary from batch to batch), this material is not ideal for precise measurements. However, for the prototype stage, it is sufficiently accurate to detect changes in chest circumference due to breathing.
Moreover, the fact that this sensor is in the form of a flexible rubber cord makes it well-suited for integration into wearable designs, such as a chest strap, where it can naturally follow the expansion and contraction of the torso during respiration. Long-term durability (e.g., degradation after thousands of stretch cycles) has not been considered at this stage, as the primary aim is to validate the prototype concept.

3.2.5 Selection of Microphone

Initially, the microphone SPH0645LM4H-B was chosen. This microphone was selected for several reasons. It had a SNR ratio of 65 dB, which is high enough to be able to reliably distinguish the cough sounds from the background [20, 21]. Furthermore, it had a power consumption of 600 μA, which is very low [20]. This is desirable as it means a better battery life for the prototype. The microphone also had an I2S (Inter-Integrated Circuit Sound) digital output, which means it sent data using a standard format, and thus does not require an extra audio chip but can be connected directly to the microcontroller [20].

There were however some problems in soldering this microphone. It was therefore decided to switch to an INMP441 microphone. Just like our previous microphone, this one has an I2S digital output. It has a slightly lower SNR, 61 dBA instead of 65, but this difference is negligible [22]. Furthermore, the INMP441 microphone has a slightly higher power consumption of 1.4 mA instead of 600 μA [22]. While this is still a relatively low power consumption, it will still result in a slightly shorter battery life than if SPH0645LM4H-B had been used. However, the differences between the two microphones were not significant, and due to the availability of the INMP441, it was ultimately selected for the prototype.

3.2.6 Selection of Microcontroller

The LOLIN S3 Pro development board was chosen. It has 8 MB of PSRAM (Pseudo Static Random-Access Memory), and the microcontroller is compatible with BLE. It also features a battery port with a 5 mA charging capability. Additionally, it includes an I²C slot for connecting a microphone and a USB-C socket that can be plugged into a computer [23].

3.3 Software Design

The software components of SmartSense include the Bluetooth communication module, a mobile application, and an AI-based detection algorithm running on the microcontroller. These components work together to process sensor data, detect symptoms, and provide real-time alerts to users.

3.3.1 Bluetooth Communication

BLE is used to transmit processed sensor data from the microcontroller to the mobile application. The communication protocol is designed to be power-efficient, ensuring minimal energy consumption while maintaining reliable real-time data transfer. The system is tested to optimize transmission frequency and packet size, reducing latency while avoiding unnecessary power usage. The communication protocol follows a structured approach in which the microcontroller functions as a BLE peripheral, broadcasting services, while the mobile application acts as the central device, scanning for and establishing connections.

3.3.2 Mobile Application

The mobile application serves as the user interface for SmartSense, providing real-time symptom tracking, historical data visualization, and alert notifications. It allows users to input additional symptoms manually, improving the accuracy of risk estimation. The app is designed with a user-friendly interface, ensuring accessibility for individuals with varying levels of technical expertise. The application is developed using React Native with the Expo framework and utilizes the react-native-ble-plx module to manage BLE connectivity.

3.3.3 AI-Based Detection Algorithm

Among the software features integrated into the device, a cough detection algorithm is implemented using machine learning techniques to identify characteristic cough patterns from audio signals captured by the onboard microphone. The audio data undergoes preprocessing, including noise filtering and feature extraction, before classification. The model is optimised to operate efficiently on the microcontroller, ensuring real-time detection with minimal computational overhead.

3.3.4 Selection of the AI Model

TensorFlow, in combination with the MobileNetV2 base model, was selected to implement cough detection on the ESP32 microcontroller. This choice was based on several factors, including compatibility, computational efficiency, and reliable performance. TensorFlow Lite, a lightweight version of TensorFlow, enables models to run directly on the ESP32 microcontroller without requiring an operating system [24]. It also reduces model size, making it well-suited for our AI application.
MobileNetV2 was chosen due to its compatibility with embedded systems and ability to efficiently deliver reliable performance with limited hardware resources. MobileNetV2 is a compact convolutional neural network architecture that has been specifically designed for resource-limited environments such as mobile and embedded systems [25]. The model employs depthwise separable convolutions and an inverted residual structure, which reduces computations and memory use while maintaining high accuracy [25]. These features make MobileNetV2 particularly suitable for implementation on the ESP32 microcontroller, a device with limited processing and memory capabilities. Another significant advantage of using MobileNetV2 as a base model is that it comes with pre-trained weights. This helps improve generalization and allows fine-tuning of the pre-trained model to improve performance and accuracy without requiring a large dataset.

3.4 Data Processing and Analysis

The SmartSense system processes sensor data using signal transformation and AI-based classification to analyze respiratory activity. The main steps in data processing and analysis are as follows:

3.4.1 Data Collection and Preprocessing

The system utilizes two types of sensors: a conductive rubber band that increases in resistance when stretched and a microphone that records audio data over time. Voltage is measured from the band, and the time-domain voltage data is transformed into the frequency domain. The audio signal from the microphone is converted into a spectrogram.

3.4.2 Data Processing

The frequency domain data from the conductive band is used directly to assess breathing patterns and detect changes in chest movement. The spectrogram generated from the microphone data is input into an AI model, which classifies potential respiratory symptoms.

3.4.3 Data Transmission and Storage

After processing, the results are transmitted via BLE to the mobile application. This application will then display real-time insights and visual trends related to breathing. It will also store historical data and allow users to manually input additional symptom information.

3.5 Testing and Validation

The SmartSense system undergoes systematic testing and evaluation to ensure accuracy, reliability, and efficiency. The testing process is divided into sensor testing, AI performance evaluation and Bluetooth communication assessment.

3.5.1 Sensor Testing

The microphone and conductive rubber sensor are tested individually to ensure accurate data collection.

- **Microphone testing:** The microphone's ability to detect sound is evaluated under various noise conditions to ensure consistent detection accuracy.
- **Conductive rubber sensor testing:** The sensor is tested by measuring chest expansion and contraction, ensuring it accurately tracks breathing depth and patterns.

3.5.2 AI Performance Evaluation

The cough detection algorithm is trained using a combination of real and publicly available datasets. It will then be validated using precision, recall, and F1-score metrics to assess classification performance. To ensure robustness, testing is conducted under different environmental conditions with varied background noise levels. The AI model is then converted into a lightweight format (e.g., TensorFlow Lite) and deployed on the microcontroller, where latency and computational efficiency are tested.

3.5.3 Bluetooth Communication Assessment

Data transmission stability is tested by sending sensor data from the microcontroller to the mobile application at different distances. The system is then assessed for latency and power consumption, ensuring an optimal balance between real-time

transmission and energy efficiency. Finally, the mobile application verifies that all transmitted data is received correctly and displayed without delays.
Testing is conducted iteratively, with refinements applied based on collected results. The final prototype is tested in a realistic usage scenario to confirm its effectiveness in monitoring respiratory symptoms.

4

Implementation and Integration

The following chapter will provide a detailed account of how the different solutions were implemented in practice. Additionally, this chapter will offer an in-depth explanation on how the different parts were integrated.

4.1 AI Model for Cough Detection

The training process for the AI model starts with the acquisition of a dataset of raw audio samples. A cough audio dataset from Kaggle was selected because of its already labeled data [26], and was later complemented with real-time audio collected from the microcontroller. The cough audio samples are manually reviewed and split into 2-s intervals that contain coughs. The samples are converted into spectrogram representations, which serve as input features for the neural network. These spectrograms capture the relevant frequency patterns over time, which distinguish cough sounds from other sounds. The spectrograms are separated between cough and non-cough sounds and split into training, validation, and test sets for model training. Each audio sample undergoes preprocessing, and STFT (Short-Time Fourier Transform) processing was explored, but due to microcontroller limitations, this option could not be applied. Instead, all audio files are first resampled to a consistent sampling rate of 8 kHz, to match the sampling rate of the microcontroller, using the Librosa Python library. To create consistent input samples for the model, each audio file is segmented into 2-second chunks. To ensure that each audio file is segmented to the same size, the application of zero-padding is used on shorter audio files, and longer audio files are split into multiple segments. For each segment, a STFT-like process is performed using FTT (Fast Fourier Transform) and a sliding window approach. A Hamming window of size 512 is applied to each frame, and an FFT is computed and converted to a magnitude spectrum. The first 128 frequency bins are retained to create a spectrogram representation of 128x128. The resulting single-channel spectrogram representation undergoes row-wise normalization and is duplicated across three channels (RGB image) to meet the input dimensions expected by the base model. Augmentation is applied to the training set to increase generalization of the data. These inputs are later used for model training using binary-crossentropy, callback (to avoid overtraining of the model), and an automated learning rate, which reduces the learning rate during training based on validation loss. The extended test results of the model accuracy and loss over each epoch during training can be found in Appendix A.1.

4.2 Microcontroller Code C++

The microcontroller, an ESP32-S3, was programmed using C++ in the PlatformIO environment. The system integrates several libraries, including TensorFlow Lite for Microcontrollers, ArduinoFFT, BLE, and ArduinoJson, to achieve real-time signal acquisition, processing, AI-based inference, and wireless transmission. The microcontroller serves as the system's core, managing the input from two distinct sensor components: the microphone and the conductive rubber band.

Microphone Signal Processing

The microphone is used to capture audio data for cough detection with a sampling rate of 8kHz. The audio is read using the I2S interface and processed in real-time using a sliding buffer of 512 samples. The same FTT approach as in the AI model is applied using the ArduinoFFT library to generate time-frequency representations of the audio signal. The result is normalized and passed to a quantized TensorFlow Lite model, which performs inference directly on the microcontroller.
The model classifies each spectrogram and returns a probability indicating whether a cough is present. A threshold is applied to this output, and binary values are stored. Every minute, the system aggregates the cough events and sends the cough frequency to the mobile app through BLE.

Breathing Detection Using Conductive Rubber Band

A voltage divider circuit is used to measure analog voltage changes caused by the expansion and contraction of the chest. The ADC (Analog-to-Digital Converter) readings are sampled at 10 Hz. Every 256 samples, a FFT is computed using ArduinoFFT to detect the dominant breathing frequency (converted to breaths per minute). These readings are averaged over one-minute windows and transmitted via BLE.

AI Interference Integration

A pre-trained and quantized TensorFlow Lite model is stored in firmware as a C++ byte array and loaded into external PSRAM at runtime. A tensor arena of 2.9MB is allocated, and the model is invoked on each new STFT segment. The AI component is fully embedded on the device and runs without external connectivity.

BLE Data Transmission

To enable wireless communication with the mobile application, the ESP32-S3 implements a custom BLE service with separate characteristics for each type of data: cough frequency, breathing rate, breathing depth, and emergency status. Cough and breathing data are handled by distinct BLE characteristics, allowing parallel, structured updates to the app.
Data is transmitted in JSON (JavaScript Object Notation) format, enriched with timestamps provided by the app via a dedicated time characteristic. Transmission

occurs at fixed intervals (typically once every fifteen minutes) or more frequently for time-sensitive values such as the emergency flag. Each characteristic uses the notify operator to push updates without requiring polling, minimizing latency and conserving energy.
This modular BLE structure ensures reliable and synchronized real-time data delivery across all vital parameters while supporting a low-power design suited for continuous wearable use.

4.3 Hardware Integration

Conductive Rubber Band Integration

The hardware setup involved integrating the conductive rubber band into a voltage divider configuration. One end of the voltage divider was connected to a 3-volt power source on the board, followed by a fixed resistor. This was then connected to an analog input pin, for example A1, on the microcontroller. From A1, the conductive band was connected in series to ground. In this configuration, A1 becomes the midpoint of the voltage divider, allowing the voltage drop across the rubber band to be measured, which in turn reflects its resistance and thus the amount of stretch.

Microphone Integration

The microphone hardware setup required connection between the microphone and the microcontroller. Initial integration was performed on a breadboard to allow for easy testing and troubleshooting before moving to a permanent soldered configuration. This approach ensured that the connections and signal integrity could be verified before final assembly.
The schematic illustrating how the microphone was connected to the microcontroller is presented in Figure 4.1.

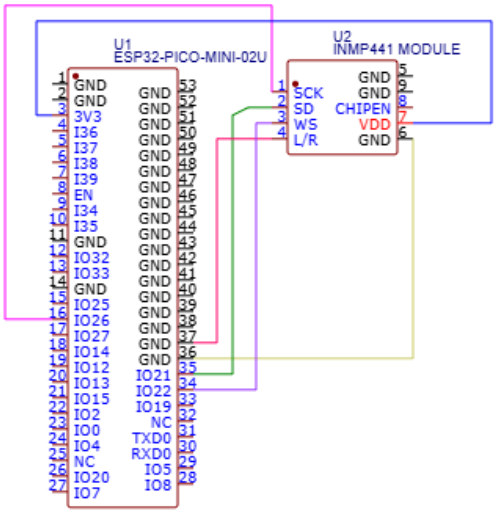


Figure 4.1: Schematics of connections between microchip and microphone.

4.4 Mobile Application

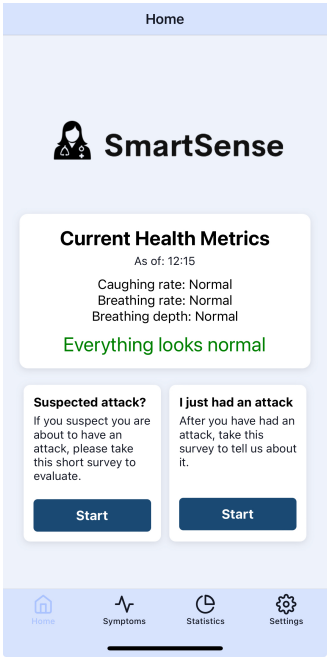
Front-end Designing

The mobile application is written in React Native, using Expo, and the main component of the front-end design is a tab interface, built with React Navigation 6. The file App.js wraps the navigation tree in a NavigationContainer that both applies a custom theme and renders a Tab.Navigator with user-friendly icons at the bottom of the screen. An overview of each tab, its underlying stack, and its main functions can be seen in Table 4.1

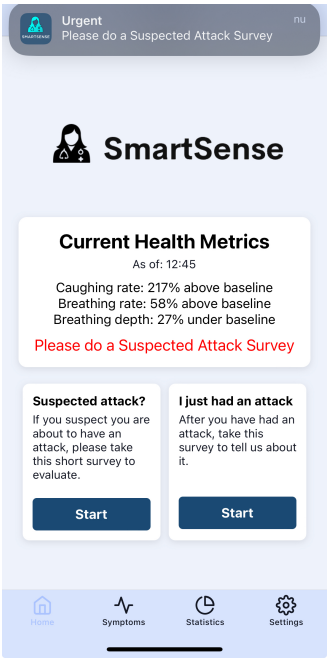
Tab	Underlying stack	Functions
Home	StartStackScreen	Main view, showing real-time status based on current symptoms. Buttons for taking attack surveys.
Symptoms	SymptomsStackScreen	Symptom survey, letting a user answer questions about their symptoms for later analysis.
Statistics	StatisticsStackScreen	Charts displaying user's symptom data, saved user surveys, and data download.
Settings	Plain scroll view	BLE connection status, baseline value view and setters, reminder setting, and developer tools.

Table 4.1: Overview of application tabs, their navigation stacks, and main functionality.

The home tab, seen in Figure 4.2, acts as the landing page of the mobile application. The tab is created to supply the user with instantaneous information about their current symptoms. As new data is received, the home tab will change appearance, prompting the user to do a SuspectedAttackSurvey if it is determined that the current symptoms are indicating a suspected attack. When this happens, a notification is automatically generated, seen in Figure 4.2b. The SuspectedAttackSurvey allows the user to input their current heart rate with the help of a timer, and then lets the user answer a few questions. When done, recommended actions are then given to the user based on their current symptoms and answers. The starting screen of a SuspectedAttackSurvey, and an example of one of the resulting recommendations, can be seen in Figures 4.3. From the home tab, the user can also do an AttackSurvey. This survey lets the user answer questions about an attack they've just had, which enables later analysis.

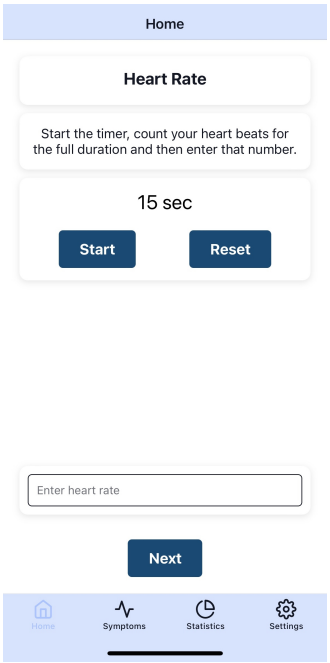


(a) Home tab during normal symptoms.

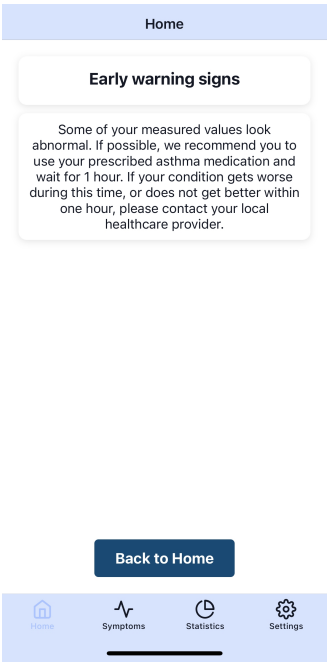


(b) Home tab during escalated symptoms.

Figure 4.2: Home tab of mobile application during different symptoms. (a) shows normal symptoms, while (b) shows escalated symptoms.



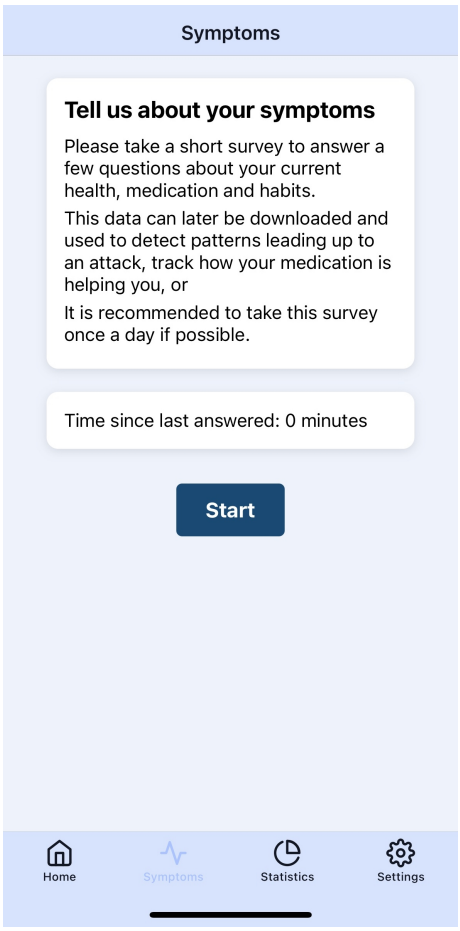
(a) Start of SuspectedAttack-Survey.



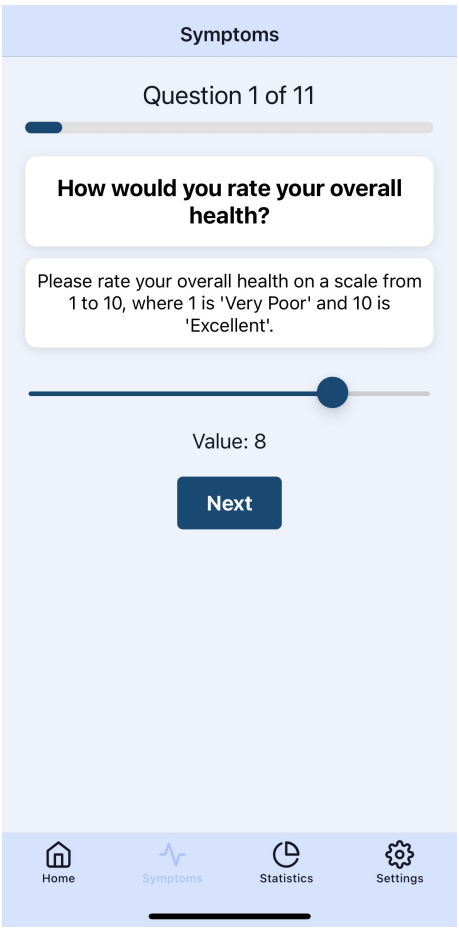
(b) Example of resulting recommended action.

Figure 4.3: The starting screen of a SuspectedAttackSurvet seen in (a), and an example of the resulting recommended action, seen in (b).

The symptoms tab, seen in Figure 4.4, is created to effortlessly answer questions about their current health. The questions are all answered either by choosing a value between one and ten or by pressing yes or no. Each question aims to capture important information about the user's symptoms, habits, environment, and medication. This data can later be downloaded and used in further analysis to detect patterns.



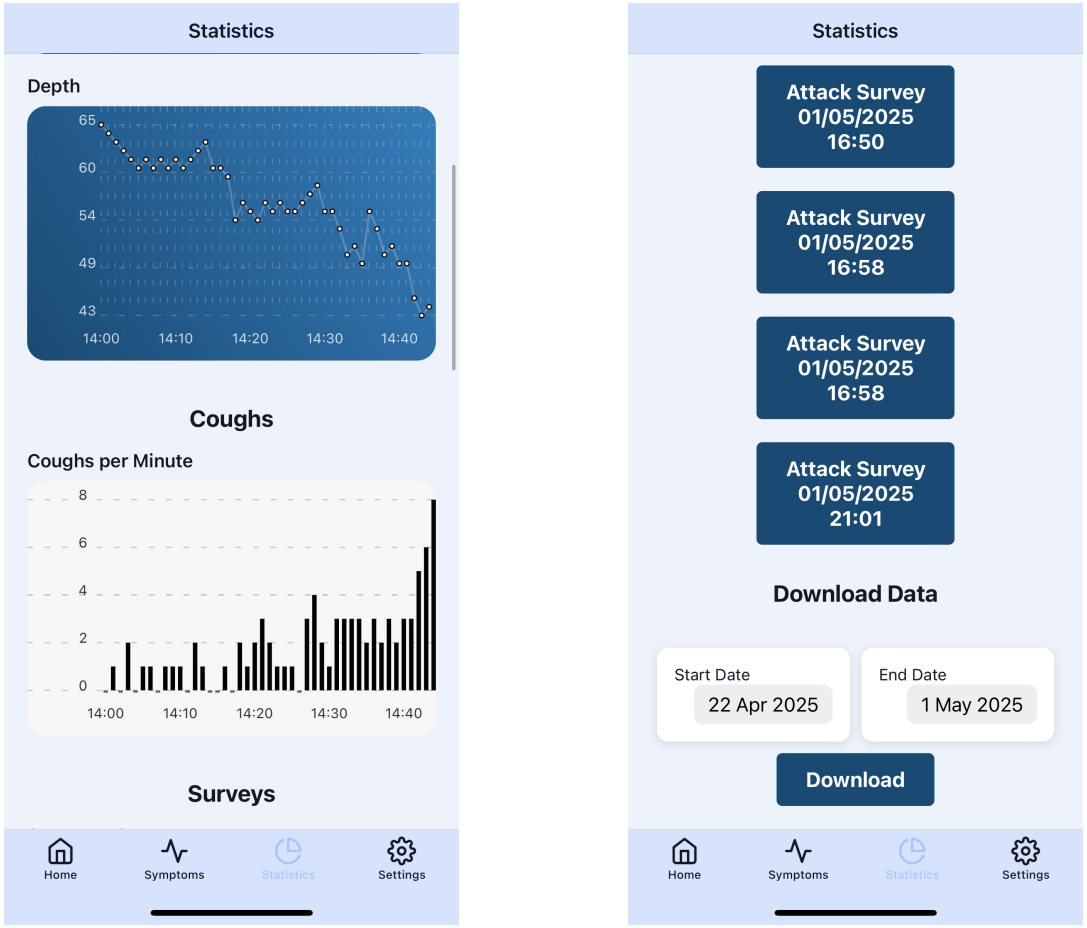
(a) Main symptoms tab view.



(b) Example of question.

Figure 4.4: The starting screen of a SuspectedAttackSurvet seen in (a), and an example of a question asked, seen in (b).

To display all received respiratory data and saved surveys, the user can navigate to the statistics tab. This tab displays breathing frequency and depth over time in line charts and the number of coughs per minute in a bar chart, seen in Figure 4.5. The user can also tap on each individual survey to see their previous answers. Two date pickers allows for the user to set two dates, between which all collected data is downloaded and stored as a JSON file. The charts and survey sections are both updated as new data is received or added.



(a) Graph section of statistics tab. (b) Download section of statistics tab.

Figure 4.5: Two views of the statistics tab. Graphs of breathing depth and coughs per minute are shown in (a), while the download section is shown in (b).

Lastly, the settings tab allows the user to see and change a few key factors about the application. An indicator shows the current status of the microcontroller, and when the last ping was received, allowing for some simple troubleshooting. Information about the currently set baselines is shown, and two ways of setting new baselines let the user easily manage them. An option to enable and disable symptom reminders lets the user control whether they want to receive notifications if they have not done a symptom survey for a set amount of time. Lastly, the user can enable developer mode to gain even further control of the application, allowing for manually adding data, manually connecting to BLE devices, enabling EmergencyMode, and removing all collected data. An example of the view in the settings tab can be seen in Figure 4.6.

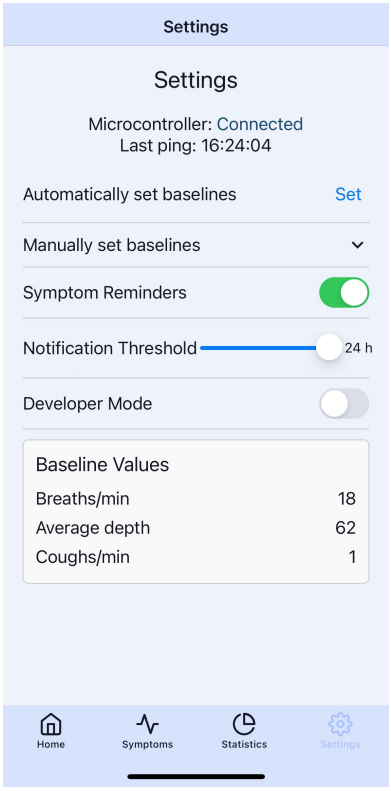


Figure 4.6: Example view of the settings tab.

Back-end Design

The mobile application executes all logic on the users device, without the use of a remote server. Consequently, the "back-end" refers to the client code that stores states, receives sensor data, evaluates risk, and trigger notifications. Like the front-end, the back-end is primarily build in React Native, and uses React Native Contexts, local storage, BLE communication, and notification handling.
Two global states, the EmergencyModeContext and BaselineContext, are handled using React Native Context:

- EmergencyModeContext maintains whether the application is in an emergency state, influencing the risk calculation algorithms.
- BaselineContext manages baseline values for cough rate, breathing frequency, and breathing depth.

Data persistence across sessions is achieved with AsyncStorage from react-native-async-storage, and is mainly used to store two types of data in JSON format. Measurement data (e.g., the user's breathing frequency) is received over BLE and stored under specific keys (e.g., measurementDataFrequency). When stored, it can later be retrieved to calculate risk, to be displayed in graphs, or to be downloaded. An example of what measurement data looks like when stored can be seen in Table 4.2.

Key	Value
time	2025-04-08T12:00:00Z
type	frequency
length	15
data	[15, 16, 17, 18, 19, 18, 19, 19, 19, 19, 20, 18, 17, 16, 15]

Table 4.2: Example of how data is stored under the key measurementDataFrequency.

In this case, the array stored under the data key refers to the average breathing frequency per minute. The timestamp under the time key indicates when the first value of the data array was measured. Consequently, each value of the data array corresponds sequentially to one minute, beginning at the given time. In the example, the first value indicates that the user had an average breathing frequency of 15 at 12:00, an average frequency of 16 at 12:01, and so fourth.
User-submitted data (e.g., symptom surveys or attack surveys) is also stored using AsyncStorage, and can be accessed to display historical data or to be downloaded. Like the measurement data, user-submitted data is also stored under specific keys (e.g., symptomSubmissions).
BLE is integrated in the app using the BleManager from react-native-ble-plx. On the application side, the BleManager handles scanning, connecting, and communication with the microcontroller. When the application is started, the system is set to listen for a specific SERVICE_UUID (Universally Unique Identifier) that the microcontroller is advertising. If the SERVICE_UUID is found, a connection is automatically made, and the application subscribes to notifications from four different characteristics. These characteristics, created by the microcontroller, each correspond to one type of data (e.g., breathing frequency or emergency mode) that the microcontroller wants to send. The system listens for characteristic updates, and when changed, they are decoded and stored locally. In addition to being used by the microcontroller to make changes to the EmergencyModeContext, the EMERGENCY_CHAR_UUID serves another purpose. This characteristic is continuously updated by the microcontroller every ten seconds. If the application does not receive an update in a given amount of time, an automatic disconnect is performed, and the system attempts to reconnect. To further handle synchronization, a fifth characteristic is used. Every time the application receives an update from the EMERGENCY_CHAR_UUID, it also updates the TIME_CHAR_UUID with the current time, allowing the microcontroller to accurately set timestamps on gathered data.
The application utilizes several key metrics to calculate risk and provide personalized feedback:

- **Measurement data:** Collected data about the user's current breathing frequency, depth, and cough rate. The application analyzes these measurements to detect potential abnormalities.
- **Emergency mode:** A specific mode activated when critical symptoms are detected. When in emergency mode, the application modifies the risk calculations to only take the most recently collected data into account.
- **Baselines:** Values specific to the current user, representing breathing frequency, depth, and cough rate. These values are either set automatically

based on historical data or manually by the user and serve as reference points for detecting abnormalities.
- **Threshold values:** Predefined critical limits for symptom data. When a threshold for a specific data type is exceeded, that data is treated as abnormal.

First, an average for each type of data is calculated, based on measurement data. Normally, this average is based on the fifteen most recent values of each type, but if emergency mode is activated, the three most recent values are instead used. The averages are then divided by their respective baseline value, resulting in change factors representing the current increase or decrease of each type of data. Lastly, the change factors are compared with their respective threshold values to determine if the change is abnormal. The application then uses the number of currently abnormal values to generate feedback or to trigger notifications.
A similar procedure is used to generate the resulting feedback given in the SuspectedAttackSurvey. In addition to the number of abnormalities calculated by the automatic risk algorithm, the feedback given in the SuspectedAttackSurvey also depends on the number of abnormal answers given in the survey. If the user inputs a heart rate that exceeds its threshold value, while also simultaneously answering "Yes" to questions regarding specific symptoms (such as difficulty speaking, drowsiness, or confusion), these responses are marked as abnormal. Consequently, the resulting feedback provided by the application will adapt accordingly.

4.5 Final Assembly

This section summarizes the final component choices, casing design, and assembly process for the SmartSense prototype. The selected components met budgetary and weight constraints, while the casing was iteratively designed for durability, comfort, and functionality. The fully assembled wearable device successfully met the project's lightweight, compact, and user-friendly design criteria.

4.5.1 Summary of Component Selection

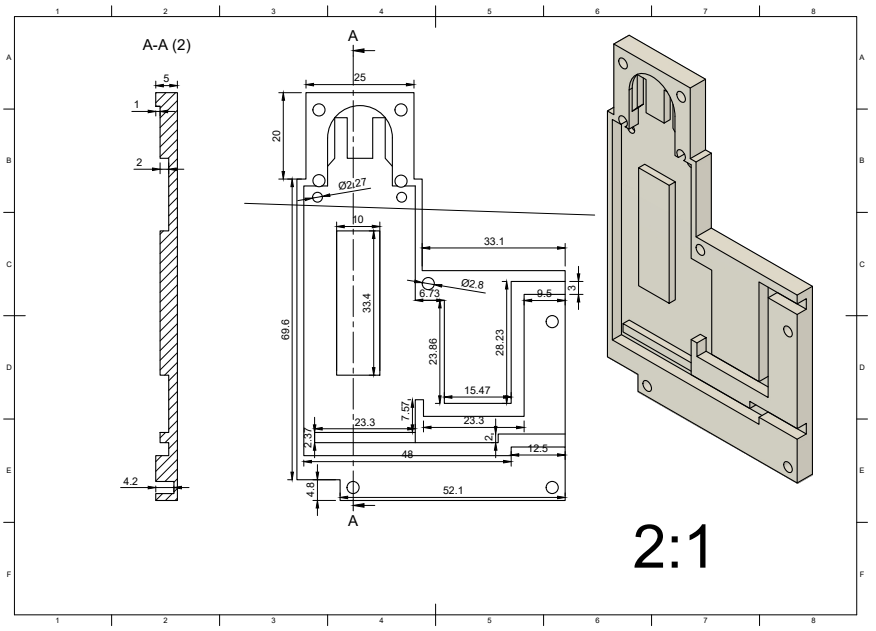
To consolidate the hardware choices and confirm alignment with the design goals, Table 4.3 presents a summary of the final components integrated into the SmartSense prototype. This includes the selected models, their approximate individual costs, and weights.
The total system weight is 154 grams, with all of it apart from the chest strap weighing 56 grams. This puts the prototype bellow the required combined mass of the microcontroller, microphone, battery, conductive band, and casing. It also stays within the $200 budget, confirming the feasibility of the low-cost, lightweight wearable design.

Component	Price (SEK)	Weight (g)
Microcontroller	254	7.2
Microphone	44	4.0
Conductive Rubber Band	109	20.0
Battery	129	14.8
Casing	0	10.0
Chest Strap	0	98.0
Total	**536**	**154**

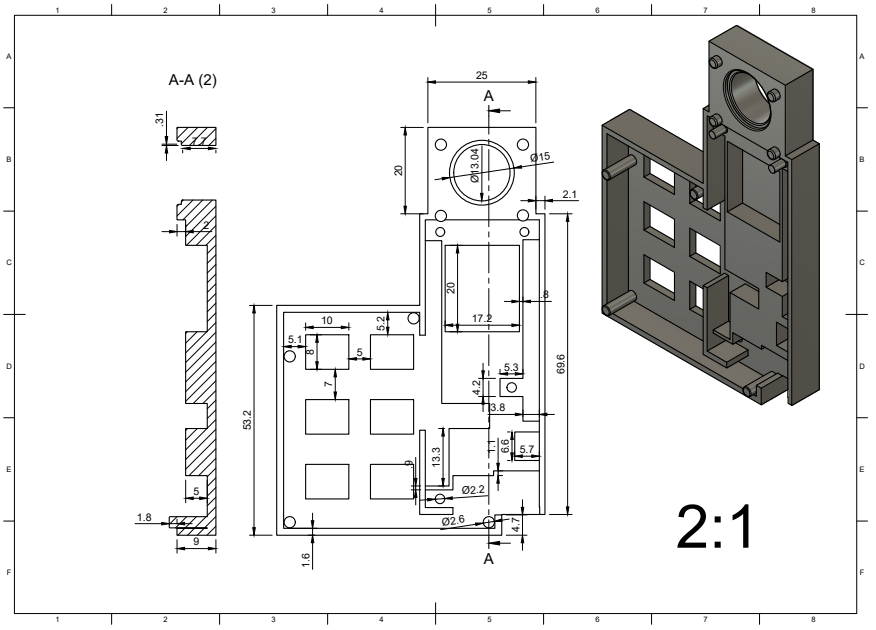
Table 4.3: Component cost and weight breakdown.

4.5.2 Casing

The casing was designed in Fusion 360 and 3D printed using thermoplastic polyurethane (TPU). TPU was chosen for several reasons. Firstly, it is soft and flexible, which makes it more comfortable to wear than many other filaments [27]. It is also durable and can withstand significant wear, which is crucial since it allows the user to go about their daily life without worrying about the case breaking [27]. TPU is also more resistant to body fluids, oils, and cleaning agents than other filaments, making it extra suitable for a device worn on the body [28]. Multiple iterations of the casing were tested in order to ensure that it was as comfortable and functional as possible. It was designed to be as thin as possible to avoid it protruding too much from the user's body. ensure In the final version, the slot for the microphone was positioned high on the casing so the microphone would be as close as possible to the sound source. The section holding the microphone was also made as thick as the rest of the case, despite housing a much thinner component. This was done to prevent anything, such as fabric, from accidentally touching the microphone and thus affecting its performance. To prevent the bands from touching each other, the casing included long, separate channels for them. Additionally, ventilation holes were incorporated so heat could dissipate, to avoid the device from getting too hot to wear, or hot enough to damage something after prolonged use. In Figure 4.7 the parts of the case are shown. The ventilation holes are the square holes on the left side of the top part of the case.



(a) Bottom part of casing.



(b) Top part of casing.

Figure 4.7: Design of the casing

4.5.3 Final Integrated Prototype

The components were initially tested and organized on a breadboard prior to being soldered and enclosed within the TPU casing. The final device was fastened to the chest using an adjustable strap, providing both comfort and stability for extended daily wear. Inside the casing, the internal layout was carefully optimized to reduce bulk, resulting in a total weight of around 56 grams and compact dimensions of 9.5

cm × 6 cm × 1.4 cm, successfully meeting the design criteria for wearable use.

5

Test Result and Discussion

This chapter presents the results from testings of the different major parts of the system as well as an integrated test with all the different parts. It will also contain discussion and analyzes of the findings. In this chapter, the results of all the tests conducted for all of the different subsystems will be presented. The findings for these tests will be analyzed and discussed, including the technical challenges that may have had an effect on the results, and suggestions for further improvement.

5.1 AI Testing

This section outlines the procedures and test results of the AI model used to evaluate its performance against defined benchmarks. The model's performance is comprehensively analyzed using established metrics, such as accuracy, precision, recall, F1 score, and confusion matrices, to determine the model's ability to generalize to new data and fulfill functional requirements.
To ensure that the AI model satisfies the requirement of an accuracy of at least 85%, the model was subjected to tests to assess its performance. A comprehensive evaluation is carried out using the test dataset to assess the performance of the AI model, which represents unseen, unbiased, real-world data. The primary metric is accuracy, which indicates the proportion of correctly classified samples across all categories. To ensure the accuracy is not misleading, a confusion matrix is created, which provides a breakdown of TP (True Positive), TN (True Negative), FP (False Positive), and FN (False Negative). From the confusion matrix, the precision (the proportion of predicted samples that were correct) and recall (the proportion of samples that were correctly identified by the model) can be derived. To balance precision and recall, an F1 score is calculated, which represents the harmonic mean of the two. This offers a more balanced and informative measure of the AI model's performance. The primary goal for the testing was to ensure that the model complies with the requirements placed on it and to illuminate potential improvements to ensure that the model is not only accurate but also reliable with high recall. Testing helps reveal weaknesses, such as bias towards a specific class or overfitting, where the model is overtrained on the training data and is not reliable in real-world scenarios.

Test Results

Using TensorFlow's built-in function to assess precision, we can determine if the trained model reaches the accuracy threshold of 85% required of the system, when

tested with the unbiased test set. The result of this can be seen in Table 5.1 A confusion matrix is constructed by comparing the expected model output with its prediction to test the model's recall and provide a more detailed summary of its accuracy, this is shown in Table 5.2.

Label	Precision	Recall	F1 score
Background	0.92	0.94	0.93
Cough	0.94	0.92	0.93
Overall Accuracy	93%		

Table 5.1: Float32 Accuracy Measurement.

		Predicted	
		Background	Cough
Actual	Background	47 (TN)	3 (FP)
	Cough	4 (FN)	46 (TP)

Table 5.2: Float32 Confusion Matrix for Background vs Cough Classification.

The model is later converted to a TensorFlow Lite model and quantized from float32 to int8 to compress the model size to fit on the limited microcontroller memory. The quantization is done by reducing the precision of the model weights from 32 bits to 8 bits, which cuts the model size to a fourth of its original size. However, the expected input and output shape is kept at float32 to comply with the base model expectations. The quantized model is subjected to an accuracy test on the test dataset, shown in Table 5.3, and a confusion matrix, shown in Table 5.4 is constructed to ensure the accuracy requirement is met after quantization.

Label	Precision	Recall	F1 score
Background	0.95	0.84	0.89
Cough	0.86	0.96	0.91
Overall Accuracy	90%		

Table 5.3: Int8 Accuracy Measurement.

		Predicted	
		Background	Cough
Actual	Background	42 (TN)	8 (FP)
	Cough	2 (FN)	48 (TP)

Table 5.4: Int8 Confusion Matrix for Background vs Cough Classification.

Discussion of Test Results

The performance of the cough detection model was evaluated using both the original float32 version and the quantized int8 TensorFlow Lite version to assess accuracy,

precision, recall, and F1-score. The original model achieved an overall accuracy of 93%, surpassing the accuracy benchmark and exhibiting strong classification capability. It yielded a precision of 92% and a recall of 94% for the background class, while the cough class achieved a precision of 94% with a recall of 92%. This indicates that the model has high accuracy while maintaining a high proportion of correctly identified samples. The corresponding confusion matrix for the float32 model confirms the high accuracy and low error rate of the model by highlighting the low fraction of FP and FN.
Following this, the model was quantized to int8 format using TensorFlow Lite to reduce the model size significantly, making it suitable for deployment on the resource-limited ESP32 microcontroller. The quantized version maintained a strong performance and achieved an overall accuracy of 90%. Although the quantized model experienced a slight reduction in accuracy compared to the float32 version, it maintains an accuracy above the requirement, indicating that the conversion did not substantially degrade model performance. The int8 version achieved a background precision of 95% and a recall of 85%, while the cough class achieved a precision of 86% with a recall of 96%. The corresponding confusion matrix reflected a continued low rate of misclassification but saw an increase in FP compared to the float32 version. Overall, the quantized version shows a favorable tradeoff between accuracy and size, and maintains a robust performance that passes the requirements.

5.2 Cough Detection Testing

To evaluate the efficiency of the cough detection system, numerous tests were carried out. The system was tested under several distinct conditions to assess both accuracy and robustness in varying real-world environments. Each condition simulates a unique use-case scenario that the wearable device may encounter during regular operation.
For all tests of the cough detection system, sounds were played from a phone. This was done to ensure that tests could be replicated with the same background noise, but also because the sound properties of fake and real coughs differ, and it might be hard for us to spontaneously produce real coughs. The links and names to all the videos with sounds played can be found in Appendix C.
The mic was in all test, unless otherwise stated, placed on a table with the phone playing the sound 20 cm away. This distance was chosen as it was determined after measurements to be a likely distance between the chest (where the microphone will be placed) and the mouth and nose of the users.
Unless otherwise mentioned, three different videos were used for each testing, each once, to ensure that the system worked for different types of coughs. The following tests were conducted:

- **Coughing:** Videos with coughing sounds but little to no background noise were used to determine the accuracy in ideal sound conditions. Each video was tested three times to determine the reliability of the system.
- **Coughing with background noise:** When the system is implemented in the real world, it will have to discern coughs even with high amounts of background noise. Therefore, another test was conducted in which the same videos as in

the previous test were used, but a video with background noise was played in the background. The phone playing the background noises was placed on the side, as shown in Figure 5.1, to simulate noises coming from an outside source. Three different background noise videos were used, and each was played once for each coughing video.

- **Coughing at different distances**: To assess the system's sensitivity relative to the user's proximity, the device was tested at three distances: 10 cm, 15 cm, and 20 cm from the sound source. This evaluates how the strength of the audio signal affects model performance.
- **Silence:** Previous tests have been intended to determine false negative rates, but more tests were conducted to find the false positive rates. The first of these was done by testing in complete silence.
- **Background noise:** Ambient sound was introduced during testing to evaluate how well the system maintains accuracy in noisy, everyday environments. The phone was placed like phone 2 in Figure 5.1.
- **High frequency noise:** To determine whether the model confuses high-frequency noise signals with coughs due to spectral overlap, a test with high-frequency noise being played was done.

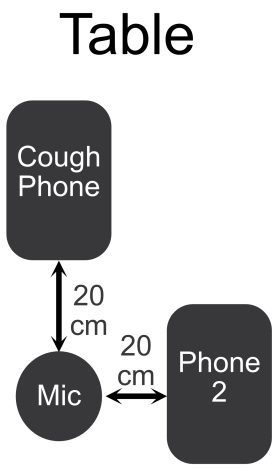


Figure 5.1: Phone placement relative to microphone.

Together, these test scenarios provide a comprehensive view of the system's detection capabilities under both ideal and challenging conditions. The next section present the results in detail, followed by an analysis of system performance and limitations.

Test Results

In Table 5.5, the results from the first cough detection accuracy test are presented. Cough 1 to 3 refer to the different cough sound videos, while Trial 1 to 3 corresponds to separate test iterations for each video. The values in the table represent the detection accuracy (in percentage) for each trial. An average accuracy for each video across all three trials is also shown.

	Cough 1 (%)	Cough 2 (%)	Cough 3 (%)
Trial 1	79	82	86
Trial 2	67	79	83
Trial 3	64	86	86
Average	70	82	85

Table 5.5: Average accuracy for each video and trial.

Presented in Table 5.6 is the accuracy for the same three cough videos, but with three different background noises. As in the previous table, individual results as well as averages are shown. This test tests the robustness of the system in more realistic, noisy environments.

	Cough 1 (%)	Cough 2 (%)	Cough 3 (%)
Background 1	59	67	61
Background 2	50	77	68
Background 3	52	73	67
Average	54	72	65

Table 5.6: Accuracy with different background noises.

To investigate the effect of the distance between the microphone and the sound source, the system was also tested at three different distances from the phone: 20 cm, 15 cm, and 10 cm. The results for these tests are summarized in Table 5.7.

Distance (cm)	Cough 1 (%)	Test 2 (%)	Test 3 (%)	Average (%)
20	72	81	81	78
15	78	82	86	82
10	83	86	90	86

Table 5.7: Cough detection accuracy at different distances (3 trials per distance, each 5 minutes).

To assess the rate of false positives, the system was tested in a completely silent environment, without any coughing. The results, shown in Table 5.8, reveal the percentage of false positives for three different trials.

Condition	Test 1 (%)	Test 2 (%)	Test 3 (%)	Average (%)
Silent (No Coughs)	23	29	27	26.3

Table 5.8: False positive cough detections during silent 5-minute intervals (no coughing present).

A similar test was performed with different background noises to determine the false positive rate under non-silent, real-world conditions. The outcome of these tests is shown in Table 5.9 below.

Test 1 (%)	Test 2 (%)	Test 3 (%)	Average (%)
38	27	32	32

Table 5.9: False positive rate for different background noises.

For the final test, a high-frequency tone was played in the background. This was done three times, and the results are shown in Table 5.10.

Test 1 (%)	Test 2 (%)	Test 3 (%)	Average (%)
0.29	0.21	0.32	0.27

Table 5.10: False negative for high frequency background noise.

The actual data from the testing can be found in appendix B.

Discussion of Test Results

The results above provide a general idea of the performance and limitations of the cough detection system in different conditions. The baseline accuracy shown in Table 5.5 indicates a relatively high average detection accuracy of between 70% and 85%. The accuracy was however significantly higher for coughs 2 and 3, at 82% and 85% respectively, than for video 1, which had an accuracy of 70%. This might be because there was more breathing between the coughs in this video, which might have confused the AI.
When background noise was introduced, the detection accuracy declined across all cough types, as shown in Table 5.6. Cough 1 dropped to an average of 54%, while coughs 2 and 3 fell to 72% and 65% respectively. This indicates that environmental noise negatively impacts the model's performance, which is especially a problem for cough 1 when the accuracy was already relatively low. Despite the background noise, cough 2 still maintained a relatively robust performance. The average accuracy across the different videos is still low enough that this will be a problem, as there is almost always background noise in real life.
The distance between the sound source and the microphone also played a significant role in accuracy, which can be seen in Table 5.7. As the distance decreased from 20 cm to 10 cm, the average detection accuracy improved from 78% to 86%. This shows that the model performs best with close-proximity input. This is probably due to increased signal clarity. The exact placement of the band around the chest and ensuring it is as close as possible to the person's mouth is, therefore, very important to consider in the final product. For this reason, the microphone was placed on the top of the casing, as mentioned in section 4.5.2.
False positive rates were also assessed both under silence and background noise scenarios. In complete silence, the system recorded a relatively high average false positive rate of around 26.3%, which is shown in Table 5.8. When background noise was present, the false positive rate was even higher, at an average of 32% (Table 5.9). This indicates a tendency to incorrectly classify ambient or internal sounds as coughs. Further adjustments of the case and filtering need to be done in the future to minimize this.

A similarly high false negative test rate, of around 27% (Table 5.10) was found when a high frequency noise was played. The false negative rate is however not higher than for the other background noise conditions, indicating that high frequency noise specifically might not be a problem.
Overall, while the cough detection system has a strong baseline performance and a high accuracy in optimal conditions, it is significantly impacted by noise and distance. This may be a significant problem for the real-world asthma monitoring system, as the conditions are then usually far from ideal.

5.3 Breathing Rate Testing

The initial phase of testing involved constructing a prototype wearable device incorporating a conductive rubber band sensor. This prototype enabled real-time data collection to evaluate the sensor's response to chest expansion during breathing.
To measure voltage changes from the conductive rubber, the sensor was implemented as part of a voltage divider circuit connected to an analog input on the ESP32 microcontroller. This setup allowed the system to record analog values corresponding to the sensor's resistance changes as the band stretched.
Early testing involved placing the prototype on the chest and recording voltage over time while the user remained stationary. Data was collected under various sensor placements and levels of band tension. The ArduinoFFT library was used to perform frequency-domain analysis on the voltage readings. FFT was chosen for its ability to detect periodic signals (like breathing) despite variations in amplitude or background noise.
The testing procedure also involved deliberately introducing real-world conditions such as light movement (e.g., walking or talking) to examine how the signal responded.

Test Results

Initial testing was performed using a prototype wearable device equipped with a conductive rubber band sensor to evaluate whether it could reliably detect breathing. This early test provided promising results, as shown in Figure 5.2, where the voltage signal demonstrated noticeable variation corresponding to chest expansion and contraction during normal breathing.
After the initial validation, two more extensive tests were conducted to assess the system's performance under different conditions. In the first test, the user was sitting still and breathing normally in a calm environment. As seen in Figure 5.3, the voltage output showed a clean, sinusoidal waveform that clearly followed the user's breathing cycle. These results confirmed the system's ability to accurately detect breathing rate in ideal conditions.
To simulate more realistic use, a second test was carried out while the user was talking and moving lightly over a longer period of time. In this case, the signal became more irregular due to motion artifacts and minor shifts in the position of the sensor. As shown in Figure 5.4, the waveform was noisier, but still retained

a periodic structure. Despite the reduced clarity, FFT analysis was still able to identify the breathing frequency with reasonable stability.

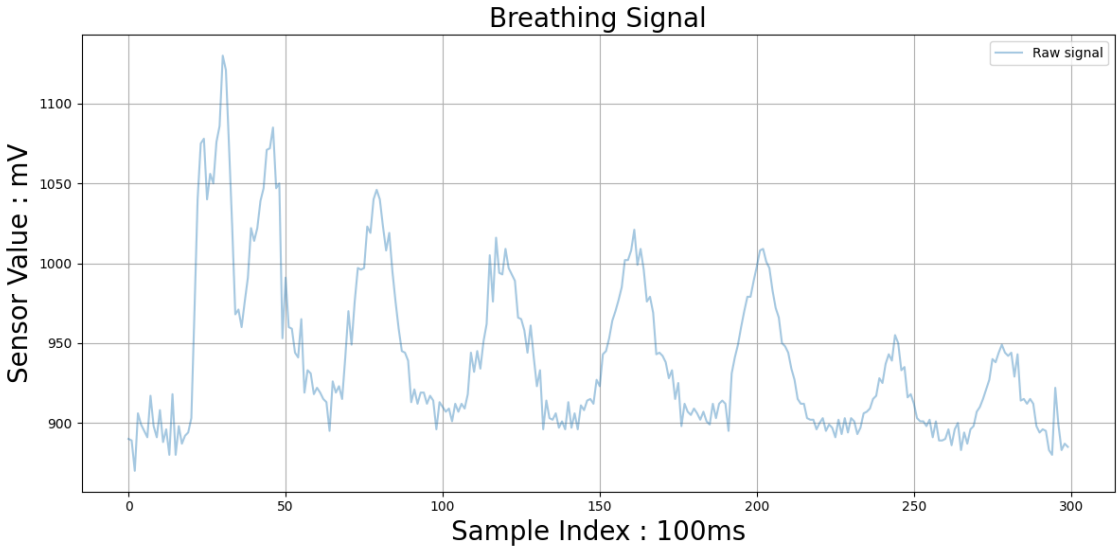


Figure 5.2: Initial voltage from the conductive rubber band during testing.

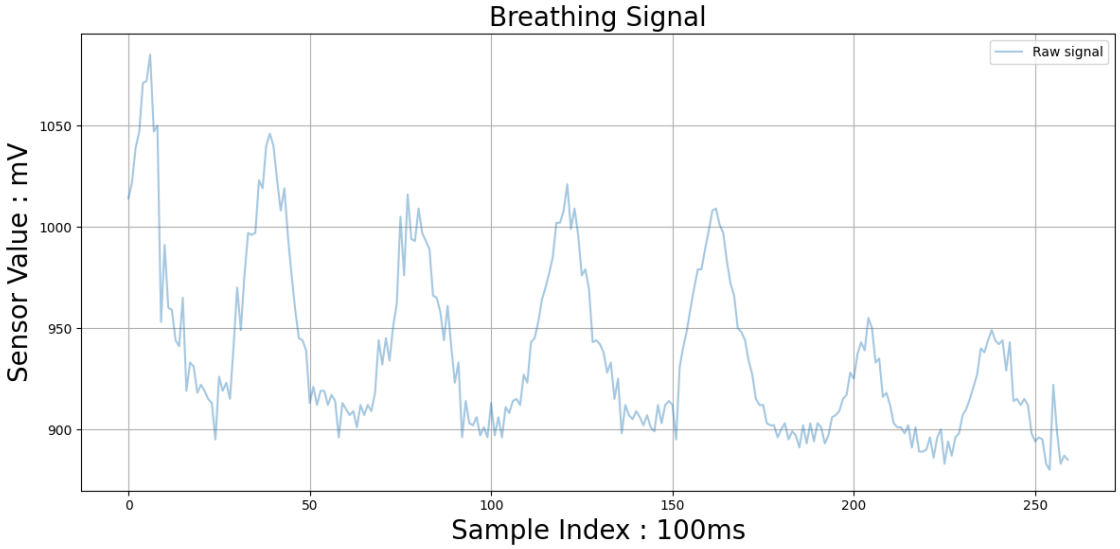


Figure 5.3: Sensor output under good conditions, showing a clean, sinusoidal waveform.

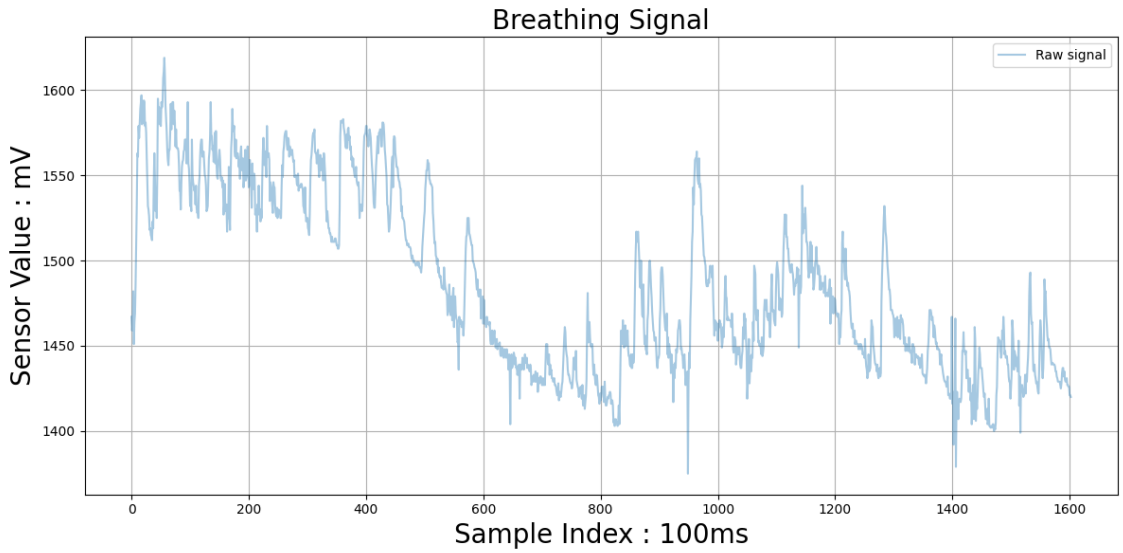


Figure 5.4: Sensor output under poor conditions, showing a noisy but somewhat periodic signal.

Discussion of Test Results

The initial phase of testing with the prototype wearable device revealed several important insights regarding the reliability and usability of the system. As shown in Figure 5.2, early data collection identified specific limitations in the mechanical design of the wearable. Notably, the conductive rubber band was observed to slip downward slightly during testing, resulting in a noticeable decrease in signal amplitude over time. Additionally, the act of putting on the wearable introduced a sudden spike at the beginning of the voltage graph, likely caused by a combination of abrupt stretching and sensor stabilization. These artifacts highlight the importance of secure and consistent sensor placement, as even small changes in position can significantly affect the quality and consistency of the measured signal.

Following the initial validation, two more structured tests were conducted. The first of these involved monitoring breathing under ideal, controlled conditions, specifically, the user sitting still and breathing naturally. As illustrated in Figure 5.3, the output signal exhibited a smooth, sinusoidal waveform that closely followed the user's respiration pattern. This clear periodicity confirmed that the system is indeed capable of accurately detecting breathing rate when the sensor is properly positioned and free from external disturbances. These findings emphasize the potential of the approach, provided that the wearable device is fitted securely and the environment remains stable.

In contrast, the second test introduced more realistic, everyday scenarios, including light movement and speech. This real-world test yielded more complex and irregular data, as seen in Figure 5.4. A notable drop in baseline voltage occurred mid-way through the recording, most likely the result of the band slipping or shifting position as the user moved. Moreover, the graph contains several sharp spikes, which could be attributed to factors such as the sensor bouncing during walking or rapid chest movements associated with speaking. Talking also disrupts regular breathing

patterns, introducing further variability into the signal.
These observations underscore the challenges associated with extracting clean respiratory signals outside of controlled environments. They illustrate how even mild user activity can introduce noise, motion artifacts, and inconsistent readings. It is important to note that this testing did not include more aggressive activities, which could further degrade signal quality. As such, while the system performs well under optimal conditions, achieving accurate and reliable measurements in dynamic, real-world settings remains a significant challenge. These results point to the necessity for improved sensor stability and mechanical design if the device is to be deployed for continuous monitoring in everyday use.
Lastly, as seen in Figure 5.2 and 5.4, the amplitude varies significantly. This large variation is the main reason why measuring breathing depth could not be reliably performed. For the amplitude to represent any meaningful physiological value, such as breathing depth, the system would require recalibration each time the band experiences a notable shift in positioning or tension.

5.4 Application Testing

To verify that the application will interact reliably with the microcontroller, a full end-to-end test is first executed using a BLE simulator, which emulates the device's services. The test is designed to test both the BLE functionality of the application, while also testing other key functions of the application.
Testing was carried out using a purpose-built BLE simulator that emulates the microcontrollers expected behavior. The simulator advertises a primary service with four characteristics, Breath, Cough, Depth and Emergency. Each characteristic is able to deliver timestamped JSON data packets via the notify operation. Three representative datasets per characteristic were included to reproduce realistic operating conditions. The application was launched, BLE was enabled, and React Natives built-in performance monitor was started. The connection logic of the application scans continuously for the advertised SERVICE_UUID and establishes a connection automatically when the simulator is started. From that point, numerous function tests are carried out in the following order:

- The simulator repeatedly sent data on all four characteristics via the notify operation.
- The application detected each notification.
- All data is displayed in the application's statistics tab.
- The home tab was loaded, and a SuspectedAttackSurvey was done.
- All data were downloaded using the statistics tab.
- The simulator repeatedly sent data on the emergency mode characteristic during 10-second intervals.
- The simulator was closed, and stopped sending data on the emergency mode characteristic. The application terminates the connection.
- The simulator is started again, and again starts sending data on the emergency mode characteristic. The application re-establishes the connection.

Lastly, the size of the application was noted, along with the peak and average RAM usage.

The primary objectives of this test were to first verify that all core feature of the application operated as intended. Second, to confirm reliable BLE behavior by ensuring scanning, connection management, and data handling all work correctly with a simulated peripheral device. Third, to capture performance metrics.

Test Results

The initial test, where the simulator sent data on all four characteristics as soon as possible, revealed a bottleneck in the application. Every notification generated by the notify operator, except the very first one, was silently dropped, and only a single data point was received by the application. In the following tests, a 200 ms delay was added between each call to the notify operator. This delay gave enough time for the application to receive and parse each notification, resulting in a 0% packet loss for subsequent tests.
Once the delay had been implemented and a 0% packet loss was achieved, the remaining functional tests were executed. All notifications and data were now received, parsed, and stored correctly. The feedback given in the home tab changed according to what data the simulator was sending, and the graphs in the statistics tab displayed them as expected. Using the download section in the statistics tab resulted in a JSON file with the values transmitted by the simulator added. When the simulator continuously sent data on the emergency mode characteristic, the application kept the connection open. Once the app stopped sending data on this characteristic, the application closed the connection and attempted to reconnect. When the simulator was restarted, a new connection was established. The final size of the application was noted down to 30 MB, average RAM usage to 117 MB, and peak RAM usage to 155 MB.

Discussion of Test Results

The end-to-end evaluation confirms that the core logic of the application functions correctly once given sufficient time to process incoming packets. The total loss of notifications during the initial test points to a queue handling or parsing bottleneck inside the application. However, by introducing a 200 ms delay between each packet sent by the simulator, reliable communication between the application and the simulator was achieved. The results of the subsequent functional tests demonstrate that once the data is received by the application, it is stored, utilized and displayed properly. Likewise, the reconnection logic performs as intended, executing automatic disconnects and reconnects when the simulator is turned on and off. Overall, testing with a simulator confirms that the application meets its stated goals, but it also exposed a latent risk. If notifications are sent in quick succession, they might end up being silently dropped. Mitigating that vulnerability, either by adding a delay or by optimizing the parsing done by the application, will be essential before the test with a real peripheral device.

5.5 Full System Testing

The purpose of the full system testing is to evaluate the integrated performance and functionality of the entire wearable cough detection and breathing monitoring system under realistic usage scenarios. This testing assesses how effectively all components interact when assembled into a cohesive unit and worn by an individual in a stationary position.
The test was carried out by first assembling the microphone, microcontroller, battery, and conductive rubber sensor into the wearable case, which was then securely positioned around the test subject's chest using adjustable straps. The microcontroller was powered on, initializing the system and beginning sensor data acquisition. Following this, the accompanying mobile application was launched, initiating a BLE connection with the microcontroller. Once the BLE connection was successfully established, the test subject was instructed to breathe naturally while periodically coughing at random intervals to simulate realistic use cases. The number of coughs produced and the number of breaths per minute taken by the test subject were noted down. Throughout the test, the microcontroller continuously collected and processed sensor data, periodically transmitting the processed cough detection results and breathing rate via BLE to the mobile application. The application then received, parsed, and integrated this data into its internal risk algorithms, updating and displaying the collected data through visual graphs and user interface elements in real-time. Lastly, the microcontroller is reset, and the application closes and re-establishes the connection.

Test Results

Once both the application and the microcontroller were started, a BLE connection was correctly established, and the application immediately began receiving data on the emergency mode characteristic. After one minute of operation, the microcontroller transmitted data indicating the number of coughs detected. However, this count did not match the actual number of coughs produced by the test subject. After another minute passed, the microcontroller transmitted breathing rate data to the application. Similar to the cough data, the breathing rate reported did not align with the subject's actual breathing rate. Despite receiving incorrect data, the application successfully parsed and incorporated this information into its risk algorithms and displayed it within the relevant graphs. Furthermore, when the microcontroller was turned off, the application correctly managed the disconnection by ending the connection and promptly attempting to re-establish it once the microcontroller was powered back on.

Discussion of Test Results

The results from the full system test show that the device successfully meets the fundamental integration and BLE communication goals. However, significant deviations from other goals were observed, particularly regarding breathing rate and cough detection. Neither the measured breathing rate nor the detected cough fre-

quency corresponded accurately to the actual physiological events occurring during the test, falling short of the system's requirements.
Extensive debugging indicated that the root cause of the breathing rate inaccuracies stems from the microcontroller requiring approximately 5 seconds to execute the cough detection algorithm, thus disrupting the sampling frequency of data from the conductive rubber sensor. This disruption leads to unpredictable and unreliable results from the FFT-based breathing rate calculations. Furthermore, the inaccuracies observed in cough detection align closely with previously identified problems detailed in Section 5.2, particularly the model's susceptibility to environmental noise and false-positive classifications.

5.6 Technical Challenges

The technological components of the wearable system were selected and integrated with a focus on balancing functionality, energy efficiency, and user comfort. In this section, there is an analysis of key takeaways from the design process and testing results.

5.6.1 Issues with Background Noise

As previously mentioned, the system had a limited ability to detect coughing accurately when background noise was present. Its accuracy dropped significantly when it was tested with background noise present. One factor behind this could be a lack of background noise in the coughing sounds that the AI was trained on, which could have made it more sensitive to less than ideal conditions. Additionally, he choice of microphone could also have effect, since the INMP441 is omnidirectional and thus captures sound equally from all directions. This makes it more likely to pick up background noise, which might drown the cough noise.

5.6.2 Elasticity Issue with the Rubber Band

One of the main challenges encountered during development was related to the nonlinear elasticity and mechanical behavior of the conductive rubber band used for measuring chest expansion. During testing, it became evident that the sensor's output was highly dependent on the initial tension of the band. When transitioning from a completely unstretched to a stretched state, the sensor produced strong and measurable changes in resistance. However, once the band had been pre-stretched by approximately 10% to 20%, further expansion resulted in significantly weaker signals. This indicated that most of the measurable resistance change occurred only within the initial stretch range.
This behavior introduced practical limitations to both data reliability and user comfort. In order to secure the band in place on the chest, users were initially required to stretch the band tightly, which inadvertently moved the sensor out of its optimal measurement range. To address this, an elastic strap was introduced to help fix the sensor in place without requiring excessive stretch. This solution allowed the

sensor to remain within its most responsive range while also improving comfort and consistency across users.
Furthermore, the inconsistencies in baseline resistance due to fitment made it difficult to reliably measure breathing depth. The absolute voltage values from the voltage divider circuit varied depending on how tightly the band was worn, making calibration difficult without manual user input. As a result, depth measurement was deprioritized in the final prototype, with a focus placed instead on reliably detecting breathing rate using frequency-domain analysis.

5.6.3 Choice of Microcontroller

There were several issues regarding the choice and use of a microcontroller in this project, particularly concerning memory capacity. We initially used an ESP32 Feather V2, but later discovered it was not capable of compiling or running the AI model. The primary reason for this was its limited RAM and the lack of external PSRAM, which made it impossible to store both the model and the required tensor arena in memory. Although the microcontroller had enough flash memory, TensorFlow Lite Micro does not run models directly from flash; it must load them into RAM, which the Feather V2 simply could not support due to its limited RAM. This limitation was not immediately obvious and caused a significant bottleneck in development.
We attempted to address this by quantizing the model, reducing its size by roughly 200%. However, even after optimization, the model still required more memory than the Feather V2 could provide. Consequently, we decided to purchase a new microcontroller, the LOLIN S3 Pro, which includes external PSRAM and has a higher memory ceiling.
Even after switching to the LOLIN S3 Pro, we encountered a new issue: the AI model failed to allocate memory during runtime due to exceeding the TensorFlow Lite Micro tensor arena size. Despite increasing the arena size significantly, AllocateTensors consistently failed. This was eventually traced back to the model being only partially quantized, it still contained floating point operations, which consumed a disproportionate amount of RAM during inference. Additionally, we discovered that including the model as a c-array caused it to be stored in RAM at boot, rather than being accessed from flash or loaded dynamically into PSRAM. As a result, both the model and its tensor arena competed for memory resources, causing inference setup to fail.
In retrospect, it would have been better to simulate the microcontroller on PlatformIO before buying hardware. We also should have invested more effort early in the project to optimize the AI model, since its size is the most critical factor for the microcontroller. Additionally, our group assumed the hardware was needed immediately, even before preliminary programming was complete, which proved unnecessary and ultimately hindered progress.

5.6.4 Complications During Integrated Testing

The challenges regarding integrated testing can be categorized into two sections: Cough detection testing and breathing rate detection.
Several challenges arose during the integration of the cough detection system. The main problem was inaccurate detection during testing. Various methods were applied to identify and resolve the issue.
First, we investigated whether the root cause was the AI model itself, the microphone code, or the logic that processed and passed data to the AI model. To determine if the model was functioning correctly, we recorded a 20 second audio sample using the microcontroller's microphone. The AI model achieved an accuracy of 82%, which, although slightly lower than expected, confirmed that the model was still capable of detecting coughs. This led us to conclude that the most likely issue was incorrect input formatting.
A long debugging process followed, involving simulations, tests, and repeated evaluations. The issue was ultimately resolved by removing a condition that only allowed every other FFT frame to be stored in the spectrogram. Once this restriction was lifted, the code began storing every frame, ensuring that the spectrogram was fully populated and correctly matched the input shape expected by the AI model.
The second critical fix involved normalizing the raw audio samples by dividing them by 32768.0f. This step scaled the 16-bit integer values into the expected float range of approximately -1.0 to 1.0, aligning the microphone input format with the format the AI model was trained on.
However, when we conducted full system tests, a new issue became apparent. We found that the AI inference calculations took approximately five seconds to complete. This caused the inference process to block the rest of the code from running continuously. As a result, the band could no longer gather data every 100 ms as it was originally intended to do. Instead of collecting evenly spaced samples over one minute, the data was now sampled in inconsistent batches and spread out over an unknown period of time. This introduced 5-second "cuts" or delays in the data stream, which disrupted the timing and integrity of the collected data. Such interruptions posed significant problems for the system's real-time functionality and its ability to deliver consistent and reliable outputs.

5.7 Potential Improvements

This section highlights potential improvements to the SmartSense prototype, focusing on enhancing sensor accuracy, cough detection reliability, and overall system integration. These enhancements would substantially elevate the system's robustness, reliability, and real-world applicability.

5.7.1 Proposed Improvement of the Conductive Rubber Band

One potential improvement for the rubber band sensor would be to explore alternative materials or sensor types with more linear resistance profiles across their stretch range. This could reduce the sensitivity to initial tension and improve the accuracy

of depth measurements. Additionally, implementing a dynamic calibration feature, either through software or user-guided setup, could compensate for differences in fitment and baseline resistance. Mechanical improvements, such as a better-designed enclosure or elastic harness that ensures consistent placement and tension, could also enhance repeatability and user comfort. If future versions aim to include breathing depth, a multi-sensor approach or improved preprocessing techniques may be necessary to isolate meaningful signal components from physical noise.

5.7.2 Proposed Improvements to Cough Detection

While the SmartSense prototype demonstrates the feasibility of real-time cough detection on a microcontroller, there are several areas for improvement to enhance its accuracy and robustness. The primary limitation lies in the size and diversity of the training dataset. The current AI model was trained on only a few hundred samples, which significantly limits its ability to generalize across real-world conditions. In particular, it struggles with false positives in the presence of non-cough high-frequency sounds and environmental noise. Expanding the dataset to include a broader range of coughs, backgrounds, and user variations would enable the model to better differentiate between actual coughs and acoustically similar events.
In addition, improvements can be made in the signal processing pipeline. The current implementation lacks advanced filtering techniques to isolate relevant acoustic features. Incorporating noise suppression, adaptive filtering, or even traditional digital filters (e.g., band-pass or notch filters) could improve the clarity of the input to the AI model and reduce misclassifications.
Furthermore, the cough detection system did not work well when background sounds were present. It may be possible to improve this by for example using multiple microphones, to help isolate sounds from a specific direction or by picking a microphone that was not omnidirectional.
It is also important to note that this project represents a proof of concept. To transition to a commercial-grade product, significantly more testing is required. This would include long-term validation across diverse environments, user profiles, and device placements, testing for variables such as distance from the mouth, microphone orientation, background activity, and even fabric occlusion. Evaluations over extended periods (weeks or months) would be necessary to assess consistency, durability, and performance under varied usage scenarios.

5.7.3 Proposed Improvements to Full System Integration

One of the main issues identified during full system testing was the blocking behavior caused by the AI inference process. This delay stemmed from the relatively high computational demands of running the inference model compared to the processing power of the current microcontroller. As a result, while inference was being calculated, the processor was unable to execute other tasks simultaneously, which disrupted the real-time data collection process.
The primary bottleneck here lies in the limited performance capabilities of the exist-

ing hardware. The inference step, taking approximately five seconds, prevented the system from gathering data at the intended 100 ms intervals, leading to the timing and sampling inconsistencies described earlier. This is because the system performs float calculation which require a large amount of processing power.
A proposed improvement is to upgrade the microcontroller to a model with a faster processor or enhanced computational support for machine learning tasks. For instance, selecting a microcontroller that includes a dedicated hardware accelerator for AI inference or features a higher clock speed and improved parallelism could significantly reduce inference time. Alternativly, the input shape can be quantized to allow the microcontroller to perform int8 calculations, which would require a lot less processing. This would help maintain the intended sampling frequency and allow for smoother and more reliable system operation.

6

Resource Management, Sustainability, and Ethics

The evolution of medical wearables, especially those aimed at high-risk demographic categories like asthma patients, represents a multifaceted set of duties that goes far beyond technical competence. They include concerns about environmental sustainability, ethical data management, access, and societal implications. Throughout this chapter, we have performed a comprehensive review of these aspects in relation to the device that we have created.

6.1 Sustainability

This chapter examines the product's sustainability and explores ways to enhance it. Key areas of focus to do this include resource use, ecological impacts and strategies for extending the product's lifespan. Power management and energy efficiency will also be discussed.

6.1.1 Resource Use and Ecological Impacts

The wearable prototype consists of an ESP32 microcontroller, an I2S microphone, standard copper wiring, an elastic rubber strap, and a custom 3D-printed casing. The ESP32 microcontroller was selected for its low power consumption and integrated AI acceleration, making it ideal for edge processing of audio signals. The I2S microphone offers a compact and reliable method for high-fidelity cough detection. The wearable device is mainly made up of an elastic rubber band and the microcontroller. The use of rubber was driven by its stretchable, durable, and comfortable nature, especially suitable for extended use on the chest. Plastic components were necessary to offer safe housing and mechanical protection of the electronic components.
Despite being effective for prototyping, the current material choices have mixed sustainability profiles. Copper wiring, microcontroller, and I2S microphone are difficult to recycle, and the embedded nature of electronic components makes disassembly for end-of-life treatment challenging.
Microcontroller recycling typically follows six main steps:

1. **Collection and Transportation:** E-waste is collected from various sources and transported to specialized recycling facilities.

2. **Manual Disassembly:** Devices are dismantled by hand to remove microcontrollers and separate components for targeted recovery.
3. **Shredding:** Remaining parts are mechanically shredded into smaller pieces to ease further processing.
4. **Separation:** Advanced techniques such as magnetic separation, eddy current separation, and density-based methods are used to isolate materials.
5. **Material Recovery:** Valuable materials, such as copper, aluminum, and rare earth metals, are extracted and prepared for reuse.
6. **Waste Treatment:** Non-recyclable residues are safely disposed of, and hazardous substances are treated to minimize environmental impact [29].

However, microcontrollers are complex components often soldered onto PCBs alongside other parts, which makes separation difficult. Furthermore, they may contain hazardous substances such as lead, requiring careful handling. As a result, recycling becomes economically unfeasible without large-scale operations. This increases the risk that microcontrollers are discarded in landfills, where toxic substances can leach into the soil and groundwater. Other less ideal outcomes include incineration or stockpiling. Similar issues apply to the I2S microphone; its small size and embedded nature make it difficult to recover or recycle, contributing to environmental pollution if improperly discarded.
However, battery sustainability remains a challenge. Lithium-ion batteries face environmental risks from extraction methods, disposal, and possible chemical spillage [30, 31]. For these reasons, our design also includes a common JST plug along with USB-C, allowing recharging while ensuring future battery swaps, promoting reuse, and reducing e-waste.
A comprehensive LCA should be conducted before large-scale deployment to quantify the environmental footprint and guide material substitutions toward a more sustainable design.

6.1.2 Extended Duration of Product Use

Unlike disposable medical devices, this wearable is designed for repeated use over an extended period. All components, including the battery and stretch sensor, are mounted on a microcontroller board that can be serviced or replaced. While the microphone is soldered in place for signal reliability, it can still be replaced using basic soldering tools if needed. The charging circuitry is based on the TP4054, a linear Li-Po (Lithium Polymer) charging IC that ensures safe recharging via USB-C [23]
The enclosure is designed to be easy to dismantle, making it easier to change individual components as they age. This modular design not only increases product longevity but also aligns with a circular economy model, where reuse and refurbishment are preferred over disposal and replacement [32].
To support long-term functionality, the system is designed with firmware-level stability on the ESP32 microcontroller, ensuring reliable operation over time. In addition, the companion mobile application, built with React Native with the Expo framework, supports remote updates through standard app distribution platforms or OTA update mechanisms. This modular software architecture enables continuous

performance improvements and the addition of new features without necessitating hardware replacement, thereby extending the effective lifecycle of the device.

6.1.3 Power Management and Energy Efficiency

Energy efficiency was a central consideration throughout the development of SmartSense, especially given the system's reliance on a small Li-Po battery. Power management strategies were implemented both in hardware selection and software design to ensure the device could operate for a minimum of two hours on a single charge, as outlined in the design requirements.
The microcontroller alternates between two operating modes: normal mode, where data sampling and AI inference are performed locally without BLE communication, and TX mode, where BLE is activated to transmit data to the mobile application. To reduce overall power consumption, data transmission occurs every 15 minutes instead of continuously. During normal mode, the microcontroller remains in modem sleep with BLE disabled, while sensors such as the INMP441 microphone and the conductive rubber band are sampled.
BLE was selected because it consumes significantly less energy compared to Classic Bluetooth [33]. This efficiency is achieved through short, efficient communication bursts and long sleep periods, making BLE ideal for battery-powered devices like wearables and IoT sensors.

6.2 Power Calculations

The microcontroller swings between two different states: idle and BLE transmit output power. Every 15 minutes (900 seconds), the microcontroller sends data through BLE to the app. The maximum value will be used to calculate the highest power consumption the microcontroller will be charged with.
To calculate a theoretical power consumption, both states have to be taken into account.

Current Consumption During Low-Power Operation

In the idle state, we need to determine the total current consumption of all components. During this state, the microcontroller is running the AI model, continuously sampling data from the microphone and the band, while operating in modem sleep mode with the CPU active and BLE disabled.

Component	**Current**
ESP32-S3 Lolin	$I_{\text{modem-sleep}} = 51.2$ mA [34]
INMP441 Microphone	$I_{\text{mic}} = 2.5$ mA [22]
Band	$I_{\text{band}} \approx 0.79$ mA

Table 6.1: Idle state current consumption

Current consumption of band

The current through the band can be calculated as shown in Equation 6.1.

$$I_{\text{band}} = \frac{3.3\,\text{V}}{2.1\,\text{k}\Omega + 2.1\,\text{k}\Omega} \approx 0.79\,\text{mA} \tag{6.1}$$

The total idle current is given by Equation 6.2:

$$I_{\text{idle}} = I_{\text{modem-sleep}} + I_{\text{mic}} + I_{\text{band}} = 53.79\,\text{mA} \tag{6.2}$$

Current Consumption During Data Transmission

Table 6.2 shows the value of TX state current consumption.

Component	Current
ESP32-S3 Lolin (transmitting)	$I_{\text{TX}} = 355\,\text{mA}$ [34]

Table 6.2: TX state current consumption.

Theoretical Foundation for Average Current Estimation

The average current consumed by a microcontroller system operating in multiple modes, such as transmit, idle, and sleep, can be calculated using duty-cycle weighting. This method is conceptually similar to pulse-width modulation (PWM) current averaging. The formula as seen in Equation 6.3 for estimating the average inductor current in a PWM signal is given as:

$$I_{\text{avg}} = \frac{t_{\text{on}}}{T} \cdot \frac{V}{R} \tag{6.3}$$

where t_{on} is the active time, T is the full PWM period, and V/R represents the peak current. This structure aligns conceptually with the generalized average current formula used in power budgeting as shown in Equation 6.4:

$$I_{\text{avg}} = \frac{\sum_i I_i\, t_i}{T} \tag{6.4}$$

Which calculates the average current draw across various operating states, each with current I_i and time duration t_i over a full cycle time T as shown in Equation 6.5 [35]. The full duty cycle time will be 900 seconds. The time duration of the idle state will be assigned 899 seconds, and the TX state will be assigned 1 second.

$$I_{\text{avg}} = \frac{132.2\,\text{mA}\cdot 1\,\text{s} + 26.1\,\text{mA}\cdot 899\,\text{s}}{900\,\text{s}} \approx 26.2\,\text{mA} \tag{6.5}$$

Battery-side Current with DC–DC Efficiency

The formula for battery current can be seen in Equation 6.2.

$$I_{\text{batt}} = \frac{V_{\text{out}}\ I_{\text{avg}}}{V_{\text{batt}}\ \eta}$$

(6.6)

The values of output voltage, battery voltage and buck convert efficiency can be seen in Table 6.3.

Parameter	Value
Output voltage (V_{out})	3.3 V
Battery voltage (V_{batt})	3.7 V
Buck converter efficiency (η)	90 %

Table 6.3: Voltage levels and converter efficiency.

A buck converter[36] is used to regulate the battery's power by the board. Hence, the lowest efficiency is calculated to be 26.0 mA, as seen in Equation 6.7.

$$I_{\text{batt}} = \frac{3.3\,\text{V} \cdot 26.2\,\text{mA}}{3.7\,\text{V} \cdot 0.90} \approx 26.0\,\text{mA} \tag{6.7}$$

Runtime

The capacitance of the LiPo battery is 750 mAh [37]. The calculated runtime of the battery can be seen in Equation 6.8.

$$\text{Runtime (h)} = \frac{750\,\text{mAh}}{26.0\,\text{mA}} \approx 28.9\,\text{h} \tag{6.8}$$

Hence, the theoretical average current through the microcontroller will be approximately 26.2 A. The current through the battery will be 26 mA, and the estimated battery lifetime will be 28.9 hours. During emergency mode the battery consumption will be more demanding and will depend on how long Tx state will run. Nevertheless, the run time is still well above the requirement of 2 hour run time.

6.3 Ethical Considerations

This section discusses the ethical considerations involved in the development and use of the product, which is an essential aspect of responsible engineering and design. Proactively considering factors such as those presented in this section makes it possible to implement the product in an ethical and inclusive way.

6.3.1 User Consent and Data Privacy

Given the sensitive nature of health data, maintaining privacy and data integrity was of utmost priority during system design. All parameters, such as respiratory patterns, rate of cough, and instances of emergencies, are stored locally in the mobile app on the user's smartphone. There is no real-time transmission of health data to third-party servers, cloud storage mechanisms, or external parties.

The specialized data storage system allows total management of healthcare data by individuals. Users are empowered to share their historical data with healthcare professionals, enhancing diagnostic activities while preserving privacy.
From an ethical standpoint, this approach aligns with GDPR principles and medical confidentiality [38]. In the future, if cloud synchronization is implemented, it will require explicit user consent, end-to-end encryption, and anonymization protocols to uphold ethical standards.

6.3.2 Accessibility and Inclusion

Usability was a key consideration in the development of both the hardware and software components of the device. The wearable strap is made from elastic material and designed to ensure comfort and adaptability for most users. However, some groups, such as individuals with limited motor function, obesity, or increased skin sensitivity, may still experience challenges when using the current version. These limitations highlight the importance of ongoing ergonomic testing and inclusive design practices to ensure accessibility across all user demographics [39].
In terms of software, the mobile application has a clear interface suitable for users of different ages and levels of technological expertise. Nevertheless, challenges remain for those unfamiliar with smartphone technology or those who don't speak English.

6.3.3 Fairness and Prejudice in Algorithmic Systems

The detection model uses a trainable convolutional neural network based on spectrograms of recorded coughs. While MobileNetV2 enables computational efficiency on wearable hardware, the model's accuracy depends heavily on the diversity of its training data.
If the dataset is skewed, e.g., overrepresenting certain age groups, genders, or accents, algorithmic performance may vary across user groups. This could result in lower sensitivity for some individuals, contributing to disparities in health monitoring outcomes. Augmenting the dataset with more representative audio samples and varying environmental noise contexts is a critical follow-up to address these issues [40].

6.4 Impact on Public Health

Asthma is a chronic respiratory disorder that affects a significant portion of the global population [3]. Our wearable technology offers real-time monitoring capable of detecting early signs of respiratory issues, such as abnormal breathing or persistent cough, thereby improving patient prognosis.
The device enables users to monitor symptoms and take precautionary actions, potentially preventing severe asthma attacks and reducing the need for urgent medical care. The ability to share objective data with health professionals further enhances diagnostic accuracy and facilitates personalized treatment. As a result, the system improves both individual health outcomes and healthcare system efficiency.

7
Conclusion

This project focused on designing and building SmartSense, a low-cost, AI-driven wearable prototype aimed at real-time detection of asthma-related symptoms through cough and breathing pattern analysis. By combining a conductive rubber sensor, a digital microphone, and an ESP32-S3 microcontroller running an embedded neural network, the system sought to provide users with early warnings via a companion mobile application. The application enabled real time visualization, symptom logging, and alerts, forming a self contained ecosystem for asthma symptom tracking.
The prototype demonstrated strong performance at the subsystem level. The AI-based cough detection model exceeded the target accuracy threshold, breathing data was captured reliably under controlled conditions, and BLE communication was established with the mobile app. However, serious challenges arise during the complete integration of the system. Simultaneous AI inference, sensor data collection, and BLE transmission occasionally led to timing conflicts, delayed updates, and communication instability. These issues limited the prototype's reliability in continuous real-time use and highlighted the complexity of synchronizing multiple tasks on a resource-constrained embedded platform.
While SmartSense confirmed that many of the design choices worked well individually, the prototype struggled to function reliably when all parts were combined. This outcome indicates areas for future research, underscoring the need to explore improved system architectures and investigate effective strategies for task management. Specifically, further studies should examine advanced timing control mechanisms, parallel processing capabilities, and robust integration methods to address these identified performance challenges.

Bibliography

[1] PNGTree. *Hand Holding Phone Illustration Transparent PNG*. Accessed: 2025-05-13. 2023. URL: https://pngtree.com/freepng/hand-holding-phone-illustration-transparent_7579126.html.

[2] Freepik. *Illustration of a Circuit*. Accessed: 2025-05-13. 2018. URL: https://www.freepik.com/free-vector/illustration-circuit_2606103.htm.

[3] WHO. *Asthma*. 2024. URL: https://www.who.int/en/news-room/fact-sheets/detail/asthma. [Accessed: 2025-03-02].

[4] Cleveland Clinic. *Asthma Attack*. 2024. URL: https://my.clevelandclinic.org/health/diseases/asthma-attack?utm. [Accessed: 2025-03-02].

[5] The Global Asthma Network. *The Global Asthma Report 2018*. URL: https://globalasthmareport.org/2018/resources/Global_Asthma_Report_2018.pdf. [Accessed: 2025-03-02].

[6] Sabrina Felson. *Asthma Attack*. 2021. URL: https://www.webmd.com/asthma/asthma-attack. [Accessed: 2025-04-18].

[7] American Lung Association. *Reduce asthma triggers*. 2024. URL: https://www.lung.org/lung-health-diseases/lung-disease-lookup/asthma/managing-asthma/reduce-asthma-triggers?utm. [Accessed: 2025-03-02].

[8] NHS. *Asthma Attacks*. 2021. URL: https://www.nhs.uk/conditions/asthma/asthma-attack/. [Accessed: 2025-03-02].

[9] Fortune Business Insights. *Wearable Medical Devices Market Size, Share & Industry Analysis, By Product (Diagnostic & Monitoring Devices Fitness Bands, Smartwatches, Smartclothing, and Others and Therapeutic Devices Wearable Defibrillators, Drug Delivery Devices, Pain Management Devices, Hearing Aids, and Others), By Application (Remote Patient Monitoring & Home Healthcare and Sports & Fitness), By Grade (Consumer Grade and Clinical Grade), By Distribution Channel (Retail Pharmacies, Online Distribution, and Hypermarkets & Others), and Regional Forecast, 2024-2032*. 2025. URL: https://www.fortunebusinessinsights.com/industry-reports/wearable-medical-devices-market-101070. [Accessed: 2025-02-13].

[10] Grand View Research. *Wearable Medical Devices Market Trends*. 2024. URL: https://www.grandviewresearch.com/industry-analysis/wearable-medical-devices-market. [Accessed: 2025-02-13].

[11] S.A. Alowais et al. "Revolutionizing healthcare: the role of artificial intelligence in clinical practice". In: *BME Medical Education* 23.689 (2023). URL: https://bmcmededuc.biomedcentral.com/articles/10.1186/s12909-023-04698-z. [Accessed: 2025-02-13].

[12] Spyros Papiris et al. “Clinical Review: Severe Asthma”. In: *Critical Care* 6.1 (2002), pp. 30–44. DOI: 10.1186/cc1451. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC137395/. [Accessed: 2025-05-06].

[13] Jocelin Isabel Hall et al. “The present and future of cough counting tools”. In: *Journal of Thoracic Disease* 12.9 (2020), pp. 5207–5223. DOI: 10.21037/jtd-2020-icc-003. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7578475/. [Accessed: 2025-05-06].

[14] Jerad Lewis and Paul Schreier. *Low Self Noise: The First Step to High-Performance MEMS Microphone Applications*. White Paper MS-2348. San Jose, CA: InvenSense, Inc., 2013. URL: https://invensense.tdk.com/wp-content/uploads/2015/03/Low-Self-Noise-The-First-Step-to-High-Performance-MEMS-Microphone-Applications.pdf. [Accessed: 2025-05-06].

[15] Aakanksha Chowdhery. *Visual Wake Words with TensorFlow Lite Micro*. 2019. URL: https://medium.com/tensorflow/visual-wake-words-with-tensorflow-lite-micro-8578e59ea6f9. [Accessed: 2025-05-06].

[16] Jieun Baek and Yosoon Choi. “Smart Glasses-Based Personnel Proximity Warning System for Improving Pedestrian Safety in Construction and Mining Sites”. In: *International Journal of Environmental Research and Public Health* 17.4 (2020), p. 1422. DOI: 10.3390/ijerph17041422. URL: https://doi.org/10.3390/ijerph17041422. [Accessed: 2025-05-08].

[17] Silicon Laboratories Inc. *EFR32BG13 Blue Gecko Bluetooth® Low Energy SoC Family Data Sheet*. Data Sheet Rev. 1.7. Austin, TX: Silicon Laboratories Inc., 2022. URL: https://www.silabs.com/documents/public/data-sheets/efr32bg13-datasheet.pdf. [Accessed: 2025-05-08].

[18] Zephyr Technology. *BioHarness™ 3.0 User Manual*. User Manual 9700.0079. Annapolis, MD: Zephyr Technology, 2012. URL: https://www.zephyranywhere.com/media/download/bioharness3-user-manual.pdf. [Accessed: 2025-05-08].

[19] Ada Fruit. *Conductive Rubber Cord Stretch Sensor + extras!* n.d. URL: https://www.adafruit.com/product/519#description. [Accessed: 2025-05-06].

[20] Knowles. *SPH0645LM4H-B MEMS Microphone Datasheet*. 2017. URL: https://mm.digikey.com/Volume0/opasdata/d220001/medias/docus/908/SPH0645LM4H-B.pdf. [Accessed: 2025-05-07].

[21] Global Audio Visual. *Choosing a Microphone: Specs Features*. n.d. URL: https://www.globalavisual.com/choosing-a-microphone-specs-features/. [Accessed: 2025-04-18].

[22] TDK InvenSense. *INMP441 MEMS Microphone Datasheet*. 2014. URL: https://www.digikey.se/htmldatasheets/production/1431884/0/0/1/inmp441-datasheet.html. [Accessed: 2025-05-07].

[23] LOLIN / WEMOS. *LOLIN S3 Pro: ESP32-S3 Development Board*. WEMOS Electronics. 2022. URL: https://www.wemos.cc/en/latest/s3/s3_pro.html. [Accessed: 2025-05-08].

[24] TensorFlow Team. *LiteRT for Microcontrollers*. 2024. URL: https://ai.google.dev/edge/litert/microcontrollers/overview. [Accessed: 2025-05-12].

[25] Mark Sandler et al. *MobileNetV2: Inverted Residuals and Linear Bottlenecks*. 2018. URL: https://arxiv.org/pdf/1801.04381v4. [Accessed: 2025-05-12].

[26] Himanshu. *Cough Audio Dataset*. 2022. URL: https://www.kaggle.com/datasets/himanshu007121/cough-audio-dataset/data. [Accessed: 2025-05-11].

[27] V. Marco, G. Massimo, and G. Manuela. "Additive manufacturing of flexible thermoplastic polyurethane (TPU): enhancing the material elongation through process optimisation". In: *Progress in Additive Manufacturing* 10 (2025), pp. 2877–2891. URL: https://link.springer.com/article/10.1007/s40964-024-00790-y?. [Accessed: 2025-05-03].

[28] ATPolymer. *Understanding TPU Material: Uses and Biocompatibility*. URL: https://www.atpchem.com/understanding-tpu-material-uses-and-biocompatibility.html. [Accessed: 2025-05-03].

[29] RTS. *The Complete E-Waste Recycling Process*. Recycle Track Systems (RTS). 2021. URL: https://www.rts.com/blog/the-complete-e-waste-recycling-process/. [Accessed: 2025-05-08].

[30] Greenly. *The Harmful Effects of our Lithium Batteries*. 2024. URL: https://greenly.earth/en-us/blog/industries/the-harmful-effects-of-our-lithium-batteries. [Accessed: 2025-04-29].

[31] Institute for Energy Research. *Environmental Impacts of Lithium-Ion Batteries*. 2023. URL: https://www.instituteforenergyresearch.org/renewable/environmental-impacts-of-lithium-ion-batteries/. [Accessed: 2025-04-29].

[32] United Nations Environment Programme. *Design for Disassembly as a Circular Economy Tool*. United Nations Environment Programme. 2023. URL: https://www.unep.org/resources/emerging-issues/sustainable-production-and-consumption-design-disassembly-circular-0. [Accessed: 2025-05-08].

[33] Sphero Team. *Bluetooth Low Energy Tech vs. Bluetooth*. 2023. URL: https://sphero.com/blogs/news/bluetooth-low-energy-vs-bluetooth. [Accessed: 2025-05-06].

[34] Espressif Systems. *ESP32-S3-WROOM-1 & WROOM-1U Datasheet, Version 1.4*. 2024. URL: https://www.espressif.com/sites/default/files/documentation/esp32-s3-wroom-1_wroom-1u_datasheet_en.pdf. [Accessed: 2025-05-06].

[35] Tim Wilmshurst. *Designing Embedded Systems with PIC Microcontrollers: Principles and Applications*. 2nd. Chapter 9.5.2: PWM and average current derivation. Newnes, 2010. ISBN: 978-0-08-096184-4. [Accessed: 2025-05-07].

[36] Monolithic Power Systems. *Buck Converters – DC-DC Converters | MPS*. 2025. URL: https://www.monolithicpower.com/en/learning/mpscholar/power-electronics/dc-dc-converters/buck-converters. [Accessed: 2025-05-12].

[37] Electrokit. *Batteri LiPo 3.7V 750mAh*. 2025. URL: https://www.electrokit.com/batteri-lipo-3.7v-750mah. [Accessed: 2025-05-07].

[38] European Union. *Artikel 9 GDPR – Behandling av särskilda kategorier av personuppgifter*. 2016. URL: https://gdpr--info-eu.translate.goog/art-

9-gdpr/?_x_tr_sl=en&_x_tr_tl=sv&_x_tr_hl=sv&_x_tr_pto=sc. [Accessed: 2025-05-05].

[39] Clint Zeagler. *Where to Wear It: Functional, Technical, and Social Considerations in On-Body Location for Wearable Technology 2.0*. 2017. URL: https://www.clintzeagler.com/where-it-body-maps/. [Accessed: 2025-05-08].

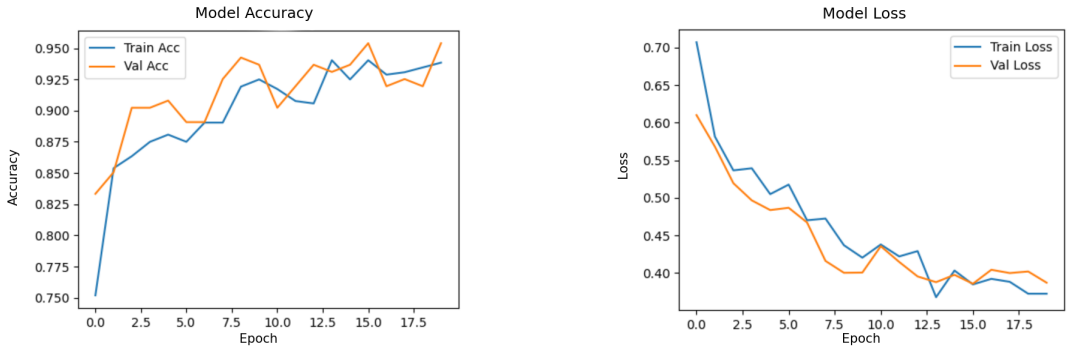
[40] Yanmin Qian. *Data augmentation using generative adversarial networks for robust speech recognition*. 2019. URL: https://www.sciencedirect.com/science/article/pii/S0167639319300044. [Accessed: 2025-04-29].

A

Appendix 1

This section contains the extended test results of the model accuracy and loss over each epoch during training.

A.1 Extended test Results



(a) Model Accuracy (b) Model Loss

Figure A.1: Model performance metrics: (a) Accuracy and (b) Loss. The model was trained using the following parameters. Epoch 20, Optimizer: Adam, Loss: Binary Crossentropy, Learning rate: 0.0005, Callback with patience 7 epoch and monitoring validation loss, Dynamic learning rate adjuster with factor 0.96 with patience 5 and monitoring validation loss, and a threshold of 0.5

B

Appendix 2

This section provides the data from the testing. The 1s represent when the AI detects a cough and the 0s when it do not detect a cough.

B.1 Coughing, no background noise

	Cough 1	Cough 2	Cough 3
Trial 1	1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1	1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1	0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1
Trial 2	1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1,1	0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0	0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1
Trial 3	1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1	0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1	0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1

Table B.1: Data from cough detection testing with no background noise.

B.2 Coughing, background noise

	Cough 1	Cough 2	Cough 3
Background 1	1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1	0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1	1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1
Background 2	1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1	0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1	0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1
Background 3	0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1	0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1	1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1

Table B.2: Data from cough detection testing with different background noises

B.3 Coughing at different distances

Distance in cm	Cough 1	Cough 2	Cough 3
20	1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1	0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1	1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1
15	1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1	1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1	1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1
10	0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1	1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0	1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1

Table B.3: Data from cough detection testing at different distances

B.4 Background noise only

Background 1	Background 2	Background 3
0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0	0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1	0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0

Table B.4: Data from cough detection with only background noise

B.5 High pitch

Trial 1	Trial 2	Trial 3
1,1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0	0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1	0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1

Table B.5: Data from cough detection test with high pitch noise

C

Appendix 3

This section provides the link to all the videos used in the testing.

C.1 Coughing videos

The videos with cough sounds:
Cough 1: https://www.youtube.com/watch?v=_EZKutZi3Fs
Cough 2: https://www.youtube.com/watch?v=yWfInypWKzI&t=379s
Cough 3: https://www.youtube.com/watch?v=15CMvY4aRM4

C.2 Background noise

The videos with background sounds:
Background 1: https://www.youtube.com/watch?v=IKB3Qiglyro
Background 2: https://www.youtube.com/watch?v=OE9bF80KQGko
Background 3: https://www.youtube.com/watch?v=IKB3Qiglyro

C.3 High-pitch noise

The video with the high pitch noise: https://www.youtube.com/watch?v=KNxFWFuaaRY&t=81s

DEPARTMENT OF ELECTRICAL ENGINEERING
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY