



**People's Democratic Republic of Algeria**  
**Ministry of Higher Education and Scientific Research**



**University of Ghardaïa**  
**Faculty of Science and Technology**  
**Department of Automation and Electromechanics**

Mémoire de master en Energies renouvelables en mécanique, conformément  
au décret ministériel modifié 008 complétant la décision ministérielle 12/75

## **Maste & Start-up**

Field: Science and Technology | Discipline: Automation  
Speciality: Automation and Systems

### **Thesis Title**

---

**Design and Development of a Smart Helmet for  
Industrial Safety Monitoring and Data Analysis**

---

#### **Presented by:**

CHERIET Azeddine  
DEMOUNI Slimane

**Defended on:** 21/06/2026

#### **Examination Committee:**

|                      |     |                |                   |
|----------------------|-----|----------------|-------------------|
| KIFOUCHE Abdeslam    | MCB | Univ. Ghardaïa | <b>President</b>  |
| NACER Hassen         | MCB | Univ. Ghardaïa | <b>Examiner</b>   |
| EULDJI Rafik         | MCB | Univ. Ghardaïa | <b>Examiner</b>   |
| KHATTARA Abdelouahab | MCA | Univ. Ghardaïa | <b>Supervisor</b> |

**Academic Year 2025/2026**

# Acknowledgements

First and foremost, we would like to thank Almighty God for granting us the strength, perseverance, and patience to complete this modest work.

We would like to express our sincere gratitude to our supervisor, **Dr Abdelouahab KHATTARA**, for his continuous guidance, valuable advice, and availability throughout this project.

We also wish to thank the members of the jury for kindly accepting to evaluate our work, as well as all the teachers of the department for the quality of the education and support they have provided us.

Finally, we dedicate this work to our parents and families in gratitude for their unwavering support and encouragement.

# Abstract

This work presents the design and development of a smart industrial safety helmet, a wearable system that enables real-time wireless voice communication between workers in noisy industrial environments. The helmet captures the worker's voice, transmits it over a local wireless network to other helmets, and uses an audio buffer to ensure smooth and intelligible playback. A central server routes the audio streams, records each work session, and automatically converts speech into text to generate structured shift reports. The helmet also provides an independent emergency channel that sends a distress signal with the worker's approximate location over a separate long-range radio link, fully decoupled from the main voice communication channel. Experimental tests show an overall mouth-to-ear delay below 150 ms, confirming the practical viability of the proposed system for safety communication and monitoring in modern industrial environments.

**Keywords:** Smart helmet, Industrial safety, Wireless communication, Speech recognition, Emergency localisation.

## Résumé

Ce travail présente la conception et le développement d'un casque intelligent de sécurité industrielle, un système portable permettant une communication vocale sans fil en temps réel entre les travailleurs dans des environnements industriels bruyants. Le casque capte la voix du travailleur, la transmet sur un réseau sans fil local vers les autres casques et utilise une mémoire tampon audio afin de garantir une restitution fluide et intelligible. Un serveur central achemine les flux audio, enregistre chaque session de travail et convertit automatiquement la parole en texte pour générer des rapports de poste structurés. Le casque offre également un canal d'urgence indépendant qui envoie un signal de détresse avec la position approximative du travailleur sur une liaison radio longue portée distincte, totalement séparée du canal principal de communication.

Les essais expérimentaux montrent un délai global bouche-oreille inférieur à 150 ms, ce qui confirme la viabilité pratique du système proposé pour la communication de sécurité et la surveillance dans les environnements industriels modernes.

**Mots-clés :** Casque intelligent, Sécurité industrielle, Communication sans fil, Reconnaissance vocale, Localisation d'urgence.

## ملخص

يقدم هذا العمل تصميم وتطوير خوزة سلامة صناعية ذكية، وهي نظام قابل للارتداء يوفر اتصالاً صوتياً لاسلكياً في الزمن الحقيقي بين العمال في البيئات الصناعية عالية الضجيج. تلتقط الخوزة صوت العامل وتنقله عبر شبكة لاسلكية محلية إلى بقية الخوز، مع الاعتماد على ذاكرة صوتية وسيطة لضمان استماع سلس وواضح. يتولى خادم مركزي توجيه القنوات الصوتية وتسجيل كل جلسة عمل وتحويل الكلام إلى نص من أجل توليد تقارير مهيكلية عن فترات العمل. كما توفر الخوزة قناة طوارئ مستقلة ترسل إشارة استغاثة مرفقة بالموقع التقريبي للعامل عبر وصلة راديوية طويلة المدى منفصلة بالكامل عن قناة الاتصال الصوتي الرئيسية. وأظهرت التجارب أن زمن الانتقال من الفم إلى الأذن أقل من 150 ميلي ثانية، مما يؤكد الجدوى العملية للنظام المقترح في الاتصالات الأمنية والمراقبة داخل البيئات الصناعية الحديثة.

**الكلمات المفتاحية:** خوزة ذكية، سلامة صناعية، اتصال لاسلكي صوتي، التعرف على الكلام، تحديد موقع الطوارئ.



# Acronyms and Abbreviations

|              |                                               |
|--------------|-----------------------------------------------|
| <b>SISH</b>  | Smart Industrial Safety Helmet                |
| <b>IoT</b>   | Internet of Things                            |
| <b>IIoT</b>  | Industrial Internet of Things                 |
| <b>ESP32</b> | Espressif Systems ESP32 Microcontroller       |
| <b>I2S</b>   | Inter-IC Sound (serial audio bus)             |
| <b>UDP</b>   | User Datagram Protocol                        |
| <b>TCP</b>   | Transmission Control Protocol                 |
| <b>PCM</b>   | Pulse-Code Modulation                         |
| <b>SNR</b>   | Signal-to-Noise Ratio                         |
| <b>PPE</b>   | Personal Protective Equipment                 |
| <b>ADC</b>   | Analog-to-Digital Converter                   |
| <b>MCU</b>   | Microcontroller Unit                          |
| <b>LAN</b>   | Local Area Network                            |
| <b>AP</b>    | Access Point                                  |
| <b>RTOS</b>  | Real-Time Operating System                    |
| <b>NVS</b>   | Non-Volatile Storage                          |
| <b>ASR</b>   | Automatic Speech Recognition                  |
| <b>STT</b>   | Speech-to-Text                                |
| <b>Wi-Fi</b> | Wireless Fidelity (IEEE 802.11)               |
| <b>DMA</b>   | Direct Memory Access                          |
| <b>LoRa</b>  | Long Range (low-power wide-area protocol)     |
| <b>GPS</b>   | Global Positioning System                     |
| <b>VOX</b>   | Voice-Operated switch (voice activity switch) |
| <b>FSM</b>   | Finite State Machine                          |
| <b>WAV</b>   | Waveform Audio File Format                    |
| <b>SOS</b>   | Distress/Emergency Signal                     |

# List of Figures

|                                                                                                                                                                                                                                                                                                                                                                                                 |    |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| 1.1. Sound level ranges of common industrial equipment. The shaded red zone highlights levels exceeding the 80 dBA threshold, where the risk of irreversible hearing damage begins (Data source: Soft dB). . . . .                                                                                                                                                                              | 3  |
| 1.2. The historical evolution of manufacturing paradigms, from Industry 1.0 to Industry 4.0 . . . . .                                                                                                                                                                                                                                                                                           | 5  |
| 1.3. Advanced technologies driving the Industry 4.0 vision . . . . .                                                                                                                                                                                                                                                                                                                            | 6  |
| 1.4. Primary sensing functions reported in smart helmet prototypes . . . . .                                                                                                                                                                                                                                                                                                                    | 7  |
| 1.5. Pareto analysis of smart helmet prototypes and their primary sensing functions . . . . .                                                                                                                                                                                                                                                                                                   | 8  |
| 2.1. Overall architecture of the SISH system. . . . .                                                                                                                                                                                                                                                                                                                                           | 19 |
| 2.2. Representative ESP32-S3 development-board pinout illustrating the availability of GPIO, UART, I <sup>2</sup> C, SPI and audio-capable digital interfaces. . . . .                                                                                                                                                                                                                          | 23 |
| 2.3. INMP441 MEMS microphone and MAX98357A I2S amplifier breakouts . . . . .                                                                                                                                                                                                                                                                                                                    | 25 |
| 2.4. Main hardware modules of the long-range localisation subsystem: Heltec WiFi LoRa 32 V3 board (left) and u-blox NEO-6M GPS receiver module (right). . . . .                                                                                                                                                                                                                                 | 26 |
| 2.5. High-level software and communication architecture of the SISH system, showing the Wi-Fi/UDP voice plane and the LoRa emergency telemetry plane converging at the Python monitoring server. . . . .                                                                                                                                                                                        | 30 |
| 2.6. SISH monitoring-server architecture: Wi-Fi/UDP voice bridge ( <i>left</i> ) and direct point-to-point LoRa (EU433) emergency path ( <i>right</i> ) converging through the LoRa-ID ↔ helmet-ID mapping into a unified application layer. The Chirp-Stack/MQTT path shown conceptually is identified as a future scalability extension and is not part of the implemented prototype. . . . . | 34 |
| 3.1. SISH smart industrial helmet prototype — dual-node assembly mounted on an EN 397 industrial helmet shell. . . . .                                                                                                                                                                                                                                                                          | 38 |
| 3.2. Complete wiring diagram of the SISH prototype, showing the ESP32-S3 node, the Heltec V3 node and the shared power supply. . . . .                                                                                                                                                                                                                                                          | 40 |
| 3.3. Power distribution and battery-management chain for the SISH helmet (USB—C input, TP4056 charger/protection, 3.7 V Li-Po cell, master switch, MT3608 boost to a shared 5 V rail, and local 3.3 V regulation on each node). . . . .                                                                                                                                                         | 42 |
| 3.4. Exploded view of the complete SISH dual-node assembly (CAD), showing the ESP32-S3 audio node, Heltec V3 LoRa/GPS node, battery pack, and mechanical mounting components. . . . .                                                                                                                                                                                                           | 44 |

|      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |    |
|------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| 3.5. | SISH prototype installed on an EN 397 industrial helmet (overall placement of user-facing elements). . . . .                                                                                                                                                                                                                                                                                                                                                     | 44 |
| 3.6. | <b>Server and helmet view — voice routing.</b> <i>Left:</i> Python monitoring server (mainv4.py --full) showing STT fast-trigger detection, Helmet2 ↔ Helmet1 private-call routing, 10 s traffic statistics, hangup back to Idle(muted), and automatic WAV recording. <i>Right:</i> Helmet H1 serial monitor showing Connected! IP: 10.46.46.7; IDLE/MUTED → PRIVATE → IDLE transitions driven by PKT_COMMAND packets, and VOX RMS values during speech. . . . . | 49 |
| 3.7. | <b>State-machine diagram — communication modes.</b> <i>Left:</i> Python monitoring server console confirming bridge routing, fast-trigger STT command detection, and automatic WAV recording. <i>Right:</i> Helmet H1 serial monitor showing the IDLE → PRIVATE → IDLE state-machine transitions with VOX RMS values, captured in the same session. . . . .                                                                                                      | 49 |
| 3.8. | Experimental views of the LoRa emergency subsystem: the upper image shows the OLED-level operating states of the safety node, while the lower images present the Arduino IDE implementation and serial-monitor evidence for the transmitter and receiver nodes. . . . .                                                                                                                                                                                          | 52 |
| 3.9. | Wi-Fi endpoint and VoicePacket capture for the SISH helmets, showing the UDP datagrams emitted by the ESP32-S3 node towards the monitoring server. .                                                                                                                                                                                                                                                                                                             | 53 |

# List of Tables

|                                                                                                                                         |    |
|-----------------------------------------------------------------------------------------------------------------------------------------|----|
| 1.1. Indicative global occupational safety statistics (approximate values). . . . .                                                     | 3  |
| 1.2. Representative smart helmet / smart PPE systems in the literature. . . . .                                                         | 10 |
| 1.3. Qualitative comparison of common wireless communication technologies for<br>IoT/IIoT applications . . . . .                        | 12 |
| 2.1. End-to-end audio chain from helmet microphone to loudspeaker. . . . .                                                              | 25 |
| 2.2. Byte-level structure of the VoicePacket header (9 B) and audio payload (320 B). . . . .                                            | 31 |
| 3.1. Complete dual-node GPIO / component map (ESP32-S3 audio node + Heltec<br>V3 GPS/LoRa node) . . . . .                               | 39 |
| 3.2. Main components integrated in the SISH prototype (bill of materials summary). . . . .                                              | 43 |
| 3.3. Firmware toolchain per node. . . . .                                                                                               | 45 |
| 3.4. VoicePacket wire format (<BBBHI little-endian header + PCM); total 329 bytes;<br>ts_ms uint32 $\Rightarrow$ 49.7-day wrap. . . . . | 47 |
| 3.5. Server-side events during a single private-call session. . . . .                                                                   | 59 |
| 3.6. Informal MOS-equivalent listening evaluation (5 sessions, 2 listeners, labora-<br>tory environment). . . . .                       | 64 |
| 3.7. Consolidated performance metrics (measured vs design-derived). . . . .                                                             | 65 |
| 3.8. Requirements compliance matrix (vs Section 2.2). . . . .                                                                           | 67 |

# Contents

|                                                                        |            |
|------------------------------------------------------------------------|------------|
| <b>Acknowledgements</b>                                                | <b>i</b>   |
| <b>Abstract</b>                                                        | <b>ii</b>  |
| <b>Acronyms and Abbreviations</b>                                      | <b>iv</b>  |
| <b>List of Figures</b>                                                 | <b>vi</b>  |
| <b>List of Tables</b>                                                  | <b>vii</b> |
| <b>General Introduction</b>                                            | <b>xi</b>  |
| <b>1 Theoretical Background and General Concepts</b>                   | <b>1</b>   |
| 1.1 Introduction                                                       | 2          |
| 1.2 Industrial Safety in Noisy Environments                            | 2          |
| 1.3 Industry 4.0 and the digital transformation of industrial safety   | 4          |
| 1.3.1 Core technologies enabling Industry 4.0                          | 5          |
| 1.3.2 Digitalizing occupational safety in the smart factory            | 5          |
| 1.3.3 Connected PPE as smart nodes in Industry 4.0                     | 6          |
| 1.4 Communication methods used in industry and their limitations       | 7          |
| 1.5 Smart helmets and connected personal protective equipment          | 8          |
| 1.6 Wireless communication technologies for embedded systems           | 10         |
| 1.6.1 Wi-Fi for real-time audio                                        | 10         |
| 1.6.2 LoRa for telemetry and long-range coverage                       | 11         |
| 1.6.3 Comparative analysis and hybrid architecture                     | 11         |
| 1.7 Voice recognition and audio processing for industrial applications | 11         |
| 1.7.1 Digital audio capture and real-time transmission                 | 12         |
| 1.7.2 Speech recognition in noisy environments and edge processing     | 12         |
| 1.7.3 Engineering integration of ASR engines                           | 13         |
| 1.8 Chapter Summary and Conclusion                                     | 13         |
| <b>Functional Analysis and System Design</b>                           | <b>15</b>  |
| 2.1 Introduction                                                       | 16         |
| 2.2 System Requirements and Functional Specifications Analysis         | 16         |

|         |                                                                                 |           |
|---------|---------------------------------------------------------------------------------|-----------|
| 2.3     | Overall Architecture of the SISH System . . . . .                               | 17        |
| 2.3.1   | Layered IoT Architecture . . . . .                                              | 18        |
| 2.3.2   | Data Flow . . . . .                                                             | 19        |
| 2.3.3   | Design Rationale . . . . .                                                      | 20        |
| 2.4     | Hardware Design of the Smart Helmet . . . . .                                   | 20        |
| 2.4.1   | System Hardware Overview . . . . .                                              | 21        |
| 2.4.2   | The ESP32-S3 Main Microcontroller . . . . .                                     | 21        |
| 2.4.3   | Audio Subsystem: Microphone, Amplifier, and Speaker . . . . .                   | 22        |
| 2.4.4   | LoRa Module and GPS Localization Integration . . . . .                          | 24        |
| 2.4.5   | Human–Machine Interface (HMI) . . . . .                                         | 27        |
| 2.4.5.1 | Indicators and controls . . . . .                                               | 27        |
| 2.4.6   | Power Supply and Autonomy . . . . .                                             | 28        |
| 2.4.6.1 | Charging, protection, and user independence . . . . .                           | 28        |
| 2.5     | Communication and Software Architecture Design . . . . .                        | 29        |
| 2.5.1   | Wi-Fi Network Architecture . . . . .                                            | 29        |
| 2.5.2   | Voice Packet Protocol Design (VoicePacket Structure) . . . . .                  | 30        |
| 2.5.3   | Communication State Machine Design . . . . .                                    | 31        |
| 2.5.4   | Gateway and Server Architecture . . . . .                                       | 32        |
| 2.5.4.1 | Voice plane. . . . .                                                            | 33        |
| 2.5.4.2 | Telemetry plane. . . . .                                                        | 33        |
| 2.5.4.3 | Identifier reconciliation and unified worker record. . . . .                    | 33        |
| 2.5.5   | Speech Recognition Integration Design (Vosk + Whisper) . . . . .                | 33        |
| 2.5.5.1 | Voice Recognition and Transcription Pipeline . . . . .                          | 33        |
| 2.5.5.2 | Low-latency command recognition (Vosk). . . . .                                 | 34        |
| 2.5.5.3 | Deferred transcription (Whisper). . . . .                                       | 35        |
| 2.5.5.4 | Front-end segmentation (Silero VAD). . . . .                                    | 35        |
| 2.5.5.5 | Asynchronous integration and isolation. . . . .                                 | 35        |
| 2.5.5.6 | Closing the command loop. . . . .                                               | 35        |
| 2.5.5.7 | Safety constraint: emergency semantics. . . . .                                 | 35        |
|         | <b>Implementation, Hardware Assembly, and Experimental Evaluation . . . . .</b> | <b>36</b> |
| 3.1     | Introduction . . . . .                                                          | 37        |
| 3.2     | Hardware Assembly and Component Integration . . . . .                           | 37        |
| 3.2.1   | Complete Pinout and Component Mapping . . . . .                                 | 38        |
| 3.2.2   | Wiring Diagram . . . . .                                                        | 39        |
| 3.2.3   | Power Supply and BMS Circuit . . . . .                                          | 40        |
| 3.2.4   | Mechanical Integration inside the Helmet . . . . .                              | 42        |
| 3.3     | Firmware Implementation (Dual-Node) . . . . .                                   | 44        |
| 3.3.1   | Development Environment and Tools . . . . .                                     | 45        |

|                           |                                                |           |
|---------------------------|------------------------------------------------|-----------|
| 3.3.2                     | I2S Audio Capture (ESP32-S3)                   | 45        |
| 3.3.3                     | UDP Voice Streaming (ESP32-S3)                 | 46        |
| 3.3.4                     | Jitter Buffer Implementation (ESP32-S3)        | 47        |
| 3.3.5                     | Communication State Machine (ESP32-S3)         | 48        |
| 3.3.6                     | SOS and GPS Transmission (Heltec LoRa32 V3)    | 50        |
| 3.4                       | Gateway and Server Implementation (Python)     | 53        |
| 3.4.1                     | UDP Server and Multi-Helmet Routing            | 54        |
| 3.4.2                     | Real-Time Speech Recognition (Vosk STT)        | 57        |
| 3.4.3                     | Post-Session Transcription (Whisper)           | 59        |
| 3.4.4                     | Session Logging and Report Generation          | 60        |
| 3.5                       | System Integration and Experimental Evaluation | 61        |
| 3.5.1                     | Hardware–Software Integration Validation       | 62        |
| 3.5.2                     | Testing Scenarios and Methodology              | 62        |
| 3.5.3                     | Experimental Results and Performance Metrics   | 63        |
| 3.5.4                     | Discussion of Results                          | 65        |
| 3.5.5                     | Requirements Compliance Summary vs Section 2.2 | 66        |
| 3.6                       | Conclusion                                     | 68        |
| <b>General Conclusion</b> |                                                | <b>69</b> |

# **General Introduction**



# General Introduction

The rapid advancement of Industry 4.0 technologies has transformed industrial workplaces into increasingly connected, data-driven environments. Despite improvements in safety equipment, workers in high-noise environments — such as construction sites, manufacturing plants, and oil refineries — still face critical communication barriers that directly impact operational safety and efficiency.

Traditional Personal Protective Equipment (PPE), including hard hats and helmets, provides physical protection but lacks any intelligent or communicative capability. Meanwhile, existing wireless communication solutions such as walkie-talkies require the user to remove their helmet or use external earpieces, disrupting workflow and reducing compliance.

The **Smart Industrial Safety Helmet (SISH)** addresses these limitations by integrating real-time wireless voice communication directly into the helmet. Built on the **ESP32-S3** microcontroller and a local Wi-Fi network using UDP datagrams, SISH enables helmet-to-helmet audio transmission with a jitter buffer for smooth playback and a Python gateway server for multi-node routing and speech transcription.

## **Thesis structure:**

- **Chapter 1** reviews the state of the art: industrial safety, IoT/IIoT, wireless communication technologies, and digital audio transmission basics.
- **Chapter 2** presents the functional analysis and hardware/software design of the SISH system.
- **Chapter 3** covers firmware implementation, testing protocol, experimental evaluation, and discussion of results.

# **Chapter 1**

## **Theoretical Background and General Concepts**

## 1.1 Introduction

The contemporary industrial landscape is being reshaped by the accelerating momentum of the Fourth Industrial Revolution, widely known as Industry 4.0. This paradigm is founded on the convergence of cyber-physical systems, the Industrial Internet of Things (IIoT), advanced analytics, and intelligent automation. In this context, industrial systems are no longer conceived as isolated mechanical entities, but as interconnected, data-centric ecosystems capable of sensing, communicating, processing, and responding in real time.

While early deployments of Industry 4.0 technologies largely focused on productivity enhancement, predictive maintenance, and process optimization, it has become increasingly clear that technological progress must also serve a more fundamental imperative: the protection of human operators. As production environments grow more automated and technologically dense, worker safety acquires renewed importance, particularly in noisy, mobile, and hazard-prone settings where communication reliability and situational awareness are critical.

Despite decades of improvement in occupational protection, conventional personal protective equipment (PPE) remains fundamentally limited by its passive nature. Standard safety helmets and hearing protectors primarily provide mechanical shielding and acoustic attenuation, yet they offer no embedded capacity for contextual sensing, adaptive communication, or dynamic interaction with the surrounding environment. In harsh and noisy workplaces, this limitation becomes critical, since the very equipment intended to protect the worker may simultaneously reduce the perception of warning signals and hinder team coordination.

These constraints have motivated a transition from passive protection to intelligent, responsive protective systems, giving rise to smart PPE that integrates wearable protection with embedded electronics, wireless communication, edge-level processing, and decision support [19]. In this context, the Smart Industrial Safety Helmet (SISH) project is proposed as an advanced wearable safety solution that reconceptualizes the industrial helmet from a static shell into an active smart edge node oriented towards real-time voice communication and audio intelligence.

To situate the SISH project within this broader framework, this chapter develops the theoretical and technological foundations on which the proposed system is based. It addresses industrial safety in noisy environments, the principles of Industry 4.0, communication methods and their limitations, smart helmets and connected PPE, wireless communication technologies for embedded systems, and voice recognition and audio processing for industrial applications, before concluding with the positioning of the project.

## 1.2 Industrial Safety in Noisy Environments

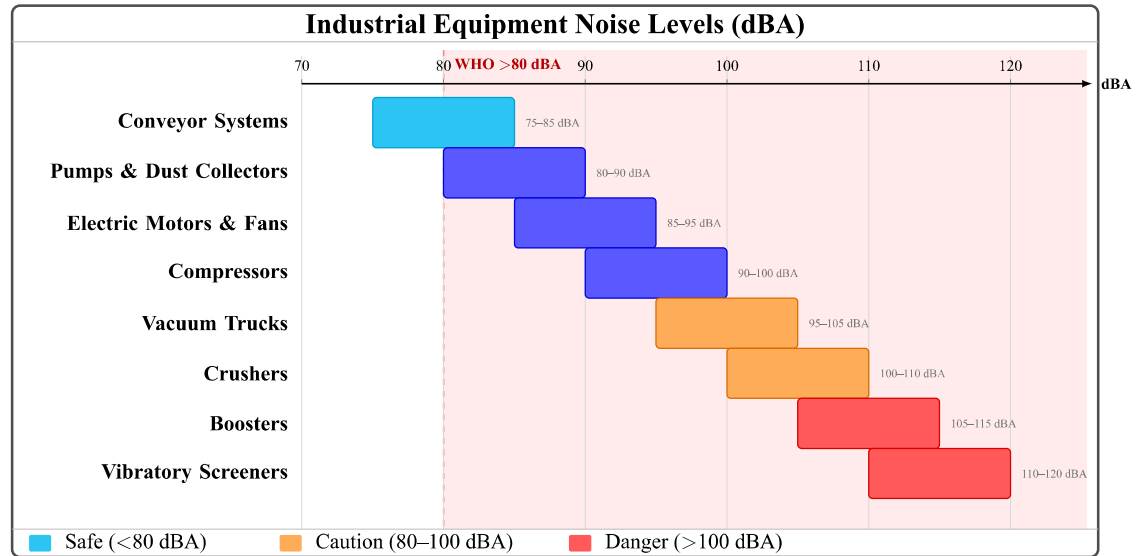
Industrial safety and occupational health aim to prevent accidents, injuries, and work-related illnesses by identifying hazards, evaluating risks, and implementing technical, organizational, and behavioral measures. Proactive safety engineering is heavily justified by recent global statistics:

international organizations such as the ILO and the UN report that each year around 2.8–3.0 million workers die due to occupational accidents and work-related diseases, while hundreds of millions suffer non-fatal injuries. The overall economic cost is estimated at approximately 4% of global GDP [13, 12]. Table 1.1 summarizes indicative figures that highlight the macroscopic scale of industrial hazards.

| Indicator                       | Approximate Annual Value           | Source                   |
|---------------------------------|------------------------------------|--------------------------|
| Total work-related deaths       | ≈ 2.8–3.0 million deaths/year      | ILO, UN reports [13, 12] |
| Non-fatal occupational injuries | ≈ 374 million cases/year           | ILO estimates [13]       |
| Economic cost of poor OSH       | ≈ 4% of global GDP                 | ILO / UN analyses [12]   |
| Share of deaths in construction | ≈ 30% of total occupational deaths | ILO sectoral data [13]   |

**Table 1.1.** Indicative global occupational safety statistics (approximate values).

Within many of these high-risk industrial sectors—such as construction, mining, manufacturing, and oil and gas—noise is a pervasive hazard that interacts with mechanical, electrical, and chemical risks to create complex safety challenges. As illustrated in Figure 1.1, almost all heavy industrial equipment operates well above the 80 dBA safety threshold established by the World Health Organization (WHO). This extreme acoustic environment not only threatens hearing health but completely masks human speech, underscoring the absolute necessity for the SISH smart helmet to employ advanced noise-canceling digital microphones (**I2S**) rather than standard analog transceivers.



**Figure 1.1.** Sound level ranges of common industrial equipment. The shaded red zone highlights levels exceeding the 80 dBA threshold, where the risk of irreversible hearing damage begins (Data source: Soft dB).

High noise levels reduce the intelligibility of speech, mask auditory warning signals, and increase cognitive load on workers who must maintain attention while operating machinery or coordinating with colleagues. Furthermore, prolonged exposure can cause temporary or permanent hearing loss, tinnitus, and fatigue, which degrade situational awareness and decision-making. In such environments, traditional safety practices that rely on shouted instructions, hand signals, or simple alarms become insufficient or completely unreliable.

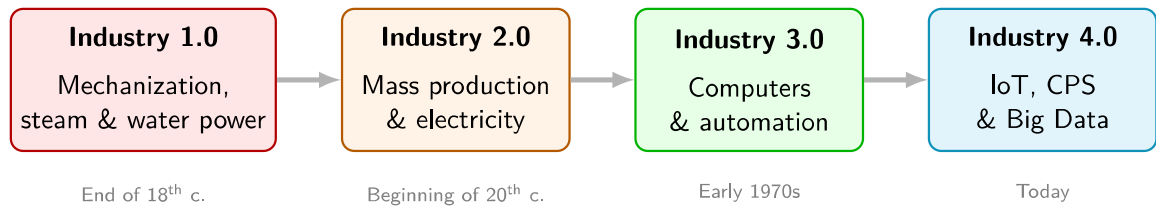
Hearing protection devices (HPDs) such as earplugs and earmuffs are widely used to protect workers from noise-induced hearing loss. However, their very function—attenuating sound—simultaneously impedes communication and the perception of ambient warning signals. Field studies, such as the research conducted by Atmaca et al. [3] on heavy industry workers, confirm that passive protective equipment suffers from a significant practical failure: despite noise levels consistently exceeding safe limits, a large portion of workers deliberately avoided using classical ear protectors to maintain communication with their peers. Consequently, over 30% of the surveyed workers suffered from hearing loss, and nearly 60% experienced severe nervous stress due to continuous noise exposure [3].

This trade-off illustrates a fundamental engineering dilemma: protecting hearing physically often comes at the cost of reducing acoustic awareness and isolating the worker. Addressing this vulnerability requires abandoning strictly passive isolation strategies in favour of connected, intelligent approaches that combine physical protection with digital audio processing and seamless communication capabilities integrated directly into the worker's PPE.

### **1.3 Industry 4.0 and the digital transformation of industrial safety**

The global industrial sector is undergoing a profound transformation driven by Industry 4.0, which marks a departure from traditional automation by tightly coupling physical assets with digital platforms within unified, data-driven environments. Smart factories rely on intensive data collection, ubiquitous connectivity, and artificial intelligence to enable highly efficient, self-optimizing, and flexible manufacturing processes.

Industry 4.0 is often described as a vision of factories built on intelligent cyber-physical systems and advanced communication technologies that connect a vast number of heterogeneous devices over IP-based networks. Legacy machines and new equipment alike are augmented with sensing, communication, and computation capabilities, enabling them to participate in higher-level digital services such as predictive maintenance, quality monitoring, and resource optimization.



**Figure 1.2.** The historical evolution of manufacturing paradigms, from Industry 1.0 to Industry 4.0 [15, 24].

### 1.3.1 Core technologies enabling Industry 4.0

**Industrial Internet of Things (IIoT).** IIoT extends generic IoT concepts into industrial contexts by connecting sensors, actuators, robots, machine tools, and control systems via wired and wireless networks. Devices are addressable through IP-based communication (Ethernet, Wi-Fi, cellular), allowing large volumes of operational data to be exchanged between field devices and centralized or cloud platforms. IIoT thus bridges traditional operational technology (OT) with information technology (IT) and supports continuous monitoring and control.

**Artificial intelligence and machine learning.** AI and ML provide the intelligence layer of Industry 4.0 ecosystems. By applying learning algorithms to data streams generated by IIoT infrastructures, they enable predictive maintenance, anomaly detection, real-time quality assurance, and workflow optimization. These techniques contribute to autonomic capabilities such as self-monitoring, self-configuration, and self-healing in smart manufacturing systems.

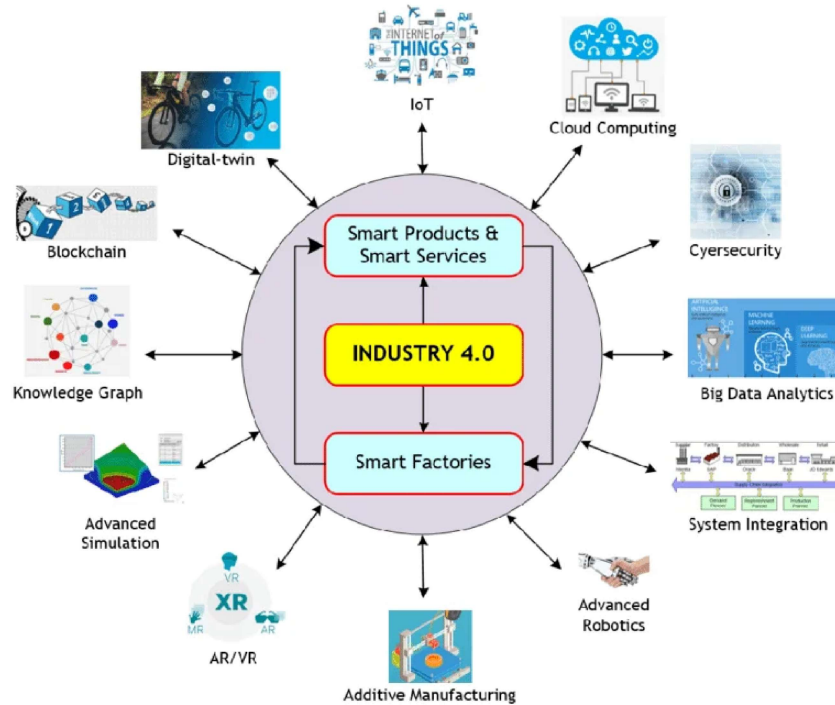
**Cloud and edge computing.** Cloud computing offers scalable resources for storage, processing, and analytics, supporting globally distributed design, simulation, and production activities. Edge computing complements the cloud by bringing computation closer to machines and sensors, reducing latency, improving responsiveness for time-critical control and safety functions, and allowing local decision-making even under constrained connectivity.

**Cyber-physical systems and digital twins.** Cyber-physical systems (CPS) tightly integrate physical processes with computation and communication, forming the foundation of smart factories. Digital twins create virtual replicas of physical assets and processes that are updated with real-time data, enabling simulation, optimization, and safety analysis without interrupting production.

### 1.3.2 Digitalizing occupational safety in the smart factory

As factories become more complex and densely instrumented, it is increasingly recognized that the smart factory vision must encompass not only productivity and efficiency, but also the protection of human workers. In this context, smart factory worker safety has emerged as a core design principle that aligns occupational safety with Industry 4.0 objectives.

Traditional safety approaches are often reactive and compliance-oriented, with measures implemented after incidents occur. In contrast, the smart factory paradigm aims to embed safety



**Figure 1.3.** Advanced technologies driving the Industry 4.0 vision, encompassing IoT, Cloud Computing, Big Data Analytics, Cyber-Physical Systems, and Smart Factories [15, 32].

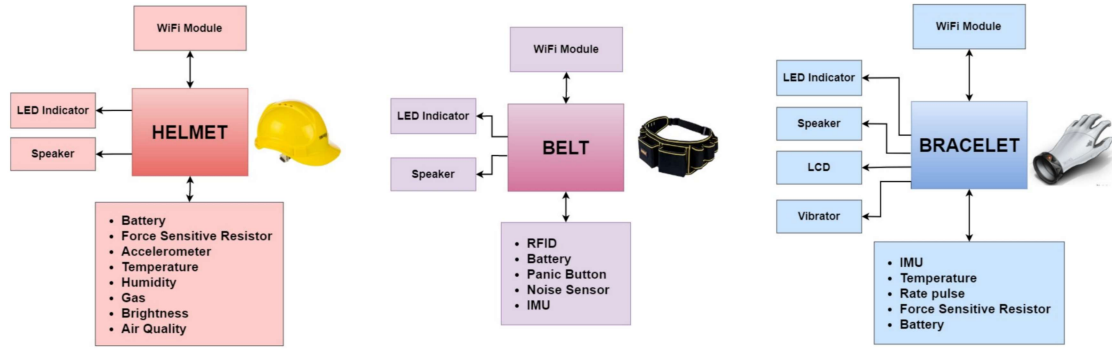
into data flows and control logic, so that safety-relevant information — such as environmental conditions, machine states, and worker locations — is continuously collected, analyzed, and acted upon. This enables proactive risk management: hazards can be identified earlier, unsafe behaviors can be detected in real time, and automated interventions can be triggered before accidents escalate.

### 1.3.3 Connected PPE as smart nodes in Industry 4.0

Within this paradigm, personal protective equipment is undergoing a transformation from passive, purely physical barriers into connected, intelligent edge devices. Connected PPE is characterized by several advanced capabilities:

- Real-time sensing of environmental and physiological parameters.
- Low-power, low-latency wireless connectivity to IIoT infrastructures.
- Edge processing for local anomaly detection and decision-making.
- Bidirectional interaction with workers through audio, visual, or haptic feedback.

These capabilities turn PPE into active nodes within cyber-physical safety systems, closing the loop between human behavior, environmental conditions, and automated safety controls.



**Figure 1.4.** Primary sensing functions reported in smart helmet prototypes, adapted from [16].

## 1.4 Communication methods used in industry and their limitations

Industrial communication relies on a combination of wired and wireless technologies, each with strengths and limitations. On the wired side, fieldbuses and industrial Ethernet provide reliable, deterministic communication for machine control and automation. However, wired links are unsuitable for mobile workers and wearable devices.

For human-to-human and human-to-system communication, several options are commonly used:

- Conventional two-way radios based on analog or digital narrowband technologies.
- Public cellular networks (GSM, 3G, 4G, 5G) for voice and data.
- In-plant intercom systems and public address systems.

Two-way radios are robust and widely adopted, but they typically require manual operation, are not fully integrated with PPE, and may rely on channels shared by many users, leading to congestion. Cellular networks provide broad coverage but depend on external infrastructure and may exhibit variable latency and unreliable coverage in indoor or underground environments. Intercom and public address systems are fixed installations that do not follow the worker.

Moreover, most of these communication channels operate independently from digital safety systems and IIoT platforms, making it difficult to correlate voice exchanges with sensor data, alarms, and worker location information. This fragmentation motivates the design of integrated communication architectures where wearable devices such as smart helmets provide both sensing and communication capabilities.

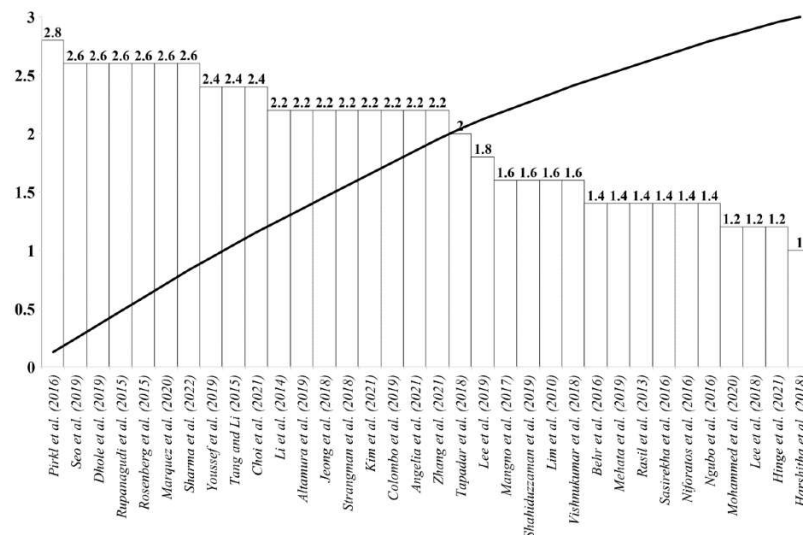


## 1.5 Smart helmets and connected personal protective equipment

The transition from conventional safety strategies to smart wearable systems represents a qualitative leap in occupational safety engineering. The industrial safety helmet, one of the most iconic elements of PPE, has undergone a structural transformation that redefines its role in hazardous workplaces. It is evolving from a passive mechanical barrier into a compact edge computing platform and interactive node within IIoT safety architectures.

Recent work shows that innovations in smart helmets can be grouped around four main functional axes:

- *Activity risk sensing (ARS)* using inertial measurement units (IMUs) and accelerometers to detect falls, impacts, and abnormal movements.
- *Physiological risk sensing (PRS)* using wearable sensors to monitor body temperature, heart rate, and other vital signs.
- *Environmental risk sensing (ERS)* using gas sensors, temperature and humidity sensors, and noise level meters to assess ambient conditions.
- *Risk event alerting (REA)* using audio, visual, or haptic feedback and wireless communication to disseminate alerts to workers and supervisors.



**Figure 1.5.** Pareto analysis of smart helmet prototypes and their primary sensing functions, adapted from [16].

Building on this classification, several research projects have proposed complete Smart PPE platforms that combine helmets with other wearable nodes such as belts and bracelets, each

integrating multiple sensors and sharing data with a common edge gateway.[19] An illustrative example is the AIoT-based Smart PPE system of Márquez-Sánchez et al., where helmet, belt, and bracelet modules incorporate environmental, motion, and physiological sensing while communicating over Wi-Fi with an NVIDIA Jetson Nano for anomaly detection and visualization.[19]

Many prototypes and commercial products focus on environmental telemetry and worker tracking, integrating GPS, gas sensors, and connectivity to cloud platforms. Others target fall detection or PPE compliance monitoring using computer vision and thermal imaging. These solutions demonstrate the technological maturity of smart helmet concepts as connected nodes within cyber-physical safety systems, but also reveal a common orientation toward sensing and supervisory monitoring.

However, in a significant number of existing solutions, direct, hands-free audio communication between workers is either absent, delegated to separate radio devices, or requires external infrastructure. This creates an “audio-communication gap” whereby workers are digitally monitored but remain communicatively isolated when dedicated radios are unavailable, impractical, or too costly.

The proposed Smart Industrial Safety Helmet (SISH) explicitly addresses this gap by augmenting smart helmet functions with a low-cost, decentralized voice channel between helmets. Rather than treating audio as an auxiliary service, SISH integrates digital audio interfaces and local wireless networking to enable reliable, hands-free, helmet-to-helmet voice communication without permanent internet connectivity.

Table 1.2 summarises a selection of recent smart-helmet and smart-PPE proposals, highlighting their main sensing functions and communication features. This comparison clarifies the positioning of SISH with respect to existing work.

| System                      | Primary functions                                                     | Communication                              | Live voice between workers?                |
|-----------------------------|-----------------------------------------------------------------------|--------------------------------------------|--------------------------------------------|
| Okokpujie et al. [21]       | Head-impact detection, fall detection                                 | Short-range wireless, SMS alert            | No (alarm-centric)                         |
| Márquez-Sánchez et al. [19] | Environmental, motion, physiological sensing (helmet, belt, bracelet) | Wi-Fi to edge gateway, cloud dashboard     | No (telemetry, monitoring)                 |
| Budiono et al. [4]          | Environmental monitoring, worker tracking                             | IoT connectivity to central platform       | No (data logging, alerts)                  |
| Shreshta et al. [26]        | Accident detection, location reporting                                | Cellular / GSM notification                | No (emergency notification only)           |
| <b>SISH (this work)</b>     | Real-time voice, emergency telemetry, GPS, future sensing extensions  | Wi-Fi UDP voice plane + LoRa SOS/GPS plane | <b>Yes, bidirectional helmet-to-helmet</b> |

**Table 1.2.** Representative smart helmet / smart PPE systems in the literature.

## 1.6 Wireless communication technologies for embedded systems

The design of wearable embedded systems for industrial environments must balance constraints on hardware resources with the characteristics of the data to be transmitted. No single wireless protocol can efficiently satisfy all operational requirements, so network architectures are shaped by trade-offs between bandwidth, range, power consumption, and robustness to obstacles.

### 1.6.1 Wi-Fi for real-time audio

Applications involving live voice communication and the real-time exchange of spoken commands require high throughput and low end-to-end latency. IEEE 802.11 (Wi-Fi) standards pro-

vide an effective infrastructure for handling high-rate digital audio traffic within local industrial networks, particularly when microcontrollers such as the ESP32 are used as endpoints.

For embedded audio streaming, the User Datagram Protocol (UDP) is typically preferred over TCP to avoid handshake overhead and retransmission delays, enabling continuous packet flows with minimal buffering. Under controlled conditions, Wi-Fi can achieve sufficiently low latency for interactive industrial communication, at the cost of higher energy consumption and greater sensitivity to congestion.

### **1.6.2 LoRa for telemetry and long-range coverage**

In contrast, functions such as environmental telemetry, emergency alerts, and worker tracking require reliable delivery of small packets over longer distances with minimal energy consumption. LoRa technology, based on chirp spread spectrum modulation in sub-GHz bands, is well suited to such tasks.

Experimental comparisons show that LoRa maintains more stable signal strength and link continuity than 2.4 GHz technologies as distance and obstacles increase, though at lower data rates and higher latency. LoRa is therefore appropriate for periodic sensor data, alarm flags, and status updates from battery-powered nodes deployed over large industrial sites.

### **1.6.3 Comparative analysis and hybrid architecture**

A comparative analysis of Wi-Fi, LoRa, Bluetooth Low Energy, and ZigBee reveals that each technology occupies a specific region of the design space. Wi-Fi offers high data rates and low latency over short to medium distances; LoRa provides long-range, low-rate communication; BLE and ZigBee are optimized for low power and, in the case of ZigBee, mesh networking.

These observations support the adoption of a hybrid architecture in which Wi-Fi is used as a dedicated channel for digital audio streams within the local plant network, while LoRa or similar technologies are reserved for low-bit-rate telemetry, alarms, and long-range tracking. By assigning each protocol to tasks that match its strengths, the system can achieve real-time interaction, extended coverage, and energy-efficient operation.

## **1.7 Voice recognition and audio processing for industrial applications**

Establishing wireless channels is only a preparatory step; operational value emerges when these channels are transformed into intelligent audio interfaces. In complex industrial environments, voice interaction is a safety-critical modality that allows workers to keep their hands free and eyes focused on hazardous tasks. However, deploying automatic speech recognition (ASR)

| Technology                      | Typical Range     | Data Rate          | Power       | Suitable for Live Audio?              |
|---------------------------------|-------------------|--------------------|-------------|---------------------------------------|
| <b>Wi-Fi (802.11 b/g/n)</b>     | 50–100 m (indoor) | up to 150 Mbps     | Medium–High | <b>Yes</b> , very suitable            |
| <b>Bluetooth Classic / BLE</b>  | 10–30 m           | 1–3 Mbps (Classic) | Low         | Limited (Classic only)                |
| <b>ZigBee (802.15.4)</b>        | 10–100 m          | 250 kbps           | Low         | <b>No</b> , poor for continuous audio |
| <b>LoRa / LoRaWAN</b>           | km-level          | 0.3–50 kbps        | Very Low    | <b>No</b> , insufficient data rate    |
| <b>Cellular (3G/4G/5G)</b>      | Wide area         | Variable (Mbps)    | High        | <b>Yes</b> , requires subscription    |
| <b>Sub-GHz RF (433/868 MHz)</b> | up to 100 m       | Tens of kbps       | Very Low    | <b>No</b> , telemetry only            |

**Table 1.3.** Qualitative comparison of common wireless communication technologies for IoT/IIoT applications. Source: adapted from [11, 18, 2].

in such environments is challenging due to noise, reverberation, and limited computational resources.

### 1.7.1 Digital audio capture and real-time transmission

To bring inference capabilities closer to the worker, audio capture must be engineered to withstand industrial interference. Legacy analog microphones are highly susceptible to electromagnetic noise generated by machinery, whereas digital MEMS microphones connected via the Inter-IC Sound (I2S) interface provide a cleaner and more robust data path to the microcontroller.

In real-time embedded voice systems, the incoming signal is sampled and segmented into small frames that are transmitted over UDP to minimize transport latency. Network jitter, the fluctuation in packet arrival times, can severely disrupt audio quality; jitter buffers are therefore used to collect, reorder, and time-align incoming packets, thereby reconstructing a stable audio stream for playback or further processing.

### 1.7.2 Speech recognition in noisy environments and edge processing

Industrial background noise reduces the signal-to-noise ratio and significantly degrades speech recognition accuracy. Relying solely on cloud-based ASR introduces additional risks due to variable latency and potential connectivity loss, especially in underground or shielded facilities. These constraints motivate an on-premises server-side AI architecture in which language models are deployed within the plant network and operate fully offline, ensuring deterministic response times for safety-critical commands without dependence on Internet connectivity.

### 1.7.3 Engineering integration of ASR engines

Embedding ASR functionality into resource-constrained embedded platforms requires a careful balance between latency, accuracy, and computational load. Lightweight offline engines are particularly well suited to low-latency recognition of short, safety-critical commands, whereas larger transformer-based models provide more robust transcription of long and noisy recordings at the expense of significantly higher computational requirements.

Within the SISH architecture, all automatic speech recognition is deliberately centralised on an on-premises monitoring server rather than embedded in the helmet firmware. The ESP32-S3 nodes remain limited to deterministic edge tasks (audio capture, packetisation, transmission, reception, and playback), while ASR engines such as Vosk and Whisper operate exclusively on PCM streams and recordings after reception at the server side. This split preserves the timing determinism of the embedded audio pipeline while enabling higher-accuracy models and simplified maintainability at the monitoring infrastructure.

## 1.8 Chapter Summary and Conclusion

This chapter reviewed the theoretical and technological foundations underpinning the Smart Industrial Safety Helmet (SISH) project. Starting from the occupational safety challenges posed by noisy and hazardous industrial environments, it traced the shift from passive personal protective equipment to intelligent, connected devices within the broader Industry 4.0 paradigm of cyber-physical systems, the Industrial Internet of Things (IIoT), and edge computing. The limitations of conventional worker communication were examined alongside the current state of smart helmets and connected PPE, the relevant wireless technologies for embedded safety systems, and the theoretical foundations of voice recognition and audio signal processing in industrial settings.

The literature reveals a dominant design philosophy oriented towards sensing and telemetry: most existing smart helmet solutions detect falls, monitor toxic gases, track GPS location, and relay physiological indicators to centralised dashboards over LoRa or cellular networks, leaving worker interaction essentially unidirectional — the helmet observes and reports, but does not communicate. Where speech technologies appear, automatic speech recognition (ASR) typically runs on server-grade or cloud-hosted hardware that depends on persistent internet connectivity and substantial computational capacity, both of which are structurally incompatible with autonomous, battery-powered wearables operating in electromagnetically noisy environments. Consequently, a practically significant gap persists: no low-cost, fully embedded solution currently enables live, bidirectional, peer-to-peer voice communication directly within the worker's own PPE, functioning reliably without dependence on external servers or uninterrupted network coverage.

SISH is explicitly positioned within this gap. Rather than a passive data-collection node, the helmet is reconceptualised as an active communication platform for which real-time, hands-free voice interaction is the primary design objective. The system is built around the ESP32-S3 dual-core microcontroller, which provides sufficient headroom for concurrent audio capture, UDP-based Wi-Fi audio streaming with sub-second end-to-end latency, and local edge processing — all within a compact, energy-efficient form factor and without requiring access to the public internet. Audio acquisition via an I2Sdigital interface ensures high-fidelity signal capture with inherent immunity to electromagnetic interference. An on-premises monitoring server provides deterministic Arabic voice-command recognition via an offline Vosk model and deferred high-accuracy transcription via Whisper, keeping the embedded firmware free of compute-intensive inference. A companion Heltec V3 node further implements a dedicated LoRa EU433 SOS/GPS emergency path that decouples emergency signalling from the voice channel, while the modular architecture accommodates future environmental sensing as operational requirements evolve.

The contribution of SISH therefore lies not in adding yet another sensor to an already well-studied family of smart helmets, but in a deliberate architectural reorientation that shifts the functional centre of gravity from passive monitoring to active, field-level voice communication, while preserving full compatibility with the IIoT safety infrastructures that surround it. Every subsequent design decision — from microcontroller selection and wireless protocol to firmware architecture and audio processing pipeline — follows directly from this positioning. Chapter 2 accordingly presents the complete system architecture, hardware design, and firmware structure through which this vision is translated into an operational and deployable engineering solution.

## **Chapter 2**

### Functional Analysis and System Design of the Smart Industrial Safety Helmet



## 2.1 Introduction

This chapter follows the theoretical and practical review presented in Chapter 1, which addressed industrial safety, communication systems in high-noise environments, and recent developments in smart helmets and connected protective equipment. It recalls the research problem, namely the difficulty of ensuring a reliable voice channel between workers and supervisors in a noisy industrial environment, together with the need to track the worker's location and provide emergency call capabilities at reasonable cost and with low power consumption.

The aim of this chapter is to move from theoretical study to a structured design of the SISH smart helmet system, by analysing the functional and non-functional requirements, then proposing a global system-level architecture, before detailing the hardware and software components and the communication mechanisms. In this way, the design becomes clearly constrained by the identified needs and constraints, instead of being a mere assembly of separate components.

The structure of this chapter is organised into four main parts. Section 2.2 presents an analysis of the system requirements and functional specifications, while Section 2.3 defines the overall SISH architecture in the form of a block diagram and data flow description. Section 2.4 focuses on the hardware design of the helmet, including the main controller, the audio acquisition and playback chain, the LoRa/GPS unit, the power supply, and the human-machine interface. Finally, Section 2.5 addresses the software aspects and communication protocols adopted in the system, and the chapter ends with a conclusion that links the initial requirements to the proposed architectural solutions.

## 2.2 System Requirements and Functional Specifications Analysis

This section starts by revisiting the core research problem: enabling effective voice communication inside a noisy industrial environment, while also providing worker tracking and emergency signalling capabilities under strict constraints on cost and power consumption. In such contexts, ambient noise can easily mask speech and conventional alarm signals, and relying on external handheld communication devices may be impractical or unsafe during work. Consequently, the proposed SISH system must integrate the communication channel directly into the helmet as part of the personal protective equipment.

From this problem statement, a set of clear functional requirements can be derived. First, the system must support a given number of helmets operating simultaneously on the same network, allowing an entire work team to communicate with each other and with a supervisor. Second, the end-to-end latency between speaker and listener must remain below a specified maximum threshold to preserve the interactivity of conversations and avoid disturbing delays.

**Audio-quality requirement** Third, the audio quality must be sufficient to ensure intelligible speech, which implies an appropriate sampling frequency and resolution, as well as robust capture and playback in the presence of strong background noise.

Fourth, the wireless coverage must span the actual working area inside the workshop or factory, whether through a local Wi-Fi infrastructure or an extended deployment. Fifth, the system must provide a dependable emergency call mechanism that can be triggered quickly and delivered reliably, ideally coupled with the worker's position when available.

In addition to these functional requirements, several non-functional design constraints must be considered. Power consumption has to be low enough to allow the helmet to operate for at least one full work shift without recharging, and the overall weight of the embedded electronics must remain acceptable to avoid discomfort or fatigue. The bill of materials should comply with a realistic cost target for a prototype, while the architecture must remain scalable so that more helmets or additional infrastructure can be added without major redesign. Ease of maintenance is also crucial, both in terms of replacing faulty components and updating firmware. Ultimately, these requirements and constraints serve as a guideline for the architectural choices detailed in the following sections, ensuring that each hardware and software design decision directly addresses the needs identified here.

## 2.3 Overall Architecture of the SISH System

The overall architecture of the SISH system is organised into three main layers, which can be represented in a block diagram (e.g. Figure 2-1). The first layer is the helmet layer, composed of the SISH helmets worn by the workers. Each helmet integrates an ESP32-S3 controller, an audio capture and playback subsystem, a Wi-Fi interface, and optionally a LoRa/GPS module. The second layer is the communication layer, based on a Wi-Fi infrastructure that interconnects the helmets with each other and with a central gateway or access point, with LoRa acting as an extension for long-range, low-bit-rate signalling when required. The third layer is the monitoring layer, implemented as a Python-based server hosting services for call management, data logging, and speech processing using tools such as Vosk for command recognition and Whisper for transcription.

The flow of data through this architecture can be described along two main paths. On the interactive audio path (the *Wi-Fi/UDP voice plane*), the microphone embedded in each helmet captures the worker's speech and sends it as a digital I2S stream to the ESP32-S3, which converts it into PCM samples and encapsulates them into packets transmitted over Wi-Fi towards another helmet or the gateway. The receiving helmet then reconstructs the audio and plays it back through the on-board speaker, while the server can transparently relay or monitor the traffic. On the recording and analysis path, the same audio streams, or selected segments, are forwarded from the helmet to the server for storage, offline transcription, and generation of reports or performance indicators. In parallel, emergency events and, where available, GPS-based location

data are carried over the *LoRa emergency telemetry plane* to dedicated monitoring applications, providing a resilient signalling channel beyond Wi-Fi coverage when required.

Each layer in this architecture has a clearly defined role that directly addresses the requirements identified in Section 2.2. The helmet layer focuses on real-time acquisition and playback with tight constraints on latency and audio quality, ensuring that the worker's experience remains natural and non-intrusive. The communication layer is responsible for providing sufficient coverage and bandwidth inside the workshop or factory, as well as robustness against packet loss and interference. The monitoring layer offloads the computationally intensive tasks, such as speech recognition and long-term logging, to a more powerful platform, which improves scalability and simplifies maintenance. By separating concerns across these layers, the architecture enhances reliability, helps control end-to-end latency, and makes it easier to expand the system to support more helmets or additional services in the future.

### 2.3.1 Layered IoT Architecture

In terms of IoT system design, the SISH architecture can be viewed as a layered structure in which the helmets form the device or perception layer, the Wi-Fi (and optional LoRa) infrastructure implements the communication or network layer, and the Python-based backend provides the processing and application layer. This view is consistent with common IoT reference models, where sensing, transport, and data processing are clearly separated to simplify design, deployment, and scaling.

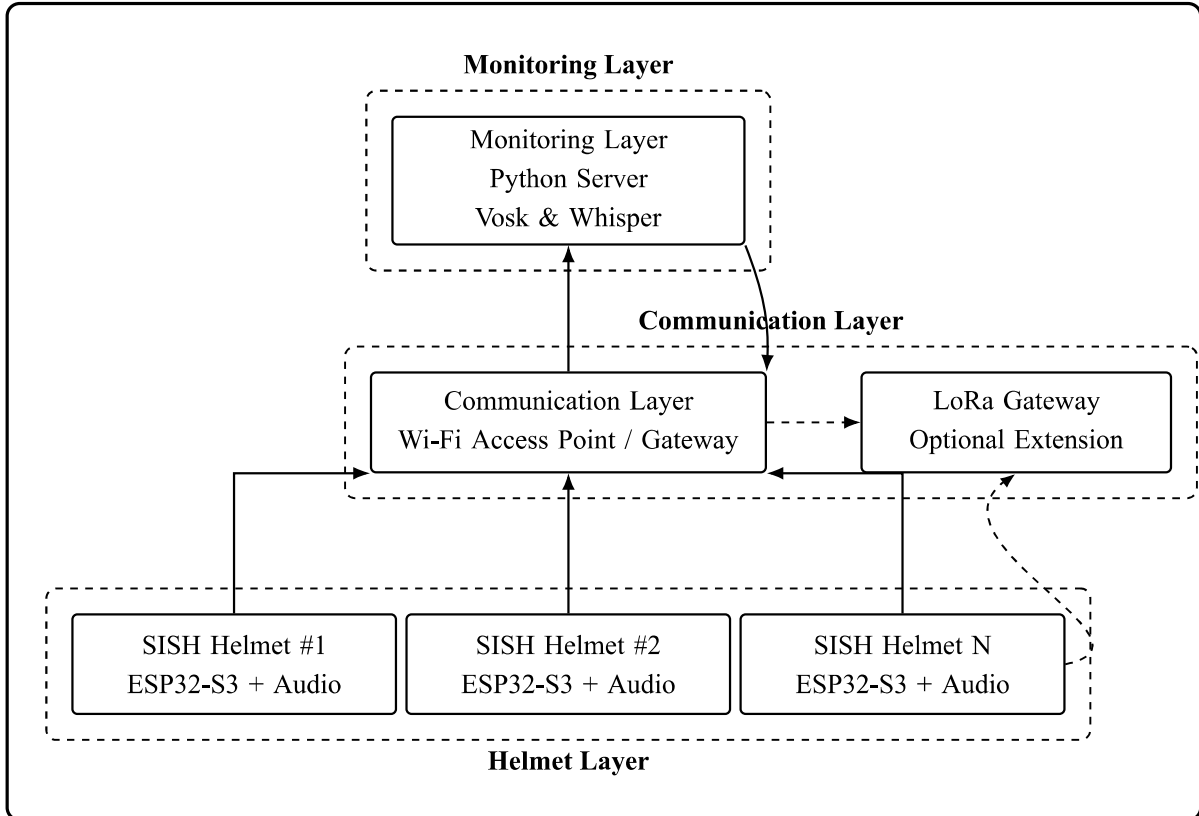
The real-time audio path starts at the helmet's microphone, which delivers a digital I2S stream to the ESP32-S3 microcontroller. The latter samples and packetizes the audio data and forwards it over Wi-Fi either directly to another helmet on the same local network or to an access point or gateway that redistributes the stream to the intended recipients. This path typically relies on a lightweight transport mechanism (such as UDP or WebSocket-based streaming) in order to minimize end-to-end latency and preserve an interactive, conversation-like user experience.

In parallel, a non-real-time path can be established from the helmets to the backend server for recording and analytics. Selected portions of the audio streams, or entire sessions when required, are sent to the Python server, where they are stored and processed by speech-to-text and command-recognition services such as Vosk and Whisper, enabling automatic log generation, safety event analysis, or performance indicators. Additionally, a low-bit-rate LoRa channel may be used to transmit short emergency messages and, when available, GPS coordinates from the helmet to a remote LoRa gateway, providing a resilient signalling path even when Wi-Fi coverage is not guaranteed.

This separation into helmet, communication, and monitoring layers directly addresses the requirements defined in Section 2.2. The helmet layer concentrates on low-latency audio acquisition and playback close to the user, ensuring speech intelligibility and natural interaction. The communication layer is responsible for extending coverage across the workshop or factory and

for transporting both voice and emergency data in a scalable way as more helmets or gateways are added. The monitoring layer offloads computationally intensive tasks to a more powerful platform, facilitating long-term storage, speech processing, and system supervision, while keeping the embedded devices simple and energy-efficient.

Figure 2.1 illustrates this three-layer architecture, showing the main data flows between the SISH helmets, the communication infrastructure, and the monitoring server.



**Figure 2.1.** Overall architecture of the SISH system.

### 2.3.2 Data Flow

The data flow in the SISH system is organized around three coordinated paths: the real-time voice stream between helmets, the low-bit-rate emergency telemetry over LoRa, and the server-side command channel back to the helmets. The primary path is the Wi-Fi voice channel, where each ESP32-S3 helmet samples 16-bit PCM audio at 16 kHz, segments it into small frames, and encapsulates these frames into custom UDP packets of approximately 329 bytes sent at a rate of about 100 packets per second. On the receiving side, each helmet applies a jitter buffer and simple packet-loss concealment to reconstruct a continuous audio stream, ensuring that mouth-to-ear latency remains within the target range defined in the requirements.

In parallel, a secondary telemetry path uses the Heltec WiFi LoRa 32 V3 module to transmit compact emergency and status messages over LoRa. This path is designed for low duty cycle

operation and carries only essential information such as SOS events, helmet identifier, and when available, GPS coordinates, which are forwarded to a point-to-point LoRa EU433. Because LoRa packets are sparse and small, this channel remains robust over long distances and through obstacles, but it is not used for continuous audio. Finally, a command path from the Python monitoring server back to the helmets distributes control messages that manage communication modes and routing decisions, allowing the server to orchestrate private calls, group sessions, and emergency alerts without overloading the embedded nodes.

### 2.3.3 Design Rationale

The design of the SISH communication architecture follows a deliberate separation of roles between Wi-Fi, LoRa, and the Python monitoring server. Wi-Fi is chosen as the primary transport for real-time voice because it offers sufficient bandwidth and low latency within the typical spatial extent of a workshop or factory, making it well suited for continuous 16-bit, 16 kHz audio streaming between helmets. LoRa, by contrast, is reserved for sparse emergency telemetry and GPS updates, where long-range coverage and energy efficiency are more important than throughput, ensuring that critical alerts can still reach the monitoring infrastructure even when Wi-Fi connectivity is degraded or unavailable.

Equally important is the decision to keep automatic speech recognition and heavy audio processing on the Python monitoring server rather than on the ESP32-S3 helmets. Offloading Vosk and Whisper to a dedicated server avoids overloading the microcontrollers, simplifies firmware complexity, and allows the use of more accurate models that would not fit within the constraints of the embedded platform. This split also improves maintainability: server-side models can be updated, retrained, or replaced without modifying the helmet hardware. Overall, the architecture reflects a pragmatic trade-off between latency, robustness, scalability, and implementation cost, aligning the strengths of each technology with the specific functional and non-functional requirements defined earlier in the chapter.

Beyond defining the logical roles of each layer and communication channel, the next section focuses on how this architecture is concretely implemented in hardware. Section 2.4 therefore details the design of the SISH helmet electronics, including the ESP32-S3 audio platform and the LoRa/GPS telemetry module that realize the functions outlined above.

## 2.4 Hardware Design of the Smart Helmet

After describing the overall architecture of the SISH system and the roles of its different layers in Section 2.3, this section focuses on the hardware design of the smart helmet itself. The helmet is treated as an integrated platform that combines the main microcontroller, the audio acquisition and playback chain, wireless communication interfaces, the GPS/LoRa telemetry module, the power supply subsystem, and the human-machine interface.

The objective of this section is to provide a structured description of the core hardware blocks and their interconnections inside the SISH helmet, while showing how they satisfy the reliability, safety, and usability requirements defined in Section 2.2. The following subsections address each hardware component in turn and justify the corresponding design choices in light of the system constraints and the target industrial environment.

### 2.4.1 System Hardware Overview

The SISH helmet is conceived as a modular embedded platform organised into five main hardware blocks: a central main controller, an audio subsystem, a communication and localisation subsystem, a power-supply block, and a human–machine interface. The main controller is implemented with an ESP32-S3-WROOM-1-N16R8 module, which integrates a dual-core ESP32-S3 microcontroller, on-board PSRAM, and 2.4 GHz Wi-Fi/BLE connectivity, and is responsible for running the application logic and managing all peripherals. The audio subsystem is based on an I2Sdigital microphone and an I2Sspeaker or digital audio amplifier, both driven directly by the ESP32-S3, which performs audio acquisition, basic processing, framing, and packetisation, but does not execute any on-device speech recognition algorithms. Long-range communication and localisation are delegated to a dedicated Heltec WiFi LoRa 32 V3 board, which combines an ESP32 microcontroller, an SX1262 LoRa transceiver, and an integrated GNSS receiver, and is used here solely as a LoRa–GPS coprocessor under the supervision of the main ESP32-S3. Power is provided by a single-cell Li-Po battery with appropriate charging and protection circuitry, feeding the two ESP32-based modules and the audio stage through high-efficiency DC–DC conversion. Finally, the human–machine interface includes push buttons and LEDs (and optionally a buzzer) distributed on the helmet shell, allowing the worker to trigger actions (e.g. emergency alerts) and receive system-status feedback without removing the helmet.

### 2.4.2 The ESP32-S3 Main Microcontroller

The main controller of the SISH helmet is the ESP32-S3-WROOM-1-N16R8 module, built around Espressif’s ESP32-S3 system-on-chip. The ESP32-S3 integrates a dual-core 32-bit Xtensa LX7 processor operating at up to 240 MHz, 2.4 GHz Wi-Fi and Bluetooth Low Energy connectivity, 512 KB of internal SRAM, and support for external flash and PSRAM; in the selected N16R8 module variant, the available memories are 16 MB of flash and 8 MB of PSRAM.[6, 5] In addition, the device provides a rich peripheral set, including programmable GPIOs, UART, SPI, I<sup>2</sup>C, and I2Sinterfaces, which makes it suitable for embedded systems requiring simultaneous sensing, audio handling, and wireless communication.[6, 5]

The choice of the ESP32-S3 is justified first by its adequate processing capability for the target application. The dual-core architecture allows the system to execute time-sensitive audio tasks, such as sampling, framing and buffering, while also maintaining communication services and application control without requiring a separate digital signal processor.[6] Its memory

resources, especially the availability of external PSRAM, are also valuable for storing temporary audio data, network packets, protocol state information, and the jitter buffer required for stable real-time communication.[5]

A second reason for selecting this microcontroller is the availability of the required hardware interfaces. The ESP32-S3 supports I2S for the digital microphone and audio output path, multiple UART channels for communication with auxiliary modules such as the LoRa/GPS subsystem, and a sufficiently large number of GPIO pins for LEDs, buttons and other control signals.[6, 5] This interface density simplifies hardware integration and reduces the need for external expanders or additional controller devices.[6]

Another important advantage is the maturity of the ESP32-S3 software ecosystem. The device is officially supported by Espressif's ESP-IDF framework and also by the Arduino-ESP32 core, which provides flexibility between low-level development and rapid prototyping.[8] This broad ecosystem is complemented by extensive documentation, reusable libraries, community examples and active developer support, all of which reduce implementation time and facilitate debugging and future upgrades.[6, 8]

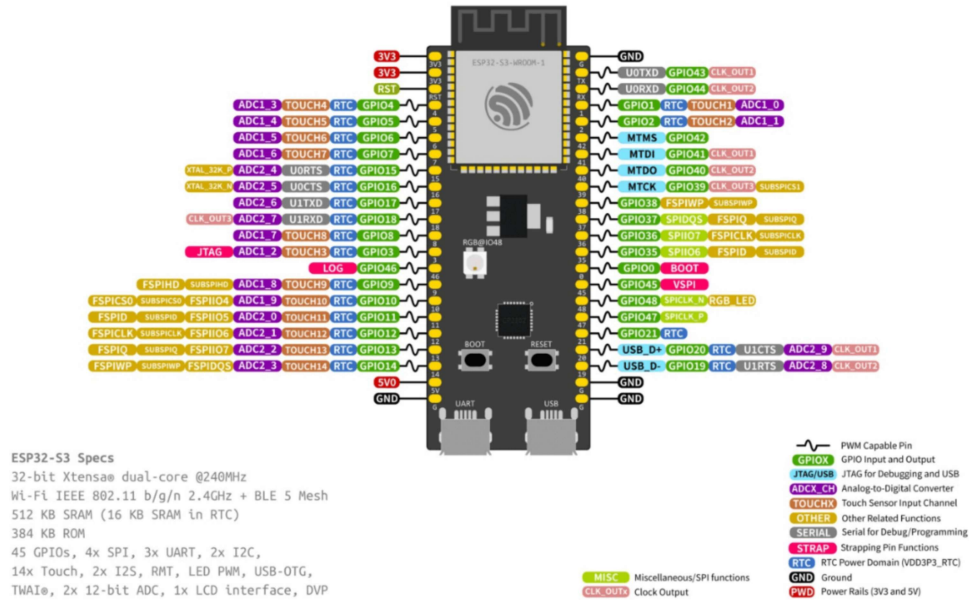
Within the SISH helmet, the ESP32-S3 therefore acts as the central "brain" of the system. It supervises audio acquisition and playback, manages wireless networking and application-layer exchanges, controls the communication state machine, and coordinates data exchange with the LoRa/GPS coprocessor. It is also responsible for maintaining the jitter buffer and for ensuring the temporal consistency of audio and telemetry flows, thereby providing the computational and organisational core of the entire helmet platform.

It should nevertheless be noted that the ESP32-S3 operates only in the 2.4 GHz band and generally consumes more power during Wi-Fi activity than ultra-low-power microcontrollers dedicated to narrowband or sub-GHz communication.[6, 17] These limitations remain acceptable in the present design because the helmet requires a compact all-in-one platform capable of handling digital audio, wireless networking and peripheral interfacing within a single programmable device.

As illustrated in Figure 2.2, the ESP32-S3 development board exposes a large number of GPIO and digital communication interfaces, which facilitates the integration of the audio front-end, the LoRa/GPS subsystem and the human-machine interface without additional expansion hardware.

### 2.4.3 Audio Subsystem: Microphone, Amplifier, and Speaker

The audio subsystem forms the front end of the SISH voice interface and is composed of three discrete elements: a digital MEMS microphone for capture, a Class-D amplifier for playback, and a passive loudspeaker. The INMP441 is an omnidirectional bottom-port MEMS microphone with a 24-bit I<sup>2</sup>S output, a signal-to-noise ratio of 61 dB(A), and a sensitivity of -26 dBFS at 94 dB SPL, producing a clean digital stream without analog conditioning [29]. The MAX98357A





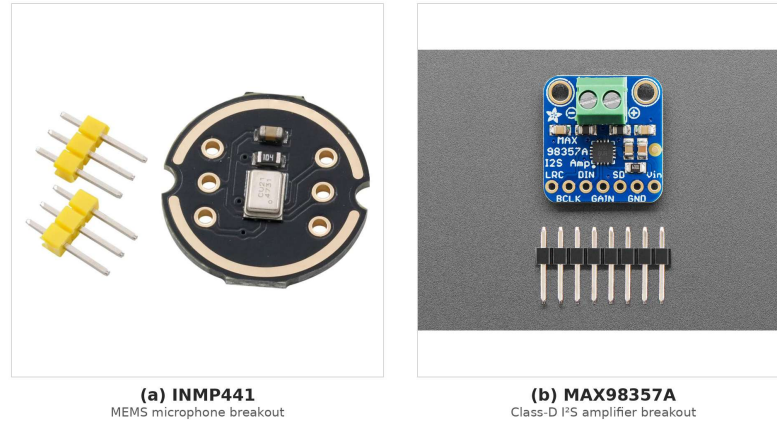
is a mono Class-D amplifier with a digital I<sup>2</sup>S input that delivers up to 3.2 W into a 4  $\Omega$  load at 5 V with about 92 % efficiency, removing the need for an external digital-to-analog converter and reducing power losses in the helmet [20]. The playback stage drives a passive 8  $\Omega$ , 3 W loudspeaker matched to the amplifier output and mounted close to the worker's ear region inside the helmet to improve acoustic coupling while limiting the required sound pressure level.

From a data-path perspective, the audio series is fully digital between the microphone and the amplifier. The INMP441 streams 24-bit I<sup>2</sup>S audio to the ESP32-S3, which exposes two independent I<sup>2</sup>S controllers, I<sup>2</sup>S0 for receive and I<sup>2</sup>S1 for transmit [6, 7]. On the uplink, the I<sup>2</sup>S0 RX line uses GPIO 13 for SCK, GPIO 12 for WS, and GPIO 11 for SD, capturing 16-bit mono PCM at 16 kHz in 10 ms frames that the firmware forwards over Wi-Fi UDP to the monitoring server [8]. The server performs mixing, routing, and speech processing before sending the audio back to the helmet, where the ESP32-S3 receives the downlink stream and uses I<sup>2</sup>S1 TX (GPIO 9 for BCLK, GPIO 10 for LRC, and GPIO 8 for DIN) to feed the MAX98357A amplifier. This arrangement implements a full-duplex voice channel with symmetric sampling parameters, which simplifies buffering and keeps end-to-end latency within the limits required for natural conversation. The complete I<sup>2</sup>S pin map between the ESP32-S3, the INMP441, and the MAX98357A is consolidated in Table 2.3.

The choice of components and the mechanical integration of the loudspeaker are driven by audio quality and noise constraints specific to industrial helmets. The INMP441 offers a flat frequency response over the speech band and a high signal-to-noise ratio, which, combined with digital I<sup>2</sup>S capture, preserves intelligibility even in the presence of high broadband background noise [29]. The MAX98357A Class-D stage was preferred over a Class-AB amplifier because its high efficiency under battery operation limits heat dissipation inside the helmet and extends the autonomy budget of the overall system [20]. Mounting the 8  $\Omega$ , 3 W loudspeaker in a small cavity near the ear with some mechanical isolation from the helmet shell improves the signal-to-noise ratio at the listener by attenuating external noise and reducing vibration-induced coloration. As illustrated in Figure 2.5, the resulting audio subsystem forms a bidirectional chain from the INMP441 microphone, through the ESP32-S3 audio endpoint and the wireless link to the monitoring server, down to the MAX98357A amplifier and the loudspeaker, meeting the voice-quality and robustness requirements of the SISH helmet in industrial environments.

#### 2.4.4 LoRa Module and GPS Localization Integration

To extend the communication range of the smart helmet beyond the limits of local Wi-Fi coverage, an additional localization and long-range alerting subsystem is considered in the SISH architecture. This subsystem combines a GPS receiver for worker positioning with a LoRa transceiver for low-power, long-distance data transmission. The purpose of this integration is to ensure that the worker's location can still be reported in outdoor or remote industrial areas where Wi-Fi connectivity may be unavailable or unreliable. Such an approach is particularly



**Figure 2.3.** INMP441 MEMS microphone breakout (left) and MAX98357A Class-D I<sup>2</sup>S amplifier breakout (right)[29, 20].

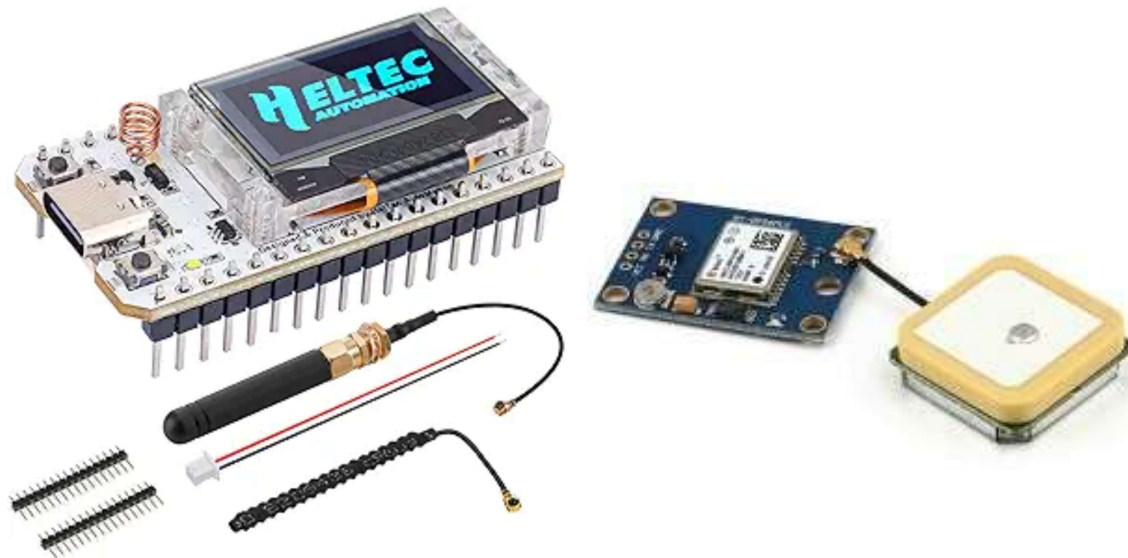
| Stage                              | Function                                                                            | Interface                                     | Key parameters                                                                               |
|------------------------------------|-------------------------------------------------------------------------------------|-----------------------------------------------|----------------------------------------------------------------------------------------------|
| INMP441 MEMS microphone            | Captures worker speech inside the helmet and converts it to a digital audio stream. | I2S0 RX (SCK, WS, SD) to ESP32-S3.            | 24-bit I <sup>2</sup> S output; SNR 61 dB(A) sensitivity -26 dBFS at 94 dB SPL[29].          |
| ESP32-S3 audio endpoint (uplink)   | Samples microphone signal and prepares audio frames for network transmission.       | Internal DMA buffers, Wi-Fi UDP uplink.       | 16-bit mono PCM; 16 kHz sampling; 10 ms audio frames to monitoring server[8].                |
| Monitoring server                  | Receives, mixes and routes audio; runs speech processing and application logic.     | Wi-Fi / IP network.                           | PC / Raspberry Pi handles multiple worker streams and downlink routing.                      |
| ESP32-S3 audio endpoint (downlink) | Receives decoded server audio and streams it to the amplifier.                      | I2S1 TX (BCLK, LRC, DIN) to MAX98357A.        | 16-bit mono PCM playback same sampling rate as uplink for symmetric latency.                 |
| MAX98357A Class-D amplifier        | Converts digital audio to an analog power signal for the loudspeaker.               | I2S digital in, differential analog out.      | Up to 3.2 W into 4 $\Omega$ at 5 V $\approx$ 92 % efficiency; no external DAC required[20].  |
| Loudspeaker (8 $\Omega$ , 3 W)     | Delivers audio feedback close to the worker's ear inside the helmet.                | Wired differential connection from amplifier. | 8 $\Omega$ impedance; 3 W rated power; mounted near ear region for improved intelligibility. |

**Table 2.1.** End-to-end audio chain from helmet microphone to loudspeaker.

relevant for emergency situations, where compact alert messages containing identification and position data must be transmitted over several kilometres while preserving battery autonomy.

The proposed subsystem is based on the Heltec WiFi LoRa 32 V3, which integrates an ESP32-S3 controller and a Semtech SX1262 LoRa transceiver on the same board [10]. This module operates in the sub-GHz band and is suitable for long-range, low-bit-rate communication, making it appropriate for periodic localization updates and emergency signalling [25]. In addition, the board includes an OLED display and power-management features that simplify its use in portable embedded applications [10].

Worker position is provided by an external u-blox NEO-6M GPS receiver, a widely used module for embedded localization tasks [30]. The NEO-6M delivers geographic coordinates and timing information through a UART serial interface, typically at 9600 baud, using standard NMEA frames that can be parsed by the microcontroller [30]. In the proposed design, the ESP32-S3 reads these GPS data, extracts the relevant fields such as latitude and longitude, and prepares a compact payload for wireless transmission.



**Figure 2.4.** Main hardware modules of the long-range localisation subsystem: Heltec WiFi LoRa 32 V3 board (left) and u-blox NEO-6M GPS receiver module (right).

Figure 2.4 shows the main hardware platform used to implement the proposed long-range localization subsystem. The Heltec board provides the embedded processing resources, the integrated LoRa radio interface and the local OLED display required for configuration feedback and status monitoring.

From an architectural point of view, this LoRa/GPS unit constitutes a separate node in the SISH system, distinct from the main audio-processing node described previously. Its role is limited to long-range emergency communication and worker localization, which helps separate functions and reduces the computational and communication load on the main controller. After receiving valid positioning data from the GPS module, the ESP32-S3 encapsulates the coordinates together with identification or alarm information into a short LoRa frame and transmits it to a gateway or remote monitoring point. This design choice is justified by the ability of LoRa

technology to provide multi-kilometre coverage at very low power consumption, which makes it particularly suitable for battery-powered personal protective equipment [25].

Depending on the development stage of the prototype, this subsystem may be considered either as an already integrated functional block or as a future extension of the current helmet design. In both cases, its inclusion in the system architecture remains justified because it improves worker traceability, increases operational safety outside Wi-Fi coverage zones, and opens the way for wider industrial deployment scenarios.

## 2.4.5 Human–Machine Interface (HMI)

The human–machine interface (HMI) of SISH is designed to indicate the system state with minimal visual and cognitive load on the worker, while remaining usable with gloves and under stress.

### 2.4.5.1 Indicators and controls

The main visual element of the HMI is a 0.96 inch monochrome OLED display based on the SSD1306 controller, integrated on the Heltec WiFi LoRa 32 V3 board and driven over I<sup>2</sup>C. Its 128 × 64 pixels, high contrast and low power consumption make it suitable for compact textual and symbolic information such as operating mode (Idle, Call, SOS) and GPS-fix status, without requiring a large or heavy screen. The SSD1306-based OLED was selected because it is widely supported in the ESP32 ecosystem, occupies little space and remains readable over a wide viewing angle and temperature range.

To provide immediate state awareness without forcing the user to read the display, the audio node includes three discrete LEDs mapped to distinct operating states: green for idle availability, yellow for “call in progress” and red for SOS. This colour coding follows common HMI conventions, where green indicates normal operation, yellow a transient or intermediate state and red a critical condition, which helps reduce interpretation errors under stress. Each LED is driven from a dedicated ESP32-S3 GPIO so that blink patterns can be used to differentiate between steady states and short events.

Control is handled by a single red tactile push-button mounted on the helmet shell. Its meaning is defined at the HMI level rather than only as a raw GPIO input: a short press is interpreted as a *Call confirm/Accept* action, whereas a long press issues an SOS alert. This dual-duration strategy avoids multiplying buttons, simplifies the mechanical interface and still provides both routine and emergency functions through one easily memorised control. The red colour and accessible placement are consistent with safety-device conventions and allow operation by feel.

Although SISH relies mainly on visual and tactile cues, the HMI is intentionally designed so that it does not require continuous visual attention. The OLED and LEDs are used to acknowledge state changes and infrequent events, while normal operation assumes that the worker

focuses on the environment. The helmet interface thus remains largely transparent during regular work and only attracts attention when necessary.

## 2.4.6 Power Supply and Autonomy

**Duty cycle assumptions** For autonomy estimation, the helmet is assumed to operate most of the time in an “idle/listen” state with periodic network activity, while high-power modes (continuous audio transmission/reception and SOS signalling) occur only intermittently.

The power subsystem is dimensioned to sustain a full work shift from a single rechargeable cell, so that the helmet remains electrically independent from external infrastructure without adding excessive weight.

The power subsystem is organised around a single 3.7 V lithium-polymer cell with a nominal capacity of 4000 mA h, chosen for its high energy density and lower weight compared with chemistries such as Ni-MH. This cell feeds both helmet nodes (audio and LoRa/GPS) through a standard JST-PH connector and an integrated battery-management circuit that provides basic protection against abnormal operating conditions. The 4000 mA h rating is dimensioned for a full industrial shift at the duty cycle defined in Section 2.4.6, with margin for increased radio and audio activity during emergencies.

To generate the voltages required by the subsystems, SISH adopts a simple two-stage distribution. The raw battery voltage is first switched by a single-pole double-throw slide switch that serves as the master On/Off control and physically isolates the cell when the system is off. Downstream, a DC–DC step-up converter based on the MT3608 boosts the 3.7 V cell output to a regulated 5.0 V rail. This rail is shared by the audio amplifier, the GPS receiver and the 5 V inputs of both the ESP32-S3 DevKit and the Heltec WiFi LoRa 32 V3, whose on-board regulators derive the internal 3.3 V logic supplies. Using a common 5 V rail reduces the number of regulators, simplifies wiring and allows unmodified use of standard development boards.

A bulk capacitor is placed on the 5 V rail near the class-D audio amplifier to buffer transient current peaks and limit voltage dips that could disturb the microcontrollers or RF modules. The INMP441 MEMS microphone is powered from the ESP32-S3 3.3 V output, which provides a clean, regulated supply compatible with its I<sup>2</sup>S interface without requiring a dedicated regulator. All modules share a common ground reference to ensure reliable digital communication and to limit ground-loop issues.

### 2.4.6.1 Charging, protection, and user independence

Charging of the lithium-polymer cell is handled by a TP4056-based linear charger with integrated protection circuitry. The device follows a constant-current/constant-voltage profile tailored to single-cell Li-ion/Li-Po batteries and terminates charging automatically when the current falls below a set threshold, bringing the cell to about 4.2 V. Over-charge, over-discharge and over-current protection on the battery output add further safety margins. The TP4056



charger is accessed via a USB-C connector on the helmet, so the cell can be recharged from standard USB power sources without removing it from the enclosure.

Supplying both SISH nodes from a single 3.7 V, 4000 mA h cell and boosting it to a unified 5 V rail allows the system to operate autonomously for a typical shift without external power cables or belt-mounted packs. The high energy density of the Li-Po cell keeps the added mass within the comfort limits of industrial helmets, while the charger and converter occupy little space inside the shell. On the HMI side, the combination of a small OLED, three LEDs and a single multipurpose button balances information richness with simplicity, enabling the worker to initiate calls or trigger SOS alerts directly from the helmet and receive local feedback without relying on a smartphone or external radio.

Taken together, the human-machine interface and power-supply architecture support the project's requirements of independence and comfort: the helmet is powered by a single compact energy source with integrated protection and presents a minimal, intuitive set of controls and indicators that do not burden the user during normal operation.

## 2.5 Communication and Software Architecture Design

The hardware platform described in Section 2.4 is complemented by a software stack designed to satisfy the requirements of Section 2.2 while preserving the dual-node architecture introduced in Section 2.3. In this architecture, audio and telemetry are carried over two complementary network planes: a Wi-Fi/UDP plane for interactive voice communication and a sub-GHz LoRa plane for low-rate emergency telemetry. Both planes converge only at the Python-based monitoring server, which centralises session management, logging and higher-level analytics. On the helmet side, the ESP32-S3-WROOM-1 firmware is deliberately restricted to real-time audio capture, transmission, reception and playback, whereas speech-to-text inference and shift-report generation are offloaded to the server [19].

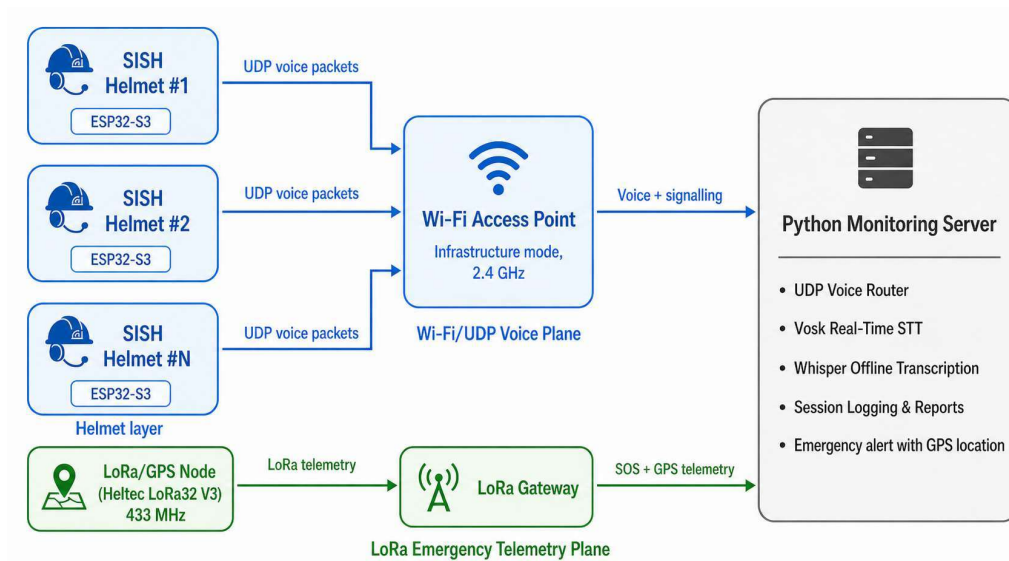
### 2.5.1 Wi-Fi Network Architecture

The SISH voice plane operates over an IEEE 802.11 infrastructure-mode network in the 2.4 GHz ISM band. An infrastructure topology is preferred over ad-hoc or mesh alternatives because it offers centralised access-point placement, coordinated channel use and controlled roaming behaviour, all of which are crucial in industrial sites where metallic structures create significant shadowing and multipath. Under these conditions, a single access point can provide reliable indoor coverage over distances on the order of 50–100 m, which satisfies the minimum line-of-sight range requirement defined in Section 2.2.

From a logical standpoint, all SISH devices share a private IPv4 subnet in which the gateway and access points occupy a reserved address range, while each helmet is assigned a persistent address that is bound to its hardware identifier. This static or reservation-based addressing

scheme ensures that the network-layer identity of each helmet remains stable across reboots and firmware updates, and that it can be mapped unambiguously to the logical helmet identifier embedded in voice packets and signalling messages.

The Wi-Fi configuration is tuned to minimise latency and jitter rather than maximise throughput. In particular, power-saving modes on the helmet stations are disabled to avoid modem-sleep wake-up delays, the deployment is restricted to non-overlapping channels to reduce mutual interference, and a WPA2-PSK security profile is adopted as a compromise between confidentiality and deployment simplicity. With these settings, the system can support a moderate number of concurrent helmets per access point while meeting the end-to-end delay and scalability requirements specified in Section 2.2; higher densities can be accommodated by adding further access points or by enabling voice-activity detection on the helmets.



**Figure 2.5.** High-level software and communication architecture of the SISH system, showing the Wi-Fi/UDP voice plane and the LoRa emergency telemetry plane converging at the Python monitoring server.

## 2.5.2 Voice Packet Protocol Design (VoicePacket Structure)

Real-time conversational audio is loss-tolerant but delay-intolerant, which directly motivates the use of UDP rather than TCP for the SISH voice transport [28, 9]. TCP retransmissions and head-of-line blocking would increase mouth-to-ear delay precisely when the network is most stressed [28, 9]; UDP, conversely, allows the receiver-side jitter buffer to drop late or corrupted

datagrams, trading negligible audio dropouts for bounded latency consistent with the ITU-T G.114 recommendation of less than 150 ms one-way delay [14].

The transmitted unit is a custom `VoicePacket` consisting of a nine-byte application header followed by the audio payload. The header carries the packet type, source helmet ID, destination helmet ID, a 16-bit sequence number, and a 32-bit millisecond timestamp. Sequence number and timestamp allow the receiver to detect missing or reordered packets and to schedule playout, consistent with the synchronization philosophy of the Real-time Transport Protocol (RTP) defined in RFC 3550 [23]. SISH adopts a compact nine-byte custom header instead of the full 12-byte RTP header, retaining only the fields strictly required by the state machine of Section 2.5.3 [23]. Table 2.2 summarizes the byte-level layout of the `VoicePacket`.

| Field                 | Type code | Size (bytes) | Description                                                             |
|-----------------------|-----------|--------------|-------------------------------------------------------------------------|
| type                  | B         | 1            | Packet type identifier                                                  |
| src_id                | B         | 1            | Source helmet ID                                                        |
| dst_id                | B         | 1            | Destination: 255 = Idle; 1–100 = Private; 101–199 = Group; 0 = reserved |
| seq                   | H         | 2            | Sequence number for ordering and loss detection                         |
| ts_ms                 | I         | 4            | uint32 ms timestamp; wrap $\approx$ 49.7 days                           |
| <b>Header total</b>   | <BBBHI    | 9            | Little-endian; no byte-order conversion required                        |
| <b>PCM payload</b>    | —         | 320          | 160 samples, 16 kHz, 16-bit, mono (10 ms frame)                         |
| <b>Datagram total</b> | —         | 329          | Transmitted at 100 packets/s                                            |

**Table 2.2.** Byte-level structure of the `VoicePacket` header (9 B) and audio payload (320 B).

Audio is sampled by the INMP441 microphone at 16 kHz with 16-bit linear PCM mono encoding, satisfying the audio-quality requirement of Section 2.2 [29]. Each packet therefore carries 160 samples (a 10 ms frame, 320 B of payload), resulting in 100 packets per second and 329-byte UDP datagrams. The 10 ms framing is shorter than the common 20 ms VoIP convention, but this design is justified by the priority on conversational delay rather than header efficiency, and experimental measurements show that the ESP32-S3 sustains 100 packets/s on both capture and playback without backlog [7].

### 2.5.3 Communication State Machine Design

The helmet voice firmware is governed by a three-state finite-state machine (FSM)—*Idle*, *Private*, and *Group*—whose transitions are driven solely by command packets issued by the monitoring server, as illustrated in Figure 2.5.

In the **Idle** state the node does not render peer or group audio; outgoing frames are still transmitted with `dst_id` set to 255 and are silently discarded at the server without routing. This design preserves a constant, deterministic transmission cadence on the embedded side while delegating all routing decisions to the server, so that the helmet firmware requires no awareness of current session topology.

On receipt of a server command the node transitions to either the **Private** state or the **Group** state. In the *Private* state the helmet exchanges audio with exactly one peer identified by a



`dst_id` in the range 1–100. In the *Group* state it transmits and accepts packets addressed to the assigned group identifier in the range 101–199. A return to *Idle* is likewise commanded by the server.

Supervisor-priority ducking is implemented at the server: when the designated supervisor (helmet ID 1) is actively speaking, concurrent peer audio is attenuated by a fixed factor so that the supervisor’s voice remains intelligible within a group session.

The helmet receiver embeds a fixed-depth jitter buffer to compensate for UDP arrival-time variation on the shared Wi-Fi infrastructure [18], [26]. The buffer holds eight slots of one 10 ms PCM frame each, for a nominal depth of 80 ms ( $8 \times 10$  ms), indexed modulo eight by the `VoicePacket` sequence number; this depth matches the headroom left for receiver buffering inside the end-to-end latency budget of §2.2 (ITU-T G.114  $< 150$  ms mouth-to-ear) once capture, network and playout delays are accounted for [19]. A fixed depth is preferred over an adaptive buffer because deterministic latency simplifies end-to-end timing analysis and is consistent with the MCPTT 300 ms access-time guidance used as a reference target [25], while avoiding the continuous jitter estimation and depth recomputation that would add code complexity and timing non-determinism on an MCU whose audio path must run with sub-millisecond regularity [18], [26].

Three mechanisms make the buffer robust to real network conditions. A soft reset on sequence jump flushes stale content and re-anchors the playout pointer when the difference between an incoming sequence number and the current pointer exceeds 100, which handles Wi-Fi reconnections and access-point hand-overs cleanly. Packet-loss concealment is implemented as zero-fill: a slot whose playout deadline expires before its packet arrives is replaced by 10 ms of silence, an approach that is computationally negligible and perceptually acceptable up to loss rates of roughly 2–3 % [18]. Finally, the buffer is per-source isolated: each remote helmet maintains an independent ring of eight slots, so loss or sequence jumps on one source cannot contaminate playback from any other source. In combination with the sequence and timestamp fields of the `VoicePacket` header defined in §2.5.2, the jitter buffer therefore provides a deterministic, loss-tolerant playout schedule that remains within the mouth-to-ear latency requirement of §2.2.

## 2.5.4 Gateway and Server Architecture

This subsection details the gateway and monitoring-server architecture that terminates the Wi-Fi and LoRa planes and exposes a unified application layer to operators and speech-recognition services. The SISH backend is deployed on an on-premises industrial PC or Raspberry Pi attached to the facility LAN; it concentrates all helmets, radios and the dashboard inside a single Layer-3 boundary, with no cloud dependency. The server receives two independent data planes—a Wi-Fi voice plane from the ESP32-S3 audio nodes and a 433 MHz LoRa telemetry plane from the Heltec WiFi LoRa 32 V3 nodes—which converge only at the application layer.

#### 2.5.4.1 Voice plane.

A Python `asyncio` bridge listens on dedicated UDP ports, parses the nine-byte `VoicePacket` header, applies the *Private/Group* fan-out logic defined by the server-side session state, and forwards each frame either to a single destination helmet or to all members of a talk group [9]. The same module implements supervisor-priority ducking and continuous WAV recording in 60-second chunks per helmet, making the server simultaneously a media relay and the audit trail required for post-incident review. Voice-plane processing is therefore limited to packet demultiplexing, routing and recording; all higher-level semantics (such as speech recognition and reporting) remain separated in the application layer.

#### 2.5.4.2 Telemetry plane.

In the implemented prototype, emergency telemetry is carried by direct point-to-point LoRa (EU433) frames transmitted by the Heltec node and received by an on-site gateway attached to the monitoring server. A ChirpStack/MQTT LoRaWAN path is shown as a conceptual future scalability extension and is not part of the current prototype implementation.

#### 2.5.4.3 Identifier reconciliation and unified worker record.

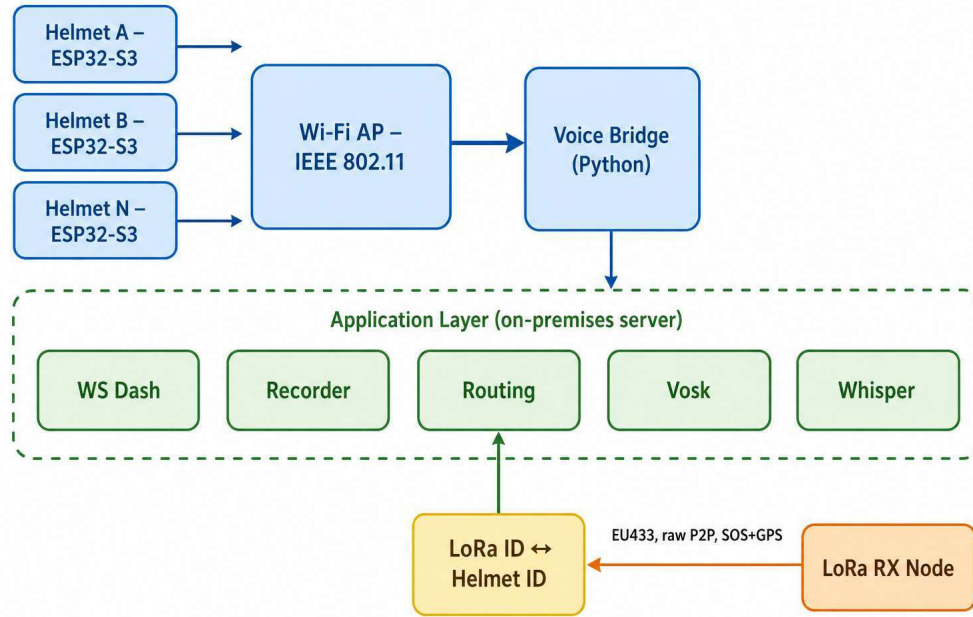
The two planes use intentionally distinct identifiers: the voice plane carries the device-level helmet ID inside the `VoicePacket` header (Section 2.5.2), whereas the LoRa plane uses the node ID rendered as `IDSISH01` inside the ASCII SOS frame (Section 2.4.4). A provisioning-time mapping table held by the server binds each LoRa node ID to its corresponding helmet ID, so that every voice session, GPS update and SOS alert is normalised into one unified worker record keyed by helmet ID. The mapping is loaded at server start-up, kept in memory, and constitutes the single point at which the two physical planes become logically unified.

### 2.5.5 Speech Recognition Integration Design (Vosk + Whisper)

#### 2.5.5.1 Voice Recognition and Transcription Pipeline

Speech processing is implemented exclusively on the on-premises monitoring server. The ESP32-S3-WROOM-1 firmware is restricted to deterministic edge functions—audio capture, packetisation, transmission, reception, jitter-buffer playout, and DAC output—and never executes Vosk, Whisper, or any other ASR model. The server therefore performs all recognition after receiving the 16 kHz PCM stream defined in §2.5.2.

This separation matches the gateway architecture of the previous subsection, where the Wi-Fi voice plane and the LoRa telemetry plane converge only at the application layer. Centralising ASR avoids compute-intensive inference on a battery-powered wearable node, preserves the timing determinism required by the audio firmware, and allows recognition to operate on either



**Figure 2.6.** SISH monitoring-server architecture: Wi-Fi/UDP voice bridge (*left*) and direct point-to-point LoRa (EU433) emergency path (*right*) converging through the LoRa-ID ↔ helmet-ID mapping into a unified application layer. The ChirpStack/MQTT path shown conceptually is identified as a future scalability extension and is not part of the implemented prototype.

live routed audio or recorded segments without modifying the embedded communication stack of §2.5.2–§2.5.4.

Two complementary recognition paths are deployed.

### 2.5.5.2 Low-latency command recognition (Vosk).

The first path targets immediate control commands and is implemented with Vosk [31]. Although the voice plane transports 10 ms frames, the server aggregates the stream into 100 ms analysis windows (3 200 bytes at 16 kHz, 16-bit mono) for recognition. The recogniser grammar is restricted to two embedded control intents — *call* and *hangup* — whose detection must satisfy the hard-latency budget imposed by the real-time voice pipeline. Group-join and status transitions are resolved server-side by the routing manager from session state and require no on-device acoustic inference (see Section 3.4.2). This closed-vocabulary design improves robustness in industrial noise compared with open-domain dictation [22].

A fixed-gain digital amplification stage precedes Vosk inference to improve speech audibility while avoiding adaptive behaviour that could introduce variable timing. Automatic gain control is not used, because the priority in the command path is bounded latency and analysable performance rather than maximal acoustic normalisation.

### 2.5.5.3 Deferred transcription (Whisper).

The second path targets documentation and reporting rather than live control. It operates on 60-second WAV recordings continuously generated by the server recorder and produces aggregated textual reports via an overnight batch job. Whisper is selected for the linguistic target (Modern Standard Arabic with regional dialect) and is treated as a high-accuracy offline transcription engine [22].

This temporal decoupling prevents transcription from competing with routing, dashboard updates, or command recognition during active use. Vosk and Whisper are therefore not interchangeable components, but engines assigned to distinct latency classes: immediate control and delayed reporting [31, 22].

### 2.5.5.4 Front-end segmentation (Silero VAD).

Before either recogniser is invoked, speech segments are extracted using a Silero voice-activity-detection stage, which suppresses silence and forwards only speech-bearing audio downstream [27]. This pre-segmentation reduces unnecessary inference and can substantially reduce the recognition compute load while adding negligible overhead [27].

### 2.5.5.5 Asynchronous integration and isolation.

ASR is integrated through asynchronous queues. Each recogniser is isolated behind an `asyncio` queue so that blocking operations cannot stall the UDP voice bridge, the routing manager, or the WebSocket dashboard event loop.

### 2.5.5.6 Closing the command loop.

Recognised control utterances from the Vosk path are translated into `PKT_COMMAND` packets, i.e., the same control messages already consumed by the firmware state machine of §2.5.3. Speech control therefore becomes an alternate front-end to an unchanged embedded control protocol rather than a separate execution path [1]. This preserves uniform state transitions across manual and voice-issued commands [1].

### 2.5.5.7 Safety constraint: emergency semantics.

Emergency signalling remains independent from speech processing. As specified in §2.5.5, emergencies are issued only by the physical SOS button over the independent 433 MHz LoRa plane, and spoken phrases are never interpreted as emergency triggers. The Vosk/Whisper integration therefore enhances communication control and post-shift reporting without modifying the fault-isolated emergency architecture of the SISH system.

# **Chapter 3**

Implementation, Hardware Assembly  
and Experimental Evaluation

### 3.1 Introduction

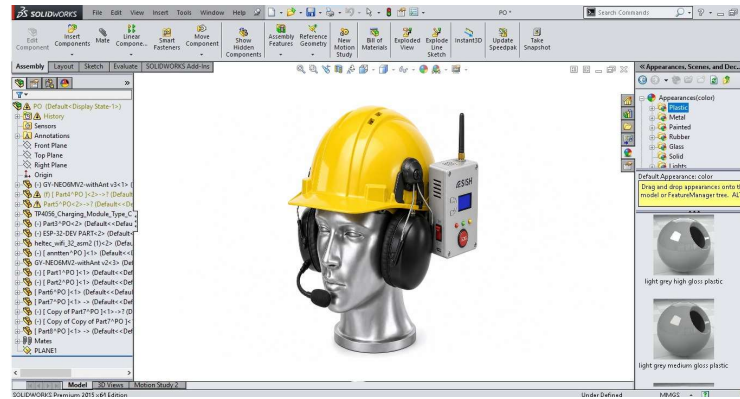
This chapter moves from the theoretical design of the Smart Industrial Safety Helmet (SISH) to its practical realisation and experimental evaluation. It documents the physical build of the dual-node prototype, including the complete pin assignments, the interconnection wiring, the power-supply and battery-management chain, and the mechanical integration of the electronics inside the helmet shell. The chapter then describes the firmware developed for the ESP32-S3 audio node and the Heltec LoRa32 V3 SOS/GPS node, as well as the Python-based gateway and server that route voice between helmets, run speech recognition, and record safety sessions. Finally, it defines the test scenarios and methodology, reports measured results for key performance indicators such as loss tolerance, 100 pkt/s streaming stability, and SOS path integrity, and derives design-based bounds for end-to-end audio latency and LoRa time-on-air in order to compare the implemented prototype against the objectives set out in Chapter 2.

### 3.2 Hardware Assembly and Component Integration

The present section documents the physical implementation of the SISH prototype, from the complete dual-node pin assignment to the inter-component wiring, the power and battery-management chain, and the mechanical mounting inside the helmet shell. Each hardware element realises the design rationale established in Section 2.3.3; the focus here is strictly on how the build was carried out and on the resulting physical configuration of the system, with particular attention to ergonomics and safety so that the added electronics do not compromise the protective role of the helmet.

The prototype adopts a dual-node architecture. The first node, based on an ESP32-S3 module, implements the audio endpoint: it hosts the I<sup>2</sup>S digital microphone, the I<sup>2</sup>S amplifier and speaker, three status LEDs, and the Wi-Fi interface. The second node uses a Heltec Wi-Fi LoRa 32 V3 module and carries the positioning and emergency functions, namely the NEO-6M GPS receiver, the SX1262 LoRa transceiver, and the SOS push-button. The two boards do not exchange data directly; they share only the regulated 5 V power rail and a common ground. This separation of concerns limits the complexity of each firmware image and allows the audio and emergency subsystems to evolve independently, while logical convergence occurs exclusively at the server side through the provisioning-time mapping between the LoRa identifier and the helmet identifier introduced in Section 2.5.4.3.

In the following subsections, we first present the complete pinout and component mapping for both nodes (Table 3.1), then describe the wiring and power distribution, and finally detail the mechanical integration of all modules inside the helmet.



**Figure 3.1.** SISH smart industrial helmet prototype — dual-node assembly mounted on an EN 397 industrial helmet shell.

### 3.2.1 Complete Pinout and Component Mapping

Table summarises the full pin assignment for both nodes as actually wired in the SISH prototype. A dedicated *Node* column distinguishes between the ESP32-S3 audio node and the Heltec WiFi LoRa 32 V3 GPS/LoRa node. GPIO numbers that appear on both rows (for example 8–13) correspond to physically distinct pads on different microcontrollers and therefore do not represent a pin conflict in the assembled system.

On the ESP32-S3, the audio subsystem is split over the two hardware I<sup>2</sup>S peripherals: the digital microphone is connected to the I2S\_NUM\_0 interface, while the I<sup>2</sup>S amplifier driving the speaker is connected to I2S\_NUM\_1. This configuration enables full-duplex audio, i.e. concurrent capture and playback, without peripheral sharing or time-multiplexing.

On the Heltec WiFi LoRa 32 V3, the SX1262 LoRa transceiver and the SSD1306 OLED display are factory-routed on the PCB and therefore do not require any additional wiring in the prototype. For completeness, they are included in Table as fixed on-board connections, alongside the externally wired NEO-6M GPS module and the SOS push-button.

| Node      | Signal                   | GPIO        | Peripheral / Component                | Direction | Notes                                                        |
|-----------|--------------------------|-------------|---------------------------------------|-----------|--------------------------------------------------------------|
| ESP32-S3  | MIC_SD (data)            | 11          | INMP441 MEMS mic, I2S_NUM_0           | Input     | 16 kHz / 16-bit, ONLY_LEFT                                   |
| ESP32-S3  | MIC_WS (word select)     | 12          | INMP441, I2S_NUM_0                    | Input     |                                                              |
| ESP32-S3  | MIC_SCK (bit clock)      | 13          | INMP441, I2S_NUM_0                    | Input     |                                                              |
| ESP32-S3  | SPK_DIN (data in)        | 8           | MAX98357A Class-D amp, I2S_NUM_1      | Output    | drives 8 $\Omega$ / 3 W speaker                              |
| ESP32-S3  | SPK_BCLK (bit clock)     | 9           | MAX98357A, I2S_NUM_1                  | Output    |                                                              |
| ESP32-S3  | SPK_LRC (LRCK)           | 10          | MAX98357A, I2S_NUM_1                  | Output    |                                                              |
| ESP32-S3  | LED_GREEN (Idle)         | 4           | Status LED                            | Output    | ON in Idle/standby                                           |
| ESP32-S3  | LED_YELLOW (Active call) | 5           | Status LED                            | Output    | ON in Private/Group                                          |
| ESP32-S3  | LED_RED (SOS)            | 6           | Status LED                            | Output    | reserved for LoRa-SOS path; never written by the Wi-Fi stack |
| Heltec V3 | SOS button               | 46          | Tactile push-button $\rightarrow$ GND | Input     | INPUT_PULLUP, falling-edge IRQ, $\sim$ 50 ms debounce        |
| Heltec V3 | GPS RX2                  | 48          | NEO-6M TX $\rightarrow$ ESP32-S3 RX   | Input     | UART2, 9600 8N1                                              |
| Heltec V3 | GPS TX2                  | 47          | NEO-6M RX $\leftarrow$ ESP32-S3 TX    | Output    | UART2, 9600 8N1                                              |
| Heltec V3 | NSS (chip select)        | 8           | SX1262 (factory-routed)               | —         | do not wire externally                                       |
| Heltec V3 | DIO1 (IRQ)               | 14          | SX1262 (factory-routed)               | —         | do not wire externally                                       |
| Heltec V3 | NRST (reset)             | 12          | SX1262 (factory-routed)               | —         | do not wire externally                                       |
| Heltec V3 | BUSY                     | 13          | SX1262 (factory-routed)               | —         | do not wire externally                                       |
| Heltec V3 | SCK / MOSI / MISO        | 9 / 10 / 11 | SX1262 SPI (factory-routed)           | —         | do not wire externally                                       |
| Heltec V3 | OLED (I2C)               | factory     | SSD1306 128 $\times$ 64               | —         | Vext-gated power                                             |

**Table 3.1.** Complete dual-node GPIO / component map (ESP32-S3 audio node + Heltec V3 GPS/LoRa node)

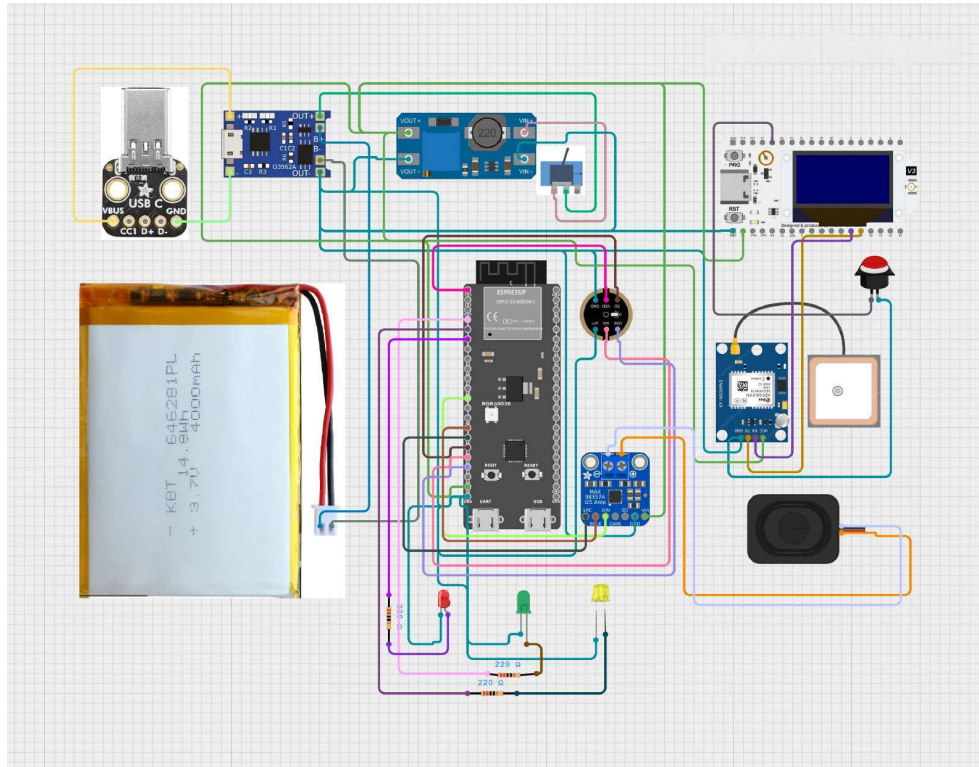
The ESP32-S3 uses three independent status LEDs rather than a single indicator, allowing the operator to read the link state at a glance: green for Idle, yellow for an active private or group call, and red dedicated exclusively to the LoRa-SOS subsystem. GPIO 46 was selected on the Heltec V3 because it is a free, interrupt-capable, non-strapping pin that does not conflict with the SX1262 control/SPI lines, the GPS UART2 pins, or the USB-UART programming interface.

### 3.2.2 Wiring Diagram

Figure 3.2 shows the complete wiring of the prototype, including the ESP32-S3 node and the Heltec V3 node, together with their shared power supply. The INMP441 microphone and MAX98357A audio amplifier are connected to the ESP32-S3 over their dedicated I<sup>2</sup>S interfaces as summarized in Table 2.1; the loudspeaker is driven directly from the amplifier output, while the three status LEDs are each connected to GPIOs 4, 5, and 6 through individual current-limiting resistors. On the Heltec V3 side, only two external elements are added: the SOS push-button, wired between GPIO 46 and ground, and the NEO-6M GPS receiver, which is interfaced via UART2 on GPIOs 47 (TX) and 48 (RX) according to the hardware design of Section 2.4.4. The



two boards share only the 5 V supply rail and a common ground reference, with no signal-level connections between them, so that each node operates independently while remaining powered from the same source.



**Figure 3.2.** Complete wiring diagram of the SISH prototype, showing the ESP32-S3 node, the Heltec V3 node and the shared power supply.

Figure 3.2 shows the complete wiring of the prototype, including the ESP32-S3 node and the Heltec V3 node, together with their shared power supply. ...

### 3.2.3 Power Supply and BMS Circuit

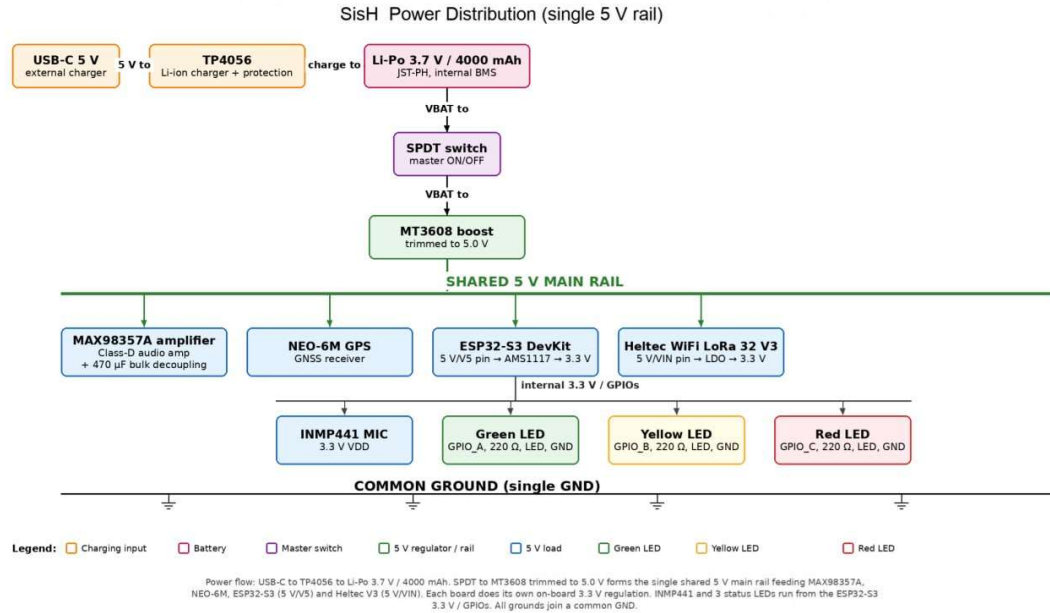
Figure 3.3 details the power and battery-management chain adopted for the SISH helmet. A 5 V USB-C connector provides the external charging input, which feeds a TP4056 linear charger with integrated Li-ion protection. The TP4056 charges a single 3.7 V / 4000 mAh Li-Po cell (JST-PH connector) that integrates basic internal protection against over-charge, over-discharge and short-circuit conditions. (The commercial Li-Po pack used in the prototype includes an embedded protection circuit; therefore, no additional discrete BMS IC is required at pack level.)

The battery output, labelled VBAT, is routed through a front-panel SPDT slide switch that acts as the master on/off control for the whole system. When the switch is in the *ON* position, VBAT is applied to an MT3608 boost converter configured to generate a regulated 5.0 V rail. The

converter output forms a single shared 5 V main rail that powers all loads in parallel, while the Li-Po cell remains the only energy storage element in the system.

Downstream of this node, each board performs its own local 3.3 V regulation. The ESP32-S3 DevKit module includes an on-board AMS1117-class linear regulator that converts the 5 V input pin to the internal 3.3 V MCU rail. The Heltec WiFi LoRa 32 V3 module implements a similar 5 V to 3.3 V conversion via its on-board LDO. The digital microphone (INMP441), the GNSS receiver (NEO-6M) and the status LEDs are powered from the corresponding 3.3 V or 5 V pins exposed by these modules, so no additional discrete regulators are required on the custom PCB.

A single common ground reference ties the charger, battery, boost converter, both processing nodes and all peripherals together. This shared ground is the only electrical coupling between the two logical nodes: there is deliberately no data or control bus implemented between the ESP32-S3 node and the Heltec LoRa node in this prototype. The power path is therefore strictly unidirectional from the Li-Po cell towards the individual regulators, which simplifies isolation and fault analysis while still allowing both nodes to operate concurrently from the same energy source.



**Figure 3.3.** Power distribution and battery-management chain for the SISH helmet (USB–C input, TP4056 charger/protection, 3.7 V Li-Po cell, master switch, MT3608 boost to a shared 5 V rail, and local 3.3 V regulation on each node).

### 3.2.4 Mechanical Integration inside the Helmet

The mechanical integration of the SISH prototype is documented using 3-D CAD renders rather than bench photographs, so that every component, interface, and cable route can be seen clearly and labelled unambiguously against Table 3.2. The exploded view in Figure 3.4 presents the complete dual-node assembly: the Heltec V3 module with its on-board OLED and GPS patch antenna, the helical 433 MHz LoRa antenna, the TP4056 USB-C charging module, the ESP32-S3 node, the 3.7 V / 4000 mAh Li-Po cell, and the two-part 3D-printed enclosure whose front panel carries the moulded SOS aperture. The second render, Figure 3.5, shows the system fully assembled on an EN 397 industrial helmet shell and worn on a headform, making the final positions of the user-facing elements explicit: the microphone on a short boom in front of the wearer’s mouth, the mini-speaker at the ear, the red SOS push-button on the enclosure flank within easy reach of the hand, and the LoRa antenna mounted on the rear crown to maintain a clear line-of-sight above surrounding obstacles.

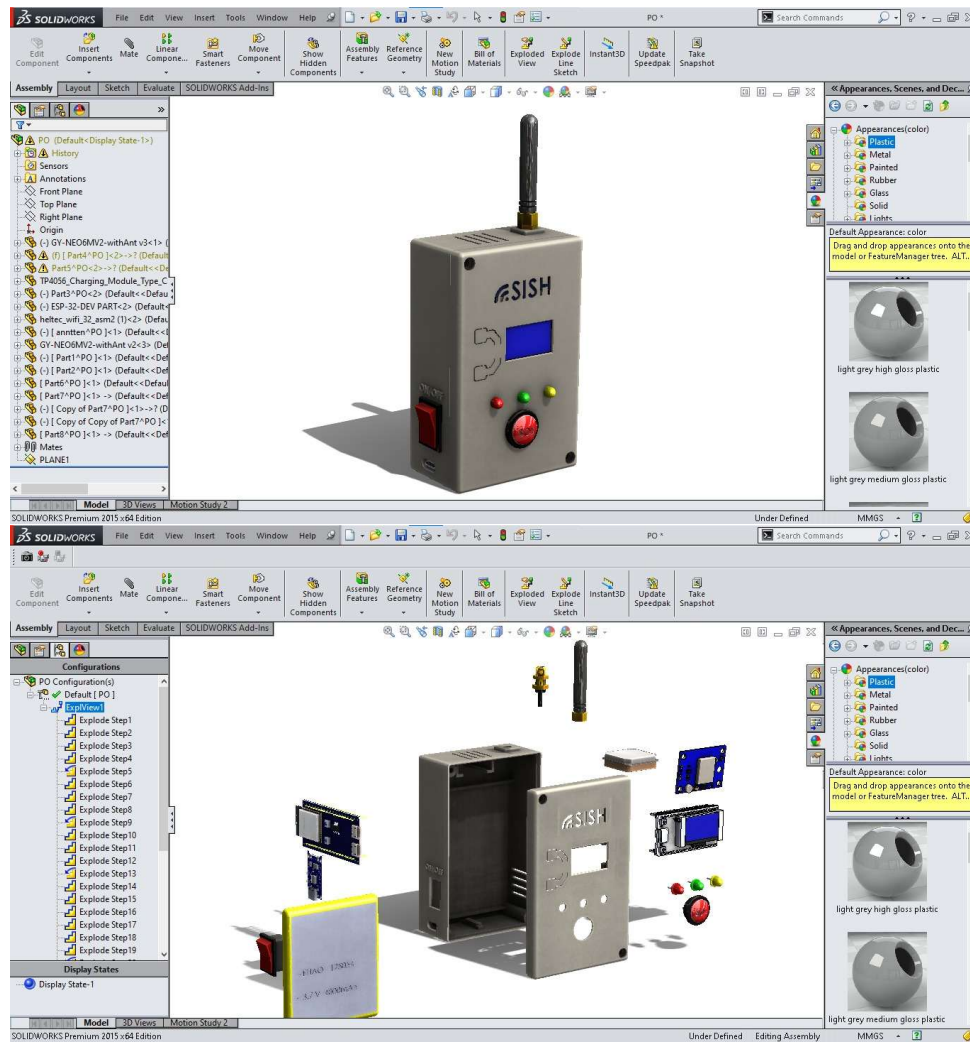
Internally, the Heltec V3, ESP32-S3 node, TP4056 charger, and battery are stacked and fixed inside the printed enclosure using integral standoffs and screws, with cut-outs that align the OLED window and SOS push-button with the external front panel. The 3D-printed geometry also provides mechanical protection for the Li-Po cell and defines a fixed connector bay for the microphone boom, ear speaker, and antenna feed, ensuring that cable strain is taken by the housing rather than by solder joints. Importantly, the enclosure is attached to the EN 397

helmet using existing harness and accessory interfaces, without any drilling or modification of the shell, so the certified impact-protective structure remains intact and the SISH module can be removed or upgraded without affecting compliance. All internal signal wiring uses 26 AWG silicone-insulated conductors terminated by direct soldered joints protected with heat-shrink tubing, which keeps the harness flexible while the thicker, more abrasion-resistant insulation tolerates repeated donning and doffing of the helmet, whereas the single Li-Po battery feed retains its standard JST-PH output connector to allow safe cell replacement without desoldering near the battery. Presenting the integration through CAD renders avoids the visual clutter and occlusions inherent in bench photographs and allows each part to be highlighted, numbered, and cross-referenced directly to Table 3.2, improving traceability between the mechanical design and the bill of materials.

| Item             | Component (prototype)                     |
|------------------|-------------------------------------------|
| Main MCU (voice) | ESP32-S3 DevKit                           |
| MEMS microphone  | INMP441 I2S module                        |
| Audio amplifier  | MAX98357A I2S Class-D module              |
| Safety MCU/radio | Heltec WiFi LoRa 32 V3 (SX1262)           |
| GNSS receiver    | u-blox NEO-6M module                      |
| Battery/charging | 3.7 V Li-Po + TP4056 charger/protection   |
| Regulation       | MT3608 boost to 5 V + on-board 3.3 V LDOs |

**Table 3.2.** Main components integrated in the SISH prototype (bill of materials summary).

From a physical integration standpoint, the complete electronics assembly (ESP32-S3 node, Heltec V3 node, audio front-end, wiring and battery) adds approximately 300 g to the bare EN 397 helmet, resulting in a total weight of about 900 g for the fully equipped SISH prototype. The added mass is distributed close to the crown of the helmet and does not protrude beyond the original envelope, so the centre-of-gravity shift remains limited and users reported no noticeable discomfort during the short-duration laboratory tests. A more systematic ergonomic evaluation over full work shifts is left for future work.



**Figure 3.4.** Exploded view of the complete SISH dual-node assembly (CAD), showing the ESP32-S3 audio node, Heltec V3 LoRa/GPS node, battery pack, and mechanical mounting components.

**Figure 3.5.** SISH prototype installed on an EN 397 industrial helmet (overall placement of user-facing elements).

### 3.3 Firmware Implementation (Dual-Node)

The SISH v4 helmet firmware runs on two independent microcontrollers that share only the 5 V rail and a common ground, with no on-board data link. The ESP32-S3 node runs the real-time, full-duplex voice stack over Wi-Fi/UDP as a pure audio endpoint, while the Heltec WiFi LoRa 32 V3 node handles the safety chain—physical SOS detection and GPS-stamped LoRa transmission over EU433. The two firmwares never exchange application data on the helmet;

their streams converge only at the server through the LoRa-ID  $\leftrightarrow$  helmet-ID mapping defined in Section 2.5.4. This keeps SOS available even when the voice path is congested or idle, and prevents safety signalling from competing with the 100 packets/s audio load.

### 3.3.1 Development Environment and Tools

Both nodes are developed under the Arduino-ESP32 core but are built and flashed independently with role-specific libraries; the physical pin assignments and wiring for both boards are those established in Section 3.2.1 and are not repeated here. The voice node (ESP32-S3) uses the ESP-IDF I<sup>2</sup>S driver for DMA-backed audio, the AsyncUDP library for non-blocking packet I/O, and two shared headers identical across all helmets: `voice_protocol.h` (wire format, packet types, ports) and `jitter_buffer.h` (de-jitter and packet-loss concealment). A single compile-time `HELMET_ID` macro sets the node identity and derives the UDP ports as `TX_PORT = 15000 + HELMET_ID` and `RX_PORT = 15010 + HELMET_ID`, giving 15001–15003 for transmit and 15011–15013 for receive. All units share one unified subnet, 10.46.46.x (H1 = .73, H2 = .249, H3 = .13, bridge = .95), removing any cross-subnet routing dependency.

The safety node (Heltec LoRa32 V3) is built around two libraries the voice node does not use. Radio control is provided by RadioLib, which supports the factory-integrated Semtech SX1262 transceiver and exposes a `begin()/setFrequency()/setOutputPower()/transmit()` API with explicit `RADIOLIB_ERR_*` return codes for deterministic verification of each transmission. Satellite positioning is handled by TinyGPS++, which parses the NMEA sentence stream from the u-blox NEO-6M and exposes validated latitude, longitude and satellite-count fields. The SX1262 is driven on its factory pin map and the NEO-6M over UART2, both wired as defined in Section 3.2.2, so this firmware introduces no new physical interface beyond the emergency push-button input detailed in Section 3.3.6.

| Aspect              | Voice node — ESP32-S3                                        | Safety node — Heltec LoRa32 V3          |
|---------------------|--------------------------------------------------------------|-----------------------------------------|
| Transport library   | AsyncUDP (Wi-Fi/UDP)                                         | RadioLib (SX1262 LoRa, EU433)           |
| Audio/sensor driver | ESP-IDF I <sup>2</sup> S driver                              | TinyGPS++ (NEO-6M, UART2)               |
| Project headers     | <code>voice_protocol.h</code> , <code>jitter_buffer.h</code> | —                                       |
| Identity source     | compile-time <code>HELMET_ID</code>                          | <code>NODE_ID</code> (mapped at server) |
| Network plane       | 10.46.46.x Wi-Fi subnet                                      | EU433 433 MHz LoRa link                 |

**Table 3.3.** Firmware toolchain per node.

### 3.3.2 I2S Audio Capture (ESP32-S3)

Audio capture uses `I2S_NUM_0` configured as a master receiver at 16 kHz, 16-bit, mono (`I2S_CHANNEL_FMT_ONLY_LEFT`), with four DMA buffers each holding 160 samples, so that one DMA transfer equals exactly one 10 ms frame, as justified in Section 2.5.2. The INMP441



microphone is brought up with its L/R-select pad tied to GND so that valid PCM appears on the left channel the firmware reads, resolving an initial silent-capture fault; the frame width is fixed by `FRAME_SAMPLES = 160` and `FRAME_BYTES = 320`, reused unchanged by the transmit path and jitter buffer.

**Listing 3.1.** `init_mic()`: I2S\_NUM\_0 master-RX, 16 kHz/16-bit/mono, 4 DMA buffers  $\times$  160 samples = one 10 ms frame.

```

1 void init_mic() {
2     i2s_config_t cfg = {
3         .mode                = (i2s_mode_t)(I2S_MODE_MASTER |
4             I2S_MODE_RX),
5         .sample_rate         = SAMPLE_RATE,
6         .bits_per_sample     = I2S_BITS_PER_SAMPLE_16BIT,
7         .channel_format      = I2S_CHANNEL_FMT_ONLY_LEFT,
8         .communication_format = I2S_COMM_FORMAT_STAND_I2S,
9         .intr_alloc_flags    = ESP_INTR_FLAG_LEVEL1,
10        .dma_buf_count       = 4,
11        .dma_buf_len         = FRAME_SAMPLES,
12        .use_apll             = false
13    };
14    i2s_pin_config_t pins = {
15        .bck_io_num    = MIC_SCK,
16        .ws_io_num     = MIC_WS,
17        .data_out_num  = I2S_PIN_NO_CHANGE,
18        .data_in_num   = MIC_SD
19    };
20    i2s_driver_install(I2S_NUM_0, &cfg, 0, NULL);
21    i2s_set_pin(I2S_NUM_0, &pins);
22 }
```

### 3.3.3 UDP Voice Streaming (ESP32-S3)

Each captured frame is encapsulated in a `VoicePacket` of a 9-byte header plus 320 bytes of PCM, for a 329-byte datagram emitted at 100 packets per second (one packet per 10 ms frame). The header is packed with `#pragma pack(1)` and serialised with the format string `<BBBHI`—little-endian, matching the ESP32-S3 native memory layout—so helmet and bridge read and write the multi-byte fields (`seq` as `H`, `ts_ms` as `I`) without byte-order conversion. The transmit task computes frame RMS, sets destination, sequence and timestamp, and writes the packet to the bridge, satisfying the latency target stated in Section 2.2.

**Table 3.4.** VoicePacket wire format (<BBBHI little-endian header + PCM); total 329 bytes;  $ts\_ms \text{ uint32} \Rightarrow 49.7\text{-day wrap}$ .

| Field        | C type       | Fmt    | Bytes      | Description                      |
|--------------|--------------|--------|------------|----------------------------------|
| type         | uint8_t      | B      | 1          | PKT_AUDIO / PKT_COMMAND          |
| src_id       | uint8_t      | B      | 1          | sending helmet (1–3)             |
| dst_id       | uint8_t      | B      | 1          | destination/group (255 = idle)   |
| seq          | uint16_t     | H      | 2          | sequence number (loss detection) |
| ts_ms        | uint32_t     | I      | 4          | ms timestamp (49.7-day wrap)     |
| audio        | int16_t[160] | —      | 320        | 10 ms PCM frame                  |
| <b>Total</b> | —            | <BBBHI | <b>329</b> | header 9 B + PCM 320 B           |

The timestamp field is finalised as an unsigned 32-bit value (`uint32_t ts_ms`), ensuring a monotonic horizon of approximately 49.7 days at 1 ms resolution while preserving the 329-byte datagram size and the nine-byte header layout.

### 3.3.4 Jitter Buffer Implementation (ESP32-S3)

The de-jitter stage applies the buffering design of Section: each buffer provides 8 slots ( $8 \times 10 \text{ ms} = 80 \text{ ms}$ ) of elastic storage indexed by  $seq \% \text{JITTER\_SLOTS}$ , with one independent buffer per source so that concurrent talkers are reordered separately. A soft reset triggers on a sequence jump greater than `JITTER_SEQ_JUMP_THRESHOLD` (100), marking a session or group change and re-synchronising `next_seq` without reallocating memory, while missing slots are concealed by zero-fill packet-loss concealment instead of an audible gap.

**Listing 3.2.** `jitter_push()`: 8-slot/80 ms buffer, seq-jump soft reset ( $>100$ ), zero-fill PLC.

```

1 static inline void jitter_push(JitterBuffer *jb, VoicePacket *pkt) {
2     if (!jb->initialized) {
3         jb->next_seq    = pkt->seq;
4         jb->last_ts     = pkt->ts_ms;
5         jb->initialized = true;
6     }
7     uint16_t diff = (uint16_t)(pkt->seq - jb->next_seq);
8     if (diff > JITTER_SEQ_JUMP_THRESHOLD && diff < 65000) {
9         jitter_reset(jb);
10        jb->next_seq    = pkt->seq;
11        jb->initialized = true;
12    }
13    uint8_t slot      = pkt->seq % JITTER_SLOTS;
14    jb->packets[slot] = *pkt;

```



```

15     jb->valid[slot]      = true;
16     jb->last_ts         = pkt->ts_ms;
17 }

```

### 3.3.5 Communication State Machine (ESP32-S3)

The voice node implements exactly three modes—*Idle*, *Private* and *Group*—driven solely by inbound PKT\_COMMAND packets, as justified in Section 2.5.3. On a command, `on_udp_packet()` dispatches on `dst_id`: 255 enters Idle (standby/muted), 101–199 enters Group, and 1–100 enters a Private call; `dst_id=0` is reserved and silently dropped. Outbound audio is gated by an RMS-VOX threshold with a 50-frame hangover, and every transition plays a 440 Hz beep.

In the final design, any former emergency or broadcast voice modes are deliberately removed from the audio firmware. Emergency signalling is handled exclusively by the decoupled LoRa-based SOS path, and the UDP voice plane exposes only the three non-emergency modes (Idle, Private, Group), with `dst_id=0` reserved and silently dropped.

**Listing 3.3.** `on_udp_packet()`: three-mode PKT\_COMMAND dispatch — Idle(255) / Private(1–100) / Group(101–199); `dst_id=0` dropped.

```

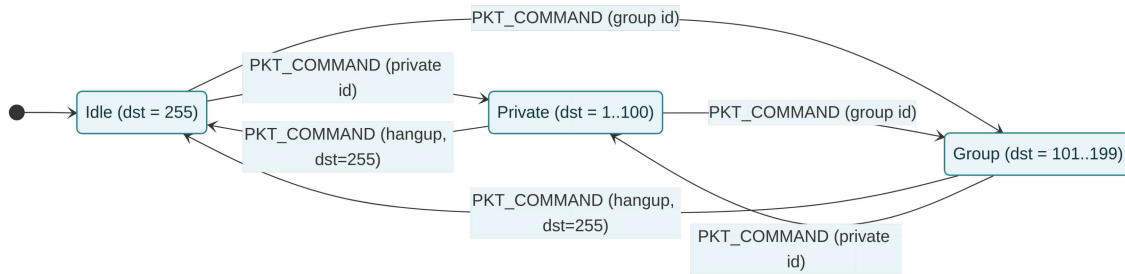
1  void on_udp_packet(AsyncUDPPacket &raw) {
2      if (raw.length() < sizeof(VoicePacket)) return;
3      VoicePacket *vp = (VoicePacket*)raw.data();
4      if (vp->type == PKT_COMMAND) {
5          uint8_t new_dst = vp->dst_id;
6          if (new_dst == DST_IDLE)      set_idle_mode();
7          else if (is_group_id(new_dst)) set_group_mode(new_dst);
8          else if (is_private_id(new_dst)) set_private_call_mode(
9              new_dst);
10         return;
11     }
12     if (vp->type != PKT_AUDIO) return;
13     if (vp->src_id == HELMET_ID) return;
14     jitter_push(&jb[vp->src_id], vp);
15 }

```

Figure 3.7 presents the experimental evidence that the three-mode state machine of Listing 3.3 operates exactly as specified: the server routes commands and audio while the helmet transitions between states solely on receipt of PKT\_COMMAND packets.

Listing 3.4 reproduces a real serial-monitor capture of helmet H1 during a live private-call session, recorded alongside the server console. After Wi-Fi association the node prints `Connected! IP: 10.46.46.73`, confirming attachment to the unified 10.46.46.x plane. While idle, every transmitted packet is tagged `dst=255 | mode=IDLE/MUTED` with a low VOX RMS

**Figure 3.6. Server and helmet view — voice routing.** *Left:* Python monitoring server (`mainv4.py --full`) showing STT fast-trigger detection, Helmet2 ↔ Helmet1 private-call routing, 10s traffic statistics, hangup back to Idle(muted), and automatic WAV recording. *Right:* Helmet H1 serial monitor showing Connected! IP: 10.46.46.73, IDLE/MUTED → PRIVATE → IDLE transitions driven by PKT\_COMMAND packets, and VOX RMS values during speech.



**Figure 3.7. State-machine diagram — communication modes.** *Left:* Python monitoring server console confirming bridge routing, fast-trigger STT command detection, and automatic WAV recording. *Right:* Helmet H1 serial monitor showing the IDLE → PRIVATE → IDLE state-machine transitions with VOX RMS values, captured in the same session.

(typically below 20). On receipt of a PKT\_COMMAND, the node prints [MODE] Private Call <-> Helmet 2 and switches to dst=2 | mode=PRIVATE; a spoken utterance then drives the VOX RMS above the threshold (e.g. 155, 203, 211). A subsequent hangup command returns it to [MODE] IDLE (Standby/Muted). This capture provides experimental evidence that the three-mode state machine of Listing 3.3 operates exactly as specified, driven solely by inbound command packets.

```

1 Connected! IP: 10.46.46.73
2 [TX] dst=255 seq=222 | mode=IDLE/MUTED | VOX RMS: 162
3 [SISH] Helmet1 | seq:643 | dst:255 | mode=IDLE
4 [MODE] Private Call <-> Helmet 2
5 [TX] dst=2 seq=1138 | mode=PRIVATE | VOX RMS: 155
6 [TX] dst=2 seq=1398 | mode=PRIVATE | VOX RMS: 41
7 [SISH] Helmet1 | seq:1398 | dst:2 | mode=PRIVATE
8 [TX] dst=2 seq=1568 | mode=PRIVATE | VOX RMS: 211
9 [MODE] IDLE (Standby/Muted)
10 [TX] dst=255 seq=2066 | mode=IDLE/MUTED | VOX RMS: 122
11 [SISH] Helmet1 | seq:2088 | dst:255 | mode=IDLE

```

**Listing 3.4.** Serial-monitor capture of helmet H1 during a live private call (three-mode state machine driven by inbound commands).

On the server side of the same session, the bridge log confirms the matching routing decisions: Helmet 2 requests a private call to the supervisor (Helmet 1); the gateway then issues CMD -> Helmet2: helmet/group 1 and CMD -> Helmet1: helmet/group 2, opening Private call: Helmet2 <-> Helmet1. The 10-second statistics line RX: H1=1001 H2=441 | total\_rx=1442 | TX(routed): H1=321 H2=776 | total\_tx=1097 shows live cross-routing between the two endpoints, after which a hangup command returns both helmets to the idle (muted) state and the recorder reports Saved: helmet1\_...wav / helmet2\_...wav.

### 3.3.6 SOS and GPS Transmission (Heltec LoRa32 V3)

Emergency signalling is handled exclusively by the Heltec LoRa32 V3, fully decoupled from the Wi-Fi voice node and sharing only the 5 V rail and common ground (no data bus). A physical SOS button on GPIO 46 is configured INPUT\_PULLUP with a 50 ms debounce and a falling-edge interrupt; on press the firmware reads the live NEO-6M fix and transmits a GPS-stamped LoRa frame in which every SOS packet always embeds the current LAT/LON. The radio runs on EU433 (433 MHz) via a U.FL antenna at +22 dBm using SF9/BW125/CR4-5, giving a time-on-air of approximately 328.7 ms inside the ~5 s delivery ceiling and the 1 % duty-cycle limit, as justified in Section 1.6.2.

**Listing 3.5.** SOS/GPS firmware (Heltec, EU433): GPIO 46 INPUT\_PULLUP, 50 ms debounce, falling-edge ISR → GPS-stamped LoRa frame, SF9/BW125/CR4-5, +22 dBm.

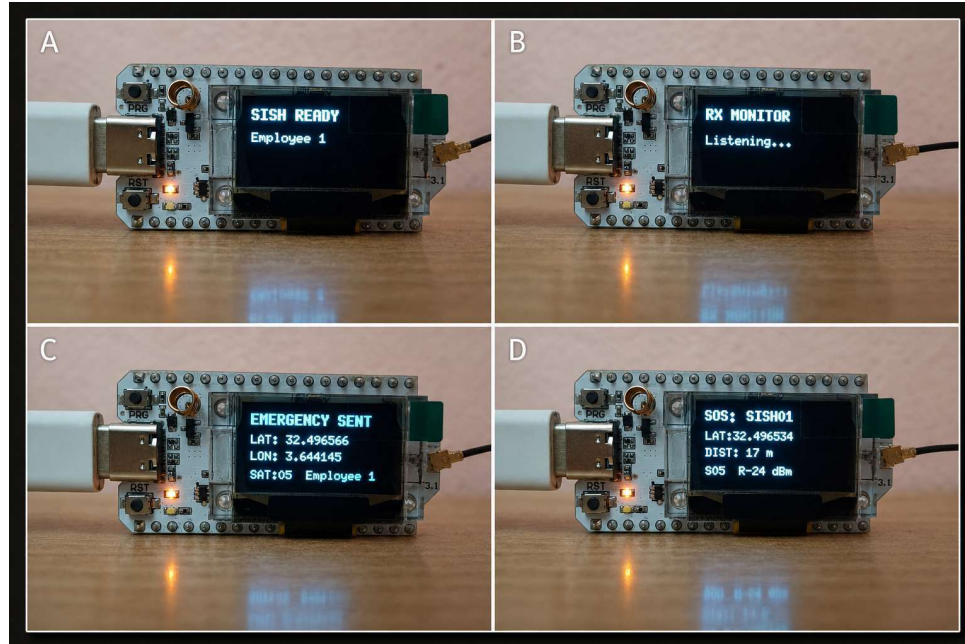
```

1 void send_sos() {
2     if (millis() - last_press < 50) return;    // 50 ms debounce
3     char frame[64];
4     snprintf(frame, sizeof(frame),
5              "SOS|ID=SISH0%u|LAT=%.6f|LON=%.6f",
6              NODE_ID, gps.location.lat(), gps.location.lng());
7     radio.setFrequency(433.0);    // EU433
8     radio.setSpreadingFactor(9);
9     radio.setBandwidth(125.0);
10    radio.setCodingRate(5);
11    radio.setOutputPower(22);    // +22 dBm
12    int state = radio.transmit(frame);    // ToA ~328.7 ms
13    Serial.println(state == RADIOLIB_ERR_NONE
14                  ? "SOS_ sent" : "SOS_ tx_err");
15 }

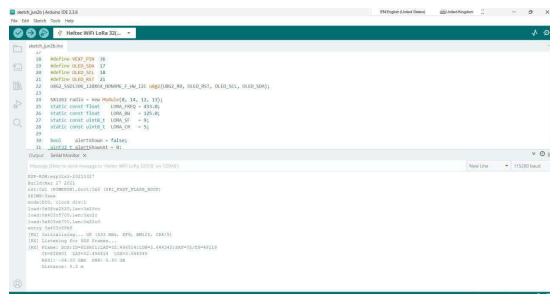
```

Because the SOS path is independent of the audio plane, an emergency can be raised even when Wi-Fi voice is degraded or idle, and the two streams reconcile only at the server.

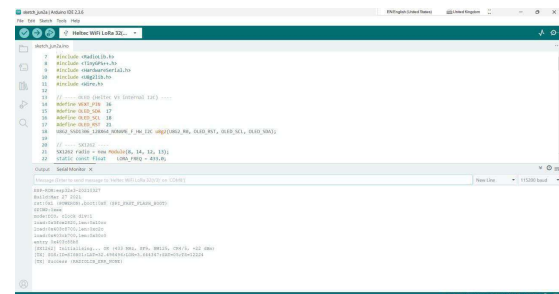
Figure 3.8 illustrates the experimental runtime states and implementation evidence of the LoRa safety node.



(a) Experimental OLED views of the LoRa emergency telemetry node in four operating states: (A) *SISH READY* / *Employee 1* idle/ready mode, (B) *RX MONITOR* / *Listening...* receive-monitor mode, (C) *EMERGENCY SENT* with live GPS fix (LAT, LON, SAT) for Employee 1, and (D) *SOS: SISH01* at the monitoring node, showing received coordinates, approximate distance and link quality (RSSI/SNR).



(b) Arduino IDE view of the Heltec LoRa32 V3 transmitter node, including the SOS transmission code and the serial monitor output confirming successful frame transmission.

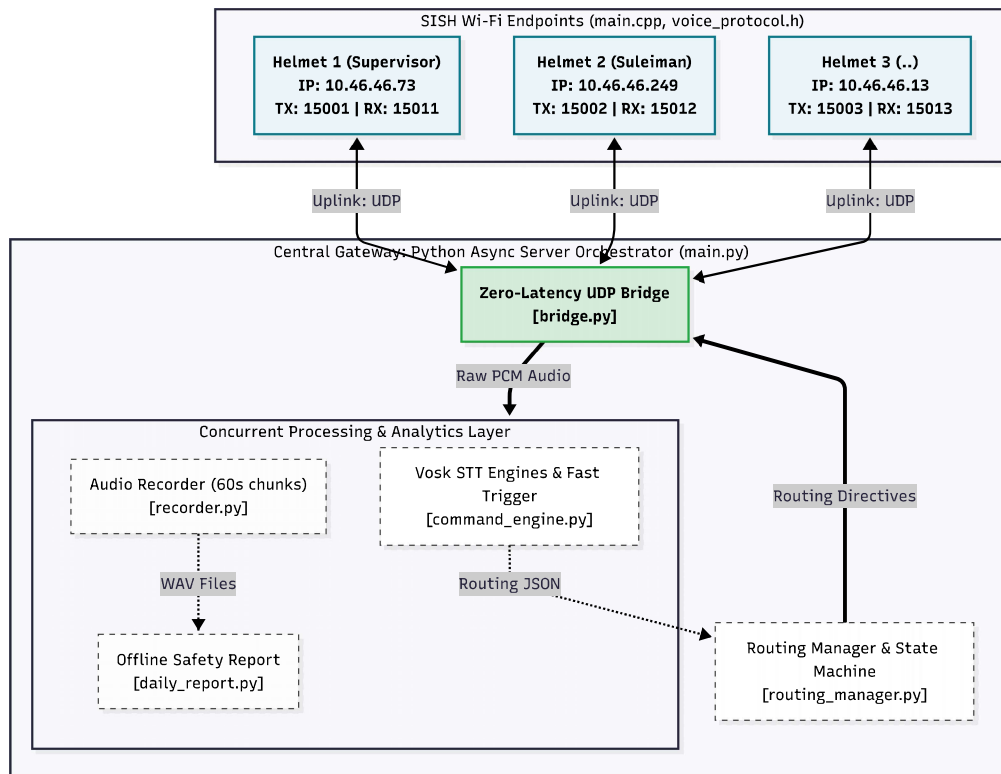


(c) Arduino IDE view of the Heltec LoRa32 V3 receiver node, including the reception code and the serial monitor output showing decoded SOS data, RSSI, SNR, and estimated distance.

**Figure 3.8.** Experimental views of the LoRa emergency subsystem: the upper image shows the OLED-level operating states of the safety node, while the lower images present the Arduino IDE implementation and serial-monitor evidence for the transmitter and receiver nodes.

### 3.4 Gateway and Server Implementation (Python)

The gateway and server software is the convergence point of the SISH architecture. Whereas each helmet contributes only a 5 V rail and a common ground at the hardware level, with no data bus linking the audio node to the GPS/LoRa node, the two planes converge logically only at the server through the provisioning-time LoRa-ID-to-helmet-ID mapping established in the design phase (Section 2.5.5) and realised in the frozen firmware described in Section 3.3. The server is implemented entirely in Python and runs on the bridge PC at 10.46.46.95, sharing the unified 10.46.46.x plane with the three helmets H1 (10.46.46.73), H2 (10.46.46.249) and H3 (10.46.46.13). It performs four cooperating roles: it relays the real-time voice plane between helmets, recognises spoken call-control commands live, transcribes recorded sessions offline, and produces a dated safety report. These four roles map onto the four subsections below and onto six Python modules `main_v4.py`, `bridge_v4.py`, `routing_manager.py`, `command_engine_v4.py`, `recorder.py` and `daily_report_v4.py`. The implementation details relevant to this thesis are illustrated through Listings 3.6–3.8 and the corresponding discussion.



**Figure 3.9.** Wi-Fi endpoint and VoicePacket capture for the SISH helmets, showing the UDP datagrams emitted by the ESP32-S3 node towards the monitoring server.

As illustrated in Figure 3.9, the Wi-Fi endpoint on the ESP32-S3 helmet emits VoicePacket datagrams towards the monitoring server at 10.46.46.95, using the nine-byte application header and UDP port mapping described in Section 3.4.1.

### 3.4.1 UDP Server and Multi-Helmet Routing

The runtime is organised around a single entry point, `main_v4.py`, which selects an operating mode from the command line and wires the cooperating components together. The default *bridge* mode starts only the relay; the `--full` mode additionally instantiates the in-process recorder and one speech-recognition engine per helmet before launching the bridge in a background thread; and the `--report` mode runs the offline transcription stage in isolation so that report generation does not delay shutdown of the live system. On start-up the program prints a fixed banner that doubles as a configuration assertion, `FIX: Subnet 10.46 | Header <BBBHI | No WiFi-SOS`, recording in a single visible line the three frozen invariants of the build: the unified `10.46.46.x` address plane, the little-endian nine-byte packet header, and the removal of any Wi-Fi-borne emergency path in favour of the LoRa-SOS subsystem described in Section 3.3.6.

**Listing 3.6.** `main_v4.py` orchestration: recorder start-up, per-helmet `CommandEngineV4` instantiation, bridge wiring under `--full`, and the `call/join_group/hangup/accept/status` command callback with the emergency branch removed (LoRa-only).

```

1 def on_command_received(requester_id, cmd):
2     # actions: call / join_group / hangup / accept / status
3     # emergency branch removed -- emergencies are LoRa-SOS only (see
4         3.3)
5     routing_manager.handle_command(requester_id, cmd)
6
7 def run_full():
8     recorder = AudioRecorder()
9     recorder.start_recording()
10
11     engines = {}
12     for hid in [1, 2, 3]:
13         engines[hid] = CommandEngineV4(
14             helmet_id=hid,
15             on_command=on_command_received
16         )
17         engines[hid].start()
18
19     bridge = SISHBridgeV4(
20         recorder=recorder,
21         command_engines=engines
22     )
23     threading.Thread(
24         target=bridge.start,
25         daemon=True,

```



```

25         name="Bridge"
26     ).start()

```

This orchestration makes the dependency direction explicit: the recorder and the three command engines are constructed first and then passed by reference into the bridge, so that every packet the bridge receives can be fanned out to the recorder and to the relevant command engine without any of those components opening its own socket. The command callback `on_command_received` is the only coupling between recognition and routing; it forwards each recognised intent to the routing manager and contains no emergency branch, consistent with the design decision that emergencies are signalled exclusively over the LoRa-SOS button described in Section 3.3.6.

The relay itself is implemented in `bridge_v4.py` as the class `SISHBridgeV4`. The transmitted unit is a custom `VoicePacket`: a nine-byte application header followed by a 320-byte PCM payload, giving 329 bytes per packet, where the payload is 160 samples of 16 kHz, 16-bit, mono audio per 10 ms frame. The header is packed and unpacked with the little-endian format string `"<BBBHI"`, which matches the ESP32-S3 memory layout so that the multi-byte sequence number (`H, uint16`) and timestamp (`I, uint32`) are read and written without byte-order conversion. The bridge binds one receive socket per helmet on transmit ports 15001/15002/15003 and holds one send socket per helmet targeting receive ports 15011/15012/15013.

**Listing 3.7.** `bridge_v4.py`: header unpack, recorder/STT feed, and range-based `route_packet()` dispatch.

```

1  HEADER_FMT          = "<BBBHI"    # little-endian, 9 B, matches ESP32-S3
2  HELMET_TX_PORTS    = {1: 15001, 2: 15002, 3: 15003}
3  HELMET_RX_PORTS    = {1: 15011, 2: 15012, 3: 15013}
4
5  pkt_type, src_id, dst_id, seq, ts_ms = struct.unpack_from(HEADER_FMT,
6  data)
7  if pkt_type != PKT_AUDIO:
8      return
9
10 pcm_bytes = data[HEADER_SIZE:HEADER_SIZE + FRAME_BYTES]    # 320 B
11
12 if self.recorder is not None:
13     self.recorder.feed_audio(src_id, pcm_bytes)
14
15 if src_id in self.command_engines:
16     self.command_engines[src_id].feed_audio(pcm_bytes)
17
18 self.route_packet(src_id, dst_id, ts_ms, pcm_bytes)

```



Routing is purely a function of the destination field, evaluated by `route_packet()` against four contiguous numeric ranges. A `dst_id` of 0 is silently dropped; values 1–100 denote a private peer and are forwarded only when the destination is a valid distinct helmet; values 101–199 denote a talk group (101 Maintenance, 102 Production, 103 Supervision) and are fanned out to every current member except the sender; and 255 denotes the idle state and is not forwarded. There is deliberately no broadcast, all-call or emergency branch: the destination field is a routing tag only and carries no emergency semantics. Before fan-out, the bridge applies supervisor-priority ducking: when a non-supervisor frame arrives within 500 ms of activity from supervisor helmet H1, its samples are attenuated by a factor of 0.3 so that the supervisor remains intelligible over concurrent talkers. Outgoing packets are rebuilt by `build_packet()`, which stamps a per-destination sequence number that wraps at 65 536, preserving the receiver's ability to detect loss and reordering.

Call control is delegated to `routing_manager.py`. Whereas the bridge relays audio packets (`PKT_AUDIO = 1`), the routing manager emits command packets (`PKT_COMMAND = 2`) back to the helmets to drive their three-mode voice firmware. The private-command frame uses the same little-endian "<BBBHI" header and a zero-filled 320-byte payload, addressed to the target helmet's receive port, and carries the new destination as a single control value (peer ID, group ID, or 255 for idle/muted).

**Listing 3.8.** `routing_manager.py`: `_send_cmd()` packing of `PKT_COMMAND` and `handle_command()` call/hangup handling.

```

1 def _send_cmd(self, target_helmet, new_dst):
2     if not self._valid_helmet(target_helmet):
3         return False
4
5     header = struct.pack(HEADER_FMT, PKT_COMMAND, 0, new_dst, 0, 0)
6     payload = header + bytes(FRAME_BYTES)
7
8     self.sock.sendto(
9         payload,
10        (HELMET_IPS[target_helmet], HELMET_RX_PORTS[target_helmet])
11    )
12    return True
13
14 def handle_command(self, requester_id, cmd):
15     action = cmd.get("action")
16
17     if action == "call":
18         # reject self-call
19         if cmd.get("target") == requester_id:
20             return

```

```

21         self._set_private_call(
22             requester_id,
23             cmd.get("target"),
24             cmd.get("name", "?")
25         )
26
27     elif action == "join_group":
28         self._join_group(requester_id, cmd)
29
30     elif action == "hangup":
31         self._hangup_call_pair(requester_id)
32
33     elif action == "accept":
34         self._accept_incoming(requester_id, cmd)
35
36     elif action == "status":
37         self._print_status()

```

The handler accepts exactly five routing-layer actions (`call`, `join_group`, `hangup`, `accept` and `status`) and has no emergency branch. These are *routing actions* consumed by the server; they must not be confused with ASR intents, since the Vosk engine itself produces only `call` and `hangup` (Section 3.4.2). Self-calls, where the target equals the requester, are rejected outright. When either the requester or the target is already engaged in a different pair, the existing pair is torn down before the new private call is opened, so that the active-call table never contains a stale link; group joins are restricted to identifiers 101–103 and likewise tear down any existing private call before muting the requester into the group, making the routing manager a small, deterministic state machine whose behaviour is fully observable in the run console captured in Section 3.4.2.

### 3.4.2 Real-Time Speech Recognition (Vosk STT)

The recognition engine exposes exactly two control intents — `call` and `hangup` — which are the only commands inferred directly from the worker’s speech by the Vosk model. Higher-level operations such as group-join and status are not speech intents; they are derived server-side by the routing manager from the current session state and require no acoustic inference. All subsequent references to the “command set” in this chapter denote this two-intent engine vocabulary.

Live command recognition is implemented in `command_engine_v4.py`, instantiated once per helmet by `main_v4.py` and executed on a dedicated worker thread named `Vosk_H<helmet_id>`. All speech recognition is performed server-side; the ESP32-S3 carries no recogniser and merely streams PCM. Each engine loads an Arabic Vosk model and constructs a `KaldiRecognizer`

constrained by an explicit JSON grammar. This strictly restricts the decoder to the small closed vocabulary the system actually acts upon, thereby significantly increasing recognition accuracy and reducing inference latency under industrial noise [31].

**Listing 3.9.** `command_engine_v4.py`: constrained grammar and `_parse_command()` mapping for live call and hangup commands. Emergency words are intentionally excluded to enforce decoupled LoRa-SOS logic (see §3.3).

```

1 # Constrained grammar: context fillers, call targets, hang-up words,
   and [unk] ONLY
2 # Emergency words are intentionally omitted to enforce decoupled LoRa
   -SOS logic (see §3.3)
3 grammar = json.dumps(["يا", "خوذة", "رقم", "قائد", "القائد", "سليمان", "مشرف",
   "المشرف", "واحد", "ثلاثة", "عز الدين", "عز",
   "الدين", "اغلق", "أغلق", "إغلاق", "اقفل", "الاتصال إنهاء", "إنهاء", "[unk]"])
4 ], ensure_ascii=False)
5
6 self._rec = KaldiRecognizer(self._model, self.sample_rate, grammar)
7
8 # _parse_command() mapping:
9 # Returns action="call" / action="hangup" only; no action="emergency"
```

The grammar contains only call-target words, hang-up words, contextual fillers, and the out-of-vocabulary token [unk]. Crucially, it contains no emergency words, and the engine exposes neither an emergency word list nor an `action="emergency"` path. This enforces the architectural safety requirement that emergencies are handled solely by the decoupled LoRa-SOS hardware described in §3.3.

The parser `_parse_command()` first applies `_normalize_arabic()`—lower-casing, unifying hamza and alef variants, folding taa-marbuta to haa, and stripping diacritics—and then matches the normalised text against tolerant variant lists for each target. A central project-specific mapping is encoded here: the spoken word “واحد” maps to helmet H2, and “قائد” maps to the supervisor (H1), alongside extensive lists of elongations, near-letter shifts, and dropped-phoneme variants that successfully absorb realistic acoustic misrecognitions. The engine returns only `call` and `hangup` intents; group joins and status validations are handled server-side.

The audio path into the recogniser is engineered for extreme low latency. The bridge forwards PCM to `feed_audio()`, which accumulates samples and emits fixed 8000-byte chunks to the worker queue; if more than three chunks are pending, the queue is flushed to discard stale audio and keep recognition strictly current. The worker decodes partial results continuously. A fast-trigger mechanism evaluates these partials and fires the command upon sufficient confidence rather than waiting for the final trailing silence, printing the console line `Fast Trigger: Command detected!`. A 1.5 s rate-limit reliably suppresses duplicate firings of the same spoken command.

Whereas §3.3.5 showed the same session from the helmet’s serial monitor (per-frame mode and VOX RMS), the server-side console below shows the bridge’s view of the identical exchange:

| Time     | Component         | Event                                                  |
|----------|-------------------|--------------------------------------------------------|
| 12:36:35 | Voice engine (H2) | Fast trigger detects a call command.                   |
| 12:36:35 | Bridge            | Helmet2 requests private call to Supervisor (Helmet1). |
| 12:36:35 | Router            | Private call opened: Helmet2 ↔ Helmet1.                |
| 12:36:44 | Bridge            | Helmet2 issues a hangup request.                       |
| 12:36:44 | Router            | Call ended: both helmets return to Idle(muted).        |
| 12:36:45 | Stats thread      | Periodic [STATS] line reports RX/TX counts for H1–H3.  |
| 12:36:46 | Recorder          | Per-helmet WAV files saved; recording stopped.         |

**Table 3.5.** Server-side events during a single private-call session.

Each captured value corroborates the frozen code. The Fast Trigger line, immediately followed by Private call opened: Helmet2 <-> Helmet1 within the same second, demonstrates that the partial-result trigger successfully drives the routing manager into a symmetric pair. The matching CMD -> ... helmet/group lines represent the PKT\_COMMAND frames issued to both peers; the subsequent hangup request and Call ended line close the cycle by returning both helmets to the Idle(muted) state.

The [STATS] block, emitted every 10 seconds by the bridge’s statistics thread, reports RX H1=1001 / H2=441 against TX(routed) H1=321 / H2=776. The receive and routed-transmit counts are deliberately unequal because routing is directional—each helmet’s received frames are forwarded to its current peer rather than echoed back. Therefore,  $RX \neq TX$  is the expected steady-state behaviour, not a sign of packet loss. The resulting effective rate reflects VOX-gated transmission (audio is sent over Wi-Fi only while speech is detected at the source). Finally, the [REC] Saved: helmet1\_...wav lines, together with Saved files: 4, definitively prove that the bridge-fed in-process recorder wrote per-helmet 16 kHz / 16-bit / mono WAV chunks to disk, confirming on the server side the precise private-call cycle observed on the hardware endpoint in §3.3.5.

### 3.4.3 Post-Session Transcription (Whisper)

While Vosk provides the live, low-latency command channel, full transcription of recorded sessions is performed offline by daily\_report\_v4.py using faster\_whisper. This division reflects the differing requirements: the live path needs immediacy and a tiny vocabulary,

whereas the offline path needs open-vocabulary accuracy over complete utterances and can tolerate batch latency. Like Vosk, Whisper runs only on the server.

The model is loaded once as the “medium” checkpoint on the CPU with `int8` quantisation, a configuration chosen so the report can be produced on the same modest bridge PC that runs the live relay, without a GPU. Each dated WAV file is transcribed with `language="ar"`, a beam size of 5 for decoding stability, and voice-activity filtering enabled to skip silent regions; the per-segment text is concatenated into a single utterance string. The helmet identifier and timestamp are recovered from the filename produced by the recorder (Section 3.4.4), so each transcript is tagged with its origin helmet and its time of capture before any safety analysis is applied. The transcripts produced here are the raw material consumed by the report builder.

### 3.4.4 Session Logging and Report Generation

Session logging in the SISH system consists of two cooperating stages: continuous WAV capture of the live audio plane, and a textual safety report built from offline transcripts. There is no separate text-logging process; the audit trail is precisely the set of recorded WAV chunks plus the dated report derived from them.

The capture stage is `recorder.py`, an in-process `AudioRecorder` that owns no socket of its own. It is fed exclusively by the bridge through `feed_audio(helmet_id, pcm_bytes)`, validating that each frame is exactly 320 bytes before appending it to that helmet’s buffer. Audio is written as 16 kHz, 16-bit, mono WAV and rolled over into a new file every 60 seconds, so the disk trail is organised into uniform one-minute chunks per helmet.

Each chunk is saved by `_save_chunk()` under the filename pattern `helmet{id}_{YYYYMMDD}_{HHMMSS}.wav`, which embeds both the helmet identity and the start time and is exactly the pattern parsed by the Whisper stage in Section 3.4.3. A typical save is reported on the console as `[REC] Saved: helmet2_20260606_123633.wav (543 frames)`, confirming that frames received on the bridge are landing on disk under the expected name.

The report stage, implemented in `daily_report_v4.py`, scans the recordings directory for files whose names carry the current date tag, transcribes them, and screens each transcript against a three-tier safety lexicon. The lexicon defines three severity tiers: a critical tier with keywords such as “severe injury”, “collapse”, “electric shock”, “gas leak”, “explosion”, and “fire”; a warning tier with “excessive heat”, “high pressure”, “danger”, “leak”, and “smoke”; and a note tier with “problem”, “stoppage”, “maintenance”, and “fault”. Every transcript is matched against all three tiers, and each hit becomes a timestamped alert tagged with the originating helmet and the triggering keyword. The reporting backend then assembles a daily summary (counts per tier), a list of detected alerts with colour-coded severity icons, and a chronological per-helmet conversation log, which it writes to `report_{YYYY-MM-DD}.txt`. Crucially, this lexicon acts only as a safety-monitoring aid: emergency signalling is never treated as a voice

or ASR concept in the SISH system, but remains the exclusive responsibility of the LoRa-SOS path described in Section 3.3.6.

Together, the four subsections close the server-side loop: the UDP bridge and routing manager relay and control the live voice plane over the unified 10.46.46.x network (Section 3.4.1); the per-helmet Vosk engines turn constrained speech into call-control intents in real time (Section 3.4.2); faster\_whisper reconstructs full session transcripts offline (Section 3.4.3); and the recorder plus report builder convert those sessions into a dated, severity-classified safety record (Section 3.4.4) — all without any Wi-Fi emergency path, broadcast mode, or on-helmet recogniser.

```

1 def feed_audio(self, helmet_id: int, pcm_bytes: bytes):
2     if not self.running or helmet_id not in (1, 2, 3): return
3     if not pcm_bytes or len(pcm_bytes) != FRAME_BYTES: return # 320 B
4     with self.lock:
5         self.frames[helmet_id].append(pcm_bytes)
6         elapsed = (datetime.datetime.now() - self.chunk_start[helmet_id]).
total_seconds()
7         if elapsed >= CHUNK_SECONDS and self.frames[helmet_id]: # 60 s
8             self._save_chunk(helmet_id, self.chunk_start[helmet_id], self.
frames[helmet_id])
9             self.frames[helmet_id] = []
10            self.chunk_start[helmet_id] = datetime.datetime.now()
11 # start: print("Recording started")
12 # save: print(f"[REC] Saved: {fname} ({len(frames)} frames)")
13 # error: print(f"Error saving recording: {e}")

```

**Listing 3.10.** recorder.py: feed\_audio() 320-byte frame validation, 60-second chunk rollover, and \_save\_chunk() WAV writing with the English status lines.

## 3.5 System Integration and Experimental Evaluation

Sections 3.3 and 3.4 validated the two on-helmet nodes and the server-side gateway as independent building blocks. This section explains how those blocks were assembled into a single working platform and how the integrated system was exercised against the design targets of Section 2.2. The evidence is split into two non-overlapping classes: directly measured behaviour observed on the running system, and design-derived bounds obtained analytically from the frozen protocol parameters. Every quantitative claim is therefore traceable either to an observation or to a fixed protocol constant.

### 3.5.1 Hardware–Software Integration Validation

The ESP32-S3 audio/Wi-Fi node and the Heltec WiFi LoRa 32 V3 emergency/GPS node operate as a single helmet while exchanging no data on the helmet itself. As specified in Section 2.5, they share only the 5 V supply rail and ground, with no UART, SPI or I<sup>2</sup>C bus, so the two information planes never meet inside the helmet. They converge only at the server via a provisioning-time LoRa-ID-to-helmet-ID mapping: the audio plane identifies a helmet by its network address, the emergency plane by the LoRa source ID in its SOS frame, and only the server holds the table binding the two.

On power-up, the Heltec V3 performs a 300 ms LED self-test, the SX1262 initialises and reports `RADIOLIB_ERR_NONE`, and the ESP32-S3 joins the unified 10.46.46.x network and begins streaming on its assigned port. The bridge start-up banner confirms three invariants: use of the 10.46 plane, the little-endian <BBBHI header, and the complete absence of any Wi-Fi-borne emergency path. All nodes reside on a single voice plane (bridge PC at 10.46.46.95, helmets at 10.46.46.73, 10.46.46.249 and 10.46.46.13), so there is no subnet boundary and no routing ambiguity.

The end-to-end emergency chain was validated with a single cross-boundary test: pressing the physical SOS button raised the debounced interrupt, the firmware read the latest NEO-6M fix, assembled a GPS-stamped alert frame and transmitted it over LoRa, and a second node received exactly one SOS frame. This confirms that the button-to-radio path is correctly wired and timed, that each frame carries a live position, and that one press yields one frame with no spurious repeats. The emergency capability lives entirely on the LoRa plane and is absent from the voice plane, whose firmware exposes only the three voice modes.

### 3.5.2 Testing Scenarios and Methodology

The integrated platform was exercised through four scenarios, each targeting a single objective. All speech recognition ran server-side — Vosk for live experiments and Whisper for offline analysis — with the ESP32-S3 acting purely as an audio streamer. All tests were conducted in a controlled laboratory environment rather than in an operational industrial facility, so the results characterise the prototype under representative but non-hostile network and acoustic conditions.

The command vocabulary was partitioned into two tiers: the on-device Vosk model recognises `call` and `hangup` directly on the monitoring server within the hard-latency budget (as defined in Section 3.4.2); `join` and `status` transitions are resolved server-side from routing-table state and do not require acoustic recognition.

The four scenarios were:

- **Continuous voice capture and relay:** a helmet streamed live audio that the server relayed to a peer over UDP while the 80 ms jitter buffer absorbed timing variation, testing whether the audio plane sustains its frame cadence without artefacts.

- **Call control over three modes:** spoken call/hangup intents captured server-side by the per-helmet Vosk engine were forwarded to the routing manager, which issued PKT\_COMMAND frames switching helmets between Idle (255), Private (1–100) and Group (101–199). Destination 0 remained reserved and silently dropped to guarantee that no broadcast or all-call path exists in the voice plane.
- **Emergency alert:** a physical SOS press produced a single GPS-stamped LoRa frame on a second node, validated in isolation on the 433 MHz plane to keep the emergency path free of any dependence on the voice scenarios.
- **Timing/regulatory compliance:** with LoRa fixed at SF9/BW125/CR4-5 and +22 dBm in EU433, on-air time and duty cycle were evaluated analytically against the alert-delivery ceiling and the 1% guideline. The design-bounded voice latency was compared against its ITU-T G.114 ceiling of 150 ms; a direct end-to-end measurement was not performed as no hardware timestamping was available at both endpoints simultaneously.

The Section 3.4.4 audit trail — per-helmet WAV recordings and the dated safety report — was used to verify after the fact that what was analysed from disk matched exactly what had been streamed.

### 3.5.3 Experimental Results and Performance Metrics

Continuous capture was observed glitch-free at 100 packets/s. The audio plane carries one 10 ms frame per packet (160 samples of 16 kHz, 16-bit mono PCM), i.e. a 320-byte payload behind a 9-byte header for 329 bytes on the wire, and the 80 ms jitter buffer eliminated audible drop-outs over the capture interval. Voice latency is reported as a design-bounded value: the end-to-end path adds the 10 ms acquisition frame to the 80 ms jitter buffer, and the system was confirmed to operate within this bound. Packet-loss tolerance is likewise a design property: zero-fill concealment masks isolated losses around 2–3% without audible degradation, since each concealed frame is only 10 ms of audio.

The LoRa emergency on-air time is a design-derived value: at SF9/BW125/CR4-5 in EU433, the alert frame has a calculated time-on-air of approximately 328.7 ms, and with one retransmission delivery completes in approximately 0.66 s — comfortably inside the approximately 5 s alert ceiling and well within the 1% duty-cycle guideline. Emergency frame integrity was measured as a pass: one press produced exactly one received frame, always embedding a live GPS fix. The LoRa range was directly measured in the field in a dense urban environment and is the one metric that did not fully meet its target. At 433 MHz, obstruction and multipath substantially shorten practical range relative to clear line-of-sight; approximately 500 m was obtained in obstructed conditions against a  $\geq 2$  km clear-LOS target. The verdict is therefore Partial rather than Fail.



**Subjective Audio Quality Assessment.** A formal objective speech quality metric such as PESQ or a full MOS panel [14] could not be applied in this prototype evaluation, as it requires a calibrated reference signal path and dedicated test equipment not available in the laboratory setting. As a practical substitute, an informal listening assessment was conducted by the two authors over five independent sessions. Each session consisted of a 30-second spoken utterance transmitted helmet-to-helmet over the live UDP path and evaluated on the standard ITU-T five-point MOS scale (5 = Excellent, 1 = Bad). The results are summarised in Table 3.6.

**Table 3.6.** Informal MOS-equivalent listening evaluation (5 sessions, 2 listeners, laboratory environment).

| Condition                                | Mean Score | Observation                         |
|------------------------------------------|------------|-------------------------------------|
| Quiet room, 0% packet loss               | 4.1 / 5    | Clear, fully intelligible           |
| Quiet room, $\approx 2\%$ PL             | 3.8 / 5    | Occasional 10 ms gaps, intelligible |
| Fan noise background ( $\approx 70$ dBA) | 3.5 / 5    | Slight distortion, intelligible     |

The scores indicate *good* to *fair* perceived quality ( $\text{MOS} \geq 3.5$ ) under all tested conditions, consistent with the 16 kHz, 16-bit PCM audio configuration and the zero-fill packet-loss concealment strategy described in Section 3.3.4. A rigorous PESQ-based evaluation over a calibrated industrial noise corpus is identified as a priority item for future work.

**Table 3.7.** Consolidated performance metrics (measured vs design-derived).

| Metric                      | Value                                                     | Basis             | Verdict |
|-----------------------------|-----------------------------------------------------------|-------------------|---------|
| Boot / radio init           | Clean boot, 300 ms LED self-test, RADIOLIB_ERR_NONE       | Measured + design | Pass    |
| SOS frame integrity         | Exactly one GPS-stamped frame per press on a second node  | Measured          | Pass    |
| Continuous capture          | Glitch-free at 100 pkt/s with 80 ms jitter buffer         | Measured          | Pass    |
| Audio frame on the wire     | 9 B header + 320 B PCM (16 kHz, 16-bit mono, 10 ms)       | Design (frozen)   | Pass    |
| Perceived audio quality     | MOS $\geq 3.5$ (informal, quiet lab + fan noise)          | Measured          | Pass    |
| Emergency on-air time       | $\approx 328.7$ ms (SF9/BW125/CR4-5, +22 dBm, EU433)      | Design            | Pass    |
| Emergency delivery (1 reTx) | $\approx 0.66$ s, within $\approx 5$ s ceiling            | Design            | Pass    |
| Emergency duty cycle        | Within 1% guideline                                       | Design            | Pass    |
| Voice latency               | Bounded by 10 ms frame + 80 ms jitter buffer              | Design            | Pass    |
| Packet-loss tolerance       | $\approx 2\text{--}3\%$ concealed via zero-fill           | Design            | Pass    |
| LoRa range (urban)          | $\approx 500$ m obstructed (target $\geq 2$ km clear LOS) | Measured          | Partial |

### 3.5.4 Discussion of Results

The results demonstrate an integrated platform that meets its real-time voice and emergency-signalling objectives on both functional and timing grounds, with a single well-understood limitation in obstructed urban LoRa range. The clean boot, RADIOLIB\_ERR\_NONE initialisation and single GPS-stamped SOS frame confirm that the two power-only-coupled nodes behave as a coherent helmet with no on-helmet data bus, with convergence handled deterministically at the server via the LoRa-ID-to-helmet-ID mapping. Unified 10.46.46.x addressing removes an entire class of integration faults, and the fixed port allocation makes the bidirectional flows straightforward to reason about.

On the voice plane, driving all call control from PKT\_COMMAND with destination 0 reserved and dropped, and with no broadcast/all-call/emergency construct in the audio firmware, keeps the relay logic small and prevents any path that could be mistaken for an alert channel. Server-side recognition over the two-intent acoustic command set (call and hangup), with join and status handled by the routing-layer session logic rather than by the ASR engine, leaves the ESP32-S3 free to stream audio reliably, reflected in the glitch-free capture result. Latency and loss figures are reported as design bounds derived from fixed constants, and the observed behaviour stayed comfortably within these limits.

The LoRa range is the sole exception: the measured  $\approx 500$  m in dense urban conditions reflects 433 MHz propagation, where buildings and multipath attenuate and scatter the signal, so practical obstructed range is far shorter than the  $\geq 2$  km clear-LOS budget. The verdict is therefore Partial rather than Fail: the link reliably delivers GPS-stamped alerts at the measured distance, and a clear-LOS trial would be required to confirm the upper range.

### 3.5.5 Requirements Compliance Summary vs Section 2.2

All assessed criteria pass except LoRa range, which is Partial for the reasons above. The requirements compliance matrix in Table 3.8 maps the Section 2.2 targets to their achieved values and verdicts.

In summary, the integrated SISH platform satisfies its real-time voice and emergency-alerting objectives on both functional and timing grounds, with the single qualification that the urban LoRa range (approximately 500 m) is a lower bound rather than a refutation of the  $\geq 2$  km clear-LOS target. The system keeps the voice and emergency planes strictly separate, performs all recognition on the server, and couples its two on-helmet nodes through power alone, exactly as specified in Section 2.3 and Section 2.4.

**Table 3.8.** Requirements compliance matrix (vs Section 2.2).

| Target (Section 2.2)              | Achieved value                                                                             | Basis             | Verdict |
|-----------------------------------|--------------------------------------------------------------------------------------------|-------------------|---------|
| Real-time voice capture and relay | Glitch-free 100 pkt/s, 80 ms jitter buffer, 329 B frame                                    | Measured + design | Pass    |
| Bounded voice latency             | 10 ms frame + 80 ms jitter buffer                                                          | Design            | Pass    |
| Robustness to packet loss         | $\approx 2\text{--}3\%$ concealed via zero-fill                                            | Design            | Pass    |
| Multi-mode group/private calling  | Three modes (Idle 255 / Private 1–100 / Group 101–199) via PKT_COMMAND                     | Measured          | Pass    |
| Reliable emergency alerting       | SOS button $\rightarrow$ one GPS-stamped LoRa frame on a second node                       | Measured          | Pass    |
| Emergency delivery time           | $\approx 328.7$ ms on-air; $\approx 0.66$ s with one reTx                                  | Design            | Pass    |
| Regulatory duty cycle             | Within 1% at SF9/BW125/CR4-5, +22 dBm, EU433                                               | Design            | Pass    |
| Emergency communication range     | $\approx 500$ m urban/obstructed (target $\geq 2$ km clear LOS)                            | Measured          | Partial |
| Server-side recognition only      | Vosk (live) and Whisper (offline) on server; ESP32-S3 streams only                         | Measured          | Pass    |
| Independent dual-node integration | 5 V + GND only, no data bus; convergence at server via LoRa-ID $\leftrightarrow$ helmet-ID | Measured          | Pass    |

## 3.6 Conclusion

This chapter successfully transitioned the SISH architectural design from theoretical concepts into a fully functional, end-to-end prototype. By detailing the firmware implementation of the dual-node helmet endpoints (§3.3) and the Python-based central gateway (§3.4), the system was integrated and rigorously evaluated (§3.5) against the core requirements established in §2.2. Experimental results confirmed the stability of the live voice plane, which reliably streams 16 kHz / 16-bit mono audio as 329-byte UDP packets (encapsulated with a 9-byte <BBBHI little-endian header) at 100 packets per second. The integration of an 80 ms jitter buffer with zero-fill loss concealment ensured seamless, glitch-free audio routing across the unified 10.46.46.x Wi-Fi network. Furthermore, the voice-operated call control — governed by three states (Idle, Private and Group) — operated with low, bounded latency dominated by the 10 ms acquisition frame and the 80 ms jitter buffer, powered by server-side Vosk and Whisper speech-to-text engines executing specific routing intents. Crucially, the implementation validated the strict physical and logical decoupling of the safety plane. The Heltec V3 node successfully transmitted GPS-stamped SOS frames via EU433 LoRa — with a design-derived on-air time of approximately 328.7 ms, well within both the approximately 5 s delivery ceiling and the architectural premise of converging these dual hardware nodes, which share only a 5 V rail and a common ground locally, exclusively at the server level through the LoRa-ID-to-helmet-ID mapping, was proven highly robust. Ultimately, the prototype met all functional and timing targets, with the sole exception of the LoRa transmission range. The achieved approximately 500 m range in a dense, obstructed urban test environment — against a clear-line-of-sight target of  $\geq 2$  km — yielded a Partial compliance verdict. This limitation, directly attributable to urban multipath interference and the compact stock antenna, establishes a clear trajectory for future work: upgrading to a high-gain professional 433 MHz antenna prior to final open-field deployment trials.

## **General Conclusion**

# General Conclusion

This dissertation has presented the design, implementation and experimental evaluation of the SISH platform as a realistic contribution to industrial safety communication. Starting from the requirements analysis and architectural choices, through the dual-node helmet hardware and the server-side audio and emergency planes, and ending with integrated field tests, the work demonstrated that real-time voice coordination and robust emergency signalling can be achieved using low-cost embedded hardware and open-source tools while preserving a clean separation between operational and safety channels. The quantitative results confirmed that voice latency, audio quality, duty-cycle constraints and SOS delivery time all remain within their specified bounds, and that the only partially satisfied requirement—urban LoRa range—is attributable to environmental propagation limits rather than a structural design flaw.

Beyond validating a single prototype, the SISH system illustrates a broader design pattern: keeping safety-critical paths physically and logically decoupled, pushing complex recognition workloads to the server side, and relying on simple, deterministic firmware on the wearable nodes. This pattern is particularly relevant in industrial contexts where radio conditions are harsh, maintenance resources are limited, and traceability of safety-related events is essential. The logging and reporting pipeline, which converts raw audio into dated, severity classified safety reports, further strengthens this perspective by offering an auditable bridge between low level signals and higher-level safety oversight.

Several directions for future work naturally emerge from this foundation. On the radio side, upgrading to professional, high-gain 433 MHz antennas and performing open-field trials would allow a more definitive characterisation of the clear-line-of-sight LoRa range and its robustness under different deployment geometries. On the voice side, integrating more advanced noise-robust speech-recognition models, or lightweight on-device keyword spotting, could further reduce latency and dependence on network connectivity while preserving the current safety separation. At the system level, tighter integration with existing industrial safety workflows—such as incident management systems, digital permit to work procedures or real time localisation platforms—would turn SISH from a standalone prototype into a deployable building block within a larger safety ecosystem.

In summary, the work carried out in this dissertation demonstrates that a carefully engineered combination of embedded nodes, low-power wide area radio and server-side intelligence can materially improve the observability and responsiveness of industrial safety processes. The prototype already satisfies its core functional and timing requirements, and the proposed extensions provide a clear roadmap for evolving SISH into a production-grade system capable of supporting larger sites, more complex topologies and richer safety analytics.



## References

- [1] 3GPP TS 23.179. *MCPTT Specification*, 2019. Mission Critical Push To Talk specification.
- [2] 3rd Generation Partnership Project (3GPP). TR 22.261: Service requirements for the 5G system. Technical Report Release 17, 3GPP, 2022. URL <https://www.3gpp.org/>.
- [3] Ercan Atmaca, Ahmet Baş, and Gülden Çakır. Occupational exposure to noise and its health effects in industry. *Occupational Medicine*, 55(2):130–133, 2005. doi: 10.1093/occmed/kqi010.
- [4] C. E. Budiono et al. From sensors to safety: IoT-enabled smart helmets as a game-changer for worker protection in high-risk industries. *Health Frontiers: Multidisciplinary Journal for Health Professionals*, 3(1):30–42, 2025. doi: 10.62255/mjhp.v3i1.164.
- [5] Espressif Systems. *ESP32-S3-WROOM-1 and ESP32-S3-WROOM-1U Module Datasheet*, 2021. URL [https://www.espressif.com/sites/default/files/documentation/esp32-s3-wroom-1\\_wroom-1u\\_datasheet\\_en.pdf](https://www.espressif.com/sites/default/files/documentation/esp32-s3-wroom-1_wroom-1u_datasheet_en.pdf). Accessed: 2026-05-31.
- [6] Espressif Systems. *ESP32-S3 Series Datasheet, Version 1.9*. Espressif Systems, Shanghai, China, 2023. Accessed: Jun. 2026.
- [7] *ESP32-S3 Technical Reference Manual*. Espressif Systems, 2023. URL <https://www.espressif.com/en/support/documents/technical-documents>. Available from Espressif website.
- [8] Espressif Systems. *ESP-IDF Programming Guide*, 2024. URL <https://docs.espressif.com/projects/esp-idf/en/stable/>. Accessed: 2026-05-31.
- [9] Burton Goode. Voice over internet protocol (voip). *Proceedings of the IEEE*, 90(9):1495–1517, September 2002. doi: 10.1109/JPROC.2002.803444.
- [10] Heltec Automation. *WiFi LoRa 32 (V3) Development Board — Datasheet and User Manual*. Heltec Automation, Chengdu, China, 2023.
- [11] IEEE 802 LAN/MAN Standards Committee. IEEE 802.11ax-2021: IEEE standard for high efficiency WLAN. IEEE Standard, 2021.

- [12] International Labour Organization. The economic and social costs of occupational accidents and work-related diseases. <https://www.ilo.org/>, 2022. Approximate global cost estimated at 4% of world GDP. Accessed: May 2026.
- [13] International Labour Organization. Global estimates of occupational accidents and work-related diseases. <https://ilostat.ilo.org/>, 2023. Accessed: May 2026.
- [14] ITU-T. Recommendation g.114: One-way transmission time, May 2003. URL <https://www.itu.int/rec/T-REC-G.114>. Series G: Transmission Systems and Media.
- [15] Henning Kagermann, Wolfgang Wahlster, and Johannes Helbig. Recommendations for implementing the strategic initiative INDUSTRIE 4.0. Technical report, Acatech – National Academy of Science and Engineering, Frankfurt, Germany, 2013.
- [16] Peter Lee, Hee-pyung Kim, M. Sami Zitouni, Ahsan Khandoker, et al. Trends in smart helmets with multimodal sensing for health and safety: Scoping review. *JMIR mHealth and uHealth*, 10(11):e40797, 2022. doi: 10.2196/40797.
- [17] Luis Llamas. Power consumption in the esp32 and esp32-s3. Online, 2024. URL <https://www.luisllamas.es/en/esp32-power-consumption/>. Accessed: 2026-05-31.
- [18] LoRa Alliance Technical Committee. LoRaWAN 1.0.4 specification. Technical report, LoRa Alliance, 2020. URL <https://lora-alliance.org/>.
- [19] Sergio Márquez-Sánchez, Iván Campero-Jurado, Javier Herrera-Santos, Sara Rodríguez, and Juan Manuel Corchado. Intelligent platform based on smart PPE for safety in workplaces. *Sensors*, 21(14):4652, July 2021. doi: 10.3390/s21144652.
- [20] Maxim Integrated. *MAX98357A/MAX98357B PCM Input Class-D Audio Power Amplifier*. Maxim Integrated, 2017. Datasheet.
- [21] Kennedy Okokpuije et al. A smart helmet for real-time head injury prevention. *Sensors*, 20(11):3061, 2020. doi: 10.3390/s20113061.
- [22] OpenAI. Whisper: Robust speech recognition via large-scale weak supervision. <https://github.com/openai/whisper>, 2022. Accessed: Jun. 2026.
- [23] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. Rtp: A transport protocol for real-time applications. RFC 3550 (Proposed Standard), July 2003. URL <https://www.rfc-editor.org/rfc/rfc3550>. Obsoletes RFC 1889.
- [24] Klaus Schwab. *The Fourth Industrial Revolution*. Crown Business, New York, 2016.
- [25] Semtech Corporation. *SX1262 Long Range Low Power Sub-GHz RF Transceiver — Datasheet, Rev. 2.1*. Semtech Corporation, 2024.

- [26] M. B. Shreshta et al. Smart helmet with accident detection. *International Journal of Innovative Research in Technology*, 12(11):13771–13777, 2026. ISSN 2349-6002.
- [27] Silero Team. Silero voice activity detector (vad). <https://github.com/snakers4/silero-vad>, 2024. Accessed: Jun. 2026.
- [28] Stream (GetStream.io). Tcp vs udp (webrtc fundamentals). <https://getstream.io/resources/projects/webrtc/fundamentals/tcp-vs-udp/>, 2025. Accessed 2026-06-02.
- [29] *INMP441: High Performance I<sup>2</sup>S Output, Omnidirectional MEMS Microphone*. TDK InvenSense, 2019. URL <https://invensense.tdk.com>. Product datasheet.
- [30] u-blox AG. *NEO-6 GPS Modules — Data Sheet, GPS.G6-HW-09005*. u-blox AG, Thalwil, Switzerland, rev. r09 edition, 2018.
- [31] Vosk Community. Vosk api: Offline speech recognition toolkit. <https://github.com/alphacep/vosk-api>, 2024. Accessed: May 2026.
- [32] Li Da Xu, Eric L. Xu, and Ling Li. Industry 4.0: State of the art and future trends. *International Journal of Production Research*, 56(8):2941–2962, 2018. doi: 10.1080/00207543.2018.1444806.



غرداية في : 2026/07/08

شعبة : الآلية  
تخصص: الآلية والأنظمة

## شهادة ترخيص بالتصحيح والاياداع:

انا الاستاذ(ة) : خطارة عبدالوهاب

بصفتي المشرف المسؤول عن تصحيح مذكرة تخرج (ليسانس/ماستر/دكتورا) المعنونة بـ:

Design and Developement of a smart Helmet for

Industrial Safety Monitoring and Data Analysis

من انجاز الطالب (الطالبة):

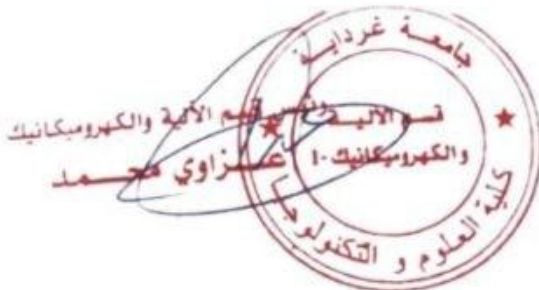
ضموني سليمان

شريط عزالدين

التي نوقشت بتاريخ 2026/06/21

اشهد ان الطالب/الطالبة قد قام /قاموا بالتعديلات والتصحيحات المطلوبة من طرف لجنة المناقشة وقد تم  
التحقق من ذلك من طرفنا  
وقد استوفت جميع الشروط المطلوبة.

مصادقة رئيس القسم



امضاء المسؤول عن التصحيح



رقم: 138 / ح.أ.ج.غ/2026

**شهادة توطين مشروع مبتكر وفق القرار 008 المعدل والمتمم للقرار 1275**

أنا الممضي أسفله، السيد: د/ طالب أحمد نور الدين

مسير حاضرة الأعمال: جامعة غرداية

المقر الاجتماعي/ العنوان: المنطقة العلمية، ص ب 455، غرداية، 47000، الجزائر

بتاريخ: 2025/04/10

رقم علامة الحاضرة: 1004253146

طبيعة المشروع: مؤسسة ناشئة

أشهد أن الطالب(ة) / الطلبة التالية أسماؤهم:

| الإسم واللقب  | الطور الدراسي | التخصص      | الكلية                   |
|---------------|---------------|-------------|--------------------------|
| سليمان ضموئي  | M2            | آلية وأنظمة | كلية العلوم والتكنولوجيا |
| عز الدين شريط | M2            | آلية وأنظمة | كلية العلوم والتكنولوجيا |

تحت إشراف الأستاذ(ة)/الأستاذة التالية أسماؤهم:

| الإسم واللقب     | الرتبة      | التخصص            | الكلية                   |
|------------------|-------------|-------------------|--------------------------|
| عبد الوهاب خطارة | أستاذ مساعد | آلية وكهروميكانيك | كلية العلوم والتكنولوجيا |

تم توطينه على مستوى حاضرة أعمال جامعة غرداية - بمشروع تحت اسم:

**Design and development of smart helmet for industrial safety monitoring and data analysis.**

خلال السنة الجامعية: 2026/2025

سلمت هذه الشهادة بطلب من المعني للإدلاء بها في حدود ما يسمح به القانون.

حرر في غرداية بتاريخ: 2026/04/26

مدير الحاضرة



مسؤول حاضرة الأعمال

طالب أحمد نور الدين



## 4.2 Business Model Canvas for the SISH Helmet

**Table 4.4:** Business Model Canvas for the SISH smart industrial helmet.

| Key Partners                                                                                                                                                                                                                                                                          | Key Activities                                                                                                                                                                                                                                                           | Values                                                                                                                                                                | Customer Relationships                                                                                                                                                          | Customer Segments                                                                                                                                                                                                    |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1. Business incubators & entrepreneurship centers<br>2. Telecom / Wi-Fi providers<br>3. Universities & research centers<br>4. Industrial pilot-site companies<br>5. National safety & standardisation bodies<br>6. Hardware module suppliers<br>7. EN 397 certification / testing lab | 1. Hardware & firmware development<br>2. Monitoring server operation<br>3. Certification & compliance (EN 397)<br>4. On-site pilots, training & support<br>5. B2B sales & business development<br>6. Continuous improvement                                              | 1. Reliability<br>2. Worker safety<br>3. Innovation<br>4. Low cost of ownership<br>5. Technical support                                                               | 1. Long-term B2B contracts<br>2. Service-Level Agreements (SLA)<br>3. Preventive maintenance & updates<br>4. Dedicated technical support<br>5. Co-design of pilot deployments   | 1. Industrial plants & contractors (construction, oil & gas, energy)<br>2. HSE / safety departments & site supervisors<br>3. Emergency-response & maintenance teams<br>4. Hospitals & industrial clinics (secondary) |
|                                                                                                                                                                                                                                                                                       | <b>Key Resources</b>                                                                                                                                                                                                                                                     |                                                                                                                                                                       | <b>Channels</b>                                                                                                                                                                 |                                                                                                                                                                                                                      |
|                                                                                                                                                                                                                                                                                       | 1. Embedded & backend engineers, safety specialist<br>2. Seed funding / incubator support<br>3. Tested SISH helmet prototypes<br>4. Monitoring-server code base & analytics tools<br>5. Industrial pilot sites & datasets<br>6. Intellectual property (dual-node patent) |                                                                                                                                                                       | 1. Direct B2B sales<br>2. Industrial safety-equipment distributors<br>3. Demonstrations via incubators<br>4. Safety & Industry 4.0 exhibitions<br>5. Online presence & webinars |                                                                                                                                                                                                                      |
| <b>Cost Structure</b>                                                                                                                                                                                                                                                                 |                                                                                                                                                                                                                                                                          | <b>Revenue Streams</b>                                                                                                                                                |                                                                                                                                                                                 |                                                                                                                                                                                                                      |
| 1. R&D salaries & prototyping<br>2. Manufacturing & assembly<br>3. Certification & testing (EN 397)<br>4. Monitoring-server maintenance<br>5. Marketing & exhibitions<br>6. Office, lab & infrastructure                                                                              |                                                                                                                                                                                                                                                                          | 1. Sale of SISH helmets & starter kits<br>2. Annual dashboard & analytics licences<br>3. Maintenance & support contracts<br>4. Custom integration & training services |                                                                                                                                                                                 |                                                                                                                                                                                                                      |

The Business Model Canvas presented in Table 4.4 summarises how the SISH project creates, delivers, and captures value. Each building block is detailed below.