**Voice-Controlled Assistive Interface for Patient Autonomy Using Offline ASR on Raspberry Pi Platform**

*Michael Schmidt, Adam Tompkins, and Dr. Jafar Saniie*

*Department of Electrical and Computer Engineering*

*Illinois Institute of Technology, Chicago, IL, U.S.A.*

*Abstract*— **This project introduces MedBridge, an affordable, voice-activated assistive device designed to restore autonomy and improve safety for individuals with limited mobility. Using a Raspberry Pi 4 and the offline Vosk speech recognition engine, MedBridge allows patients to perform essential tasks, such as calling for help or controlling aspects of their environment, through simple voice commands or physical inputs. The system operates entirely offline to ensure user privacy and functions reliably even in noisy environments. With components like a buzzer for nurse alerts, a push-button panic switch, and optional lighting control via LEDs, MedBridge is designed to be intuitive and adaptable for users with varying speech clarity and physical ability. Unlike commercial systems that cost thousands of dollars, MedBridge delivers comparable core functionality using open-source tools and modular hardware. Future enhancements include 3D-printed enclosures, expanded voice command capabilities, and integration of lightweight Large Language Models for more natural communication. MedBridge ultimately aims to empower patients by giving them greater control over their environment and a faster, simpler way to reach caregivers when needed.**

Keywords— speech recognition engine, large language model, vosk

# Introduction

Medbridge is a modular, low-cost assistive platform built to restore environmental control, communication, and health monitoring for individuals with limited mobility. Examples include patients affected by amyotrophic lateral sclerosis (ALS), spinal cord injuries, stroke, and other neuromuscular conditions. While ALS provides a clear use case, affecting about 1.7–2.2 per 100,000 people each year (Rentfrow), Medbridge’s flexible design supports any user who can speak or use a simple physical interface [1].

At its heart, Medbridge uses a Raspberry Pi 4 single-board computer because of its processing power, memory, and general-purpose I/O capabilities (Broadcom 2025) [2]. The device runs entirely offline to protect patient privacy and ensure reliable operation even when internet service is unavailable. An open-source Vosk speech-recognition engine transcribes voice commands in under 200 ms on the Pi’s quad-core ARM CPU (alphacephei.com/vosk) [3]. A Python intent parser then reads Vosk’s JSON output and issues GPIO commands to control lighting relays or activate a bedside buzzer when the user says, “lights on” or “help me.” For users with unclear speech, a large push-button panic switch provides a manual fallback to send immediate alerts.

Medbridge’s hardware bundle includes a plug-and-play USB microphone, an audio amplifier with speaker for text-to-speech feedback, and optional modules such as eye-gaze trackers or flex-sensor gloves to broaden accessibility. Integrated health sensors, such as heart rate and temperature probes, continuously monitor vital signs. If any measurement strays beyond safe limits, the system automatically sends SMS or email alerts to caregivers. A companion mobile app displays real-time vitals and conversation logs, so caregivers can respond quickly without compromising patient confidentiality.

By keeping the total cost under $100, Medbridge offers a dramatic price advantage over commercial speech-generating devices and proprietary nurse-call systems, which often exceed several thousand dollars and require ongoing subscriptions. Its open-platform architecture makes it easy for hospitals, long-term care facilities, nonprofits, and individual caregivers to customize and deploy. Future improvements include optional cloud synchronization for remote monitoring, machine-learning based language prediction to speed input, and eventual brain-computer interface integration. Medbridge closes a crucial gap in assistive technology by combining robust offline operation, multiple input methods, and extreme affordability to extend autonomy and dignity to all mobility-impaired patients.

# Background

Patients with severe motor impairments such as those resulting from high-level spinal cord injuries, advanced amyotrophic lateral sclerosis (ALS), or stroke often retain the ability to speak even when they can no longer control their limbs. Yet most environmental control systems in hospitals or home settings rely on manual switches, sip-and-puff devices, or eye-tracking interfaces, each of which carries its own limitations in terms of cost, learning curve, and reliability under varying lighting or ambient conditions. Recent advances in embedded speech-recognition technology have made it possible to deploy fully offline, low-latency transcription on compact hardware. Vosk, for example, provides a Kaldi-based engine that can reliably convert speech to text in under 200 ms on a Raspberry Pi 4, eliminating any dependence on cloud connectivity or patient data leaving the premises. By combining this offline ASR capability with simple, rule-based intent parsing and GPIO-driven relay control, our project aims to deliver a cost-effective, privacy-preserving assistive device. This system will empower paralyzed users to independently toggle room lighting with intuitive voice commands (“light on,” “light off”) and summon nursing staff (“help me”) via a discrete buzzer or network-logged alert, thereby enhancing patient autonomy, safety, and quality of life.

# III. Competition and Related Works

Several assistive technologies currently exist on the market to help individuals with limited mobility or speech impairments, but they often come at a high cost and require specialized setup. For example, the Tobii Dynavox I-12 is a comprehensive eye-tracking communication tablet priced at $7,990, while the Tobii PCEye 5 eye-tracking bar retails around $3,000 and still requires a separate computer. Other options like the Jouse+ mouth joystick, which uses mouth and facial movements to control a computer mouse, costs approximately $2,045, and the Sip-and-Puff Switch, though cheaper at $210, only serves as a basic input switch and cannot function independently. In contrast, our device provides an all-in-one, voice-controlled solution that performs tasks and can alert nurses at a lower cost. Built using open-source software (VOSK) and affordable hardware components such as LEDs, buttons, and a Raspberry Pi, our system is not only inexpensive but also easier to build, modify, and adapt to different users' needs, making it a more cost-effective and accessible alternative to existing commercial solutions.

*Table I. Open Market Comparisons to Medbridge*

|  |  |  |
| --- | --- | --- |
| Name | Price | Notes |
| Tobii Dynavox I-12 | $7,990 | Eye-Tracking |
| Tobii PCEye 5 | $3,000 | Eye-Tracking |
| Sip-and-Puff | $210 | Alert Switch |
| Mouth Joystick | $2,045 | Mouth Control Device |

# IV. Device Description

1. *Hardware Description*

*Raspberry Pi 4 -* $55

The Raspberry Pi 4 is a compact, credit-card–sized single-board computer powered by a Broadcom BCM2711 SoC featuring a quad-core Cortex-A72 CPU clocked at 1.5 GHz and up to 8 GB of LPDDR4 RAM. Unlike earlier Arduino-class microcontrollers, it runs a full Linux operating system off a microSD card, giving a familiar desktop-style environment with Python, C/C++, and shell utilities. It includes dual-band 802.11ac Wi-Fi, Bluetooth 5.0, and four USB ports (two USB 3.0 and two USB 2.0) that let us plug in high-quality USB microphones or other I/O devices. On the hardware side, its 40-pin GPIO header provides dedicated I²C, SPI, UART, PWM, and digital I/O channels, so we can directly drive a relay module to switch mains power for room lights and control a buzzer or LED indicator without needing extra microcontrollers.

For a voice-activated assistive system, the Pi 4’s combination of processing power, memory, and flexible I/O is invaluable. Its quad-core CPU and generous RAM allow us to run an offline speech-recognition engine such as Vosk—or even experiment with TensorFlow Lite models—while keeping recognition latency under a few hundred milliseconds. Because it hosts the entire software stack locally, there’s no dependence on cloud services for transcription, which is critical in hospital settings where data privacy and network reliability are paramount. The built-in networking makes it trivial to send email or SMS alerts to nurse staff, and its GPIO pins let us seamlessly connect to a relay board to control lights and trigger a nurse-call buzzer.

*Buzzer -* $0.75

The buzzer is a compact PCB-mount cylinder with an integrated driver. It will be connected to a Raspberry Pi board’s GPIO pin and the 5 V rail and the Pi’s 3.3 V GPIO controlling a small transistor as a low-side switch.

*LED -* $0.10

The LED will be a simple Light Emitting Diode that will be connected to the Raspberry Pi via one of the GPIO pins. When a command like “Lights on” or “Lights off” is perceived by the Raspberry Pi, it will send a signal to the LED to communicate which state it should be. If we were to integrate our design into a hospital setting, the LED would be replaced with the lights in the patient’s room so they could have a better sense of control over their environment.

*SunFounder USB Mini Microphone -* $8.00

The SunFounder USB 2.0 Mini Microphone is a compact, plug-and-play audio input device designed for quick and reliable voice capture on any USB-enabled host—PC, Mac, or single-board computer such as the Raspberry Pi. Housed in a low-profile metal shell just 25 mm in diameter, it connects directly to a USB 2.0 port and is recognized instantly by modern operating systems with no additional drivers required. Its omnidirectional electret condenser capsule delivers clear vocal pickup across a 100 Hz–16 kHz frequency range at up to 48 kHz sampling rate, making it ideal for speech-driven applications, online meetings, podcasting, or DIY voice-control projects. With a built-in high-SNR amplifier and a sensitivity of –38 ± 3 dB, the SunFounder Mini Mic captures even quiet commands faithfully while suppressing background noise, and its durable metal housing and integrated USB cable ensure a sturdy, tangle-free installation.

*Sound Bar -* $34

The Redragon GS560 RGB Desktop is a compact and budget-friendly speaker that is designed to be used on TV. It has an aux 3.5mm that supports audio from a Raspberry Pi and can be powered via a usb port. This device serves as an easy connection for the Raspberry Pi to produce audio feedback for voice recognition software. When users want to speak a command, the Pi will reply to an audio to inform the users that their command has been received, and this device is sufficient to accomplish that.

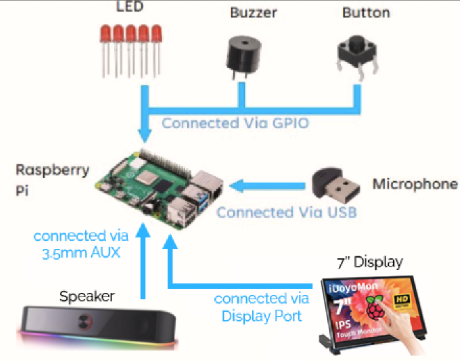
*Raspberry Pi Touch Screen -* $37

The 7” touch screen built by iUoyoMon is designed to be compatible with a Raspberry Pi. It offers a resolution of 800x480 and can be powered entirely by the Pi. It requires a 5v input which can be connected to the 5v GPIO pin on the Pi, and the data can be transmitted via a Display Port line. It offers a touch screen functionality that serves as another way to interact with the device to make it user friendly for all cases. With a simple GUI, users can interact and control the device with buttons that appear on the screen.

*Push Button -* $0.05

The pushbutton is a standard momentary single-pole, single-throw switch mounted on a breakout board with integrated pull-down resistor. It connects to one of the Pi’s GPIO pins and to 3.3 V through the board’s resistor, so it reads HIGH only when pressed. In MedAssist, pressing this button triggers an immediate “panic” alert, bypassing voice input, by sending an SMS or email notification to caregivers via a lightweight Python script. The button’s tactile feel ensures users can activate it easily, even with limited hand dexterity. We’ll mount it on a sturdy, oversized cap so it’s easy to locate and press. Debouncing logic in software prevents false triggers, ensuring that each deliberate press reliably dispatches an alert without repeated or missed signals. The main purpose of including this is to provide multiple ways of contacting the nurse. This would mainly pertain to the limited mobility patients if something was wrong with the speech or the patient can't talk, they can still resort to the basic push button call light for help.

*Figure 1. Interface Layout*



1. *Device Interface Description*

The Raspberry Pi will be the main connection to the hardware. 4 buttons will be used for this device, and they are all connected to a specific pin on the Pi. We have a 3.3v connected to one side of the button, with the same side connected to a ground pin with a 220Ω resistor. This is necessary as it will act as a voltage pulldown so that when the button is pressed and connected with an open pin on the Pi, the device will read it as an active High voltage meaning the button is pressed. Without it, the button can cause a false fire.

The buzzer and LED are connected in a similar way, where one end is connected to a specific Pin on the Pi, so there we can control them individually, and the other side is connected to the GND pin. The buzzer will act as a “help call” for nurses to become alerted if triggered. The buzzer will be powered on if users use the command “help” or press the push button/gui button. The microphone is connected to a usb port on the device to act as a recoreding stream to pick up audio.

The 7’’ display will be connected to the 5V and GND pin to power it, while information will be connected via a ribbon cable to the display port on the Pi. With a gui display, users can also press the buttons on the screen to give input to the device. A speaker is connected to a 3.5mm AUX port on the Pi to output audio, and it is powered via a USB on the Pi. The speaker is for audio feedback whenever users speak with a voice command, so the device can let them know that the command is registered.

*C. Software Description*

*ASR – VOSK:*

Vosk is an open-source, offline speech-to-text engine based on Kaldi that is well suited for embedded applications like a Raspberry Pi-powered assistive device. Its compact English models (on the order of a few hundred megabytes) can run entirely on the Pi’s quad-core Cortex-A72 CPU with latencies typically under 200 ms, ensuring near-real-time transcription without any cloud dependency or network delays. Because Vosk streams audio and outputs JSON-formatted results, it integrates smoothly with Python code for downstream intent parsing and GPIO control. Its robust noise-handling and support for multiple languages make it reliable in a variety of room conditions, and its permissive Apache 2.0 license means we can adapt and extend it freely. All of these features combine to give a private, responsive, and easy-to-deploy voice-recognition backbone for controlling lights and summoning help in environments where reliability and data privacy are paramount.

*Programming Language: Python*

Python is a high-level, interpreted programming language known for its simplicity, readability, and versatility. Designed with an emphasis on code clarity, Python uses a clean and easy-to-understand syntax that allows developers to write fewer lines of code compared to many other languages. It supports multiple programming paradigms, including object-oriented, procedural, and functional programming. Python has become one of the most popular languages in the world due to its wide range of applications, from web development and data analysis to artificial intelligence, automation, and scientific computing. Its extensive standard library and rich ecosystem of third-party packages—such as NumPy for numerical computing, Django for web development, and TensorFlow for machine learning—make it highly adaptable for both beginners and professionals. Python is also cross-platform, meaning programs written in it can run on various operating systems with little or no modification. Its community is large and active, contributing to continuous growth, support, and innovation around the world. We mostly chose python as it is simple to work with, has an extensive library for help, and we are familiar with the language as our background worked with it.

All the code written for our project is original to our group, with the help of usage examples provided in the documentation of software libraries.

*Pyttsx3:*

Pyttsx3 is a Python library that enables text-to-speech (TTS) functionality using offline speech synthesis. Unlike many TTS tools that require internet access, pyttsx3 works entirely locally by interfacing with the speech engines built into your operating system—such as SAPI5 on Windows, NSSpeechSynthesizer on macOS, and espeak on Linux. It allows developers to convert any string of text into spoken audio, making it useful for accessibility tools, voice assistants, and interactive Python applications. The library offers control over various speech properties such as rate (speed of speech), volume, and voice selection (including male and female voices if available). One of its advantages is that it queues speech commands, allowing for non-blocking use in applications that need to continue running while speech is being processed. Because it is lightweight and easy to integrate, pyttsx3 is a popular choice for projects on devices like Raspberry Pi, especially in setups that need consistent, offline voice output.

*D. Real-Time*

- Audio Stream via sounddevice.RawInputStream:The stream is processed in 8000-sample blocks at 16 kHz, meaning roughly every 0.5 seconds a block is queued. If the callback function is delayed and cannot enqueue fast enough (q.put()), the stream could underrun, and audio input may be lost or garbled.

- Speech Recognition (rec.AcceptWaveform(data)): Vosk can process ~0.5s of audio per call. If system load is high (e.g., GUI freeze, CPU spike), this processing may fall behind or miss parts of the command.

- pyttsx3 Text-to-Speech (tts.runAndWait()): This blocks execution until the TTS finishes. If the command takes a long time to speak or multiple commands queue up, the system may lag or become unresponsive.

- GPIO Callbacks (e.g., toggle\_led): GPIO event callbacks are relatively lightweight but synchronous. If a callback takes too long (like help\_ASSIST() using sleep(1)), it could block or delay subsequent events, especially under high CPU load.

*If our system were to encounter any unexpectedly large delays, the cause would most likely be one of the problems listed, and the results explained.*

*E. Security*

1. Voice Command Hijacking (Unrestricted Access)

- Risk: Anyone nearby who says "Hey Bridge" can activate the system and issue commands.

- Implication: A person could trigger lights, the buzzer, or spam commands without authentication.

- Mitigation:

a. Add a simple speaker recognition filter or a physical proximity check (e.g., PIR sensor).

b. Implement a "PIN word" or gesture confirmation after waking up.

2. TTS Misuse (pyttsx3 Exploit Potential)

- Risk: If commands are extended to take input from dynamic sources (e.g., network or file), malicious input could be spoken out loud.

- Implication: Could be used to communicate phishing messages, fake instructions, or psychological manipulation.

- Mitigation: Sanitize all dynamic content before sending it to tts.say().

3. GUI (Tkinter) Exposure

- Risk: The GUI is locally hosted, but if run in a remote-accessible environment (e.g., via VNC or X11 forwarding), anyone with access can interact with it.

- Mitigation:

a. Run the GUI locally.

b. Lock access to the Raspberry Pi via strong SSH credentials and firewall rules.

4. No Exception Handling in GPIO Callbacks

- Risk: If one of the GPIO callbacks throws an exception (e.g., due to a hardware fault), it could crash the app or leave hardware in an undefined state.

- Mitigation: Add try-except blocks around GPIO event functions.

5. No Runtime Permissions or Command Filtering

- Risk: All commands are executed without checking who sent them or how frequently.

- Implication: Potential for command spamming (especially help call buzzer).

- Mitigation: Add command rate limiting or sleep mode lockout for repeated help calls.

6. Long-Running Thread Without Exit Signal

- Risk: GUI and audio threads don’t coordinate shutdown or state cleanly. This could result in:

a. Zombie processes

b. Hanging GPIO pins

c. Orphaned resources

- Mitigation: Implement a graceful shutdown signal and use threading.Event() flags for inter-thread coordination.

# V. Results and Discussion

The initial prototype of the device was successfully developed and integrated with the VOSK offline speech recognition system. Functionality testing revealed that VOSK performs well when recognizing clearly spoken commands, even in moderately noisy environments. For instance, when tested in a gym setting with background noise, the system was still able to reliably pick up commands such as “lights on” and “lights off.” However, limitations were observed when dealing with slurred or unclear speech, which poses a challenge for users with speech impairments; a key target group for this device.

The hardware implementation included LEDs and push buttons, where the LEDs were configured to simulate room lighting. Users can control these LEDs either by voice command through VOSK or manually via a physical push button. An additional button was incorporated to adjust the brightness of the LEDs, giving users an alternate method of light control. While the manual brightness control functioned as intended, further development is required to enable VOSK to adjust brightness levels through voice commands.

Future improvements will focus on both hardware and software enhancements. On the hardware side, the addition of a speaker is planned to provide audible feedback in response to user commands, along with the installation of a buzzer for additional alerts. On the software side, the code will need to be extended to allow for more nuanced voice-controlled actions, such as brightness modulation. Finally, a custom case will need to be designed and 3D-printed to safely and neatly house all components of the device, ensuring it is user-friendly and portable.

*User Manual:*

1. Look at script comments to wire the hardware to specific pins.
2. Create a python virtual environment and place the code in Appendix A in the same folder, then download software dependencies. You can check by running the code “python RunTranscript” on the command line in the same directory as the script.
3. The code should be functioning, and a GUI screen should appear. Press the respective buttons to toggle on/off the LED, increase/decrease the LED brightness, and sound the buzzer.
4. Pressed the buttons on the GUI to do the same actions
5. Say “Hey Bridge” to wake up the device, then say these commands:
6. “Lights On”: To turn on LED
7. “Lights Off”: To Turn Off LED
8. “Increase Brightness”: Increase LED brightness
9. “Decrease Brightness”: Decrease LED Brightness
10. “Help”: To sound the buzzer

*Looking Ahead:*

*Hospital Integration and Housing Unit:*

Our hospital-integration research will begin by reviewing interoperability and compliance standards that govern medical-device deployment in clinical settings. We will examine Health Level Seven and DICOM protocols to ensure seamless data exchange with existing hospital IT systems, and we will map device management workflows against The Joint Commission’s equipment-maintenance and environment-of-care requirements to guarantee accreditation readiness. We will also survey FDA classification rules for general-use hospital devices and study CDC guidelines for device disinfection and sterilization to maintain infection-control standards. Power and mounting needs such as outlet types, voltage regulations, wall-bracket specifications, and cable routing, will be documented, along with cybersecurity considerations like AAMI TIR57 risk-management principles.

In parallel, our enclosure design research will focus on selecting materials and IP ratings appropriate for a clinical environment. We will assess ingress protection requirements based on IEC 60529 classifications, exploring IPX6–IPX7 options to guard against fluid exposure during routine cleaning or accidental spills. Material studies will compare ABS, polycarbonate, and stainless-steel housings for strength, chemical resistance, electromagnetic shielding, and ultrasonic-cleaner compatibility. Ergonomic factors, such as rounded edges, mounting brackets, and service-access panels, will be prototyped in CAD and evaluated for ease of installation, user interaction, and maintenance under hospital workflows.

*Think about LLM Integration into Device to Replace VOSK:*

To implement the LLM, we plan to replace Vosk’s rule-based speech-to-text and intent parsing system with a lightweight, locally hosted LLM that can handle both transcription and natural language understanding in a single pipeline. This will involve using an edge-optimized model like Whisper for audio transcription, followed by a compact transformer-based LLM such as TinyLlama or Mistral-Instruct to interpret the user’s intent. Audio input from the USB microphone will be transcribed using Whisper, then passed as text to the LLM, which will analyze the context and determine the appropriate GPIO action, such as turning lights on or sending an alert. All inference will run directly on the Raspberry Pi or a local server to maintain offline operation and protect patient privacy.

# VI. Conclusion

The MedBridge project successfully demonstrates that it is possible to design a low-cost, offline, voice-activated assistive device that empowers individuals with limited mobility to regain control over their environment. By leveraging the Vosk speech recognition engine and integrating it with a Raspberry Pi, basic GPIO-controlled hardware, and a user-friendly GUI, we have developed a fully functional prototype capable of responding to simple voice commands even in noisy environments. The device proved to be reliable in recognizing clear speech and executing commands to control room lighting and call for assistance. Through extensive testing and component integration, including LEDs, push buttons, a USB microphone, a buzzer, and a speaker, we’ve built a flexible platform that mimics commercial solutions at a fraction of the cost. Unlike expensive competitors on the market, MedBridge remains modular, customizable, and free from cloud dependency, preserving privacy and offering real-world utility at an accessible price point. As we look ahead, planned enhancements such as incorporating a 3D-printed enclosure, expanding voice command capabilities, and integrating lightweight Large Language Models will further improve functionality, usability, and inclusivity. With continued development, MedBridge has the potential to become a vital tool for hospitals, caregivers, and individuals seeking independence, helping bridge the gap between affordability, accessibility, and assistive technology

VII. *References*

[1] Rentfrow, Melissa. “ALS Statistics and Recent Research You Need to Know.” *Target ALS*, 20 Feb. 2025, www.targetals.org/2025/01/30/als-statistics/.

[2] Broadcom. “BCM2711 SoC Product Brief.” Raspberry Pi 4 Model B Datasheet, 2025.

[3] “VOSK Offline Speech Recognition API,” Vosk offline speech recognition API, https://alphacephei.com/vosk/ (accessed May 28, 2025).

[4]Raspberrypi4modelB, https://datasheets.raspberrypi.com/rpi4/raspberry-pi-4-product-brief.pdf (accessed May 28, 2025).

[5]“Activepassivebuzzer,”Components101, https://components101.com/misc/buzzer-pinout-working-datasheet (accessed May 28, 2025).

[6] T. Cooper, “All about leds,” Adafruit Learning System, https://learn.adafruit.com/all-about-leds/the-led-datasheet (accessed May 28, 2025).

[7] “Raspbeery Pi Mini USB 2.0 Microphone Buy Online,” Main Website, https://probots.co.in/raspbeery-pi-mini-usb-2-0-microphone.html (accessed May 28, 2025).

[8] “Mini Metal Speaker – 3 W 8 Ω,” Adafruit, https://www.adafruit.com/product/1669 (accessed June 3, 2025).

[9] “PAM8403 Super Mini 5V 3W\*2 Digital Amplifier Board USB Powered Volume Control.” *PAM8403 Super Mini 5V 3W\*2 Digital Amplifier Board USB Powered Volume Control,Shenzhen HiLetgo Technology Co., Ltd*, www.hiletgo.com/ProductDetail/2159777.html. Accessed 3 June 2025.

[10]“TactileOn/OffPushbuttonSwitch,”Adafruit, https://www.adafruit.com/product/1114 (accessed June 3, 2025).

[11] Complete Guide To Medical Device Integration Implementation. OSPLabsInsights.https://www.osplabs.com/insights/medical-device-integration

[12] Medical Equipment Compliance – Joint Commission standards. STERIS Knowledge Center. https://www.steris.com/healthcare/knowledge-center/equipment-maintenance-and-device-repair/medical-device-compliance

[13] 21 CFR Part 880 Subpart G – General Hospital and Personal Use Devices. ECFR. https://www.ecfr.gov/current/title-21/chapter-I/subchapter-H/part-880/subpart-G

[14] Guideline for Disinfection and Sterilization in Healthcare Facilities. CDC.https://www.cdc.gov/infection-control/media/pdfs/guideline-disinfection-h.pdf

[15] IP code (IEC 60529) – Ingress Protection. Wikipedia. https://en.wikipedia.org/wiki/IP\_code

[16] Choosing the Right Enclosure for Your Medical Device. Onechassis. https://gpuservercase.com/blog/medical-device-enclosures/

[17] Han, Song, Huizi Mao, and William J. Dally. “Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding.” *International Conference on Learning Representations (ICLR)*, 2016. https://arxiv.org/abs/1510.00149 (accessed June 12, 2025).

[18] Google Coral. “Coral Edge TPU: Delivering High Performance ML at the Edge.” *Coral.ai*, 2024. https://coral.ai/docs/edgetpu/overview/ (accessed June 12, 2025).

[19] U.S. Department of Health and Human Services. “Summary of the HIPAA Security Rule.” *HHS.gov*, 2013. https://www.hhs.gov/hipaa/for-professionals/security/laws-regulations/index.html (accessed June 12, 2025).

[20] Reddy, Chiranjeevi, et al. “Implementing Real-Time Speech Recognition on Edge Devices.” *IEEE Internet of Things Journal*, vol. 7, no. 8, Aug. 2020, pp. 7123–7134. https://doi.org/10.1109/JIOT.2020.2980451 (accessed June 12, 2025).

[21] Holtzman, Ari, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. “The Curious Case of Neural Text Degeneration.” *International Conference on Learning Representations (ICLR)*, 2020. https://arxiv.org/abs/1904.09751 (accessed June 12, 2025).

[22] “Amazon.Com: Riowois Sound Bar for Smart TV, Soundbar TV Speakers with Customized Bass Reflector, Compact Design and Easy Set-up, Home Audio Surround Sound System for TV PC Projector, Wall Mountable: Electronics.” *Amazon*, www.amazon.com/RIOWOIS-Soundbar-Customized-Reflector-Projector/dp/B0D3PJM8D8. Accessed 29 June 2025.

[23]*Amazon.Com:RaspberryPi:Computers*, www.amazon.com/stores/RaspberryPi/page/0F286773-FF30-489A-BD7F-4A9F35ED1A59. Accessed 29 June 2025.