Engineering Experiences 3, Electronics-ICT, May 2025, Bruges, Belgium

Low-energy backup communication system for hydrogen racecar

Jarno Mechele Joey De Smet Robijn Ameye
Faculty of Engineering Technology, KU Leuven - Bruges Campus
Spoorwegstraat 12, 8200 Bruges, Belgium
{jarno.mechele, joey.desmet, robijn.ameye}@student.kuleuven.be

Abstract

Summary of the research and conclusion, Problem, method, results, conclusion....

Keywords—Low power, Long-range wireless, Embedded systems

I. Introduction

In hydrogen-powerd endurance racing, uninterupted telmetry and communication between the pitwall and the car are essential for both competitive performance and for driver safety. As systems could fail due to multiple reasons, and cause a loss of communication costing laps or even endager lives, a dedicated low-power backup is required. This paper therefore proposes a long-range wireless solution capable of transmitting and receiving both critiacal sensor data and messages to the driver over distances up to 2km (the approximate diameter of the Le Mans circuit) while only consuming a few milliwatts. By combining a encrypted LoRa-based RF link with embedded speech synthesis and WAV playback on an ultra-low-power STM32U5 microcontroller, our design ensures that even in the event of primary-system failure, the pit-crew retains awareness of the car's most critical data and issue instructions to the driver.

II. Overall system design

III. Firmware Design and Implementation

This section describes the firmware developed for the STM32U5 microcontroller [1], which forms the core of our low-energy backup communication system. It handles real-time LoRa communication, sensor data acquisition, and voice output via speech synthesis or WAV playback. To guarantee deterministic timing, we build on FreeRTOS [2], as illustrated in Figure 1. Task isolation and priority levels make the codebase modular and maintainable.

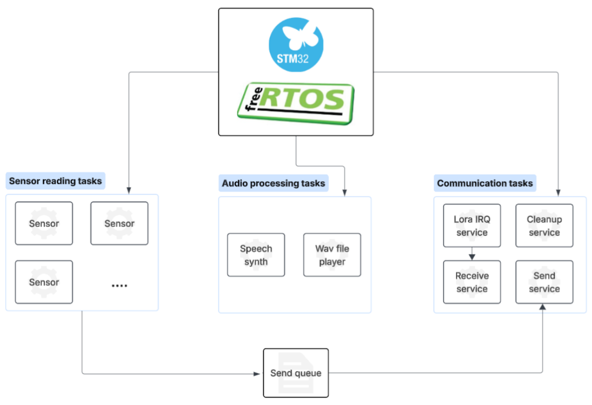

Fig. 1. Overview of the FreeRTOS-based firmware architecture

A. Overall Architecture

Figure 1 shows three primary functional domains, each implemented as one or more FreeRTOS tasks communicating via inter-task communication mechanisms provided by FreeRTOS.

B. LoRa Communication

- **IRQ Handler** Waits on the SX1276 [3] interrupt line to detect packet RX/TX completion, then gives a binary semaphore.
- **Receive Task** Blocks on that semaphore, retrieves incoming packets, decrypts them with hardware-accelerated AES-128 in CTR mode, and forwards them for processing.
- **Transmit Task** Pulls outgoing messages from a FreeRTOS queue, encrypts and formats them, then issues the LoRa send command.
- **Cleanup Task** Periodically scans stored packet buffers for timeouts and frees associated heap memory.

The message formatting steps mentioned above, including encryption, fragmentation, and encapsulation, are detailed in Section III-F.

C. Sensor Management

Each sensor (e.g. temperature, pressure, speed) runs its own task at a low priority. Tasks periodically sample the hardware interface, package readings, and enqueue them for transmission.

D. Audio Processing

- **Speech Synthesis Task** A port of `espeak-ng` [4] with all file I/O replaced by in-memory C arrays. Which dequeues strings from a FreeRTOS queue, synthesises, streams audio to I2S hardware.
- **WAV playback Task** Streams hard-coded WAV data (e.g. racing flags, standard phrases) to the I2S hardware.

E. Power and Memory Management

All tasks are assigned carefully chosen priorities (Table I) so that time-critical communication tasks preempt lower-priority work. We enable FreeRTOS tickless idle to allow the STM32U5 to enter deep sleep whenever the system is idle.

TABLE I. Task priorities and stack usage

Task	Priority	Stack (bytes)
LoRa IRQ Handler	8	128
Speech Synthesis	7	48 000
WAV Playback	7	256
LoRa Transmit	5	128
Cleanup	5	128
Sensor (each)	3	256

F. Fragmentation and Reassembly of Encrypted Messages

The transmission process, illustrated in Figure 2, begins with raw message data assigned a unique ID. This data is then processed through the following steps:

Message Transmission Pipeline

1) The raw message is padded using PKCS#7 to ensure its length is a multiple of 16 bytes, satisfying the requirements of the hardware-accelerated AES module.
2) The padded message is encrypted using AES-128 in CTR mode.

3) The resulting ciphertext is fragmented into fixed-size chunks. Each fragment is encapsulated in a packet containing the meassage ID, fragment index, and total number of fragments.
4) Packets are enqueued for transmission via LoRa.

Message Reception Pipeline

5) Recieved packets are collected in a queue.
6) Each packet is decapsulated and sorted by message ID. Once all fragments of a message are received, they are reassembled into the full encrypted message.
7) The complete ciphertext is decrypted to recover the padded plaintext.
8) Padding is removed, restoring the original raw message.

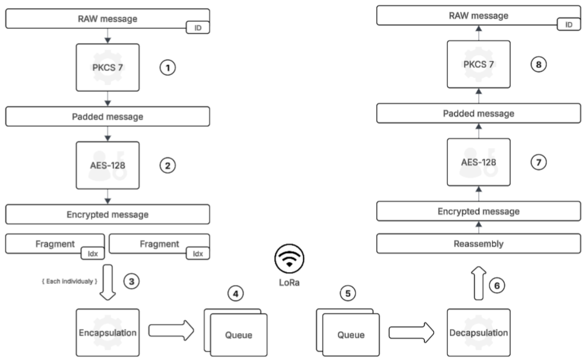

Fig. 2. Flowchart illustrating the message encryption fragmentation process prior to LoRa transmission.

IV. Backend and graphical user interface Design and Implementation

V. Hardware Design and Implementation

A. System overview

The hardware functionalities can be divided into six main components: power supply, microcontroller, audio recording, audio playback, wireless communication, and power switching. Apart from the microcontroller, these components are implemented on a custom PCB designed as a shield compatible with an STM32 Nucleo board. All components are designed to be low-power and operate from a 12V power source. The power supply includes a switching regulator that generates both 3.3V and 5V rails to power the remaining circuitry. Audio input is handled via an external analog-to-digital converter (ADC), while audio output is managed through an external digital-to-analog converter (DAC). The power to these audio components can be switched on or off by the microcontroller using MOSFETs, allowing the system to conserve energy when the components are not in use. Processed speech data can be transmitted or received via a LoRa module, and subsequently played back if needed.

B. Power supply

The power supply uses two buck converters to step down the 12V input to 5V and 3.3V, based on the TPS629203. This converter supports input voltages up to 17V and delivers up to 300mA of output current [5]. Thanks to its high switching frequency, it achieves excellent efficiency, which is further optimized by dynamically adjusting both the switching frequency and mode depending on the load. It operates in either PWM (Pulse Width Modulation) or PFM (Pulse Frequency Modulation), depending on the current demand. This feature, known as Automatic Efficiency Enhancement (AEE), maintains high efficiency even at very low duty cycles, making it particularly suitable for low-power applications.

C. Microcontroller

The STM32U5 microcontroller was selected for this design. This latest generation of STM32 ultra-low-power microcontrollers is well-suited for applications that require minimal energy consumption. Despite its low power profile, certain variants offer up to 4 MB of Flash memory and 3 MB of SRAM, enabling complex processing and data storage [1]. The microcontroller supports multiple low-power modes, allowing it to significantly reduce power consumption during periods of inactivity, such as while waiting to send or receive messages. Wake-up from these modes is achieved using Low-Power General-Purpose Input/Output (LPGPIO) pins. In this setup, an LPGPIO pin is connected to both the LoRa module and a digital input. This configuration enables the microcontroller to wake up upon receiving incoming LoRa data or when the talk butto connected to the digital input is pressed.

D. LoRa

For wireless communication, LoRa was selected. This technology serves as the physical layer for the more well-known LoRaWAN protocol. The chosen module, the RFM96W by HopeRF [6], is readily available and supported by numerous examples and open-source libraries online, making integration and development straightforward. This compact PCB can be hand-soldered and includes all the necessary hardware for LoRa communication, except for the antenna. Communication with the module is established via the SPI bus and GPIO pins. Data to be transmitted is sent to the module via the SPI interface, while incoming data is signaled through a dedicated I/O pin. The antenna is connected to the PCB using an SMA connector. Both $\frac{\lambda}{2}$ and $\frac{\lambda}{4}$ antennas can be used. A $\frac{\lambda}{2}$ antenna may also be placed remotely using a coaxial cable, allowing for more flexible antenna positioning.

E. ADC

Om audio in te lezen is gebruik gemaakt van een externe ADC, de PCM1808.

F. DAC

G. Power switching

H. PCB

VI. Testing

A. Test Setup

The system was evaluated primarily in a laboratory environment. Key aspects such as power consumption, LoRa communication integrity, and audio output functionality were tested. Most tests were conducted manually, involving the transmission of periodic test messages and monitoring system responses via serial output and audio feedback. Power draw was observed using an inline current measurement tool, and LoRa transmissions were validated using logic analyzers and debug logs.

B. Communication Testing

To verify LoRa communication, encrypted packets were periodically transmitted and received between two system nodes. Tests confirmed successful decryption, CRC validation, and end-to-end data integrity.

C. Audio Output Testing

WAV file playback was validated by connecting the system's analog output to external speakers and assessing the clarity of preloaded audio samples. Speech synthesis was tested by transmitting predefined sentences to the device and evaluating intelligibility through blind listening tests. Testers rated intelligibility qualitatively. While overall clarity was sufficient for basic commands, future improvements may include tuning output filters and optimizing speech synthesis parameters for better naturalness.

VII. Conclusion

Acknowledgment

The authors would like to thank the supervising professors for their guidance throughout the development of this project. Special thanks go to the Hydro Team KU Leuven for providing valuable context and support related to the vehicle's communication needs.

Additionally, the authors used generative AI tools to assist with language refinement and grammar correction during the preparation of this paper.

References

[1] STMicroelectronics, *STM32U5-series overview*, 2023. Available: https://www.st.com/en/microcontrollers-microprocessors/stm32u5-series.html

[2] Real Time Engineers Ltd., *FreeRTOS Kernel Reference Manual*, 2024. Available: https://www.freertos.org/Documentation/RTOS_book.html

[3] Semtech Corporation, *SX1276 LoRa Transceiver IC*, 2025. Available: https://www.semtech.com/products/wireless-rf/lora-connect/sx1276

[4] eSpeak-NG, *eSpeak NG – Next Generation Text to Speech*, 2025. Available: https://github.com/espeak-ng/espeak-ng

[5] STMicroelectronics, *Datasheet TPS629203*, 2025. Available: https://www.ti.com/product/TPS629203

[6] HOPERF, *RFM96W overview*, 2025. Available: https://www.hoperf.com/modules/lora/RFM96W.html

[7] STMicroelectronics, *Datasheet PCM1808*, 2025. Available: https://www.ti.com/lit/ds/symlink/pcm1808.pdf

[8] Texas Instruments, *Design example*, 2025. Available: https://www.ti.com/lit/an/sboa290a/sboa290a.pdf?ts=1747751396287