



UDC 004.89

A MULTIFUNCTIONAL SMARTCLOCK FOR VOICE INTERACTION AND ADAPTIVE TASK SCHEDULING

Dmytro Kozliuk , **Halyna Klym** 
Lviv Polytechnic National University,
12 Bandera St., Lviv 79013, Ukraine

Kozliuk, D., & Klym, H. (2025). A Multifunctional Smart Watch Clock for Voice Interaction and Adaptive Task Scheduling. *Electronics and Information Technologies*, 31, 31–44.
<https://doi.org/10.30970/eli.31.3>

ABSTRACT

Background. Current smart devices are often limited to separate functions such as timekeeping, environmental sensing, or voice assistance. This fragmentation hinders a unified solution for productivity in modern workspaces, where indoor conditions and time management are key. Cloud systems face latency and connectivity issues, while local ones are limited by hardware. This study presents a multifunctional smart clock combining environmental monitoring, voice interaction, and task scheduling to enhance comfort, focus, and efficiency.

Methods. The system uses an Edge–Cloud architecture: the ESP32-S3 edge runs latency-critical functions under FreeRTOS with an OOP design. Audio from a MEMS microphone (I²S) is windowed and converted via short-time FFT to log-mel spectrograms; a quantized CNN (TFLM) performs on-device keyword spotting for wake-word detection. After wake-word detection, commands are sent to Wit.ai for ASR/NLU. Audio output is driven by a Class-D amplifier and speaker. Environmental sensing covers temperature, humidity, illuminance, and CO₂ (NDIR), with filtered readings shown in an event-driven LVGL touch GUI and periodically uploaded for analysis to Firebase.

Results and Discussion. The CNN wake-word detector achieved ~90% activation accuracy in quiet-to-moderate office noise with FAR <1 trigger/hour at ~10% FRR; median detection latency remained <200 ms after sufficient context accumulation. Under RTT ≤100 ms, cloud ASR/NLU yielded end-to-end wake→intent latency ≈1–1.5 s. Concurrent environmental monitoring at a 2-s cadence did not perturb the audio pipeline, GUI sustained a 25 Hz refresh rate, and after WWD the system opened a bounded 4-s command window for user utterances. Wi-Fi provisioning via an embedded web server and hourly cloud uploads produced coherent. Threshold-driven voice/visual prompts increased awareness of indoor conditions, while integrated Pomodoro cycles supported sustained focus without auxiliary tools.

Conclusion. The proposed platform integrates voice assistance, time management, and microclimate monitoring in an affordable edge device. Its hybrid speech design balances latency and flexibility, while FreeRTOS ensures reliable multitasking across sensing, GUI, networking, and audio subsystems.

Keywords: Edge computing, RTOS, NDIR, Pomodoro, embedded voice interaction, microclimate monitoring.

INTRODUCTION

In recent years, smart devices have become deeply embedded in daily life, providing users with seamless access to information, communication, and automation [1–3]. Among



© 2025 Dmytro Kozliuk & Halyna Klym. Published by the Ivan Franko National University of Lviv on behalf of Електроніка та інформаційні технології / Electronics and information technologies. This is an Open Access article distributed under the terms of the [Creative Commons Attribution 4.0 License](https://creativecommons.org/licenses/by/4.0/) which permits unrestricted reuse, distribution, and reproduction in any medium, provided the original work is properly cited.

them, smartwatches have gained particular prominence, offering not only timekeeping functions but also health monitoring, fitness tracking, communication, and integration with broader smart ecosystems. Depending on their intended use, these devices vary from fitness trackers, which focus primarily on monitoring physical activity and health indicators, to multifunctional smartwatches such as the Apple Watch and Samsung Galaxy Watch, which combine smartphone-level functionality with portability [4,5]. Sports-oriented devices like Garmin and Suunto further emphasize durability, GPS navigation, and extended battery life, catering to outdoor enthusiasts [6].

While wearable devices continue to dominate the consumer market, desktop smart clocks represent a growing but less explored niche. Unlike smartphones or laptops, which often contribute to information overload and distraction, desktop smart clocks provide quick, distraction-free access to essential information such as time, weather conditions, reminders, or indoor climate data. Furthermore, they often function as central hubs for smart home automation, enabling voice-based interaction with other devices, while also integrating additional features such as alarms, timers, night lights, and environmental monitoring through sensors. Recent studies highlight the importance of such devices in supporting productivity and well-being, as they combine convenience with minimal cognitive load [7].

One of the most important aspects of modern smart devices is time management. The Pomodoro technique has been widely studied as a method of enhancing productivity by breaking work into structured intervals. In the work [8] it was demonstrated that while the Pomodoro technique helps establish clear structures, it may also accelerate fatigue compared to alternative approaches such as Flowtime. Other research has suggested positive effects on learning outcomes and writing skills, particularly among students [9], though findings on memory retention remain mixed (De Guzman & Abad, 2023). This indicates that Pomodoro-based tools, when properly integrated into daily routines, may support concentration and efficiency, but their impact is highly context-dependent.

Equally relevant is the role of environmental monitoring in enhancing productivity. Recent advances in Internet of Things (IoT) technologies have enabled real-time tracking of indoor parameters such as temperature, humidity, and air quality [10]. Systems like IoFClima [7] apply fuzzy logic and IoT sensors to dynamically control indoor conditions, demonstrating that comfort and energy efficiency can be jointly optimized. However, most of these solutions exist as specialized systems rather than integrated desktop devices.

Another important dimension is voice interaction. Cloud-based assistants such as Amazon Alexa, Google Assistant, or Siri dominate the consumer market, yet their integration into resource-constrained embedded systems remains challenging. Alternatives such as TensorFlow Lite, ESP-Skainet, and Picovoice provide opportunities for on-device speech recognition, which reduces latency and enhances privacy. Recent open-source projects demonstrate that microcontrollers like the ESP32-S3 are increasingly capable of supporting voice-based interactions while simultaneously handling sensor data and user interfaces [11,12].

Despite these developments, current products typically remain limited to a narrow set of functions, either as fitness-focused wearables, environmental monitors, or standalone timers. There is still a lack of integrated, affordable solutions that combine voice control, environmental monitoring, and structured time management in a single desktop device.

The aim of this work is therefore to design and develop a hardware–software system for a desktop smart clock with an integrated voice assistant and Pomodoro timer, based on the ESP32-S3 microcontroller. This system seeks to unify time management, environmental monitoring, and voice interaction into one device, thereby enhancing user comfort, concentration, and productivity in modern workspaces.

METHODS

Development of the structural diagram and operating algorithm.

The design of the proposed smart clock was guided by the need to combine several independent functionalities into a single, user-friendly device. Unlike existing commercial solutions, which are often tailored to specific tasks and lack a holistic approach, this system integrates microclimate monitoring, a Pomodoro timer, voice control, and visual feedback within one compact platform. The structural design of the device reflects this integration, enabling seamless interaction between hardware, software, and cloud-based services.

At the core of the system is the user interface, which relies on multimodal communication through voice commands, touch interaction, and visual indicators. The smart clock incorporates an array of environmental sensors, including a BME280 for temperature and humidity, an MH-Z19B for air quality monitoring, and a BH1850 for illumination measurement. These components provide real-time data about the surrounding environment, which is essential for maintaining healthy working and learning conditions. To complement these sensing capabilities, the device is equipped with an INMP441 microphone for voice input and a MAX98357-driven speaker for auditory feedback, ensuring responsive and natural communication with the user. Visual output is delivered via an ILI9341 display, which presents essential information such as time, environmental parameters, and timer status, while RGB LEDs (WS2812B) provide intuitive notifications through color-coded signals.

The computational backbone of the system is the ESP32-S3 microcontroller, which orchestrates data acquisition, processing, and communication with external services. A real-time clock (DS3231) guarantees accurate timekeeping, even in cases of temporary power loss. Beyond its local processing capabilities, the device is designed for cloud connectivity. Sensor readings are transmitted to a Firebase database, where they can be stored and analyzed over time, allowing the user to monitor long-term trends in microclimate conditions. The voice assistant functionality is implemented through integration with WitAI, a machine-learning-based web service that enables natural language recognition and command execution. This open-source approach ensures transparency, extensibility, and adaptability to user-specific requirements, while also fostering community-driven improvements.

The operational algorithm of the device was carefully structured to guarantee reliability and user convenience. Following system initialization, which activates the processor and peripheral modules, the smart clock attempts to connect to a Wi-Fi network. If stored credentials are available, the device automatically connects as a station; otherwise, it activates its access point mode to allow the user to provide network details. Once connectivity is established, the system initializes secure communication with Firebase and WitAI servers. Successful connections enable real-time synchronization of environmental data and support the functionality of the voice assistant.

The workflow of the device follows a logical sequence of states. After initialization and server connection, the system continuously collects environmental parameters, displays them on the screen, and evaluates them against predefined thresholds. Deviations trigger visual or auditory alerts, prompting the user to adjust their surroundings, such as ventilating the room or modifying lighting conditions. Simultaneously, the system remains in standby mode, waiting for a voice activation command. When such a command is detected, the audio is processed locally on the ESP32-S3 before being transmitted to WitAI for recognition. If the command is successfully identified, the corresponding action is executed; otherwise, the device provides auditory feedback indicating that the request could not be processed. To ensure smooth operation, task scheduling is implemented using a real-time operating system (RTOS), which manages parallel processes such as sensor data acquisition, screen updates, logging, and voice communication.

The proposed design demonstrates how a combination of modular hardware, cloud integration, and open-source philosophy can result in a versatile and accessible smart device. By embedding environmental awareness, productivity-enhancing tools, and natural interaction modalities into one platform, the system addresses the limitations of existing solutions and sets the foundation for further improvements by the developer community.

Design choices in hardware and software.

During the development of the smart clock, a detailed analysis of both hardware and software solutions was carried out, which made it possible to determine the optimal set of components for the implementation of the project. Based on a comparison of different microcontrollers, the ESP32-S3 was selected as the core unit. It combines high computational performance, advanced communication capabilities (Wi-Fi and Bluetooth), and low power consumption. These features make it suitable for working with multiple sensors, processing voice commands, and integrating with cloud services, which are essential for modern IoT devices.

For monitoring environmental parameters, the BH1750 (illumination), BME280 (temperature, humidity, and pressure), and MH-Z19 (CO₂ concentration) sensors were chosen. The selection of these modules was determined by their measurement accuracy, low energy consumption, and support for the I²C interface, which ensures efficient use of the microcontroller's GPIO pins.

The audio subsystem was implemented using the INMP441 microphone and MAX98357 amplifier paired with a speaker, providing high-quality audio capture and playback. This configuration is optimal for implementing a voice assistant. For data visualization, the ILI9341 display was selected due to its adequate resolution, color rendering, refresh speed, and cost efficiency, while the WS2812B RGB LED module was included to provide visual status indications. To ensure precise timekeeping, the DS3231 real-time clock module was integrated, guaranteeing stability even in cases of power failure.

On the software side, the C++ programming language was chosen, as it offers a balance between performance, object-oriented development, and efficient resource utilization. Visual Studio Code in combination with PlatformIO was used as the development environment, enabling effective project organization, broad library support, and extended debugging capabilities. The voice assistant was implemented through a hybrid approach: TensorFlow Lite was used for on-device processing, while Wit.ai provided cloud-based natural language analysis. The Arduino framework was selected due to its simplicity in integrating modules and sensors, which accelerates development and facilitates testing.

RESULTS AND DISCUSSION

Software architecture and RTOS integration.

The software architecture of the developed smart clock is designed to ensure efficient management of real-time data processing, peripheral control, and user interaction (see **Fig. 1**). The system operates on the ESP32-S3 microcontroller under a real-time operating system (RTOS), which allows concurrent execution of multiple tasks while maintaining predictable response times for critical operations such as audio processing and sensor data acquisition. This architecture provides the necessary flexibility to integrate both voice and touchscreen interfaces while maintaining reliable performance in a resource-constrained embedded environment.

At the core of the system, low-level hardware abstraction facilitates seamless interaction with the microcontroller's peripherals, including GPIO, I²C, SPI, UART, and I²S interfaces. This approach isolates hardware-specific details from higher-level modules, enabling peripheral drivers to manage device initialization, data acquisition, and calibration

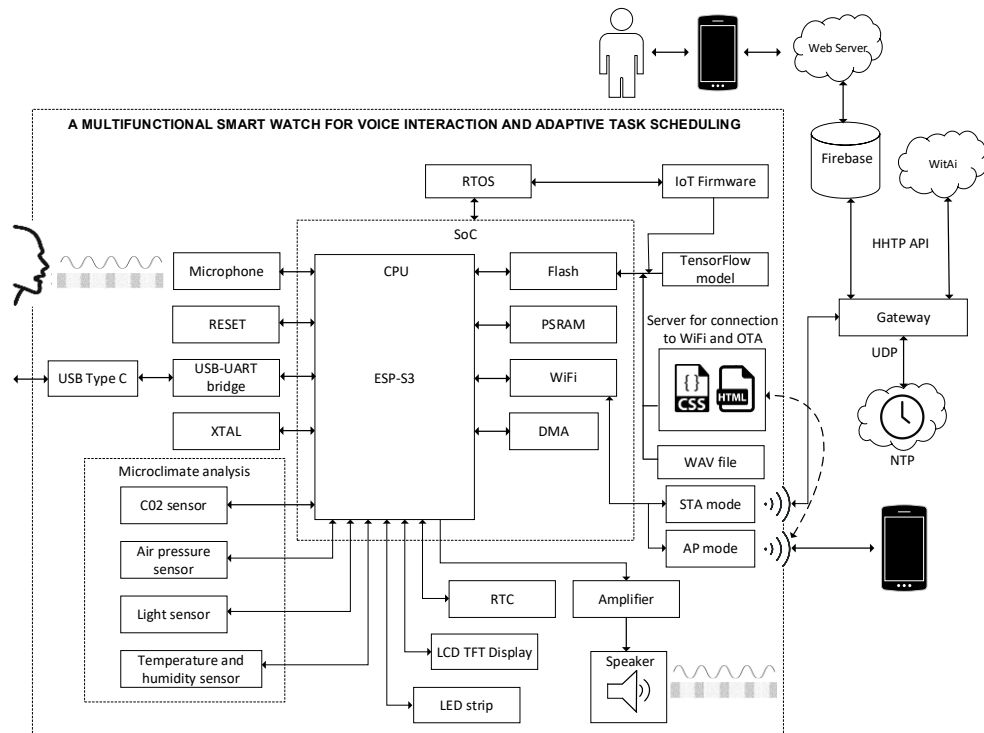


Fig. 1. Structural block diagram of the smart watch.

independently. Sensors such as the BME280, BH1750, and DS3231 are connected via a shared I2C bus, allowing precise measurement of environmental parameters with minimal GPIO usage. Similarly, the CO₂ sensor MH-Z19B communicates through UART, providing calibrated measurements after a manual zero-point calibration procedure to ensure long-term accuracy.

Dynamic memory management is critical for the system due to the use of graphical user interfaces and real-time audio processing. To optimize memory utilization, the software relies on the PSRAM of the ESP32-S3, overriding default allocation operators to direct memory-intensive objects and graphical buffers to the external RAM. This configuration, combined with the LVGL graphics library, reduces the load on the main RAM while allowing smooth rendering of user interface elements, including fonts, images, and interactive widgets. The resulting efficiency supports complex graphical operations without compromising responsiveness or stability.

While external PSRAM greatly increases the available heap, it exhibits higher access latency and lower sustained bandwidth than the ESP32-S3's internal SRAM. In this device configuration, external PSRAM is clocked at 80 MHz over MSPI, whereas on-chip SRAM runs at core speed up to 240 MHz. In embedded HMI/audio workloads, this may manifest as reduced GUI responsiveness or, under contention, audio glitches if time-critical buffers are placed off-chip. To prevent such regressions, the design adopts a split-placement policy and buffer choreography: (i) time-critical data structures - I²S DMA rings, the STFT window and working arrays for feature extraction, and the TFLM tensor arena - are pinned to internal RAM using capability-qualified allocation; (ii) bulky, immutable GUI assets (fonts, images, theme resources) are kept in PSRAM; (iii) LVGL employs two small draw buffers in internal RAM (line/strip buffering) with partial, region-based flushes rather than full-frame framebuffers; (iv) display transfers and PSRAM fetches are scheduled via DMA and kept sequential to remain cache-friendly; (v) GUI and audio execute in separate RTOS tasks

with priorities chosen to guarantee deadlines on the audio path. In practice, this arrangement sustained ≥ 25 Hz GUI refresh with no I²S underruns during WWD and command capture, while preserving the RAM headroom provided by PSRAM. Additionally, RGB565 color depth is used, and full-screen alpha-blended layers are avoided to reduce memory bandwidth.

The RTOS orchestrates the execution of multiple concurrent tasks, assigning priorities to ensure that time-sensitive operations, such as voice recognition and sensor data acquisition, are executed promptly, while background tasks, including cloud synchronization and web server communications, are performed without interfering with real-time processes. Integration with cloud services enhances system functionality: Firebase provides real-time data storage and synchronization for environmental parameters and Pomodoro session records, Wit.AI enables natural language understanding for voice commands, and NTP ensures accurate timekeeping. Additionally, a web server hosted on an external platform allows remote monitoring of device status and user data, extending the clock's utility beyond local interaction.

User interface management is closely tied to the underlying software architecture (see **Fig. 2**). Touchscreen input is calibrated through software routines to accurately map analog-to-digital converter signals to screen coordinates, ensuring precise user interaction. The ILI9341 display, interfaced via SPI, benefits from DMA transfers that enable rapid graphical updates, while audio input and output through the INMP441 microphone and MAX98357 amplifier are processed via I²S with minimal latency. The combination of real-time audio processing, touchscreen control, and graphical rendering creates an interactive and responsive system, capable of simultaneously handling multiple streams of input and output without noticeable delays.

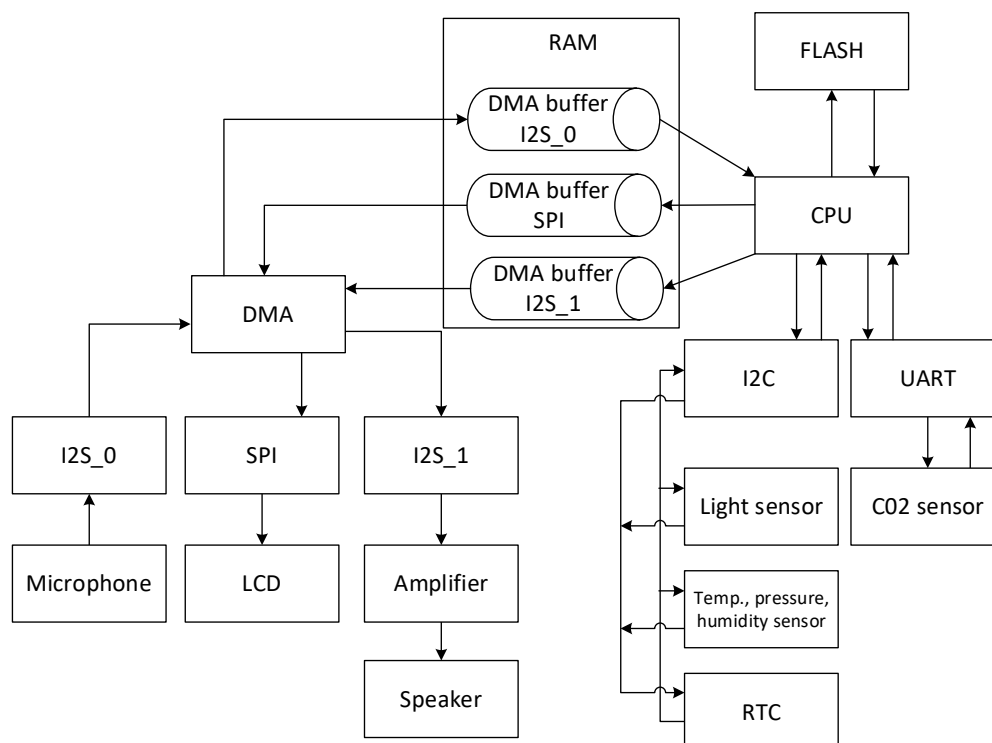


Fig. 2. DMA interaction with peripheral interfaces.

Finally, persistent data storage is implemented using the SPIFFS file system on the onboard Flash memory, which organizes system resources, audio files, and machine learning models for wake-word detection. Memory partitioning and the dual-application OTA update mechanism ensures reliability and ease of maintenance, allowing the firmware to be updated safely while preserving critical system data.

Overall, the software architecture combines real-time task management, efficient memory allocation, peripheral abstraction, and cloud connectivity to create a robust, adaptive, and interactive embedded system. This design enables the smart clock to operate seamlessly across multiple modalities, integrating environmental monitoring, voice control, graphical display, and remote connectivity into a coherent and reliable user experience.

Software architecture development.

The software architecture of the smart watch is built on FreeRTOS, enabling concurrent execution of independent tasks (threads) that share the same address space. Object-oriented programming (OOP) is employed to structure the codebase, encapsulating different functionalities such as sensor data management, voice interaction, and user interface operations into separate classes [19]. This design promotes modularity, code reuse, and ease of maintenance.

The system includes multiple tasks, each responsible for a specific operation. These tasks handle real-time updates of the display, processing of user commands, management of Pomodoro sessions, sensor data acquisition, audio input/output, and network communication. Two periodic timers coordinate scheduled operations such as hourly logging and Pomodoro timing. Synchronization mechanisms, including queues, semaphores, and task notifications, ensure safe data exchange and prevent conflicts when accessing shared resources. Task creation and management are handled through FreeRTOS APIs, with some tasks pinned to specific cores to optimize parallel execution across the ESP32's dual-core processor.

The task-level architecture is illustrated in **Fig. 3**, showing the interaction of tasks, timers, and synchronization objects. This diagram highlights how FreeRTOS enables concurrent execution, allowing the system to respond in real time to voice commands, sensor readings, and user interactions while maintaining efficient resource usage.

At the object level, the software is organized into classes representing key components of the system, including the Pomodoro timer, display manager, environment analysis, command handler, speaker, and LED controller. **Fig. 4** presents a simplified class diagram, showing relationships such as composition, aggregation, inheritance, and usage. This structure ensures modularity, facilitates testing, and supports extension of the system's functionality without disrupting existing components.

Voice interaction. The voice interface follows a two-tier Edge–Cloud pipeline that partitions low-latency detection from semantic understanding. The edge tier remains always on and performs wake-word detection (WWD) directly on the microcontroller to guarantee immediate responsiveness, while the cloud tier is invoked only for large-vocabulary speech recognition and intent extraction. An overview of the pipeline and data flow is shown in **Fig. 5**.

Audio is acquired from a digital MEMS microphone over I²S and buffered in short, overlapping frames. Each frame is windowed and transformed by a short-time Fourier transform (STFT); the resulting magnitude spectra are mapped to log-mel spectrograms that compactly capture phonetic structure and mitigate amplitude variance (illustrative example in **Fig. 6**). This time-frequency representation is well suited to image-like inference and provides a stable input for keyword spotting on embedded hardware.

Wake-word detection is performed by a lightweight CNN-based keyword-spotting (CNN-KWS) model trained on Google Speech Commands with the target wake phrase “Marvin,” augmented by “silence” and “unknown” classes. Training proceeds for a fixed number of epochs with standard augmentations (time shift, additive noise) to improve

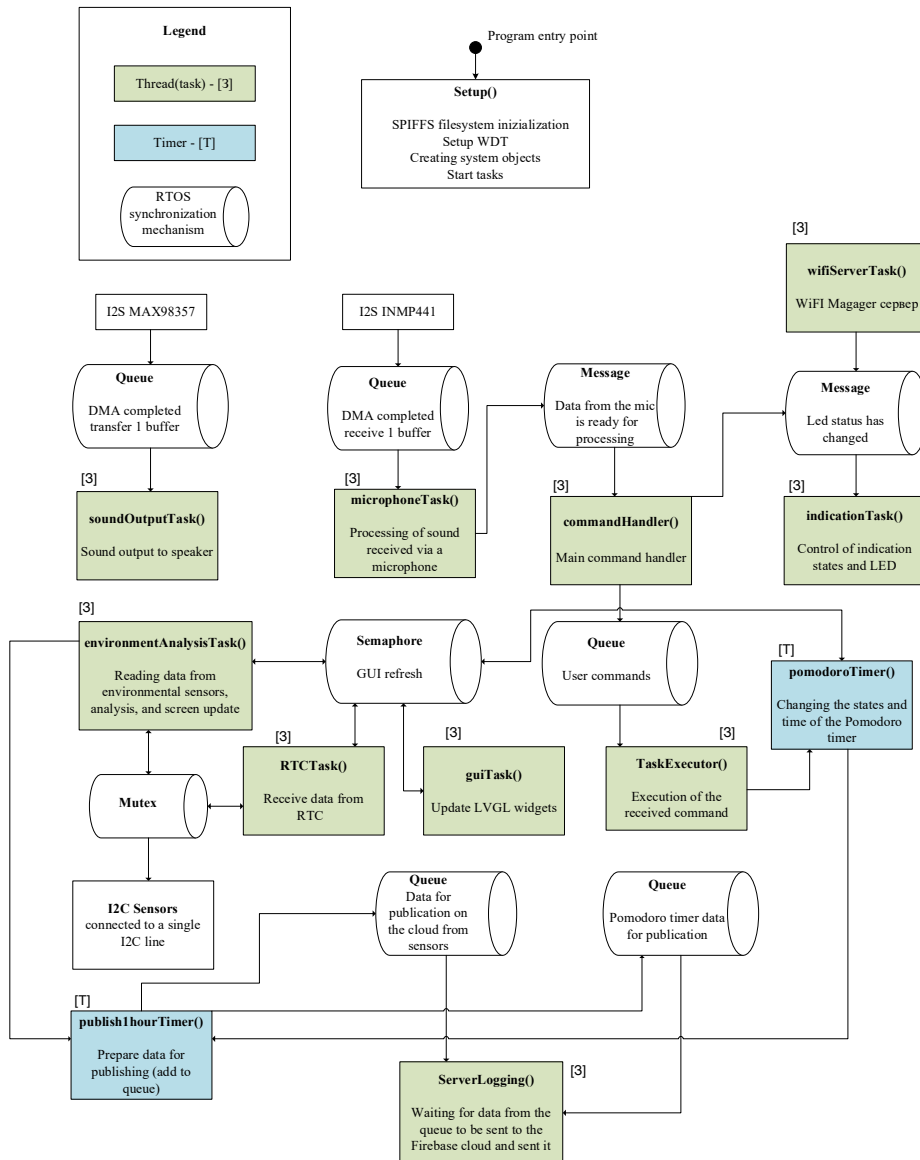


Fig.3. FreeRTOS architecture of the smart watch.

robustness. The trained network is quantized to 8-bit and deployed with TensorFlow Lite Micro as a C array linked into firmware, ensuring a small memory footprint and deterministic execution on the MCU. During inference, posterior smoothing and a calibrated decision threshold reduce both false rejections and false activations. A successful WWD event opens a bounded command window, during which the user utterance is captured for downstream understanding.

Command interpretation uses a selective processing strategy. Essential device operations (e.g., timer and mode controls) are matched locally by compact grammar to preserve functionality under variable connectivity. When network conditions are adequate, the utterance is forwarded over HTTPS to a cloud service (Wit.ai) for ASR/NLU; the response is returned as a JSON structure containing an intent label and typed entities (e.g., a duration), enabling flexible phrasing without firmware changes. This Edge–Cloud division focuses on on-device availability while leveraging cloud-scale language models for broader semantics.

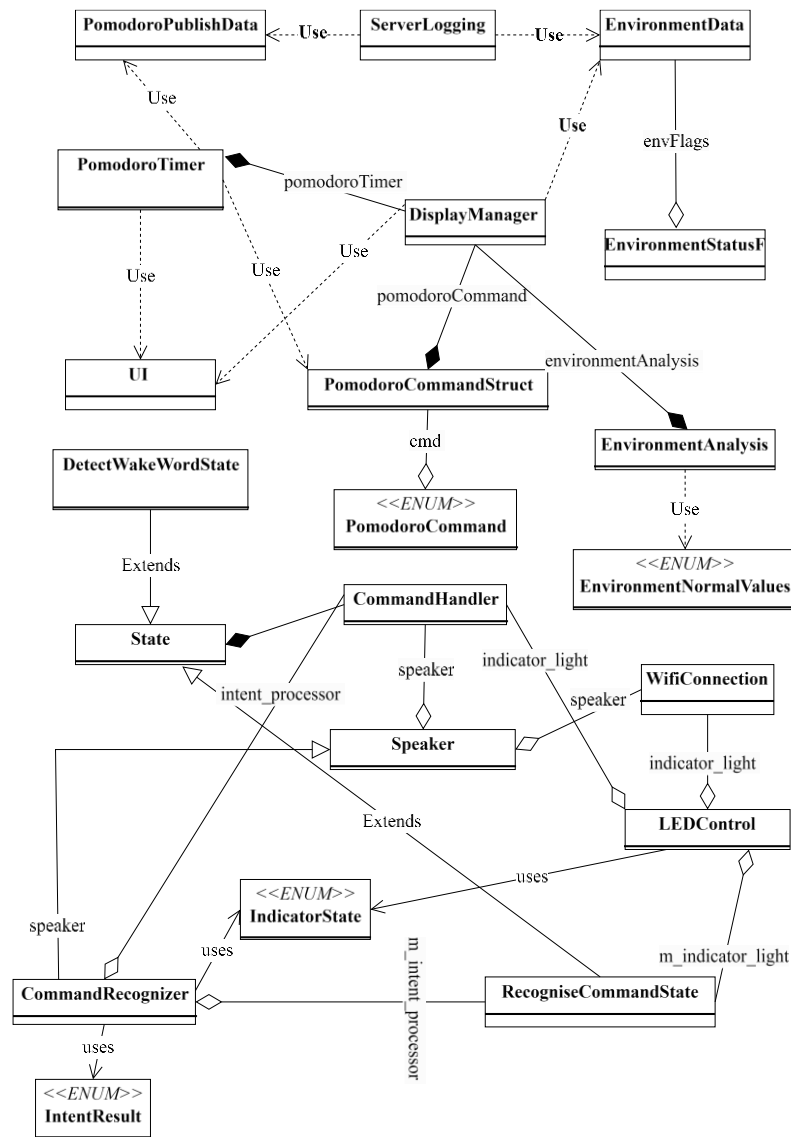


Fig.4. Class diagram of the smart watch.

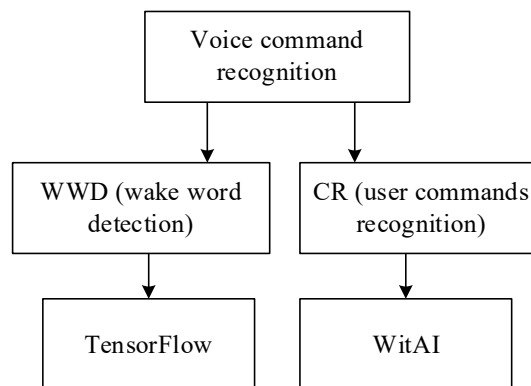


Fig.5. Voice recognition system structure.

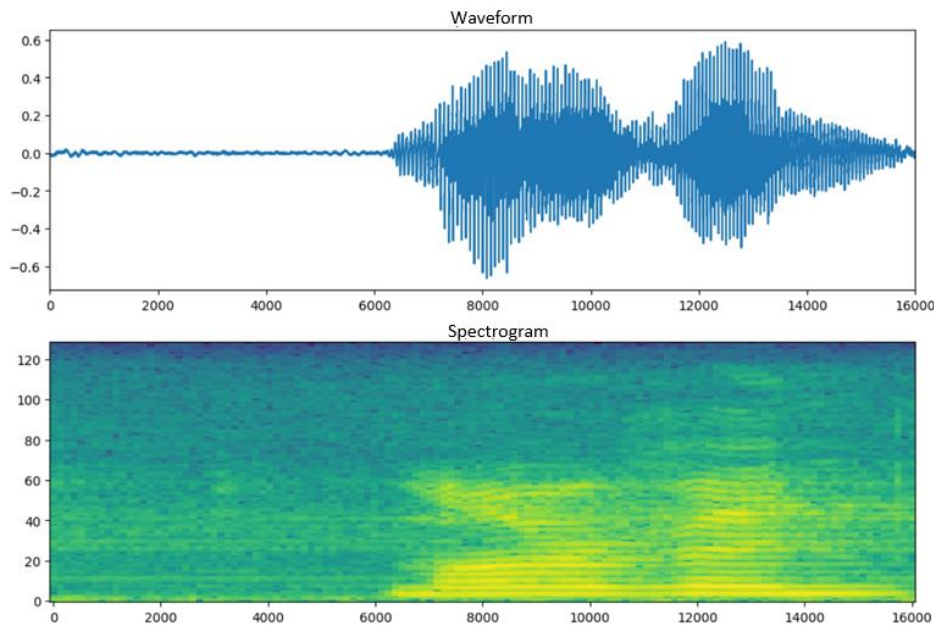


Fig. 6. Log-mel spectrogram of 'Marvin' word.

Run-time control is organized by a finite-state machine (FSM) that sequences the assistant through listening and command handling. In the WWD state, the edge continuously analyzes incoming audio; a detection triggers a transition to the CMD state, which manages utterance capture, local matching, or cloud submission, and action dispatch. On completion—or on timeout or error—the FSM returns to WWD. This explicit state structure simplifies timing, error recovery, and user-visible behavior; the state diagram is shown in Fig. 7.

User feedback combines non-intrusive auditory cues and visual status. Short tones and speech prompts are rendered through a Class-D amplifier and speaker to indicate activation, confirmation, and error conditions. In parallel, the LVGL-based touch GUI presents lightweight indicators for listening/processing states and command outcomes, and aligns voice interaction with task-management views (e.g., Pomodoro status) without forcing context switches. Together, these elements yield a responsive, resource-aware assistant: on-device CNN-KWS over STFT/log-mel features enables reliable activation; the FSM cleanly separates listening and action phases; and selective delegation to cloud NLU provides semantic breadth with minimal impact on latency or energy (Figs. 5–7).

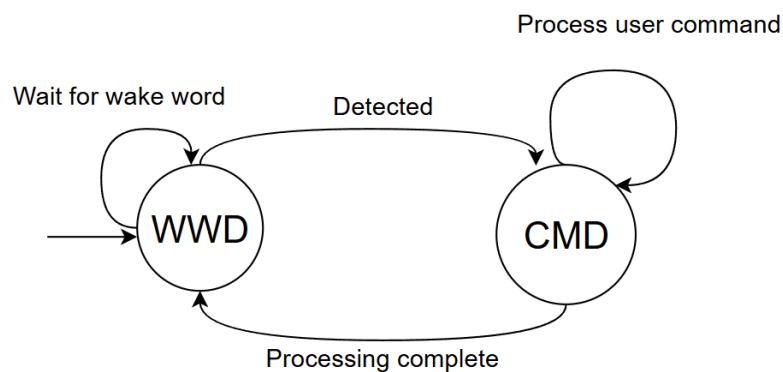


Fig.7. Voice assistant Finite State Machine.

Testing and Measurements. Resource usage on ESP32-S3 was 28.7% of RAM (93,908/327,680 bytes) and 46.3% of flash (2,185,093/4,718,592 bytes); the breadboarded prototype is shown in **Fig. 8**.

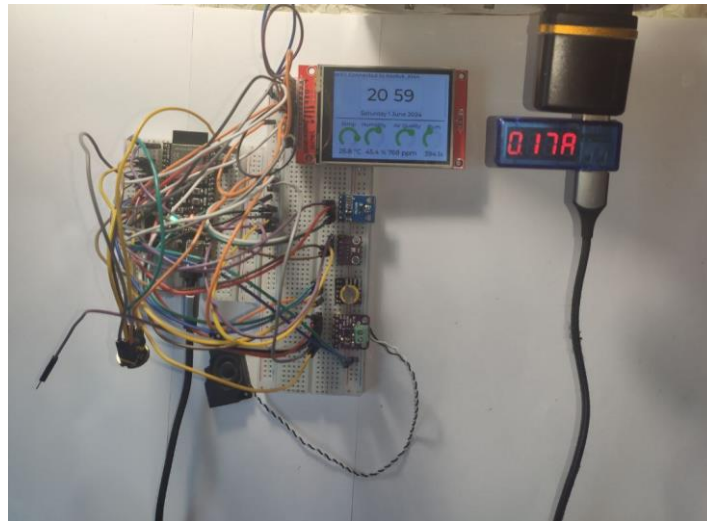


Fig.8. Smart clock on breadboard.

Average power draw measured with a Keweisi KWS-V20 USB tester was 0.17 A at 5.2 V (≈ 0.88 W). Wake-word testing in a quiet office (five participants; eight attempts each, 40 trials) yielded 36 correct activations ($\approx 90\%$ overall; per-participant 85–95%); a separate distance sweep (1–3 m) showed the expected decline, with 16/20 correct at 3 m. After activation, the assistant captured user utterances within a bounded 4-s command window with ($\approx 97\%$ overall); The system maintained concurrent sensing and GUI operation without audio dropouts during these tests.

CONCLUSION

This work demonstrates an edge–cloud architecture for a desktop smart-clock platform that unifies voice interaction, microclimate monitoring, and Pomodoro-based task management on MCU-class hardware. On the edge, a quantized CNN keyword-spotting model (TensorFlow Lite Micro) operating on STFT-derived log-mel spectrograms enables reliable wake-word detection, coordinated by an OOP design on FreeRTOS with a finite-state machine for listening and command handling. The audio chain (MEMS microphone over I²S with Class-D output) and the LVGL touch GUI maintained real-time responsiveness, with GUI refresh ≥ 25 Hz and a bounded 4s post-activation command window. Environmental sensing—including NDIR CO₂, temperature, humidity, and illuminance—ran at a 2 s cadence without perturbing the audio pipeline, while threshold-driven voice/visual alerts improved awareness of adverse conditions. Wi-Fi onboarding via an embedded web server simplified provisioning, and hourly cloud uploads produced coherent, time-aligned datasets for longitudinal analysis of environment and work sessions. Empirically, at an operating point tuned to FAR < 1 activation per hour, the wake-word detector achieved TPR $\approx 90\%$ (FRR $\approx 10\%$) with median detection latency < 200 ms in quiet-to-moderate office noise, and the hybrid edge–cloud flow preserved interactive latency by executing essential commands locally and delegating flexible, large-vocabulary understanding to cloud NLU when connectivity permitted. Collectively, the results validate that careful partitioning across edge and cloud, combined with RTOS-based concurrency and lightweight on-device ML, can deliver a responsive, energy-aware, and functionally

complete assistant on resource-constrained hardware. Future work includes expanding offline NLU for broader command coverage, personalizing thresholds and models using on-device adaptation, applying energy-aware duty cycling, and integrating with wider IoT ecosystems for context-aware scheduling and actuation.

ACKNOWLEDGMENTS AND FUNDING SOURCES

This work was supported by the Ministry of Education and Science of Ukraine (project No. 0125U001883).

COMPLIANCE WITH ETHICAL STANDARDS

The authors declare that they have no competing interests.

AUTHOR CONTRIBUTIONS

Conceptualization, [D.K., H.K.]; methodology, [D.K.]; investigation, [D.K.]; writing – original draft preparation, [D.K., H.K.]; writing – review and editing, [D.K., H.K.]; visualization, [D.K.].

All authors have read and agreed to the published version of the manuscript.

REFERENCES

- [1] Shafik, W. (2024). Smart devices and Internet of Things for sustainable energy. In *Advanced Technology for Smart Environment and Energy* (pp. 67-93). Cham: Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-50871-4_5
- [2] Fakhruddin, H. F., Saadh, M. J., Khan, S., Salim, N. A., Jhamat, N., & Mustafa, G. (2025). Enhancing smart home device identification in WiFi environments for futuristic smart networks-based IoT. *International Journal of Data Science and Analytics*, 19(4), 645-658. <https://doi.org/10.1007/s41060-023-00489-3>
- [3] Yedilkhan, D., & Smakova, S. (2024). Machine Learning Approaches for Smart Home Device Recognition from Network Traffic. *Procedia Computer Science*, 231, 709-714. <https://doi.org/10.1016/j.procs.2023.12.157>
- [4] Masoumian Hosseini, M., Masoumian Hosseini, S. T., Qayumi, K., Hosseinzadeh, S., & Sajadi Tabar, S. S. (2023). Smartwatches in healthcare medicine: assistance and monitoring; a scoping review. *BMC Medical Informatics and Decision Making*, 23(1), 248. <https://doi.org/10.1186/s12911-023-02350-w>
- [5] Alzahrani, S., Nadershah, M., Alghamdi, M., Baabdullah, R., Bayoumi, M., Bawajeih, O., ... & Bayoumi, A. (2025). The use of Apple smartwatches to obtain vital signs readings in surgical patients. *Scientific Reports*, 15(1), 1-10. <https://doi.org/10.1038/s41598-024-84459-0>
- [6] Navalta, J. W., Montes, J., Bodell, N. G., Salatto, R. W., Manning, J. W., & DeBeliso, M. (2020). Concurrent heart rate validity of wearable technology devices during trail running. *Plos one*, 15(8), e0238569. <https://doi.org/10.1371/journal.pone.0238569>
- [7] Meana-Llorián, D., García, C. G., G-bustelo, B. C. P., Lovelle, J. M. C., & Garcia-Fernandez, N. (2017). IoFClima: The fuzzy logic and the Internet of Things to control indoor temperature regarding the outdoor ambient conditions. *Future Generation Computer Systems*, 76, 275-284. <https://doi.org/10.1016/j.future.2016.11.020>
- [8] Smits, E. J., Wenzel, N., & de Bruin, A. (2025). Investigating the Effectiveness of Self-Regulated, Pomodoro, and Flowtime Break-Taking Techniques Among Students. *Behavioral Sciences*, 15(7), 861. <https://doi.org/10.3390/bs15070861>
- [9] Renacido, J. M. D., Mayordo, E. L., & Biray, E. T. (2025). A Comparative Study Between Pomodoro and Flowtime Techniques Among College Students. *International Journal of Multidisciplinary: Applied Business and Education Research*, 6(8), 3953-3973. <https://doi.org/10.11594/ijmaber.06.08.17>

- [10] Santiago, C., & Gurat, M. (2023). The Effect of Pomodoro Technique on Student Mendelian Genetics Concept Mastery during Synchronous Remote Learning. *International research journal of management, IT and social sciences*, 10(4), 233-243. <https://doi.org/10.21744/irjmis.v10n4.2287>
- [11] Litayem, N. (2024, December). Scalable smart home management with ESP32-S3: A low-cost solution for accessible home automation. *2024 International Conference on Computer and Applications (ICCA)*, 1-7. <https://doi.org/10.1109/ICCA62237.2024.10927887>
- [12] Ouyang, L., Zhu, G., & Zhang, Y. (2025, June). Design of Deepseek Big Model Intelligent Voice Assistant Based on ESP32 Microcontroller. In *2025 IEEE International Conference on Pattern Recognition, Machine Vision and Artificial Intelligence (PRMVAI)*, 1-5. <https://doi.org/10.1109/PRMVAI65741.2025.11108406>
-

БАГАТОФУНКЦІОНАЛЬНИЙ РОЗУМНИЙ ГОДИННИК ДЛЯ ГОЛОСОВОЇ ВЗАЄМОДІЇ ТА АДАПТИВНОГО ПЛАНУВАННЯ ЗАВДАНЬ

Дмитро Козлюк, Галина Клим

Національний університет «Львівська політехніка»,
вул. . Бандери 12, 79013 м. Львів, Україна

АНОТАЦІЯ

Вступ. Сучасні смарт-пристрої зазвичай обмежуються ізольованими функціями, такими як відображення часу, моніторинг параметрів середовища чи хмарні голосові асистенти. Така фрагментація не забезпечує єдиного комплексного рішення для підвищення продуктивності у сучасних робочих просторах, де важливими є як мікрокліматичні умови, так і ефективне управління часом. Системи, що працюють виключно у хмарі, страждають від затримок та залежності від підключення, тоді як повністю локальні рішення обмежені ресурсами. Це дослідження має на меті розробку багатофункціонального «розумного годинника», що об'єднує моніторинг довкілля, голосову взаємодію та планування завдань на єдиній платформі, призначеній для підвищення комфорту, концентрації та ефективності.

Методи. Система реалізована за принципом Edge–Cloud архітектури: ESP32-S3 на рівні «краю» виконує критичні до затримок функції під FreeRTOS з об'єктно-орієнтованим підходом до проектування. Аудіосигнал із MEMS-мікрофона (I²S) розбивається на вікна та перетворюється за допомогою короткочасного FFT у log-mel спектрограми; квантована згорткова нейронна мережа (TensorFlow Lite Micro) виконує локальне розпізнавання ключового слова для активації. Після цього команди передаються у Wit.ai для ASR/NLU. Аудіовихід реалізований через підсилювач класу D та динамік. Моніторинг параметрів довкілля охоплює температуру, вологість, освітленість та рівень CO₂ (NDIR-сенсор), із фільтрацією даних, які відображаються у подієво-орієнтованому сенсорному інтерфейсі LVGL та періодично завантажуються у Firebase для подальшого аналізу.

Результати. Розпізнавач ключового слова на основі CNN досяг приблизно 90% точності активації в умовах тихого та помірного офісного шуму при рівні хибних спрацювань <1 раз/год і частоті пропусків близько 10%. Медіанна затримка активації не перевищувала 200 мс після накопичення достатнього контексту. При RTT ≤100 мс хмарні сервіси ASR/NLU забезпечували загальну затримку від активації до розпізнавання наміру на рівні ≈1–1,5 с. Паралельний моніторинг довкілля з інтервалом 2 с не заважав аудіопроцесу, графічний інтерфейс зберігав частоту оновлення 25 Гц, а після активації система відкривала 4-секундне вікно для голосових команд користувача. Налаштування Wi-Fi здійснювалося через вбудований веб-сервер, а щогодинні завантаження даних у хмару були стабільними. Порогові голосові та

візуальні сповіщення підвищували обізнаність про стан мікроклімату, тоді як інтегровані цикли Pomodoro підтримували концентрацію без потреби у сторонніх інструментах.

Висновки. Запропонована платформа ефективно об'єднує голосову взаємодію, структуроване управління часом та моніторинг мікроклімату в доступному за ціною пристрої. Гібридна архітектура обробки мовлення забезпечує баланс між низькою затримкою та гнучкістю, а використання FreeRTOS гарантує безпечне та паралельне функціонування підсистем сенсорики, GUI, мережі та аудіо. Система становить практичну основу для майбутніх розширень, зокрема адаптивного планування, глибшої IoT-інтеграції, локального NLU-резерву та енергоефективного керування робочими циклами.

Ключові слова: Обчислення на рівні краю; RTOS; NDIR; Pomodoro; вбудована голосова взаємодія; моніторинг мікроклімату.

Received / Одержано
23 September, 2025

Revised / Доопрацьовано
06 October, 2025

Accepted / Прийнято
10 October, 2025

Published / Опубліковано
31 October, 2025