TED UNIVERSITY

CMPE 491 – Senior Project I

Fall 2025

AKKE – High Level Design Report

(SMART COMMAND AND CONTROL GLOVE)

Webpage: https://ilter-akke.github.io/website/

Name Surname Department

Abdullah ESİN (CMPE)

Berk ÇAKMAK (CMPE)

Ömer Efe DİKİCİ (CMPE)

Şevval KURTULMUŞ (EEE)

Supervisors: Ali BERKOL and Hüseyin Uğur YILDIZ

Jury Members: Hakkı Gökhan İLK and Mehmet Evren COŞKUN

Table of Contents

1. Introduction

In modern tactical and field operations, intra-team command delivery is typically achieved through verbal communication (radio/shouting) and visual hand signals. However, these methods rely on two critical assumptions: (i) speaking is always safe and feasible, and (ii) team members can maintain line-of-sight with the person issuing commands. In real operational environments, secrecy requirements, smoke/darkness, physical obstacles, rugged terrain, and harsh weather conditions frequently break these assumptions. As a result, commands may be delayed or not delivered at all, weakening disciplined and timely command-and-control.

To address this problem, this project introduces AKKE, a wearable command-and-control smart glove system. AKKE aims to detect predefined hand gestures and convert them into audio-based commands that can be broadcast wirelessly to multiple receivers. The system integrates flex sensors (to measure finger bending) and an IMU sensor (to capture hand orientation and motion). The sensor stream is processed in real time on an embedded ESP32-S3 unit mounted on the glove. Once a gesture is recognized, it is mapped to a corresponding command, and the associated audio file stored on a microSD card is played via DFPlayer Mini and transmitted through an RF-based communication module. This design enables team members to receive commands simultaneously without requiring speech or visual contact.

AKKE is primarily intended for military and security-oriented field scenarios where speaking is risky or impractical and where visual hand signaling may fail. The overall approach combines wearable sensing, embedded software, and gesture classification methods to produce a prototype that can operate under realistic constraints and conditions.

1.1 Purpose of the System

The primary purpose of AKKE is to provide instant and reliable command delivery among field teams without spoken communication, while supporting requirements for secrecy, speed, and operational robustness. The system is designed for scenarios in which speaking may compromise position, were noise or protective equipment limits verbal communication, or where hand signals cannot be seen due to limited visibility or obstructed line-of-sight.

To achieve this purpose, AKKE provides the following end-to-end functionality:

- Collects and processes flex and IMU sensor data from the glove in real time.
- Recognizes at least five distinct gestures and targets a practical recognition accuracy level (approximately 75–80%) under realistic operating conditions.
- Maps each recognized gesture to a predefined command set (e.g., “Advance”, “Hold Position”, “Retreat”).
- Plays the command’s corresponding audio file from microSD storage and transmits it via RF to multiple receivers simultaneously, enabling coordinated action without direct visual contact.
- Aims to keep end-to-end response time (gesture completion → command broadcast) below 1 second for timely command-and-control.

In this context, AKKE treats gesture recognition not as an isolated objective, but as a component of a complete operational pipeline: sensing → classification → command mapping → audio generation → multi-receiver wireless broadcast.

1.2 Design goals

The design goals of AKKE translate previously identified requirements and constraints into high-level objectives that guide architectural decisions, subsystem boundaries, and key technology choices (sensing, embedded processing, audio generation, and wireless broadcasting). The overall goal is to deliver a functional, reliable, and field-appropriate wearable system that enables silent command-and-control through gesture recognition.

G1 — Operational Effectiveness (Mission Fit)

- Enable command delivery without spoken communication and without requiring line-of-sight between team members.
- Map recognized gestures to a clear, non-overlapping command set to reduce ambiguity during operation.
- Support simultaneous command reception by multiple receivers to maintain team synchronization.

G2 — Real-Time Performance

- Keep the end-to-end response time (gesture completion → classification → audio trigger → broadcast) below 1 second.
- Ensure smooth operation under continuous flex + IMU sensor streaming without noticeable lag or system stalls.
- Provide fast and clear user feedback via the RGB LED indicating only whether the command was successfully transmitted ("sent") or not transmitted ("not sent"), to prevent unnecessary gesture repetition.

G3 — Recognition Quality and Robustness

- Reliably recognize at least five distinct gestures and associate them with operational commands.
- Target a practical recognition accuracy of ≥ 75–80% under realistic conditions.
- Reduce the impact of sensor noise and user-to-user variation through calibration, preprocessing, and robust classification strategies.
- Reduce incorrect command transmission risk by keeping the decision logic stable and consistent; when recognition fails, the system must not broadcast any command and must indicate "not sent" via the LED.

G4 — Field Usability and Practical Durability

- Design the system (component selection, placement, enclosure/wiring) with field use in mind; however, explicitly state that resistance to factors such as dust, humidity, and temperature variation has not been fully validated by comprehensive environmental testing at this stage and remains a design intention.
- Target a minimum wireless communication range of ~10 meters (with potential improvements depending on RF module and antenna conditions).
- If recognition or transmission fails, the system must avoid broadcasting an incorrect command and must provide "not sent" feedback via the LED.

G5 — Usability, Learnability, and Ergonomics

- Ensure the glove is easy to wear and remove and does not significantly restrict hand movement.
- Prefer a headless physical UI over complex screens/menus: power switch, calibration button, and RGB LED status feedback.
- Keep calibration simple and repeatable, enabling preparation without technical expertise in the field.

G6 — Maintainability and Extensibility

- Structure the software as modular components (sensor acquisition, preprocessing, gesture recognition, command mapping, audio playback, RF broadcast) to support:
  - adding/removing gestures,
  - updating the recognition model,
  - changing command–audio mappings,
  - adjusting communication parameters.
- Keep hardware components (battery, sensors, modules) reasonably accessible for repair, replacement, and iterative prototyping.

G7 — Power Architecture and Practical Deployment

- Ensure the selected power solution is practical for field use (feasibility of rechargeable/replaceable options) and that the prototype includes a power architecture that is operationally workable.
- Consider reproducibility/manufacturability at a prototype level: clean wiring, reasonable module placement, and repeatable assembly.

1.3 Definitions, acronyms, and abbreviations

- **AKKE**: The wearable smart-glove system developed in this project for gesture-based command transmission.
- **ESP32-S3**: The embedded microcontroller platform used as the main processing unit.
- **IMU (Inertial Measurement Unit)**: Sensor module used to capture hand motion and orientation.
- **Flex Sensor**: Bend sensor used to measure finger bending.
- **DFPlayer Mini**: Audio playback module used to play stored command audio files.
- **microSD**: Removable memory card used to store command audio files.
- **RF (Radio Frequency)**: Wireless communication used to transmit commands from the glove to receivers.
- **Receiver Unit**: Device that receives RF transmissions and outputs/triggers the corresponding command audio.
- **RGB LED**: Status indicator; in this project it provides only "sent" / "not sent" feedback.
- **Headless UI**: Screenless user interface based on physical controls/indicators (switch, button, LED).
- **Calibration**: User-triggered procedure to align/normalize sensor readings for reliable recognition.
- **Latency (End-to-End Response Time)**: Time from completing a gesture to transmitting the corresponding command.

1.4 Overview

This report transforms the analysis results of the AKKE project into a high-level (system) design. AKKE is a wearable command-and-control solution that collects flex and IMU sensor data, processes it in real time on an ESP32-S3 to recognize predefined gestures, maps each recognized gesture to a command, and delivers the command as an audio message by playing a corresponding microSD/DFPlayer file and broadcasting it via RF to multiple receivers. Instead of a screen, the system uses a headless UI (power switch, calibration button, RGB LED); the LED provides only “sent / not sent” feedback to the user.

The remainder of the report explains not only what the system does, but also why the design is structured this way, clarifying the end-to-end architecture. Specifically:

- **Section 2** summarizes any existing architecture (if applicable) and the limitations of the current approach.
- **Section 3** presents the proposed software/system architecture, including subsystem decomposition, hardware–software mapping (deployment perspective), the persistent data management strategy (e.g., storage of command audio files), global control flow, and handling of boundary conditions. (If environmental durability is described as a design intent, the report explicitly states whether it has been validated by comprehensive environmental testing.)
- **Section 4** describes the services/interfaces provided by each subsystem and how subsystems interact at a high level.
- **Section 5** provides a glossary of key terms used throughout the report.
- **Section 6** lists references (e.g., Bruegge & Dutoit and earlier project reports).

With this structure, the reader can follow AKKE’s components, responsibilities, and the complete pipeline from gesture execution to command broadcast, in a way that is directly connected to the project’s design goals.

2. Current software architecture

2.1 Architectural Style: Layered Architecture

The AKKE software system follows a Layered Architecture to ensure modularity, hardware independence, and maintainability. Given the use of the ESP32-S3, this structure allows for efficient separation of real-time sensor processing from high-level gesture classification.

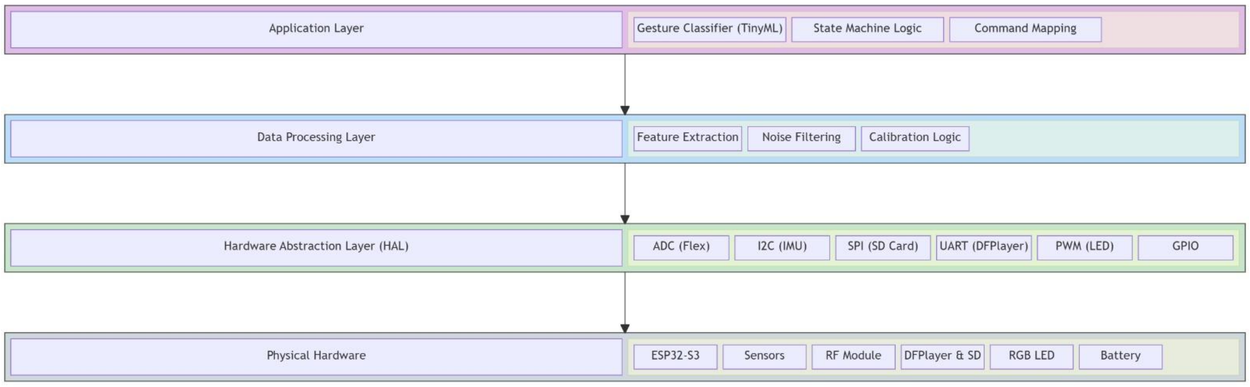

Figure 1: Multi-layered system architecture of AKKE.

2.2 Software Component Diagram

The system is divided into five primary functional modules that interact through the Glove Device coordinator:

A. Sensor Manager

- **Purpose:** Continuously samples data from flex sensors (0° to 180°) and the 3-axis IMU.
- **Mechanism:** Uses high-frequency timers to ensure a consistent sampling rate for accurate gesture recognition.

B. Gesture Classifier (TinyML Core)

- **Purpose:** Processes sensors feature representations to identify one of the five predefined tactical commands.
- **Logic:** Employs a confidence-based threshold; if confidence is low, it triggers an ErrorState instead of broadcasting.

C. Audio Engine

- **Purpose:** Plays pre-recorded command audio files stored on the microSD card using the DFPlayer Mini module.
- **Mechanism:** After a gesture is recognized and mapped to a command, the ESP32-S3 sends serial (UART) control commands to the DFPlayer Mini (e.g., select track, play, stop). Audio decoding and analog output are handled by the DFPlayer Mini, so the microcontroller is not responsible for audio streaming.

D. Broadcast Controller

- **Purpose:** Configures the transmitter via I2C.
- **Functions:** Handles frequency selection, power management, and RF modulation.

E. Headless UI Manager

- **Purpose:** Manages user interaction without a screen to maintain low cognitive load and light discipline.
- **Feedback:** Controls the RGB Status LED (e.g., Solid Blue for Idle, Blinking Red for Low Battery) and handles the Calibration Button.

3. Proposed Software Architecture

3.1 Overview

The proposed software architecture for AKKE transitions the conceptual models from the analysis phase into a structured system design centered on a Layered Architectural Style. The architecture is designed to handle real-time sensor data acquisition, complex gesture classification via TinyML, and synchronized wireless audio broadcasting. By decoupling low-level hardware abstractions from high-level application logic, the system ensures modularity, allowing for independent updates to the machine learning model or communication protocols without disrupting the core pipeline.

3.2 Subsystem Decomposition

The AKKE system is decomposed into five primary functional subsystems, each responsible for a specific stage of the command-and-control pipeline:

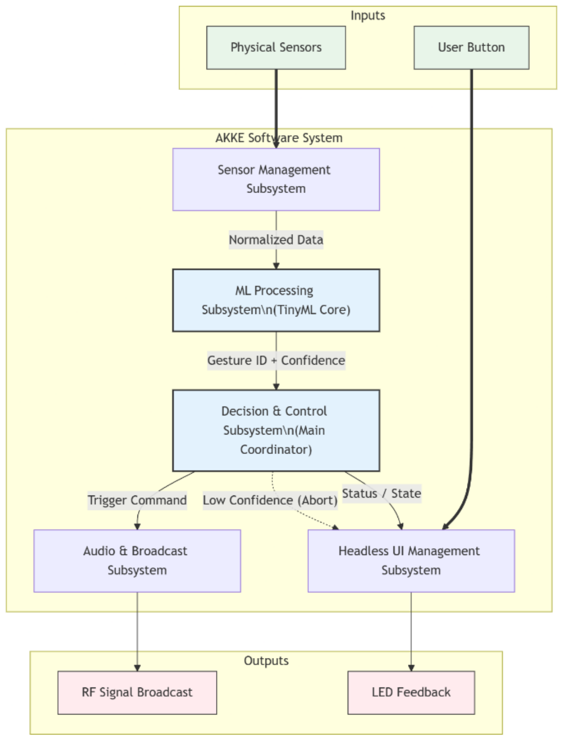

Figure 2: Subsystem Decomposition Diagram showing the logical data flow and interactions between functional subsystems.

- **Sensor Management Subsystem:** Responsible for continuous, high-frequency sampling and normalization of data from the flex and IMU sensors.
- **ML Processing Subsystem (TinyML Core):** Executes the gesture recognition model on the ESP32-S3 to classify hand movements with a focus on confidence-based validation .
- **Decision and Control Subsystem:** Acts as the system coordinator, managing state transitions and enforcing the "Accuracy Gate" to prevent incorrect command transmissions.
- **Audio and Broadcast Subsystem:** Handles the retrieval of command audio by sending UART commands to the DFPlayer Mini, which then outputs the analog audio signal to the RF module.
- **Headless UI Management Subsystem:** Manages user interactions through physical controls and provides status feedback via the RGB LED, maintaining low cognitive load for the operator.

3.3 Hardware/Software Mapping

The software components are mapped to the hardware platform to optimize performance and resource utilization:

- **ESP32-S3 Microcontroller:** Serves as the central processing hub, hosting the TinyML inference engine and the main control state machine, while offloading audio processing to the external module.
- **Peripheral Integration:** Sensor drivers utilize ADC (flex) and I2C (IMU) interfaces, while the audio playback is fully offloaded to the DFPlayer Mini via UART serial communication to feed the RF broadcast.
- **Deployment Diagram:**

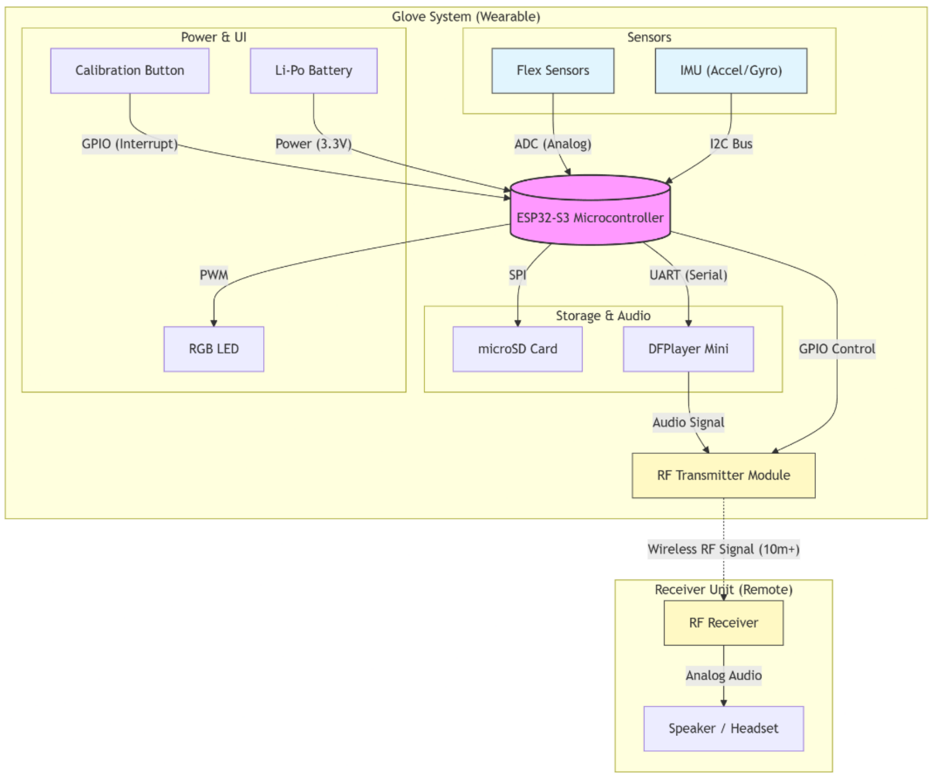

Figure 3: UML Deployment Diagram illustrating the physical connections between the ESP32-S3, sensors, storage, audio modules, and the RF transmission unit.

3.4 Persistent Data Management

Persistent data in AKKE is primarily managed via a **microSD card** interface:

- **Audio Storage:** Pre-recorded command audio files (e.g., "Advance," "Retreat") are stored in uncompressed WAV format to ensure immediate playback and broadcast without CPU-intensive decoding.
- **Configuration and Models:** The TinyML model weights and gesture-to-command mapping tables are stored in the ESP32-S3's internal flash memory, while user-specific calibration baselines can be optionally cached for rapid startup.

3.5 Access control and Security

Security and privacy are maintained through a multi-layered approach:

- **Signal Isolation:** By utilizing RF-based communication isolated from external Wi-Fi or cellular networks, the system prevents unauthorized remote access and data leakage.
- **Role-Based Interaction:** Access to system configurations and the "Update Gesture Set" use case is restricted to authorized maintenance personnel through physical hardware jumpers or secure serial interfaces.
- **Operational Integrity:** The "Accuracy Gate" serves as a safety mechanism, ensuring that only high-confidence recognitions result in a broadcast, thereby mitigating the risk of issuing false commands.

3.6 Global Software Control

The global control flow is governed by an **event-driven state machine** that ensures predictable behavior across all operational scenarios:

- **Lifecycle States:** The system transitions through *Booting*, *Idle/Ready*, *Collecting Data*, *Recognizing*, and *Transmitting* .
- **Event Handling:** Transitions are triggered by physical interrupts (e.g., the calibration button) or data-driven events (e.g., movement detection exceeding a threshold).
- **Synchronization:** The Main Controller Subsystem orchestrates the pipeline, ensuring that the Audio Engine and RF Subsystem are synchronized during the broadcast phase.

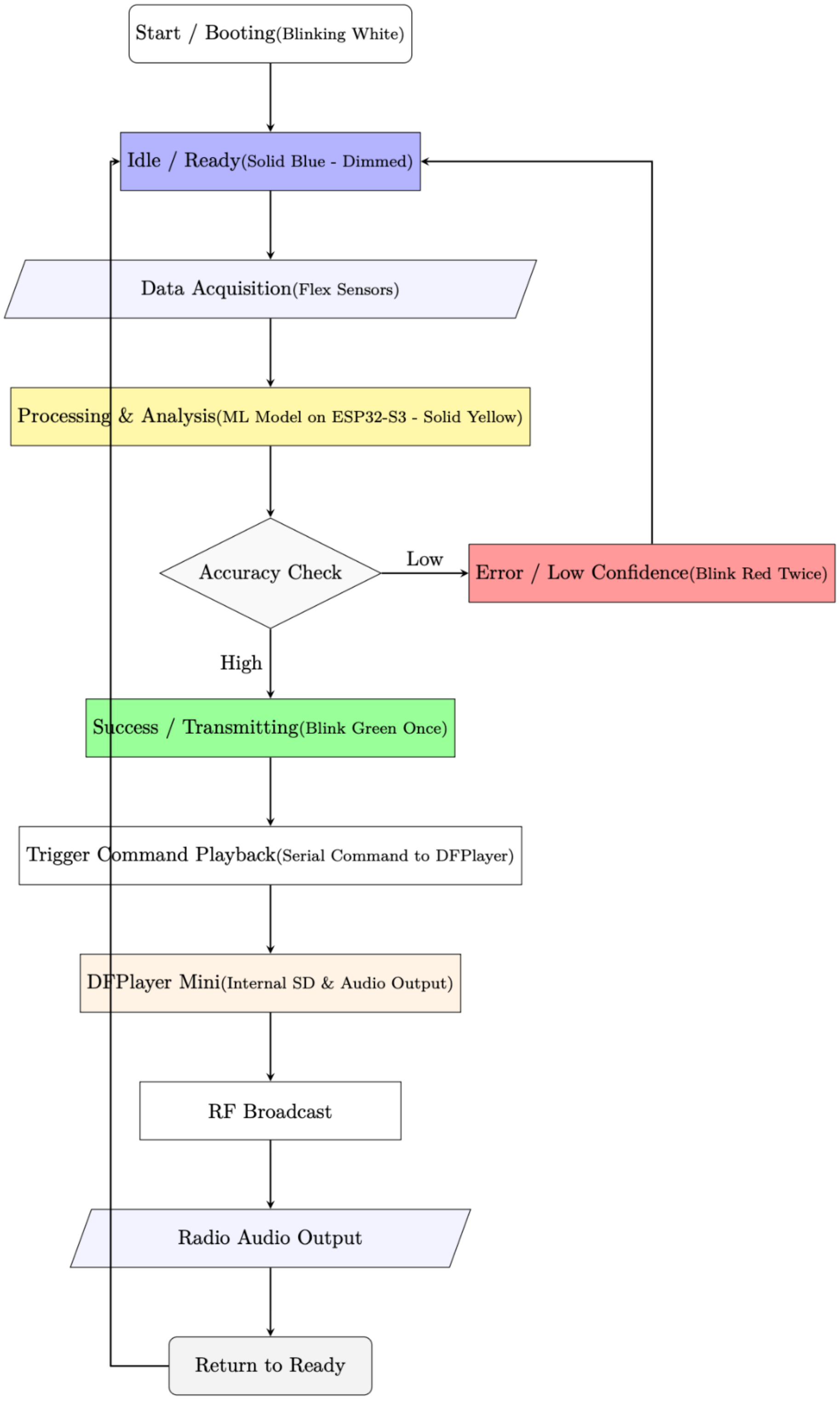
Start / Booting(Blinking White)
Idle / Ready(Solid Blue - Dimmed)
Data Acquisition(Flex Sensors)
Processing & Analysis(ML Model on ESP32-S3 - Solid Yellow)
Accuracy Check
Low
Error / Low Confidence(Blink Red Twice)
High
Success / Transmitting(Blink Green Once)
Trigger Command Playback(Serial Command to DFPlayer)
DFPlayer Mini(Internal SD & Audio Output)
RF Broadcast
Radio Audio Output
Return to Ready

Figure 4: Global Control Flowchart illustrating the system lifecycle, LED feedback mechanisms, and decision logic.

3.7 Boundary Conditions

The system is designed to handle exceptional and boundary cases to maintain reliability in the field:

- **System Initialization Failures:** If sensors or the microSD card fails to initialize during booting, the system enters an *ErrorState* and alerts the user via a persistent red LED signal.
- **Low Confidence/Recognition Failure:** When gesture recognition results fall below the 75-80% accuracy threshold, the system cancels the transmission and provides "not sent" feedback to prevent mission errors.
- **Resource Constraints:** In the event of a "Low Battery" condition, the system prioritizes critical status indicators over new command processing to prevent erratic behavior due to voltage drops.
- **Movement Timeouts:** If the "Collecting Data" state does not receive a buffer-full signal within a predefined window, the system automatically returns to *Idle* to conserve power.

4. Subsystem Services

This section describes AKKE subsystems as service providers, aligned with the end-to-end flow in the final flowchart (Booting → Ready → Sensor data acquisition → ADC → ML processing → Accuracy gate → Track resolve → DFPlayer playback (UART) → RF broadcast → Radio audio output → Return to Ready).

4.1 UI & Control Subsystem

Responsibility: Provide a headless user interface and status feedback.
Services:

- IndicateBooting() → LED **Blinking White**
- IndicateReady() → LED **Solid Blue (Dimmed)**
- IndicateProcessing() → LED **Solid Yellow**
- IndicateSuccess() → LED **Blink Green Once**
- IndicateLowConfidence() → LED **Blink Red Twice**

Inputs/Outputs: State decisions from the controller → LED output.
Flowchart mapping: The Booting/Ready/Processing/Success/Low-confidence LED steps.

4.2 Main Controller Subsystem

Responsibility: Orchestrate the full pipeline and return the system to the Ready state after each cycle.
Services:

- InitializeSystem() → prepares the system after power-on
- RunCycle() → Sensor data acquisition →ADC → inference → decision → (track resolve + DFPlayer play) → RF broadcast → receiver output
- ReturnToReady() → ends the cycle and waits for the next gesture

Inputs/Outputs: SensorData, gesture_id, accuracy → triggers or cancels the transmit pipeline + drives LED states.
Flowchart mapping: The entire main flow from Start to "Wait for next gesture."

4.3 Data Acquisition Subsystem (Flex & IMU)

Responsibility: Collect and package sensor data required for recognition.

Services:

- ReadFlexIMU() → raw sensor readings
- BuildSensorData() → window/frame formatted for ML input

Inputs/Outputs: Sensor readings → SensorData.

Flowchart mapping: "Collect/Acquire flex & IMU data."

4.4 ADC Conversion Subsystem (Flex Analog → Digital)

Responsibility: Convert the analog output of the flex sensors into digital samples and prepare them for the ML input pipeline. *(IMU data is already digital over I2C; the ADC step primarily applies to the flex channels.)*

Services:

- **InitADC()** → configure ADC (channel selection, attenuation, sampling settings)
- **SampleFlexADC()** → acquire raw ADC samples (single or multi-sample)
- **ProcessADC(samples)** → optional filtering/normalization → flex_digital_values

Inputs/Outputs: Flex analog voltage → flex_digital_values (digital readings).

Flowchart mapping: The "Analog-to-Digital Conversion (ADC)" block (after Data Acquisition, before ML Processing).

4.5 ML Processing Subsystem (ESP32-S3 Inference)

Responsibility: Run ML processing on ESP32-S3 and produce a gesture prediction with an accuracy/confidence value.

Services:

- RunInference(SensorData) → (gesture_id, accuracy)

Inputs/Outputs: SensorData → gesture_id + accuracy.

Flowchart mapping: "Process and analyze data with ML."

4.6 Decision / Accuracy Gate Subsystem

Responsibility: Apply the accuracy threshold decision; if the result is not reliable, stop the pipeline and return to Ready.

Services:

- CheckAccuracyThreshold(accuracy) → HighConfidence | LowConfidence
- HandleLowConfidence() → LED blinks red twice + ReturnToReady() (no DFPlayer playback and no RF broadcast)
- HandleHighConfidence() → success indication and transition into the audio transmission pipeline

Inputs/Outputs: accuracy → decision + next-step trigger/cancel.

4.7 Command Audio Retrieval Subsystem

Responsibility: Map the recognized gesture to the corresponding audio track on DFPlayer.

Services:

- ResolveAudio(gesture_id) → track_id

Inputs/Outputs: gesture_id → track_id.

4.8 DFPlayer Playback Control Subsystem (UART)

Responsibility: Trigger DFPlayer Mini to play the selected track via UART control

Services:

- TriggerPlayback(track_id) → analog_audio_out

Inputs/Outputs: track_id → analog_audio_out.

4.9 RF Broadcast Subsystem

Responsibility: Broadcast the DFPlayer analog audio output over RF

Services:

- BroadcastRF(analog_audio_out) → RF_signal

Inputs/Outputs: analog_audio_out → RF_signal

4.10 Receiver & Audio Output Subsystem

Responsibility: Receive RF transmission and output audio to speaker/headset.

Services:

- ReceiveRF() → received_audio;

OutputAudio(received_audio) → speaker/headset

5. Glossary

Term	Definition
AKKE	The wearable smart-glove system developed in this project for gesture-based command transmission and multi-receiver wireless audio broadcasting.
Access Control	Measures that restrict who can change configuration/model/gesture set settings (e.g., maintenance-only access).
Accuracy (Recognition Accuracy)	The practical performance level of gesture recognition targeted by the system (e.g., ~75–80% under realistic conditions).
Accuracy Gate	The control logic that blocks broadcasting when recognition confidence is below the threshold to avoid incorrect commands.
Accuracy Threshold	The minimum confidence/accuracy value required for a gesture prediction to be accepted and transmitted.
ADC (Analog-to-Digital Converter)	The hardware interface used to digitize analog flex sensor readings on the microcontroller.
Application Layer	The software layer containing core logic such as the TinyML classifier, state machine, and command mapping.
Audio & Broadcast Subsystem	The subsystem responsible for retrieving command audio from storage and transmitting it wirelessly via RF.
Audio Engine	The control component that triggers DFPlayer Mini playback via UART; audio decoding and analog output are handled by DFPlayer.
Audio Generation	The stage where the recognized gesture is turned into an audible command (via stored audio files).
Booting	The startup state where the system initializes sensors, storage, and modules before entering Ready/Idle.
Boundary Conditions	Exceptional/edge cases the system must handle (e.g., init failures, low confidence, low battery, timeouts).

Term	Definition
Broadcast Controller	The module that manages RF transmission behavior (e.g., configuration and initiating broadcast).
Calibration	A user-triggered procedure to align/normalize sensor readings to improve recognition reliability.
Calibration Button	Physical input used to trigger calibration, typically via a GPIO interrupt.
Command Mapping	The mapping from a recognized gesture to a predefined command and its corresponding audio file.
Command Set	The set of non-overlapping operational commands supported by the system (e.g., “Advance”, “Hold Position”, “Retreat”).
Confidence	The classifier’s certainty measure for a predicted gesture; used to decide whether to broadcast (“sent”) or abort (“not sent”).
Current Software Architecture	The existing/assumed architecture baseline described before the proposed design (Section 2).
Data Processing Layer	The layer responsible for normalization, filtering, and feature extraction from sensor streams.
Decision and Control Subsystem (Main Controller / Coordinator)	The subsystem that orchestrates pipeline stages and enforces the accuracy gate and state transitions.
Deployment Diagram (UML)	A UML diagram showing how software components are mapped onto hardware nodes (e.g., glove unit vs receiver unit).
DFPlayer Mini	The audio playback module used to play stored command audio files from a microSD card.
End-to-End Pipeline	The full operational chain: sensing → classification → command mapping → audio retrieval/playback → RF broadcast → receiver audio output.
End-to-End Response Time (Latency)	Time from completing a gesture to broadcasting the corresponding command; targeted to be below 1 second.

Term	Definition
Environmental Durability (Design Intention)	The intended resistance to dust/humidity/temperature variation, explicitly stated as not fully validated by comprehensive testing yet.
ErrorState	A failure state entered when critical components (e.g., sensors or microSD) fail to initialize or operate correctly.
ESP32-S3	The embedded microcontroller platform used as the main processing unit for real-time sensing, control, and TinyML inference.
Event-Driven Control	A control style where state transitions are triggered by events such as button interrupts or sensor-derived conditions.
Feature Extraction	The process of transforming raw sensor readings into compact features usable by the ML classifier.
Flex Sensor	A bend sensor used to measure finger bending for gesture recognition.
Gesture	A predefined hand pose/motion pattern used as an input command representation.
Gesture Classifier (TinyML Core)	The embedded ML component that classifies sensor data into one of the predefined gestures and outputs a confidence score.
GPIO (General Purpose Input/Output)	Microcontroller pins used for button interrupts, LED control, and module control signals.
Global Software Control	The top-level control logic (often a state machine) that governs system states such as Booting, Ready, Recognizing, Transmitting.
HAL (Hardware Abstraction Layer)	Low-level drivers for hardware interfaces (e.g., ADC, I2C, SPI, UART/I2S, GPIO) that isolate hardware details from upper layers.
Hand Signals (Visual Signaling)	Traditional line-of-sight based method of command delivery that may fail in darkness, smoke, or obstruction.
Headless UI	A screenless UI using physical controls and indicators (power switch, calibration button, RGB LED).

Term	Definition
Headless UI Manager	The module that handles user input and provides status feedback through the RGB LED.
I2C (Inter-Integrated Circuit)	A serial bus used to communicate with peripherals such as IMU sensors and (in the design) transmitter configuration.
IMU (Inertial Measurement Unit)	Sensor module capturing acceleration/gyro data to represent hand motion and orientation.
Idle / Ready	The standby operational state where the system waits for a gesture.
Initialization Failures	Boot-time failures such as sensor or microSD initialization issues that trigger an ErrorState and user alert.
Layered Architecture	An architectural style that separates system concerns into layers (HAL, data processing, application, output/communication).
Li-Po Battery	Rechargeable battery used to power the wearable glove device.
Low Battery Condition	A boundary condition where system behavior prioritizes status indication to avoid unstable behavior from voltage drops.
Low Confidence / Recognition Failure	A condition where the classifier's confidence is below threshold; transmission is canceled and "not sent" feedback is provided.
Maintainability	The ease of updating/repairing software and hardware, supported by modular components and accessible parts.
microSD Card	Removable storage used to store DFPlayer-compatible command audio file (e.g., MP3/WAV)
Modularity	Designing components so they can be updated independently (e.g., model updates or mapping changes without breaking the pipeline).
Movement Timeout	A boundary case where sensor data collection does not complete within the expected window and the system returns to Idle/Ready.
Multi-Receiver Broadcast	Sending the same command audio to multiple receiver units simultaneously to synchronize team actions.
Noise Filtering	Processing that reduces sensor noise to stabilize recognition results.

Term	**Definition**
Operational Integrity	Reliability/safety of behavior during operation (e.g., avoiding incorrect broadcasts via the accuracy gate).
Output / Communication Layer	The layer responsible for audio streaming/retrieval and RF transmission handling.
Persistent Data Management	Strategy for storing long-lived data such as audio files (microSD) and model/mapping/configuration (flash).
Preprocessing / Normalization	Transformations applied to raw sensor data to reduce variation and improve recognition robustness.
PWM (Pulse Width Modulation)	A control technique used to drive the RGB LED brightness/color and implement blink patterns.
Radio / Receiver Unit	The remote device that receives RF transmissions and outputs the command audio via speaker or headset.
RF (Radio Frequency)	Wireless communication method used to transmit command audio from glove to receivers.
RF Broadcast	The act of transmitting the prepared audio signal over RF to receiver units.
Robustness	The ability to maintain acceptable performance despite noise, user variation, and operational constraints.
Sampling Rate	Frequency at which sensors are read; consistent sampling improves gesture recognition reliability.
Security (Signal Isolation)	Reducing exposure by using RF broadcast rather than external network connectivity (Wi-Fi/cellular).
Sensor Manager (Sensor Management Subsystem)	The module responsible for continuous, high-frequency sampling and normalization of flex and IMU data.
Sensor Streaming	Continuous flow of sensor readings from flex and IMU used for real-time processing.
SPI (Serial Peripheral Interface)	Serial bus commonly used for microSD card access and fast peripheral communication.

Term	Definition
State Machine	The control mechanism that defines states (Booting, Ready, Collecting Data, Recognizing, Transmitting) and transitions between them.
Subsystem Decomposition	The breakdown of the system into smaller subsystems with defined responsibilities and interfaces (Section 3.2).
Subsystem Services	A subsystem-level “service catalog” describing what operations each subsystem provides and how subsystems interact (Section 4).
TinyML	Lightweight ML designed for microcontrollers, used here for on-device gesture classification.
UART (Serial)	Serial interface used to communicate with modules (e.g., DFPlayer Mini control).
Usability / Ergonomics	Design goal ensuring the glove is easy to wear, does not restrict movement, and is learnable with minimal cognitive load.
Visual Line-of-Sight	Requirement for traditional hand signals; AKKE reduces dependence on line-of-sight by using wireless audio broadcast.
WAV (Uncompressed Audio)	One possible audio format; decoding/playback is handled by DFPlayer Mini.
Wireless Range (~10 m)	A target minimum RF communication range for the prototype, subject to RF module and antenna conditions.

6. References

[1] A. ESİN, B. ÇAKMAK, Ö. E. DİKİCİ, Ş. KURTULMUŞ, **"AKKE – Project Proposal (SMART COMMAND AND CONTROL GLOVE)"**, TED University, CMPE 491 – Senior Project I, Fall 2025. (Webpage: https://ilter-akke.github.io/website/)

[2] A. ESİN, B. ÇAKMAK, Ö. E. DİKİCİ, Ş. KURTULMUŞ, **"AKKE – Project Specifications Report (SMART COMMAND AND CONTROL GLOVE)"**, TED University, CMPE 491 – Senior Project I, Fall 2025. (Webpage: https://ilter-akke.github.io/website/)

[3] A. ESİN, B. ÇAKMAK, Ö. E. DİKİCİ, Ş. KURTULMUŞ, **"AKKE – Analysis Report (SMART COMMAND AND CONTROL GLOVE)"**, TED University, CMPE 491 – Senior Project I, Fall 2025. (Webpage: https://ilter-akke.github.io/website/)

[4] V. Belcamino, A. Carfì, and F. Mastrogiovanni, "A systematic review on custom data gloves," *IEEE Transactions on Human-Machine Systems*, pp. 1–16, 2024. https://doi.org/10.1109/THMS.2024.3394674

[5] B. Bruegge and A. H. Dutoit, *Object-Oriented Software Engineering: Using UML, Patterns, and Java*, 2nd ed. Prentice Hall, 2004. (Website)

[6] DFRobot, "DFPlayer Mini MP3 player for Arduino (SKU: DFR0299)," DFRobot Wiki. (Website)

[7] Espressif – ESP32-S3 Series Datasheet (PDF): (Website)

[8] A. Filipowska, W. Filipowski, P. Raif, M. Pieniążek, J. Bodak, P. Ferst, K. Pilarski, S. Sieciński, R. J. Doniec, and J. Mieszczanin, "Machine learning-based gesture recognition glove: Design and implementation," *Sensors*, vol. 24, no. 18, p. 6157, 2024. https://doi.org/10.3390/s24186157

[9] C. Leite, P. Byvshev, H. Mauranen, and Y. Xiao, “Simulation-driven design of smart gloves for gesture recognition,” *Scientific Reports*, vol. 14, p. 14873, 2024. https://doi.org/10.1038/s41598-024-65069-2

[10] J. H. Ong, “Data-glove-based hand gesture recognition system using flex sensors and an IMU sensor,” Project Report, Universiti Sains Malaysia, School of Electrical & Electronic Engineering, 2017. (Website)

[11] ACM, “ACM Code of Ethics and Professional Conduct,” 2018. (Online): https://www.acm.org/code-of-ethics

[12] IEEE Computer Society, “Software Engineering Code of Ethics and Professional Practice,” 1999. (Online): https://www.computer.org/education/code-of-ethics

[13] IEEE, “IEEE Code of Ethics,” 2020. (Online): https://www.ieee.org/about/corporate/governance/p7-8.html

[14] T. W. Bynum, “Computer and Information Ethics,” *The Stanford Encyclopedia of Philosophy*, 2020. (Online): https://plato.stanford.edu/entries/ethics-computer/

[16] Anthropic, “Claude AI,” 2025. (Online): https://claude.ai

[17] OpenAI, “ChatGPT,” 2025. (Online): https://chatgpt.com/

[18] Arduino, “ESP32 Core Documentation,” 2025. (Online): https://docs.arduino.cc/hardware/esp32