PROJECT ABSTRACT

TinyML Voice Command Switch

A microcontroller-based voice command switch: a MEMS microphone captures speech and a quantized depthwise-separable CNN spots keywords like “on” and “off” fully on-device to switch a real relay-driven appliance — no internet. Delivered built-to-order with the switch node, relay load fixture, trained model, firmware, report, PPT and viva kit.

Project type: Hardware

Availability: Built to order

Suitable branches: AI & Machine Learning, Electronics / E&TC, IoT & Embedded

Technologies: ESP32, INMP441 microphone, TensorFlow Lite Micro, Google Speech Commands, DS-CNN, relay module, OLED, Arduino IDE

Price: Request a quotation

Prepared by Projectech · projectech.in — for academic project reference.

1. Title

TinyML Voice Command Switch

2. Introduction / Background

Voice assistants made “turn on the light” feel ordinary, but the usual implementation streams speech to a distant server — needing Wi-Fi, a cloud account, and continuous listening through someone else's infrastructure. TinyML offers the opposite architecture: the entire speech pipeline runs on a microcontroller next to the switch. This project builds that as a complete appliance-control prototype. An I2S MEMS microphone captures one-second audio windows; on-device preprocessing extracts MFCC features; a depthwise-separable CNN (DS-CNN) — the architecture family designed for small-footprint keyword spotting — classifies each window into a command word, “unknown” or silence; and a relay switches a real AC demo lamp with no internet involved.

3. Problem Statement

Keyword spotting is usually demonstrated as a serial-monitor toy that prints detected words — the control loop is missing. Students learn the classifier but never the appliance: relay actuation, confidence gating, feedback and safety interlocks. This project builds the complete loop: a real speech dataset (Google Speech Commands, 105,829 utterances), a genuinely small DS-CNN quantized for the board, and a relay-driven load with opto-isolation, OLED command feedback, a manual override and a hardware kill switch — voice control engineered like a product, at student scale.

4. Objectives

- Capture speech with an I2S MEMS microphone and on-device MFCC feature extraction.
- Train a DS-CNN keyword spotter on the real Google Speech Commands dataset (command words, unknown, silence).
- Quantize the model to int8 and deploy it with TensorFlow Lite Micro for real-time inference.
- Drive a relay-switched AC demo load with confidence gating, OLED feedback and safety interlocks.
- Validate command reliability with the student's own spoken test procedure at fixed distances.
- Deliver the complete kit: node, relay fixture, model, firmware, report, PPT and viva Q&A.;

5. Proposed Methodology

The work proceeds in three stages. (1) Offline training: Speech Commands utterances are converted to MFCC features through the exact pipeline the firmware will use; a DS-CNN is trained to separate the command words from unknown speech and silence, then post-training quantized to int8. (2) Firmware: the microcontroller streams overlapping 1-second windows through MFCC extraction and inference, applies a confidence threshold plus a confirmation rule across consecutive windows, and on a confirmed command drives the relay and updates the OLED. (3) Validation: the student runs the spoken test procedure — commands at 1 m and 2 m in a quiet room — and logs recognition vs false-trigger rates.

6. System Architecture / Working

The microphone captures audio continuously; firmware slices it into overlapping 1-second windows and computes MFCC features on-device. The int8 DS-CNN outputs a probability per class (command words, unknown, silence). Firmware requires the winning keyword to exceed the confidence threshold across consecutive windows — rejecting casual conversation containing the words — then energizes the opto-isolated relay to switch the AC demo lamp and echoes the recognized word, confidence and new ON/OFF state on the OLED. A hardware kill switch always overrides voice. Training uses the identical MFCC pipeline offline, so deployed features match trained features.

7. Hardware / Software Requirements

- ESP32 or Arduino Nano 33 BLE Sense

- INMP441 I2S MEMS microphone
- Relay module with opto-isolation + AC demo lamp fixture
- 0.96-inch OLED display, manual override switch
- TensorFlow Lite Micro (int8 keyword model)
- Python training pipeline (MFCC features, TensorFlow)
- Google Speech Commands v2 dataset (public research data)
- Arduino IDE / PlatformIO (C/C++ firmware)

8. Expected Outcomes

- A working voice switch that toggles a real AC load on spoken commands with OLED feedback.
- A trained, quantized DS-CNN with test accuracy reported honestly from the training run.
- A documented MFCC pipeline and confidence-tuning procedure.
- Student-measured command reliability from the spoken test procedure.
- A complete report, PPT and viva Q&A; covering speech features, DS-CNN design, quantization and safety.

9. Applications

- Teaching keyword spotting and speech feature extraction on microcontrollers.
- Demonstrating offline, privacy-preserving voice control as an alternative to cloud assistants.
- A reference build for accessibility-oriented switch-control projects.

10. Future Scope

- Custom wake-word training with the student's own voice data.
- Multi-command vocabulary (fan speed, dimmer levels) with retraining.
- Speaker verification so only enrolled users can switch.
- Battery-powered wireless switch node with BLE status.