**Project Specification: Voice-Controlled Piano Page Turner**

**Overview**

This project implements a voice-controlled page-turner for piano players. The system is based on a Teensy 4.1 microcontroller with a Teensy Audio Shield and a MAX9814 microphone (Adafruit). It connects via USB-C to a tablet, acting as a USB HID keyboard that sends page-up and page-down events.

The device recognises four spoken commands: next, weiter, zurück, and back. These are trained and recognised using TensorFlow Lite Micro (TFLM) with embedded MFCC (Mel-frequency cepstral coefficients) features. The firmware is developed using PlatformIO.

**Hardware Components**

* **Microcontroller**: Teensy 4.1
* **Audio interface**: Teensy Audio Shield
* **Microphone**: MAX9814
* **User Interface**:
  + Red LED: Recording active
  + Green LED: Listening mode active
  + Yellow LED: Model upload in progress
  + Push button: Initiates recording
* **Connectivity**:
  + USB-C to tablet (HID interface)
  + Serial connection to PC for training/testing
  + Debug serial output (via USB-serial converter)
  + U**ART to USB adapter:** usedduring training to get a log

**Audio & MFCC Configuration**

* **Format**: 16-bit PCM mono
* **Sampling Rate**: 16 kHz
* **Sample Duration**: 2 seconds per recording
* **MFCC Parameters**:
  + Window size: 30 ms
  + Stride: 10 ms
  + Coefficients: 13

**File Structure (on PC)**

/dataset/

/next/

/back/

/weiter/

/zurück/

/silence/

/background/

/model/

model\_v1.tflite

model\_v2.tflite

* Augmented data is stored in the same label directories
* Model files are versioned; the latest is renamed to model.tflite before upload

**Use Cases**

**1. Recording**

* Triggered by pressing the push button.
* Red LED lights up.
* Audio is streamed to the PC over serial.
* Recording ends after button release or 2s timeout.
* PC:
  + Receives audio
  + Pads/trims to 2s
  + Validates sample count
  + Saves .wav file under current label mode
  + Plays back the recording

**2. Testing**

* Teensy acts like in recording mode.
* PC uses incoming audio to run inference with current model.
* Detected category is printed.
* Same MFCC preprocessing is used as in training and Teensy.

**3. Training**

* Activated from the PC menu by pressing t
* PC trains a TensorFlow model using Speech Commands format
* Augmentation is run automatically before training
* Output: model\_vN.tflite in /model

**4. Transfer Model**

* Activated from the PC menu by pressing u
* Yellow LED lights up
* PC chunks model into 512-byte blocks and sends with CRC
* Teensy reassembles, validates full CRC, renames after successful upload
* Teensy confirms via serial
* Model is stored in flash memory including version number

**5. Listening**

* Default mode after startup if model is present
* Green LED is on
* Teensy runs real-time MFCC + model inference
* If keyword detected:
  + next/weiter → HID PAGE\_DOWN
  + back/zurück → HID PAGE\_UP
* If no model is found, listening is disabled

**Serial Communication**

* Packet Structure: [2 bytes header][payload length][payload][CRC-8]
* Supported Commands:
  + RECORDING\_START
  + RECORDING\_STOP
  + MODEL\_UPLOAD\_START
  + MODEL\_UPLOAD\_END
  + ACK / NACK / ERROR / READY
* Error Handling:
  + On CRC or protocol failure: discard input for 1s, then reset state
  + Teensy maintains a state machine (recording/listening/uploading)

**Audio Augmentation**

* Automatic before training
* All next/back/weiter/zurück files are mixed with silence and background (piano sound)
* SNR levels: 5 and 10
* Output file format: next/next\_5\_aug1.wav, etc.
* Augmented files are included automatically in training
* Additional augmented files are created by shifting audio snippets to get the model accustomed to the word not always been at the same position.

**Debouncing & Safeguards**

* **Button debounce**: Timing-based logic
* **Inference debounce**: 1s minimum wait between PAGE\_UP/PAGE\_DOWN triggers
* **Startup check**:
  + If model.tflite is found, green LED turns on
  + Otherwise, listening is disabled until upload

**Silence Detection**

* Silence is determined based on RMS threshold (e.g. −40 dB)
* Python applies:

rms = np.sqrt(np.mean(audio\*\*2))

rms\_db = 20 \* np.log10(rms + 1e-10)

if rms\_db < threshold\_db:

label = "silence"

* Background is defined as non-speech ambient piano noise
* Both are set manually via the PC menu mode

**Logging**

* Debug logs are printed to terminal over second serial port
* Format: [INFO] Recording started at t=123ms
* Logging is not saved to disk

**Python Console Interface**

* Keyboard-controlled menu (non-blocking input)
* Microcontroller-style loop (no threads)
* Polls keyboard and serial input alternately
* Menu states:
  + Change label mode
  + Start training
  + Start model transfer
  + Play back dataset
  + Switch to test mode

**Notes**

* All models and data structures follow TensorFlow Speech Commands conventions
* Teensy only switches to listening mode if model is present
* Teensy always loads model.tflite on boot
* CRC is applied per model chunk and for full model file after reassembly
* PC always sends the most recent model version