**Project Summary**

**Goal:**  
A multi-platform (~~desktop~~ + Android) system using C++ and V4L2 (Video4Linux) to recognize sign language via webcam, display subtitles, and use TTS (text-to-speech).  
Plus, on Android: if the phone is shaken (via accelerometer), it starts or stops the webcam-based detection.

**Hardware Stack**

| **Device / Sensor** | **Purpose** |
| --- | --- |
| **Webcam** **480p–720p USB webcam** | Capture video for sign detection |
| **Android Phone Accelerometer** | Shake detection to toggle recording |

**Software Stack**

**1. Language & Core Frameworks**

| **Component** | **Stack / Tool** |
| --- | --- |
| **Main Language** | C++ (for OS-level and native performance) |
| **Camera Interface** | **V4L2 (Linux)** - API, **OpenCV (cross-platform)** - library |
| **AI/ML Model** | **TensorFlow Lite** or ONNX (for gesture detection) |
| **Text-to-Speech (TTS)** | **eSpeak**, **RHvoice** |
| **Subtitle Engine** | C++ string output + OpenCV GUI overlay |
| **Shake Detection** | Android Sensor API (Java/Kotlin) |
| **Communication (Phone <-> PC)** | Sockets or WebSocket over WiFi or USB |

**2. Cross-Platform Bridge**

* **Android App (Java/Kotlin):**
  + Detects phone shake
  + Sends start/stop command to PC software
* **PC-side Listener (C++):**
  + Socket listener waits for phone command
  + Starts/stops camera capture and AI

**Structure and Architecture**

**PC Side (C++ App)**

**main.cpp**

**├── capture\_module.cpp/h** // Uses V4L2/OpenCV to get webcam input

**├── sign\_recognition.cpp/h** // Loads trained model and runs detection

**├── subtitle\_module.cpp/h** // Renders subtitles to screen

**├── tts\_module.cpp/h** // Converts text to speech (eSpeak/Festival)

**├── socket\_listener.cpp/h** // Listens for shake-trigger via socket

**└── utils.cpp/h** // Helpers, config reader etc.

**Android Side (App)**

* MainActivity.java
  + Registers accelerometer listener
  + On shake, sends TCP/UDP/WebSocket command to PC

**Development Steps**

**Step 1: Basics**

* Learn:
  + OS concepts (process, threads, file I/O)
  + Basic C++ (especially file/stream handling)
* Set up OpenCV + V4L2 camera capture
* Make a simple GUI to show webcam feed + draw boxes

**Step 2: AI Inference**

* Train or use a pre-trained **sign language recognition model**
  + Convert to ONNX or TensorFlow Lite
  + Load model using ONNX Runtime / TFLite C++ API
  + Feed video frames into it

**Step 3: Subtitles + TTS**

* Based on recognized signs, print subtitles
* Feed recognized words into a TTS engine (e.g., espeak "Hello")

**Step 4: Android Shake App**

* Use Android SensorManager to detect shake
* On shake, send a message to PC (IP + port)
* Use simple TCP/UDP or HTTP client

**Step 5: Connect Both**

* On shake, PC toggles state (recording ↔ paused)
* Show on GUI: “Recording...” or “Paused”

**APIs**

**PC-side:**

* **V4L2** → Video capture (Linux)
* **OpenCV** → Frame processing + GUI
* **ONNX Runtime** → Load ML model
* **eSpeak / RHvoice** → TTS
* **Sockets (Berkeley/Unix)** → For communication

**Android-side:**

* **SensorManager** → Accelerometer
* **Socket/HTTP API** → Network connection
* **Permissions:** INTERNET, BODY\_SENSORS

**Tools to Use**

| **Purpose** | **Tools** |
| --- | --- |
| IDE | VS Code, Android Studio |
| Build System | CMake |
| Model Testing | Python + OpenCV (optional) |
| Communication Test | Netcat, Wireshark |
| Version Control | Git + GitHub |
| Model Training | Google Colab (for free GPU) |

**Project Milestone Plan**

| **Phase** | **Tasks** |
| --- | --- |
| Week 1 | Set up camera input using V4L2 + OpenCV |
| Week 2 | Display frame on GUI, overlay fake subtitles |
| Week 3 | Integrate a sign detection model |
| Week 4 | Connect to eSpeak/Festival TTS |
| Week 5 | Build Android shake app |
| Week 6 | Set up socket communication |
| Week 7 | Final integration & testing |
| Week 8 | OS-level packaging & polishing |

Optional

* A GitHub folder structure for the full project?
* Code templates for webcam and socket modules?
* Help training a lightweight sign model?