Firmware Developer – Embedded Systems (ESP32 / Audio / IoT)

Type: Freelance · Remote
Start Date: Immediate
Project Duration: ~2 months, with extension possibility

About the Opportunity

We’re building an audio device for children that brings stories to life through sound, imagination, and cultural depth. Our mission is to help kids explore values, emotions, and big ideas in a fun and age-appropriate way - without relying on screens or flashy visuals.

We’re looking for a hands-on Firmware Developer who enjoys solving hardware-software challenges, optimizing for audio performance, and working with ESP32-based systems. This is a freelance project ideal for someone who thrives in fast-paced prototyping and is passionate about clean, modular embedded design.

If you enjoy building real-world products, love debugging embedded puzzles, and want to be part of something meaningful for kids and families — we’d love to work with you.

Your Role

- Build and optimize embedded firmware for our ESP32-based platform, designed for real-time audio playback and interaction.
- Implement and manage communication interfaces — I2C, I2S, SPI, UART — for audio modules, sensors, and figurine detection.
- Program and control amplifier modules to ensure smooth, high-quality sound output.
- Set up secure Wi-Fi and Bluetooth provisioning using BLE, OTA firmware updates, and backend connectivity (e.g., HTTP, MQTT).
- Work with C++ (object-oriented), Arduino/VSCode environments, and use Git + CMake for build and version control.
- Collaborate with our product team to support feature testing, board bring-up, and manufacturing trials.

What We’re Looking For

- Proficiency with ESP32 firmware development and embedded C/C++
- Experience in real-world use of I2S, I2C, UART, SPI
- Knowledge of amplifier chips and audio interface handling (TAS2563, DACs, etc.)
- Strong understanding of wireless provisioning (BLE/Wi-Fi), OTA updates, and security best practices
- Ability to debug hardware-firmware interaction issues (serial tools, logic analyzers, etc.)
- Familiarity with Arduino CLI, CMake, VSCode, and Git

Bonus If You Have

- Audio DSP experience or work in sound-heavy consumer devices
- Experience working on low-volume hardware prototypes
- Familiarity with battery management systems, power optimization, or low-power firmware modes

---

Assignment

We have created an assignment that aims to check your cpp coding skills, understanding of the problem, and object-oriented approach towards development.

There are 3 modules.

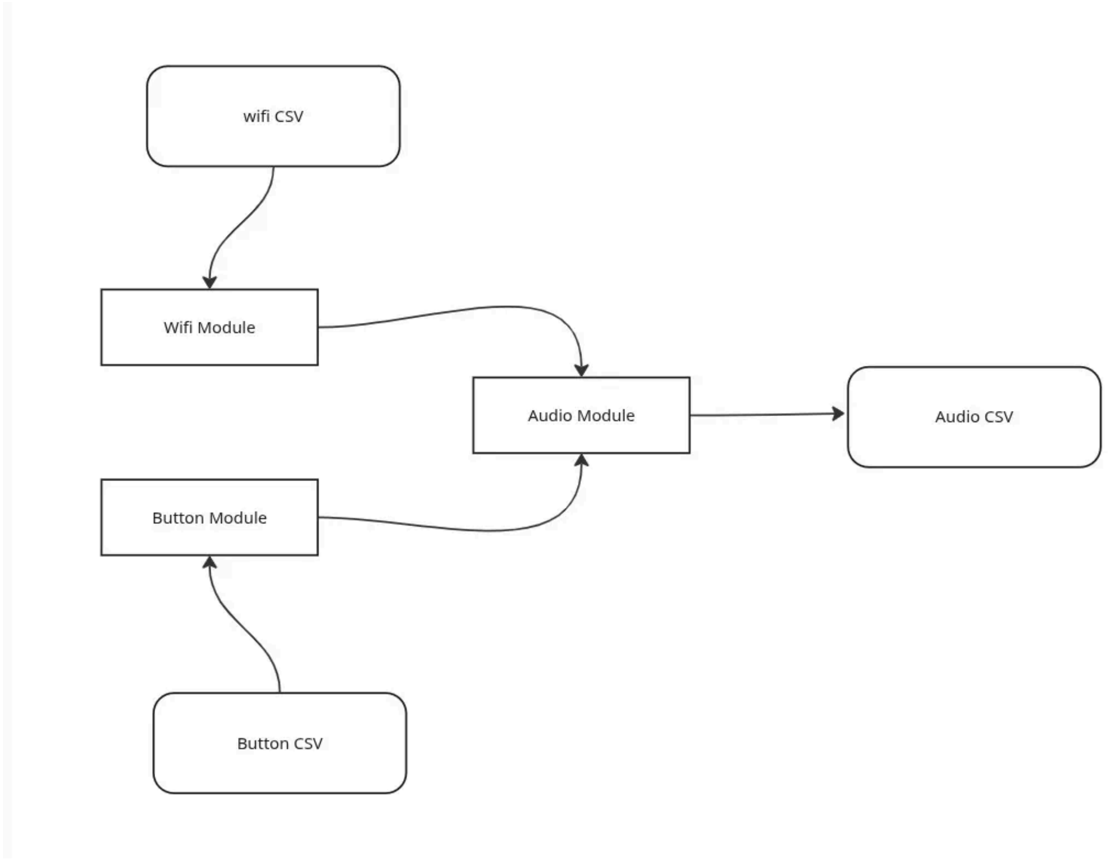
- Wifi Module, that keeps checking a csv file at a rate of 10 Hz.
- Audio Module that runs at 10Hz
- Button module that read a csv file at 0.1 Hz

Note: Since you don't have hardware, hence we are simulating by reading CSV File.

Module Explanation:

- Wifi Module:
  - The aim of the wifi module is just to keep checking the connectivity to wifi.
  - Since we are simulating the connection, we keep reading a csv file that has CONNECTED or DISCONNECTED written separated by “,” .
  - The wifi module is supposed to inform the AUDIO module if it is connected or disconnected from wifi.
  - Reads the “wifi.csv” file at 10 Hz
- Button Module:

    - The button module simulates physical buttons, which can be, PLAY, PAUSE, NEXT, PREVIOUS
    - The buttons module reads the “button.csv” at 0.1 Hz file and informs the Audio module about it.
- Audio Module:
    - The module is simulating if it is playing an audio or it is paused or if shifting track to next or Previous.
    - If the Audio Module receives info from WIFI Module, about it is connected to WIFI it must print to the CSV file “PLAYING AUDIO ONLINE”. This should only be printed, only when it transitions from DISCONNECTED to CONNECTED.
    - If the Module receives info from WIFI Module about it being DISCONNECTED from wifi, it must print to the CSV file “PLAYING AUDIO OFFLINE”. This should only be printed, only when it transitions from CONNECTED to DISCONNECTED.
    - The Audio Module must be printing into the csv file “PLAYING AUDIO” at a frequency of 10Hz.
    - The Audio Module if receives PAUSE from the Button Module it must print “PAUSED” only once when the event happens. It must stop printing “ “PLAYING AUDIO”.
    - The Audio Module if receives NEXT/PREVIOUS it must print “PLAYING THE <NEXT/ PREVIOUS> TRACK” accordingly once, only when the event happens.
    - All the print from the Audio module goes to Audio CSV.



NOTE:
1) Smartly open and close the csv files in your code.
2) Please download the wifi.csv & button.csv file at this link: Embedded Assignment

---

Please reach out to us with your assignment or any queries at a.adani@yoomeplaylab.in , kushal.j.10@gmail.com