Alarm Clock Project Journey

Ameer Tabri

Mohammad Abo Foul

Muhammad Bakri

spring 2023 IOT 236332/3

Thought process.

1) We started our project simple, focusing on implementing the time functionality:
    - During setup, the ESP32 synchronized with global time.
    - The ESP32 updated the time every second without relying on an internet connection.
    - By testing the display with a built-in function that takes a string, we successfully created a display that showed the current time.
2) Once real-time functionality was in place, we proceeded to download an MP3 file for the alarm and saved it on the ESP32 file system, and we make sure that the amplifier was working correctly.
3) Next, we wrote the functions to handle the alarm, initially setting the alarm time manually.
4) Shifting our attention to the app, we started with a basic approach, creating sliders for changing volume and brightness as well an auto brightness button, to test the connections between Flutter, Firebase, and the ESP32.
5) As the app took shape, we added an option to set the alarm time, including selecting specific days of the week. Starting with a basic design, we later improved it by dedicating a separate page for alarm settings, allowing users to add multiple alarms.
6) We wanted to add something unique for the app, we decided to combine a game element with the alarm system. Thus, we introduced difficulty levels, one of which involves playing a snake game and reaching a certain score to turn off the alarm.
7) Finally, we added more features, including a "study mode" with two timers - one for studying and one for rest. Additionally, we integrated a "reminders" functionality, enabling users to set reminders using the app. As we progressed, decided that all these features made a device suitable for students, helping them in studying and managing their time effectively.

Through this thought process, our project evolved from a simple clock implementation to a feature packed and student-friendly device, providing an all-in-one solution for time management and productivity in one device.

The evolution of our app

1. Initially, we made sliders for clock volume and brightness, alongside an auto-brightness button (which will work in the clock using a light sensor). These features are part of the settings page.
2. Then, we made it possible to set a single alarm using the app.
3. After that we expanded the alarm functionality, enabling users to set multiple alarms.
4. To enhance the alarm customization, we added various options for each alarm, such as selecting specific days of the week, choosing personalized ringtones, and assigning unique alarm names.
5. A feature that was initially a joke was adding a snake game to the app, but it was not related to the project whatsoever. Later we had the idea to add multiple difficulty levels to turn of the alarm, one of which was to play snake game to turn it off (so the user can wake up properly).
6. As the project neared completion, we decided to add even more features:
    1. Study mode page: Users can set up 2 timers one for study sessions and one for rest, with repetitions.
    2. A reminders page: This feature allows users to add multiple reminders for each day at specific times.

actions we verified separately.

- display words on screen
- display time (format and font)
- play sound from file system
- play sound from SD card
- connect to wifi, given ssid and password
- get time and date, given successful wifi connection
- get wifi from the user
- read alarms list from firebase
- check if there is an alarm for the current minute
- check text-to-speech
- say time with text-to-speech
- say time with mp3 files
- read from firebase in app
- write to firebase in app

Problems

1. **Display:** At the beginning of our project, we encountered a problem with the display. It was printing random patterns, even though the code was correct. Eventually, we discovered that changing the hardware type to MD_MAX72XX::FC16_HW in the code resolved the issue.

   Surprisingly this simple problem took us the most to solve.

2. **Sound:** We had two audio devices at our disposal, and we attempted to use one of them. However, it presented a problem: upon starting, it kept repeating a small segment until we pressed the boot button on the ESP32.

3. **Radio:** Initially, we thought about adding a radio to the clock's features. However, as we proceeded and added more functionalities, we faced difficulties in running the radio without stuttering. Due to the technical challenges of multitasking, we decided to not including it.

4. **RAM and Storage:** Unfortunately, the ESP32 does not have an "infinite" amount of RAM like a regular PC. This constraint required us to come up with creative solutions to solve problems without sacrificing features. We made that happen this in several ways:

    1. We minimized the use of Firebase variables since each variable consumes a significant amount of RAM.

    2. We removed unnecessary libraries to free up memory.

    3. The audio variable requires a substantial amount of RAM during initiation, making it challenging to read from Firebase and initialize the variable simultaneously. To overcome this, we implemented a solution where we destroy the audio variable after completing an audio task. If a change is detected from Firebase, the audio has to be stopped to make room for the Firebase task. Fortunately, such occasions are rare.