Project Scope: ESP32 Badminton Court Timer (Revised)

1. Project Overview

The objective is to create a timer system for a badminton court using an ESP32 microcontroller. The system will be controlled via a simple web page accessible from any device on the same local WiFi network. The ESP32 will manage the game timing, including the total game duration and an initial break period within each round, and will activate a relay connected to a siren to signal key events.

2. Core Features

- **Web-Based Control Panel:** A user-friendly, responsive web interface to control the timer.
- **Easy Network Access (mDNS):** Access the control panel via a friendly name (e.g., badminton-timer.local) instead of an IP address.
- **Time of Day Display:** Shows the current time on the web interface, synchronized via the internet.
- **WiFi Connectivity:** The ESP32 connects to an existing local WiFi network to serve the web page.
- **Timer Controls:**
  - **Start Button:** To start the entire match sequence.
  - **Pause/Resume Button:** To temporarily pause and resume all timers.
  - **Stop/Reset Button:** To stop the sequence and reset the system to its initial state.
- **Configurable & Persistent Settings (on the web page):**
  - **Game Duration:** An input field to set the total length of each round in minutes.
  - **Break Time:** An input field to set the length of the initial break period in seconds.
  - **Number of Rounds:** An input field to set how many consecutive rounds to play.
  - *All settings are saved on the ESP32 and remembered after a reboot.*
- **Real-Time Display:**
  - **Main Timer:** Displays the time remaining in the current round.
  - **Break Timer:** A smaller, secondary display showing the countdown of the initial break period.
  - **Round Counter:** Displays the current round (e.g., "Round 1 of 3").
- **Visible Status Indicator:** A prominent GUI element will act as an "Enable Display":
  - **White (or neutral):** Indicates the system is idle or the match is finished.

    - **Green:** Indicates the match is active and a round is in progress.
- **Audible Alarm:** The ESP32 will control a relay connected to a siren with distinct signals:
    - **End of Break:** A single siren blast.
    - **End of Round/Game:** A double siren blast.

3. Hardware Requirements

1. **ESP32 Development Board:** The core of the project (Model: **XC3800**).
2. **5V Relay Module:** A single-channel relay module (Model: **XC4419**).
3. **Siren/Buzzer:** An audible alarm that can be powered by a separate power source.
4. **Power Supply:**
    - A 5V USB power supply for the ESP32.
    - A separate power supply suitable for the chosen siren.
5. **Jumper Wires:** To connect the ESP32 to the relay module.
6. **Enclosure (Recommended):** A project box to safely house all components.
7. **Wiring Details & I/O Assignment:**
    - **Relay Control Pin:** The relay will be controlled using **GPIO 26** on the ESP32.
    - **Connections:**
        - ESP32 **GND** → Relay Module **-** (GND)
        - ESP32 **V5** → Relay Module **+** (5V)
        - ESP32 **G26** → Relay Module **S** (Signal)

4. Software Architecture

The project will be built using the Arduino framework for the ESP32.

- **ESP32 Firmware (C++/Arduino):**
    1. **WiFi & Settings Manager:** On first boot, the ESP32 creates a WiFi Access Point. A captive portal allows the user to configure local WiFi credentials and saves settings to non-volatile memory.
    2. **NTP Client:** The ESP32 will connect to an NTP server to fetch the current time.
    3. **Web Server:** An asynchronous web server will serve the index.html page and handle WebSocket connections for real-time communication.
    4. **mDNS Service:** Broadcasts the badminton-timer.local address.
    5. **Timer Logic:** The core timing logic runs on the ESP32, tracking the state (IDLE, RUNNING, PAUSED, FINISHED) and managing the new timer sequence and alarm patterns.
    6. **GPIO Control:** The firmware will manage GPIO 26 to control the relay.
- **Web Interface (HTML / CSS / JavaScript):**
    1. **HTML:** Will provide the structure for the clock, settings, controls, all timer

displays, and the "Enable Display" status indicator.
2. **CSS:** Styling for a clean, responsive interface, including the color change for the enable display.
3. **JavaScript:** Manages WebSocket communication, sends user commands, and dynamically updates all UI elements based on data from the ESP32.

5. Development and Testing Strategy

- **Web Interface Testing (No Hardware):** A "simulation mode" in the JavaScript will allow for UI development and testing in a browser without needing the ESP32.
- **Firmware Logic Testing (On-Device):** Extensive Serial.print() statements will be used with the Arduino Serial Monitor to verify the new state machine and timer logic on the hardware.

6. User Workflow (Revised)

1. **Initial State:** The system is **Idle**. The on-screen "Enable Display" is **white**.
2. **User Action:** Clicks the **"Start"** button.
3. **Round Begins:**
    - The "Enable Display" turns **green**.
    - The round counter updates to "Round 1 of X".
    - The visible **Main Game Timer** starts counting down from the configured Game Duration.
    - The visible **Break Timer** starts counting down simultaneously from the configured Break Time.
4. **End of Break:**
    - When the Break Timer reaches zero, a **single siren blast** sounds. The break timer display can disappear or show "00:00".
    - The Main Game Timer continues to run without interruption.
5. **End of Round:**
    - When the Main Game Timer reaches zero, a **double siren blast** sounds.
    - The system checks if this was the final round.
        - **If NOT the final round:** The round counter increments ("Round 2 of X"), and the sequence immediately repeats from **Step 3**.
        - **If it IS the final round:** The sequence proceeds to **Step 6**.
6. **End of Match:**
    - The "Enable Display" turns back to **white**.
    - The system state becomes **Finished**. It will remain in this state until the user presses the **"Stop/Reset"** button, which returns it to the **Idle** state (Step 1).
7. **Pause/Reset Controls:**
    - The **"Pause/Resume"** button can be used at any time to freeze and un-freeze

all running timers.
- The **"Stop/Reset"** button can be used at any time to immediately end the entire sequence and return to the **Idle** state (Step 1).