Real-Time Basketball Scoring System Using ESP32

Project Procedure and Design Document

August 7, 2025

1 Project Overview

This document details the complete procedure to build a **real-time basketball scoring and player stats tracking system** using an **ESP32 microcontroller**. The system supports tracking scores, fouls, assists, rebounds, steals, game clock management, and displays live updates on a Smart TV and control panel accessible via web browsers over Wi-Fi.

Key Features:

- Team score and foul tracking
- Player-level statistics (points, fouls, assists, rebounds, steals)
- Game clock with start, pause, reset, and quarter control
- Real-time updates via WebSocket communication
- Web-based scoreboard (for Smart TV) and control panel (for mobile devices)
- Physical buttons input integrated with web control input

2 Hardware Setup

2.1 Components Required

- ESP32 development board (Wi-Fi enabled)
- Physical push buttons for input (score increments, fouls, clock control, player selection)
- Pull-up or pull-down resistors (typically 10 kΩ)
- Breadboard and jumper wires or custom PCB
- Optional: OLED display for debugging/local display
- Wi-Fi router or use ESP32 Access Point mode for standalone operation

2.2 Button Assignments

Map physical buttons to following functions (example):

- Team A and Team B score increments (+1, +2, +3 points)
- Team fouls increment
- Game clock controls (start, pause, reset)
- Player selection (via buttons or rotary encoder)
- Player stat increments (fouls, assists, rebounds, steals)

3 Software Architecture

3.1 ESP32 Firmware Responsibilities

- Manage physical button inputs using GPIO and interrupts or polling
- Maintain internal game state: team scores, fouls, player stats, and game clock
- Host a web server with WebSocket support to:
  - Serve scoreboard and control panel web pages stored in SPIFFS/LittleFS
  - Send real-time game state updates to connected clients
  - Receive control commands from web clients
- Synchronize game state across all connected clients and hardware inputs

3.2 Data Structures

Define a player data structure to track full stats:

```
struct Player {
    String name;
    int number;
    int points;
    int fouls;
    int assists;
    int rebounds;
    int steals;
    bool onCourt;
};
```

Use arrays or vectors for team rosters (limit 10-12 players/team).

4 Networking and Communication

4.1 Wi-Fi Modes

- **Access Point Mode:** ESP32 creates its own Wi-Fi network to which Smart TV and control devices connect.
- **Station Mode:** ESP32 connects to an existing Wi-Fi network shared with Smart TV and control devices.

4.2 Communication Protocol

- Use WebSockets (e.g., via AsyncWebSocket library) for bidirectional real-time communication.
- ESP32 broadcasts game state updates as JSON to all connected clients.
- Clients send control commands (stat updates, clock control) to ESP32.

5 Web Interface Design

5.1 Scoreboard Page

- Large, clear display optimized for Smart TV screens
- Shows team scores, fouls, current quarter, and game clock
- Displays player stats: points, fouls, assists, rebounds, steals
- Scrollable or paginated roster display for each team
- Auto-updates on receiving WebSocket data

5.2 Control Panel Page

- Touch-friendly interface for mobile devices
- Player selection controls (dropdown or numeric input)
- Buttons to increment stats (points, fouls, assists, rebounds, steals)
- Game clock controls: start, pause, reset, quarter change
- Sends commands via WebSocket to ESP32

6 Game Clock Management

- Use `millis()` timer in ESP32 to track elapsed game time
- Support start, pause, reset, and quarter transitions
- Periodically broadcast clock updates to connected clients

7 Data Persistence and Export (Optional)

- Store current game stats in SPIFFS/LittleFS as JSON or CSV files
- Allow exporting match summaries for post-game analysis
- Limit data retention due to ESP32 memory constraints

8 Integration and Testing

1. Test physical button input and debouncing
2. Verify ESP32 Wi-Fi connection and web server availability
3. Load scoreboard and control panel on devices and test real-time updates
4. Test synchronization between physical inputs and web commands
5. Simulate a full game flow including player stat updates and clock management
6. Perform stress test with multiple clients connected simultaneously

9 Project Enhancements (Future Work)

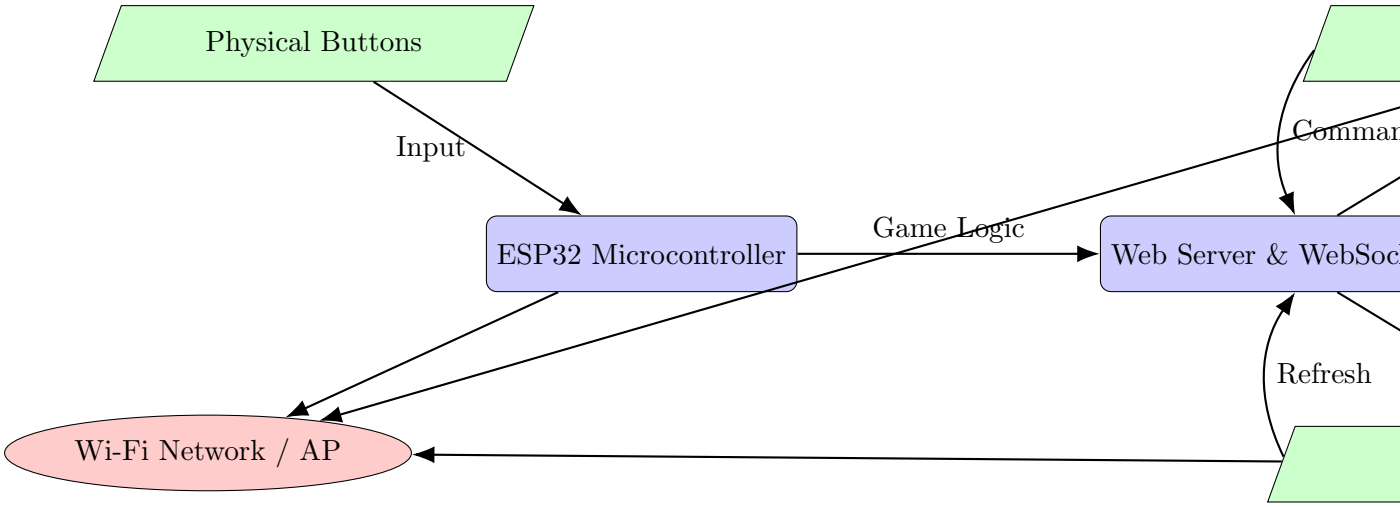
- Add audio feedback (buzzers) for game events
- Implement OTA (Over-the-Air) firmware updates

- Integrate with external databases or cloud services for long-term storage
- Enhance UI with team logos, colors, and animations
- Implement player substitution and lineup management

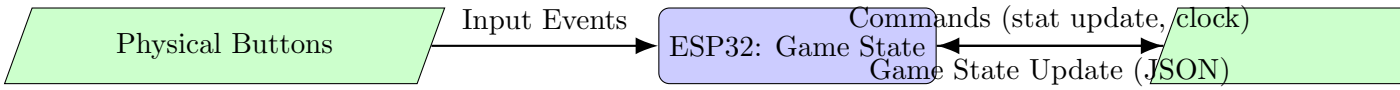
Summary of Skills Demonstrated

Category	Skills
Embedded Systems	GPIO handling, interrupts, timers, memory management
Networking	Wi-Fi setup, HTTP server, WebSocket protocol
Software Development	C++ programming, JSON handling, file system (SPIFFS)
Web Development	HTML, CSS, JavaScript, responsive UI, WebSocket client
Systems Integration	Multi-device real-time communication and synchronization
User Experience	Touch UI design, scalable display layout for TV and mobile

10 System Architecture Diagram



11 Data Flow Diagram



This document serves as a complete guideline for developing a real-time basketball scoring and player stats system using ESP32, blending embedded and web technologies for a professional, industry-relevant project.