National Textile University

Faisalabad

Department of Computer Science

Course Name: Internet Of Things
Course Code: CSE-4079
Assignment Number: 01

Class: BS Artificial Intelligence
Semester: 6th

Submitted By:

- Muneeb Ur Rehman (22-NTU-CS-1368)
- Swaiba Shahid (22-NTU-CS-1377)
- Adifa Jahangir (22-NTU-CS-1338)

Submitted To:

Dr. Nasir Mehmood

Submission Date: 12-03-2025

Table of Contents

IoT Assignment on ESP32-S3
Environmental Monitoring & Interactive Dice Game

Team Members:

- **Muneeb Ur Rehman** (22-NTU-CS-1368)
- Swaiba Shahid (22-NTU-CS-1377)
- Adifa Jahangir (22-NTU-CS-1338)

Abstract

This report presents two distinct IoT tasks developed for the ESP32-S3 microcontroller, showcasing its versatility in handling diverse applications ranging from environmental monitoring to interactive gaming. The first task integrates web-based control of an RGB LED, real-time environmental sensing with a DHT11 sensor, and local display capabilities through an OLED screen. The second project implements a multiplayer dice game with chat functionality, demonstrating the ESP32's ability to handle networked interactive applications. Both tasks utilize Wi-Fi connectivity in dual modes (STA + AP) and employ socket programming for web server functionality.

1. Introduction

The ESP32-S3 microcontroller has emerged as a powerful platform for IoT applications due to its dual-core processing capabilities, integrated wireless communication, and rich peripheral support. This report details two tasks that leverage these capabilities to create sophisticated IoT solutions:

1. An environmental monitoring system with web-based control and display capabilities
2. A networked multiplayer dice game with chat functionality

These tasks demonstrate the ESP32-S3's ability to handle both sensor-driven applications and interactive user experiences through web interfaces.

2. Environmental Monitoring System

2.1 System Architecture

The environmental monitoring system comprises:

- ESP32-S3 microcontroller
- NeoPixel RGB LED
- DHT11 temperature/humidity sensor
- SSD1306 OLED display
- Dual WiFi connectivity (STA + AP modes)

2.2 Implementation Details

2.2.1 Network Configuration

The system establishes dual WiFi connectivity:

- Station (STA) mode for connecting to external networks
- Access Point (AP) mode for direct device communication

```
sta.connect(TampleDiago, 12345676)  # Station mode
ap.config(essid=ssid_ap, password=password_ap)  # AP mode
```

```
def set_color(r, g, b):
    np[0] = (r, g, b)
    np.write()

def read_sensor():
    sensor.measure()
    return sensor.temperature(), sensor.humidity()

set_color(r, g, b)
temp, hum = read_sensor()
```

2.2.2 Hardware Integration

- **NeoPixel Control**: RGB values parsed from HTTP requests to dynamically update LED colors
- **DHT11 Sensor**: Reads temperature and humidity with error handling
- **OLED Display**: Updates messages via I2C interface with character limit enforcement

2.2.2.1 Wiring Diagram:

```
ESP32-S3          NeoPixel LED
GPIO 48 (GPIO48) → Data In (DI)
3.3V             → VCC
GND              → GND

ESP32-S3          DHT11 Sensor
GPIO 4 (GPIO4)   → Data
3.3V             → VCC
GND              → GND


ESP32-S3          SSD1306 OLED
GPIO 8 (GPIO8)   → SDA
GPIO 9 (GPIO9)   → SCL
3.3V             → VCC
GND              → GND
```

2.2.3 Web Server Development

A socket-based HTTP server listens on port 80, parsing requests and generating dynamic responses with embedded sensor data and weather conditions.

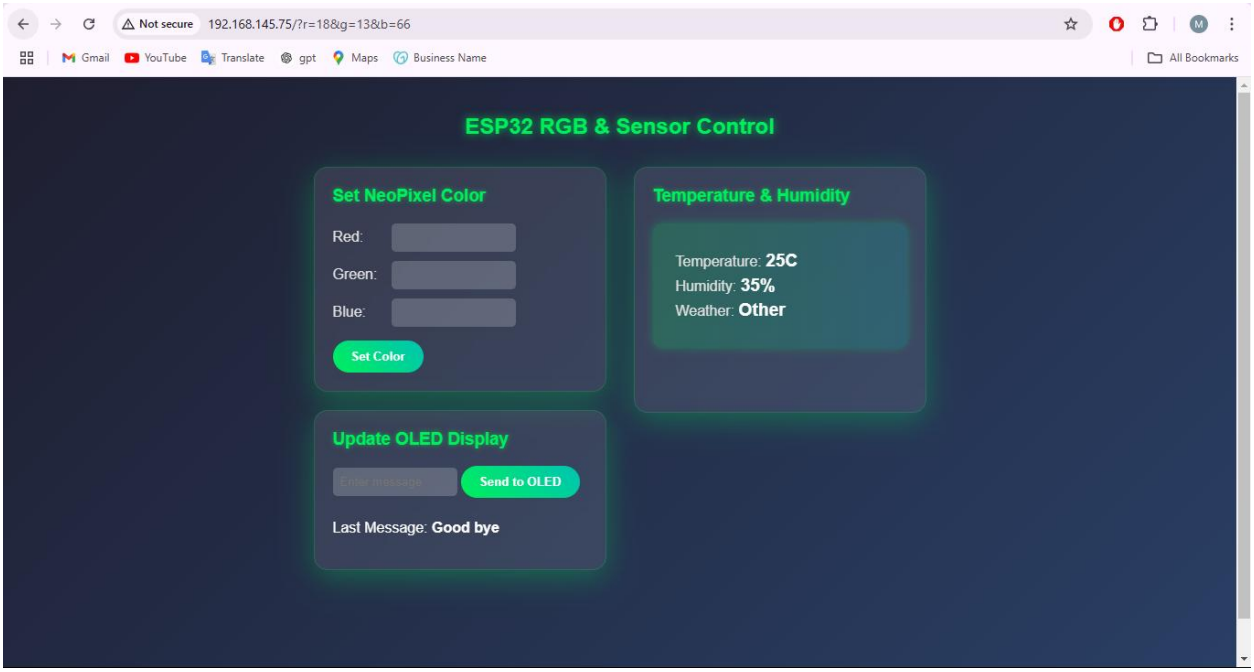


Fig 1.1 Environment monitoring Dashboard

2.2.4 Weather Condition Logic

A rule-based system categorizes weather conditions based on temperature and humidity thresholds:

Condition	Temperature (°C)	Humidity (%)
Normal	25–30	40–50
Dry or Cool	<25	<40
Cool or Moist	20–30	>64
Hot or Dry	>40	<50

2.3 Results

- Successful integration of web control, environmental sensing, and local display
- Real-time updates of sensor data on the web interface
- Accurate weather condition inference based on sensor readings
- Responsive control of RGB LED through web interface
- Timely updates of messages on the OLED display

3. Interactive Dice Game

3.1 System Architecture

The dice game system includes:

- ESP32-S3 microcontroller
- Dual WiFi connectivity (STA + AP modes)
- Web-based interface for multiplayer interaction
- Game state management with JSON statistics

3.2 Implementation Details

3.2.1 Network Configuration

Similar to the monitoring system, this task utilizes dual Wi-Fi modes for robust connectivity.

3.2.2 Game Logic

- Players take turns rolling a virtual die
- Each player has 6 turns
- Game tracks scores and determines winner
- Chat functionality for player interaction

3.2.3 Web Interface

The web interface features:

- Player registration and game initialization
- Real-time game state updates
- Chat functionality
- Game statistics display

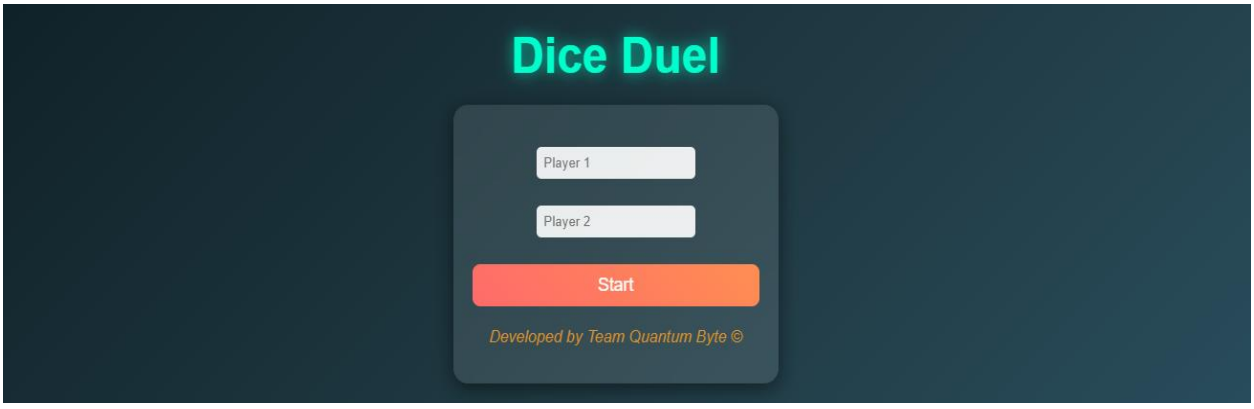


Fig 1.1 Entering Dashboard

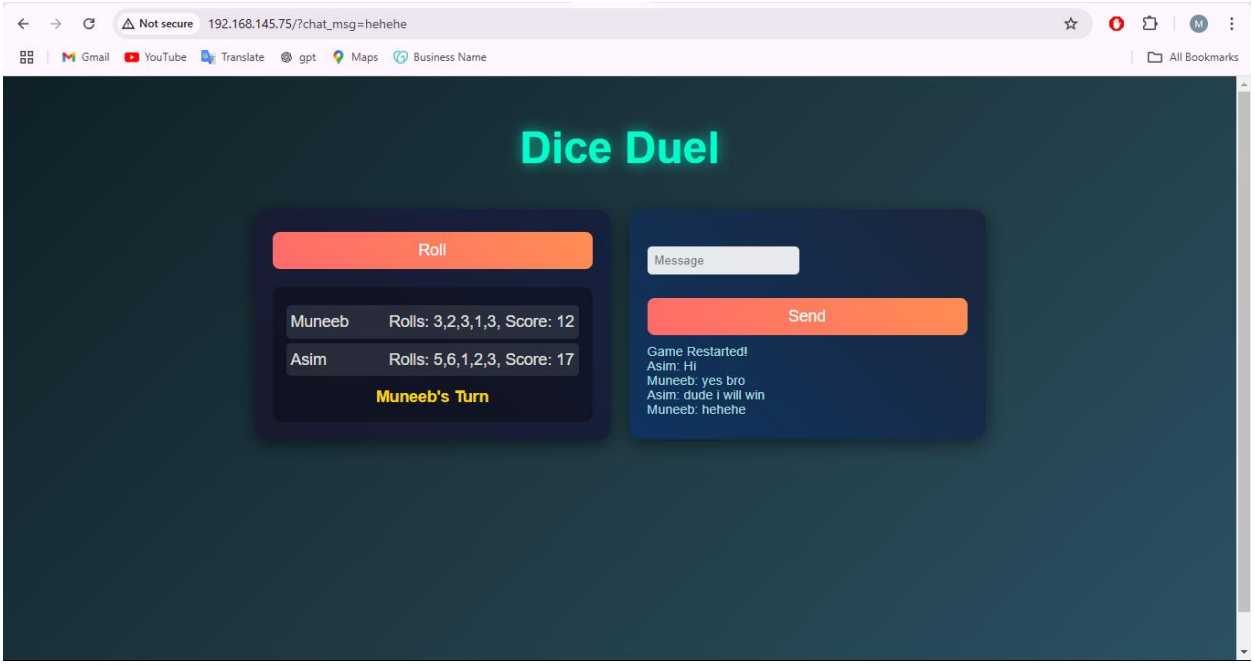


Fig 1.2 Game Dashboard

3.2.4 Game State Management

Game state is maintained in memory with periodic JSON updates for web interface rendering.

```
def roll_dice():
    return random.randint(1, 6)

def switch_player():
    global current_player
    current_player = "P2" if current_player == "P1" else "P1"

def determine_winner():
    p1_score = players["P1"]["score"]
    p2_score = players["P2"]["score"]
    if p1_score > p2_score:
        return f"{players['P1']['name']} Wins! {p1_score}-{p2_score}"
    elif p2_score > p1_score:
        return f"{players['P2']['name']} Wins! {p2_score}-{p1_score}"
    return "Tie!"
```

3.3 Results

- Successful implementation of multiplayer dice game
- Real-time game state synchronization
- Functional chat system for player communication

- Accurate turn management and score tracking
- Responsive web interface for game interaction

4. Discussion

4.1 Technical Challenges

- Managing concurrent network connections
- Ensuring timely updates across all system components
- Optimizing memory usage for MicroPython environment
- Implementing turn-based game logic with web interface synchronization

4.2 Solutions and Innovations

- Modular architecture allowing independent component operation
- Efficient memory management techniques
- Run module on new thread instead of hole code
- JSON-based game state management for easy web integration
- CSS animations and responsive design for enhanced user experience

5. Team Contributions

Environmental Monitoring System

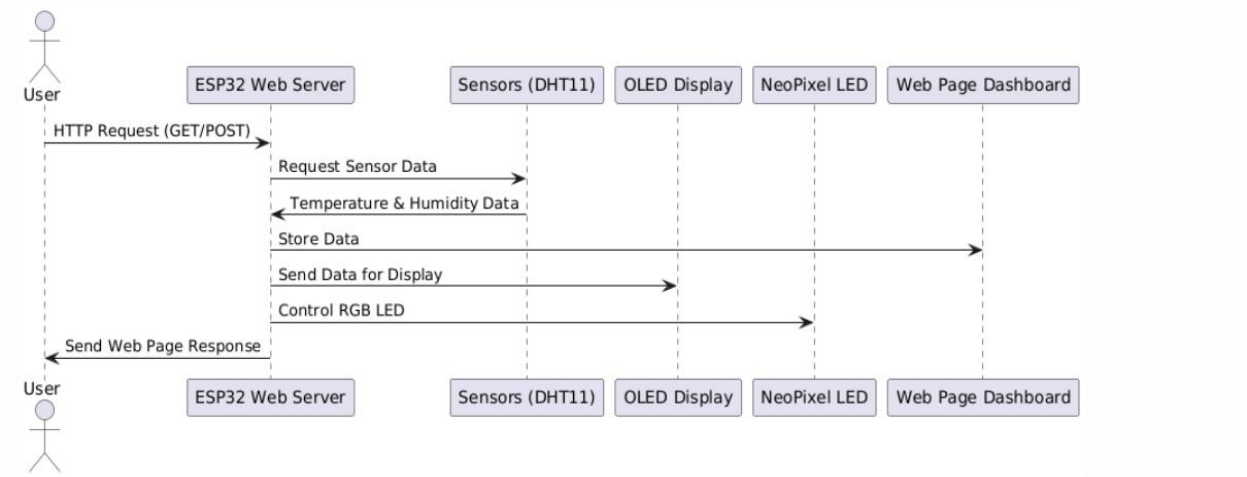
- **Muneeb Ur Rehman**: Designed the additional weather inference module that analyzes temperature and humidity data to determine weather conditions. Also led the overall webpage design and user interface implementation.
- **Swaiba Shahid**: Developed the core functionality for displaying real-time temperature and humidity data on the web interface and implemented the text display feature for the OLED screen.
- **Adifa Jahangir**: Created the module for controlling the RGB NeoPixel light, enabling web-based color customization through HTTP requests.

Interactive Dice Game Server

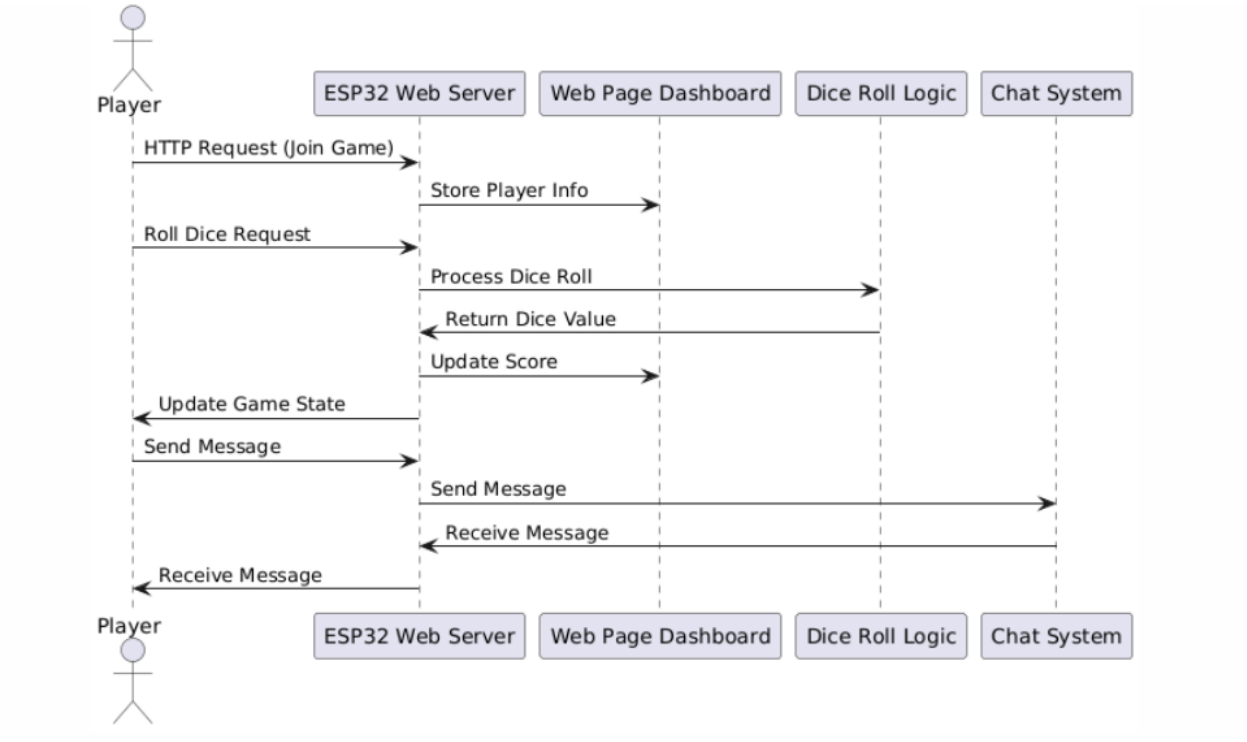
- **Muneeb Ur Rehman**: Architected the game logic, implemented the turn-based mechanics, and designed the web interface for the multiplayer dice game.
- **Adifa Jahangir & Muneeb Ur Rehman**: Collaboratively developed the real-time chat functionality, enabling players to communicate during the game.

6. Data Flow Diagrams

6.1 DFD of Environmental Monitoring System



6.2 DFD of Interactive Dice System Server



7. Conclusion

These Tasks successfully demonstrate the ESP32-S3's capabilities in creating diverse IoT applications ranging from environmental monitoring to interactive gaming. The implementation adheres to principles of modularity, error handling, and user-centric design, making it adaptable to various IoT use cases.

References

1. Socket Programming Fundamentals
2. W3school for web page design

Appendices

Appendix A: Environmental Monitoring System Code

[https://github.com/Muneeb-0/IoT_Assignment_WebServer/blob/main/rgb_temp_message_webPage.py]

Appendix B: Dice Game System Code

[https://github.com/Muneeb-0/IoT_Assignment_WebServer/blob/main/game_chat_server_webpage.py]