General Description

The `main.py` file contains the main code for controlling an ESP32 using MicroPython. It provides the following features:

- A **LED clock** that displays the current time using RGB LEDs.
- A **web interface** for interacting with the ESP32.
- **Temperature and humidity monitoring** using a DHT11 sensor.
- A **church bell simulation** that rings at regular intervals.
- A **colorful LED animation**.

The program uses MicroPython libraries for handling LEDs, an OLED display, the DHT11 sensor, and a web server.

Key Features

1. WiFi Connection

The ESP32 connects to a WiFi network using credentials provided in the code. Once connected, the device's IP address is displayed on the OLED screen.

2. LED Clock

The LED clock uses an RGB LED ring to display the current time: - **Red**: Hours. - **Green**: Minutes. - **Blue**: Seconds.

The clock can be toggled on or off via the web interface.

3. Church Bell

A buzzer simulates a church bell that rings at specific intervals: - 1 chime every hour. - 2 chimes every half-hour. - 3 chimes every quarter-hour.

4. Temperature and Humidity Monitoring

The DHT11 sensor measures temperature and humidity, which can be accessed via the web interface.

5. LED Animation

A colorful animation sequence is displayed on the LED ring.

6. Web Interface

A web server allows users to: - View temperature and humidity data. - Enable or disable the LED clock. - Start the LED animation. - Set the real-time clock (RTC).

Code Overview

1. Initialization

- **Constants**: Define the number of LEDs, GPIO pins, and other hardware configurations.
- **WiFi Setup**: Connects to a WiFi network and displays the IP address on the OLED screen.
- **RTC**: Initializes the real-time clock for timekeeping.

2. Core Functions

a. OLED Initialization Initializes the OLED display using the I2C protocol.

b. Display Information Displays the IP address, LED clock status, and current time on the OLED screen.

c. Church Bell Simulates a church bell that rings at intervals (hourly, half-hourly, and quarterly).

d. LED Clock Lights up LEDs on the ring to represent hours, minutes, and seconds.

e. LED Animation Displays a sequence of colors on the LED ring.

f. Temperature and Humidity Reads temperature and humidity data from the DHT11 sensor.

---

3. Web Server

The web server is built using the `Microdot` library. Key routes include:

- `/`: Serves the main HTML page.
- `/init_rtc`: Sets the RTC with a provided date and time.
- `/sensor_data`: Returns temperature and humidity data.
- `/clock_status`: Returns the current status of the LED clock.
- `/enable_led_clock`: Enables the LED clock.
- `/disable_led_clock`: Disables the LED clock.
- `/led_animation`: Starts the LED animation.

Hardware Setup

- **ESP32**: Main microcontroller.
- **DHT11**: Temperature and humidity sensor.

- **RGB LED Ring**: Displays the LED clock and animations.
- **Buzzer**: Simulates the church bell.
- **OLED Display**: Shows device information.

Usage

1. Update the WiFi credentials in the code.
2. Deploy the `main.py` file to the ESP32.
3. Access the web interface using the IP address displayed on the OLED screen.
4. Use the web interface to interact with the device (e.g., enable the LED clock, view sensor data).

Summary of Functions

Function	Description
`init_oled()`	Initializes the OLED display.
`display_oled_info()`	Displays IP address, clock status, and time on the OLED.
`play_church_bell()`	Simulates a church bell that rings at intervals.
`enable_led_clock()`	Activates the LED clock to display the current time.
`animation()`	Displays a colorful animation on the LED ring.
`get_temphum()`	Reads temperature and humidity data from the DHT11 sensor.
`main()`	Starts the web server and updates the OLED display.

Dependencies

- **Microdot**: For the web server.
- **ssd1306**: For the OLED display.
- **dht**: For the DHT11 sensor.
- **neopixel**: For controlling the RGB LEDs.

Example Workflow

1. **Start the Device**: The ESP32 connects to WiFi and displays its IP address on the OLED.
2. **Access the Web Interface**: Open the IP address in a browser to interact with the device.

3. **Enable the LED Clock**: Use the web interface to start the LED clock. Terminal output: ao ao 54% ↓ 21:09
4. **View Sensor Data**: Check the temperature and humidity readings via the web interface.
5. **Run LED Animation**: Trigger the LED animation for a colorful display.

“‘