src/main.cpp

```cpp
#include <Arduino.h>
#include <EEPROM.h>
#include <ESP32Servo.h>

#include "freertos/FreeRTOS.h"
#include "freertos/event_groups.h"
#include "freertos/task.h"
#include "freertos/timers.h"
#include "pin_config.h"
#include "time_func.h"
#include "ui/lv_setup.h"
#include "ui/ui.h"
bool shouldTakePill = false;
int servoPin = 13;
int buzzerPin = 12;
Servo myservo;
int pos = 0;

int BUTTON1 = 0;   // Define the pin for BUTTON1
int BUTTON2 = 14;  // Define the pin for BUTTON2

enum {
    EDITING_HOUR,
    EDITING_MINUTE,
    EDITING_SECOND
} EDITING_MODE;

bool isCounting = true;
bool isEditing = false;
int editing = EDITING_HOUR;

void setInterval(uint32_t interval) {
    EEPROM.write(0, (interval >> 24) & 0xFF);  // Most significant byte
    EEPROM.write(1, (interval >> 16) & 0xFF);
    EEPROM.write(2, (interval >> 8) & 0xFF);
    EEPROM.write(3, interval & 0xFF);  // Least significant byte
    EEPROM.commit();                   // Required for ESP32
}

uint32_t getInterval() {
    return (EEPROM.read(0) << 24) |
           (EEPROM.read(1) << 16) |
           (EEPROM.read(2) << 8) |
           EEPROM.read(3);
}

uint32_t getRemainingTime() {
    // Read 32-bit remaining time from EEPROM addresses 4-7
    return (EEPROM.read(4) << 24) |
           (EEPROM.read(5) << 16) |
           (EEPROM.read(6) << 8) |
```

```
52            EEPROM.read(7);
53  }
54
55  void setRemainingTime(uint32_t remainingTime) {
56      // Write 32-bit remaining time to EEPROM addresses 4-7
57      EEPROM.write(4, (remainingTime >> 24) & 0xFF);
58      EEPROM.write(5, (remainingTime >> 16) & 0xFF);
59      EEPROM.write(6, (remainingTime >> 8) & 0xFF);
60      EEPROM.write(7, remainingTime & 0xFF);
61      EEPROM.commit();  // Required for ESP32
62  }
63
64  void lv_set_times(uint32_t interval, uint32_t remainingTime) {
65      char interval_buf[16];
66      char remainingTime_buf[16];
67      // interval and remainingTime are in seconds, we want to represent them in
    hours, minutes and seconds
68      uint32_t interval_hours = interval / 3600;
69      uint32_t interval_minutes = (interval % 3600) / 60;
70      uint32_t interval_seconds = interval % 60;
71      uint32_t remainingTime_hours = remainingTime / 3600;
72      uint32_t remainingTime_minutes = (remainingTime % 3600) / 60;
73      uint32_t remainingTime_seconds = remainingTime % 60;
74      sprintf(remainingTime_buf, "%02d:%02d:%02d", remainingTime_hours,
    remainingTime_minutes, remainingTime_seconds);
75      char interval_Hours_buf[16];
76      char interval_Minutes_buf[16];
77      char interval_Seconds_buf[16];
78      sprintf(interval_Hours_buf, "%02dHr", interval_hours);
79      sprintf(interval_Minutes_buf, "%02dMin", interval_minutes);
80      sprintf(interval_Seconds_buf, "%02dSec", interval_seconds);
81      lv_label_set_text(ui_Interval_Hours_Value, interval_Hours_buf);
82      lv_label_set_text(ui_Interval_Minutes_Value, interval_Minutes_buf);
83      lv_label_set_text(ui_Interval_Seconds_Value, interval_Seconds_buf);
84      lv_label_set_text(ui_Next_Pill_Time_Value, remainingTime_buf);
85      setRemainingTime(remainingTime);
86  }
87
88  void buzzerTask(void* parameter) {
89      while (true) {
90          if (shouldTakePill) {
91              tone(buzzerPin, 1000);
92              delay(1000);
93              noTone(buzzerPin);
94              delay(1000);
95          } else {
96              delay(1000);
97          }
98      }
99  }
100
101 void buttonTask(void* parameter) {
102     pinMode(BUTTON1, INPUT);
103     pinMode(BUTTON2, INPUT);
```

```
104     bool b1Pressed = false;
105     bool b2Pressed = false;
106     double b2PressedTime = 0;
107     while (true) {
108         bool b1 = !digitalRead(BUTTON1);
109         bool b2 = !digitalRead(BUTTON2);
110         if (b1 && !b1Pressed) {
111             b1Pressed = true;
112             Serial.println("Button 1 pressed");
113             shouldTakePill = false;
114             if (isEditing) {
115                 uint32_t interval = getInterval();
116                 uint32_t interval_hours = interval / 3600;
117                 uint32_t interval_minutes = (interval % 3600) / 60;
118                 uint32_t interval_seconds = interval % 60;
119                 if (editing == EDITING_HOUR) {
120                     interval_hours += 1;
121                 } else if (editing == EDITING_MINUTE) {
122                     interval_minutes += 15;
123                 } else if (editing == EDITING_SECOND) {
124                     interval_seconds += 5;
125                 }
126                 if (interval_seconds >= 60) {
127                     interval_seconds = 0;
128                 }
129                 if (interval_minutes >= 60) {
130                     interval_minutes = 0;
131                 }
132                 if (interval_hours >= 24) {
133                     interval_hours = 0;
134                 }
135                 interval = interval_hours * 3600 + interval_minutes * 60 +
    interval_seconds;
136                 setInterval(interval);
137                 lv_set_times(interval, interval);
138             }
139         } else if (!b1 && b1Pressed) {
140             b1Pressed = false;
141         }
142         if (b2 && !b2Pressed) {
143             b2Pressed = true;
144             b2PressedTime = millis();
145             Serial.println("Button 2 pressed");
146             if (isEditing) {
147                 editing = (editing + 1) % 3;
148             }
149         } else if (!b2 && b2Pressed) {
150             if (millis() - b2PressedTime > 3000) {
151                 Serial.println("Button 2 long pressed");
152                 isCounting = !isCounting;
153                 isEditing = !isEditing;
154             }
155             b2Pressed = false;
156             b2PressedTime = 0;
```

```
157         }
158         // Serial.println("Button 1: " + String(b1));
159         // Serial.println("Button 2: " + String(b2));
160         delay(1);
161     }
162 }
163 
164 void editTask(void* parameter) {
165     bool onoff = false;
166     while (true) {
167         if (isEditing) {
168             if (editing == EDITING_HOUR) {
169                 lv_obj_set_style_text_color(ui_Interval_Hours_Value,
    lv_color_hex(0xFB0000), LV_PART_MAIN | LV_STATE_DEFAULT);
170                 lv_obj_set_style_text_color(ui_Interval_Minutes_Value,
    lv_color_hex(0x000000), LV_PART_MAIN | LV_STATE_DEFAULT);
171                 lv_obj_set_style_text_color(ui_Interval_Seconds_Value,
    lv_color_hex(0x000000), LV_PART_MAIN | LV_STATE_DEFAULT);
172             } else if (editing == EDITING_MINUTE) {
173                 lv_obj_set_style_text_color(ui_Interval_Hours_Value,
    lv_color_hex(0x000000), LV_PART_MAIN | LV_STATE_DEFAULT);
174                 lv_obj_set_style_text_color(ui_Interval_Minutes_Value,
    lv_color_hex(0xFB0000), LV_PART_MAIN | LV_STATE_DEFAULT);
175                 lv_obj_set_style_text_color(ui_Interval_Seconds_Value,
    lv_color_hex(0x000000), LV_PART_MAIN | LV_STATE_DEFAULT);
176             } else if (editing == EDITING_SECOND) {
177                 lv_obj_set_style_text_color(ui_Interval_Hours_Value,
    lv_color_hex(0x000000), LV_PART_MAIN | LV_STATE_DEFAULT);
178                 lv_obj_set_style_text_color(ui_Interval_Minutes_Value,
    lv_color_hex(0x000000), LV_PART_MAIN | LV_STATE_DEFAULT);
179                 lv_obj_set_style_text_color(ui_Interval_Seconds_Value,
    lv_color_hex(0xFB0000), LV_PART_MAIN | LV_STATE_DEFAULT);
180             }
181             onoff = !onoff;
182         } else {
183             lv_obj_set_style_text_color(ui_Interval_Hours_Value,
    lv_color_hex(0x000000), LV_PART_MAIN | LV_STATE_DEFAULT);
184             lv_obj_set_style_text_color(ui_Interval_Minutes_Value,
    lv_color_hex(0x000000), LV_PART_MAIN | LV_STATE_DEFAULT);
185             lv_obj_set_style_text_color(ui_Interval_Seconds_Value,
    lv_color_hex(0x000000), LV_PART_MAIN | LV_STATE_DEFAULT);
186         }
187         delay(100);
188     }
189 }
190 
191 void setup() {
192     EEPROM.begin(128);
193     Serial.begin(115200);
194     lv_begin();  // Initialize lvgl with display and touch
195     ui_init();   // Initialize UI generated by Square Line
196 
197     // setInterval(10);       // Set interval to 1 hour
198     // setRemainingTime(10);  // Set remaining time to 1 hour
199     Serial.println("Interval: " + String(getInterval()));
```

```
200     Serial.println("Remaining Time: " + String(getRemainingTime()));
201     lv_set_times(10, 10);
202 
203     ESP32PWM::allocateTimer(0);
204     ESP32PWM::allocateTimer(1);
205     ESP32PWM::allocateTimer(2);
206     ESP32PWM::allocateTimer(3);
207     myservo.setPeriodHertz(50);  // standard 50 hz servo
208     // myservo.attach(servoPin, 1000, 2000);  // attaches the servo on pin 18 to the servo object
209 
210     pinMode(buzzerPin, OUTPUT);
211     noTone(buzzerPin);
212     digitalWrite(buzzerPin, HIGH);
213 
214     xTaskCreate(buzzerTask, "buzzerTask", 2048, NULL, 1, NULL);
215     xTaskCreate(buttonTask, "buttonTask", 2048, NULL, 1, NULL);
216     xTaskCreate(editTask, "editTask", 2048, NULL, 1, NULL);
217 
218     setInterval(3600);
219     setRemainingTime(3600);
220 }
221 
222 void loop() {
223     lv_handler();   // Update UI
224     update_time();  // Update time and date on UI
225     if (isCounting) {
226         uint32_t interval = getInterval();
227         uint32_t remainingTime = getRemainingTime();
228         if (remainingTime > 0) {
229             remainingTime -= 1;
230             lv_set_times(interval, remainingTime);
231             setRemainingTime(remainingTime);
232             if (remainingTime <= 0) {
233                 // Turn off the relay
234                 myservo.attach(servoPin, 1000, 2000);      // attaches the servo on pin 18 to the servo object
235                 for (pos = 3; pos <= 48 + 45; pos += 1) {  // goes from 0 degrees to 180 degrees
236                     // in steps of 1 degree
237                     myservo.write(pos);  // tell servo to go to position in variable 'pos'
238                     delay(2);            // waits 15 ms for the servo to reach the position
239                 }
240                 delay(200);
241                 for (pos = 48 + 45; pos >= 3; pos -= 1) {  // goes from 180 degrees to 0 degrees
242                     myservo.write(pos);                    // tell servo to go to position in variable 'pos'
243                     delay(2);                              // waits 15 ms for the servo to reach the position
244                 }
245                 myservo.detach();
246                 shouldTakePill = true;
```

```
            }
        } else {
            if (interval > 0) {
                remainingTime = interval;
                setRemainingTime(remainingTime);
                lv_set_times(interval, remainingTime);
                // Turn on the relay
                // shouldTakePill = false;
            }
        }
    }
    delay(1000);

    Serial.println("Interval: " + String(getInterval()));
    Serial.println("Remaining Time: " + String(getRemainingTime()));
}
```