Comsats University Islamabad

Abbottabad Campus

Real Time Embedded System

Lab Task # 1

Submitted by,

Aazan Ali Khan	FA20-EEE-026

Submitted to,

Dr. Syed Mashood Murtaza

Department of Electrical &

Computer Engineering

Task 1:

Esp32 on Wokwi:

The following code is for an ESP32 microcontroller, and it controls five LEDs (Red, Yellow, Green, White, Blue) connected to different GPIO pins. The LEDs start blinking in a pattern when a button is pressed. The blinking pattern can be toggled between LSB-to-MSB (left to right) and MSB-to-LSB (right to left) based on button presses. The current LED that is on is printed to the Serial Monitor.

Code:

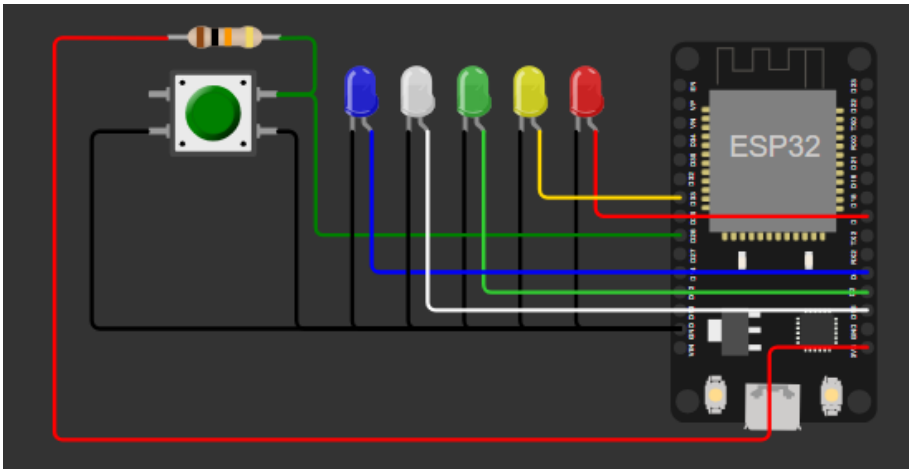
```
#include <Arduino.h>

#include "freertos/FreeRTOS.h"

#include "freertos/task.h"

#include "driver/gpio.h"


// LED pin definitions

#define LED_RED GPIO_NUM_5

#define LED_GREEN GPIO_NUM_2

#define LED_BLUE GPIO_NUM_4

#define LED_YELLOW GPIO_NUM_33

#define LED_WHITE GPIO_NUM_15

#define BUTTON GPIO_NUM_26


// LED pin array in the order: Red, Yellow, Green, White, Blue

gpio_num_t leds[] = {LED_RED, LED_YELLOW, LED_GREEN, LED_WHITE,
LED_BLUE};

const char* led_names[] = {"Red", "Yellow", "Green", "White", "Blue"};


volatile bool led_running = false; // Start with LEDs off

volatile bool button_pressed = false;

volatile uint32_t button_press_time = 0;
```

```
volatile bool pattern_lsb_to_msb = true;

void IRAM_ATTR handleButtonPress() {
  if (millis() - button_press_time > 50) { // debounce
    button_pressed = true;
    button_press_time = millis();
  }
}

// Function to control the LED pattern
void pattern_task(void *parameter) {
  int led_index = 0;
  int num_leds = sizeof(leds) / sizeof(leds[0]);

  while (1) {
    if (led_running) {
      // Turn off all LEDs
      for (int i = 0; i < num_leds; i++) {
        gpio_set_level(leds[i], 0);
      }

      // Turn on the current LED and print its name
      gpio_set_level(leds[led_index], 1);
      Serial.print("LED On: ");
      Serial.println(led_names[led_index]);

      // Calculate the next LED index
      if (pattern_lsb_to_msb) {
```

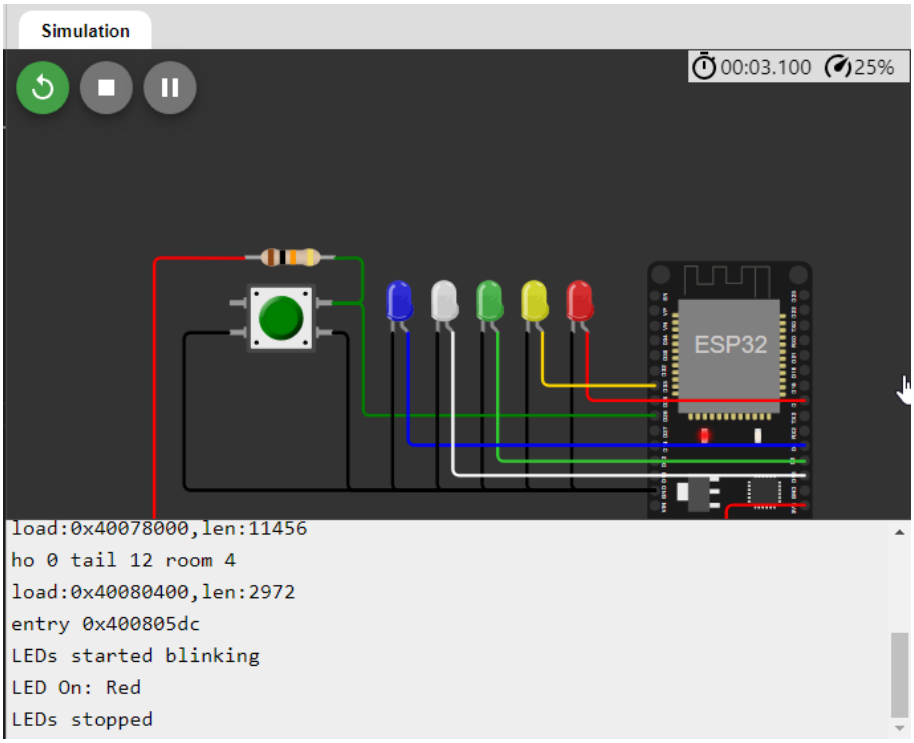
```
      led_index = (led_index + 1) % num_leds;
    } else {
      led_index = (led_index - 1 + num_leds) % num_leds;
    }

    vTaskDelay(500 / portTICK_PERIOD_MS);
  } else {
    // Turn off all LEDs when not running
    for (int i = 0; i < num_leds; i++) {
      gpio_set_level(leds[i], 0);
    }
    vTaskDelay(100 / portTICK_PERIOD_MS);
  }
 }
 vTaskDelete(NULL);
}

// Function to handle button press
void button_task(void *parameter) {
 pinMode(BUTTON, INPUT_PULLUP);
 attachInterrupt(digitalPinToInterrupt(BUTTON), handleButtonPress, FALLING);

 while (1) {
  if (button_pressed) {
    led_running = !led_running;
    button_pressed = false; // Reset the button_pressed flag
    Serial.println(led_running ? "LEDs started blinking" : "LEDs stopped");
  }
```

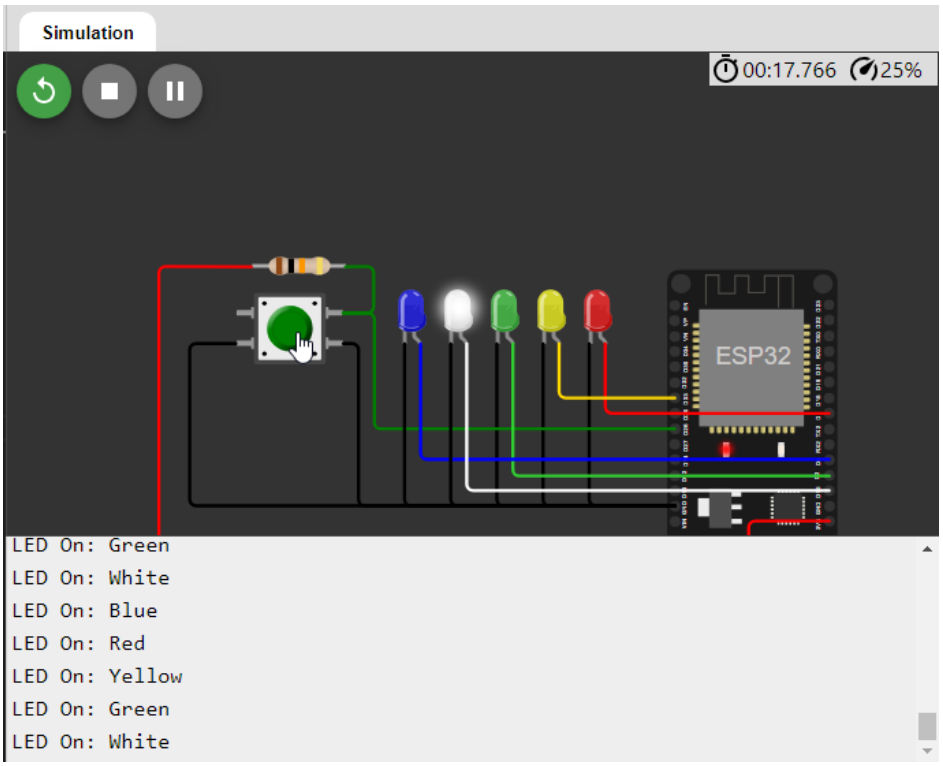
```
    vTaskDelay(100 / portTICK_PERIOD_MS);
  }
}

void setup() {
  Serial.begin(115200);

  // Configure the LED pins as output
  for (int i = 0; i < sizeof(leds) / sizeof(leds[0]); i++) {
    gpio_reset_pin(leds[i]);
    gpio_set_direction(leds[i], GPIO_MODE_OUTPUT);
  }

  // Create the pattern task
  xTaskCreatePinnedToCore(
    pattern_task, "pattern_task", 2048, NULL, 5, NULL, 1);

  // Create the button task
  xTaskCreatePinnedToCore(
    button_task, "button_task", 1024, NULL, 3, NULL, 1);
}

void loop() {
  // Loop is empty because tasks are running in FreeRTOS
}
```

Circuit Diagram:



Simulation:




Simulation
00:17.766
25%
ESP32
LED On: Green
LED On: White
LED On: Blue
LED On: Red
LED On: Yellow
LED On: Green
LED On: White