INSTITUTE OF TECHNOLOGY OF CAMBODIA

DEPARTMENT OF ELECTRICAL AND ENERGY ENGINEERING

Embedded Electronics

Lab Part1 Report: TP-02 (ESP32)

PREPARED FOR:

Mr. TEP Sovichea

PREPARED BY:

HUT Hokkey	**e20180335**
CHHUN Pichpisal	**e20180149**
HAK Menghour	**e20180239**

Engineering's Degree

Department of Electrical and Energy Engineering

Institute of Technology of Cambodia

2022-2023

Task

TP-02:

1. 1.Task1: handle LED blink rate (timer)
2. 2.Task2: handle ADC read every 50ms(vTaskDelay)
3. 3.Task3: handle UART command(Task)

Content

I. Introduction

Software timers are used to schedule the execution of a function at a set time in the future, or periodically with a fixed frequency. The function executed by the software timer is called the software timer's callback function. Software timers are implemented by and are under the control of, the FreeRTOS kernel. They do not require hardware support and are not related to hardware timers or hardware Counters. Note that, in line with the FreeRTOS philosophy of using innovative design to ensure maximum efficiency, software timers do not use any processing time unless a software timer callback function is actually executing.

II. Objective

- Understanding the process of using software timer management.
- Be able to create the UART console task to control the two timers.
- Create the Alert LED control task to toggle fast for some milliseconds and pause for some milliseconds
- then set alert on or off from the UART console command.
- Create the ADC control task to connect one of the ADC1 GPIOs to a potentiometer; then use the following
- Command to read the average values of the potentiometer after that an implementation on the ADC control
- Command to allow multiple channel selections from the UART console command.

III. Procedure

The purpose of the laboratory is to design the three console commands to control two software timers. The UART console task should be considered first before other tasks since it received a command from the CMD console and processed the command to control the software timers.

1.Task1: handle LED blink rate (timer)

```c
void vTaskBlinkLED (TimerHandle_t xTimer) {
    if (count == 0) {
        xTimerChangePeriod(xTimer,pdMS_TO_TICKS(period_short), 0);

    }
    else if (count == 8) {
        xTimerChangePeriod(xTimer,pdMS_TO_TICKS(period_long), 0);
        count = 0;
        led_state = 0;
        gpio_set_level(LED1_GPIO, led_state); //
        return;
    }

    count++;
    led_state = !led_state;
    gpio_set_level(LED1_GPIO, led_state);
}

```

We set timer, the LED will on every 100ms and stop 1ms. We adjust short period 100ms and long period 1000ms.

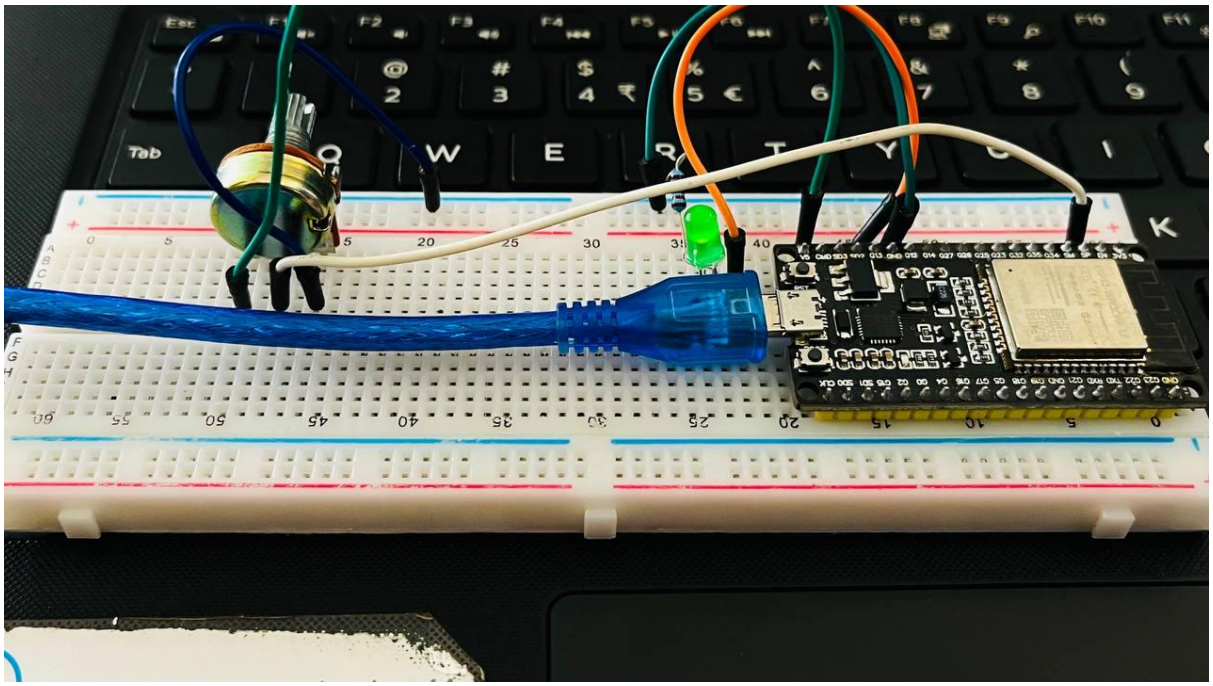


Figure 1:Handle LED1 Blink rate

2.Task2: handle ADC read every 50ms(vTaskDelay)

```c
void adc_init(adc_channel_t adc_cha) {

    /* ADC Config */
    adc1_config_width(width);
    adc1_config_channel_atten(adc_cha, atten);
    adc_chars = calloc(1, sizeof(esp_adc_cal_characteristics_t));
    esp_adc_cal_value_t val_type = esp_adc_cal_characterize(unit, atten, width, DEFAULT_VREF, adc_chars);
}
```

```c
void adc_callback (TimerHandle_t xTimer) {
    if (iter < num_of_cycle) {
        adc_reading += adc1_get_raw((adc1_channel_t)channel[ch_id]);
        xTimerChangePeriod(xTimer, pdMS_TO_TICKS(sample_time),0);
        iter++;
        printf("ADC Reading = %d and Num_of_Cycle = %d\n",adc_reading, iter);
    }
    else if (iter == num_of_cycle) {
        adc_reading /= num_of_cycle;
        uint32_t voltage = esp_adc_cal_raw_to_voltage(adc_reading, adc_chars);
        xTimerStop(ADCTimer,0);
        ESP_LOGI(TAG_ADC, ">>> ADC Channel[%d]:  Raw: %d  Voltage: %dmV\n", ch_id,adc_reading, voltage);
        adc_reading = 0;
        iter = 0;
    }

}
```

We set timer, the LED will on every 100ms and stop 1ms. We adjust short period 100ms and long period 1000ms.

```
adc 50 20
I (31742) UART Console: adc 50 20
I (31742) Read_CMD_Info: ADC Command recieved ADC Channel 0: sample_time: 50 and num_of_cycle: 20
ADC Reading = 4095 and Num_of_Cycle = 1
ADC Reading = 8190 and Num_of_Cycle = 2
ADC Reading = 12285 and Num_of_Cycle = 3
ADC Reading = 16380 and Num_of_Cycle = 4
ADC Reading = 20475 and Num_of_Cycle = 5
ADC Reading = 24570 and Num_of_Cycle = 6
ADC Reading = 28665 and Num_of_Cycle = 7
ADC Reading = 32760 and Num_of_Cycle = 8
ADC Reading = 36855 and Num_of_Cycle = 9
ADC Reading = 40950 and Num_of_Cycle = 10
ADC Reading = 45045 and Num_of_Cycle = 11
ADC Reading = 49140 and Num_of_Cycle = 12
ADC Reading = 53235 and Num_of_Cycle = 13
ADC Reading = 57330 and Num_of_Cycle = 14
ADC Reading = 61425 and Num_of_Cycle = 15
ADC Reading = 65520 and Num_of_Cycle = 16
ADC Reading = 69615 and Num_of_Cycle = 17
ADC Reading = 73710 and Num_of_Cycle = 18
ADC Reading = 77805 and Num_of_Cycle = 19
ADC Reading = 81900 and Num_of_Cycle = 20
I (32752) ADC_Info: >>> ADC Channel[0]:  Raw: 4095  Voltage: 3134mV
```

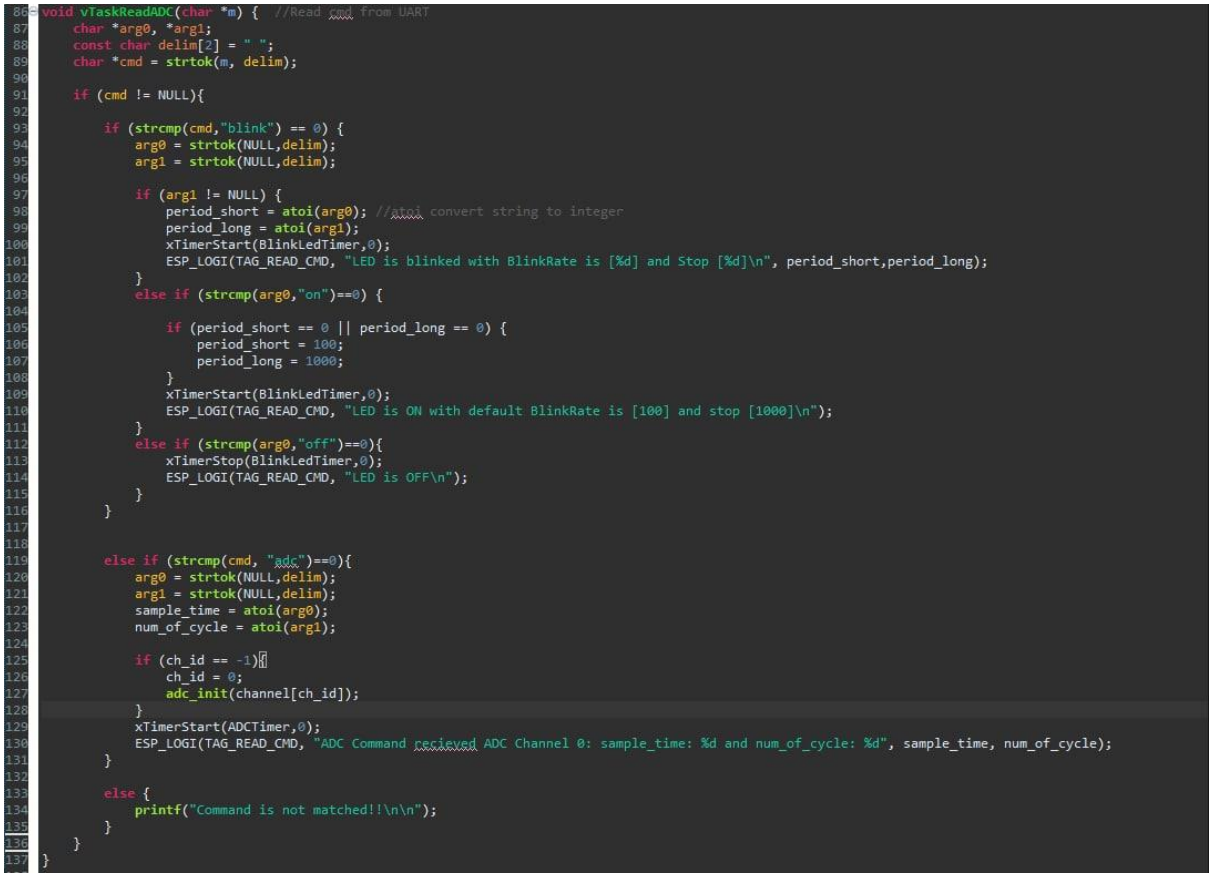
Figure 2:Handle ADC read every50ms

3.Task3: handle UART command (Task)

```c
void vTaskHandleUART (void *pvParameters){
    char input_cmd[128];
    uint8_t count = 0;

    while (1) {
        uint8_t ch;
        ch = getchar();

        if (ch != 0xff) {
            //echo character to the console
            putchar(ch);
            if (ch == '\n') {
                printf("\n");
                input_cmd[count] = '\0';
                count = 0;
                ESP_LOGI(TAG_UART_CONSOLE, "%s", input_cmd);
                vTaskReadADC((char*)&input_cmd);
            }
            else if (ch == '\b'){
                if (count > 0) {
                    count--;
                    input_cmd[count] = '\0';
                    putchar(' ');
                    putchar('\b');
                }
            }
            else {
                input_cmd[count] = ch;
                count++;
            }
        }

        vTaskDelay(pdMS_TO_TICKS(10));
    }
}
```

```
Getting real time stats over 1000 ms
-------------------------------------------------------------------------
|       Task       | Core | Stack (byte) | Run Time (us) | Percentage |
-------------------------------------------------------------------------
|  vTaskHandleUAR  |   1  |         1220 |          5900 |      0.30% |
| main             |   0  |         2248 |           929 |      0.05% |
| IDLE             |   1  |         1112 |        994100 |     49.71% |
| IDLE             |   0  |         1108 |        999071 |     49.96% |
| esp_timer        |   0  |         1076 |             0 |      0.00% |
| ipc1             |   0  |         3652 |             0 |      0.00% |
| Tmr Svc          |   1  |         1104 |             0 |      0.00% |
| ipc0             |   0  |         1596 |             0 |      0.00% |
-------------------------------------------------------------------------
blink on
I (10172) UART Console: blink on
I (10172) Read_CMD_Info: LED is ON with default BlinkRate is [100] and stop [1000]

blink off
I (16292) UART Console: blink off
I (16292) Read_CMD_Info: LED is OFF
```

Figure 3:Read Command in UART



```
void vTaskReadADC(char *m) {  //Read cmd from UART
    char *arg0, *arg1;
    const char delim[2] = " ";
    char *cmd = strtok(m, delim);

    if (cmd != NULL){

        if (strcmp(cmd,"blink") == 0) {
            arg0 = strtok(NULL,delim);
            arg1 = strtok(NULL,delim);

            if (arg1 != NULL) {
                period_short = atoi(arg0); //atoi convert string to integer
                period_long = atoi(arg1);
                xTimerStart(BlinkLedTimer,0);
                ESP_LOGI(TAG_READ_CMD, "LED is blinked with BlinkRate is [%d] and Stop [%d]\n", period_short,period_long);
            }
            else if (strcmp(arg0,"on")==0) {

                if (period_short == 0 || period_long == 0) {
                    period_short = 100;
                    period_long = 1000;
                }
                xTimerStart(BlinkLedTimer,0);
                ESP_LOGI(TAG_READ_CMD, "LED is ON with default BlinkRate is [100] and stop [1000]\n");
            }
            else if (strcmp(arg0,"off")==0){
                xTimerStop(BlinkLedTimer,0);
                ESP_LOGI(TAG_READ_CMD, "LED is OFF\n");
            }
        }


        else if (strcmp(cmd, "adc")==0){
            arg0 = strtok(NULL,delim);
            arg1 = strtok(NULL,delim);
            sample_time = atoi(arg0);
            num_of_cycle = atoi(arg1);

            if (ch_id == -1){
                ch_id = 0;
                adc_init(channel[ch_id]);
            }
            xTimerStart(ADCTimer,0);
            ESP_LOGI(TAG_READ_CMD, "ADC Command recieved ADC Channel 0: sample_time: %d and num_of_cycle: %d", sample_time, num_of_cycle);
        }

        else {
            printf("Command is not matched!!\n\n");
        }
    }
}
```

Figure 4:UART Console

IV. Conclusion

In conclusion, the purpose of this lab is to perceive the process of FreeRTOS software time. This timer has a limitation and restriction allowed to perform. The execution time should be short and should not call vTaskDealy() into the block state. Alert LED control use one command “alert” and 2 arguments “period_short” “period_long” allow LED to toggle fast and pause within a timer. However, this auto-reload timer could go to the Dormant state whenever receiving the command “alert off”. In addition, the ADC control uses a timer to do the average ADC value read from the potentiometer. Which is successfully read and minimized noise by using a ceramic capacitor. Last but not least, the user can choose the multiple ADC1 channels to read the average ADC values according to the number of samples and sample time input by a user.