REPORT LAB

EMBEDDED SYSTEM - CO3054

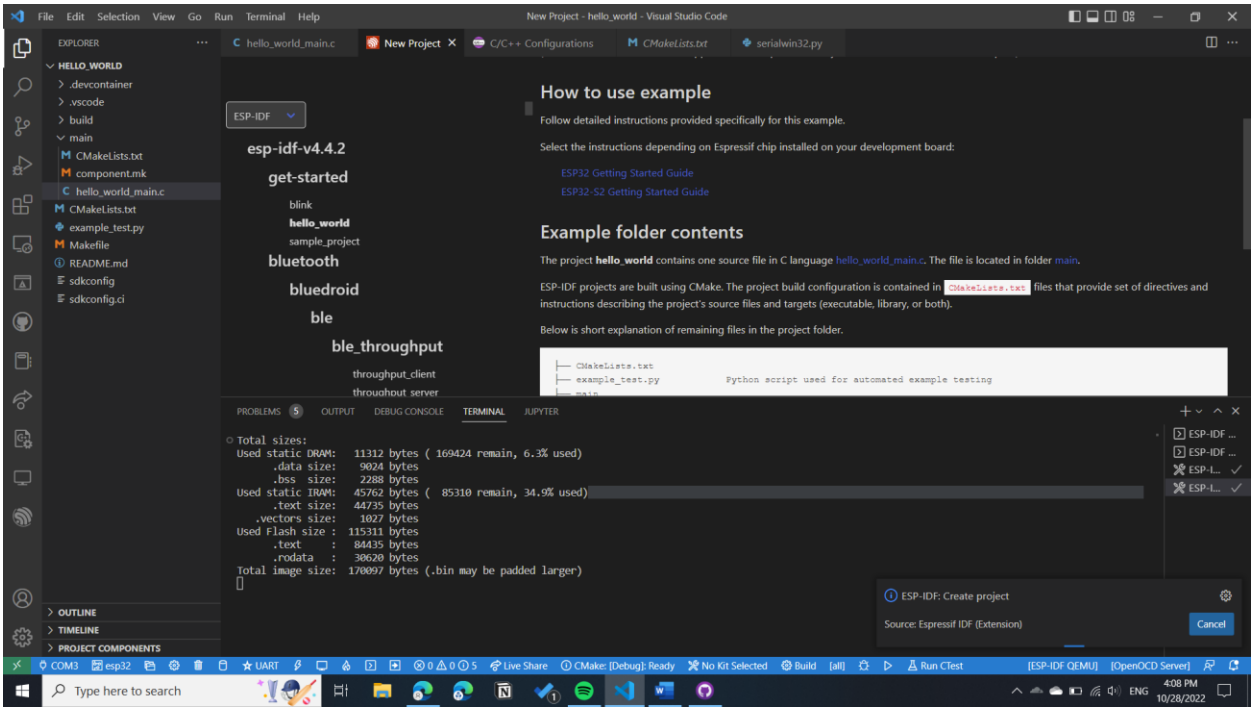
Group: **CC02**

Student: **Dương Gia An – 1952163**

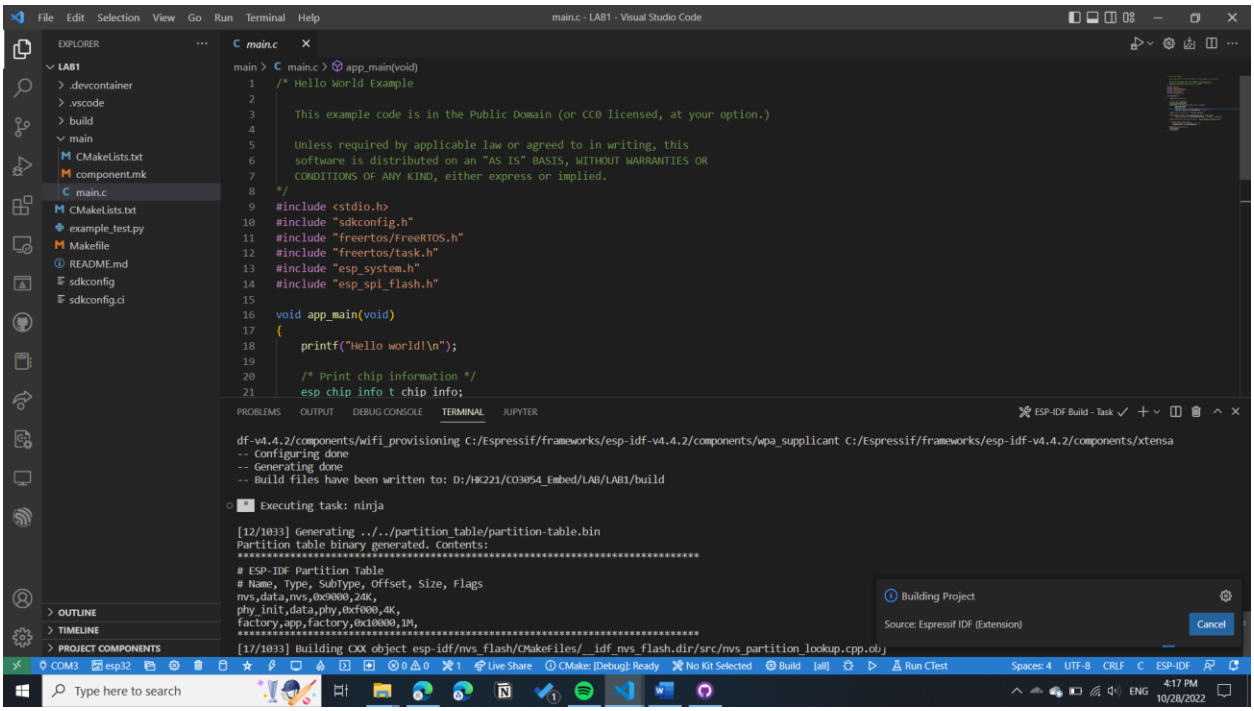
Contents

I. INTRODUCTION TO ESP32 AND ESP-IDF

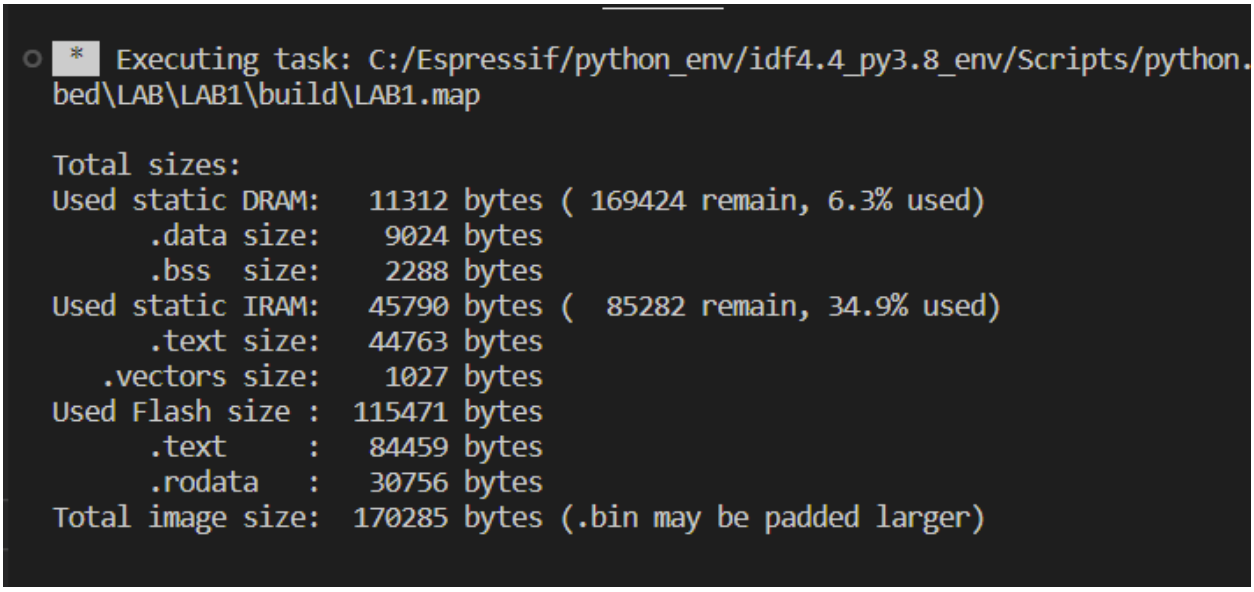
1. After install ESP-IDF extension on VS code, I create a project with HelloWorld Template.



2. Build the Project and connect ESP32 (COM3).



Finished Build



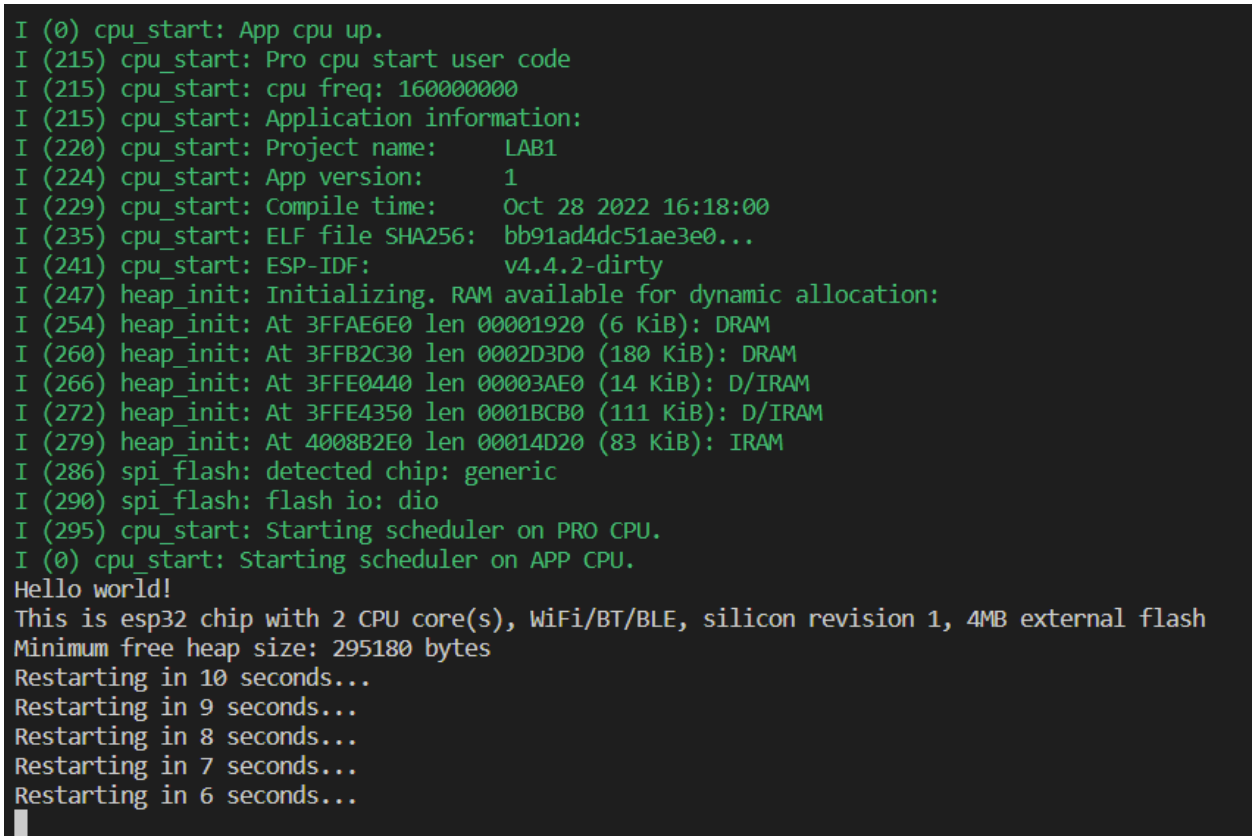
3. Flash (UART) project code to ESP32 by Press BOOT button on ESP32 while Flashing.

```
loader/bootloader.bin 0x10000 LAB1.bin 0x8000 partition_table/partition-table.bin

esptool.py v3.3.2-dev
Serial port COM3
Connecting....
Chip is ESP32-D0WDQ6 (revision 1)
Features: WiFi, BT, Dual Core, 240MHz, VRef calibration in efuse, Coding Scheme None
Crystal is 40MHz
MAC: 24:62:ab:d6:ef:e8
Uploading stub...
Running stub...
Stub running...
Changing baud rate to 460800
Changed.
Configuring flash size...
Auto-detected Flash size: 4MB
Flash will be erased from 0x00001000 to 0x00007fff...
Flash will be erased from 0x00010000 to 0x00039fff...
Flash will be erased from 0x00008000 to 0x00008fff...
Flash params set to 0x0220
Compressed 25392 bytes to 15887...
Wrote 25392 bytes (15887 compressed) at 0x00001000 in 0.7 seconds (effective 276.5 kbit/s)..
Hash of data verified.
Compressed 170416 bytes to 89840...
Wrote 170416 bytes (89840 compressed) at 0x00010000 in 2.3 seconds (effective 582.9 kbit/s).
Hash of data verified.
Compressed 3072 bytes to 103...
Wrote 3072 bytes (103 compressed) at 0x00008000 in 0.1 seconds (effective 435.1 kbit/s)...
Hash of data verified.

Leaving...
Hard resetting via RTS pin...
```

4. Press Monitor Device to see ESP execute flashed code.



II. ESP32 GPIO AND FREERTOS TASK

Code in file **main.c**

```
1.  #include <stdio.h>
2.  #include "sdkconfig.h"
3.  #include "freertos/FreeRTOS.h"
4.  #include "freertos/task.h"
5.  #include "esp_system.h"
6.  #include "esp_spi_flash.h"
7.
8.
9.  void print_id(void *pvParameter){
10.    while(1){
11.        printf("DUONG GIA AN : %d \n",1952163);
12.        vTaskDelay(1000/portTICK_PERIOD_MS);
13.
14.    }
15.    vTaskDelete(NULL);
16.
17.}
18.
19.void blinky(void *pvParameter){
20.    while(1){
21.        printf("Press Button\n");
22.        int  rd = rand() % (5000 + 1 - 0) + 0;
23.        vTaskDelay(rd /portTICK_PERIOD_MS);
24.    }
25.    vTaskDelete(NULL);
26.}
27.
28.
29.void app_main(){
30.    xTaskCreate(&print_id, "print_id", 2048, NULL, 0, NULL);
31.    xTaskCreate(&blinky, "blinky", 2048,NULL,0,NULL );
32.
33.    for (int i = 20; i >= 0; i--) {
34.        printf("Remaing %d seconds...\n", i);
35.        vTaskDelay(1000 / portTICK_PERIOD_MS);
36.    }
```

```
37.    printf("Restarting now.\n");
38.    vTaskDelay(5000 / portTICK_PERIOD_MS);
39.    fflush(stdout);
40.    esp_restart();
41.}
```

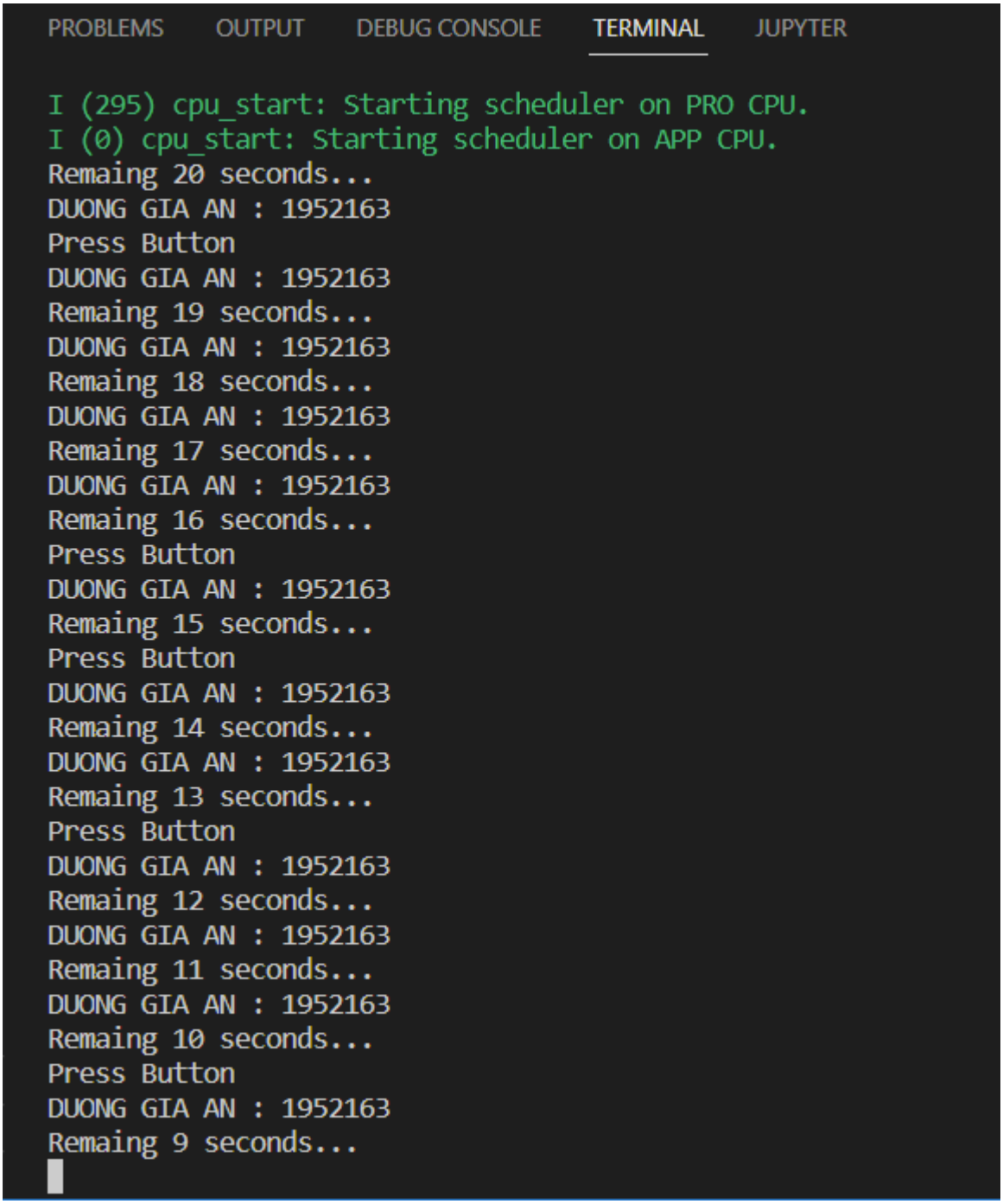
Link Github: CO3054_ES_LAB/LAB1 at main · kinggiaan/CO3054_ES_LAB (github.com)

Explain:

- Cyclic task: void **print_id()** is task that print my student ID every 2 seconds.
- Acylic task: void **Blinky()** is alternated for button in GPIO in ESP32. I change to a random time to press button from 0 – 5000ms.
- **app_main()** will print time stamp every 1 second and restart ESP after 20 seconds.

The *priority* and *usStackDepth* of task cyclic/acylic is the same as 0 and 2048 (mean 2048*4 bytes will be allocated for these tasks).

Result:


PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
JUPYTER
I (295) cpu_start: Starting scheduler on PRO CPU.
I (0) cpu_start: Starting scheduler on APP CPU.
Remaing 20 seconds...
DUONG GIA AN : 1952163
Press Button
DUONG GIA AN : 1952163
Remaing 19 seconds...
DUONG GIA AN : 1952163
Remaing 18 seconds...
DUONG GIA AN : 1952163
Remaing 17 seconds...
DUONG GIA AN : 1952163
Remaing 16 seconds...
Press Button
DUONG GIA AN : 1952163
Remaing 15 seconds...
Press Button
DUONG GIA AN : 1952163
Remaing 14 seconds...
DUONG GIA AN : 1952163
Remaing 13 seconds...
Press Button
DUONG GIA AN : 1952163
Remaing 12 seconds...
DUONG GIA AN : 1952163
Remaing 11 seconds...
DUONG GIA AN : 1952163
Remaing 10 seconds...
Press Button
DUONG GIA AN : 1952163
Remaing 9 seconds...

Does the ESP-IDF need the vTaskStartScheduler() routine?

No, because ESP-IDF will call vTaskStartScheduler() automatically.

Unlike Vanilla FreeRTOS, users must not call `vTaskStartScheduler()`. Instead, ESP-IDF FreeRTOS is started automatically. The entry point is a user defined `void app_main(void)` function.

- Typically, users would spawn the rest of their applications task from `app_main`.
- The `app_main` function is allowed to return at any point (i.e., before the application terminates).
- The `app_main` function is called from the `main` task.

The `main` task is one of multiple tasks that are automatically spawned by ESP-IDF during startup.[1]

[1] FreeRTOS - ESP32 - — ESP-IDF Programming Guide latest documentation (espressif.com)