

## EMBEDDED SYSTEMS – CC01

### LAB 1

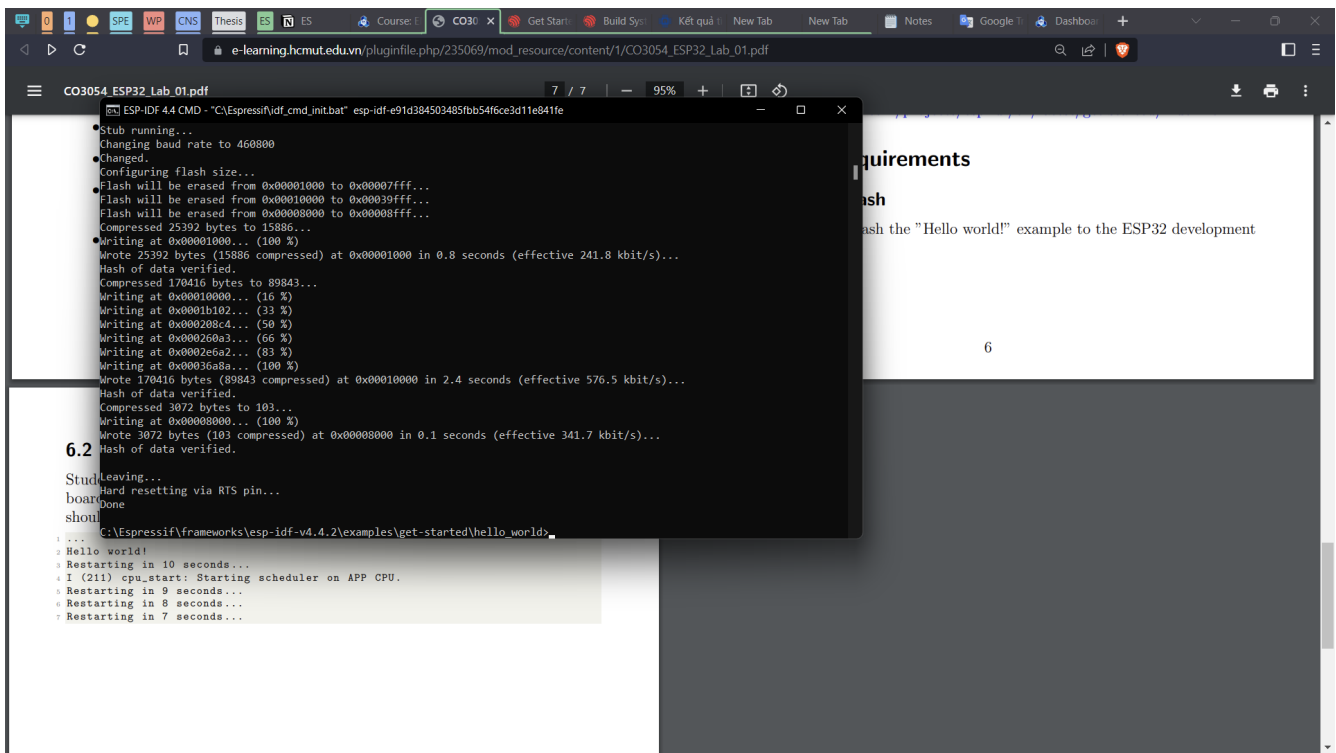
### ESP

#### ESP-L01

*Source code:*

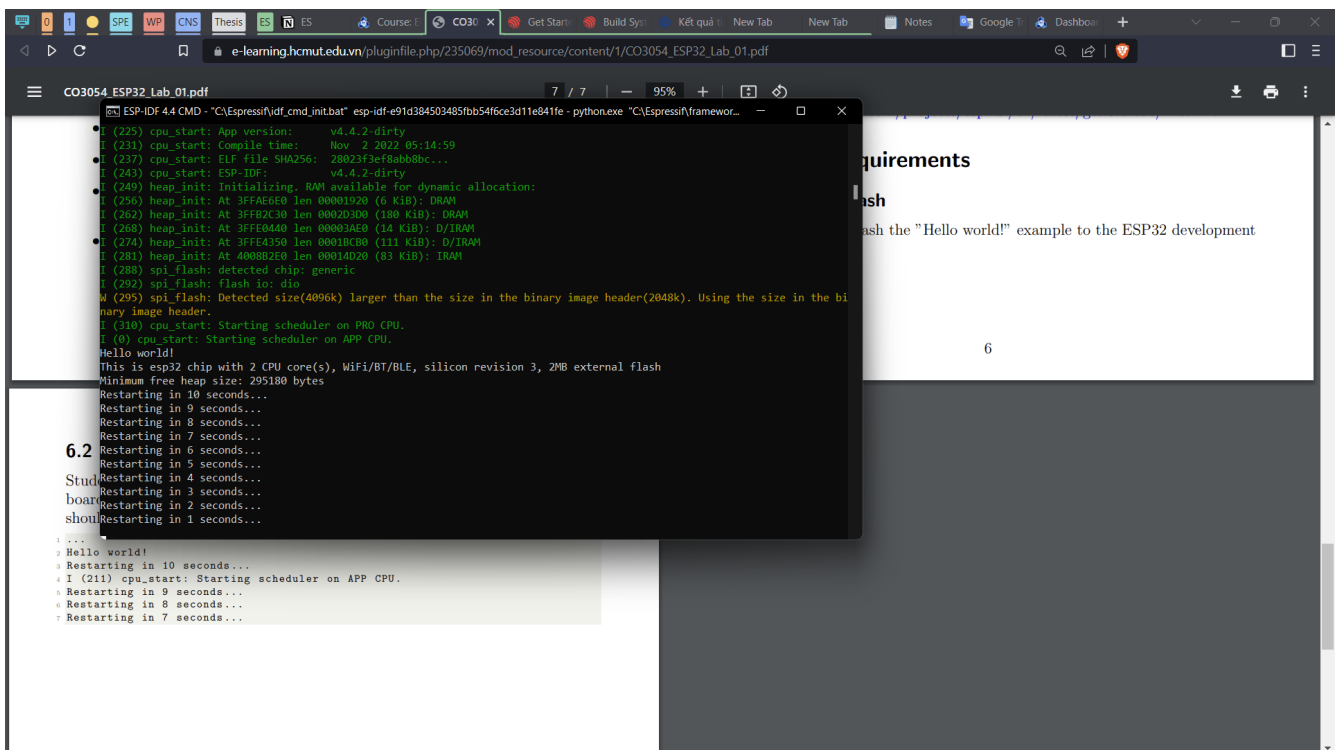
```
8  */
9  #include <stdio.h>
10 #include "sdkconfig.h"
11 #include "freertos/FreeRTOS.h"
12 #include "freertos/task.h"
13 #include "esp_system.h"
14 #include "esp_spi_flash.h"
15
16 void app_main(void)
17 {
18     printf("Hello world!\n");
19
20     /* Print chip information */
21     esp_chip_info_t chip_info;
22     esp_chip_info(&chip_info);
23     printf("This is %s chip with %d CPU core(s), WiFi%s, ",
24           CONFIG_IDF_TARGET,
25           chip_info.cores,
26           (chip_info.features & CHIP_FEATURE_BT) ? "/BT" : "",
27           (chip_info.features & CHIP_FEATURE_BLE) ? "/BLE" : "");
28
29     printf("silicon revision %d, ", chip_info.revision);
30
31     printf("%dMB %s flash\n", spi_flash_get_chip_size() / (1024 * 1024),
32           (chip_info.features & CHIP_FEATURE_EMB_FLASH) ? "embedded" : "external");
33
34     printf("Minimum free heap size: %d bytes\n", esp_get_minimum_free_heap_size());
35
36     for (int i = 10; i >= 0; i--) {
37         printf("Restarting in %d seconds...\n", i);
38         vTaskDelay(1000 / portTICK_PERIOD_MS);
39     }
40     printf("Restarting now.\n");
41     fflush(stdout);
42     esp_restart();
43 }
44
```

## Results:



```
ESP-IDF 4.4 CMD: "C:\Espressif\idf_cmd_init.bat" esp-idf-e91d384503485fbb54f6ce3d11e841fe
Stub running...
Changing baud rate to 460800
Changed.
Configuring flash size...
Flash will be erased from 0x00001000 to 0x00007fff...
Flash will be erased from 0x00010000 to 0x00039fff...
Flash will be erased from 0x00008000 to 0x00008fff...
Compressed 25392 bytes to 15886...
Writing at 0x00010000... (100 %)
Wrote 25392 bytes (15886 compressed) at 0x00010000 in 0.8 seconds (effective 241.8 kbit/s)...
Hash of data verified.
Compressed 170416 bytes to 89843...
Writing at 0x00010000... (16 %)
Writing at 0x0001b102... (33 %)
Writing at 0x000208c4... (50 %)
Writing at 0x000260a3... (66 %)
Writing at 0x0002e6a2... (83 %)
Writing at 0x00036a8a... (100 %)
Wrote 170416 bytes (89843 compressed) at 0x00010000 in 2.4 seconds (effective 576.5 kbit/s)...
Hash of data verified.
Compressed 3072 bytes to 103...
Writing at 0x00008000... (100 %)
Wrote 3072 bytes (103 compressed) at 0x00008000 in 0.1 seconds (effective 341.7 kbit/s)...
Hash of data verified.

6.2
Stub running...
Hard resetting via RTS pin...
Done
shout
1 ... C:\Espressif\frameworks\esp-idf-v4.4.2\examples\get-started\hello_world\
2 Hello world!
3 Restarting in 10 seconds...
4 I (211) cpu_start: Starting scheduler on APP CPU.
5 Restarting in 9 seconds...
6 Restarting in 8 seconds...
7 Restarting in 7 seconds...
```



```
ESP-IDF 4.4 CMD: "C:\Espressif\idf_cmd_init.bat" esp-idf-e91d384503485fbb54f6ce3d11e841fe - python.exe "C:\Espressif\frameworks\
(225) cpu_start: App version: v4.4.2-dirty
(231) cpu_start: Compile time: Nov 2 2022 05:14:59
(237) cpu_start: ELF file SHA256: 28023f3ef8abb8bc...
(243) cpu_start: ESP-IDF: v4.4.2-dirty
(249) heap_init: Initializing. RAM available for dynamic allocation:
(256) heap_init: At 3FFAE6E0 len 00001920 (6 KiB): DRAM
(262) heap_init: At 3FFB2C30 len 00002D08 (180 KiB): DRAM
(268) heap_init: At 3FFE9448 len 00003AE9 (14 KiB): D/IRAM
(274) heap_init: At 3FFEA350 len 0001BC00 (111 KiB): D/IRAM
(281) heap_init: At 400802E0 len 00014D20 (83 KiB): IRAM
(288) spi_flash: detected chip: generic
(292) spi_flash: flash io: dio
(295) spi_flash: Detected size(4096k) larger than the size in the binary image header(2048k). Using the size in the binary image header.
(310) cpu_start: Starting scheduler on PRO CPU.
(0) cpu_start: Starting scheduler on APP CPU.
Hello world!
This is esp32 chip with 2 CPU core(s), WiFi/BT/BLE, silicon revision 3, 2MB external flash
Minimum free heap size: 295180 bytes
Restarting in 10 seconds...
Restarting in 9 seconds...
Restarting in 8 seconds...
Restarting in 7 seconds...
Restarting in 6 seconds...
Restarting in 5 seconds...
Restarting in 4 seconds...
Restarting in 3 seconds...
Restarting in 2 seconds...
Restarting in 1 seconds...

6.2
Stub running...
Hard resetting via RTS pin...
Done
shout
1 ...
2 Hello world!
3 Restarting in 10 seconds...
4 I (211) cpu_start: Starting scheduler on APP CPU.
5 Restarting in 9 seconds...
6 Restarting in 8 seconds...
7 Restarting in 7 seconds...
```

## ESP-L02

*Source code:*

```
1  #include <stdio.h>
2  #include <stdbool.h>
3  #include <unistd.h>
4  #include "freertos/FreeRTOS.h"
5  #include "freertos/task.h"
6  #include "driver/gpio.h"
7  #include "esp_system.h"
8  #include "esp_spi_flash.h"
9
10 void cyclicktask()
11 {
12     while (1)
13     {
14         printf("1952521\n");
15         vTaskDelay(2200 / portTICK_PERIOD_MS);
16     }
17     vTaskDelete(NULL);
18 }
19
20 void acyclicktask()
21 {
22     while (1)
23     {
24         printf("ESP32\n");
25         vTaskDelay(3000 / portTICK_PERIOD_MS);
26     }
27     vTaskDelete(NULL);
28 }
29
30 void app_main(void)
31 {
32     xTaskCreate(
33         cyclicktask,
34         "cyclic ",
35         1024 * 2,
36         NULL,
37         1,
38         NULL);
39     xTaskCreate(
40         acyclicktask,
41         "acyclic ",
42         1024 * 2,
43         NULL,
44         1,
45         NULL);
46 }
```

## Results:

The screenshot shows a web browser window displaying a PDF document titled "CO3054\_ESP32\_Lab\_02.pdf". A terminal window is overlaid on the PDF, showing the output of an ESP-IDF 4.4 command. The terminal output includes:

```
ESP-IDF 4.4 CMD: "C:\Espressif\idf_cmd_init.bat" esp-idf-e91d384503485fbb54f6ce3d11e841fe
//Set
gpio: Stub running...
Changing baud rate to 460800
Changed.
Configuring flash size...
If
Flash will be erased from 0x00001000 to 0x00007fff...
1 or
Flash will be erased from 0x00001000 to 0x000039fff...
Flash will be erased from 0x00008000 to 0x00008fff...
gpio: Compressed 25392 bytes to 15887...
2 gpio: Writing at 0x00001000... (100 %)
3 while
4 pr
wrote 25392 bytes (15887 compressed) at 0x00001000 in 0.8 seconds (effective 241.7 kbit/s)...
5 gpio:
Flash of data verified.
6 pr
Compressed 170048 bytes to 89590...
7 pr
Writing at 0x00001000... (16 %)
8 pr
Writing at 0x0001b0c7... (33 %)
9 pr
Writing at 0x00020884... (50 %)
10 }
Writing at 0x00026070... (66 %)
Writing at 0x0002e673... (83 %)
Writing at 0x00036a7e... (100 %)
Wrote 170048 bytes (89590 compressed) at 0x00001000 in 2.3 seconds (effective 582.7 kbit/s)...
As
Flash of data verified.
attr
compressed 3072 bytes to 103...
This
Writing at 0x00008000... (100 %)
Wrote 3072 bytes (103 compressed) at 0x00008000 in 0.1 seconds (effective 331.4 kbit/s)...
pull
Flash of data verified.
1 gpio:
2 gpio:
3 gpio:
4 gpio:
5 gpio:
6 gpio:
7 gpio:
Code 8: Using GPIO config

In the ESP32 SDK, we can define a GPIO as being pulled-up or pulled-down
by using the gpio_set_pull_mode() function. This function takes as input the pin
number we wish to set and the pull mode associated with that pin.
```

The PDF document contains text on the right side, including:

- try point of the program here
- An ESP-IDF application program entry point
- tasks and schedule them using FreeRTOS's task management
- printing your student identifier every second.
- polling a button and print "ESP32" every when the button
- need the vTaskStartScheduler() routine?

The page number 9 is visible at the bottom of the PDF document.

The screenshot shows a web browser window displaying a PDF document titled "CO3054\_ESP32\_Lab\_02.pdf". A terminal window is overlaid on the PDF, showing the output of an ESP-IDF 4.4 command. The terminal output includes:

```
ESP-IDF 4.4 CMD: "C:\Espressif\idf_cmd_init.bat" esp-idf-e91d384503485fbb54f6ce3d11e841fe
//Set
gpio: (0) cpu_start: App cpu up.
(216) cpu_start: Pro cpu start user code
(216) cpu_start: cpu freq: 160000000
(216) cpu_start: Application information:
(220) cpu_start: Project name: main
(225) cpu_start: App version: v4.4.2-dirty
(230) cpu_start: Compile time: Nov 2 2022 17:16:01
(236) cpu_start: ELF file SHA256: 68b5e9bb896274a7...
4 while
5 pr
(242) cpu_start: ESP-IDF: v4.4.2-dirty
6 pr
(248) heap_init: Initializing. RAM available for dynamic allocation:
7 pr
(255) heap_init: At 3FFAE6E0 len 00001920 (6 KiB): DRAM
8 pr
(261) heap_init: At 3FFB2C30 len 0002D300 (100 KiB): DRAM
9 pr
(267) heap_init: At 3FFB0440 len 00003A60 (14 KiB): 0/IRAM
10 }
(274) heap_init: At 3FFC4350 len 00001C00 (11 KiB): 0/1/IRAM
(280) heap_init: At 4008082C len 00014D3C (83 KiB): IRAM
(287) spi_flash: detected chip: generic
(291) spi_flash: flash io: dio
(295) spi_flash: Detected size(4096k) larger than the size in the binary image header(2048k). Using the size in the bi
ary image header.
attr
(309) cpu_start: Starting scheduler on PRO CPU.
This
(0) cpu_start: Starting scheduler on APP CPU.
pull
1952521
ESP32
1 gpio:1952521
2 gpio:ESP32
3 gpio:1952521
4 gpio:ESP32
5 gpio:1952521
6 gpio:ESP32
7 gpio:1952521
Code 8: Using GPIO config

In the ESP32 SDK, we can define a GPIO as being pulled-up or pulled-down
by using the gpio_set_pull_mode() function. This function takes as input the pin
number we wish to set and the pull mode associated with that pin.
```

The PDF document contains text on the right side, including:

- try point of the program here
- An ESP-IDF application program entry point
- tasks and schedule them using FreeRTOS's task management
- printing your student identifier every second.
- polling a button and print "ESP32" every when the button
- need the vTaskStartScheduler() routine?

The page number 9 is visible at the bottom of the PDF document.

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

→ No, because the startup flow of an ESP-IDF application will call this automatically.