

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH  
TRƯỜNG ĐẠI HỌC BÁCH KHOA  
KHOA KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH



**HỆ THỐNG NHÚNG (CO3053)**

---

**Lab 3: FreeRTOS Tasks Scheduling**  
**Lab 4: FreeRTOS Queue Management**

---

Giảng viên hướng dẫn: Vũ Trọng Thiên  
Nhóm sinh viên thực hiện - Nhóm 33: Phạm Duy Quang - 2011899  
Dương Đức Nghĩa - 2011671

Thành phố Hồ Chí Minh, Tháng 11/2023



## Mục lục

<b>1</b>	<b>Github</b>	<b>2</b>
<b>2</b>	<b>Lab 3</b>	<b>2</b>
2.1	Cấu hình và định nghĩa task . . . . .	2
2.1.1	Cấu hình (configuration) . . . . .	2
2.1.2	Định nghĩa task . . . . .	4
2.2	Prioritized Pre-emptive Scheduling with Time Slicing . . . . .	5
2.3	Prioritized Pre-emptive Scheduling without Time Slicing . . . . .	5
2.4	Co-operative Scheduling . . . . .	6
2.5	Extra Exercise . . . . .	7
<b>3</b>	<b>Lab 4</b>	<b>9</b>
3.1	Đặc tả (specification) . . . . .	9
3.2	Hiện thực (implementation) . . . . .	9
3.3	Kết quả . . . . .	13

## 1. Github

Link Github các bài lab của nhóm 33: <https://github.com/ULTIMATE-Mystery/Embedded-System-HCMUT-Semester-231>.

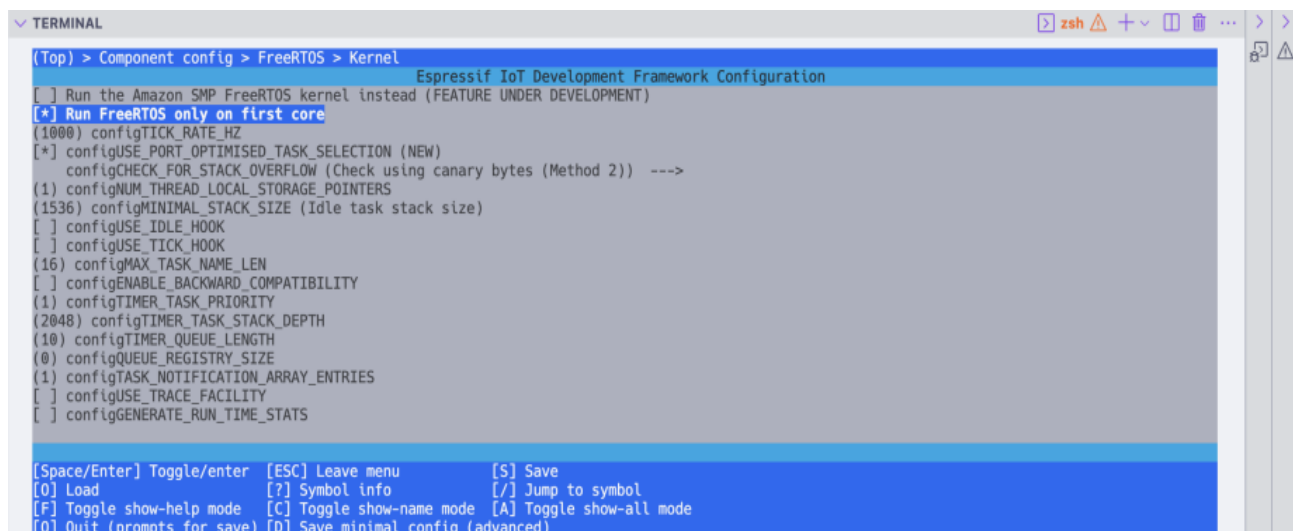
## 2. Lab 3

### 2.1 Cấu hình và định nghĩa task

#### 2.1.1 Cấu hình (configuration)

- Config project chạy trên 1 core của ESP32 như **hình 1**, với cách truy cập sau:

```
idf.py menuconfig
```



Hình 1: Vào component config → FreeRTOS → Kernel

- Config preemptive với time slicing như **hình 2**:



Hình 2: Chỉnh sửa thành preemptive with time slicing ở file FreeRTOSConfig.h

- Config preemptive không có time slicing như **hình 3**:

```

84  /* ----- Scheduler Related ----- */
85
86  #define configUSE_PREEMPTION                1
87  #define configUSE_TICKLESS_IDLE            CONFIG_FREERTOS_USE_TICKLESS_IDLE
88  #if configUSE_TICKLESS_IDLE
89      #define configEXPECTED_IDLE_TIME_BEFORE_SLEEP CONFIG_FREERTOS_IDLE_TIME_BEFORE_SLEEP
90  #endif /* configUSE_TICKLESS_IDLE */
91  #define configCPU_CLOCK_HZ                  ( CONFIG_ESP_DEFAULT_CPU_FREQ_MHZ * 1000000 )
92  #define configTICK_RATE_HZ                  CONFIG_FREERTOS_HZ
93  #define configUSE_TIME_SLICING              1
94  #define configUSE_16_BIT_TICKS              0
95  #define configIDLE_SHOULD_YIELD              0
96  #define configKERNEL_INTERRUPT_PRIORITY      1 /*Todo: This currently isn't used anywhere */
97  Darian Leung, 7 months ago • freertos: Uncrustify FreeRTOSConfig files
98  /* ----- Synchronization Primitives ----- */
99

```

Hình 3: Chỉnh sửa thành preemptive without time slicing scheduling

- Config co-operative như **hình 4**:

```

86  #define configUSE_PREEMPTION                0
87  #define configUSE_TICKLESS_IDLE            CONFIG_FREERTOS_USE_TICKLESS_IDLE
88  ✓ #if configUSE_TICKLESS_IDLE
89      #define configEXPECTED_IDLE_TIME_BEFORE_SLEEP CONFIG_FREERTOS_IDLE_TIME_BEFORE_SLEEP
90  #endif /* configUSE_TICKLESS_IDLE */
91  #define configCPU_CLOCK_HZ                  ( CONFIG_ESP_DEFAULT_CPU_FREQ_MHZ * 1000000 )
92  #define configTICK_RATE_HZ                  CONFIG_FREERTOS_HZ
93  #define configUSE_TIME_SLICING              0
94  #define configUSE_16_BIT_TICKS              0
95  #define configIDLE_SHOULD_YIELD              0
96  #define configKERNEL_INTERRUPT_PRIORITY      1 /*Todo: This currently isn't used anywhere */
97
98  /* ----- Synchronization Primitives ----- */
99

```

Hình 4: Chỉnh sửa thành co-operative scheduling

- Ngoài ra, để dễ dàng chuyển đổi giữa các cách scheduling, nhóm đã định nghĩa trong file header **FreeRTOSConfig.h**:

```

C: > Espressif > frameworks > esp-idf-v5.1.1 > components > freertos > esp_additions > include > freertos > h FreeRTOSConfig.h > ...
83
84 /* ----- Scheduler Related ----- */
85
86 /* User scheduler define */
87 #ifdef SCHEDULING_MODE_PREEMPTIVE_WITHOUT_TIME_SLICING
88     #define configUSE_PREEMPTION 1
89     #define configUSE_TIME_SLICING 0
90 #endif
91
92 #ifdef SCHEDULING_MODE_PREEMPTIVE_WITH_TIME_SLICING
93     #define configUSE_PREEMPTION 1
94     #define configUSE_TIME_SLICING 1
95 #endif
96
97 #ifdef SCHEDULING_MODE_CO_OP
98     #define configUSE_PREEMPTION 0
99     #define configUSE_TIME_SLICING 0
100 #endif
101
102 /* Default marco for scheduler */
103 #ifndef configUSE_PREEMPTION
104     #define configUSE_PREEMPTION 1
105 #endif
106
107 #ifndef configUSE_TIME_SLICING
108     #define configUSE_TIME_SLICING 0
109 #endif

```

Hình 5: Định nghĩa trong file header **FreeRTOSConfig.h**

### 2.1.2 Định nghĩa task

- Hàm **vBusyWaitMs** hoạt động như delay nhưng không đưa task vào block state.

```

1 /* Do nothing for specific milliseconds */
2 void vBusyWaitMs(int iMilliseconds) {
3     int64_t lStart = esp_timer_get_time();
4     int64_t lEnd = lStart + iMilliseconds * 1000;
5     while (esp_timer_get_time() < lEnd) {
6         ; // Do nothing
7     }
8 }

```

- **vTask1**: In từng kí tự một của biến toàn cục **msg** sau mỗi 1000ms. Task này có độ ưu tiên là 1.

```

1 void vTask1(void* pvParameters) {
2     int iLen = strlen(pcMsg);
3     while (true) {
4         for (int i = 0; i < iLen; i++) {
5             printf("%c", pcMsg[i]);
6             vBusyWaitMs(50);
7         }
8     }
9 }

```







## 2.5 Extra Exercise

- Ý tưởng: Đầu tiên ta tạo 2 task dùng để tính toán sự sử dụng của 2 core CPU. Để tính toán thông số đó, thì ta cần biết 2 số là:

- Khoảng thời gian, hay số ticks mà idle task hook nó được gọi.
- Khoảng thời gian mà task tính toán sử dụng CPU nêu trên được gọi lại (trừ lần đầu tiên).

$$\text{CPU Usage (\%)} = 100 - \left( \frac{\text{Idle Task Run Time}}{\text{Total Time}} \right) \times 100$$

- Ta cần định nghĩa lại hàm idle task hook function để tính ticks mỗi lần hàm này được gọi.

```
1 void vApplicationIdleHook() {  
2     if (xPortGetCoreID() == 0) {  
3         ulIdleTicksCore0++;  
4     } else {  
5         ulIdleTicksCore1++;  
6     }  
7 }
```

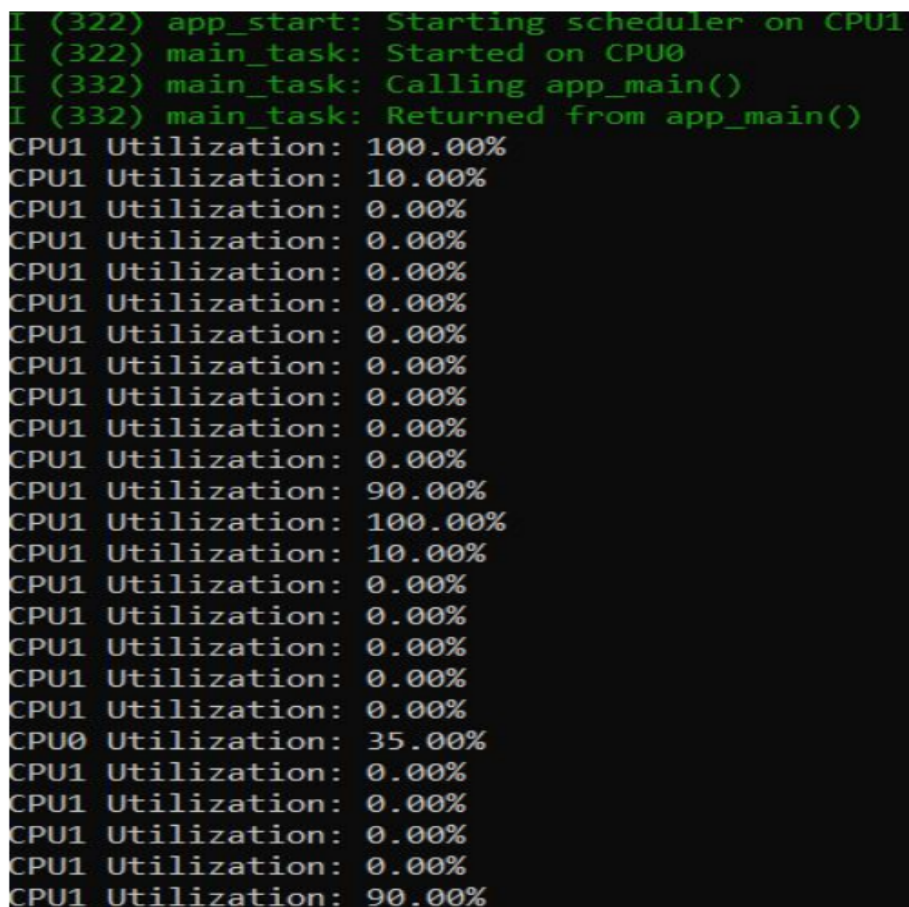
- Hai tasks dùng để tính toán mức độ sử dụng của CPU dựa trên số ticks đã đếm trong Idle task hook function.

```
1 /* Monitor task for both core */  
2 static void vMonitorCore1(void* arg) {  
3     while (1) {  
4         vTaskDelay(MONITOR_PERIOD_TASK1);  
5  
6         TickType_t currentTick = xTaskGetTickCount();  
7         TickType_t elapsedTicks = currentTick - xPrevTickCore0;  
8  
9         float cpu0Utilization =  
10            100.0 - ((float)ulIdleTicksCore0 * 100.0 / elapsedTicks);  
11  
12         if (!ucFirstTimeTask1)  
13             printf("CPU0 Utilization: %.2f%%\n", cpu0Utilization);  
14         else  
15             ucFirstTimeTask1 = 0;  
16  
17         // Reset the idle ticks and update the previous tick count  
18         ulIdleTicksCore0 = 0;  
19         xPrevTickCore0 = currentTick;  
20     }  
21 }  
22  
23 static void vMonitorCore2(void* arg) {  
24     while (1) {  
25         vTaskDelay(MONITOR_PERIOD_TASK2);  
26  
27         TickType_t currentTick = xTaskGetTickCount();
```



```
28     TickType_t elapsedTicks = currentTick - xPrevTickCore1;
29
30     float cpu1Utilization =
31         100.0 - ((float)ulIdleTicksCore1 * 100.0 / elapsedTicks);
32
33     if (!ucFirstTimeTask2)
34         printf("CPU1 Utilization: %.2f%%\n", cpu1Utilization);
35     else
36         ucFirstTimeTask2 = 0;
37
38     // Reset the idle ticks and update the previous tick count
39     ulIdleTicksCore1 = 0;
40     xPrevTickCore1 = currentTick;
41 }
42 }
```

- o Kết quả sau khi chạy:



```
I (322) app_start: Starting scheduler on CPU1
I (322) main_task: Started on CPU0
I (332) main_task: Calling app_main()
I (332) main_task: Returned from app_main()
CPU1 Utilization: 100.00%
CPU1 Utilization: 10.00%
CPU1 Utilization: 0.00%
CPU1 Utilization: 0.00%
CPU1 Utilization: 0.00%
CPU1 Utilization: 0.00%
CPU1 Utilization: 0.00%
CPU1 Utilization: 0.00%
CPU1 Utilization: 0.00%
CPU1 Utilization: 0.00%
CPU1 Utilization: 0.00%
CPU1 Utilization: 90.00%
CPU1 Utilization: 100.00%
CPU1 Utilization: 10.00%
CPU1 Utilization: 0.00%
CPU1 Utilization: 0.00%
CPU1 Utilization: 0.00%
CPU1 Utilization: 0.00%
CPU1 Utilization: 0.00%
CPU0 Utilization: 35.00%
CPU1 Utilization: 0.00%
CPU1 Utilization: 0.00%
CPU1 Utilization: 0.00%
CPU1 Utilization: 0.00%
CPU1 Utilization: 90.00%
```

Hình 9: Mức sử dụng CPU

### 3. Lab 4

#### 3.1 Đặc tả (specification)

Một hệ thống có các sensor dùng để đo những thông số khác nhau, cụ thể:

- eMotoSpeed
- eSpeedSetPoint
- eSPISetMode
- Other

Các thông số này sẽ không được lấy theo chu kỳ cụ thể, nhóm sẽ thực hiện việc trích xuất dữ liệu ngẫu nhiên để phục vụ cho việc hiện thực Lab 4.

Các thông số sau khi đã trích xuất được ở Task Sender thì sẽ liền được gửi cho Task Receiver tương ứng để có thể xử lý thông qua một Queue duy nhất.

#### 3.2 Hiện thực (implementation)

Định nghĩa các tham số sẽ sử dụng trong quá trình hiện thực:

- `QUEUE_SIZE`: Số phần tử mà hàng đợi có thể chứa.
- `QUEUE_WAITS`: Khoảng thời gian tối đa sẽ block để chờ có task trong hàng đợi, nếu hàng đợi trống.
- `QUEUE_SEND_WAITS`: Khoảng thời gian tối đa mà task sẽ block để chờ có chỗ trống trên hàng đợi, nếu hàng đợi đã đầy.
- `QUEUE_SEND_DELAY`: Khoảng thời gian delay giữa việc sinh dữ liệu.
- Tổng kết:

```
1 #define QUEUE_SIZE 10
2 #define QUEUE_WAITS 200
3 #define QUEUE_SEND_WAITS 300
4 #define QUEUE_SEND_DELAY 500
5
6 #define REJECTS_TIMES 3
```

Hiện thực struct **ID** cho đối tượng **QueueData** và **TaskType**:

```
1 typedef enum {
2     eMotoSpeed = 0, // Can Bus Task
3     eSpeedSetPoint, // HMI
4     eSPISetMode, // SPI
5     Other,
6 } ID_t;
```

Hiện thực struct **QueueData** để có thể thống nhất dữ liệu gửi và nhận giữa các Sender và Receiver bên dưới. Tuy nhiên, vì các Receiver sẽ chỉ thực hiện đúng kiểu dữ liệu thuộc về nó để xử lý, để tránh việc mất các dữ liệu trong lúc các Receiver nhận dữ liệu không dành cho nó thì nhóm đã định nghĩa thêm đặc tính *rejectTimes*, nếu *rejectTimes* vượt qua ngưỡng nhất định thì ta tiến hành bỏ dữ liệu đấy.

```
1 struct QueueData {
2     ID_t eRequestID;
3     char cMessage[20];
4     uint32_t lDataValue;
5     int8_t rejectTimes;
6 };
```

Hiện thực struct cho đối tượng kiểu "Task" **TaskType**:

```
1 struct TaskType {
2     ID_t eDataID;
3     char cTaskName[20];
4 };
```

Hàm main sẽ bao gồm:

- o Tạo một Queue để giao tiếp giữa Sender và các Receiver:

```
1 xQueue = xQueueCreate(QUEUE_SIZE, sizeof(struct QueueData *));
```

- o Tạo Sender Task để thực hiện việc sinh dữ liệu:

```
1 // create sender task
2 xTaskCreate(&vSenderTask, "Sender Task", 2048, NULL, 2, NULL);
```

- o Tạo 3 Receiver Task(CAN, HDMI, SPI) để nhận dữ liệu tương ứng với kiểu mà Sender Task gửi:

```
1 // create functional task
2 xTaskCreate(&vReceiverTask, "CAN Task", 2048, (void *) & CAN, 2, NULL);
3 xTaskCreate(&vReceiverTask, "HMI Task", 2048, (void *) & HMI, 2, NULL);
4 xTaskCreate(&vReceiverTask, "SPI Task", 2048, (void *) & SPI, 2, NULL);
```

- o Tổng kết hàm main():

```
1 // create functional task
2 void app_main(void) {
3     xQueue = xQueueCreate(QUEUE_SIZE, sizeof(struct QueueData *));
4
5     // create sender task
6     xTaskCreate(&vSenderTask, "Sender Task", 2048, NULL, 2, NULL);
7
8     // create functional task
9     xTaskCreate(&vReceiverTask, "CAN Task", 2048, (void *) &CAN, 2, NULL);
10    xTaskCreate(&vReceiverTask, "HMI Task", 2048, (void *) &HMI, 2, NULL);
11    xTaskCreate(&vReceiverTask, "SPI Task", 2048, (void *) &SPI, 2, NULL);
12 }
```

Hàm **vSenderTask()** dùng để khởi tạo task Sender được mô tả bên dưới. Hàm này sẽ gửi dữ liệu cho task Receiver sau mỗi khoảng **QUEUE\_SEND\_DELAY**. Để hiện thực được chức năng gửi nhiều kiểu dữ liệu do nhiều sensor khác nhau sinh ra thì nhóm đã sử dụng hàm *random()* để sinh giá trị ngẫu nhiên từ 0 đến 3 tương ứng với 4 kiểu dữ liệu đẩy vào Queue do eMotoSpeed, eSpeedSetPoint, eSPISetMode, Other sinh ra.

```
1 void vSenderTask(void * pvParameter) {
2     time_t t;
3     srand((unsigned)time(&t));
4
5     while (1) {
6         int ranTask = (rand() % 4);
7
8         struct QueueData *xData = malloc(sizeof(struct QueueData));
9
10        // Create task based on random number
11        if (xData != NULL) {
12            switch (ranTask) {
13                case eMotoSpeed:
14                    xData -> eRequestID = eMotoSpeed;
15                    strcpy(xData -> cMessage, "CAN");
16                    xData -> rejectTimes = 0;
17                    xData -> lDataValue = rand() % 100;
18                    break;
19
20                case eSpeedSetPoint:
21                    xData -> eRequestID = eSpeedSetPoint;
22                    strcpy(xData -> cMessage, "HMI");
23                    xData -> rejectTimes = 0;
24                    xData -> lDataValue = rand() % 100;
25                    break;
26
27                case eSPISetMode:
28                    xData -> eRequestID = eSPISetMode;
29                    strcpy(xData -> cMessage, "SPI");
30                    xData -> rejectTimes = 0;
31                    xData -> lDataValue = rand() % 100;
32                    break;
33
34                case Other:
35                    xData -> eRequestID = 99;
36                    strcpy(xData -> cMessage, "DUNNO");
37                    xData -> rejectTimes = 0;
38                    xData -> lDataValue = rand() % 100;
39                    break;
40            }
41
42            // Send this random task to the queue
43            if (xQueueSendToBack(xQueue, (void *) & xData, QUEUE_SEND_WAITS) !=
44                pdTRUE) {
45                printf("Failed to send %s to queue !\n", xData -> cMessage);
46            }
47        } else {
48            printf("Allocated queue failed !");
49        }
50    }
```

```
50     vTaskDelay(pdMS_TO_TICKS(QUEUE_SEND_DELAY));  
51 }  
52 vTaskDelete(NULL);  
53 }
```

Hàm **vReceiverTask()** dùng để phục vụ cho 3 Receiver task. Các Receiver chỉ nhận đúng task về kiểu của mình để có thể xử lý, nếu khác thì Receiver sẽ Warning đồng thời tăng giá trị của *rejectTimes*. Nếu dữ liệu có *rejectTimes* vượt số lượng Receiver Task (hiện tại là 3) thì sẽ bị "DROP". Lý do chủ yếu để đặt ngưỡng giá trị cho *rejectTimes* là số lượng Receiver Task là do tránh việc thực thi các dữ liệu do Other sinh ra.

```
1 void vReceiverTask(void* pvParameter) {  
2     TaskCount++;  
3     for (; ;) {  
4         struct TaskType *pData = (struct TaskType *)pvParameter;  
5         struct QueueData *xReceivedStruct;  
6  
7         if (xQueue != NULL) {  
8             // Check if there are any items in the queue. If yes, handle them;  
9             // otherwise, increase the reject variable by 1. If the reject variable  
10            // reaches its maximum, skip this task.  
11            if (xQueueReceive(xQueue, &xReceivedStruct, (TickType_t)QUEUE_WAITS) ==  
12                pdPASS) {  
13                if (xReceivedStruct -> eRequestID == pData -> eDataID) {  
14                    printf(  
15                        "SUCCEDED -- I'm %s --- Received from %s task, data = "  
16                        "%ld\n",  
17                        pData -> cTaskName, xReceivedStruct -> cMessage,  
18                        xReceivedStruct -> lDataValue);  
19  
20                    free(xReceivedStruct);  
21                } else {  
22                    printf(  
23                        "WARNING -- I'm %s: Received from %s, but it's not my "  
24                        "task\n",  
25                        pData -> cTaskName, xReceivedStruct -> cMessage);  
26                    if (xReceivedStruct -> rejectTimes < TaskCount - 1) {  
27                        xReceivedStruct -> rejectTimes++;  
28                        xQueueSendToFront(xQueue, (void *) & xReceivedStruct,  
29                            (TickType_t)10);  
30                    } else {  
31                        printf(  
32                            "REJECTED -- This task %s is rejected %d times, skipping "  
33                            "the task\n",  
34                            xReceivedStruct -> cMessage,  
35                            (xReceivedStruct -> rejectTimes + 1));  
36                        free(xReceivedStruct);  
37                    }  
38                }  
39            } else {  
40                // Handle empty queue
```

```

41     }
42 }
43
44     vTaskDelay(pdMS_TO_TICKS(10));
45 }
46 vTaskDelete(NULL);
47 }

```

Kiểu dữ liệu cho từng task được đặt tầm toàn cục (global scope) như sau:

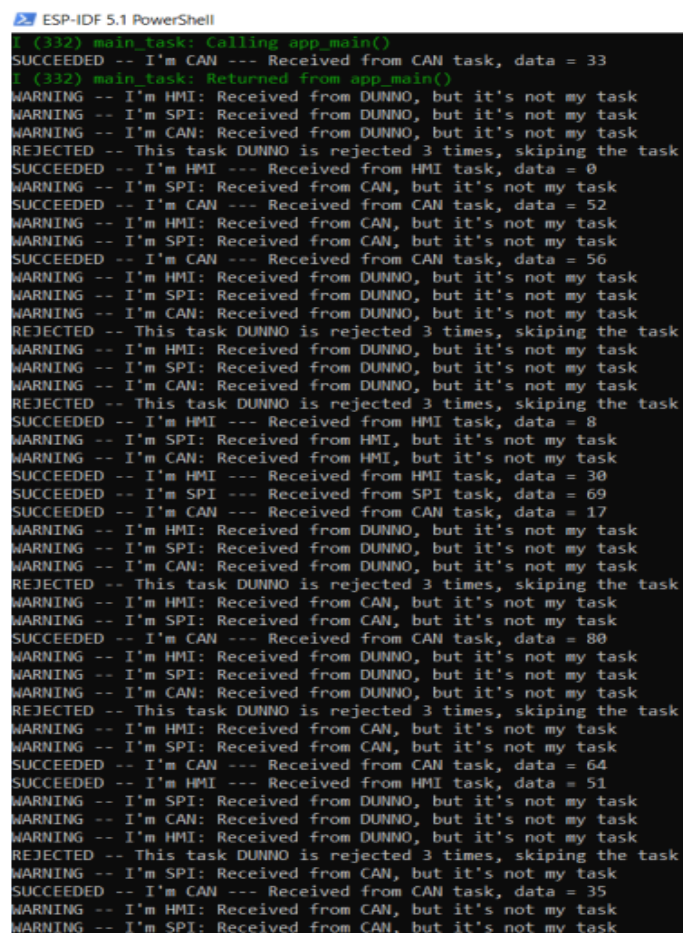
```

1 struct TaskType CAN = {.cTaskName = "CAN", .eDataID = 0};
2 struct TaskType HMI = {.cTaskName = "HMI", .eDataID = 1};
3 struct TaskType SPI = {.cTaskName = "SPI", .eDataID = 2};

```

### 3.3 Kết quả

Kết quả sau khi chạy mô phỏng:



```

ESP-IDF 5.1 PowerShell
I (332) main_task: Calling app_main()
SUCCEEDED -- I'm CAN --- Received from CAN task, data = 33
I (332) main_task: Returned from app_main()
WARNING -- I'm HMI: Received from DUNNO, but it's not my task
WARNING -- I'm SPI: Received from DUNNO, but it's not my task
WARNING -- I'm CAN: Received from DUNNO, but it's not my task
REJECTED -- This task DUNNO is rejected 3 times, skipping the task
SUCCEEDED -- I'm HMI --- Received from HMI task, data = 0
WARNING -- I'm SPI: Received from CAN, but it's not my task
SUCCEEDED -- I'm CAN --- Received from CAN task, data = 52
WARNING -- I'm HMI: Received from CAN, but it's not my task
WARNING -- I'm SPI: Received from CAN, but it's not my task
SUCCEEDED -- I'm CAN --- Received from CAN task, data = 56
WARNING -- I'm HMI: Received from DUNNO, but it's not my task
WARNING -- I'm SPI: Received from DUNNO, but it's not my task
WARNING -- I'm CAN: Received from DUNNO, but it's not my task
REJECTED -- This task DUNNO is rejected 3 times, skipping the task
WARNING -- I'm HMI: Received from DUNNO, but it's not my task
WARNING -- I'm SPI: Received from DUNNO, but it's not my task
WARNING -- I'm CAN: Received from DUNNO, but it's not my task
REJECTED -- This task DUNNO is rejected 3 times, skipping the task
SUCCEEDED -- I'm HMI --- Received from HMI task, data = 8
WARNING -- I'm SPI: Received from HMI, but it's not my task
WARNING -- I'm CAN: Received from HMI, but it's not my task
SUCCEEDED -- I'm HMI --- Received from HMI task, data = 30
SUCCEEDED -- I'm SPI --- Received from SPI task, data = 60
SUCCEEDED -- I'm CAN --- Received from CAN task, data = 17
WARNING -- I'm HMI: Received from DUNNO, but it's not my task
WARNING -- I'm SPI: Received from DUNNO, but it's not my task
WARNING -- I'm CAN: Received from DUNNO, but it's not my task
REJECTED -- This task DUNNO is rejected 3 times, skipping the task
WARNING -- I'm HMI: Received from CAN, but it's not my task
WARNING -- I'm SPI: Received from CAN, but it's not my task
SUCCEEDED -- I'm CAN --- Received from CAN task, data = 80
WARNING -- I'm HMI: Received from DUNNO, but it's not my task
WARNING -- I'm SPI: Received from DUNNO, but it's not my task
WARNING -- I'm CAN: Received from DUNNO, but it's not my task
REJECTED -- This task DUNNO is rejected 3 times, skipping the task
WARNING -- I'm HMI: Received from CAN, but it's not my task
WARNING -- I'm SPI: Received from CAN, but it's not my task
SUCCEEDED -- I'm CAN --- Received from CAN task, data = 64
SUCCEEDED -- I'm HMI --- Received from HMI task, data = 51
WARNING -- I'm SPI: Received from DUNNO, but it's not my task
WARNING -- I'm CAN: Received from DUNNO, but it's not my task
WARNING -- I'm HMI: Received from DUNNO, but it's not my task
REJECTED -- This task DUNNO is rejected 3 times, skipping the task
WARNING -- I'm SPI: Received from CAN, but it's not my task
SUCCEEDED -- I'm CAN --- Received from CAN task, data = 35
WARNING -- I'm HMI: Received from CAN, but it's not my task
WARNING -- I'm SPI: Received from CAN, but it's not my task

```

Hình 10: Kết quả mô phỏng