ĐẠI HỌC QUỐC GIA

ĐẠI HỌC BÁCH KHOA TP HỒ CHÍ MINH

– Embedded System–

LAB 4

Nhóm 2 – L02

Họ và tên	MSSV
Trần Nguyễn Minh Duy	1910095
Đặng Trung Kiên	1911437
Nguyễn Hải Long	1911517
Nguyễn Nhật Trường	1912344

Thành phố Hồ Chí Minh – 2022

MỤC LỤC

ĐÓNG GÓP CỦA CÁC THÀNH VIÊN

Họ và tên	MSSV	Đóng góp
Trần Nguyễn Minh Duy	1910095	100%
Đặng Trung Kiên	1911437	100%
Nguyễn Hải Long	1911517	100%
Nguyễn Nhật Trường	1912344	100%

LINK GITHUB

https://github.com/duytran1511/Embedded_System_LAB

BÁO CÁO

1. Ý tưởng hiện thực

1.1. Mô tả queue

```
// struct data for queue
typedef struct
{
    uint8_t ID;
    uint32_t val;
} Data;
// this variable hold queue handle
xQueueHandle xQueue;
```

- Ta tạo 1 struct dữ liệu là Data với:
    - ID: giúp các task lấy dữ liệu từ queue kiểm tra xem dữ liệu này có dành cho task đó hay không (bằng cách so sánh ID của data với ID của task).
    - val: giá trị bổ sung để thực hiện request.

1.2. Tạo request

```
// variables and constants for request
/*
    0 - get temperature
    1 - Turn on LED
    2 - Turn off LED
    the other - unacceptable request
*/
#define GET_TEMP        0
#define TURN_ON_LED     1
#define TURN_OFF_LED    2

#define N_REQUEST       3
uint8_t request[N_REQUEST] = {0, 0, 0};
// check if there is any request
bool flagRequest = false;
```

```
void vRequest()
{
    uint8_t i = 0;

    while (1)
    {
        // create 3 requests every 5 seconds

        // create request randomly from 0 to 3
        // if request == 3 then error
        bootloader_random_enable();
        request[0] = esp_random() % 4;
        request[1] = esp_random() % 4;
        request[2] = esp_random() % 4;
        bootloader_random_disable();

        // set flag to notify there is request
        flagRequest = true;

        vTaskDelay(5000 / portTICK_RATE_MS);
    }

    vTaskDelete(NULL);
}
```

- Để mô phỏng việc tạo request, ta giả sử có 3 loại request:
    - Request = 0: Lấy giá trị nhiệt độ.
    - Request = 1: Bật đèn.
    - Request = 2: Tắt đèn.
    - Request = các giá trị khác: request không hợp lệ.
- Tiếp theo, ta tạo mảng **request[3]**, để lưu request.
- Cứ mỗi 5 giây, ta sẽ tạo ra 3 request bằng hàm **vRequest**. Ta tạo bằng cách gán biến request bằng 1 số ngẫu nhiên từ 0 đến 3, với request = 3 là không hợp lệ).

- Sau khi tạo request, ta sẽ set cờ **flagRequest** để báo hiệu rằng đã tạo request. Bằng cờ **flagRequest**, Repception task sẽ biết rằng có request mới hay không.

1.3. Functional task id = 0: Task lấy nhiệt độ

```
void vGetTemp(void *parameter)
{
    // keep the status of receiving data
    BaseType_t xStatus;
    // time to block the task until data is available
    const TickType_t xTicksToWait = pdMS_TO_TICKS(100);

    Data data;
    uint8_t id = (uint8_t)parameter;

    uint32_t temperature = 0;

    while (1)
    {
        // Peek data from the queue
        // to check if the next request is for it
        xStatus = xQueuePeek(xQueue, &data, xTicksToWait);

        if (xStatus == pdPASS)
        {
            // check if request
            if (data.ID == id)
            {
                // get random temperature
                bootloader_random_enable();
                temperature = 25 + (esp_random() % 5);
                bootloader_random_disable();

                // pop data from the queue
                xStatus = xQueueReceive(xQueue, &data, xTicksToWait);

                if (xStatus == pdPASS)
                {
```

```
                    printf("Response: %d*C\n", temperature);
                }
                else
                {
                    printf("Response: Cound not recieve data\n");
                }
            }
        }

        vTaskDelay(50 / portTICK_RATE_MS);
    }

    vTaskDelete(NULL);
}
```

- Chức năng chính: nếu có request = 0 (lấy nhiệt độ), task sẽ trả về (in ra) giá trị của nhiệt độ.
- Cách hoạt động: task sẽ kiểm tra xem ID (struct Data) của phần tử đầu của Queue có bằng 0 (id của task) hay không.
    - Nếu có, task sẽ tạo ra 1 giá trị nhiệt độ temperature ngẫu nhiên từ 25 tới 29, và in giá trị nhiệt độ. Sau đó, sẽ lấy (pop) phần tử data đã kiểm tra ra khỏi queue.
    - Nếu không, task sẽ bỏ qua.

1.4. Functional task id = 1: Task điều khiển LED

```
void vSetLed(void *parameter)
{
    // keep the status of receiving data
    BaseType_t xStatus;
    // time to block the task until data is available
    const TickType_t xTicksToWait = pdMS_TO_TICKS(100);

    Data data;
    uint8_t id = (uint8_t)parameter;

    while (1)
```

```
    {
        // Peek data from the queue
        // to check if the next request is for it
        xStatus = xQueuePeek(xQueue, &data, xTicksToWait);

        if (xStatus == pdPASS)
        {
            // check if request
            if (data.ID == id)
            {
                xStatus = xQueueReceive(xQueue, &data, xTicksToWait);

                // pop data from the queue
                if (xStatus == pdPASS)
                {
                    printf("Response: Set LED to %d\n", data.val);
                }
                else
                {
                    printf("Response: Cound not recieve data\n");
                }
            }
        }

        vTaskDelay(50 / portTICK_RATE_MS);
    }

    vTaskDelete(NULL);
}
```

- Chức năng chính: tắt/bật đèn dựa trên giá trị val của struct Data.
- Cách hoạt động: task sẽ kiểm tra xem ID (struct Data) của phần tử đầu của Queue có bằng id = 1 hay không.
    - Nếu có, task dựa vào giá trị val (struct Data) để bật (nếu val = 1) hoặc tắt (nếu val = 0) đèn.
    - Nếu không, task sẽ bỏ qua.

1.4. Reception task

```
void vReceptionTask(void)
{
    /* keep the status of sending data */
    BaseType_t xStatus;
    /* time to block the task until the queue has free space */
    const TickType_t xTicksToWait = pdMS_TO_TICKS(100);
    /* Data is added to queue*/
    Data requestedData;

    uint8_t i = 0;

    while (1)
    {
        // check if there is any request
        if (flagRequest)
        {
            // classify 3 tasks,
            // then add data to queue
            for (i = 0; i < N_REQUEST; i++)
            {
                if (request[i] == GET_TEMP)
                {
                    requestedData.ID = 0;
                    requestedData.val = 0;
                    printf("Request: get temperature\n");
                }
                else if (request[i] == TURN_ON_LED)
                {
                    requestedData.ID = 1;
                    requestedData.val = 1;
                    printf("Request: Turn on LED\n");
                }
                else if (request[i] == TURN_OFF_LED)
                {
                    requestedData.ID = 1;
                    requestedData.val = 0;
                    printf("Request: Turn off LED\n");
                }
                else
                {
                    // If no functional task receives the request,
                    // raise an error
                    requestedData.ID = request[i];
                    requestedData.val = 0;
                    printf("Request: Error - id = %d\n", request[i]);

                    // unset flag, and ignore that request
```

```
                    flagRequest = false;
                    continue;
                }

                // send data to queue
                xStatus = xQueueSendToBack(xQueue, &requestedData,
xTicksToWait);

                // check if sending is ok or not
                if (xStatus == pdPASS)
                {
                    ;
                }
                else
                {
                    printf("Request %d: Could not send data\n", request);
                }
            }

            // unset flag
            flagRequest = false;
        }

        vTaskDelay(50 / portTICK_RATE_MS);
    }

    vTaskDelete(NULL);
}
```

- Chức năng chính: task nhận request và phân loại các request này và gửi chúng đến hàng đợi để các functional task xử lý
- Cách hoạt động:
    - Nếu có request mới được tạo (flagRequest = 1), task sẽ tiến hành phân loại, và thêm data vào Queue với id, và val tương ứng với request đó.
        - Nếu request = 0: ID = 0 (tương ứng với task lấy nhiệt độ), val = x (tùy ý).
        - Nếu request = 1: ID = 1 (tương ứng với task điều khiển LED), val = 1 (bật đèn).
        - Nếu request = 2: ID = 1 (tương ứng với task điều khiển LED), val = 0 (tắt đèn).
        - Nếu request == các giá trị còn lại: báo lỗi, vào bỏ qua.

- Hạ cờ flagRequest (flagRequest = 0), để chờ request tiếp theo.

1.5. Hàm main

```
void app_main()
{
    /* create the queue which size can contains 5 elements of Data */
    xQueue = xQueueCreate(5, sizeof(Data));

    xTaskCreate(vRequest, "vRequest", 2048, NULL, 3, NULL);

    xTaskCreate(vReceptionTask, "vReceptionTask1", 2048, NULL, 2, NULL);

    xTaskCreate(vGetTemp, "vGetTemp1", 2048, (uint8_t *)0u, 1, NULL); //
task get temp, id = 0
    xTaskCreate(vSetLed, "vSetLed", 2048, (uint8_t *)1u, 1, NULL);    //
task set led, id = 1
}
```

- Đầu tiên, ta tạo 1 queue với size = 5.
- Tiếp theo, ta tạo các task như sau:
    - Request task: có priority lớn nhất.
    - Reception task: task có priority lớn thứ nhì.
    - Functional task thứ nhất: task lấy nhiệt độ, priority nhỏ nhất = 1, và bằng priority với các functional task khác.
    - Functional task thứ hai: task điều khiển đèn, priority nhỏ nhất = 1, và bằng priority với các functional task khác.

2. Kết quả chạy chương trình

```
I (0) cpu_start: Starting scheduler on APP CPU.
Request: Turn on LED
Request: Turn off LED
Request: Turn on LED
Response: Set LED to 1
Response: Set LED to 0
Response: Set LED to 1
Request: Error - id = 3
Request: Turn on LED
Request: Error - id = 3
Response: Set LED to 1
Request: Turn on LED
Request: Turn on LED
Request: Turn on LED
Response: Set LED to 1
Response: Set LED to 1
Response: Set LED to 1
Request: Error - id = 3
Request: get temperature
Request: Turn off LED
Response: 25*C
Response: Set LED to 0
```

Lần 1:

- 3 request lần lượt là: bật đèn, tắt đèn, bật đèn.
- 3 phản hồi bao gồm: điều chỉnh tín hiệu của LED lên mức 1, điều chỉnh tín hiệu của LED xuống mức 0, điều chỉnh tín hiệu của LED lên mức 1.

Lần 2:

- 3 request lần lượt là: request lỗi id = 3, bật đèn, request lỗi id = 3.
- 1 phản hồi bao gồm: điều chỉnh tín hiệu của LED lên mức 1.

Lần 3:

- 3 request lần lượt là: bật đèn, bật đèn, bật đèn.
- 3 phản hồi bao gồm: điều chỉnh tín hiệu của LED lên mức 1, điều chỉnh tín hiệu của LED lên mức 1, điều chỉnh tín hiệu của LED lên mức 1.

Lần 4:

- 3 request lần lượt là: request lỗi id = 3, lấy nhiệt độ, tắt đèn.
- 2 phản hồi bao gồm: trả về (in ra) giá trị nhiệt độ (ngẫu nhiên), điều chỉnh tín hiệu của LED xuống mức 0.