ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA

VI XỬ LÝ - VI ĐIỀU KHIỂN (TN) (CO3049)

Báo cáo Lab 4 - A cooperative scheduler

Tp. Hồ Chí Minh, Tháng 12/2022

Contents

1 Link

- Link Github : https://github.com/HappyFalcon22/Lab-MCU-Repo/tree/Lab-4/Lab%204
- Link mô phỏng Proteus : https://drive.google.com/file/d/1K-JqXwUF8HqSf3xI57KwljhY3uXxZ3Li/view

2 Quy ước

- Task *display7SEG* hiển thị số ngẫu nhiên mỗi $1s$
- Đèn LED *red* sẽ chớp tắt mỗi $0.5s$
- Đèn LED *green* sẽ chớp tắt mỗi $1s$
- Đèn LED *yellow* sẽ chớp tắt mỗi $1.5s$
- Đèn LED *orange* sẽ chớp tắt mỗi $2.0s$
- Đèn LED *pink* sẽ thực hiện *oneshot*, chỉ sáng.
- Đơn vị thời gian hiển thị (kí hiệu là t) : $1t = 10ms$

3 Hiện thực scheduler

3.1 Hàm SCH_Init

```
void SCH_Init(void)
{
  for (int i = 0; i < SCH_MAX_TASKS; i++)
    SCH_Delete_Tasks(i);
  current_idx_task = 0; // Just in case (could delete this line)
}
```

Code 1: Code hàm SCH_Init

3.2 Hàm SCH_AddTask

```
void SCH_Add_Task (void (*pFunction)(), uint32_t delay, uint32_t period)
{
  if (current_idx_task < SCH_MAX_TASKS)
  {
    SCH_Tasks_G[current_idx_task].pTask = pFunction;
    SCH_Tasks_G[current_idx_task].delay = delay/CLOCK_TICK;
    SCH_Tasks_G[current_idx_task].period = period/CLOCK_TICK;
    SCH_Tasks_G[current_idx_task].runMe = 0;
    SCH_Tasks_G[current_idx_task].taskID = current_idx_task; // Temporarily set
    current_idx_task++; // Update the index to the next slot
  }
}
```

Code 2: Code hàm SCH_Add_Tasks

3.3 Hàm SCH_Update

```
void SCH_Update(void)
{
  time_unit++;
  for(int i = 0; i < current_idx_task; i++) // Travese all active tasks
  {
    if (SCH_Tasks_G[i].pTask == 0)
      continue;
    if (SCH_Tasks_G[i].delay > 0)
      SCH_Tasks_G[i].delay--;
    else
```

```
    {
      SCH_Tasks_G[i].delay = SCH_Tasks_G[i].period;
      SCH_Tasks_G[i].runMe += 1;
    }
  }
}
```

Code 3: Code hàm SCH_Update

3.4 Hàm SCH_Dispatch_Tasks

```
void SCH_Dispatch_Tasks(void) // Runs in the super-loop
{

  for (int i = 0; i < current_idx_task; i++)
  {
    if (SCH_Tasks_G[i].pTask == 0)
      continue;
    if (SCH_Tasks_G[i].runMe > 0)
    {
      (*SCH_Tasks_G[i].pTask)();
      SCH_Tasks_G[i].runMe--; // Must put in line first
      if (SCH_Tasks_G[i].period == 0)
        SCH_Delete_Tasks(i);
    }
  }
}
```

Code 4: Code hàm SCH_Dispatch_Tasks

3.5 SCH_Delete_Tasks

```
void SCH_Delete_Tasks(int index)
{
  if (SCH_Tasks_G[index].pTask == 0){}
  else
  {
    SCH_Tasks_G[index].pTask = 0x0000;
    SCH_Tasks_G[index].delay = 0;
    SCH_Tasks_G[index].period = 0;
    SCH_Tasks_G[index].runMe = 0;
  }
  return;
}
```

Code 5: Code hàm SCH_Delete_Tasks

4 Hiện thực UART đơn giản

4.1 Chuyển khai báo sang file global.c

```
#include "global.h"
#include "main.h"

UART_HandleTypeDef huart2;
int time_unit = 0;
```

Code 6: Khai báo ở file global.c

4.2 Hàm HAL_UART_RxCpltCallback

```
uint8_t temp = 0;
void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart)
{
  if (huart->Instance == USART2)
  {
    HAL_UART_Transmit(&huart2, &temp, 1, 50);
    HAL_UART_Receive_IT(&huart2, &temp, 1);
  }
```

```
}
```

Code 7: Hàm HAL_UART_RxCpltCallback

4.3 In thời gian thực hiện các tasks

Vì thời gian có thể là 1, 2, 3, 4 hoặc 5 chữ số, em thực hiện hàm *calc_digit* để tính chính xác số chữ số của thời gian để truyền UART.

```
int calc_digit(int num)
{
  int res = 1;
  while(1)
  {
    if (num % 10 != num)
    {
      res++;
      num = num / 10;
    }
    else
      return res;
  }
}
```

Code 8: Thực hiện hàm calc_digit

Sau đó em thực hiện đoạn code truyền UART về thời gian và tên task bắt đầu thực hiện. Giả sử ở task *toggle_yellow* :

```
void toggle_yellow()
{
  char str1[] = "Time : ";
  char str2[calc_digit(time_unit)];
  sprintf(str2, "%d", time_unit);
  char str3[] = ", Task toggle_yellow scheduled\r\n";
  HAL_UART_Transmit(&huart2, str1, sizeof(str1), 1000);
  HAL_UART_Transmit(&huart2, str2, sizeof(str2), 1000);
  HAL_UART_Transmit(&huart2, str3, sizeof(str3), 1000);
  HAL_GPIO_TogglePin(GPIOA, LED3_Pin);
}
```

Code 9: Truyền UART ở task toggle_yellow

5 Init trên hàm main

```
int main()
{
  ...
  MX_GPIO_Init();
  MX_TIM2_Init();
  MX_USART2_UART_Init();
  HAL_UART_Receive_IT(&huart2, &temp, 1);
  HAL_TIM_Base_Start_IT(&htim2);
  SCH_Init();
  SCH_Add_Task(display7SEG, 100, 1000);
  SCH_Add_Task(toggle_red, 100, 500);
  SCH_Add_Task(toggle_green, 100, 1000);
  SCH_Add_Task(toggle_yellow, 100, 1500);
  SCH_Add_Task(toggle_orange, 100, 2000);
  SCH_Add_Task(toggle_pink, 100, 0);
  while(1)
  {
    SCH_Dispatch_Tasks();
  }
  ...
}
```

Code 10: Khởi tạo ở hàm main

6 Thiết kế trên Proteus

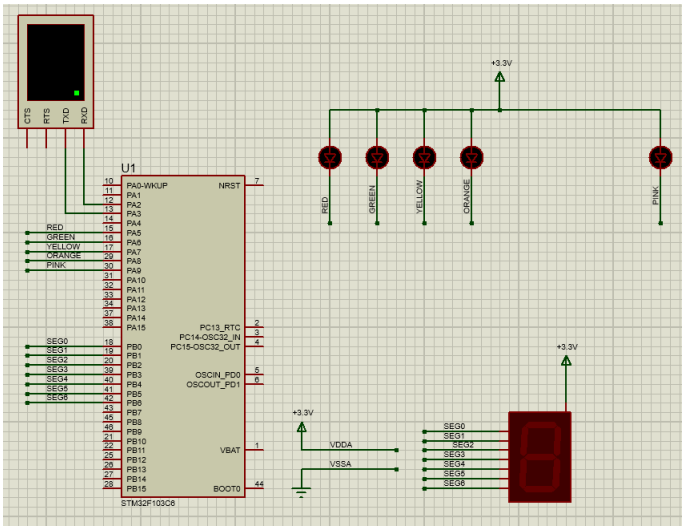

Figure 1: *Proteus Schematic*

Video mô phỏng kết quả :

https://drive.google.com/file/d/1K-JqXwUF8HqSf3xI57KwljhY3uXxZ3Li/view?usp=share_link