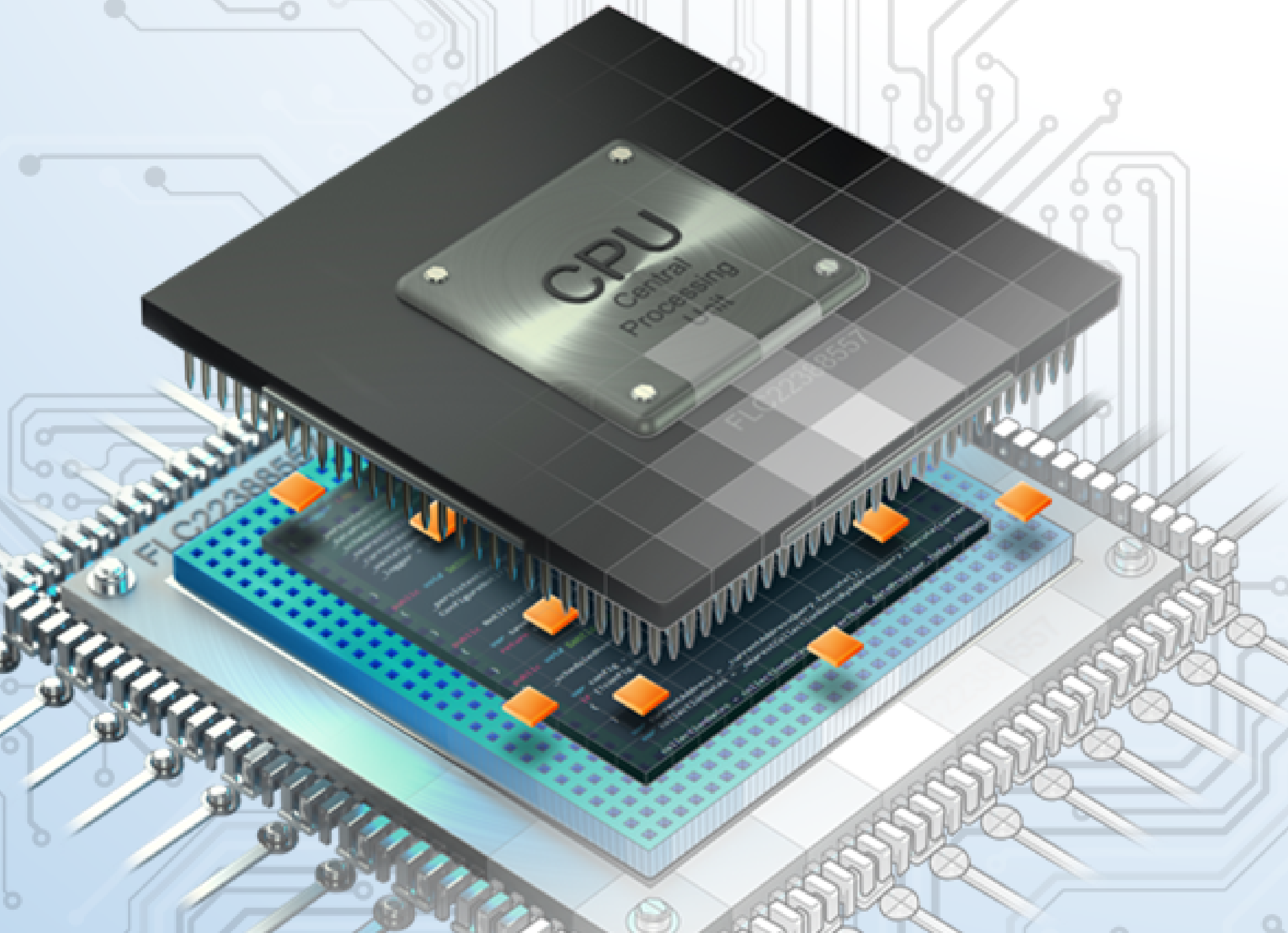
HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY
COMPUTER ENGINEERING

Microcontroller

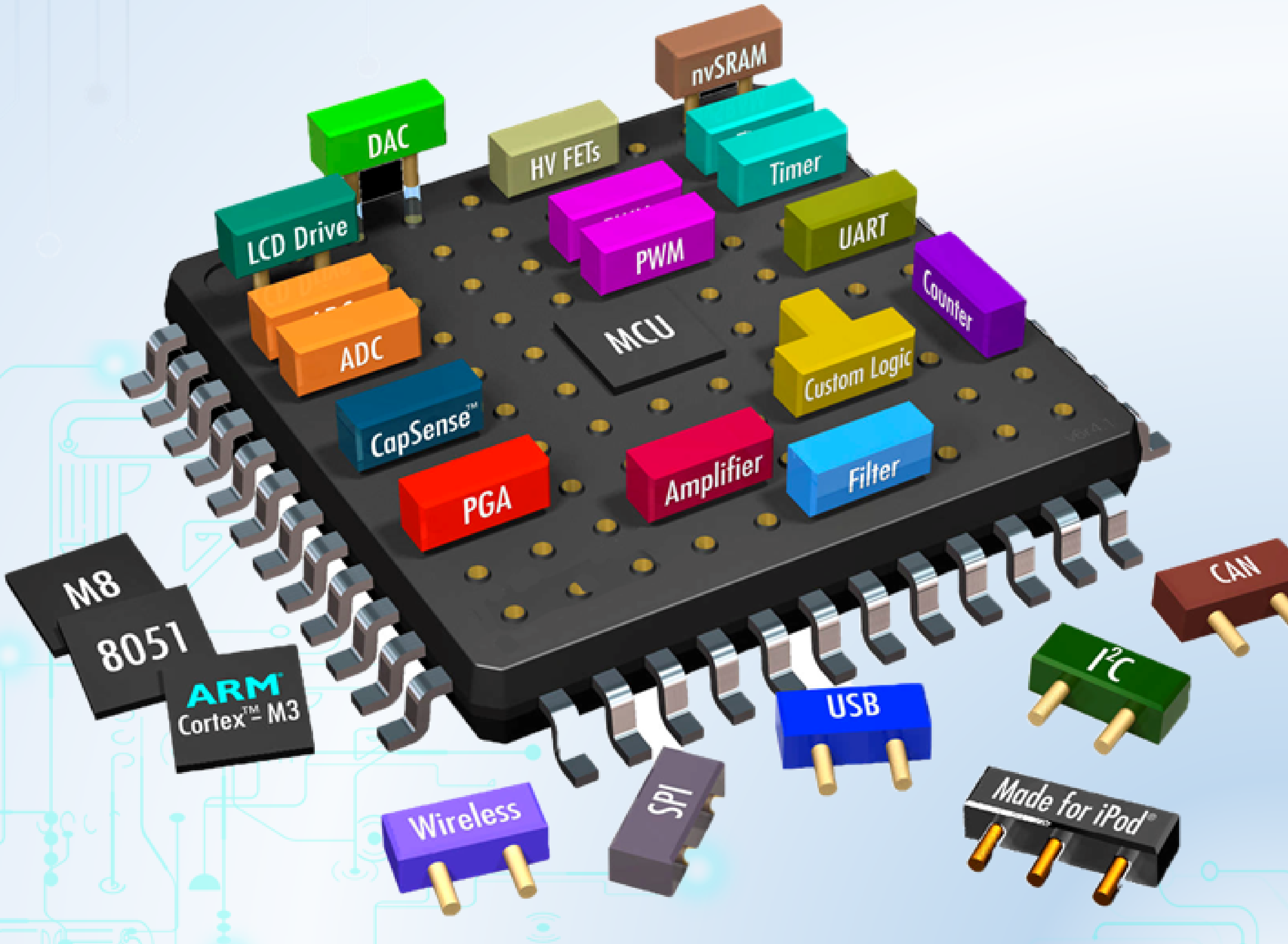


Dr. Le Trong Nhan

Mục lục

CHƯƠNG 1

Digital Clock Project



1 GITHUB LINK

Click here

2 Code reference

2.0.1 The scheduler

Containing 4 functions to solve the problem

```c
/*
 * scheduler.h
 *
 *  Created on: Nov 11, 2022
 *      Author: Dinh Luu
 */

#ifndef INC_SCHEDULER_H_
#define INC_SCHEDULER_H_

#include <stdint.h>

#define EMPTY 0
#define ACTIVE 1
#define DELETED 2

typedef struct{
  void (*pTask)(void);
  uint32_t  Delay;
  uint32_t  Period;
  uint8_t   RunMe;

  uint32_t  TaskID;
  uint32_t  State;
}sTasks;

#define SCH_MAX_TASKS 40

void SCH_Init(void);

void SCH_Add_Task ( void (*pFunction)() ,
          uint32_t DELAY,
          uint32_t PERIOD);

void SCH_Update(void);

void SCH_Dispatch_Tasks(void);

void SCH_Delete_Task(uint32_t taskID);

#endif /* INC_SCHEDULER_H_ */
```
Program 1.1: scheduler.h

```c
#include "scheduler.h"
#include "main.h"
sTasks SCH_tasks_G[SCH_MAX_TASKS];
uint8_t current_index_task = 0;

int printTime = 0;

void SCH_Init(void){
  current_index_task = 0;
  for(int i = 0; i < SCH_MAX_TASKS; i++)
  {
    SCH_tasks_G[i].State = EMPTY;
  }
}

void SCH_Add_Task ( void (*pFunction)() , uint32_t DELAY,
   uint32_t PERIOD){
  for(int i = 0; i < SCH_MAX_TASKS; i++)
  {
    current_index_task = i;

    if(SCH_tasks_G[i].State != ACTIVE && current_index_task
    < SCH_MAX_TASKS){
      SCH_tasks_G[current_index_task].pTask = pFunction;
      SCH_tasks_G[current_index_task].Delay = DELAY;
      SCH_tasks_G[current_index_task].Period =  PERIOD;
      SCH_tasks_G[current_index_task].RunMe = 0;

      SCH_tasks_G[current_index_task].State = ACTIVE;

      // not important now
      SCH_tasks_G[current_index_task].TaskID =
   current_index_task;


      current_index_task++;

      break;
    }
  }
}


```

```c
void SCH_Update(void){
  for(int i = 0; SCH_tasks_G[i].State != EMPTY && i <
   SCH_MAX_TASKS; i++)
  {
    if(SCH_tasks_G[i].State == ACTIVE){
      if (SCH_tasks_G[i].Delay > 0)
      {
//          display7SEG(SCH_tasks_G[i].Delay / 10);
        SCH_tasks_G[i].Delay--;
      }
      else
      {
        SCH_tasks_G[i].Delay = SCH_tasks_G[i].Period;
        SCH_tasks_G[i].RunMe++;
      }
      if (SCH_tasks_G[i].Delay > 0 && printTime == 50)
      {
        display7SEG(SCH_tasks_G[i].Delay / 10);
        printTime = 0;
      }
    }
  }
      printTime++;
}

void SCH_Dispatch_Tasks(void){
  for(int i = 0; SCH_tasks_G[i].State != EMPTY && i <
   SCH_MAX_TASKS; i++){
    if(SCH_tasks_G[i].State == ACTIVE && SCH_tasks_G[i].
   RunMe > 0){
      SCH_tasks_G[i].RunMe--;
      (*SCH_tasks_G[i].pTask)();
    }
  }
}

void SCH_Delete_Task(uint32_t taskID) {
  if(SCH_tasks_G[taskID].State = ACTIVE && taskID <
   SCH_MAX_TASKS){
    SCH_tasks_G[taskID].Delay = 0;
    SCH_tasks_G[taskID].Period =  0;
    SCH_tasks_G[taskID].RunMe = 0;

    SCH_tasks_G[taskID].State = DELETED;
  }
}
```

Program 1.2: scheduler.c

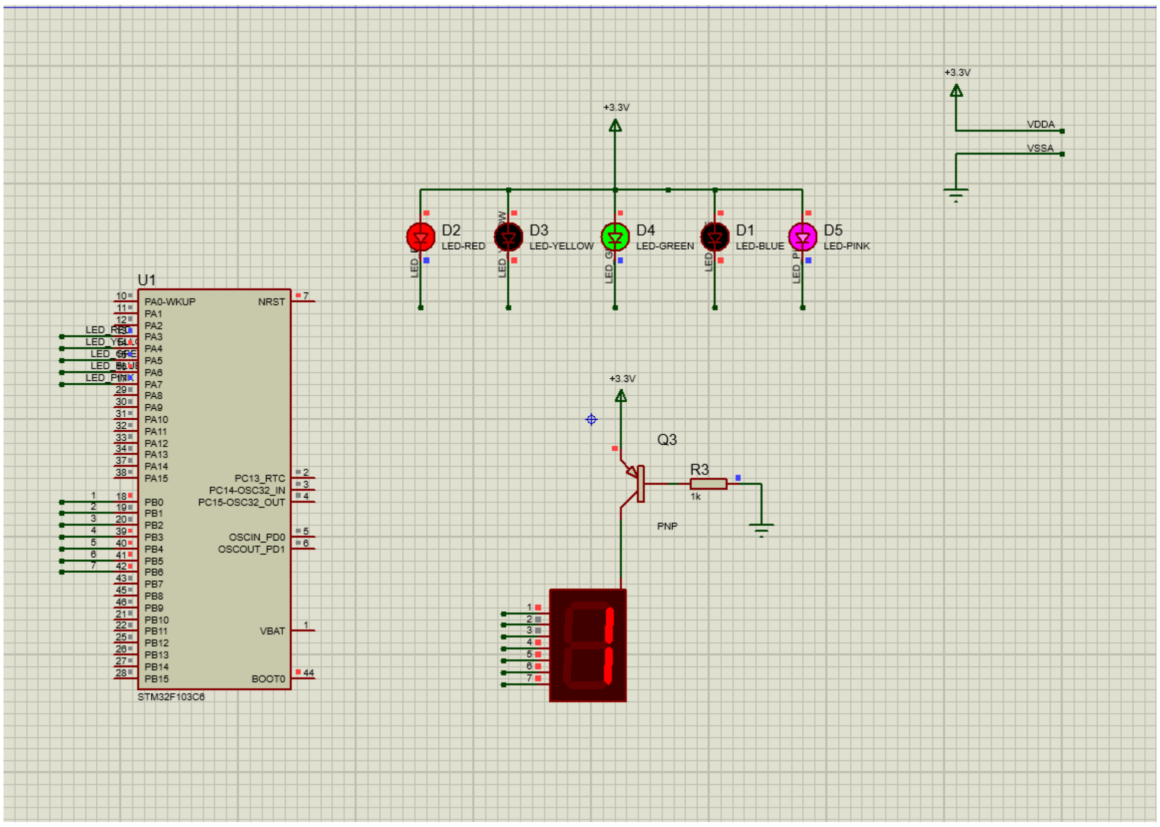
2.0.2 The main

```c

void ledredtest(){
  HAL_GPIO_TogglePin(LED_RED_GPIO_Port, LED_RED_Pin);
}
void ledyellowtest(){
  HAL_GPIO_TogglePin(LED_YELLOW_GPIO_Port, LED_YELLOW_Pin);
}
void ledgreentest(){
  HAL_GPIO_TogglePin(LED_GREEN_GPIO_Port, LED_GREEN_Pin);
}
void ledbluetest(){
  HAL_GPIO_TogglePin(LED_BLUE_GPIO_Port, LED_BLUE_Pin);
}
void ledpinktest(){
  HAL_GPIO_TogglePin(LED_PINK_GPIO_Port, LED_PINK_Pin);
}
int main(void)
  SCH_Add_Task(ledredtest, 0, 50);
  SCH_Add_Task(ledyellowtest, 0, 100);
  SCH_Add_Task(ledgreentest, 0, 150);
  SCH_Add_Task(ledbluetest, 0, 200);
  SCH_Add_Task(ledpinktest, 0, 250);
{
  while (1)
  {
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
    SCH_Dispatch_Tasks();
//    SCH_Delete_Task(3);

  }
}
```

Program 1.3: Add tasks to queue and run tasks in while loop

```c

void HAL_TIM_PeriodElapsedCallback ( TIM_HandleTypeDef*
   htim )
{
  SCH_Update();
}
```

Program 1.4: Timer interrupt



Hình 1.1: The schematic