TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG
ĐẠI HỌC BÁCH KHOA HÀ NỘI

Báo cáo thực hành hệ nhúng
Bài thực hành số 1

Giảng viên hướng dẫn: Mai Xuân Ngọc
Nhóm thực hiện: Phan Đức Duy – 20225831
Phạm Tùng Dương – 20225825
Sinh viên báo cáo: Phạm Tùng Dương – 20225825
Ngày nộp: Tuần 34 – 26/4/2025

Nội dung

I. **Bài 3.1** 3

II. **Bài 3.6** 3

III. **Bài 3.7** 6

IV. **Bài 3.8** 6

V. **Bài 3.9** 7

I. Bài 3.1

1. **Hai IC này (DS1307 và AT24C32) có thể cùng hoạt động được không? Tại sao?**
   Hai IC này hoàn toàn có thể đồng thời hoạt động.
   Lý do:
   - Cả hai IC đều dùng chuẩn I2C
   - Mà I2C lại hỗ trợ nhiều thiết bị hoạt động cùng lúc và mỗi thiết bị đều có địa chỉ riêng ở trên bus
2. **Các chân tín hiệu cần để ghép nối với module Tiny RTC là gì?**
   - Các chân nguồn:
     - VCC: cấp nguồn.
     - GND: nối đất
     - BAT: cấp nguồn 3V thay nguồn pin nuôi chip RTC (không cần nối khi có pin)
   - Các chân tín hiệu:
     - SCL (SERIAL CLOCK): chân clock của bus I2C, tín hiệu đồng bộ truyền dữ liệu.
     - SDA (SERIAL DATA): chân dữ liệu hai chiều của bus I2C.
3. **Địa chỉ của DS1307 và AT24C32 tương ứng là bao nhiêu?**
   - Địa chỉ của DS1307 là 0x68
   - Địa chỉ của AT24C32 mặc định là 0x50 nhưng có thể thay đổi bằng các chân nối A0, A1, A2 (từ 0x50 đến 0x57)

II. Bài 3.6

```
67  uint8_t bin2bcd(uint8_t val) {
68      return ((val / 10) << 4) | (val % 10);
69  }
70
71  // Hàm chuyển đổi BCD sang binary
72  uint8_t bcd2bin(uint8_t val) {
73      return ((val >> 4) * 10) + (val & 0x0F);
74  }
```

Mục đích: Chuyển từ giá trị nhị phân (decimal thông thường) sang **BCD (Binary-Coded Decimal)** và ngược lại vì BCD là định dạng mà DS1307 sử dụng.

- **Hàm bin2bcd(uint8_t val):**
  - **Cơ chế:** BCD lưu mỗi chữ số thập phân (0–9) thành 4 bit (1 nibble).
  - **Ví dụ:** Decimal 45 -> 0x45 trong BCD:

Biểu thức	Giá trị	Ý nghĩa
val / 10	4	Lấy chữ số hàng chục
4 << 4	0x40	Đưa chữ số hàng chục vào 4 bit cao
val % 10	5	Lấy chữ số hàng đơn vị

- **Hàm *bcd2bin(uint8_t val):***
  - **Cơ chế:** Giải mã 2 nibble:
    - **4 bit cao** là hàng chục ⟶ nhân 10
    - **4 bit thấp** là hàng đơn vị ⟶ cộng vào
  - **Ví dụ:** val = 0x45

Biểu thức	Giá trị	Ý nghĩa
val >> 4	4	Lấy hàng chục từ 4 bit cao
4*10	40	Hàng chục
val & 0x0F	5	Lấy chữ số hàng đơn vị từ 4 bit thấp

```
75  struct Time {
76      uint8_t sec;
77      uint8_t min;
78      uint8_t hour;
79      uint8_t weekday;
80      uint8_t day;
81      uint8_t month;
82      uint8_t year;
83  };
84
```

Mục đích: Khai báo cấu trúc đại diện cho thời gian

```
85  // Hàm ghi thời gian vào DS1307
86  void SetTime(struct Time *time) {
87      struct Time bcdTime;
88
89      bcdTime.sec = bin2bcd(time->sec) & 0x7F; // Clear CH bit (bit7)
90      bcdTime.min = bin2bcd(time->min);
91      bcdTime.hour = bin2bcd(time->hour) & 0x3F; // 24-hour mode (bit6=0)
92      bcdTime.weekday = bin2bcd(time->weekday);
93      bcdTime.day = bin2bcd(time->day);
94      bcdTime.month = bin2bcd(time->month);
95      bcdTime.year = bin2bcd(time->year);
96
97      HAL_I2C_Mem_Write(&hi2c3, 0xD0, 0, 1, (uint8_t *)&bcdTime, 7, 1000);
98  }
```

- **Mục đích**: Chuyển đổi giá trị giờ từ kiểu decimal sang BCD, gửi 7 byte dữ liệu BCD qua I2C đến địa chỉ 0xD0 (ghi dữ liệu).
- **& 0x7F**: Clear bit 7 trong thanh ghi giây (bit CH = Clock Halt, nếu bit = 1 thì DS1307 sẽ dừng đếm).
- **& 0x3F**: Clear bit 6 trong thanh ghi giờ (để chọn chế độ 24 giờ).

```
void GetTime(struct Time *time) {
    struct Time bcdTime;
    HAL_I2C_Mem_Read(&hi2c3, 0xD1, 0, 1, (uint8_t *)&bcdTime, 7, 1000);

    time->sec = bcd2bin(bcdTime.sec & 0x7F); // Mask out CH bit
    time->min = bcd2bin(bcdTime.min);
    time->hour = bcd2bin(bcdTime.hour & 0x3F); // Mask out 12/24 mode bit
    time->weekday = bcd2bin(bcdTime.weekday);
    time->day = bcd2bin(bcdTime.day);
    time->month = bcd2bin(bcdTime.month);
    time->year = bcd2bin(bcdTime.year);
}
```

Mục đích: Đọc 7 byte từ DS1307 qua I2C tại địa chỉ 0xD1 (địa chỉ đọc). Dữ liệu được đọc là dạng BCD, nên cần dùng bcd2bin() để chuyển về decimal. Cũng sử dụng & 0x7F và & 0x3F để bỏ các bit điều khiển không liên quan đến giá trị thời gian.

```
// Thiết lập thời gian một lần duy nhất
    struct Time currentTime = {
        .sec = 30,
        .min = 04,
        .hour = 10,
        .weekday = 5,
        .day = 17,
        .month = 4,
        .year = 25
    };
    SetTime(&currentTime);  // Gọi một lần để set thời gian ban đầu

    char buff[30];
    struct Time readTime;
```

Khởi tạo cấu trúc Time để thiết lập giờ hiện tại (tự chọn thời gian)

```
while (1)
{
            GetTime(&readTime);
            sprintf(buff, "%02d:%02d:%02d - %02d - %02d/%02d/%02d\r\n",
                          readTime.hour,
                          readTime.min,
                          readTime.sec,
                          readTime.weekday,
                          readTime.day,
                          readTime.month,
                          readTime.year);
                    HAL_UART_Transmit(&huart1, (uint8_t*)buff, strlen(buff), 1000);
                    HAL_Delay(500);
```

- Gọi GetTime() để:
  - Đọc 7 byte từ DS1307 qua I2C (địa chỉ 0xD1)

- Chuyển giá trị từ BCD ⟶ decimal
- Gán vào readTime

- Định dạng chuỗi kiểu: HH:MM:SS - Thứ - Ngày/Tháng/Năm rồi gửi ra Hercules qua UART1.

III. Bài 3.7

```
//              In tiêu đề
                SH1106_GotoXY(0, 0);
                SH1106_Puts("Current Time:", &Font_7x10, 1);

                // In thời gian
                sprintf(buff, "%02d:%02d:%02d", readTime.hour, readTime.min, readTime.sec);
                SH1106_GotoXY(20, 15);
                SH1106_Puts(buff, &Font_11x18, 1);

                // In ngày tháng
                sprintf(buff, "%02d/%02d/20%02d", readTime.day, readTime.month, readTime.year);
                SH1106_GotoXY(15, 40);
                SH1106_Puts(buff, &Font_7x10, 1);

                SH1106_UpdateScreen();
                HAL_Delay(1000);
```

- **sprintf()**: định dạng thành chuỗi "HH:MM:SS" vào buff.
  - readTime.hour, min, sec là giờ thực từ DS1307.
  - %02d đảm bảo có 2 chữ số (ví dụ 04, 09...).
- **SH1106_GotoXY(20, 15):** đặt vị trí hiển thị giờ nằm **dưới dòng tiêu đề** một chút, căn giữa ngang.
- **Font_11x18**: font lớn, dễ nhìn (chiều cao 18 pixel), dùng để nổi bật thời gian hiện tại.
  Tương tự với hiển thị ngày tháng năm.
- **SH1106_UpdateScreen()** ⟶ Cập nhật bộ đệm màn hình OLED.
  (Các hàm GotoXY() và Puts() chỉ ghi dữ liệu vào RAM đệm, chưa hiển thị ngay).

IV. Bài 3.8

```
        uint8_t CardID[5];
            HAL_Delay(100);

            if (TM_MFRC522_Check(CardID) == MI_OK) {
                SH1106_Fill(0);
                sprintf(buf, "ID: %02X%02X%02X%02X%02X",
                        CardID[0], CardID[1], CardID[2], CardID[3], CardID[4]);
                SH1106_GotoXY(10, 30); // goto 10, 10
                SH1106_Puts(buf, &Font_7x10, 1);
                SH1106_UpdateScreen(); // update screen
            }
            else {
                SH1106_Fill(0);
                sprintf(buf, "%s", "NONE?");
                SH1106_GotoXY(20, 15); // goto 10, 10
                SH1106_Puts(buf, &Font_11x18, 1);
                SH1106_UpdateScreen(); // update screen
            }
```

- CardID[5]: mảng chứa 5 byte ID của thẻ RFID (thường là UID 5 byte).
- HAL_Delay(100): chờ 100ms để tránh đọc quá nhanh hoặc nhiễu.
-

Nếu có thẻ gần đầu đọc, clear màn hình trước khi hiển thị ID thẻ theo dạng HEX. (ID đã được đọc nhờ các hàm trong thư viện)

VD: "ID: 23AB7C194F" (mỗi byte in 2 chữ số hex, viết hoa).

Nếu không có thể gần đầu đọc, hiện chữ NONE.

V. Bài 3.9

```
/* USER CODE BEGIN PD */
#define MAX_LOGS 100
#define MAX_KEYS 10
/* USER CODE END PD */
```

- Khai báo số lần lưu logs tối đa và số mã thẻ có thể lưu tối đa.

```
uint8_t authKeys[MAX_KEYS][5];
int authCount = 1;
```

- Khai báo mã hợp lệ và số lượng mã hợp lệ hiện có.

```
int isAuthorized(uint8_t *id) {
    for (int i = 0; i < authCount; i++) {
        if (memcmp(id, authKeys[i], 5) == 0)
            return 1;
    }
    return 0;
}
```

- Hàm trên có tác dụng kiểm tra thẻ RFID có hợp lệ hay không
- **Cách hoạt động**:
    - Duyệt qua danh sách các thẻ được phép (authKeys – gồm authCount thẻ).
    - So sánh từng thẻ trong danh sách với id bằng memcmp.
    - Nếu khớp, trả về 1 (hợp lệ).
    - Nếu không có thẻ nào khớp, trả về 0 (không hợp lệ).

```c
typedef struct {
    uint8_t time[3];      // [giây, phút, giờ]
    uint8_t cardID[5];    // mã thẻ RFID
} DoorLog;
DoorLog logs[MAX_LOGS];
uint8_t logCount = 0;
void save_log(uint8_t *cardID, uint8_t *time) {
    if (logCount < MAX_LOGS) {
        memcpy(logs[logCount].cardID, cardID, 5);
        memcpy(logs[logCount].time, time, 3); // time[0]=sec, time[1]=min, time[2]=hour
        logCount++;
    }
}
void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart) {
    if (huart->Instance == USART1) {
        if (uart_rx == 'L') {
            show_logs();  // Gửi toàn bộ log khi nhận ký tự 'L'
        }

        // Nhận lại tiếp ký tự tiếp theo
        HAL_UART_Receive_IT(&huart1, &uart_rx, 1);
    }
}
```

- Cấu trúc DoorLog: Lưu thời gian mở cửa và mã thẻ RFID tương ứng.
- Hàm save_log(): Lưu một log mới nếu chưa đầy. -> Sao chép mã thẻ và thời gian vào log tiếp theo ->Tăng logCount.
- Hàm HAL_UART_RxCpltCallback(): Hàm ngắt khi nhận dữ liệu UART.Nếu nhận ký tự 'L' từ máy tính (PC) -> gọi show_logs() để gửi log về -> Sau đó, tiếp tục chờ ký tự tiếp theo.

```c
void show_logs() {
    char buffer[100];
    for (int i = 0; i < logCount; i++) {
        sprintf(buffer, "Log %02d: %02X%02X%02X%02X%02X at %02d:%02d:%02d\r\n", i + 1,
                logs[i].cardID[0], logs[i].cardID[1], logs[i].cardID[2],
                logs[i].cardID[3], logs[i].cardID[4],
                logs[i].time[2], logs[i].time[1], logs[i].time[0]);
        HAL_UART_Transmit(&huart1, (uint8_t*)buffer, strlen(buffer), HAL_MAX_DELAY);
    }
}
/* USER CODE END 0 */
```

- Duyệt toàn bộ logs.
- Format từng log thành chuỗi dạng:
  Log 01: 23ABCD0198 at 12:34:56
- Gửi qua UART về PC.

```
//
    uint8_t defaultCard[5] = {0xA3, 0x1E, 0x31, 0xB9, 0x35};
    memcpy(authKeys[0], defaultCard, 5);
    authCount = 1;
    struct Time currentTime = {
        .sec = 50,
        .min = 59,
        .hour = 23,
        .weekday = 5,
        .day = 17,
        .month = 4,
        .year = 25
    };
    SetTime(&currentTime);  // Gọi một lần để set thời gian ban đầu
```

Đoạn mã trên khai báo mã thẻ RFID hợp lệ và khai báo thời gian hiện tại.

Bên trong hàm while(1):

```
        if (TM_MFRC522_Check(cardID) == MI_OK) {
            HAL_GPIO_WritePin(GPIOG, GPIO_PIN_13, GPIO_PIN_SET);

            if (isAuthorized(cardID)) {
                HAL_GPIO_WritePin(GPIOG, GPIO_PIN_14, GPIO_PIN_SET);
                SH1106_GotoXY(10, 10);
                SH1106_Puts("Welcome", &Font_11x18, 1);
                SH1106_UpdateScreen();
                struct Time readTime;
                GetTime(&readTime);
                sprintf(buff, "%02d:%02d:%02d - %02d - %02d/%02d/%02d\r\n",
                              readTime.hour,
                              readTime.min,
                              readTime.sec,
                              readTime.weekday,
                              readTime.day,
                              readTime.month,
                              readTime.year);
                    HAL_Delay(500);
```

- **Nếu có thẻ RFID:** Bật đèn báo (LED3) khi có thẻ.
- **Nếu thẻ hợp lệ:** Đồng thời bật đèn LED4 và hiển thị dòng chữ Welcome lên SH1106.
- **GetTime và sprintf**: Lấy lại thời gian mở cửa.

```
                                readTime.year);
                        HAL_Delay(500);
                                  HAL_UART_Transmit(&huart1, (uint8_t*)buff, strlen(buff), 1000);
                                  HAL_Delay(500);
                                  char time[3];
                                 sprintf(time, "%02d:%02d:%02d - %02d - %02d/%02d/%02d\r\n",
                                               readTime.hour,
                                               readTime.min,
                                               readTime.sec,
                                               readTime.weekday,
                                               readTime.day,
                                               readTime.month,
                                               readTime.year);
                                  save_log(cardID, time);

            } else {
                SH1106_GotoXY(10, 10);
                SH1106_Puts("Rejected", &Font_11x18, 1);
                SH1106_UpdateScreen();
            }
            HAL_Delay(1000);
        } else {
            HAL_GPIO_WritePin(GPIOG, GPIO_PIN_13, GPIO_PIN_RESET);
            HAL_GPIO_WritePin(GPIOG, GPIO_PIN_14, GPIO_PIN_RESET);
        }

        HAL_Delay(200);
```

- **<u>Lỗi ở đây</u>**: char time[3] không đủ chỗ chứa chuỗi thời gian đầy đủ nên khi thực hiện thời gian đọc được sẽ bị sai
- **Nếu thẻ không có quyền:** Hiển thị dòng chữ “Rejected” lên SH1106.
- **Nếu không có thẻ nào được quét:** Tắt cả 2 đèn báo.