# Báo Cáo - Giải Thích Từng Dòng Code DHT11 (STM32F10x)

## Phần 1: Khai báo thư viện và biến toàn cục

```
#include "stm32f10x.h"
#include <stdint.h>
```

- `#include "stm32f10x.h"`: Gọi file header chính thức của STM32F1, trong đó chứa toàn bộ định nghĩa thanh ghi, cấu trúc và hàm cho vi điều khiển STM32F10x. - `#include <stdint.h>`: Thư viện chuẩn của C, cung cấp kiểu dữ liệu chuẩn có độ rộng cố định (uint8_t, uint32_t, ...).

```
int hum = 0;
int temp = 0;
```

- Khai báo biến toàn cục lưu giá trị độ ẩm ( `hum` ) và nhiệt độ ( `temp` ). Giá trị kiểu số nguyên.

```
uint8_t I_RH, D_RH, I_Temp, D_Temp;
uint8_t Rh_byte1, Rh_byte2, Temp_byte1, Temp_byte2, CheckSum, presence;
```

- Các biến trung gian lưu dữ liệu trả về từ cảm biến: - `I_RH` , `D_RH` : Phần nguyên và thập phân của độ ẩm. - `I_Temp` , `D_Temp` : Phần nguyên và thập phân của nhiệt độ. - `Rh_byte1` , `Rh_byte2` : Hai byte dữ liệu độ ẩm. - `Temp_byte1` , `Temp_byte2` : Hai byte dữ liệu nhiệt độ. - `CheckSum` : Byte kiểm tra dữ liệu. - `presence` : Biến đánh dấu cảm biến có phản hồi hay không.

---

## Phần 2: Prototype (khai báo hàm)

```
void delay_us(uint32_t us);
void delay_ms(uint32_t ms);

void DHT11_Start(void);
uint8_t DHT11_Check_Response(void);
uint8_t DHT11_Read(void);
```

- Khai báo nguyên mẫu các hàm sẽ sử dụng: - `delay_us()` : Tạo trễ micro giây. - `delay_ms()` : Tạo trễ mili giây. - `DHT11_Start()` : Gửi tín hiệu bắt đầu đến cảm biến. - `DHT11_Check_Response()` : Kiểm tra phản hồi từ DHT11. - `DHT11_Read()` : Đọc 1 byte dữ liệu từ cảm biến.

---

## Phần 3: Tạo Delay bằng SysTick

```c
void delay_us(uint32_t us)
{
    SysTick->LOAD = 72 * us;      // 72 MHz
    SysTick->VAL  = 0;
    SysTick->CTRL = 5;            // Enable SysTick
    while ((SysTick->CTRL & (1 << 16)) == 0);
    SysTick->CTRL = 0;
}
```

- `SysTick->LOAD = 72 * us;`: Với xung clock 72 MHz, mỗi chu kỳ mất ~1/72 µs. Nhân với `us` để tính số chu kỳ cần. - `SysTick->VAL = 0;`: Reset bộ đếm. - `SysTick->CTRL = 5;`: Bật SysTick, chọn nguồn clock CPU. - `while ((SysTick->CTRL & (1 << 16)) == 0);`: Chờ đến khi cờ đếm tràn ( `COUNTFLAG` ) bật lên. - `SysTick->CTRL = 0;`: Tắt SysTick sau khi trễ.

```c
void delay_ms(uint32_t ms)
{
    while(ms--) delay_us(1000);
}
```

- Vòng lặp gọi `delay_us(1000)` để tạo trễ mili giây.

---

## Phần 4: Cấu hình GPIO cho DHT11

```c
#define DHT11_PIN 0   // PA0
```

- Chọn chân PA0 để giao tiếp với cảm biến DHT11.

```c
static void GPIO_Config(void)
{
    RCC->APB2ENR |= RCC_APB2ENR_IOPAEN;   // Enable clock GPIOA

    // PA0 input floating
    GPIOA->CRL &= ~(0xF << (DHT11_PIN * 4));
    GPIOA->CRL |=  (0x4 << (DHT11_PIN * 4));
}
```

- Mở clock cho port A. - Cấu hình PA0 là input floating (chế độ ban đầu để đọc tín hiệu từ cảm biến).

```c
void DHT11_Set_Output(void)
{
    GPIOA->CRL &= ~(0xF << (DHT11_PIN * 4));
    GPIOA->CRL |=  (0x3 << (DHT11_PIN * 4));   // Output 50MHz push-pull
}
```

- Cấu hình PA0 thành output push-pull (dùng khi MCU kéo đường dữ liệu xuống 0 hoặc lên 1).

```
void DHT11_Set_Input(void)
{
    GPIOA->CRL &= ~(0xF << (DHT11_PIN * 4));
    GPIOA->CRL |=  (0x4 << (DHT11_PIN * 4));   // Input floating
}
```

- Trả PA0 về input để đọc dữ liệu trả về từ cảm biến.

---

## Phần 5: Các hàm DHT11

```
void DHT11_Start(void)
{
    DHT11_Set_Output();
    GPIOA->BSRR = (1 << (DHT11_PIN + 16)); // Kéo xuống 0
    delay_ms(20);                          // Ít nhất 18ms
    GPIOA->BSRR = (1 << DHT11_PIN);        // Kéo lên 1
    delay_us(30);
    DHT11_Set_Input();
}
```

- `DHT11_Set_Output()`: Đặt chân ở chế độ output. - `GPIOA->BSRR = (1 << (DHT11_PIN + 16))`: Kéo xuống 0 (ghi 1 vào bit reset của PA0). - `delay_ms(20)`: Giữ mức thấp ít nhất 18 ms (theo chuẩn DHT11). - `GPIOA->BSRR = (1 << DHT11_PIN)`: Kéo chân lên 1. - `delay_us(30)`: Giữ mức cao ~20–40 µs. - `DHT11_Set_Input()`: Đổi chân thành input để chờ phản hồi.

```
uint8_t DHT11_Check_Response(void)
{
    uint8_t response = 0;
    delay_us(40);
    if (!(GPIOA->IDR & (1 << DHT11_PIN)))
    {
        delay_us(80);
        if (GPIOA->IDR & (1 << DHT11_PIN)) response = 1;
        delay_us(40);
    }
    return response;
}
```

- Chờ 40 µs. - Nếu chân kéo xuống thấp → cảm biến đã phản hồi. - Sau đó cảm biến kéo lên 80 µs → xác nhận. - Trả về `1` nếu có phản hồi, ngược lại trả `0`.

```
uint8_t DHT11_Read(void)
{
```

```
    uint8_t i, j = 0;
    for (i = 0; i < 8; i++)
    {
        while (!(GPIOA->IDR & (1 << DHT11_PIN))); // Chờ lên 1
        delay_us(40);
        if (GPIOA->IDR & (1 << DHT11_PIN))
            j |= (1 << (7 - i));
        while (GPIOA->IDR & (1 << DHT11_PIN));    // Chờ xuống 0
    }
    return j;
}
```

- Vòng lặp 8 lần để đọc 1 byte. - Chờ tín hiệu lên mức cao. - Delay 40 µs, nếu tại thời điểm đó chân vẫn cao → bit = 1, nếu thấp → bit = 0. - Dữ liệu được lưu dần vào biến `j`.

---

## Phần 6: Hàm `main`

```
int main(void)
{
    GPIO_Config();
    delay_ms(1000); // Chờ cảm biến ổn định

    while (1)
    {
        DHT11_Start();
        presence = DHT11_Check_Response();

        Rh_byte1   = DHT11_Read();
        Rh_byte2   = DHT11_Read();
        Temp_byte1 = DHT11_Read();
        Temp_byte2 = DHT11_Read();
        CheckSum   = DHT11_Read();

        if ((Rh_byte1 + Rh_byte2 + Temp_byte1 + Temp_byte2) == CheckSum)
        {
            I_RH   = Rh_byte1;
            D_RH   = Rh_byte2;
            I_Temp = Temp_byte1;
            D_Temp = Temp_byte2;

            hum  = I_RH;   // giá trị % độ ẩm
            temp = I_Temp; // giá trị °C nhiệt độ
        }

        delay_ms(2000); // Đọc mỗi 2 giây
    }
}
```

- `GPIO_Config()`: Cấu hình GPIO ban đầu. - `delay_ms(1000)`: Chờ cảm biến DHT11 khởi động (≥1s theo datasheet). - Vòng `while(1)`: Vòng lặp vô hạn, liên tục đọc dữ liệu. - `DHT11_Start()`: Gửi tín hiệu bắt đầu. - `presence = DHT11_Check_Response()`: Kiểm tra cảm biến phản hồi. - Đọc 5 byte: 2 byte độ ẩm, 2 byte nhiệt độ, 1 byte checksum. - Nếu tổng 4 byte bằng checksum → dữ liệu hợp lệ. - Gán giá trị đọc vào biến toàn cục `hum` và `temp`. - Trễ 2 giây rồi lặp lại.

---

## Kết luận

- Code trên cấu hình STM32F103 đọc dữ liệu cảm biến DHT11 qua 1 dây (PA0).
- Sử dụng SysTick để tạo delay chính xác µs và ms.
- Tuân thủ chuẩn giao tiếp của DHT11 (Start signal, Response, Data frame 40 bit).
- Giá trị thu được: `hum` (% độ ẩm) và `temp` (°C).