**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

****

**BÁO CÁO NGHIÊN CỨU KHOA HỌC ĐỀ TÀI**

***“*Xây dựng module phần mềm đọc dữ liệu các loại cảm biến chuyên dụng*”***

**Mã số ………….**

**HỢP ĐỒNG NGHIÊN CỨU KHOA HỌC CÔNG NGHỆ**

**Số: ……………**

|  |  |
| --- | --- |
| **Người thực hiện :** |  |
| **Chủ nhiệm đề tài:** |  |

Mục Lục

[CHƯƠNG 1: GIỚI THIỆU CHUNG 3](#_Toc3415583)

[CHƯƠNG 2: XÂY DỰNG MODULE PHẦN MỀM ĐỌC CÁC LOẠI CẢM BIẾN CHUYÊN DỤNG 5](#_Toc3415584)

[2.1. Giới thiệu phần mềm STM32CubeMX: 5](#_Toc3415585)

[2.2. Giới thiệu phần mềm Keli C MDK-ARM: 6](#_Toc3415586)

[2.3. Thiết kế module giao tiếp STM32 và module Rhf76-052 7](#_Toc3415587)

[2.4. Thiết kế module phần mềm đọc nhiệt độ độ ẩm từ cảm biến SHT75 9](#_Toc3415588)

[2.4. Thiết kế module phần mềm đọc dữ liệu từ cảm biến độ ẩm đất và cảm biến đo pH 12](#_Toc3415589)

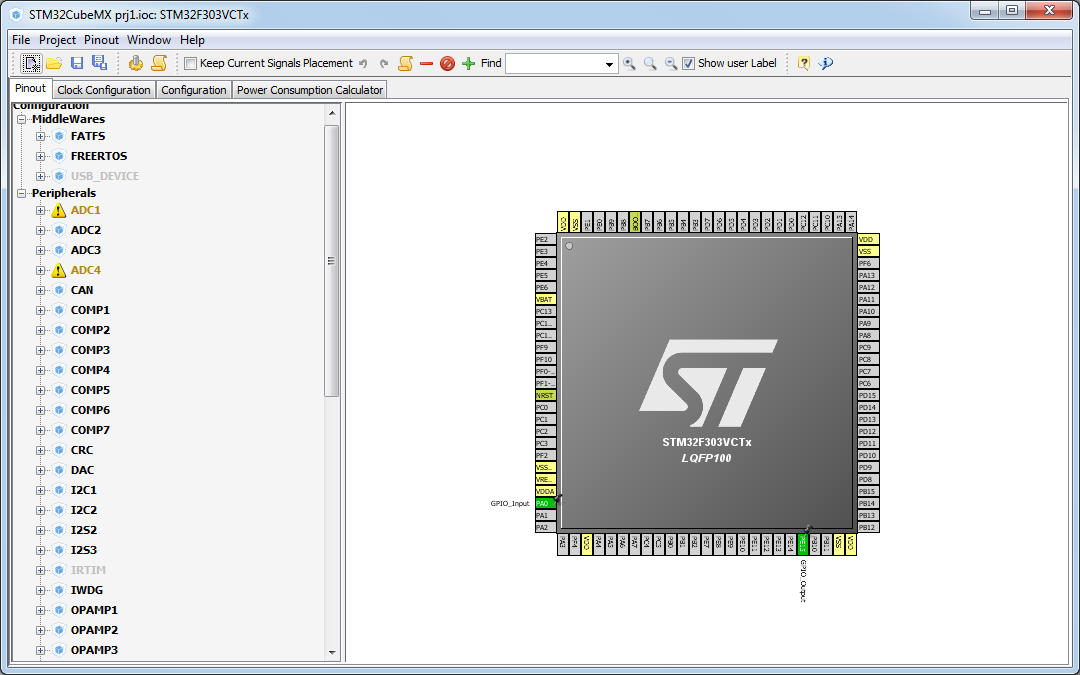
[2.5. Thiết kế module phần mềm ghi dữ liệu vào EEPROM 13](#_Toc3415590)

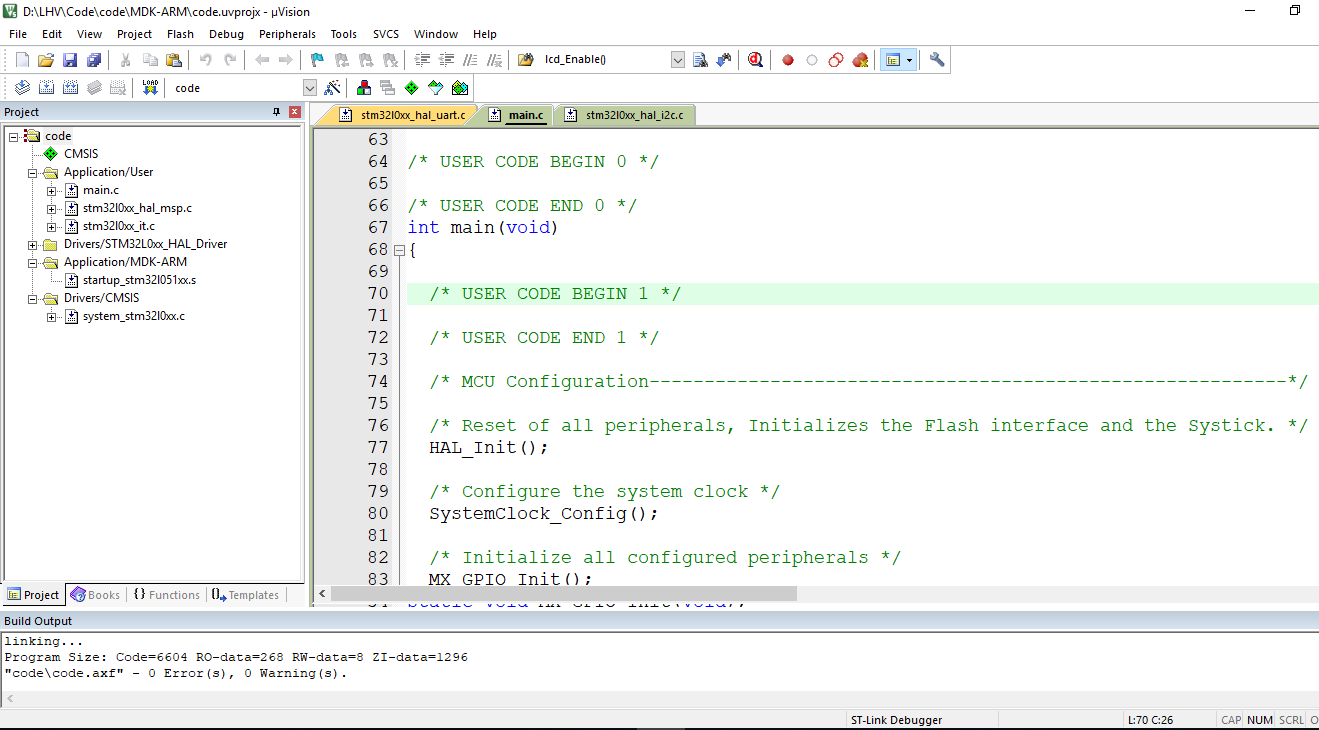
[CHƯƠNG 3: KẾT LUẬN 16](#_Toc3415591)

**CHƯƠNG 1: GIỚI THIỆU CHUNG**

**Giới thiệu chung**

Sau khi đã thiết kế phần cứng và layout bo mạch hoàn chỉnh. Cần phải lập trình cho chíp điều khiển chính của bo mạch là STM32L152 để nó có thể giao tiếp với module Lora và các cảm biến. Có khá nhiều trình biên dịch cho STM32 như IAR Embedded Workbench, Keil C… Thiết kế này sử dụng phần mềm Keil C v5 và phần mềm **STM32CubeMX**.

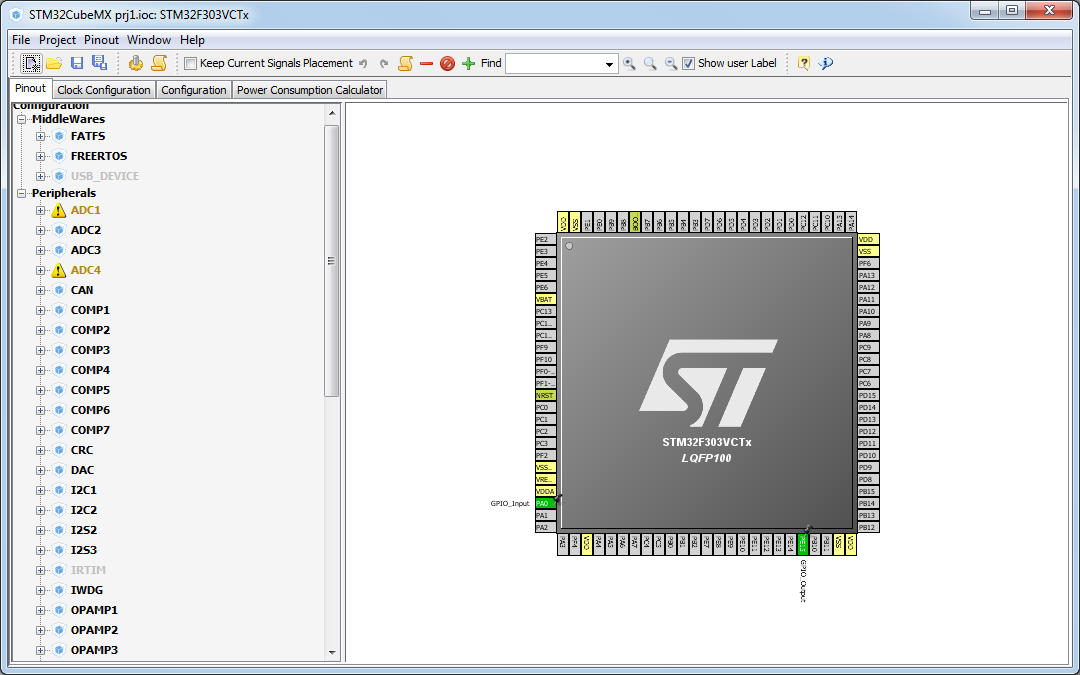


****

**CHƯƠNG 2: XÂY DỰNG MODULE PHẦN MỀM ĐỌC CÁC LOẠI CẢM BIẾN CHUYÊN DỤNG**

**2.1. Giới thiệu phần mềm STM32CubeMX:**

Trong thiết kế sử dụng chíp STM32L152 để làm chíp điều khiển trung tâm nên cần phải thiết kế các module để có thể giao tiếp nó với module Lora và đọc dữ liệu từ các cảm biến. Nếu thiết kế một thư viện dựa trên các thanh ghi của chíp này sẽ rất phức tạp. Thay vì điều đó các nhà sản xuất đã phát triển các thư viện/driver chuẩn cho các con chip của họ. Thư viện này cung cấp các hàm API của nhà sản xuất, giúp cho việc viết chương trình trở nên dễ dàng hơn nhiều. Hãng ST trước đây cũng cung cấp thư viện ngoại vi chuẩn (Standard Peripheral Libraries) để cho chúng ta sử dụng. Tuy nhiên sử dụng nó cũng còn quá nhiều điều phức tạp, và những ai mới bước vào thế giới ARM sẽ rất dễ nản lòng. Ví dụ đơn giản nhất là việc khởi tạo phần cứng ( Cài đặt RCC cho hệ thống, cài đặt ngoại vi…) cũng khá là phức tạp. Sau đó ST ra mắt công cụ STM32 MicroExplorer để giúp cấu hình ngoại vi, cũng như tạo project dựa trên các cấu hình đó, việc còn lại của chúng ta chỉ là viết code của mình. Trải qua nhiều version, STM32 MicroExplorer đã phát triển thành STM32CubeMX, một chương trình hoàn thiện hơn rất nhiều và giúp cho việc lập trình trên STM32 dễ dàng hơn bao giờ hết.



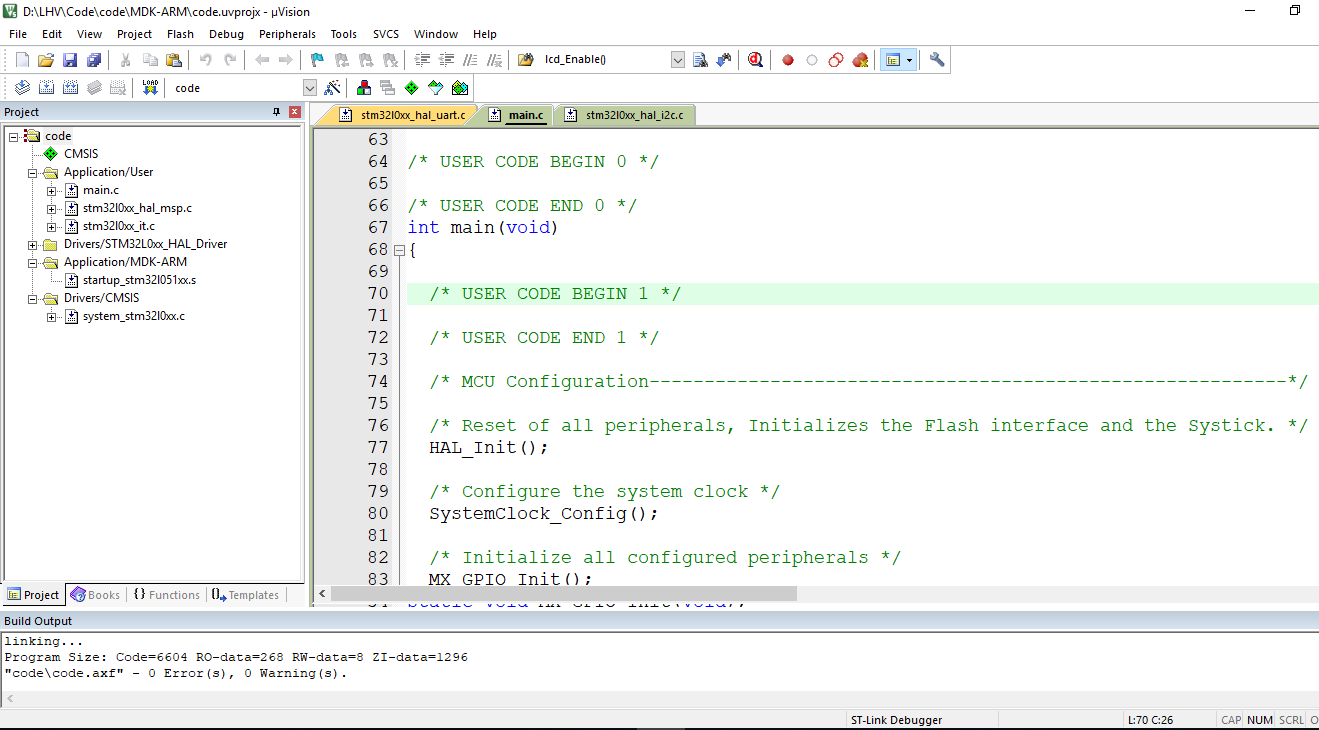
Hình 51- Phần mềm CubeMX

***Ưu điểm của STM32CubeMX:***

* Giúp cấu hình ngoại vi dễ dàng: Chọn các pin trên chip và chọn các tính năng mong muốn gắn với nó. Cấu hình Middlewares (FATS, FREERTOS), các ngoại vi như CRC, IWDG, TIMERS…, cấu hình Clock và tính toán mức độ tiêu hao năng lượng.
* Việc cấu hình đó cho phép người lập trình cấu hình trên giao diện đồ họa vô cùng trực quan và dễ dàng thao tác.
* STM32CubeMX tự động download các driver mới nhất của ST dành cho các dòng chip của mình. ST đã không còn phát triển Standard Peripheral Libraries nữa, thay vào đó họ phát triển cấu trúc firmware mới bao gồm lớp cách ly phần cứng (HAL) bao gồm các driver cho ngoại vi, lớp Middleware bao gồm hỗ trợ TCP/IP, USB, Graphics, FAT file system, Touch library, và hệ điều hành mã nguồn mở RTOS. Cấu trúc firmware mới này có mức độ trừu tượng cao hơn, tập trung vào các tính năng phần cứng chung thay vì tập trung thuần túy vào phần cứng. Mức độ trừu tượng cao hơn giúp phát triển các API thân thiện và có thể dễ dàng chuyển từ phần cứng này sang phần cứng khác. Lưu ý là Standard Peripheral Libraries không còn được khuyên dùng để phát triển các ứng dụng mới nữa.
* Sau khi đã cấu hình xong lập trình viên có thể lựa chọn để tạo mã nguồn cho trình biên dịch mà họ muốn sử dụng như Keli C hay IAR.

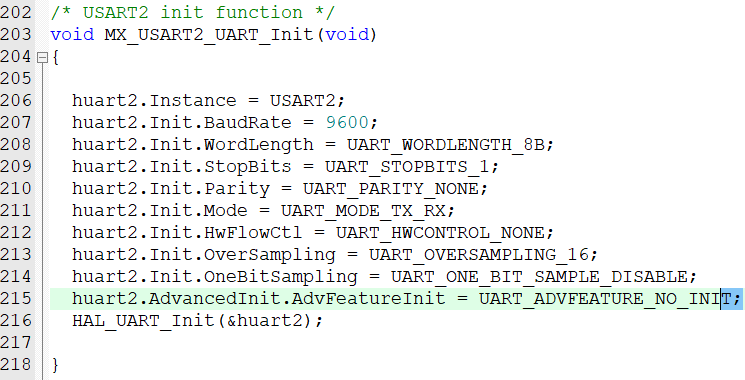
**2.2. Giới thiệu phần mềm Keli C MDK-ARM:**

**Đây là phần mềm hỗ trợ cho người dùng lập trình cho vi điều khiển các dòng khác nhau. Nó hỗ trợ soạn thảo và biên dịch cả ngôn ngữ C và ngôn ngữ ASM thành ngôn ngữ máy để nạp vào vi điều khiển. Nó cũng hỗ trỡ debug chương trình. Thiết kế sử dụng phần mềm này để lập trình và nạp code cho vi điều khiển STM32L152.**

****

**2.3. Thiết kế module giao tiếp STM32 và module Rhf76-052**

Do phần cứng đã thiết kế STM32 giao tiếp với module thông qua UART2 nên thiết kế sử dụng các funtion có sẵn trong thư viện để giao tiếp UART giữa STM32 và modun Lora. Sau khi đã cấu hình các chân dành cho UART và cấu hình clock bằng phần mềm STM32Cube Mx.



Hình 52- Khởi tạo cho UART

Hàm trên sẽ cấu hình sử dụng Uart2 với tốc độ baund 9600, độ dài dữ liệu 8 bít 1 bít stop, không sử dụng bit parity. Sau khi đã cấu hình UART với những lệnh như trên. Có thể sử dụng hàm sau trong thư viện để gửi mã lệnh đến cho module Lora trong chế độ ngắt.

***HAL\_StatusTypeDef HAL\_UART\_Transmit\_IT(UART\_HandleTypeDef \*huart, uint8\_t \*pData, uint16\_t Size)***

* huart : bộ uart đang được sử dụng (UART2)
* pData: con trỏ đến bộ đệm dữ liệu
* Size: kích thước của dữ liệu để gửi

Sử dụng funtion sau để có thể nhận lại được dữ liệu từ module Lora:

***HAL\_StatusTypeDef HAL\_UART\_Receive\_IT(UART\_HandleTypeDef \*huart, uint8\_t \*pData, uint16\_t Size)***

* huart : bộ uart đang được sử dụng (UART2)
* pData: con trỏ đến bộ đệm dữ liệu
* Size: kích thước của dữ liệu để nhận

Sử dụng hàm sau để có thể nhận lại được dữ liệu từ module Lora:

Cả 2 funtion này đều trả về trạng thái của HAL là OK nếu thành công hay BUSY nếu ngược lại.

Ngoài ra cũng có thể sử dụng các funtion khác trong thư viện hal\_uart.h

Với 2 funtion trên đã có thể giao tiếp dễ dàng để gửi và nhận dữ liệu từ module Lora.

Các hàm khác như:

***HAL\_StatusTypeDef HAL\_UART\_Transmit(UART\_HandleTypeDef \*huart, uint8\_t \*pData, uint16\_t Size, uint32\_t Timeout)***

* huart : bộ uart đang được sử dụng (UART2)
* pData: con trỏ đến bộ đệm dữ liệu
* Size: kích thước của dữ liệu để gửi
* Timeout: thời gian timeout

Hàm này truyền dữ liệu UART theo cơ chế polling.

***HAL\_StatusTypeDef HAL\_UART\_Receive(UART\_HandleTypeDef \*huart, uint8\_t \*pData, uint16\_t Size, uint32\_t Timeout)***

* huart : bộ uart đang được sử dụng (UART2)
* pData: con trỏ đến bộ đệm dữ liệu
* Size: kích thước của dữ liệu để nhận
* Timeout: thời gian timeout

Hàm này nhận dữ liệu UART theo cơ chế polling.

***HAL\_StatusTypeDef HAL\_UART\_Transmit\_DMA(UART\_HandleTypeDef \*huart, uint8\_t \*pData, uint16\_t Size)***

* huart : bộ uart đang được sử dụng (UART2)
* pData: con trỏ đến bộ đệm dữ liệu
* Size: kích thước của dữ liệu để truyền
* Timeout: thời gian timeout

Hàm này truyền dữ liệu UART trong chế độ DMA(Direct Memory Access).

***HAL\_StatusTypeDef HAL\_UART\_Receive\_DMA(UART\_HandleTypeDef \*huart, uint8\_t \*pData, uint16\_t Size)***

* huart : bộ uart đang được sử dụng (UART2)
* pData: con trỏ đến bộ đệm dữ liệu
* Size: kích thước của dữ liệu để nhận
* Timeout: thời gian timeout

Hàm này nhận dữ liệu UART trong chế độ DMA(Direct Memory Access).

**2.4. Thiết kế module phần mềm đọc nhiệt độ độ ẩm từ cảm biến SHT75**

Module này được xây dựng để đọc dữ liệu nhiệt độ độ ẩm từ cảm biến chuyên dụng SHT75.

Define chân DATA và SCK trong thư viện main.h

#define DATA\_OUT PORTB\_12

#define DATA\_IN PINB\_12

#define SHT\_CLK PORTC\_14

*Hàm gửi tín hiệu start:*

void SHT\_Start()

{

GPIO\_Direction(GPIOB,GPIO\_Pin\_12,DIR\_OUT); // Data la chan ra

DATA\_OUT = 1;

SHT\_CLK = 0;

SHT\_CLK = 1;

DATA\_OUT = 0;

SHT\_CLK = 0;

SHT\_CLK = 1;

DATA\_OUT = 1;

SHT\_CLK = 0;

}

*Hàm viết 1 byte dữ liệu:*  
uint8\_t SHT\_WriteByte(uint8\_t Data)

{

uint8\_t i, error = 0;

GPIO\_Direction(GPIOB,GPIO\_Pin\_12,DIR\_OUT); // set la cong ra // Data la chan ra

delay\_us(2);

for(i = 0x80; i > 0; i /= 2)

{

SHT\_CLK = 0;

if(i & Data) DATA\_OUT = 1;

else DATA\_OUT = 0;

delay\_us(1);

SHT\_CLK = 1;

delay\_us(1);

}

SHT\_CLK = 0;

GPIO\_Direction(GPIOB,GPIO\_Pin\_12,DIR\_IN); // Data la chan vao

delay\_us(1);

SHT\_CLK = 1;

delay\_us(1);

error = DATA\_IN;

SHT\_CLK = 0;

delay\_ms(250);

return(error);

}

*Hàm đọc 1 byte dữ liệu:*  
uint8\_t SHT\_ReadByte(uint8\_t ack)

{

uint8\_t i, val = 0;

GPIO\_Direction(GPIOB,GPIO\_Pin\_12,DIR\_IN); // Data la chan vao

for(i = 0x80; i > 0; i /= 2)

{

SHT\_CLK = 1;

delay\_us(1);

if(DATA\_IN) val = val | i;

delay\_us(1);

SHT\_CLK = 0;

}

GPIO\_Direction(GPIOB,GPIO\_Pin\_12,DIR\_OUT); // Data la chan ra

delay\_us(1);

DATA\_OUT = ! ack;

SHT\_CLK = 1;

delay\_us(1);

SHT\_CLK = 0;

return(val);

}

*Hàm đọc 16 bít dữ liệu:*

uint16\_t SHT\_ReadSenSor(uint8\_t Command)

{

uint8\_t msb, lsb, crc;

SHT\_ResetConection();

SHT\_WriteByte(Command);

while(DATA\_IN);

msb = SHT\_ReadByte(SHT\_ACK);

lsb = SHT\_ReadByte(SHT\_ACK);

crc = SHT\_ReadByte(SHT\_NOACK);

return(((uint16\_t) msb << 8) | (uint16\_t) lsb);

}

*Hàm chuyền đổi nhiệt độ và độ ẩm:*

void SHT\_ReadTemHumi(float \*tem, float \*humi)

{

uint16\_t SOT;

uint16\_t SORH;

SOT=SHT\_ReadSenSor(SHT\_MEASURE\_TEMP);

SORH=SHT\_ReadSenSor(SHT\_MEASURE\_HUMI);

if(SHT\_Resolution==SHT\_14\_12\_BIT)

{

\*tem=(H\_D1+H\_D2\*SOT);

\*humi=((H\_D1+H\_D2\*SOT- 25)\*(H\_T1+H\_T2\*SORH)+H\_C1+H\_C2\*SORH+H\_C3\*SORH\*SORH);

}

else

{

\*tem=(L\_D1+L\_D2\*SOT);

\*humi=((L\_D1+L\_D2\*SOT-25)\*(L\_T1+L\_T2\*SORH)+L\_C1+L\_C2\*SORH+L\_C3\*SORH\*SORH);

}

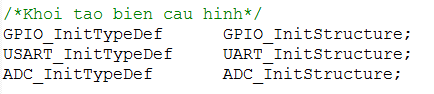
}

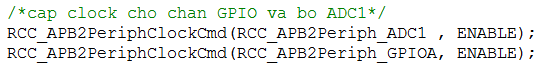
**2.4. Thiết kế module phần mềm đọc dữ liệu từ cảm biến độ ẩm đất và cảm biến đo pH**

Cảm biến độ ẩm đất sử dụng là loại cảm biến có thể giao tiếp với STM32 bằng cách đọc giá trị ADC qua 1 kênh ADC đầu vào.

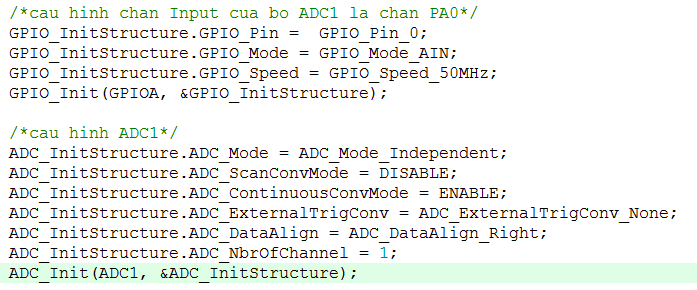
Tương tự cảm biến đo pH cũng sử dụng kênh ADC đầu vào để đo dữ liệu. Cho nên các cấu hình bên dưới đều có thể dùng để đọc dữ liệu từ cảm biến đo pH.

Khởi tạo các định nghĩa và cấp clock cho ngoại vi cũng như bộ ADC.:

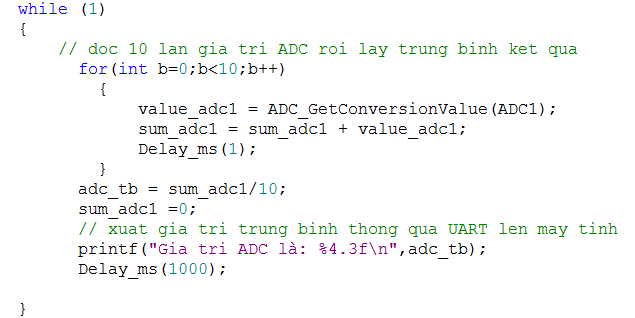




Cấu hình chân GPIO Input ADC và cấu hình chung ADC:



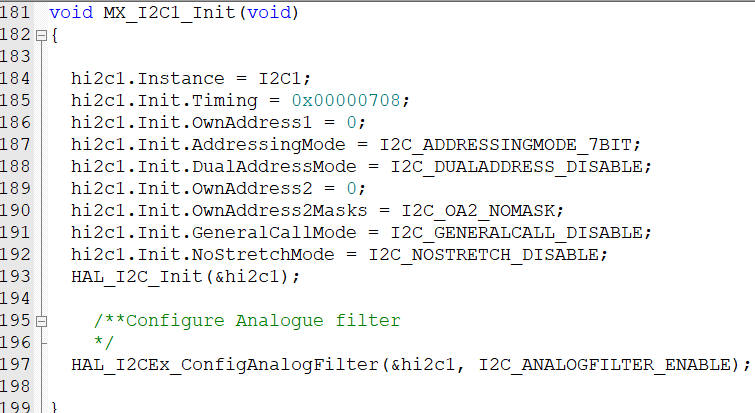
Sử dụng phương pháp trung bình để đọc kết quả chính xác hơn:



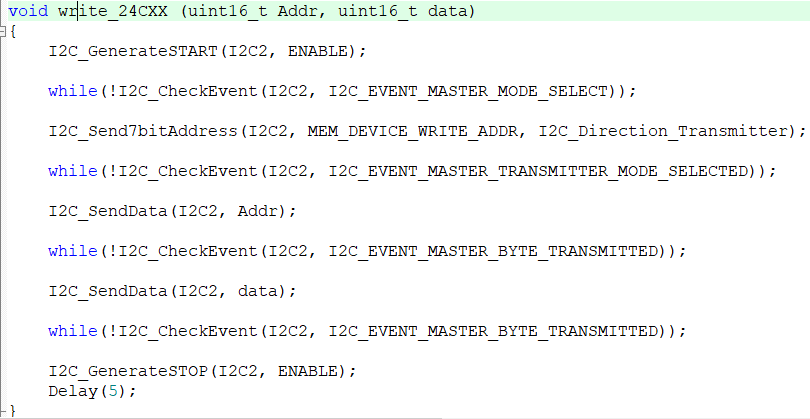
**2.5. Thiết kế module phần mềm ghi dữ liệu vào EEPROM**

Để dữ liệu được lưu trữ vào trong EEPROM xây dựng module để ghi dữ liệu vào EEPROM thông qua I2C.

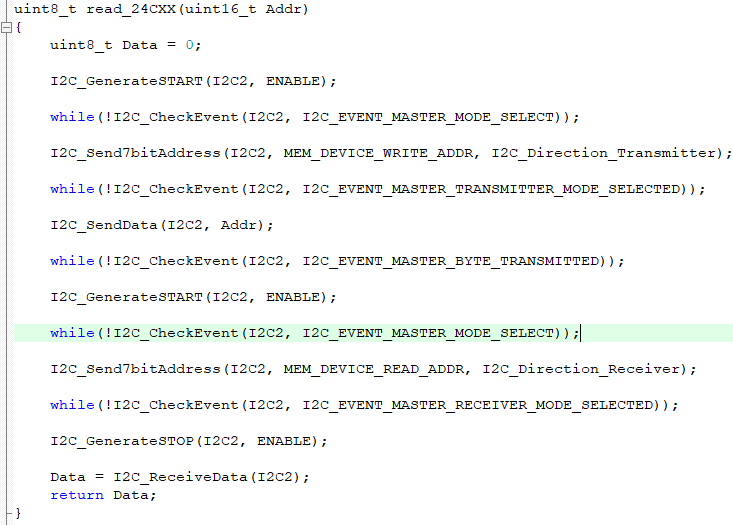
Khởi tạo I2C:

****

Hàm ghi 1byte vào EEPROM:

****

Hàm trình đọc 1byte từ EEPROM:



# CHƯƠNG 3: KẾT LUẬN

**Kết luận:**

Để có thể giao tiếp thuận tiện giữa MCU và các cảm biến cần có các moulde phần mềm được lập trình để có thể đọc dữ liệu và truyền dữ liệu 1 cách tốt nhất. Trong khi lập trình cần chú ý về độ chính xác của dữ liệu và thời gian truyền dữ liệu được đảm bảo.