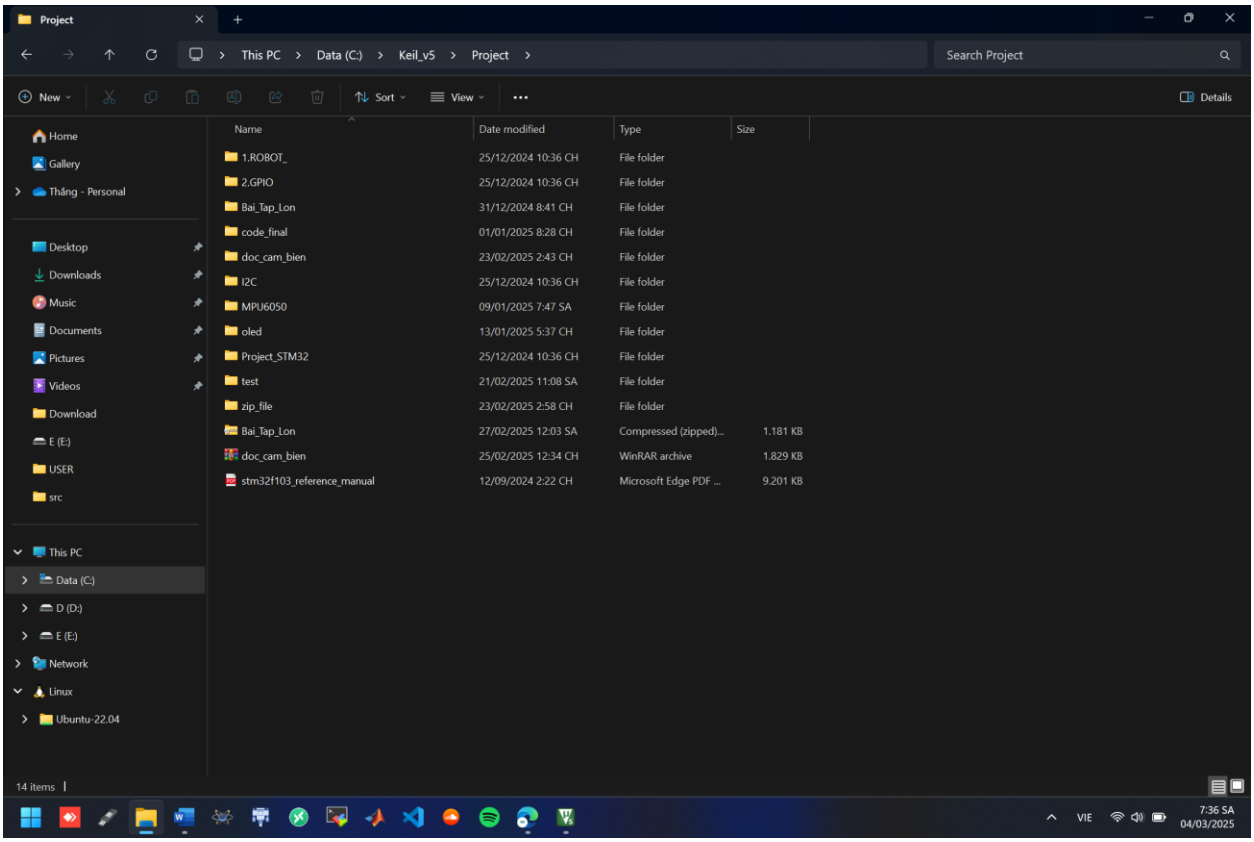
Đọc cảm biến DHT11 bằng stm32f103cc8t6

Note: đọc datasheet của cảm biến và tài liệu rm của stm32 trước khi code

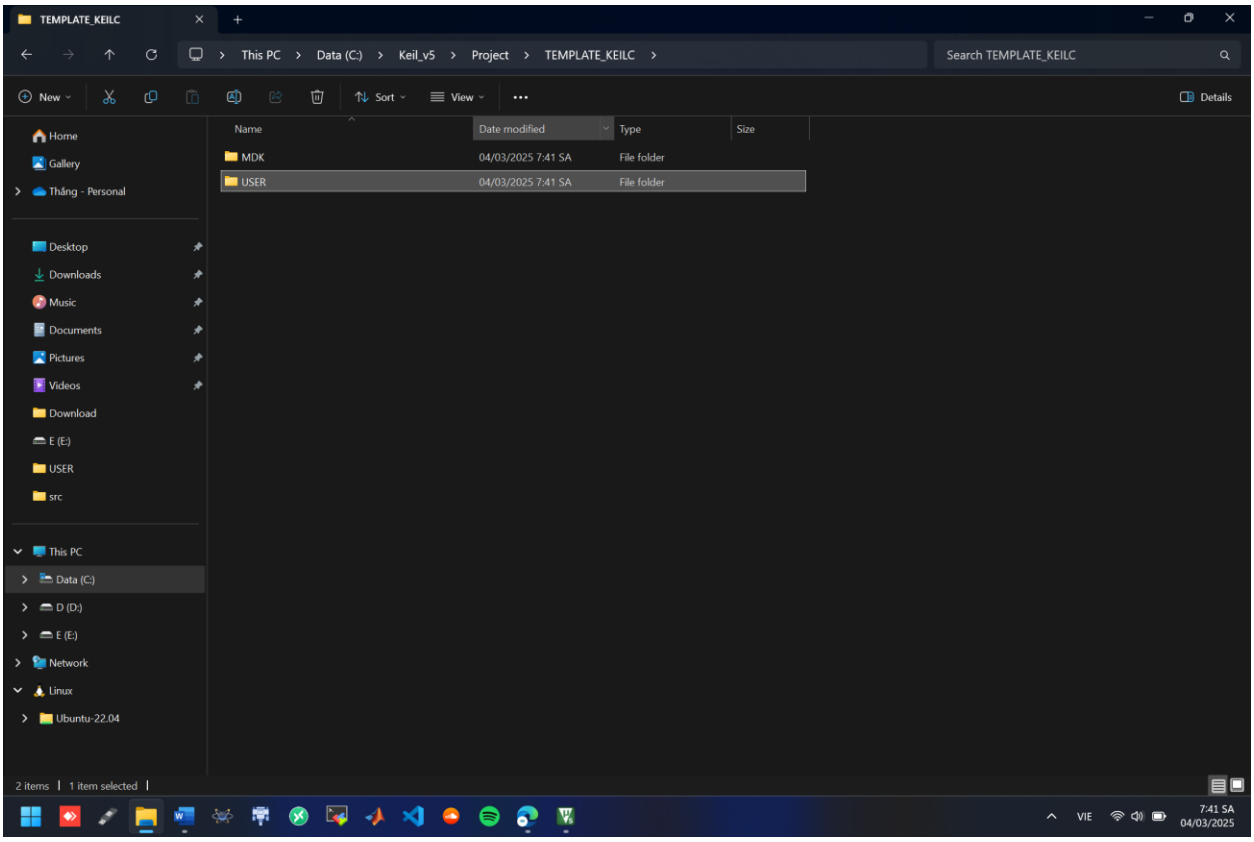
I. Cách tạo project keilc

(trước khi tạo project mới thì nên tạo 1 folder để lưu các dự án trong các folder dự án nên chia thành các folder nhỏ hơn để dễ dàng quản lý dưới đây là 1 template mẫu)

- Đây là thư mục để lưu các dự án

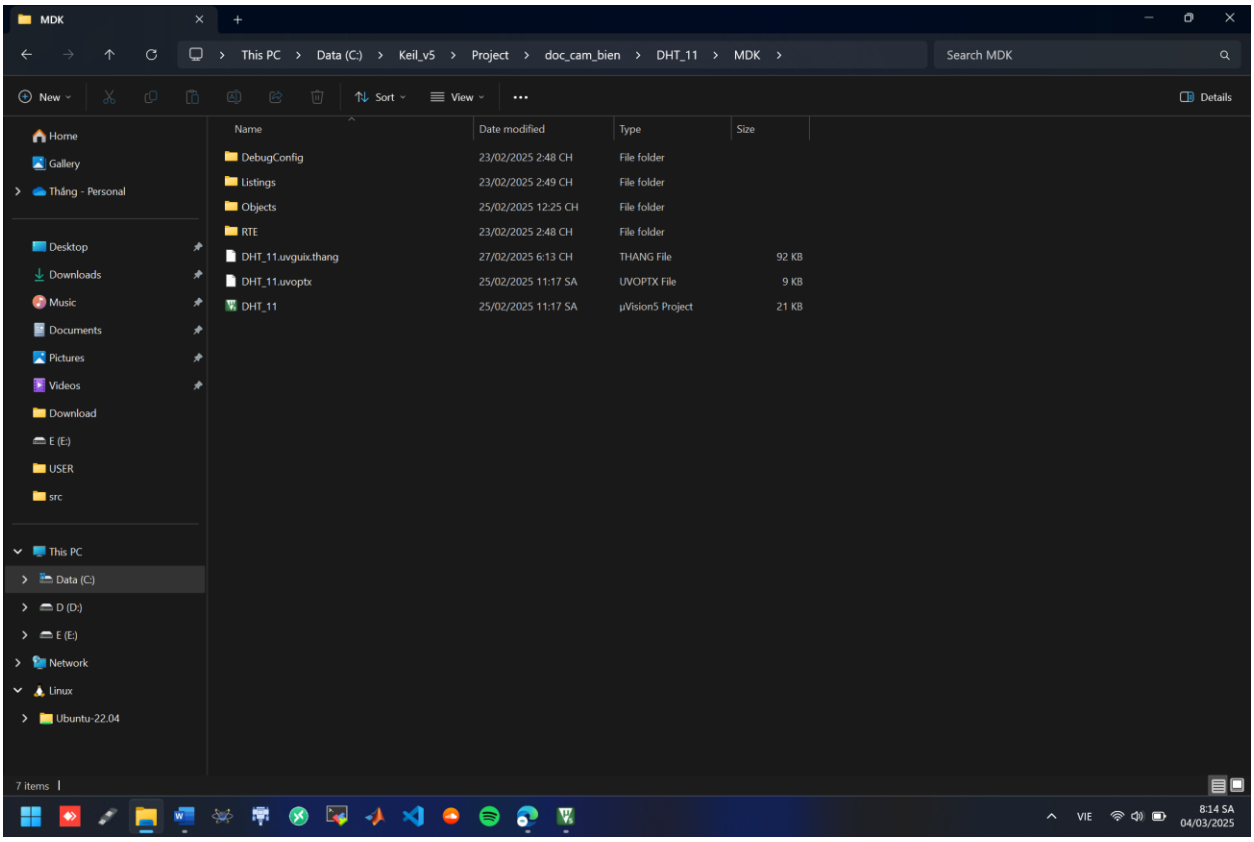


- Trong folder dự án tạo 2 thư mục , 1 thư mục để lưu các file cấu hình (MDK), 1 thư mục để lưu Src code (USER)

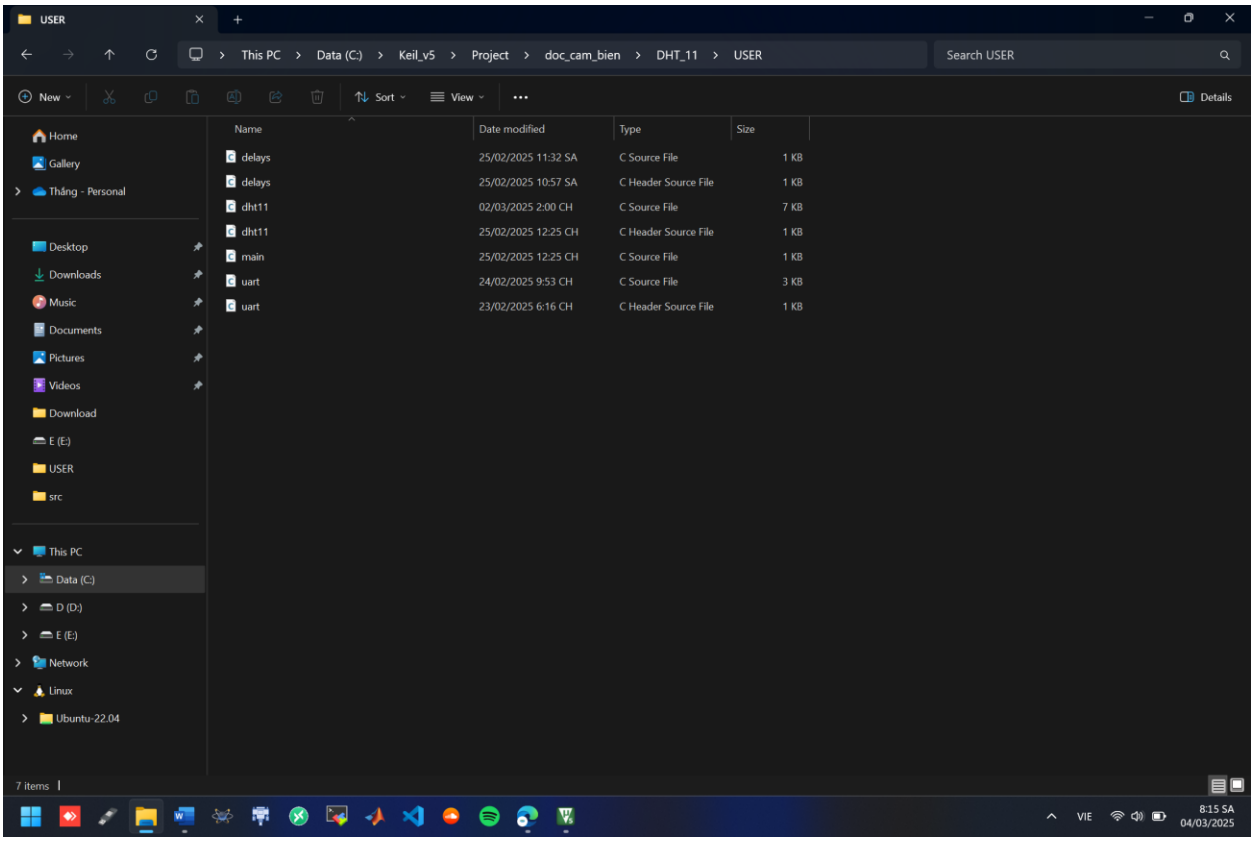


Ví dụ:

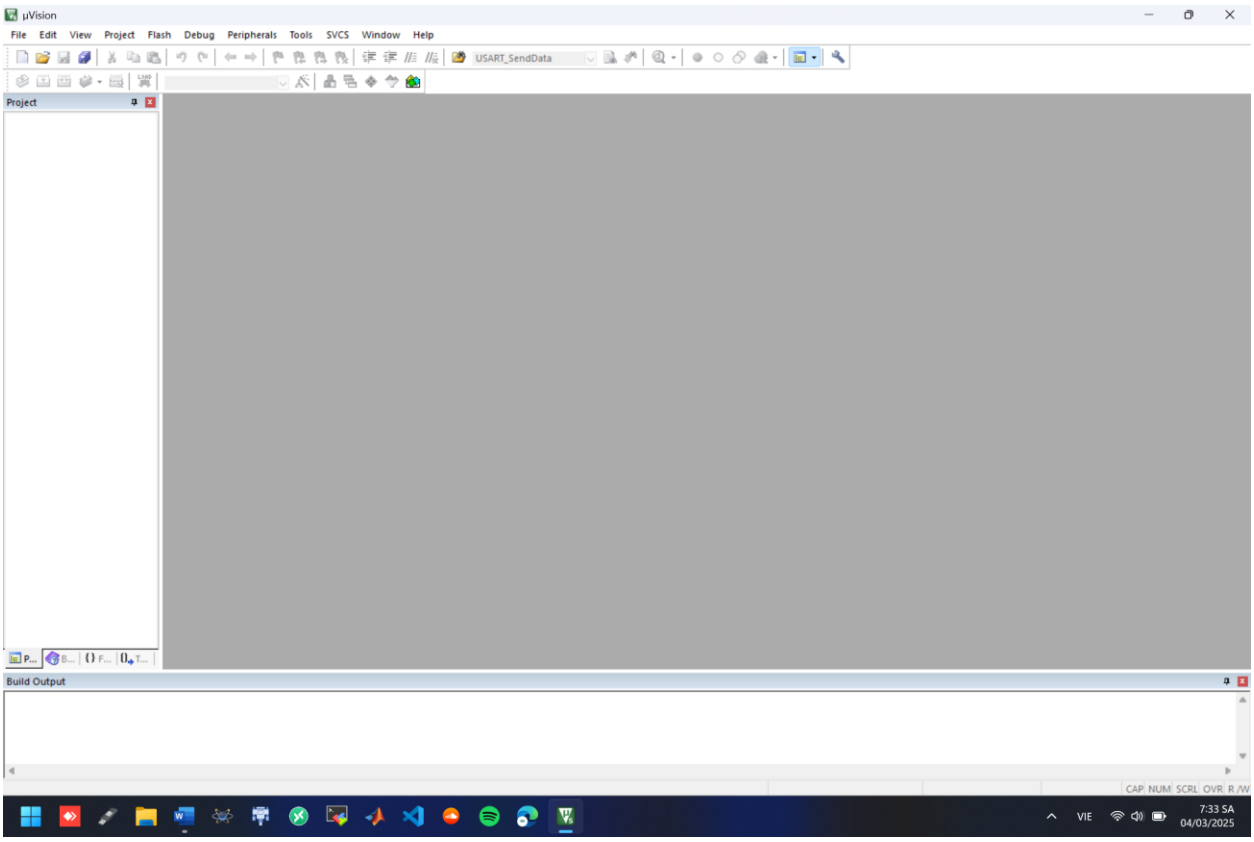
- Trong MDK sẽ có như sau:



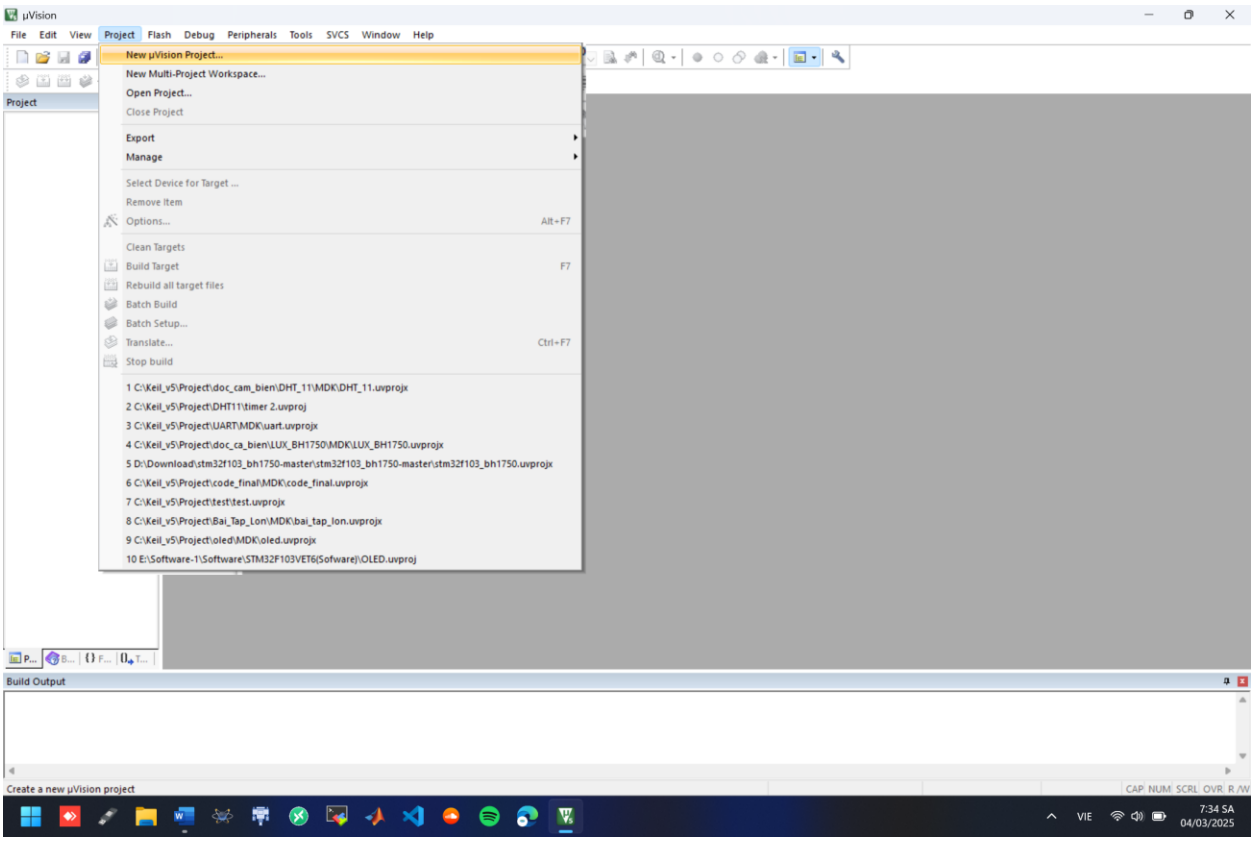
- Trong USER sẽ có



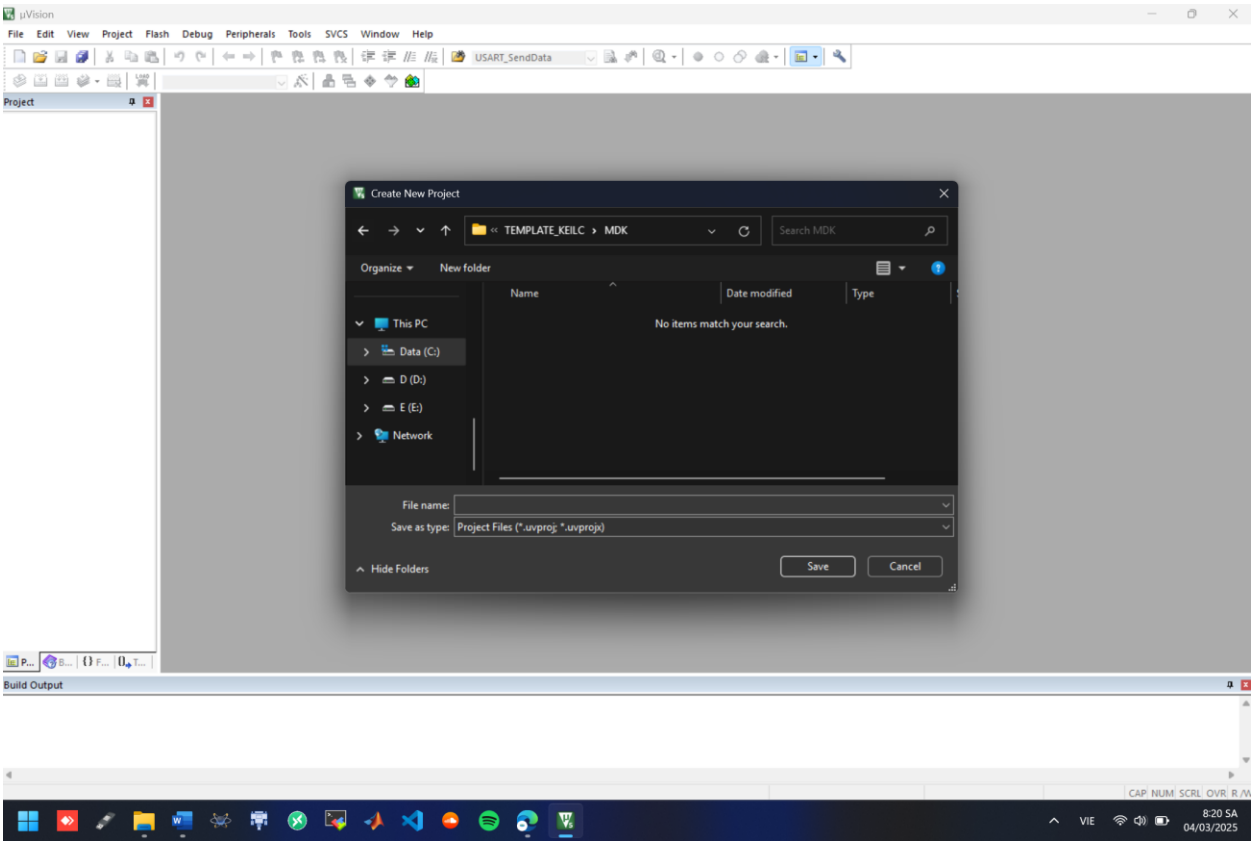
1. Cài và crack keilc(nếu chưa có)
2. Mở keilc lên và sẽ có giao diện sau



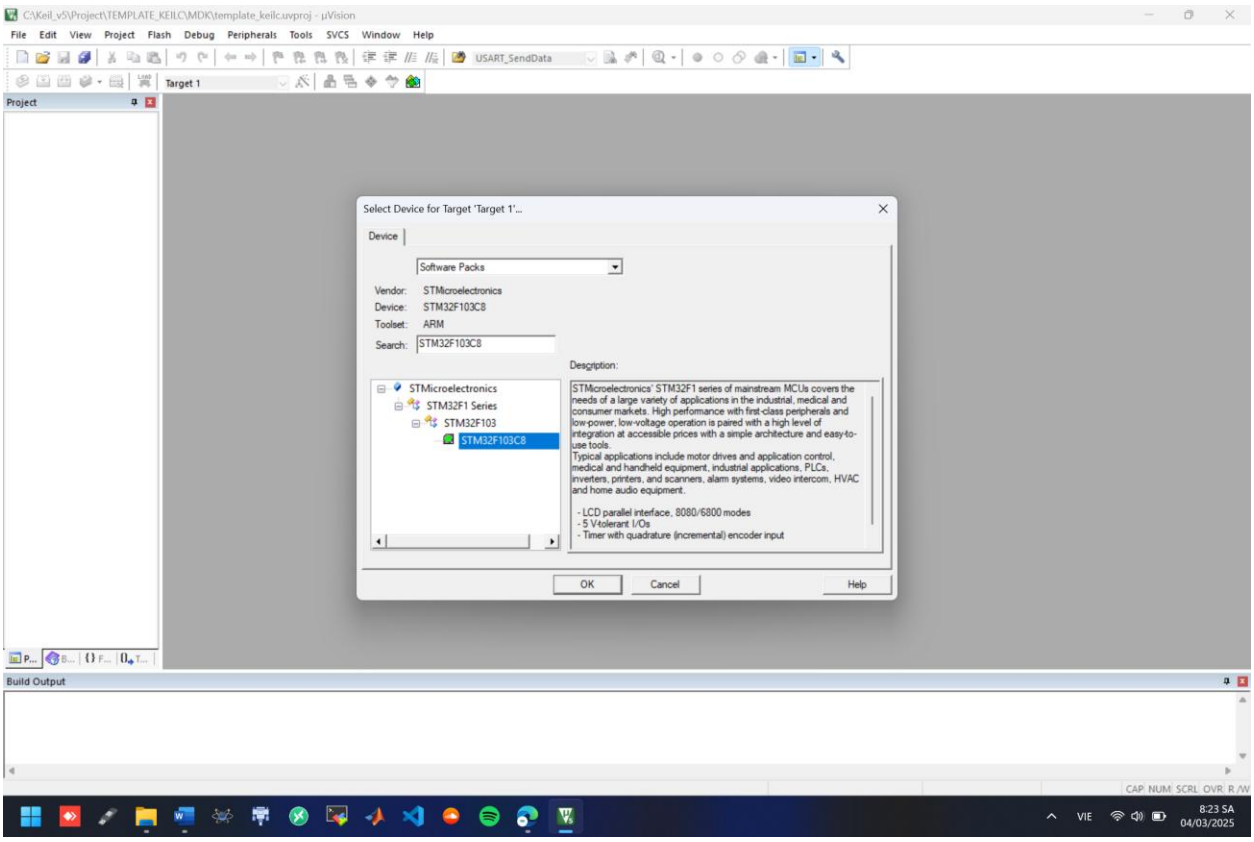
3. Chọn vào project -> new uvision project



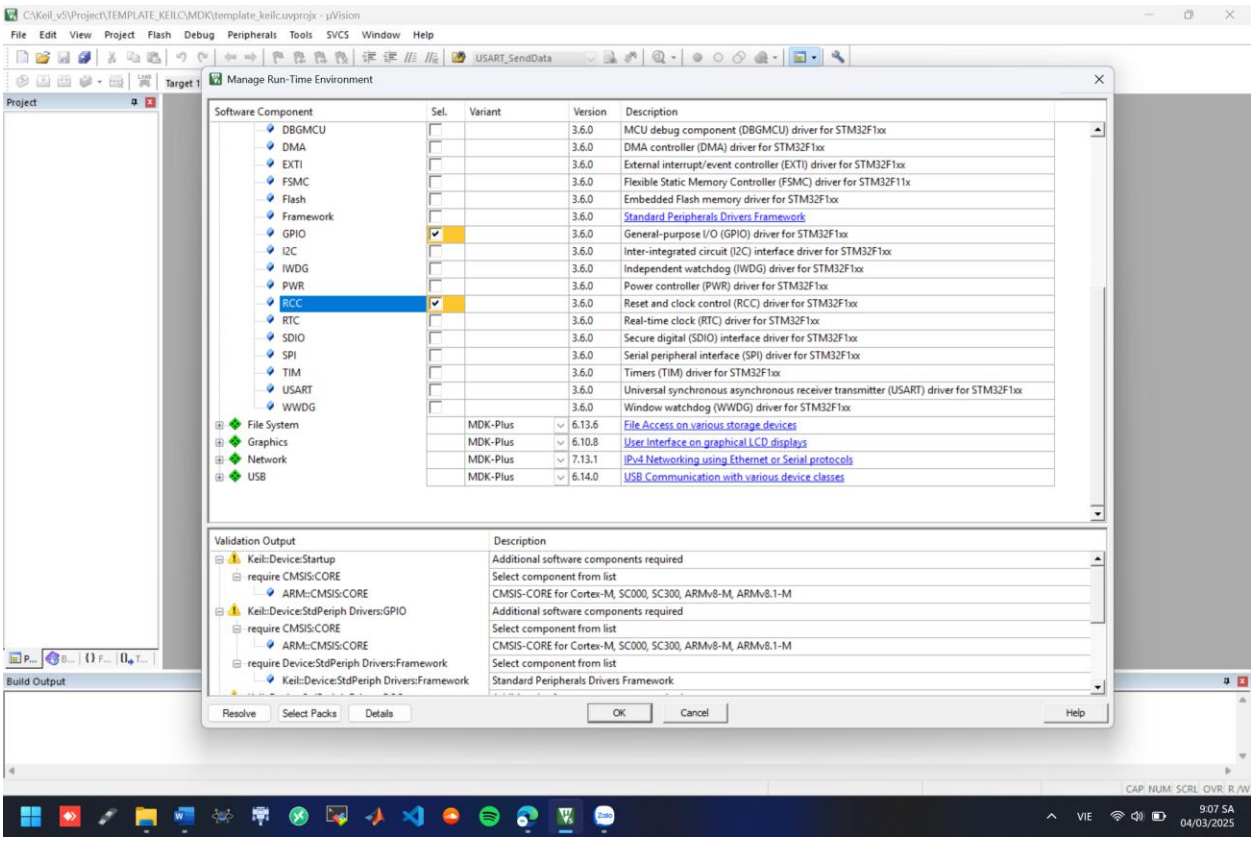
4. Trỏ đến chỗ folder MDK vừa tạo và lưu vào đó (nhớ chọn file name : cái này tùy chọn



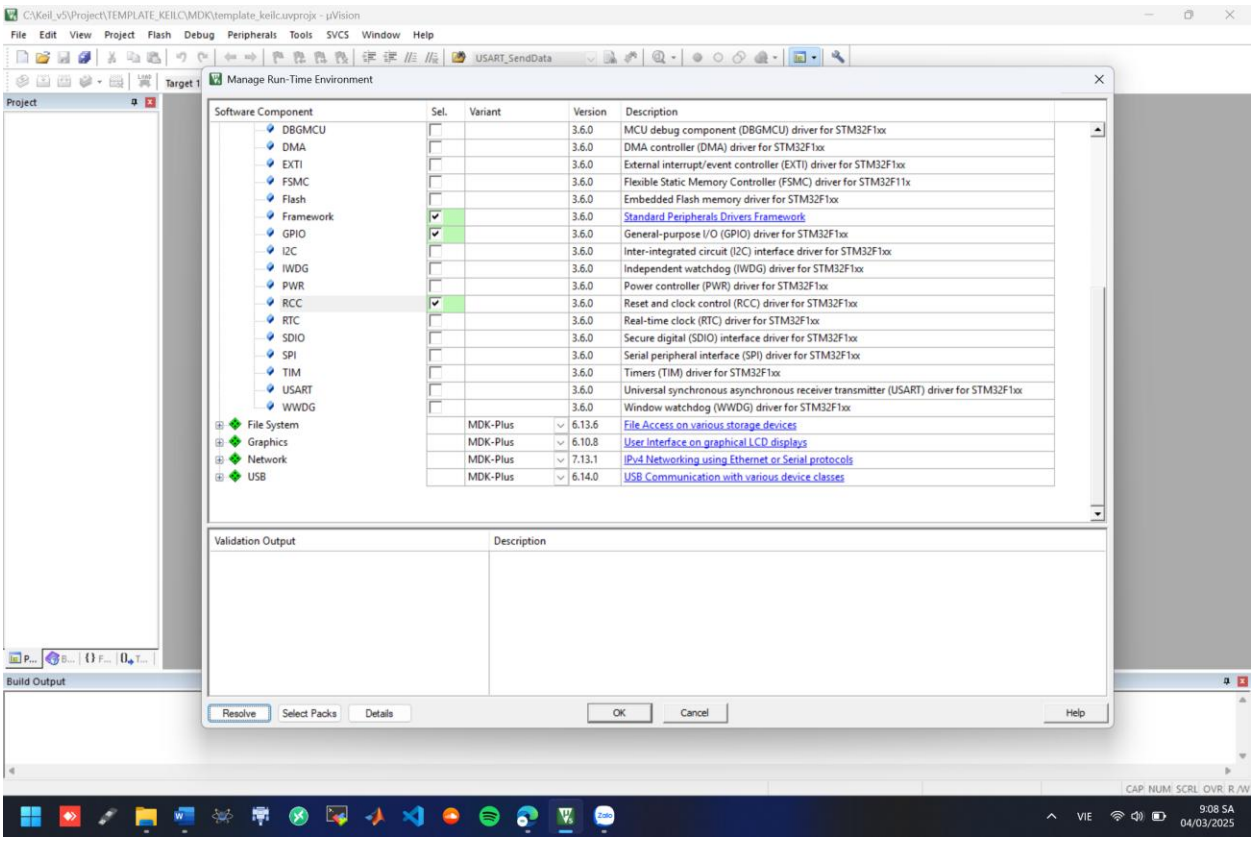
- Sau đó sẽ hiện ra giao diện sau , ở đây sẽ chọn device mà ta cần(STM32F103C8)



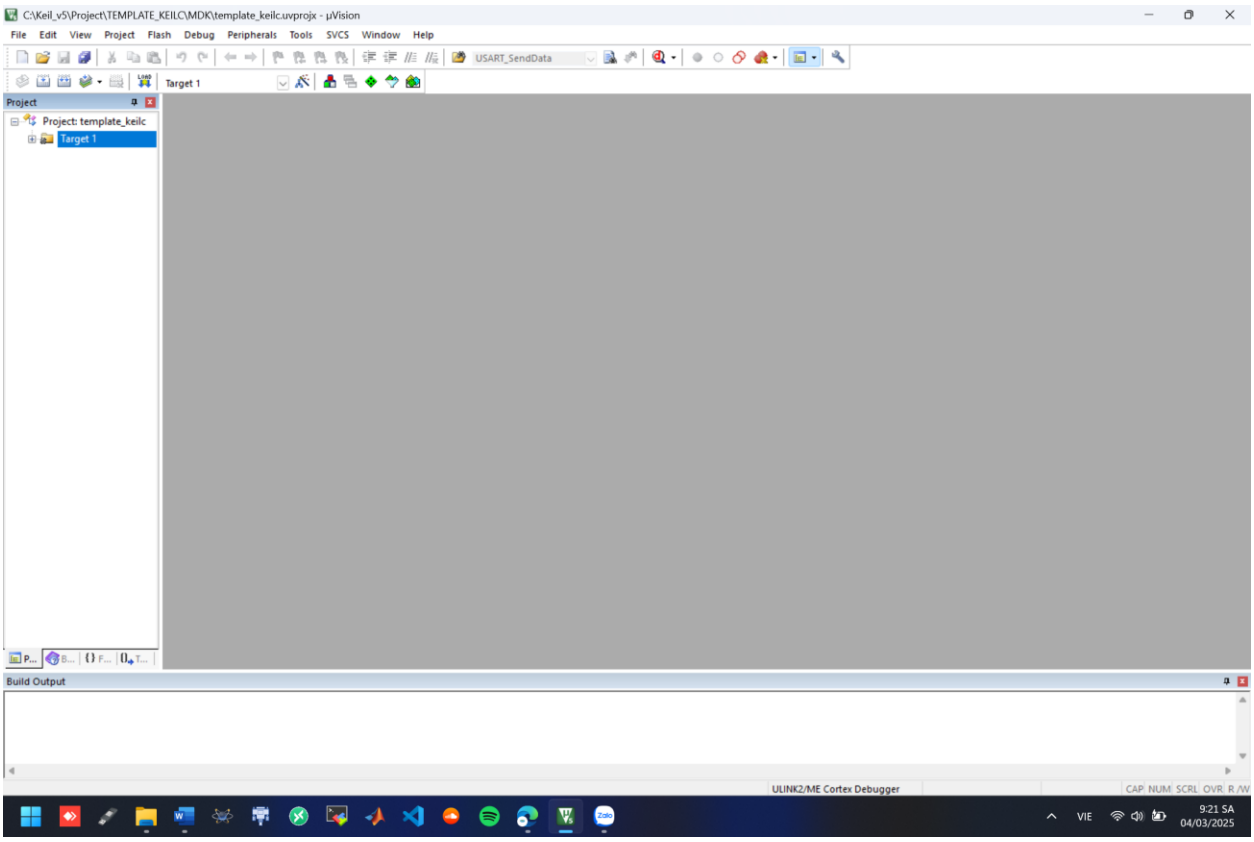
- Sau khi chọn device xong thì sẽ có giao diện sau , ở đây chúng ta sẽ chọn device -> tích vào startup , tiếp đến chọn std driver ở đây ta sẽ chọn các driver cần để code như gpio , rcc , ...



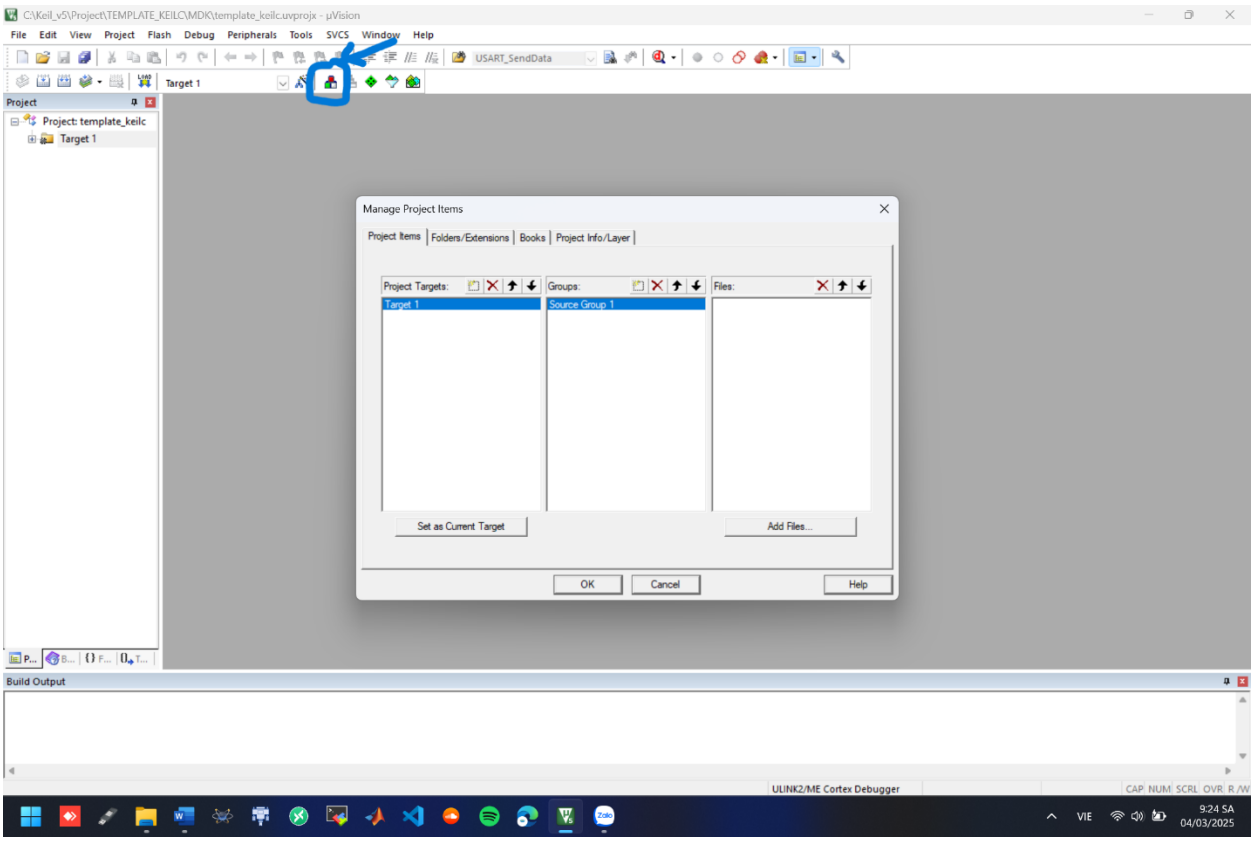
- cuối cùng tích vào Resolve và chọn ok.



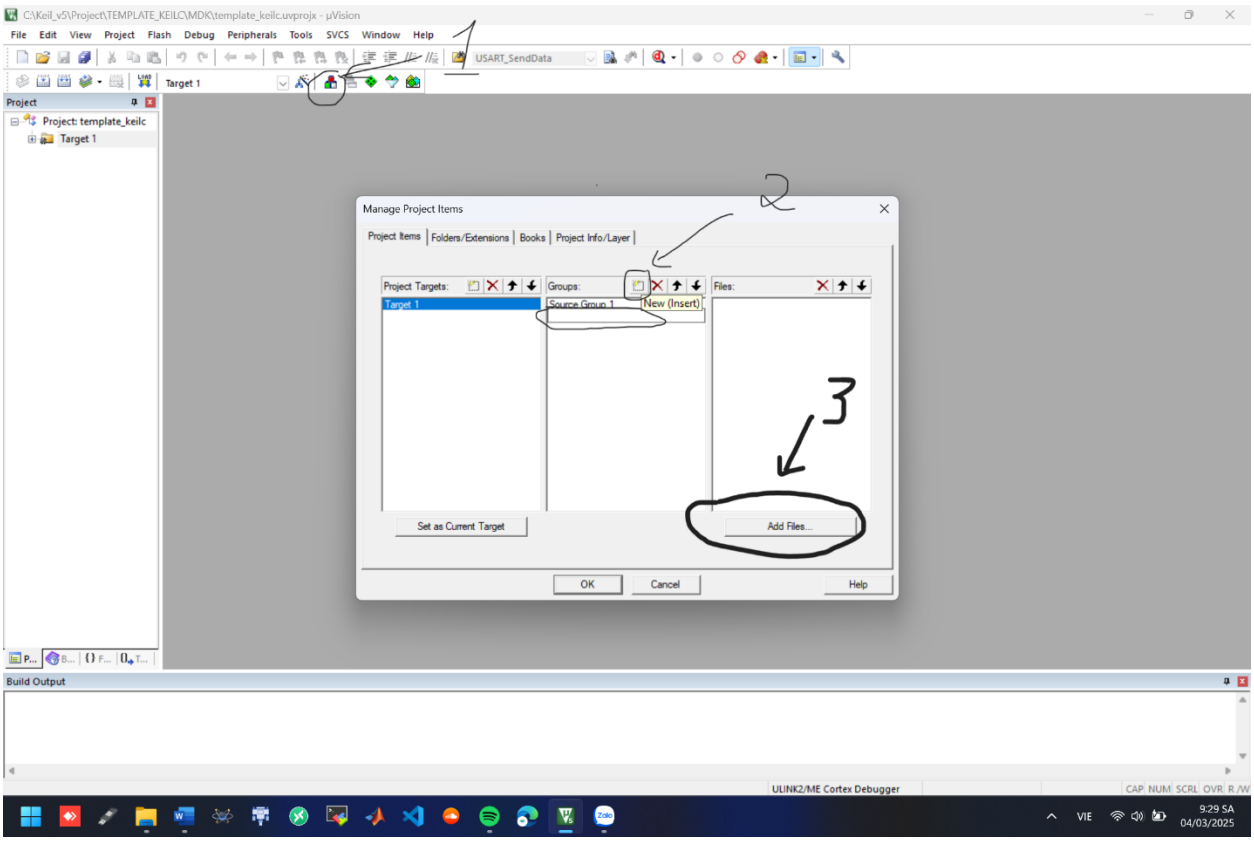
- Sau đó hiện ra:



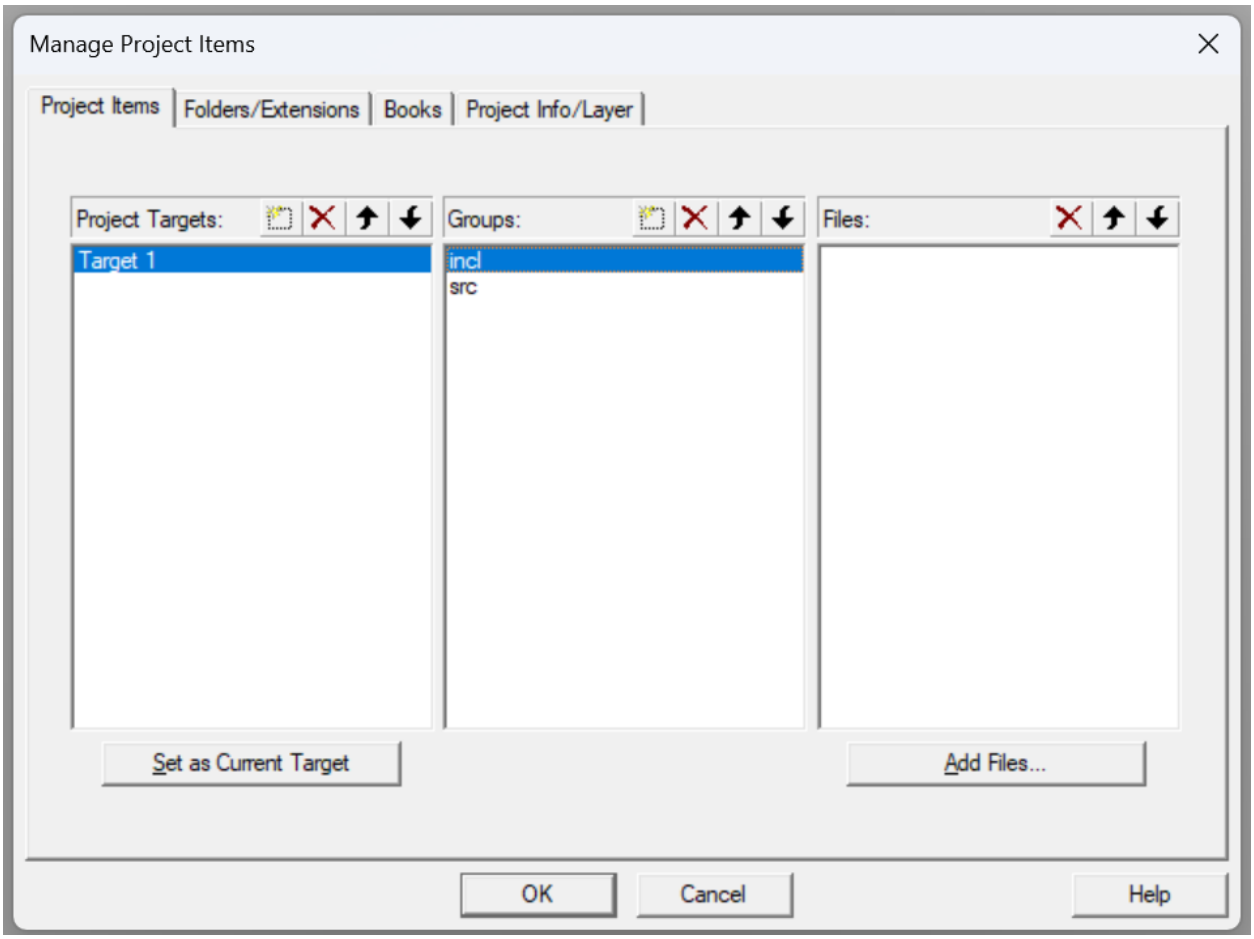
5. Tiếp chúng ta sẽ thêm các thư mục vào dự án



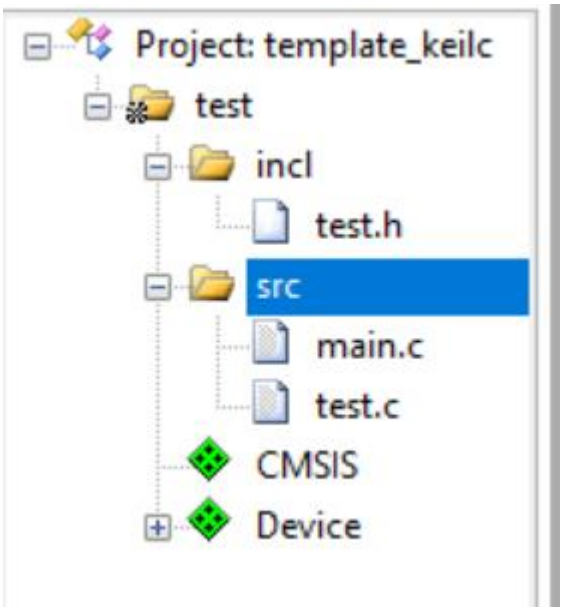
- ở dây chúng ta sẽ tạo 2 thư mục để lưu code :
    - incl : nơi để lưu các file.h (các thư viện mà mình tự viết ra) : dht11.h , delay.h ,....
    - src : nơi để lưu các file.c: dht11.c , delay.c, main.c,.....



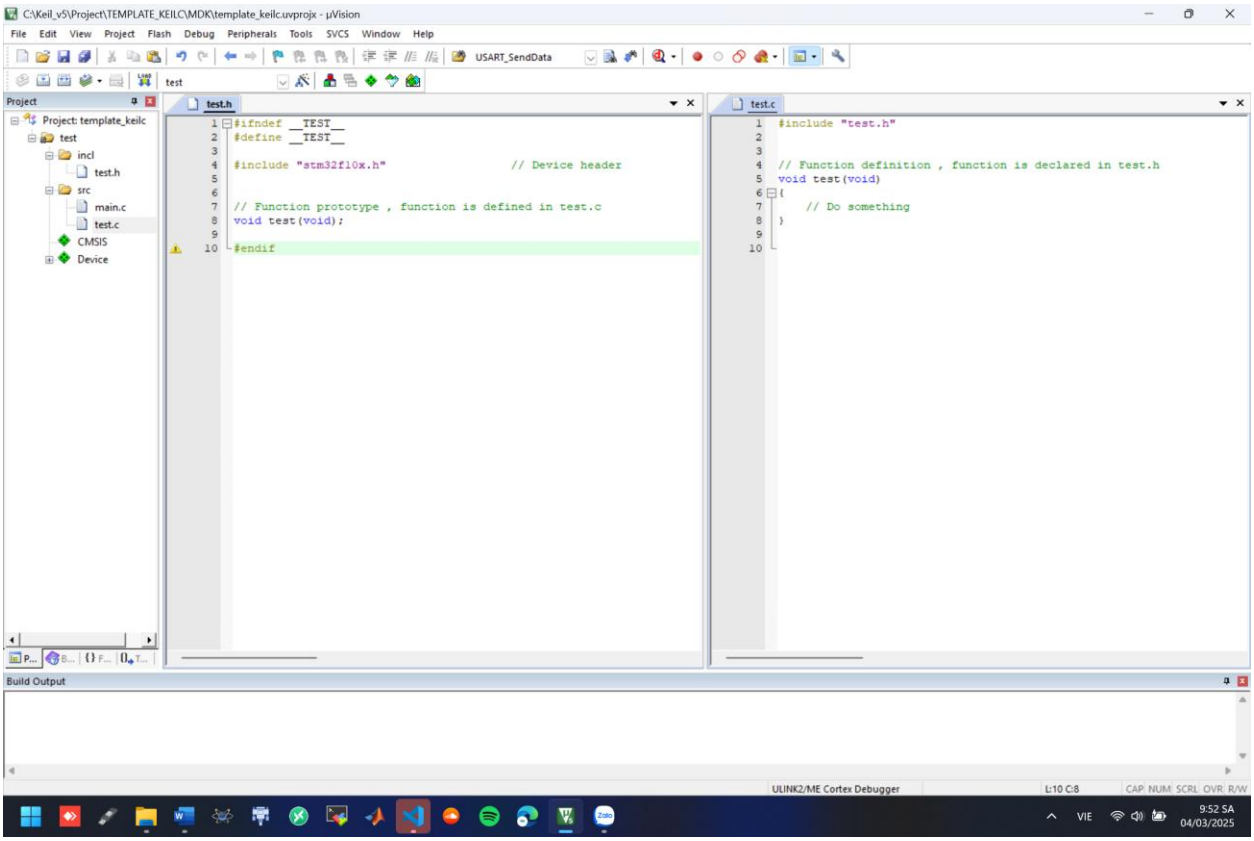
- sửa tên các Group thành incl và src


Manage Project Items
Project Items
Folders/Extensions
Books
Project Info/Layer
Project Targets:
Target 1
Groups:
incl
src
Files:
Set as Current Target
Add Files...
OK
Cancel
Help

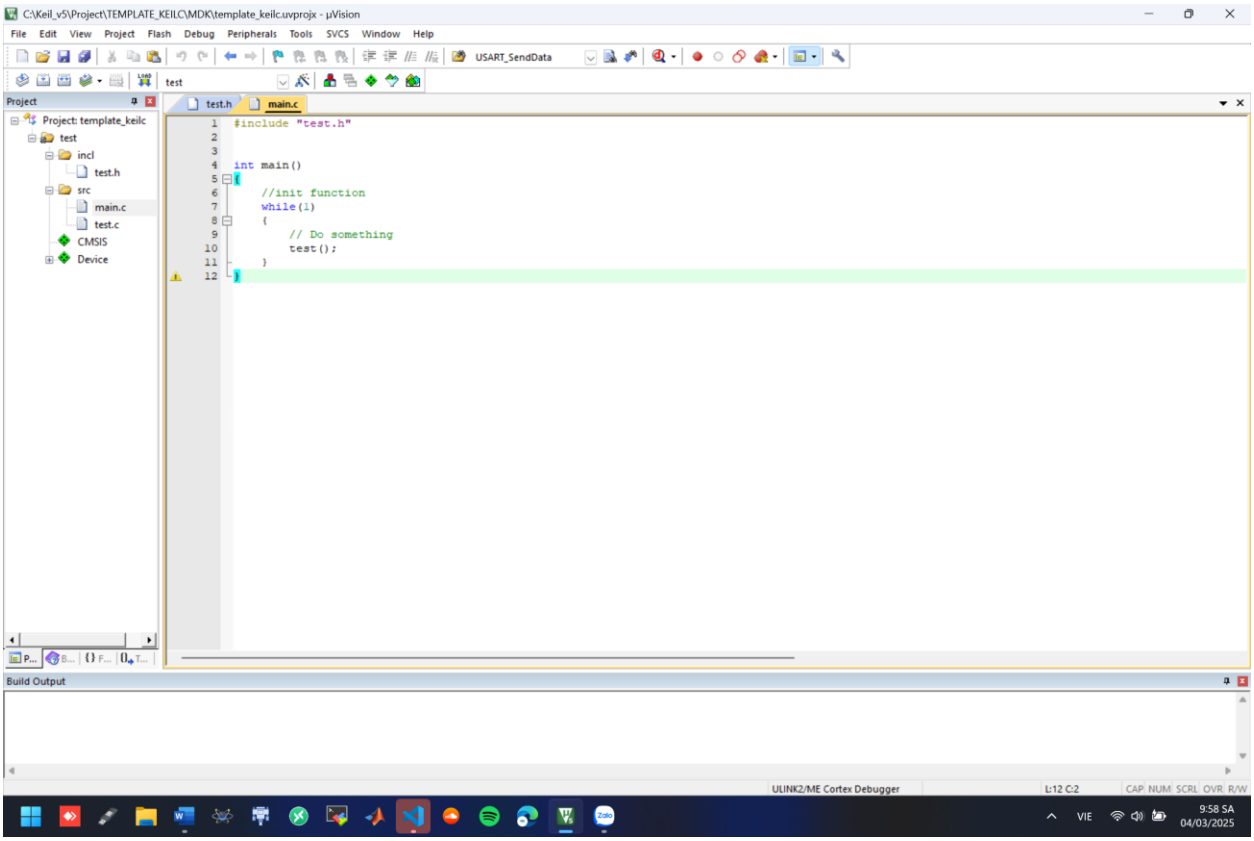
- sau khi thêm các file thì ta sẽ có như thế này



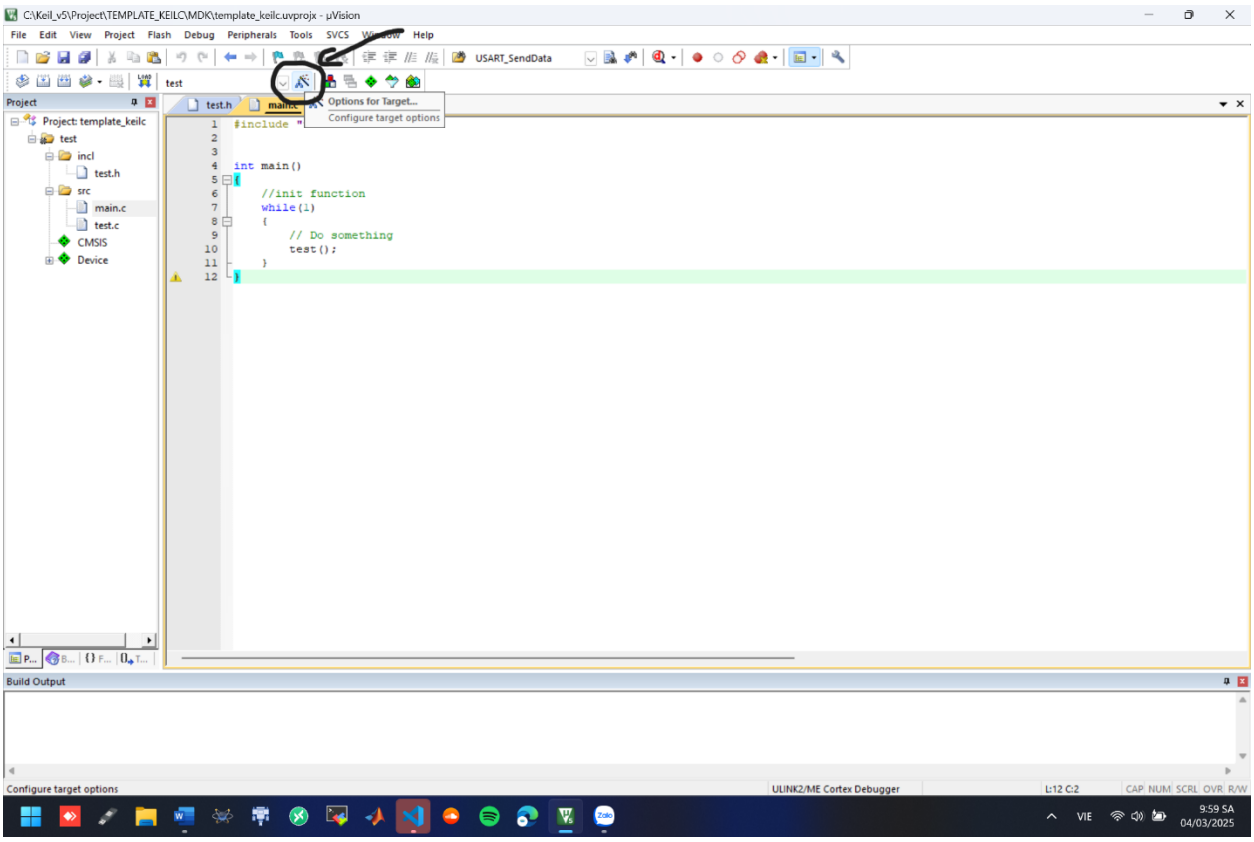
6. template mẫu của test.h , test.c



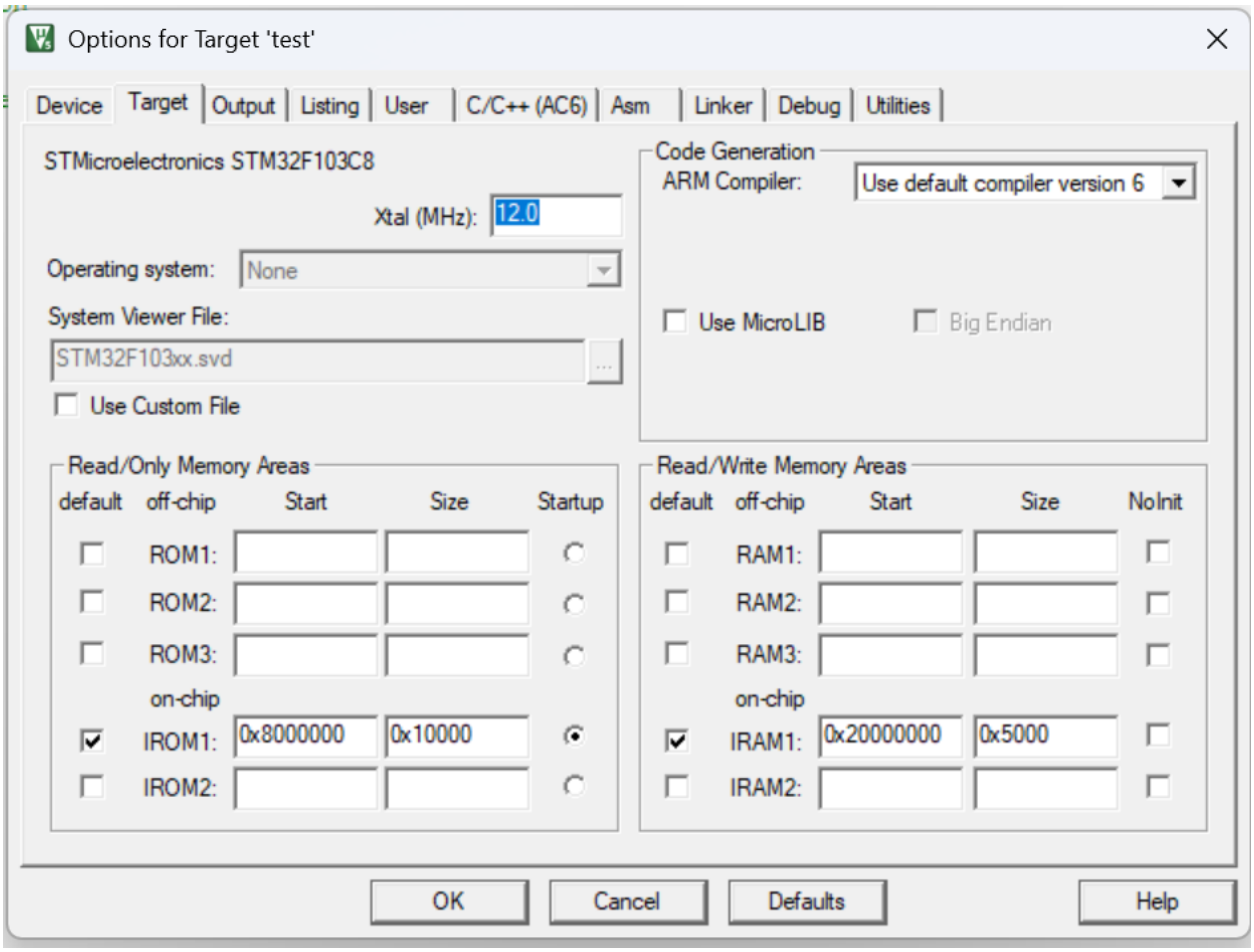
- ở main.c chúng ta sẽ include file test.h và gọi hàm test để sử dụng



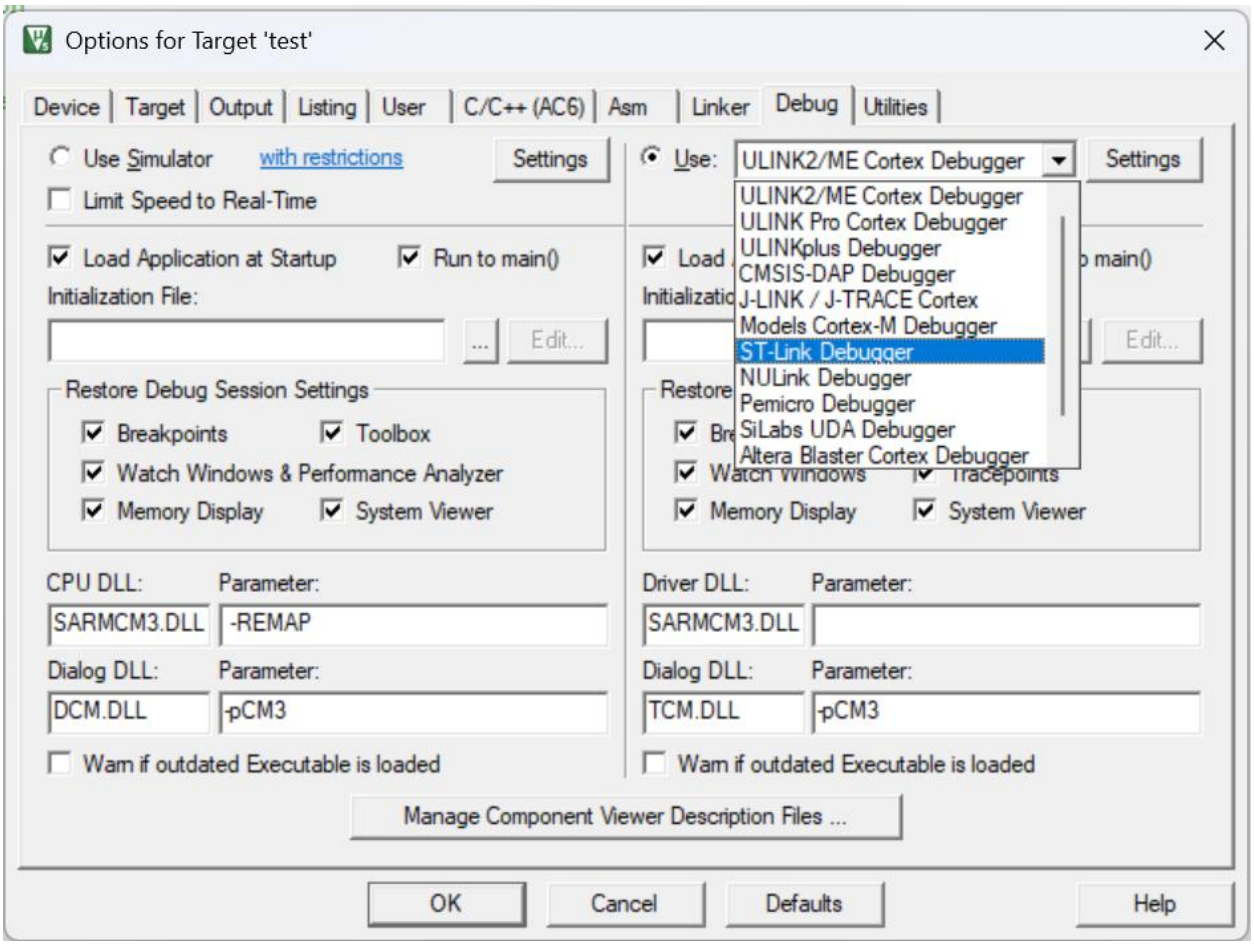
7. cuối cùng chúng ta cần config thêm vài thứ ở trong dự án



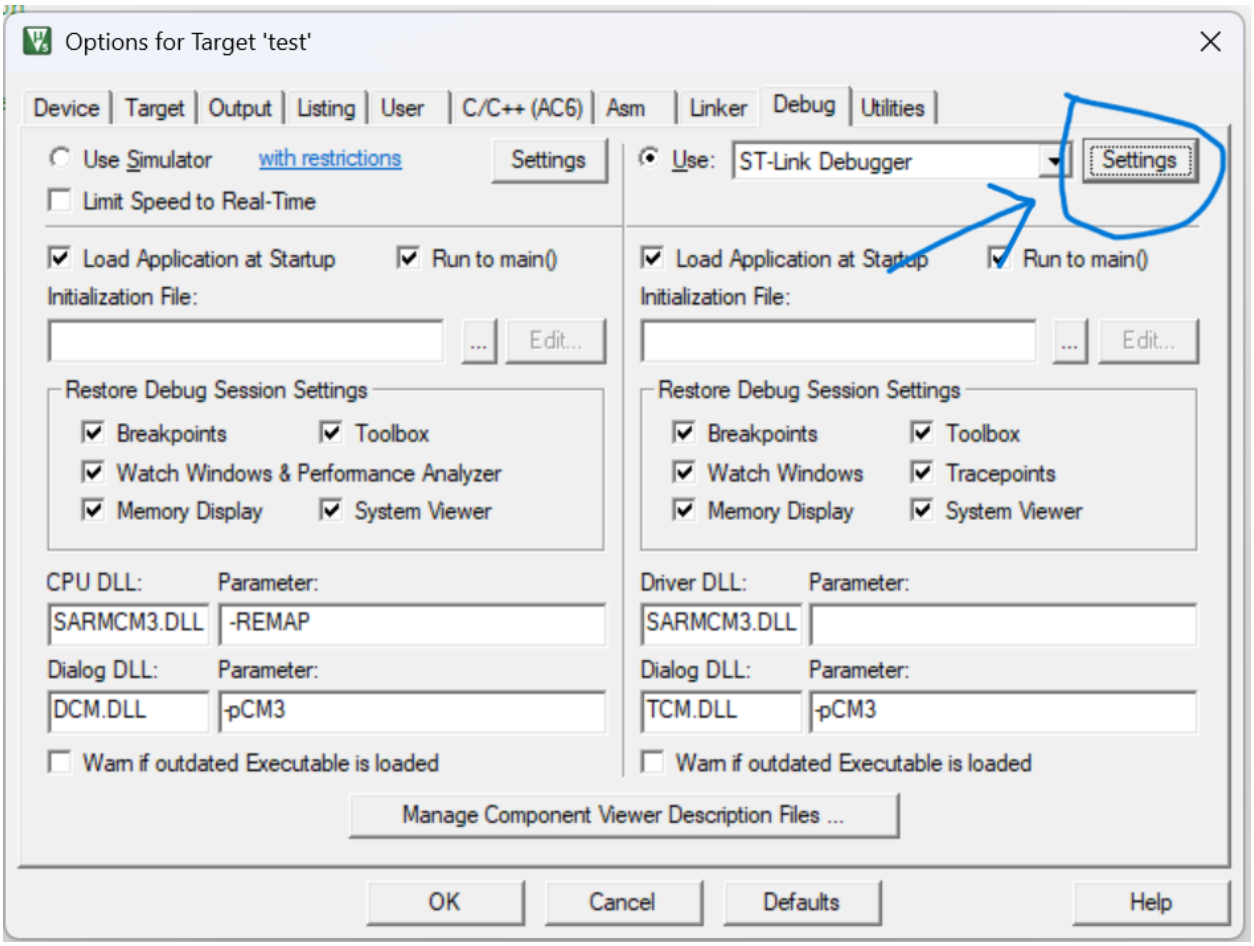
- sau khi chọn vào đây sẽ hiện ra cửa sổ giao diện



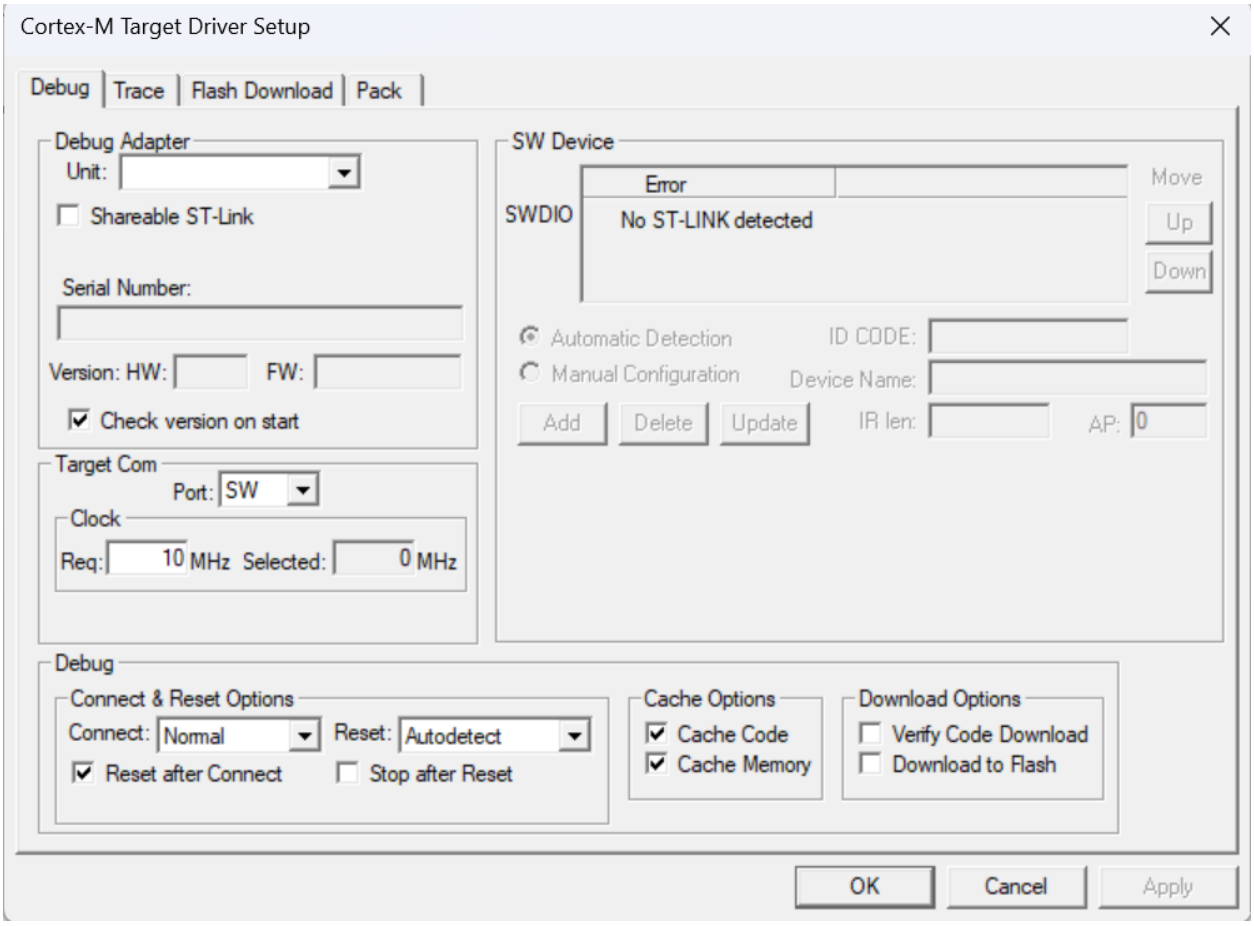
- trong giao diện này chúng ta sẽ sửa những thứ sau:
    - chọn Debug -> sửa thành stlink



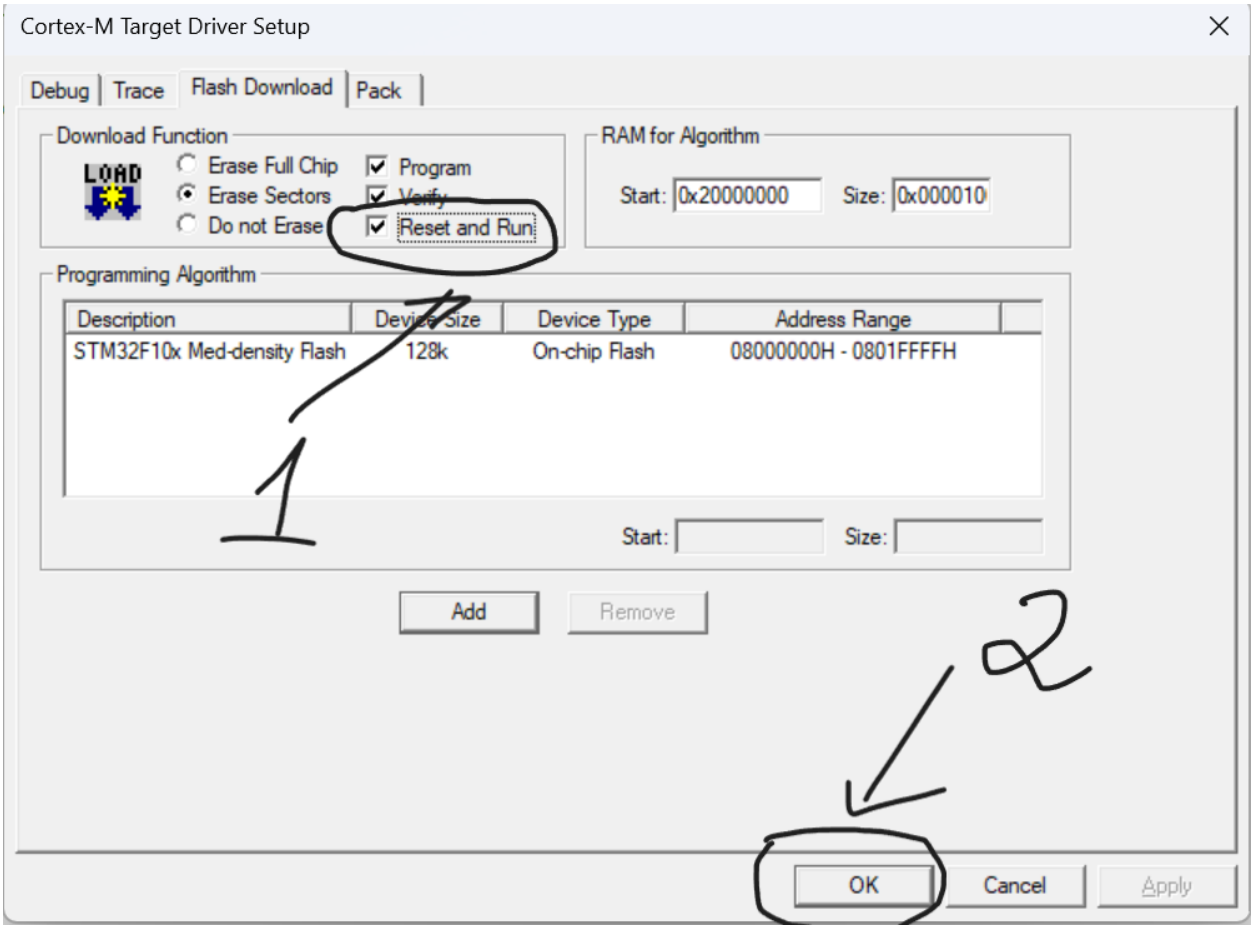
- tiếp đến chọn settings



- sau đó sẽ hiện ra giao diện sau



- tiếp đó chọn vào mục falsh download



Sau khi làm xong các bước trên hãy build thử project xem có lỗi không nếu không lỗi thì đã thành công

```
Build Output
Build started: Project: template_keilc
*** Using Compiler 'V6.14', folder: 'C:\Keil_v5\ARM\ARMCLANG\Bin'
Build target 'test'
".\Objects\template_keilc.axf" - 0 Error(s), 0 Warning(s).
Build Time Elapsed:  00:00:00
```

II. Cấu hình timer2

1. Cấu hình cho timer2
   - Bật clock cho timer2
   - Thiết lập mode đếm cho timer2

- Thiết lập giá trị cho chu kì của bộ đếm thời gian (đọc rm để rõ hơn).
- Thiết lập giá trị cho bộ chia.
- Khởi tạo bộ đếm thời gian với các thiết lập trên.
- Kích hoạt bộ đếm thời gian.

```
void Timer2_Init(void)
{
    // Enable clock for TIM2
    RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM2, ENABLE);
    // Init timer
    TIM_TimeBaseInitTypeDef timerInit;
    // mode up counter
    timerInit.TIM_CounterMode = TIM_CounterMode_Up;
    // period = 0xFFFF
    timerInit.TIM_Period = 0xFFFF;
    // prescaler = 72 - 1
    timerInit.TIM_Prescaler = 72 - 1;
    // clock division
    TIM_TimeBaseInit(TIM2, &timerInit);
    // Enable TIM2
    TIM_Cmd(TIM2, ENABLE);
}
```

2. Viết các hàm cần dùng
   - Hàm delay_1ms() : dùng bộ đếm của timer2 đếm đến khi đạt đến 1000
   - Hàm delay_ms () : gọi hàm delay_1ms để tạo số thời gian muốn delay

Note : cần khởi tạo timer trước khi khởi tạo cảm biến

I. cảm biến DHT11

1. Kích hoạt clock cho các chân GPIO

```
// Enable clock for GPIOC and GPIOB
RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOC, ENABLE);
RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOB, ENABLE);
```

2. Cấu hình các chân GPIO

- Chân PC13(led on board) để debug nếu cần , không cần thì có thể xóa bỏ (mode output push pull/ speed 50 MHz)

```
// LED DEBUG PC13 - OUTPUT
gpioInit.GPIO_Mode = GPIO_Mode_Out_PP;
gpioInit.GPIO_Pin = GPIO_Pin_13;
gpioInit.GPIO_Speed = GPIO_Speed_50MHz;
GPIO_Init(GPIOC, &gpioInit);
```

- Chân PB12 chân để đọc dữ liệu từ cảm biến DHT11
  (mode output open drain / speed 50MHz)

```
// DHT11 PB12 - OUTPUT
gpioInit.GPIO_Mode = GPIO_Mode_Out_OD;
gpioInit.GPIO_Pin = GPIO_Pin_12;
gpioInit.GPIO_Speed = GPIO_Speed_50MHz;
GPIO_Init(GPIOB, &gpioInit);
```

3. Bắt đầu quá trình khởi động và kiểm tra phản hồi từ cảm biến DHT11

- Đầu tiên là gửi tín hiệu bắt đầu , cảm biến DHT11 được kích hoạt bằng cách kéo chân dữ liệu xuống mức thấp trong ít nhất 18ms và sau đó kéo lên mức cao

```
// send start signal
GPIO_ResetBits(GPIOB, GPIO_Pin_12);
Delay_Ms(20);
GPIO_SetBits(GPIOB, GPIO_Pin_12);
```

- Chờ phản hồi từ cảm biến
  - Đặt bộ đếm của timer2 về 0 để bắt đầu đo thời gian.
  - Vòng lặp đầu tiên để check xem trong khoảng thời gian ngắn 10 xung clock của timer2 cảm biến DHT11 có phản hồi bằng cách kéo chân dữ liệu lên mức cao hay không.
  - Nếu mà cảm biến không phản hồi thì chương trình sẽ rơi vào vòng lặp vô hạn.

```
// wait for response
TIM_SetCounter(TIM2, 0);
while (TIM_GetCounter(TIM2) < 10) {
    if (GPIO_ReadInputDataBit(GPIOB, GPIO_Pin_12)) {
        break;
    }
}
// check if DHT11 response
u16Tim = TIM_GetCounter(TIM2);
if (u16Tim >= 10) {
    while (1) {
    }
}
```

- Tín hiệu thấp đầu tiên
    - Đặt bộ đếm của timer2 về 0 và chờ tín hiệu thấp
    - Vòng lặp chờ trong thời gian ngắn(45 xung clock của timer 2) để kiểm tra xem chân dữ liệu có được kéo xuống mức thấp hay không.
    - Kiểm tra xong thời gian phản hồi có nằm trong khoảng thời gian mong đợi không , nếu không nằm trong khoảng thời gian mong đợi thì sẽ rơi vào vòng lặp vô hạn

```
TIM_SetCounter(TIM2, 0);
while (TIM_GetCounter(TIM2) < 45) {
    if (!GPIO_ReadInputDataBit(GPIOB, GPIO_Pin_12)) {
        break;
    }
}
// check if DHT11 response
u16Tim = TIM_GetCounter(TIM2);
if ((u16Tim >= 45) || (u16Tim <= 5)) {
    while (1) {
    }
}
```

- Tín hiệu cao tiếp theo
    - Đặt bộ đếm của timer2 về 0 và chờ tín hiệu cao
    - Vòng lặp để chờ trong thời gian 90 xung của timer2 để xem DHT11 có kéo chân dữ liệu lên mức cao hay không.
    - Kiểm tra xong thời gian phản hồi có nằm trong khoảng thời gian mong đợi không , nếu không nằm trong khoảng thời gian mong đợi thì sẽ rơi vào vòng lặp vô hạn,

```
TIM_SetCounter(TIM2, 0);
while (TIM_GetCounter(TIM2) < 90) {
    if (GPIO_ReadInputDataBit(GPIOB, GPIO_Pin_12)) {
        break;
    }
}
// check if DHT11 response
u16Tim = TIM_GetCounter(TIM2);
if ((u16Tim >= 90) || (u16Tim <= 70)) {
    while (1) {
    }
}
```

- Tín hiệu thấp tiếp theo
    - Đặt bộ đếm của timer2 về 0 và chờ tín hiệu thấp
    - Vòng lặp này chờ trong khoảng thời gian ngắn (95 xung clock của Timer 2) để kiểm tra xem DHT11 có kéo chân dữ liệu xuống mức thấp hay không.
    - Nếu thời gian phản hồi không nằm trong khoảng mong đợi, chương trình sẽ vào vòng lặp vô hạn để báo hiệu lỗi.

```
TIM_SetCounter(TIM2, 0);
while (TIM_GetCounter(TIM2) < 95) {
    if (!GPIO_ReadInputDataBit(GPIOB, GPIO_Pin_12)) {
        break;
    }
}
u16Tim = TIM_GetCounter(TIM2);
if ((u16Tim >= 95) || (u16Tim <= 75)) {
    while (1) {
    }
}
```

4. Quá trình đọc cảm biến

- Sau khi cảm biến phản hồi xong thi sẽ bắt đầu gửi 40 bit dữ liệu theo thứ tự:
    1. 8 bit độ ẩm nguyên .
    2. 8 bit độ ẩm thập phân(luôn là 0);
    3. 8 bit nhiệt độ nguyên.
    4. 8 bit nhiệt độ thập phân.
    5. 8 bit checksum.

- Quá trình nhận 8bit đầu tiên
    - Vòng lặp này sẽ đọc 8 bit dữ liệu
    - Chờ chân PB12 được đặt lên mức cao. Đặt lại bộ đếm của Timer 2 về 0 và chờ cho chân PB12 lên cao. Nếu chân PB12 lên cao trước khi bộ đếm đạt đến 65, vòng lặp sẽ kết thúc sớm.
    - Lấy giá trị của bộ đếm Timer 2 và kiểm tra xem nó có nằm trong khoảng từ 45 đến 65 hay không. Nếu không, chương trình sẽ vào vòng lặp vô hạn để báo hiệu lỗi.
    - Đặt lại bộ đếm của Timer 2 về 0 và chờ cho chân PB12 xuống thấp. Nếu chân PB12 xuống thấp trước khi bộ đếm đạt đến 80, vòng lặp sẽ kết thúc sớm.

- Dịch trái giá trị trong u8Buff[0] để chuẩn bị lưu trữ bit mới. Nếu thời gian chân PB12 lên cao lớn hơn 45, bit nhận được là 1, ngược lại là 0.

```
// recive 8 bit -> 1 byte
for (int i = 0; i < 8; ++i) {
    /* cho chan PB12 len cao */
    TIM_SetCounter(TIM2, 0);
    while (TIM_GetCounter(TIM2) < 65) {
        // wait for PB12 to go high
        if (GPIO_ReadInputDataBit(GPIOB, GPIO_Pin_12)) {
            break;
        }
    }

    // check if time is out of range
    u16Tim = TIM_GetCounter(TIM2);
    if ((u16Tim >= 65) || (u16Tim <= 45)) {
        while (1) {
        }
    }

    /* cho chan PB12 xuong thap */
    TIM_SetCounter(TIM2, 0);
    while (TIM_GetCounter(TIM2) < 80) {
        // wait for PB12 to go low
        if (!GPIO_ReadInputDataBit(GPIOB, GPIO_Pin_12)) {
            break;
        }
    }
    // check if time is out of range
    u16Tim = TIM_GetCounter(TIM2);
    if ((u16Tim >= 80) || (u16Tim <= 10)) {
        // time out
        while (1) {
        }
    }
    // shift left 1 bit
    u8Buff[0] <<= 1;
    if (u16Tim > 45) {
        /* nhan duoc bit 1 */
        u8Buff[0] |= 1;
    } else {
        /* nhan duoc bit 0 */
        u8Buff[0] &= ~1;
    }
}
```

- Đọc các byte còn lại y hệt như byte đầu tiên
- Byte checksum

- Tính toán checksum bằng cách cộng các giá trị của 4 byte đầu tiên (u8Buff[0], u8Buff[1], u8Buff[2], u8Buff[3]). Checksum là một giá trị dùng để kiểm tra tính toàn vẹn của dữ liệu nhận được.
- So sánh giá trị checksum tính toán được (u8CheckSum) với giá trị checksum nhận được từ cảm biến (u8Buff[4]). Nếu hai giá trị này không khớp, chương trình sẽ vào vòng lặp vô hạn để báo hiệu rằng có lỗi xảy ra trong quá trình truyền dữ liệu.

```
u8CheckSum = u8Buff[0] + u8Buff[1] + u8Buff[2] + u8Buff[3];
if (u8CheckSum != u8Buff[4]) {
    while (1) {
    }
}
```

6. Debug bằng uart

```
//debug using uart1
USART1_Send_String("Temperature: ");
USART1_Send_Number(u8Buff[2]);
USART1_Send_String("*C\n");
USART1_Send_String("Humidity: ");
USART1_Send_Number(u8Buff[0]);
USART1_Send_String("%\n");
```