

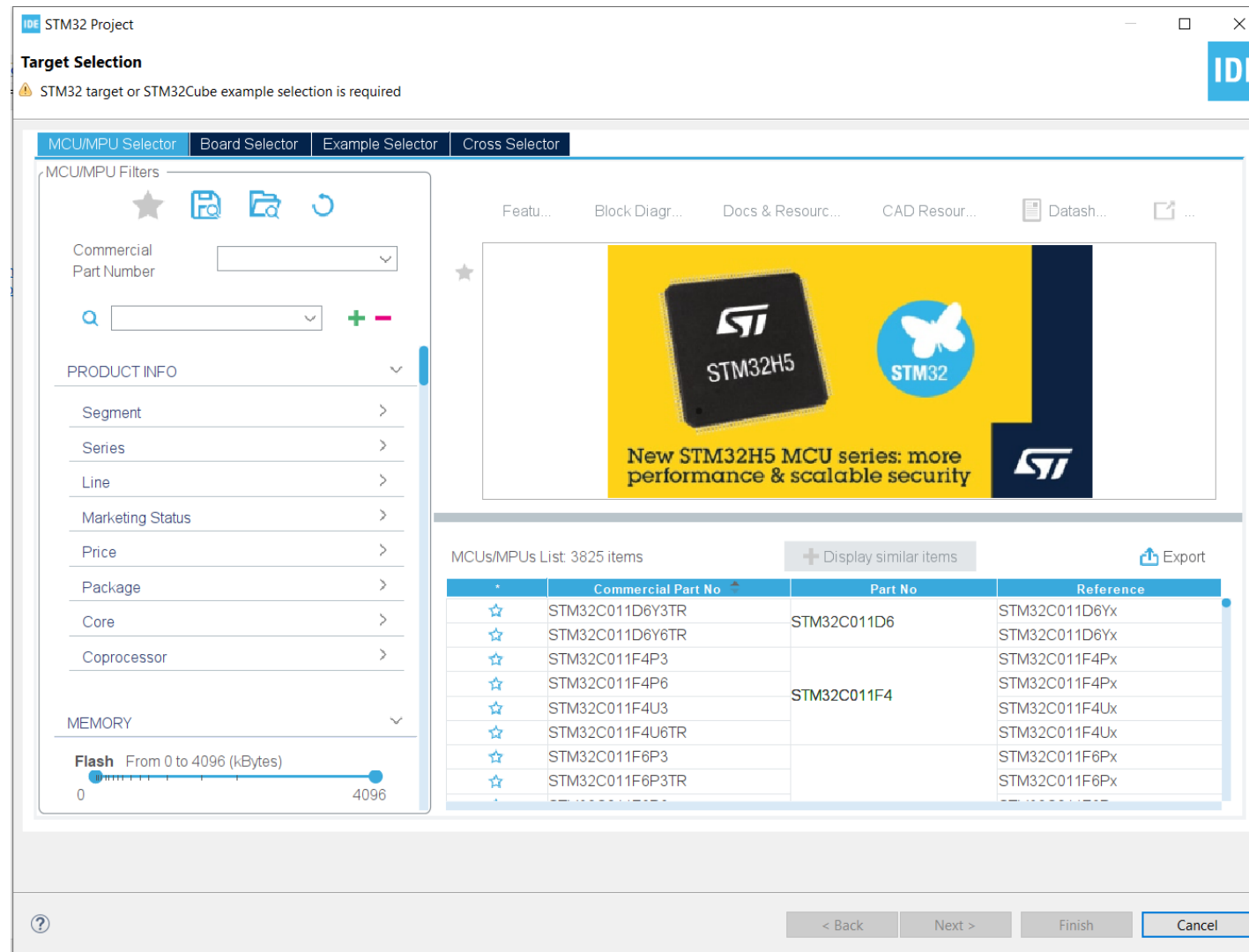
UCSD Embedded C Assignment 8

By

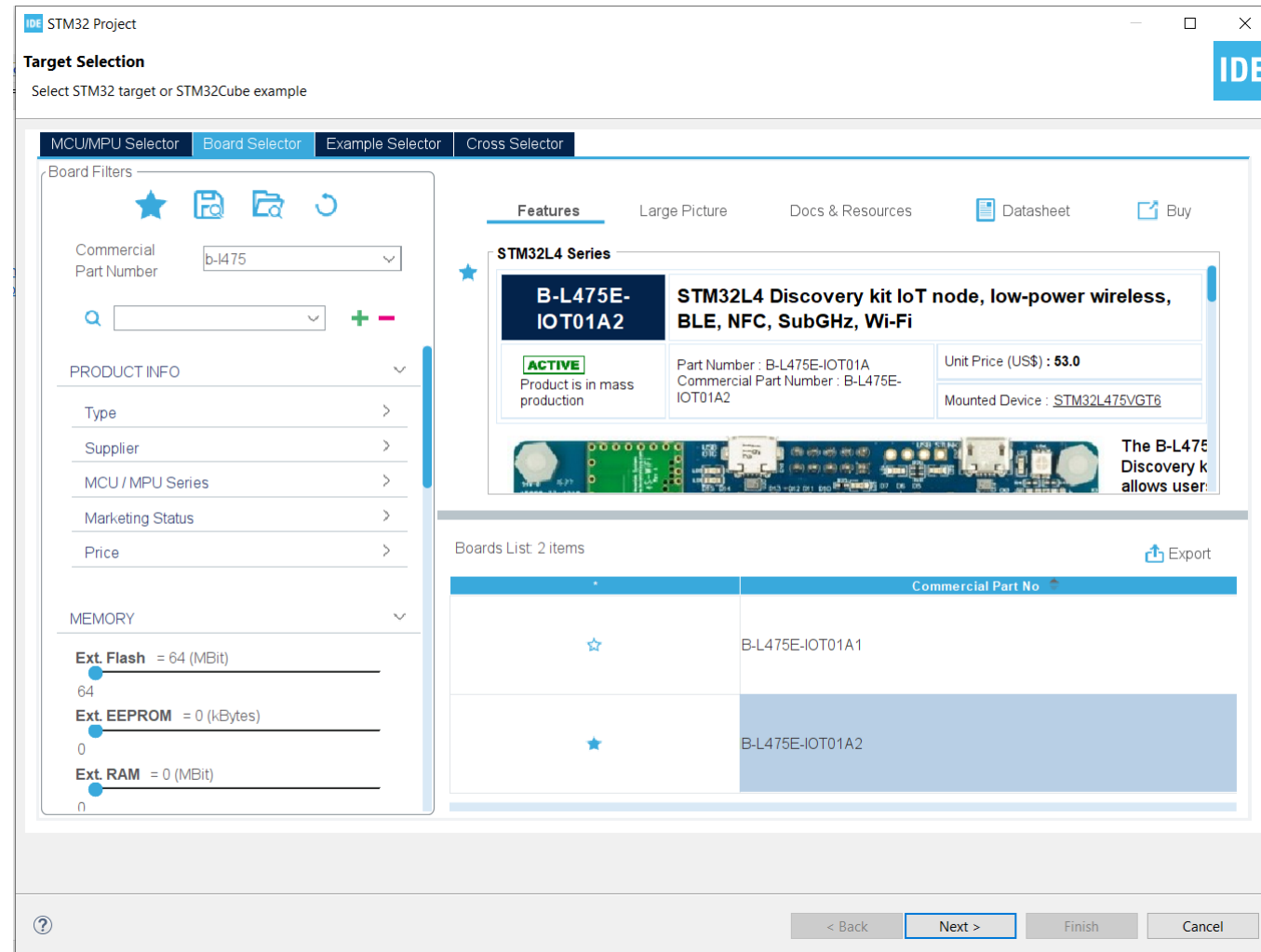
Hsuankai Chang

hsuankac@umich.edu

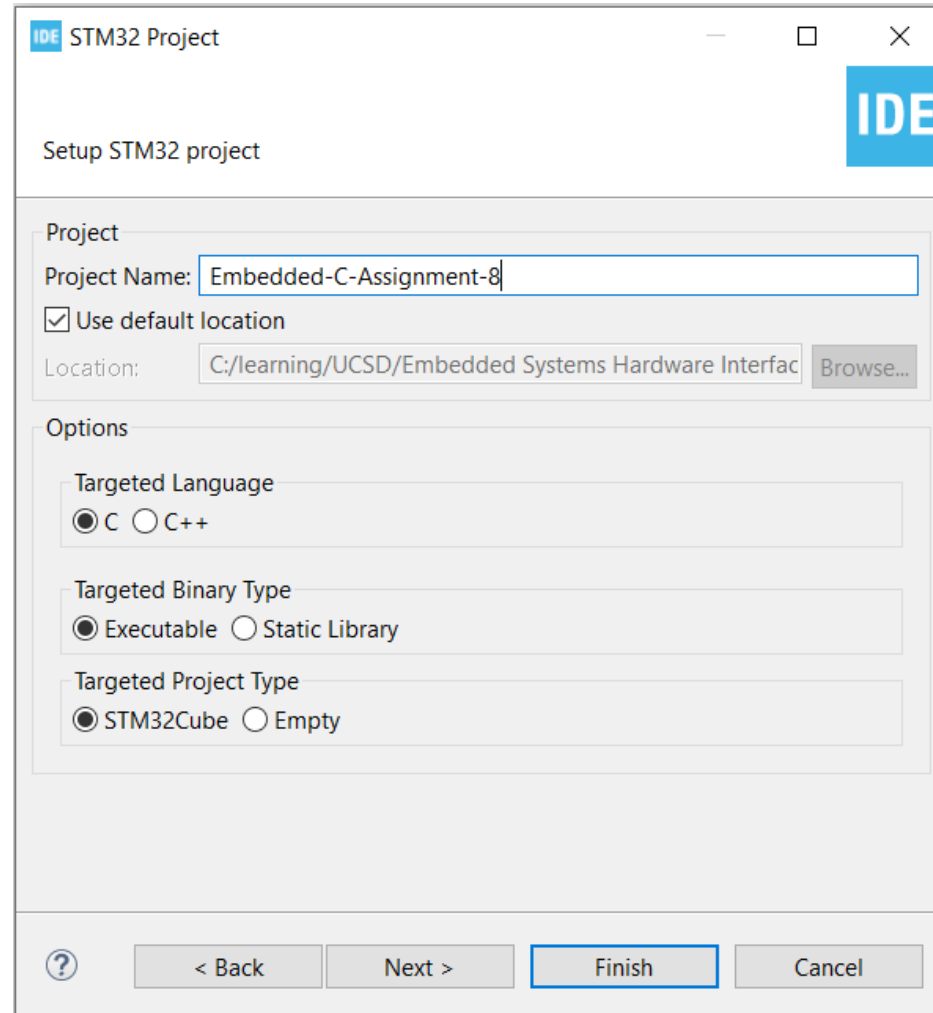
Step 1. Startup STM32CubeIDE and create new STM32 project



Step 2. Access board selector and type in the board you use, click Next



Step 3. Enter the project name then click Next



The image shows a 'Setup STM32 project' dialog box from an IDE. The window title is 'IDE STM32 Project'. The main heading is 'Setup STM32 project'. The 'Project' section contains a 'Project Name' field with the text 'Embedded-C-Assignment-8', a checked 'Use default location' checkbox, and a 'Location' field with the path 'C:/learning/UCSD/Embedded Systems Hardware Interfac' and a 'Browse...' button. The 'Options' section contains three groups of radio buttons: 'Targeted Language' with 'C' selected, 'Targeted Binary Type' with 'Executable' selected, and 'Targeted Project Type' with 'STM32Cube' selected. At the bottom, there is a help icon, and buttons for '< Back', 'Next >', 'Finish' (highlighted with a blue border), and 'Cancel'.

IDE STM32 Project

Setup STM32 project

Project

Project Name: Embedded-C-Assignment-8

☒ Use default location

Location: C:/learning/UCSD/Embedded Systems Hardware Interfac Browse...

Options

Targeted Language

☒ C ☐ C++

Targeted Binary Type

☒ Executable ☐ Static Library

Targeted Project Type

☒ STM32Cube ☐ Empty

? < Back Next > Finish Cancel

Step 4. See the firmware package name, version and location



The screenshot shows a dialog box titled "STM32 Project" with a subtitle "Firmware Library Package Setup". The main text says "Setup STM32 target's firmware". The dialog is divided into three sections: "Target and Firmware Package", "Firmware and Software Package Repository", and "Code Generator Options".

Target and Firmware Package

Target Reference: B-L475E-IOT01A2

Firmware Package Name and Version: STM32Cube FW_L4 **V1.17.2** ▼

Firmware and Software Package Repository

Location:
C:\Users\hsuankai.chang\STM32Cube\Repository

See ['Firmware Updater'](#) for settings related to package installation

Code Generator Options

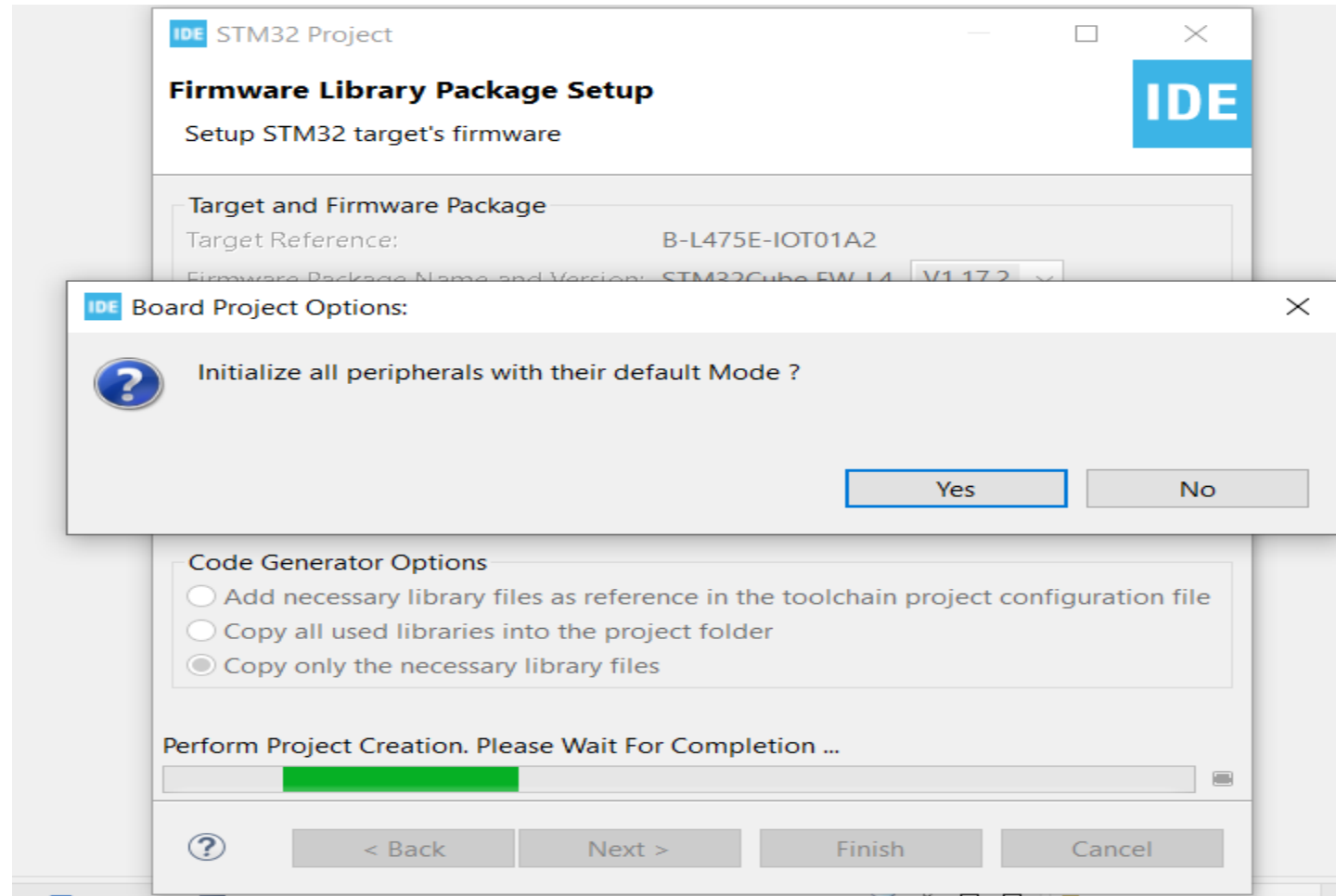
☐ Add necessary library files as reference in the toolchain project configuration file

☐ Copy all used libraries into the project folder

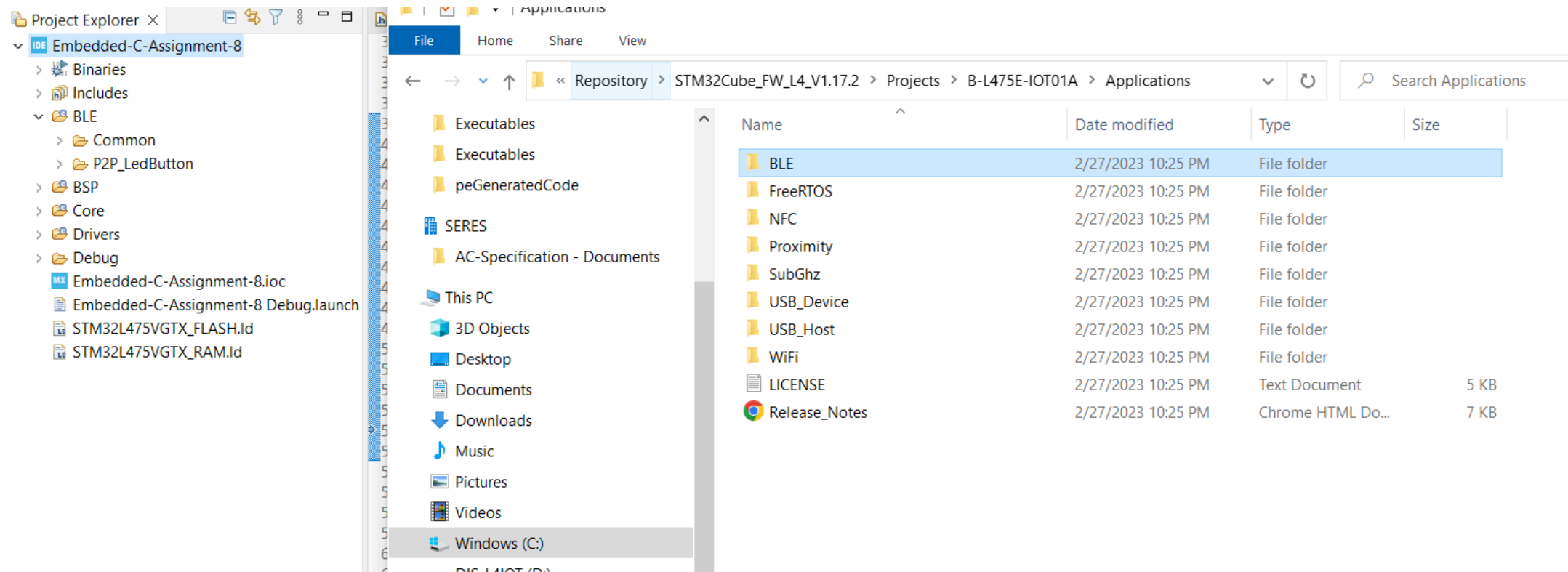
☒ Copy only the necessary library files

At the bottom, there are four buttons: a help button (question mark icon), "< Back", "Next >", and "Finish" (highlighted with a blue border). A "Cancel" button is also present.

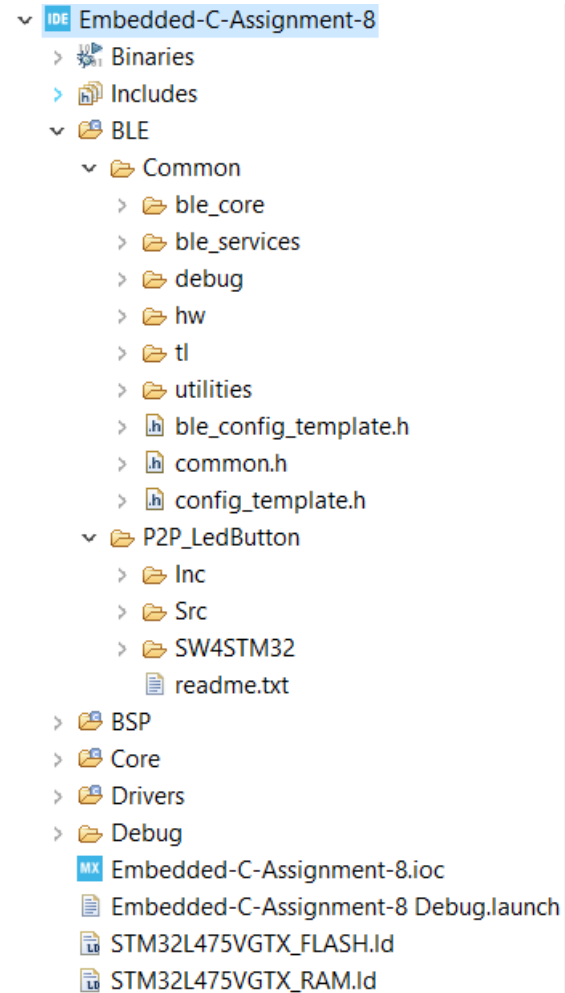
Step 5. Click yes to initialize all peripherals to default



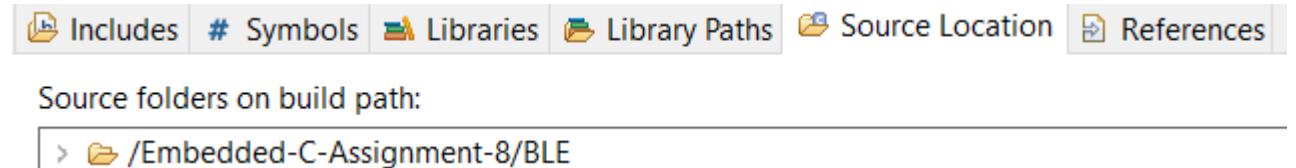
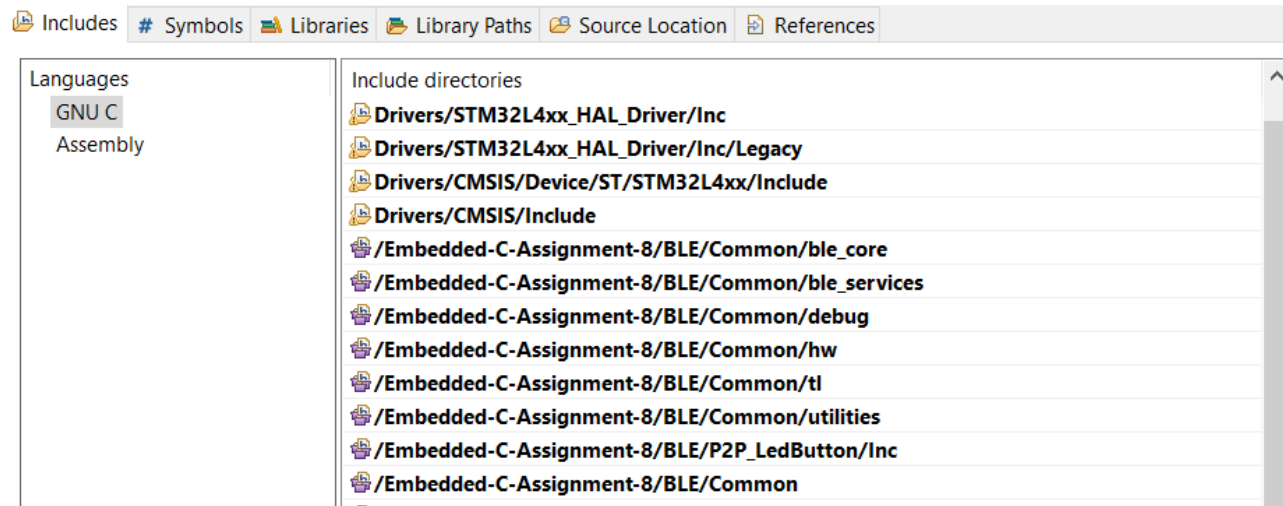
Step 6. Copy the BLE folder into the project, and delete the HeartRate folder



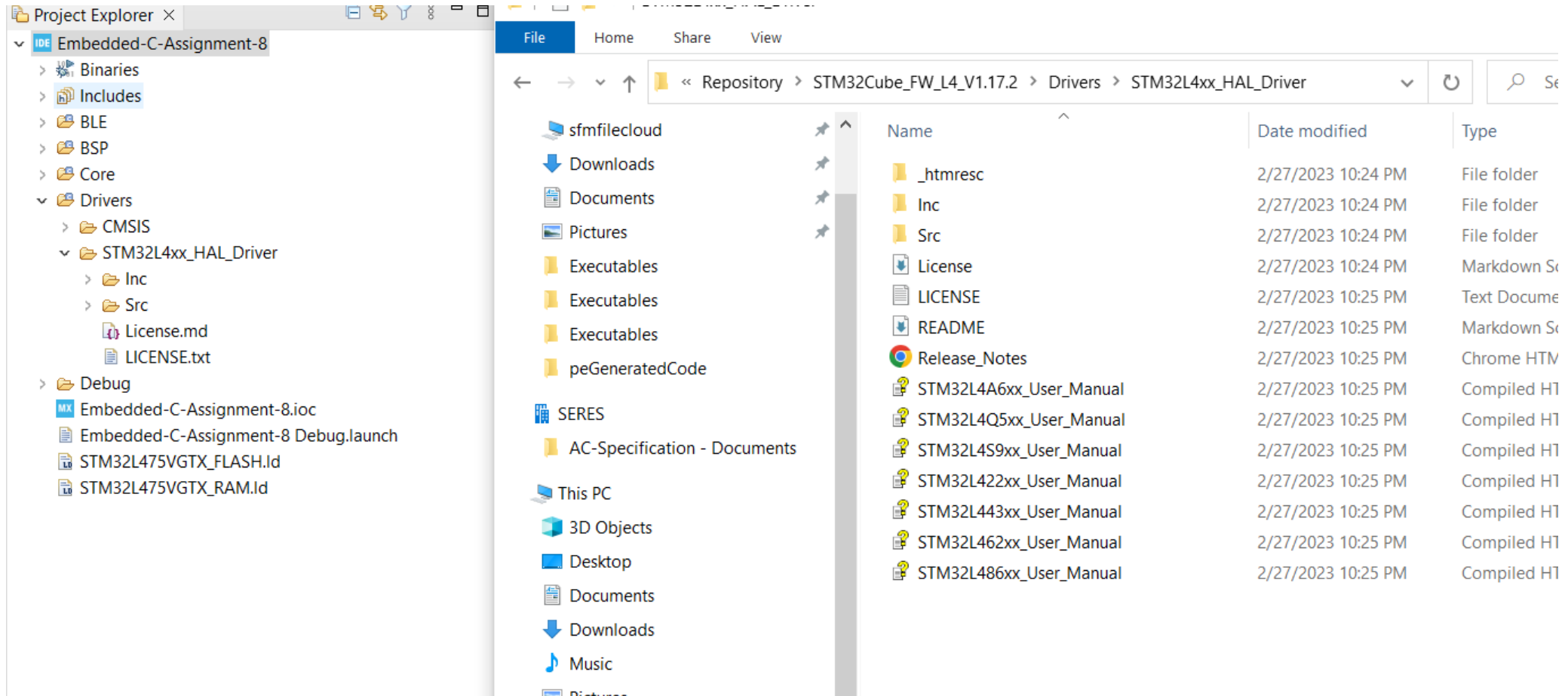
Step 7. Delete EWARM and MDK-ARM folder



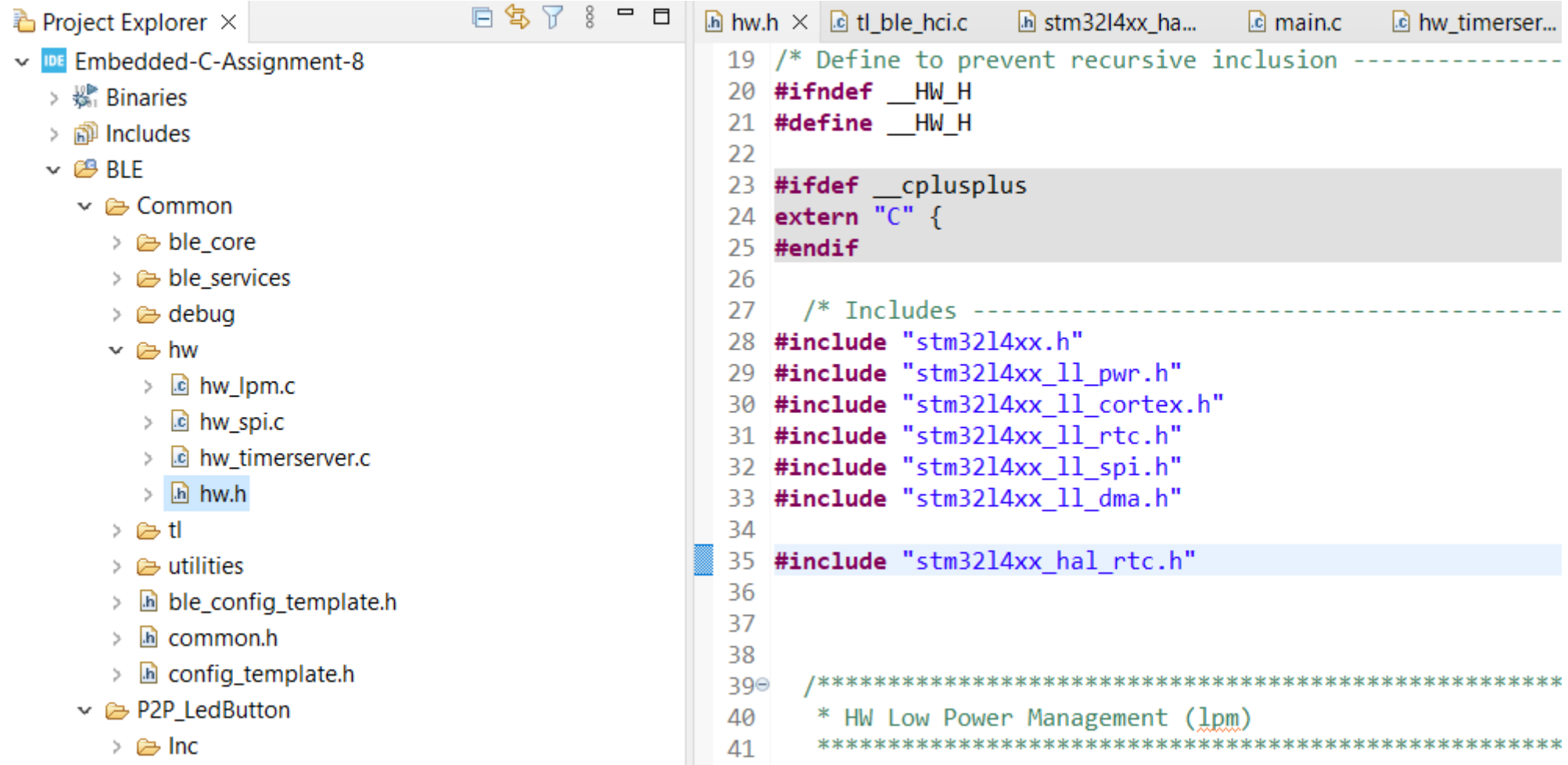
Step 8. Add the BLE source path and include path



Step 9. Copy the missing LL and HAL header and source file into the project, which includes stm32l4xx_ll_pwr.h, stm32l4xx_ll_rtc.h, stm32l4xx_ll_cortex.h, stm32l4xx_ll_spi.h, stm32l4xx_ll_dma.h, stm32l4xx_hal_rtc.h, stm32l4xx_hal_rtc_ex.h



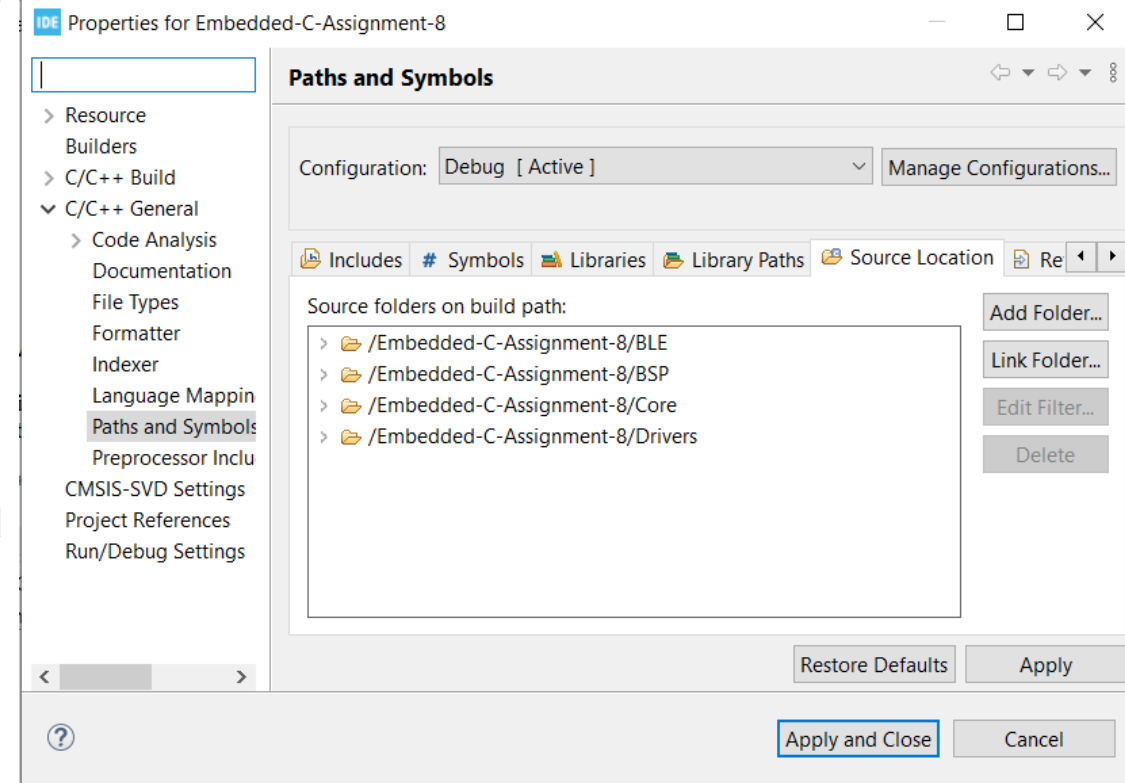
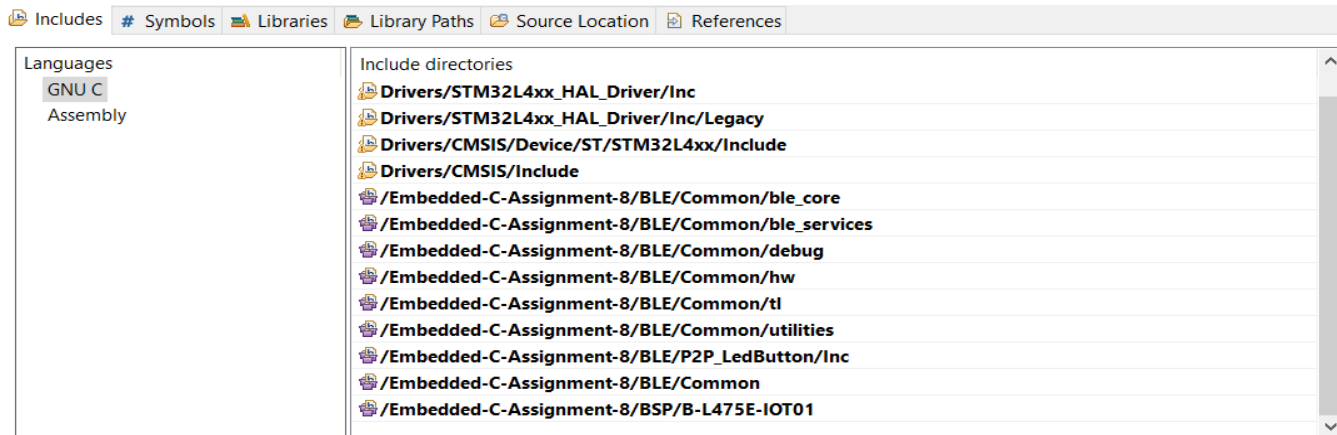
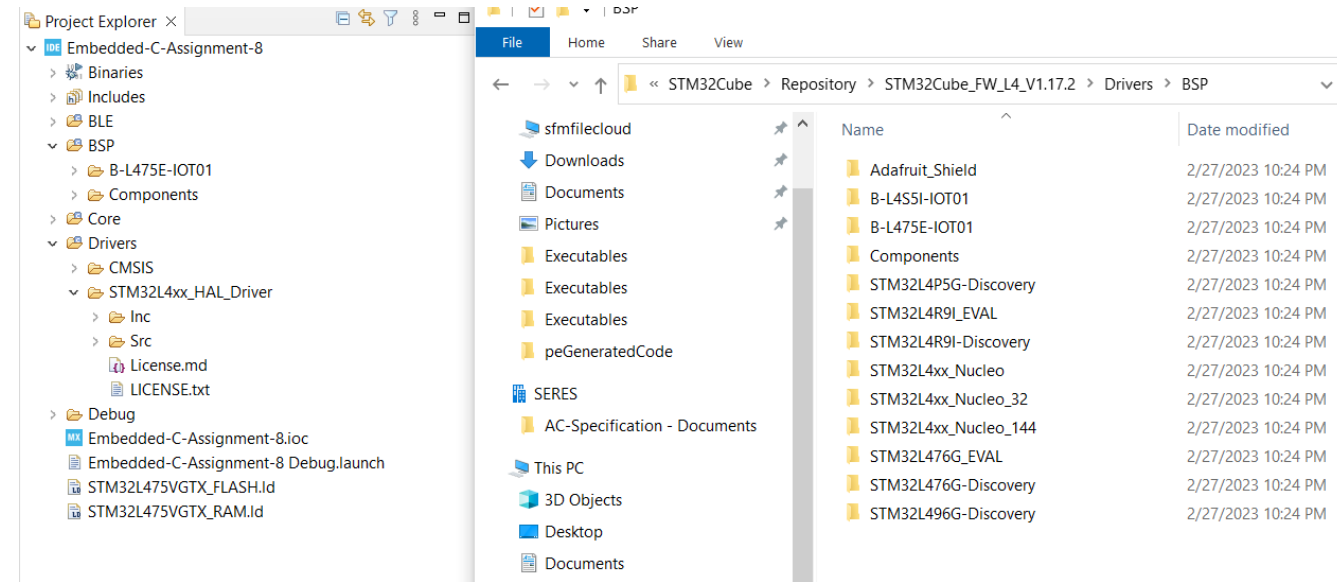
Step 10. Add stm32l4xx_hal_rtc.h in hw.h file



The screenshot shows an IDE interface with two main panels. The left panel is the Project Explorer, showing a project named 'Embedded-C-Assignment-8'. Under the 'BLE' folder, there is a 'Common' folder containing 'ble_core', 'ble_services', 'debug', and 'hw'. The 'hw' folder is expanded, showing files 'hw_lpm.c', 'hw_spi.c', 'hw_timerserver.c', and 'hw.h'. The 'hw.h' file is selected. The right panel shows the content of 'hw.h'. The file starts with a recursive inclusion guard for 'HW_H'. It then includes several STM32L4xx HAL and LL headers. Line 35, which is highlighted in blue, shows the addition of the header 'stm32l4xx_hal_rtc.h'. The file ends with a multi-line comment block for 'HW Low Power Management (lpm)'.

```
19 /* Define to prevent recursive inclusion -----
20 #ifndef __HW_H
21 #define __HW_H
22
23 #ifdef __cplusplus
24 extern "C" {
25 #endif
26
27 /* Includes -----
28 #include "stm32l4xx.h"
29 #include "stm32l4xx_ll_pwr.h"
30 #include "stm32l4xx_ll_cortex.h"
31 #include "stm32l4xx_ll_rtc.h"
32 #include "stm32l4xx_ll_spi.h"
33 #include "stm32l4xx_ll_dma.h"
34
35 #include "stm32l4xx_hal_rtc.h"
36
37
38
39- /* *****
40  * HW Low Power Management (lpm)
41  * *****
```

Step 11. Add BSP folders into the project, and also add the include path and source path



Step 12. Add BLUENRG_MS into preprocessor symbols, since our aci_gap_init function use 5 parameters

The image shows an IDE interface with three main components: a Project Explorer on the left, a code editor in the center, and a Properties/Settings window on the right.

Project Explorer: Shows a project named "Embedded-C-Assignment-8". The "BSP" folder is expanded, showing subfolders like "binaries", "includes", "ble", "common", "ble_services", "ble_svch", "dis.c", "dis.h", "hrs.c", "hrs.h", "lbs_stm.c", "lbs_stm.h", "svc_ctl.c", "svc_ctl.h", "svc_private.h", "debug", "hw", "hw_jpm.c", "hw_spi.c", "hw_timerserver.c", "hw.h", "utilities", "ble_config_template.h", "common.h", "config_template.h", "P2P_LedButton", "Inc", "Src", "SW4STM32", and "readme.txt".

Code Editor: Displays the file `hw.h`. The code defines a function `aci_gap_init` and includes preprocessor symbols. The relevant code is as follows:

```
28 #if BLUENRG_MS
29 //cond BLUENRG_MS
30 /**
31  * @brief Initialize the GAP layer.
32  * @note Register the GAP service with the GATT.
33  * All the standard GAP characteristics will also be added:
34  * @li Device Name
35  * @li Appearance
36  * @li Peripheral Preferred Connection Parameters (peripheral role only)
37  * @code
38  *
39  * tBleStatus ret;
40  * uint16_t service_handle, dev_name_char_handle, appearance_char_handle;
41  *
42  * ret = aci_gap_init(1, 0, 0x07, &service_handle, &dev_name_char_handle, &appearance_char_handle);
43  * if(ret){
44  *     PRINTF("GAP_Init failed.\n");
45  *     reboot();
46  * }
47  * const char *name = "BlueNRG";
48  * ret = aci_gatt_update_char_value(service_handle, dev_name_char_handle, 0, strlen(name), name);
49  * if(ret){
50  *     PRINTF("aci_gatt_update_char_value failed.\n");
51  * }
52  * }
53  * @endcode
54  * @param role Bitmap of allowed roles: see @ref gap_roles "GAP roles".
55  * @param privacy_enabled Enable (1) or disable (0) privacy.
56  * @param device_name_char_len Length of the device name characteristic
57  * @param[out] service_handle Handle of the GAP service.
58  * @param[out] dev_name_char_handle Device Name Characteristic handle
59  * @param[out] appearance_char_handle Appearance Characteristic handle
60  * @retval tBleStatus Value indicating success or error code.
61  */
62 tBleStatus aci_gap_init(uint8_t role, uint8_t privacy_enabled,
63                        uint8_t device_name_char_len,
64                        uint16_t* service_handle,
65                        uint16_t* dev_name_char_handle,
66                        uint16_t* appearance_char_handle);
67 //endcond
68
```

Properties for Embedded-C-Assignment-8: The "Settings" tab is selected. The "Preprocessor" section is expanded, showing the "Define symbols (-D)" list. The symbols defined are:

- DEBUG
- BLUENRG_MS
- USE_HAL_DRIVER
- STM32L475xx

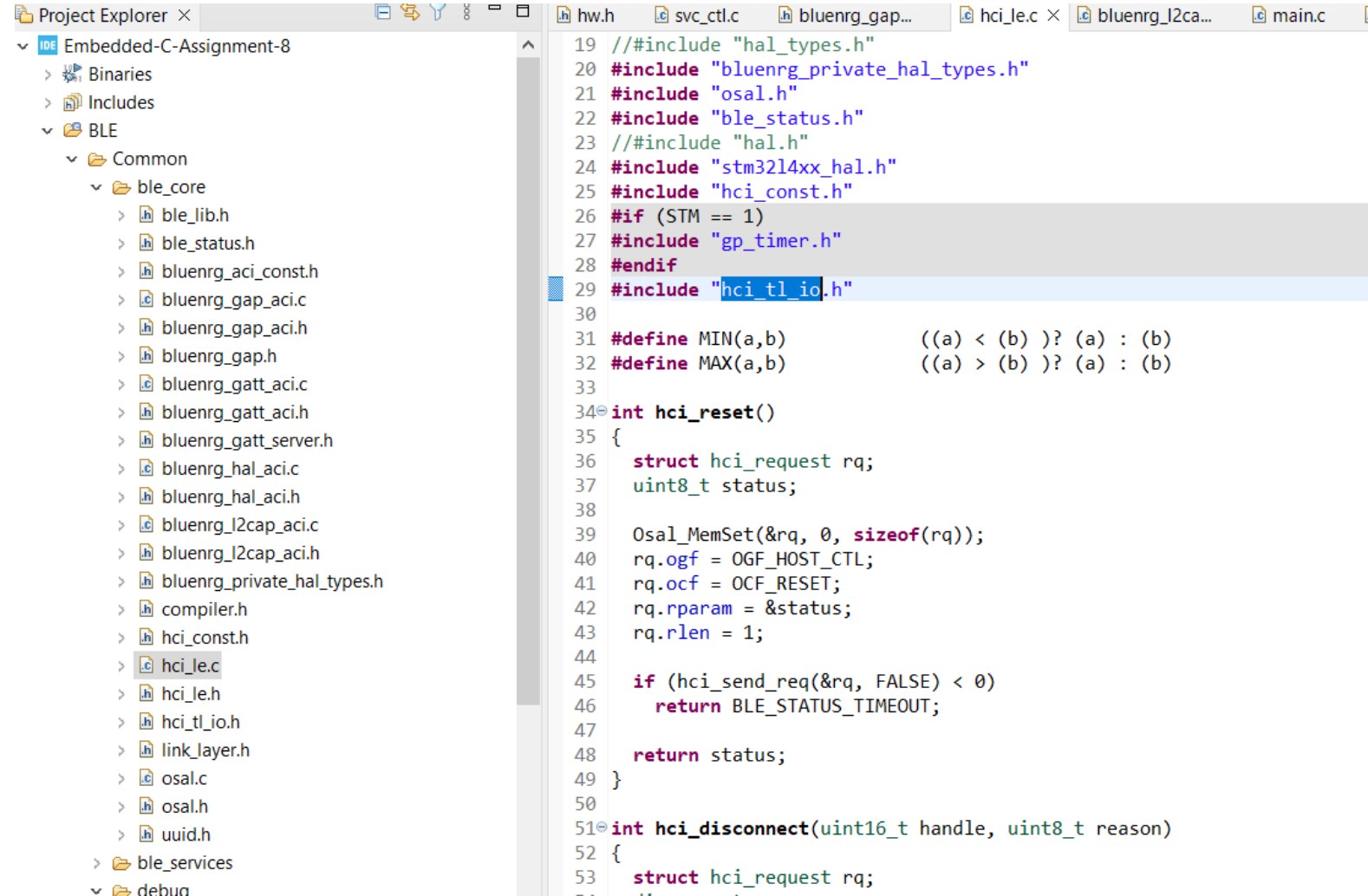
The "Undefined symbols (-U)" list is empty.

Step 13. Change the hal_types.h and hal.h include header files to bluenrg_private_hal_types.h and stm32l4xx_hal.h file in hci_ie.c and bluenrg_l2cap_aci.c file

```
10 * All rights reserved.
11 *
12 * This software is licensed under terms that can be fou
13 * in the root directory of this software component.
14 * If no LICENSE file comes with this software, it is pr
15 *
16 *****
17 */
18
19 //include "hal_types.h"
20 #include "bluenrg_private_hal_types.h"
21 #include "osal.h"
22 #include "ble_status.h"
23 //include "hal.h"
24 #include "stm32l4xx_hal.h"
25 #include "hci_const.h"
26 #if (STM == 1)
27 #include "gp_timer.h"
28 #endif
29 #include "hci_tl_io.h"
30
31 #define MIN(a,b) ((a) < (b)) ? (a) : (b)
32 #define MAX(a,b) ((a) > (b)) ? (a) : (b)
33
34 int hci_reset()
35 {
36     struct hci_request rq;
37     uint8_t status;
38
39     Osal_MemSet(&rq, 0, sizeof(rq));
40     rq.ogf = OGF_HOST_CTL;
41     rq.ocf = OCF_RESET;
42     rq.rparam = &status;
43     rq.rlen = 1;
44
45     if (hci_send_req(&rq, FALSE) < 0)
46 ..
```

```
1 ***** (C) COPYRIGHT 2013 STMMicroelectronics *****
2 * File Name      : bluenrg_hci.c
3 * Author         : AMS - HEA&RF BU
4 * Version        : V1.0.0
5 * Date          : 4-Oct-2013
6 * Description    : File with HCI commands for BlueNRG FW6.0 and above.
7 *****
8 * This software is licensed under terms that can be found in the LICENSE file
9 * in the root directory of this software component.
10 * If no LICENSE file comes with this software, it is provided AS-IS.
11 *****/
12
13 //include "hal_types.h"
14 #include "bluenrg_private_hal_types.h"
15 #include "osal.h"
16 #include "ble_status.h"
17 //include "hal.h"
18 #include "stm32l4xx_hal.h"
19 #include "osal.h"
20 #include "hci_const.h"
21 #include "bluenrg_aci_const.h"
22 #include "bluenrg_hal_aci.h"
23 #include "bluenrg_gap.h"
24
25 #define MIN(a,b) ((a) < (b)) ? (a) : (b)
26 #define MAX(a,b) ((a) > (b)) ? (a) : (b)
27
28 tBleStatus aci_l2cap_connection_parameter_update_request(uint16_t conn_handle, uint16_t interval_min,
29                                                         uint16_t interval_max, uint16_t slave_latency,
30                                                         uint16_t timeout_multiplier)
31 {
32     struct hci_request rq;
33     uint8_t status;
34 ..
```

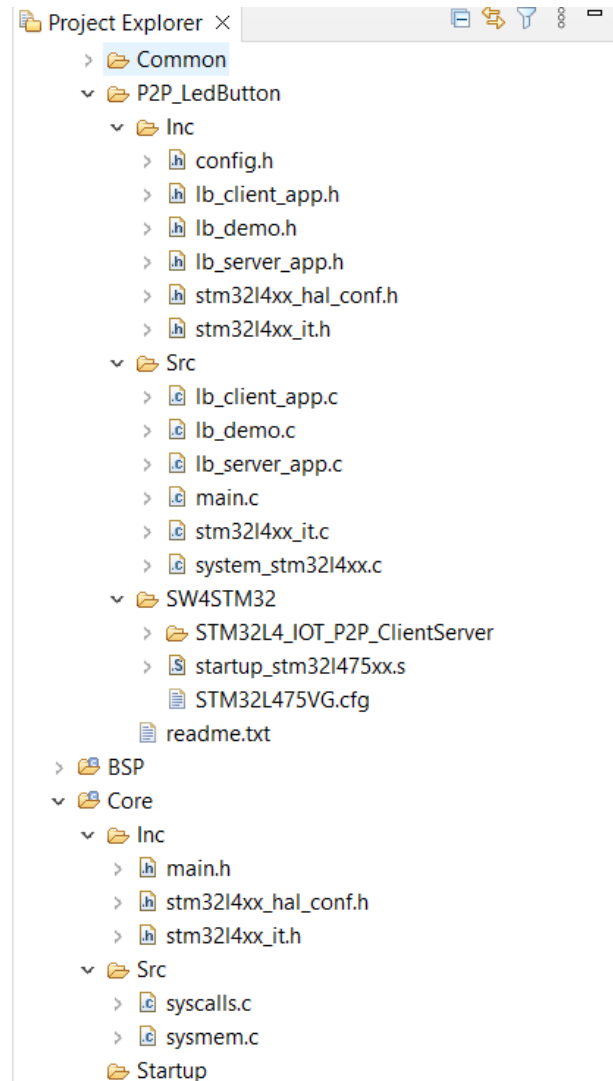
Step 14. Add hci_tl_io.h include file into hci_le.c file













The screenshot shows an IDE with the Project Explorer on the left and the hci_le.c file open in the editor on the right. The Project Explorer shows a project named 'Embedded-C-Assignment-8' with a folder structure including 'binaries', 'includes', 'BLE', and 'Common'. The 'Common' folder contains a 'ble_core' sub-folder with various header and source files. The 'hci_le.c' file is highlighted in the Project Explorer. The editor window shows the code for hci_le.c, with line 29 highlighted: `#include "hci_tl_io.h"`. The code includes several other headers and defines macros for MIN and MAX. It also contains functions for hci_reset and hci_disconnect.

```
19 //include "hal_types.h"
20 #include "bluenrg_private_hal_types.h"
21 #include "osal.h"
22 #include "ble_status.h"
23 //include "hal.h"
24 #include "stm32l4xx_hal.h"
25 #include "hci_const.h"
26 #if (STM == 1)
27 #include "gp_timer.h"
28 #endif
29 #include "hci_tl_io.h"
30
31 #define MIN(a,b) ((a) < (b)) ? (a) : (b)
32 #define MAX(a,b) ((a) > (b)) ? (a) : (b)
33
34 int hci_reset()
35 {
36     struct hci_request rq;
37     uint8_t status;
38
39     Osal_MemSet(&rq, 0, sizeof(rq));
40     rq.ogf = OGF_HOST_CTL;
41     rq.ocf = OCF_RESET;
42     rq.rparam = &status;
43     rq.rlen = 1;
44
45     if (hci_send_req(&rq, FALSE) < 0)
46         return BLE_STATUS_TIMEOUT;
47
48     return status;
49 }
50
51 int hci_disconnect(uint16_t handle, uint8_t reason)
52 {
53     struct hci_request rq;
```

Step 15. Delete main.c, stm32l4xx_it.c, system_stm32l4xx.c, stm32l4xx_hal_msp.c and also startup file in Core folder



Step 16. Delete ble_hci_ti_io_template.c file to solve multiple definition of hci_send_rq function

 ble_core	2/27/2023 10:25 PM
 ble_services	2/27/2023 10:25 PM
 debug	2/27/2023 10:25 PM
 hw	2/27/2023 10:25 PM
 tl	2/27/2023 10:25 PM
 utilities	2/27/2023 10:25 PM
 ble_config_template	2/27/2023 10:25 PM
 ble_hci_tl_io_template	2/27/2023 10:25 PM
 common	2/27/2023 10:25 PM
 config_template	2/27/2023 10:25 PM

Step 17. Add stm32l4xx_hal_rcc, rtc and rtc_ex source files in driver folder, and uncomment HAL_RTC_MODULE_ENABLED to use the feature

The screenshot displays an IDE interface with three main components: a Project Explorer on the left, a Project Explorer on the right, and a code editor on the right.

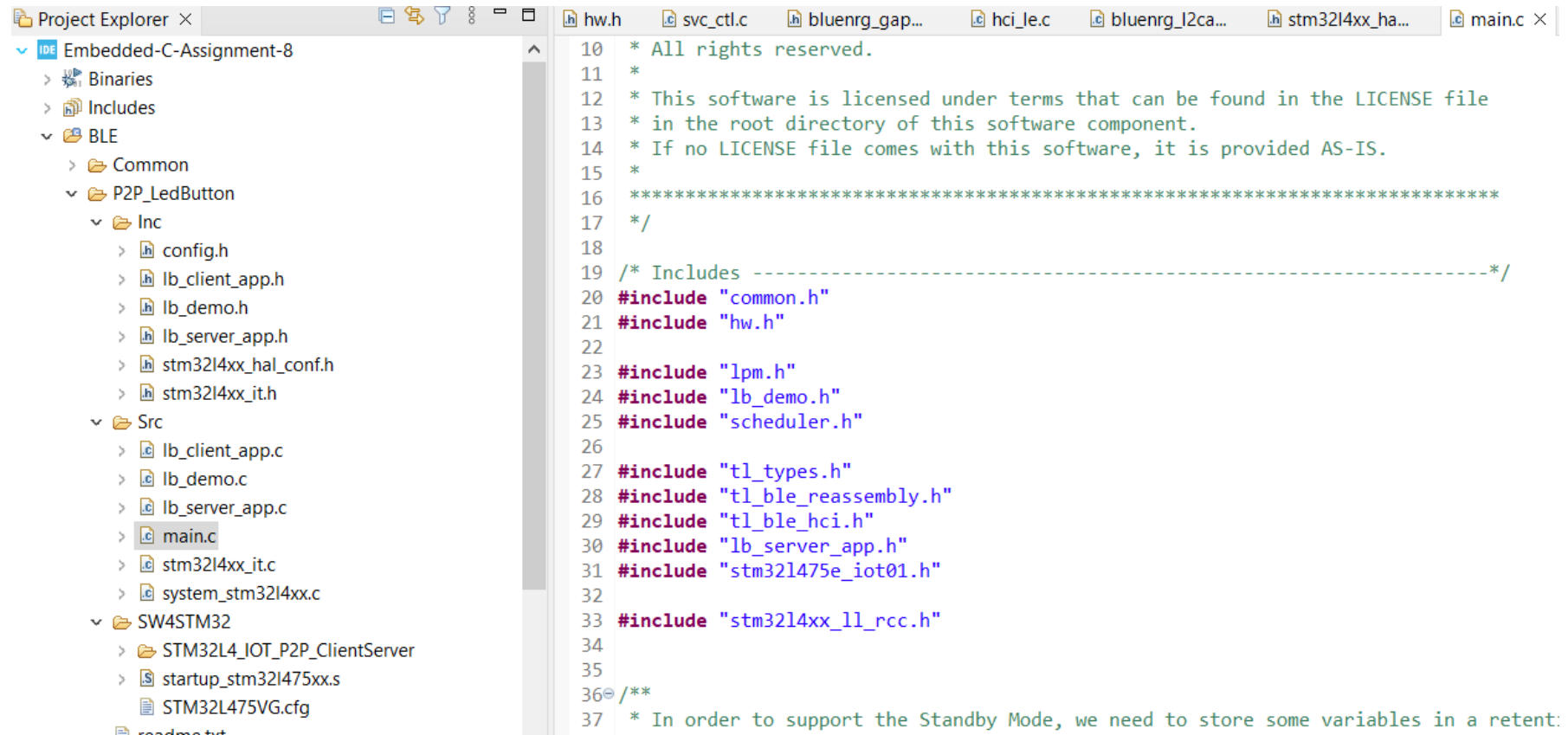
Left Project Explorer: Shows the 'Src' folder of a project. The files listed include various HAL modules for STM32L4xx. The files `stm32l4xx_hal_rcc.c`, `stm32l4xx_hal_rtc.c`, and `stm32l4xx_hal_rtc_ex.c` are highlighted in blue.

Right Project Explorer: Shows the 'Src' folder of a project named 'Embedded-C-Assignment-8'. The files listed include various HAL modules for STM32L4xx. The files `stm32l4xx_hal_rcc.c`, `stm32l4xx_hal_rtc.c`, and `stm32l4xx_hal_rtc_ex.c` are highlighted in blue.

Code Editor: Shows the content of `stm32l4xx_hal_rtc.c`. The file contains a series of `/*#define` and `#define` statements for enabling various HAL modules. The line `#define HAL_RTC_MODULE_ENABLED` is highlighted in blue, indicating it has been uncommented.

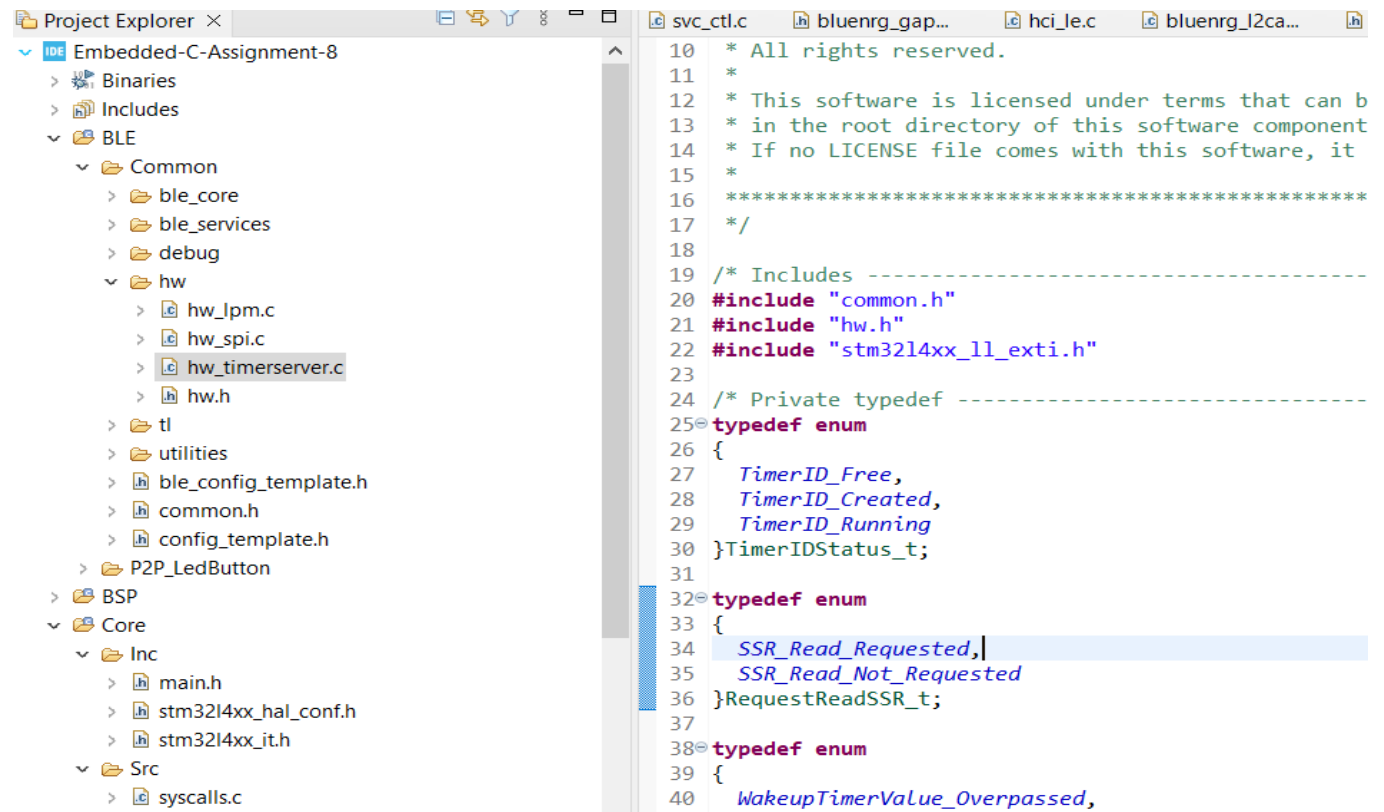
```
63 /*#define HAL_NOR_MODULE_ENABLED */
64 /*#define HAL_OPAMP_MODULE_ENABLED */
65 /*#define HAL_OSPI_MODULE_ENABLED */
66 /*#define HAL_OSPI_MODULE_ENABLED */
67 #define HAL_PCD_MODULE_ENABLED
68 /*#define HAL_PKA_MODULE_ENABLED */
69 /*#define HAL_QSPI_MODULE_ENABLED */
70 #define HAL_QSPI_MODULE_ENABLED
71 /*#define HAL_RNG_MODULE_ENABLED */
72 #define HAL_RTC_MODULE_ENABLED
73 /*#define HAL_SAI_MODULE_ENABLED */
74 /*#define HAL_SD_MODULE_ENABLED */
75 /*#define HAL_SMBUS_MODULE_ENABLED */
76 /*#define HAL_SMARTCARD_MODULE_ENABLED */
77 #define HAL_SPI_MODULE_ENABLED
78 /*#define HAL_SRAM_MODULE_ENABLED */
79 /*#define HAL_SWPMI_MODULE_ENABLED */
80 /*#define HAL_TIM_MODULE_ENABLED */
81 /*#define HAL_TSC_MODULE_ENABLED */
82 #define HAL_UART_MODULE_ENABLED
83 /*#define HAL_USART_MODULE_ENABLED */
84 /*#define HAL_WWDG_MODULE_ENABLED */
85 /*#define HAL_EXTI_MODULE_ENABLED */
86 /*#define HAL_PSSI_MODULE_ENABLED */
87 #define HAL_GPIO_MODULE_ENABLED
88 #define HAL_EXTI_MODULE_ENABLED
89 #define HAL_DMA_MODULE_ENABLED
90 #define HAL_RCC_MODULE_ENABLED
91 #define HAL_FLASH_MODULE_ENABLED
92 #define HAL_PWR_MODULE_ENABLED
93 #define HAL_CORTEX_MODULE_ENABLED
94
95 /* ##### Oscillator Values adaptation ##### */
96 /**
97  * @brief Adjust the value of External High Speed oscillator (HSE) used in your
98  *        This value is used by the RCC HAL module to compute the system frequen
99  *        (when HSE is used as system clock source, directly or through the PLL)
100  */
```

Step 18. Add stm32l4xx_ll_rcc.h file in main.c file under BLE folder to resolve LL_RCC errors

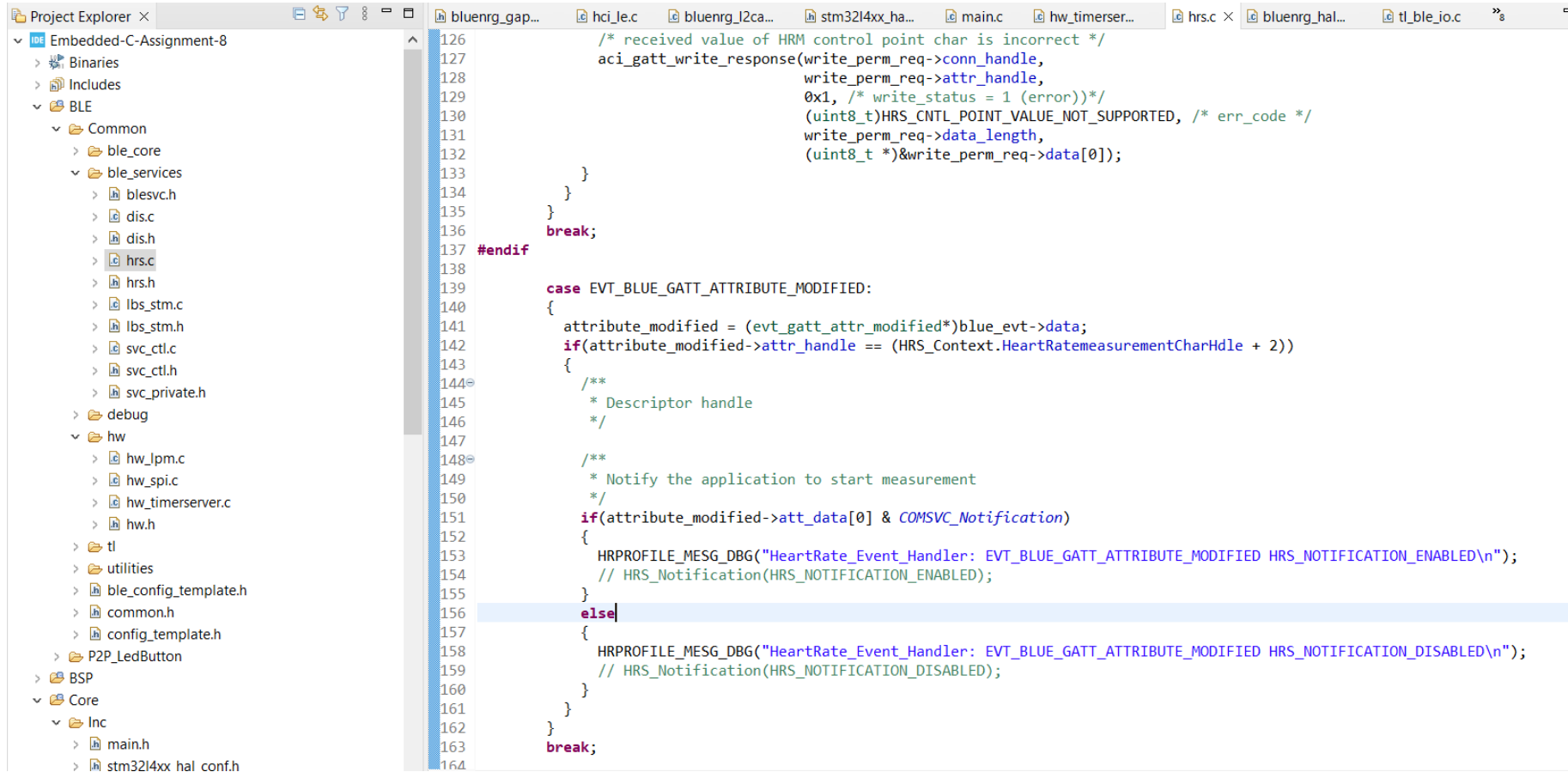


```
10  * All rights reserved.
11  *
12  * This software is licensed under terms that can be found in the LICENSE file
13  * in the root directory of this software component.
14  * If no LICENSE file comes with this software, it is provided AS-IS.
15  *
16  *****
17  */
18
19  /* Includes -----*/
20  #include "common.h"
21  #include "hw.h"
22
23  #include "lpm.h"
24  #include "lb_demo.h"
25  #include "scheduler.h"
26
27  #include "tl_types.h"
28  #include "tl_ble_reassembly.h"
29  #include "tl_ble_hci.h"
30  #include "lb_server_app.h"
31  #include "stm32l475e-iot01.h"
32
33  #include "stm32l4xx_ll_rcc.h"
34
35
36  /**
37  * In order to support the Standby Mode, we need to store some variables in a retent:
```

Step 19. Add stm32l4xx_ll_exit.h and .c file into driver inc and src folder, then add the header file into hw_timerserver.c file

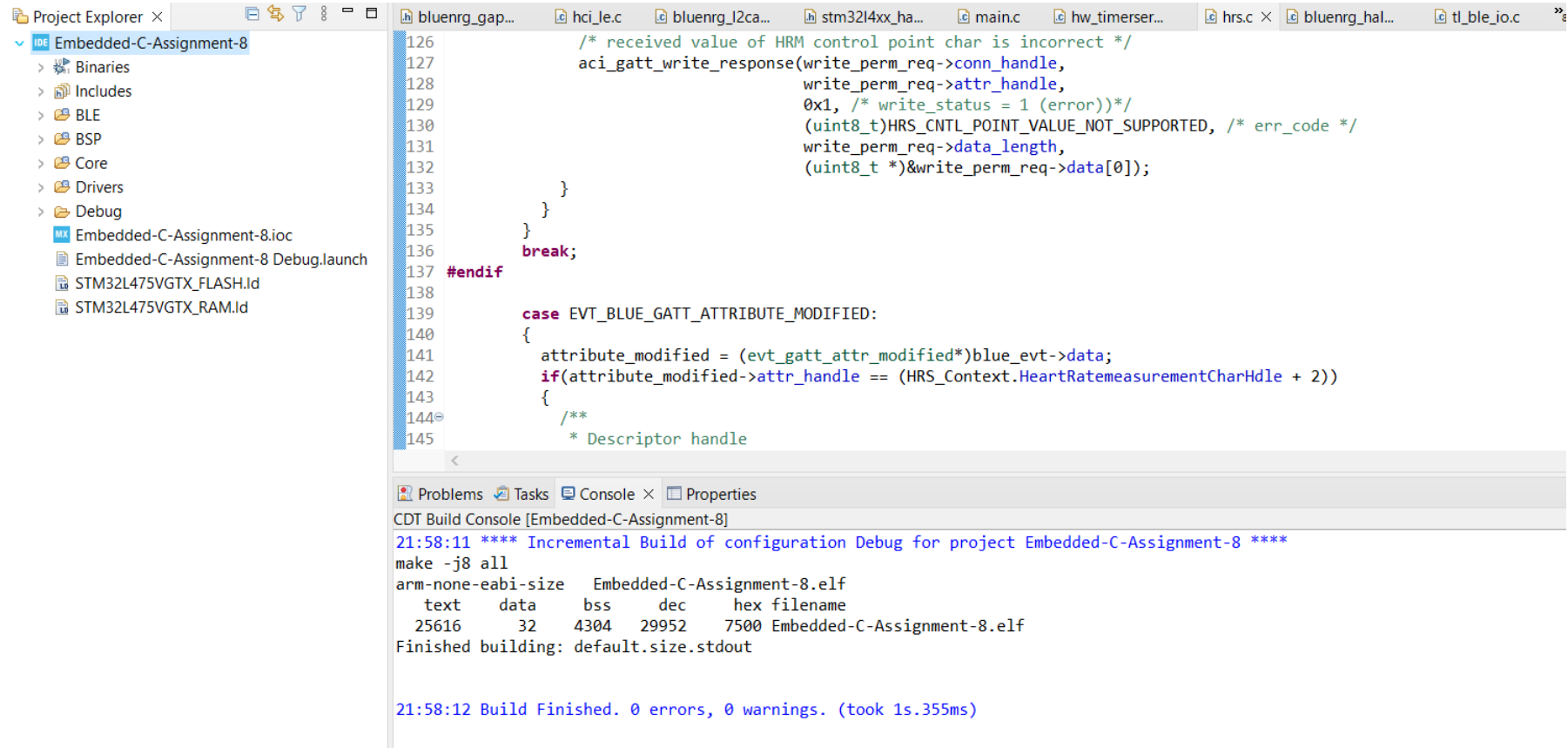


Step 20. Comment out all the HRS_Notification function in hrs.c file



```
126      /* received value of HRM control point char is incorrect */
127      aci_gatt_write_response(write_perm_req->conn_handle,
128                             write_perm_req->attr_handle,
129                             0x1, /* write_status = 1 (error) */
130                             (uint8_t)HRS_CNTL_POINT_VALUE_NOT_SUPPORTED, /* err_code */
131                             write_perm_req->data_length,
132                             (uint8_t *)&write_perm_req->data[0]);
133    }
134  }
135  }
136  break;
137 #endif
138
139 case EVT_BLUE_GATT_ATTRIBUTE_MODIFIED:
140 {
141   attribute_modified = (evt_gatt_attr_modified*)blue_evt->data;
142   if(attribute_modified->attr_handle == (HRS_Context.HeartRateMeasurementCharHdle + 2))
143   {
144     /**
145      * Descriptor handle
146      */
147
148     /**
149      * Notify the application to start measurement
150      */
151     if(attribute_modified->att_data[0] & COMSVC_Notification)
152     {
153       HRPROFILE_MESG_DBG("HeartRate_Event_Handler: EVT_BLUE_GATT_ATTRIBUTE_MODIFIED HRS_NOTIFICATION_ENABLED\n");
154       // HRS_Notification(HRS_NOTIFICATION_ENABLED);
155     }
156   else
157   {
158     HRPROFILE_MESG_DBG("HeartRate_Event_Handler: EVT_BLUE_GATT_ATTRIBUTE_MODIFIED HRS_NOTIFICATION_DISABLED\n");
159     // HRS_Notification(HRS_NOTIFICATION_DISABLED);
160   }
161 }
162 }
163 break;
164
```

Step 21. Finally, we can build the project successfully, and when run in debug mode, we can search the Bluetooth module of our stm board



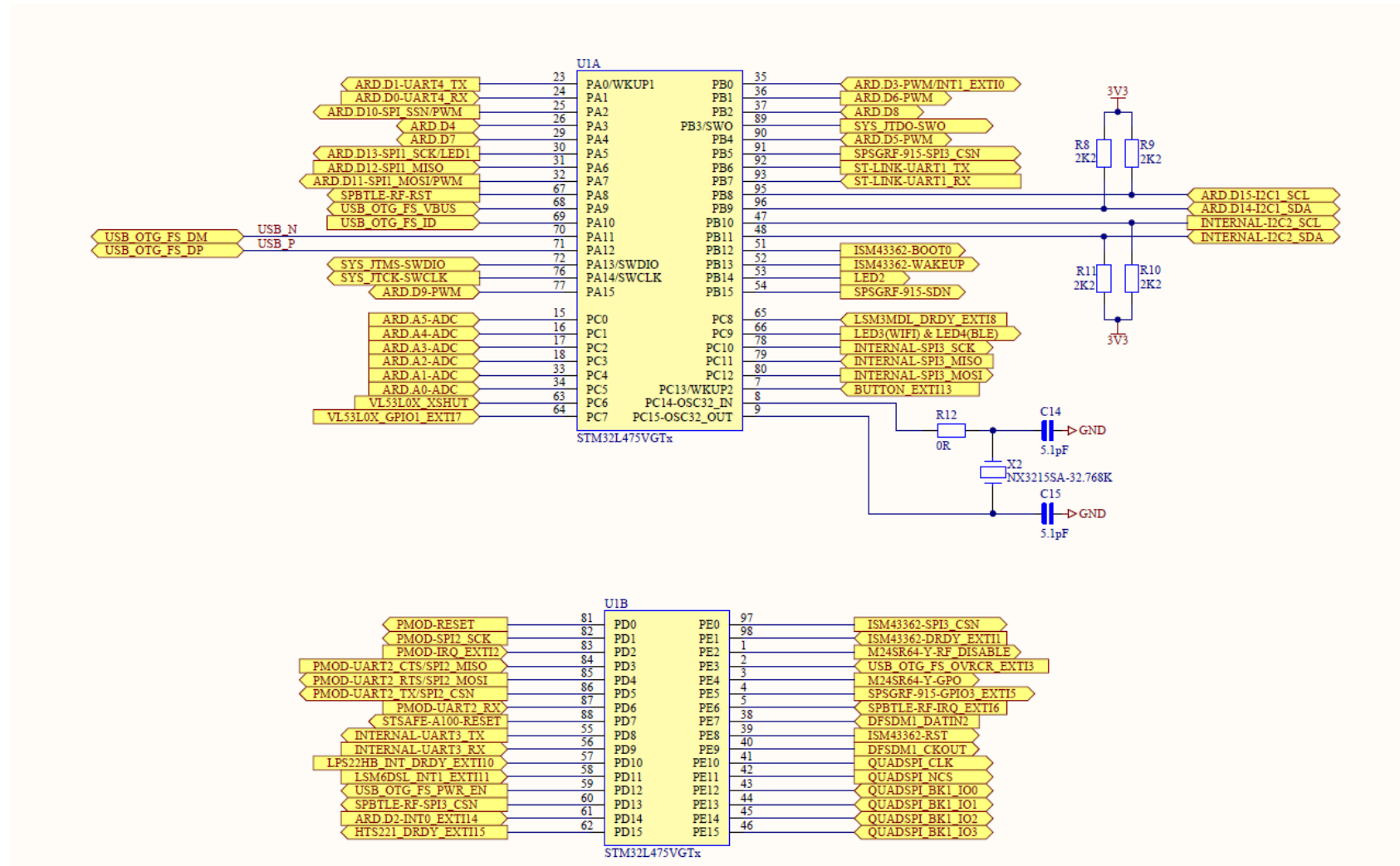
The screenshot displays an IDE interface with a Project Explorer on the left and a main editor window showing C code. The Project Explorer shows a project named 'Embedded-C-Assignment-8' with sub-items: Binaries, Includes, BLE, BSP, Core, Drivers, Debug, Embedded-C-Assignment-8.ioc, Embedded-C-Assignment-8 Debug.launch, STM32L475VGTX_FLASH.ld, and STM32L475VGTX_RAM.ld. The main editor window shows a C file with line numbers 126 to 145. The code includes comments and function calls related to Bluetooth GATT operations. The console at the bottom shows the build output for the project 'Embedded-C-Assignment-8', indicating a successful incremental build with 0 errors and 0 warnings.

```
126      /* received value of HRM control point char is incorrect */
127      aci_gatt_write_response(write_perm_req->conn_handle,
128                             write_perm_req->attr_handle,
129                             0x1, /* write_status = 1 (error) */
130                             (uint8_t)HRS_CNTL_POINT_VALUE_NOT_SUPPORTED, /* err_code */
131                             write_perm_req->data_length,
132                             (uint8_t *)&write_perm_req->data[0]);
133    }
134  }
135  }
136  break;
137 #endif
138
139 case EVT_BLUE_GATT_ATTRIBUTE_MODIFIED:
140 {
141     attribute_modified = (evt_gatt_attr_modified*)blue_evt->data;
142     if(attribute_modified->attr_handle == (HRS_Context.HeartRateMeasurementCharHdle + 2))
143     {
144         /**
145         * Descriptor handle
```

Problems Tasks Console Properties
CDT Build Console [Embedded-C-Assignment-8]
21:58:11 **** Incremental Build of configuration Debug for project Embedded-C-Assignment-8 ****
make -j8 all
arm-none-eabi-size Embedded-C-Assignment-8.elf
text data bss dec hex filename
25616 32 4304 29952 7500 Embedded-C-Assignment-8.elf
Finished building: default.size.stdout

21:58:12 Build Finished. 0 errors, 0 warnings. (took 1s.355ms)

Appendix, schematic for BLE module, processor side connection



Appendix, schematic for BLE module

