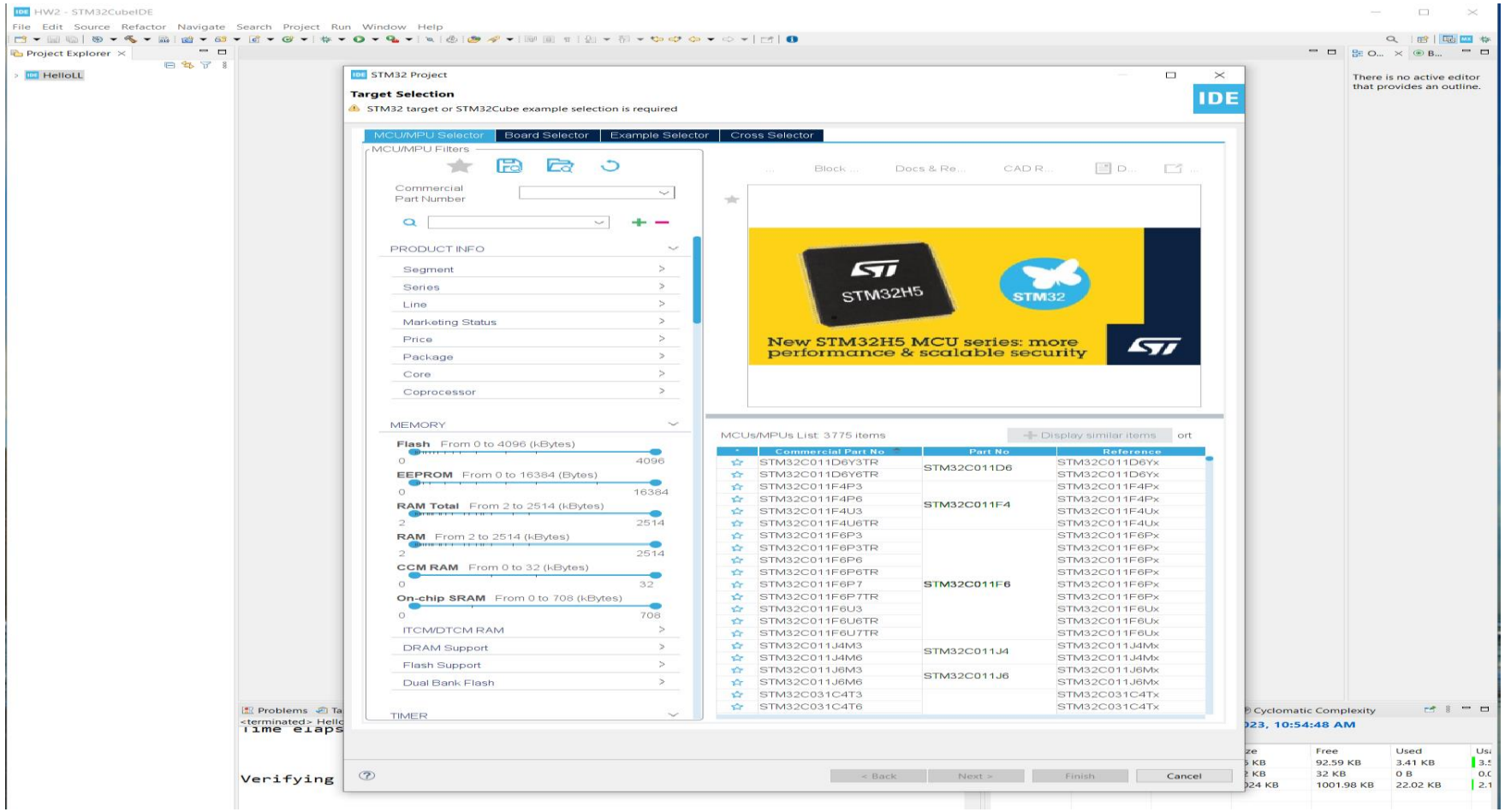
UCSD Embedded RTOS Assignment 4

By

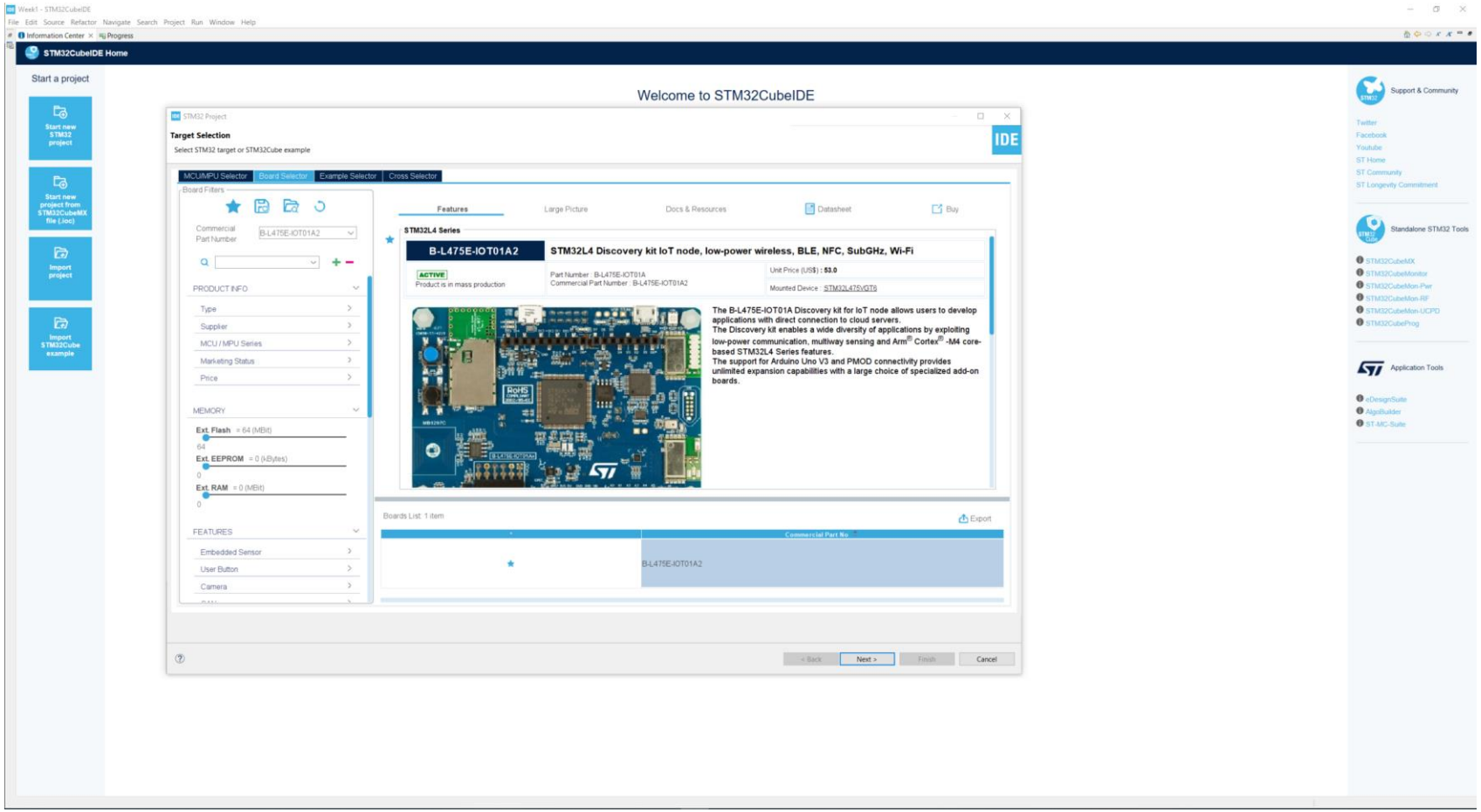
Hsuankai Chang

hsuankac@umich.edu

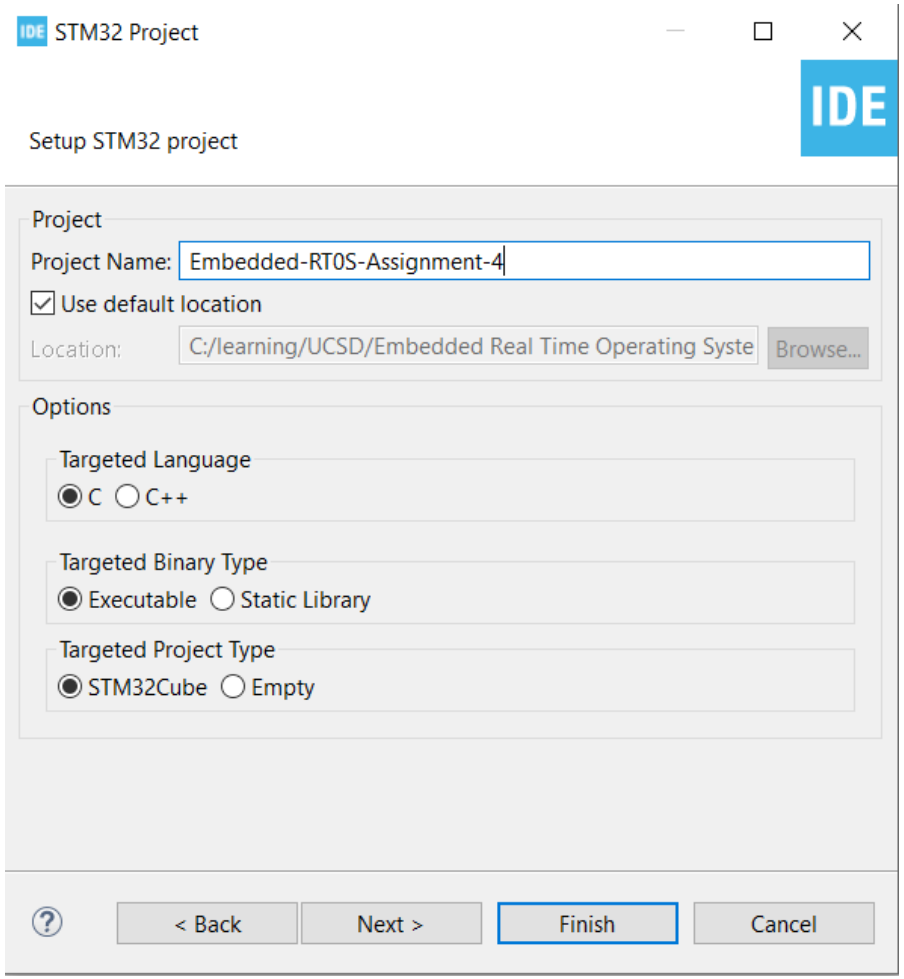
Step 1. Startup STM32CubeIDE and create new STM32 project



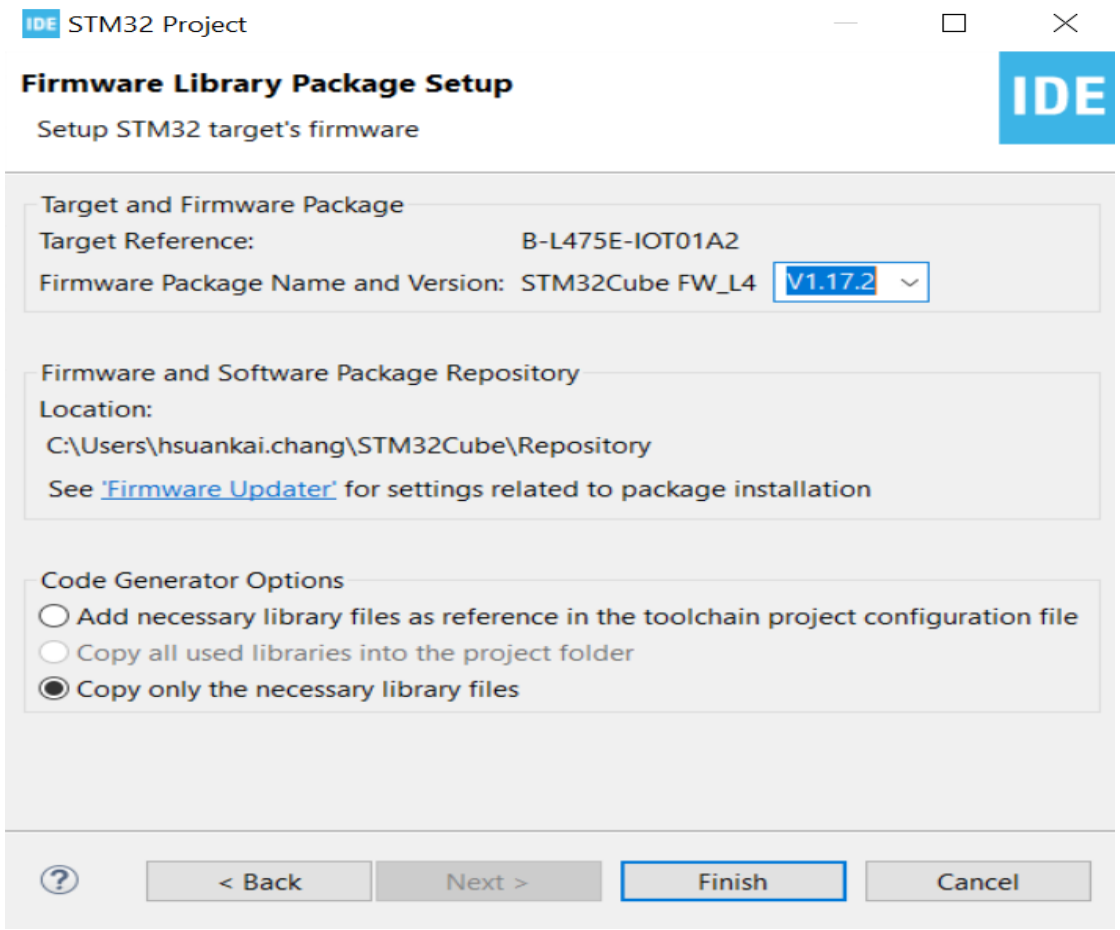
Step 2. Access board selector and type in the board you use, click Next



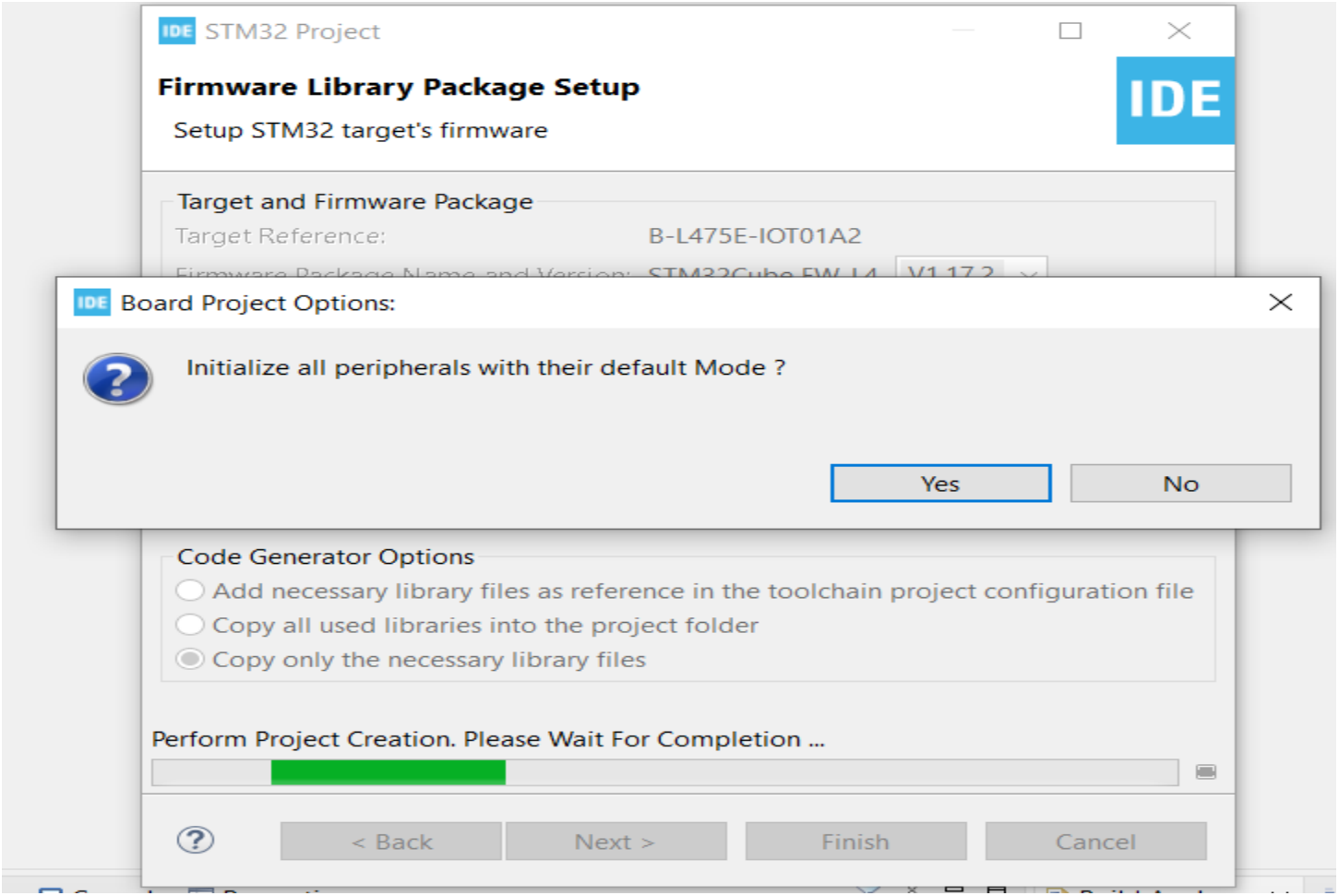
Step 3. Enter the project name then click Next



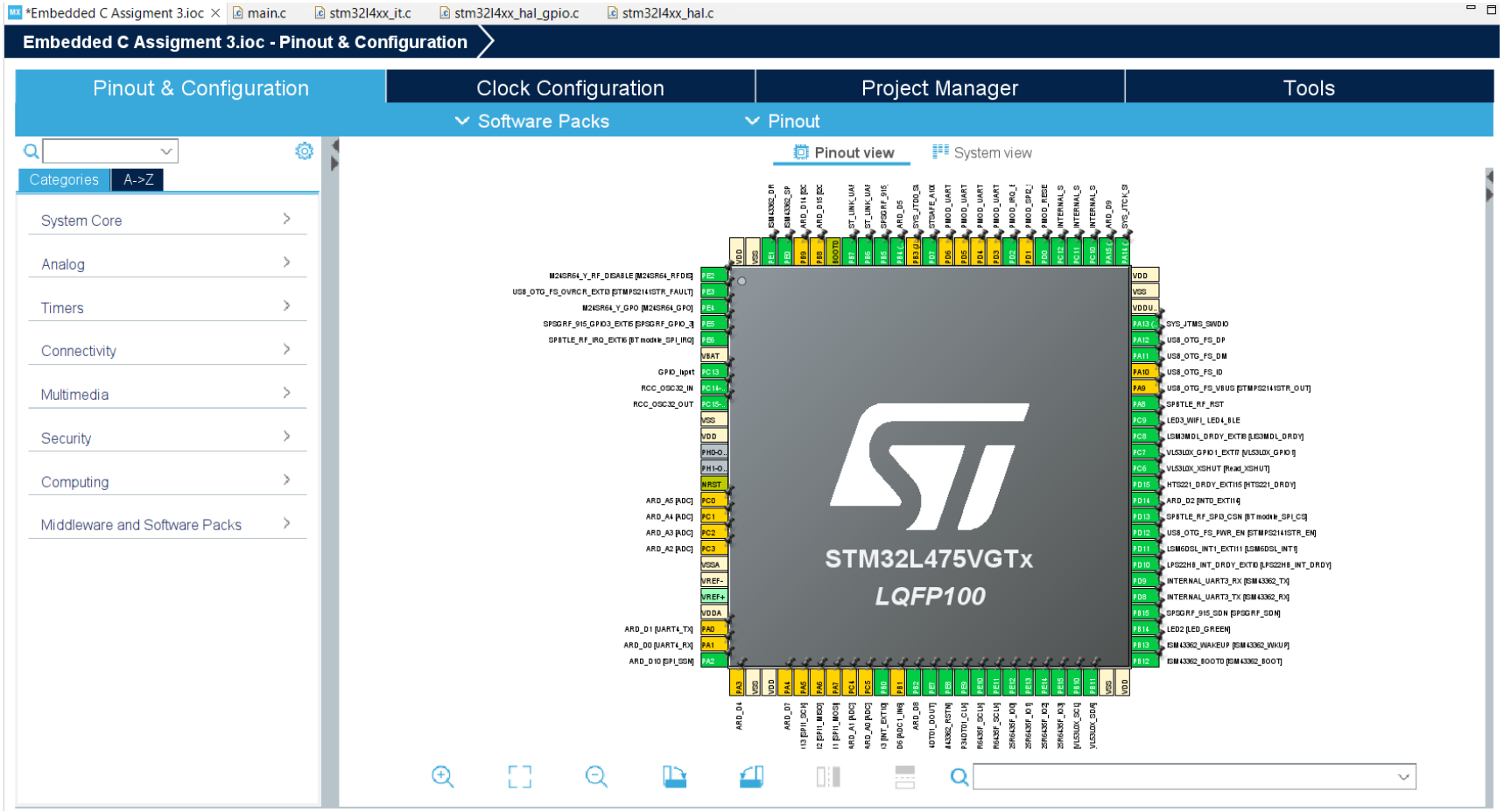
Step 4. See the firmware package name and version



Step 5. Click yes to initialize all peripherals to default



Step 6. When in .ioc file, click Pinout & Configurations



Step 7. Enable the CMSIS_V1 RTOS, and add one more tasks.

Embedded-RT0S-Assignment-4.ioc × main.c

Embedded-RT0S-Assignment-4.ioc - Pinout & Configuration

Pinout & Configuration | Clock Configuration | Project Manager

Software Packs | Pinout

Categories | A->Z

- Timers
- Connectivity
- Multimedia
- Security
- Computing
- Middleware and ...
  - AIROC-Wi-Fi-Blue
  - FATFS
  - FP-ATR-SIGFOX1
  - FP-SNS-SMARTA
  - FREERTOS
  - I-CUBE-Cesium
  - I-CUBE-ITTIADB
  - I-CUBE-embOS
  - I-CUBE-wolfSSL
  - I-Cube-SoM-uGOA
  - TOUCHSENSING
  - USB_DEVICE
  - USB_HOST
  - X-CUBE-AI
  - X-CUBE-ALGOBL
  - X-CUBE-ALS

FREERTOS Mode and Configuration

Configuration

Reset Configuration

Tasks and Queues | Timers and Semaphores | Mutexes | Events | FreeRTOS Heap Usage

Config parameters | Include parameters | Advanced settings | User Constants

Tasks

Task Name	Priority	Stack Size...	Entry Function	Code Generati...	Parameter	Allocation	Buffer Name	Control Block N...
defaultTask	osPriorityNormal	128	StartDefaultTask	Default	NULL	Dynamic	NULL	NULL
myTask02	osPriorityLow	128	StartTask02	Default	NULL	Dynamic	NULL	NULL

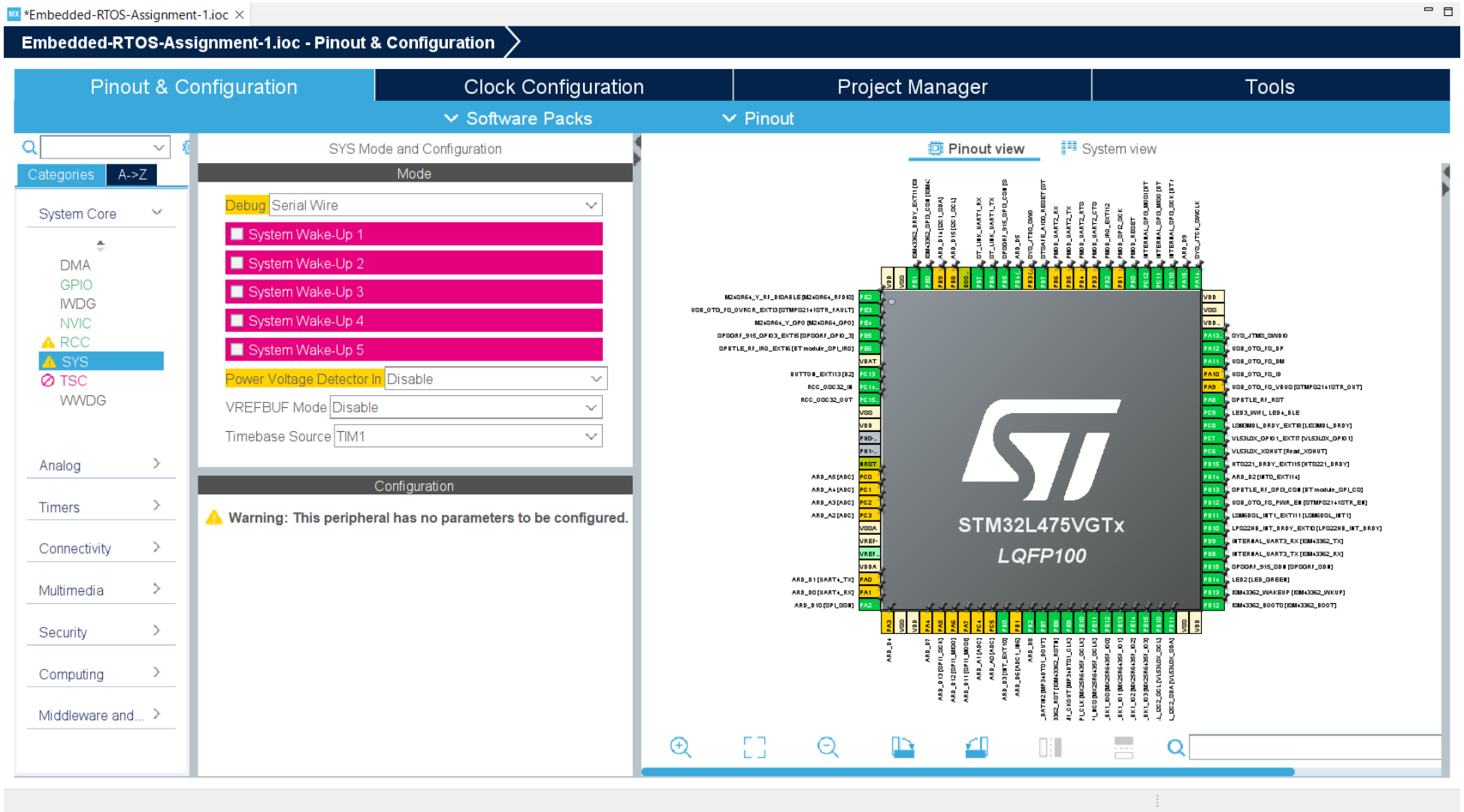
Add | Delete

Queues

Queue Name	Queue Size	Item Size	Allocation	Buffer Name	Control Block Name

Add | Delete

Step 8. Change Timebase from systick to TIM1



Step 9. Enable the software timer in FreeRTOS settings and add two more timers. One one-shot timer and one auto reload timer

Tasks and Queues | Timers and Semaphores | Mutexes | Events | FreeRTOS Heap Usage

Config parameters | Include parameters | Advanced settings | User Constants

Configure the below parameters :

Search (Ctrl+F)

USE_DAEMON_TASK_STARTUP_HOOK	Disabled
CHECK_FOR_STACK_OVERFLOW	Disabled
Run time and task stats gathering related definitions	
GENERATE_RUN_TIME_STATS	Disabled
USE_TRACE_FACILITY	Disabled
USE_STATS_FORMATTING_FUNCTIONS	Disabled
Co-routine related definitions	
USE_CO_ROUTINES	Disabled
MAX_CO_ROUTINE_PRIORITIES	2
Software timer definitions	
USE_TIMERS	Enabled
TIMER_TASK_PRIORITY	2
TIMER_QUEUE_LENGTH	10
TIMER_TASK_STACK_DEPTH	256 Words
Interrupt nesting behaviour configuration	
LIBRARY_LOWEST_INTERRUPT_PRIORITY	15
LIBRARY_MAX_SYSCALL_INTERRUPT_PRIORITY	5
Added with 10.2.1 support	
MESSAGE_BUFFER_LENGTH_TYPE	size_t
USE_POSIX_ERRNO	Disabled

Embedded-RT0S-Assignment-4.ioc | main.c

Embedded-RT0S-Assignment-4.ioc - Pinout & Configuration

Pinout & Configuration | Clock Configuration | Project Manager

Software Packs | Pinout

Categories | A->Z

- Timers
- Connectivity
- Multimedia
- Security
- Computing
- Middleware and ...

FREERTOS Mode and Configuration

Configuration

Reset Configuration

Tasks and Queues | Timers and Semaphores | Mutexes | Events | FreeRTOS Heap Usage

Config parameters | Include parameters | Advanced settings | User Constants

Timers

Timer Name	Callback	Type	Code Generation O...	Parameter	Allocation	Control Block Name
myTimer01	prvMyTimerOneShot	osTimerOnce	Default	NULL	Dynamic	NULL
myTimer02	prvMyTimerAutoRel...	osTimerPeriodic	Default	NULL	Dynamic	NULL

Add | Delete

Step 10. Create "Task 1" that blinks the LED2 at a rate of 1 second. Create "Task 2" that blinks the "Wifi/BLE" LED at a rate of 2 seconds

Embedded-RTOS-Assignment-4.ioc × main.c ×

```c
void StartDefaultTask(void const * argument)
{
  /* USER CODE BEGIN 5 */
  /* Infinite loop */
  for(;;)
  {
    HAL_GPIO_TogglePin(LED2_GPIO_Port, LED2_Pin);
    osDelay(1000);
  }
  /* USER CODE END 5 */
}

/* USER CODE BEGIN Header_StartTask02 */
/**
* @brief Function implementing the myTask02 thread.
* @param argument: Not used
* @retval None
*/
/* USER CODE END Header_StartTask02 */
void StartTask02(void const * argument)
{
  /* USER CODE BEGIN StartTask02 */
  /* Infinite loop */
  for(;;)
  {
    HAL_GPIO_TogglePin(LED3_WIFI__LED4_BLE_GPIO_Port, LED3_WIFI__LED4_BLE_Pin);
    osDelay(2000);
  }
  /* USER CODE END StartTask02 */
}
```

Step 11. Create "Timer 1" that is a One-Shot timer function named prvMyTimerOneShot(). The timer should fire 15 seconds after startup and display a message "prvMyTimerOneShot" on the console.

Embedded-RTOS-Assignment-4.ioc × main.c ×

```c
  }
  /* USER CODE END StartTask02 */
}

/* prvMyTimerOneShot function */
void prvMyTimerOneShot(void const * argument)
{
  /* USER CODE BEGIN prvMyTimerOneShot */
    char buf[100];
    snprintf(buf, sizeof(buf), "prvMyTimerOneShot\r\n");
    HAL_UART_Transmit(&huart1, (uint8_t*)buf, strlen(buf), 1000);
  /* USER CODE END prvMyTimerOneShot */
}
```

Step 12. Create "Timer 2" that is an auto-reload timer function named prvMyTimerAutoReload(). The timer should fire every 5 seconds and display a count and a message on the console, for example, "prvMyTimerAutoReload: 1".

Embedded-RTOS-Assignment-4.ioc | main.c

```c
    snprintf(buf, sizeof(buf), "prvMyTimerOneShot\r\n");
    HAL_UART_Transmit(&huart1, (uint8_t*)buf, strlen(buf), 1000);
  /* USER CODE END prvMyTimerOneShot */
}

/* prvMyTimerAutoReload function */
void prvMyTimerAutoReload(void const * argument)
{
  /* USER CODE BEGIN prvMyTimerAutoReload */
    static int count = 1;
    char buf[100];
    snprintf(buf, sizeof(buf), "prvMyTimerAutoReload: %d\r\n", count);
    HAL_UART_Transmit(&huart1, (uint8_t*)buf, strlen(buf), 1000);
    count ++;
  /* USER CODE END prvMyTimerAutoReload */
}
```

Step 13. Don’t forget to start the timer and set the timing parameter

Embedded-RTOS-Assignment-4.ioc | main.c

```c
  MX_USART3_UART_Init();
  MX_USB_OTG_FS_PCD_Init();
  /* USER CODE BEGIN 2 */

  /* USER CODE END 2 */

  /* USER CODE BEGIN RTOS_MUTEX */
  /* add mutexes, ... */
  /* USER CODE END RTOS_MUTEX */

  /* USER CODE BEGIN RTOS_SEMAPHORES */
  /* add semaphores, ... */
  /* USER CODE END RTOS_SEMAPHORES */

  /* Create the timer(s) */
  /* definition and creation of myTimer01 */
  osTimerDef(myTimer01, prvMyTimerOneShot);
  myTimer01Handle = osTimerCreate(osTimer(myTimer01), osTimerOnce, NULL);

  /* definition and creation of myTimer02 */
  osTimerDef(myTimer02, prvMyTimerAutoReload);
  myTimer02Handle = osTimerCreate(osTimer(myTimer02), osTimerPeriodic, NULL);

  /* USER CODE BEGIN RTOS_TIMERS */
  /* start timers, add new ones, ... */
  osTimerStart(myTimer01Handle, 15000);
  osTimerStart(myTimer02Handle, 5000);
  /* USER CODE END RTOS_TIMERS */

  /* USER CODE BEGIN RTOS_QUEUES */
  /* add queues, ... */
```

Step 14. Debug and run the code, test is successful