

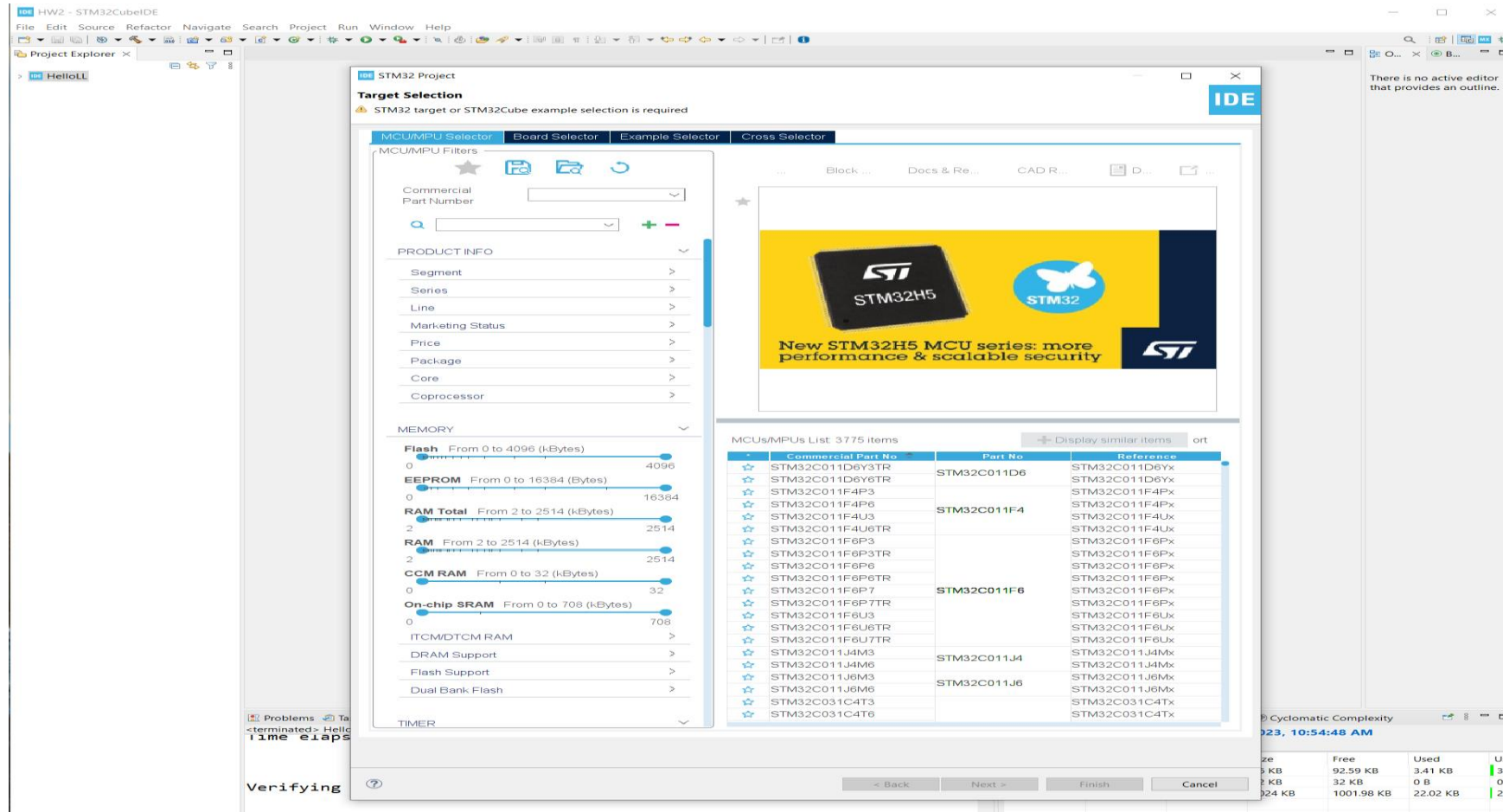
# UCSD Embedded RTOS Final Assignment

By

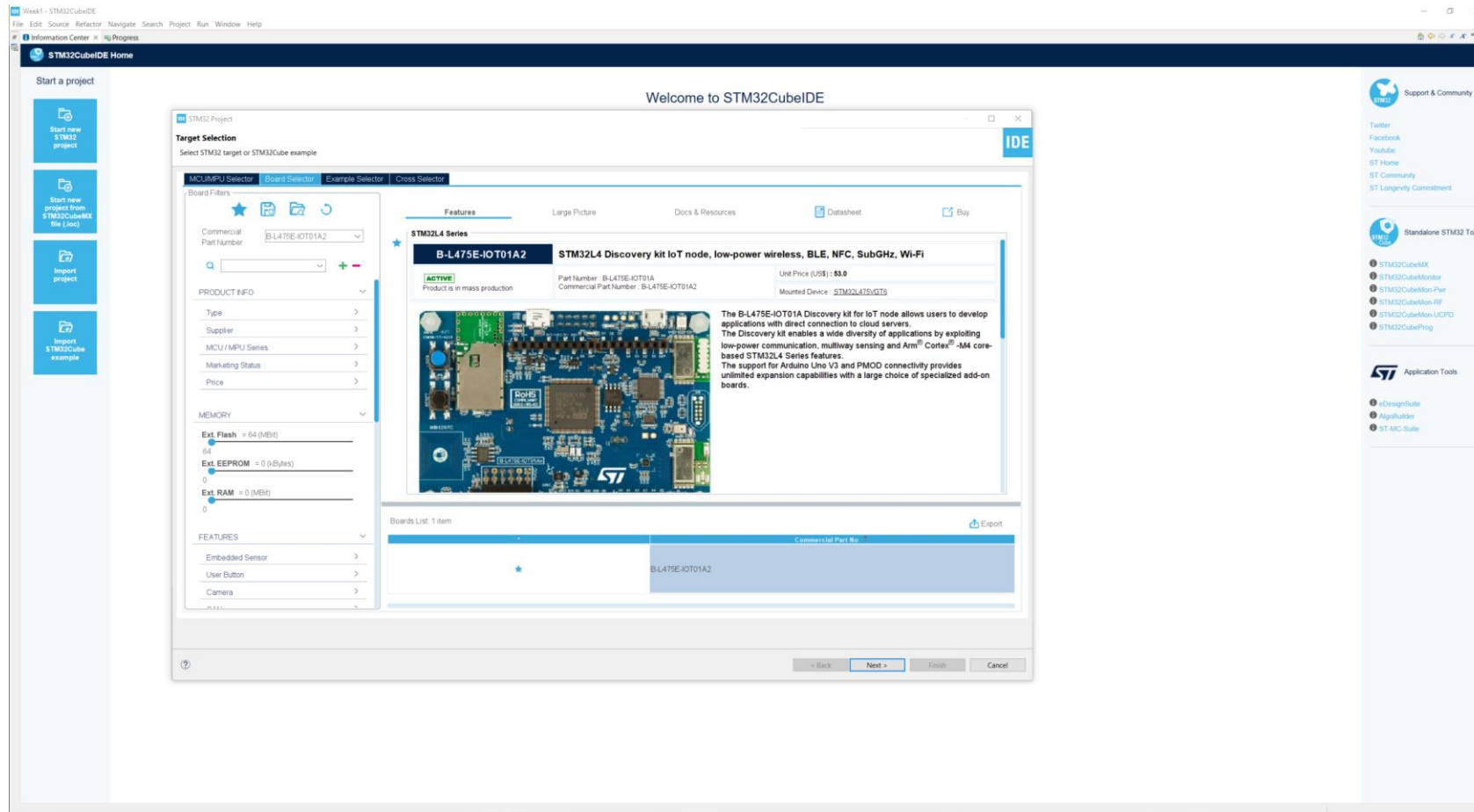
Hsuankai Chang

[hsuankac@umich.edu](mailto:hsuankac@umich.edu)

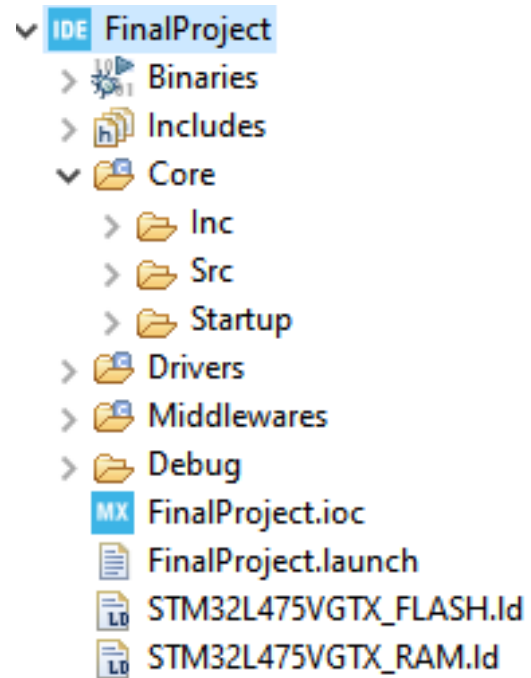
## Step 1. Startup STM32CubeIDE and create new STM32 project



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



The image shows a Windows-style dialog box titled "STM32 Project" with a subtitle "Firmware Library Package Setup". The subtitle is followed by the instruction "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". In the first section, "Target Reference" is set to "B-L475E-IOT01A2" and "Firmware Package Name and Version" is set to "STM32Cube FW\_L4" with a dropdown menu showing "V1.17.2". The second section shows the "Location" as "C:\Users\hsuankai.chang\STM32Cube\Repository" and includes a link to the "Firmware Updater". The third section contains three radio button options for code generation, with the third option, "Copy only the necessary library files", being selected. At the bottom, there are buttons for "?", "< Back", "Next >", "Finish", and "Cancel".

IDE STM32 Project

**Firmware Library Package Setup**

Setup STM32 target's firmware

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

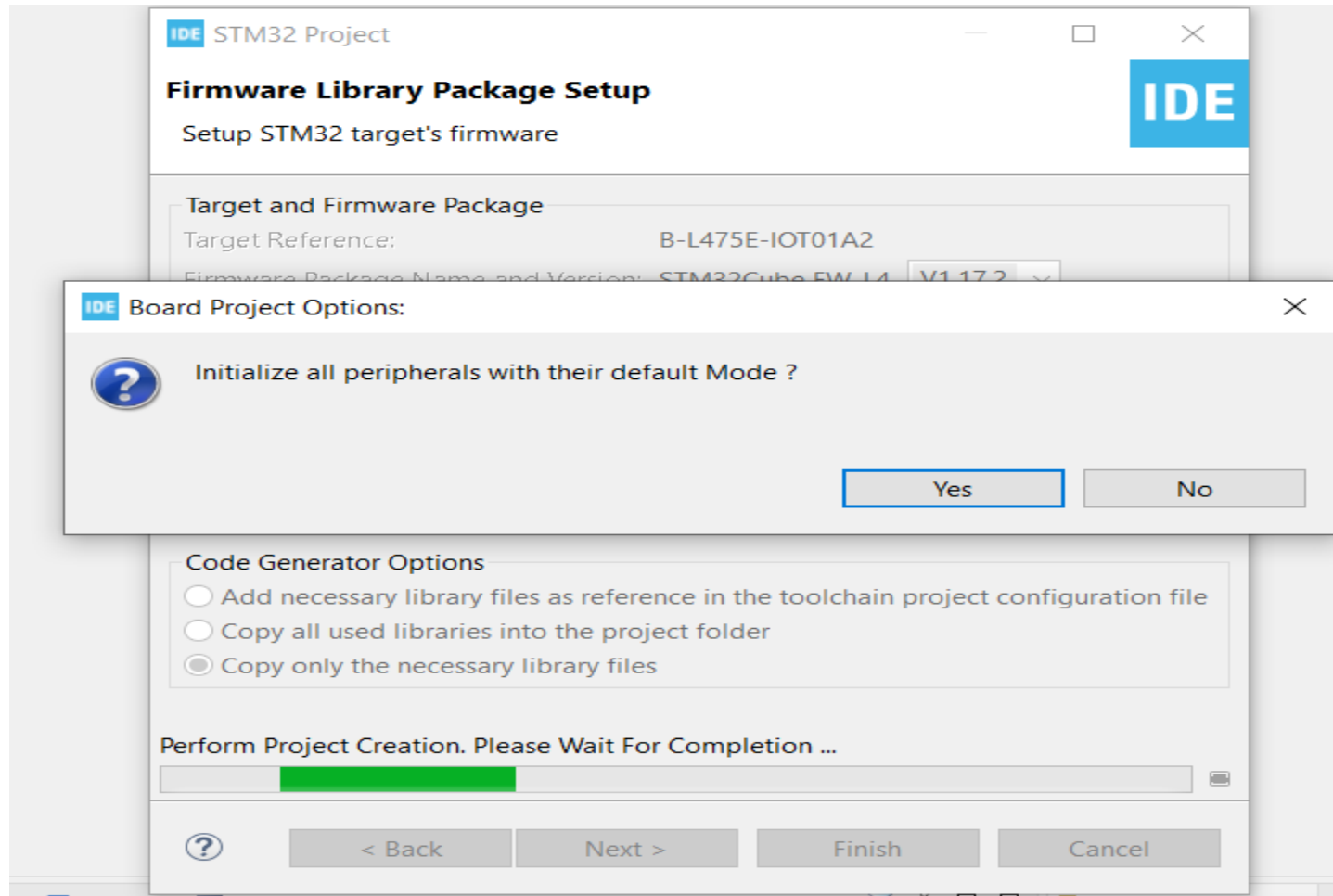
☐ Add necessary library files as reference in the toolchain project configuration file

☐ Copy all used libraries into the project folder

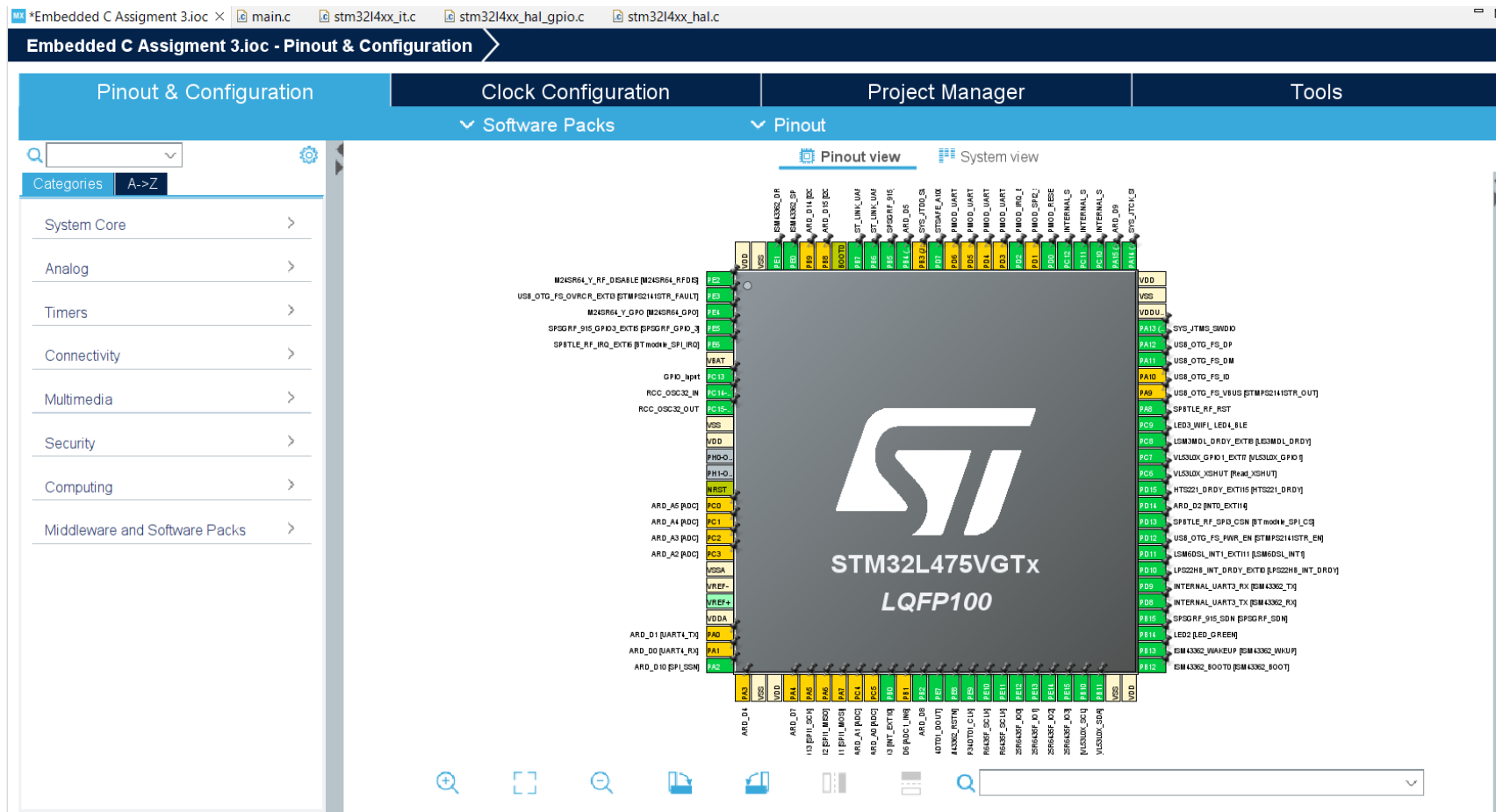
☒ Copy only the necessary library files

? < Back Next > Finish Cancel

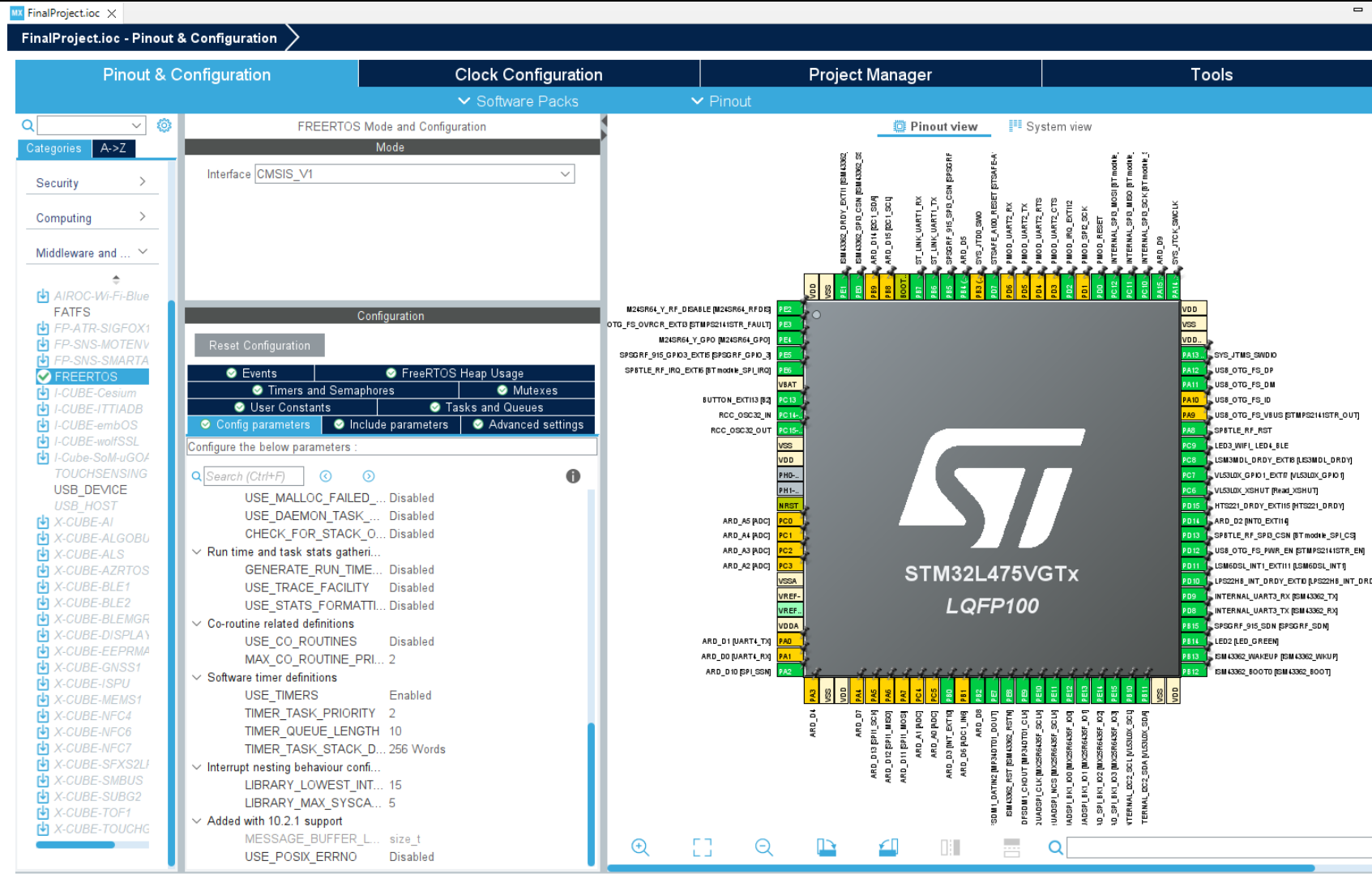
Step 5. Click yes to initialize all peripherals to default



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



Step 7. I pick option 1 as my final project, and will demonstrate required features 1 – 4 in my following slides.  
Enable CMSIS\_V1 FreeRTOS, and enable software timers in the config parameters.





Step 8. Add one periodic timer task, for one shot timer, I manually create it using FreeRTOS API

The screenshot displays the STM32CubeIDE Pinout & Configuration window for the STM32L475VGTX LQFP100. The window is divided into several sections:

- Pinout & Configuration:** This section is active and shows the configuration for the **Interface** (CMSIS\_V1) and the **Configuration** of the **Timers**. The **Timers** table lists the configuration for **myTimer02**.
- Configuration:** This section shows the configuration for the **Timers** and **Binary Semaphores**. The **Timers** table lists the configuration for **myTimer02**.
- | Timer Name | Callback   | Type           | Code Gener... | Parameter | Allocation | Control Bloc... |
|------------|------------|----------------|---------------|-----------|------------|-----------------|
| myTimer02  | Callback02 | osTimerPeri... | Default       | NULL      | Dynamic    | NULL            |
- Binary Semaphores:** This section shows the configuration for the **Binary Semaphores**. The table lists the configuration for the **Binary Semaphores**.
- | Semaphore Name | Allocation | Control Block Name |
|----------------|------------|--------------------|
|----------------|------------|--------------------|
- Counting Semaphores:** This section shows the configuration for the **Counting Semaphores**. The table lists the configuration for the **Counting Semaphores**.
- | Semaphore Name | Count | Allocation | Control Block Name |
|----------------|-------|------------|--------------------|
|----------------|-------|------------|--------------------|

The **Pinout view** on the right shows the pinout of the STM32L475VGTX LQFP100 package, with pins labeled from PA0 to PA15 and PB0 to PB15.

Step 9. Besides default task, add two more tasks, one ping task and one pong task for feature 4

FinalProject.ioc - Pinout & Configuration

Pinout & Configuration | Clock Configuration | Project Manager | Tools

Software Packs | Pinout

Categories | A-Z

Security | Computing | Middleware and ...

AIROC-Wi-Fi-Blue  
FATES  
FP-ATR-SIGFOX1  
FP-SNS-MOTENV  
FP-SNS-SMARTA  
FREERTOS  
I-CUBE-Cesium  
I-CUBE-ITTIADB  
I-CUBE-embOS  
I-CUBE-wolfSSL  
I-Cube-Sol-uGO4  
TOUCHSENSING  
USB\_DEVICE  
USB\_HOST  
X-CUBE-AI  
X-CUBE-ALGObU  
X-CUBE-ALS  
X-CUBE-AZRTOS  
X-CUBE-BLE1  
X-CUBE-BLE2  
X-CUBE-BLEMGR  
X-CUBE-DISPLA  
X-CUBE-EPRMA  
X-CUBE-GNSS1  
X-CUBE-ISPU  
X-CUBE-MEMS1  
X-CUBE-NFC4  
X-CUBE-NFC6  
X-CUBE-NFC7  
X-CUBE-SFXS2LI  
X-CUBE-SMBUS  
X-CUBE-SUBG2  
X-CUBE-TOF1  
X-CUBE-TOUCHG

Interface: CMSIS\_V1

Configuration

Reset Configuration

Mutexes | Events | FreeRTOS Heap Usage  
User Constants | Tasks and Queues | Timers and Semaphores  
Config parameters | Include parameters | Advanced settings

Tasks

Task Na...	Priority	Stack Si...	Entry Function	Cod...	Parameter	Allocation	Buffer Na...	Control ...
defaultTa...	osPriorit...	128	StartDefaultT...	Def...	NULL	Dynamic	NULL	NULL
pingTask	osPriorit...	128	PingTask	Def...	NULL	Dynamic	NULL	NULL
pongTask	osPriorit...	128	PongTask	Def...	NULL	Dynamic	NULL	NULL

Queues

Queue Name	Queue Size	Item Size	Allocation	Buffer Name	Control Block ...
pingQueue	16	uint32_t	Dynamic	NULL	NULL
pongQueue	16	uint32_t	Dynamic	NULL	NULL

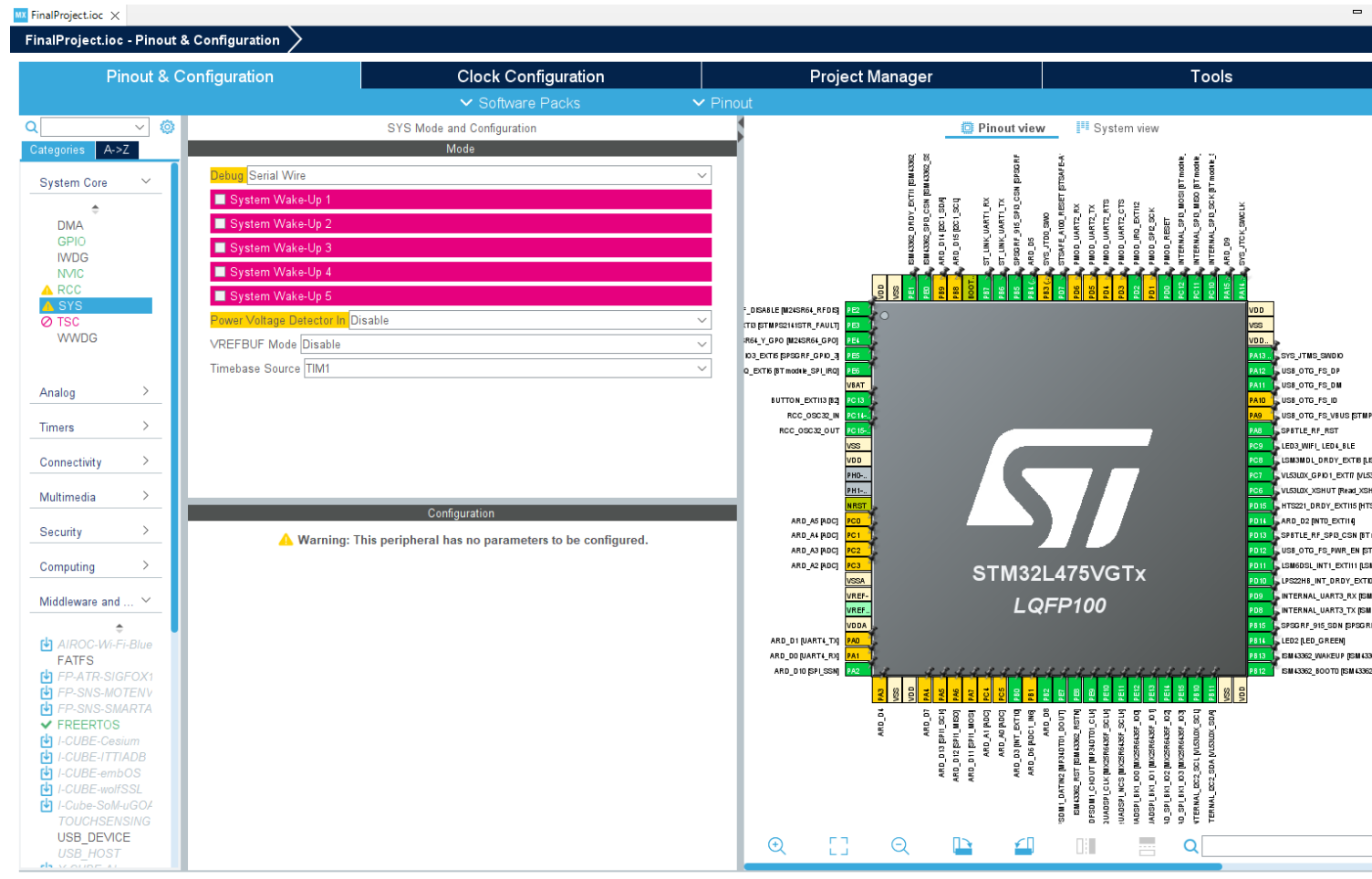
Add | Delete

Pinout view | System view

STM32L475VGx LQFP100

Pinout diagram showing various pins and their functions, including VDD, VSS, I2C, SPI, UART, and other peripherals.

## Step 10. Change Timebase source to use TIM1



Step 11. Code the main.c file, include more header files and TimerHandle\_t variable for one shot timer

```
MX FinalProject.ioc  main.c X
1 /* USER CODE BEGIN Header */
2 /**
3  * *****
4  * @file      : main.c
5  * @brief     : Main program body
6  * *****
7  * @attention
8  *
9  * Copyright (c) 2023 STMicroelectronics.
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 /* USER CODE END Header */
19 /* Includes -----*/
20 #include "main.h"
21 #include "cmsis_os.h"
22
23 /* Private includes -----*/
24 /* USER CODE BEGIN Includes */
25 #include <stdio.h>
26 #include <string.h>
27 /* USER CODE END Includes */
28
29 /* Private typedef -----*/
30 /* USER CODE BEGIN PTD */
31
```

```
MX FinalProject.ioc  main.c X
46
47 I2C_HandleTypeDef hi2c2;
48
49 QSPI_HandleTypeDef hqspi;
50
51 SPI_HandleTypeDef hspi3;
52
53 UART_HandleTypeDef huart1;
54 UART_HandleTypeDef huart3;
55
56 PCD_HandleTypeDef hpcd_USB_OTG_FS;
57
58 osThreadId defaultTaskHandle;
59 osThreadId pingTaskHandle;
60 osThreadId pongTaskHandle;
61 osMessageQId pingQueueHandle;
62 osMessageQId pongQueueHandle;
63 osTimerId myTimer02Handle;
64 /* USER CODE BEGIN PV */
65 TimerHandle_t oneShotTimer;
66 /* USER CODE END PV */
67
```

## Step 12. Timer, task and queue creation

```
FinalProjectioc main.c X
118 /* Initialize all configured peripherals */
119 MX_GPIO_Init();
120 MX_DFSDM1_Init();
121 MX_I2C2_Init();
122 MX_QUADSPI_Init();
123 MX_SPI3_Init();
124 MX_USART1_UART_Init();
125 MX_USART3_UART_Init();
126 MX_USB_OTG_FS_PCD_Init();
127 /* USER CODE BEGIN 2 */
128 HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_2);
129 /* USER CODE END 2 */
130
131 /* USER CODE BEGIN RTOS_MUTEX */
132 /* add mutexes, ... */
133 /* USER CODE END RTOS_MUTEX */
134
135 /* USER CODE BEGIN RTOS_SEMAPHORES */
136 /* add semaphores, ... */
137 /* USER CODE END RTOS_SEMAPHORES */
138
139 /* Create the timer(s) */
140 /* definition and creation of myTimer02 */
141 osTimerDef(myTimer02, Callback02);
142 myTimer02Handle = osTimerCreate(osTimer(myTimer02), osTimerPeriodic, NULL);
143
144 /* USER CODE BEGIN RTOS_TIMERS */
145 /* start timers, add new ones, ... */
146 oneShotTimer = xTimerCreate("One shot timer", pdMS_TO_TICKS(2000), pdFALSE, NULL, vTimerOneShotCallback);
147 osTimerStart(myTimer02Handle, 2000);
148 /* USER CODE END RTOS_TIMERS */
149
150 /* Create the queue(s) */
151 /* definition and creation of pingQueue */
152 osMessageQDef(pingQueue, 16, uint32_t);
153 pingQueueHandle = osMessageCreate(osMessageQ(pingQueue), NULL);
154
155 /* definition and creation of pongQueue */
156 osMessageQDef(pongQueue, 16, uint32_t);
157 pongQueueHandle = osMessageCreate(osMessageQ(pongQueue), NULL);
158
159 /* USER CODE BEGIN RTOS_QUEUES */
160 /* add queues, ... */
161 /* USER CODE END RTOS_QUEUES */
162
```

```
160 /* add queues, ... */
161 /* USER CODE END RTOS_QUEUES */
162
163 /* Create the thread(s) */
164 /* definition and creation of defaultTask */
165 osThreadDef(defaultTask, StartDefaultTask, osPriorityNormal, 0, 128);
166 defaultTaskHandle = osThreadCreate(osThread(defaultTask), NULL);
167
168 /* definition and creation of pingTask */
169 osThreadDef(pingTask, PingTask, osPriorityIdle, 0, 128);
170 pingTaskHandle = osThreadCreate(osThread(pingTask), NULL);
171
172 /* definition and creation of pongTask */
173 osThreadDef(pongTask, PongTask, osPriorityIdle, 0, 128);
174 pongTaskHandle = osThreadCreate(osThread(pongTask), NULL);
175
176 /* USER CODE BEGIN RTOS_THREADS */
177 /* add threads, ... */
178 /* USER CODE END RTOS_THREADS */
179
180 /* Start scheduler */
181 osKernelStart();
182 /* We should never get here as control is now taken by the scheduler */
183 /* Infinite loop */
184 /* USER CODE BEGIN WHILE */
185 while (1) {
186 /* USER CODE END WHILE */
187
188 /* USER CODE BEGIN 3 */
189 }
190 /* USER CODE END 3 */
191 }
```

## Step 13. Feature 1, flashing LED2

```
747 void StartDefaultTask(void const * argument)
748 {
749     /* USER CODE BEGIN 5 */
750     /* Infinite loop */
751     for (;;) {
752         HAL_GPIO_TogglePin(LED2_GPIO_Port, LED2_Pin);
753         osDelay(2000);
754     }
755     /* USER CODE END 5 */
756 }
757
```

## Step 14. Feature 2, periodic timer and one-shot timer creation, periodic timer start, and periodic timer callback function

```
/* Create the timer(s) */
/* definition and creation of myTimer02 */
osTimerDef(myTimer02, Callback02);
myTimer02Handle = osTimerCreate(osTimer(myTimer02), osTimerPeriodic, NULL);

/* USER CODE BEGIN RTOS_TIMERS */
/* start timers, add new ones, ... */
oneShotTimer = xTimerCreate("One shot timer", pdMS_TO_TICKS(2000), pdFALSE, NULL, vTimerOneShotCallback);
osTimerStart(myTimer02Handle, 2000);
/* USER CODE END RTOS_TIMERS */

~..
818 /* Callback02 function */
819 void Callback02(void const * argument)
820 {
821     /* USER CODE BEGIN Callback02 */
822     static int countAutoReload = 1;
823     char buf[100];
824     snprintf(buf, sizeof(buf), "autoReloadTimer triggered: %d\r\n", countAutoReload);
825     HAL_UART_Transmit(&huart1, (uint8_t*) buf, strlen(buf), 1000);
826     // ARD_D5
827     HAL_GPIO_TogglePin(GPIOB, GPIO_PIN_4);
828     countAutoReload++;
829     /* USER CODE END Callback02 */
830 }
```

Step 15. Feature 3, one shot timer code. The reason why I did not use CMSIS API to create one shot timer is, I find out osTimerStart can not be used in ISR, so change to use FreeRTOS API

```
709 void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
710 {
711     if(GPIO_Pin == BUTTON_EXTI13_Pin)
712     {
713         BaseType_t xHigherPriorityTaskWoken = pdFALSE;
714         if(xTimerStartFromISR(oneShotTimer, &xHigherPriorityTaskWoken) != pdPASS)
715         {
716             char buf[100];
717             snprintf(buf, sizeof(buf), "oneShotTimer start failed\r\n");
718             HAL_UART_Transmit(&huart1, (uint8_t*) buf, strlen(buf), 1000);
719         }
720
721         if( xHigherPriorityTaskWoken != pdFALSE )
722         {
723             portYIELD_FROM_ISR(xHigherPriorityTaskWoken);
724         }
725     }
726 }
727
728 void vTimerOneShotCallback( TimerHandle_t xTimer )
729 {
730     static int countOneShot = 1;
731     char buf[100];
732     snprintf(buf, sizeof(buf), "oneShotTimer triggered: %d\r\n", countOneShot);
733     HAL_UART_Transmit(&huart1, (uint8_t*) buf, strlen(buf), 1000);
734     // ARD_D8
735     HAL_GPIO_TogglePin(GPIOB, GPIO_PIN_2);
736     countOneShot++;
737 }
```



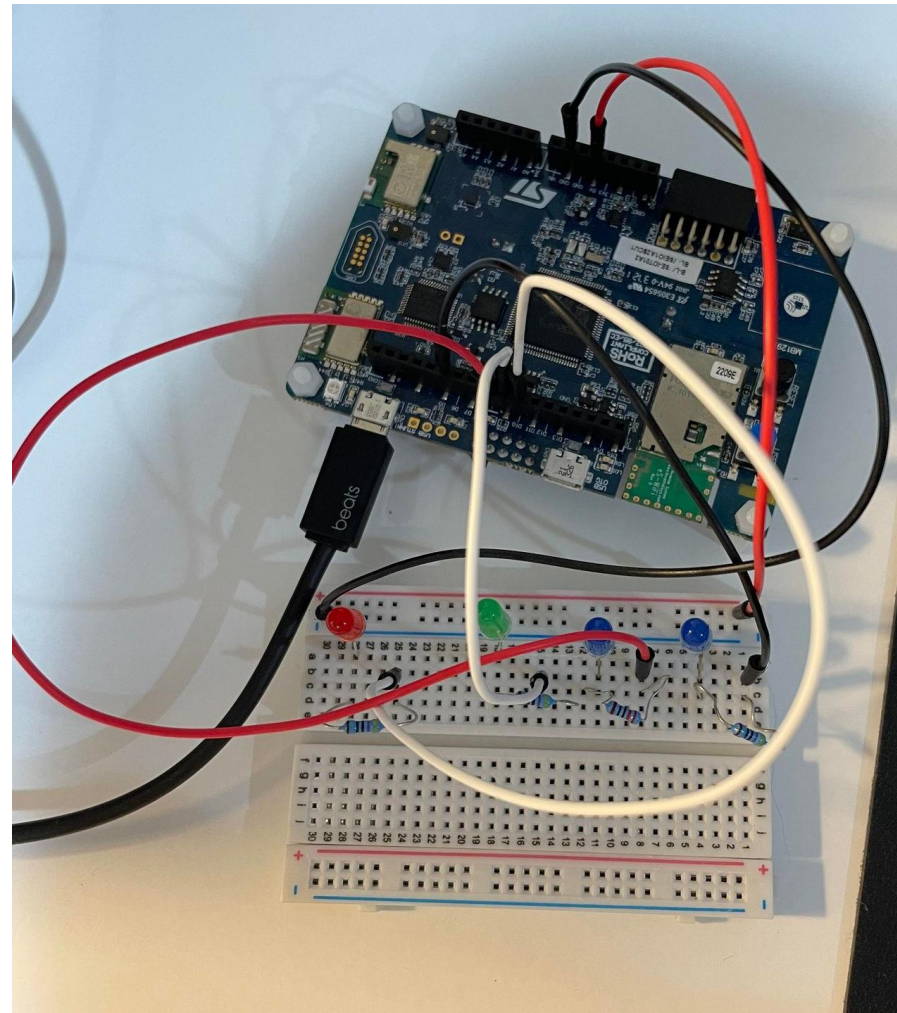
## Step 15. Feature 4, ping pong task

```
152 osMessageQDef(pingQueue, 16, uint32_t);
153 pingQueueHandle = osMessageCreate(osMessageQ(pingQueue), NULL);
154
155 /* definition and creation of pongQueue */
156 osMessageQDef(pongQueue, 16, uint32_t);
157 pongQueueHandle = osMessageCreate(osMessageQ(pongQueue), NULL);
158
```

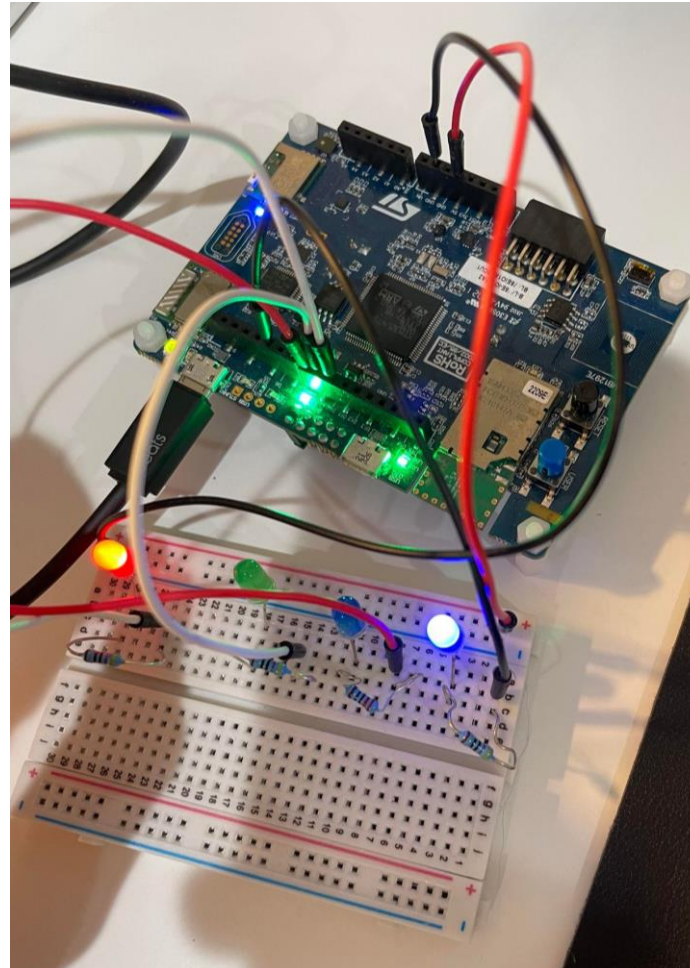
```
765 void PingTask(void const * argument)
766 {
767     /* USER CODE BEGIN PingTask */
768     static int pingcmd = 0;
769     osEvent pingEvt;
770     char buf[100];
771     /* Infinite loop */
772     for(;;)
773     {
774         osMessagePut(pingQueueHandle, pingcmd, osWaitForever);
775         pingcmd++;
776         pingEvt = osMessageGet(pongQueueHandle, osWaitForever);
777         if(pingEvt.status == osEventMessage){
778             snprintf(buf, sizeof(buf), "ping command count: %d\r\n", (int)(pingEvt.value.p));
779             HAL_UART_Transmit(&huart1, (uint8_t*)buf, strlen(buf), 1000);
780             // ARD_D10
781             HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_15);
782         }
783         osDelay(2000);
784     }
785     /* USER CODE END PingTask */
786 }
787
```

```
788 /* USER CODE BEGIN Header_PongTask */
789 /**
790  * @brief Function implementing the pongTask thread.
791  * @param argument: Not used
792  * @retval None
793  */
794 /* USER CODE END Header_PongTask */
795 void PongTask(void const * argument)
796 {
797     /* USER CODE BEGIN PongTask */
798     static int pongcmd = 0;
799     osEvent pongEvt;
800     char buf[100];
801     /* Infinite loop */
802     for(;;)
803     {
804         pongEvt = osMessageGet(pingQueueHandle, osWaitForever);
805         if(pongEvt.status == osEventMessage){
806             snprintf(buf, sizeof(buf), "pong command count: %d\r\n", (int)(pongEvt.value.p));
807             HAL_UART_Transmit(&huart1, (uint8_t*)buf, strlen(buf), 1000);
808             // ARD_D9
809             HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_2);
810             osMessagePut(pongQueueHandle, pongcmd, osWaitForever);
811             pongcmd++;
812         }
813         osDelay(2000);
814     }
815     /* USER CODE END PongTask */
816 }
```

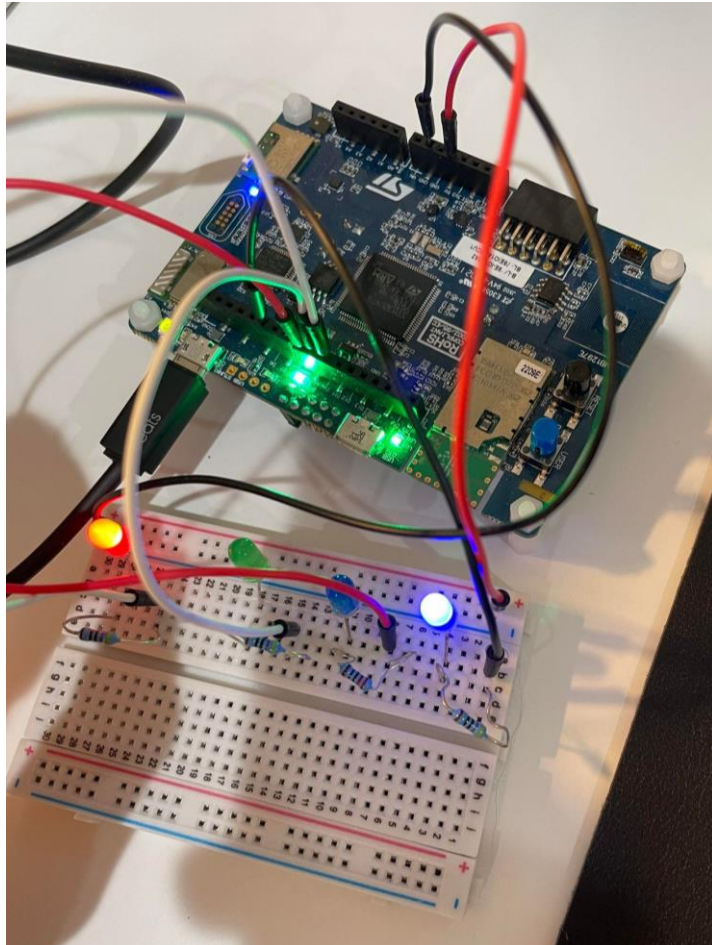
Step 16. Connect the hardware, from right to left is ARD\_D5, ARD\_D8, ARD\_D9, ARD\_D10



Step 17. Feature 1 test successful, LED2 pin flash at a rate of 2 seconds



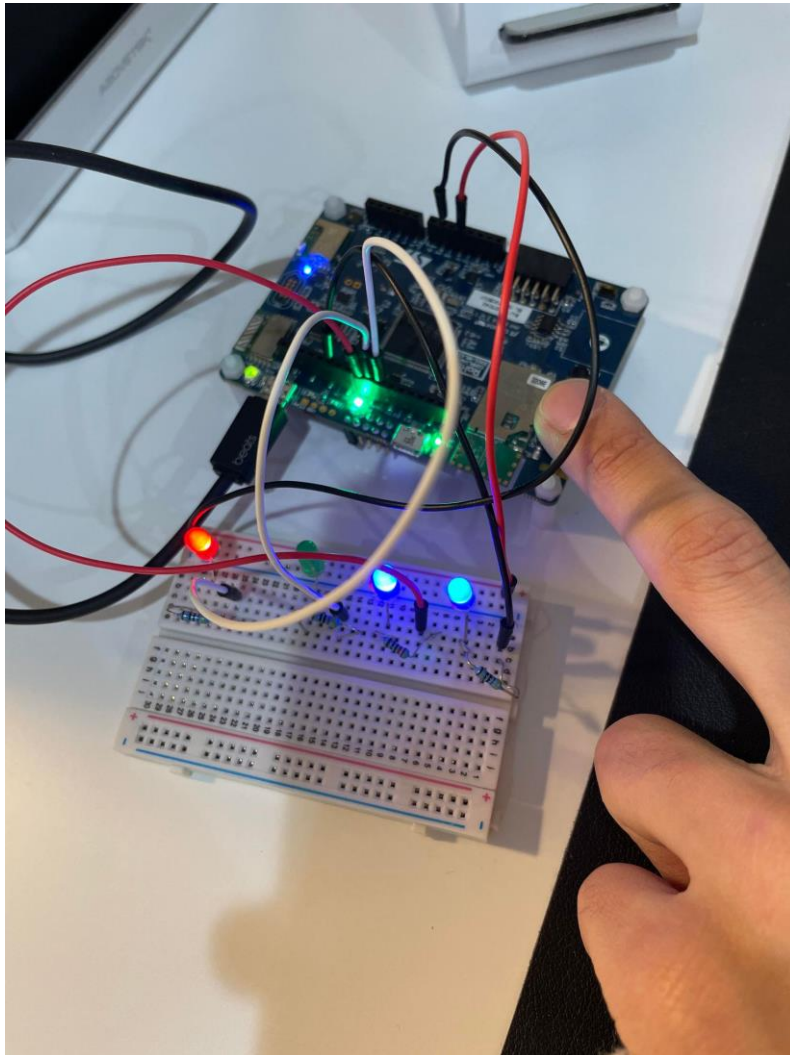
Step 18. Feature 2 test successful, ARD\_D5 LED flash at a rate of 2 seconds, timer call back function triggered



```
COM4 - Tera Term VT
File Edit Setup Control Window Help
ping command count: 112
pong command count: 112
autoReloadTimer triggered: 113
ping command count: 113
pong command count: 113
autoReloadTimer triggered: 114
ping command count: 114
pong command count: 114
autoReloadTimer triggered: 115
ping command count: 115
pong command count: 115
autoReloadTimer triggered: 116
ping command count: 116
pong command count: 116
autoReloadTimer triggered: 117
ping command count: 117
pong command count: 117
autoReloadTimer triggered: 118
ping command count: 118
pong command count: 118
autoReloadTimer triggered: 119
ping command count: 119
pong command count: 119
```

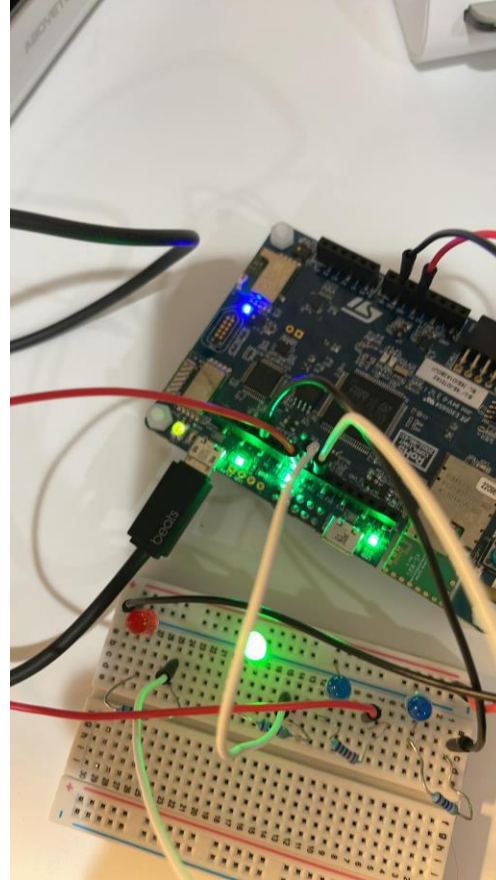
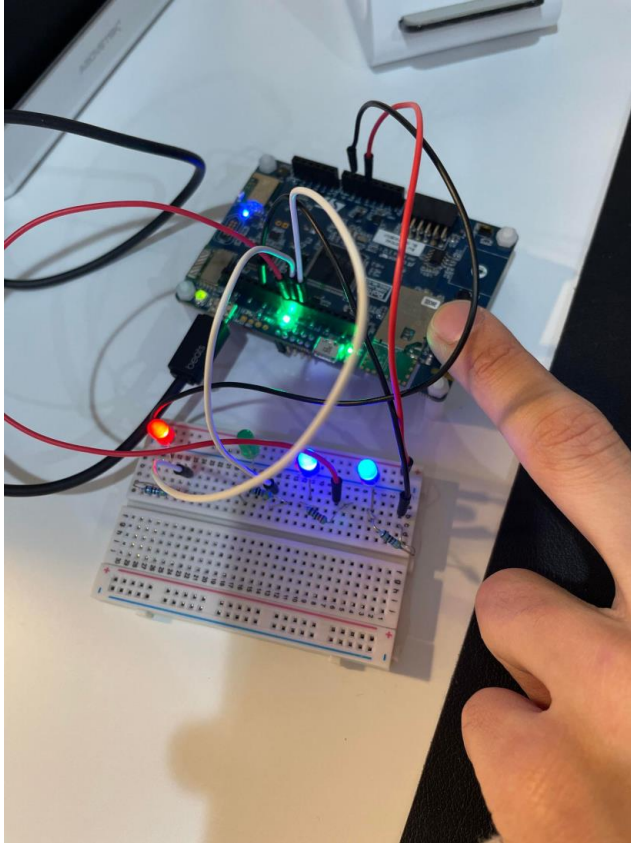


Step 19. one shot timer test successful, call back function triggered, ARD\_D8 LED flash with time out 2 seconds



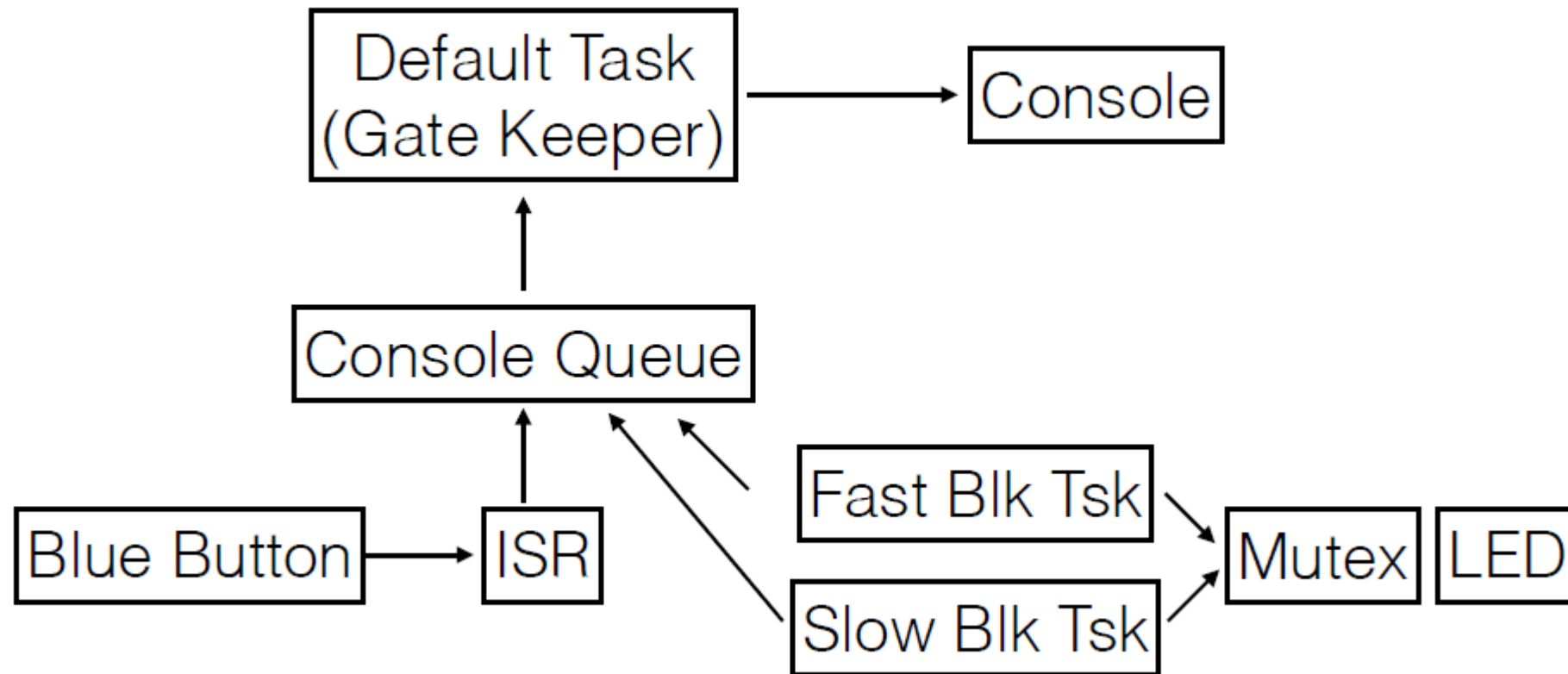
```
COM4 - Tera Term VT
File Edit Setup Control Window Help
autoReloadTimer triggered: 179
ping command count: 179
pong command count: 179
autoReloadTimer triggered: 180
ping command count: 180
pong command count: 180
autoReloadTimer triggered: 181
ping command count: 181
pong command count: 181
oneShotTimer triggered: 2
```

Step 20. Feature 4 test successful, ping pong task send command to queue to toggle LED (ARD\_D9, ARD\_D10 pin)



```
COM4 - Tera Term VT
File Edit Setup Control Window Help
pong command count: 266
autoReloadTimer triggered: 267
ping command count: 267
pong command count: 267
autoReloadTimer triggered: 268
ping command count: 268
pong command count: 268
autoReloadTimer triggered: 269
ping command count: 269
pong command count: 269
autoReloadTimer triggered: 270
ping command count: 270
pong command count: 270
autoReloadTimer triggered: 271
ping command count: 271
pong command count: 271
autoReloadTimer triggered: 272
ping command count: 272
pong command count: 272
autoReloadTimer triggered: 273
ping command count: 273
pong command count: 273
autoReloadTimer triggered: 274
```

Step 20. Resource LED bonus project. I will create a new project and follow the code demo like below



## Step 21. Add two more tasks, fast and slow blink tasks

ResourceLED.ioc - Pinout & Configuration

Pinout & Configuration | Clock Configuration

Software Packs | Pinout

FREERTOS Mode and Configuration

Interface: CMSIS\_V1

Configuration

Reset Configuration

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

Config parameters | Include parameters | Advanced settings | User Constants

Task Name	Priority	St	Entry Function	Code Gene	Parameter	Allocation	Buffer Name	Control Blo
defaultTask	osPrior...	128	StartDefaultTask	Default	NULL	Dynamic	NULL	NULL
myFastBlinkLEDTask	osPrior...	128	StartFastBlinkT...	Default	NULL	Dynamic	NULL	NULL
mySlowBlinkTask	osPrior...	128	StartSlowBlinkT...	Default	NULL	Dynamic	NULL	NULL

Add | Delete

Queues

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

Add | Delete



## Step 22. Add a queue

**Queues**

Queue Name	Queue Size	Item Size	Allocation	Buffer Name	Control Block Name
ConsoleQueue	16	uint16_t	Dynamic	NULL	NULL

## Step 23. Add a mutex

✓ Tasks and Queues	✓ Timers and Semaphores	✓ <b>Mutexes</b>	✓ Events	✓ FreeRTOS Heap Usage
✓ Config parameters	✓ Include parameters	✓ Advanced settings	✓ User Constants	

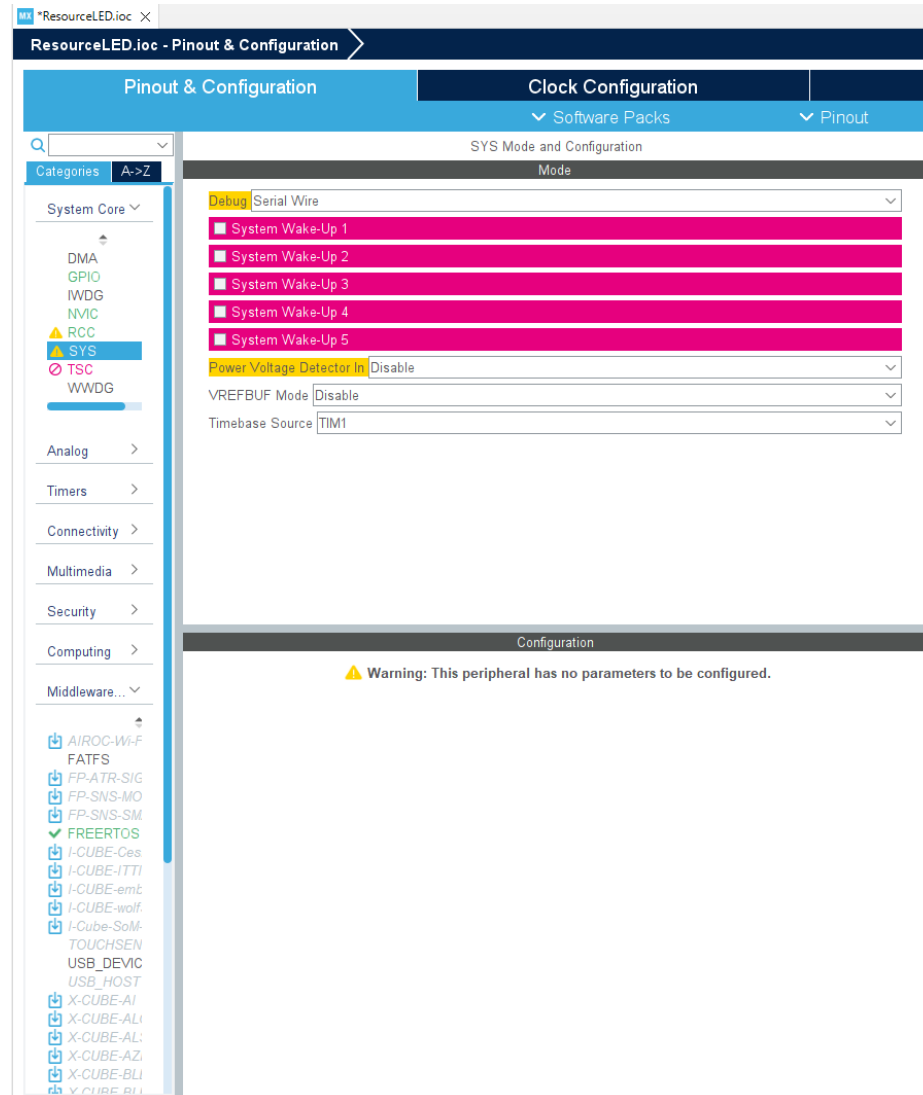
**Mutexes**

Mutex Name	Allocation	Control Block Name
LEDFlashMutex	Dynamic	NULL

AddDelete

**Recursive Mutexes**

## Step 24. Select Timbase to TIM1



## Step 25. Code the default task

```
ResourceLED.ioc  main.c  stm32l4xx_it.c  stm32l4xx_hal_gpio.c  startup_stm32l475vgtx.s
712 /**
713  * @brief Function implementing the defaultTask thread.
714  * @param argument: Not used
715  * @retval None
716  */
717 /* USER CODE END Header_StartDefaultTask */
718 void StartDefaultTask(void const * argument)
719 {
720     /* USER CODE BEGIN 5 */
721     char buf[100];
722     /* Infinite loop */
723     for(;;)
724     {
725         // Wait to receive command
726         osEvent event = osMessageGet(ConsoleQueueHandle, osWaitForever);
727         // Format message for console
728         snprintf(buf, sizeof(buf), "event.value.v: %lu\r\n", event.value.v);
729         HAL_UART_Transmit(&huart1, (uint8_t*)buf, strlen(buf), 1000);
730     }
731     /* USER CODE END 5 */
732 }
```

## Step 26. Fast LED blink task

```
741 void StartFastBlinkTask(void const * argument)
742 {
743     /* USER CODE BEGIN StartFastBlinkTask */
744     /* Infinite loop */
745     for(;;)
746     {
747         osDelay(1000);
748         // Grab mutex
749         osMutexWait(LEDFlashMutexHandle, osWaitForever);
750         osMessagePut(ConsoleQueueHandle, CMD_FAST_LED_FLASH, osWaitForever);
751         HAL_GPIO_WritePin(LED2_GPIO_Port, LED2_Pin, SET);
752         osDelay(250);
753         HAL_GPIO_WritePin(LED2_GPIO_Port, LED2_Pin, RESET);
754         osMutexRelease(LEDFlashMutexHandle);
755     }
756     /* USER CODE END StartFastBlinkTask */
757 }
```

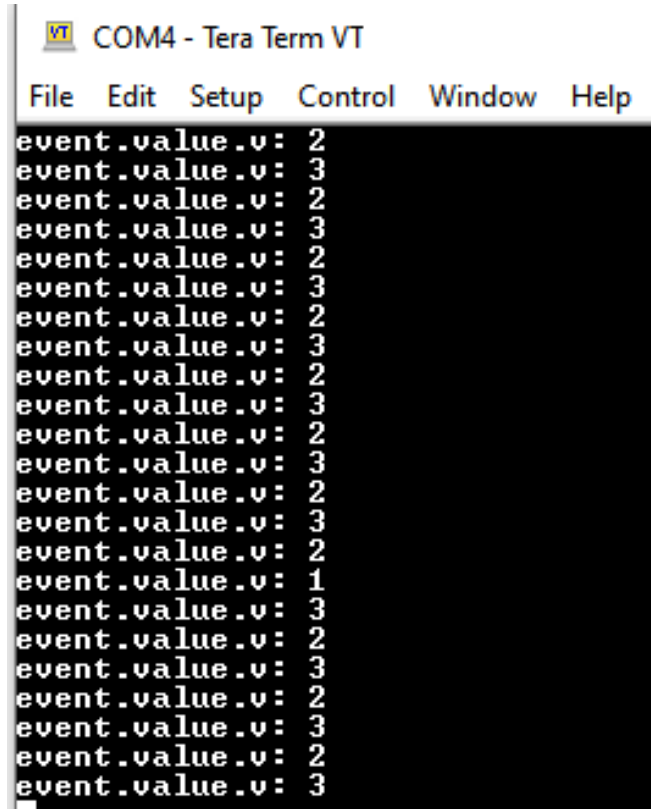
## Step 27. slow LED blink test

```
766 void StartSlowBlinkTask(void const * argument)
767 {
768     /* USER CODE BEGIN StartSlowBlinkTask */
769     /* Infinite loop */
770     for(;;)
771     {
772         osDelay(1000);
773         // Grab mutex
774         osMutexWait(LEDFlashMutexHandle, osWaitForever);
775         osMessagePut(ConsoleQueueHandle, CMD_SLOW_LED_FLASH, osWaitForever);
776         HAL_GPIO_WritePin(LED2_GPIO_Port, LED2_Pin, SET);
777         osDelay(1000);
778         HAL_GPIO_WritePin(LED2_GPIO_Port, LED2_Pin, RESET);
779         osMutexRelease(LEDFlashMutexHandle);
780     }
781     /* USER CODE END StartSlowBlinkTask */
782 }
```

## Step 28. GPIO call back

```
704 /* USER CODE BEGIN 4 */
705 void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
706 {
707     osMessagePut(ConsoleQueueHandle, CMD_BUTTON_PRESS, osWaitForever);
708 }
709 /* USER CODE END 4 */
```

Step 29. Build and run the code, test is successful



The screenshot shows a terminal window titled "COM4 - Tera Term VT". The menu bar includes "File", "Edit", "Setup", "Control", "Window", and "Help". The terminal output consists of 25 lines, each starting with "event.value.v:" followed by a value. The values are: 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 1, 3, 2, 3, 2, 3, 2, 3.

```
COM4 - Tera Term VT
File Edit Setup Control Window Help
event.value.v: 2
event.value.v: 3
event.value.v: 2
event.value.v: 3
event.value.v: 2
event.value.v: 3
event.value.v: 2
event.value.v: 3
event.value.v: 2
event.value.v: 3
event.value.v: 2
event.value.v: 3
event.value.v: 2
event.value.v: 3
event.value.v: 2
event.value.v: 3
event.value.v: 2
event.value.v: 3
event.value.v: 2
event.value.v: 1
event.value.v: 3
event.value.v: 2
event.value.v: 3
event.value.v: 2
event.value.v: 3
event.value.v: 2
event.value.v: 3
```



## Step 30. Bonus 2, Event group LED project, create 2 more tasks

The screenshot shows the STM32CubeIDE Pinout & Configuration window for the EventGroupLED.ioc project. The window is divided into two main sections: Pinout & Configuration and Clock Configuration. The Pinout & Configuration section is active, showing the 'FREERTOS Mode and Configuration' section. The 'Interface' dropdown is set to 'CMSIS\_V1'. The 'Configuration' section shows the 'Tasks and Queues' tab selected. The 'Tasks' table lists three tasks: defaultTask, myTask02, and myTask03. The 'Queues' section is empty.

Task Name	Priority	Stack Siz...	Entry Fun...	Code Gen...	Parameter	Allocation	Buffer Name	Control Bl...
defaultTask	osPriorityL...	128	StartDefa...	Default	NULL	Dynamic	NULL	NULL
myTask02	osPriorityL...	128	StartTask02	Default	NULL	Dynamic	NULL	NULL
myTask03	osPriorityL...	128	StartTask03	Default	NULL	Dynamic	NULL	NULL

## Step 31. Create even group handle and flags

```
main.c × EventGroupLED.ioc FreeRTOSConfig.h startup_stm32l47
46 I2C_HandleTypeDef hi2c2;
47
48 QSPI_HandleTypeDef hqspi;
49
50 SPI_HandleTypeDef hspi3;
51
52 UART_HandleTypeDef huart1;
53 UART_HandleTypeDef huart3;
54
55 PCD_HandleTypeDef hpcd_USB_OTG_FS;
56
57 osThreadId defaultTaskHandle;
58 osThreadId myTask02Handle;
59 osThreadId myTask03Handle;
60 /* USER CODE BEGIN PV */
61 #define mainDEFAULT_BIT (1UL << 0UL)
62 #define mainTASK_BIT_1 (1UL << 1UL)
63
64 EventGroupHandle_t xEventGroup;
65 /* USER CODE END PV */
66
```

```
main.c × FreeRTOSConfig.h stm32l4xx_it.c
102 HAL_Init();
103
104 /* USER CODE BEGIN Init */
105 /* USER CODE END Init */
106
107 /* Configure the system clock */
108 SystemClock_Config();
109
110 /* USER CODE BEGIN SysInit */
111
112 /* USER CODE END SysInit */
113
114 /* Initialize all configured peripherals */
115 MX_GPIO_Init();
116 MX_DFSDM1_Init();
117 MX_I2C2_Init();
118 MX_QUADSPI_Init();
119 MX_SPI3_Init();
120 MX_USART1_UART_Init();
121 MX_USART3_UART_Init();
122 MX_USB_OTG_FS_PCD_Init();
123 /* USER CODE BEGIN 2 */
124 xEventGroup = xEventGroupCreate();
125 /* USER CODE END 2 */
126
```

## Step 32. Set flags in default task and task02

```
main.c x EventGroupLED.ioc FreeRTOSConfig.h startup_stm32l475vgtx.s
701 void StartDefaultTask(void const * argument)
702 {
703     /* USER CODE BEGIN 5 */
704     /* Infinite loop */
705     for(;;)
706     {
707         osDelay(1000);
708         xEventGroupSetBits(xEventGroup, mainDEFAULT_BIT);
709     }
710     /* USER CODE END 5 */
711 }
712
713 /* USER CODE BEGIN Header_StartTask02 */
714 /**
715  * @brief Function implementing the myTask02 thread.
716  * @param argument: Not used
717  * @retval None
718  */
719 /* USER CODE END Header_StartTask02 */
720 void StartTask02(void const * argument)
721 {
722     /* USER CODE BEGIN StartTask02 */
723     /* Infinite loop */
724     for(;;)
725     {
726         osDelay(3000);
727         xEventGroupSetBits(xEventGroup, mainTASK_BIT_1);
728     }
729     /* USER CODE END StartTask02 */
730 }
```

Step 33. Blink LED2 in task3, build and run the code, test is successful

```
739 void StartTask03(void const * argument)
740 {
741     /* USER CODE BEGIN StartTask03 */
742     /* Infinite loop */
743     for(;;)
744     {
745         xEventGroupWaitBits(xEventGroup,
746                             mainDEFAULT_BIT | mainTASK_BIT_1,
747                             pdTRUE,
748                             pdTRUE,
749                             portMAX_DELAY);
750         HAL_GPIO_WritePin(LED2_GPIO_Port, LED2_Pin, SET);
751         osDelay(1000);
752         HAL_GPIO_WritePin(LED2_GPIO_Port, LED2_Pin, RESET);
753     }
754     /* USER CODE END StartTask03 */
755 }
```