Lab Objective

To demonstrate FreeRTOS task management using the following features: -

- GPIO interfacing with LEDs, Switches
- Task synchronisation using Semaphores

Reference: FreeRTOS Reference Manual Chapter

- Semaphore / Mutexes
  - xSemaphoreCreateBinary()
  - xSemaphoreCreateBinaryStatic()
  - vSemaphoreCreateBinary()
  - xSemaphoreCreateCounting()
  - xSemaphoreCreateCountingStatic()
  - xSemaphoreCreateMutex()
  - xSemaphoreCreateMutexStatic()
  - xSem'CreateRecursiveMutex()
  - xSem'CreateRecursiveMutexStatic()
  - vSemaphoreDelete()
  - xSemaphoreGetMutexHolder()
  - uxSemaphoreGetCount()
  - **xSemaphoreTake()**
  - xSemaphoreTakeFromISR()
  - xSemaphoreTakeRecursive()
  - xSemaphoreGive()
  - xSemaphoreGiveRecursive()
  - xSemaphoreGiveFromISR()

Part A: GPIO Interfacing using Task Suspend and Resume

Create 2 tasks called ButtonTask and LED_Task in main().

ButtonTask polls the active-low push-button switch. ButtonTask should suspend itself after detecting a switch press.

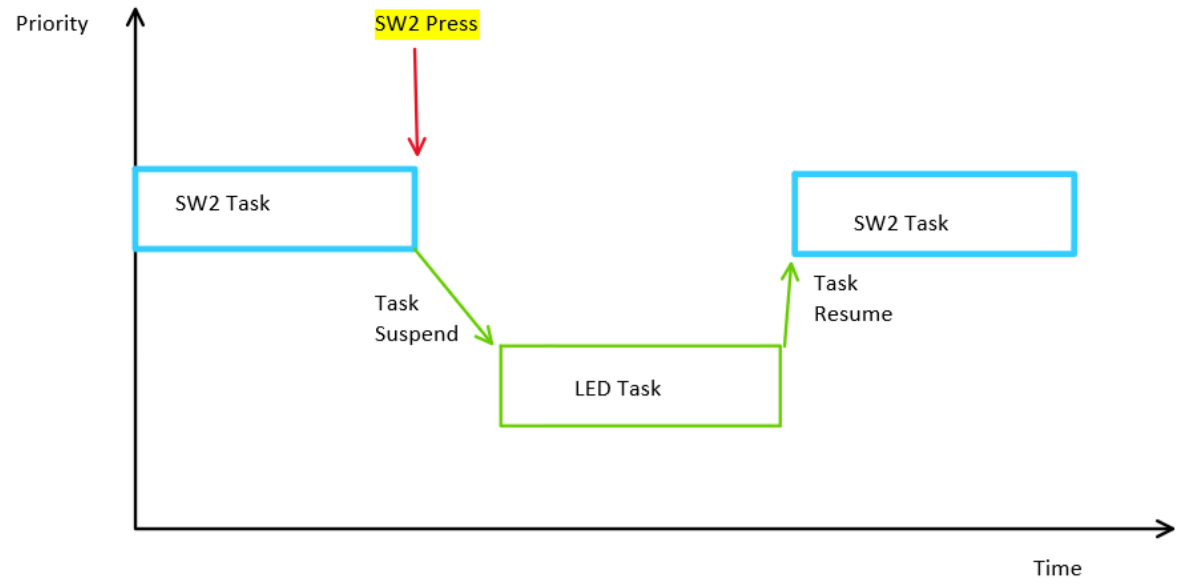
The LED_Task toggles the LED2. After toggling the LED, the LED_Task should cause the SW2_Task to resume.

```
Starting button task
Button pressed, suspending button task...
Starting LED task
Toggling LED, resuming button task...

Button pressed, suspending button task...
Toggling LED, resuming button task...
```

Console | Problems | Executables | Debugger Console | Memory | FreeRTOS Task List

Name	Priority (B...	Start of S...	Top of St...	State	Event Ob...	Stack Usa...	Run Time...
Button Task	6/6	0x20001...	0x20001...	RUNNING		312B / 8...	>99%
IDLE	0/0	0x20000...	0x20000...	READY		44B / 51...	0%
LED Task	5/5	0x20001...	0x20001...	READY		312B / 8...	<1%
Tmr Svc	6/6	0x20000...	0x20000...	BLOCKED	TmrQ	168B / 1...	<1%



Part B: GPIO Interfacing using Delay Blocking

Part B implements the same functionality as part A but uses task delay blocking instead of task suspend/resume. Use vTaskDelay() in the ButtonTask to poll the switch every 100ms.

Hint: Use a global flag variable to signal that a switch press has occurred.

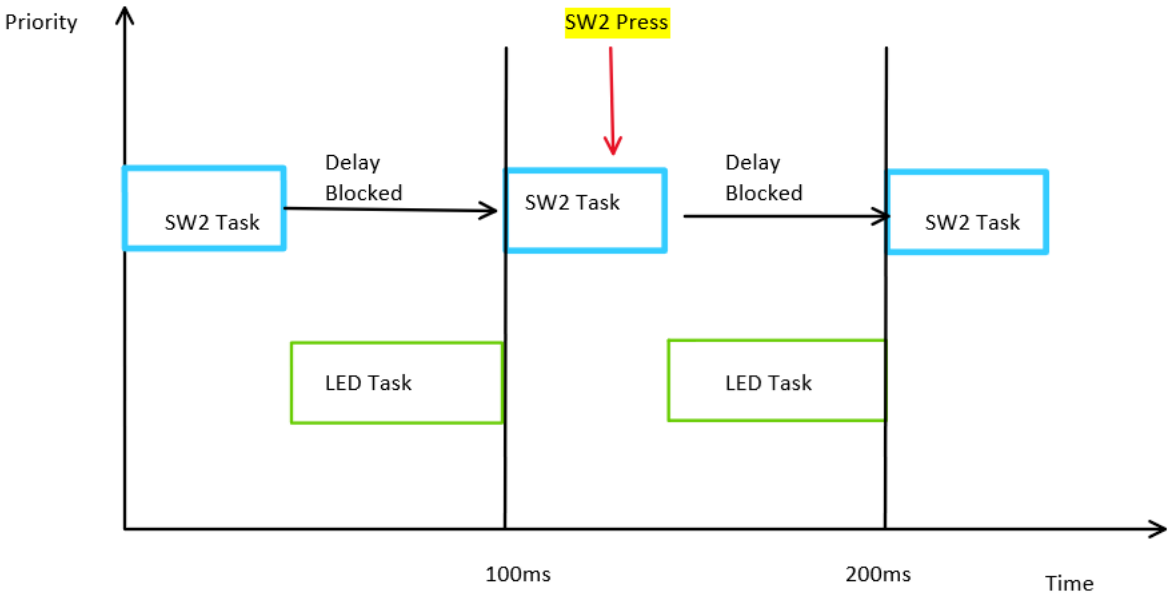
```
Starting button task
Starting LED task
Button pressed...
Toggling LED...

Button pressed...
Toggling LED...

Button pressed...
Toggling LED...
```

Console | Problems | Executables | Debugger Console | Memory | FreeRTOS Task List

Name	Priority (B...	Start of S...	Top of St...	State	Event Ob...	Stack Usa...	Run Time...
Button Task	6/6	0x20001...	0x20001...	DELAYED		312B / 8...	5%
IDLE	0/0	0x20000...	0x20000...	READY		44B / 51...	0%
→ LED Task	5/5	0x20001...	0x20001...	RUNNING		312B / 8...	95%
Tmr Svc	6/6	0x20000...	0x20000...	BLOCKED	TmrQ	168B / 1...	<1%



Part C: Task Synchronisation using a Semaphore

Create a semaphore. Create 2 tasks called ButtonTask and LED_Task in main().

ButtonTask polls the active-low push-button switch every 100ms. The semaphore is given when a switch press is detected.

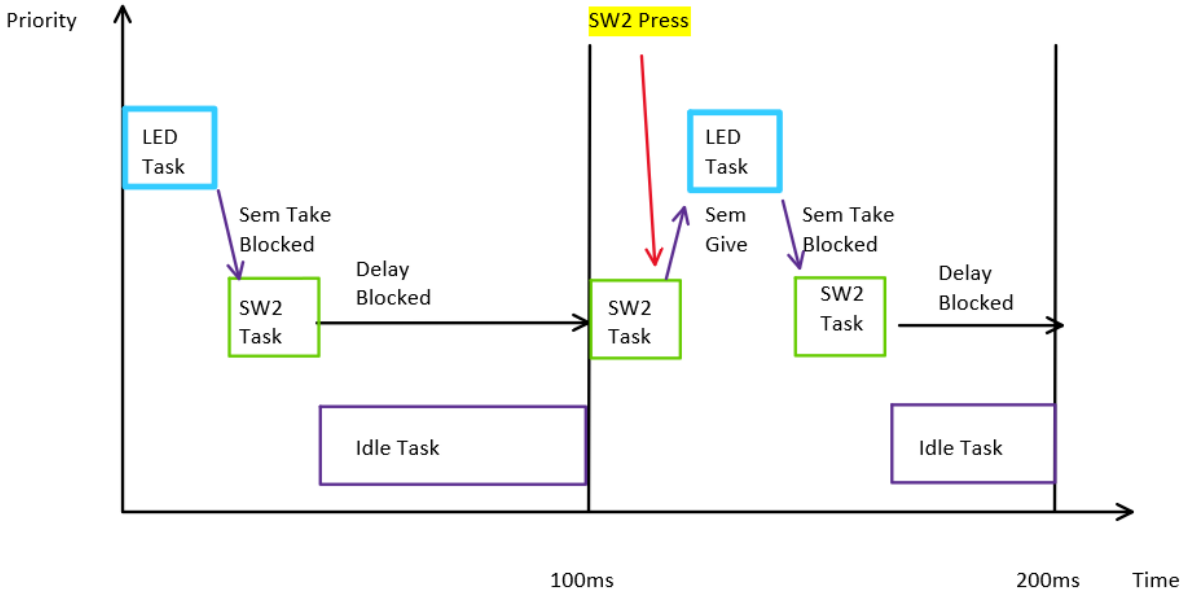
The LED_Task blocks waiting to take the semaphore. The LED should toggle when the LED task unblocks.

```
Starting LED task
Starting button task
Button pressed, giving semaphore...
Semaphore taken, Toggling LED...

Button pressed, giving semaphore...
Semaphore taken, Toggling LED...
```

Console | Problems | Executables | Debugger Console | Memory | FreeRTOS Task List

Name	Priority (B...	Start of S...	Top of St...	State	Event Ob...	Stack Usa...	Run Time...
Button Task	5/5	0x20001...	0x20001...	DELAYED		312B / 8...	3%
IDLE	0/0	0x20000...	0x20000...	RUNNING		96B / 51...	97%
LED Task	6/6	0x20001...	0x20001...	BLOCKED	0x20001...	312B / 8...	<1%
Tmr Svc	6/6	0x20000...	0x20000...	BLOCKED	TmrQ	168B / 1...	<1%



Part D: Task synchronisation with multiple semaphores

Create 3 tasks: LED_Task, ButtonTask, TimerTask.

Create 2 semaphores, buttonSemaphore and timerSemaphore.

The LED_Task should block waiting to take one of the two semaphores.
ButtonTask gives the buttonSemaphore when the button is pressed and the LED_Task should toggle LED2.
TimerTask gives the timerSemaphore every second and the LED_Task should print the time in seconds when this semaphore is taken.

```
Starting LED task
Starting button task
Starting timer task
Time: 001 seconds
Time: 002 seconds
Time: 003 seconds
Time: 004 seconds
Time: 005 seconds
Time: 006 seconds
Button pressed, giving semaphore...
Semaphore taken, Toggling LED...
Time: 007 seconds
Time: 008 seconds
Button pressed, giving semaphore...
Semaphore taken, Toggling LED...
Time: 009 seconds
Time: 010 seconds
```

Console | Problems | Executables | Debugger Console | Memory | FreeRTOS Task List ×

Name	Priority (B...	Start of S...	Top of St...	State	Event Ob...	Stack Usa...	Run Time...
Button Task	5/5	0x20001...	0x20001...	DELAYED		312B / 8...	2%
→ IDLE	0/0	0x20000...	0x20000...	RUNNING		88B / 51...	98%
LED Task	6/6	0x20001...	0x20001...	DELAYED		504B / 8...	<1%
Timer Task	4/4	0x20001...	0x20001...	DELAYED		312B / 8...	<1%
Tmr Svc	6/6	0x20000...	0x20000...	BLOCKED	TmrQ	168B / 1...	<1%