Assignment 7

The question

Create a RTOS Application
Create Task1 and Task2 with their proper TaskHandles.
Pass the following parameters:
Task 1 (handle)----Pass to ----> Task1 as parameter
Task 2 (handle)----Pass to ----> Task2 as parameter
Inside Task1:
Blink LED1 and LED2 at every 3ms for 10 times, after that delete itself.
Inside Task2:
Blink LED3 and LED4 at every 5ms for 5 times, after that delete itself.

The following are the task functions defined for the same purpose:

```
void Task1(void *T1)
{
	uint16_t count = 10;
	for(;;)
	{
		HAL_GPIO_TogglePin(GPIOD, GPIO_PIN_12 | GPIO_PIN_13);
		HAL_Delay(3);
		if(count == 1)
		{
			vTaskDelete(NULL);
		}
		count--;
	}
}

void Task2(void *T2)
{
	uint16_t count = 5;
	for(;;)
	{
		HAL_GPIO_TogglePin(GPIOD, GPIO_PIN_14 | GPIO_PIN_15);
		HAL_Delay(5);
		if(count == 1)
		{
			vTaskDelete(NULL);
		}
		count--;
	}
}
```

The task handles are created, before the creation of the tasks, shown below:

```
TaskHandle_t T1;
TaskHandle_t T2;
xTaskCreate(Task1, "Task1", 200, NULL, 1, &T1);
xTaskCreate(Task2, "Task2", 200, NULL, 1, &T2);
```

The vTaskDelete() function:

Deletes an instance of a task that was previously created using a call to xTaskCreate() or xTaskCreateStatic().

Deleted tasks no longer exist so cannot enter the Running state.

Do not attempt to use a task handle to reference a task that has been deleted. When a task is deleted, it is the responsibility of the idle task to free the memory that had been used to hold the deleted task's stack and data structures (task control block). Therefore, if an application makes use of the vTaskDelete() API function, it is vital that the application also ensures the idle task is not starved of processing time (the idle task must be allocated time in the Running state).

The Trace of the code is shown below.