Assignment #3

NAME: Tejas Shingala (C0732079)
Jaivik Patel (C0732075)
Akshit Patel (C0724035)

By using queue function in FreeRTOS, In this assignment create sender task for send LED color to queue in random time interval between 0 to 1, and sender task stop transmitting after 4 seconds for 30 seconds. Receiver task take data from queue and blow LED after every 2 seconds.

In this assignment we create 3 main task , one is for sender, one for receiver and last one is for timer task. In sender task, we select color to send and then take random number and manage limit between 0 to 1 by dividing random number by it's maximum number which is RAND_MAX, then take delay for 4 second then stop sending data for 30 seconds by using another delay function for 30 seconds.
Then in receiver task, we take data from queue and so, LED light will be blow for 2 seconds and change color after 2 seconds and repeat the same.

Program:

```
#include "board.h"
#include "FreeRTOS.h"
#include "task.h"
#include "queue.h"
#include "croutine.h"



static void prvSetupHardware(void)
{
	SystemCoreClockUpdate();
	Board_Init();

	/* Initial LED0 state is off */
	Board_LED_Set(0, false);
	Board_LED_Set(1, false);
	Board_LED_Set(2, false);
}
int sendok;
/* LED1 toggle thread */
static void vSenderTask(void *pvParameters)
{
	int32_t lValueToSend;
	int32_t xStatus;
	lValueToSend = ( int32_t ) pvParameters;
	int sendok;
	float RAND_MAX;

	while (1)
	{


		if (sendok == 1);
		{
			Board_LED_Set(0, false);
			float i = rand ();
			float r = ((float)i)/((float)RAND_MAX);
			vTaskDelay(configTICK_RATE_HZ);
		}
}
}
```

```
static void vReceiverTask( void *pvParameters )
{
	int32_t lReceivedValue;
	int32_t xStatus;
	const TickType_t xTicksToWait = pdMS_TO_TICKS( 100 );

	while (1)
	{
		if( uxQueueMessagesWaiting( xQueue ) != 0 )
		{
			Board_LED_Set(0, false);
		}
	xStatus = xQueueReceive( xQueue, &lReceivedValue, xTicksToWait );
	}
}

void vTimerTask(void *pvParameters)
{
	while(1)
	{
		if (sendok == 1)
		{
			vTaskDelay(40000);
		}
		else if (sendok == 0)
		{
			vTaskDelay(300000);
		}
	}
}


int main( void )
{
	xQueue = xQueueCreate( 5, sizeof( int32_t ) );
	if( xQueue != NULL )
	{
```

```
		while (1);

		xTaskCreate( vSenderTask, "Sender1", 1000, ( void * ) 100, 1, NULL );
		xTaskCreate( vSenderTask, "Sender2", 1000, ( void * ) 200, 1, NULL );

		xTaskCreate( vReceiverTask, "Receiver", 1000, NULL, 2, NULL );
		vTaskStartScheduler();

w
	}
}
/* UART (or output) thread */
static void vUARTTask(void *pvParameters) {
	int tickCnt = 0;

	while (1) {
		DEBUGOUT("Tick: %d\r\n", tickCnt);
		tickCnt++;

		/* About a 1s delay here */
		vTaskDelay(configTICK_RATE_HZ);

	}
}
```