[1]BN	Section	Student ID	اسم الطالب
10	**1**	**9220061**	**احمد عصام سيد**
37	**1**	**9220205**	**بلال رمضان حلمي**

1 System Design

1.1 Pseudocode:

1. **System Initialization**
   - Define constants: queue size and random timers bounds (lower and upper for each state).
   - Define variables: state index, sender times, average times, sent/blocked messages counters and received messages counter.
   - Create semaphores (one for each sender and one for the receiver).
   - Create timers (one for each sender and one for the receiver).
   - Create a message queue.
2. **Sender Task Function**
   - Loop continuously:
     - Generate message content ("time is xyz").
     - Wait on sender semaphore which is released by a random-time timer.
     - If queue is not full:
       - Send the generated message to the queue.
       - Update successfully sent messages counter.
     - Else:
       - Update blocked messages counter.
     - Calculate new random delay within bounds for current state.
     - Update sender timer period with new delay.
     - Restart sender timer.
3. **Receiver Task Function**
   - Loop continuously:
     - Wait on receiver semaphore which is released by a constant-time timer of 100 ms
     - Receive message from the queue.
     - If message received:
       - Increment received messages counter.
     - Restart receiver timer.
4. **Timers Callback Functions**
   - Sender Timer Callback: Release corresponding sender semaphore.
   - Receiver Timer Callback
     - Release receiver semaphore.
     - If received messages reach threshold (1000 messages):
       - Calculate average time per sender
       - Print statistics for each sender (total sent/blocked messages).
       - Reset counters (received messages, sent/blocked messages).
       - Reset message queue.
       - Update state index (refers to the random-time bounds).
       - If state index reaches the end:
         - Print "Game Over".
         - End task scheduler.
         - Exit program.
5. **Main Function**
   - Initialize system (as described in the 1st point).
   - Start sender and receiver timers.
   - Create sender and receiver tasks.
   - Start task scheduler.
   - If Queue creation failed:
     - Print a message.

1.2 Figures

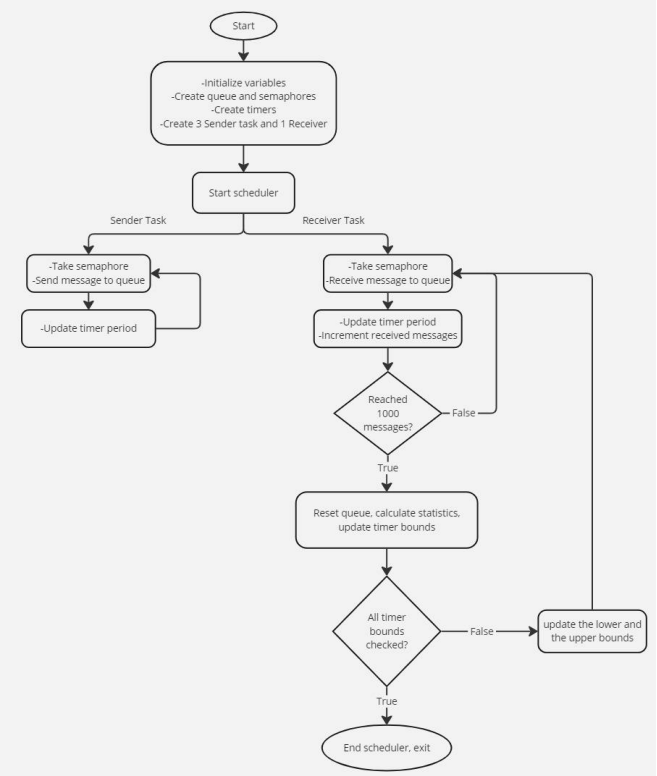

Figure 1 : Program sequence flowchart

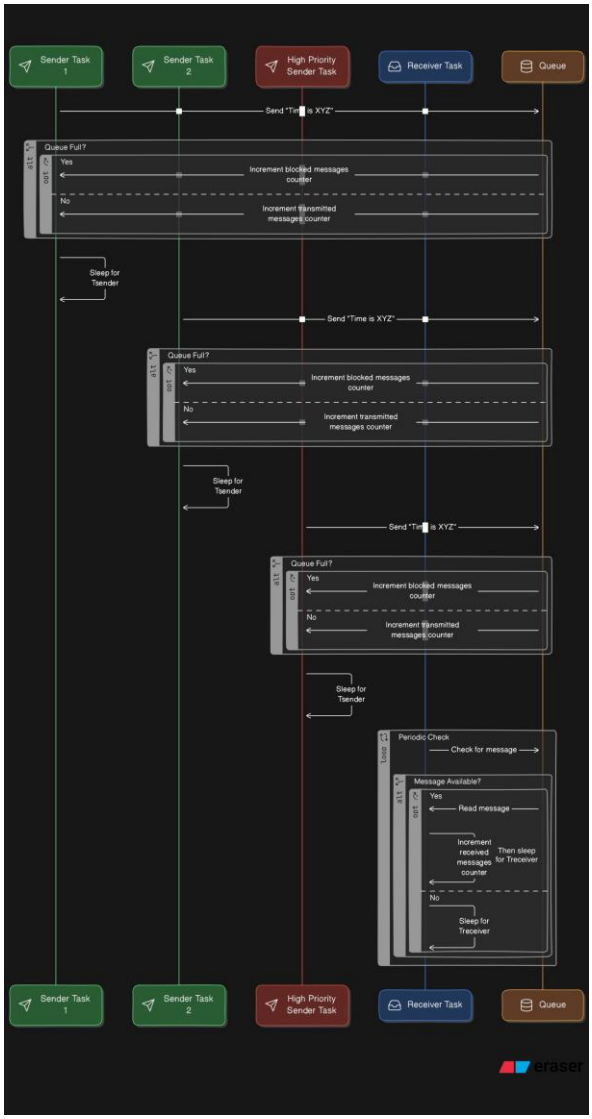

Figure 2: Tasks interaction diagram

1.3 Code Snippets

```
//This line Creates the message
snprintf(ValueToSend,sizeof(ValueToSend)," Time is %dms",xTaskGetTickCount());

//This line computes random number in a certain range
int newperiod = (rand() % (UpperBound[StateIndex] - LowerBound[StateIndex] + 1)) +
LowerBound[StateIndex];

//This line changes the timer period
xTimerChangePeriod(autoReloadTimer[j], pdMS_TO_TICKS(newperiod), 0);

//This line gets the timer ID
int k = (int)pvTimerGetTimerID(xTimer);

//This line terminates the program
exit(0);
```

2 Results and Discussion

Average Time	successfully sent	blocked	Total sent
98	1002	2034	3036
139	1002	1146	2148
181	1002	654	1656
221	1002	353	1355
260	1001	151	1152
302	1000	9	1009

Table 1: Queue 3

Average Time	successfully sent	blocked	Total sent
98	1009	2026	3133
139	1009	1137	2285
181	1009	647	1837
221	1009	347	1577
260	1009	145	1414
302	1001	0	1303

Table 2: Queue 10

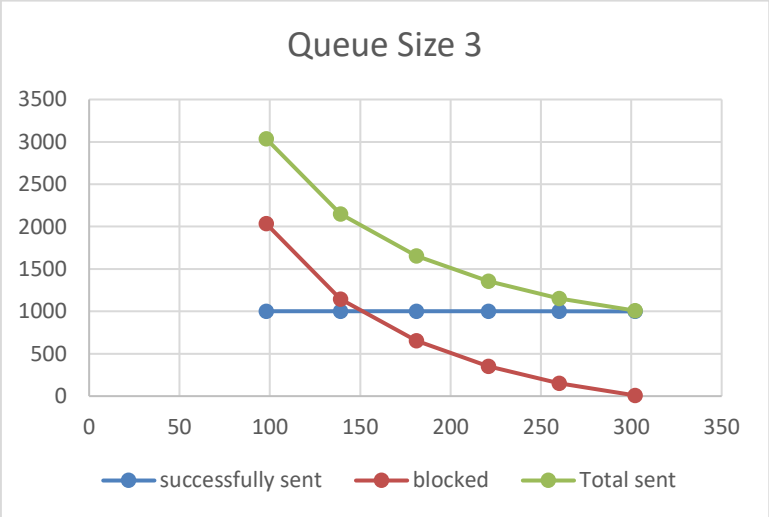

Figure 3: Queue 3

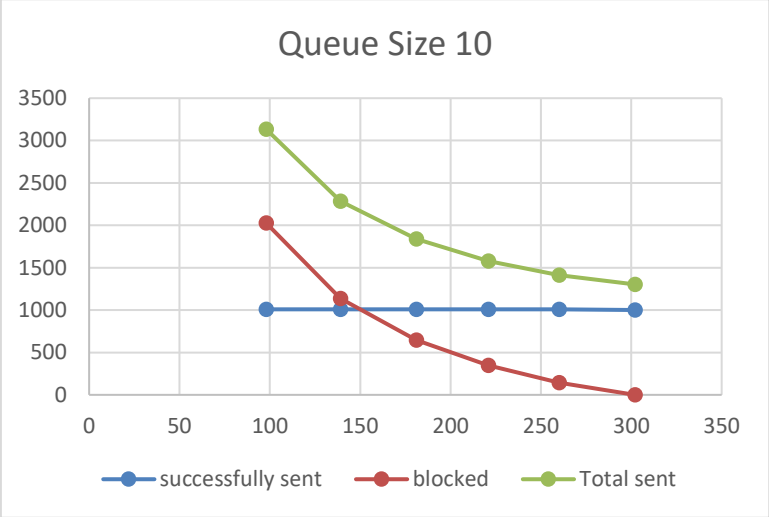

Figure 4: Queue 10

The gap between Total Sent Messages and Received Messages is the blocked messages, which decay as average Tsender increases because the receiver task gains time advantage over sender tasks. And another gap which depends on the queue size and it's between 0 and (queue size – 1) due to the presence of other messages in the queue when the Receiver task receives the last message.

The blocked messages differs a little as the queue size changes from 3 to 10 as the difference is small relative to the number of required messages.

Average Time	Sender 1	Sender 2	Sender 3
98	308	309	385
139	331	365	306
181	324	337	341
221	335	353	314
260	341	325	335
302	339	330	331

Table 3: Queue 3 (Successfully Sent of each sender)

Average Time	Sender 1	Sender 2	Sender 3
98	330	356	323
139	340	322	347
181	348	321	340
221	343	350	316
260	330	347	332
302	329	336	336

Table 4: Queue 10 (Successfully Sent of each sender)

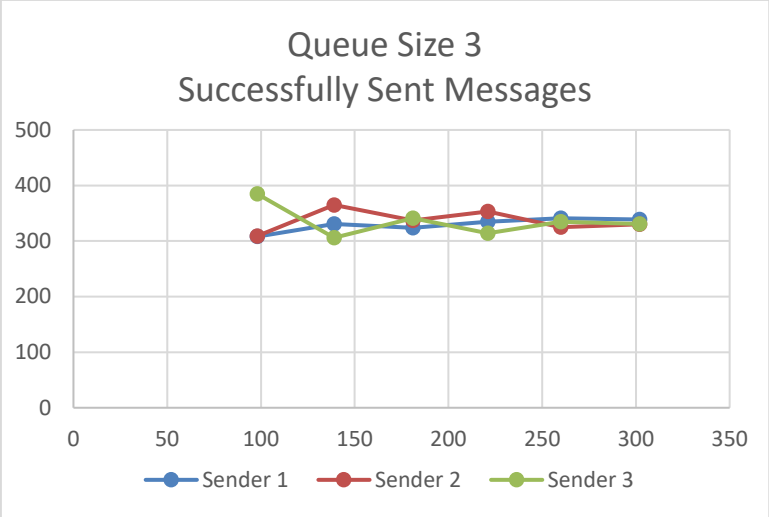

Figure 5: Queue 3 (Successfully Sent Messages of each sender)

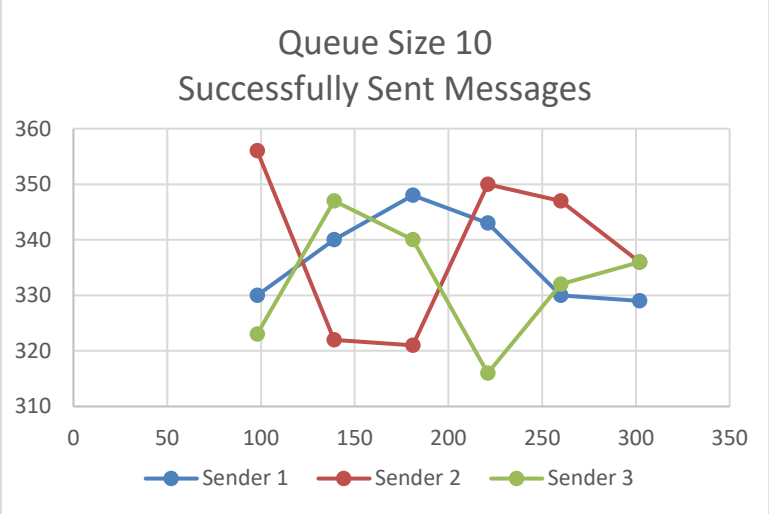

Figure 6: Queue 10 (Successfully Sent Messages of each sender)

As we see the number of successfully sent messages for each task is totally random and doesn't depend on the average Tsender nor the queue size, as the sleep period is random and the higher priority task doesn't take any advantage over the low priority ones unless they wake up at the same instant.

Average Time	Sender 1	Sender 2	Sender 3
98	693	718	623
139	387	351	408
181	229	213	212
221	118	101	134
260	44	60	47
302	1	5	3

Table 5: Queue 3 (Blocked Messages of each sender)

Average Time	Sender 1	Sender 2	Sender 3
98	675	650	701
139	372	396	369
181	205	232	210
221	110	101	136
260	57	38	50
302	0	0	0

Table 6: Queue 10 (Blocked Messages of each sender)

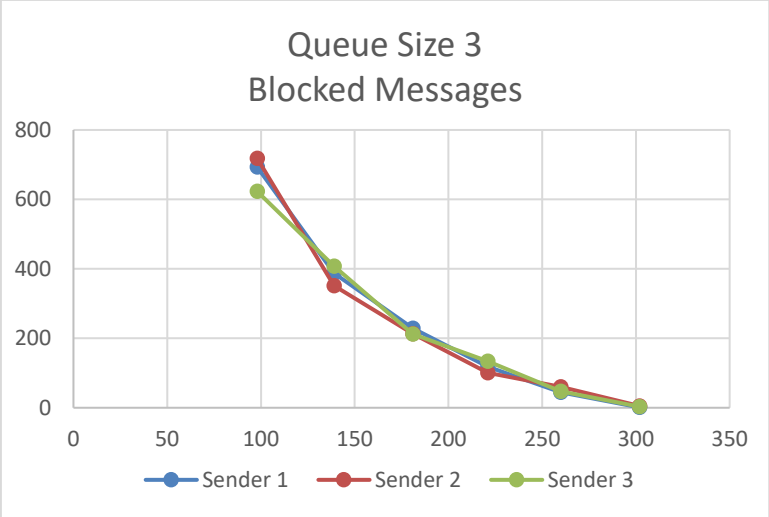

Figure 7: Queue 3 (Blocked Messages of each sender)

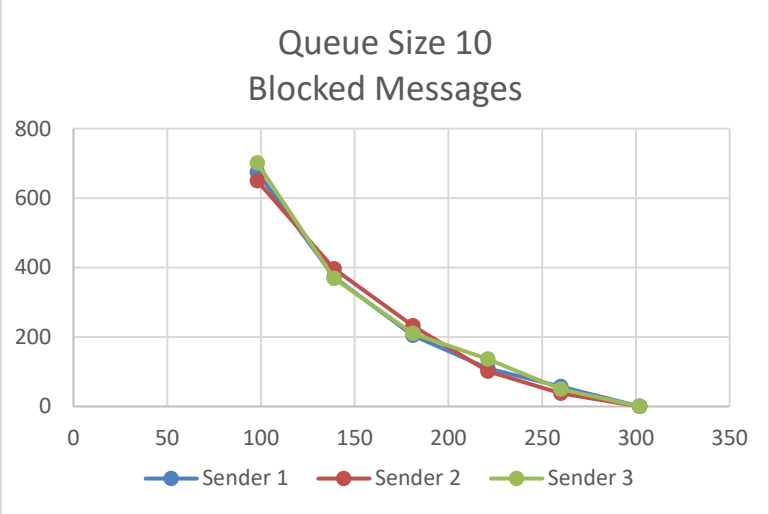

Figure 8 : Queue 10 (Blocked Messages of each sender)

Since the number of blocked messages per task doesn't depend on the priority so we conclude that most of the blocked messages are due to the queue is full at that time, or the receiver task is using the queue at the same instant not collision between two sender tasks.

3 References

[1] R. Barry, "FreeRTOS - Market leading RTOS (Real Time Operating System) for embedded systems with Internet of Things extensions.," [Online]. Available: https://www.freertos.org/index.html.