

BN	Section	Student ID	اسم الطالب
5	2	9220276	خالد أحمد حامد عبدالعزيز
15	2	9220349	سارة عبدالعاطي إبراهيم عبدالعاطي

1 System Design

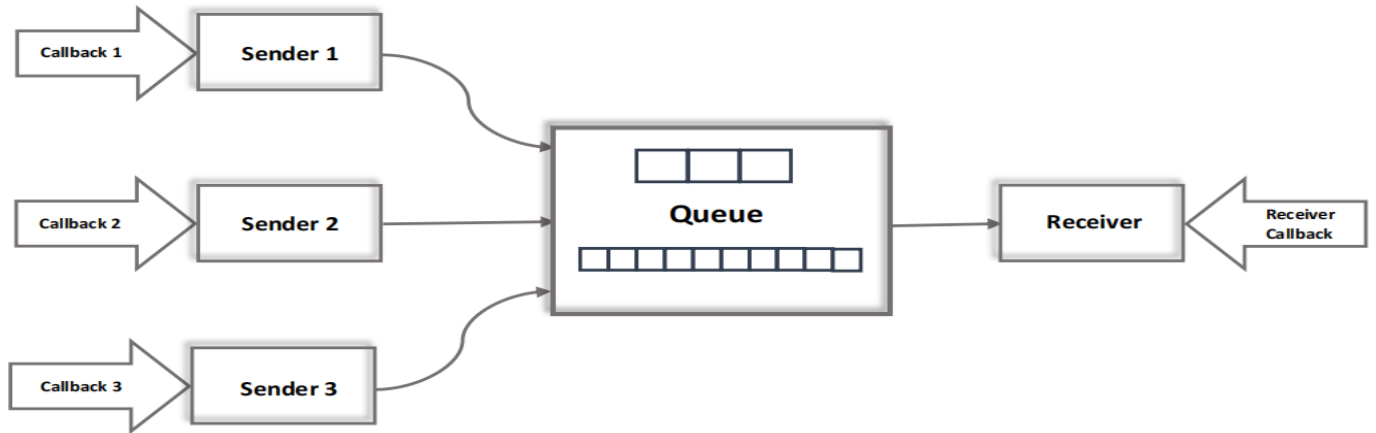


Figure1 : data structure design

1.1 Flow charts: It illustrate the sequence of every function in the code and their relation with each other.

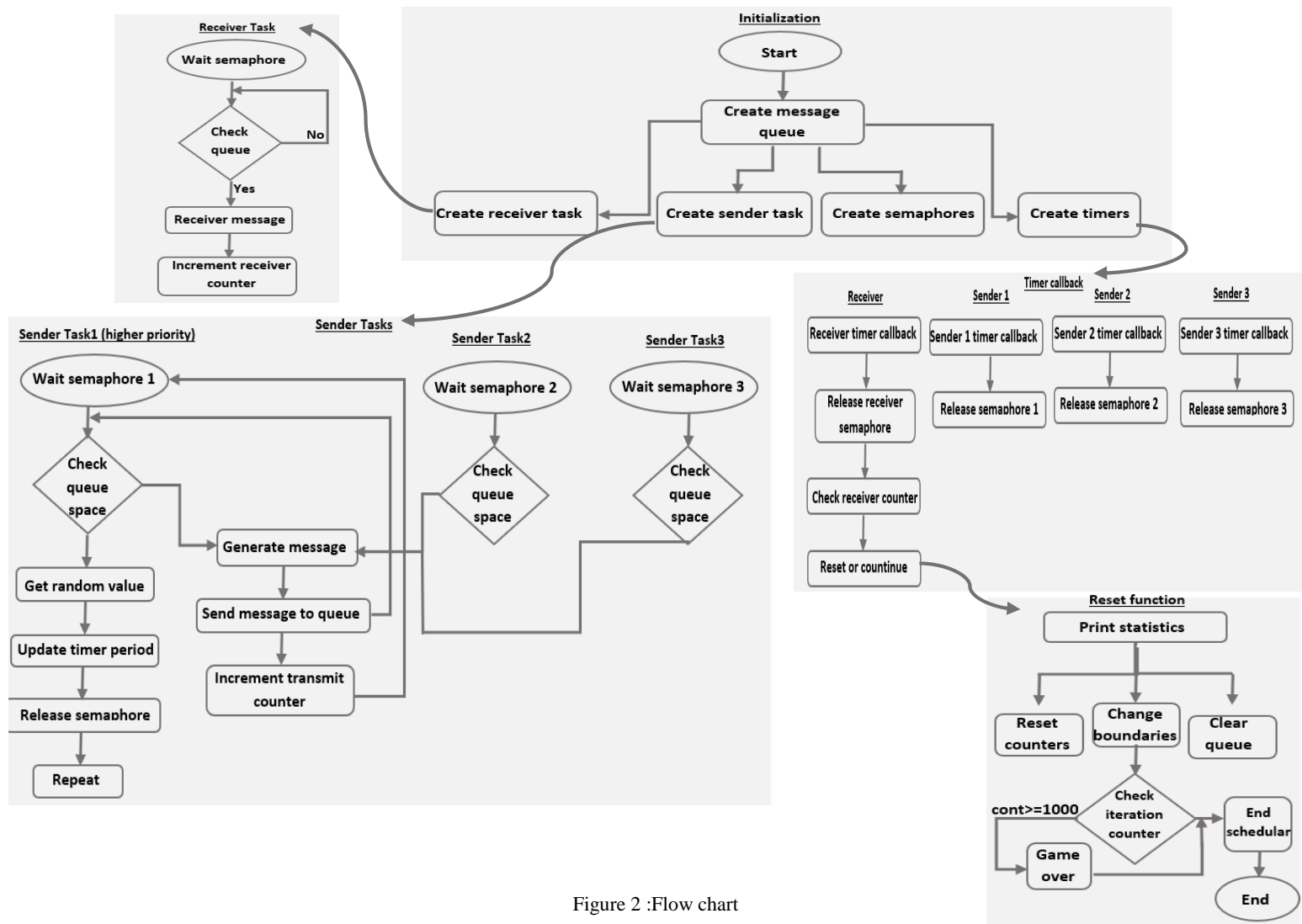


Figure 2 :Flow chart

1.2 Main Structures:

1. **Task Handles:** Handles are used to manage the sender and receiver task
2. **Semaphores:** Semaphores are used to synchronize the sender and receiver tasks with their respective timers.
3. **Queue:** A queue has size of 3 or 10, Is created to hold the messages sent by the sender tasks
4. **Timers:** Timers are used to periodically release the semaphores, which in turn trigger the sender and receiver tasks
5. **Functions:**
 - Reset Function: Resets counters, prints statistics, and handles end-of-game conditions.
 - Random Number Generator: Generates random numbers using uniform distribution.
 - Initialization Function: Initializes the tasks, queue, semaphores, and timers.
6. **Tasks:**
 - Sender Tasks: There are 3 sender tasks, each sender task waits for its semaphore, checks if queue has space, and sends a message or increments the blocked counter. It then changes its timer period based on a random value
 - Receiver Tasks: The receiver task waits for its semaphore and receives messages from the queue.
7. **Timer Callback Functions:**
 - 3 Sender Timer Callbacks each callback function releases semaphore for the corresponding sender task.
 - Receiver Timer Callback releases semaphore for receiver task and calls Reset function if 1000 messages have been received.

2 Code Snippets

Sender Task	Receiver Task
<p>Manages message sending for senders. It waits for semaphore, after getting the semaphore, it attempts to send a message with the current system tick to the queue. If successful, it increments the success counter, if blocked, it increments the blocked counter. After attempting to send, it sets a new random period for its timer</p>	<p>Waits for its semaphore released by its timer, When the semaphore is acquired, it attempts to receive messages from the queue. If a message is successfully received, it increments the receive counter. If the receive counter reaches the limit (1000 messages), it triggers the Reset function.</p>
<pre>void Sender1Task(void* pvParameters) { while (1) { xSemaphoreTake(SenderSem[0], portMAX_DELAY); if (uxQueueSpacesAvailable(Queue) > 0) { char SenderMessage[20]; sprintf(SenderMessage, "Time is %lu", xTaskGetTickCount()); if (xQueueSendToBack(Queue, SenderMessage, 0) == pdPASS) { MessageTransmittedCounter[0]++; } } else { MessageBlockedCounter[0]++; } int SenderRandomValue = RandomNumbersUniformDist(); // Convert SenderRandomValue to ticks SenderOnePeriod = pdMS_TO_TICKS(SenderRandomValue); xTimerChangePeriod(SenderOneTimer, SenderOnePeriod, 0); } }</pre>	<pre>void ReceiverTask(void* pvParameters) { while(1) { //Take Semaphore released bt callback xSemaphoreTake(ReceiverSem, portMAX_DELAY); //Checks if queue has a free space if (uxQueueMessagesWaiting(Queue) > 0) { char ReceiverString[20]; // Send the string to the queue if(xQueueReceive(Queue, ReceiverString, 0) == pdPASS) { MessageReceivedCounter++; } } } }</pre>

3 Results and Discussion

3.1 Number of total sent & blocked messages

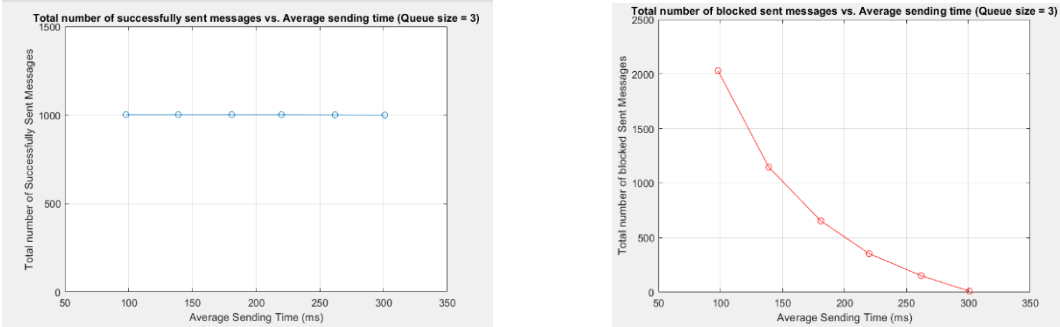
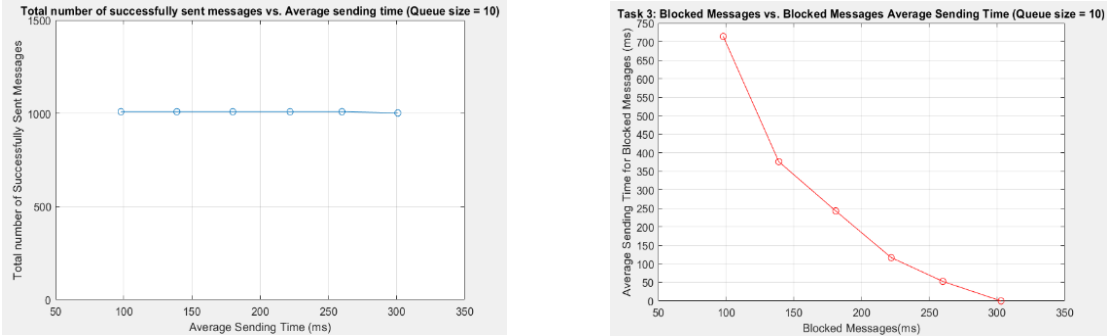
Queue of size 3	 <p style="text-align: center;">Figure 3 :total number of sent VS blocked messages</p>
Queue of size 10	 <p style="text-align: center;">Figure 4 :total number of sent VS blocked messages</p>

Table 1 :Sent & block total statistics

Explanation of the gap:

The gap between sent and received messages is due to difference in frequencies of sender and receiver operations (They aren't perfectly synchronized), Messages are being sent by sender tasks more quickly than the receiver can process them. As a result, there are more messages sent than received in the queue, which causes an accumulation of messages.

A recommended solution: Optimizing synchronization and balancing the rates of sender and receiver tasks.

Effect of increasing the queue size:

The increase in queue size from 3 to 10 reduces the frequency of blocked messages, generally improves system efficiency and enhances the flow and overall performance.

Size 3	Size 10
More blocked messages	Fewer blocked messages
Lower average sending times	Higher average sending times
Senders wait often	Smoother operation

Table 2 :Effect of increasing the queue size

3.2 Statistics per sender task

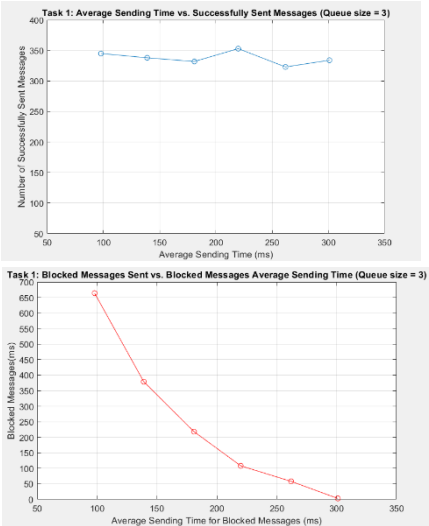
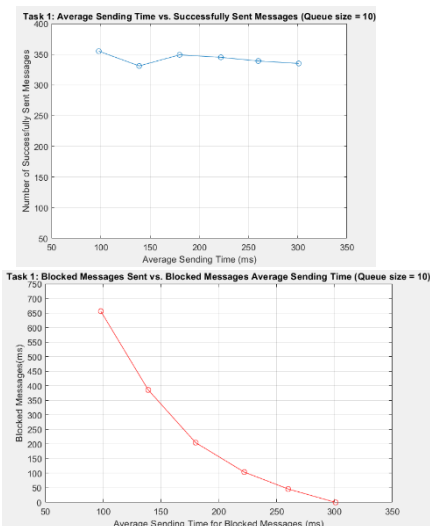
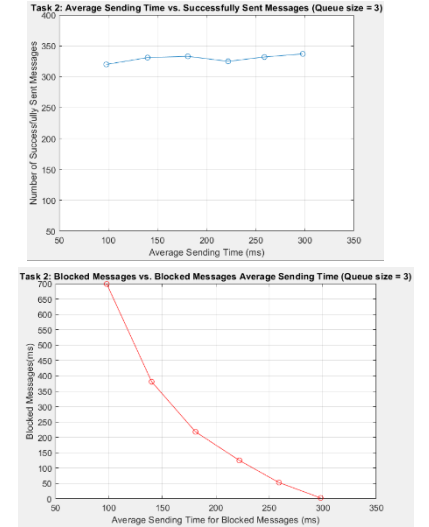
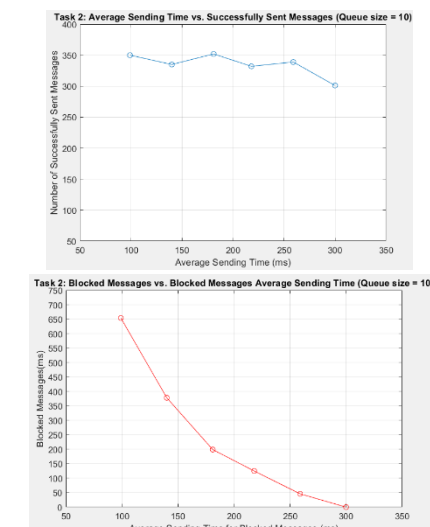
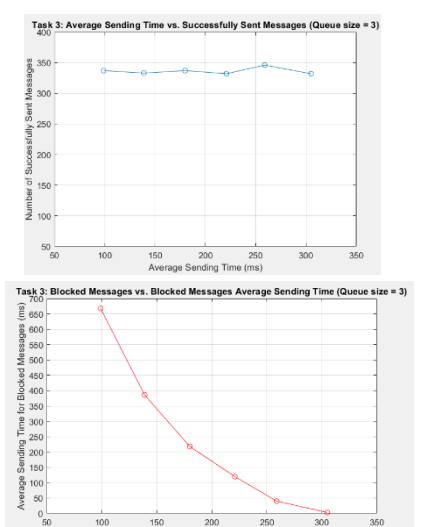
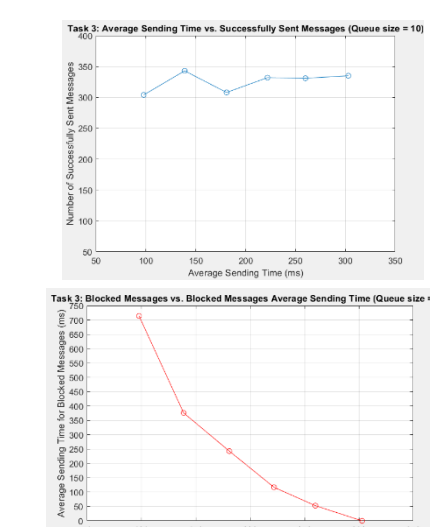
Queue	Size 3	Size 10
<u>Higher priority sender</u>	 <p>Figure 5 :Higher priority sender task</p>	 <p>Figure 6 :Higher priority sender task</p>
<u>Lower priority sender 1</u>	 <p>Figure 7 : Lower priority sender task 1</p>	 <p>Figure 8 :Lower priority sender task 1</p>
<u>Lower priority sender 2</u>	 <p>Figure 9 :Lower priority sender task 2</p>	 <p>Figure 10 :Lower priority sender task 2</p>

Table 3 :Sender tasks statistics

3.3 Total messages statistics for each iteration

Queue of size 3	Queue of size 10
<p>-----</p> <p>Iteration number : 1 Total number of successfully sent messages :1002 Total number of blocked messages :2031</p> <p>-----</p> <p>Iteration number : 2 Total number of successfully sent messages :1002 Total number of blocked messages :1146</p> <p>-----</p> <p>Iteration number : 3 Total number of successfully sent messages :1002 Total number of blocked messages :654</p> <p>-----</p> <p>Iteration number : 4 Total number of successfully sent messages :1002 Total number of blocked messages :353</p> <p>-----</p> <p>Iteration number : 5 Total number of successfully sent messages :1002 Total number of blocked messages :151</p> <p>-----</p> <p>Iteration number : 6 Total number of successfully sent messages :1001 Total number of blocked messages :9</p> <p>Figure 11 :Total messages statistics (Queue size=3)</p>	<p>-----</p> <p>Iteration number : 1 Total number of successfully sent messages :1009 Total number of blocked messages :2024</p> <p>-----</p> <p>Iteration number : 2 Total number of successfully sent messages :1009 Total number of blocked messages :1139</p> <p>-----</p> <p>Iteration number : 3 Total number of successfully sent messages :1009 Total number of blocked messages :647</p> <p>-----</p> <p>Iteration number : 4 Total number of successfully sent messages :1009 Total number of blocked messages :346</p> <p>-----</p> <p>Iteration number : 5 Total number of successfully sent messages :1009 Total number of blocked messages :144</p> <p>-----</p> <p>Iteration number : 6 Total number of successfully sent messages :1001 Total number of blocked messages :0</p> <p>Figure 12 :Total messages statistics (Queue size=10)</p>

Table 4 :Total number of successfully sent messages and blocked messages for each iteration

3.4 Discussion

We can evaluate the approximate number of blocked messages using the following equation:

$$B = (S - R) \times \left(T - \frac{L}{R}\right)$$

where: B : represents the number of blocked messages, S : denotes the average sending rate, R : stands for the average receiving rate ($R=0.01$ messages/ms), T : represents the total time of operation ($T=100000$ ms), L : corresponds to the length of the message queue (either 3 or 10).

since there are three sender tasks that send three messages at the same average sending rate, we can calculate S as:

$$S = \frac{3}{\frac{(\text{Upper Bound} + \text{Lower Bound})}{2}}$$

Iteration Boundaries: Upper Bounds = [150,200,250,300,350,400], Lower Bounds = [50,80,110,140,170,200].

Let's calculate B for both queue sizes $L=3$ & $L=10$, for an example evaluate for first iteration:

Upper Bound =150, Lower Bound=50, average sending interval: $\frac{150+50}{2} = 100$ ms, $S = \frac{3}{100} = .03$ Message/ms,

$B = (0.03 - 0.01) \times \left(100000 - \frac{L}{0.03}\right) = 1994$ for $L=3$ and for $L=10$ $B = 1980$.

Predicted Blocked Messages Table:

Iteration	1	2	3	4	5	6
Queue Size L=3	1994	1137	668	359	150	0
Queue Size L=10	1980	1129	663	356	149	0

Table 5 :Total number of successfully sent messages and blocked messages for each iteration (Queue size= 3&10)