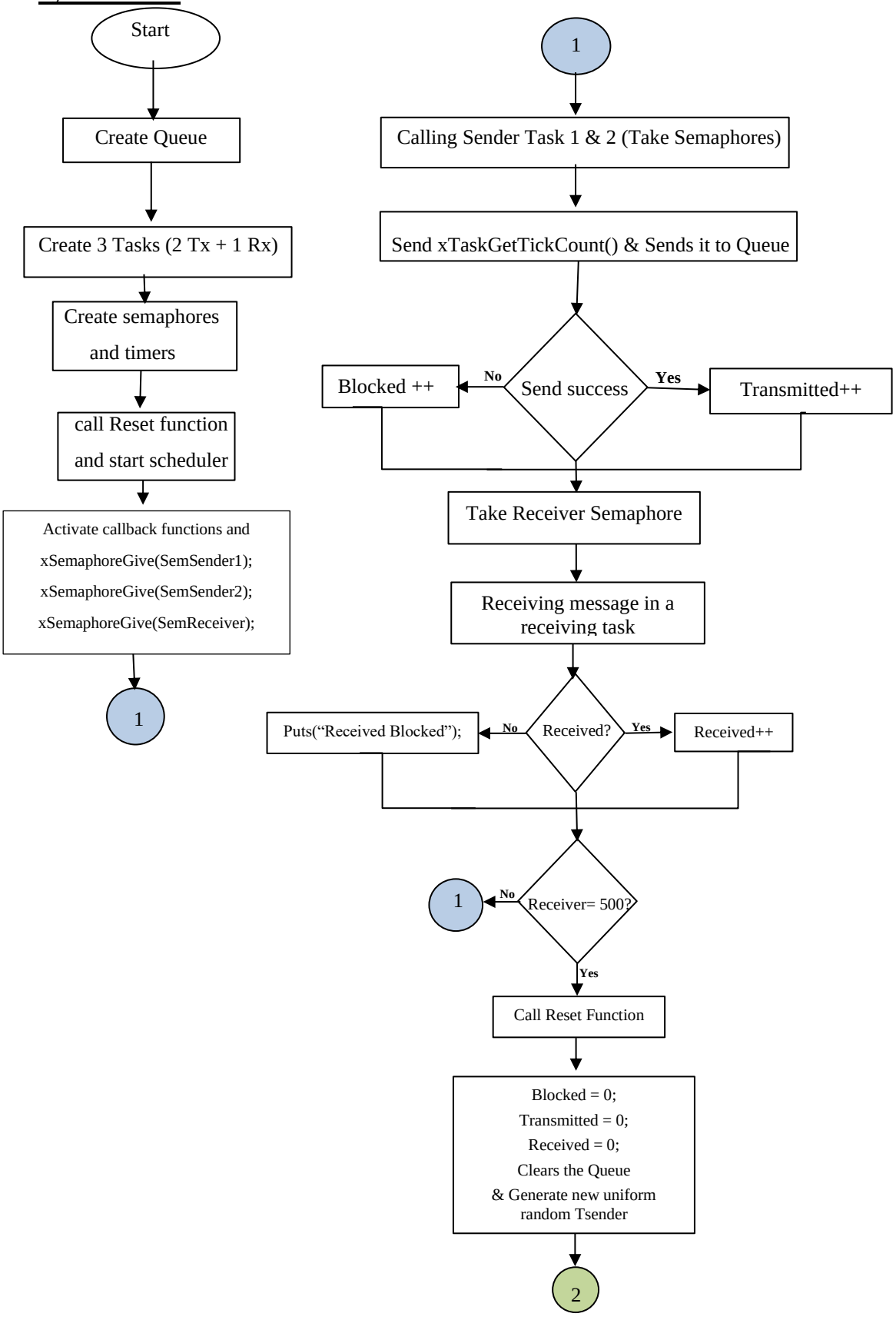
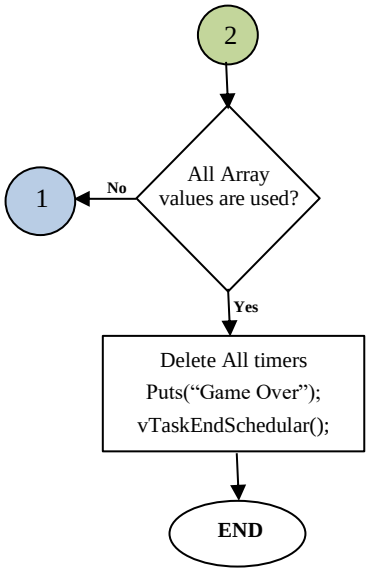
1 System Design

We have in our system we have 3 tasks that communicate with each other. Including 2 Sender Tasks and 1 Receiver Task, we created also for each task its own semaphore and its own timer. Where sender/receiver tasks take semaphore inside their own tasks and the call-back function of timer give the semaphore back. The basis of our communication system is that we have a queue that takes values from sender and give those values back to the receiver.

System Flow:





Reset Function:

```
void Reset_fun()
{  static int Reset_call_count=0;
   int random_no=0;
   int lower_bound[]={50,80,110,140,170,200};
   int upper_bound[]={150,200,250,300,350,400};
   srand(time(NULL));
      //below line will generate a random no bet upper and lower bounds every time Reset_fun is called
   random_no= (rand() % (upper_bound[Reset_call_count]-lower_bound[Reset_call_count]+1) )
        +lower_bound[Reset_call_count];
   if(received==MaxReceived)
   { printf("\n\nReceived 500 !!!!!!!! for the %d Time\n",Reset_call_count);
     printf("The number of successfully send messages = %d\n",transmitted);
     printf("The Total Number of Blocked messages = %d\n",blocked);
     transmitted=0;                      //reset total no. of transmitted messages
     blocked=0;                          //reset total no. of blocked messages
     received=0;                         //reset total no. of received messages
     xQueueReset(Queue);} //clears queue
   if (Reset_call_count==6)                 //if all values of array are used , destroy program
   { xTimerDelete(xTimer1,(TickType_t) 0);
     xTimerDelete(xTimer2,(TickType_t) 0);
     xTimerDelete(xTimer3,(TickType_t) 0);
     printf("**************Game Over**************\r\n");
     vTaskEndScheduler();
     exit(0);
     return;}
   xTimerChangePeriod(xTimer1,random_no,(TickType_t) 0);
   xTimerChangePeriod(xTimer2,random_no,(TickType_t) 0);
   printf("------Sender Time now is %d -------\n",random_no);
   Reset_call_count++; }
```

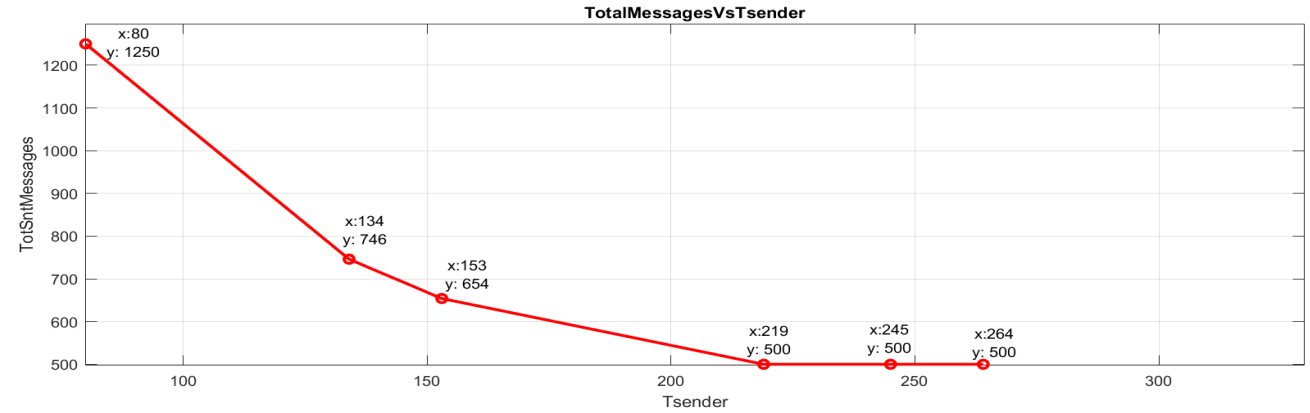
2 Results and Discussion

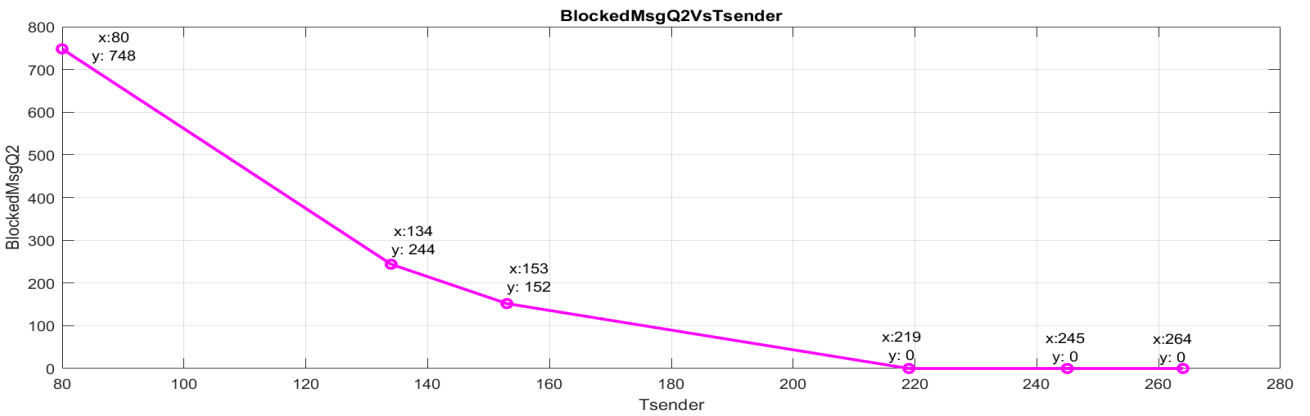
	Queue Size = 2		Queue Size = 20	
Tsender	Sent Messages	Blocked Messages	Sent Messages	Blocked Messages
80	502	748	520	730
134	502	244	520	226
153	502	152	520	134
219	500	0	500	0
245	500	0	500	0
264	500	0	500	0

****Note**

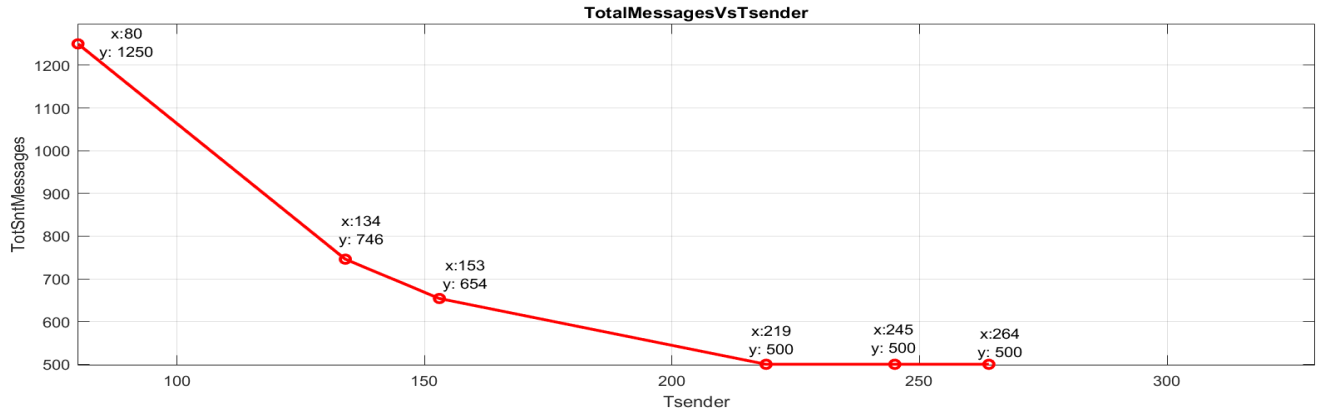
Since *Tsender* is a uniform random number, we got these random numbers from the queue of size 2 and used them in the queue size 20 to notice the difference of changing the queue size on the program in that case

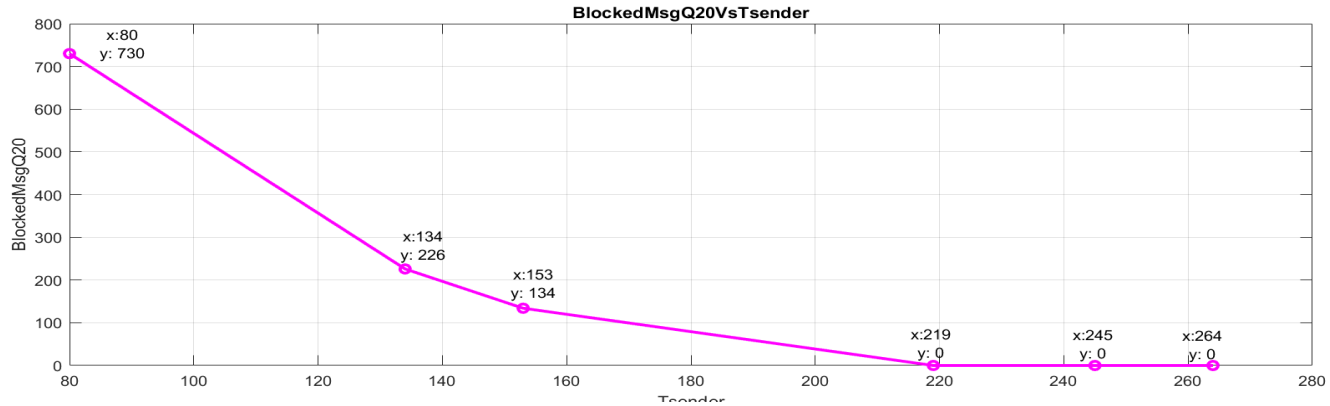
Queue Size 2

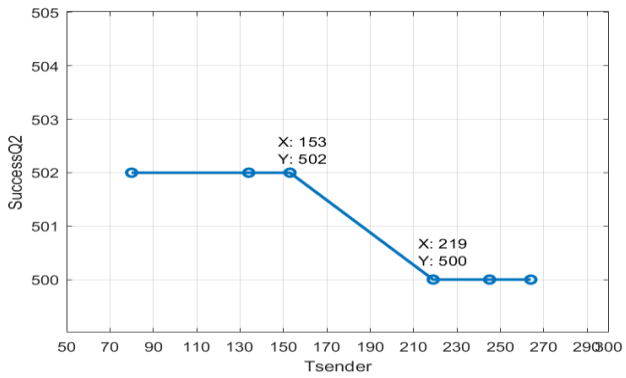


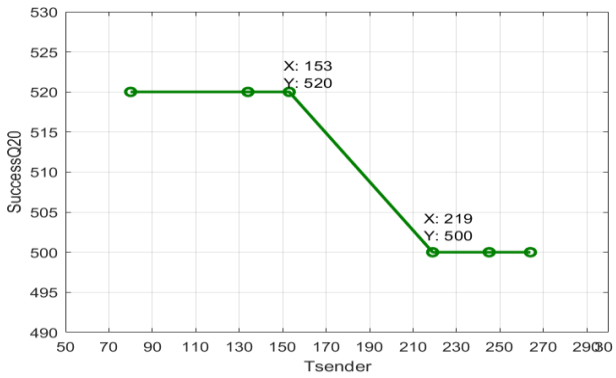


Queue Size 20









3 Observations:

It is noticed that if Tsender is bigger than Treceiver, then there are no blocked messages, as the receiver has less time to read from the queue while sender takes longer time to send. While if Tsender is smaller than Treceiver then there must be a number of blocked messages that exist in the system.

It is also noticed that in case of Tsender is smaller than Treceiver, as Tsender decrease, the number of blocked messages increase. As noticed from above table at Tsender=80, number of blocked messages were 748, if Tsender increased to be 134, it is observed that number of blocked messages become 245. Which proves our observations.

If we increased the size of queue to be equal 20, the total number of sent messages increase by 18, and total number of blocked messages decreases by 18, that's because we were using queue of size 2 and when we changed to a queue of size 20 it resulted in 18 new extra places in queue, which resulted in increasing send message by 18 and decreasing the number of blocked messages by 18. We also notice that when the Tsender is bigger than Treceiver the number of the sent messages are 500. And when the Tsender is less than Treceiver there will be waiting in the queue so when we reach 500 received messages there will always be (500 + the one extra sent message + the messages that are still waiting in the queue) that's why we get 502 sent messages in case of queue size = 2 and get 520 sent messages in case of queue size = 20.

Sender Task1: (it is the exact same code as SenderTask2)

```
void SenderTask1(void *p)
{ char tx[10];              //string to store in it
  TickType_t xTimeNow;
  while(1){
    if(xSemaphoreTake(SemSender1,portMAX_DELAY) )    //take semaphore of senderTask1 (SemSender1)
    {
      xTimeNow =xTaskGetTickCount();                //obtain current tick count
      sprintf(tx,"\nTime is : %d\n", xTimeNow);     //store the word "Time is : " and the current
                                                    //tick count inside the string tx
    if(!xQueueSend(Queue,tx,0))          //we send the string tx to the queue
    {puts("Sender1 Blocked");            //if sender1 failed to send we print "Sender1 Blocked"
    blocked++;}
    else
    { printf("Sender1 *Sends*\n");      //if sender1 send successfully we print "Sender1 *Sends*"
      transmitted++;}                    //and we increment the total number of transmitted messages
    }
  }
}
```

SenderTask1 Timer Call-back function:

```
static void Sender1_TimerCallback( TimerHandle_t xTimer )
{
xSemaphoreGive(SemSender1);        //give semaphore back
}
```

Receiver Task:

```
void ReceiverTask(void*p)
{   char rx[10];
    while(1)
    {  if(xSemaphoreTake(SemReceiver,portMAX_DELAY))    //take receiver semaphore
       { if( xQueueReceive(Queue,rx,100) )              //receive from queue
          { printf(" %s\n\n",rx);       //if successfull receive , it prints the Time sended by sendertasks
          received++; }                 //increments total number of received messages
          else
             puts("Receiver Blocked");     //if not recieved successfully , prints "Receiver Blocked"
       }
    }
}
```

ReceiverTask Timer Call-back function:

```
static void Receiver_TimerCallback( TimerHandle_t xTimer )
{
  xSemaphoreGive(SemReceiver);  //give semaphore
  if(received==MaxReceived)  {Reset_fun();}        //calls Reset_function
}
```