Primer 1. Napraviti tri zadatka:
Ime: Sender1 – Prioritet: osPriorityLow – Parametar 100 – Stack:128 – Entry function: sender
Ime: Sender2 – Prioritet: osPriorityLow – Parametar 200 – Stack:128 – Entry function: sender
Ime: Receiver – Prioritet: osPriorityHigh – Stack:128 – Entry function: receiver
Napraviti red:
Ime: MyQueue1 – Size: 5 – ItemSize: uint32_t

```
void sender(void *argument)
{
  uint32_t value_to_send = (uint32_t) argument;
  osStatus_t status;

  const char txt[] = "Slanje nije uspelo!\n";
  for(;;)
  {
       status =  osMessageQueuePut(myQueue01Handle, &value_to_send, NULL, 0);
       if(status != osOK){
            HAL_UART_Transmit(&huart2, (uint8_t *)txt, strlen(txt), HAL_MAX_DELAY);
            //osDelay(1);
       }
  }
}
```

```
void receiver(void *argument)
{

      char txt[20];
      uint32_t received;
      osStatus_t status;

      for(;;)
  {
      status = osMessageQueueGet(myQueue01Handle, &received, NULL, 100);
      //osDelay(1);
      if(status == osOK){
            sprintf(txt,"Primljeno je: %lu \n",received);
            HAL_UART_Transmit(&huart2, (uint8_t *)txt, strlen(txt), HAL_MAX_DELAY);
      }
      else{
            strcpy(txt,"Prijem nije uspeo");
            HAL_UART_Transmit(&huart2, (uint8_t *)txt, strlen(txt), HAL_MAX_DELAY);
      }

  }
}
```

Primer 2. Napraviti tri zadatka:
Ime: Sender1 – Prioritet: osPriorityHigh – Parametar &(structs_to_send[0]) – Stack:128 – Entry function: sender
Ime: Sender2 – Prioritet: osPriorityHigh – Parametar &(structs_to_send[1]) – Stack:128 – Entry function: sender
Ime: Receiver – Prioritet: osPriorityLow – Stack:128 – Entry function: receiver
Napraviti red:
Ime: MyQueue1 – Size: 5 – ItemSize: Data

```
-------------------------------------------------------------------------------
/* USER CODE BEGIN PV */
#define mainSender1 1
#define mainSender2 2

typedef struct
{
      uint8_t value;
      uint8_t source;
}Data;

const Data structs_to_send[2]=
{
            {100, mainSender1},
            {200, mainSender2}
};
-------------------------------------------
void sender(void *argument)
{
  osStatus_t status;

  const char txt[] = "Slanje nije uspelo!\n";
  for(;;)
  {
       status =  osMessageQueuePut(myQueue01Handle, argument, NULL, 100);
       if(status != osOK){
          HAL_UART_Transmit(&huart2, (uint8_t *)txt, strlen(txt), HAL_MAX_DELAY);
       }
  }
}
-------------------------------------------
void receiver(void *argument)
{
      char txt[20];
      Data received;
      osStatus_t status;

      for(;;)
  {

      if(osMessageQueueGetCount(myQueue01Handle) == 5){
            strcpy(txt,"Red je pun\n");
            HAL_UART_Transmit(&huart2, (uint8_t *)txt, strlen(txt), HAL_MAX_DELAY);
      }

      status = osMessageQueueGet(myQueue01Handle, &received, NULL, 0);

      if(status == osOK){
          if(received.source == mainSender1){
            sprintf(txt,"Zadatak 1: %hu \n",received.value);
            HAL_UART_Transmit(&huart2, (uint8_t *)txt, strlen(txt), HAL_MAX_DELAY);
```

```
          }
          if(received.source == mainSender2){
            sprintf(txt,"Zadatak 2: %hu \n",received.value);
            HAL_UART_Transmit(&huart2, (uint8_t *)txt, strlen(txt), HAL_MAX_DELAY);
            }
      }
      else{
            strcpy(txt,"Prijem nije uspeo\n");
            HAL_UART_Transmit(&huart2, (uint8_t *)txt, strlen(txt), HAL_MAX_DELAY);
      }

  }
}
```