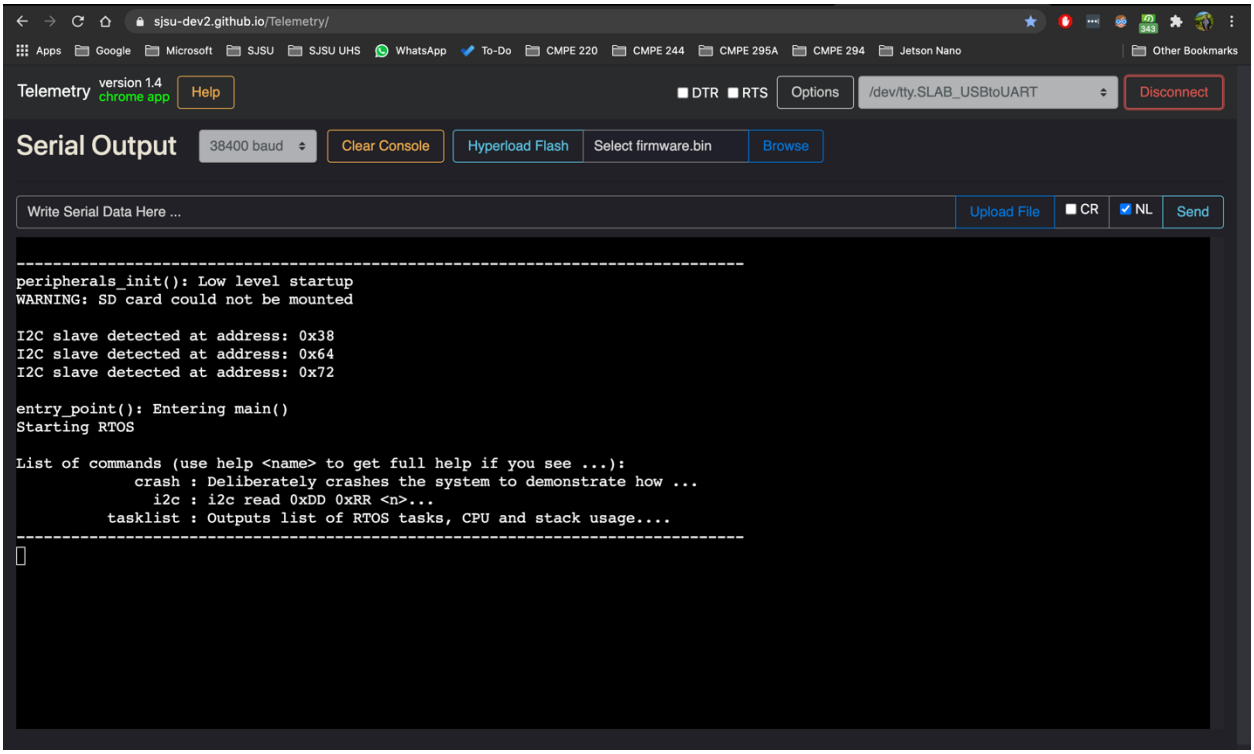
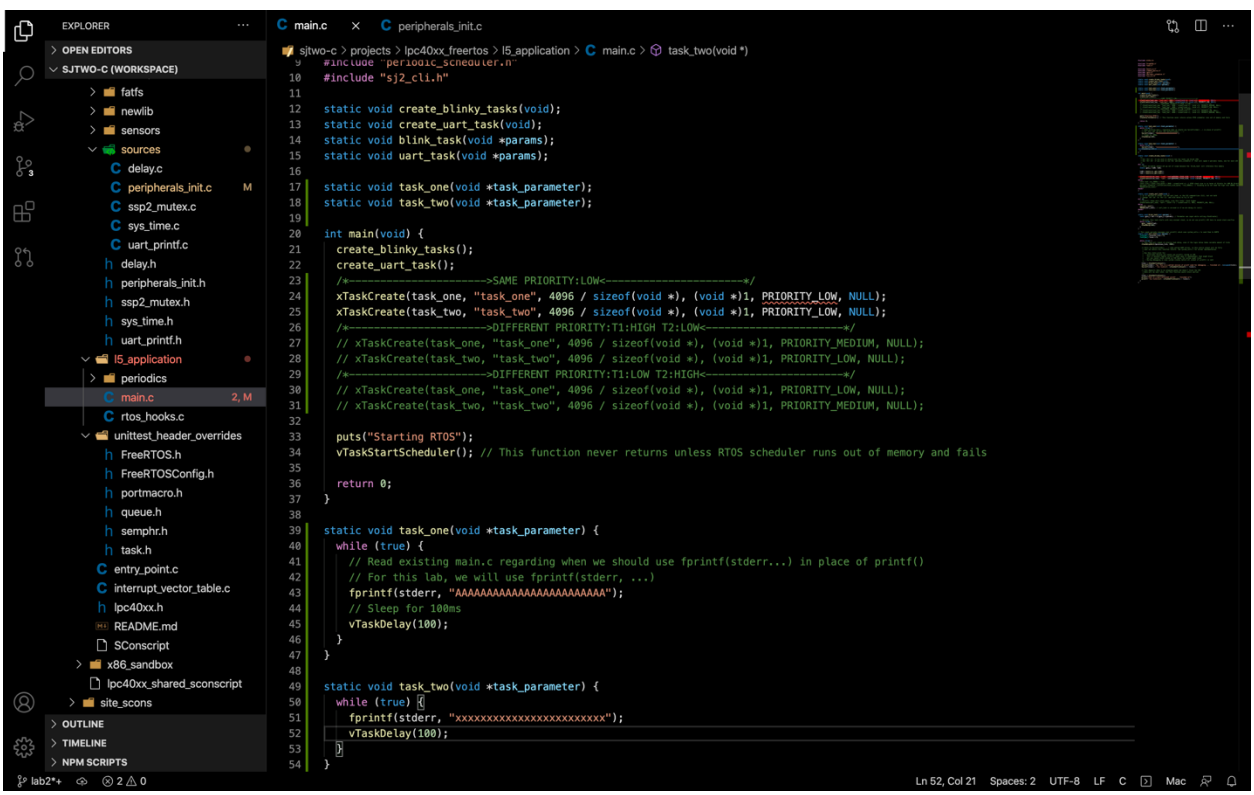
AKHIL CHERUKURI
CMPE 244 Assignment: Multiple Tasks

Part 0a: Change UART speed



Part 0b & 1: Create Task Skeleton and Create RTOS tasks



Part 2: Further Observation (Critical thinking question)

Q) How come 4(or 3 sometimes) characters are printed from each task? Why not 2 or 5, or 6?
A) We have set the UART speed to 38400 bits per second. Taking into account that a single character comprises of 8 bits and UART send data packets with a start and stop bit along with the data bit, therefore each dataset is made of [1 Start bit + 8 Data bits (character) + 1 Stop bit] 10 bits. So overall it will send [UART speed/bits per character = 38400/10] 3840 characters per second. In FreeRTOS, time slicing occurs for every tick (tick is set to 1ms) when both tasks have the same priority, so 3840/1000 = 3.84 characters per millisecond, that is why the output is varying in the range of 3-4 characters.

Part 3: Change the Priority Levels

<u>Same Priority (Task 1: Low; Task 2: Low):</u>

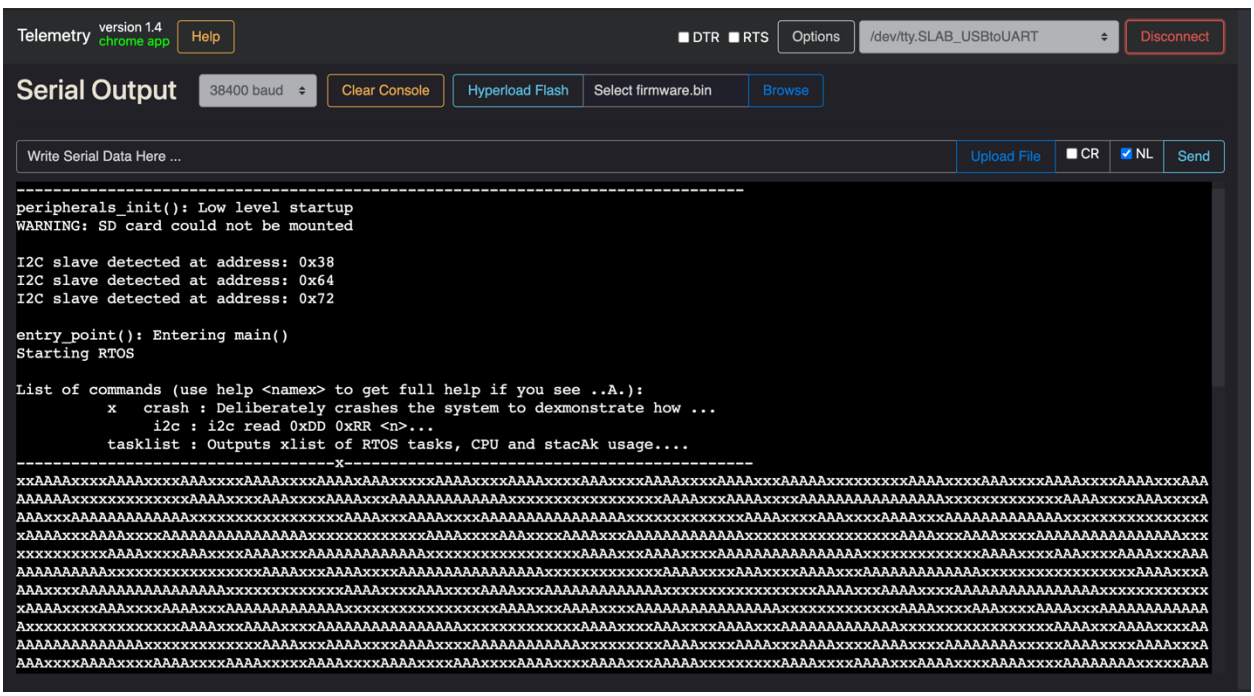


Figure 1: Same Priority with vTaskDelay

We can observe that two tasks with equal priority, the FreeRTOS schedular is using time slice of 1ms to allow both the tasks to run. As written in Part 2 observation, we can see that 3-4 characters printing from each task with UART speed set to 38400. Time needed to print 25 characters of each task is [25/3.84] 6.51 milliseconds. The difference between Figure 1 and Figure 2 is that Figure 1 has vTaskDelay(100), this is the reason the time slicing stops for 100ms so that each task has the ability to print all 25 characters before time slicing starts again. In Figure 2, we can observe that without vTaskDelay, timing slicing is continuous, therefore you can see a constant output of 4 characters of task1 and task2 continuously.

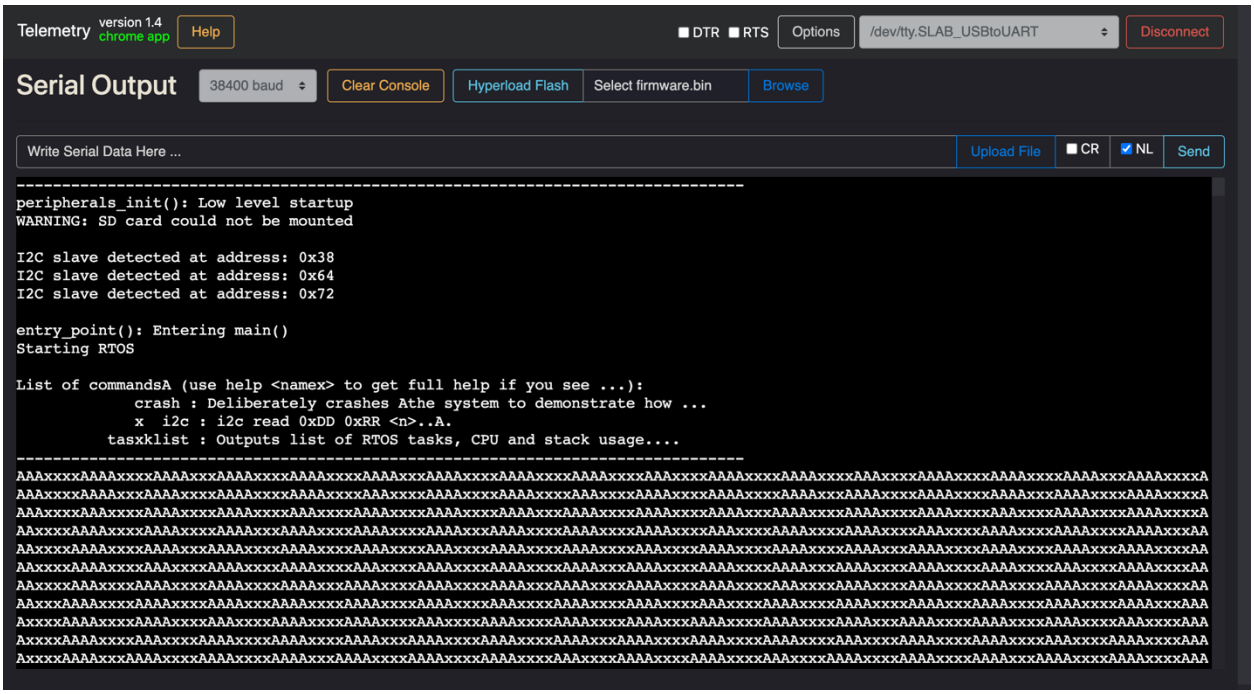


Figure 2: Same Priority without vTaskDelay.

Different Priority (Task 1: High; Task 2: Low):

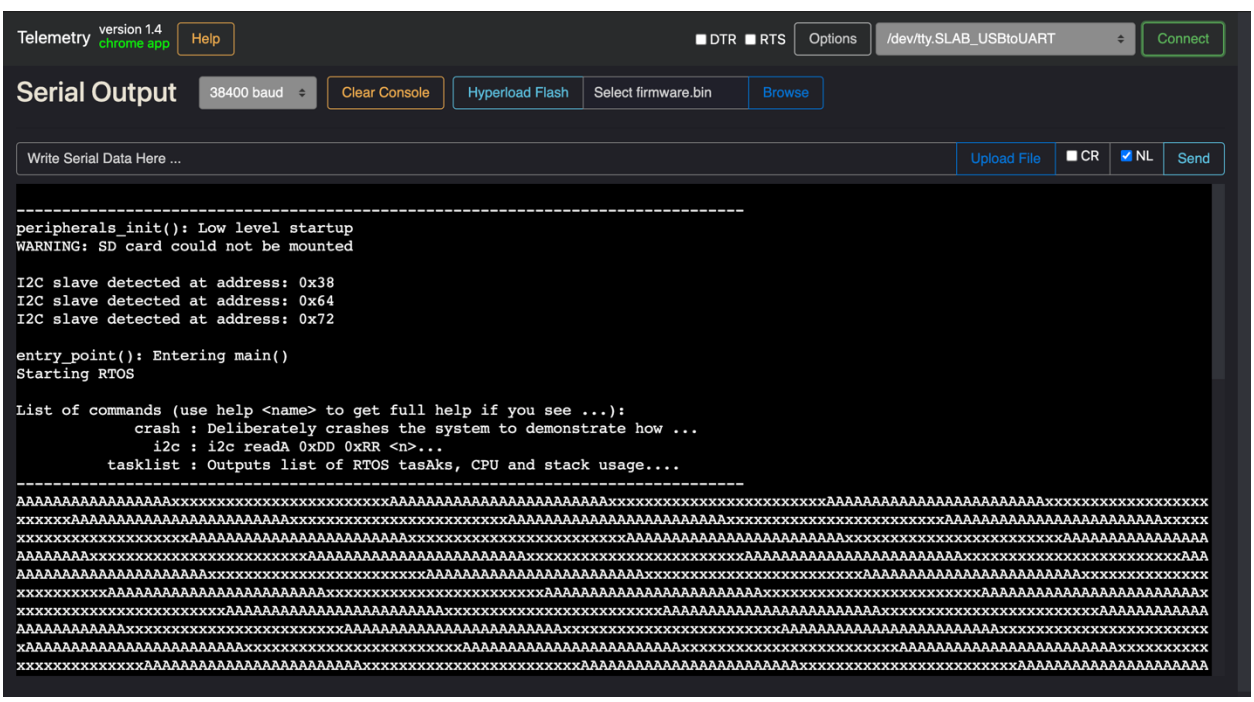


Figure 3: Different Priority (Task1: High; Task2: Low)

Here we have task1 with high priority than task2, that is why the scheduler runs task1 (char[25] = 'AAAAAAAAAAAAAAAAAAAAAAAAA') first and with vTaskDelay(100) this allows the higher priority task1 to stop for 100ms to allow the lower priority task2 to run, the 100ms is

enough to print the content of task2 (char[25] = ‘xxxxxxxxxxxxxxxxxxxxxxxxx’) before context switching back to task1 with the rate of 3.84 characters per millisecond. Without vTaskDelay, task2 of lower priority would not have the chance to run, and we would be able to see the output of task1 only.

<u>Different Priority (Task 1: Low; Task 2: High):</u>

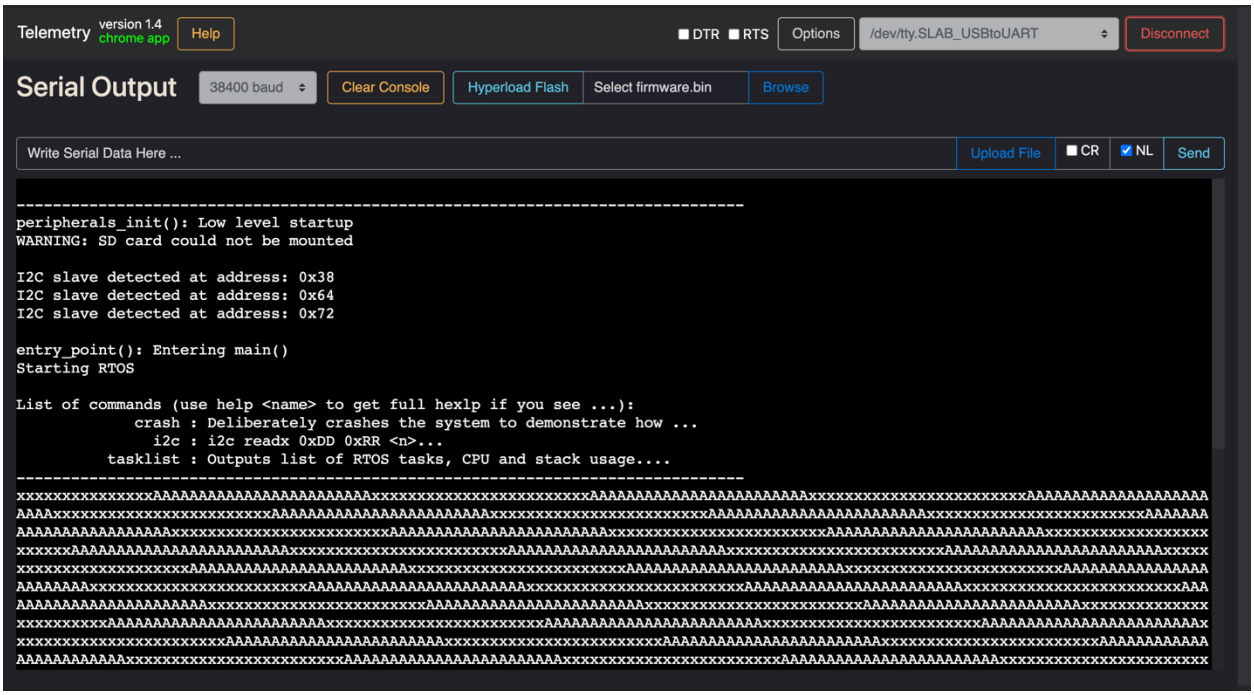


Figure 4: Different Priority (Task1: Low; Task2: High)

Here we have task1 with high priority than task2, that is why the scheduler runs task1 (char[25] = ’ xxxxxxxxxxxxxxxxxxxxxxxxx’) first and with vTaskDelay(100) this allows the higher priority task1 to stop for 100ms to allow the lower priority task2 to run, the 100ms is enough to print the content of task2 (char[25] = ‘AAAAAAAAAAAAAAAAAAAAAAAAA’) before context switching back to task1 with the rate of 3.84 characters per millisecond. Without vTaskDelay, task1 of lower priority would not have the chance to run, and we would be able to see the output of task2 only.