**Sanket Vinod Agarwal**

**Assignment 2**

**Git: https://github.tamu.edu/agarwal-220196/FreeRTOS.git**

Q: Why is "matrixtask" using most of the CPU utilization?

A: Matrix task is a CPU dependent task. To simulate the actual CPU dependency, a long delay has been added and thus it keeps the CPU holding for a longer duration and thus the CPU utilization is higher for matrixtask.

Q:Why must the priority of "communicationtask" increase in order for it to work properly?

A: Communication task has a lower priority as compared to matrix task. Thus, it is possible that matrixtask may suspend communicationtask. Since matrixtask's CPU utilization is high, it may cause the comunicationtask to be suspended for a long time and that may cause the communicationtask to miss its deadline. Thus, in order for it to work properly, it should changed to a higher priority.

Q: What happens to the completion time of "matrixtask" when the priority of "communicationtask" is increased?

A: When the priority of communicationtask is increased, it gets the authority to suspend matrix task and thus the duration of matrixtask gets increased as waiting time also gets added to it.

Q: How many seconds is the period of "matrixtask"? (Hint: look at vApplicationTickHook() to measure it)

A: The matrixtask period gets changed to 3.6 sec or 3647 ms as seen from the following output

**Additional learning:**

Matrixtask uses dynamic memory allocation by using pvPortMalloc(). I was curious about why we can't use the standard malloc and free functions

*and I found the following answer on the reference website mentioned below:

*If RTOS objects are created dynamically then the standard C library

*malloc() and free() functions can sometimes be used for the purpose, but …

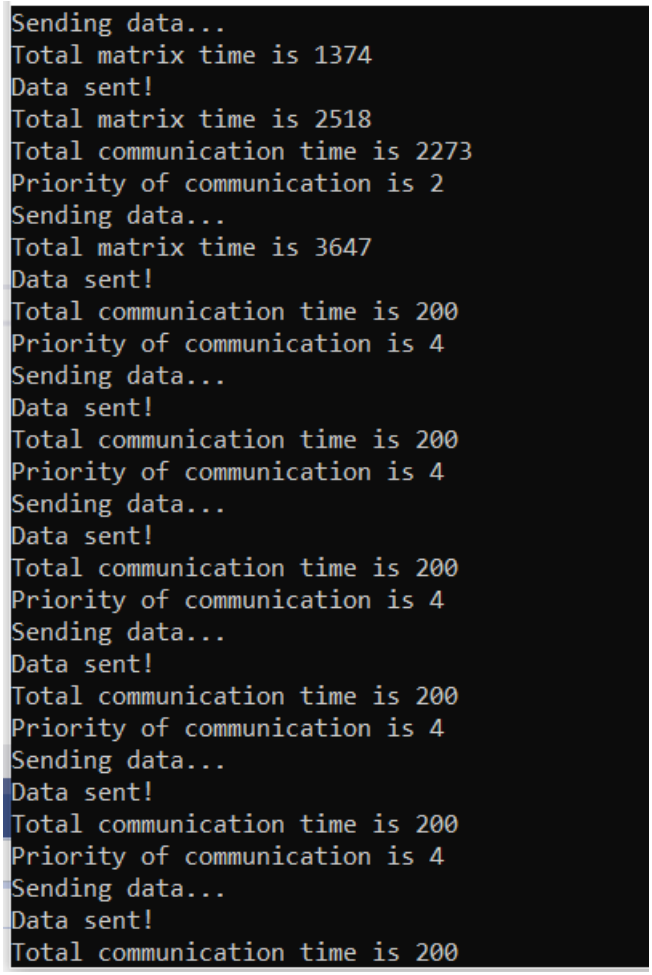
*they are not always available on embedded systems,

*they take up valuable code space,

*they are not thread safe, and

*they are not deterministic (the amount of time taken to execute the function will differ from call to call)

*Reference: https://www.freertos.org/a00111.html




Output Result