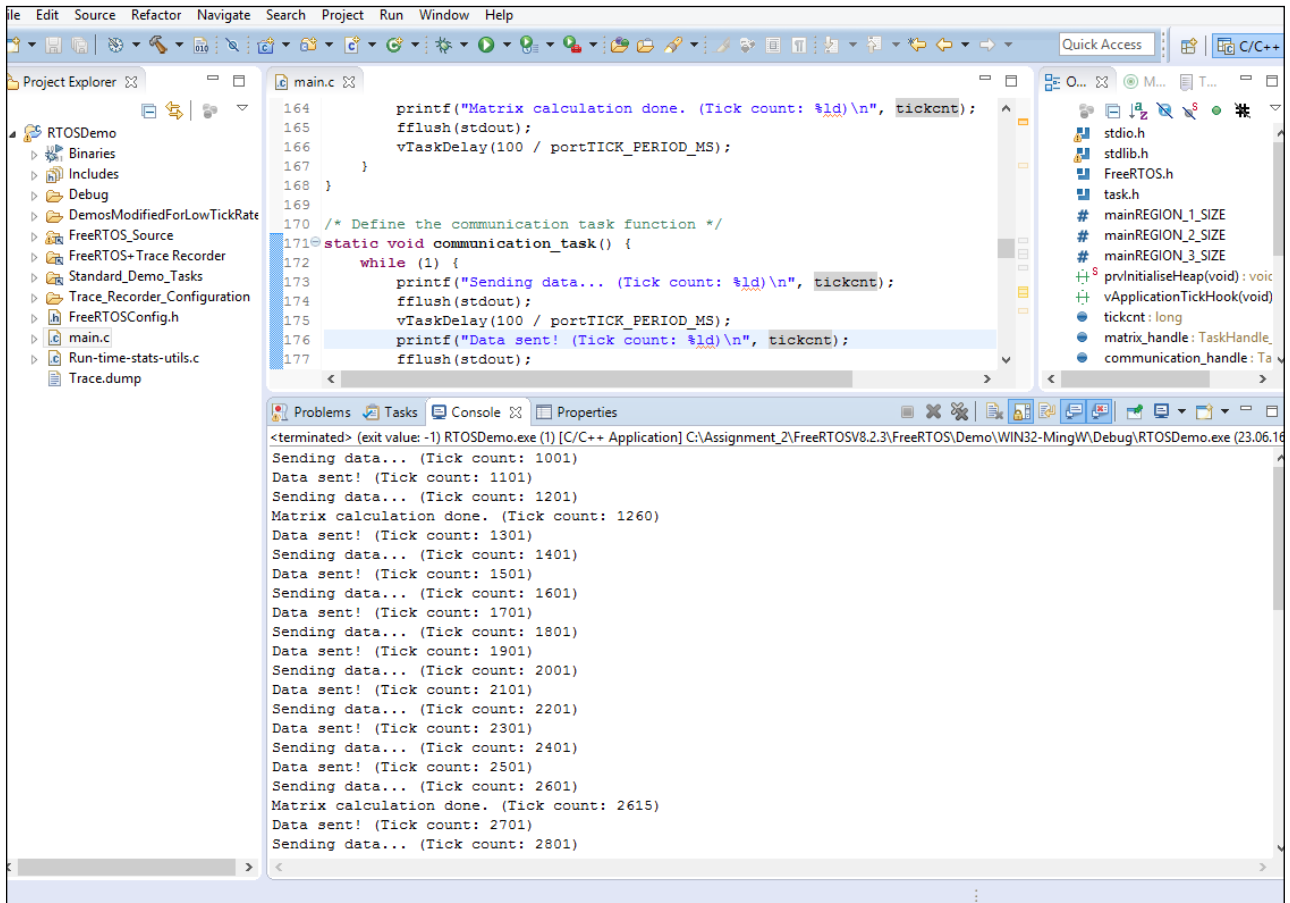


# Development of Real-Time Systems

## Assignment 2

Elyasin Shaladi - 23 June 2016



```
164     printf("Matrix calculation done. (Tick count: %ld)\n", tickcnt);
165     fflush(stdout);
166     vTaskDelay(100 / portTICK_PERIOD_MS);
167 }
168 }
169
170 /* Define the communication task function */
171 static void communication_task() {
172     while (1) {
173         printf("Sending data... (Tick count: %ld)\n", tickcnt);
174         fflush(stdout);
175         vTaskDelay(100 / portTICK_PERIOD_MS);
176         printf("Data sent! (Tick count: %ld)\n", tickcnt);
177         fflush(stdout);
178     }
179 }
```

Console Output:

```
<terminated> (exit value: -1) RTOSDemo.exe (1) [C/C++ Application] C:\Assignment_2\FreeRTOSV8.2.3\FreeRTOS\Demo\WIN32-MingW\Debug\RTOSDemo.exe (23.06.16)
Sending data... (Tick count: 1001)
Data sent! (Tick count: 1101)
Sending data... (Tick count: 1201)
Matrix calculation done. (Tick count: 1260)
Data sent! (Tick count: 1301)
Sending data... (Tick count: 1401)
Data sent! (Tick count: 1501)
Sending data... (Tick count: 1601)
Data sent! (Tick count: 1701)
Sending data... (Tick count: 1801)
Data sent! (Tick count: 1901)
Sending data... (Tick count: 2001)
Data sent! (Tick count: 2101)
Sending data... (Tick count: 2201)
Data sent! (Tick count: 2301)
Sending data... (Tick count: 2401)
Data sent! (Tick count: 2501)
Sending data... (Tick count: 2601)
Matrix calculation done. (Tick count: 2615)
Data sent! (Tick count: 2701)
Sending data... (Tick count: 2801)
```

---

## Assignment 2 Report

Why is *matrix\_task* using most of the CPU utilisation?

*matrix\_task* used most of the CPU utilisation because it had a higher priority than *communication\_task*. *matrix\_task* was created with priority 3. *communication\_task* was created with priority 1.

Also *matrix\_task* is very computation intensive. On the other hand *communication\_task* merely prints out "Sending data" and "Data sent" and does not do intense computing.

Why must the priority of *communication\_task* increase in order for it to work properly?

*communication\_task*'s priority must increase in order to be scheduled accordingly. Otherwise *communication\_task* would only get scheduled after the *matrix\_task* is blocked for 100ms by a delay, i.e. when the *matrix\_task* is done.

What happens to the completion time of *matrix\_task* when the priority of *communication\_task* is increased?

The completion time of the *matrix\_task* is later because, due to the increase of the priority of *communication\_task*, the scheduler now assigns CPU time to the *communication\_task*.

However, since *communication\_task* is not computation intensive, the difference in completion time is actually small.

How many seconds is the period of *matrix\_task*? (Hint: look at `vApplicationTickHook()` to measure it)

The period of *matrix\_task* is approximately 1.3 seconds on my machine.