Assignment 2

I used Eclipse IDE as described in instruction file you can use visual studio code also by editing main.c and adding lines that I will explain below :

The required exercise :

Create a new task "prioritysettask" which:

1. Sets the priority of "communicationtask" to 4 in case its execution time is more than 1000 milliseconds (Hint: look at vApplicationTickHook() to measure it)
2. Sets the priority of "communicationtask" to 2 in case its execution time is less than 200 milliseconds (Hint: look at vApplicationTickHook() to measure it)

Provide a screenshot of the execution and answer the following questions in a report:

- Why is "matrixtask" using most of the CPU utilization?
- Why must the priority of "communicationtask" increase in order for it to work properly
- What happens to the completion time of "matrixtask" when the priority of "communicationtask" is increased?
- How many seconds is the period of "matrixtask"? (Hint: look at vApplicationTickHook() to measure it)

Solution:matrix_period 1274

Why is "matrix task" using most of the CPU utilization

Because in real life embedded system matrix multiplication causes too much delay .however, in the provided code which doesn't run on embedded system but on desktop , it has hard coded block of code that makes it takes too much time

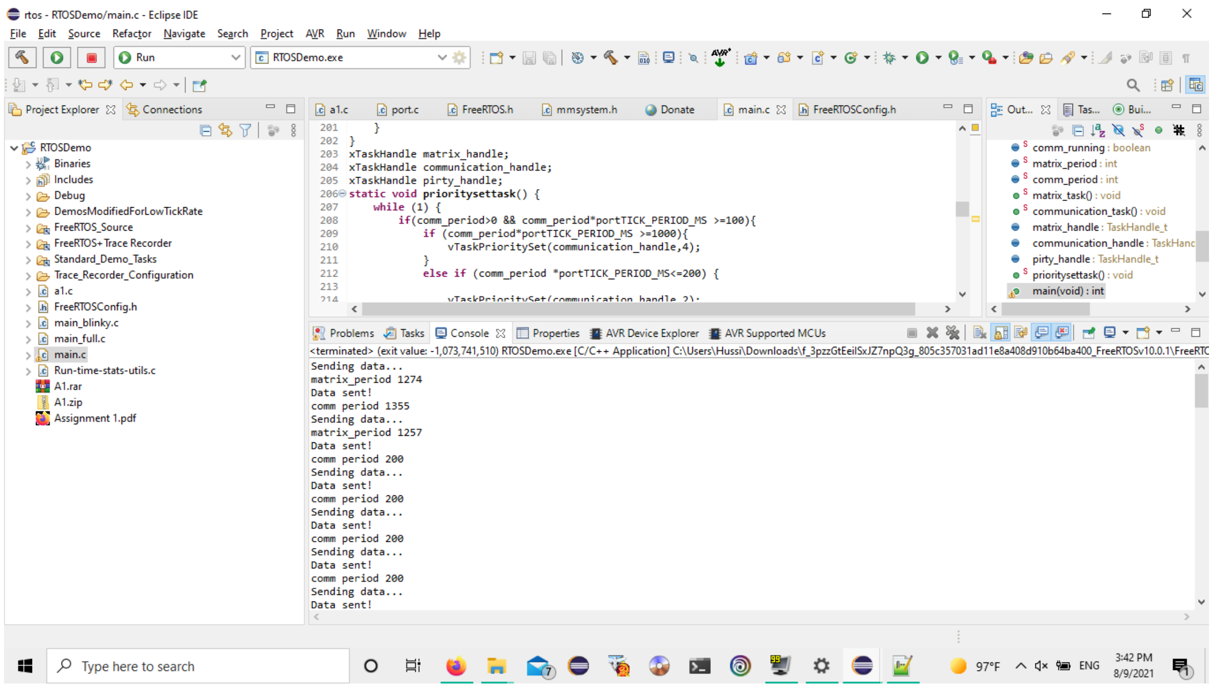
```
for (simulationdelay = 0; simulationdelay < 1000000000;simulationdelay++);
```

Why must the priority of "communication task" increase in order for it to work properly

Because matrix multiplication has higher priority, and it finishes after its deadline by large amount which leading to blocking lower priority communication task

What happens to the completion time of "matrix task" when the priority of "communication task" is increased?

It finishes in time



```
}
}
xTaskHandle matrix_handle;
xTaskHandle communication_handle;
xTaskHandle pirty_handle;
static void prioritysettask() {
    while (1) {
        if(comm_period>0 && comm_period*portTICK_PERIOD_MS >=100){
            if (comm_period*portTICK_PERIOD_MS >=1000){
                vTaskPrioritySet(communication_handle,4);
            }
            else if (comm_period *portTICK_PERIOD_MS<=200) {

                vTaskPrioritySet(communication_handle,2);
```

```
<terminated> (exit value: -1,073,741,510) RTOSDemo.exe [C/C++ Application] C:\Users\Hussi\Downloads\f_3pzzGtEeilSxJZ7npQ3g_805c357031ad11e8a408d910b64ba400_FreeRTOSv10.0.1\FreeRTC
Sending data...
matrix_period 1274
Data sent!
comm period 1355
Sending data...
matrix_period 1257
Data sent!
comm period 200
Sending data...
Data sent!
comm period 200
Sending data...
Data sent!
comm period 200
Sending data...
Data sent!
comm period 200
Sending data...
Data sent!
```

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

```
Total time = 1274
```

Code:

Matrix task and communication task :

```
static void matrix_task() {
    int i;
    double **a = (double**) pvPortMalloc(ROW * sizeof(double*));
    for (i = 0; i < ROW; i++)
       a[i] = (double*) pvPortMalloc(COL * sizeof(double));
    double **b = (double**) pvPortMalloc(ROW * sizeof(double*));
    for (i = 0; i < ROW; i++)
       b[i] = (double*) pvPortMalloc(COL * sizeof(double));
    double **c = (double**) pvPortMalloc(ROW * sizeof(double*));
    for (i = 0; i < ROW; i++)
       c[i] = (double*) pvPortMalloc(COL * sizeof(double));

    double sum = 0.0;
    int j, k, l;

    for (i = 0; i < SIZE; i++) {
       for (j = 0; j < SIZE; j++) {
             a[i][j] = 1.5;
             b[i][j] = 2.6;
       }
    }

    while (1) {
       /*
        * In an embedded systems, matrix multiplication would block the CPU for a long
time
        * but since this is a PC simulator we must add one additional dummy delay.
        */
       matrix_running = TRUE;
       matrix_period=0;
       long simulationdelay;
       for (simulationdelay = 0; simulationdelay < 1000000000;
                   simulationdelay++)
             ;
       for (i = 0; i < SIZE; i++) {
             for (j = 0; j < SIZE; j++) {
                   c[i][j] = 0.0;
             }
       }

       for (i = 0; i < SIZE; i++) {
             for (j = 0; j < SIZE; j++) {
                   sum = 0.0;
                   for (k = 0; k < SIZE; k++) {
                         for (l = 0; l < 10; l++) {
                               sum = sum + a[i][k] * b[k][j];
                         }
                   }
                   c[i][j] = sum;
             }
       }
       vTaskDelay(100);
      matrix_running = FALSE;
      printf("matrix_period %d\n",matrix_period * portTICK_PERIOD_MS);
    }

}
static void communication_task() {
    while (1) {
       comm_running = TRUE;
       comm_period=0;
       printf("Sending data...\n");
       fflush(stdout);
       vTaskDelay(100);
       printf("Data sent!\n");
       fflush(stdout);
       vTaskDelay(100);
       comm_running = FALSE;
       printf("comm period %d\n",comm_period * portTICK_PERIOD_MS);
    }
}
```

Prioritysettask

```
xTaskHandle matrix_handle;
xTaskHandle communication_handle;
xTaskHandle pirty_handle;
static void prioritysettask() {
    while (1) {
       if(comm_period>0 && comm_period*portTICK_PERIOD_MS >=100){
             if (comm_period*portTICK_PERIOD_MS >=1000){
                   vTaskPrioritySet(communication_handle,4);
             }
             else if (comm_period *portTICK_PERIOD_MS<=200) {

                   vTaskPrioritySet(communication_handle,2);

             }
       }
    }
}
```

vApplicationTickHook

```
void vApplicationTickHook(void) {

    if (matrix_running == TRUE) {
        matrix_period++;
    }


    if (comm_running == TRUE) {
        comm_period++;
    }

}
```

main

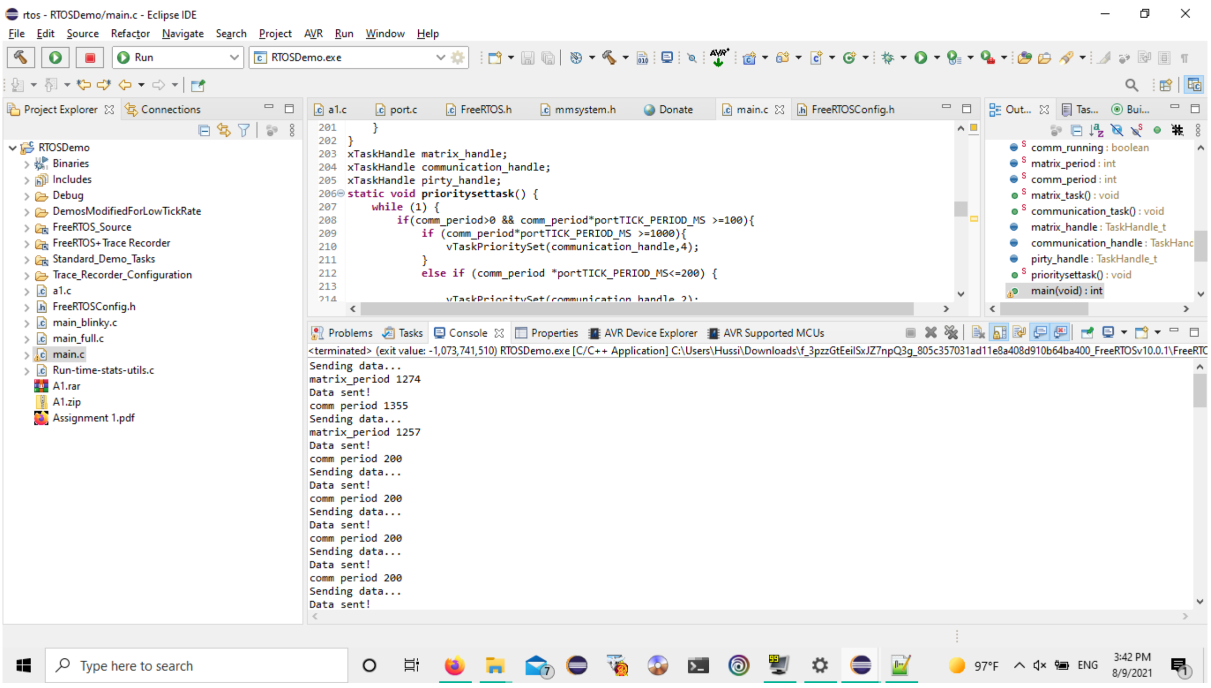
```c
int main(void) {
    /* This demo uses heap_5.c, so start by defining some heap regions.  heap_5
     is only used for test and example reasons.  Heap_4 is more appropriate.  See
     http://www.freertos.org/a00111.html for an explanation. */
    prvInitialiseHeap();

    /* Initialise the trace recorder.  Use of the trace recorder is optional.
     See http://www.FreeRTOS.org/trace for more information. */
    vTraceEnable( TRC_START);

    //xTaskCreate(HelloTask,"HELLO TASK",configMINIMAL_STACK_SIZE,NULL,1,&HT);
    xTaskCreate((pdTASK_CODE) matrix_task, (signed char*) "Matrix", 1000, NULL,
             3, &matrix_handle);
    xTaskCreate((pdTASK_CODE) communication_task,
             (signed char*) "Communication", configMINIMAL_STACK_SIZE, NULL, 1,
             &communication_handle);

    xTaskCreate((pdTASK_CODE) prioritysettask, (signed char*) "pirty",
configMINIMAL_STACK_SIZE, NULL,
             1, &pirty_handle);

    /* The mainCREATE_SIMPLE_BLINKY_DEMO_ONLY setting is described at the top
     of this file. */
    vTaskStartScheduler();
    while (1)
       ;
    return 0;
}
/*-----------------------------------------------------------*/
```



```c
    }
}
xTaskHandle matrix_handle;
xTaskHandle communication_handle;
xTaskHandle pirty_handle;
static void prioritysettask() {
    while (1) {
        if(comm_period>0 && comm_period*portTICK_PERIOD_MS >=100){
            if (comm_period*portTICK_PERIOD_MS >=1000){
                vTaskPrioritySet(communication_handle,4);
            }
            else if (comm_period *portTICK_PERIOD_MS<=200) {

                vTaskPrioritySet(communication_handle,2);
```

```
<terminated> (exit value: -1,073,741,510) RTOSDemo.exe [C/C++ Application] C:\Users\Hussi\Downloads\f_3pzzGtEeilSxJZ7npQ3g_805c357031ad11e8a408d910b64ba400_FreeRTOSv10.0.1\FreeRTC
Sending data...
matrix_period 1274
Data sent!
comm period 1355
Sending data...
matrix_period 1257
Data sent!
comm period 200
Sending data...
Data sent!
comm period 200
Sending data...
Data sent!
comm period 200
Sending data...
Data sent!
comm period 200
Sending data...
Data sent!
```