Industrial Automation

Assignment 6

Professor Behzad Moshiri

Samira Hajizadeh - 810198378

Introduction

Rate Monotonic Scheduling

Rate-monotonic scheduling (RMS) is a priority assignment algorithm used in real-time operating systems (RTOS) with a static-priority scheduling class. The static priorities are assigned according to the cycle duration of the job, so a shorter cycle duration results in a higher job priority. (**Wikipedia**)

Code Explanation

First, we define class ”Task” in order to organize tasks more neatly. Then we find the hyper period of the tasks by calculating the least common multiple.

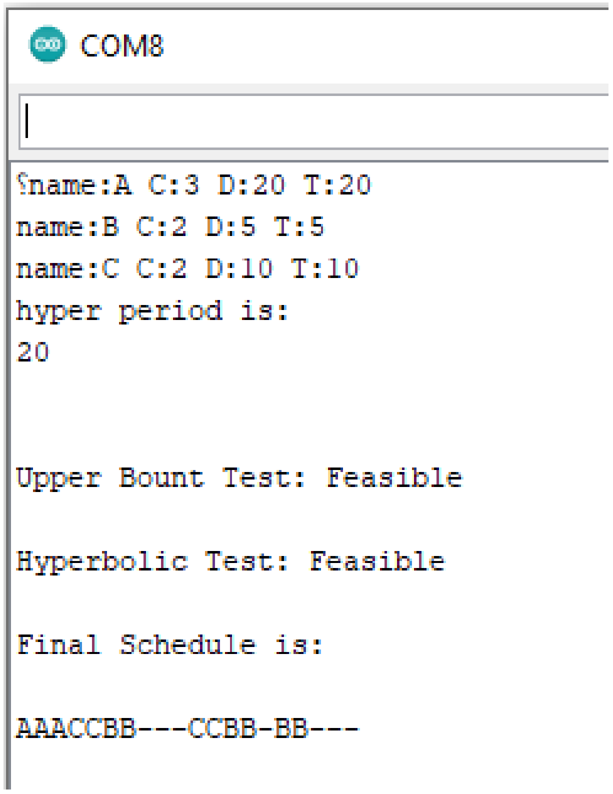
In the second part we print the tasks once and start sorting them by their cycle duration. Then we do upper bound (equation 1) and hyperbolic (equation 2) tests to check whether rate monotonic scheduling is possible or not. At the end, we start scheduling by executing the task with higher priorities first. The output of the code can be seen in figure 1.

A picture of arduino IDE containing the code is shown in figure 2 to 6.

** NOTE: In order to run the code, copy the ”Task” folder containing a header file with the same name and paste it in your arduino libraries folder (Usually: Documents Arduino libraries).

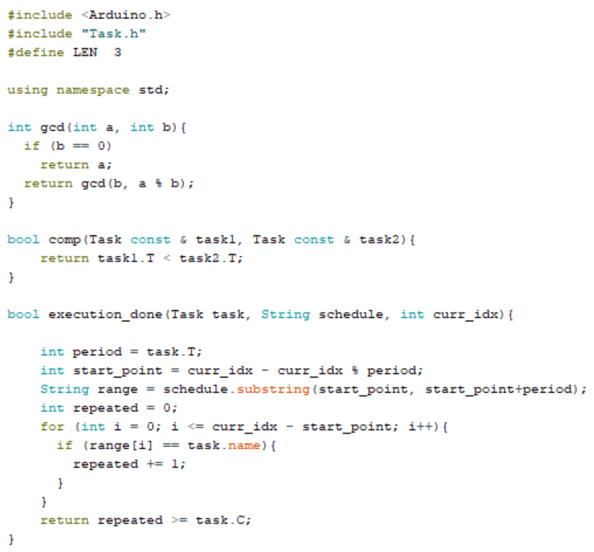
$$U = \sum_{i=1}^{n} U_i = \sum_{i=1}^{n} \frac{C_i}{T_i} \leq n(2^{1/n} - 1) \tag{1}$$

$$\prod_{i=1}^{n} (U_i + 1) \leq 2 \tag{2}$$



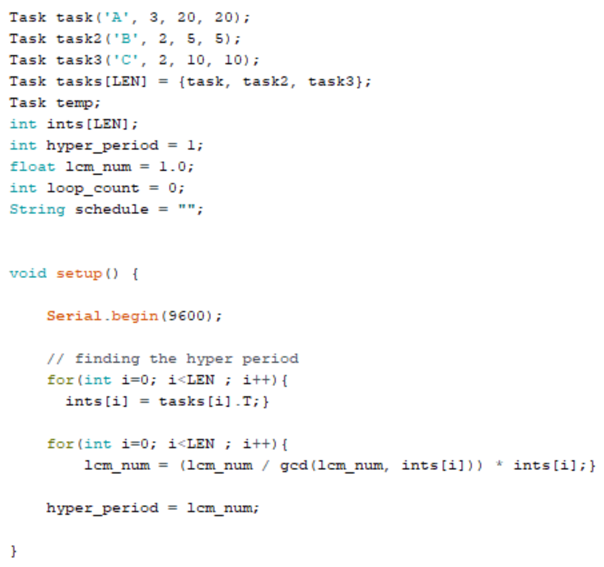
```
⸮name:A C:3 D:20 T:20
name:B C:2 D:5 T:5
name:C C:2 D:10 T:10
hyper period is:
20


Upper Bount Test: Feasible

Hyperbolic Test: Feasible

Final Schedule is:

AAACCBB---CCBB-BB---
```

Figure 1: The serial output of RMS code in arduino IDE



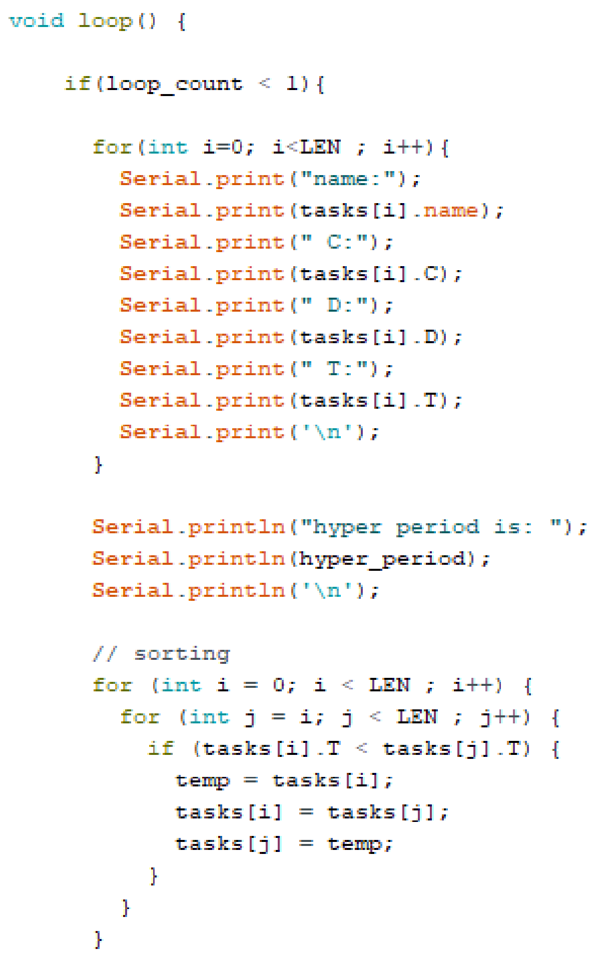
```cpp
#include <Arduino.h>
#include "Task.h"
#define LEN  3

using namespace std;

int gcd(int a, int b){
  if (b == 0)
    return a;
  return gcd(b, a % b);
}

bool comp(Task const & task1, Task const & task2){
    return task1.T < task2.T;
}

bool execution_done(Task task, String schedule, int curr_idx){

    int period = task.T;
    int start_point = curr_idx - curr_idx % period;
    String range = schedule.substring(start_point, start_point+period);
    int repeated = 0;
    for (int i = 0; i <= curr_idx - start_point; i++){
      if (range[i] == task.name){
        repeated += 1;
      }
    }
    return repeated >= task.C;
}
```

Figure 2: CPP code for RMS - PART 1



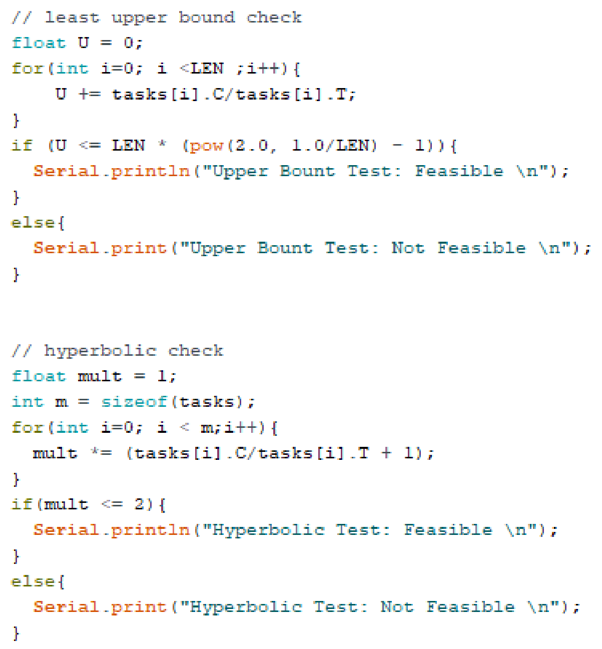
```cpp
Task task('A', 3, 20, 20);
Task task2('B', 2, 5, 5);
Task task3('C', 2, 10, 10);
Task tasks[LEN] = {task, task2, task3};
Task temp;
int ints[LEN];
int hyper_period = 1;
float lcm_num = 1.0;
int loop_count = 0;
String schedule = "";


void setup() {

    Serial.begin(9600);

    // finding the hyper period
    for(int i=0; i<LEN ; i++){
      ints[i] = tasks[i].T;}

    for(int i=0; i<LEN ; i++){
        lcm_num = (lcm_num / gcd(lcm_num, ints[i])) * ints[i];}

    hyper_period = lcm_num;

}
```

Figure 3: CPP code for RMS - PART 2



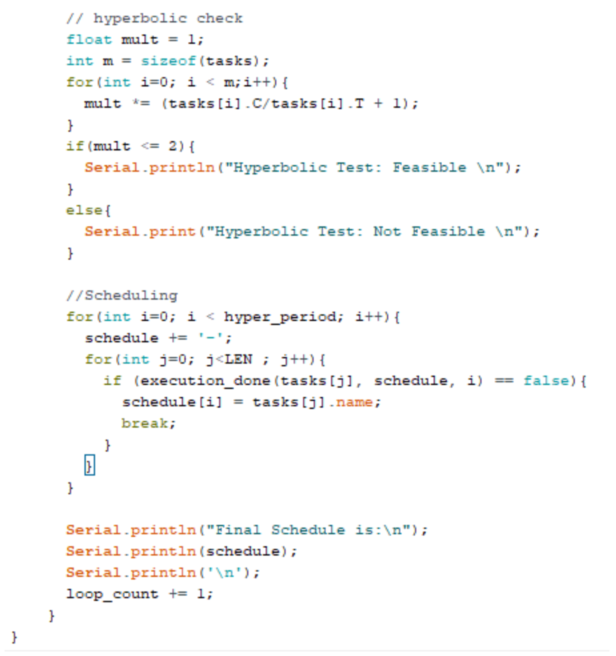
```cpp
void loop() {

    if(loop_count < 1){

      for(int i=0; i<LEN ; i++){
        Serial.print("name:");
        Serial.print(tasks[i].name);
        Serial.print(" C:");
        Serial.print(tasks[i].C);
        Serial.print(" D:");
        Serial.print(tasks[i].D);
        Serial.print(" T:");
        Serial.print(tasks[i].T);
        Serial.print('\n');
      }

      Serial.println("hyper period is: ");
      Serial.println(hyper_period);
      Serial.println('\n');

      // sorting
      for (int i = 0; i < LEN ; i++) {
        for (int j = i; j < LEN ; j++) {
          if (tasks[i].T < tasks[j].T) {
            temp = tasks[i];
            tasks[i] = tasks[j];
            tasks[j] = temp;
          }
        }
      }
```

Figure 4: CPP code for RMS - PART 3



```
// least upper bound check
float U = 0;
for(int i=0; i <LEN ;i++){
    U += tasks[i].C/tasks[i].T;
}
if (U <= LEN * (pow(2.0, 1.0/LEN) - 1)){
  Serial.println("Upper Bount Test: Feasible \n");
}
else{
  Serial.print("Upper Bount Test: Not Feasible \n");
}


// hyperbolic check
float mult = 1;
int m = sizeof(tasks);
for(int i=0; i < m;i++){
  mult *= (tasks[i].C/tasks[i].T + 1);
}
if(mult <= 2){
  Serial.println("Hyperbolic Test: Feasible \n");
}
else{
  Serial.print("Hyperbolic Test: Not Feasible \n");
}
```

Figure 5: CPP code for RMS - PART 4



```
      // hyperbolic check
      float mult = 1;
      int m = sizeof(tasks);
      for(int i=0; i < m;i++){
        mult *= (tasks[i].C/tasks[i].T + 1);
      }
      if(mult <= 2){
        Serial.println("Hyperbolic Test: Feasible \n");
      }
      else{
        Serial.print("Hyperbolic Test: Not Feasible \n");
      }

      //Scheduling
      for(int i=0; i < hyper_period; i++){
        schedule += '-';
        for(int j=0; j<LEN ; j++){
          if (execution_done(tasks[j], schedule, i) == false){
            schedule[i] = tasks[j].name;
            break;
          }
        }
      }

      Serial.println("Final Schedule is:\n");
      Serial.println(schedule);
      Serial.println('\n');
      loop_count += 1;
    }
}
```

Figure 6: CPP code for RMS - PART 5