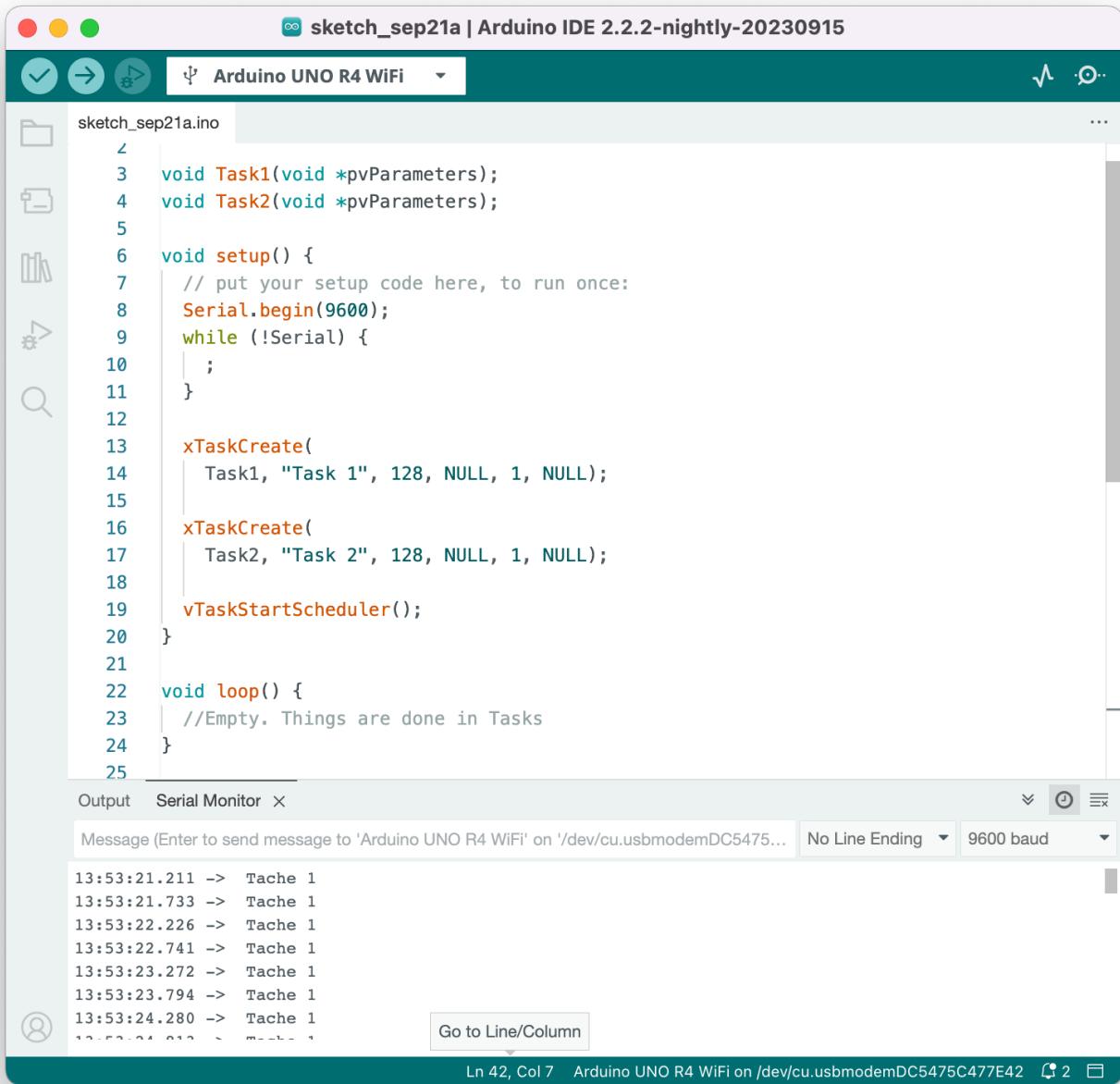


Programmation multi-tâches – Rendu TP02

Partie 1

Le code ci-dessous nous permet d'obtenir le résultat affiché dans la console.



The screenshot shows the Arduino IDE interface with the title bar "sketch_sep21a | Arduino IDE 2.2.2-nightly-20230915". The toolbar includes standard icons for file operations. The left sidebar shows the file "sketch_sep21a.ino" is open. The code editor contains the following FreeRTOS-based Arduino sketch:

```
sketch_sep21a.ino
1
2 void Task1(void *pvParameters);
3 void Task2(void *pvParameters);
4
5 void setup() {
6     // put your setup code here, to run once:
7     Serial.begin(9600);
8     while (!Serial) {
9         ;
10    }
11 }
12
13 xTaskCreate(
14     Task1, "Task 1", 128, NULL, 1, NULL);
15
16 xTaskCreate(
17     Task2, "Task 2", 128, NULL, 1, NULL);
18
19 vTaskStartScheduler();
20 }
21
22 void loop() {
23     //Empty. Things are done in Tasks
24 }
25
```

The "Serial Monitor" tab is selected at the bottom, showing the output window. The output window displays the following log entries:

```
Message (Enter to send message to 'Arduino Uno R4 WiFi' on '/dev/cu.usbmodemDC5475...' No Line Ending 9600 baud
13:53:21.211 -> Tache 1
13:53:21.733 -> Tache 1
13:53:22.226 -> Tache 1
13:53:22.741 -> Tache 1
13:53:23.272 -> Tache 1
13:53:23.794 -> Tache 1
13:53:24.280 -> Tache 1
13:53:24.812 -> m--- 1
```

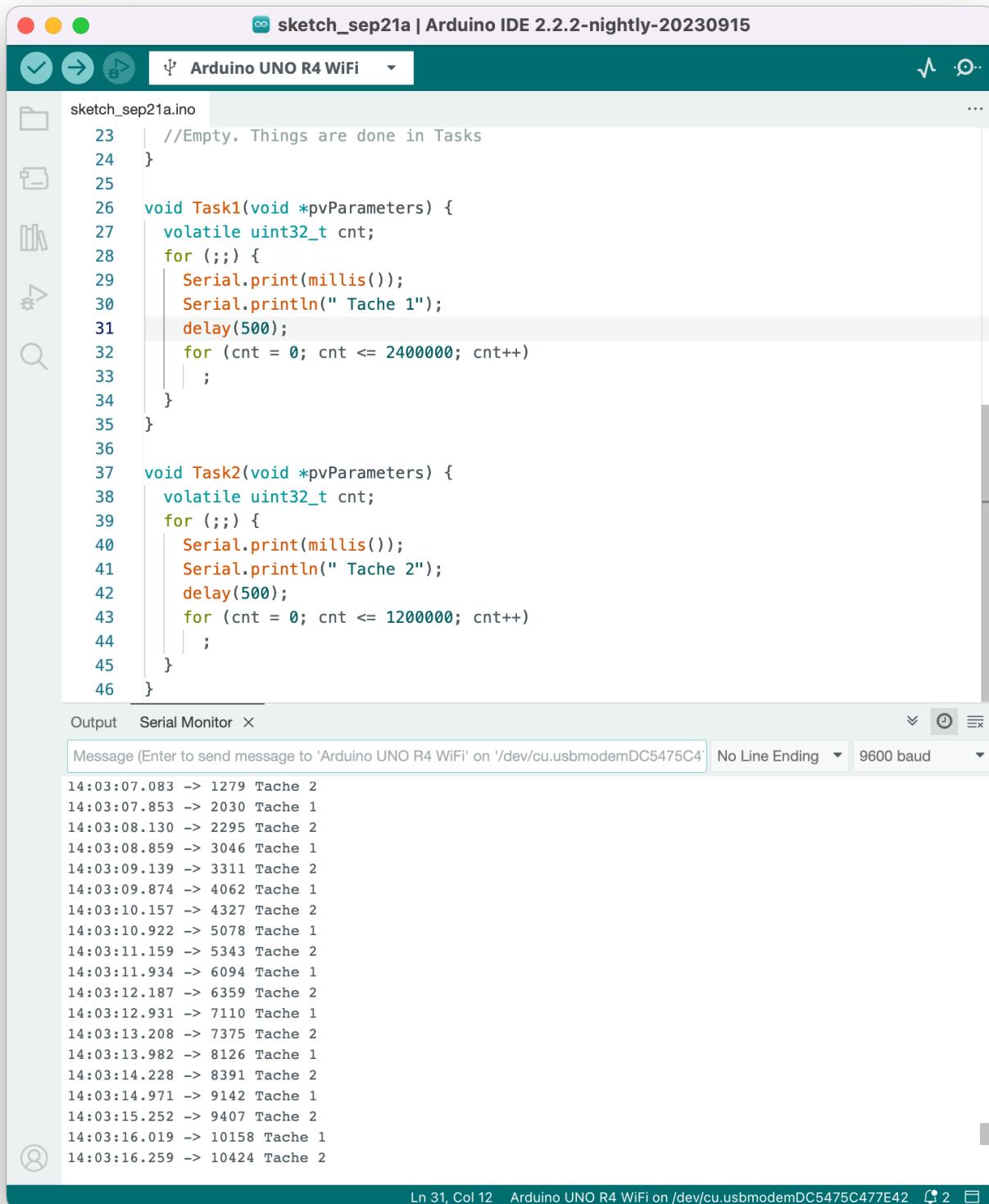
The status bar at the bottom indicates "Ln 42, Col 7" and "Arduino Uno R4 WiFi on /dev/cu.usbmodemDC5475C477E42".

Avec Task1 et Task2 définies plus bas dans le code. Nous pouvons noter que le temps d'exécution est variable, mais dépasse rarement les 500ms, et descend rarement au-dessous de 20ms. Nous pouvons par ailleurs noter que seule la tâche 1 est visible à l'exécution du programme.

Note : La bibliothèque sur macOS contenant les fonctions de freeRTOS se trouve sur le chemin suivant :

/Users/<user>/Library/Arduino15/packages/arduino/hardware/renesas_uno/1.0.4/libraries/Arduino_FreeRTOS/src/Arduino_FreeRTOS.h

En décommentant les lignes permettant de faire un delay, nous pouvons constater que nous avons bien une « simultanéité » lors de l'exécution des tâches.



The screenshot shows the Arduino IDE interface with the sketch `sketch_sep21a`. The code contains two tasks, `Task1` and `Task2`, each printing its execution time and a task identifier to the Serial Monitor. Both tasks run in parallel, with Task 1 taking approximately 2.4 million iterations and Task 2 taking approximately 1.2 million iterations. The Serial Monitor output shows the timestamp, task ID, and iteration count for each task.

```
sketch_sep21a | Arduino IDE 2.2.2-nightly-20230915
Arduino UNO R4 WiFi

sketch_sep21a.ino
23 //Empty. Things are done in Tasks
24 }
25
26 void Task1(void *pvParameters) {
27     volatile uint32_t cnt;
28     for (;;) {
29         Serial.print(millis());
30         Serial.println(" Tache 1");
31         delay(500);
32         for (cnt = 0; cnt <= 2400000; cnt++)
33             ;
34     }
35 }
36
37 void Task2(void *pvParameters) {
38     volatile uint32_t cnt;
39     for (;;) {
40         Serial.print(millis());
41         Serial.println(" Tache 2");
42         delay(500);
43         for (cnt = 0; cnt <= 1200000; cnt++)
44             ;
45     }
46 }

Output Serial Monitor ×

Message (Enter to send message to 'Arduino UNO R4 WiFi' on '/dev/cu.usbmodemDC5475C4': No Line Ending ▾ 9600 baud ▾

14:03:07.083 -> 1279 Tache 2
14:03:07.853 -> 2030 Tache 1
14:03:08.130 -> 2295 Tache 2
14:03:08.859 -> 3046 Tache 1
14:03:09.139 -> 3311 Tache 2
14:03:09.874 -> 4062 Tache 1
14:03:10.157 -> 4327 Tache 2
14:03:10.922 -> 5078 Tache 1
14:03:11.159 -> 5343 Tache 2
14:03:11.934 -> 6094 Tache 1
14:03:12.187 -> 6359 Tache 2
14:03:12.931 -> 7110 Tache 1
14:03:13.208 -> 7375 Tache 2
14:03:13.982 -> 8126 Tache 1
14:03:14.228 -> 8391 Tache 2
14:03:14.971 -> 9142 Tache 1
14:03:15.252 -> 9407 Tache 2
14:03:16.019 -> 10158 Tache 1
14:03:16.259 -> 10424 Tache 2

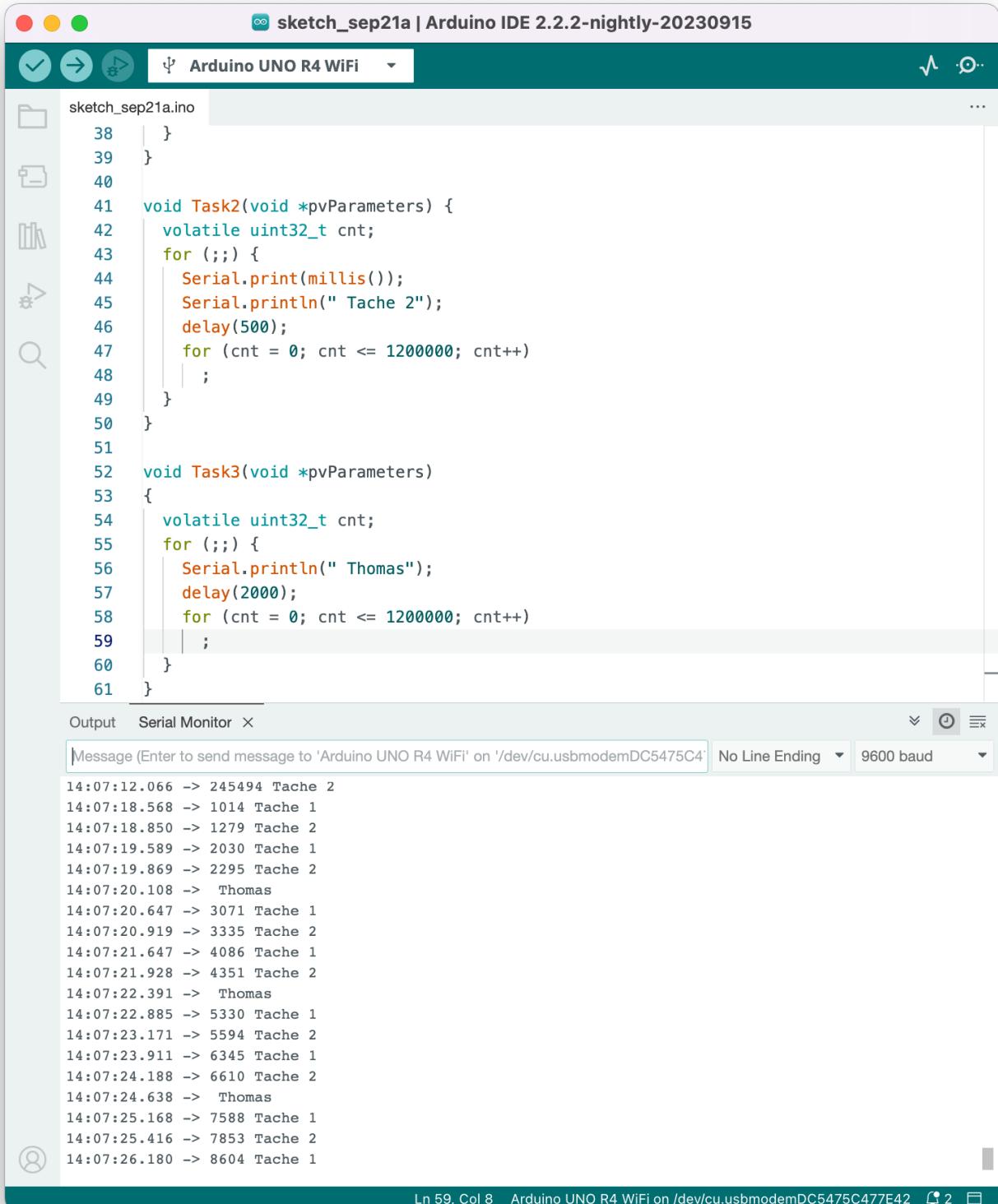
Ln 31, Col 12  Arduino UNO R4 WiFi on /dev/cu.usbmodemDC5475C477E42  ↻ 2  □
```

Après avoir commis l'erreur de décommenter le `delay()`, il sera re commenté pour la majorité des exercices suivants.

Par ailleurs, nous pouvons constater les différences entre les vitesses d'exécution entre nos différentes machines. Pour ma part, sur une machine avec un i7 (7eGen), $3\ 200\ 000 \approx 2\text{s}$.

Exercice 01

La troisième tâche est créée de façon identique à celles précédemment créées précédemment. Il était cependant nécessaire de modifier le délai afin de ne la faire apparaître



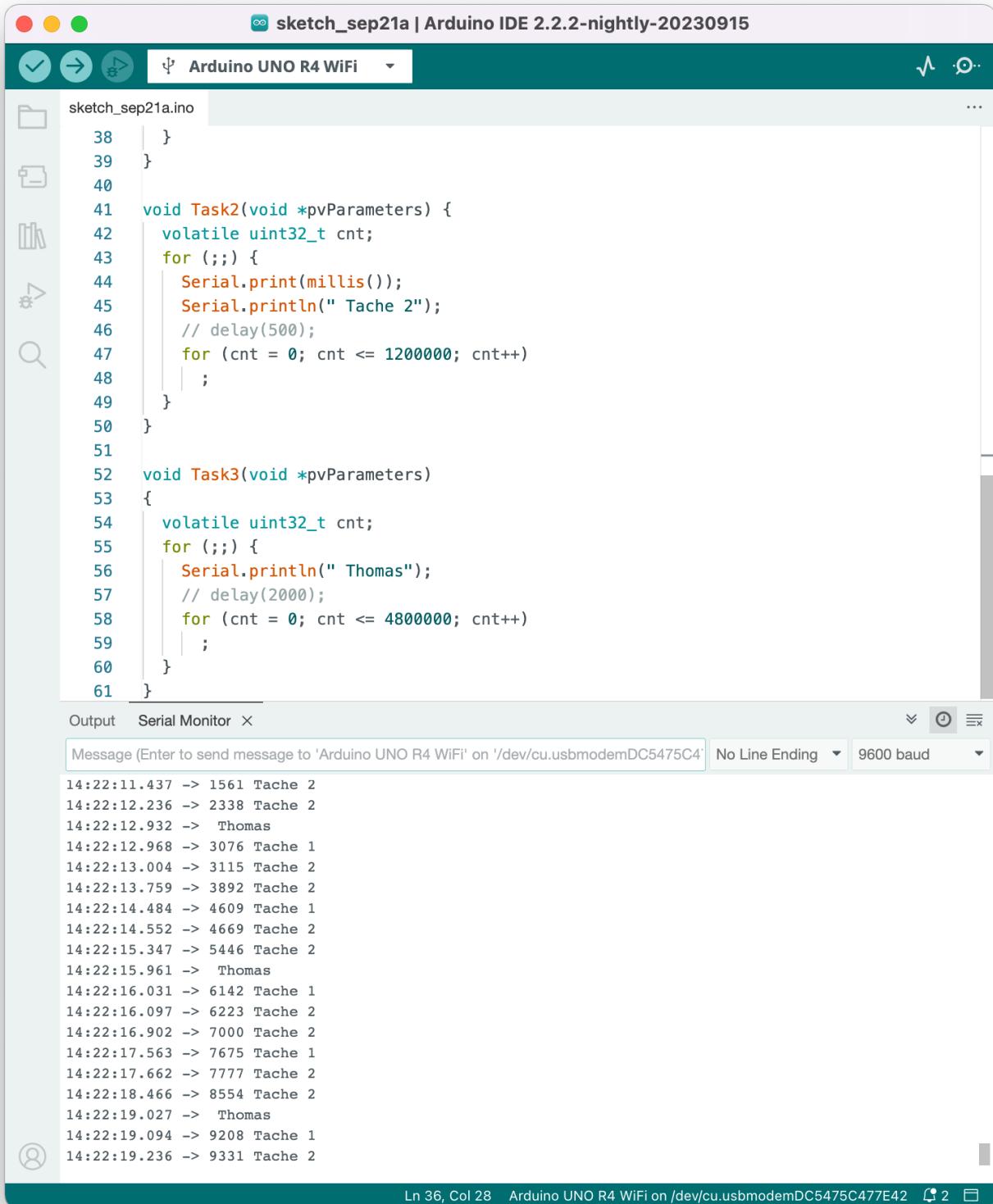
The screenshot shows the Arduino IDE interface. The top bar displays "sketch_sep21a | Arduino IDE 2.2.2-nightly-20230915". The left sidebar shows the file structure with "sketch_sep21a.ino" selected. The main code editor contains the following C++ code:

```
sketch_sep21a.ino
38 }
39 }
40
41 void Task2(void *pvParameters) {
42     volatile uint32_t cnt;
43     for (;;) {
44         Serial.print(millis());
45         Serial.println(" Tache 2");
46         delay(500);
47         for (cnt = 0; cnt <= 1200000; cnt++)
48             ;
49     }
50 }
51
52 void Task3(void *pvParameters)
53 {
54     volatile uint32_t cnt;
55     for (;;) {
56         Serial.println(" Thomas");
57         delay(2000);
58         for (cnt = 0; cnt <= 1200000; cnt++)
59             ;
60     }
61 }
```

The bottom section shows the "Serial Monitor" tab open, displaying a log of messages sent from the Arduino to the computer. The messages are timestamped and show alternating prints of "Tache 2" and "Thomas".

Time	Message
14:07:12.066	-> 245494 Tache 2
14:07:18.568	-> 1014 Tache 1
14:07:18.850	-> 1279 Tache 2
14:07:19.589	-> 2030 Tache 1
14:07:19.869	-> 2295 Tache 2
14:07:20.108	-> Thomas
14:07:20.647	-> 3071 Tache 1
14:07:20.919	-> 3335 Tache 2
14:07:21.647	-> 4086 Tache 1
14:07:21.928	-> 4351 Tache 2
14:07:22.391	-> Thomas
14:07:22.885	-> 5330 Tache 1
14:07:23.171	-> 5594 Tache 2
14:07:23.911	-> 6345 Tache 1
14:07:24.188	-> 6610 Tache 2
14:07:24.638	-> Thomas
14:07:25.168	-> 7588 Tache 1
14:07:25.416	-> 7853 Tache 2
14:07:26.180	-> 8604 Tache 1

Note : Il n'était pas autorisé d'utiliser delay(), le code modifié est donc le suivant, avec le résultat en console.



The screenshot shows the Arduino IDE interface with the following details:

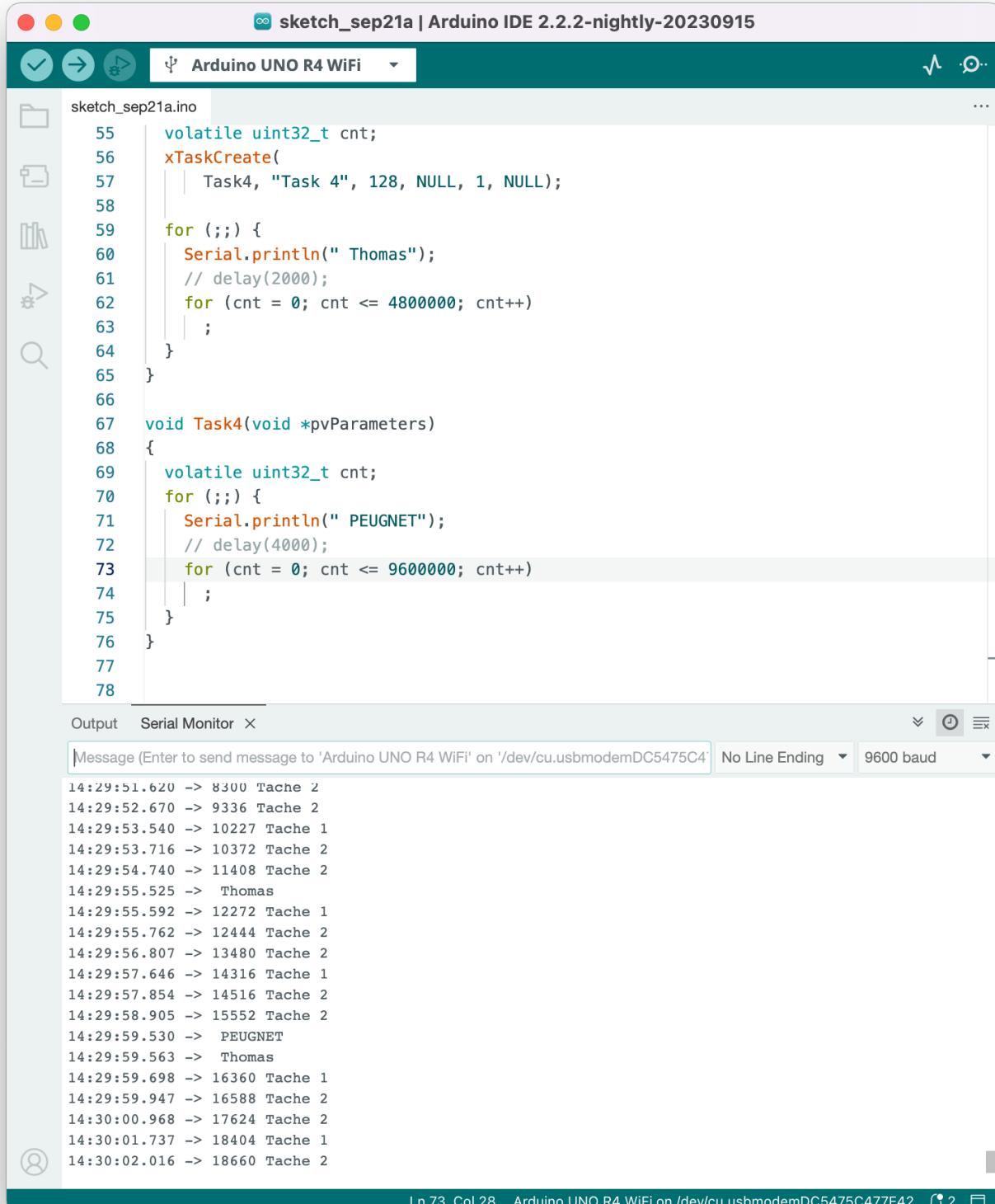
- Title Bar:** sketch_sep21a | Arduino IDE 2.2.2-nightly-20230915
- Sketch:** sketch_sep21a.ino
- Code Content:**

```
38  }
39 }
40
41 void Task2(void *pvParameters) {
42     volatile uint32_t cnt;
43     for (;;) {
44         Serial.print(millis());
45         Serial.println(" Tache 2");
46         // delay(500);
47         for (cnt = 0; cnt <= 1200000; cnt++)
48             ;
49     }
50 }
51
52 void Task3(void *pvParameters)
53 {
54     volatile uint32_t cnt;
55     for (;;) {
56         Serial.println(" Thomas");
57         // delay(2000);
58         for (cnt = 0; cnt <= 4800000; cnt++)
59             ;
60     }
61 }
```
- Serial Monitor:**
 - Message input field: "Enter to send message to 'Arduino UNO R4 WiFi' on '/dev/cu.usbmodemDC5475C4'"
 - Settings: No Line Ending, 9600 baud
 - Output:

```
14:22:11.437 -> 1561 Tache 2
14:22:12.236 -> 2338 Tache 2
14:22:12.932 -> Thomas
14:22:12.968 -> 3076 Tache 1
14:22:13.004 -> 3115 Tache 2
14:22:13.759 -> 3892 Tache 2
14:22:14.484 -> 4609 Tache 1
14:22:14.552 -> 4669 Tache 2
14:22:15.347 -> 5446 Tache 2
14:22:15.961 -> Thomas
14:22:16.031 -> 6142 Tache 1
14:22:16.097 -> 6223 Tache 2
14:22:16.902 -> 7000 Tache 2
14:22:17.563 -> 7675 Tache 1
14:22:17.662 -> 7777 Tache 2
14:22:18.466 -> 8554 Tache 2
14:22:19.027 -> Thomas
14:22:19.094 -> 9208 Tache 1
14:22:19.236 -> 9331 Tache 2
```
- Status Bar:** Ln 36, Col 28 Arduino UNO R4 WiFi on /dev/cu.usbmodemDC5475C477E42 2

Exercice 02

Le code permettant d'afficher le nom toutes les 4 secondes est le suivant.

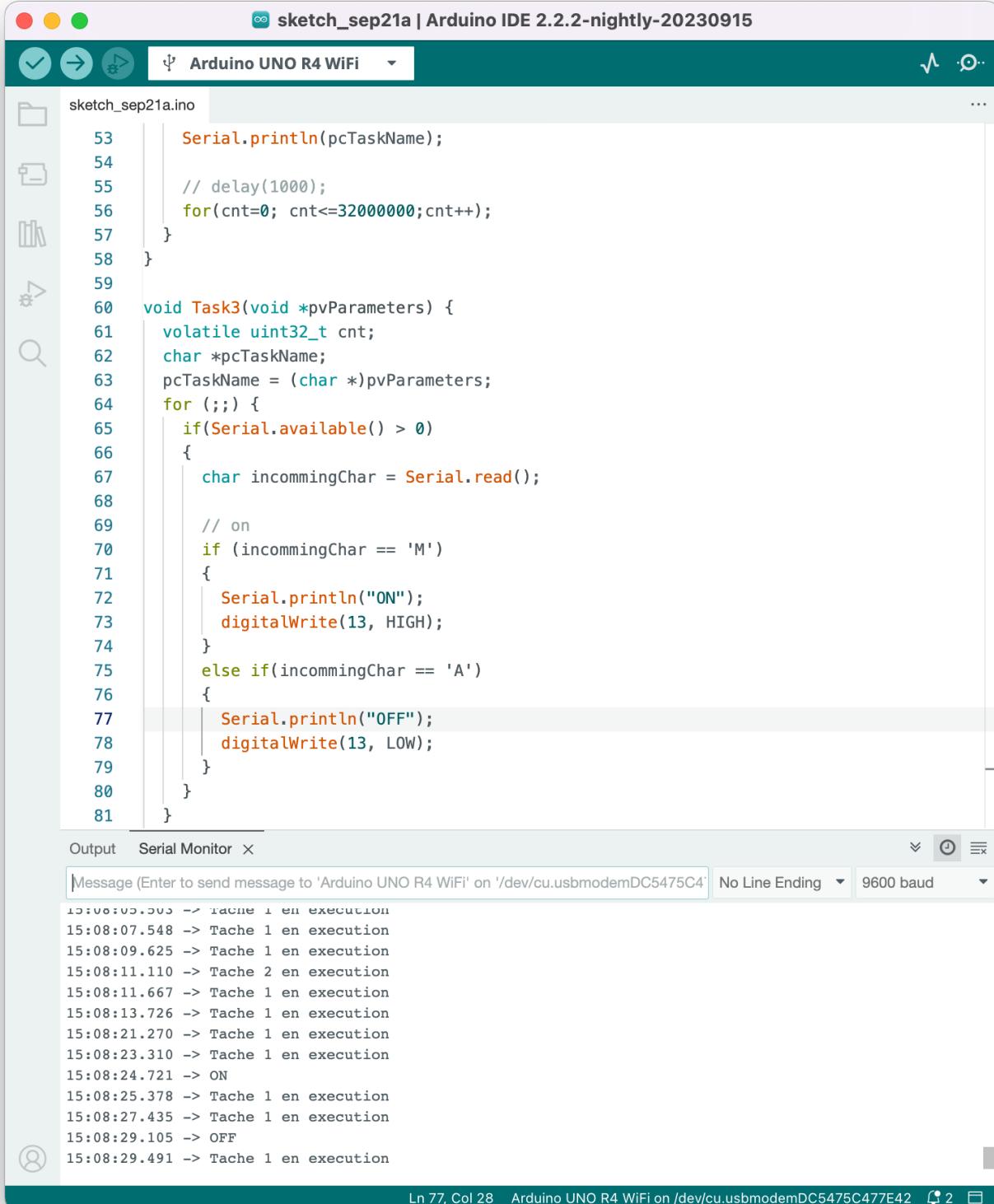


The screenshot shows the Arduino IDE interface with the following details:

- Title Bar:** sketch_sep21a | Arduino IDE 2.2.2-nightly-20230915
- Sketch Selection:** Arduino UNO R4 WiFi
- Code Editor:** The code is named sketch_sep21a.ino. It contains two tasks: Task4 and Task4. Task4 is triggered by a timer interrupt. Both tasks print their respective names to the serial port every 4 seconds. The code is as follows:55 volatile uint32_t cnt;
56 xTaskCreate(
57 Task4, "Task 4", 128, NULL, 1, NULL);
58
59 for (;;) {
60 Serial.println(" Thomas");
61 // delay(2000);
62 for (cnt = 0; cnt <= 4800000; cnt++)
63 ;
64 }
65
66
67 void Task4(void *pvParameters)
68 {
69 volatile uint32_t cnt;
70 for (;;) {
71 Serial.println(" PEUGNET");
72 // delay(4000);
73 for (cnt = 0; cnt <= 9600000; cnt++)
74 ;
75 }
76 }
- Serial Monitor:** The monitor shows the output of the printed strings. The text "Thomas" and "PEUGNET" are repeated every 4 seconds. The output is as follows:14:29:51.620 -> 8300 Tache 2
14:29:52.670 -> 9336 Tache 2
14:29:53.540 -> 10227 Tache 1
14:29:53.716 -> 10372 Tache 2
14:29:54.740 -> 11408 Tache 2
14:29:55.525 -> Thomas
14:29:55.592 -> 12272 Tache 1
14:29:55.762 -> 12444 Tache 2
14:29:56.807 -> 13480 Tache 2
14:29:57.646 -> 14316 Tache 1
14:29:57.854 -> 14516 Tache 2
14:29:58.905 -> 15552 Tache 2
14:29:59.530 -> PEUGNET
14:29:59.563 -> Thomas
14:29:59.698 -> 16360 Tache 1
14:29:59.947 -> 16588 Tache 2
14:30:00.968 -> 17624 Tache 2
14:30:01.737 -> 18404 Tache 1
14:30:02.016 -> 18660 Tache 2

Exercice 03

Le code permettant d'expérimenter le programme de l'exercice 07 du TP01 est le suivant.



The screenshot shows the Arduino IDE interface. The top bar displays "sketch_sep21a | Arduino IDE 2.2.2-nightly-20230915". The left sidebar shows the file structure with "sketch_sep21a.ino" selected. The main code editor window contains the following C++ code:

```
sketch_sep21a.ino
53     Serial.println(pcTaskName);
54
55     // delay(1000);
56     for(cnt=0; cnt<=32000000;cnt++);
57 }
58 }
59
60 void Task3(void *pvParameters) {
61     volatile uint32_t cnt;
62     char *pcTaskName;
63     pcTaskName = (char *)pvParameters;
64     for (;;) {
65         if(Serial.available() > 0)
66         {
67             char incommingChar = Serial.read();
68
69             // on
70             if (incommingChar == 'M')
71             {
72                 Serial.println("ON");
73                 digitalWrite(13, HIGH);
74             }
75             else if(incommingChar == 'A')
76             {
77                 Serial.println("OFF");
78                 digitalWrite(13, LOW);
79             }
80         }
81     }
}
```

The bottom section shows the "Serial Monitor" tab open, displaying the following log output:

```
Message (Enter to send message to 'Arduino UNO R4 WiFi' on '/dev/cu.usbmodemDC5475C4'
15:08:00.000 -> Tache 1 en execution
15:08:07.548 -> Tache 1 en execution
15:08:09.625 -> Tache 1 en execution
15:08:11.110 -> Tache 2 en execution
15:08:11.667 -> Tache 1 en execution
15:08:13.726 -> Tache 1 en execution
15:08:21.270 -> Tache 1 en execution
15:08:23.310 -> Tache 1 en execution
15:08:24.721 -> ON
15:08:25.378 -> Tache 1 en execution
15:08:27.435 -> Tache 1 en execution
15:08:29.105 -> OFF
15:08:29.491 -> Tache 1 en execution
```

The status bar at the bottom indicates "Ln 77, Col 28" and "Arduino UNO R4 WiFi on /dev/cu.usbmodemDC5475C477E42".

Afin de ne pas devoir faire de photo, nous avons affiché **ON** ou **OFF** lorsque la LED apparaît allumée ou éteinte sur la carte Arduino.

Exercice 05

Le code permettant d'afficher ces deux tâches est le suivant.



The screenshot shows the Arduino IDE interface with the title bar "sketch_sep21a | Arduino IDE 2.2.2-nightly-20230915". The central area displays the code for "sketch_sep21a.ino". The code uses the FreeRTOS library to create two tasks: Task1 and Task2. Task1 prints its name repeatedly, while Task2 does nothing. The "Serial Monitor" tab is open at the bottom, showing the output of the tasks.

```
#include <Arduino_FreeRTOS.h>

const char *pcTextForTask1 = "Tache 1 en execution";
const char *pcTextForTask2 = "Tache 2 en execution";

void Task1(void *pvParameters);
void Task2(void *pvParameters);

void setup() {
    // put your setup code here, to run once:
    Serial.begin(9600);
    while (!Serial) {
        ;
    }

    xTaskCreate(
        Task1, "Task 1", 128, (void*) pcTextForTask1, 1, NULL);
    xTaskCreate(
        Task2, "Task 2", 128, (void*) pcTextForTask2, 1, NULL);
    vTaskStartScheduler();
}

void loop() {
    //Empty. Things are done in Tasks
}

void Task1(void *pvParameters) {
    volatile uint32_t cnt;
    char *pcTaskName;
    pcTaskName = (char *)pvParameters;
    for (;;) {
        Serial.println(pcTaskName);
        // delay(1000);
        for (cnt = 0; cnt <= 3200000; cnt++)
            ;
    }
}
```

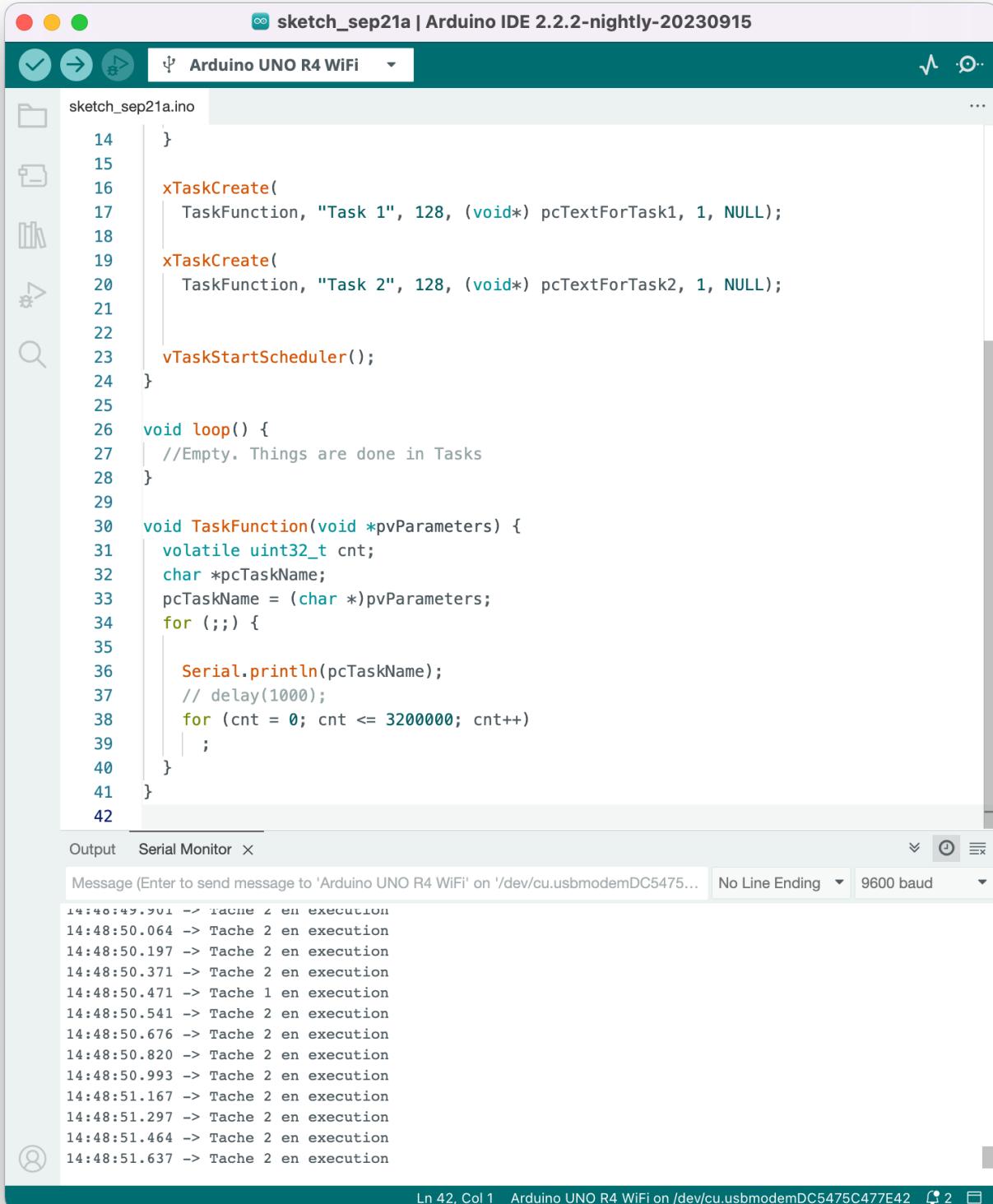
Output Serial Monitor

Message (Enter to send message to 'Arduino UNO R4 WiFi' on '/dev/cu.usbmodemDC5475C477E42') No Line Ending 9600 baud

Ln 37, Col 15 Arduino UNO R4 WiFi on /dev/cu.usbmodemDC5475C477E42 2

Exercice 05 - suite

Le code permettant de refaire l'exercice 05 mais en effectuant une optimisation du code est le suivant.



The screenshot shows the Arduino IDE interface with the following details:

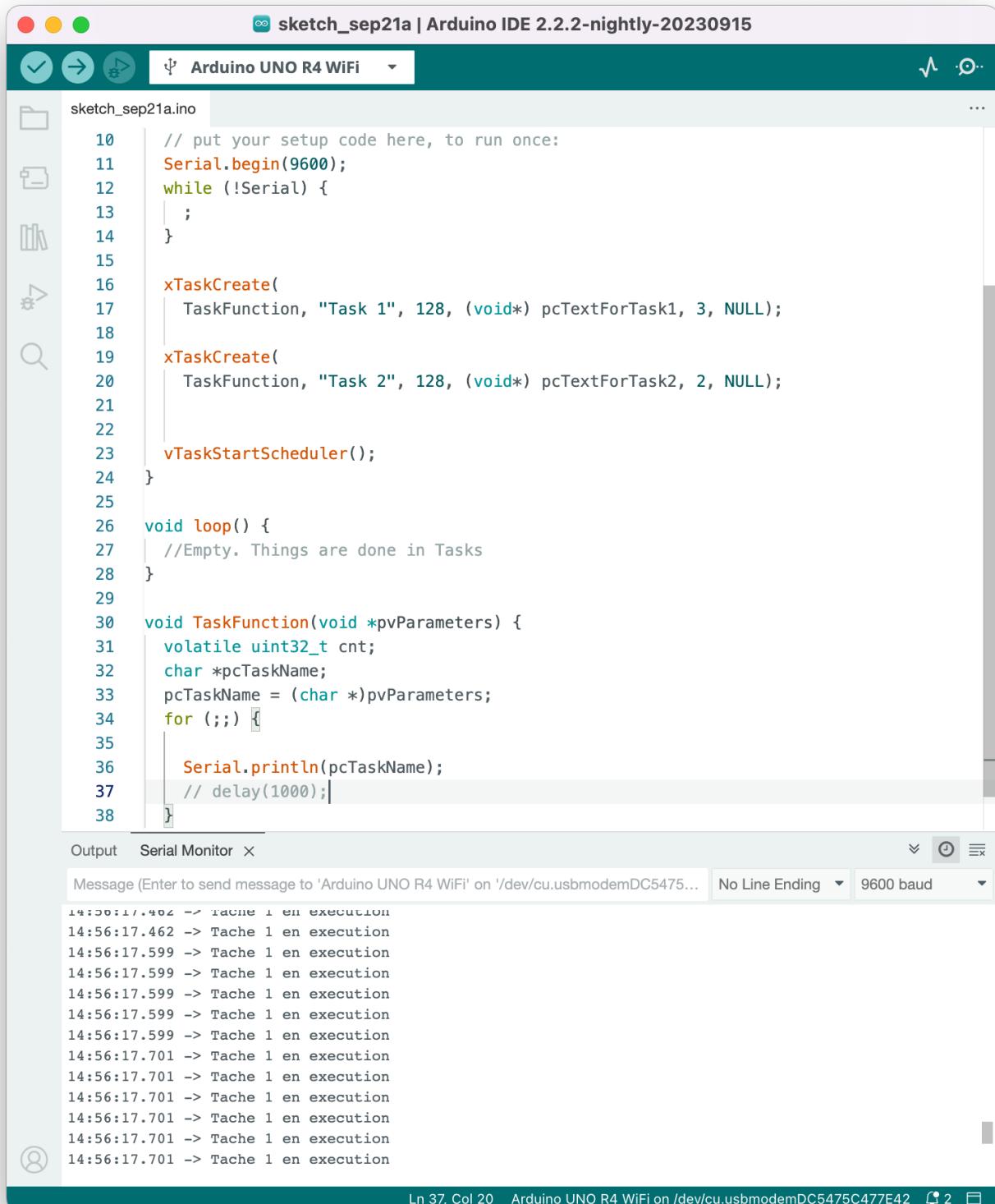
- Title Bar:** sketch_sep21a | Arduino IDE 2.2.2-nightly-20230915
- Sketch Name:** sketch_sep21a.ino
- Board:** Arduino Uno R4 WiFi
- Code Content:**

```
14 }
15
16 xTaskCreate(
17     TaskFunction, "Task 1", 128, (void*) pcTextForTask1, 1, NULL);
18
19 xTaskCreate(
20     TaskFunction, "Task 2", 128, (void*) pcTextForTask2, 1, NULL);
21
22 vTaskStartScheduler();
23 }
24
25
26 void loop() {
27     //Empty. Things are done in Tasks
28 }
29
30 void TaskFunction(void *pvParameters) {
31     volatile uint32_t cnt;
32     char *pcTaskName;
33     pcTaskName = (char *)pvParameters;
34     for (;;) {
35
36         Serial.println(pcTaskName);
37         // delay(1000);
38         for (cnt = 0; cnt <= 3200000; cnt++)
39             ;
40     }
41 }
```
- Serial Monitor:**
 - Message (Enter to send message to 'Arduino Uno R4 WiFi' on '/dev/cu.usbmodemDC5475C477E42...')
 - No Line Ending
 - 9600 baud
 - Output:
14:48:49.901 -> Tache 2 en execution
14:48:50.064 -> Tache 2 en execution
14:48:50.197 -> Tache 2 en execution
14:48:50.371 -> Tache 2 en execution
14:48:50.471 -> Tache 1 en execution
14:48:50.541 -> Tache 2 en execution
14:48:50.676 -> Tache 2 en execution
14:48:50.820 -> Tache 2 en execution
14:48:50.993 -> Tache 2 en execution
14:48:51.167 -> Tache 2 en execution
14:48:51.297 -> Tache 2 en execution
14:48:51.464 -> Tache 2 en execution
14:48:51.637 -> Tache 2 en execution

Nous pouvons constater un comportement identique.

Exemple

Le code suivant permet d'expérimenter le comportement qui aura lieu si jamais la priorité de deux tâches identiques n'est pas la même. Nous pouvons constater que seule la tâche de priorité supérieure est traitée.



The screenshot shows the Arduino IDE interface with the following details:

- Title Bar:** sketch_sep21a | Arduino IDE 2.2.2-nightly-20230915
- Sketch:** sketch_sep21a.ino
- Code Content:**

```
10 // put your setup code here, to run once:
11 Serial.begin(9600);
12 while (!Serial) {
13     ;
14 }
15
16 xTaskCreate(
17     TaskFunction, "Task 1", 128, (void*) pcTextForTask1, 3, NULL);
18
19 xTaskCreate(
20     TaskFunction, "Task 2", 128, (void*) pcTextForTask2, 2, NULL);
21
22 vTaskStartScheduler();
23
24 }
25
26 void loop() {
27     //Empty. Things are done in Tasks
28 }
29
30 void TaskFunction(void *pvParameters) {
31     volatile uint32_t cnt;
32     char *pcTaskName;
33     pcTaskName = (char *)pvParameters;
34     for (;;) {
35
36         Serial.println(pcTaskName);
37         // delay(1000);
38     }
}
```
- Output Tab:** Serial Monitor
- Serial Monitor Content:**

```
Message (Enter to send message to 'Arduino UNO R4 WiFi' on '/dev/cu.usbmodemDC5475C477E42...')
```

Execution logs from the serial monitor:

```
14:56:17.402 -> Tache 1 en execution
14:56:17.462 -> Tache 1 en execution
14:56:17.599 -> Tache 1 en execution
14:56:17.701 -> Tache 1 en execution
```