University of
South-Eastern Norway

www.usn.no

IIA2017: Industrial IT

Multitasking and real-time assignment

Isak Skeie, 245362

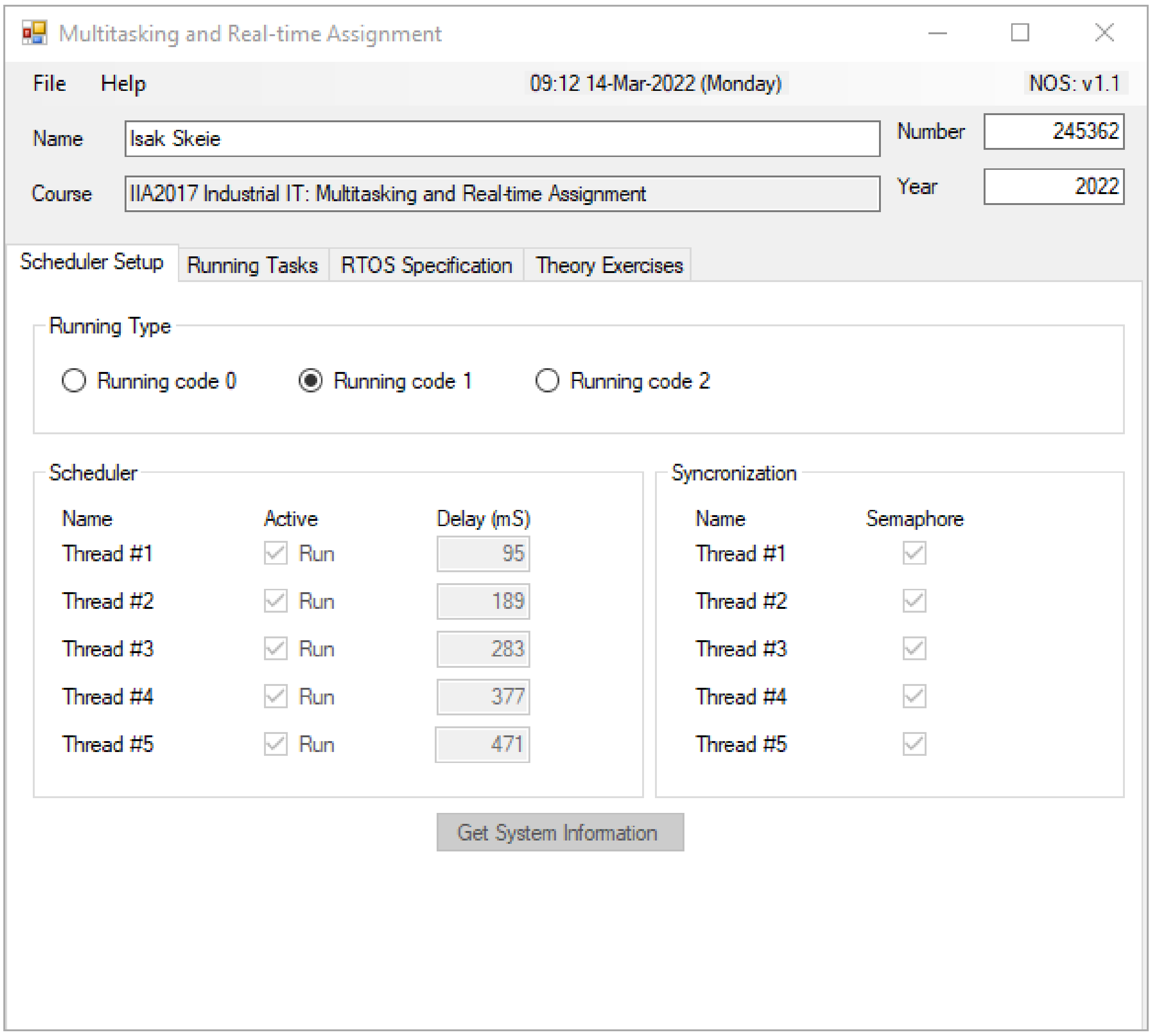
Faculty of Technology, Natural sciences and Maritime Sciences
Campus Porsgrunn

Contents

1 Introduction

The purpose of this assignment is to be familiar with Multitasking and real time systems. This is done through different tasks, highlighting the different aspects. Prior to this assignment, theory given in the lecture material is needed.

1.1 Scheduler Setup



09:12 14-Mar-2022 (Monday)

NOS: v1.1

Name	Isak Skeie
Course	IIA2017 Industrial IT: Multitasking and Real-time Assignment
Number	245362
Year	2022

Running Type: Running code 0 [✓ Running code 1] Running code 2

Scheduler

Name	Active	Delay (mS)
Thread #1	✓ Run	95
Thread #2	✓ Run	189
Thread #3	✓ Run	283
Thread #4	✓ Run	377
Thread #5	✓ Run	471

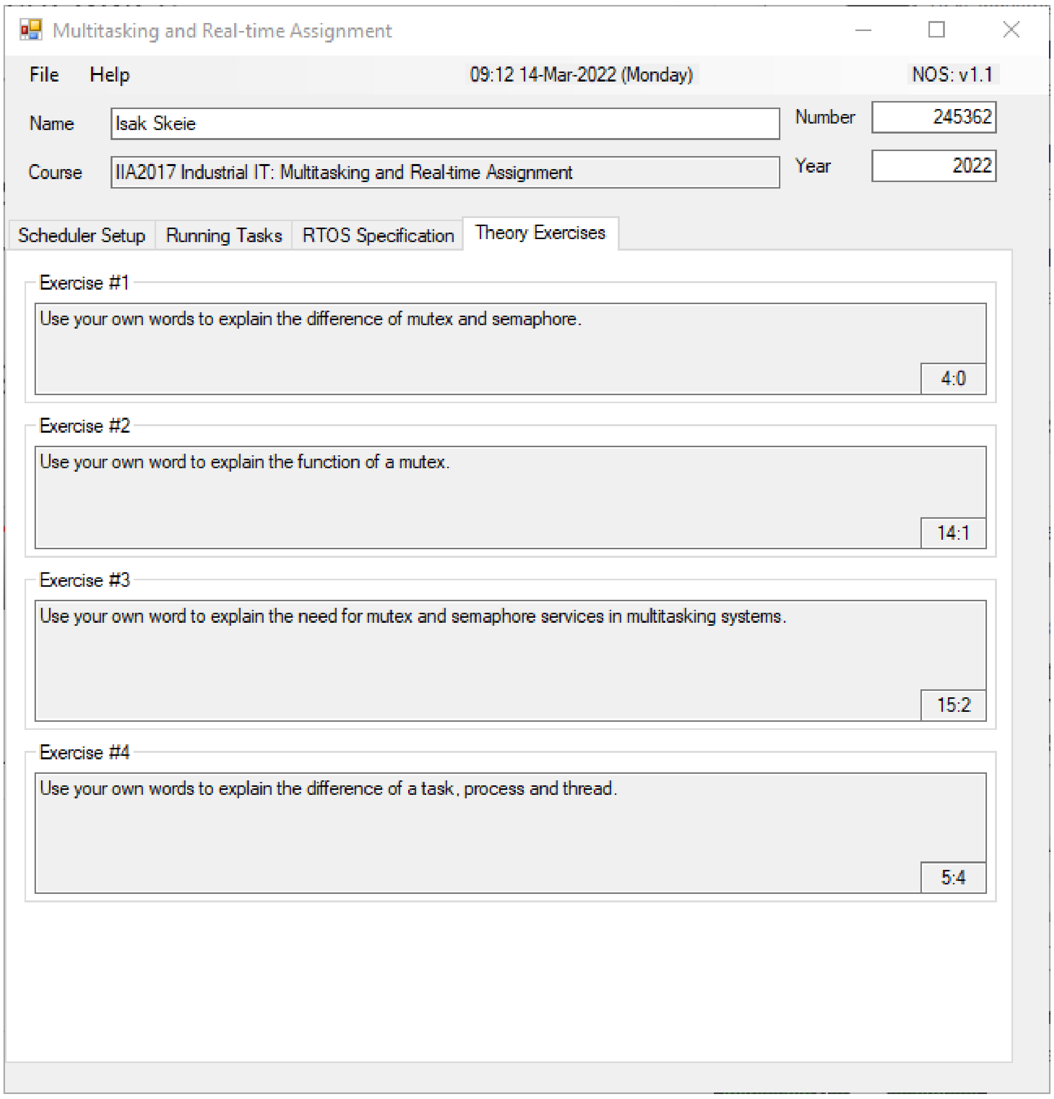
Syncronization

Name	Semaphore
Thread #1	✓
Thread #2	✓
Thread #3	✓
Thread #4	✓
Thread #5	✓

Figure 1: Specifications for the scheduler

1.2 Theory

1. The difference between Mutex and Semaphore is the usage in MultiTasking in programming. Both Mutex and Semaphore is used to allocate resources to tasks, with the difference being that a tasks uses Mutex to lock the task, while Semaphore is used to signal the use of resources to tasks. [1]
2. The function of mutex is to make sure a resource is only used by one task at time. This is achieved by the task using Mutex to lock the resource.
3. Without a way to control tasks use of resources, several tasks could try accessing one resource at the same time. This will cause issues with tasks writing over memory before its used by another task. As well as Deadlocking. [2]
4. The main difference between a Process, Threads and Task is their relation to memory. A Process uses a Memory Management unit for memory access, so does the Thread, but with the addition that it shares data with its process. A task runs within a Thread, and doesn’t utilize MMU. [3]


Multitasking and Real-time Assignment

File Help 09:12 14-Mar-2022 (Monday) NOS: v1.1

Name	Isak Skeie	Number	245362
Course	IIA2017 Industrial IT: Multitasking and Real-time Assignment	Year	2022

Exercise #1

Use your own words to explain the difference of mutex and semaphore.

4:0

Exercise #2

Use your own word to explain the function of a mutex.

14:1

Exercise #3

Use your own word to explain the need for mutex and semaphore services in multitasking systems.

15:2

Exercise #4

Use your own words to explain the difference of a task, process and thread.

5:4

Figure 2: Theorethical questions

2 Evaluation Of a Multitasking system

<Start each chapter with a short introduction about what the chapter is about.>

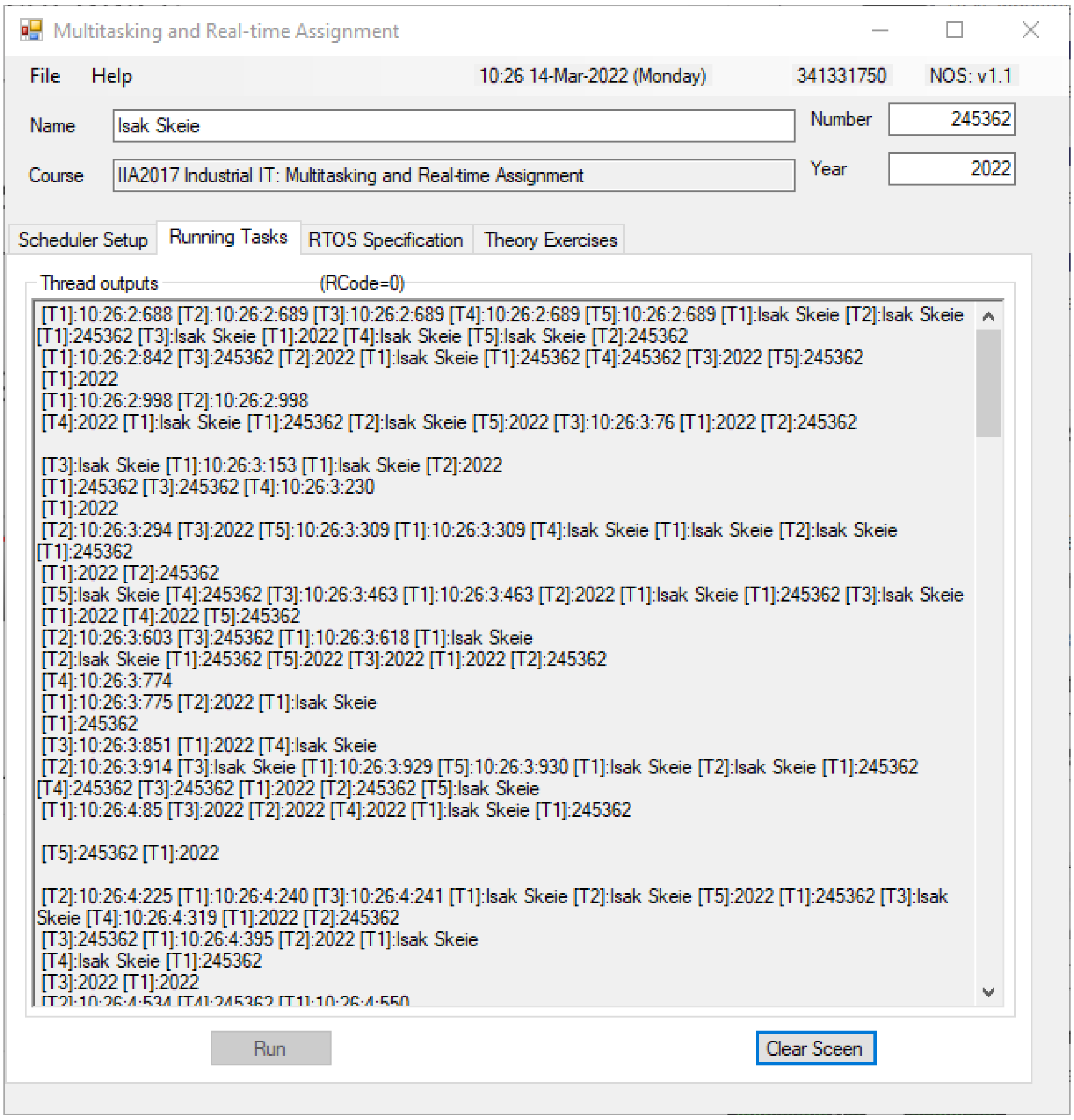
2.1 Analysis

The System Sequence Diagram shows a simple process for the threads to execute. With the shared resource being the communication with the console. This is used to display information for the user. This common resources doesn't affect the running time in any significant way. The delay however, is the component deciding the running time.

To be able to share resources between threads, one could use Semaphores or Mutex. Theses components make sure resources are used correctly by Thread. With the addition of Thread.Sleep(), its possible for threads to share resources in runtime.

2.1.1 Running Code 0

In Running Code 0, all the threads run in 137 iterations of a loop. Between each loop iteration, there is a delay in the threads, this is given from the parameters in the "MultiTasking and Realtime"-application. The Duration for each Thread is measured from the printed information and verified by multiplying number of loops by delay. Based on the output, its clear that all the threads start concurrently and finish in the order of the delay between the loops. With a noticeably difference in finish time, depending on the delay in the loops. The start and end of Code 0 is shown in figure # and #



10:26 14-Mar-2022 (Monday) 341331750 NOS: v1.1

Name: Isak Skeie

Course: IIA2017 Industrial IT: Multitasking and Real-time Assignment

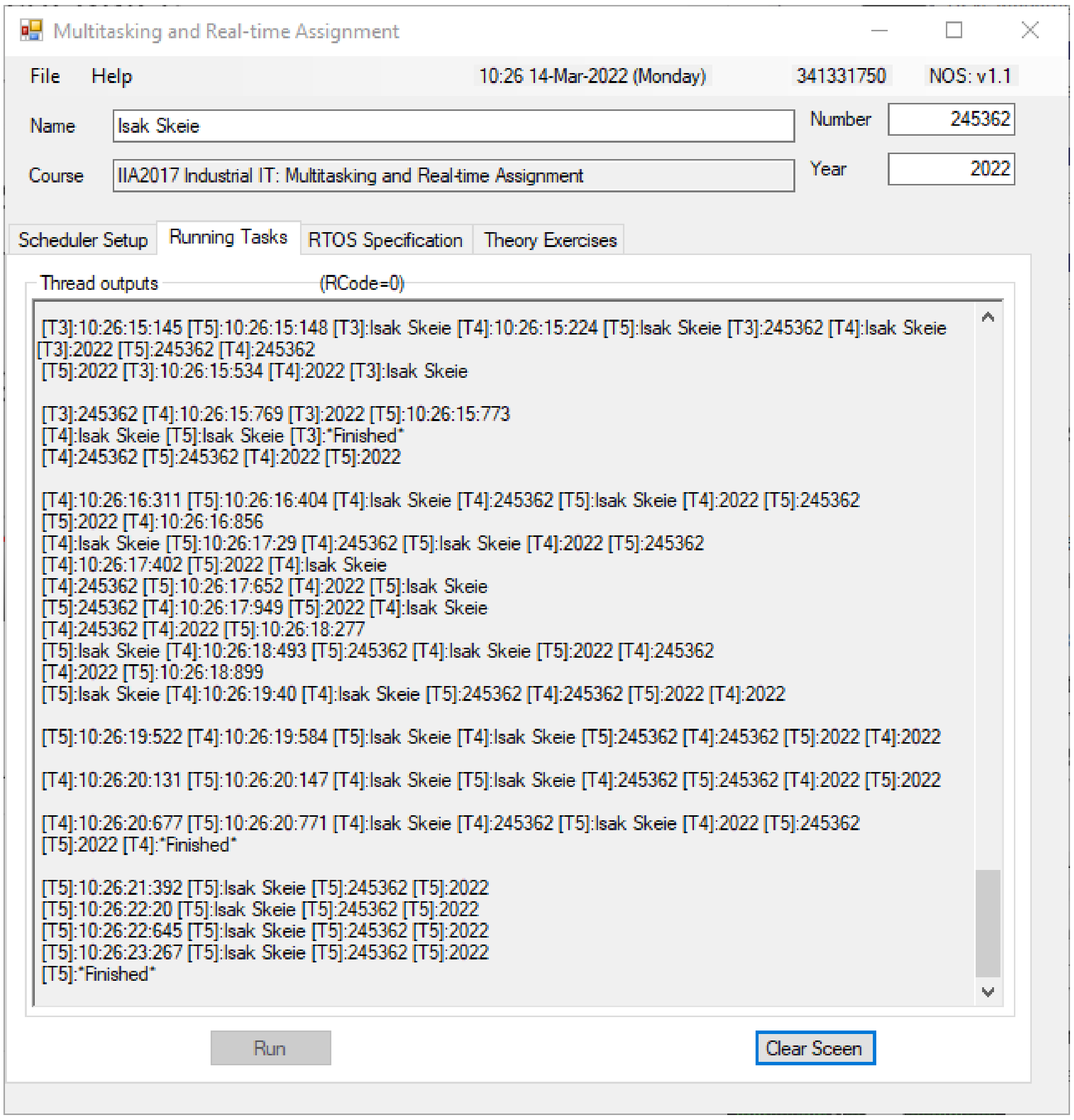
Number: 245362

Year: 2022

Thread outputs (RCode=0)

```
 [T1]:10:26:2:688 [T2]:10:26:2:689 [T3]:10:26:2:689 [T4]:10:26:2:689 [T5]:10:26:2:689 [T1]:Isak Skeie [T2]:Isak Skeie
[T1]:245362 [T3]:Isak Skeie [T1]:2022 [T4]:Isak Skeie [T5]:Isak Skeie [T2]:245362
 [T1]:10:26:2:842 [T3]:245362 [T2]:2022 [T1]:Isak Skeie [T1]:245362 [T4]:245362 [T3]:2022 [T5]:245362
 [T1]:2022
 [T1]:10:26:2:998 [T2]:10:26:2:998
 [T4]:2022 [T1]:Isak Skeie [T1]:245362 [T2]:Isak Skeie [T5]:2022 [T3]:10:26:3:76 [T1]:2022 [T2]:245362

 [T3]:Isak Skeie [T1]:10:26:3:153 [T1]:Isak Skeie [T2]:2022
 [T1]:245362 [T3]:245362 [T4]:10:26:3:230
 [T1]:2022
 [T2]:10:26:3:294 [T3]:2022 [T5]:10:26:3:309 [T1]:10:26:3:309 [T4]:Isak Skeie [T1]:Isak Skeie [T2]:Isak Skeie
[T1]:245362
 [T1]:2022 [T2]:245362
 [T5]:Isak Skeie [T4]:245362 [T3]:10:26:3:463 [T1]:10:26:3:463 [T2]:2022 [T1]:Isak Skeie [T1]:245362 [T3]:Isak Skeie
 [T1]:2022 [T4]:2022 [T5]:245362
 [T2]:10:26:3:603 [T3]:245362 [T1]:10:26:3:618 [T1]:Isak Skeie
 [T2]:Isak Skeie [T1]:245362 [T5]:2022 [T3]:2022 [T1]:2022 [T2]:245362
 [T4]:10:26:3:774
 [T1]:10:26:3:775 [T2]:2022 [T1]:Isak Skeie
 [T1]:245362
 [T3]:10:26:3:851 [T1]:2022 [T4]:Isak Skeie
 [T2]:10:26:3:914 [T3]:Isak Skeie [T1]:10:26:3:929 [T5]:10:26:3:930 [T1]:Isak Skeie [T2]:Isak Skeie [T1]:245362
[T4]:245362 [T3]:245362 [T1]:2022 [T2]:245362 [T5]:Isak Skeie
 [T1]:10:26:4:85 [T3]:2022 [T2]:2022 [T4]:2022 [T1]:Isak Skeie [T1]:245362

 [T5]:245362 [T1]:2022

 [T2]:10:26:4:225 [T1]:10:26:4:240 [T3]:10:26:4:241 [T1]:Isak Skeie [T2]:Isak Skeie [T5]:2022 [T1]:245362 [T3]:Isak
Skeie [T4]:10:26:4:319 [T1]:2022 [T2]:245362
 [T3]:245362 [T1]:10:26:4:395 [T2]:2022 [T1]:Isak Skeie
 [T4]:Isak Skeie [T1]:245362
 [T3]:2022 [T1]:2022
 [T2]:10:26:4:534 [T4]:245362 [T1]:10:26:4:550
```

Figure 3: Start of Running Code 0



10:26 14-Mar-2022 (Monday) 341331750 NOS: v1.1

Name: Isak Skeie
Number: 245362
Course: IIA2017 Industrial IT: Multitasking and Real-time Assignment
Year: 2022

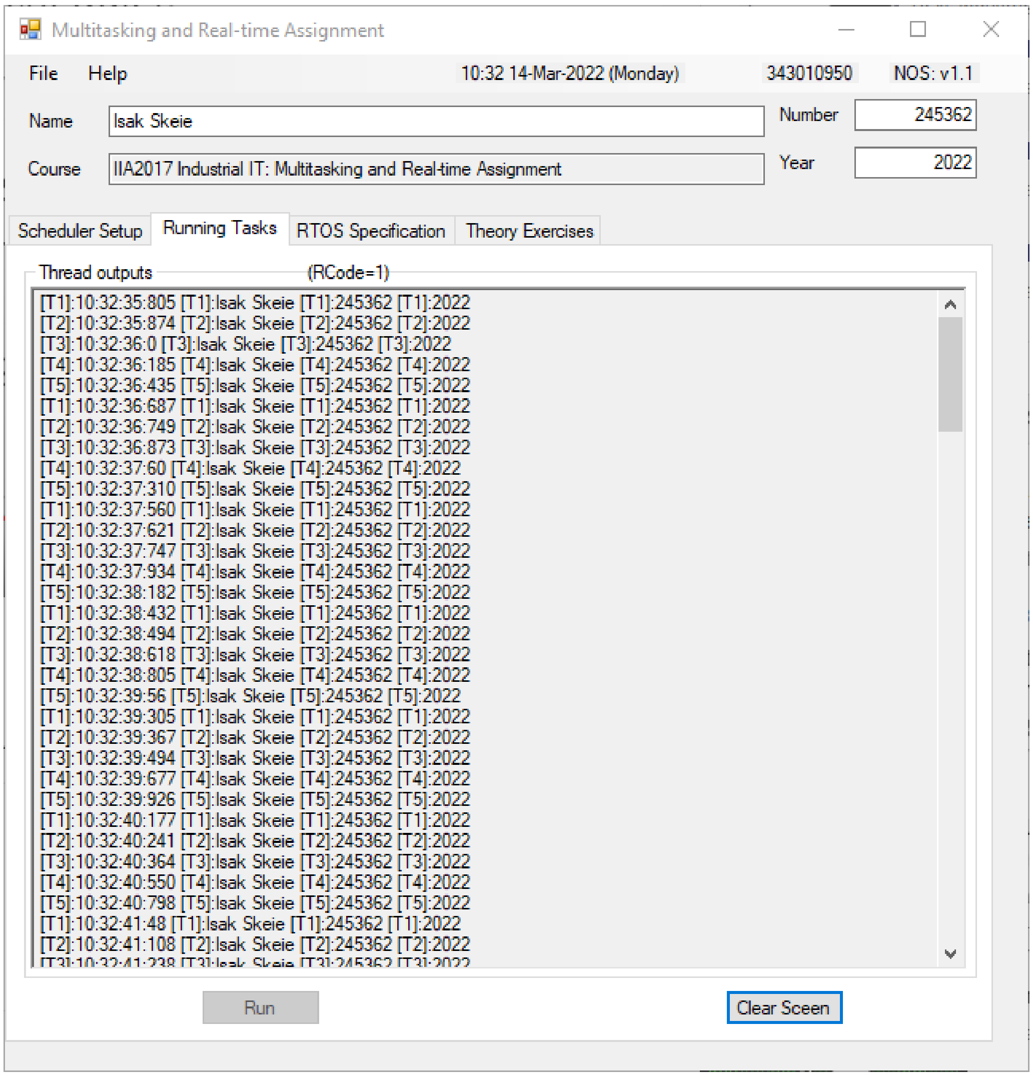
Thread outputs (RCode=0)

```
[T3]:10:26:15:145 [T5]:10:26:15:148 [T3]:Isak Skeie [T4]:10:26:15:224 [T5]:Isak Skeie [T3]:245362 [T4]:Isak Skeie
[T3]:2022 [T5]:245362 [T4]:245362
[T5]:2022 [T3]:10:26:15:534 [T4]:2022 [T3]:Isak Skeie

[T3]:245362 [T4]:10:26:15:769 [T3]:2022 [T5]:10:26:15:773
[T4]:Isak Skeie [T5]:Isak Skeie [T3]:*Finished*
[T4]:245362 [T5]:245362 [T4]:2022 [T5]:2022

[T4]:10:26:16:311 [T5]:10:26:16:404 [T4]:Isak Skeie [T4]:245362 [T5]:Isak Skeie [T4]:2022 [T5]:245362
[T5]:2022 [T4]:10:26:16:856
[T4]:Isak Skeie [T5]:10:26:17:29 [T4]:245362 [T5]:Isak Skeie [T4]:2022 [T5]:245362
[T4]:10:26:17:402 [T5]:2022 [T4]:Isak Skeie
[T4]:245362 [T5]:10:26:17:652 [T4]:2022 [T5]:Isak Skeie
[T5]:245362 [T4]:10:26:17:949 [T5]:2022 [T4]:Isak Skeie
[T4]:245362 [T4]:2022 [T5]:10:26:18:277
[T5]:Isak Skeie [T4]:10:26:18:493 [T5]:245362 [T4]:Isak Skeie [T5]:2022 [T4]:245362
[T4]:2022 [T5]:10:26:18:899
[T5]:Isak Skeie [T4]:10:26:19:40 [T4]:Isak Skeie [T5]:245362 [T4]:245362 [T5]:2022 [T4]:2022

[T5]:10:26:19:522 [T4]:10:26:19:584 [T5]:Isak Skeie [T4]:Isak Skeie [T5]:245362 [T4]:245362 [T5]:2022 [T4]:2022

[T4]:10:26:20:131 [T5]:10:26:20:147 [T4]:Isak Skeie [T5]:Isak Skeie [T4]:245362 [T5]:245362 [T4]:2022 [T5]:2022

[T4]:10:26:20:677 [T5]:10:26:20:771 [T4]:Isak Skeie [T4]:245362 [T5]:Isak Skeie [T4]:2022 [T5]:245362
[T5]:2022 [T4]:*Finished*

[T5]:10:26:21:392 [T5]:Isak Skeie [T5]:245362 [T5]:2022
[T5]:10:26:22:20 [T5]:Isak Skeie [T5]:245362 [T5]:2022
[T5]:10:26:22:645 [T5]:Isak Skeie [T5]:245362 [T5]:2022
[T5]:10:26:23:267 [T5]:Isak Skeie [T5]:245362 [T5]:2022
[T5]:*Finished*
```

Figure 4:End of Running Code 0

2.1.2 Running Code 1

For Running code 1, the threads go through 137 iterations. For this run, the delay in the loops of the threads are disregarded. This is because the threads run in an ascending queue. The total run time of all the threads will in this case have a minimal difference. The start of the process is shown in figure #, and the end of the run in figure #



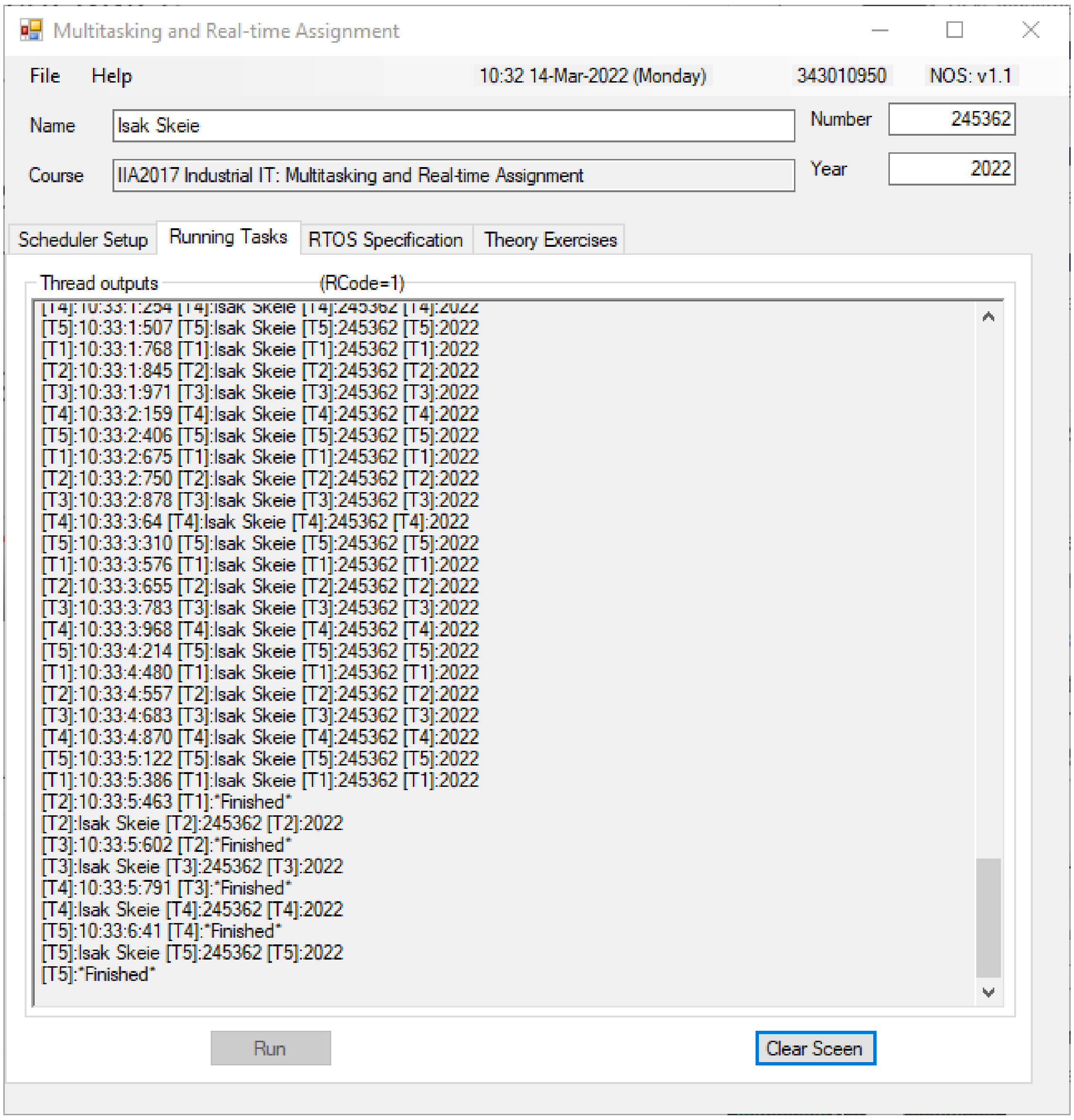
10:32 14-Mar-2022 (Monday) 343010950 NOS: v1.1

Name: Isak Skeie
Number: 245362
Course: IIA2017 Industrial IT: Multitasking and Real-time Assignment
Year: 2022

Thread outputs (RCode=1)

```
[T1]:10:32:35:805 [T1]:Isak Skeie [T1]:245362 [T1]:2022
[T2]:10:32:35:874 [T2]:Isak Skeie [T2]:245362 [T2]:2022
[T3]:10:32:36:0 [T3]:Isak Skeie [T3]:245362 [T3]:2022
[T4]:10:32:36:185 [T4]:Isak Skeie [T4]:245362 [T4]:2022
[T5]:10:32:36:435 [T5]:Isak Skeie [T5]:245362 [T5]:2022
[T1]:10:32:36:687 [T1]:Isak Skeie [T1]:245362 [T1]:2022
[T2]:10:32:36:749 [T2]:Isak Skeie [T2]:245362 [T2]:2022
[T3]:10:32:36:873 [T3]:Isak Skeie [T3]:245362 [T3]:2022
[T4]:10:32:37:60 [T4]:Isak Skeie [T4]:245362 [T4]:2022
[T5]:10:32:37:310 [T5]:Isak Skeie [T5]:245362 [T5]:2022
[T1]:10:32:37:560 [T1]:Isak Skeie [T1]:245362 [T1]:2022
[T2]:10:32:37:621 [T2]:Isak Skeie [T2]:245362 [T2]:2022
[T3]:10:32:37:747 [T3]:Isak Skeie [T3]:245362 [T3]:2022
[T4]:10:32:37:934 [T4]:Isak Skeie [T4]:245362 [T4]:2022
[T5]:10:32:38:182 [T5]:Isak Skeie [T5]:245362 [T5]:2022
[T1]:10:32:38:432 [T1]:Isak Skeie [T1]:245362 [T1]:2022
[T2]:10:32:38:494 [T2]:Isak Skeie [T2]:245362 [T2]:2022
[T3]:10:32:38:618 [T3]:Isak Skeie [T3]:245362 [T3]:2022
[T4]:10:32:38:805 [T4]:Isak Skeie [T4]:245362 [T4]:2022
[T5]:10:32:39:56 [T5]:Isak Skeie [T5]:245362 [T5]:2022
[T1]:10:32:39:305 [T1]:Isak Skeie [T1]:245362 [T1]:2022
[T2]:10:32:39:367 [T2]:Isak Skeie [T2]:245362 [T2]:2022
[T3]:10:32:39:494 [T3]:Isak Skeie [T3]:245362 [T3]:2022
[T4]:10:32:39:677 [T4]:Isak Skeie [T4]:245362 [T4]:2022
[T5]:10:32:39:926 [T5]:Isak Skeie [T5]:245362 [T5]:2022
[T1]:10:32:40:177 [T1]:Isak Skeie [T1]:245362 [T1]:2022
[T2]:10:32:40:241 [T2]:Isak Skeie [T2]:245362 [T2]:2022
[T3]:10:32:40:364 [T3]:Isak Skeie [T3]:245362 [T3]:2022
[T4]:10:32:40:550 [T4]:Isak Skeie [T4]:245362 [T4]:2022
[T5]:10:32:40:798 [T5]:Isak Skeie [T5]:245362 [T5]:2022
[T1]:10:32:41:48 [T1]:Isak Skeie [T1]:245362 [T1]:2022
[T2]:10:32:41:108 [T2]:Isak Skeie [T2]:245362 [T2]:2022
[T3]:10:32:41:238 [T3]:Isak Skeie [T3]:245362 [T3]:2022
```

Figure 5: Start of running code 1



Name: Isak Skeie
Number: 245362
Course: IIA2017 Industrial IT: Multitasking and Real-time Assignment
Year: 2022

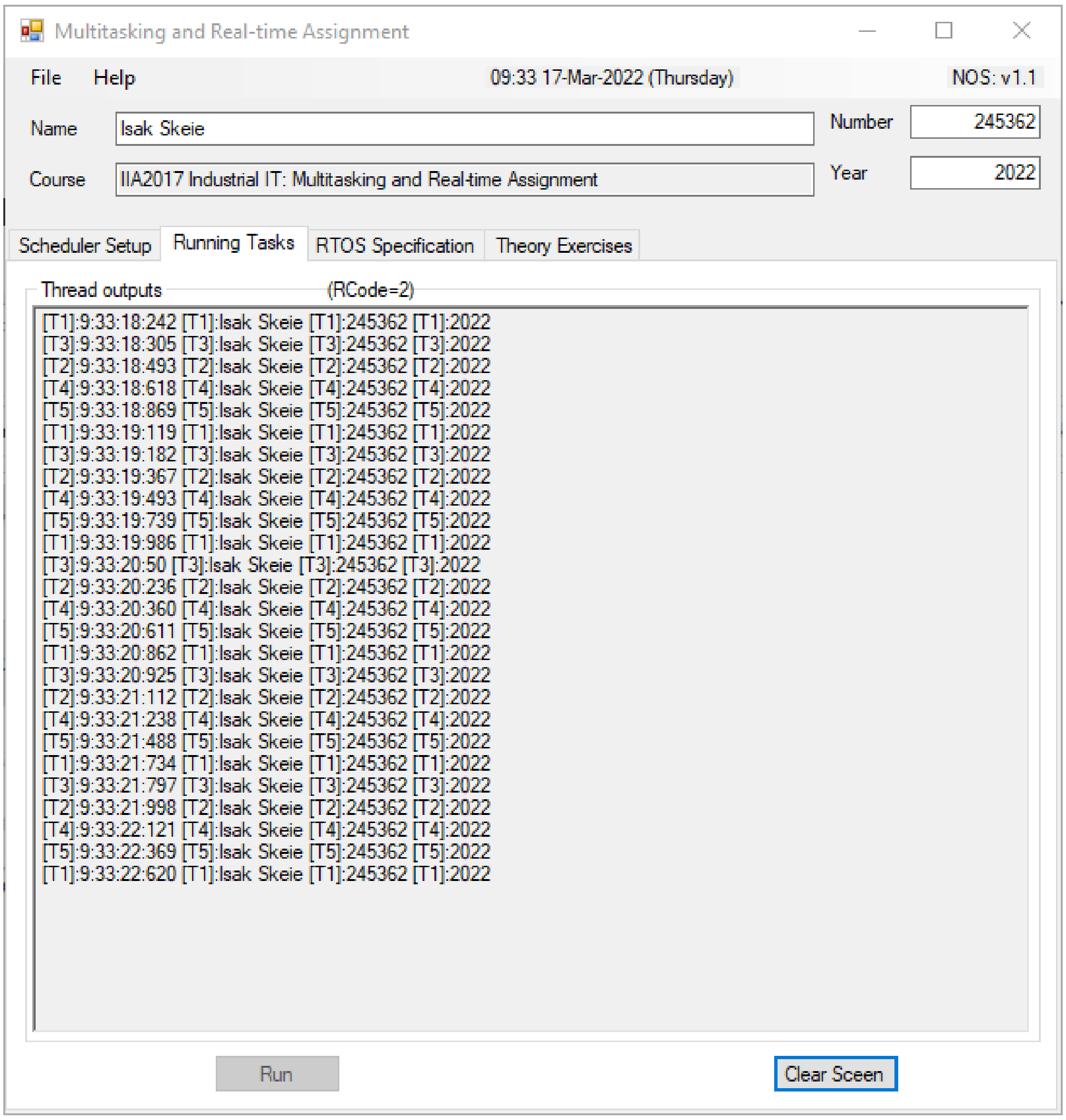
Thread outputs (RCode=1)

```
[T4]:10:33:1:254 [T4]:Isak Skeie [T4]:245362 [T4]:2022
[T5]:10:33:1:507 [T5]:Isak Skeie [T5]:245362 [T5]:2022
[T1]:10:33:1:768 [T1]:Isak Skeie [T1]:245362 [T1]:2022
[T2]:10:33:1:845 [T2]:Isak Skeie [T2]:245362 [T2]:2022
[T3]:10:33:1:971 [T3]:Isak Skeie [T3]:245362 [T3]:2022
[T4]:10:33:2:159 [T4]:Isak Skeie [T4]:245362 [T4]:2022
[T5]:10:33:2:406 [T5]:Isak Skeie [T5]:245362 [T5]:2022
[T1]:10:33:2:675 [T1]:Isak Skeie [T1]:245362 [T1]:2022
[T2]:10:33:2:750 [T2]:Isak Skeie [T2]:245362 [T2]:2022
[T3]:10:33:2:878 [T3]:Isak Skeie [T3]:245362 [T3]:2022
[T4]:10:33:3:64 [T4]:Isak Skeie [T4]:245362 [T4]:2022
[T5]:10:33:3:310 [T5]:Isak Skeie [T5]:245362 [T5]:2022
[T1]:10:33:3:576 [T1]:Isak Skeie [T1]:245362 [T1]:2022
[T2]:10:33:3:655 [T2]:Isak Skeie [T2]:245362 [T2]:2022
[T3]:10:33:3:783 [T3]:Isak Skeie [T3]:245362 [T3]:2022
[T4]:10:33:3:968 [T4]:Isak Skeie [T4]:245362 [T4]:2022
[T5]:10:33:4:214 [T5]:Isak Skeie [T5]:245362 [T5]:2022
[T1]:10:33:4:480 [T1]:Isak Skeie [T1]:245362 [T1]:2022
[T2]:10:33:4:557 [T2]:Isak Skeie [T2]:245362 [T2]:2022
[T3]:10:33:4:683 [T3]:Isak Skeie [T3]:245362 [T3]:2022
[T4]:10:33:4:870 [T4]:Isak Skeie [T4]:245362 [T4]:2022
[T5]:10:33:5:122 [T5]:Isak Skeie [T5]:245362 [T5]:2022
[T1]:10:33:5:386 [T1]:Isak Skeie [T1]:245362 [T1]:2022
[T2]:10:33:5:463 [T1]:*Finished*
[T2]:Isak Skeie [T2]:245362 [T2]:2022
[T3]:10:33:5:602 [T2]:*Finished*
[T3]:Isak Skeie [T3]:245362 [T3]:2022
[T4]:10:33:5:791 [T3]:*Finished*
[T4]:Isak Skeie [T4]:245362 [T4]:2022
[T5]:10:33:6:41 [T4]:*Finished*
[T5]:Isak Skeie [T5]:245362 [T5]:2022
[T5]:*Finished*
```

Figure 6: End of running code 1

2.1.3 Running Code 2

When running code 2, the process hangs, as seen in figure #. There could be several explanations as to why the process hangs. A higher priority task could be needing a semaphore that’s taken by a lower priority task. From the figure #, thread 1 is the last to be executed, another explanation is that this thread doesn’t release the Semaphore properly.

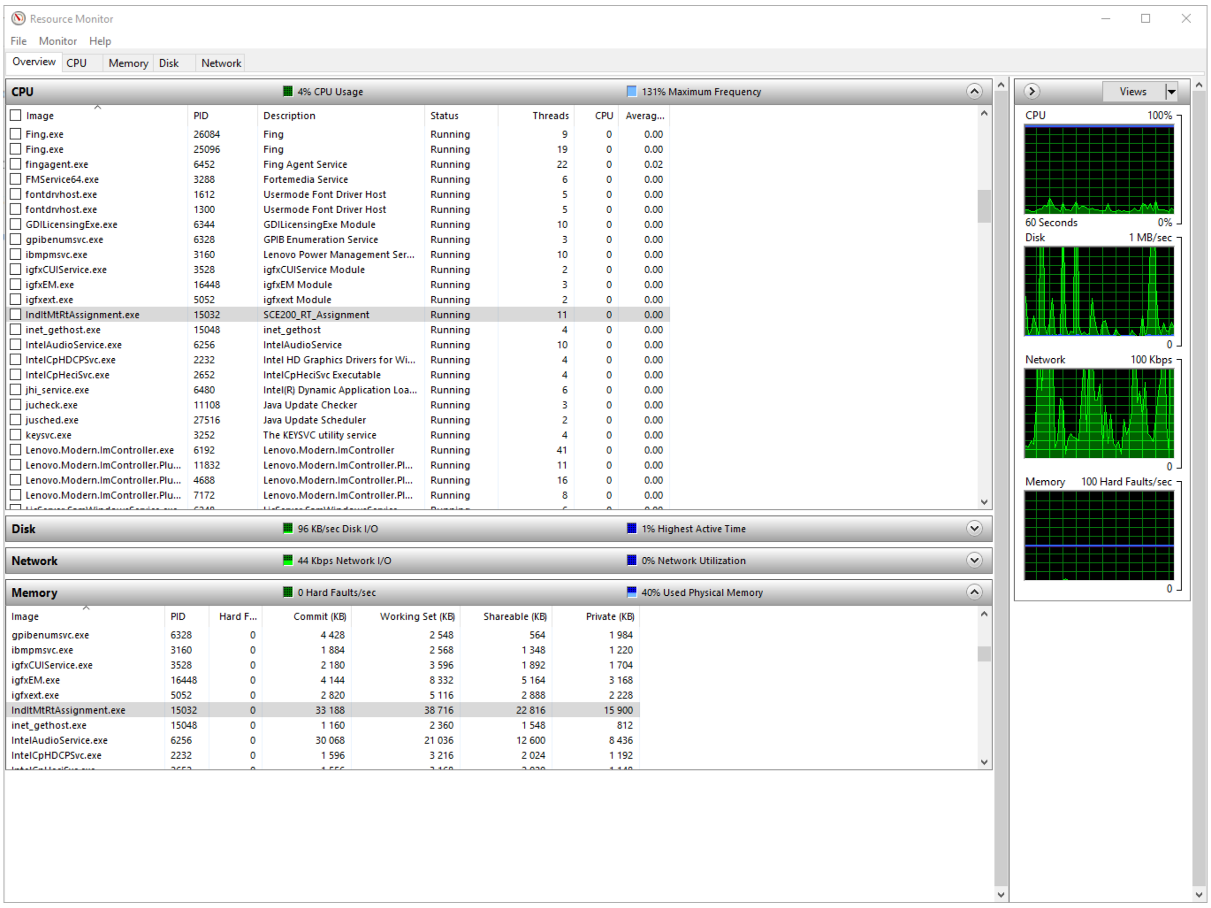


Name: Isak Skeie
Number: 245362
Course: IIA2017 Industrial IT: Multitasking and Real-time Assignment
Year: 2022

Thread outputs (RCode=2)

```
[T1]:9:33:18:242 [T1]:Isak Skeie [T1]:245362 [T1]:2022
[T3]:9:33:18:305 [T3]:Isak Skeie [T3]:245362 [T3]:2022
[T2]:9:33:18:493 [T2]:Isak Skeie [T2]:245362 [T2]:2022
[T4]:9:33:18:618 [T4]:Isak Skeie [T4]:245362 [T4]:2022
[T5]:9:33:18:869 [T5]:Isak Skeie [T5]:245362 [T5]:2022
[T1]:9:33:19:119 [T1]:Isak Skeie [T1]:245362 [T1]:2022
[T3]:9:33:19:182 [T3]:Isak Skeie [T3]:245362 [T3]:2022
[T2]:9:33:19:367 [T2]:Isak Skeie [T2]:245362 [T2]:2022
[T4]:9:33:19:493 [T4]:Isak Skeie [T4]:245362 [T4]:2022
[T5]:9:33:19:739 [T5]:Isak Skeie [T5]:245362 [T5]:2022
[T1]:9:33:19:986 [T1]:Isak Skeie [T1]:245362 [T1]:2022
[T3]:9:33:20:50 [T3]:Isak Skeie [T3]:245362 [T3]:2022
[T2]:9:33:20:236 [T2]:Isak Skeie [T2]:245362 [T2]:2022
[T4]:9:33:20:360 [T4]:Isak Skeie [T4]:245362 [T4]:2022
[T5]:9:33:20:611 [T5]:Isak Skeie [T5]:245362 [T5]:2022
[T1]:9:33:20:862 [T1]:Isak Skeie [T1]:245362 [T1]:2022
[T3]:9:33:20:925 [T3]:Isak Skeie [T3]:245362 [T3]:2022
[T2]:9:33:21:112 [T2]:Isak Skeie [T2]:245362 [T2]:2022
[T4]:9:33:21:238 [T4]:Isak Skeie [T4]:245362 [T4]:2022
[T5]:9:33:21:488 [T5]:Isak Skeie [T5]:245362 [T5]:2022
[T1]:9:33:21:734 [T1]:Isak Skeie [T1]:245362 [T1]:2022
[T3]:9:33:21:797 [T3]:Isak Skeie [T3]:245362 [T3]:2022
[T2]:9:33:21:998 [T2]:Isak Skeie [T2]:245362 [T2]:2022
[T4]:9:33:22:121 [T4]:Isak Skeie [T4]:245362 [T4]:2022
[T5]:9:33:22:369 [T5]:Isak Skeie [T5]:245362 [T5]:2022
[T1]:9:33:22:620 [T1]:Isak Skeie [T1]:245362 [T1]:2022
```

Figure 7: Attemptet run of Running code 2



CPU 🟩 4% CPU Usage 🟦 131% Maximum Frequency

Image	PID	Description	Status	Threads	CPU	Averag...
Fing.exe	26084	Fing	Running	9	0	0.00
Fing.exe	25096	Fing	Running	19	0	0.00
fingagent.exe	6452	Fing Agent Service	Running	22	0	0.02
FMService64.exe	3288	Fortemedia Service	Running	6	0	0.00
fontdrvhost.exe	1612	Usermode Font Driver Host	Running	5	0	0.00
fontdrvhost.exe	1300	Usermode Font Driver Host	Running	5	0	0.00
GDILicensingExe.exe	6344	GDILicensingExe Module	Running	10	0	0.00
gpibenumsvc.exe	6328	GPIB Enumeration Service	Running	3	0	0.00
ibmpmsvc.exe	3160	Lenovo Power Management Ser...	Running	10	0	0.00
igfxCUIService.exe	3528	igfxCUIService Module	Running	2	0	0.00
igfxEM.exe	16448	igfxEM Module	Running	3	0	0.00
igfxext.exe	5052	igfxext Module	Running	2	0	0.00
IndItMtRtAssignment.exe	15032	SCE200_RT_Assignment	Running	11	0	0.00
inet_gethost.exe	15048	inet_gethost	Running	4	0	0.00
IntelAudioService.exe	6256	IntelAudioService	Running	10	0	0.00
IntelCpHDCPSvc.exe	2232	Intel HD Graphics Drivers for Wi...	Running	4	0	0.00
IntelCpHeciSvc.exe	2652	IntelCpHeciSvc Executable	Running	4	0	0.00
jhi_service.exe	6480	Intel(R) Dynamic Application Loa...	Running	6	0	0.00
jucheck.exe	11108	Java Update Checker	Running	3	0	0.00
jusched.exe	27516	Java Update Scheduler	Running	2	0	0.00
keysvc.exe	3252	The KEYSVC utility service	Running	4	0	0.00
Lenovo.Modern.ImController.exe	6192	Lenovo.Modern.ImController	Running	41	0	0.00
Lenovo.Modern.ImController.Plu...	11832	Lenovo.Modern.ImController.Pl...	Running	11	0	0.00
Lenovo.Modern.ImController.Plu...	4688	Lenovo.Modern.ImController.Pl...	Running	16	0	0.00
Lenovo.Modern.ImController.Plu...	7172	Lenovo.Modern.ImController.Pl...	Running	8	0	0.00

Disk 🟩 96 KB/sec Disk I/O 🟦 1% Highest Active Time

Network 🟩 44 Kbps Network I/O 🟦 0% Network Utilization

Memory 🟩 0 Hard Faults/sec 🟦 40% Used Physical Memory

Image	PID	Hard F...	Commit (KB)	Working Set (KB)	Shareable (KB)	Private (KB)
gpibenumsvc.exe	6328	0	4 428	2 548	564	1 984
ibmpmsvc.exe	3160	0	1 884	2 568	1 348	1 220
igfxCUIService.exe	3528	0	2 180	3 596	1 892	1 704
igfxEM.exe	16448	0	4 144	8 332	5 164	3 168
igfxext.exe	5052	0	2 820	5 116	2 888	2 228
IndItMtRtAssignment.exe	15032	0	33 188	38 716	22 816	15 900
inet_gethost.exe	15048	0	1 160	2 360	1 548	812
IntelAudioService.exe	6256	0	30 068	21 036	12 600	8 436
IntelCpHDCPSvc.exe	2232	0	1 596	3 216	2 024	1 192

Figure 8: Resource monitoring while Running code 2 is deadlocked

2.2 Conclusion on analysis results

When looking at the running time for each of the processes. Its clear that its determined by the delay given for each of the threads. With the run time for Running code 0 and Running code 1 being the same. The synchronization used in Running code 1 makes better use of the resources and the execution better for the user, with all of the threads starting and stopping concurrently. When it comes to Running code 2, the synchronization of the threads fault, making the code go into a Deadlock.

3 Development of a multitasking system

The code from the lectures # is used source code. When running this code, it acts the same way as Running Code 0. With the delays in the threads deciding when to print information. By using Semaphores, the goal is to make the code act more like Running Code 1. This means that the threads print out their information concurrently. Disregarding the delay that's been added. This is done by added a Semaphore that can be occupied by only one Thread at a time. The Semaphore is closed when the thread is initialed, and closed when the initialization is over. It will be closed again when each of the threads enters their loop for writing to the console. And released at the end of the loop. By using Semaphores on initialization, as well as Thread.Sleep(), the threads are queued in the right order for them to execute ascending. The finished code can be viewed in appendix A.

```
Initialize static Sephamore;
ThreadClass
{
Initialize integer for loop count;
Initialize integer for delay;
Initialize thread class;

ThreadClass constructor(name, delay)
      {
      Initialize integer LoopCount = 0;
      Sets delay equal to input delay;
      Sephamore.WaitOne():
      Initializes thread;
      Sets name of thread;
      }
Run Method()
      {
      Semaphore release;
      Write to console starting thread;
      Do While
```

```
            {
            Semaphore WaitOne;
            Loop count ++;
            Thread sleep with delay;
            Semaphore release;
            Write Thread name and loop;
            }
      Write ending of thread;
      }
}


Class program
{
Static void main()
      {
      Write starting program;
      Create array of thread classes;
      Create string array of thread names;
      Create int array of thread loop delay;
      Loop that instantiate array of thread classes;
      Loop through a count with Thread.Sleep(0);
      Write end of program;
      }
}
```

4 Time Requirements for a real-time system

Based on the requirements given in the Assignment Description, a maximum response time is needed for the system. The requirement given is a maximin latency of 250ms from energization of level input, until the pump acts. When the pump receives a signal to change state, it takes 200ms until the process is finished. This leaves the application a window of 50ms to receive a signal from one of the switches, until a control signal is sent to the Pump. Within 50ms the application needs to receive the signal interrupt, run Interrupt Service Routine, run the scheduler, and then run the task for the pump.

$$Time = Interupt\ Latency + Context\ Switch + Interupt\ Instructions + Context\ Switch + Task\ instructions$$

Based on the given equation, all of the RTOS fulfill the time requirement, although with different margins. For this use case, RTOS 1 would be a good choice, it doesn't have the highest margins in milliseconds, nor the highest number of maximum tasks. But it has a low Context switch time, and low instruction time. This is assumed favorable for the speed of the system, over the scalability which is assumed, will stay at a minimum. The timing diagram for RTOS 1 is shown in figure #

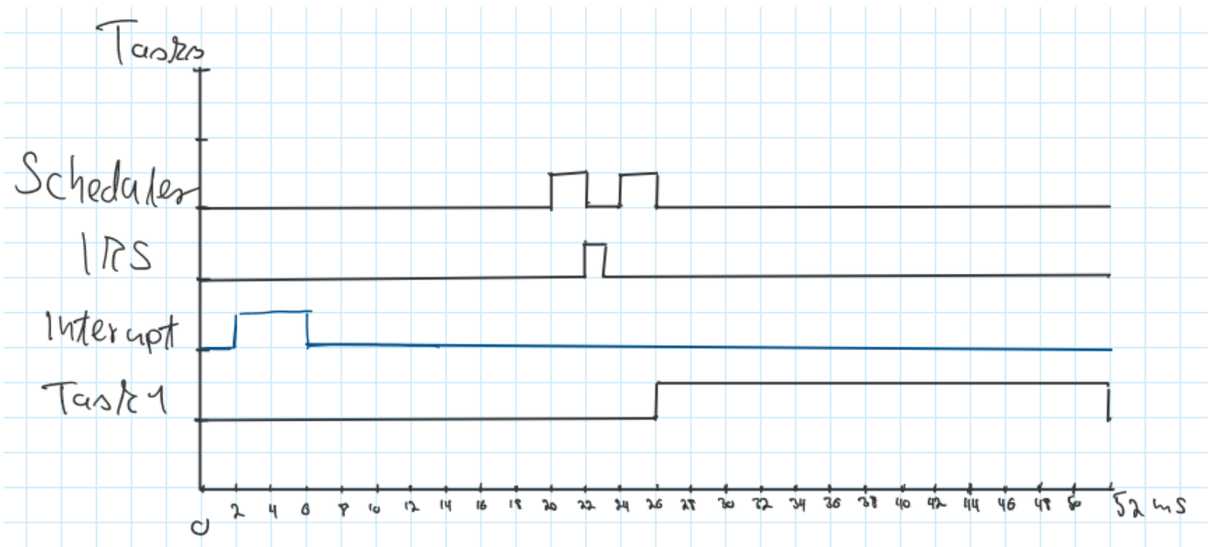


Figure 9: Timing Diagram for the system

5 Conclusion

The purpose of this assignment have been to explore the different aspects of Multitasking and Real-Time systems. This is done through several tasks. The first one being an evaluation of several similar multithreaded processes . By comparing the differences and how they act with respect to time, its possible to make an analysis. In doing this you need fundamental knowledge of the process of threading. With the development of a threaded system, you obtain knowledge in using Semaphores and how threads relate to each other with the use of semaphores. Lastly , the time required forces you to regard the different elements that make the threading process when calculating the time spent for a process. This includes Threaded tasks, interrupts, and scheduling.

Going through these tasks a general understanding is obtained. Useful for development of embedded systems, and deeper knowledge in how more complex software systems functions. Vital capabilities for an Engineer working with industrial IT and Automation.

6 References

[1] Geeks for Geeks, «Geeks For Geeks,» 1 04 2021. [Internett]. Available: https://www.geeksforgeeks.org/mutex-vs-semaphore/. [Funnet 16 03 2022].

[2] M. Barr, «BarrGroup,» 04 05 2016. [Internett]. Available: https://barrgroup.com/embedded-systems/how-to/rtos-mutex-semaphore. [Funnet 16 03 2022].

[3] N.-O. Skeie, «Course IIA2017 - Industrial Information Technology,» i *Lecture notes with sections for systems engineering, process.....*, 2022, p. 599.

Appendices

Appendix A: Threading program

```
// Isak Skeie, 245362, 2022-Spring
using System;
using System.Threading;


namespace Threads
{

    class ThreadClass
    {

        int loop_cnt;
        int loop_delay;
        Thread cThread;


        public ThreadClass(string name, int delay)
        {
            loop_cnt = 0;
            loop_delay = delay;
            Sema.sem.WaitOne();


            cThread = new Thread(new ThreadStart(this.run));
            cThread.Name = name;

            cThread.Start();

        }

        void run()
        {
            Sema.sem.Release();

            Console.Write(" Starting " + cThread.Name);
            do
            {
                Sema.sem.WaitOne();


                loop_cnt++;
                Thread.Sleep(loop_delay);

                Sema.sem.Release();
                Console.WriteLine(" " + cThread.Name + ": Loop" + loop_cnt);
            } while (loop_cnt < 5);

            Console.WriteLine(" Ending " + cThread.Name);


        }
    }
    class Program
    {


        static void Main(string[] args)
        {
            Console.WriteLine(" Start of main Program ");
            ThreadClass[] ct = new ThreadClass[5];

            string[] names = new string[]
            {
                "Thread1",
                "Thread2",
                "Thread3",
                "Thread4",
                "Thread5"
            };

            int[] Delay = new int[]
            {
                95,
                189,
                283,
                377,
                471
            };




            for (int i = 0; i < 5; i++)
            {
                ct[i] = new ThreadClass(names[i], Delay[i]);
            }


            for (int cnt = 0; cnt < 30; cnt++)
            {
                Console.Write(".");
                Thread.Sleep(0);
            }

            Console.WriteLine(" End of Main Program");
        }
    }

    public static class Sema
    {
        public static Semaphore sem = new Semaphore(1, 1);
    }
}
```