TDDD07 Lab report

By Emil Nilsson & Tommy Lindman

Table of contents

# Introduction

The main goal for this lab series was to give us the opportunity to get some hand-on experience in building a distributed real-time system. This includes analyze and evaluate timing properties and how to structure a real-time system with regard to deadlines and communication constraints.

In lab 1 of this series the goal was the construct a schedule for a set of tasks by measuring and analyzing the WCET and others factors under different conditions.

In lab 2 we were to ensure that the wireless communication between the robots was working given a set of conditions: prioritize the different kinds of messages being sent to ensure that critical messages are delivered, a robot are only allowed to send data under a predefined time-slot, there is a resource constraint on the communication bandwidth.

# Laboration 1

## Method

To generate measurement points for execution times we defined conditions in which to test the execution time of the given tasks. The varying conditions included robot location, robot situation and execution order of tasks. Robot situation is dependent on location, i.e. in some location the robot will find a victim, in another the robots hits a wall and so on.

Round 1:

When we begun measuring the execution times we started by running the robot for short periods of time under the different conditions specified above whilst gathering data. This step of the measurement process gave us huge amounts of data with varying execution times. Since we were only interested in the worst case execution times we discarded all data except the worst case for each task.

Round 2:

The gathered data were used as a filter throughout the rest of the measurements. In the second round we only gathered data when the execution time was equal to or exceeded the worst case execution times in the first round. Giving us less but more relevant data.

Schedule:

We designed the schedule by taking into account the WCET and the minimum and maximum interarrival times for each task. See table 1.

## Results

By analyzing and evaluating our measurements we decided the WCETs as shown in Table 1. The measured worst cased execution time for each task was rounded up to the nearest integer and used as each tasks WCET. As a result of the solid and thorough test and measurement phase we were confident that we had measurements very close to each tasks real WCET.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **TASK** | **WCET** | **MINIMUM** | **MAXIMUM** | **ACTUAL** |
| Avoid | 27 | 0 | 100-150 | 112 |
| Communicate | 5 | 0 | 1300 | 560 |
| Control | 19 | 300 | 1300 | 560 |
| Mission | 1 | 0 | 1300 | 560 |
| Navigate | 1 | 0 | 1300 | 560 |
| Refine | 56 | 0 | 1300 | 112 |
| Report | 1 | 0 | 1300 | 560 |

Table 1- Task periodicity and WCET (in ms)

From the information in Table 1 we constructed our schedule as seen in Figure 1.

Our minor cycle was 112ms and our major cycle 560ms.

* The Avoid task must be run at least every 100-150ms to avoid overheating the motors.

Avoid task in scheduled to run every 112ms.

* The time from when a victim is read by the RFID reader to when a message is sent must be less than 1300ms.

The time between a victim is read and when a message is sent can never be more than one major cycle (560ms).

* If a stop command is received from the mission control, then the robot must stop within 1300ms.

Robot stops within one major cycle (560ms).

# Laboration 2

## Method

To ensure working communication between the robots we had to make sure to only send data in a predefined time-slot using TDMA. As bandwidth was restricted we had to ensure that all the critical messages were prioritized.

To be able to communicate within our designated time-slot we had to make some modifications to our schedule from lab 1 as well as to our task periodicity table. The revised table (Table 2) contains an updated version of the information available. The revised schedule (Figure 2) has a minor cycle of 125ms and a major cycle of 1000ms.

We also had to make sure that the communication task was only run within the appropriate time-slot.

## Result

In order to dynamically synchronize with the time-slot we had to create a an algorithm which calculated time until the next communication slot and slept until that time-slot occurred before running the scheduler, hence starting execution. This algorithm is called timer\_wait\_for\_timeslot and is implemented in scheduler.c (APPENDIX LOCATION??). With this strategy all we needed to do to ensure that the communication task was run in the predefined time-slot was to place it in the beginning of the first minor cycle.

In lab 1 all messages in a doubly linked list were sent. The priority requirement for lab 2 was solved by checking what type of message that was next in the queue. If a high priority message was found it was send directly whilst low priority messages were placed in another queue depending on message type.

In order not to exceed the time-slot in which we were to send we kept track on the amount of bytes sent. Since we had knowledge of the time-slot time in ms and the transfer rate on the channel we could calculate the maximum total packet(s) size we were able to send.

## Lab 2

# Discussion

As a result of the solid and thorough test and measurement phase we were confident w

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **TASK** | **WCET** | **MINIMUM** | **MAXIMUM** | **ACTUAL** |
| Avoid | 27 | 0 | 100-150 | 125 |
| Communicate | 5 | 0 | 1300 | 1000 |
| Control | 19 | 300 | 1300 | 500 |
| Mission | 1 | 0 | 1300 | 1000 |
| Navigate | 1 | 0 | 1300 | 500 |
| Refine | 56 | 0 | 1300 | 125 |
| Report | 1 | 0 | 1300 | 125 |

Table 2

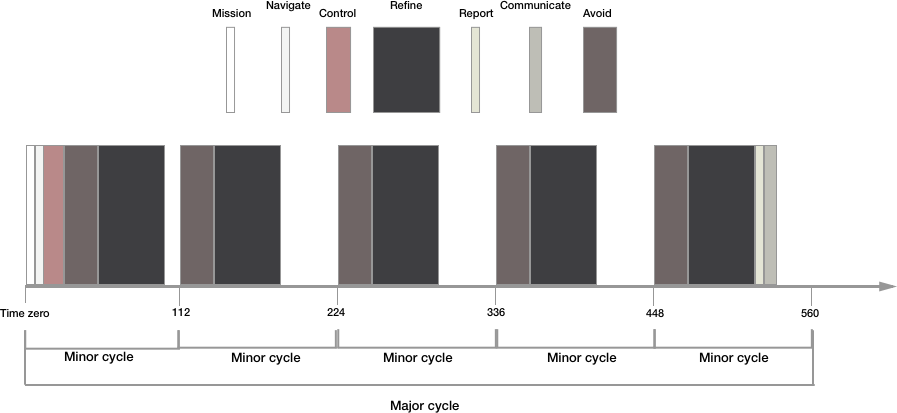


Figure 1

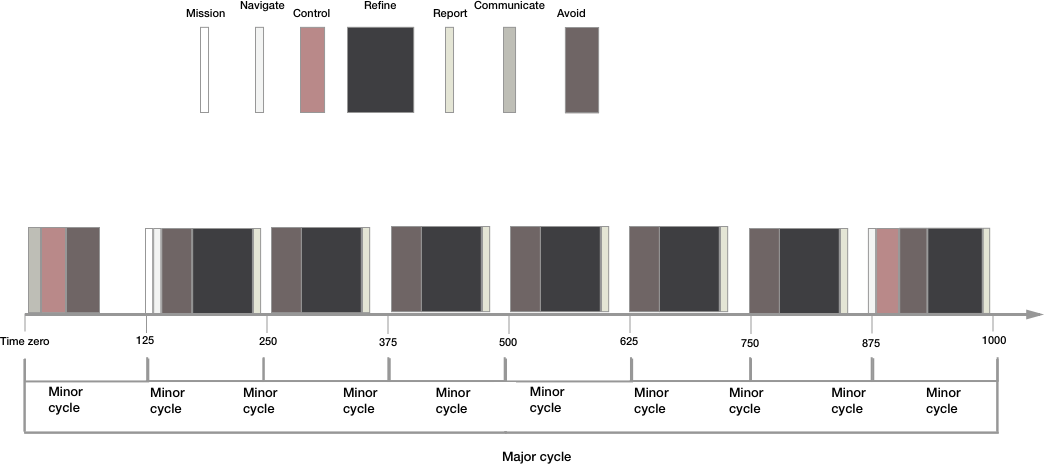


Figure 2