# Embedded Control Systems

## Exam 2016-08-18

8:30 – 12.30: M Building

Course code: SSY190

Teachers: Knut Åkesson

The teacher will visit examination halls twice to answer questions. This will be done approximately one hour after the examination started and one hour before it ends.

The exam comprises 30 credits. For the grades 3, 4, and 5, is respectively required 15, 20 and 25 credits.

Solutions and answers should be complete, written in English and be unambiguously and well motivated. In the case of ambiguously formulated exam questions, the suggested solution with possible assumptions must be motivated. The examiner retains the right to accept or decline the rationality of assumptions and motivations.

*Exam results* will be reported in Ladok. *The results* are open for review 2016-09-08 between 12:30-13:30 at the department.

*No aids are allowed on the written exam except:*

- Standard pocket calculator (no hand computer). Erased memory.
- The report: "Computer Control: An Overview" Björn Wittenmark, Karl-Johan Åström, and Karl-Erik Årzén. International Federation of Automatic Control (IFAC) Professional Briefs. No comments are allowed in the article.
- The report: "C for Java Programmers", J. Maassen. No comments are allowed in the article.
- Dictionary from/to your native language to/from English

**1**

a) Define what characterize a preemptive scheduler.
(2p)

b) Explain the difference between safety and liveness specifications.
(1p)

c) What is priority inversion? Explain by giving a simple example.
(1p)

d) Explain shortly how the priority inheritance protocol works.
(1p)

**2**

In a real-time control system there are three threads, one control thread, one thread for reference generation, and a thread for user interaction. The control thread must execute at a sampling rate of 4 milliseconds, the reference generator should update the references every 12 milliseconds, and the user interaction thread should poll for user interaction and update the plotters 50 times per second. The total execution times are $x$ milliseconds for the control thread, 4 milliseconds for the reference generator, and 9 milliseconds for the user interaction thread. You may assume that the real-time kernel is ideal.

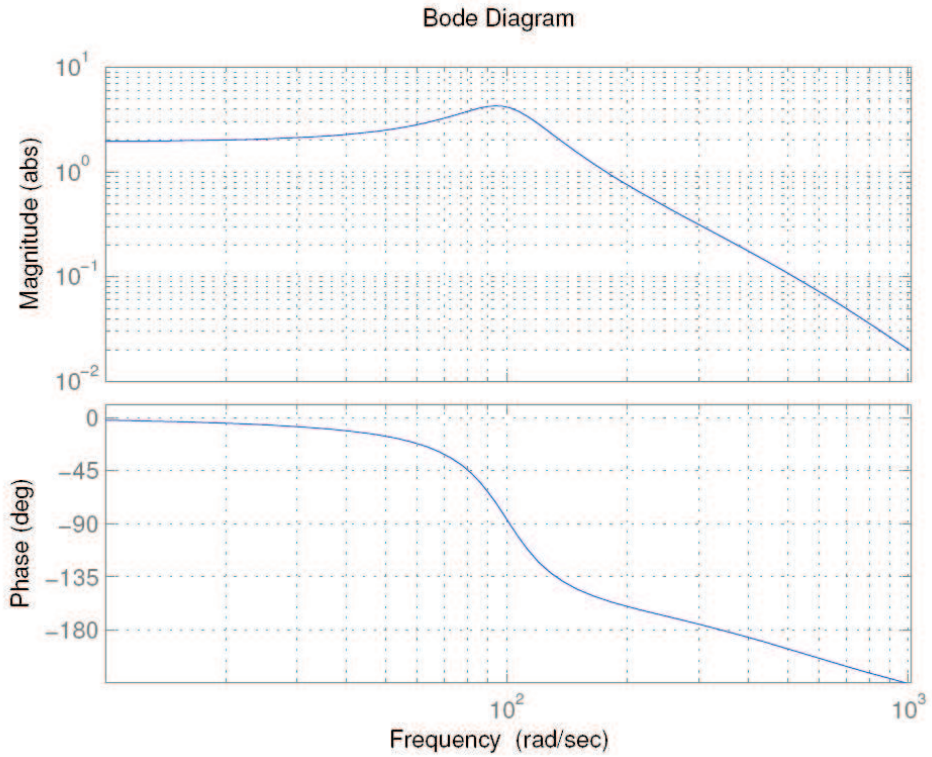
a) Assume that the deadline $D_i$ for each thread is equal to the period $T_i$. Assume that $x = 0.5$, that is, the controller execution time is 0.5 milliseconds and that all blocking due to interprocess communication can be ignored. Will the task set be schedulable using rate monotonic priority assignments?

(3p)

b) What is the maximum controller execution time $x$ allowed if the task set is to be schedulable using Earliest Deadline First scheduling?

(2p)

c) The open-loop Bode diagram of the system is shown below.

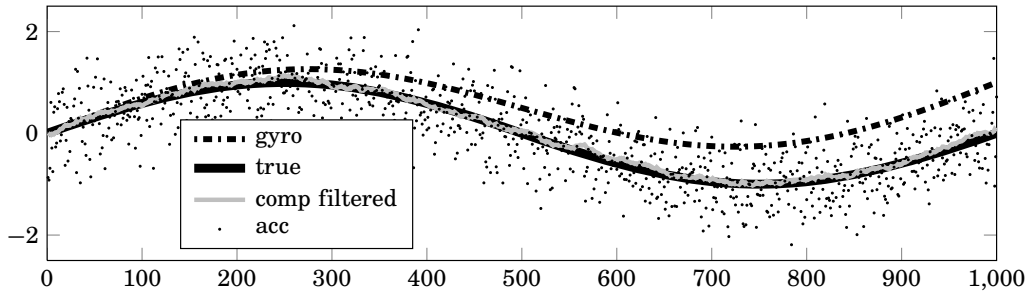



What is the maximum controller execution time $x$ allowed if the closed-loop system should be stable?

(2p)

**3**

Inertial Measurement Units (IMU) often contain both gyroscopes and accelerometers for positioning purposes. For the purpose of pose estimation, the gyroscope can be assumed to measure angular velocities and the accelerometer to measure an angle. Gyroscopes exhibit desirable high frequency properties, but suffer from low frequency drift. Accelerometers on the other hand, often suffer from high frequency noise, but exhibit no drift. A common method to fuse measurements from the two sensors is a complementary filter. The idea behind the filter is to obtain an estimate of the angle by combining a low-pass filtered value of the accelerometer signal with a high-pass filtered value of the integrated gyroscope signal. The result typically obtained can be seen in the figure below.




The simple complementary filter is characterized by two transfer functions, $G_g(s)$ and $G_a(s)$, and the relations

$$1 = G_g(s) + G_a(s) \tag{1}$$

$$Y_f(s) = G_g(s)Y_g(s) + G_a(s)Y_a(s) \tag{2}$$

It has been determined that a suitable low-pass filter for the accelerometer measurements is given by

$$G_a(s) = \frac{1}{s+1}.$$

a) Design the complementary high-pass filter for the gyroscope measurements, $G_g(s)$, such that eq. (1) is satisfied. Assume that the signal from the gyroscope is pre-integrated, such that the gyroscope returns an angle measurement.

(1p)

b) Sample the two filters using zero-order hold. The filters are to be implemented with sample time $h$.

(1p)

c) Implement the discrete time filters obtained in b) in the pseudo-code skeleon below. The output `yf` is the filtered signal, the inputs `ya` and `yg` are the measurements from the accelerometer and the (pre-integrated) gyroscope respectively. A variable declared `static` will keep its value between two consecutive calls to the function.

(3p)

```
double compFilt (double ya, double yg, double h) {
    /* Declare and initialize variables */
    static double yf = 0.0; // Filtered signal
    ... Your code here ...

   /* Perform calculation */
    ... Your code here ...

   return yf;
}
```

**4**

Suppose we have N threads with identities $0, 1, \ldots, N-1$. The threads share a critical region of code that only one of the threads is allowed to execute at a time, in addition they are required to repeatedly cycle through this section in the natural order $(0, 1, \ldots, N-1)$.

a) Let `turn` be a shared integer variable initialized to zero and let `current_pid` be the identity of the currently executing process. Each thread executes the code below. Assume that reading or writing to a single variable is an atomic operation. Can a race-condition occur? Motivate your answer!

```
while( TRUE )
{
    while( turn != current_pid ) /* wait */ ;
    critical_region();
    turn = (turn + 1) % N;
    noncritical_region();
}
```

Note: x % N returns the remainder of division of x by N.

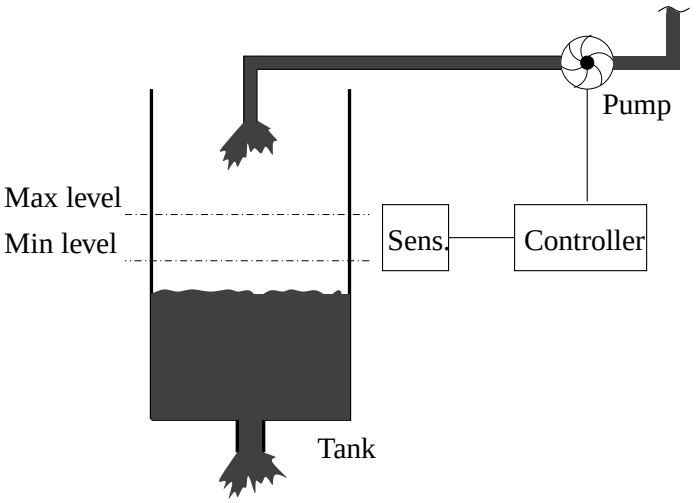
(2p)

b) Develop a solution using semaphores for the problem described above - you are not allowed to used a turn variable or a similar solution all coordination should be based on sempahores. Clearly specify the initial value of each semapahore. Write pseduo-code for the solution. Notation: if `x` is a sempahore then `wait(x)` and `signal(x)` are the semaphore operations.

(2p)

**5**

Consider the water tank control system shown below.




The control objective is to maintain the water level $l$ within the maximum and minimum levels indicated in the figure. The controller can either turn on the pump or turn it off. When the pump is on, the dynamics of the tank is

$$On : \frac{dl(t)}{dt} = -l(t) + 25$$

and when it is off

$$Off : \frac{dl(t)}{dt} = -l(t)$$

The controller will turn the pump on if $l < 10$ and will turn it off if $l > 15$.

a) Sketch a solution of the hybrid system with the continuous part of the initial state equal to $l(0) = 12$ and with three discrete transitions.
(1p)

b) Is the hybrid automaton Zeno, i.e., does it have Zeno solutions?
(2p)

**6**

In the following, give an LTL formula that formalizes the given English wording. If the English is subject to any ambiguity, as it frequently is, describe how you are disambiguating it, and why.

a) $p$ is true.

(1p)

b) $p$ becomes true before $r$.

(1p)

c) $p$ will happen at most once.

(1p)

d) $p$ will happen at most twice.

(1p)

e) The light always blinks. Use the following proposition: $p$ = the light is on.

(1p)

f) The lights of a traffic signal always light in the following sequence: green, yellow, red, and back to green, etc., with exactly one light on at any time. Use the following propositions: $g$ = the green light is on, $y$ = the yellow light is on, and $r$ = the red light is on.

(1p)

**Good Luck!**

# Formula sheet

Liu-Layland schedulability test

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

Response-time analysis

$$R_i = C_i + \sum_{\forall j \in hp(i)} \left\lceil \frac{R_i}{T_j} \right\rceil C_j$$

1/ See book

2/

| | T | C |
|---|---|---|
| control | 4 | 1 |
| ref | 12 | 4 |
| ui | 20 | 9 |

50 times/second $\Rightarrow T = 20$ ms

a/ Assume $D = T$ for all threads and $x = 0,5$

Is the task schedulable with RM

$$\sum \frac{C_i}{T_i} \leq n\left(2^{1/n} - 1\right) \qquad \text{(Sufficient condition for schedulability)}$$

$$\frac{x}{4} + \underbrace{\frac{4}{12} + \frac{9}{20}}_{0,78} \leq \underbrace{3\left(2^{1/3} - 1\right)}_{\approx 0,779} \Rightarrow$$

Liu-Leyland not satisfied.
Exact response time analysis needed.
$\Rightarrow$ Schedulable.

b/ $\frac{x}{4} + \frac{4}{12} + \frac{9}{20} \leq 1 \Rightarrow x \leq 0,87$

c/ Deadtime in controller $e^{-sL}$, $L = x$

$\omega_c \approx 190$ rad/s

arg deadtime $= -\omega x \cdot \frac{180}{\pi}$

$\varphi_m$ (from figure) is $\sim 20°$

Closed-loop system is stable if arg Deadtime $(j\omega_c) \leq \varphi_m$

$\Rightarrow 190 \cdot x \cdot \frac{180}{\pi} \leq 20 \Rightarrow x \leq \frac{20 \cdot \pi}{180 \cdot 190} \approx 0,0018$ seconds $= \underline{1,8 \text{ ms}}$

3/

a/ $G_a = \frac{1}{s+1}$ $\qquad G_g(s) = 1 - G_a(s) = 1 - \frac{1}{s+1} = \frac{s}{s+1}$

b/ From Table 3 in Computer-Control: An overview we have

$$H_a(z) = \frac{1 - e^{-h}}{z - e^{-h}}$$

$$H_g(z) = 1 - H_a(z) = \frac{z-1}{z - e^{-h}}$$

c/ To implement the filters they need to be transformed into a difference equation

$$y_f(k) = H_a(z)y_a(k) + H_g(z)y_g(k) = \frac{1 - e^{-h}}{z - e^{-h}} y_a(k) + \frac{z-1}{z - e^{-h}} y_g(k)$$

$\Longleftrightarrow$

$$(z - e^{-h}) y_f(k) = (1 - e^{-h}) y_a(k) + (z-1) y_g(k)$$

$$y_f(k+1) = e^{-h} y_f(k) + (1 - e^{-h}) y_a(k) + y_g(k+1) - y_g(k)$$

Shift the difference equation one step to get $y_f(k)$

$$\Rightarrow y_f(k) = e^{-h} y_f(k-1) + (1 - e^{-h}) y_a(k-1) + y_g(k) - y_g(k-1)$$

C-code:

```
double compFilt(double ya, double yg, double h)
{
    static double yf = 0.0; // Filter state
    static double ys = 0.0; // Gyroscope state
    static double yas = 0.0; // Accelerometer state
    double e = exp(-h); // Filter constant
    double em = 1 - e; // Filter constant
    yf = e * yf + em * yas + yg - ygs;
    ygs = yg;
    yas = ya;
    return yf;
}
```

4/

a/ No race-condition can occur.

current_pid is constant for each thread, turn is a shared variable.

A context switch after turn == current_pid will not allow any other thread to have the condition (turn == current_pid) to be satisfied. Turn is only updated after the critical region is left.

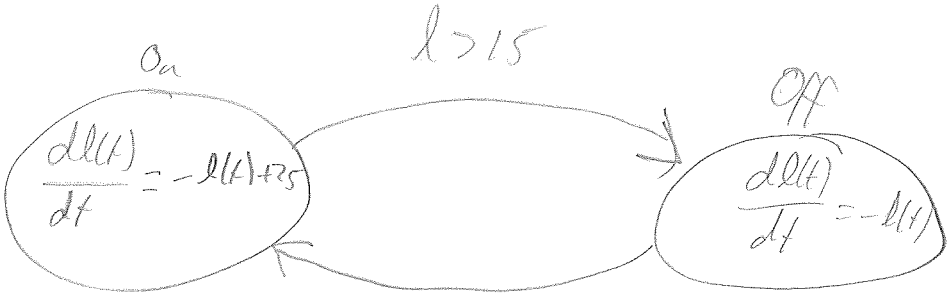
b/ Semaphore mutex

Initialize mutex to 1.

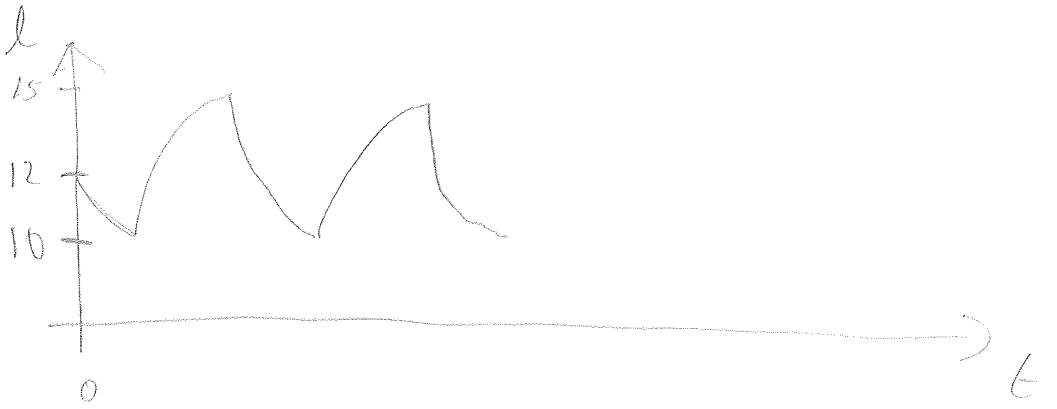
Every thread:

```
while (true)
    wait(mutex)
    critical_region()
    signal(mutex)
    noncritical_region()
```

5a/




a/ $l(0) = 12$ Start in off state




b/ It is not zeno, because infinite frequent switchers are not possible.

6/

a) If p should be true in all states, then $G_p$

b) $\neg r \, U (p \vee \neg r)$ This allows p to never happen. Not clear from spec if this should be allowed or not

c) $\neg(F(p \wedge X F p))$

d) $\neg(F(p \wedge X(p \wedge X F p)))$

e) $G((p \wedge F \neg p) V (\neg p \wedge F p))$

f) Exactly one light on at any time:

$$P_1 = G((\underbrace{g \vee y \vee r}_{\text{at least one}}) \wedge \overbrace{(\neg(g \wedge y) \wedge \neg(g \wedge r) \wedge \neg(y \wedge r))}^{\text{at most one}})$$

Prevent r from follow y, g from follow y, and y from r

$$P_2 = G(\neg(g \wedge X r) \wedge \neg(y \wedge X g) \wedge \neg(r \wedge X y))$$

$$P_3 = GF g \wedge GF y \wedge GF r$$

Total spec: $P_1 \wedge P_2 \wedge P_3$