Exam solutions FYS3240/4240 2014

Problem 1

a) Explain how the accuracy (constant frequency output) of quartz crystal oscillators is improved.
The accuracy is improved using **temperature compensation** (temperature compensated crystal oscillator - TCXO), or **temperature control** using an oven controlled crystal oscillator (OCXO), where the oscillator is enclosed in a temperature controlled oven.

b) Explain the **IEEE 1588** and the **NTP** protocols.

IEEE 1588:

Gives **sub-microsecond synchronization** in distributed systems, and provides a standard protocol for synchronizing clocks connected via a multicast capable network, such as **Ethernet**. All participating clocks in the network are <u>synchronized to the highest quality clock in the network</u>. The highest ranking clock is called the ***grandmaster clock***, *and* synchronizes all other ***slave clocks***. IEEE 1588 uses a protocol known as the **precision time protocol (PTP)**, and the level of precision achievable using PTP depends heavily on the jitter (the variation in latency) present in the underlying network topology.

NTP:

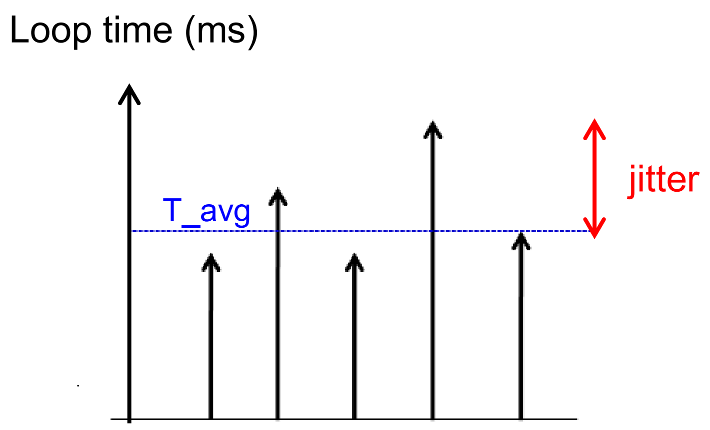
A protocol designed to synchronize the clocks of computers over a network, and can provide **accuracies of better than 10 ms** over Ethernet. The accuracy depends on the network, and point-to-point connections provide the highest precision. The **User Datagram Protocol (UDP)** is used. (A time request is sent to a time server or to a GPS with an NTP server).

c) Explain the limitations of software timing based on the computer clock when you try to measure the time between events in your software code.

The **Real Time Clock (RTC**) is an integrated circuit on the motherboard, with a typical resolution of **1 millisecond (1 kHz).** The software clock, called the computer clock, is maintained by the operating system based on the RTC interrupts. At computer start-up (power turned on) the computer clock (system time) is set to a value based on the RTC and then regularly updated based on interrupts from the RTC chip. Therefore, **the <u>time resolution</u> of the computer clock is limited to 1 ms**, and **the <u>accuracy</u> of the computer clock is hardware and operating system dependent.**

If you use a software timer functions (which use the computer clock) to control a loop, then you **must expect differences in the time interval (period) between each iteration of the loop (this is called <u>jitter</u>).**

(In addition, the computer clock drifts away from the correct time. At the time of synchronization with a time server the clock is reset to the "correct time", but with a small offset).



d) How can you make more accurate relative time measurements in your software then what you can obtain using the computer clock?

The question is about **relative time** **measurements**; e.g. number of milliseconds elapsed to calculate a result. That means that synchronization to UTC time or GPS time is not important. What is important is to have a clock with high time resolution (millisecond resolution is often not sufficient), and the clock must have high accuracy (low drift). Since we are only interested in accurate relative time, not the absolute time, **we should use a high precision counter/timer**

Available computer hardware timers are:

- **High Precision Event Timer (HPET)**
  - 64-bit up-counter with a frequency higher than 10 MHz.
- **Time Stamp Counter (TSC)**
  - 64-bit register in the CPU (cores) that increment each processor clock cycle.
  - However, can be unreliable on a modern multicore computer due to:
    - multicore computers can have different values in their time-keeping registers.
    - variability of the CPU frequency due to power management technologies or performance technologies such as *Intel Turbo Boost Technology*.

To get **µs resolution:** *Use the functions* ***QueryPerformanceCounter*** and ***QueryPerformanceFrequency***

$$\Delta T = \frac{\boldsymbol{QueryPerformanceCounter(n)} - \boldsymbol{QueryPerformanceCounter(n-1)}}{\boldsymbol{QueryPerformanceFrequency}}$$

Synchronizing the computer clock to GPS will not be sufficient, since we still get the same resolution problem (1 ms) from the computer clock (and the computer clock will also drift after synchronization). So, an alternative to use HPET and TSC is to add a timing/counter card to your computer. These counter/timing cards are available with high precision oscillators (OCXO). So instead of reading the computer clock you read a counter value from this card, from your software. Since most DAQ-cards also have counters an alternative is to read a counter value from a DAQ-card, but a DAQ card has a less accurate oscillator compared to a dedicated timing card. (With a known oscillator frequency you can calculate the number of microseconds from the number of counts).

e) You are provided two separate text files with measurements recorded from two different instruments. The first column in both files should contain a GPS timestamp for each data sample. However, comparing the measurement data in the two files you discover that it must be an offset in time between the timestamps in the two files. What could be the reason for this time offset?

Coordinated Universal Time (**UTC**) is the world time standard based on atomic clocks, but with leap seconds added at irregular intervals to compensate for the slowing of the Earth's rotation. Remember that **GPS time does not include leaps seconds**, and as of July 2012 **GPS time is 16 seconds ahead of UTC** because of the leap seconds added to UTC. So, if you see at 16 seconds difference in the time columns (when comparing corresponding measurement data) this is due to that one instrument has converted to UTC time, while the other has not.

Another error source (if not the 16 sec difference) could be in the time stamping process; causing that the time stamp is not associated to the correct data sample (e.g. due to buffered data acquisition). This can happen in a computer-based DAQ system unless you do time stamping directly in hardware in the instrument/sensor. (For instance many high-speed cameras timestamp each image in hardware, inside the camera, before the image containing the timestamp as part of the image data is transmitted / recorded).

There are of course several other possible explanations. The text describes **a time offset**, and with offset we usually think of a constant (time invariant) deviation from the true/correct value. Some GPS errors could cause an offset time drift between two instruments, such as one instrument loosing GPS signals, and with only the instrument’s internal quartz clock available the time will drift until the next GPS time fix. But the question was more about a constant time offset, and not a time drift.

It is of course also possible that the two instruments used a different GPS, and if they had different constant errors this would give a constant time offset. But this error is likely to be so small that you would not discover it given the problem described. Remember that the GPS system depends on very accurate clocks in order not to get very large position errors. Given a position error of 100 meter, which is very large, this corresponds to a time error of about 33 µs. ($S = v * t$, where $v = 3*10^8$ m/s). So this is not a likely explanation. If the instruments where placed at different geographic locations with large separation each GPS could experience different effects on the GPS signals through the transmission path from the satellites. This could also cause a time offset. But again, this is not so likely to be the explanation, because the error is very likely to be too small to be discovered her. (Also remember that 1 µs sample interval of an instruments correspond to a sample frequency of 1 MHz).

Problem 2

a) Explain why UDP is used instead of TCP in applications such as video streaming.

UDP is not reliable (packets can be lost)

- No flow control
- No saturation control
- No retransmission of data

However, UDP is a fast protocol with small overhead (compared to TCP), and shorter delay (reaction time down to about 10 ms). Sometimes losing a few data packets is not a problem, and a **short delay** is more important than receiving 100 % of the data packets, such as in video steaming. Therefore UDP is used for broadcast and multicast of data, such as video. Also it is used in timing protocols such as NTP. Sometimes there is also not sufficient time to retransmitt the data if data are missing (and still get a "real-time" data stream), such as when video signals are distributed from satellites.

b) What is a **jumbo frame**, and why is it used?

A typical network data package (frame) used to be about 1500 bytes. A jumbo frame is a larger data packet, and a common jumbo frame size is about 9 kB. IPv4 supports jumbo packets up to 64 kB. By using jumbo packets you can transmit the same amount of data with fewer packets. Though you save a small amount of bandwidth (by using fewer headers), you dramatically reduce CPU usage because the PC spends less time analyzing packets.

c) What is required in order to run data acquisition software on a modern computer running Windows with protected mode? Explain.

Need a **device driver** in order to allow hardware I/O operations from application programs.

In protected mode, there are four privilege levels or rings, numbered from 0 to 3, with ring 0 being the most privileged and 3 being the least. The operating system and some device drivers run in ring 0 and applications run in ring 3

d) What are the advantages and disadvantages of selecting RAID-10 when doing data acquisition?

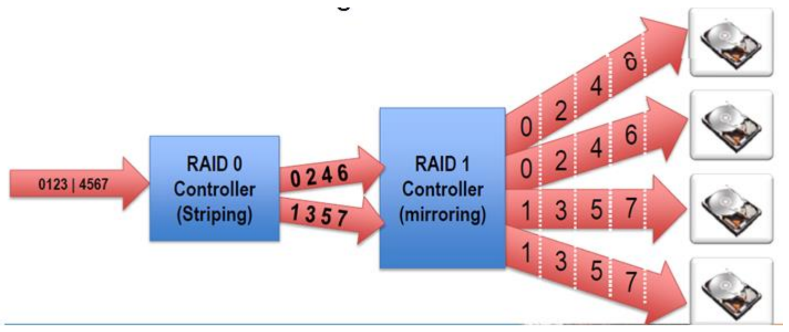
RAID-10 = RAID0 + RAID1 = Striping and mirroring

Advantages:

- Both increased speed and redundancy compared to a single drive
- Can sustain multiple drive failures
- Fast rebuild as data is copied block for block from the source to the target. (No parity calculations are required as in RAID-5 and RAID-6).

Disadvantages:

- Configuration requires twice the number of hard drives (compared to RAID-0)

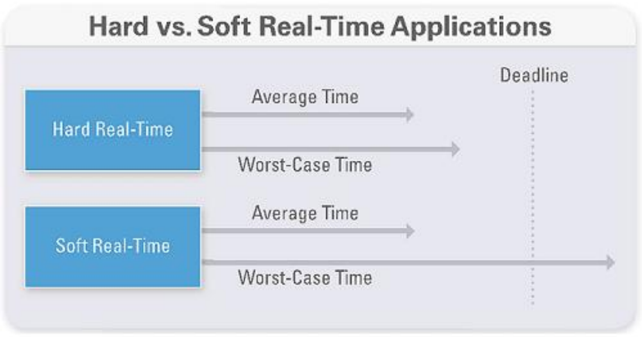


e) What is the "definition" of a real time system?

A real-time system gives you **determinism,** meaning that you can predict when a section of your program will execute.

<u>Hard real-time</u>: system where it is absolutely necessary that responses occur within the required **deadline**

<u>Soft real-time</u>: allows for some deadlines to be missed with only a slight degradation in performance but not a complete failure.



f) Many computer-based data acquisition systems are running on non-real-time hardware using operating systems such as Windows 7. Explain the main hardware and software solutions used to make this possible.

We need to add <u>data buffers</u> in our system, and to make the data transfer and our software as efficient as possible (to avoid overwrite and overflow errors).

<u>Hardware solutions:</u>

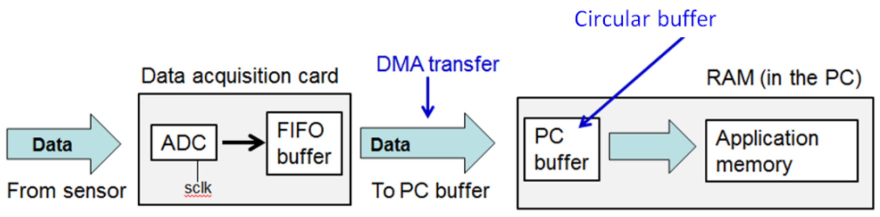
- Direct Memory Access (**DMA**) transfer mechanism (the fastest way to move data, and allows the CPU to do other work), using the <u>DMA-controller</u>.

- Buffered acquisition **- FIFO buffer on the DAQ card**
- **RAID**-hardware and/or SSD drives (to increase write speed)

Software solutions:

Since it is not possible to ensure that a software code will finish within a specified deadline on a non-real-time computer and operating system the following techniques are used in software:

- Buffered acquisition - using a **circular PC-buffer in RAM**
- **Producer-consumer structure** using **multiple threads** ("independent" acquisition and write-to-file speed).
- Data **queues** to send data between producer and consumer loops
  - Queues contain a buffer (it is a FIFO) to make sure that no data are lost.



g) How can you to increase the signal-to-noise-ratio (SNR) in a data acquisition system?

- **Using an amplifier** (as close to the sensor as possible, to amplify the signal before the noise enter e.g. the transmission cable), for instance with automatic gain control (AGC), to use the entire analog-to-digital converter (ADC) dynamic range.
- **Use an ADC with more bits per sample (**The SNR of an ideal N-bit ADC is SNR(dB) = 6.02*N + 1.76
- **Use oversampling** (followed by digital LP-filtering and down-sampling).
- **Filtering** (to remove noise and limit the signal bandwidth), in hardware or software
  - (Remember from basic electronics that thermal noise (Johnson noise) in a resistors is proportional to the square root of the signal bandwidth)
- Averaging (when constant or periodic signals and random noise);
- Apply general noise reduction techniques:
  - Position noise sources away from data acquisition device, cable, and sensor if possible
  - Place data acquisition device as close to sensor as possible to prevent noise from entering the system

- Twisted pairs, coax cable, shielding
- Use differential signals
- Avoid ground loops.

Topics not covered in lectures:

- Smoothing
- Lock-in amplifier

h) The equivalent circuit for a sensor and the DAQ front-end electronics is shown in Figure 1. The sensor is modelled as an ideal voltage generator with output voltage V_{in} followed by a parallel connection of a resistor R_s and a capacitance C_s on the output. The DAQ front-end electronics is modelled as an ideal amplifier with gain 1, with a parallel connection of a resistor R_L and a capacitor C_L on the input. The signal cable between the sensor output and the front-end electronics input is assuemd ideal, with zero resistance and zero capacitance.

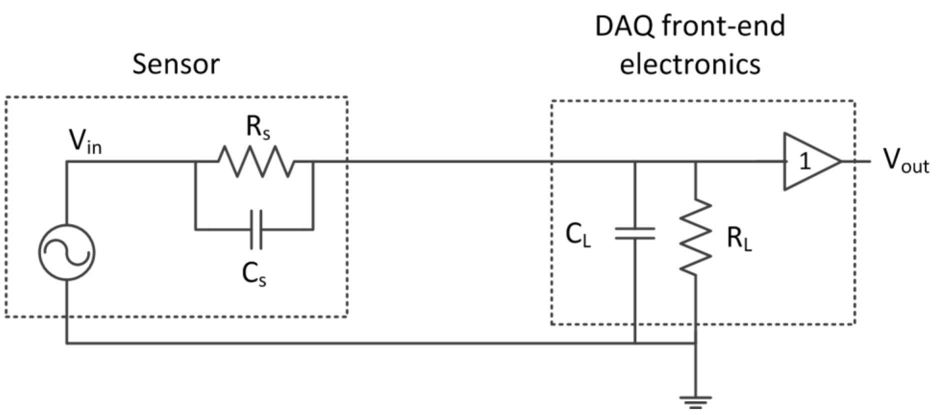


Figure 1: Equivalent circuit for a sensor and the DAQ front-end electronics (the signal cable between the sensor output and the DAQ front-end electronics input is assuemd ideal, with zero resistance and zero capacitance).

Look at the following two cases:

1) very low frequency of the input signal from the sensor ($f \rightarrow 0$)

2) very high frequency of the input signal ($f \rightarrow \infty$)

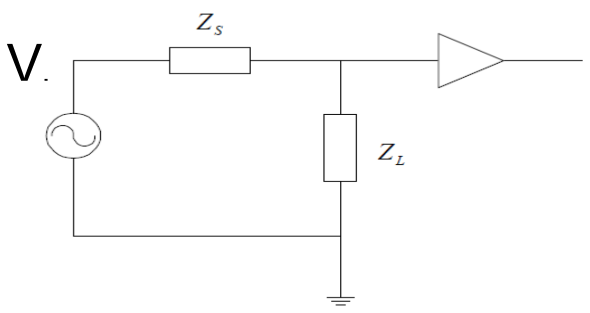
Show that the conditons $R_L >> R_s$ and $C_L << C_s$ are required in order to get Vout ≈ Vin

Hints:
All calculations can be based on V = Z * I (Ohm's law with a complex impedance). Look at the circuit as a voltge divider between the sensor impedance and the load impedance.

The following relations are provided to assist in the calculations:

- Voltage divider: $V_{out} = \frac{Z_L}{Z_s + Z_L} \cdot V_{in}$

- Impedance of a resistor: $Z_R = R$
- Impedance of a capacitor : $Z_C = \frac{1}{j\omega C}$, where $\omega = 2\pi f$
- Impedance of a parallel connection of a resistor and a capacitor: $Z_p = Z_R \parallel Z_C = \frac{Z_C \cdot R}{Z_C + R}$



$$V_{out} = \frac{Z_L}{Z_s + Z_L} \cdot V_{in}$$

1) <u>Low frequency case where f → 0:</u>
The absolute value of the capacitor impedance ($|Z_C|$) is very high, and the impedance of the parallel connection can be approximated as:

$$Z_p \approx \frac{Z_C \cdot R}{Z_C} = R$$

The voltge divider between the sensor impedance and the load impedance therefore gives:

$$V_{out} \approx \frac{R_L}{R_S + R_L} \cdot V_{in}$$

This requires $\underline{R_L >> R_s}$ in order to get a denominator that is approximately equal to R_L, and <u>Vout ≈ Vin</u>

2) <u>High frequency case where f → ∞:</u>

The absolute value of the capacitor impedance is very low, and the impedance of the parallel connection can be approximated as:

$$Z_p \approx Z_C$$

The voltge divider between the sensor impedance and the load impedance therefore gives:

$$V_{out} \approx \frac{Z_{C,L}}{Z_{C,S} + Z_{C,L}} \cdot V_{in} = \frac{\frac{1}{C_L}}{\frac{1}{C_S} + \frac{1}{C_L}} \cdot V_{in}$$

This requires $C_L << C_S$ in order to get the denominator approximately equal to $1/C_L$, and Vout ≈ Vin