System Programming

Final assignment

2016 - 2017

Bachelor Electronics/ICT

Course coördinator: *Luc Vandeurzen*
Lab coaches: *Jeroen Van Aken*
Stef Desmet
Tim stas
Luc Vandeurzen
Floris De Feyter

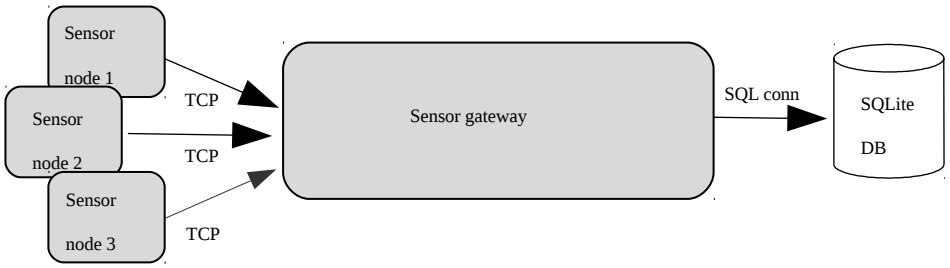
Last update: May 11, 2017

Academic Integrity

*This is an **individual** assignment! It is only allowed to submit **your own work**. You may discuss the assignment with others but you are not allowed to share work or use (part of) another's solution. If you include work (e.g. code, technical solutions,...) from external sources (internet, books …) into your solution, you must clearly indicated this in your solution and cite the original source. If two students present very similar solutions, no distinction will be made between the 'maker' and the 'copier'.*

Sensor Monitoring System

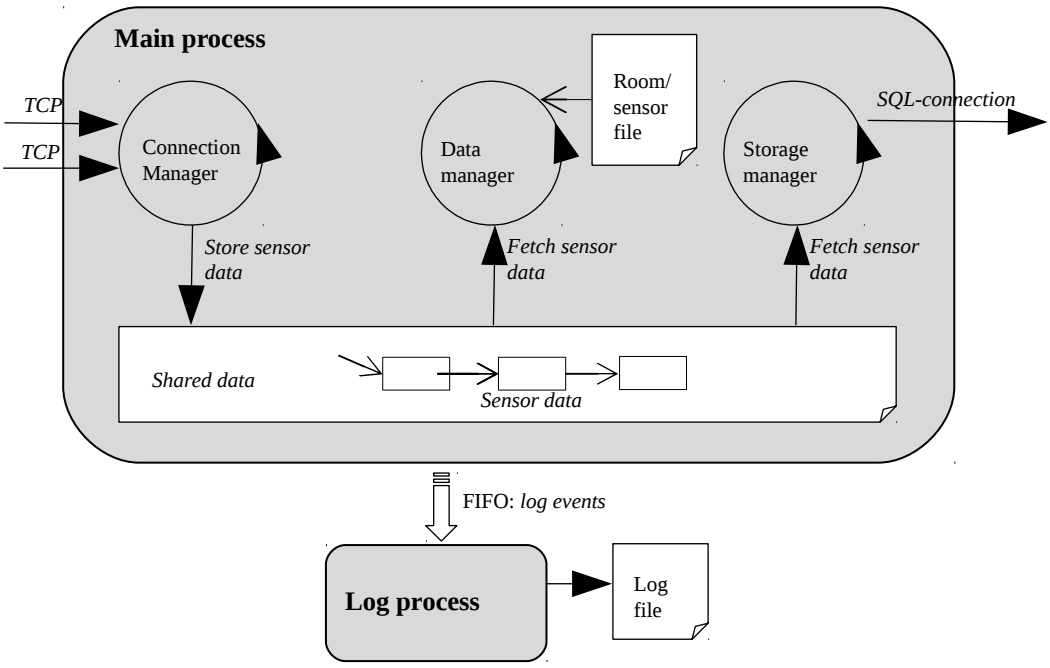
The sensor monitoring system consists of sensor nodes measuring the room temperature, a sensor gateway that acquires all sensor data from the sensor nodes, and an SQL database to store all sensor data processed by the sensor gateway. A sensor node uses a private TCP connection to transfer the sensor data to the sensor gateway. The SQL database is an SQLite system (see lab 7). The full system is depicted below.



The sensor gateway may **not** assume a maximum amount of sensors at start up. In fact, the number of sensors connecting to the sensor gateway is not constant and may change over time.
Working with real embedded sensor nodes is not an option for this assignment. Therefore, sensor nodes will be simulated in software (see lab 8).

Sensor Gateway

A more detailed design of the sensor gateway is depicted below. In what follows, we will discuss the minimal requirements of both processes in more detail.



Minimal requirements

Req 1. The sensor gateway consists of a main process and a log process. The log process is started (with fork) as a child process of the main process.

Req 2. The main process runs three threads: the connection, the data, and the storage manager thread. A shared data structure (see lab 9) is used for communication between all threads. Notice that read/write/update-access to the shared data needs to be *thread-safe*!

Req 3. The connection manager listens on a TCP socket for incoming connection requests from **new** sensor nodes. The port number of this TCP connection is given as a command line argument at start-up of the main process. e.g.: `./server 1234`

Req 4. The connection manager captures incoming packets of sensor nodes as defined in lab 8. Next, the connection manager writes the data to the shared data structure.

Req 5. The data manager thread implements the sensor gateway intelligence as defined in lab 6. In short, it reads sensor measurements from shared data, calculates a running average on the temperature and uses that result to decide on 'too hot/cold'. It doesn't write the running average values to the shared data – it only uses them for internal decision taking.

Req 6. The storage manager thread reads sensor measurements from the shared data structure and inserts them in the SQL database (see lab 7). If the connection to the SQL database fails, the storage manager will wait a bit before trying again. The sensor measurements will stay in shared data until the connection to the database is working again. If the connection did not succeed after 3 attempts, the gateway will close.

Req 7. The log process receives log-events from the main process using a FIFO called "`logFifo`". If this FIFO doesn't exists at startup of the main or log process, then it will be created by one of the processes. All threads of the main process can generate log-events and write these log-events to the FIFO. This means that the FIFO is shared by multiple threads and, hence, access to the FIFO must be thread-safe.

Req 8. A log-event contains an ASCII info message describing the type of event. For each log-event received, the log process writes an ASCII message of the format

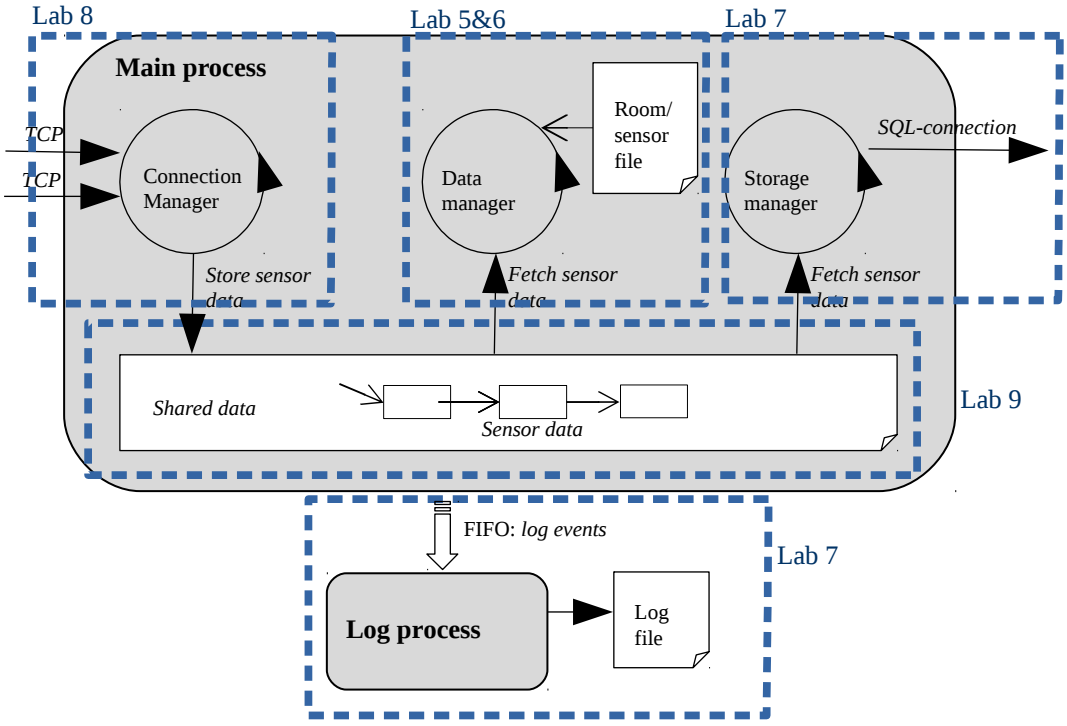
`<sequence number> <timestamp> <log-event info message>` to a new line on a log file called "`gateway.log`".

Req 9. At least the following log-events need to be supported:

1. From the connection manager:
   a. A sensor node with <sensorNodeID> has opened a new connection
   b. The sensor node with <sensorNodeID> has closed the connection
2. From the data manager:
   a. The sensor node with <sensorNodeID> reports it's too cold (running avg temperature = <value>)
   b. The sensor node with <sensorNodeID> reports it's too hot (running avg temperature = <value>)
   c. Received sensor data with invalid sensor node ID <node-ID>
3. From the storage manager:
   a. Connection to SQL server established.
   b. New table <name-of-table> created.
   c. Connection to SQL server lost.
   d. Unable to connect to SQL server.

How to start?

The sensor gateway is a kind of integration result of the code of lab 5 up to lab 9. The picture below shows the relationship between the different components of the sensor monitoring system and the lab sessions. It should be clear from the description of the requirements which pieces of code of these labs are required for the sensor gateway.



Deliverables and acceptance criteria

On **syssoft.groept.be** you will find a description of what exactly and in which format (source code files, directory structure, make file, etc.) you need to prepare and upload your solution. Once you have finished the assignment, you upload your solution on **syssoft.groept.be**.

Take the following thoughts into account when designing a solution:

- Re-factor and clean up the code you upload. Code structure and readability are also evaluation criteria!
- Implement *efficient code* (e.g. optimize your algorithms, use efficient loops, etc.). 'But it works ...' should not be your design mantra! Design and performance are also evaluation criteria!
- **syssoft.groept.be** is not a compilation nor test tool. Write test code – also non-trivial test code – yourself and thoroughly test your code. For instance, test your solution with multiple sensor nodes running concurrently using very small sleep times between two measurements. Include a debug-mode.

Your solution will only be accepted for grading if the following criteria are fullfilled:

- Your solution is available on **syssoft.groept.be** before the deadline;
- Your solution has passed all the available tests on **syssoft.groept.be**;
- Your solution must be a reasonable try to implement **all** minimal requirements. This excludes, for instance, solutions that consist of an (almost) empty .c file, incomplete code, or code that has no or very little relationship to the exercise, code that implements a different assignment (e.g. from a previous academic year), etc.;
- Your solution has no or too little exception/error handling (e.g. malloc without NULL-test, empty list checks, switches without breaks, no asserts, etc);
- Your source code is unstructured and/or unreadable. For example, you don't logically structure the code into source and header files, you apply bad naming of variables and functions, you don't use typedefs to create logical names, the code is not indented well, you write 'spaghetti-code' with goto's, …;
- You must be able to present and defend your solution. During the defense of your solution, you are supposed to be able to answer on the technical questions of the evaluator. This includes technical questions related to programming (C, Linux system calls), source code building (preprocessor, compiling, linking, Valgrind, …) and Linux command line basics. The reference guide for the minimal knowledge on Linux are the Linux lab manuals. You are supposed to be able to work with the commands and tools introduced in these lab sessions. You should also be familiar with the usage of the programming tools introduced in the labs. More concretely, we target at least the following tools **and their options**:
  - gcc tool set (preprocessor, compiler, linker, assembly) and options (preprocessor symbols, code optimization, debug flag, ...)
  - creating static and shared libraries (ar, ldd, ldconfig, gcc), linking libraries (gcc)
  - valgrind
  - make tool and make files (you have to be able to compile and run your code using a simple make file)

Due date: June 9, 2017 before 17 p.m.

Solutions uploaded after the deadline will be rejected. The code presented during the defense of your solution must be the **same** code as uploaded and tested on **syssoft.groept.be.**

You must defend your solution on the date and location stated on the examination schedule. To create a more efficient planning, you must subscribe **before the due date of this assignment** to one of the events available on https://tolapps.kuleuven.be/Tolinto/event/8656776836_gII .