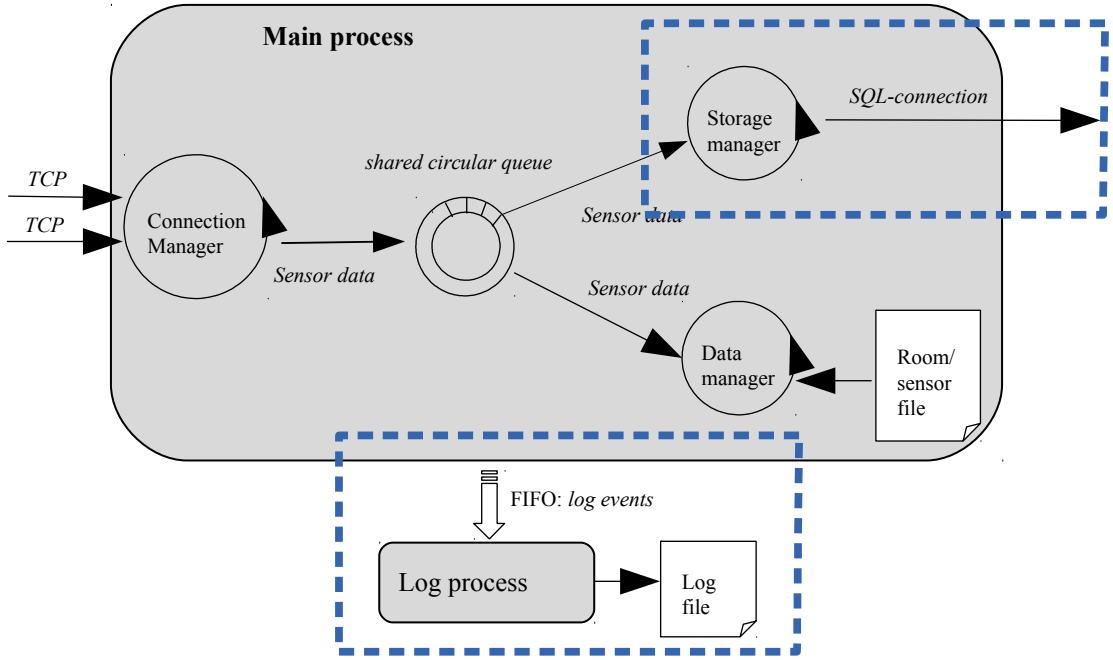
# *Lab 7 – File I/O, database interfacing, processes and pipes*

*For your information*

*The picture below visually sketches the final assignment of this course. The relationship of this lab to the final assignment is indicated by the dashed blue line.*




***Exercise: MySQL interfacing***

Implement a program that connects to the MySql server running on “studev.groept.be”. Login using the account “a13_syssoft” with password “a13_syssoft”. (Connection will only work from within the GroupT network, outside you can test on a local MySQL server) Once connected to the database server, the program creates a new table “yourname” in the existing database “a13_syssoft”. This table should have the following columns:

- id: automatically generated unique id
- sensor_id (INT)
- sensor_value (DECIMAL(4,2))
- timestamp (TIMESTAMP)

The C API of MySQL is required in order to implement this program. You can install the MySql client developer package with the following command:

> sudo apt-get install libmysqlclient-dev

Compile your code as follows:

> gcc yourfile.c $(mysql_config –cflags –libs)

*Some information sources:*

- A tutorial that will guide you all the way through this exercise: http://zetcode.com/tutorials/mysqlcapitutorial/
- The MySql reference manual discusses the full C API: dev.mysql.com/doc/refman/5.0/en/c-api.html

***Exercise: File I/O and MySQL interfacing***

Adapt the program of the first exercise such that all sensor measurements read from the file 'sensor_data' (see lab 6) are now inserted into the database table created in previous exercise.

Experiment with some queries to read sensor data from database**.**

*THE SOLUTION OF THIS EXERCISE NEEDS TO BE UPLOADED ON TOLEDO BEFORE THE DEADLINE.*

*Your solution is only accepted if the following criteria are satisfied:*

1. *Compilation with '-Wall -Werror' option generates no warnings or errors*
2. *Running 'cppcheck –enable=all --suppress=missingIncludeSystem' on the source code results in no warnings or errors*
3. *Write a test program main.c to check if the implementation of all methods is correct and gives result as expected.*

    - *Make sure the binary file containing sensor measurements is named 'sensor_data'*
    - *The program to communicate with database is named 'sensor_db.c'*
    - *Needless to mention, restrict your implementation to config.h and sensor_db.h*

*DO NOT upload code that doesn't satisfy these criteria.*

***Exercise: Processes and pipes***

Change the program above (= storage manager) such that a new log process is created and a communication pipe between the storage manager and the newly created log process is installed. The log process receives log-events from the storage manager. and writes them to a text file (the log file).

A log-event contains an ASCII info message describing the type of event. A few examples of log-events are:

- Connected to SQL server.
- Unable to connect to SQL server.
- New table <name-of-table> created.
- Data insertion failed.

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".