Motionlogger 2

– Software implementation of data logging at high frequencies

Jens Svensson, IEA, LTH April 2010

In today's industry the need for fast logging of measurements is increasing. Existing systems used by Tetra Pak AB have a maximum logging frequency which is inadequate. As a result of this Tetra Pak gave Industriprojektbyrån AB the task of developing a measuring system for this purpose. Industriprojektbyrån AB has before the start of the thesis work developed the hardware set up for the system. This article describes the further development of the system which is about developing the software part of the system. The software development includes designing the software and implementing the code. And the final part of the thesis is testing of the system.

Tetra Pak has need for fast logging of data from encoder sensors. Encoding sensors are primarily used for measuring position of rotating axes. According to Anders Sundberg the frequency of the phenomena's that the system is designed to detect is typically up to 3-400 Hz. For further uses it also could be interesting to see the output from the control loop which goes up to 2 kHz. Tetra Pak uses an existing system for logging measurements from incremental encoders that has a maximum logging frequency of 2 kHz. The logging frequency of 2 kHz is not enough today. As a result of this Tetra Pak gave Industriprojekbyrån AB the task of developing a measuring system with high logging frequency. Industriprojektbyrån AB has before the start of the thesis work developed the hardware set up for the system. The specification for the thesis was to develop a system that could log data under 30 s with a logging frequency of at least 50 kHz. In addition to the logging of data from encoder sensor the system can log data from analog and digital sensors.

Operating system

The first step in development of the system was to choose and configure the operating system. Since the application is developed as a real time application, the operation system needs to have a configuration that supports the real time. Real time (RT) means that the application is subjected to timing deadlines i.e. the application needs to finish a task within a certain time. A small evaluation was made to determine what operating system to use. The result fell on the Linux based, open source Xubuntu. To configure Xubuntu for running a real time application it is necessary to recompile the kernel. The kernel is a central component for the operation system. The kernel handles the communications between the applications and the hardware. The reason for the recompilation of the kernel is that some non standard options needs to be enabled. The most important option here is to enable the real time support for the kernel and to disable as many as possible of the system management interrupts. System management interrupts is a special kind of hardware driven interrupts that handles power management for the computer.

Software design

The best way to explain the system designed is by a block flow diagram, see Figure A.

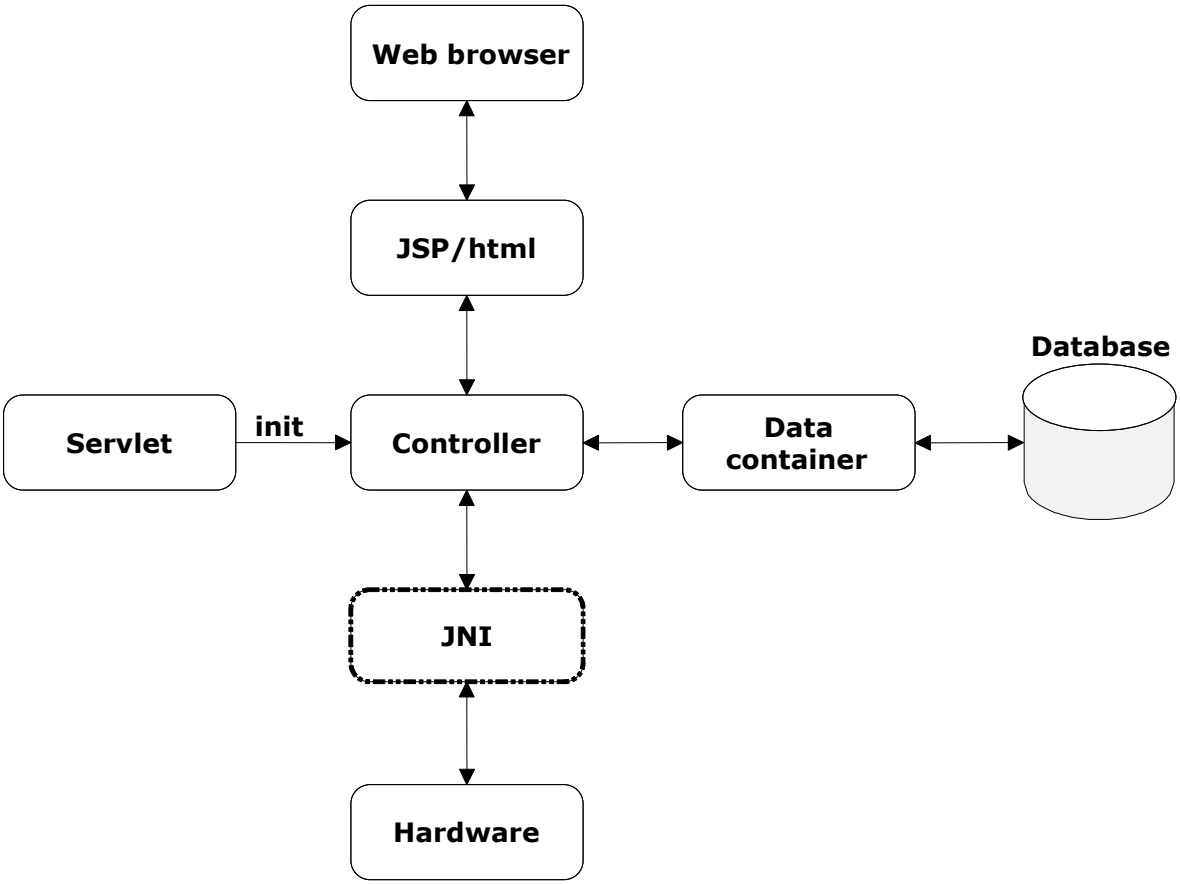

Figure A. Block flow diagram for the application. Arrows represent communication paths.

The application is designed as a servlet and will run under Apache Tomcat. A servlet is server application that is implemented entirely in Java. Apache Tomcat is a web server for servlets. The design is based on and use parts of the Open Process Logger framework. The Open Process Logger is an open source framework for collecting, storing and presenting data from PLC (Programmable Logic Controller) systems and is developed by Industriprojektbyrån AB.

The communication with the hardware is done via a Java Native Interface (JNI). This is by far the largest and most complicated part of the software. All logging algorithms and RT-logic are located here.

When logging is complete the data is kept in a special container class. Logics for post processing of the data are located in this container. Database communication is also done from this class. The data container and the controller are implemented in pure Java. The only purpose for the controller is to work as a communications' hub. Both the controller and the data container are constructed in such a way that only one instance of each may exists simultaneously.

The user interface is constructed as a webpage using JSP and html. The webpage can naturally be accessed via a web browser. The logging functions are activated from the webpage and the data presented as graphs. For data presentation the embedded graphviewer from the Open Process Logger is used.

Java Native Interface

The Java Native Interface is a feature of the Java language which makes it possible to declare methods in Java and implement them in native language. In this application, C is the native language used. Using Java Native Interface compromises the portability feature of Java but instead gets access to the C real time features.

Post processing

The system also has implemented functions for derivation and filtering of the data. The system has functions for the 1st or 2nd derivative. It is also possible to low pass filter the raw data from the logging and in additions the data can be stored in a database. The database is part of the Open Process Logger framework and implemented in Apache Derby.

System testing

Continuously during the development the system has been subjected to a series of test regarding the functionality and performance of the system. The most important results are the logging frequencies for the system. The frequencies are shown in Figure B. The final system is able to log data for time periods up to 15 s. Also important is the tests regarding the determinism of the system. The tests showed that the system was not fully deterministic but relatively close.

Conclusions

The thesis work resulted in fully functional system according to the original specifications. All of the specifications are not accommodated due to physical limitations of the hardware. The system is however not complete for delivery to the customer due to additional requests from the costumer. These requests have appeared naturally during development of the system. The extensive workload made it impossible to address these during thesis.

Future works

The suggested future works is divided into three steps. The first step is about finishing the development so the system can be delivered. There are only two issues that need to be addressed and they are: modifications in the memory management functions and a method for deleting of old data from the database. The second step is about improving the performance for the system and eliminating the short comings related the hardware. In very short form this could be done by upgrading the computer and compile a new kernel. The last step is about post processing of the data and improvement of the user interface. The post processing of the data include filters, Fourier transformation and exporting data.

Logging type:	Frequency:
One encoder channel	127 kHz
All encoder channels	43 kHz
One analog channel	48 kHz
Two analog channels	15.5 kHz
Digital	160 kHz

Figure B. Logging frequencies for the system