

Creating a climate chamber test system with Compact Fieldpoint, LabVIEW and GOOP

Author

Jeffrey Habets, NBG Industrial Automation

Industry

Research, Industrial Controls/Devices/Systems

Product

Compact FieldPoint, Distributed I/O, LabVIEW, Endevo GOOP Inheritance Toolkit, Endevo UML Modeller

The Challenge

Update a 10 years old climate chamber test system from LabVIEW 2.2 Macintosh with old NI DAQ hardware to the latest hardware and software standards.

The Solution

The customer ([Barcol-Air](#)) does comfort measurements on air distribution systems they develop for office buildings by performing measurements in an adjustable (in size) room. The room is equipped with a whole bunch of Air Velocity Transducers and temperature sensors attached to a moveable carriage which is driven by a stepper motor. This system was 10 years old and running on LabVIEW 2.2 on an Apple Macintosh using very old NI DAQ hardware. The data acquisition system is positioned outside the climate room. Quite a large bunch of wires was going from the Mac to the sensor carriage. Apart from the sensors on the carriage there are also a couple of static sensors inside and outside the room, mainly temperature and pressure sensors.

This system needed to be upgraded to latest standards and so it was decided to go to a PC platform with Windows and LabVIEW 7.1. The DAQ hardware was replaced with two Compact Fieldpoint systems. One is positioned on the carriage and the other one outside the room. So the bunch of cables from the Mac to the PC was replaced with a single Ethernet cable. The cFP controllers are used in passive mode which means their sole function is to connect the modules with the PC and no LabVIEW Realtime programming needs to be done.

Hardware used

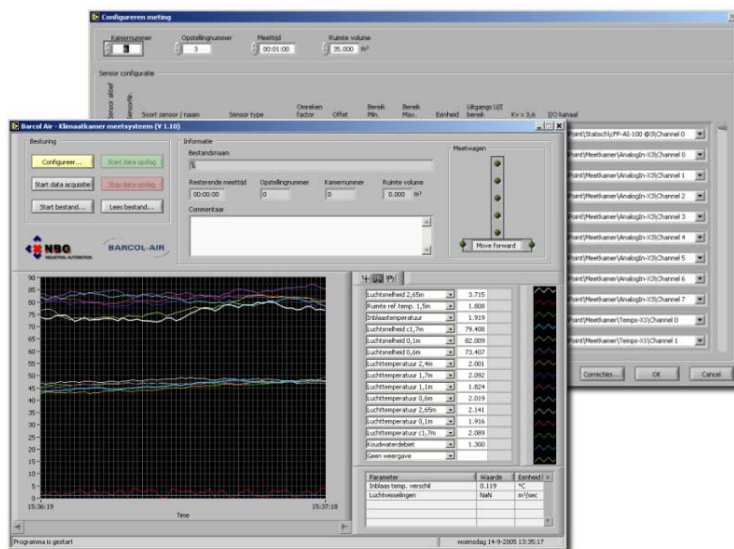
- 8-slot cFP with cFP2000, cFP-RTD-122, cFP-DI-301, 2 x cFP-AI-100, cFP-RLY-421, cFP-TC-120
- 8-slot cFP with cFP2000, 2 x cFP-RTD-122, cFP-AI-100, cFP-CTR-500
- PC with WindowsXP

Development tools used

- LabVIEW 7.1 + OpenG libraries
- Endevo GOOP Inheritance Toolkit
- Endevo UML Modeler

Requirements

The application should be able to acquire data from approximately 40 sensors and control the carriage motor. Acquisition rate is fixed (although easily changeable) at 1 Hz. The duration of the acquisition is configurable, as well as some other parameters. The acquired data is saved in a csv file so it can easily be imported in Excel for further data analysis. The actual data saved is a moving average of 100s on most of the sensors (also easily adjustable if necessary). The application has the ability to playback the data from a file in real-time (like simulating reading from the cFP modules) as well as read the data and display it directly without delay on the onscreen graph. The user is able to select which channels are displayed in the graph on screen and is able to add comments to the datasets written to disk. Channels are fully configurable by the user. The user can name channels, enter gain, offset, etc.



The application UI - simple and effective

Development

Development of the application was done using GOOP, Graphical Object Oriented Programming in LabVIEW. We used GOOP and mix it with 'normal' dataflow programming which gives us best of both worlds in one programming environment. Using the GOOP tools we are able to use UML to design our software and generate the needed classes and their methods from the class diagram. We then code the functionality (the methods) for the classes and integrate the classes into the main program. If the blueprint for a class changes during implementation (e.g. addition of methods or attributes) these changes are automatically reflected back to the UML class design by using the UML Modeler's reverse engineering feature. This way we always have an up-to-date description of the global application functionality.

