Remote temperature measurement with and without FPGA

Aurel Gontean, Roland Szabó

***Abstract* — This paper presents three ways of measuring the temperature. For measuring the temperature we used LM35 Centigrade sensor. One approach is using no FPGA (cheap and expensive), another method is using FPGA. The programming was done in LabVIEW. The most important result is that the experiment can be accessed over the internet (http://plst.etc.upt.ro) and anyone can measure the temperature from our laboratory just by using a simple web browser. We use it to check if the air conditioning system works correctly for our servers. We do a continuous temperature monitoring of the laboratory's temperature.**

***Keywords* — FPGA, LabVIEW, LM35, real-time, reconfigurable I/O, remote server, temperature, web.**

I. INTRODUCTION

THIS paper presents a very interesting experiment, we made three experiments to measure the temperature.

The first one is using the NI USB-6251 M Series Multifunction DAQ. This method is probably the best, because the equipment has a 5V DC power supply which is needed for the LM35 sensor, this way there is no need for external power supply.

The second method is using a the NI USB-6009 Low-Cost Multifunction DAQ. This second method requires an external power supply for the temperature sensor, but is has some advantages. One advantage should be that is about 5 times cheaper than the NI USB-6251. Another advantage should be that it's driver can be downloaded from the vendor's internet page for no charge.

The third method is maybe the most expensive, but it has some advantages too. The program is made on FPGA, it has a special casing (metal pored over the circuit) so it has resistance to shocks up to 50g, supports temperatures from -40 to 70 °C and it has a reliable real-time operating system called VxWorks Wind River. This method has the disadvantages of being expensive and the programming requires a better knowledge in LabVIEW.

II. THE FIRST EXPERIMENT WITH NO FPGA

A. *The NI USB-6251 Multifunction DAQ*

This equipment is from the M Series family and is one of the most recommended by the producer.

Aurel Gontean is with Applied Electronics Department, Faculty of Electronics and Telecommunications, "Politehnica" University of Timişoara, România (e-mail: aurel.gontean@etc.upt.ro).

Roland Szabó is Master student, Faculty of Electronics and Telecommunications, "Politehnica" University of Timişoara, România (e-mail: roland.szabo@etc.upt.ro).

Why, because it has a lot of functions in a relatively little box. It's easy to install it because it connects to the PC with a USB cable. It's size is approximately like an A5 paper. It has 16 analog inputs, 2 analog outputs and 24 digital I/O lines. It comes in two forms factors: Mass Term and Screw Term. To the Mass Term the user can connect a connector block via a cable, the Screw Term has a built in connector block. We used this second one.

B. *The Block Schematics of the Experiment*

The LM35 sensor is an analog sensor, which is mostly in a TO92 capsule and it has 3 connectors, one for V_{CC}, one for the ground and one for the signal. The signal that provides is a voltage that is equal with the temperature in Centigrade divided with 100. We had to measure mV, so we needed great accuracy. Simply multiplying the measured voltage with 100 we obtained the temperature in Centigrade.

The method of connecting the temperature sensor is illustrated on the block schematics from Fig. 1.

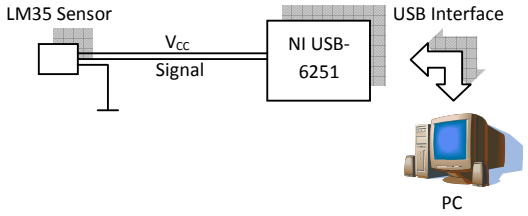

Fig. 1. Block schematics of the experiment

C. *Front Panel of the Program in LabVIEW*

The Front Panel (Fig. 2.) it's basic, it shows the device's name, the maximum and minimum value of the channel (between -10 and 10V), the temperature converted in °C and viewed on Waveform Graph and the sample rate (bigger – better accuracy, lower speed; smaller – the opposite).

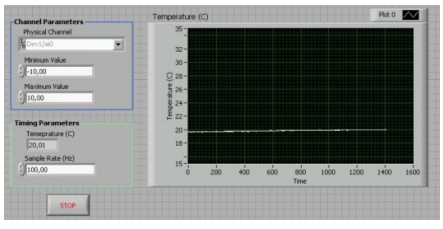

Fig. 2. Front Panel of the experiment

D. The Block Diagram of the Program in LabVIEW

The Block Diagram uses a single While loop with the configuration of the channel to measure voltage. The voltage is multiplied with 100 to obtain the temperature in Centigrade. We used a mean function to make the signal smoother. The shift registers are used to update in real-time the static Waveform Graph. One of the most important setting of this block diagram is at the first icon when we configure the channel, we have to specify if we connected the sensor differentially or single ended. When is differentially, then is a positive and negative connection, when is single ended then is only a V_{CC} and a GND. The options are: default, RSE, NRSE, Differential, Pseudo differential. On Fig. 3. we can see the Block Diagram of the experiment.

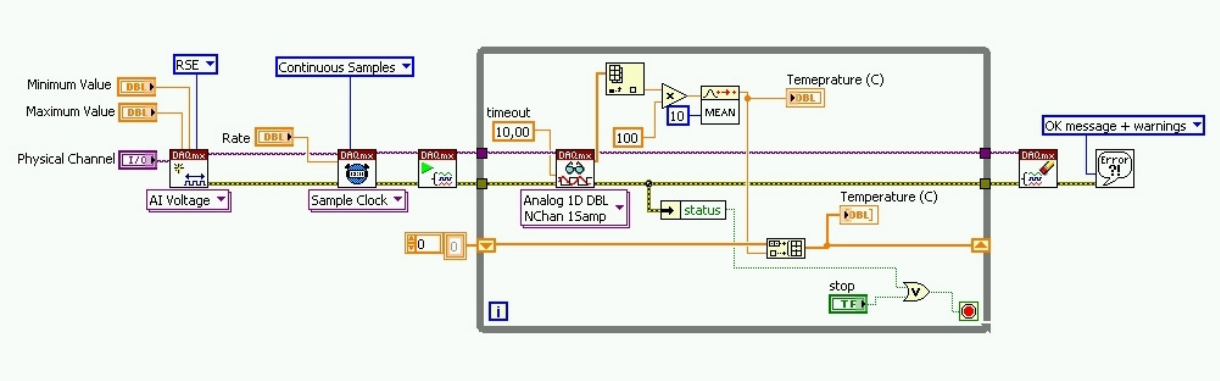

Fig. 3. The Block Diagram of the experiment

III. The Second (Cheap) Experiment With NO FPGA

A. The NI USB-6009 Multifunction DAQ

This equipment is one of the cheapest and smallest USB data acquisition cards. The producer provides a driver for free that is perfectly compatible with this equipment. It has 8 analog inputs, 2 analog outputs and 12 digital I/O lines. It is important to know that not only it has fewer I/O lines then the NI USB-6251 card. It's speed is lower, it's speed is only 48kS/s compared to the other one which has 1.25 MS/s. It is only on 14-bits comparing to the 16-bits card introduced in chapter II. It also has some restricted functions comparing to his "bigger brother". The other card can do frequency and temperature measurements directly, this one works perfectly only in voltage measurements and it's channels' configuration is reduced, it cannot support all of the: default, RSE, NRSE, Differential, Pseudo differential settings, it supports only single ended and differential.

For this experiment the Front Panel is exactly the same with the other experiment (Fig. 2.), even the Block Diagram is the same (Fig. 3.) and even the free driver icons are the same with the driver icons that are non free, the only difference is that on the non free driver icons is written DAQmx and on the free icons is written mx Base.

The Block Schematics is similar to the one on Fig. 1., but with the difference, that this equipment has no DC 5V and needs an external power supply.

IV. The FPGA Experiment

A. The reconfigurable I/O Equipment

This equipment is a robust industrial equipment. It requires a Virtex 5 FPGA chassis with a real-time controller and some acquisition modules. For this temperature measurement we have chosen the NI 9201 C Series Analog Input Module.

It is maybe the poorest in characteristics. It has only 8 analog inputs, it is on 12 bits and it's speed is 500kS/s. Only it's speed is better than the USB-6009, but according the direct connection to the FPGA chassis and real-time controller and the connection to the PC on Ethernet, this speed is not the best.

B. The Block Schematics of the Experiment

As we can see on Fig. 4. the main idea is the same with the other block schematics (Fig. 1.), but slightly different, because it needs an external 5V DC power supply and the equipment is more interesting, we need an FPGA chassis, a real-time controller and an analog input (AI) module and all of these are connected trough Ethernet interface.

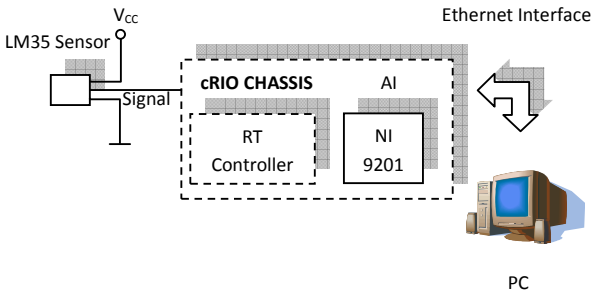

Fig. 4. The block schematics of the experiment

C. The Front Panels of the Programs in LabVIEW

For this reconfigurable I/O program to work on our Windows environment we have to make four different programs: one on FPGA, one as the real-time host, one as

the networked real-time host (this one will send the variables to Windows) and one as the Windows host. Not all these programs will run in parallel. There are two ways of running real-time program: we can run the FPGA program trough the RT host, or we can run the FPGA program trough the networked RT host (this one will send the variables to Windows trough Ethernet) and then we use a Windows GUI to receive them. The second method requires to run two programs in parallel the networked RT host program and the Windows GUI.

On Fig. 5. we can see the FPGA program's Front Panel with clock settings, the temperature and an error cluster.

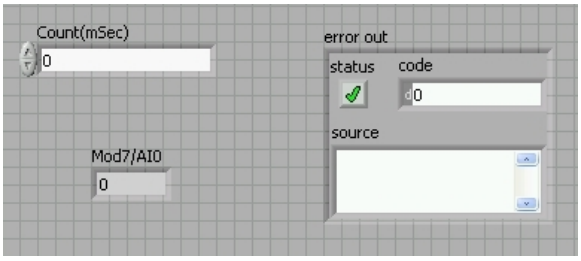

Fig. 5. FPGA program Front Panel

On Fig. 6. we can see the RT host's and the networked RT host's front panel (these differ only in the Block Diagram). We can see here two graphs: a Waveform Graph for logging and a Waveform Chart for real-time measurements. The Waveform Graph has a big log of a whole day showing a big temperature change caused of shutting on and off the air conditioning system.

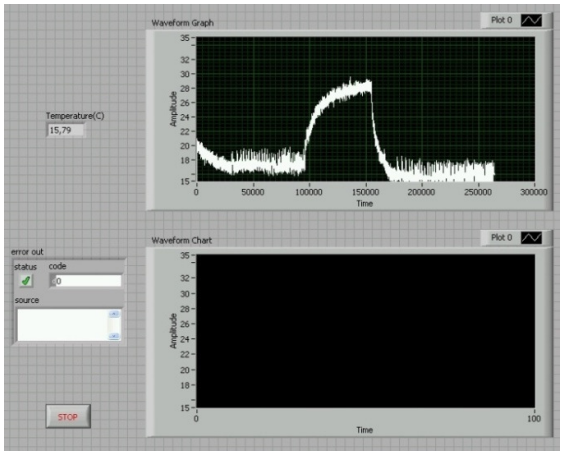

Fig. 6. (Networked) RT host Front Panel

On Fig. 7. We can see the Windows GUI. It has more functions like the sample interval dial and a stop button that will stop only this windows interface, not the RT program. It indicates the temperature (on an indicator and a Waveform Graph) received from the RT program.

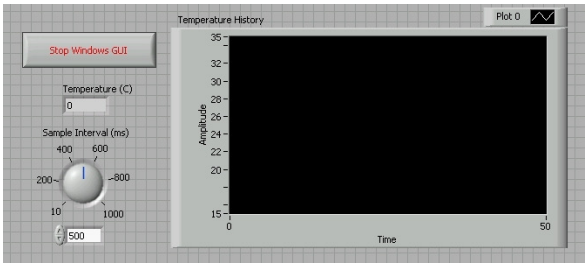

Fig. 7. Windows GUI Front Panel

D. The Block Diagrams of the Programs in LabVIEW

In this section we will show what hides in the back of each Front Panel described in the C. section.

On Fig. 8. we can see the FPGA program's Block Diagram. It's quite simple, it has a Stacked Sequence in a While loop with the clock settings and the AI0 FPGA Node. The AI0 FPGA node is the channel with number 0 from the NI 9201 C Series Analog Input Module.

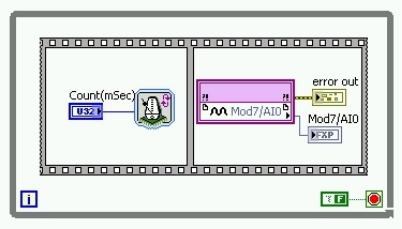

Fig. 8. FPGA program Block Diagram

On Fig. 9. we can see the RT host Block Diagram. The RT host represents the host of the FPGA program (Fig. 8.) running on the real-time operating system. The FPGA program is loaded in the "FPGA Target RIO0" icon. The FPGA program is reduced, because of saving resources, it reads only raw data. The signal conditioning and the graphical displaying is made on the real-time host. The RT host is software based (runs on an operating system) and the FPGA program is hardware based (reconfigurable I/O with FPGA), so this is the explanation why we need to do the FPGA program as simple as possible.

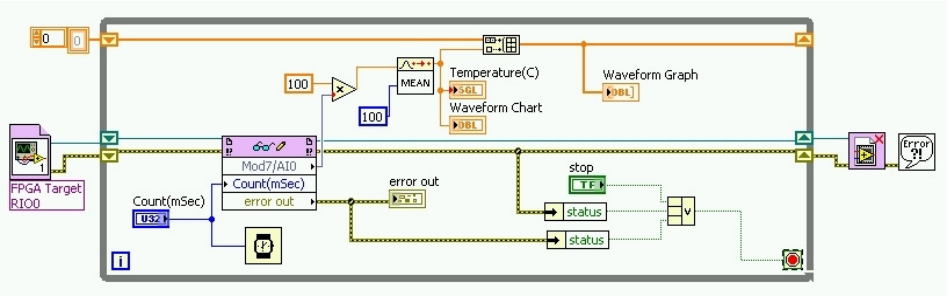

Fig. 9. RT host Block Diagram

On Fig. 10. we can see the networked RT host Block Diagram. The networked RT host is a copy of the real-time program (Fig. 9.), but the global variables are added (the variables which will send de data trough Ethernet).

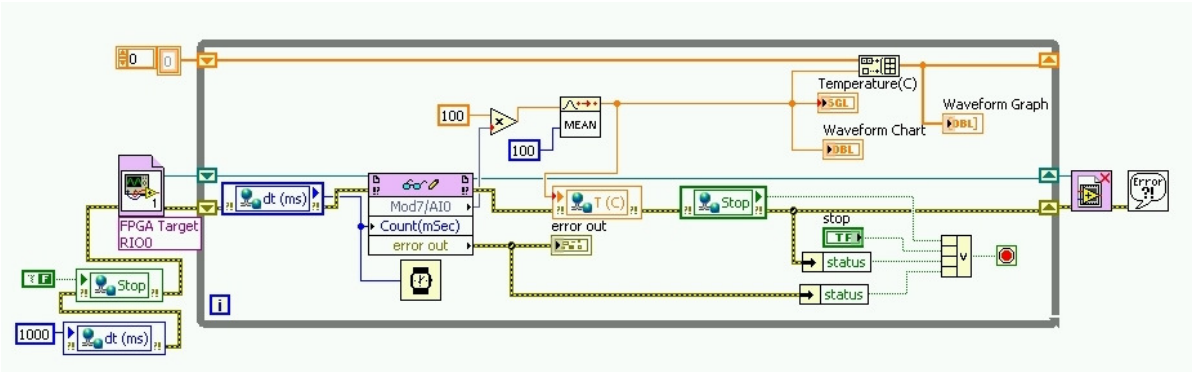

Fig. 10. Networked RT host Block Diagram

On Fig. 11. We can see the Windows GUI Block Diagram. The Windows GUI is the mirror image of the networked real-time program (Fig. 10.). the global variables are the colored squares with a little icon with two world globes connected. If you observe on Fig. 10. and Fig. 11., every pair of global variables (the global variables with same name, but in a different program) has it's arrow on the opposite side. If the arrow is on the left side, then it means that the variable is written, and if it's on the right side, then the variable it's read. You can imagine an invisible connection between every pare of global variables, that are the values that are sent on the Ethernet. The traffic on the Ethernet is between the pairs of the global variables.

If you want to run the programs, then it's enough for you to run only the RT host program (Fig. 9.), because the FPGA program is loaded in it. If you want a full professional implementation, then you will need to run the networked RT host (this one has the FPGA loaded too) (Fig. 10.) and the Windows GUI (Fig. 11.) in parallel, because the programs are communicating trough Ethernet. Professional implementation means if you want to allow modifying access to the Windows GUI users only for certain values, variables. Other professional implementation should be that if you want to upload the program to a web server. It's true that the real-time controller has a built-in web server, but it uses many resources and it's not safe to connect the controller directly to the internet. It's much safer to create the web server on the Windows computer. The Windows computer has more memory and a better CPU. It has also a better protection from hackers. If something happens, then the Windows operating system it's much easier to reinstall then the real-time operating system.

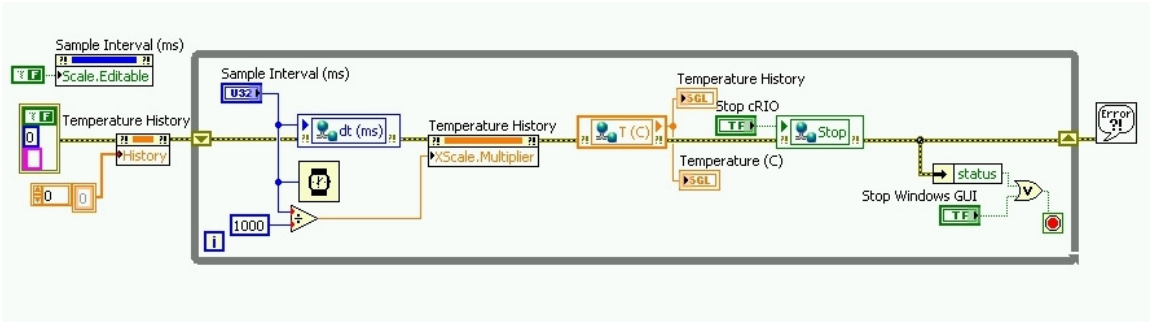

Fig. 11. Windows GUI Block Diagram

V. CONCLUSION

These experiments were made to compare the three methods of temperature measuring: the no FPGA method (cheap and expensive) and the method using FPGA and a real-time controller.

The significance of these experiments are that we can always know the temperatures from the server room, this is important when you have 9 computers running and a lot of equipment in a four by four room.

The no FPGA methods are simple and cheap, the FPGA with real-time controller method is more reliable.

REFERENCES

[1] Mihela Lascu, *Advanced programming techniques in LabVIEW*, Politehnica Publishing House, Timişoara, 2007.

[2] * * * National Instruments, *CompactRIO™ and LabVIEW™ Development Fundamentals – Course Manual*, 2008.

[3] * * * National Instruments, *LabVIEW™ Data Acquisition and Signal Conditioning Exercises*, 2008.

[4] * * * National Instruments, *Data Acquisition and Signal Conditioning – Course Manual*, 2008.

[5] * * * National Instruments, *LabVIEW™ Basics II: Development – Course Manual*, 2007.

[6] * * * National Instruments, *LabVIEW™ Basics II: Introduction – Course Manual*, 2007.