Microcontroller-based network for meteorological sensing and weather forecast calculations

Á. Vas*, Á. Fazekas*, B. Lehotai**, G. Nagy** and L. Tóth*

* University of Debrecen, Faculty of Informatics, Department of Informatics Systems and Networks, H-4032, Debrecen, Egyetem tér 1, Hungary, e-mail: vas.adam@inbox.com, adam.j.fazekas@gmail.com, toth.laszlo@inf.unideb.hu

** National Instruments Hungary Kft, H-4031, Debrecen, Határ út 1/A, Hungary, e-mail: balazs.lehotai@gmail.com, gabor3.nagy@ni.com

***Abstract*— Weather forecasting needs a lot of computing power. It is generally accomplished by using supercomputers which are expensive to rent and to maintain. In addition, weather services also have to maintain radars, balloons and pay for worldwide weather data measured by stations and satellites.**

Weather forecasting computations usually consist of solving differential equations based on the measured parameters. To do that, the computer uses the data of close and distant neighbor points. Accordingly, if small-sized weather stations, which are capable of making measurements, calculations and communication, are connected through the Internet, then they can be used to run weather forecasting calculations like a supercomputer does. It doesn't need any central server to achieve this, because this network operates as a distributed system.

We chose Microchip's PIC18 microcontroller (µC) platform in the implementation of the hardware, and the embedded software uses the TCP/IP Stack v5.41 provided by Microchip.

I. Introduction

The improvement of meteorology is strongly correlated to the development of science and technology like electronics, where only few better examples exist than that of numerical weather forecasting.

The development of thermodynamics in the 19th century resulted in a completion of the set of fundamental physical principles. It was recognized that meteorology is essentially the application of hydro- and thermodynamics to the atmosphere that requires the measurement of seven basic variables (pressure, temperature, density, humidity and the vector of velocity) and the solution of seven independent equations (three dimensional hydrodynamic equations of motion, the continuity equation, the equation of state and the equations of laws of thermodynamics). Therefore, the weather forecasts are made by collecting data about the current state of the atmosphere and using models to predict how the atmosphere will evolve [1,2].

The effective data collection was not possible until the invention of electric telegraph in 1835 but by the next decade, the telegraph allowed reports of weather conditions collecting from a wide area [1].

That time the way of making a forecast was based only on visual pattern recognition. This method required remembering a previous similar weather event which was expected to be similar to an upcoming event. However, this technique was difficult to use since there is rarely a perfect analog for an event in the future [1].

A great progress was made then in the science of meteorology during the middle of 20th century. The complexity of numerical weather prediction models have increased dramatically when the earliest modeling work was done by John von Neumann and Jule Charney likewise when the first electronic computer appeared. As the computing power grew, so did the speed capacity and complexity of the models. And as new observations became available from weather balloons, radar systems and satellites the forecast methods received more sophisticated initial data [1].

As mentioned above, weather forecasting methods are based on complex physical and mathematical models, which are very compute-intensive to calculate and consist of solving differential equations numerically based on the measured parameters. This task is typically done by supercomputers, but to maintain such systems is quite costly (considering not only the price but energy consumption, cooling etc.), and this is true for buying computing time, too.

Our idea is to construct a network, in which every node communicates with each other and does calculations based on its near and far neighbors' data. The nodes are connected to each other through the Internet. This network can be used to approximate solutions of differential equations, without using a central computer. If one uses more nodes, the calculations become more accurate as the field becomes better covered. However, the computing power required by one node does not change, because they use their neighbors' data only.

Our work for developing an autonomous meteorological sensor network that can also function as a forecasting supercomputer has started three years ago [3,4] and here we present our first results.

II. System Description

Our present system consists of three main parts. The µC based hardware with temperature, humidity and pressure sensors, as well as Ethernet connector and power supply circuits. The software on the microcontroller is

responsible for the system and measurement control along with making the necessary calculations and controlling the network connections. Finally, since at the present stage we are working with only three test panels a physical and software simulated test environment was necessary to be built.

A. *The Hardware of the Sensor and Computation Network*

Regarding the available development experiences and tools we have within doors and also considering the goal of the project the hardware (Fig. 1) has been designed on the basis of the following features:

- Low power Ethernet interface for networking
- Integrated sensors providing easy access to measurement results
- Minimize power supply and digital noise to get accurate results
- Sufficient computation power to solve planned differential equations
- Low voltage (and current) system to provide safe laboratory usage
- Low layer count of PCB to ease debugging and any necessary modification
- Preferably an 8-bit µC based
- Possible migration with SW reusability
- Low system and development cost

The Ethernet based network capability is an essential part of the system as it is the base of the geographical distribution. As the Ethernet communication is widely used in commercial and industrial applications, several Integrated Circuit (IC) manufacturers provide a proven physical layer device (PHY), media access control protocol (MAC) or PHY + MAC in one package. These solutions are targeting the best performance that can be achieved with the chosen network media and for that usually providing a (8/)16/32Bit parallel interface with a maximum flexibility to interface with industry standard µCs/µProcessors. The recently available µCs developed for network enabled embedded systems often include MAC and provide standard MII/RMII/GRMII... interfaces to the PHY. With this integration the system can use a low pin count package for µC and PHY as well which results a lower system cost. This integration still provides a maximum flexibility on MAC to µC (direct internal bus connection) and MAC to PHY for using any network media with choosing the sufficient PHY only. Several semiconductor companies stepped further and integrated the whole Ethernet chain into a chip, providing a maximum integration (single chip solution) for the most widely used media (10Base-T, 100Base-TX). Microchip is providing 10Base-T and 100Base-TX solutions integrating the MAC+PHY and µC+MAC+PHY in small footprint circuits for targeting low-cost Ethernet enabled embedded systems. As the team already had some experience with Microchip's µCs we chose the µC+MAC+PHY part for the central of our experiment.

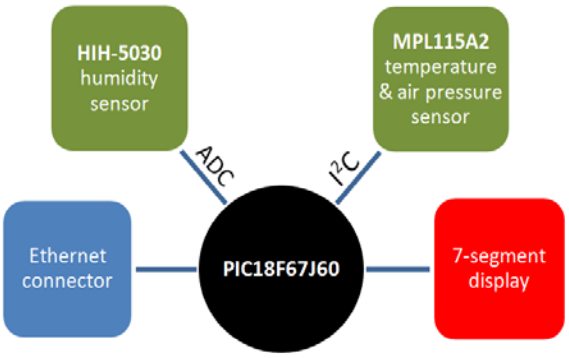

Figure 1. The block diagram of the hardware of our first experimental weather station.

B. *The Software of the Microcontroller and the Simulated Test Environment*

Microchip's MPLAB X is a single integrated environment to develop code for embedded microcontrollers. It also includes an ANSI compliant C compiler called MPLAB C18, which comes with many essential libraries containing functions for many applications like I^2C, Timers and Analog-to-Digital Converters (ADC). The software that we have been developing uses the TCP/IP Stack library by Microchip. This library includes the implementation of many network protocols, e.g. UDP, TCP, IP or DHCP. Two important parts of our software are the TCP server and TCP client running on each board. These are responsible for the exchange of weather data (temperature, air pressure, humidity) with the neighbors. This exchange process is done once in every second. When the neighbors' data arrive, the station does calculations based on the measured parameters.

Until now two calculation methods have been implemented: temperature gradients and isotherms. The temperature gradient as a vector shows the direction where the temperature rises most quickly and its length shows how fast the temperature rises in that direction. In a two-dimensional Euclidean space it can be calculated by

$$grad\ T(x_0,y_0)=\left(\frac{\partial T(x_0,y_0)}{\partial x},\frac{\partial T(x_0,y_0)}{\partial y}\right), \quad (1)$$

where $T(x_0,y_0)$ is the measured temperature at point (x_0,y_0). The partial derivatives are approximated in our case with the first-order central difference method (2,3).

$$\frac{\partial T(x_0,y_0)}{\partial x}\approx\frac{T(x_0+\Delta x,y_0)-T(x_0-\Delta x,y_0)}{2\Delta x} \quad (2)$$

$$\frac{\partial T(x_0,y_0)}{\partial y}\approx\frac{T(x_0,y_0+\Delta y)-T(x_0,y_0-\Delta y)}{2\Delta y} \quad (3)$$

Now we use the temperature but later also the pressure and wind velocity data from the neighbors of the station to approximate the solution of the differential equation.

Isotherms are curves connecting points that have the same temperature. In our case each node calculates

isotherms with 5°C differences taking into account its eastern and southern neighbors. If there is a point between them whose temperature is a multiple of 5 (or one of them has that temperature), the node calculates its position with linear interpolation and reports it as a point on an isotherm curve.

To verify the operation of the microcontroller grid we have made a software environment that can monitor and visualize the measurements, calculations and communication between the nodes. At the present state the test system with the simulated test environment, written in Java language, gathers the temperature, pressure and humidity measurement data from the microcontrollers and displays a thermal map with isotherms and gradients. It also logs the measured and calculated data transmitted by the nodes to verify their calculations.

The software accepts incoming TCP connections from the nodes and renders coordinates to them in a predefined way. Each node placed in the thermal map according to the given coordinates. It was necessary to simulate nodes because of the small number of hardware stations. For that reason we made a TCP application on the PC. Multiple instances of that program are running, and they work the same way as the boards except that they cannot measure their own temperature, obviously.

The values in the thermal map could be assigned in two ways: a point in the thermal map is either a microcontroller unit with a temperature sensor or a virtual point with a calculated temperature. To calculate the temperature in the virtual points the first-order central difference approximation of the Heat Equation was used:

$$T(x_0,y_0)_{t_{i+1}} \approx \frac{\begin{pmatrix} T(x_0+\Delta x,y_0)_{t_i} \\ +T(x_0-\Delta x,y_0)_{t_i} \\ +T(x_0,y_0+\Delta y)_{t_i} \\ +T(x_0,y_0-\Delta y)_{t_i} \end{pmatrix}}{4}, \qquad (4)$$

where $T(x_0,y_0)_{t_{i+1}}$ is the measured temperature at point (x_0,y_0) at time t_{i+1}.

The temperature and gradient values in the real nodes are assigned by the measured and computed data, which are transmitted over the network. The communication process between a node and the test environment is divided into two parts. In the first part the nodes send the measured data to the test software, and after that they send the results of the gradient and isotherm calculations. Using these data the Java application draws the vectors and isotherms (fitting a curve on the isotherm points received from the stations) on the thermal map. As we mentioned before, the nodes transmit data once in every second thus the thermal map always represents the momentary conditions in the grid.

III. Test Environment and Results

At the moment we're working with 3 stations connected to each other and to a PC through an Ethernet switch (Fig. 2). Each board is placed on a plastic can filled with water. The temperature of the water is different at each position. One is filled with hot (~60°C, red), the second is with warm water (~35°C, orange) while the third one with mixture of ice and water (~0°C, blue). This way heat propagation on the test field can be simulated efficiently. Also, due to the large temperature differences between the stations, the temperature gradient vectors become clearly visible. The stations are placed on a grid paper to make it easier to show their positions in the Java application (Figure 2), and they display the measured temperature in Celsius degrees on their 7-segment displays for debugging purposes.

Based on the measured and calculated data by the real and virtual stations the Java application can display the thermal map of the test field (Fig. 3), the gradient vectors of the real nodes (Fig. 4) and isotherms (Fig. 5).

According to the pictures, the calculation of the gradient vectors and the isotherms seems to be accurate: the gradients at nodes 2 and 3 point at the direction of node 1, the hottest point of the field; and there are many isotherms drawn around node 1, where the temperature changes most quickly.

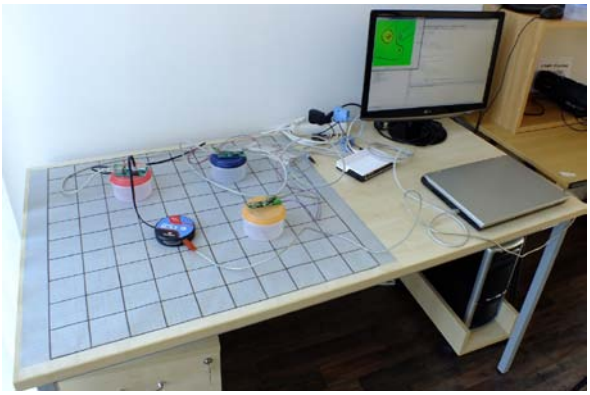

Figure 2. The test environment with the 3 weather stations and the Java application running on the PC.

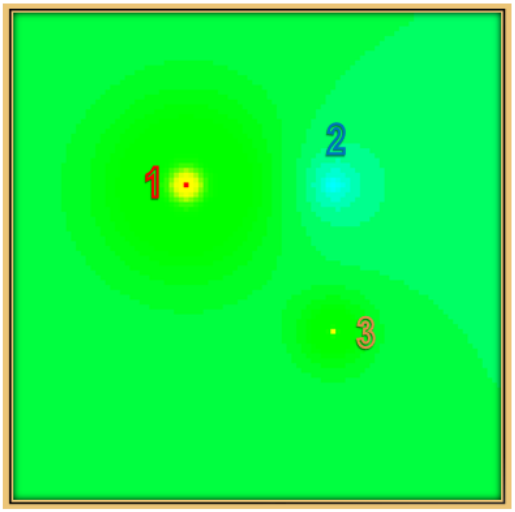

Figure 3. The thermal map of the test field.

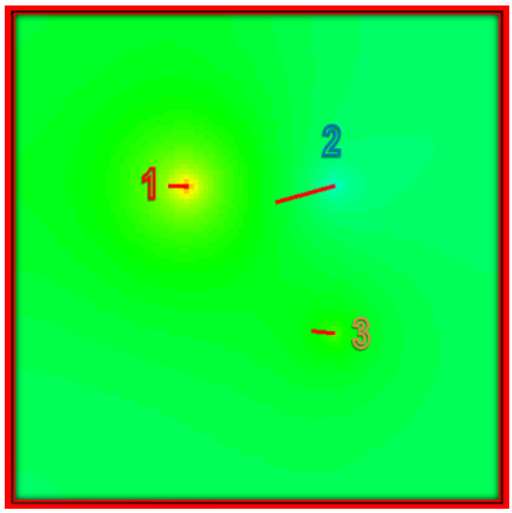

Figure 4. The temperature gradients displayed on the thermal map.

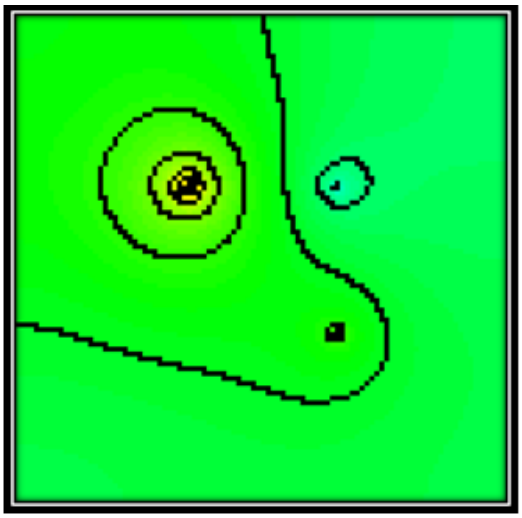

Figure 5. Isotherms displayed on the thermal map.

IV. SUMMARY

We succeeded to build a simple network which is capable of calculating isotherms and temperature gradients on a test field by distributed calculations. Although it consists of 3 stations now, we could understand the basic physical processes, test several different experimental configurations, the necessary basic instrumentation, algorithms and software environments as well as get sufficient experiences for further work towards using more stations and implementing further meteorological models on the boards.

ACKNOWLEDGEMENT

The authors wish to thank National Instruments Hungary Kft. for providing the possibility to use their facilities and SciTech Műszer Kft. for financially and technically supporting our work. This work is supported by the TÁMOP-4.2.2.C-11/1/KONV-2012-0001 project. The project has been supported by the European Union, co-financed by the European Social Fund.

REFERENCES

[1] P. Lynch, "The origins of computer weather prediction and climate modeling," *J. Computational Physics*, vol. 227, pp. 3431–3444, 2008.

[2] F. G. Shuman, "Weather prediction," U.S. Department of Commerce National Oceanic and Atmospheric Administration National Weather Service, Office Note 198, April 1979.

[3] B. Simon, G. Pásztor, "Mikrokontrolleren alapuló időjárás előrejelző hálózat," University of Debrecen, Faculty of Informatics, BSc thesis, supervisor: L. Tóth and G. Nagy, 2010.

[4] B. T. Kulcsár, D. Virág, "Mikrokontrolleren alapuló időjárás előrejelző hálózat II.," University of Debrecen, Faculty of Informatics, BSc thesis, supervisor: L. Tóth and G. Nagy, 2011.