Building a Smart Sensor using Windows CE

Ganesh Gopalan

COAS, Oregon State University

104 Ocean Administration Building

Corvallis OR-97333

Abstract: **In this paper we discuss how to develop a smart instrument i.e. one that has built in scientific logic that allows it to behave intelligently. The instrument will be capable of providing near-real time updates and respond directly to user requests for data. We demonstrate this technique using a Davis Instrument as the "sensor" that is connected to an SBC MediaGX processor board running Windows CE 3.0. The board supports 100 base-T Ethernet and TCP/IP. Communication with the Davis Instrument is through serial I/O. Requests for data are sent to the instrument over a serial interface. Specific tasks in the project included making hardware connections between the various components and software configuration on the device. A host PC is also set up for the development and transfer of embedded software using ActiveSync. The tools used include Embedded Visual C++ 3.0 and other remote tools for editing files and working with the registry. Software approaches explored to provide access to the data include ASP, DCOM and XML data sources. A server on the Windows CE device connects to the Davis Instrument and logs temperature, pressure and humidity values to a file. Data is also recorded as XML. The server also exposes a COM interface so that a remote DCOM client can connect to it and read the data. The frequency with which values are logged can be changed remotely. Plots of the data are also supported using MATLAB on the host PC. Pre-defined conditions that occur in the data can be detected using XML and XSL and the user can be alerted when they occur.**

I. Introduction

A key component to data access and event detection in Earth Science is how fast the data becomes available to client applications. As technology improves, it is becoming increasingly easy to get data directly from its source. In this paper we demonstrate how data can be accessed directly from an instrument in a near-real-time fashion using an embedded operating system. Remote control of the instrument is also demonstrated. We illustrate a solution that uses a Davis Instrument but the technique can easily be extended to other sensors.

II. Goals

The goals for this project are:

- To demonstrate how an instrument can be made accessible through the Web.
- To demonstrate how data from an instrument can be accessed in a near real-time manner.
- To provide a means of controlling the instrument from a remote location.

III. Project Description

We now discuss the selection of hardware, operating systems, software tools and architectures for the system. Hardware and software configuration are also covered. In this paper, “Windows CE device”, “embedded device”, “device”, “SBC MediaGX” all refer to the piece of hardware that is connected to the Davis Instrument unless otherwise specified.

A. Selecting Hardware

After looking at a variety of embedded hardware available, the SBC-Media GX by Arcom Control Systems was selected. The reasons for this choice are discussed below.

The SBC (Single Board Computer) Media-GX is an EBX compliant board based on the MMX-enhanced 233 MHz Geode processor. The kit contains a complete set of components to enable the rapid development of new Windows CE applications. The board has 32 MB DRAM and 16MB flash supported as standard memory with expansion up to 128 MB DRAM. Standard peripheral I/O ports supported by the SBC Media GX include dual USB connectors, 10/100 base-TX Ethernet, four fast serial ports, floppy disk drive & IDE interfaces. Standard PC type mouse and keyboard ports are supplied via PS/2 mini DIN connectors. The kit also included a flat panel display.

B. Selecting an Operating System

The development of this device started with choosing an appropriate embedded operating system. The focus was on two operating systems, Windows CE 3.0 and Windows NT

Embedded 4.0, since there existed a number of proven products developed using these two operating systems. Windows CE 3.0 was eventually selected since Windows NT Embedded 4.0 is not suitable for battery-powered devices and its memory requirements were greater than that of CE. Windows NT Embedded 4.0 required 12 MB of RAM and 8 MB of persistent storage while Windows CE 3.0 required a minimum of 450 KB of RAM and 350 KB for storage.

We now give a brief overview of Windows CE 3.0. Windows CE is an open, scalable, 32-bit operating system that is designed to meet the needs of a broad range of intelligent devices [1]. It provides a powerful set of features and tools including:

- A highly modular architecture to rapidly configure an operating system from over 200 pre-built and extensible modules.
- Real time support.
- Flash memory support for diskless operation.
- Supports advanced application services like DCOM, ADO and MSMQ.
- Rich Internet services with HTTP 1.0 server for serving HTML, ideal for simplified remote device management.
- Supports latest communication and networking technologies.

C. Setting up the Hardware

Fig. 1 shows a picture of the Windows CE device. Refer to Fig. 2 for the hardware configuration. The system supports PC style peripherals such as a keyboard and mouse for input. A flat panel display is also connected to the board. A host PC is connected to the system using a serial interface. The PC is used to develop embedded programs and transfer them onto the device.

The Davis Instrument is also connected to the system by a serial cable. Commands and data are sent over this cable. The device also has an on- board Ethernet connector that is used to plug the device into the Local Area Network.

D. Configuring the System

The SBC MediaGX processor board comes pre-installed with Windows CE 3.0. There are a few things that need to be done on the device before embedded programs can be deployed on the system. These include the following:

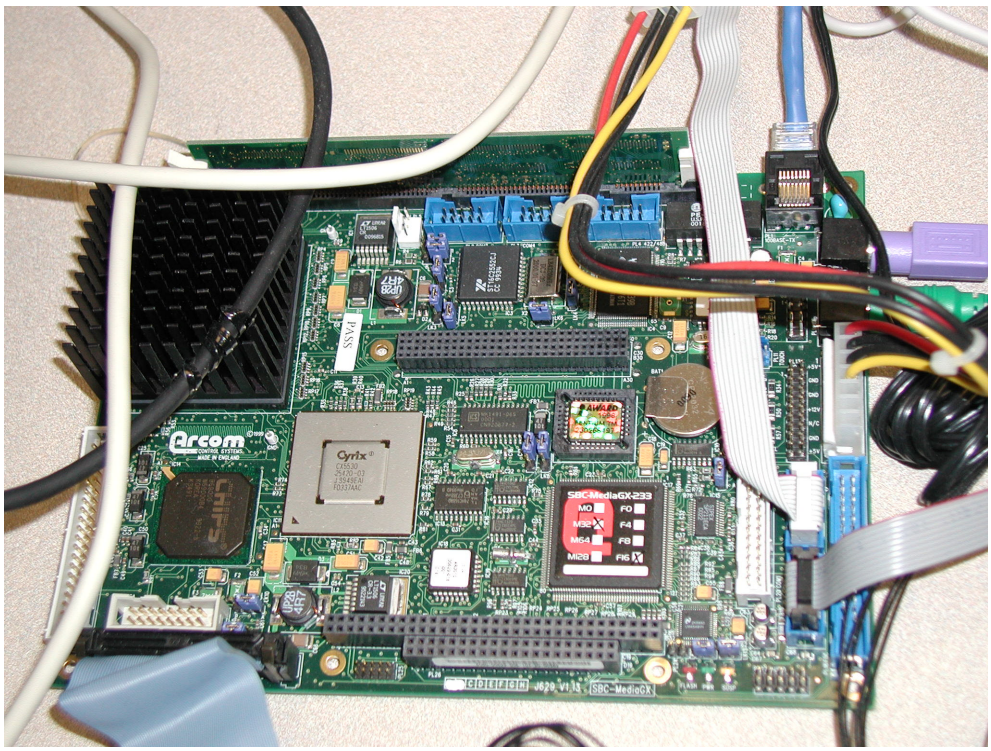


Fig. 1. The SBC MediaGX board with hardware connections.

- Saving the registry - this is done when system parameters, such as HTTP settings, are changed.
- Setting up a COM port – a COM port must be configured to make a connection to the host PC.
- Setting up an ActiveSync connection to the host PC – ActiveSync facilitates the synchronization of data between a desktop and a companion application running on the Windows CE device [2]. Once a partnership is established with the host machine, files and/or data can be exchanged between the two systems using this connection.
- Ethernet connection – network parameters such as IP addresses are configured here.

E. Setting up the Host PC

The host PC is the system where all software development is done including writing and compiling embedded programs. Once an ActiveSync connection is established with the host PC, data and files can be moved between the host PC and the system using the connection.

Software tools used in embedded development such as Windows CE Platform Builder 3.0 and Microsoft Embedded Visual Tools are installed on this machine. The latter includes Embedded Visual C++ 3.0, Remote File Viewer, Remote Process Viewer, Remote Registry Editor and Remote Heap Walker. One or more of these tools maybe used during the software development process.

F. Programming Environments

Both Embedded Visual Basic and Embedded Visual C++ can be used for building embedded software. However, based on the type of software components that were required, Embedded Visual C++ was the primary programming environment. We do not use Platform Builder to generate and upload an image of the operating system onto the device since the hardware comes pre-installed with the O/S. However, remote tools accessible from within Platform Builder are used when required.

G. Developing Embedded Projects

Embedded Visual C++ 3.0 is used to build executables or dynamic link libraries for the Windows CE device. First a Windows CE platform is selected from available types (such as Palm-size PC 2.11 or Pocket PC 3.0). Next, the programming environment needs to be configured with the appropriate paths for header and library files used when compiling and linking code. These files have to be installed from the Arcom installation CD. Project Settings have to be created for each project as appropriate. These include Preprocessor and Linker Settings that may point to the MediaGX header files/libraries and any flags specific to the Windows CE architecture. The project is then compiled. The resulting binary can be tested using an emulator if required. Finally the binary is transferred to the device using Remote File Viewer for further testing and deployment.

H. Software Architectures

The goal here was to explore different methods of accessing data from a file on the device before actually logging and retrieving readings from the Davis Instrument. We tried to take advantage of the fact that the Windows CE device supports a web server.

The methods explored were:

1. Active Server Pages (ASP)
2. Internet Server API (ISAPI)
3. Distributed COM (DCOM)

Using ASP:
A built-in Visual Basic COM object created using ASP was used to access the file system on the device. This method seemed to fail because of security reasons and the Windows CE documentation did not help resolve this.

Using ISAPI:
This was similar to ASP except that the piece of code that accessed the file system was written in C++ and loaded in to the address space of the web-server as a dynamically linked library (DLL). This method did not work either because the DLL would not load due to security reasons.

Using DCOM:
This technique seemed to work the best. Here, the PC client launches a DCOM object that accesses the file system on the device. The contents of the file are returned as a method argument.

Eventually, two approaches were implemented. One was to use a DCOM client on the host PC that connects to a DCOM server running on the device to get the data while the second approach was to read the data directly from the device in the form of XML. The DCOM client is in the form of an ISAPI DLL so that it can be accessed from the web.

I. Logging Data from the Instrument using serial I/O.

In order for data to be logged to the device, serial communications with the Davis Instrument must be established. Commands to read data or change settings on the instrument must be sent using the Windows CE serial API.

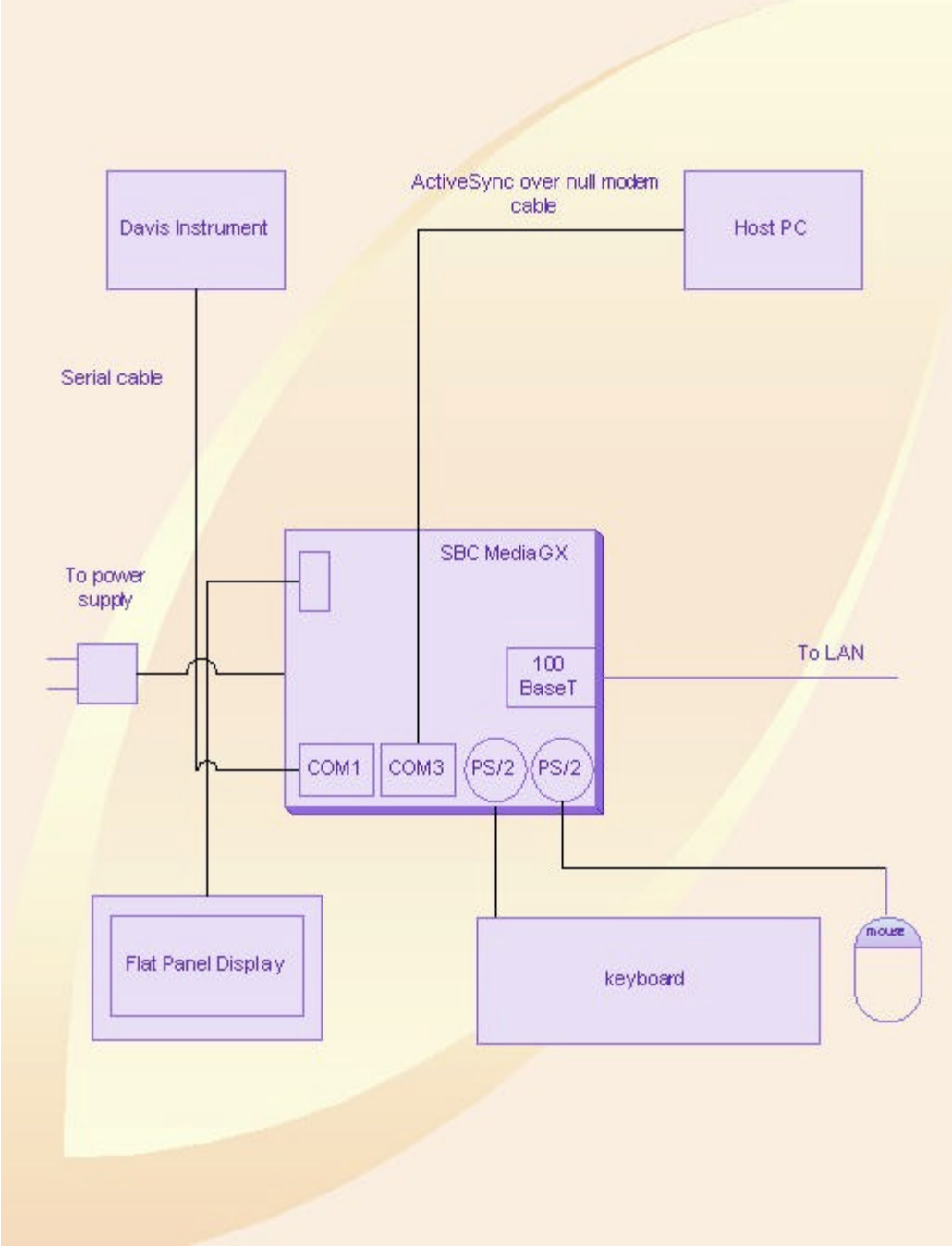


Fig. 2. Hardware Configuration.

Temperature, pressure and humidity are read using the serial I/O functions.

A separate thread is responsible for communications with the Davis Instrument. The purpose of creating a thread is to make use of as much of the CPU's time as possible [3]. This thread sends bytes across the serial cable, waits for acknowledgments and reads the results of commands. Specific bytes are sent for specific tasks e.g. the byte sequence 'W', 'R', 'D', '0x00', '0x00', '0x44', '0x52', '0x0d' followed by a '0x44', '0x30', '0x0d' tells the Davis Instrument to send the current temperature. The main thread receives requests from DCOM clients and processes them.

Fig. 3. The SBC MediaGX board with the flat-panel display, keyboard and mouse. The flat-panel display is the one with the Arcom logo. The Davis Instrument is to the left of the display.

J. Applications and User Interface

Applications that display the latest data on a browser have been developed. Plots of the retrieved data are also generated using MATLAB (see figures 5 and 6).

The DCOM client that is built into the ISAPI DLL on the PC, displays the data and also accesses MATLAB through OLE automation to generate plots of the data.

A second application displays data from an XML data source that is accessed directly from the device, using HTTP. A plot is generated as mentioned earlier using MATLAB.

Events that the user maybe interested in, such as high/low temperatures, can be specified using an XSL style-sheet. Alerts can be sent automatically using the style-sheet.

IV. Conclusion

Although several of the technologies available on the PC platform such as ASP, ISAPI and COM are supported on the CE platform, we found that these technologies are not readily usable. However, this was a good start in working with embedded operating systems and applications.

V. Future Work

Future work includes the use of new embedded operating systems such as Windows CE .NET or Windows XP Embedded on the device. There is also the possibility of using an actual drifter as the measuring instrument and demonstrating network connectivity using wireless protocols.

Glossary

ADO – Active Data Objects, a technology used for database access.

COM – Microsoft's Component Object Model which is a standard means of communication between objects.

DCOM – Distributed Component Object Model. A version of COM where objects can be instantiated on remote machines.

EBX – A form factor that is about 5.5"x8".

ISAPI – Internet Server Application Programming Interface, an architecture for the development of interactive web-based applications.

MSMQ – Microsoft Message Queue, a technology used in asynchronous messaging applications.

OLE Automation – A technique where COM objects can be accessed from a scripting environment such as a web-browser.

SBC – Single Board Computer, a form factor that's used in developing embedded devices.

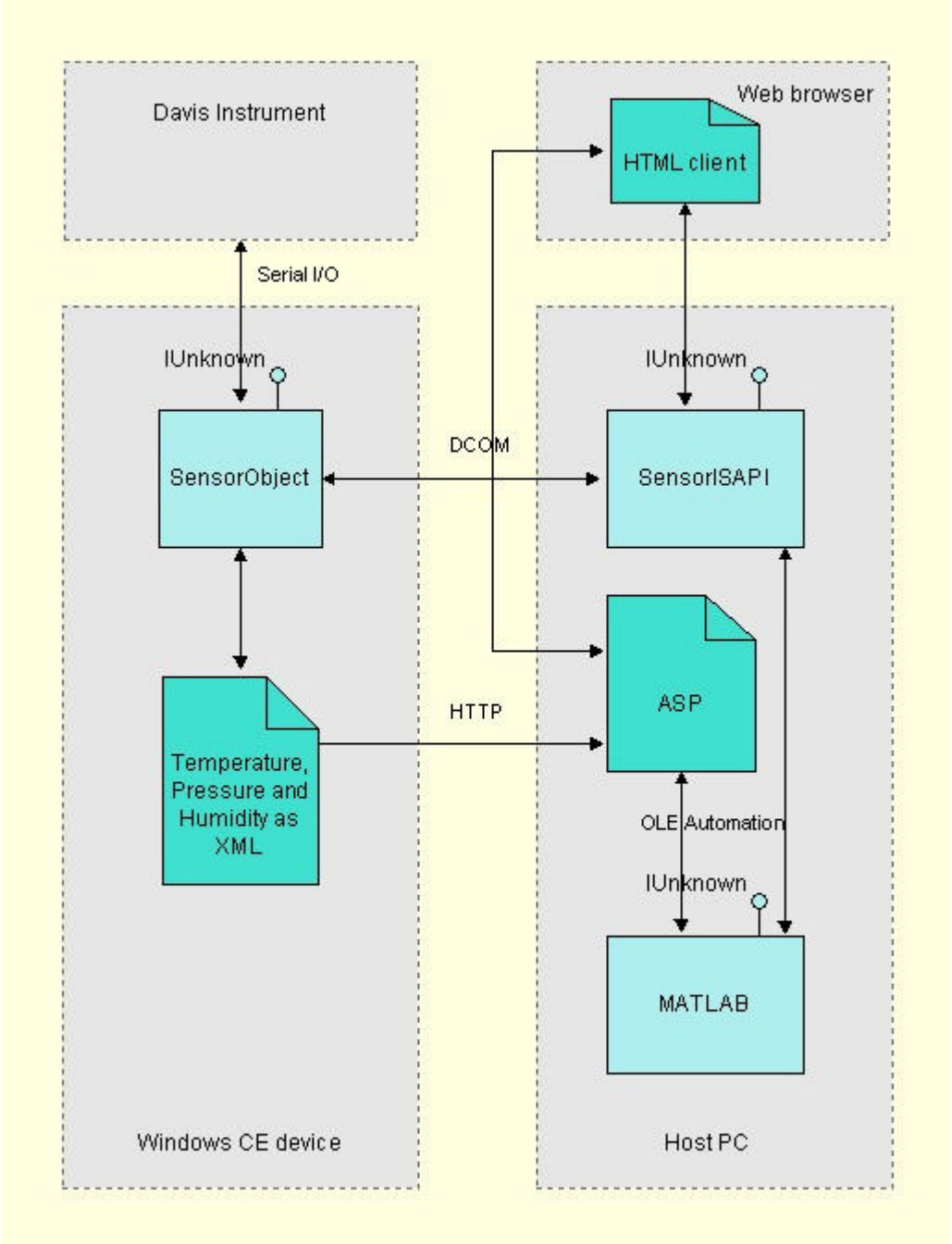


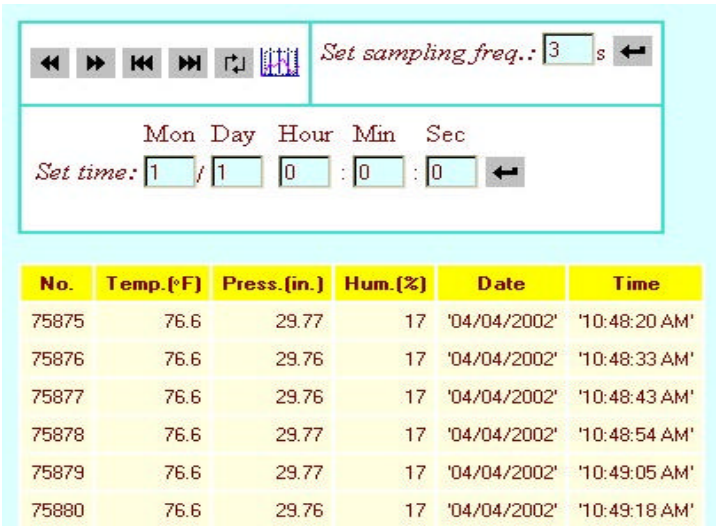
Fig. 4. Software architecture for the smart sensor - "SensorObject" is the DCOM server and "SensorISAPI" the DCOM client. The DCOM server is also responsible for logging the data from the Davis Instrument. Plots are generated using MATLAB and OLE Automation on the host PC.

Windows CE – One of Microsoft's embedded operating systems.

XSL – Extensible style-sheet language; used to describe the layout and format of XML data.

REFERENCES

[1] *Introducing Windows CE 3.0*, http://msdn.microsoft.com/library, Microsoft Corporation, 2001.



No.	Temp.(°F)	Press.(in.)	Hum.(%)	Date	Time
75875	76.6	29.77	17	'04/04/2002'	'10:48:20 AM'
75876	76.6	29.76	17	'04/04/2002'	'10:48:33 AM'
75877	76.6	29.76	17	'04/04/2002'	'10:48:43 AM'
75878	76.6	29.77	17	'04/04/2002'	'10:48:54 AM'
75879	76.6	29.77	17	'04/04/2002'	'10:49:05 AM'
75880	76.6	29.76	17	'04/04/2002'	'10:49:18 AM'

Fig. 5. The user interface to the SmartSensor application. The table shows logged data from the Davis Instrument. The sampling frequency can be changed from here. The time on the instrument can also be set from this interface.

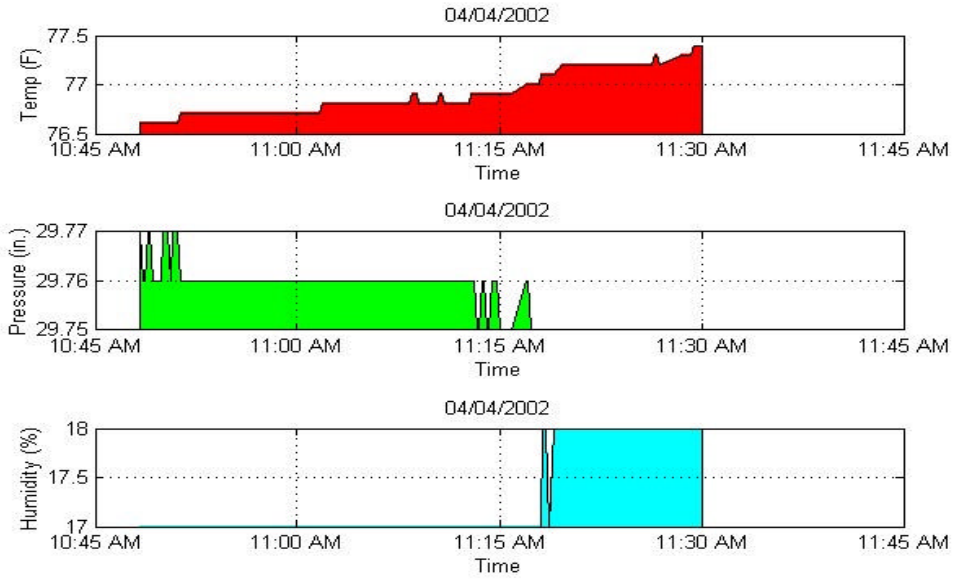


Fig. 6. A plot of temperature, pressure and humidity values from the Davis Instrument. This was generated using MATLAB.

[2] N. Grattan, M. Brain, *Windows CE 3.0 Application Programming.* Prentice Hall Inc., 2001.

[3] *Microsoft Windows CE 3.0 Kernel Services: Multiprocessing and Thread Handling*, http://msdn.microsoft.com/library, Microsoft Corporation, 2001