REMOTE OPERATION OF AN AXIAL TURBOFAN WINDTUNNEL VIA THE WORLD WIDE WEB

Robert M. McFarlane, B.S.M.E., James D. McBrayer, D. Sc., P.E., Professor Emeritus

University of Central Florida

Introduction - The Engineering Laboratory and Engineering Education

Contemporary engineering training requires learning on two parallel tracks. One is learning the basic science and theoretical foundations of science, in subjects such as physics, chemistry, thermodynamics, and mechanics. The other track is the practical application of this theoretical groundwork to the investigation and solution of actual problems. The place that these two aspects of the syllabus come together is in the engineering laboratory. It is here that the student is presented with a concrete problem to be solved, and they learn to apply the lessons of the classroom. The larger goal is to prepare the graduate to enter today's engineering profession with not only the proper problem-solving strategies and skills, but also to have gained experience that can be directly transferred to the needs of the contemporary career. This means exposure to the latest technologies and an understanding of their concepts and applications.

Perhaps the most significant aspect of engineering practice in the last decades has been the impact of computer technology in practically every field, from basic research, to applied research and product design and development. From the first implementation of electronic calculating machines, the engineering discipline has been one of the most heavy users and developers of computing technology. The ability to use this technology will determine the effectiveness of the engineering student as they go to industry, perhaps only second in importance to understanding the basic fundamentals of the chosen field of study. Thus, one of the goals of any successful curriculum will be to include training and experience in the use of computing hardware and software. The engineering laboratory is an excellent opportunity for this purpose.

One of the important aspects of using laboratory experiments is the fact that not all phenomenon act entirely as the theoretical models may predict. This is caused by a host of factors, such as wiring effects, signal noise, hysteresis, calibration effects, and second-order effects. These may fall under the categories of uncertainty and error in measurements, which are fundamental concepts in engineering in general. Learning how to deal with these effects, to

identify, correct or finally accept and quantify these effects is an integral part of the engineering education[1]. Many computer simulations of experimental results do not or can not take these factors into account in a meaningful way. Therefore, working with a physical apparatus has many advantages over simulations, but has a disadvantage in that the student must gain access to the lab, and scheduling access to scarce resources may limit the use of those lab resources. Being able to allow distributed access to a single apparatus during a certain time window can leverage the benefit of that apparatus, and may allow sharing between campuses or schools as well as between individual students.

There are many examples of the use of the Internet to leverage the use of expensive or unique engineering laboratories around the world. Collaboration has always been an important part of research, especially in an era of increasing specialization and dispersal of resources. Scientists in quantum physics and fusion research have been dealing with the problem of extremely expensive plant and experiment costs for many years[2]. The recent developments in networking technology and infrastructure have proven to be of great use in solving this problem. Examples include the MIT Plasma Science and Fusion Center, the Oak Ridge National Laboratory[3], the DIII-D National Fusion Facility[4], the Joint European Torus (JET)[5], and the Large Helical Device (LHD)[6] in Japan. In addition, the Department of energy has made the development of collaborative techniques a priority in its present and future programs[3].

The Oak Ridge National Laboratory (ORNL) has developed a software solution for monitoring and controlling a harsh environment test bed over the Internet[3]. A server computer, data acquisition hardware and LabVIEW (Laboratory Virtual Engineering Workbench) software make up the data acquisition and control system (DACS). LabVIEW is a virtual instrument software from National Instruments, which is also used in the experiment that this paper documents.

Description of Problem

The intent of this project is to research, design and develop the systems required to make the Technovate Scott Turbofan System, Model 9005 (TS9005) experiment accessible to engineering students from a remote location using the World Wide Web. The TS9005 is a multi-mode, interdisciplinary apparatus that allows students to explore the relationship between applied power and the fluid dynamics of air. It is essentially a benchtop turbofan windtunnel that is powered by a dc motor. In order to accomplish this goal, there are four major areas that must be addressed.

First is the conversion of the TS 9005 apparatus from a manually controlled system to one that can be electronically controlled. This means all the input experimental parameters have to be electronically controlled. These are usually the voltage and current from the dc power source, but in some configurations, the mechanical power or pressure readings may be used in control feedback loops. Second the data from the experimental runs must be digitally captured by installation of DAQ hardware. These are primarily the pressure taps and transducers which measure the pressure of the air at various locations in the tube, but also the voltage and amperage are measurands when the air stream is forced into the tunnel from an outside source, running the motor as a generator or dynamometer. Third, the Virtual Instruments necessary to control the

input parameters and process the data are designed. These are computer programs that control the inputs and record the outputs, including a GUI interface for the operator. The combination of the input control, data acquisition, and processing hardware and software is called the Data Acquisition and Control System (DACS). Finally, the entire system is connected to the Internet and given a URL address for students to access. Once the remote terminal has gained access through the Internet, this experiment can be controlled from the GUI on the client computer, and the operator has the option to handle the data in several ways, in the same manner as if they were present in the lab.

Description of Solution

The solution to this problem will have to meet several requirements. Most importantly, we want the users of the system to have an experience which will be as close as possible to being in the lab and using the equipment.

- It is desirable that the student exercise there own judgment in the design of the experiment to as great a degree as is possible, or at least have some input into the manner in which the data will be acquired and processed.
- Since their interface will be through a computer screen, it should have a graphical user interface (GUI) that has the appearance of an engineering instrument. This part of the system must be simulated, of course, and is called a "Virtual Instrument", or VI.
- We want the GUI to be self-contained, that is, each experiment requires only one screen presentation, without the need for extensive look-ups or successive page loading.
- We would like the student to be able to control the apparatus to as great a degree as is possible, within the limits of lab safety.
- The GUI should have a dynamic nature during the running of the experiment itself, indicating the collection of data.
- The data must be available to the user for download at the remote location.
- Peripheral devices that add to the perceptive experience can enhance the learning experience, so including video and auditory presentation of the device in action would be an important enhancement.

Use of LabVIEW as Solution

The characteristics that made LabVIEW the choice of the developers at ORNL are also the reason it has been incorporated into several University laboratories. One of its main attractions is its proprietary high-level programming language, called "G". LabVIEW is in fact a true 32-bit compiler, and as such can create stand-alone executables. G is a graphical language, using icons to represent code statements, functions, and modules. These are combined in a GUI environment and linked together in a "data-flow" sequence, rather than the procedural style of many text based languages. This style of programming is more intuitive than text-based styles, as the flow of the program follows the "flow" of the data through the processing stages. This high level treatment allows for modularity and development of libraries of simple, low level VIs, which can be combined to form more complex VIs. This frees the student from the rigors and challenges of syntax and allows greater concentration on solving the engineering task at hand.

Apparatus Hardware

The Turbofan System, Model 9005 (TS9005) is a bench-top sized, interdisciplinary learning system centered on the fluid dynamics of air, utilizing the Scott Turbo-fan system. It permits students to conduct experiments on a variety of phenomena such as laminar versus turbulent flow, pressure and velocity relationships, turbine blade properties, electric power generation, behavior of aerodynamic shapes, nozzle behavior and flowmeter and manometer usage. The basic components are:

1) An air turbine mounted in a clear cylindrical tube 5.125" in diameter.
2) The turbine is driven by a ¼ hp electric motor, which has a d-c power supply, providing speeds up to 35 mph.
3) Air-drive venturis, exit venturi and exit nozzle, air straightener and orifices.
4) Instrumentation includes pressure gauges and manometers, Pitot static tube, torque load cell, air velocity meter, air taps for pressure measurements, and stroboscopic disc for measuring shaft rpm.

The system can be operated in four major modes:

1) Turbine rotor is driven electrically to provide airflow for wind tunnel experiments.
2) Turbine rotor is driven in reverse electrically to permit variety of electric motor experiments coupled with reverse flow wind tunnel experiments.
3) Turbine rotor is driven in reverse against the electric drive by a high velocity outside air source; the motor acts as dynamometer for various power measurements.
4) Turbine rotor is driven by high velocity outside air with electric power off; the motor acts as an electric generator.

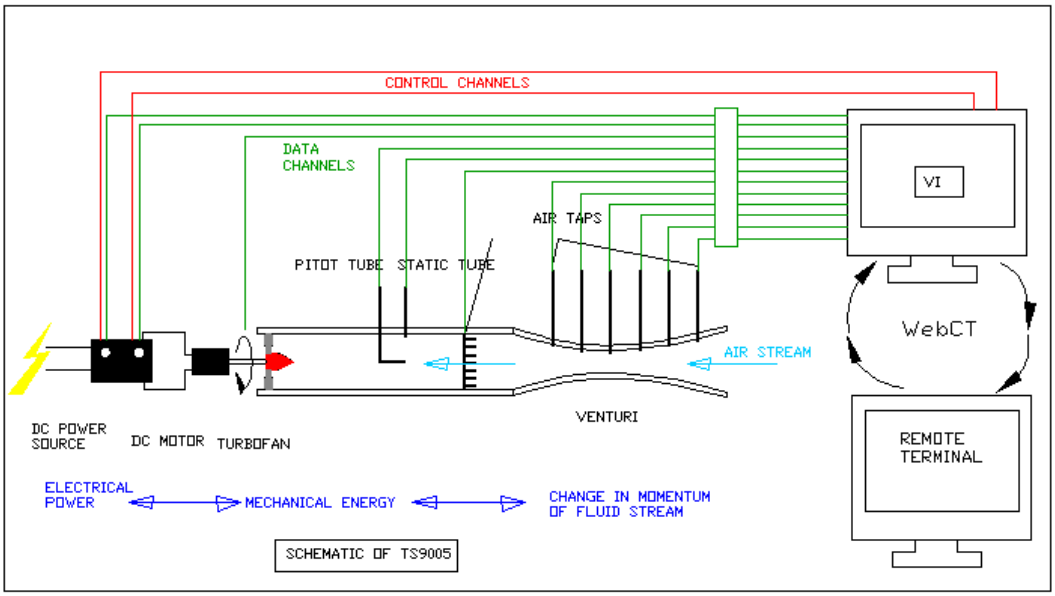

Figure 1 Schematic of the Scott axial turbofan windtunnel showing DACS

Because of the many configurations of the air channel with venturi, nozzle or orifice and the major modes of operation, the system has a wide range of applications. In any given configuration, experimental parameters may be selected as either input or output. The requirement for a remote system means that only the aspects of the device that are electronically

operated can be controlled. For example, orifices, nozzles or student-designed airfoils must be physically mounted in the laboratory. Experiments requiring such on-site assistance require scheduling and remote conferencing capabilities. Once the apparatus is configured for the run, the parameters can be grouped into three main areas. These are electric power, rotational speed of the turbofan, and pressure measurements.

The apparatus had to be converted from manual control to digital control as the first step in implementing the remote system. The primary components in this conversion were a DC motor control board from KB Electronics, a Digital Acquisition System (DAQ) control board , from National Instruments (NI), which is mounted in the host computer, Digital to Analog and Analog to Digital converters from National Instruments , and the program to control the hardware and apparatus, which is written in LabVIEW 6.0i.

Controls

The primary component to be controlled is the 1/4 hp motor that powers the rotating turbofan. This motor is rated at 24 VDC, but the manufacturer's placard on the original power supply states that the voltage should not exceed 18 VDC in operation of the turbine. The turbine is mounted directly to the shaft of the motor, so the turbine has the same rotational speed as the motor. The speed of a DC motor is proportional to the DC voltage applied across the motor circuit, but the maximum speed is determined by the motor and load combination.

In order to control a DC motor, the standard 110 VAC current available must first be transformed to a DC current. This current then must be controlled by a signal from the controlling computer. This is accomplished by the inclusion of a motor control electronic circuit board, which rectifies the AC current into DC, and regulates the DC voltage applied to the motor circuit. To fill this requirement, the model KBIC 120 (36 mod), designed and manufactured by KB Electronics, was used. This is a silicon rectifying circuit (SRC), combined with an analog voltage control which can be manually controlled by a rotary potentiometer, or set to voltage following mode, which will accept a 0 to 7 VDC analog signal from a controlling computer. The SRC circuit component converts the line AC voltage to a DC voltage, such that the DC voltage out is equal to the root-mean-square (RMS) VAC input. Since the rated maximum voltage for the motor is 24 VDC, the AC voltage entering the board must be maintained at 36VAC or less, in order to keep the RMS DC voltage from exceeding the maximum rated voltage for the motor. A commercially available Variac, or variable AC transformer, was used to provide AC voltage at 36VAC.

Data Acquisition

The voltage across the motor is being set by the user input and then is scaled by the KB control board. However, the actual voltage delivered across the motor may not be exactly equal to the desired value, so it must be measured close to the motor terminals to determine the true value. This is accomplished by connecting two signal wires at the wire connection to the motor, which then carry this voltage signal to the SCXI 1122 module, through the SCXI 1322 terminal.

The current drawn by the motor is measured indirectly by taking a voltage measurement across the shunt resistor in the KB control board. This resistor is in series with the motor load, so the current passing through both is the same. Since its value is known and constant, the current value can be calculated in the software program from Ohm's Law. This signal is carried to the SCXI 1122 module though the SCXI 1322 terminal.

The motor housing is kept from rotating by a bracket mounted on its side, which is fixed to the ground through a load cell. This is a device that contains a pre-calibrated strain gage, designed specifically for measuring tension and compression forces. The device we use is a model 31/B331-03 from Sensotec. The cell requires a five volt excitation input, and outputs a voltage signal with a calibration factor of 15.8226 mV / V, linearly proportional to the load experienced by the cell. This signal is sent to the SCXI 1122 module, through the SCXI 1322 terminal. This signal represents the force necessary to hold the motor with a known moment arm.

Pressure transducers are electronic devices that measure fluid pressures, and output a voltage or current signal based on the magnitude of the pressure. The Analog to Digital converter module, SCXI 1122, has capacity for 16 channels. In addition, the SCXI 1000 chassis has the capacity to add two more converter modules within it. The number of channels required depends on the experimental setup.

Software - Functional Requirements and Description

The requirements of the DACS program are that it completely controls the process of running an experiment on the TS 9005. This includes making a connection via the Internet, powering up the apparatus, checking that the operation of the motor is within normal limits, prompting the user for inputs and experimental parameters, collecting the data, then saving the data, allowing the user to download the data, and terminating the web connection. The TS 9005 can be set to many experimental configurations, as discussed previously, and each hardware configuration requires a software solution. The main change is in the selection of the data sampling channels that will be used in the desired experimental configuration. While these channels will change, there are some parameters that are used for all the various set-ups. These are the data parameters, which determine the number of trials to run, the number of samples to collect, and the sampling rates for the data.

The graphical user interface (GUI) that presents to the user, called the "front panel" should resemble the face of an engineering laboratory instrument. That is, it should give the user the sense of "being there" in the lab, and operating the actual apparatus. To this end, controls and displays should be as realistic as possible. The operation of the program should also be as intuitive as possible. The progression of inputs and indicators should follow a logical, left to right sequence across the interface. When the user is required to interact with the display, he should be prompted that the next operation is now ready, and if an error occurs, he should be notified. It is also desirable for the user to confirm their data and control inputs before execution, to avoid typing errors. It is an important requirement that the user be able to concentrate on the content of the experiment itself, rather than on the operation of the program. This means that many low-level details that are not related to the experimental task being studied should be transparent to the user. These include initialization, operational status checking, file handling, closing the program and most importantly, the web connection should be simple to execute.

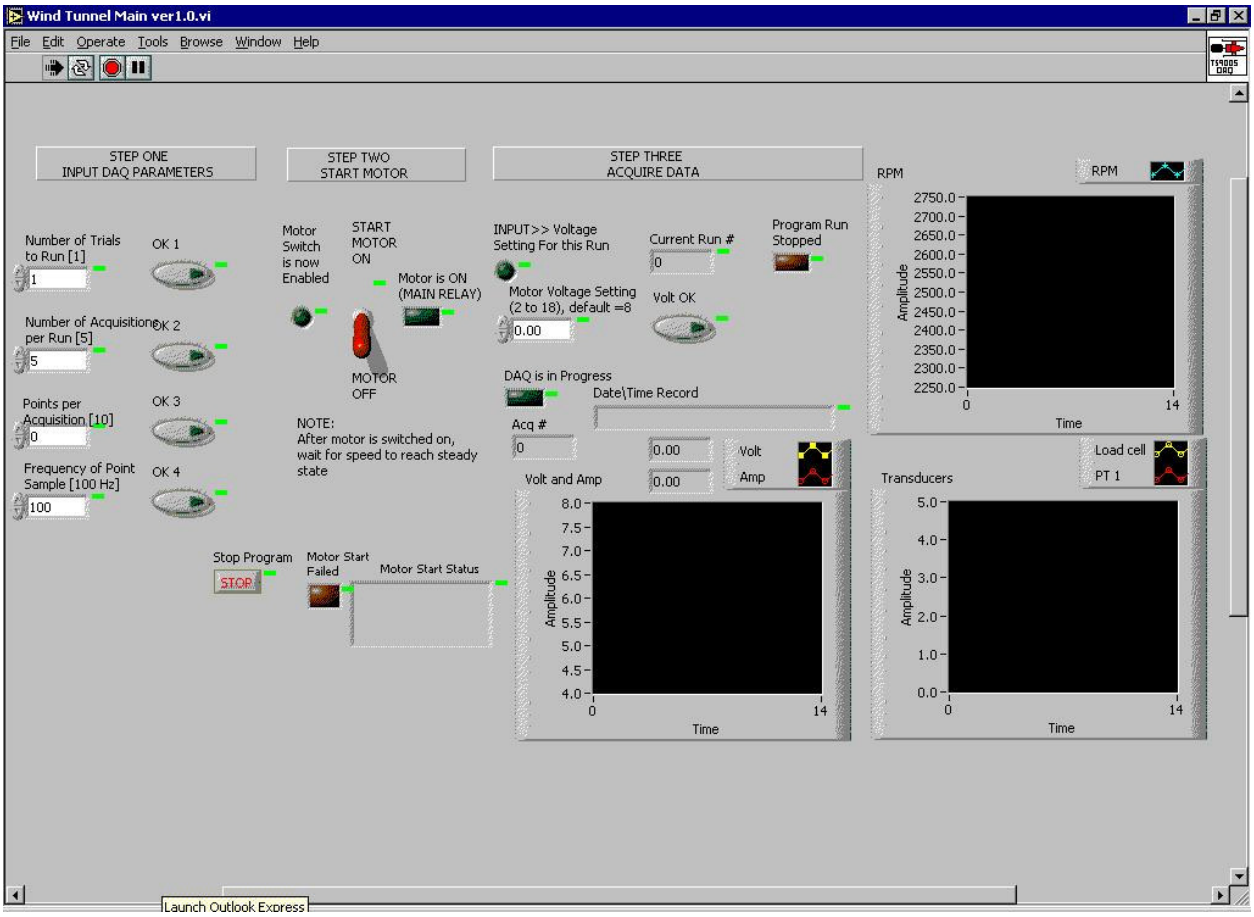

Figure 2 Front Panel

The front panel as designed for the prototype is represented in Fig. 2. There are three major steps in its operation, each of which has operations that include user inputs and prompts and status indicators. Each of these steps are covered in detail in the sections that follow, but it should be pointed out that each step is a separate functional program entity, which must be successfully completed in its entirety before the next step can be accessed. This insures that no

step is incomplete due to user omission, which is quite likely in a learning environment. Furthermore, all user numerical inputs are not recorded in the program and program control does not advance, until the user confirms the values. Once a step of the program is completed and exited, it cannot be re-entered.

The first program task is to initialize all the variables to their default values. This includes the DAQ parameters and the Boolean controls, which the user will operate, as well as the Boolean prompts and indicator LED's. Also, the values of the analog output channels are set as well. The KB channel is set to zero volts, so no unexpected operation occurs when the program is begun. The string containing the IP address of the host computer and the data socket connection for the waveform data is also initialized and passed forward to the Open Data Socket VI function. This is necessary for the real-time waveform data to be passed over the TCP/IP connection. When all these steps have occurred, control passes to the data parameter input section.

The data are collected according to the parameters set by the user. Each experimental setup uses this input, and it is the first thing that must be completed by the user in the program flow. Each parameter has a default value, shown on the front panel, and a minimum and maximum value, which cannot be changed by the end user. The first parameter is the number of trials to complete. The user may select up to ten runs, each of which will be saved to its own data file. The second parameter is the number of acquisitions to be made per run, the default value is ten, ranging up to twenty. An acquisition is made by taking several data point samples, at a specified frequency, then averaging the values to yield one acquisition. The third input is the number of points per acquisition, ranging from 10 to 100, defaulting to 50, and the fourth is the frequency of the point samples, ranging from 20 Hz to 200 HZ, defaulting to 100Hz. As the user sets each parameter, he or she is prompted to confirm the values by pressing a Boolean control button, which turns green in color to indicate that the user has confirmed the value. Once these parameters are set, they cannot be changed until all the trials are completed or the program is terminated. This insures each trial has the same DAQ characteristics. Each trial run, however, prompts the user for a value for the variable input to the experiment. In the prototype, this is for the value for the voltage across the fan motor. So as an example, five runs could be completed, consisting of 20 acquisitions, each acquisition being the average of 50 points taken at 100 Hz. Each run could be done at varying voltages, such as 6, 8, 10, 12, and 14 volts, giving a profile of the experimental data at varying power levels. After user confirmation of all four parameters, control passes to the motor start sequence.

After the user defined DAQ parameters are entered, the user is prompted that the motor on-off switch is now active by the presence of a green LED. The motor switch will not be active until all the data parameters are confirmed. When the user activates the switch, the KB channel is set to provide a voltage to the motor of 8VDC, and the main relay is turned on. After five seconds, the value of the current flowing thorough the shunt resistor is checked for normal operation. Since the KB controller is set to bring the motor acceleration up to speed in four seconds, the current flow should be in the nominal range in five seconds. If the current is not below 10 amps at this time, the motor start is deemed to fail, and the program is terminated, with a dialog box informing the user of the reason.

After a good start, program control passes to the data acquisition loop. This is a series of nested loops. The outer loop is a “Case” structure, which tests the exit condition from the motor start sequence. If False, then no operations take place and control passes to the termination routine. If True, the next layer is a “while” loop, which runs as long as the number of iterations is less than the data parameter value for the number of experimental runs. Inside this while loop, two other loops are accomplished. First the Volt Input sequence is run, since the “For” loop cannot execute until it has received the input from the Volt Input sequence. In this sequence, the user is prompted for the desired voltage for this experimental run. After entering this numeric value, the user confirms the value by pressing a Boolean control. At this point, the voltage input prompt is turned off and a “DAQ in Progress” indicator is turned on. These steps are accomplished in the sequence structure named “Input Voltage Setting”. The voltage input is passed to the KB1.vi sub-program, which performs a linear scaling to convert the zero to 18-volt range of the motor to the zero to 7-volt range of the Kb signal. This 0 - 7 VDC signal is sent via the Analog Out Write One Point VI to the KB channel. This value remains on the channel output until re-updated. At this point, the motor responds to the new voltage level, and data acquisition can begin.

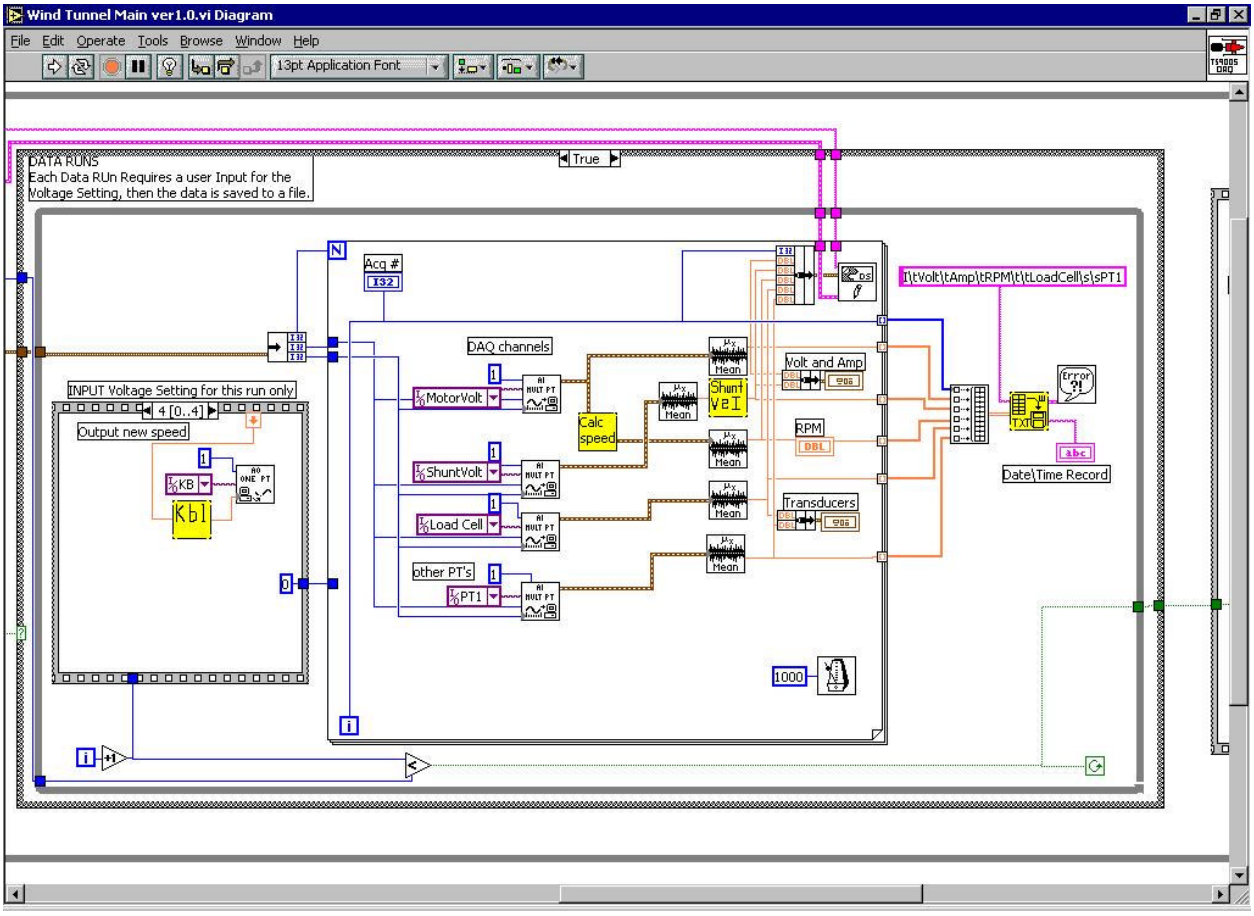

Figure 3 Data Acquisition Loop

After the voltage is set for the current run, control passes into the "For" loop, which is set to execute for the number of acquisitions. In each iteration, the data is acquired from the analog data channels by the Acquire Waveform VI. This routine takes the device number, the channel name, the number of samples, and the sampling rate as inputs. It then acquires the data and sends out a waveform data type. This waveform is a vector of the data values acquired, in sequence. This subroutine is used for each of the four channels in the prototype program, the voltage across the motor, the voltage across the shunt resistor, the voltage from the load cell, and the voltage across a pressure transducer. In another experimental configuration, there may be more data channels to be sampled. The output from the motor voltage channel is sent to a VI that calculates the RPM based on the voltage, and the voltage across the shunt resistor is sent to another VI that calculates the current through the resistor.

The data as acquired is then sent to a function that calculates the arithmetic mean of the values. This average value is stored in a vector array at the edge of the loop, that is, these average values accumulate in a vector as the loop is running, then are passed out to the next function after the loop has exited. These one-dimensional vectors are then sent out of the loop, into an array building function, which creates a two-dimensional array from the data vectors. This larger array is then sent to a VI which transposes the array, so that the row vectors for each data type are converted into columns, then writes the two-dimensional array to a spreadsheet file, adding the time and date and data column headers. The user is prompted by a standard "save file" dialog box to save the file. While the data handling and file saving operations are in progress, no data acquisition takes place, and the motor remains at the last speed setting.

This entire process is repeated for the number of trials that the user has input. Once all the runs are completed, control passes to the program termination function, which sets all the variables and channel values prior to program termination. The user is notified that the program has ended by the presence of a red LED that is labeled "Host Program has Stopped".

Connection to Internet and World Wide Web

The host PC in the lab was connected to the Internet by a direct fast modem line, which was connected via a router in the lab to the main web server connection in the Mechanical, Materials and Aerospace Engineering Department. The IP address of the host was 132.170.203.109. This could change in the future, if the connection is broken and re-established using a different line. This would require some software data changes, which are detailed in the next section. No other special hardware was required for this.

The host PC in the lab had LabVIEW 6i, Professional Developer Version 6.0, installed. This software includes the LabVIEW programming environment, as well as a built-in web server and application builder. The web server is a separate application, which resides on the host, and establishes a connection through TCP/IP with the Internet. The application builder is a compiler that allows the building of stand-alone executables to run on any Windows platform. It also allows the building of an installer program for LabVIEW run-time engine.

In order for the data to be passed over a TCP/IP connection, there must be a web server application, which handles the connection, creation and distribution of data packets, and packet

checking. The NI program "Data Socket" handles this function. The types of data to be handled by the server are configured one of two ways; either through directly configuring them in the Data Socket Manager, or programmatically. The Data Socket Manager is a component of the Data Socket application which configures the data types used in a real-time VI, assigns them addresses in the Windows OS registry, and makes them available to the Data Socket server. Thus when a remote client requests to subscribe to a data type, the Data Socket server knows exactly where to find that data in the host computer. Also, when a remote client publishes data, as in the case of a control signal being sent, the server places the new value in the proper data register, which immediately updates the value in the running application.

There is a limitation to this method of creating data connections. This method will only support single data values or arrays of similar types of values. That is, it will not support clusters of dissimilar types, such as numeric and Boolean combined, or real-time waveform data, which contains time information as well as the numeric values of the data waveform. This type of data must have its connection made programmatically. This allows the programmer the flexibility to set the types that will be included in the connection, which can be any of LabVIEW's supported types or structures. This method can also be used for single data types, such as the controls and indicators used in this application, but programming some 26 variables is a task that would clutter the program diagram, making the more important control flow less easily discernible. In essence, the programming decision to use the DS Manager for the single data types was made on this basis. Once the DS connections have been made, a small LED appears on the front panel next to each data type. When the DS sever is active, if there is a valid connection between the server and the data item, the LED shows green; if there is not a valid connection, it shows red. Moving the cursor over the red LED reveals an error message.

Remote Application

Once the host application has been created and the DS connections are operable, the remote application can be created. This is done by copying the entire front panel, or the part that is need for the remote application, into a new program file, and saving it under a different name. This has the result of placing in the program diagram only the data types represented on the front panel, with no program control. This application can now only operate when it is connected to the corresponding host program, which will provide the program flow and control. In order to allow the remote to operate, the data types must be placed into a "while" loop in the program diagram, and a Boolean STOP switch is paced on the front panel, to be wired to the while loop conditional test. This allows the remote application to continue to operate until either the host or the remote user terminates the program. The DS connection LED's appear next to each data type again, and if one shows red, the error message must be investigated to correct the problem.

At this point, the remote application is still a LabVIEW VI; that is, it can only operate on a PC that has LabVIEW installed. In order to allow users who do not have this program installed to use the remote, it must be compiled into a standalone executable. This is handled by the Lab VIEW application builder. From the remote application front panel, this compiler is launched from the "Tools" menu. Here the name of the file and the destination directory, are entered. Next the top-level VI is selected, and any sub-VI's are included. A custom icon can be created, and Active -X server can be included. For any target computer that does not have LabVIEW

installed, the LabVIEW runtime engine must be included. This is a self-installing program that will actually allow the remote client to run and communicate with the host program, allowing LabVIEW data types to be passed and correctly interpreted. Once all the desired options are entered, running the compiler will create the installer for the run-time engine and the remote application. The completed file can then be stored on magnetic media or to a web-site, and distributed.

Summary of Internet Connection

In order for the two applications to communicate across the Internet requires several components. First, the host application must be running on the host computer. The host must also have the Data Socket server operating. The remote client must have installed either LabVIEW or the LabVIEW run-time engine, and the remote application created especially for the host application. When the remote program is launched, it will automatically attempt to connect via Data Socket Connection to the IP address of the host. This is done programmatically; the user does not have to know the address. Once a connection is established, the remote application can control, display and download data as though it was the host program, to the extent that the program allows remote access.

Summary

It is important in the modern engineering environment to be able to create computer controlled and monitored systems for engineering education. It is also important for engineers, who may have some general programming experience, to be able to tailor these computer controlled systems to their immediate needs, without long design times and recourse to computer programming specialists. This was a project in which National Instruments programming language LabVIEW, in combination with DAQ hardware, was shown to be an effective tool, to be able to design and implement Data Acquisition and Control Systems, including remote access and control via the World Wide Web.

The conversion of the TS 9005 wind tunnel from a manually controlled system to one programmatically controlled by a computer was effected successfully. The Digital Acquisition and Control System was demonstrated, and real-time data was acquired. The successful demonstration of the Internet connection via Data Socket server was demonstrated, with both the host application and remote client application being compiled and successfully demonstrated. In this demonstration, the client application, running on a computer with Internet access, was able to connect to the host computer, which was running the host application, and control the operation of the TS 9005. Further, the client was able to display the data acquired from the DAQ system as it ran on the host system.

Bibliography

1) Wheeler, A., Ganji, R. (1996). Introduction to Engineering Experimentation. Upper Saddle River, N.J.: Prentice Hall.

2) Fredian, T., Stillerman, J. (1999) MDSplus remote collaboration support-internet and worldwide web. Fusion Engineering and Design 43, p. 327-334.

3) Holmes, W., McMillan, D., Smith, R. (1999) Using LabVIEW to Access and Control a Harsh Environment Laboratory through the Internet. National Instruments' Academic Solutions http://digital.ni.com/articles.nsf/4e809e481ad5f764862563ef0053261d/0d2718db14a41b2c8625646100613b0c

4) McHarg, B., Casper, T., Davis, S., Greenwood, D. (1999). Tools for remote collaboration on the DIII-D National Fusion Facility. Fusion Engineering and Design 43. p. 343-355.

5) Krom, J. (1999) The evolution of control and data acquisition at JET. Fusion Engineering and Design 43. p. 265-273.

6) Nakasnishi, H., Kojima, M., Hidekuma, S. (1999). Distributed processing and network of data acquisition and diagnostics control for large helical device (LHD). Fusion Engineering and Design 43. p. 293-300.

Biographical Information

McFARLANE, ROBERT M., B.S.M.E.

This prototype for this project was completed as an undergraduate honors thesis by Mr. McFarlane in the Department of Mechanical, Materials and Aerospace Engineering, College of Engineering and Computer Science, at the University of Central Florida. The thesis is titled "Remote Operation of the Scott Axial Turbofan via the World Wide Web", December 2001.

McBRAYER, JAMES D.