Bradley University

Automated Industrial Wind Tunnel Controller

Project Proposal

Nick Detrempe and Daniel Monahan
11/29/2011

Project Description

The Automated Industrial Wind Tunnel Controller project involves integrating a microcontroller into the pre-existing wind tunnel system and utilizing a server-client approach to control the wind tunnel either remotely or locally using a computer interface. Using a LabView based client users will be able to remotely manipulate the actuators on the wind tunnel and the wind tunnel itself to create different testing conditions. Data will then be relayed back to the client which will both display it for user feedback and export it to a spreadsheet to save testing data. The user will also be able to view the wind tunnel using 3 web cams that will be placed to allow remote users the most informative viewing experience.

Previous Work Completed

2010: Benjamin Morrison and Michael Firman

- Performed system analysis on the wind tunnel.
- Acquired solid state relays to control damper and blower through a microcontroller.
- Designed H-Bridge circuitry to control linear actuators through a microcontroller.

2011: Adam Green

- Performed system analysis on the wind tunnel.
- Acquired solid state relays to control damper and blower through a microcontroller.
- Designed H-Bridge circuitry to control linear actuators through a microcontroller.

Changes to Previous Work

- Removal of the National Instruments digital to analog converter
- Microcontroller automatic shutdown to protect components
- Electronic and manual control switching method

Current System Concept

Our proposed system will utilize 3 main parts: the microcontroller, the computer with the server, and a remote client. The microcontroller is a Silicon Labs C8051F120. It has an 8-bit 8051 processor and an 8 channel analog to digital converter. This allows us to collect data from sensors on the wind tunnel and to send control signals to the control interfaces on the wind tunnel.

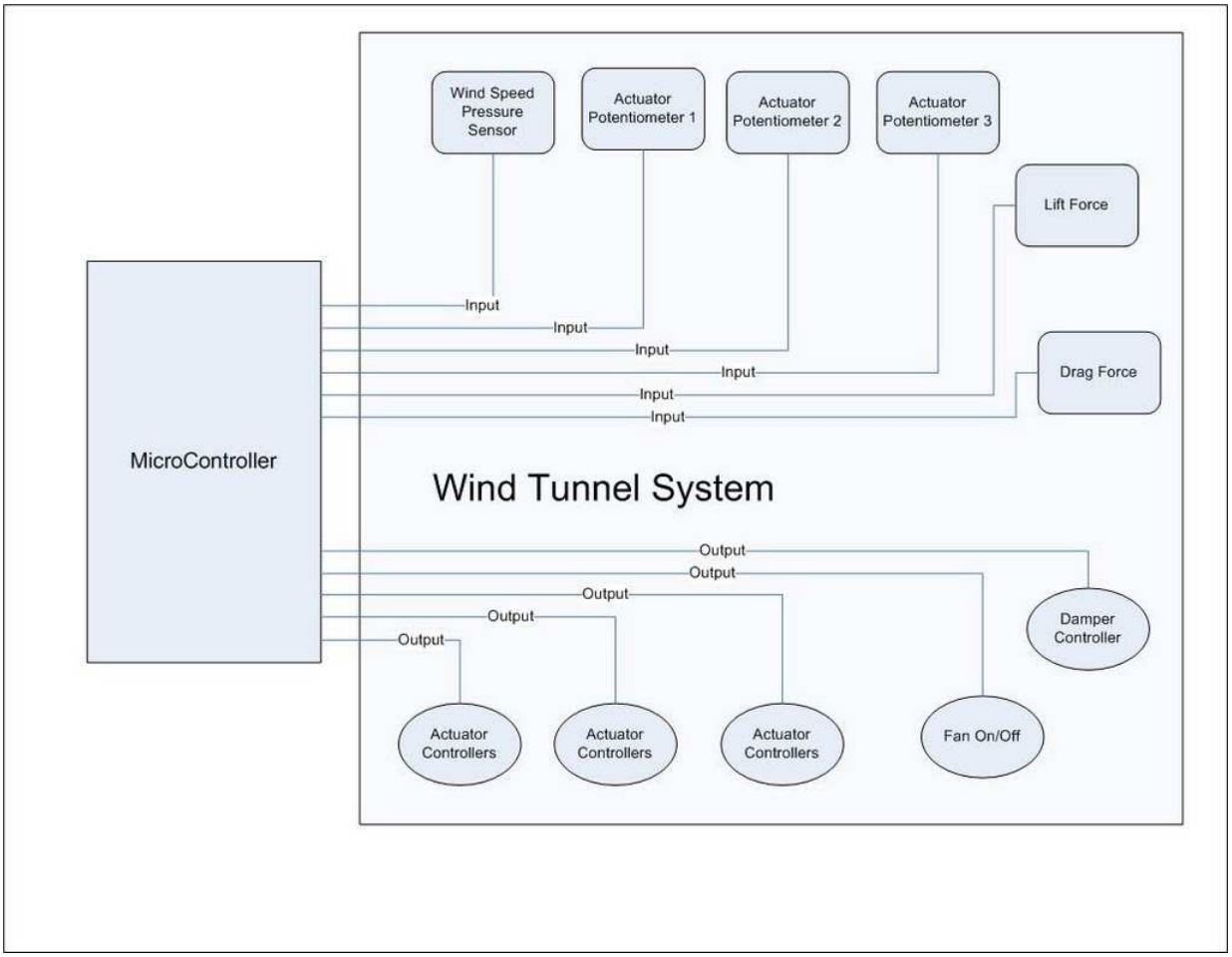

Figure 1: Wind Tunnel System Inputs and Outputs

The Silicon Labs microcontroller then communicates to the Server subsystem via a serial RS232 connection. The microcontroller will be sending a steady stream of sensor data to the server via this connection. At the same time it can accept commands from the server and execute them. The microcontroller will have built in checks to prevent over-extending or taxing the wind tunnel to mitigate any possible damage to both the tunnel and to the microcontroller and electronic components. The server system will take received data and condition it and send it on to the client via TCP. It will also relay commands from the client to the microcontroller. The client will be the user interface. Able to be operated remotely or locally, it will allow the user both to view the data sent by the microcontroller and to send commands to the microcontroller via the server. The client will also have some capability to view the wind tunnel either integrated in the main user interface or through a separate interface.

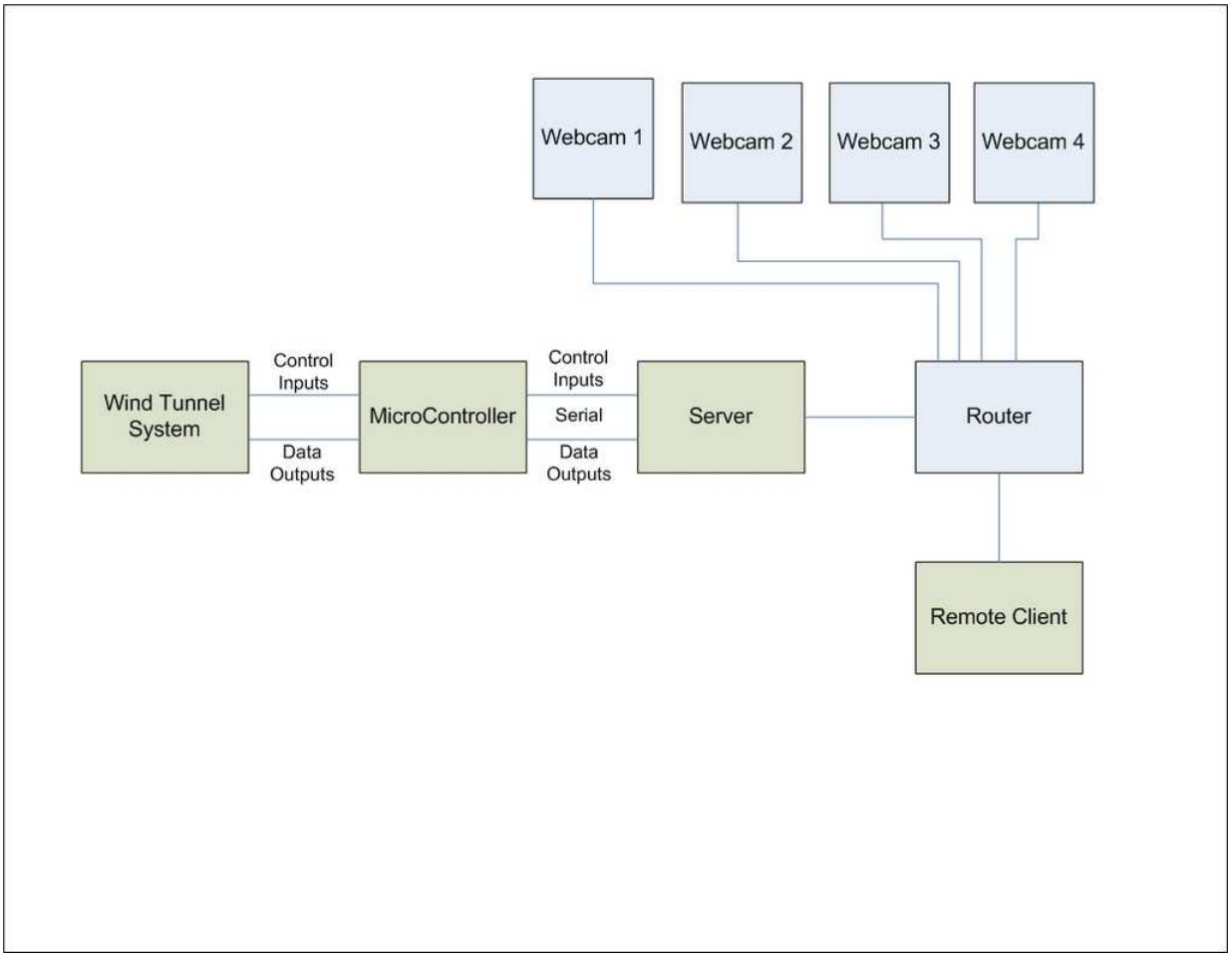

Figure 2: Overall System Block Diagram

The microcontroller code will use interrupt based A/D conversation. The 6 channels being converted will execute in a round robin fashion at 600 Hz, giving a 100 Hz sampling rate for each channel. Each time a conversion is completed, a flag will be set to inform the main loop that new data is ready.

The main loop will poll three sources. The conversion complete flag, the character received flag and the command queue. When a conversion is complete, the main loop will average the new value along with the values of the fifteen most recent samples to implement a basic filter. This averaged value will be sent to the server via a serial port.

When a character has been received via the serial port, the main loop will call a function that will begin looping to receive characters. This loop will continue until an end of line character has been received. When a command is found in the command queue, this command will be executed.

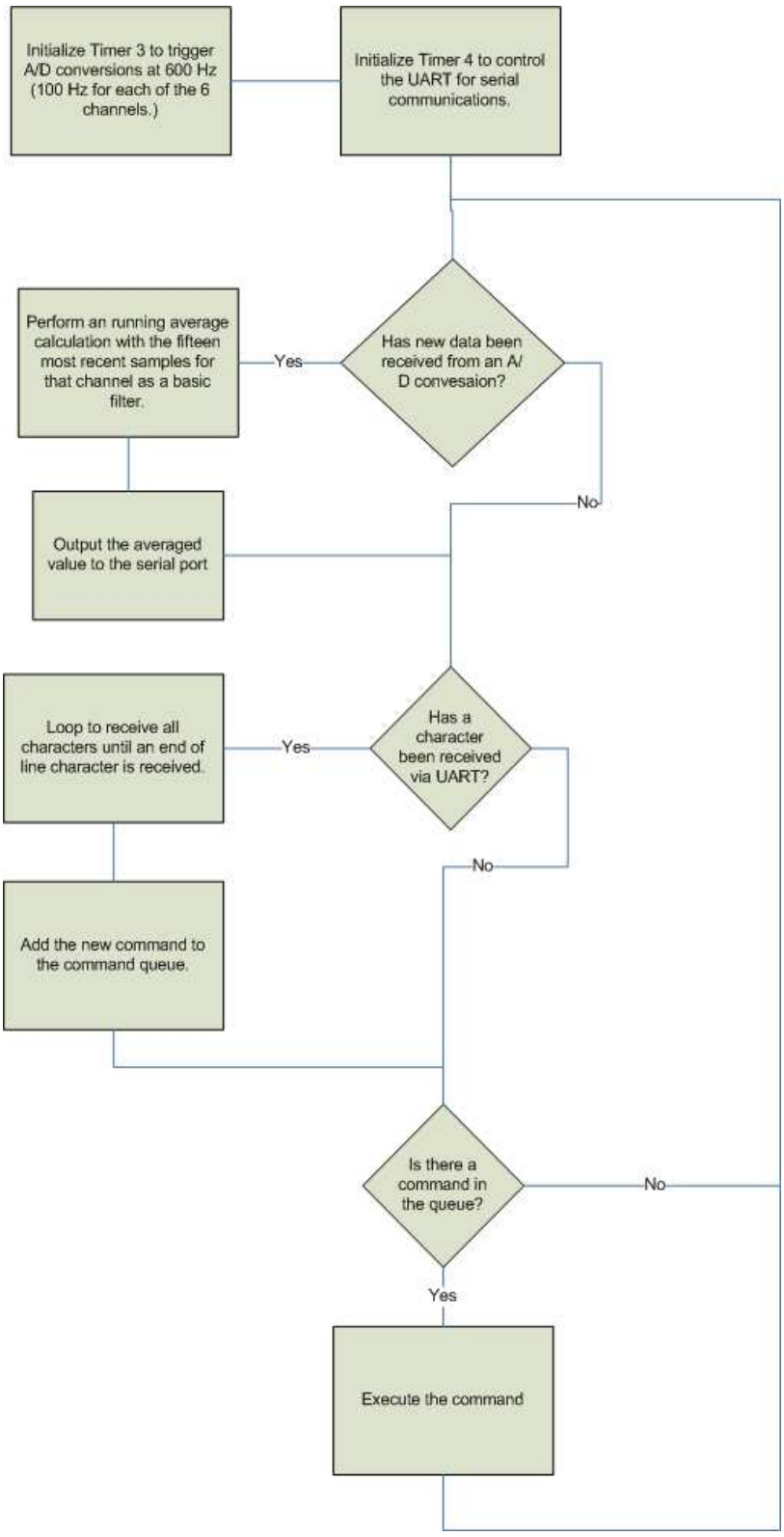

Figure 3: Overall Flowchart of Microcontroller Code

The server component for the project is currently being written using Java. Java seemed the best solution for both capability and ease of development. The server began in C but the need for multiple threads arose and we moved to Java. So far the server has functioned as little more than a communication relay between the LabView client and the microcontroller. The server juggles TCP read and write functions as well as reading and writing to the computer's COM port. Currently the server accepts up to ten clients and will send data received from the microcontroller to all of them. The end configuration will have the server further distinguish which client has control of the wind tunnel and will restrict control to that client.

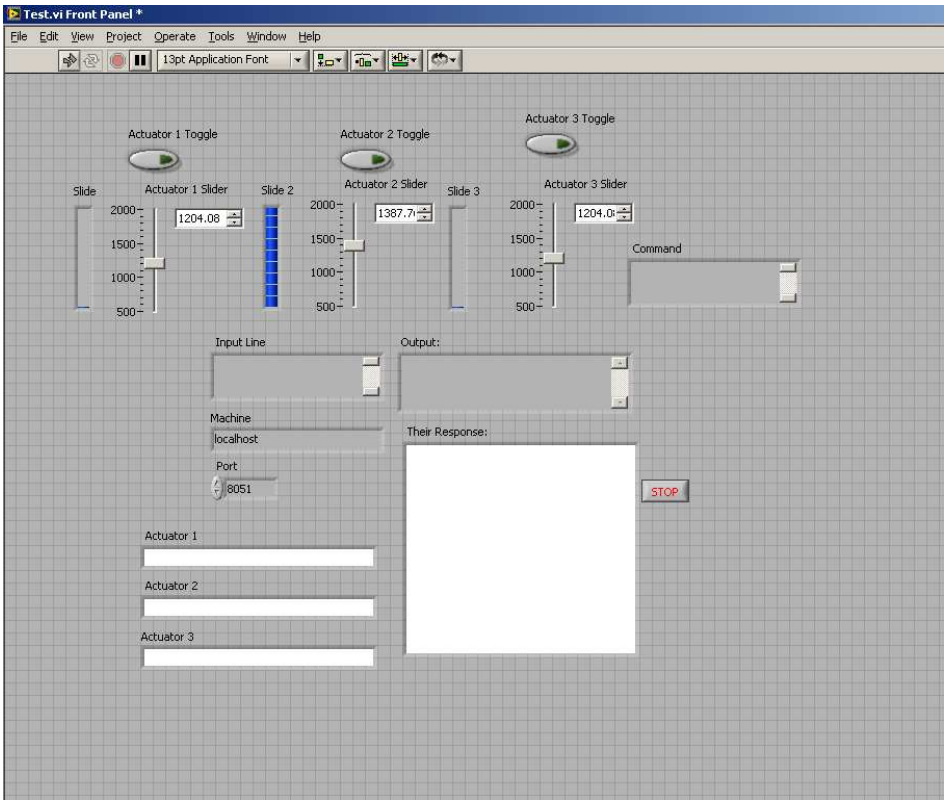

Figure 4: Current LabView Client Program

Figure 4 depicts the current status of the LabView client. The client communicates using TCP to the server on the host machine. The server is written in Java. The client currently has successfully been tested to both send and receive text from the microcontroller. Integration of graphical interfaces to both display data and send commands has also been tested successfully albeit with some bugs. The code for LabView is "written" graphically (Figure 5). To some degree the code flows like one main program, however using independent while loops, separate threads can essentially be created to allow for independent processes. This allows for more reliable communication.

LabView provides a library of functions that enable developers to more quickly develop more complex code. For example there are numerous TCP communication functions as well as other communication protocols. Hardware interface is another strength of LabView even though we are not utilizing it in this project. LabView also provides the ability to create custom functions, “VI” files, that allow developers to condense and simplify the reading of their Block Diagrams. We will start creating VI’s once we begin establishing effective and efficient code segments for the various components of the client.

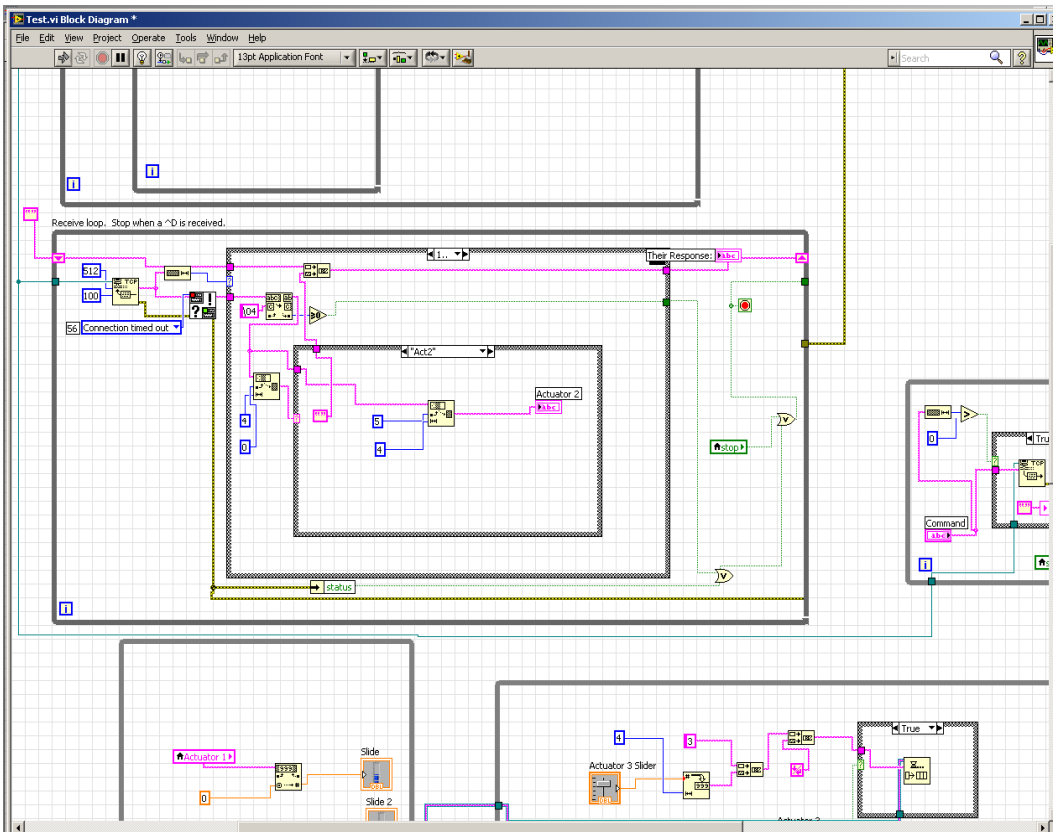

Figure 5: Segment of LabView Block Diagram for Client Program. Center block is a while loop for TCP read functionality

Work Completed

- Initial analysis on existing setup (some of it has been disconnected by ME staff to allow for manual control.)
- National Instruments Digital to Analog converter has been disconnected from microcontroller.
- Code for microcontroller currently is being rebuilt (Nick).
- LabView user interface and server is currently being updated (Daniel).

Schedule of Upcoming Work

	Fall 2011	Weeks 1 - 2	Weeks 3 – 5	Weeks 6 - 7	Weeks 8 - 10	Weeks 11 - 14
Daniel	**Complete LabView and server setup.**	**Connect all relays back to wind tunnel.**	**Test and debug LabView on wind tunnel.**	**Analyze existing transducers measuring lift, drag and wind speed and determine how to measure and display values**	**Set up and test remote access.**	**Prepare for final presentation.**
Nick	**Complete UC setup.**		**Test and debug UC on wind tunnel.**		**Set up and test webcam network.**	

----------Critical Path----------->>>>

Possible Project Extensions

- The current software allows for a user to turn on the wind tunnel and open or close the damper, while manually monitoring the wind speed. A possible upgrade will be to allow the user to input a desired wind speed and the controller adjusts the damper appropriately.
- The microprocessor and local PC could be removed and replaced with a Beagle Board.