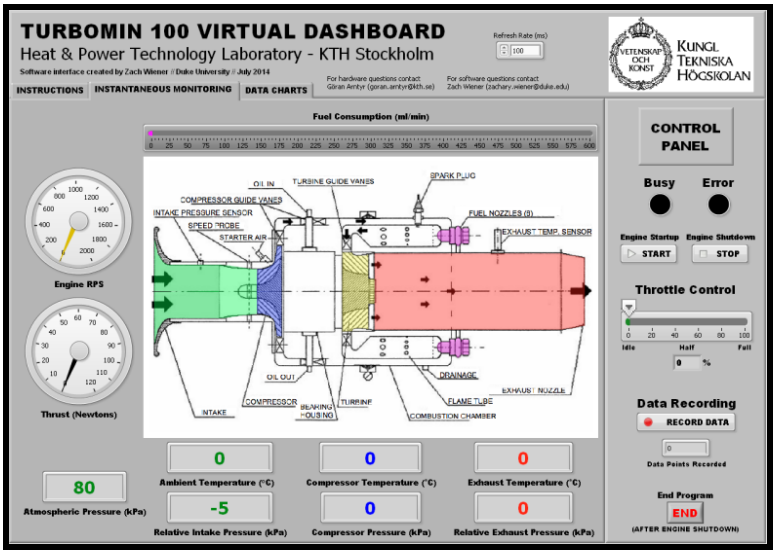
Creating a Virtual Dashboard for a Turbojet with LabVIEW



Screenshot of Turbojet_Virtual_Dashboard.vi

Zachary S. Wiener

Civil & Environmental Engineering
Energy & The Environment Program
Duke University c/o 2015
Durham, North Carolina, USA

2014 Summer Researcher
Heat & Power Technology Laboratory
KTH Royal Institute of Technology
Stockholm, Sweden

Table of Contents

Introduction: How I Ended Up In Sweden

During the fall of 2013, I (like every other student at Duke) was in search of something to do during the summer. I was searching for internships in the Civil Engineering field, as well as research positions in laboratories at Duke. I also applied to several engineering firms in France, as I speak fluent French. During my search, I received the following application via an email from the Duke engineering school encouraging students to apply to a research opportunity in Stockholm.

Duke-KTH Internship Application
The Duke-KTH Internship Program offers Duke students the opportunity to spend a summer in Sweden at the Royal Institute of Technology (KTH) doing project research. This 10 week program begins in early June and includes funding to cover most of your travel and living expenses. KTH will provide assistance finding student housing in Stockholm.

Current Project Research at KTH includes:
Zero Emission House - design and build a "net-zero" house that operates totally off the grid
Membrane Distillation - produce ultra pure water using membrane distillation and solar heat
Aerospace - turbine or micro-turbine research to produce power or electricity
Other research areas - students are encouraged to submit other project or research ideas

This program involved various aspects of energy production and turbines, as well as the chance to submit my own research project if I so chose. After some searching online, I decided that a summer in Stockholm interested me very much. I was finishing a great semester in Metz, France, and the chance to go back to Europe in a few months appealed to me. Shortly after applying, the program director, Jim Gaston at Duke University, emailed me to say that I was selected, and I immediately accepted the position.

The Royal Institute of Technology (Kungliga Tekniska Högskolan, or KTH), is a research university located in Stockholm, Sweden. My task for the summer was to move to Stockholm, and spend nine weeks as a research assistant in the Heat & Power Technology Laboratory. As I would soon find out, the laboratory is heavily focused on turbine dynamics and performance, a field with which I was not very familiar. But my computer skills as a Duke engineering student allowed me to tackle a programming project that would benefit the students who take classes in this department at KTH.

Project Overview

Background Information

I was assigned a supervisor for this project, Dr. Nenad Glodic, a full-time researcher at KTH. One of the classes that he regularly teaches is focused on jet propulsion, and incorporates a miniature turbojet engine (Figure 1) as a teaching activity. The engine is the cylindrical object on top of the cart in the figure. The functions of the engine are regulated by a control box, as shown on the left-hand side of the cart in Figure 1. This control box is responsible for the fuel pump, oil pump, and starting of the engine, as well as monitoring a multitude of sensors that measure pressures, temperatures, and other vital information at various locations in the engine.

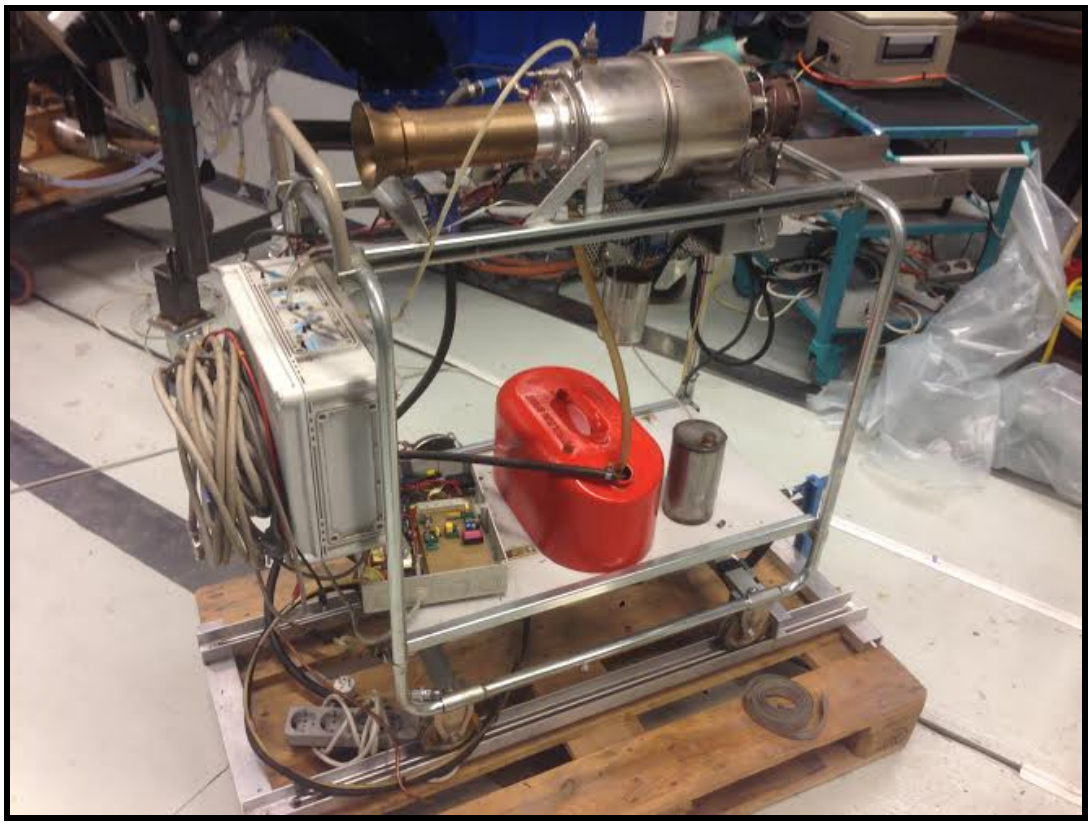


Figure 1: Miniature Turbojet Engine

Currently, the students and lab engineers operate the engine with a "dashboard" (Figure 2) which is connected via serial cable to the control box. This dashboard has analog dials that display data from the sensors, such as engine speed, fuel consumption, and temperatures and pressures at various locations in the engine. There are a total of nine dials that are operational. At the center of the dashboard is a cutaway of the engine. It is exactly one half of the Turbomin 100 engine, but sliced in half for display purposes. Some of the dials have a wire that points to the location where their reading is taken.

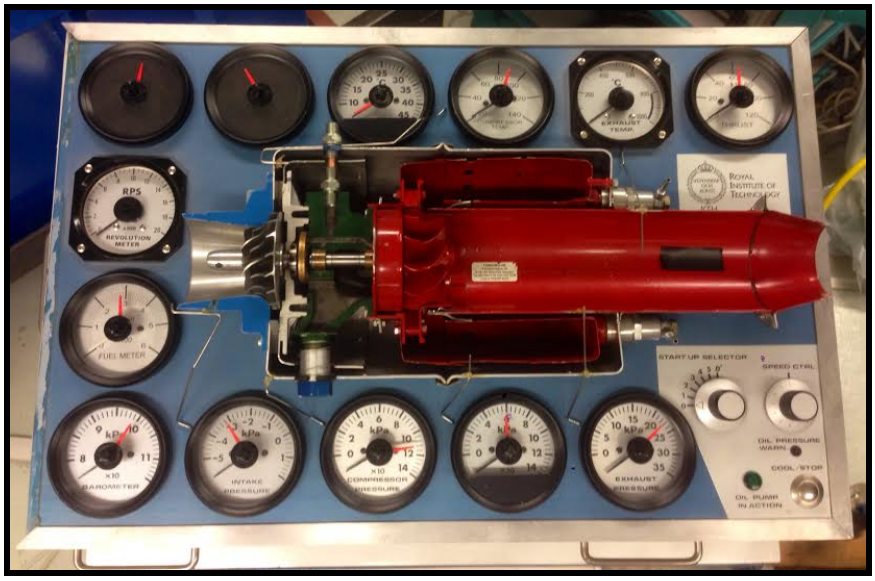


Figure 2: Analog Dashboard with Engine Cutaway

User inputs on the dashboard include a "startup mode" selector switch, where the operator switches the system through a sequence of modes in order to start the engine (see Figure 3). There is also a throttle knob, and a shutdown button. Starting the engine with the "startup mode" selector is a process that must be done very precisely, and is currently only done by the professional lab engineers. The students can manipulate the throttle knob once the engine is running in order to change the speed, and thereby change the temperatures, pressures, and fuel consumption.

Figure 3: Analog Dashboard Control Panel

The system functions as intended in its current configuration, which was assembled in the mid-1990s by the engineers at the HPT Lab. Twenty years later, there are some things that could be improved upon, and that is the motivation for my summer project.

Desired Improvements

The main complaint from the faculty that use this engine as a teaching exercise is that there is no data recording capability. The students can only observe the needles on the dials moving as they manipulate the throttle, but there is no way to go back and analyze the data after finishing the activity. It is more of a qualitative experiment, and a more quantitative interface would be a vast improvement. With so many scientific experiments controlled by computers these days, it was time to update the turbojet engine to be controlled by a computer interface.

A secondary complaint is that the startup and shutdown processes are very delicate to perform, and currently can only be done by the engineers who assembled the control system twenty years ago. This problem can also be solved by adding computer controls to the system.

My Task

The job I was tasked with this summer was to create a computer interface for students to control the turbojet and record data from its sensors. It needed to simplify the process of starting, running, and shutting down the engine, to the point where the engineers can stand back and monitor safety while the students run the engine.

The platform for this interface was chosen by my supervisor, and his selection was National Instruments LabVIEW software. This software is heavily used at the HPT laboratory to run the experimental apparatuses and collect data. LabVIEW (short for Laboratory Virtual Instrument Engineering Workbench) is described as being a tool for all of the following applications: Data Acquisition, Instrument Control, Test Automation, Analysis and Signal Processing, Industrial Control, and Embedded Design. National Instruments also produces hardware that works seamlessly with LabVIEW. It was decided that the HPT Lab would purchase a USB Data Acquisition card, and I would create the software and wiring scheme this summer to be assembled in the fall after my return to Duke. The following report describes the work I performed this summer.

Current System Setup

As described previously, the control box communicates with the dashboard via analog controls. Each circuit (of which there are 15) runs on 5V power, meaning that the magnitude of the voltage (0V - 5V) is the “information” passing through the wires. There is no “intelligent” communication, or passing of data, between the dashboard and control box. This proved to be very convenient for the use of a USB DAQ card (which can read voltages between -10V and 10V) and LabVIEW software.

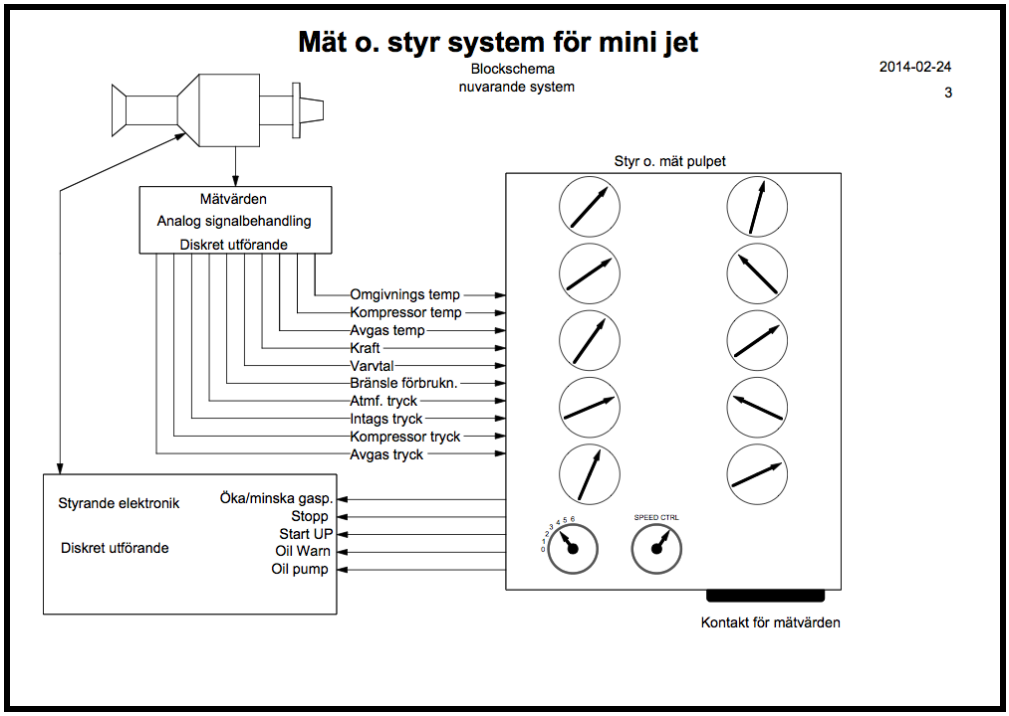


Figure 4: Current System Wiring Diagram (Swedish)

Figure 4 depicts the current wiring scheme for the system. The box with the dials on the right-hand side is the analog dashboard, and the box at the bottom-left is the control box that processes the signals from the operator and communicates them to the engine . The engine is in the top-left. The ten arrows coming from the engine into the dashboard are the analog signals that student users of this engine are concerned with. Each wire carries a voltage between 0V and 5V that is then fed to a gauge on the dashboard that is readable in terms of units like degrees Celsius and kPa. The ten signals are as follows (translated from Swedish):

- Ambient Temperature (°C)
- Compressor Temperature (°C)
- Exhaust Temperature (°C)
- Thrust (N)
- Engine Speed (Revolutions/second, or RPS)
- Fuel Consumption (Milliliters/minute, or ml/min)
- Atmospheric Pressure/Barometer (kPa)
- Intake Pressure (kPa)
- Compressor Pressure (kPa)
- Exhaust Pressure (kPa)

There is a port on the back of the dashboard where a data recording device can be plugged in (Kontakt för mätvärden), but this is currently unused by the HPT Laboratory. There are no wires from this port to the controls for the engine, only to the gauges. This renders the port of little use for this project.

On the diagram, the dashboard also has arrows that point directly to the control box. The control box processes these requests, and then activates the circuits necessary for the engine to respond as requested. The oil warning light on the dashboard illuminates if a signal is received from the control box. This arrow's direction is incorrectly labeled on the diagram and should be reversed. These are the "dashboard ↔ control box" signals:

- Engine speed increase/decrease
- Shutdown
- Startup
- Oil warning
- Oil pump

Future System Setup

After meeting with one of the lab engineers, Göran Arntyr, I was informed of the new configuration for the turbojet system. The HPT Lab would purchase a National Instruments USB-6211 DAQ card, and this would be wired to a new microcomputer that would interact with the current control box. The microcomputer (to be built by Mr. Arntyr) will control the engine, but operator inputs and data output will go through the USB DAQ card. The microcomputer will be built after my summer internship, but my responsibility was the wiring for the DAQ card and the creation of the LabVIEW interface that communicates with it. Figure 5 depicts the future diagram for the system.

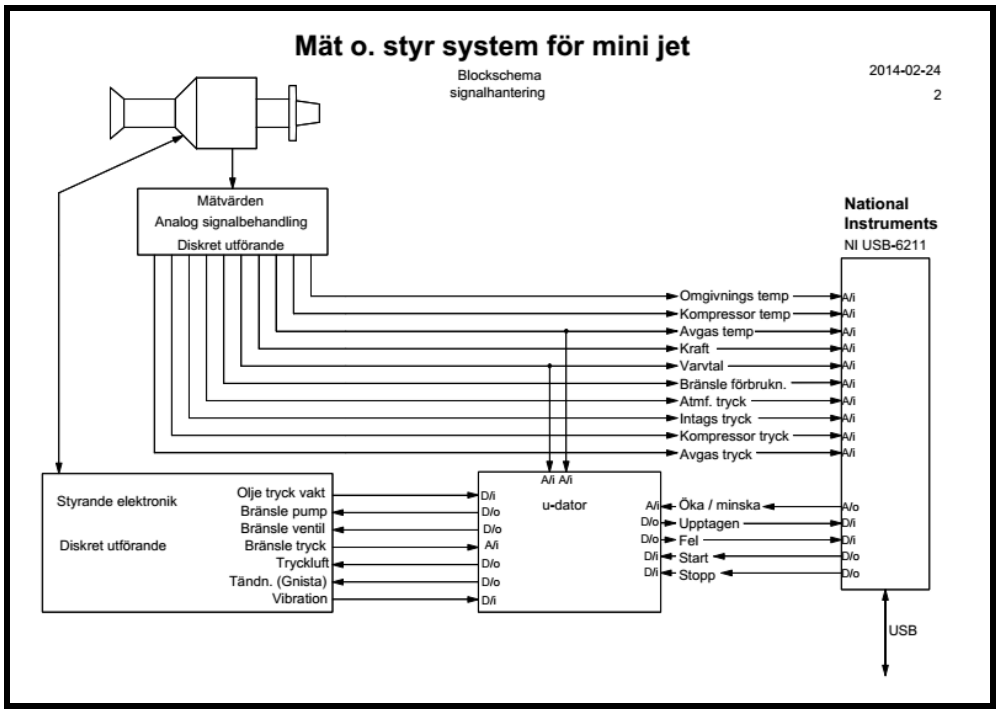


Figure 5: Future System Wiring Diagram (Swedish)

In Figure 5, the ten data outputs remain the same. But instead of being displayed on analog gauges, the signals are read by the National Instruments USB device and fed into the LabVIEW interface. Additionally, the microcomputer will send a "busy" (upptagen) signal and an "error" (fel) signal to the interface. Outputs from the LabVIEW interface are only a "start" command, a "stop" command, and a speed control (öka/minska). The additional box in Figure 5, abbreviated "µ-dator," is the microcomputer that will be built by Mr. Arntyr. The signals that are sent from the microcomputer to the control box are outside the scope of this project, thus, this report focuses on the signals sent and received by the NI USB-6211 device (far right side of Figure 5).

Each of the signals sent and/or received by the USB device is described on the following page in Table 1.

Signal Name	**Signal Description**
Ambient Temperature Compressor Temperature Exhaust Temperature Thrust Engine Speed Fuel Consumption Atmospheric Pressure/Barometer Intake Pressure Compressor Pressure Exhaust Pressure (analog in; 0V-5V)	Collectively, these are the ***analog inputs.*** The respective sensors on and in the engine send a signal between 0V and 5V directly to the USB device. The relationship between voltage and real units (degrees Celsius, kPa, etc.) were determined during the calibration process (described later in the report). Each gauge on the front panel of the LabVIEW Virtual Instrument displays one of these 5V analog signals.
Throttle/Speed Control (analog out; 0V-5V)	This is an ***analog output.*** The USB device sends a signal between 0V and 5V to the microcomputer, which has a linear relationship with throttle percentage. (0V = 0%; 5V = 100%)
Busy (digital in; 0V or 5V)	This is a ***digital input.*** When the microcomputer indicates that the engine is busy (in the middle of starting up or shutting down), 5V will be sent to the USB device. Otherwise, there will be 0V in this circuit.
Error (digital in; 0V or 5V)	This is also a ***digital input.*** When the microcomputer indicates that there is an error in the system, 5V will be sent to the USB device. Otherwise, there will be 0V in this circuit.
Engine Start (digital out; 0V or 5V)	This is a ***digital output.*** The signal to the microcomputer to start the engine comes in the form of a one-second 5V pulse. Otherwise, there will be 0V in this circuit.
Engine Stop (digital out; 0V or 5V)	This is also a ***digital output.*** This circuit has an idle position of "high," which means it always carries 5V, unless the signal to shut down the engine is being sent. This signal comes in the form of a one-second 0V pulse. This reversal is a safety measure so that the engine will shut down if the USB device is disconnected.

Table 1: Signals sent/received by NI USB-6211

National Instruments USB-6211

The USB device chosen for this application is the National Instruments USB-6211. The specifications from the National Instruments website are as follows:

16-Bit, 250 kS/s M Series Multifunction DAQ,
Bus-Powered.

MSRP: $792 USD

- 16 analog inputs (16-bit, 250 kS/s)
- 2 analog outputs (16-bit, 250 kS/s); 4 digital inputs; 4 digital outputs; 2 32-bit counters
- Bus-powered USB for high mobility; built-in signal connectivity
- NI signal streaming for sustained high-speed data streams over USB; OEM version available
- Compatible with LabVIEW, LabWindows™/CVI, and Measurement Studio for Visual Studio .NET
- NI-DAQmx driver software and NI LabVIEW SignalExpress LE interactive data-logging software

Source: http://sine.ni.com/nips/cds/view/p/lang/en/nid/203224

This device was chosen because of its easy USB connectivity, and the capacity to accept the necessary number of inputs and outputs (10 analog inputs, 1 analog output, 2 digital inputs, 2 digital outputs). The decision was made by Mr. Göran Arntyr of the HPT Laboratory engineering staff.

Learning LabVIEW

In order to learn the skills necessary to create the virtual dashboard, I spent my first few weeks at the HPT Lab taking a crash course in LabVIEW. The staff at the lab sent me a .zip file of tutorials to use as my study material. This included a six-hour course published by National Instruments in 2003, and a series of 10 shorter tutorials published by a physics professor (James Drummond) in 1996. While these materials were useful, I found the most helpful source to be the self-paced video training series found in the online NI Academic Resource Center. The link to this site is as follows: http://www.ni.com/academic/students/learn/. These videos feature a LabVIEW engineer who guides the viewer through simple programming techniques, while explaining them in terms that I could understand, as someone completely new to LabVIEW. Familiarizing myself with the concept of data flow programming was the most difficult challenge. Having learned MATLAB programming at Duke, it was not easy to change my mindset to the graphical data flow technique that LabVIEW uses.

Another resource I found very useful was the National Instruments toll-free helpline (1-800-531-5066). As long as you have an account with NI, they will connect you in a matter of seconds to an applications engineer who can answer any questions about software or hardware. Even after the conversation, they will follow up via email to make sure that they have completely answered your question, and you have up to a week to respond to keep the conversation going. I ended up calling this helpline several times over the course of the summer.

Once I had spent several weeks learning the intricacies of LabVIEW software, the USB-6211 DAQ card arrived in the mail, and I was ready to begin work on the dashboard.

Signal Calibration

Before beginning the software programming, it was necessary to determine how the voltage in each signal circuit related to an actual data value. Before Mr. Arntyr left for his summer vacation, we had the opportunity to work together on calibrating the signals. This is the process of relating voltages to real units (kPa, °C, etc.). This was accomplished by attaching a variable voltage power supply to each gauge on the old analog dashboard, and recording the voltages needed to point the needles to their minimum and maximum values. For example, when the power supply was connected to the "engine speed" gauge, it took 0.108 volts to make the needle display 0 revs/second, and it took 5.00 volts to make the needle display 2000 revs/second. These values were recorded to be input later into the "DAQ assistant," a part of the LabVIEW virtual instrument. It was assumed that the relationships were all linear, so the two minimum and two maximum values were the only points necessary for each signal.

Creating the Dashboard

VI Requirements

The first step in the process was to determine what tasks were required of the LabVIEW virtual instrument. The signals to be handled by the DAQ card were outlined previously in Table 1, but the required user functions of the program had to be outlined during a meeting with Nenad Glodic. Listed below are these key functions:

- Display data from signals instantaneously (gauges like the original dashboard).
- Display engine diagram with color coding of gauges corresponding to their sensors' position in the engine.
- Display most important data streams over time in scrolling charts (like an EKG machine):
    - Compressor Temperature, Pressure
    - Exhaust Temperature, Pressure
    - Throttle
    - Thrust
- Record snapshots of all data streams to a text file at the click of a button.
- Allow easy "one-click" access to operator controls: start, stop, throttle, record data.
- Display "busy" and "error" notifications prominently.

After this meeting, I began to construct the front panel of the dashboard. This involved finding a cutaway diagram of the Turbomin 100 engine online and color coding the important regions. Then I created gauges with colors to match their corresponding locations on the engine. This technique is cleaner than drawing arrows to point out the locations. The labels on the diagram also had to be translated from Swedish, as this was the only available language for the diagram. Once a basic functional front panel had been created, the next step was to program the analog inputs.

VI Basics

The majority of the VI (virtual instrument) is contained within a "while loop." This means that the entire operation of the VI (reading data, displaying data, etc.) happens repeatedly until the loop is ended (by a "stop" button on the front panel). There is a "wait" function inside the loop, so that the loop repeats at a specified "refresh rate." This rate can be changed on the front panel, and the default is 100 milliseconds.

The only parts of the VI that fall outside of the loop are the engine startup and shutdown commands. These are in a separate loop because their signal duration of one second would inhibit the operation of the main loop. This separate loop can be found at the bottom of the block diagram. This loop is also ended by the "stop" button on the front panel.

The user must simply press the "run VI" button at the top of the LabVIEW interface to begin using the VI. Instructions to do this are displayed on screen (in the "instructions" tab) as soon as the user opens the VI.

DAQ Assistant Analog Inputs

LabVIEW has a function called an "Express VI," which consists of a prewritten program that can be integrated into another VI. The main Express VI used in the dashboard is called "DAQ Assistant." This VI allows user-friendly access to the inputs and outputs of the USB device without extensive and unnecessarily complex programming. In my interface, *Turbojet_Virtual_Dashboard.vi*, one DAQ Assistant VI is used for each type of signal (analog input, analog output, digital input, digital output).

The DAQ Assistant for the analog inputs handles the data streams from the engine. It is shown on the left-hand side of Figure 6:

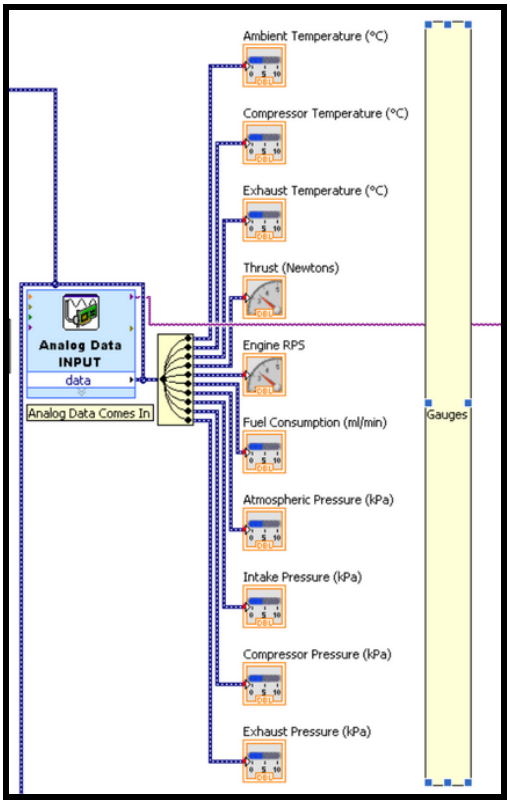


Figure 6: Analog Data Inputs

This express VI contains the calibration information determined earlier. It maps the minimum voltage to the minimum gauge value, and the maximum voltage to the maximum gauge value using the "map ranges" function. These values can be easily changed by opening the express VI, clicking a signal (on the left), and clicking the wrench icon by the "scaling" option (on the right). Details can be found in the instruction manual created for the dashboard. See Figure 7 below, which depicts the DAQ Assistant dialog box.

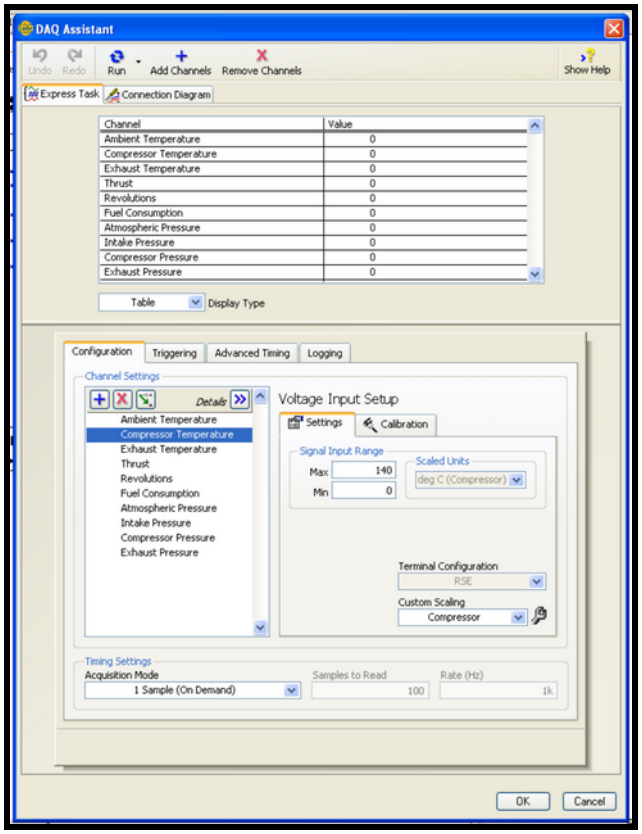


Figure 7: Analog Input DAQ Assistant Dialog

As shown in Figure 6, this DAQ Assistant is wired directly to the gauges on the front panel. This instantaneously displays the value from each of the 10 analog inputs on the gauges ("instantaneous monitoring" tab). It is also wired to the waveform charts that display the most important signals in the form of a continuously scrolling chart ("data charts" tab).

Data Recording

The dashboard is required to record only one data point for each press of the "record" button, as opposed to many applications where data is recorded continuously over a period of time. There is an express VI that was implemented to record the data to a text file, which LabVIEW calls ".lvm" for "LabVIEW measurement." The express VI is called "Write to Measurement File." It was configured using the dialog box associated with the VI, and is wired to receive all ten analog signals. It also receives the current throttle percentage and the elapsed time since the VI started to run. Each of these 12 signals is passed through another VI first that gives the datastream a name, such as "Comp. Press." This allows the text file to display a header on top of each column with the abbreviated name of the signal.

The first time a user clicks the "record" button, a dialog appears, asking for a filename. The default location to save this file is the "Data Files" folder on the desktop. This can be changed in the dialog box. Each subsequent click of the button records a data point with all 12 data streams. A display on the front panel shows the user how many data points he or she has recorded. The text

file created by the VI is displayed with sample data in Figure 8. The 21-line header can be disregarded, and is standard to all .lvm files. The highlighted "file description" line describes the units of each datastream, but that is the only relevant line of the header. In this example, the record button was pressed seven times (the first was at 1.2 seconds, and the last was at 39.3 seconds).

Reader_Version 2
Separator Tab
Decimal_Separator ,
Multi_Headings No
X_Columns No
Time_Pref Relative
Operator adm.hpt
Description **Note about units** Temperature is degrees Celsius, Pressure is kPa, Thrust is Newtons, Revs is Revolutions/Sec, Fuel consumption is ml/min
Date 2014/07/16
Time 16:15:35,46875
End_of_Header

Channels 12

Samples 1 1 1 1 1 1 1 1 1 1 1 1

Date 2014/07/16 2014/07/16 2014/07/16 2014/07/16 2014/07/16 2014/07/16 2014/07/16 2014/07/16 2014/07/16 2014/07/16 2014/07/16 2014/07/16
Time 16:15:35,46875 16:15:35,46875 16:15:35,46875 16:15:35,46875 16:15:35,46875 16:15:35,46875
16:15:35,46875 16:15:35,46875 16:15:35,46875 16:15:35,46875 16:15:35,46875 16:15:35,46875

X_Dimension Time Time Time Time Time Time Time Time Time Time Time Time

X0 0,00000000000000000E+0 0,00000000000000000E+0 0,00000000000000000E+0 0,00000000000000000E+0 0,00000000000000000E+0
0,00000000000000000E+0 0,00000000000000000E+0 0,00000000000000000E+0 0,00000000000000000E+0 0,00000000000000000E+0 0,00000000000000000E+0 0,00000000000000000E+0
Delta_X 1,000000 1,000000 1,000000 1,000000 1,000000 1,000000 1,000000 1,000000 1,000000 1,000000 1,000000 1,000000

End_of_Header

Time (sec)	Throttle %	Ambient Temp.	Comp. Temp.	Ex. Temp.	Thrust (N)	Revs/Sec	Fuel Cons.	Atm. Press.	Int. Press.	Comp. Press.	Exhaust Press.	Comment
1,218750	0,000000	10,000000	0,000000	0,000000	0,000000	0,000000	2,281323	80,000000	-5,000000	0,000000	0,000000	
14,812500	20,000000	10,000000	0,000000	0,319002	0,000000	0,000000	0,000000	80,000000	-5,000000	0,000000	0,000000	
20,312500	40,000000	10,000000	0,000000	3,032403	0,000000	0,000000	0,000000	80,000000	-5,000000	0,000000	0,000000	
28,906250	60,000000	10,000000	0,000000	0,000000	0,000000	0,000000	0,000000	80,000000	-5,000000	0,000000	0,000000	
34,406250	80,000000	10,000000	0,000000	0,000000	0,000000	0,000000	0,000000	80,000000	-5,000000	0,000000	0,000000	
36,718750	100,000000	10,000000	0,000000	0,436976	0,000000	0,000000	0,000000	80,000000	-5,000000	0,000000	0,000000	
39,312500	0,000000	10,000000	0,000000	0,000000	0,000000	0,000000	0,000000	80,000000	-5,000000	0,000000	0,000000	

Figure 8: Example Data File (.lvm)

Once the operation of the engine is complete and the VI has been ended, the user can find the file in the "Data Files" folder on the desktop in order to copy to a flash drive or send via email.

DAQ Assistant Digital Inputs

The two digital inputs to the system are the "busy" and "error" signals. A separate DAQ assistant was created for these signals, and when they receive a 5V signal, the corresponding light illuminates in red on the front panel. When the "busy" signal is active, the throttle control is disabled and greyed out in addition to the illumination of the red light. The programming for these functions can be seen in Figure 9.

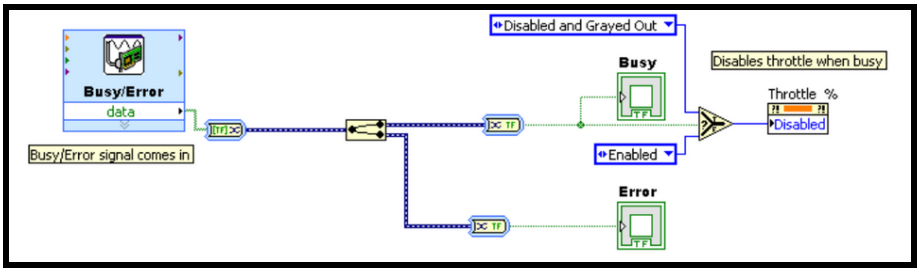


Figure 9: Programming for busy/error signals

DAQ Assistant Analog Output

The sole digital output for the system is the throttle control. The operator uses the mouse to move a slider control on the front panel between zero and full, or 0% to 100% (see figure 10).

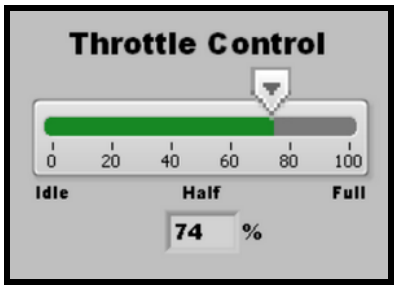


Figure 10: Throttle Control

A simple divide function converts this percentage to a voltage value between 0V and 5V, and sends it to the DAQ card through a DAQ Assistant labeled "throttle output." The current throttle value is displayed below the slider, as well as on the "data charts" page in the form of a vertical gauge. The throttle value is also connected to the data recording function, so each data point contains the throttle value at that moment in time.

DAQ Assistant Digital Outputs

The digital outputs on the DAQ card are controlled by the engine startup and shutdown buttons on the front panel. The idle signal of the engine startup button is 0V, and when clicked, a one-second 5V pulse is sent through the circuit. The engine shutdown button is configured in the opposite manner. When the VI is running, the idle position is 5V, and when clicked, a one-second 0V pulse is sent. This is a safety measure, so that if the system loses power or is disconnected, the engine will simply shut down.

This task is accomplished with two separate DAQ Assistant VIs, each placed in an event structure that sits inside a separate while loop from the rest of the program. The separate loop exists so that the one-second pulses do not interrupt the continuous refreshing of the rest of the program. The event structure/loop programming that controls these functions is shown in Figure 11.

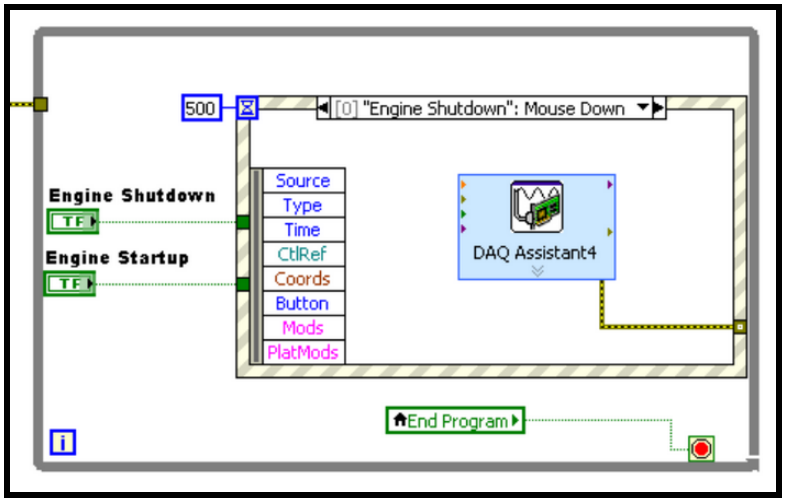


Figure 11: Event structure inside while loop

Testing and Conclusion

Once the VI was completed, the functionality was tested as well as possible without access to the microcomputer. Voltages were generated with a PS3005L DC Power Supply to simulate the analog and digital inputs (signals from the microcomputer). The outputs were tested with a UNI-T UT61D Multimeter. It read the varying throttle voltages as well as the engine startup and shutdown signals, confirming their operation as intended.

The last week of my summer at KTH was spent cleaning up the program, and composing this report, as well as an instruction manual for the software. The instruction manual begins with a student instructions section, intended to guide the student through operation of the engine. The second section of the manual details step-by-step how to change the parameters of the software for HPT Lab engineers. This is so the software and microcomputer can be joined as seamlessly as possible, without my presence in Stockholm.

I would like to thank Jim Gaston of Duke University for selecting me for this amazing opportunity. It has been quite an adventure, as well as an extremely beneficial experience for my education and future career in the engineering field. Thank you to Nenad Glodic as well, for taking time out of his busy schedule to mentor me, and giving me the responsibility of such an interesting project.

If there are ever any questions that arise regarding *Turbojet_Virtual_Dashboard.vi*, please do not hesitate to contact me at zachary.wiener@duke.edu.