Internet Data Acquisition

Matt Matoushek
Invocon, Inc.
mmatoushek@invocon.com

Abstract

This paper shall present the results thus far of Invocon's Small Business Innovation Research (SBIR) Phase II project for NASA Stennis Space Center entitled the Wireless Ethernet-based Data Acquisition System (WEBDAS), as well as describe two flight system developments underway for the International Space Station based on this technology; EWIS and HDMAX. Our recent efforts have focused on the concept of enhancing wireless data acquisition by making collected data available to a user or feedback system through standardized IT resources over the Internet, Internet Data Acquisition (IDA). The goals of IDA are three fold; Hardware Modularity, Software Modularity, and Pervasiveness of Data. The design of IDA has created a three prong attack for developing a quick to market data integration solution; Core Application and Services, Web Application, and a Sound Documentation Methodology.

1. IDA Born from WEBDAS

The SBIR Phase II program WEBDAS, Wireless Ethernet-based Data Acquisition System, commenced in January of 2003. The WEBDAS goals of Hardware Modularity, Software Modularity and Pervasiveness of data were to be met through the network topology shown in Figure 1. WEBDAS network.

Existing Invocon micro sensor systems such as MicroWIS-XG (http://www.invocon.com/MicroWIS-XG_tech_overview.html) and MITEWIS (http://www.invocon.com/MITE_WIS.html) were the instrument of choice due to the facilities maintenance requirements levied by the WEBDAS project of small size, low power, and programmable acquisition rates. Within the WEBDAS network the sensor units (complete with transducer, front end data capture electronics, data processor and radio) communicate with the Network Interface Unit (NIU).

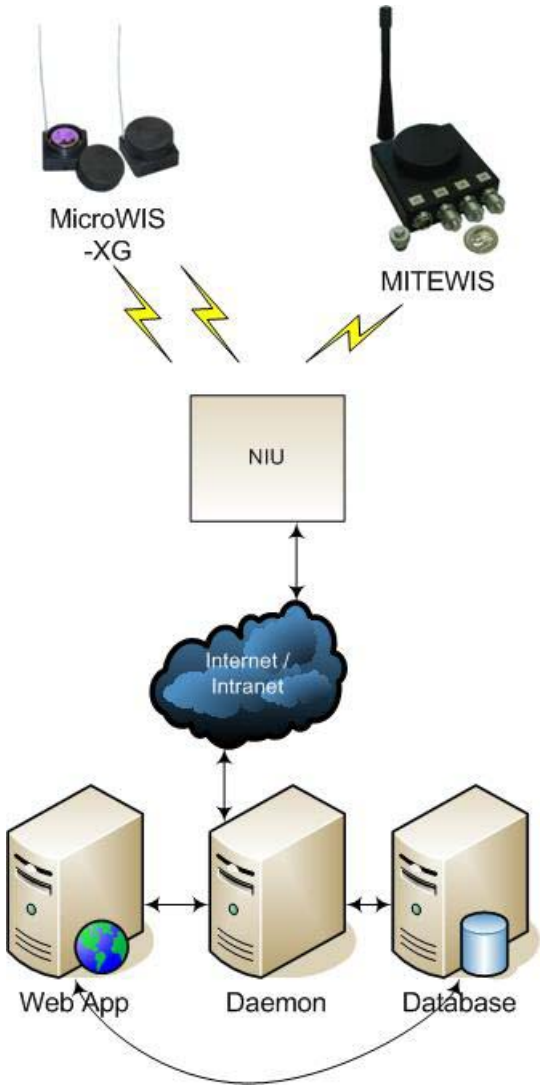

Figure 1. WEBDAS network

The NIU is responsible for gathering the data measured by the sensors, correlating the data to a known time base if necessary, and storing the data until such time that a transmission of the data to the Daemon is appropriate. Upon transmission of the data, the NIU must make a new, or utilize an existing, Internet

connection to the WEBDAS Daemon. In Figure 1. WEBDAS network, this connection is made through the cloud of the Internet / Intranet. This depicts that the NIU's view of the world is limited only by physical access to the Internet, that is anywhere Ethernet, Wireless, Cellular or Space Telemetry permits, data can be propagated.

The Daemon is responsible for assimilating the data collected from all NIUs, and subsequently, all sensors into a relational database for easy access to an end user or customer. The Daemon, as well as the remaining server components of the WEBDAS network, are depicted as separate machines in Figure 1. WEBDAS network to show the flexibility and scalability of the servers' architectures, while in fact, the server components are software entities. For a given problem, these entities (Daemon, Database and Web App) could coincide on the same hardware. The Web App is indicative of the users' gateway into the data collection, viewing and reporting, as well as acquisition mode commanding, of the remote sensor networks.

The WEBDAS network can be viewed as a multi-channel remote data collector, with varying levels of multiplexing, and decoding of data throughout the necessary communication links. The handling of the data flow along these links, and the desire to create a robust, re-usable architecture, one that can operate on "middle-man" hardware such as the NIU, and servers such as the Daemon has led to the design and development of Internet Data Acquisition as a cross platform, multi-threaded, scalable C++ architecture.

2. IDA Core design

At the heart of the IDA design is the Application Core library, or Appcore. The Appcore, a static C++ library provides a base implementation of the following services:

- Thread handling
- Safe inter-thread communication and memory access
- Communication link abstraction
  - TCP clients and servers (listeners)
  - UDP clients and servers (listeners)
  - UART / serial port access
- Packeting and handling
  - Communication link packet reception and transmission
  - Byte stream packet determination
- DLL Management
- Logging, time functions and exception handling

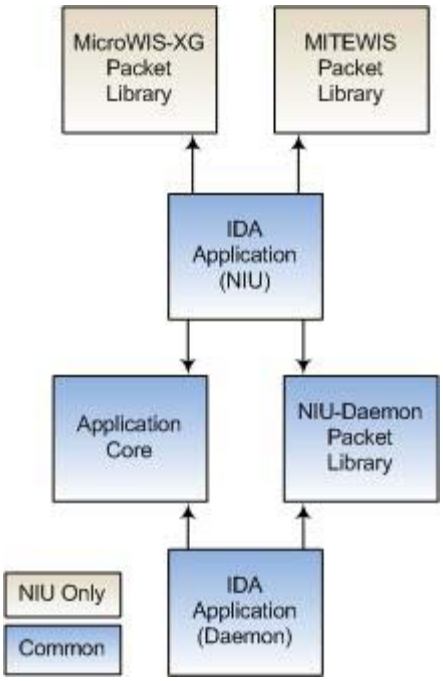

Figure 2. IDA communication link components

The Application Core (Appcore) can be seen situated between the IDA Application for the NIU and the Daemon of Figure 2. IDA communication link components, highlighted in the darker tint of a common component. The IDA Application (IDAApp) provides the most common run-time instantiation of the Appcore. The IDAApp improves the core functionality with the loading of global initialization parameters, as well as dynamically loading communication link threads and general service threads, as specified in the XML configuration file.

The actual connection type (socket, serial port, etc) is not depicted in Figure 2. IDA communication link components, for the protocols set out in the Packet Libraries, are capable of utilizing the Appcore's abstraction of communication links. Therefore, the Packet Libraries are capable of operating over any media and can adapt to changing networking and wireless topologies. The communication links and threads are specified through DLL (dynamic link library), or shared object, interfaces that are provided by the Appcore and are well documented throughout the IDA system.

3. WEBDAS NIU as an IDAApp

Providing the WEBDAS NIU as a base example of an IDA Application, there are two communication links loaded for use by the NIU. The first is to communicate with the receiver hardware over a serial port, and the second to handle the communication with

the Daemon, connecting as a TCP client. A trimmed down example configuration file is shown in Figure 3. IDA configuration file example.

```
<?xml version="1.0" ?>
<config>
  <lib_path>.</lib_path>
  <state_controller lib_name="libtl_sc_ims_niu.so">
    <init_param name="niu_id" value="1" />
    <init_param name="daemon_id" value="2000" />
    <serial_link comm_thread_op="receiver"
                 port_name="/dev/ttyS1">
      <psl lib_name="libpsl_microrf.so">
        <lower_support embedded="false">
          <phl lib_name="libphl_ims_mitewis.so"/>
        </lower_support>
      </psl>
    </serial_link>
    <tcp_client_link comm_thread_op="daemon"
                     port_num="8888"
                     server_name="IP">
      <psl lib_name="libpsl_gp.so">
        <lower_support embedded="false">
          <phl lib_name="libphl_ims_daemon.so" />
        </lower_support>
      </psl>
    </tcp_client_link>
  </state_controller>
  <service_thread lib_name="libtl_ims_storage.so">
    <init_param name="data_dir" value="/data/"/>
  </service_thread>
</config>
```

Figure 3. IDA configuration file example

The serial_link tag element of Figure 3. IDA configuration file example, specifies the physical device to utilize, /dev/ttyS1, as well as the software interfaces to load, libpsl_microrf and libphl_ims_mitewis. The particular implementations of these libraries is left for a different exercise, however libphl_ims_mitewis corresponds to the MITEWIS Packet Library of Figure 2. IDA communication link components. A PSL is a packet support library, while a PHL is a packet handling library. The PSLs generally provide the closest interface with the Appcore, pulling the bytes off of the stream, while the PHL is the most pluggable element to the system, defining protocol specific timeouts, retries, and usage. WEBDAS has the potential to extend to a next generation sensor, for example, MicroWIS-XG, with the simple upload of a new DLL complete with support for MicroWIS-XG packet handling to the NIU.

The tcp_client_link tag element of Figure 3. IDA configuration file example, specifies the abstract device to create a TCP Socket over, through the port_num and server_name parameters. To simulate a telnet connection to a local server, the parameters might read 23 and localhost (127.0.0.1), respectively. The IDAApp and Appcore take care of establishing the client connection, and providing an API for the packet handling library, in this case, libphl_ims_daemon, to send and receive packets over this link. As with the serial_link in the WEBDAS NIU example, the tcp_client_link has a psl name libpsl_gp which pulls the appropriate packets off of the TCP byte stream. It should be noted that the packeting of the IDA communication links is full duplex, supporting transmission and reception of packets that various protocols can adapt to their needs. The communication links will continually check for new bytes on the link, as well as check for new bytes to be transmitted and act accordingly.

The two communication links are linked together through the application specific brain of each IDA instantiation. This central operating loop, or main controller, manifests itself as a State Controller, and is specified through the state_controller tag element of the configuration file. State Controllers implement the Appcore's state controller interface, and can be designed to be as complicated as an advanced Internet router, shuffling data between a plethora of disparate communication links, as simple as a pass through for sensor data, or as a buffer, deciding how long to store sensor data locally before transmitting. The buffer design could be most beneficial when dealing with a high power link on a low power device, such as a cellular interface, where time on the wire equates with significant battery loss. The State Controller is the IDA way of pulling everything together.

If a State Controller needs a proverbial break, there is one last resort for good design within a typical IDA Application. Along side of the state controller entry within the configuration file is the potential to add any number of generic service thread libraries. These threads act as specialized candidates for performing time or media intensive tasks, as well as yet another avenue for code re-utilization throughout differing projects. The most common application for a service thread has been storage; that is, file I/O and file content management. The NIU utilizes its storage thread to store data away, buffering until a user defined forward period is breached and data cleared for transmission.

4. WEBDAS Daemon as an IDAApp

The WEBDAS Daemon architecture can be closely compared to that of the NIU, both utilizing IDA. The Daemon has a State Controller that manipulates the inputs and outputs of its links to the outside world. The Daemon only needs one communication link, a TCP Server, configurable via a tcp_server tag element within an IDA configuration file. The software

protocol libraries for the Daemon precisely mirror those of the NIU, as NIUs are the client to the WEBDAS Daemon.

Where the Daemon dramatically differs from the NIU is two fold. First, in its storage service thread, shown in Figure 4. Daemon database service thread the Deamon gains the capability to query a relational database.

```
<service_thread lib_name="libtl_ims_db.dll">
  <init_param name="database" value="ida_data"/>
  <init_param name="host" value="IP address"/>
  <init_param name="username" value="ida_user"/>
  <init_param name="password" value="ida_pass"/>
</service_thread>
```

Figure 4. Daemon database service thread

Secondly, to the scrutinous reader, the Daemon's database thread library has been specified with a ".dll" extension as opposed to the ".so" extension used for all libraries within the NIU. DLLs are the Windows operating system's shared libraries, while SOs are the Linux operating system's method of dynamically linking code. The Deamon, and the NIU, constructed as IDA applications, are capable of being compiled for either Linux or Windows, as well as cross-compiled for embedded architectures and processors such as ARM and PowerPC. IDA requires only an operating system and standard C++ libraries for operation within an embedded environment, though extensions such as database libraries may only exist for a particular platform or require more horsepower than an embedded architecture can deliver. Thus, all dramatic effect aside, the present configuration example, and WEBDAS deliverable design, is to utilize embedded Linux for the NIU and Windows for a Daemon, quantifying IDA's re-usability, scalability and cross-platform capability.

5. IDA Core documentation

The IDA Core elements and application framework, as described herein, make for a quick turn development bed, while providing robust features for the advanced application. These features cannot be utilized properly by future developers, nor extended through third party development, without proper specifications, guidelines, as well as limitations.

IDA Core Documentation, as well as protocol specifications, sensor control and usages and thread library specifications have enhanced IDA as a whole, and provided a proven methodology for ramping up new developers into a complex architecture. This is of un-calculable value to future developers, as well as the IDA design.

6. Web application

The design in Figure 1. WEBDAS network, coupled with the continued discussion of WEBDAS NIU and Daemon components as IDA applications, has left the Web Application server component (Web App) to the reader's imagination. Taking a cue from the extended design and development of the IDA Core C++ framework, and the critical documentation methodology, the IDA web application has been designed as a modular, pluggable framework as well.

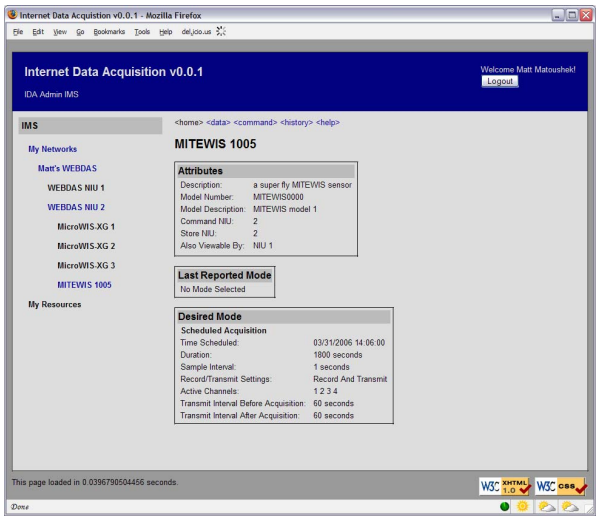

Figure 5. IDA web application

The core IDA web application, portrayed in Figure 5. IDA web application, offers services to an IDA web developer consisting of the following:

- User configuration and management
- Database abstraction and connection
- Extensible module framework
- Communication link abstraction
  - TCP Client
- Packeting and handling
- Logging, time functions and exception handling

Considering these base functionalities of the IDA web application, and the operational differences working within a web site framework, as opposed to a traditional application, WEBDAS required a need provide one further level of abstraction. This abstraction level allows for pluggable sensor types without the need for re-inventing the data transport mechanism. Thus, the WEBDAS web application is implemented as a specific instance of the IMS (Invocon Micro System) IDA Module.

The IMS, and the associated IMS database, have been designed to handle generic core concepts of data acquisition (data format, command format, calibration and conversion equation information), and provides a plug-in interface for sensor specific implementation of these concepts.

7. IDA in summary

Internet Data Acquisition (IDA) consists of core application and services, a unique web application, and a sound documentation methodology. These key elements, coupled with Invocon's unique market position, capable of providing space qualified electronics equipment, efficiently, within a quality environment, makes IDA a powerful tool for the IT and data acquisition communities commercially as well as for flight.

8. Flight developments

Flight developments have benefited from the power of IDA and utilized the core elements in developing the EWIS NCU, the HDMAX CCB, and the BioNet CO2 HAB.

8.1. External Wireless Instrumentation System (EWIS)

EWIS is an accelerometer network for micro-gravity data acquisition on the external outboard solar array trusses of the International Space Station. The Network Control Unit (NCU) of the EWIS network performs the same function as the NIU of the WEBDAS network of Figure 1. WEBDAS network, accumulating data from the Remote Sensor Units (RSUs) and propagating to the ISS network. The RSUs of the EWIS network communicate with the NCU using a 900 Mhz radio. The IDAApp of the NCU utilizes a custom IDA communication link to establish a connection with the radio driver. The NCU also utilizes an IDA serial communication link to communicate with the power supply and a number of service threads to handle data storage and command retrieval to and from an on-board ISS server.

8.2. HDMAX - Ultra High Definition Video Recorder for ISS

The HDMAX payload, an Ultra High Definition (Quad HD) Video Recorder, deployed as an ExPRESS Rack payload has been built on top of IDA. The Communication Control Board (CCB) of the HDMAX payload is a PC/104 Intel ARM based board that utilizes IDA to control the third party components utilizing the mechanisms as prescribed in the matrix of Table 1. HDMAX links. Each link utilizes a custom protocol, supplied by the particular component vendors, that have been implemented as modular IDA packet libraries.

Table 1. HDMAX links

Component	Communication Link Type
Florida Atlantic University (FAU) HDMAX Camera	TCP Client
S.TWO Corporation Digital Data Recorder	UDP Client
Invocon, Inc. Audio Daughter Board	UART / Serial Port
Invocon, Inc. HDMAX Station GUI	TCP Server
ExPedite the PRocessing of Experiments to the Space Station (ExPRESS) Rack	TCP Server

Subsequently, the Station GUI and the Ground GUI were developed on top of the IDA framework.

8.3. BioNet CO2 HAB

IDA conceptually provides middle-ware architectural elements to Invocon sensor controllers. During BioServe SBIR Phase I, to showcase the flexibility of IDA, an IDA application that collected data from CO2 Sensors (http://www.invocon.com/MicroWIS-CO2_tech_overview.html) was incorporated as a Hardware Abstractor (HAB) to Colorado University's BioNet (http://www.colorado.edu/engineering/BioServe/index.html) middle-ware.

9. Future space application

Potential near-term space applications of the described Information Technology developments include vehicle health monitoring of the Shuttle and CEV, structural and environmental monitoring in and around the International Space Station (ISS), and crew medical monitoring. As part of NASA's Exploration initiatives, such networks could easily be deployed on the Moon and Mars, to provide imagery collection, crew communication, remote sensing nodes for scientific applications, robotic command and control, and environmental and safety monitoring of crew habitats. By enabling NIUs and Sensors to provide data through standardized IT resources, the data

acquisition and IT communities can utilize existing infrastructures to enhance the acquisition and processing of remote data. Their data becomes readily available to anyone, anywhere, even space.