Benthic Imaging Application Notes

Contents

1 Introduction

Starting in 2014, development of a rework to the benthic imaging payload began, intended on expanding the functionality of the system to full duration missions. Along with this improvement came a need for better internal storage, improved power electronics, and inclusion of a more intelligent subsea controller. This project is near completion and this document hopes to serve as an information brief summating improvement and provide references for all aspects of the existing design.

2 Technical Overview

2.1 System Overview

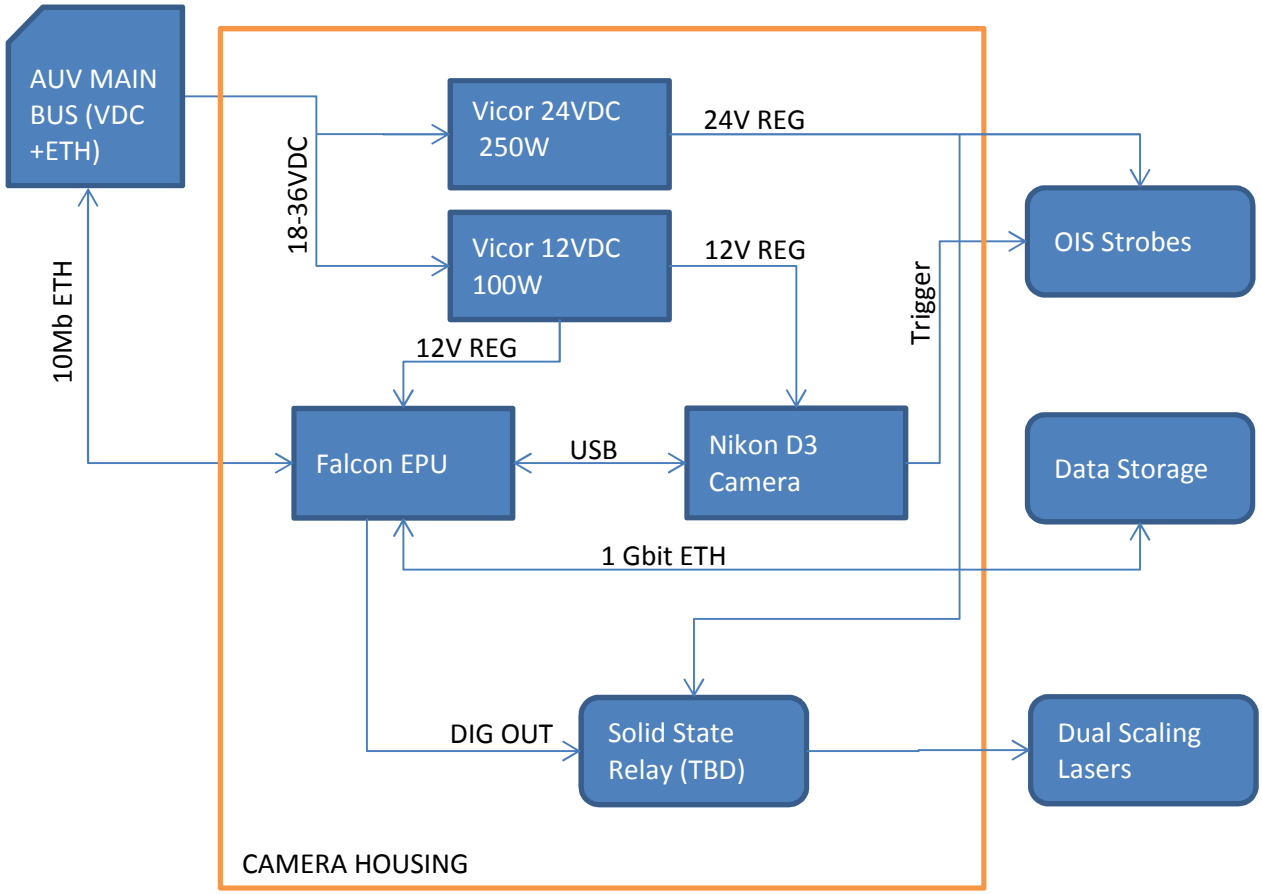


Figure 1: System component overview.

2.2 Design Improvements

2.2.1 Increasing storage capacity

Previous designs stored all data from the camera on internal compact flash storage. This was problematic because available storage was not sufficient to record data at maximum rates for a mission length under full battery charge. Also, added trouble was that the data could only be downloaded via USB interface to the camera. In order to

satisfy both of these conditions and simplify electronics within the housing, a single embedded processor unit from Versalogic, the Falcon EPU, was selected to interface with the camera, providing command and control to the camera body through the USB interface, downloading images to solid state storage attached to a mSATA interface.

2.2.2 Gigabit Connectivity

The Falcon EPU has a gigabit network interface, which should improve post-mission download times. A custom designed carrier board provides a second, 10/100mbit network interface, used for low-bandwidth communication to the rest of the AUV, most notably drivers running in the Main Vehicle Controller (MVC).

2.2.3 Upgraded DC/DC Electronics

MBARI has developed new, wider voltage input compatible Vicor interface boards. Leveraging these improves reliability of power supply to the imaging strobe ballast. These new boards have been tested to 240W load on 24V input successfully in the laboratory. Final testing of this power stage requires hookup to the actual strobe ballast load, and is yet to be conducted.

2.2.4 Fully Autonomous Capability

Because a fully functional embedded processor running Windows 7 was a requirement to reliable interaction with the camera, the camera itself now carries enough computational power to operate nearly autonomously. The software has been designed so that simple network interactions command the camera, strobes, and lasers, including constant rate, user-defined, periodic acquisition. Simple modifications to the software could expand timing functions to allow for delayed start and conditional acquisition as well.

3 Electrical Modifications

In the process of this upgrade, switching to a USB based control versus communication through digital IO combined with RS232, many items could be removed from the original camera housing. As a result, the entire control board, carrying the Matchport and supporting electronics, were deemed unnecessary. This large board was removed to make room for the components to be installed.

As previously mentioned, the Versalogic Falcon EPU was selected to be the central control and communications unit within the housing. A custom fan-out board was designed to breakout all of the major peripheral items into field-serviceable and highly-retentive connectors. This fan-out board also attached a USB 100mBit network adaptor, providing a low-bandwidth communication to the MVC for control.

The gigabit connector onboard the core falcon EPU has been connected to a gigabit-certified Subconn wet mate-able connecter (DBH8M). This connection should enable on deck download of the image post-dive. Either connection to the Falcon should support Windows Remote Desktop.

Intended for later specification and installation is a solid-state relay as controlled by the optically isolated transistors on the fan-out board. As the lasers have not yet been selected, it would be pre-emptive to select a relay at this time.

Standard fuse blocks have been installed to house 3AG size fuses. These fuses protect the input to the 12V DC/DC and the output regulated 24VDC to the strobe ballasts. The new Vicor mini carrier board features fuse protection for the input on-board.

Please see the schematics for further detail. They can be found at:

\\atlas\ProjectLibrary\368000_DMO_AUV\Documents\Benthic Imaging\Electrical

4 Mechanical Modifications

Aside from the afore-mentioned bulkhead change, the mechanical changes to the system were very small. New tapped holes were laid out on the the mounting plate to accommodate the Falcon EPU and the 3AG fuseholders. Updated assemblies have been moved to the following location:

\\atlas\ProjectLibrary\368000_DMO_AUV\Documents\Benthic Imaging\Mechanical

5 Software Description

The following software description is a copy of the README.md file found on the bitbucket.org/mbari/nikonserver website. Please refer there for the latest developments and documentation. The version at the time of this writing is tagged at version 1.0 in the repository.

5.1 Introduction

The Nikon D3 application suite is a set of tools developed at the Monterey Bay Aquarium Research Institute in order to interact with Nikon D3 model DSLR camera. These are all win32 console applications and build currently under MS Visual Studio 2012 for Desktop. Included solution files will properly build and debug all programs.

To checkout a copy of this program using git, please visit https://bitbucket.org/mbari/nikonserver for instruction and http link to clone against. A copy has also been placed in the following location:

```
\\atlas\ProjectLibrary\368000_DMO_AUV\Software\nikon
```

5.3 Applications

The suite consists of three applications:

- **nikonD3Test**: a standalone program test unit for menu based camera interaction for camera functionality testing purposes.
- **nikonD3Server**: a multi-threaded tcp server that also handles hardware triggering through a serial port and camera interaction through USB using PTPP.
- **nikonD3Client**: a tcp client for basic interaction and control of the server.
- **spoggle**: a simple tool to open a serial port and test digital output control. This includes using both the TXD line as a pulse trigger and the RTS handshaking line as a digital output.

Files associated uniquely with those three programs are included in folders by their program name. There are a number of common classes and libraries also included in the root/common folder. There is a common MSVS solution file in the root directory that refers to individual project files in each program subdirectory.

5.4 External Binaries

Nikon's SDK has two precompiled binaries that nikonD3Server and nikonD3Test use and load in runtime. They are `Type0001.md3` and `NkdPTP.dll`. They are located in their project directories and are copied over to their build locations post-build.

5.5 Requirements

Compiling this program requires the use of Microsoft Visual Studio. Initial development was done under *MSVS 2012 for Desktop*.

Runtime requirements for the executable are Windows 7 or newer on a computer providing USB 1.0+ and a single serial port (server only).

5.6 Nikon D3 TCP Socket Interface Control

The `nikonD3Server` application employs a binary communication scheme intended for transmission over TCP sockets. File `nikonD3Messages.h` provides the structures. Each message consists of a fixed width header followed by a variable width message whose length is contained within the header.

The `nikonD3Message` class provides a convenient interface for encoding and decoding these messages within your application.

5.6.1 Table of Values

Below is a table of the format as of January 30th, 2014. Please refer to the source files in case field or commands have been added.

Byte(s)	Type	Contents
0-1	unsigned short	Message Start (0x1801)

Byte(s)	Type	Contents
2	unsigned char	Version
3	unsigned char	NikonCommandType (get/set/error)
4-5	unsigned short	NikonMessageType (message)
6	unsigned char	cameraID (reserved)
7-8	unsigned char[2]	empty (reserved for future use)
9-12	unsigned long	byteCount (msg size to follow)
13-byteCount	variable	message as defined by bytes 4-5 and 9-12

For the data field, a NikonMessageXXXXType should be defined.

5.7 NikonD3Server Console Application

The `nikonD3Server` is a multi-threaded application designed to trigger and download images from a Nikon DSLR (D3 and D700) in an autonomous fashion. Because of the nature of the USB PTPP interactions and their timing, multiple threads have been programmed in order to avoid unresponsive communications and improve overall acquisition speeds. Those threads are as follows:

- main thread: Used for initialization and tcp server
- trigger thread: Used for uart issued camera hardware triggering on a fixed interval.
- camera thread: Used for camera USB PTPP interactions, including configuration and image downloads.

5.7.1 Program Usage

The `nikonD3Server` is configured from a windows-style INI file. Program execution will not occur if the file is not specified as an arguments (i.e. `nikonD3Server.exe <ini filename>`) or if the default file `nikonD3Server.ini` is not present in the working directory.

5.7.2 Operation

On program start the ini file is loaded, and an attempt to find the camera on the USB bus occurs. Also a "syslog"-style log file is written out as specified in the INI file. One note on the INI file, proper escapes for backslashes must exist in any file path fields in order to work properly.

There are three main commands the user will want to interact with the server during runtime via the TCP protocol:

- Camera Connection: Checking or changing the connection status to the camera.
- Acquisition Interval: The time period when frame acquisition is hardware triggered to the camera.
- Acquisition Enabling: Start or stop acquiring.

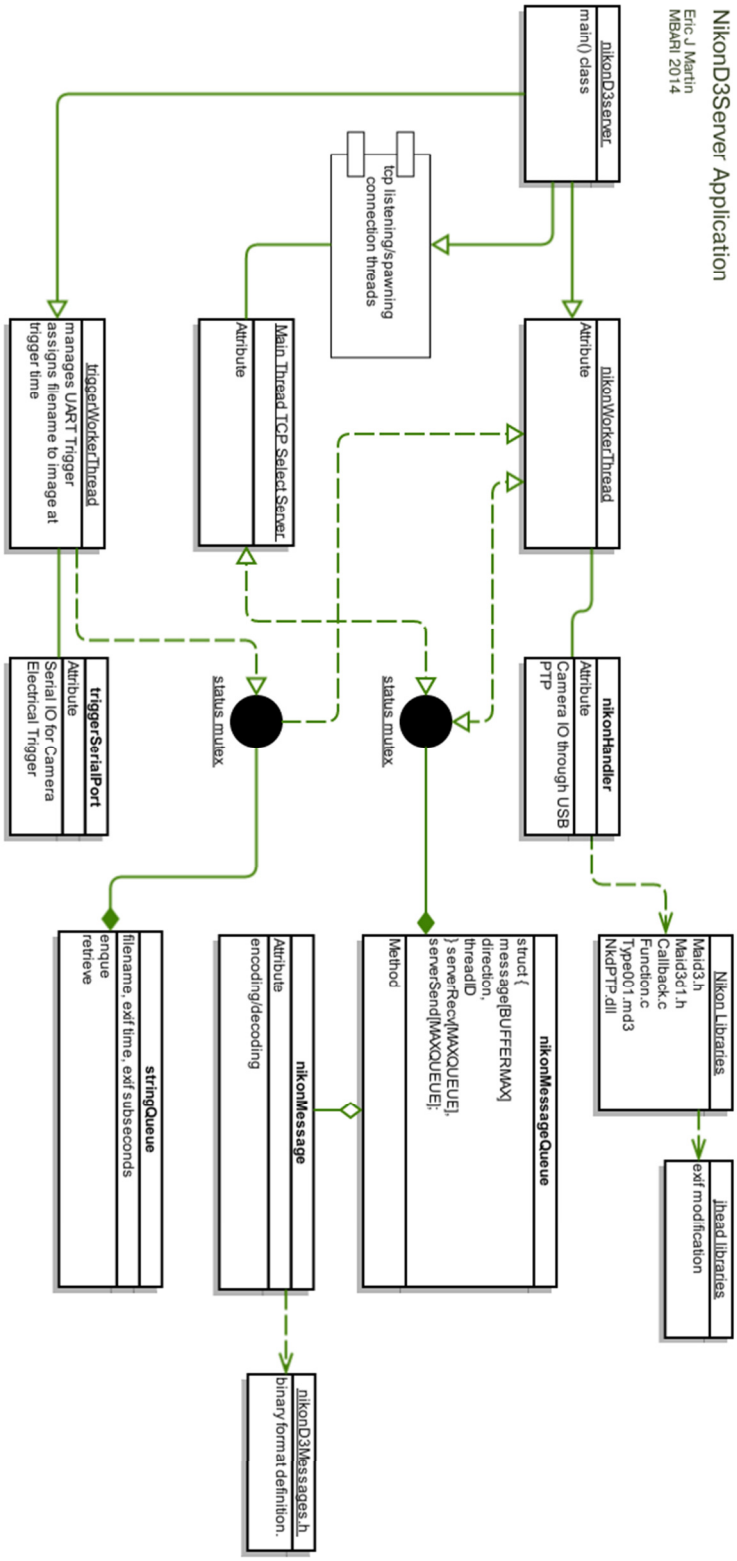
When images are asked to acquire more quickly than they can be downloaded from the camera, the DRAM onboard the camera begins to store the images not yet downloaded. The program will only allow 15 triggers

without image downloads. However, the ability to separate triggering and acquisition allows for smaller intervals because the staggered downloads de-linearize the acquisition steps and allow for downloads from DRAM while another images is being read in from the sensor. For this reason it is important for the user to note the differences in frame download counts and trigger counts. Too large of a difference implies lost images, false triggering, or a problem in the camera.

5.7.3 Requirements

Compiling this program requires the use of Microsoft Visual Studio. Initial development was done under MSVS 2012 for Desktop. Runtime requirements for the executable are Windows 7 or newer on a computer providing USB 1.0+ and a two serial ports (server only).

5.7.4 Program Flowchart



5.7.5 Sample INI File

```
# Sample INI FILE
# ---------------------------------
#   file: nikonD3Server.ini
#   author: Eric J Martin
#           2014 MBARI
# ---------------------------------

# SERVER SETTINGS
[server]

# port is any avaiable tcp port on your machine.
port=2020

# currently the code only supports one client
maxclients=1

# this is the file name and path for the operating syslog style log.
logfilename="E:\\\data\nikonD3server.log"

# CAMERA SETTINGS
[camera]

# This setting is deprecated
pretrigger=0

# Initial value for image triggering interval.
interval=5000

# This prefix should also contain the absolute path for destination
# images. The prefix is appended with the trigger time and date as well.
fileprefix="E:\\\\data\"

# TRIGGER SETTINGS
[trigger]

# string of the comm port name windows serial only.
serialport="COM2"

# baudrate in in bps, only standard rates are recognized.
baudrate=600

# bits per character: only 6/7/8
bits=8
# parity: string with E/O/N
parity="N"
# stop bits: 1 or 0
stop=1

# DIGITAL OUT
[laserpower]

# string of the comm port name windows only. The UART must have handshaking lines.
serialport="COM1"

# baudrate in in bps, only standard rates are recognized.
baudrate=600

# bits per character: only 6/7/8
bits=8
# parity: string with E/O/N
parity="N"
# stop bits: 1 or 0
stop=1
```

5.8 NikonD3Client Console Application

The `nikonD3Client` is win32 console application which runs a very simple tcp client and a set of textual commands to interact with the `nikonD3Server` program. This program is intended to be an example for someone writing a more autonomous driver to interact with the `nikonD3Server`. A

5.8.1 Program Usage

Typical usage of the program is as follows:

```
nikonD3Client.exe <server ip> <server port>
```

Where `<server ip>` is an IPv4 address or dynamic name that can be resolved. The `<server port>` is an integer representing the TCP listening socket on the server side.

5.8.2 Runtime Commands

There are only 8 commands currently implemented, and are outlined in the code block below:

```
-----------------------------------------------
n      - Send NONE message to test communication
c[0,1] - Tell System to Connect
a[0,1] - start acquisition
l[0,1] – Test secondary DO to control laser relay power
i[X]   - set interval to X milliseconds
tc     - retrieve trigger count
fc     - retrieve frame count
x      - Exit program
-----------------------------------------------
```

In the case there is a `[0,1]` annotation, 1 activates the property and 0 disables it. An `[X]` requires that the command be directly followed by an integer.

As this is a sample program, the TCP client aspects of it are coded in a rather rudimentary way. If the server fails to respond to any command, the program will hang and you will have to escape out of it and reconnect.

5.9 NikonD3Test Application

The NikonD3Test application is a slightly modified version of the test sample program that ships with the Nikon SDK. These modifications are small, and streamline use for a single camera application and faster frame acquisition.

5.9.1 Usage

The program is a keyboard interactive console application. Use should be fairly self-evident.

5.10 Spoggle Console Application

A contraction of "serial port toggle", this simple console application is a menu driven application used to test use of your serial ports for use as digital output. In the context of the Nikon D3 upgrades, to minimize electronics, the Falcon EPU fan-out board included an optoisolated pair of transistors intended to be used to trigger the camera capture in hardware. The first opto set is attached to the transmit lines of one UART. The second trigger is attached to a handshaking (RTS) line on another serial port, which can be used for a monostate purpose, such as controlling a solid state switch used to power external devices.

5.10.1 Program Usage

The application has only one option, used to define the device to attach to, usually a COM<x> designation.

```
spoggle -d <COMX>
```

5.10.2 Requirements

This code was built using Microsoft Visual Studio Express 2013 and is compiled for Win32 machines running Windows 7 or newer.

6 Remaining Tasks

The following list outlines tasks identified to be completed previous to the next deployment of the system:

- **DBH8M Bulkhead Installation:** Expected delivery of the connector is by the middle of February, and installation should only take 2 hours including final assembly and a vacuum test setup.
- **MVC QNX Driver:** Already started and checked in, the driver requires a lot of work for full installation. Also a Nikon behavior needs to be prototyped. 10-15 hours to complete.
- **Deck Cable Construction**
- **Network configuration:** The falcon currently uses DHCP to retrieve an IP address. The machine name is currently set to BIAUV-CAMERA-2. Login is user “biauv-camera-a\dorado1” and password is the standard for the username.
- **Laser Selection & Appropriate Relay Selection**