Novel Control System for Robotic Devices via USB

Blake Hannaford, Jesse Dosher, Arash Aminpour, Ken Fodero, Hawkeye King

INTRODUCTION:

As the number and complexity of projects in the Bio-Robotics Laboratory has increased, so have our control system needs. Most importantly we need an I/O platform that offers easy mobility and scales easily to higher degrees of freedom. The requirements for our system include lightweight cabling, portability among robotics applications, high-bandwith and low-latency (data refresh rates of 1kHz, minimum). Our solution is the BRL FHD3.1 board based on a USB 2.0 connection to a Linux host. USB gives us high-speed bandwith enough to control high-degree of freedom robotic systems over a single cable. An on-board processor, custom software and tailored hardware give us a tremendously powerful and versatile control platform. This platform will be the beating heart of many robots we have in the works; it will 'pump' data from encoders, to a Linux host, and out to CAN bus, analog output, Serial, I2C, and can do low level processing along the way.

THE PROBLEM:

While some off-the-shelf control systems were sufficient for our simpler robots, we could not find a solution that would scale to additional degrees of freedom with reasonable cabling requirements. Cables for PCI-based systems added too much bulky weight and were often short, prohibitive tethers (see fig 1). Furthermore, those systems required costly expansion cards and dedicated hardware, unsuitable for use on multiple projects. Perhaps most importantly, the commercial solutions were often provided 'black-box', offering only high-level programming APIs and little or no low-level access to the control hardware. Lack of low-level access meant we could not engineer the most efficient data-paths and perform fast, low-level data manipulation (bit-banging). Thus conventional control platforms were limiting the complexity and fine-tuning of our robots.

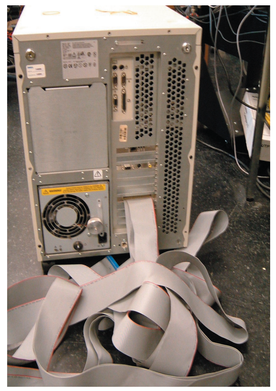

Fig.1 The Problem: PCI control systems required prohibitive cabling and expensive dedicated hardware.

We would have to develop our own control system, customizable to our specific performance needs.

DESIGN REQUIREMENTS:

- Scalable to additional degrees of freedom
- "White box" solution, maximally configurable
- Simple and versitile cabling.
- Capable of controlling many types of robotic systems
- Minimum of dedicated hardware
- "Plug and play" can be taken down, transported and set up without hassle
- High performance suitable for human interfaces (>=1KHz)
- Connect seamlessly to a Linux host for programmable controls software

Fig.2 The BRL control board requires only one USB cable for all communication with control host. The fully programmable board mediates all data I/O.

FHD investigators: Blake Hannaford, Jesse Dosher, Rainer Leuschke, Elizabeth Kurihara and Hawkeye King
Surgical Robot and Pulley board investigators: Blake Hannaford, Jacob Rosen, Denny Trimble, Jesse Dosher, Mitch Lum, Brandon Mander, Tim Ramsey, Arash Aminpour, Ken Fodero

Projects presented on this poster are funded by Samsung, National Science Foundation (NSF), and US Army Medical Research Command (MRMC).

Poster design by Hawkeye King

BOARD DESIGN:

The brain of the BRL board is an 8-bit RISC microprocessor runnnig at 16Mhz. On-board code is developed in C and compiled with processor specific GCC modules. Machine-code is uploaded directly from a Linux host to the on-chip 4KB EEPROM. Programming in C gives us the potential to perform fast computations on the data as it goes in/out. For example we can low-pass filter incoming position data, providing a cleaner signal for the Linux host's kinematics calculations. The processor provides a host of other services including I2C and on-chip ADC converters available should the need for those capabilities arise.

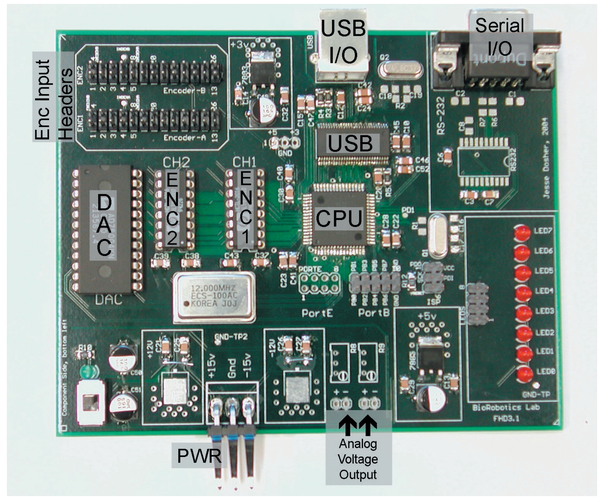

Fig.3: "The Board" version 1.

USB 2.0 is a powerful and versitile data-transfer standard offering a range of speeds and tranfer types, optional data-integrity verification, and the ability to combine those into selectable transfer-mode configurations. Our board can be programatically transformed to any of those configuration states. We chose to use bulk-mode microframe transfers.

A single bulk-mode microframe transfer requires 128 microseconds and carries a 6KB data payload. This timing is sufficient for seven transfers within 1ms timing constraint (1 KHz refresh rate). A 6KB data payload is plenty to carry data for 8 24-bit encoders (192 bits=24 bytes) and/or 16 16-bit DACs (36 bytes) The spare capacity can be used for additional data such as a first derivative (velocity) calculation of each position encoder, board configuration parameters, or any other pertinent data.

Data inputs to the first generation board come from two 16-bit encoder chips, whose value is queried in two sequential reads across the 8-bit bus. Generation two of the control board will accept inputs from 8 24-bit encoders.

Control outputs are via two 12-bit DAC channels outputting zero to five volts. Version two of the board will add a CAN bus networking connection taking advantage of processor-native CAN control to provide more output possibilities. Additional output channels on the current board are useful for debugging: RS-232 (serial) output can connect back to a host for traditional "printf: outputs, and a bank of 8 LEDs can be configured to display a byte of information in lights.

VERSION ONE DESIGN FEATURES:

- 8-bit CPU runs at 16 Mhz
- USB 2.0 configured to run at 53MBps broken into 125us "microframes" carrying payloads of 6KB.
- 2-channel 12-bit D/A converter outputs two 0-5V signals
- 2 16-bit quatrature encoder chips
- DAC and encoder chips run on a single 8-bit data bus. USB module is on a separate 8-bit bus.
- Board has a built-in UART to assist debugging via RS232.

VERSION TWO DESIGN FEATURES:

- Same CPU and USB configuration provides 8-16 independent degrees of freedom.
- 8 24-bit quadrature encoder channels.
- 16 16-bit DAC outputs.
- 16 16-bit A/D converters
- On-board CAN bus connector provides additional output channels

SOFTWARE, DRIVER AND HOST SYNCHRONIZATION:

In order to meet stringent timing requirements of high-performance force-feedback robotics the entire host-based control system executes in a "hard-real-time" environment provided by RTAI (Real-Time Aplication Interface) extensions to the Linux kernel. RTAI Linux ensures code execution at exact times or intervals of time (specified in nanoseconds) by preempting the Linux kernel and taking exclusive control of the processor as necessary. Coding for the RTAI computing context demands adherence to a stringent, high-performance coding paradigm.

Our USB device driver is originally based on the usb_skeleton.c driver- a model Linux driver for USB devices. Rewrites of the driver were necessary to execute bulk-mode microframe transfers. Great effort went into ensuring RTAI compliance, as all sleeping and blocking function calls in the driver and USB subsystem were tracked down and rewritten, removed or avoided.

A similar driver was written for the CAN bus, in preparation for using the built-in CAN functionality of our board. The available open-source CAN drivers provided a user-space API unsuitable for access in kernel-space or from RTAI. Again we had to crack open the black box and re-wire it to our needs.

The control-loop for the system is implemented as a hard real time task-loop that executes with a given frequency (specified at compile time, we've chosen to use 1ms). At the tick of one millisecond the loop begins by sending a USB packet from the host to the board requesting the latest encoder values. Next, we read the position values from the USB bus into a data structure holding all information for all degrees of freedom. From there we call a function calculate_force() that will update force values in our data structure. That function is a modular system; parts of it will be modified for the unique dynamics of each robot. The force value is output to the board and the loop sleeps until the start of the next period.

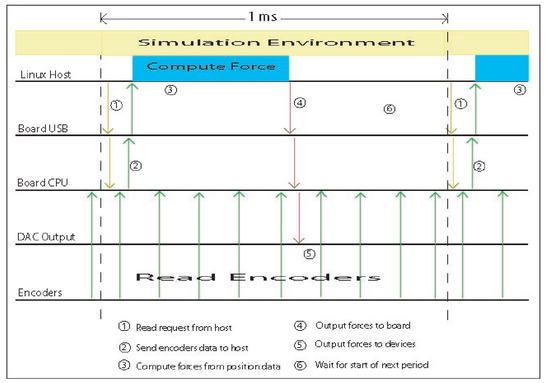

Fig. 4 Control loop timing diagram.

PROJECT STATUS, CURRENT DIRECTIONS:

Version one of the board is complete with all beta-level software in place. Drivers for the system are working in real-time under RTAI Linux . The current control loop is hard coded at compile time, and a control toolkit is under development to allow interactive control of attatched devices from user-space on the PC host.

Version one of the board will be used for controlling two of our Lab's test-platforms, the Pulley Board (see fig. 7) and one two-degree-of-freedom digit of the four digit Fingertip Haptic Display (see fig. 8). Experimental results from those test-platforms will go into the design of the systems to be controlled by version two of the board.

Version two hardware is under development, slated for completion within two months. We are designing it for maximum versatility, and with two robotic systems in mind: the BRL Surgical Robot (See fig. 9) and also the four-digit, eight degree-of-freedom Fingertip Haptic Display (See fig.8). Of those two, the four-finger FHD is ready to be hooked up as soon as the board is finished.

Drivers and real-time control software from the first version will be readily ported to version two. The second version of the user-interface is on the drawing board and will be a VR-type simulated environment showing a 3D model of the control environment.

SYSTEMS UNDER DEVELOPMENT:

At BRL we have several systems already developed or under development that will be controled by versions one and two of our control system.

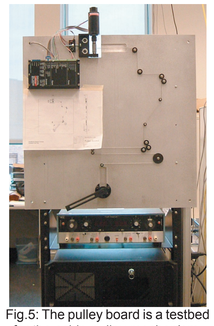

Fig.5: The pulley board is a testbed for the cable-pulley mechanism.

PULLEY BOARD:

The pulleyboard is a one-degree-of-freedom testbed for robotic assemblies. A brushless Maxon DC motor is drives an armature via a pretensioned cable. The cable runs through a series of pulleys between the motor and the armature to simulate the surgical robot's cable-pulley configuration.

This test platform will be used to test various pulley and cable options for the surgical robot. The Pulley board system also provides a simplified robotic system for exploring related problems and as a control development testbed.

BRL SURGICAL ROBOT:

A teleoperated laproscopic device with haptic feedback. 7 degrees of freedom give the robot a spherical range of laproscopic access. The low inertia design uses motors located at a stationary base and connected to axes by pretensioned cables. The robot is divided into two parts, a goss-scale upper arm with a higher-precision lower arm holding laproscopic tools.

Control of the surgical robot's 7 degrees of freedom will be via the BRL control board. Cabling requirements between the base-platform and the PC-controller (with user-interface) will reduce to a single USB cable. This will simplify set-up and use, and make the robot more easily servicable. Complete access to control system hardware will allow fine-tuning of this high-precision robot.

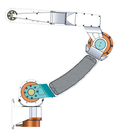

Fig.7- Lower arm is attatched to stationary base

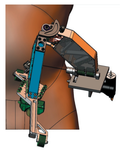

Fig.8: Upper arm.

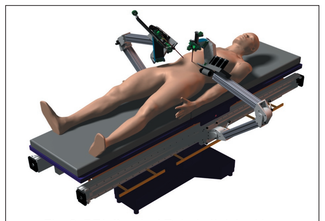

Fig. 6: BRL Surgical Robot attatched to a stationary base.

Fingertip Haptic Display:

The Fingertip Haptic Display (FHD) is a five bar mechanism that is operated with the fingertip of the user. It is designed to accommodate the workspace of the human finger in flexion/extension. The FHD enables touch based user interaction with a virtual environment. The user can essentially feel a virtual object, exploring shape, texture and compliance. Promising applications are palpation training for medical personnel, museum displays: enabling the visitor to "touch" art, as well as psychophysics research: exploring the limits of human touch perception.

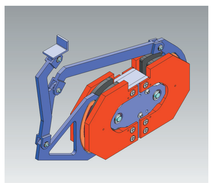

Fig.8: CAD drawing shows a single digit of the FHD

The device uses two voice coil actuators in the armatures to achieve low inertia, low friction and high power capabilities for rendering virtual objects with a high degree of realism. High-resolution position sensors are integrated, allowing four devices to be stacked for use with index, middle, ring and little finger.

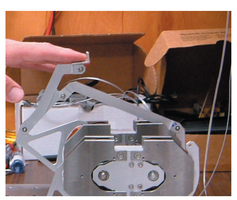

Fig. 9: Rainer gives the FHD the finger

BioRobotics Laboratory
http://brl.ee.washington.edu/