Abstract:

Determine the current state of affairs for the Copernicus robot. Investigate and assess existing hardware and software for the project, prepare Kinect control program for future use with robot. Outline the current status of Copernicus robot and develop a high-level workflow, resource library and toolset for the superseding team, thus furthering the project.

Immediate tasks:

-Install robot body onto roving base.

-Assess motor driver “black-box.”

-Gather missing components.

-Attempt control of base motors.

-Locate, install and configure any existing software needed.

-Powerpoint program for presentation related to Copernicus robot.

-PowerPoint program outlining project.

Attempted:

Leverage software provided by J.R. Kerr website. The software is written for Visual Studio 2003. We attempted to recompile the libraries, DLLs and example projects from source, we were unable to complete this task in the several hours dedicated to it. As we were unable to compile this source into a usable program, we were not able to merge it directly with our Kinect program and thereby control the robot via the Kinect as intended. The motors, serial communication and motor driver boards all tested functional when using the NMC test utility. The NMC test utility program does work on current versions of Windows, however, as it is a \*.exe Windows executable file.

An outline of the required experimental equipment, platform, etc. follows below:

Provided by: J.R Kerr LLC (mfg. of motor controller/RS-232 Serial device)

Software:  
<http://www.jrkerr.com/software.html>

Documentation:

<http://www.jrkerr.com/picsrvsc.pdf> ← (command summary included here)  
<http://www.jrkerr.com/pioboard.pdf>  
<http://www.jrkerr.com/piodata.pdf>  
<http://www.jrkerr.com/boards.html>

Other Software:

Bryan Rudy, 3rd party Linux:  
<http://www.praecogito.com/~brudy/blue_cube/code/blue_move/>

FTDI Drivers:  
<http://www.ftdichip.com/Drivers/D2XX.htm>

Software: (continued)

IDE and compiler toolchain:

Visual Studio 2015 Enterprise→ Install VC++ tools through Setup

RealTerm:

Serial console for Windows. Can be used to transmit/receive bytes with I/O board, capture serial data, etc.

<http://sourceforge.net/projects/realterm/>

Operating System:

Windows 7 Professional (Purely due to Kinect development)

Hardware platform:

-Evolution Robotics servo controller.

-(3 total) Internal boards:

-(1) RS-232 → Serial comm board

-(2) PIC-SERVO v.4, PIC\_ENC v.1 boards

-Saelig RS-232 → USB Serial cable, uses FT232 by FTDI, be sure to use appropriate drivers.

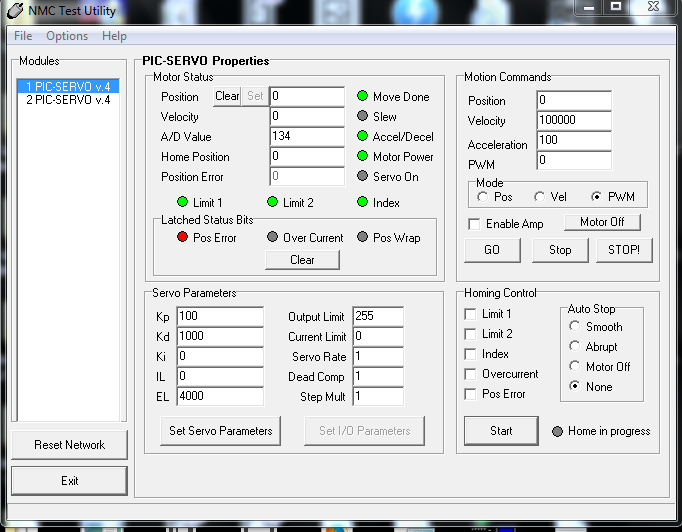
-(2) Motors provided, (mounted on base chassis)

Power for servo control unit:

9V, 3-4 Amp.connector polarity GND -> - ( + )

Using the platform:

Using the NMC Test Utility allows the user to enable the amplifiers, initialize and send motion control signals to the motor drivers, which in turn control the motors, allowing for motion. A step by step outline for using the test utility is provided below with arrows indicating mandatory fields. There is another utility called “stepresponse.exe” if using the stepper drivers. As we are using the PIC-SERVO, not the PIC-STEP, it's usage is not discussed, but was deemed worthy of mention should users intend on using steppers in the future.

NMC Test Utility:

 Servo Modules





Reset Network



Figure 1: Screenshot of NMC Test Utility

**Workflow:**

-Install FTDI drivers, or appropriate for your RS-232 to USB Serial cable.

-Install J.R.Kerr NMC Test Utility, “NMCTest.exe” from repository or link provided in this paper.

-Connect motors to servo controller, they only connect one way. Take note of orientation.

-Connect RS-232 Serial cable to servo controller.

-Plug 12 VDC, 3-4 Amp power-supply barrel plug into servo controller.

-Plug power-supply into wall outlet.

-Connect Serial cable to PC.

-Open NMC Test Utility, you will be prompted for which COM device and baud rate.

-Choose 115200 for baud rate.

\*Note: If you choose incorrectly, simply select the button in the bottom left of the window,

that says, “Reset Network” this will allow you to modify the COM Port/baud.

-Upon successful selection a prompt will state how many modules were found. It should report (2).

-The modules will be displayed in the left-hand column of the application.

-Enable the amplifiers with the check-box under “Motion Commands” on the right.

-PWM values 0-255 are possible, negative numbers, (i.e. -20) are valid and designate reversal.

-By modifying the magnitude of the PWM field under “Motion Commands,” user can alter the velocity.

-Click “Start” in bottom right-hand corner of the Application window.

Troubleshooting Serial communications:

-Ensure device driver is installed correctly.

.Click Start→Search, start typing “Device Manager.”

.Once populated, click “Device Manager.”

.Under “Ports (COM & LPT)” you should see your Serial device, right-click it, select

Troubleshooting: (continued)

properties.

.The opening screen of properties tells you about the driver and it's functionality.

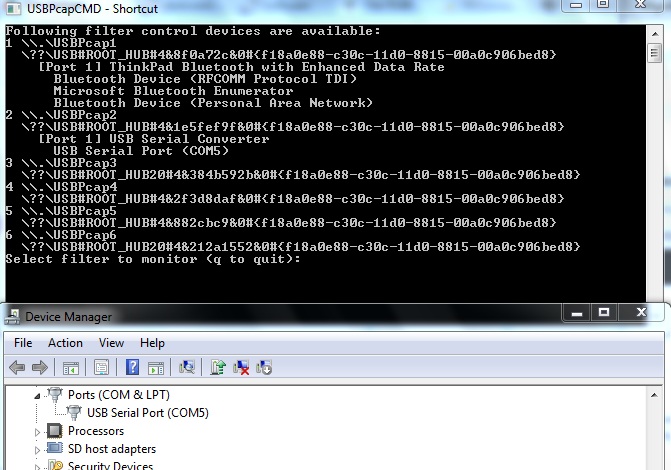
.See the github repository, FTDI link or appropriate for your specific cable.

Now that the motors have been verified, it is very helpful to capture the USB bus traffic. USBPcap is a tool that allows exactly that. Note that after installing, a reboot is necessary so that the program can enumerate the bus. Leave the Serial cable plugged into the PC through the reboot so that it will be included in the process.

Once installed and rebooted, locate “USBPcapCMD.exe,” running this with the serial cable plugged into the computer will open a command prompt window and give you a list. The leftmost column number is the device number. Look in Device Manager, under “Ports(COM & LPT)” The resultant field shows the number to enter into USBPcapCMD prompt as shown below.

USBPcap:

<http://desowin.org/usbpcap/>



User enters '2'

for device on

COM5

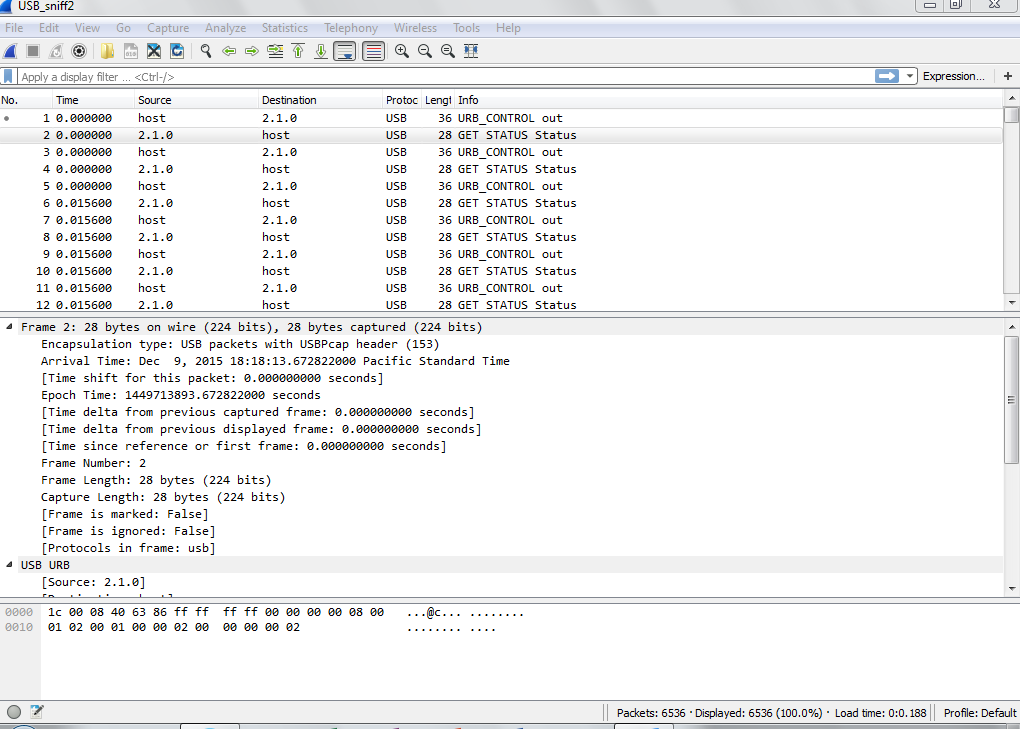




Figure 2: COM Port number discovery.

Once the selection is entered into USBPcap, user is prompted for a filename. Enter it withou any extension. The file will be saved to the USBPcap directory and the capture is now running. Open and run NMC Test Utility, all of the bus traffic for the device will be logged to the filename entered into the command prompt. When finished capturing data, enter <ctrl+c> into the terminal to kill the process. To view the \*.pcap file, Wireshark is used. The program can apply filters, determines directionality and exposes full packet payload.

In the example below, the type of command, direction and data are all nicely outlined for inspection. By capturing this data while sending commands to the device via test utility, etc. algorithmically, the protocol can be discerned even with out any additional tools. This is a worst-case approach but is possible. In a most likely event it would be used to verify communications while debugging, testing and verifying passed datagrams. In a proper format it may be possible to feed captured or crafted sequences to the servo controller via Realterm or a similar Serial console.



 Type and Direction

Packet Data Payload



Figure 3: Analyzing USB packet data captured with USBPcap in Wireshark

The captured data can now be compared against the intended strings by following the command breakdown in the documentation provided by J.R.Kerr. The command summary in the repository is under \*/MOTOR\_DRIVERS/picsrvsc.pdf, or available from the J.R.Kerr website under the software link provided in the Software section of this document. Below is an excerpt of that document explaining the CMD Byte:

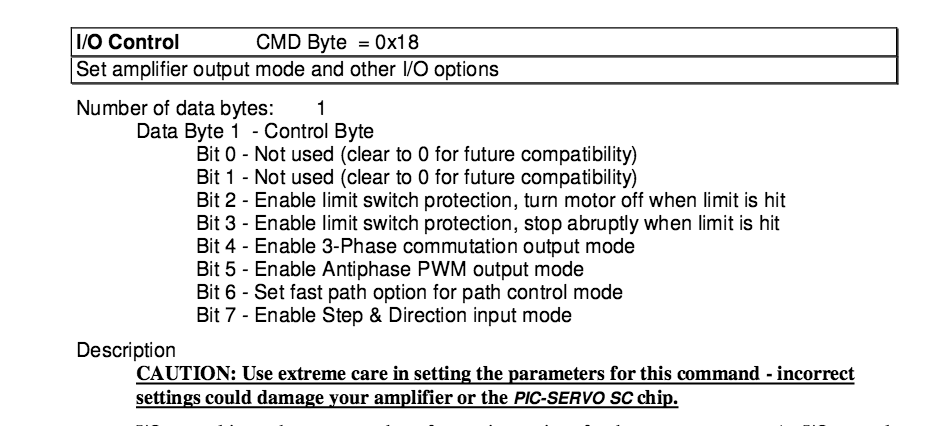


Figure 4: Excerpt from CMD outline in documentation from J.R. Kerr

Results:

We have created operating code for the Kinect using both gestures and transparent button lay-overs to exercise control programmatically. This code can be coupled with any

Running the NMC test program not only allowed us to verify the functional state of the motors and control unit, but also to sample the communications data providing unique insight into the protocols. This advantage, paired with the outlined information in the documentation from the manufacturer this allows for a more complete view of the protocol.

As of current, the robot body is not mounted on the base as it is unclear whether or not the base will indeed be used. However, based on the documentation now located and the to verify the USB communication in transit, we do believe it possible, (though non-trivial), to build an interpreter program and revive the hardware's usability. Given that multiple of these devices exist in the lab, it may prove a worthwhile undertaking. Given this newly assembled information, toolset and workflow, we estimate approximately 40-60 man hours would be necessary for that section of the project to completion, without attempting to repair the deprecated software from the manufacturer. Should software be created this would be an ideal platform for an Intel NUC or similar.