Data acquisition from educational plants over USB in GNU/Linux

Michal Sedlák

Slovak University of Technology, Faculty of Electrical Engineering and Information Technology
Ilkovičova 3, 812 19 Bratislava, Slovak Republic
Tel.: +421 2 60291111 Fax: +421 2 65420415
e-mail: michal.sedlak@stuba.sk

Abstract: The paper describes possibilities of implementation of USB communication in GNU/Linux. It proposes use of USB as data acquisition bus with use of FTDI chip as a serial to USB converter for communication with educational plants. It deals with implementation of communication methods using open source libraries libFTDI, OpenUSB and libUSB.

Keywords: USB, GNU, Linux, Open Source, FOSS, FTDI, libFTDI, OpenUSB, libUSB

1 INTRODUCTION

On site experiments as well as remote experiments are nowadays frequently used in education of automation theory. Real experiments makes students more aware of difficulties that come with implementation of control methods in real environment and real tasks. On the other hand, education on real experiment brings new costs linked with educational plants, which are expensive. To lower the costs is possible by use of virtual experiments, but simulation is bounded with models and that means that it can not fully substitute contact with real experiment. Eminent part of costs for educational plant is data acquisition board, which is required for connection of educational plant with PC. USB DAQ can cost only a fraction of PCI DAQ and are able to save resources[1].

USB DAQ are produced by lot of companies[2][3][4][5][6][7]. If we were able to substitute these costly DAQ cards with some other low-cost communication device that will fulfill the requirements of the educational plants and requirements of the methods taught in the class, we would be able to save resources or we will be able to provide the plants for more students[1]. USB DAQ should be the adequate substitution of communication channel for most of the education plants. USB is implemented in majority of modern PCs or laptops, and in addition it is widely used as well in embedded systems and router boards, or other low cost platforms, which makes it candidate for low-cost on site communication. It is implemented in most operating systems and widely spread. Implementation of USB on router boards and other embedded platforms gives us option for low power consumption plants that can run 24/7 which lowers costs for operation of remote laboratory.

2 PROBLEM STATEMENT

USB is not designed for real-time access. We would like to show that general implementation of USB in GNU/Linux is able to sample fast enough for use with educational plants, where is 1kHz sampling frequency more than satisfactory. With USB as a communication bus for DAQ communication comes a lot of implementation problem. First, we would like to communicate at latencies close to 1 millisecond, that is hardware limit of USB where every communication frame takes 1 millisecond. Implementation of communication close to this limit requires that both communication sides are prepared for this type of load. We have to examine latencies that are caused by operating system as well as linked with the transfer over shared bus.

Next problem of USB communication, is how can we communicate over USB or what device can we communicate with. We want communicate with the educational plants. This type of device is in most cases not prepared for communication over USB.

USB have more options how to communicate, these options are called transfer types[8] or communication modes. Presented communication type is bulk transfer, which is the most commonly used transfer type, for I/O devices like hard drives, or flash drives. [4][9]. Bulk transfer guarantees that sent data will be delivered to recipient, which costs some overhead and latency, but is easy to implement on host side. Most capable transfer type for data acquisition is isochronous transfer type, which is using preallocated bandwidth. That means that we can communicate with plant in every 1ms frame, which gives theoretical round trip latency near to 2 ms.

We have used uDAQ24/T[4] an Arduino[10] board as USB data acquisition device used for tests. Both these devices are communicating over USB with serial to USB converter based on FTDI chip[11].

3 UDAQ24/T AND ARDUINO DUEMILANOVE

Communication over USB is shown in the communication diagram shown in Figure 1. As one can see from the diagram, we have to decide which device to use as an intermediary device to the plant, that is able to communicate over USB and to send action as normalized signals. We chose between older version of thermal plant, uDAQ24/T, and Arduino Duemilanove development board.

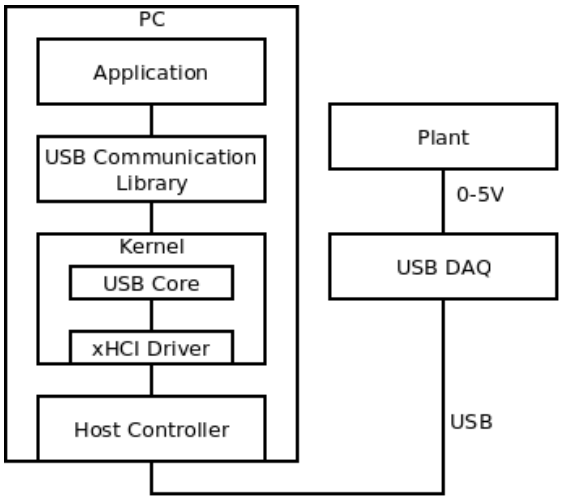

Figure 1: USB communication diagram in GNU/Linux

Arduino is development board, with software tools enabling rapid application development. It is very easy to use development board based on Atmel ATMEGA328 micro controller. It uses FTDI FT232R serial to USB converter chip .

uDAQ24/T is thermo-optical plant based on ATMEGA8 micro processor from Atmel and FT232BM as USB to serial converter. We have removed thermo-optical subsystem and used this device as USB DAQ. We have developed alternative firmware that have implemented functions that were essential for our solution, like attaching of debugging information needed for testing purposes.

We have chosen uDAQ24/T because it has full support of isochronous transfer, which could be used in future and 12 bit D/A converters instead of 10 bit D/A used in Arduino.

4 FTDI AND GNU/LINUX

USB does have four transfer types that could be used for data transfer:

1. isochronous
2. interrupt
3. control
4. bulk

Isochronous transfer is used in application where should be guaranteed timing and bandwidth. USB is not checking if data were transferred. Isochronous transfer have highest priority in transfer frame. On the other hand bulk transfers which are using bandwidth that other types does not consume have implemented acknowledge of correct delivery, but lowest priority in SUB frame. Bulk transfer is mostly used by devices which use to transfer big amounts of data, like data storages. Control type is used for transfer of small amounts of data with acknowledged delivery. It is generally used for configuration of USB device. Every USB device has to have one control end point called “endpoint 0”. Interrupt transfer type is used in case that application use to transfer small amount of data in certain time. Interrupt transfer is used by devices where is responsiveness one of the most important parameters, like human/machine interfaces. USB Transfer type schedule order is shown in Figure 2, USB 2.0 is using more complicated scheduling of transfers [8].

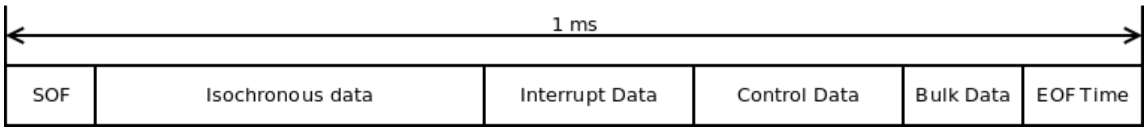

Figure 2: USB Transfer Type Schedule Order in USB 1.1 frame

Low latency communication over USB has to be set up on both end points. We use communication subsystem of real plant, based on FTDI chip, connected to the GNU/Linux operating system. FTDI chip is USB to serial converter with full implementation of USB protocol. FTDI chip FT232BM supports both bulk and isochronous transfers so it fulfill our requirements. Chip is fully configurable over USB connection from host computer.

5 LIBFTDI

User can set all properties of communication with FTDI chip with communication library libFTDI written in C language based on libusb 0,1. LibFTDI is open source library maintained by Intra2net[12]. This library provide very easy access to most of FTDI chips. To open a FTDI device use function `int` ftdi_usb_open (struct ftdi_context *ftdi, int vendor, int product). After that you can use any function from libFTDI, like setting baudrate int ftdi_set_baudrate (struct ftdi_context *ftdi, int baudrate). Unfortunately this library does not have implemented the isochronous type. This led us to writing our own communication libraries. First based on OpenUSB library and second one based on libusb1.0, which are more mature and have big community that is supporting these projects.

6 OPENUSB AND LIBUSB 1.0

Performance of this After implementation of the basic communication libraries we were able to communicate with the device very easily. Both OpenUSB and LibUSB are mature USB communication libraries, its syntax is not compatible, but main difference is how they work with threads. OpenUSB was forked from LibUSB because LibUSB is not thread-safe. Main idea of OpenUSB is to create thread safe communication library. These two libraries are functionally identical, more about differences in [13]

When we were able to communicate with FTDI chip, we have to write program for the device with FTDI chip that enables to make some benchmark. At first we developed simple program that was echoing sent packets. Than we enhanced communication utility. We programmed to send debugging data like microseconds from start of program and packet count to evaluate latencies.

7 EXPERIMENTS

We were testing uDAQ24/T and Arduino Dueilanove directly connected to USB port and over USB hub. We were sending data from PC to USB DAQ every millisecond, when device received the sample it echoed it back. We were using blocking functions for bulk transfers and measuring time of sending and receiving of frame. Histograms of measured latencies are shown in Figure 3

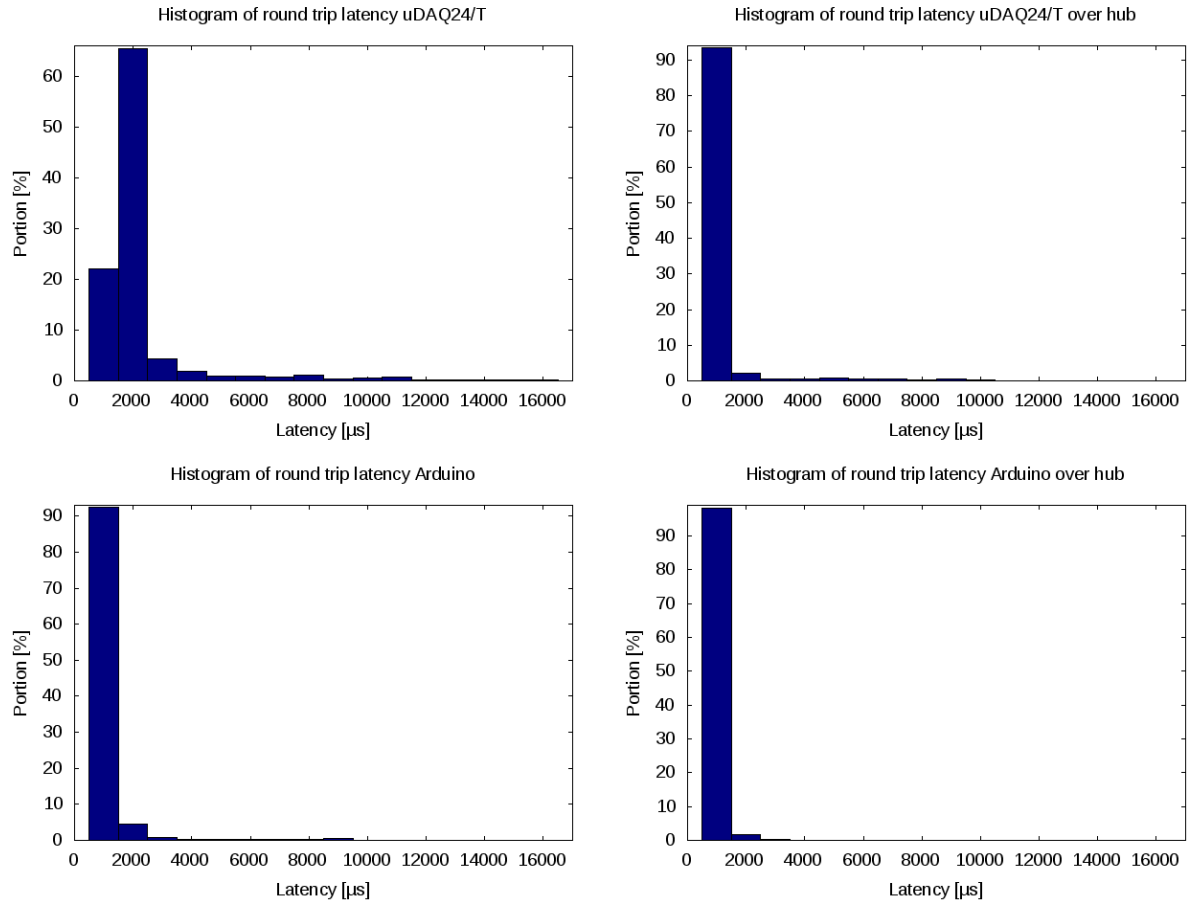

Figure 3: Histograms of measured latencies

From histograms shown in Figure 3 can be seen, that over 90 %of communication is made in 2 ms, as well as that Arduinos FT232R is better tuned for latency that FT232BM which is in the uDAQ24/T. This difference is caused by better buffering and usage of event character[12]. More precise number is shown in Table 1.

Table 1: Comparison of USB latencies

	> 5 ms [%]	> 10 ms [%]	max. latency [µs]
uDAQ24/T	5,72	1,74	43 935
uDAQ24/T+hub	2,70	0,39	25 407
Arduino	1,85	0,31	22 622
Arduino+hub	0,08	0,01	11 733

As you can see in Table 1 maximal latency with Arduino over hub is nearly 12 ms and only 0,01% is more than 10 ms which makes this device suitable for robust and nonlinear control with sampling times over 10 milliseconds. These number can change depending on hardware or load of the system and load of USB.

8 CONCLUSION

For simulation of DAQ we have used uDAQ24/T and Arduino board as USB data acquisition device. Both these devices are communicating over USB with serial to USB converter based on FTDI chip. We have created applications which used for communication wit these USB devices most used USB communication libraries libFTDI, OpenUSB and libUSB 1.0. We have made benchmarks that have shown that USB can be used for real time control with sampling times higher than 10ms not dependent on backend library. Most of the delay was produced by operating system I/O and by FTDI chip buffers. We were able to send and receive small amounts of data, which was lower than 64 bytes per sample, in 10ms with number of lost samples close to zero. These results could be enhanced by use of isochronous USB transfer or preemptive Linux kernel.

9 ACKNOWLEDGEMENT

The work has been partially supported by the Grant KEGA No. 3/7245/09 and by the Grant VEGA No. 1/0656/09. It was also supported by a grant (No. NIL-I-007-d) from Iceland, Liechtenstein and Norway through the EEA Financial Mechanism and the Norwegian Financial Mechanism. This project is also co-financed from the state budget of the Slovak Republic.

REFERENCES

[1] WYNN R. (2005): Plug&play, networked data acquisition systems reduce test costs, *Computing & Control Engineering Journal,Volume: 16 Issue: 1* Feb.-March 2005, Page(s): 23- 25
[2] http://www.bmcm.de/us/pr-usb-pio.html
[3] http://www.linux-usb-daq.co.uk/prod2_usbdux/
[4] SINGH, G.; CONRAD, J.M. (2008): *Easy-to-use communication interfaces for data acquisition, Southeastcon*, IEEE, 3-6 Page(s): 111-116, 10.1109/SECON.2008.4494269
[5] http://iotech.com/products/pdaq3s.htm
[6] http://www.linux-usb-daq.co.uk/
[7] http://labjack.com/u12
[8] http://developer.intel.com/technology/usb/ehcispec.htm
[9] RAMADOSS, L. HUNG, J.Y. (2008): *A study on universal serial bus latency in a real-time control system*, Industrial Electronics,IECON 2008. 34th Annual Conference of IEEE, Pages: 67-72, ISSN: 1553-572X, 10.1109/IECON.2008.4757930
[10] http://www.arduino.cc/
[11] http://www.ftdichip.com
[12] http://www.intra2net.com/en/developer/libFTDI/index.php
[13] http://www.libusb.org/wiki/Libusb1.0