Proceedings of the World Congress on Mechanical, Chemical, and Material Engineering (MCM 2015)
Barcelona, Spain, July 20 - 21, 2015
Paper No. 247

Low-cost RCP System for Control Courses using Matlab and the Open-source Hardware

Young Sam Lee, Yeong Sang Park, Bong Eon Jo, Sugkil Seo
Inha University
Inharo 100, Namku, Nam-ku, Incheon City, Korea
lys@inha.ac.kr; pys0728k@hotmail.co.kr; jbw0803@gmail.com; smooms121@naver.com

Abstract -In this paper, we propose a new and cost-effective rapid control prototyping (RCP) system based on the Arduino Due and Matlab/Simulink. The proposed RCP system has a feature that a computer on which Simulink is running acts as a realtime controller and the Arduino Due performs data acquisition, transmission of the data to and from a computer through a built-in high speed USB interface, and the application of control data received from Simulink. For its implementation, we develop 7 input and output blocks for use with Simulink, all of which are implemented from a single S-function. We also propose a method through which we can overcome the problem arising when the sampling time of the control system is not constant. The proposed RCP system has several advantages over existing methods such as good maintainability, portability due to the USB interface, low cost, and no necessity for C-code generation even though it can only be applied to control systems with moderate sampling rates. It is expected that the proposed RCP system can be useful in teaching control-related topics to undergraduate and graduate students.

Keywords: Arduino Due, Rapid Control Prototyping (RCP), Block libraries, High speed USB interface.

1 Introduction

Rapid control prototyping (RCP) system is a kind of development environment that is used for the design, development, and verification of the controller prototype in an efficient way. Several RCP systems are available commercially in the market. Matlab/Simulink is the most well known and widely used among those systems. Realtime Workshop (RTW), the add-on product of Simulink, generates C-code for the block diagram-based model constructed by Simulink (The Mathworks Inc., 2005*b*). Embedded coder, another add-on product of Simulink, generates C-code specific to a certain embedded processor and thus reduce the time for the development (The Mathworks Inc., 2005*a*). Matlab/Simulink and RTW are open architecture and thus several lab-developed RCP systems for custom-developed hardware have been proposed in academia (Lee, Shin and Sunwoo, 2004; Hercog and Jezernik, 2005; Bucher and Balemi, 2006). Furthermore, several researches related with the application of RCP system are published (Lin, Tseng and Tseng, 2006; Kennel, 2006).

In many engineering colleges, the courses on control are provided to students and those courses include the design of controllers and experiments. Controller design and experiments using RCP systems will save students the tedious and error-prone manual coding and make them focused on the control algorithm itself. This will greatly improve the students' interest on the design and experiments. However, the commercial RCP systems or academically developed RCP systems are based on expensive hardwares and softwares. Furthermore, the hardware lacks in portability. Due to these reasons, take-home experiments for students are hardly possible.

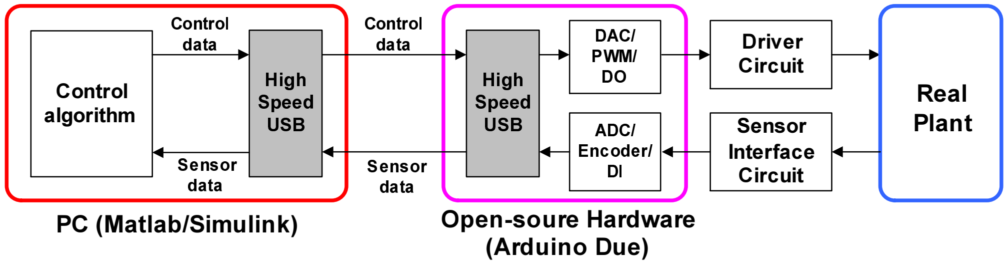

Fig. 1: The conceptual diagram of the proposed RCP system

One good way of making a hardware with good portability is to give it the USB communication interface. Many approaches using DAQ units with the USB interfaces have been published for just gathering data, not for control purpose (Khalil, 2006; Jiang, Ojaruega, Becchetti, Griffin and Torres-Isea, 2011). The USB supports plug-and-play and there is no need to turn off a PC when we connect the hardware to a PC. As a result, the hardware unit with the USB interface has good portability because the connection and disconnection to a PC are so easy (Web 2 : Universal Serial Bus Specification, 2015). However, some existing RCP systems where a PC acts as a controller such as xPC Target do not use a DAQ unit interfaced through the USB. This is because the USB communication is not appropriate for realtime control where a small amount of data is transmitted repeatedly. Considering that there are many experiments for undergraduate or graduate students that do not require high sampling rates, the DAQ unit with the USB interface can be a good candidate for communicating data for realtime control.

C-code generation of the existing RCP systems require expensive softwares like Embedded coder and compilers. If we can construct a controller using Simulink without using code generation, we don't have to use much money to by those softwares. In reality, many engineering colleges already have the basic Matlab and Simulink. Furthermore, this software is usually for concurrent users and therefore students can use the software even at home.

Nowadays, various approaches for education and research using open-source hardware platforms are actively being done around the world. The Arduino Due, which has been recently released, is an open-source microcontroller board based on Cortex-M3 CPU (Web 1 : Arduino Due, 2015). It has plentiful input and output peripherals that can be used for data acquisition. Furthermore, it has the built-in high speed USB interface. Therefore, the open-source hardware platform like the Arduino Due can be a good candidate that can be used for building RCP system that features low-cost and good portability.

In this paper, we propose a new RCP system that can solve the portability and the cost of the existing RCP systems. For this purpose, we use the Arduino Due with the high speed USB interface and a controller model constructed from Simulink acts as a realtime controller directly.

2 Introduction of the Proposed RCP System

The proposed RCP system consists of two subsystems except the plant system as shown in Figure 1. The first subsystem is a PC system on which Matlab/Simulink is running under MS windows. The second subsystem is an Arduino Due board, which has a built-in high speed USB interface. A PC and an Arduino Due communicate control data and sensor data through the high speed USB interface. In conventional RCP systems, the controller model constructed from Simulink is translated into a C-code by the automatic code generator, compiled, and then downloaded to a embedded control unit. On the other hand, the proposed RCP system does not have a code generation process. Instead, the Simulink controller model directly computes the control value in the proposed RCP system. The way of performing the control action is summarized as follows:

S1: The Arduino Due measures the data needed for control and sends them to the PC through the USB communication.

S2: Simulink receives the sensor data transmitted from the Arduino Due.

S3: Simulink computes the control data using the received sensor data.

S4: Simulink sends the computed control data to the Arduino Due through the USB communication.

S5: The Arduino Due apply the received control data to the output peripherals (ex: DAC, PWM, Digital output).

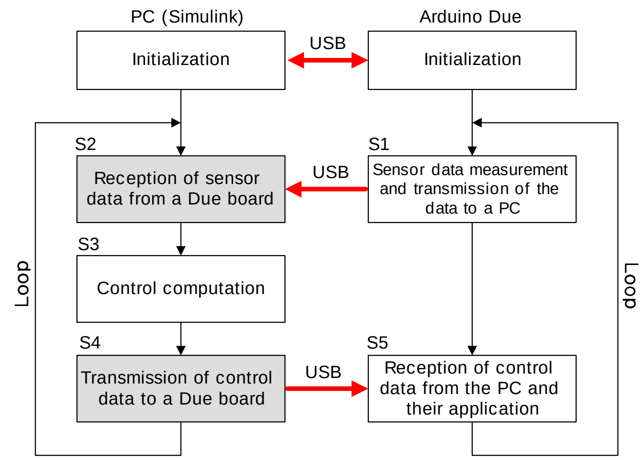

Fig. 2: The flowchart on the proposed RCP system

Figure 2 is the flowchart that explains the operation of the proposed RCP system conceptually. The kinds of sensor data to be received from the Arduino Due and to be transmitted to a PC are specified in the controller model constructed using Simulink. The controller model is built using various built-in blocks provided by Simulink and the input/output blocks provided by the proposed RCP system. The control computation is performed entirely on the PC and the Arduino Due is just responsible for measuring sensor data and applying the control data to the output peripherals. S1 and S5 are performed in the Arduino Due and S2, S3, and S4 are performed on the PC. For the periodic control operation, we utilize the SysTick timer of the Arduino Due. Even though the sample time has a little bit jitter, it is overall periodic, which will be shown later in the paper.

2.1 Arduino Due for Data Acquisition

In this paper, we use the Arduino Due, which is a well-known open-source hardware platform, for data acquisition. The Arduino Due is a microcontroller board based on the Atmel SAM3X8E ARM Cortex-M3 CPU. It is the first Arduino board based on a 32-bit ARM core microcontroller. It has 54 digital input/output pins of which 12 can be used as PWM outputs, 12 analog inputs, 2 DAC, a 84 MHz clock, a built-in native USB port, etc. The reasons why we choose the Arduino Due as a main microcontroller for data acquisition are manifold:

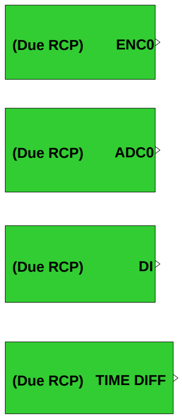

Fig. 3: Input/output blocks provided by the proposed RCP system

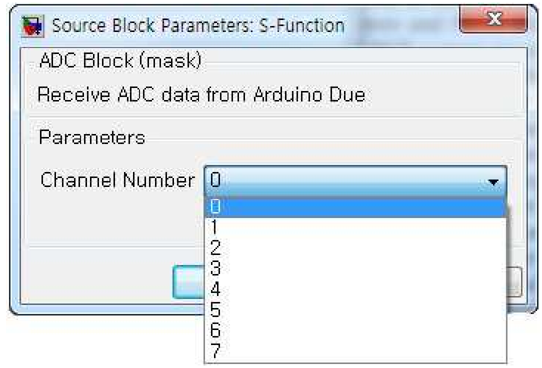

Fig. 4: Channel selection of the ADC block through a dialog box

- It is an open-source hardware platform and hence well-known, widely used, easily accessible, and not that expensive.
- It has plentiful input and output peripherals that can be used for control purposes.
- It provides the built-in high speed USB interface that can transfer data at fast speed.
- A free development environment is available.

In the proposed RCP system, the Arduino Due is not responsible for the control computation. Instead, it only measures the necessary sensor data and then send them to the PC so that Simulik can compute the required control value on the basis of the sensor data and send the computed control data to the Arduino Due to apply the control. The Arduino Due is widely used and its price is not that expensive. In the proposed RCP system, the actual computation of the control is performed on the PC side. Thus, we do not need such a high speed microcontroller. The Arduino Due has a built-in high speed USB interface. As widely known, USB supports the plug-and-play and can be interfaced with any PC easily irrespective of laptops and desktops. Nowadays, the engineering education put much emphasis on experiments and design. Therefore, the Arduino Due with lots of peripherals and USB interface may be the best choice for take-home experiments for students. The proposed RCP system provides 7 input/output blocks that are used to control some important input/output peripherals. By using those blocks, transfer of data to and from the Arduino Due can be easily achieved.

2.2 RCP Input/Output Blocks

The important feature of RCP systems is that they provide input and output abstraction blocks so that controller designers can handle the input and output peripherals easily. Controller designers don't have to spend much time in struggling with handling input/output peripherals. Instead, they have only to focus on the control algorithm itself with the help of the provided input/output blocks. The proposed RCP system provides 7 input and output blocks so that the controller can receive the required sensor data from the Arduino Due and send the computed control data to the Arduino Due within the Simulik environment. Figure 3 shows those input/output blocks. The blocks belonging to the left column are input blocks and the blocks belonging to the right column are output blocks. Blocks that has a number beside the block name are blocks that have multiple channels. For example, the ADC block has 8 channels. If one wants to use other channels, he can double-click the block and then choose the channel number through a dialog box. Figure 4 shows the dialog box of the ADC block activated by double-click.

Among those 7 input/output blocks, PWM block and Time block requires further explanation as follows:

- PWM block: PWM is usually used for controlling electric motors and PWM block supported by the proposed RCP system provides two types of PWM interface, which are PWM/Dir interface and complementary PWM interface. Furthermore, the frequency of PWM can be configured to be 1 KHz or 10 KHz or 20 KHz. Sometimes PWM signals should be synchronized together, which means that those PWM signals are all generated from a single counter. For this purpose, PWM block also provides the option for synchronization. Furthermore, dead time for each PWM channel can also be specified independently. Figure 5 shows the dialog box of PWM block by which all options mentioned above can be specified.

- Time block: Time block can be used for getting the system time from the Arduino Due. Two types of time information can be obtained: time and time difference. Time is the system time itself and time difference is the difference in the system time for the previous sample and the current sample. Figure 6 shows the dialog box of Time block.

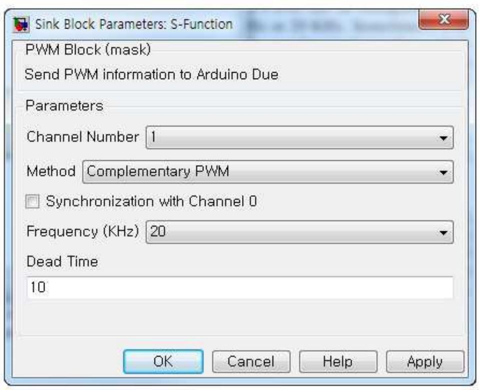

Fig. 5: The dialog box of PWM block

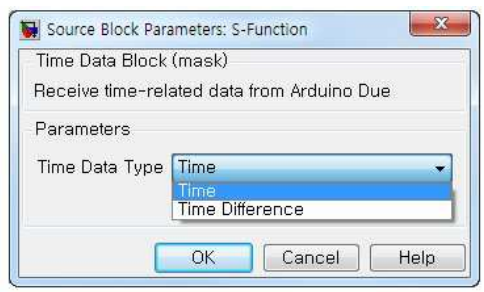

Fig. 6: The dialog box of Time block

Currently, 7 input/output blocks are supported. However, additional input/output blocks are to be developed so that one can construct the control algorithm more flexibly.

3 Realtime Control Experiment Using the Proposed RCP System

In order to check out the usefulness of the proposed RCP system, we constructed a PI velocity control model using the I/O blocks provided by the proposed RCP system. Figure 7 shows the constructed Simulink model. It implements the PI control with anti-windup compensation. The block labeled as 'Maxon Motor' in Figure 8 is a subsystem which consists of several blocks. To get the velocity of a motor, it uses Encoder block with the channel number 1. To generate complementary PWM signals, it uses two PWM blocks with the channel number 0 and 1, respectively. In the controller model given in Figure 7, one can choose a reference trajectory among three candidate references: a periodic square wave, a sinusoidal wave, and a user-driven reference that is generated from ADC value of a potentiometer. The sample rate of the control system is 1 KHz, which is fast enough for controlling the velocity of a motor. It is mentioned that the control algorithm can be easily built by drag-and-drop of function blocks and graphical editing.

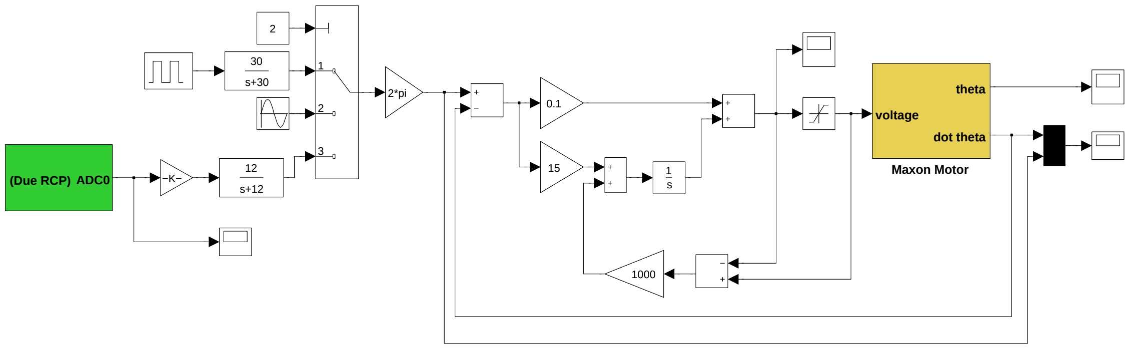

Fig. 7: Simulink model for the PI velocity control of a DC motor constructed using the proposed RCP system

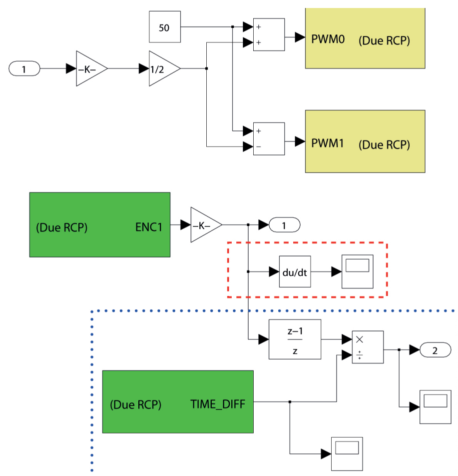

Fig. 8: Construction of a subsystem 'Maxon Motor'

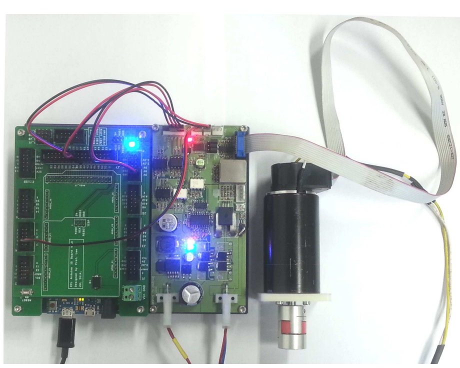

Fig. 9: Experimental setup for a DC motor control

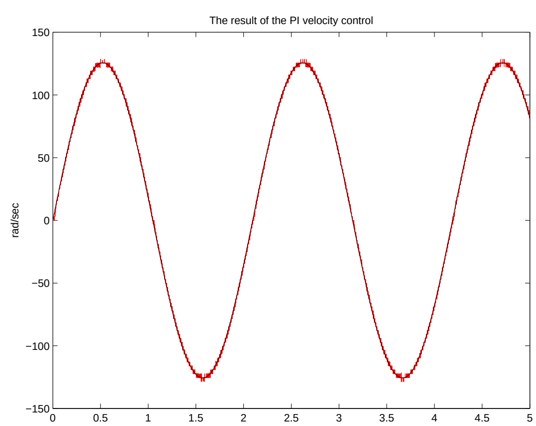

Fig. 10: The result of the PI velocity control implemented using the proposed RCP system.

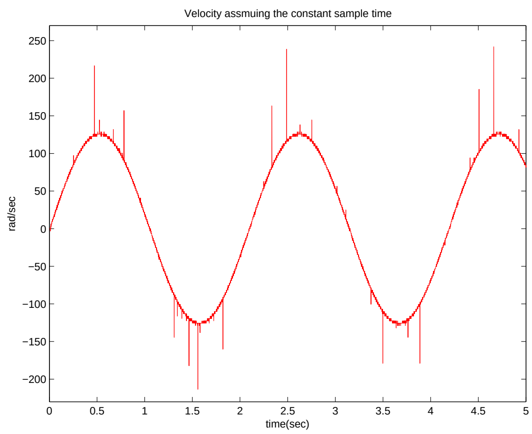

Fig. 11: Velocity assuming that the sample time is constant

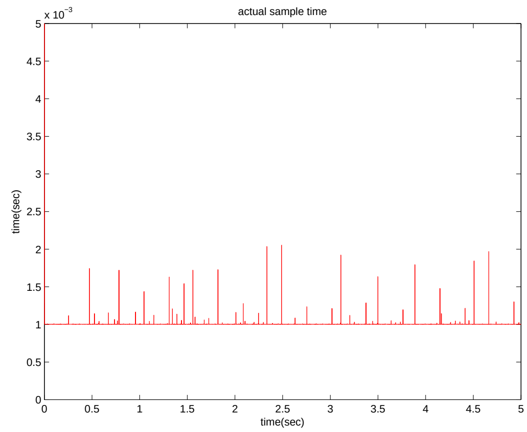

Fig. 12: Actual sample time at the sample rate of 1 KHz

Figure 9 shows the picture of the experimental setup of the DC motor control. We used a Maxon DC motor (RE 40) for control. It is shown in the figure that the wire connection between the motor driver and the IO expansion board are done using jumper wires with female connectors on both ends.

Figure 10 shows the result of the control experiment with the reference being a sinusoidal wave. It is seen that the actual velocity of the motor follows the reference trajectory successfully.

Let's consider some more detail about how Time block is used to improve the control performance. Let's denote the encoder counter value at i as $N(i)$ with $i = 0, 1, 2, \cdots$. Then the angular velocity $w(i)$ of a motor can be obtained as

$$w(i) = \frac{K[N(i) - N(i-1)]}{T(i) - T(i-1)},$$

where $T(i)$ is the time obtained from the Arduino Due at i and K is a coefficient that changes the counter value to the angle. Defining $\Delta(i) = T(i) - T(i-1)$, $\Delta(i)$ represents the actual sample time size. The block diagram shown in Figure 8 has two implementations for getting the angular velocity of the motor: the one contained in a dashed box and the other contained in a dotted box. The blocks contained in a dashed box obtains the angular velocity assuming the sample time to be constant. On the other hand, the blocks contained in a dotted box obtains the angular velocity using the actual sample time size obtained from Time block. If the actual sample time is constant, the two implementations will give the same velocity. However, it is not the case because Windows is not a realtime operating system. The angular velocity shown in Figure 10 is obtained through the implementation contained in the dotted box. Figure 11 shows the angular velocity assuming that the sample time is constant (dashed box). It has big spikes in velocity. We can guess that the control performance will be very poor if we use this velocity value for feedback. Figure 12 shows the actual sample time size obtained using Time block supported by the proposed RCP system. Overall the sample time is 1 millisecond. However, we see lots of jitters in the sample time size, which shows that the control computation sometimes is not finished within 1 millisecond. From this, we know that appropriate use of time information retrieved from the Arduino Due can greatly improve the performance of the control system constructed by the proposed RCP system.

4 Conclusions

In this paper, we proposed a new RCP system based on Matlab/Simulink and the Arduino Due with the built-in native high speed USB communication interface. The proposed RCP system currently proivdes 7 input/output library blocks for use with Simulink. Using those library blocks, users can build a prototype control algorithm with ease and fast. Since the control algorithm is computed by Simulink running on a PC, we can make full use of the strength of Simulink. Furthermore, one can use various functions provided by Matlab and thus complicated control algorithms can be easily built. The proposed RCP system has limited realtime property and can be used at moderate sample rate, i.e. 2 KHz. However it has several advantages such as good portability, no need for C-code generation, and cost effectiveness. Future work will include the extension of I/O blocks such that the proposed RCP system can be applied more versatilely.

Acknowledgements

This research was supported by the MSIP(Ministry of Science, ICT and Future Planning), Korea, under the C-ITRC(Convergence Information Technology Research Center) (IITP-2015-H8601-15-1003) supervised by the IITP(Institute for Information & Communications Technology Promotion)

References

Bucher, R. and Balemi, S. (2006), ‘Rapid controller prototyping with Matlab/Simulink and Linux’, *Control Engineering Practice* **14**, 185–192.

Hercog, D. and Jezernik, K. (2005), ‘Rapid control prototyping using Matlab/Simulink and a DSP-based motor controller’, *Int. J. Engng Ed.* **21**(3), 1–9.

Jiang, H., Ojaruega, M., Becchetti, F. D., Griffin, H. C. and Torres-Isea, R. O. (2011), ‘A USB-based portable data acquisition system for detector development and nuclear research’, *Nuclear Instruments and Methods in Physics Research A* **652**, 483–486.

Kennel, R. (2006), ‘Improved direct torque control for induction motor drives with rapid prototyping system’, *Energy Conversion and Management* **47**, 1999–2010.

Khalil, M. I. (2006), ‘A USB-based data acquisition system for neutron TOF measurements’, *Meas. Sci. Technol.* **17**, N1–N7.

Lee, W., Shin, M. and Sunwoo, M. (2004), ‘Target-identical rapid control prototyping platform for model-based engine control’, *Proc. Instn Mech. Engrs Part D, J. Automobile Engineering* **218**, 755–765.

Lin, C. F., Tseng, C. Y. and Tseng, T. W. (2006), ‘A hardware-in-the-loop dynamics simulator for motorcycle rapid controller prototyping’, *Control Engineerning Practice* **14**, 1467–1476.

The Mathworks Inc. (2005*a*), *Real-time workshop embedded code user’s guide (ecoder_ug.pdf), Version 4*.

The Mathworks Inc. (2005*b*), *Real-time workshop user’s guide (rtw_ug.pdf), Version 6*.

Web 1 : Arduino Due (2015), *http://arduino.cc/en/Main/ArduinoBoardDue* consulted 1 Feb 2015 .

Web 2 : Universal Serial Bus Specification (2015), *http://www.usb.org* consulted 1 Feb 2015 .