Ball & Beam for Ele

Grind your teeth on PID design and programming

Jose Luis Rupérez Fombellida & José Manuel Escobosa Bravo

Like it or not, Ball & Beam is a compulsory item on the Electronics Engineering curriculum, specifically for classes on PID (proportional-integral-derivative) control systems. The idea is to try to balance a ball on a beam using a control loop. If the ball is pushed, the system moves the beam to return the ball to its initial position. So let's get out our USB DAQ card and a PC to show off!

Apart from your curiosity and eagerness to learn about PID (one of the great pillars of electronic engineering), the ingredients of this project can be summarised as

- the Elektor USB Data Acquisition (DAQ) Card [1];
- a circuit which acts as interface between the DAQ and the mechanical system;
- mechanics, i.e. construction of the ball & beam system, which is very easy and cheap compared to commercial units;
- A personal computer running a program.

This project can be interesting for anyone keen on electronics and/or computers; even more so if he or she is curious about control and regulation systems or about discrete PID controllers using Windows C++/CLI programming. It may also trigger old hands at electronics to re-live the PID experiments they once did at College in the dim past.
The project has a low threshold as there is no need to do any Laplace or Z transform. There is no need to do any calculation either; apart perhaps from an empirical tweaking of the PID controller by adjusting certain controls in the PC program. Of course, those of you with solid mathematics education can rigorously analyse the system.
On the other hand, seeing how a ball remains stable in a certain position on the beam not only has didactic or academic interest! Aircraft of the vertical takeoff type such as the Harrier employ similar control loop systems to maintain stability while stationary in the air.

Watch the Elektor Ball & Beam scale model on YouTube

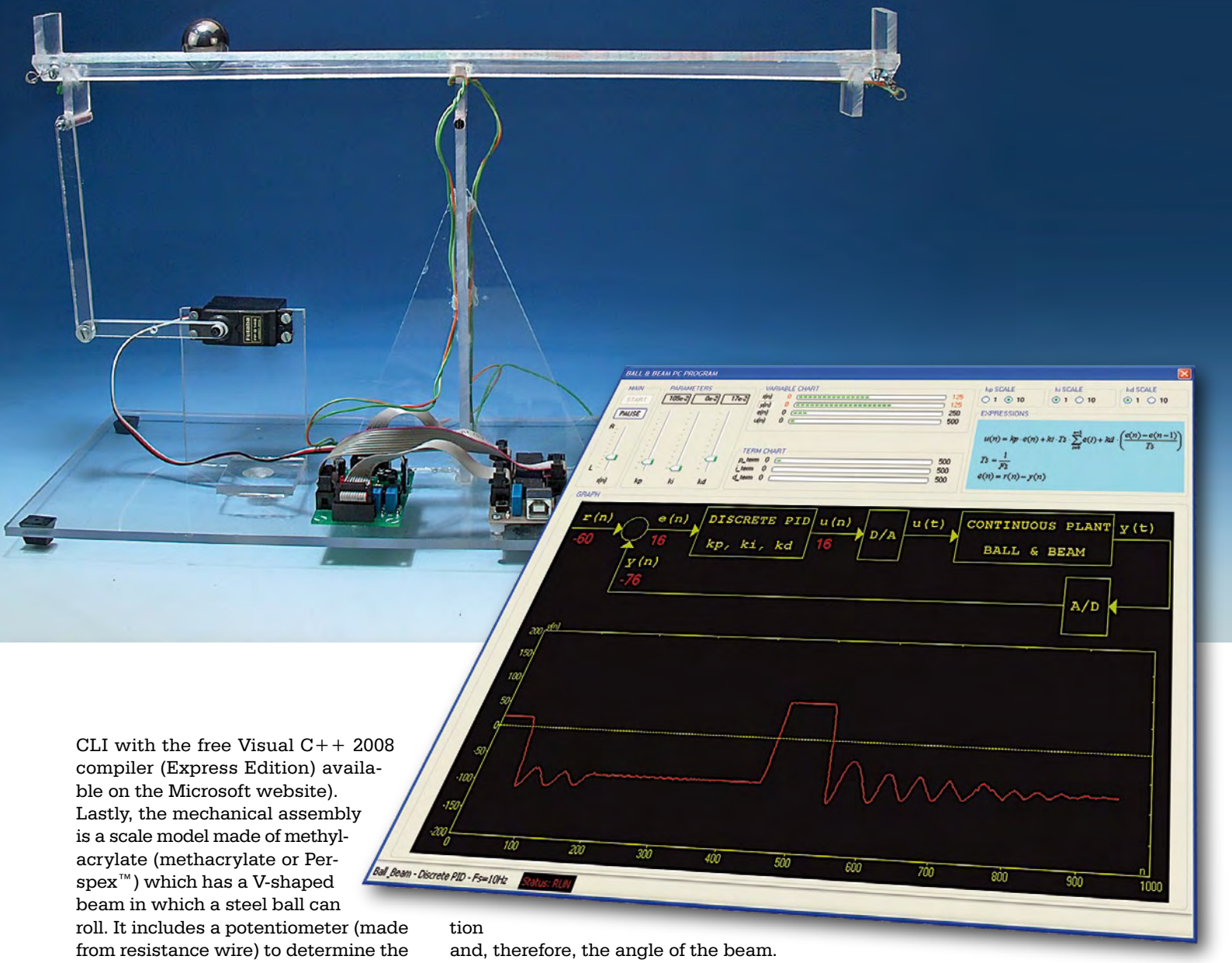
Recipe for the system

The Ball & Beam system consists of electronics, software and mechanical elements and so makes a nice project to run by a group of students from different fields of study. The elements are shown in **Figure 1**.
The electronics comprises two printed circuit boards:
1. Elektor USB Data Acquisition Card (November 2007). This card has 8 digital inputs, 8 digital outputs, 8 analogue inputs and 2 analogue outputs. From all this I/O capacity we employ just one digital output, one analogue output and one analogue input. For a deeper knowledge of this card, the article in ref. [1] should be read.

2. Ball & Beam Interface Card. The card described in this article is used to adapt the Ball & Beam Mechanics to the USB Data Acquisition Card.

For software, we add the **Ball & Beam PC Program**. This application for PC communicates with the Data Acquisition Card to send and receive information from the Ball & Beam mechanical assembly through the Ball & Beam Interface Card. It is developed in C++/

ktor USB DAQ Card



CLI with the free Visual C++ 2008 compiler (Express Edition) available on the Microsoft website).
Lastly, the mechanical assembly is a scale model made of methylacrylate (methacrylate or Perspex™) which has a V-shaped beam in which a steel ball can roll. It includes a potentiometer (made from resistance wire) to determine the position of the ball on the beam, and a servo to make the beam swing up and down.

USB data acquisition card

From the USB Data Acquisition Card, the following resources are used:

1. AN0 — this analogue input is available via pin 1 of connector K5 on the card. This input receives a voltage proportional to the position of the ball on the beam.

2. CCP1 — this analogue output is found on the pin 1 of connector K4. It is programmed to control the servo position and, therefore, the angle of the beam.

3. RD0 — this digital output is available on pin 1 of K1. An LED is connected to it, which will blink at the PID sampling frequency F_s when the system is operating.

Ball & Beam interface card

The Ball & Beam Interface Card was purposely kept a simple as possible, using cheap and easy to find components. The price to be paid for this simplicity is some tolerance but that's not a problem since we have an adjustment process available.
The circuit diagram of the interface card is shown in Figure 2. It can be thought of as consisting of three parts.

1. LED — this indicator (with its associated current limiting resistor) is connected to the RD0 digital output on the USB Data Acquisition Card. The Ball & Beam PC program will make this LED blink to the sampling frequency F_s.

2. Filter — this is for the ball position 'potentiometer'. As the steel ball touches two lengths of Nickel-Chrome wire, on occasions false contacts will be produced which would give wrong values. To counteract erratic values as much as possible, a low-pass filter consisting of R1 and C1 is included.

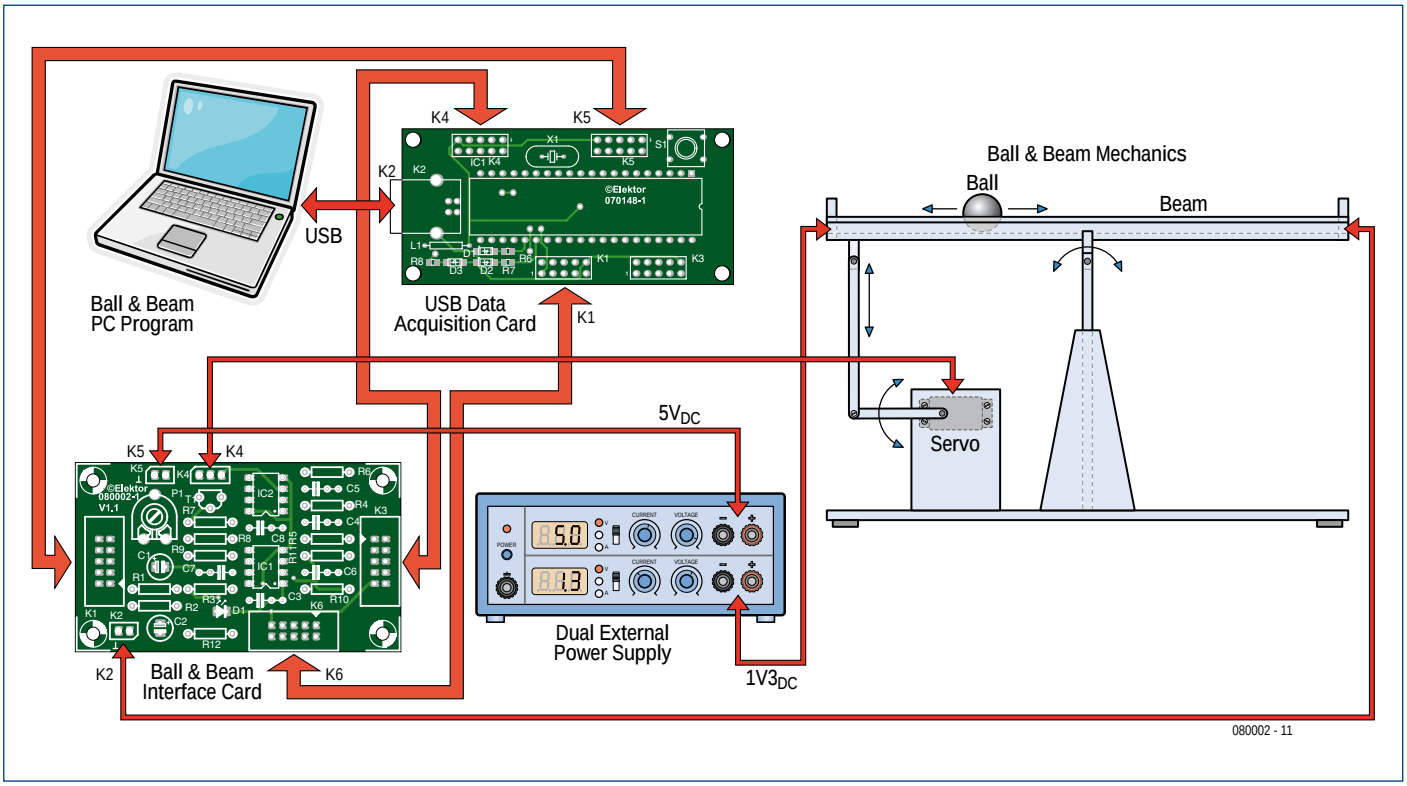


Figure 1. Overview of electronic and mechanical components connected up.

This 6 dB/octave (first order) filter effectively sits between the ball position 'potentiometer' and the AN0 analogue input on the USB Data Acquisition Card. The simulation result of this filter is shown in **Figure 3**. Note that the roll-off frequency is very low at just 7.5 Hz, but enough to perfectly register the ball position due to the movement by the beam.

3. Pulse generator to the servo — the beam slope is determined by the position of a servo. The beam will remain horizontal if the servo is in the centre position. If the servo spindle is at one extreme, the beam slopes some 15°. If it is at the other extreme, the beam slopes 15° opposite.

The signal fed to a servo is composed of pulses with a frequency between 50 Hz and 60 Hz. The pulsewidth determines the servo position and varies between 1 ms and 2 ms. The servo's central position corresponds to 1.5 ms. These times may vary slightly from a servo manufacturer to another. The pulsewidth is controlled by the CCP1 analogue output on the USB DAQ Card.

The relevant electronics is simple: one astable and one monostable implemented using the ICM7555 or TLC555, i.e., the low-power CMOS version of the famous LM/NE555 (which, by the way, should not be used). Both IC1 and IC2 are supplied from the USB port on your computer through the USB DAQ Card. More on the 7555 and relevant calculations in the **inset**.

Around IC2 a monostable has been assembled. Its trigger signal is provided by IC1; that is why the pulses generated by the monostable have the same frequency as the IC1 astable. The output pulsewidth controls the servo position and must have a frequency between 50 Hz and 60 Hz. The monostable pulsewidth should vary between 1 ms and 2 ms (typically).

The pulsewidth is determined by the CCP1 analogue output. CCP1 supplies a PWM signal with a frequency of about 2.9 kHz whose average in volts can be calculated with the expression:

$$U_{CCP1} = 5D$$

Where D is a constant between 0 and 1. As the CCP1 signal is going to modulate the monostable's pulsewidth, it is necessary to filter it to turn it into a continuous level. This is done by R5, C4, R4 and C5, the components forming a low-pass filter of modest features with a roll-off at about 30 Hz. That's high enough for the sampling frequency F_s and low enough for suppressing harmonics of the 2.9 kHz PWM signal on CCP1. The slope of the filter is 12 dB/octave (2nd order). Notice that the charging of the monostable's capacitor is not done with the usual simple resistor — here, a constant-current source is used. Built around T1, it keeps the capacitor's charge build-up linear, enabling the monostable pulsewidth to vary in linear fashion with the CCP1 signal. The constant current flows in T1's collector line and is adjusted by P1. See the **Monostable IC2 dimensioning inset** for the component value calculations.

The servo, besides the pulse signal, needs a supply voltage. As its current consumption is rather high (hundreds

Provide video proof to Elektor if you did the project with a ball heavier than 1 kg.

of milliamps), it is impossible to supply it from the USB DAQ Card (remember where that card gets its power from?). Hence it is essential to power the servo from an external supply rated at 5 V and 1 A, connected up to K5.

Construction

The component mounting plan of the circuit board designed for the interface card is shown in **Figure 4**. As usual the true-scale copper track layout (reflected and non-reflected) is in a free .pdf file found on www.elektor.com/080002 i.e. .the web page for the project.
Construction of this board should be a breeze as only traditional through-hole components are involved.

The Ball & Beam PC program

The program developed for the project implements a discrete PID controller with an F_s (sampling frequency) of 10 Hz. The PID mathematical derivation is given in the **PID maths inset**. The PC program is also a free download from the project web page.
All PID parameters can be controlled and observed in the program, see **Figure 5**. A brief program description is given below.

- MAIN. Using START, you launch the system and with PAUSE you stop it. You can adjust the reference (desired position of the ball on the beam) by adjusting *r(n)*.

- PARAMETERS. Here the PID controller's *kp, ki,* and *kd* constants can be adjusted. For each of these there are two scales: 1 and 10.

- VARIABLE CHART. This shows the evolution of *r(n), y(n), e(n), u(n)* signals. These are, respectively: reference (desired position of the ball on the beam); real position of the ball on the beam; error and PID controller output to the servo. The starting numbers and end of scale remain in black if the value is positive and they change into red for negative values.

- TERM CHART. Here you can view the evolution of proportional, integral, derivative (PID) terms, as parts of the controller maths. As before, black is positive and red is negative.

- EXPRESSIONS: In a frame you view the mathematical expression used for the PID controller implementation.

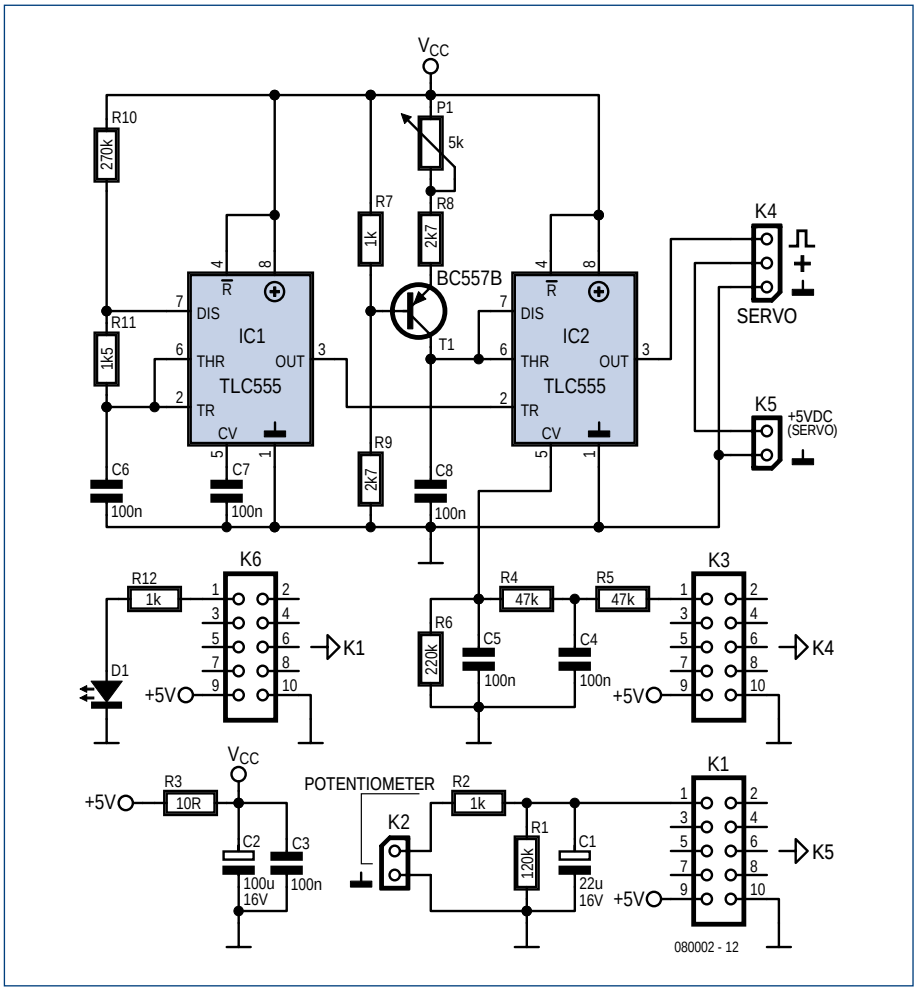


Figure 2. Circuit diagram of the Ball & Beam interface card.

- GRAPH. There are two charts: the first one is the whole system's block diagram consisting of the discrete PID controller (implemented by software in the Ball & Beam Program), a D/A converter and an A/D (both on the USB DAQ) and the 'continuous plant' which is the Ball & Beam mechanical assembly. The Ball & Beam Interface Card only adapts the USB Data Acquisition Card to the Ball and Beam mechanics, that's why it does not appear in the

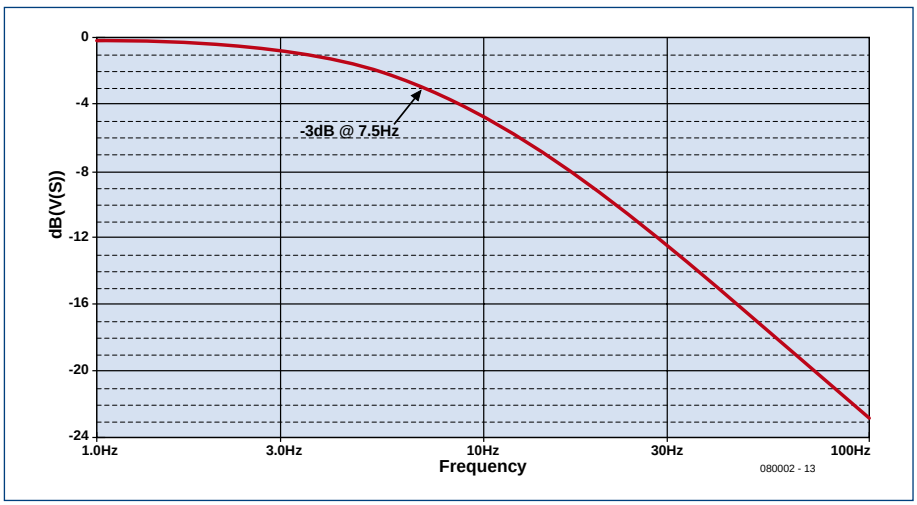


Figure 3. Frequency response of the filter designed for the wire potentiometer.

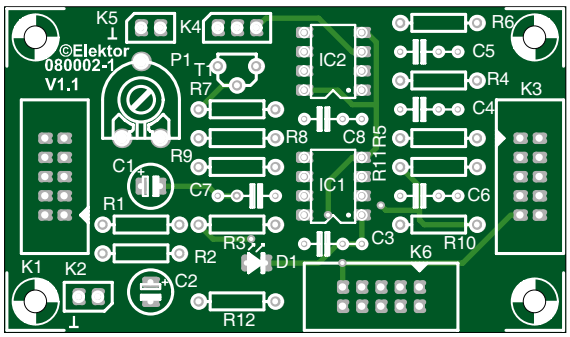


Figure 4. PCB design for the interface. The copper track layout is a free download.

COMPONENTS LIST

Resistors
R1 = 120kΩ
R2,R7,R12 = 1kΩ
R3 = 10Ω
R4,R5 = 47kΩ
R6 = 220kΩ
R8,R9 = 2kΩ7
R10 = 270kΩ
R11 = 1kΩ5
P1 = 5kΩ preset

Capacitors
C1 = 22µF 16V radial
C2 = 100µF 16V radial
C3,C6,C7,C8 = 100nF
C4,C5 = 100nF

Semiconductors
D1 = LED, low current, 3mm
T1 = BC557B
IC1,IC2 = TLC555 or ICM7555 (do not use plain 555)

Miscellaneous
K1,K3,K6 = 10-way boxheader
K2,K5 = 2-way SIL pinheader
K4 = 3-way SIL pinheader
PCB, ref. 080002-1 from www.thepcbshop.com
PCB artwork, free download from www.elektor.com/080002
Project software, free download from www.elektor.com/080002

block schematic.
The second chart, located in the lower part of the screen, shows the evolution of a selected (n) function. The signal can be any one of those included in the VARIABLE or TERM CHART. By clicking on its name, it will appear represented.

The program was written using C++/CLI with the free of charge Visual C++ 2008 (Express Edition) compiler. The most important function is called 'Void timer1_Tick' and it is found in the archive 'Form1.h'. In this function, among others, sits the discrete PID controller code.
The free of charge .NET XYGraph was also used for the control of componentXtra.com.

Ball & Beam mechanics

Parts lists, parts photos and details of the mechanical assembly of the Ball & Beam scale model may be found in a supplementary document that can be downloaded free of charge from the project web page as archive file **080002-W.zip**. The document also has an interesting discussion on selection of the ball material (steel) and its weight (approx. 64 g).

The beam has two wires made of nickel-chrome (NiCr 80%/20%) between which the ball rolls. The steel ball representing the wiper, the lot acts like a linear potentiometer (or 'rheostat') that's perfect to determine the ball position on the beam. The resistance wire is normally used for manufacturing and repairing heating elements. The wire has a diameter of 0.5 mm. Resistivity for the nickel-chrome wire at 80%/20% is about 1.1 Ω·mm²/m. The beam length being 40 cm, the resistance works out as

$$R = \rho \cdot \frac{L}{S} =$$

$$1.1 \cdot \frac{40 \cdot 10^{-2}}{196.35 \cdot 10^{-3}} = 2.24\Omega$$

$$S = \pi \cdot r^2 =$$

$$\pi \cdot \left(\frac{0.5}{2}\right)^2 = 196.35 mm^2$$

How this acts like a potentiometer is illustrated in **Figure 6**. If the ball is at the left end of the beam, the voltage between the wires is (close to) 0 V. This voltage keeps increasing as the ball moves to the right, until getting to a maximum of 1.3 V.
The servo used is the Futaba type S148 or equivalent. It is connected to K4 on the Ball & Beam Interface Card. The servo's white wire connects to pin 1 of K4, the red one to pin 2 and the black one to pin 3.

Starting up

The USB DAQ Card connection to the PC and the installation of the corresponding driver is explained in [1].
Once the USB DAQ is installed, you can launch the Ball & Beam PC program. Make sure that the application is stopped (Status: STOP). Connect the 5 V, 1 A external power supply and adjust preset P1 until the 'C' arm (see mechanical assembly document) gets horizontal — the beam must be horizontal too. Click on the START button (Status: RUN); this will take controls *r(n)* and *kp* to their highest position (the latter in the ×10 scale), whilst checking that the 'C' arm is at an angle of 30° to 50° relative to the base. Then stop the application (Status: STOP).
Now put the ball on the beam and connect the 1.3 V/1 A auxiliary power supply. Move the ball by hand to the far right of the beam, keeping it there, and write down the voltage measured across resistor R2. This should be approximately 1.3 V. Adjust the aux power supply to get exactly 1.3 V. Now everything is ready for the first try.

Maiden run

Push the START button on the program and slide reference *r(n)* to the middle. Slowly raise the *kp* proportional constant until the ball rocks slowly on the beam. Carefully increase the derivative constant until the ball is stable. If so, push it a little by hand — the system should respond by keeping the ball steady.
Now increase the integral constant little by little until the error is eliminated and the ball is steady as before, but this time approximately in the middle of the beam. Got it! PID in action… tell everyone about it at College and in the Elektor forum!

From here, you and your classmates can try as much you want, for example:

- choose different positions for the ball using *r(n)*.
- Make the system respond slower or quicker, with increased or reduced accuracy, with more or fewer tendency to oscillation, and so on, by changing *kp, kd* and *ki*.
- study the system's maths and adjust the constants for determined starting premises.

About Windows

MS Windows is not a real-time operating system. In fact, it's constantly waiting for events coming from peripherals (keyboard, mouse, etc.) and from applications. This can affect our system's sampling frequency F_s hence cause irregularities in the operation (the Ball & Beam Interface Card LED will be our witness).

To counteract this inconvenience as much as possible, certain precautions must be taken:

- Use a PC with fast CPU and a vast amount of RAM memory.
- Execute the Ball & Beam PC program only. Pull the plug on MSN & Co.
- Give your application a high level of priority in the Windows multitasking scheme (this is done by the program).
- While the Ball & Beam PC program application is executing, do not maximize not minimize windows or push the Start button, etc. Allow the PC to interact with the application only.

If you stick to the above guidelines, the sampling frequency will be practically free from irregularities.

If something does not properly work

If something does not work, take some time to read this section, it will help you locate the cause either in the PC program, the interface card, or the mechanical assembly.

First, make sure of a running, fully updated Windows XP or Vista. Ball & Beam does not work under Windows 2000 or lower. Also check if you have the latest update of .NET Framework (from version 3.5).
The executable program Ball_Beam.exe is in the same folder as mpusbapi.dll and XYGraph.dll. To run the program under Windows Vista, right-click on 'Ball_Beam.exe' and from the context menu choose 'Execute as Administrator'.

The program has been successfully tested on Windows XP and Vista PCs without encountering any problem. However, the program may indicate some error if there is a problem with the USB Data Acquisition Card. If so, go back to ref. [1]. Note that a correction was published for the project.

If the program is modified, it must be compiled again with the Visual C++ 2008 Express Edition — do not use the 2005 version. Besides, you should have NET XYGraph and Visual C++ 2008 Express Edition, all correctly registered.

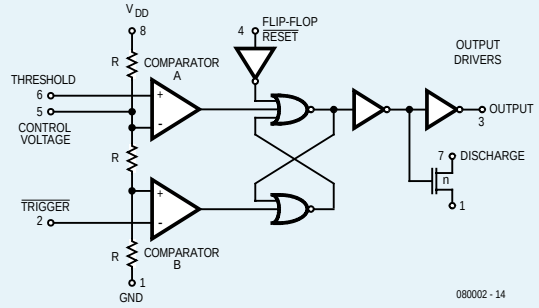
The program needs a screen resolution of at least 1280×1024 so send the old 15 inch CRT off to recycling along with the 486 or Win98 PC.

On the Ball & Beam Interface Card it may happen that it is impossible to get the adjustment done properly. This is probably due to tolerances, mainly in the internal voltage divisor of the (ICM)7555 or TLC555 integrated circuit (in the prototype we used a Philips IC). With an oscilloscope, check up the signal on IC1 pin 3 (0 V and 5 V levels, frequency between 50 Hz and 60 Hz, low level of some 100 μs). If not, IC1 is probably at fault or you have mounted a wrong component value.

If it proves impossible the get the 'C' arm horizontal by adjusting P1, change C8 to a different value (47 nF or 150 nF).
If during the start-up the 'C' arm is noticeably higher than 30-50° relative to the base, try different values for R4 and R5 (68 kΩ or 100 kΩ).

(ICM)7555 astable multivibrator

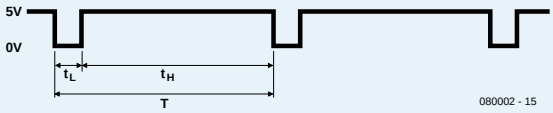
The (ICM)7555 chip is the CMOS low-power version of the renowned LM555/NE555. With it, RC timers can be easily designed. Its functional schematic is given below.



For some later calculations, it is necessary to know the value of R. For the present application, measurements indicate that a value of 100 kΩ is best.

The signal generated by the astable multivibrator around IC1 is periodical but with the High level time much longer than the Low level time. The Low level pulse is used for triggering a second 7555 (IC2) wired as a monostable multivibrator.

Theoretically, the signal the astable should supply at its output (pin 3 of IC1) is:



$$t_H = (R10 + R11)\cdot C6\cdot \ln 2 = (270k + 1k5)\cdot 100n\cdot 0.69 = 18.7ms$$

$$t_L = R11\cdot C6\cdot \ln 2 = 1k5\cdot 100n\cdot 0.69 = 103.5\ \ s$$

$$T = t_H + t_L = 18.7ms + 103.5\ \ s = 18.8ms$$

$$F = \frac{1}{T} = \frac{1}{18.8ms} = 53.2Hz$$

The actually measured values are:

t_H = 18 ms

t_L = 100 μs

T = t_H + t_L = 18.1ms, hence F = 1/T = 55.2 Hz

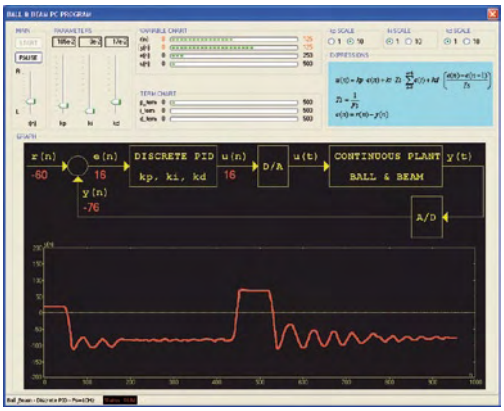


Figure 5. The Ball & Beam PC software in action.
Food for C++ programmers and quite a lot to tweak to get the ball stable on the beam!

If it's

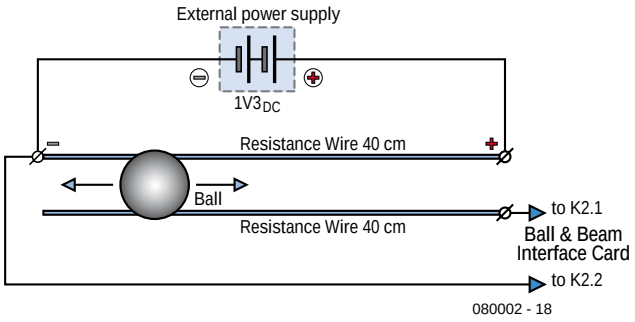
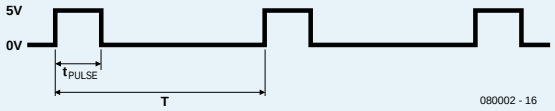


Figure 6. The position of the ball on the beam is read by a DIY rheostat or wire potentiometer.

Monostable IC2 dimensioning

The expected signal at the monostable output (pin 3 of IC2) is:



Allowing for a little headroom at either extreme, the servo timing range of 1.0 - 2.0 ms is calculated from

$$t_{PULSE} = \frac{C8 \cdot U_{CV}}{I_{Q1}}$$

$$U_{CV} = 5 \cdot \frac{(2 \cdot R) \| R6 \| (R4 + R5)}{R + \left[(2 \cdot R) \| R6 \| (R4 + R5)\right]} + U_{CCP1} \cdot \frac{R \| R6 \| (2 \cdot R)}{(R4 + R5) + \left[R \| R6 \| (2 \cdot R)\right]} = 1.65 + 0.54 \cdot U_{CCP1}$$

R=100K

$$I_{Q1(\alpha=0.2)} = \frac{\left(\frac{5 \cdot R7}{R9 + R7} - U\gamma\right)}{(R8 + \alpha \cdot P1)} = 203 \ A$$

Uγ=0.6V

$$t_{PULSE} = \frac{C8 \cdot U_{CV}}{I_{Q1}} = \frac{100nF \cdot (1.65 + 0.54 \cdot U_{CCP1})}{203 \ A} = \begin{matrix} 0.8ms @ U_{CCP1} = 0V \\ 1.5ms @ U_{CCP1} = 2.5V \\ 2.1ms @ U_{CCP1} = 5V \end{matrix}$$

Note that the variation of t_{PULSE} is linear with respect to U_{CCP1}. In practice, the pulse durations obtained after adjusting P1 are well within the range 1ms to 2ms.

PID maths

$$u(n) = k_p \cdot e(n) + k_i \cdot Ts \cdot \sum_{i=0}^{n-1} e(i) + k_d \cdot \left(\frac{e(n) - e(n-1)}{Ts}\right)$$

$$Ts = \frac{1}{Fs}$$

$$e(n) = r(n) - y(n)$$

been necessary to change C8, R4 and R5, check the signal at IC2 pin 3 (levels 0 V and 5 V, frequency between 50 and 60 Hz, high level of 1.5 ms). For this last measurement, the Ball & Beam PC program has to be halted (Status: STOP) and P1 adjusted to reach the indicated 1.5 ms.

If everything is correct so far, ask yourself if the Ball & Beam Interface Card and the servo are properly powered, the latter from the 5 V / 1 A supply. Finally, are you powering the wire potentiometer with an external power supply of approximately 1.3 V and a minimum current of 1 A? If so, all that's left to do is verify the mechanical assembly.

(080002-I)

Reference

[1] USB Data Acquisition Card, Elektor November 2007. www.elektor.com/070148

About the Authors

Jose Luis Rupérez Fombellida and José Manuel Escobosa Bravo are teachers in electronics at San Blas Secondary School, Madrid, Spain.