Int. J. Engng Ed. Vol. 21, No. 5, pp. 814–837, 2005
Printed in Great Britain.

0949-149X/91 $3.00+0.00
© 2005 TEMPUS Publications.

Analysis and Synthesis in the Simulink Environment of Control Laws for DC Motors: A Real-time Implementation with the Microcontroller BASIC Stamp II as a Simple Tool for an Education Control Laboratory*

ANTONIO TORNAMBÈ
Dipartimento di Informatica, Sistemi e Produzione, Università di Roma Tor Vergata, via del Politecnico, 1, 00133 Roma, Italy. E-mail: tornambe@disp.uniroma2.it

This paper describes a simple and cheap experimental apparatus that has been developed at Università di Roma Tor Vergata for educational purposes; this tool is currently used in the course "Laboratorio d'Automatica" (Laboratory of Automatic Control), which is offered in the first year of a three-year university programme ("Laurea" degree) in Automatic Control. The most notable feature of this experimental apparatus is that it can be controlled in real time from Simulink; the paper reports on several experimental tests for students using this tool who are only familiar with the fundamentals of calculus, geometry, physics and computer sciences.

INTRODUCTION

THIS PAPER DESCRIBES a simple and cheap experimental apparatus that has been developed at Università di Roma Tor Vergata for educational purposes; this tool is presently used in the course "Laboratorio d'Automatica" (Laboratory of Automatic Control), which is offered in the first year of a three-year university programme ("Laurea" degree) in Automatic Control. The aim of the course is to present the basic principles of Automatic Control from a practical point of view, particularly before students are introduced to the theory. In the first year of the "Laurea" degree in Automatic Control, the students are only familiar with the fundamentals of calculus, geometry, physics and computer sciences; hence, introducing the basics of Automatic Control theory from a practical point of view, using "visual" environments such as Simulink, is almost mandatory. The experimental apparatus allows the real-time control of a DC motor, using the microcontroller BASIC Stamp II (BS2) for real-time implementation of the control laws; through the serial port of the PC, the BS2 communicates in real time with a Simulink program; directly in Simulink, the student can read the state of the DC motor encoder, which can be used for velocity estimation with a high-gain observer; the student can also directly apply to the DC motor a suitable input voltage whose value can be computed in real time on the basis of measured position and of estimated velocity through simple control laws.

Students on the course are arranged in small groups (two or three students per group) and the instructor provides each group with a PC connected to the experimental apparatus, which constitutes a sort of simulative/experimental personal desktop lab; this can be done in our institution, thanks to the low cost of the experimental apparatus, by using the computer science laboratory PCs, which are already available. The experimental set-up is already assembled by the instructor, so that the students are only asked to control it by programming in Simulink and/or directly at low level using BS2, after a set of simulation tests. The most notable feature of our software implementation is that the same Simulink program used in the simulation can be used to control the actual experimental apparatus in real time; it is sufficient to replace the Simulink block corresponding to the model of the DC motor with another Simulink block which is concerned with real-time communication between the experimental set-up and the PC.

Several projects can be assigned to each group: implementation both in Simulink and with BS2 of a simple velocity estimation algorithm (a high-gain observer); tuning of the parameters appearing on a simple model of the DC motor through a series of

* Accepted 6 July 2005.

experiments that exert different input voltages on the motor; design and simulation of simple control laws in Simulink (proportional control law, PID control law, Sliding mode control); real-time implementation in Simulink of the above control laws; real-time implementation of the same control laws with BS2. In this way, the students can familiarize themselves with all the phases necessary for the actual design of a real control system: 1) design of a dynamic model of the process to be controlled and parameter tuning/identification, 2) analysis and design of control laws by simulation tests, 3) implementation of the control law on a prototype system, and 4) implementation on a microcontroller.

THE EQUIPMENT USED

The experimental set-up (schematic circuit shown in Fig. 1) was realized using the following equipment: one BASIC Stamp II, including the `NZ-1000` universal training board with the requisite 12V (DC) wall transformer (the BS2); the integrated circuit `DS1803-010`, which comprises two digitally controlled potentiometers, two synchronous up/down binary counters `HEF40193B` (four bits each), one rotary optical encoder `ENT1J-B28-L00128`, one DC motor `IG33` (± 12 V), one wideband amplifier `WA301`, one manually controlled potentiometer, and some resistors and capacitors.

The digitally controlled device `DS1803-010` and the wide-band amplifier `WA301` are used to generate an analogue positive/negative voltage suitable for the DC motor `IG33`. The two counters `HEF40193B` and the encoder `ENT1J-B28-L00128` are used to digitally measure the position of the motor shaft. The manually controlled potentiometer is used to generate an analogue reference input, proportional to the position of its shaft. All these operations are controlled and coordinated by the `BS2`, possibly with the interaction of a Simulink program running on a PC communicating with the `BS2` through the serial ports COM1/COM2.

In particular, the `BS2` can be used for the implementation of various control and observation/estimation algorithms, and for data exchange with the other devices and with the PC; for the `BS2` $\rightarrow$ PC (serial port COM1) communication, serial data are transmitted using the `DEBUG` instruction (or, equivalently, with a `SEROUT` instruction) on a `BS2` pin dedicated to programming (`PIN16=SOUT`, which is not directly available on the `NX-1000` board but is available through the programming cable), whereas for the (serial port COM2) PC $\rightarrow$ `BS2` communication, serial data are received using the `SERIN` instruction on the `BS2 PIN15` using a second communication cable. Using appropriate commands executed with the BS2, the position of the wiper terminals of the two digitally controlled potentiometers available within `DS1803-010` can be set independently, thus digitally defining two voltages (V_0 and V_1). The difference between these two voltages ($V := V_0 - V_1$) is suitably amplified with the wide-band amplifier, thus generating a ± 12 V voltage that can be directly applied to the DC motor, which can rotate clockwise and counterclockwise, depending on the sign of the amplified voltage (clockwise if $V_0 < V_1 - \Delta$, counterclockwise if $V_0 > V_1 + \Delta$, at rest if $abs(V_0 - V_1) \leq \Delta$, with 2Δ being the amplitude of the motor dead-zone, assuming symmetric). The encoder shaft is directly joined to the motor shaft: the data coming from the two channels A and B of the encoder can then be fed to the counters, which are connected in cascade to count from 0 up to 255 (eight bits), before resetting. The shaft position of the manually controlled potentiometer is read by the `BS2` with the `RCTIME` instruction, and this is used as an analogue reference input, both for position and velocity control.

A brief description of each device is given below.

The BASIC Stamp II is a hybrid microcontroller which is designed to be programmed in a version of the BASIC programming language called PBASIC; version 2.5 of the PBASIC language is used, with the freeware BASIC STAMP Editor 2.0 for programming. All the subsequent fragments of code are written in PBASIC 2.5.

The main program is simply an infinite loop constituted by the following sequence of commands: reading (or, real-time acquisition from Simulink) of the reference value (either reference position or reference velocity), reading of the state of the encoder, possible velocity estimation with a high-gain observer, computation (or real-time acquisition from Simulink) of the control input, application of the requisite voltage to the DC motor, and debugging for PC data acquisition and elaboration in real time using Simulink, which receives the data from the `BS2` with the format "`start;%d,%d,%d;end`".

```
DO
 GOSUB Reading  ' Reading of the reference
    value
 GOSUB Measure  ' Reading of the encoder
 GOSUB Observer ' Velocity estimation
 GOSUB Control  ' Computation of the
    control law
 GOSUB Potentiometer0 ' Writing the
    control law on Pot 0
 GOSUB Potentiometer1 ' Writing the
    control law on Pot 1
 ' The reference and measured positions
    and the control law
 ' are sent to the PC in a format suitable
    for interfacing Simulink
 DEBUG ''start;'',DEC
    RPosition,'','',DEC
    MPosition,'','',DEC
    SControlInput,'';end''
LOOP
```

The subroutines `Observer` and `Control` are specified in the subsequent sections, according to

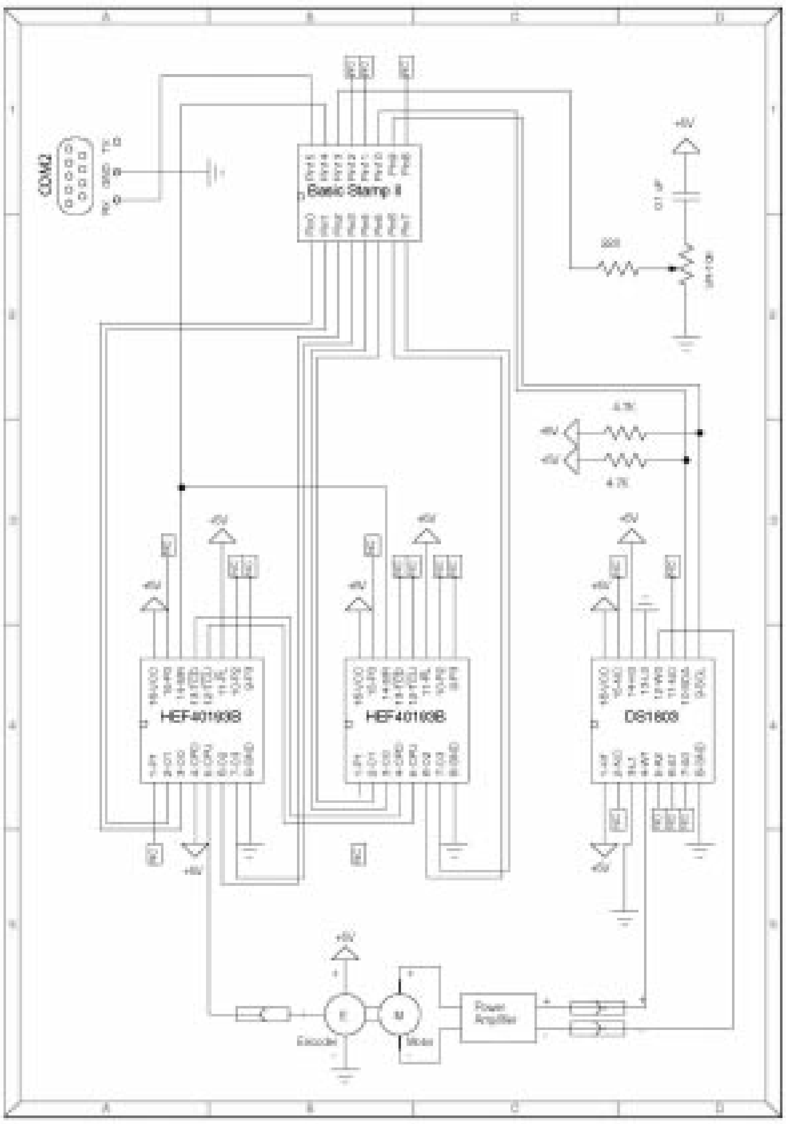

Fig. 1. The schematic circuit of the experimental set-up.

the particular assignment (see [6–8] for details on general control theory and digital applications).

The subroutines Reading, Measure, Potentiometer0 and Potentiometer1 are specified in the section where the devices and main procedures are described.

ASSIGNMENT 1: HIGH-GAIN OBSERVER IMPLEMENTATION IN SIMULINK USING THE BS2

The first assignment is the implementation, in Simulink and using the `BS2`, of a high-gain observer which, using the measured position `MPosition` (provided by the `BS2`) of the motor shaft, gives an asymptotic (and, hopefully, accurate) estimate of the motor shaft velocity. The equations for the high-gain observer for estimating the velocity are [9]:

$$\dot{\hat{x}}_1(t) = \hat{x}_2(t) + k_1(y(t) - \hat{x}_1(t)),$$
$$\dot{\hat{x}}_2(t) = k_2(y(t) - \hat{x}_1(t)),$$

where $y(t)$ is the measured shaft position at time t, $\hat{x}_1(t)$ and $\hat{x}_2(t)$ are the estimated shaft position and velocity, respectively, at the same time t, $k_1 := 2\mu$ and $k_2 := \mu^2$, with μ being the high-gain (in fact, the value chosen after some trials is not too high: $\mu = 4$).

Notice that the above system is a full order observer, which estimates not only the velocity but also the measured position; this has been done in order to provide students with a way to judge the accuracy of the estimated velocity $\hat{x}_2(t)$, which is the comparison of the estimated position $\hat{x}_1(t)$ with the measured one $y(t)$.

The high-gain observer is realized in Simulink with a continuous-time S-Function (the one marked with `hg` in Fig. 2), so that the students

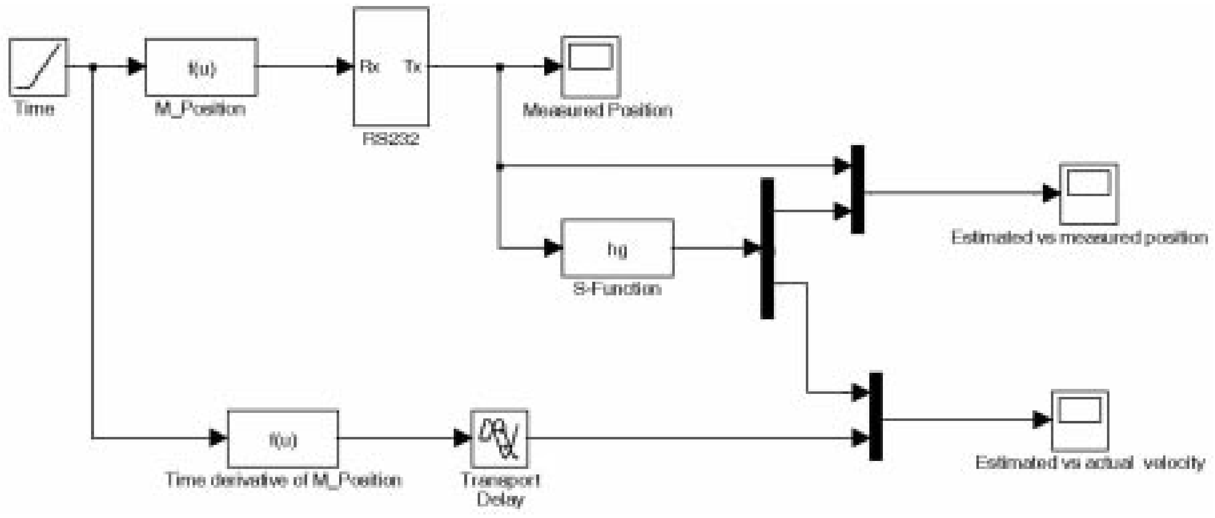
Time
f(u)
M_Position
Rx Tx
RS232
Measured Position
hg
S-Function
Estimated vs measured position
Estimated vs actual velocity
f(u)
Time derivative of M_Position
Transport Delay

Fig. 2. Simulink scheme for designing a high-gain observer with BS2.

do not have to concern themselves initially with its discrete-time implementation.

Serial communication with the PC is realized with the following fragment of program (the Simulink program with the "RS232" block transmits the data with the format "`A%d;`" and receives the data with the format "`start;%d;end`"):

```
Baud96 CON 84 ' 9600-8-N-1
'-------
SERIN 15,Baud96,[WAIT(''A''),
    DEC MPosition]
SEROUT 16,Baud96,[''start;'',
    DEC MPosition,'';end'']
```

In particular, the value of `MPosition` is sent from Simulink through the serial port COM2 and received by the `BS2`, which transmits back to the PC the same value through the serial port COM1. In this way the data received from the `BS2` constitute a function of time known in closed form, whose time derivative can be computed in closed form too. The output of the "RS232" block is used as input for the high-gain observer (variable $y(t)$ in the equations of the high-gain observer), which asymptotically estimates both `MPosition` and its time derivative (variables $\hat{x}_1(t)$ and $\hat{x}_2(t)$, respectively, in the equations of the high-gain observer). For validation purposes, such asymptotic estimates are compared with the actual `MPosition` and with its time derivative.

The time histories reported in Fig. 3 are referred to a measured output $y(t) = 100(1 + \sin(t/2))$, whose time derivative (used for comparison with the estimated velocity) is $y(t) = 50\sin(t/2)$. As one

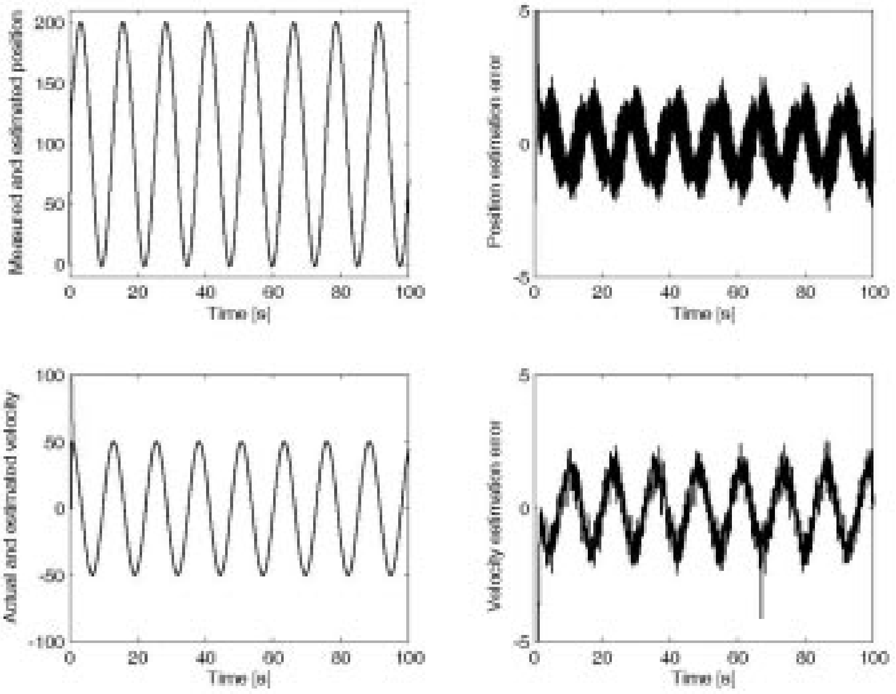

Fig. 3. The high-gain observer realized in Simulink, which uses the values received in real-time by the BS2.

can see from Fig. 3, the estimates provided by the high-gain observer are sufficiently accurate (the actual and the estimated time histories almost overlap), apart from a small estimation error due to the quantisation and sampling necessary for communication with the BS2.

Then, the following discrete-time version (obtained by approximating the time derivative with the backward difference) of the high-gain observer was realized with the BS2:

$$\hat{x}_1(t+T) = \hat{x}_1(t) + T(\hat{x}_2(t) + k_1(y(t) - \hat{x}_1(t))),$$
$$\hat{x}_2(t+T) = \hat{x}_2(t) + T(k_2(y(t) - \hat{x}_1(t))),$$

where T is the sampling time. The following subroutine has been used for implementing the discrete-time version of the high-gain observer:

```
MPosition VAR Word ' Measured position
x1 VAR Word         ' Estimated position
x2 VAR Word         ' Estimated velocity
x1Old VAR Word     ' Old estimated position
x2Old VAR Word     ' Old estimated velocity
'---------------
mu CON 4           ' High-gain
K1 CON 2*mu        ' Observer coefficient
K2 CON mu*mu       ' Observer coefficient
Delta VAR Byte     ' Observer coefficient

Observer:
 x1Old=x1
 x2Old=x2
  IF MPosition>x1Old THEN
     x1=K1*(MPosition-x1Old)
     x2=K2*(MPosition-x1Old)/Delta
 ELSE
     x1=-(K1*(x1Old-MPosition))
     x2=-(K2*(x1Old-MPosition)/Delta)
 ENDIF

 x2=x2Old+x2
 IF x2Old+x1 > 32766 THEN
     x1=-((-(x2Old+x1))/Delta)
 ELSE
     x1=(x2Old+x1)/Delta
 ENDIF
 x1=x1Old+x1
 IF x1 > 32766 THEN
     x1=0
 ENDIF
RETURN
```

Note that the main difficulty in the implementation of the high-gain observer is the handling (in particular, the multiplication and the division) of variables that can assume both positive and negative values, as the negative numbers have been represented using the “two’s complement rule”.
The high-gain observer implemented with the BS2 is compared with the version implemented in Simulink, using the scheme shown in Fig. 4. The DC motor is manually controlled by varying its input voltage (using an external signal generator), whereas the motor shaft position is measured in real time by the BS2, which (always in real time) estimates the motor shaft velocity; hence, the BS2 transmits to the PC (through serial port COM1, with the format “start;%d,%d,%d;end”) the values of three variables: the measured position, the estimated position and the estimated velocity:

```
DEBUG ''start;'',DEC MPosition,'','',
      SDEC x1,'','',SDEC x2,'';end''
```

The measured position is used in Simulink as input for the second high-gain observer, which again estimates the motor shaft position and velocity, which are finally compared with those estimated with the BS2.
The time histories of the estimated positions (by the PC and the BS2), which are compared with the measured one, and of the estimated velocities are shown in Fig. 5. As can be seen from Fig. 5, the two high-gain observers behave accordingly (in particular, the estimated positions almost overlap with the measured one).

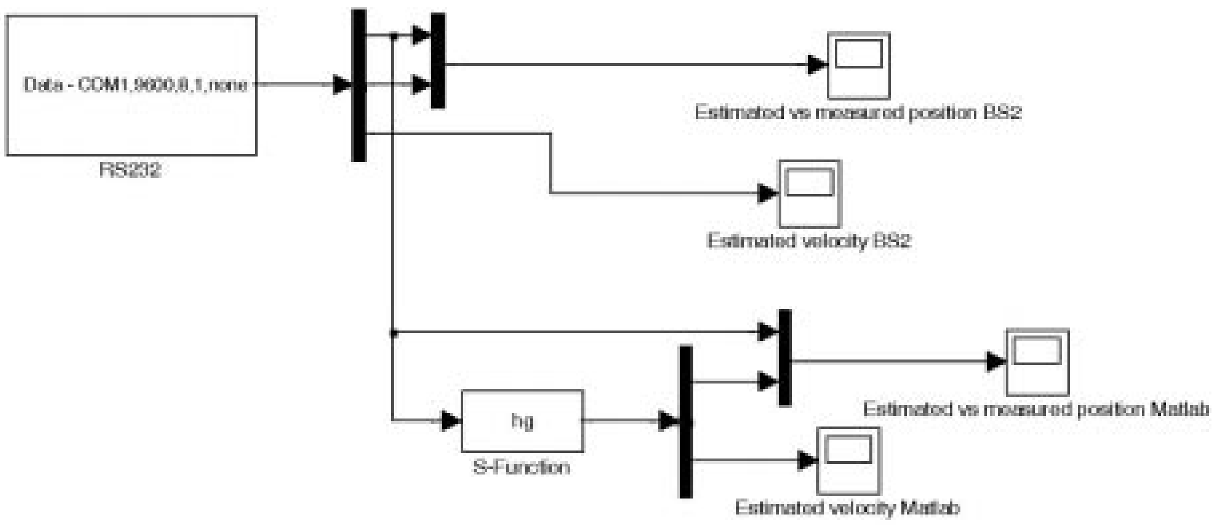
Data - COM1,9600,8,1,none
RS232
Estimated vs measured position BS2
Estimated velocity BS2
Estimated vs measured position Matlab
hg
S-Function
Estimated velocity Matlab

Fig. 4. Simulink scheme for comparison of the high-gain observer implemented in Simulink with the one implemented with the BS2.

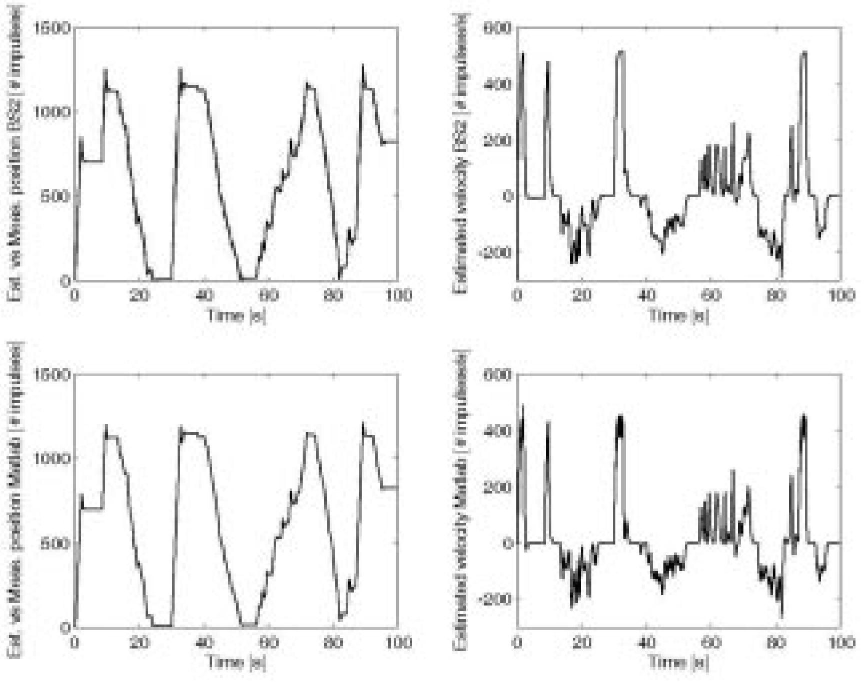

Fig. 5. High-gain observer implemented both with Simulink and BS2: comparisons between the respective time histories.

ASSIGNMENT 2: MODEL DESIGN

The second assignment is the design of the input-output dynamic model of the experimental set-up. The model is very simple (see Fig. 6) and comprises various linear and non-linear blocks: input saturation, dead-zone, Coulomb and viscous friction, sampling and a two-poles linear transfer function (as usual in low frequency applications, the high frequency electrical mode is omitted in the description of the linear part):

$$\frac{K_M}{s(s+\lambda_M)}.$$

The upper and lower limits (U_S and L_S) of the input saturation, the start and the end (S_{DZ} and E_{DZ}) of the dead zone, the offset O_C of the Coulomb friction value, the coefficient C_F of viscous friction, and the two parameters K_M and λ_M of the linear system are not provided in the motor data sheet.

The Laboratorio d'Automatica students have no prior knowledge about system identification, and the presence of nonlinear terms makes the theoretical identification problem less trivial. A simple solution is to ask the students to apply the same real-time input both to the experimental set-up and to the model simulated, with a starting guess for the unknown parameters. Then, by comparing the two responses in a series of trials, the students have to tune the parameters in order to obtain pretty much the same behaviour (Fig. 7).

With the "Signal Generator" block, the students can graphically define different inputs in order to appreciate the differences between actual and simulated responses as functions of the model parameters. After a few trials, a set of fairly representative parameters can be obtained (Table 1).

Table 1.

U_S	115
L_S	−115
S_{DZ}	−35
E_{DZ}	35
O_C	20
C_F	1
K_M	40
λ_M	10

Fig. 8 shows the comparison between the measured and simulated time histories (which are similar) obtained with the estimated parameters, when the voltage input is a ramp from 0V to +12V in 100 seconds. In Fig. 9 one can see the

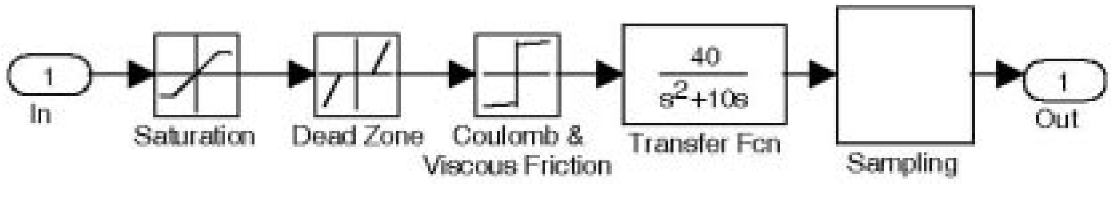

Fig. 6. The Simulink model of the DC motor.

comparison of the measured and simulated time histories (which also in this case are similar) obtained with the same parameters, when the input voltage is the piecewise continuous function reported on the left of Fig. 9.

Notice that the very fast oscillations present in both estimated velocities are due to sampling; the students can appreciate this fact by omitting the "Sampling" block in the model of Fig. 6; in fact, in the absence of such a block, the oscillations disappear from the velocities estimated from the model.

ASSIGNMENT 3: DESIGN AND SIMULATION OF SIMPLE CONTROL LAWS IN SIMULINK

Using the model of the previous section, the Students are asked to design simple control laws

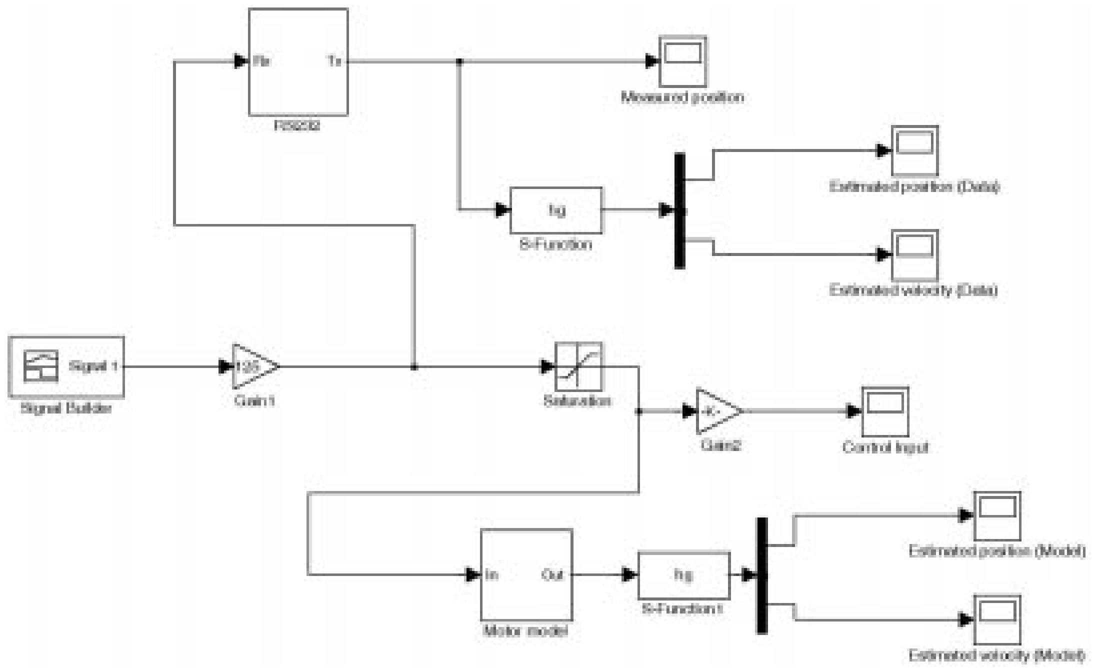

Fig. 7. Simulink scheme for tuning the model parameters.

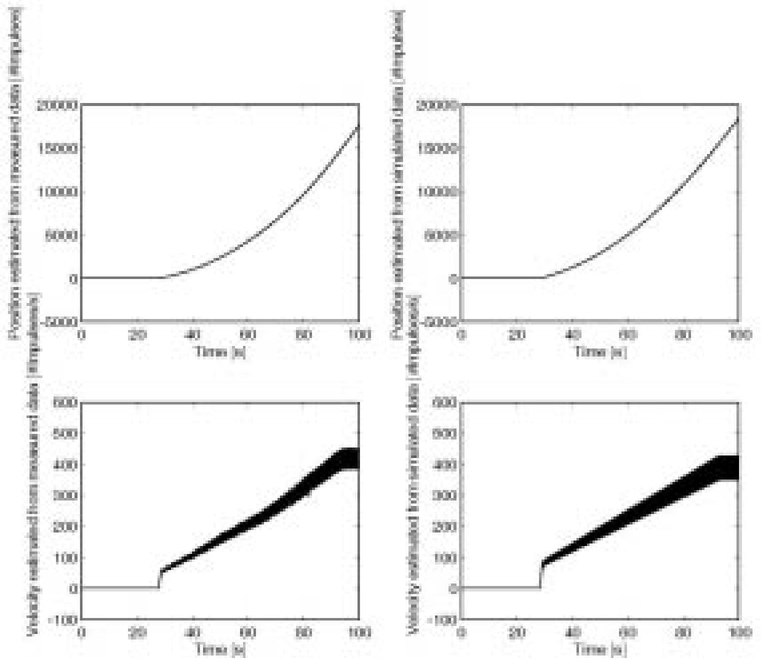

Fig. 8. Comparison of measured and simulated time histories (position and velocity) when the input voltage is a ramp from 0V to +12V in 100s.

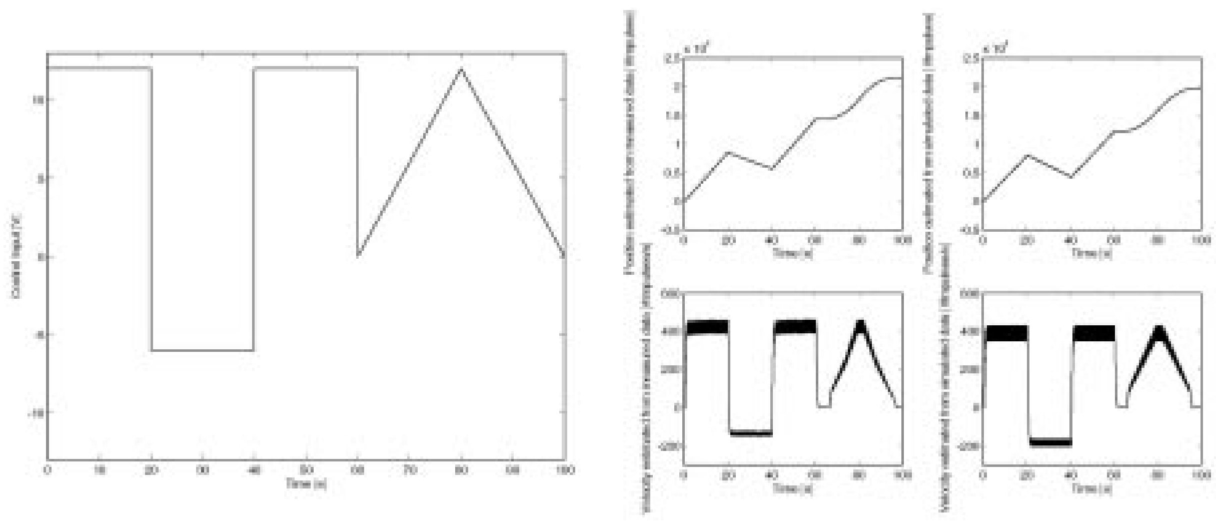

Fig. 9. Comparison of measured and simulated time histories: piecewise continuous input.

(standard regulators and sliding mode control), and to prove their effectiveness in simulation before implementing them on the experimental apparatus.

Proportional control law

The first standard regulator analysed by the students is a proportional control law

$$u(t) = -K_P(x_1(t) - r(t)),$$

where $x_1(t)$ is the output of the "Motor model" block and $r(t)$ is the reference signal generated with the "Signal Builder" block; the regulator has been implemented with the Simulink scheme shown in Fig. 10, with $K_P = 1$. With this control law, the student encounters, for the first time, two important notions in basic control theory: the comparison of the actual position with its reference value and negative feedback. In addition, the student can appreciate the fact that the high viscous friction present in the experimental apparatus allows one to omit the derivative action. With the "Signal Builder" block, the student can generate different reference signals, observing the respective time responses, possibly under different values of K_P. The high-gain observer is always used for velocity estimation.

The simulated results are reported in Fig. 11 and the student can learn from this figure that a

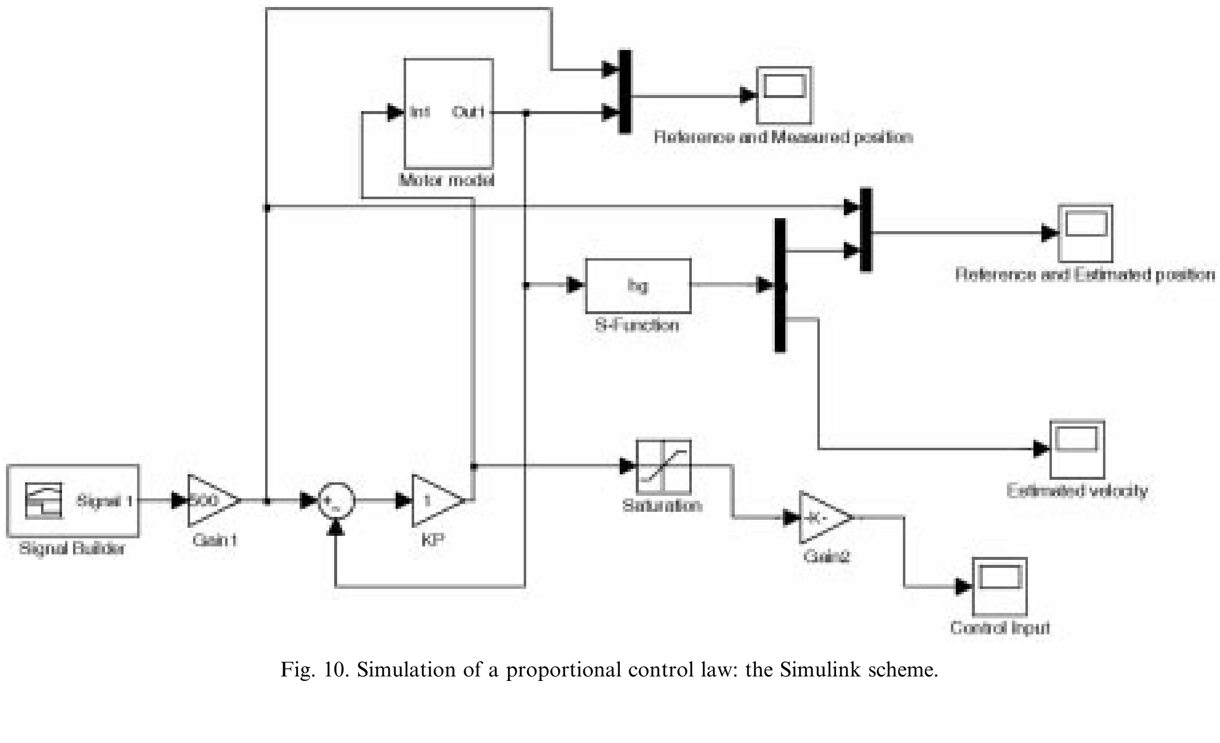

Fig. 10. Simulation of a proportional control law: the Simulink scheme.

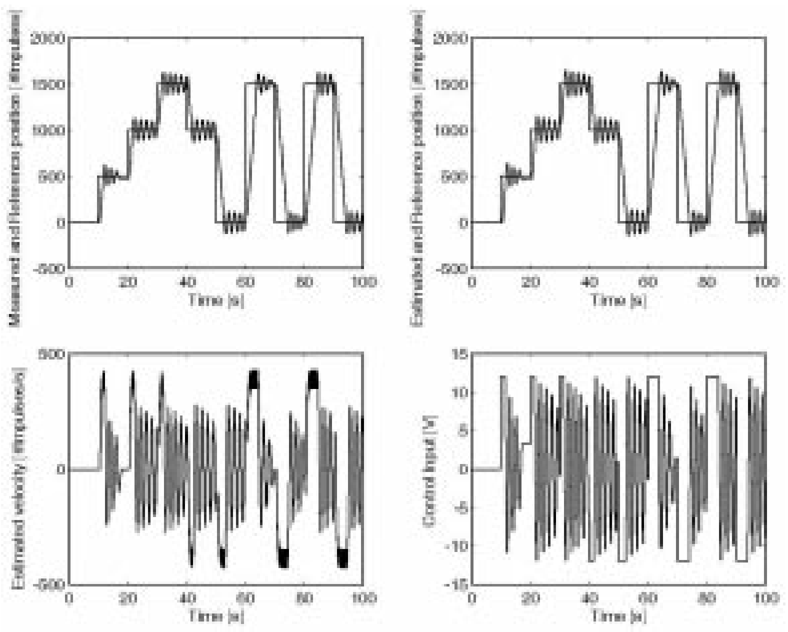

Fig. 11. Simulation of a proportional control law (K_P=1): time histories.

sufficiently high proportional coefficient is sufficient to obtain a small "steady-state" error (for the intervals in which the reference inputs are constant).

PID control law

The second standard regulator analysed by the students is a PID control law, implemented on the basis of the measured position and the estimated velocity supplied by the high-gain observer (see Fig. 12):

$$u(t) = -K_P(x_1(t) - r(t)) - K_I\int_0^t (x_1(\theta) - r(\theta))d\theta - K_D\hat{x}_2(t),$$

where $x_1(t)$ and $\hat{x}_2(t)$ are, respectively, the output of the "Motor model" block and the estimate of its time derivative, and $r(t)$ is the reference signal generated with the "Signal Builder" block. The chosen values of the regulator parameters are $K_P = 0.1$ (ten times less than the value of K_P

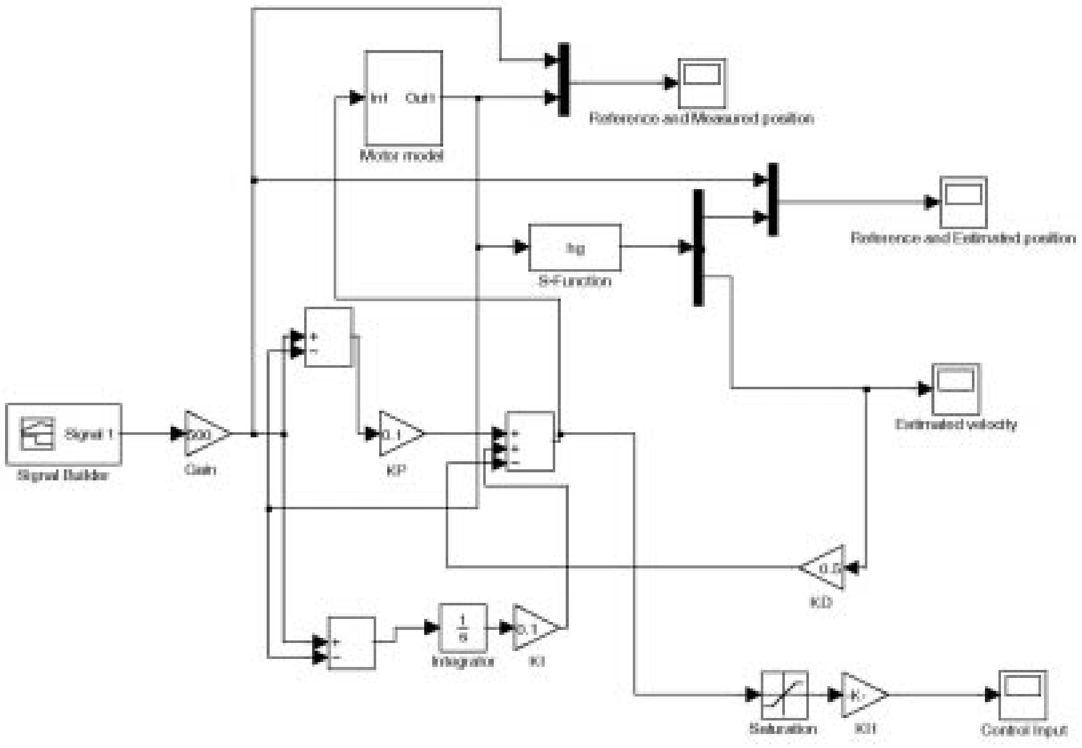

Fig. 12. Simulation of a PID control law: the Simulink scheme.

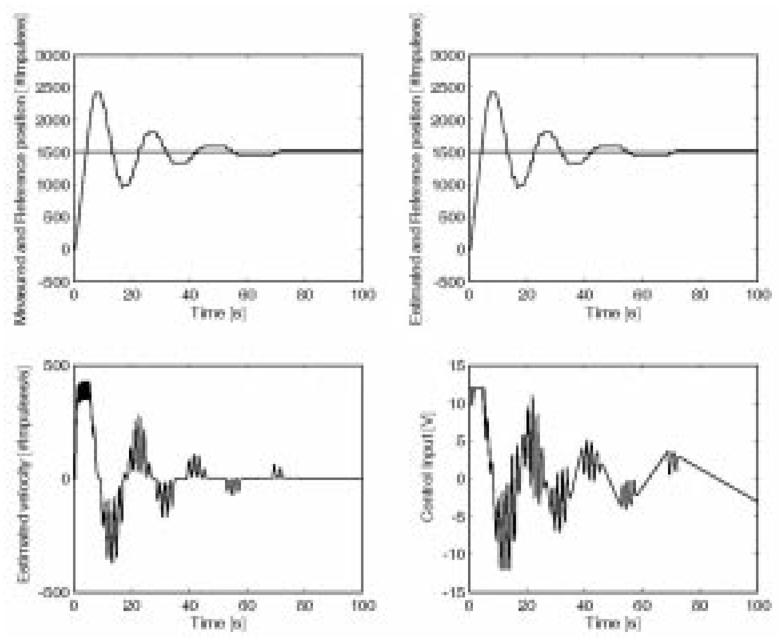

Fig. 13. Simulation of a PID control law: time histories K_P=0.1, K_I=0.1 and K_D=0.5.

chosen in the case of the proportional controller), $K_I = 0.1$ and $K_D = 0.5$.

The simulated results are reported in Fig. 13, in the case of a high constant value for the reference position; from such a figure, the student can discover that, although the proportional coefficient is ten times less than the value chosen for the proportional control law, the presence of the integral action (used in conjunction with a suitable derivative action, to preserve stability) allows one to obtain an almost zero steady-state error at the end of the simulation interval. In addition, the student can observe that the cost to be paid for having this property is a high overshoot.

Sliding mode control

Finally, the third implemented control law analysed by the students is a sliding mode regulator:

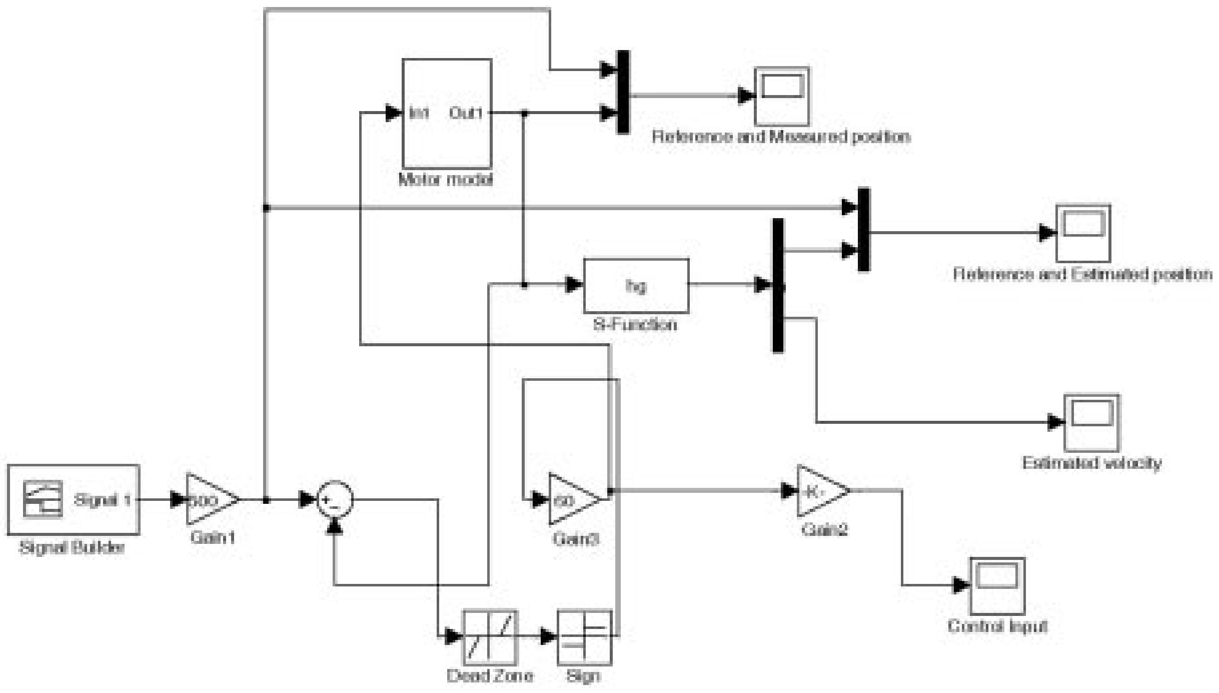

Fig. 14. Simulation of a sliding mode control law: the Simulink scheme.

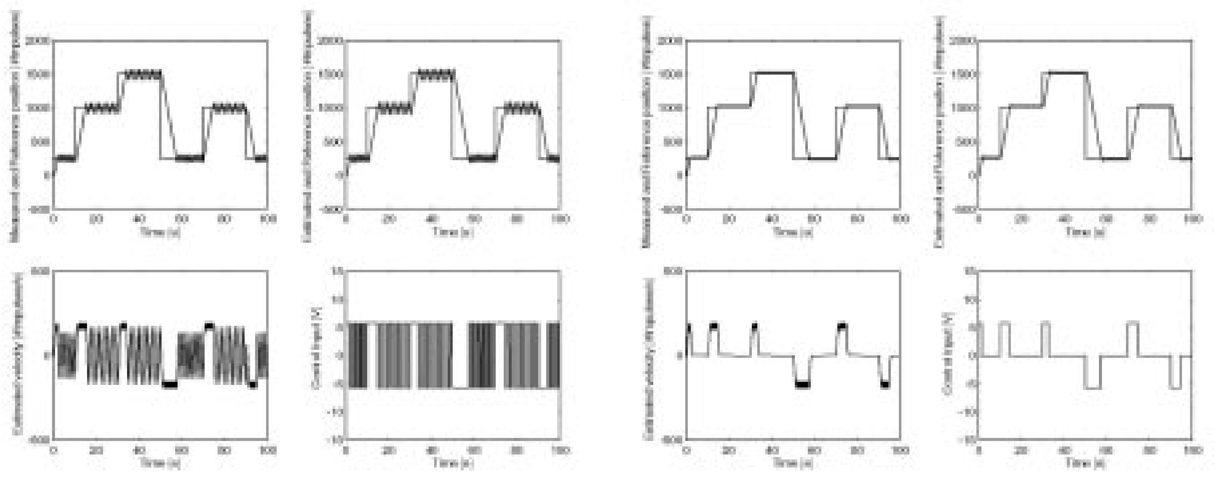

Fig. 15. Simulation of a sliding mode control law: time histories with (on the left) and without (on the right) dead-zone.

$$u(t) = \begin{cases} K_S & \text{if} \quad r(t) - x_1(t) > \Delta \\ 0 & \text{if} \quad abs(r(t) - x_1(t)) < \Delta \\ -K_S & \text{if} \quad r(t) - x_1(t) < \Delta \end{cases}$$

where K_S is a constant parameter chosen so that when $u(t) = K_S$ the actual voltage applied to the motor is 6V, and 2Δ is the amplitude of the dead-zone. Such a control law has been implemented with the Simulink scheme shown in Fig. 14, where $r(t)$ is generated by the "Signal Generator" block and $x_1(t)$ is the output of the "Motor model" block.

This control law is intuitively appreciated by the students in the following sense: if `MPosition` must be increased to reach `RPosition`, then the motor shaft must rotate clockwise, whereas if `MPosition` must be decreased to reach `RPosition`, then the motor shaft must rotate counter-clockwise; otherwise (if `MPosition` is close to `RPosition`), the motor must rest. The voltage input must be applied accordingly.

Two different simulation tests can be carried out: the first one without the dead-zone and the second one with a sufficiently large dead-zone (the corresponding time histories are shown in Fig. 15). The difference between the two strategies can be clearly seen from this figure: without the dead-zone there is a relevant chattering, whereas with a sufficiently large dead-zone the chattering disappears, but this feature is paid with a no-null (a sometimes relevant) steady-state error.

ASSIGNMENT 4: SIMULINK IMPLEMENTATION OF CONTROL LAWS

Once a control law has been tested in simulation, its Simulink real-time implementation on the experimental apparatus is trivial: the "Motor model" block must be replaced with a serial communication block with the `BS2`, for which the `Reading` and `Control` subroutines consist simply in serial communications and data interpretation, whereas the observer subroutine is not called on at all:

```
RxD     CON 15    ' PIN15
TxD     CON 16    ' PIN16
Baud96  CON 84    ' 9600-8-N-1
CInput  VAR Word  ' Signed control input
'-----------
Reading:
 SERIN RxD, Baud96, [WAIT(''A''),SDEC
      CInput]
RETURN

Control:
 IF CInput<30000 THEN
    Direction=0
    ControlInput=CInput MAX 125
 ELSE
    Direction=1
    ControlInput=(-CInput) MAX 125
 ENDIF
 DEBUG ''start;'',DEC MPosition,'';end''
RETURN
```

The Simulink program transmits the data to the `BS2` with the format "`A%d;`", whereas it receives the data from the `BS2` with the format "`start;%d;end`".

Proportional control law

The Simulink scheme for the real-time implementation of a proportional control law is simply obtained by the scheme reported in Fig. 10, by replacing the "Motor model" block with a serial communication block with the `BS2`, thus obtaining the Simulink scheme reported in Fig. 16.

Various experimental tests can be carried out with this Simulink scheme (see Fig. 17); in particular, it is interesting to relate the time responses of the system with the proportional coefficient K_P. In particular, increasing the value of K_P from 0.1 to 1, the student can see how the steady-state error decreases as K_P increases. In addition, for $K_P = 1$ we obtain time histories similar to those obtained in the simulation reported in Fig. 17.

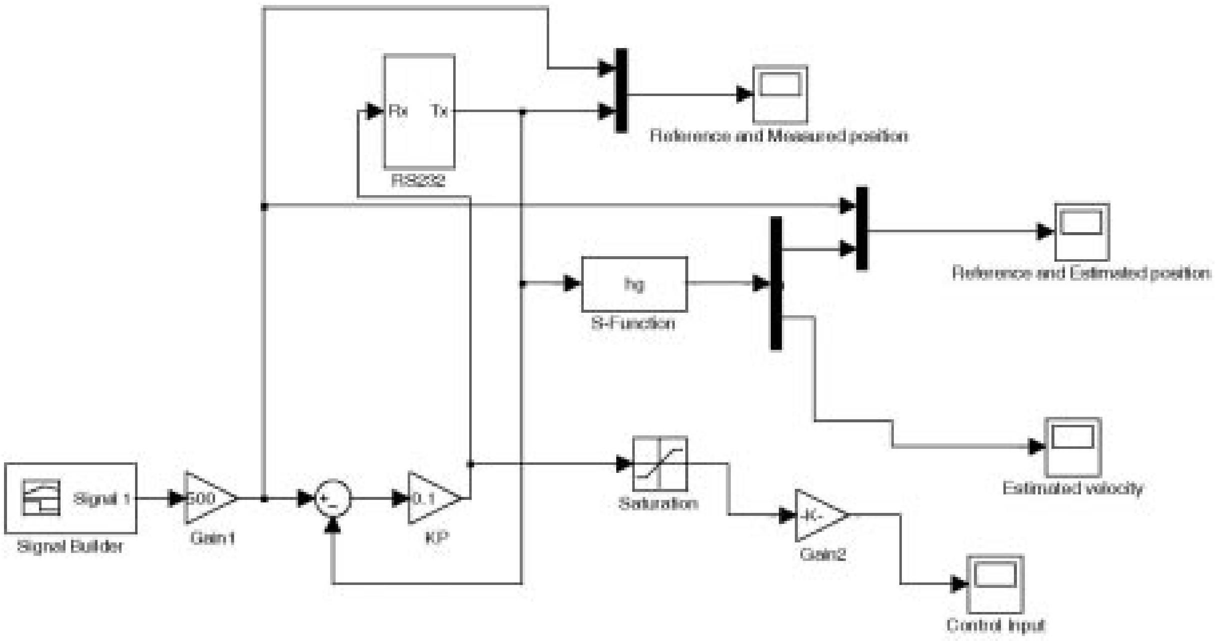

Fig. 16. Real-time control from Simulink of the DC motor with a proportional control law: the Simulink scheme.

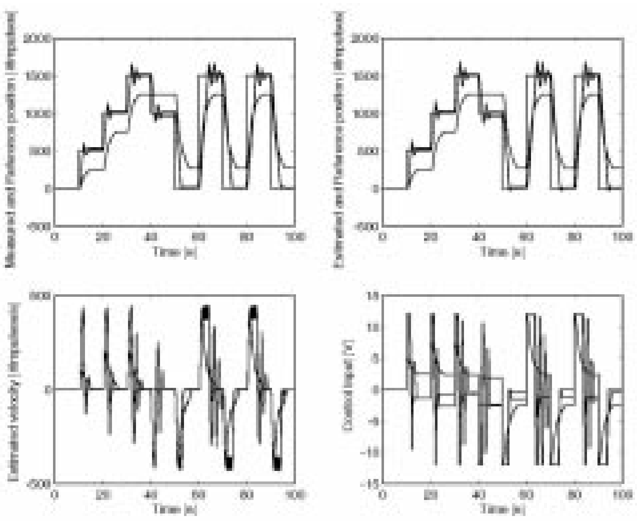

Fig. 17. Real-time control from Simulink of the DC motor with a proportional control law: comparison of the time histories corresponding to K_P=0.1, K_P=0.5 and K_P=1.

PID control law

The Simulink scheme for the real-time implementation of a PID control law is simply obtained by the scheme reported in Fig. 12, by replacing the “Motor model” block with a serial communication block with the BS2, thus obtaining the Simulink scheme shown in Fig. 18.

Various experimental tests can be carried out with this Simulink scheme (see Fig. 19): in particular, it is interesting to relate the time responses of the system with the derivative coefficient K_D. In particular, by increasing the value of K_D from 0 to 1, the student can see how the overshoot decreases as K_D increases. In addition, for $K_D = 0.5$, we obtain time histories similar to those obtained in the simulation shown in Fig. 13.

Sliding mode control

The Simulink scheme for the real-time implementation of a sliding mode control law is simply obtained by the scheme reported in Fig. 14, by replacing the “Motor model” block with a serial communication block with the BS2, thus obtaining the Simulink scheme reported in Fig. 20.

Two different experimental tests have been carried out: the first one without the dead-zone and the second one with a sufficiently large dead-zone (the time histories are reported in Fig. 21). As

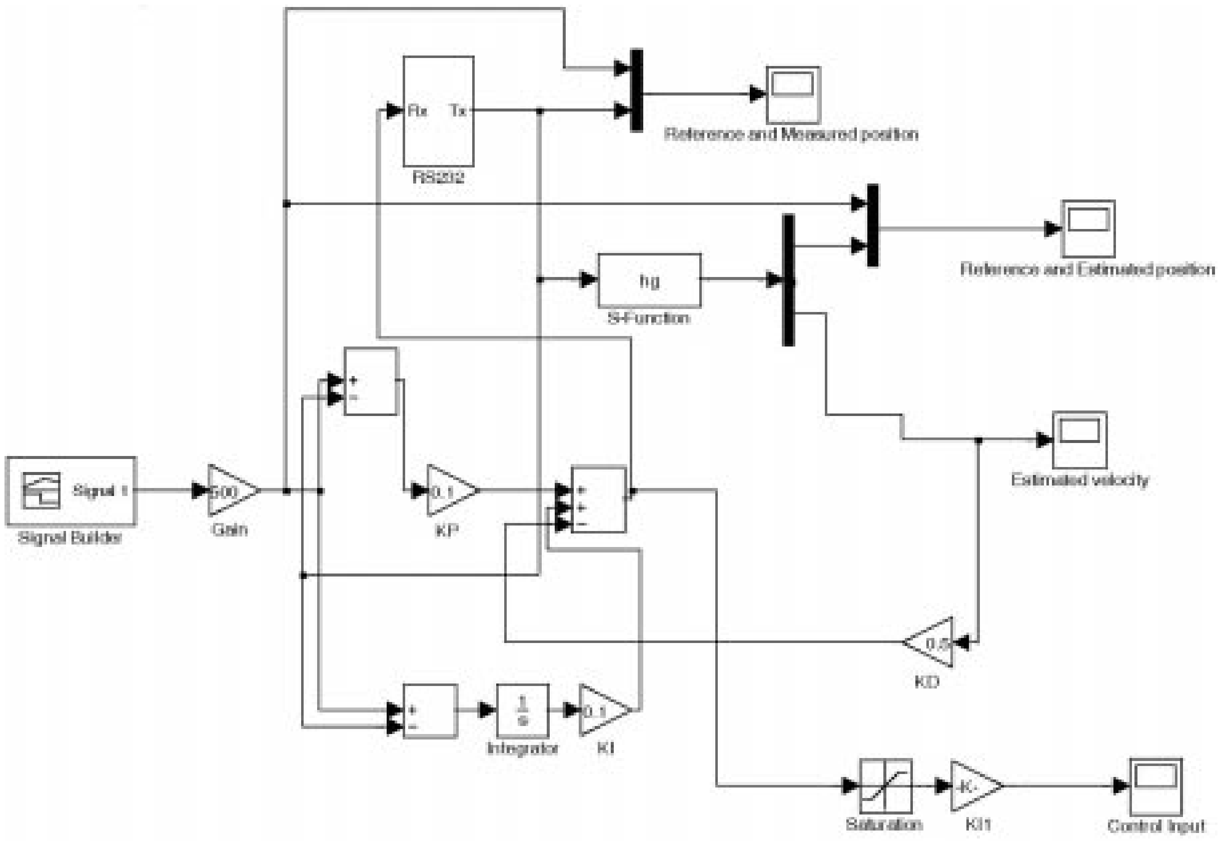

Fig. 18. Real-time control from Simulink of the DC motor with a PID control law: the Simulink scheme.

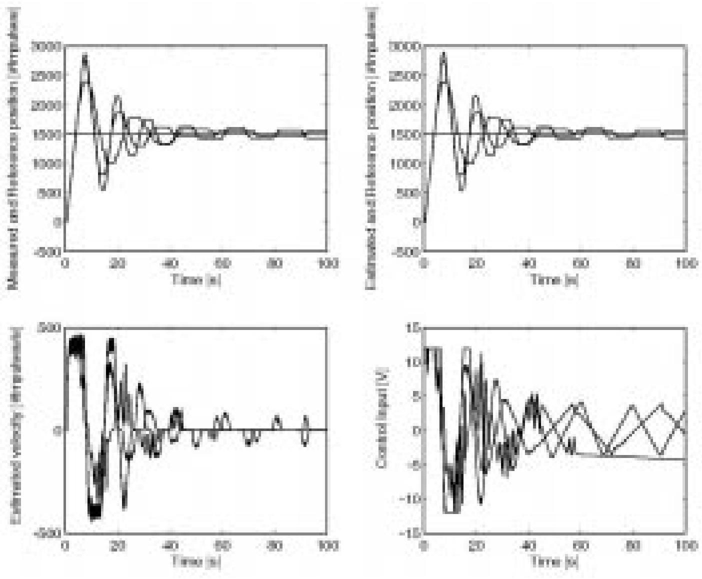

Fig. 19. Real-time control from Simulink of the DC motor with a PID control law: comparison of the time histories corresponding to K_D=0, K_D=0.5 and K_D=1.

with the simulated case (notice that the time histories of Fig. 21 are pretty much the same as those reported in Fig. 15), the difference between the two strategies can be clearly seen from these figures: without the dead-zone there is a relevant chattering, whereas with a sufficiently large dead-zone the chattering disappears, but this feature is paid with a non-null (and sometimes relevant) steady-state error.

ASSIGNMENT 5: BS2 IMPLEMENTATION OF CONTROL LAWS

In Assignment 3 various control laws were designed and tested on the basis of a simple dynamic model; then, the same control strategies were experimentally validated implementing them directly in Simulink for Assignment 4. Now, it is

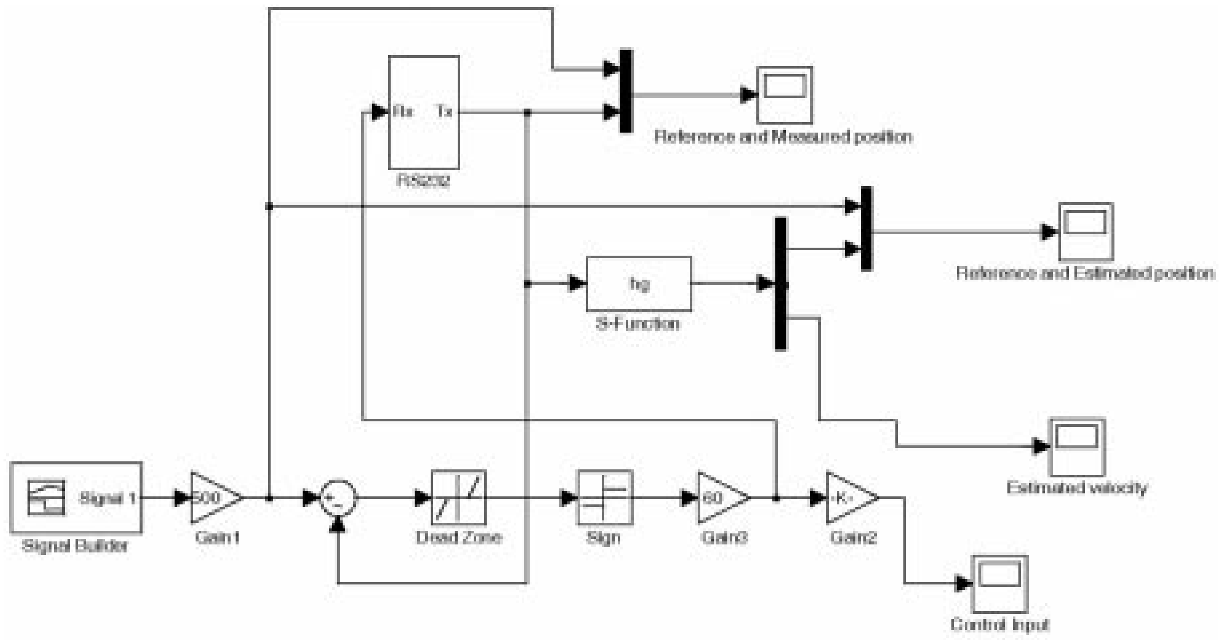

Fig. 20. Real-time control from Simulink of the DC motor with a sliding mode control: Simulink scheme.

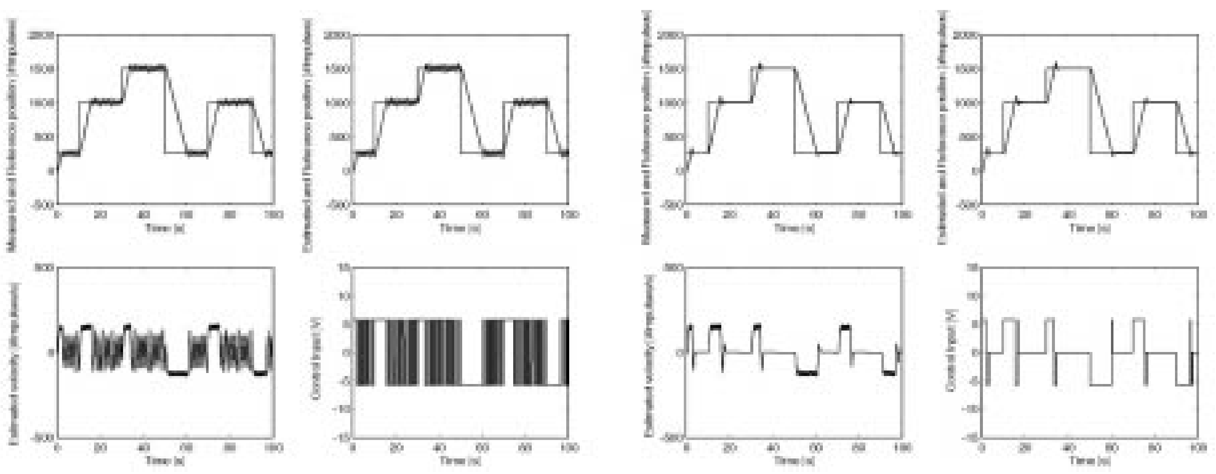

Fig. 21. Real-time control from Simulink of the DC motor with a sliding mode control: time histories obtained with and without the dead-zone.

time to implement such control laws directly on the microcontroller BS2.

The Simulink scheme for the position control is reported in Fig. 22, which now has the only objective of receiving data from the BS2 and estimating motor shaft velocity (the Simulink scheme could be removed if we were not interested in recording the position of the motor shaft).

Proportional control law

The procedure implemented on the BS2 for real-time control of the DC motor with a proportional control law is the following:

```
MPosition VAR Word ' Measured position
RPosition VAR Word ' Reference position
Direction VAR Bit ' Clockwise (0),
    Counter clockwise (1)
ControlInput VAR Byte ' Voltage to be
    applied to the DC motor (from 0 to 255)
SControlInput VAR Word ' Signed voltage
    to be applied to the DC motor

ControlInputOld VAR Byte ' Old value of
    Voltage,
KP CON 2 ' Proportional gain

Control:
IF (MPosition>RPosition) THEN
   Direction=1
   ControlInputOld=KP*(MPosition-
     RPosition) MAX 255
   SControlInput=300-ControlInputOld
   ELSE
   Direction=0
```

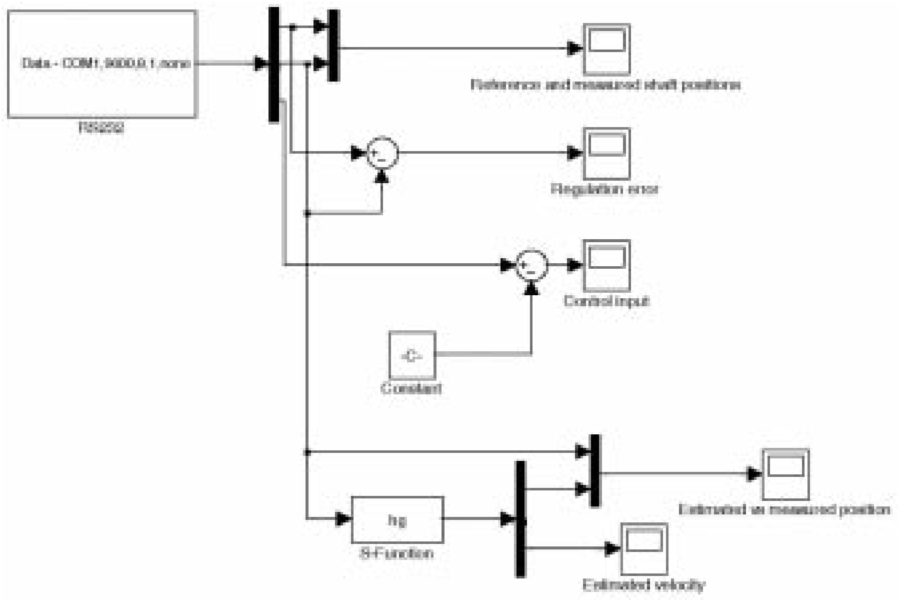
Data - COM1,9600,8,1,none
RS232
Reference and measured shaft positions
Regulation error
Control input
-C-
Constant
hg
S-Function
Estimated vs measured position
Estimated velocity

Fig. 22. Simulink implementation of data acquisition and high-gain observer design.

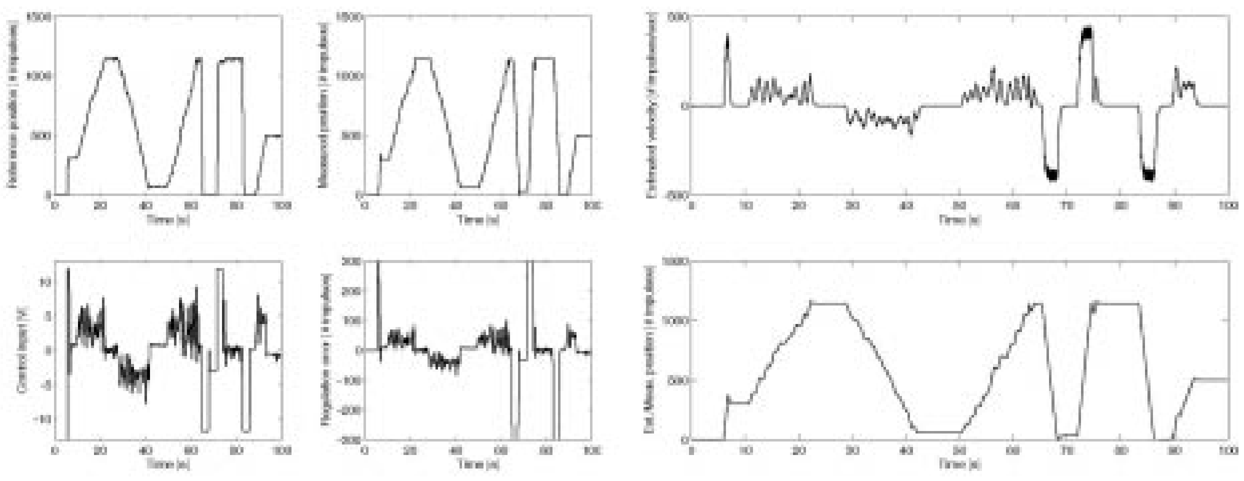

Fig. 23. Proportional control law.

```
    ControlInputOld=KP*(RPosition-
      MPosition) MAX 255
    SControlInput=300+ControlInputOld
ENDIF
ControlInput=ControlInputOld/2
RETURN
```

The reference value `RPosition` (which is controlled by manually turning the shaft of the analogue potentiometer) is read from the `BS2`. Some experimental results are reported in Fig. 23, which shows the manually defined reference position, the measured position, the control input, the regulation error, and the estimates obtained in real time of position and velocity, the former being practically coincident with the measured one.

With this experimental test, it is very interesting for the student to manually turn the reference potentiometer shaft, thus experimenting with the effects of various movements on the real motion of the motor shaft.

Motion planning

The disadvantage of manually defining the reference input is that the same experiment cannot be repeated exactly. With this experiment the student is asked to *a priori* define a desired motion and then to implement it on the `BS2` through a sequence of instructions. This has been realized with the following procedures:

```
Index1  VAR Word ' Index
Index2  VAR Word ' Index
Maximum VAR Word ' Index
'------------------
Maximum = 47
RPosition = 500
GOSUB Main
```

```
Maximum = 46
RPosition = 1000
GOSUB Main
Maximum = 45
RPosition = 1500
GOSUB Main
Maximum = 45
RPosition = 500
GOSUB Main
Index2=0
Maximum = 1

DO WHILE Index2<90
 Index2=Index2+1
 GOSUB Main
 RPosition=RPosition+11
LOOP
Index2=0
Maximum = 1
 DO WHILE Index2<90
  Index2=Index2+1
  GOSUB Main
  RPosition=RPosition-11
 LOOP
Maximum = 45
RPosition = 1000
GOSUB Main
Maximum = 46
RPosition = 0
GOSUB Main
END

Main:
Index1=0
DO WHILE (Index1 < Maximum)
 Index1=Index1+1
 GOSUB Measure
 GOSUB Control
 DEBUG ''start;'',DEC RPosition,'','',
 DEC MPosition,'','',DEC
     SControlInput,'';end''
 GOSUB Measure
 GOSUB Potentiometer0
 GOSUB Potentiometer1
 GOSUB Measure
LOOP
RETURN
```

The experimental results are reported in Fig. 24.

Comparison of proportional and proportional/ derivative control laws

The procedure implemented on the BS2 for the real-time control of the DC motor with a proportional/derivative control law is the following:

```
x1  VAR Word ' Reference position
x2  VAR Word ' Reference position
RPosition  VAR Word ' Reference position
ControlInput VAR Byte ' Voltage to be
     applied to the DC motor (from 0 to 255)
SControlInput VAR Word ' Signed voltage to
     be applied to the DC motor
ControlInputOld VAR Word ' Old value of
     Voltage, to be used for possible
     filtering
Direction VAR Bit ' Clockwise (0),
     Counter clockwise (1)
KP  CON 2 ' Control gain
KD  CON 0 ' Control gain (0/1)
DKD  CON 10 ' Control gain

Control:
IF (x1>RPosition) THEN
ControlInputOld=-((KP*(x1-RPosition))
     MAX 255)
ELSE
ControlInputOld=KP*(RPosition-x1) MAX
     255
ENDIF
IF (x2<32766) THEN
ControlInputOld=ControlInputOld-
     (KD*x2/DKD)
ELSE
ControlInputOld=ControlInputOld+(KD*(-
     x2)/DKD)
ENDIF

IF ControlInputOld < 32766 THEN
Direction=0
ControlInput=(ControlInputOld MAX 255)/2
SControlInput=300+ControlInput
ELSE
Direction=1
ControlInput=((-ControlInputOld) MAX
     255)/2
SControlInput=300-ControlInput
ENDIF
RETURN
```

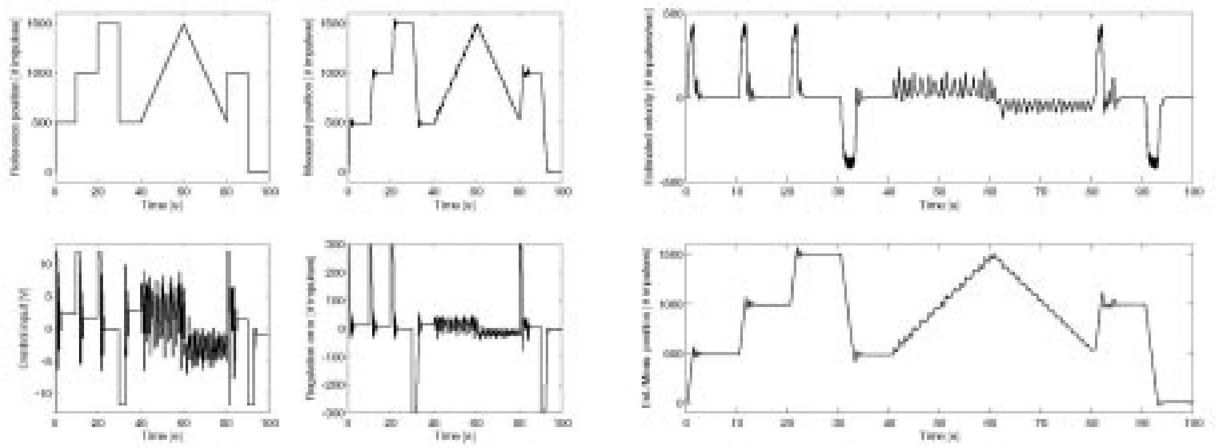

Fig. 24. Motion planning implemented on the BS2.

The Simulink scheme for the position control is reported in Fig. 25.

With this experiment the student can discover the differences between a P and a PD control; by comparing the time histories reported in Figs. 26 and 27, the student can see that the control input is in saturation for a shorter period when a PD is used (the steady-state values are similar in both cases, due to the chosen value of K_P).

Sliding mode

The procedure implemented on the `BS2` for real-time control of the DC motor with a sliding mode control law is the following:

```
ControlInput VAR Byte ' Voltage to be
    applied to the DC motor (from 0 to 255)
SControlInput VAR Word ' Signed voltage to
    be applied to the DC motor
ControlInputOld VAR Byte ' Old value of
    Voltage, to be used for possible
    filtering
Direction VAR Bit ' Clockwise (0),
    Counter clockwise (1)
MPosition VAR Word ' Measured position
RPosition VAR Word ' Reference position
Delta  CON 20 ' Gap

Control:
IF (ABS(MPosition-RPosition)>Delta) AND
    (MPosition>RPosition) THEN
 Direction=1
 ControlInputOld=100
 SControlInput=300-ControlInputOld
ELSEIF (ABS(MPosition-RPosition)>Delta)
    AND (MPosition<RPosition) THEN
 Direction=0
 ControlInputOld=100
 SControlInput=300+ControlInputOld
ELSE
```

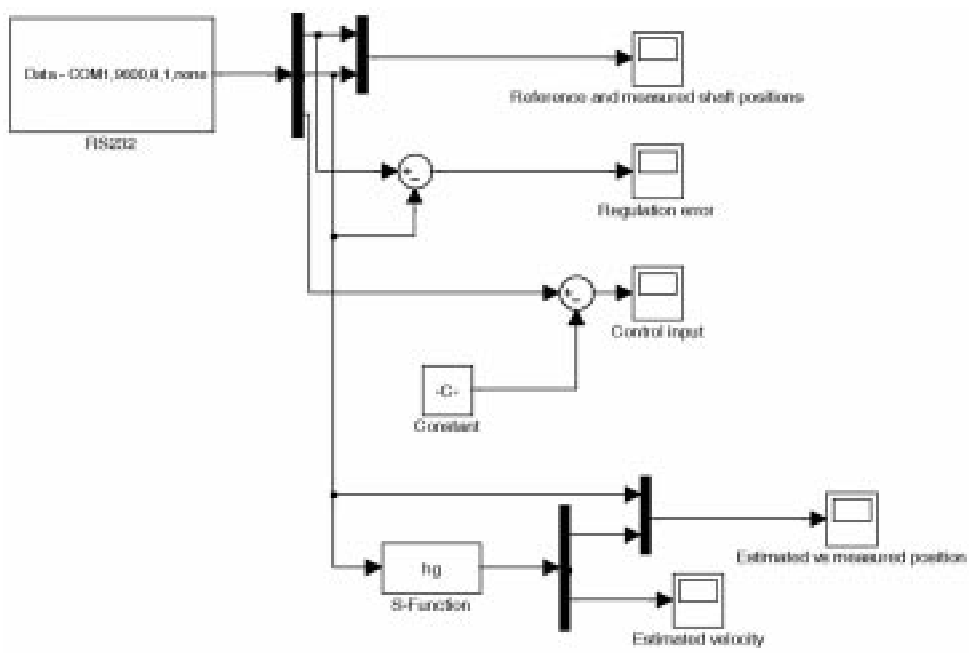
Data - COM1,9600,8,1,none
RS232
Reference and measured shaft positions
Regulation error
Control input
-C-
Constant
hg
S-Function
Estimated vs measured position
Estimated velocity

Fig. 25. Simulink diagram.

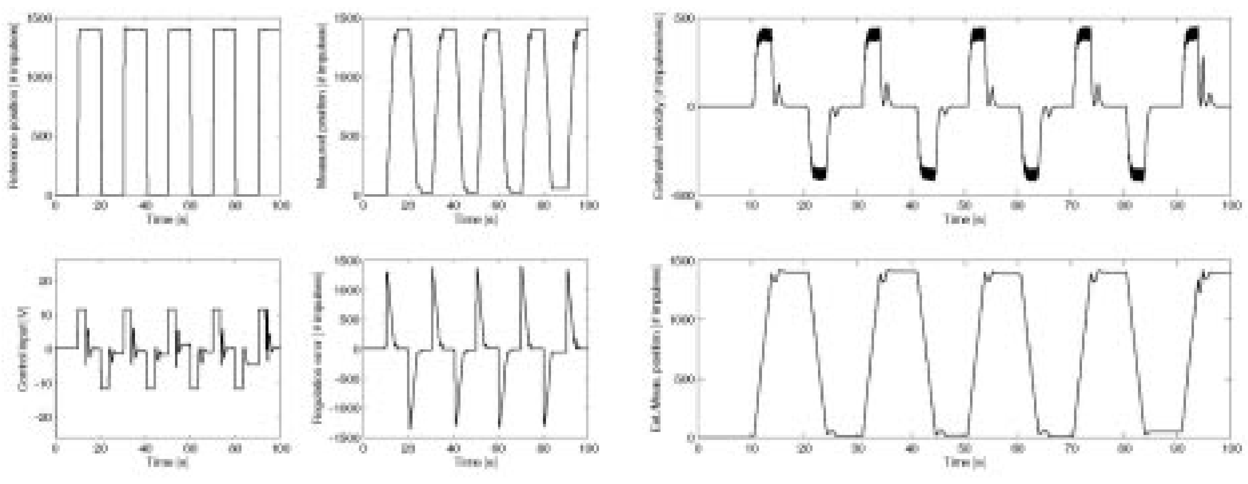

Fig. 26. Proportional control law (K_D=0).

```
 ControlInputOld=0
 SControlInput=300
ENDIF
ControlInput=ControlInputOld/2
RETURN
```

Two different experimental tests were carried out: the first one without the dead-zone and the second one with a sufficiently large dead-zone (the time histories are reported in Figs. 28 and 29, respectively). As with the simulated case, the difference between the two strategies can be clearly seen from these figures: without the dead-zone there is a relevant chattering, whereas with a sufficiently large dead-zone the chattering disappears.

Speed control

In order to consider a PI control, the regulation of the velocity of the DC motor to a constant value (speed control) has been considered. The procedure implemented on the `BS2` for real-time control of the DC motor with a speed control law is the following:

```
Measure:
 MVelocityOld=(6*MVelocity)
 HIGH RES
 LOW RES
RETURN

Reading:
HIGH Potentiometer  ' Pin 13 is High
PAUSE 1   ' 1 ms pause
RCTIME Potentiometer,1,RPosition
    ' Discharge time measurement
RPosition=(45*RPosition)/64 ' Scaling
RETURN

Control:
Integral=Integral+(MVelocityOld)-
    (RPosition)
IF Integral<32766 THEN
 UIntegral=KI*Integral/DKI
ELSE
 UIntegral=-(KI*(-Integral)/DKI)
ENDIF
IF ((MVelocityOld-RPosition)<32766)
    THEN
  ControlInputOld=-
    ((KP*(MVelocityOld-RPosition)/
    DKP) MAX 255)
```

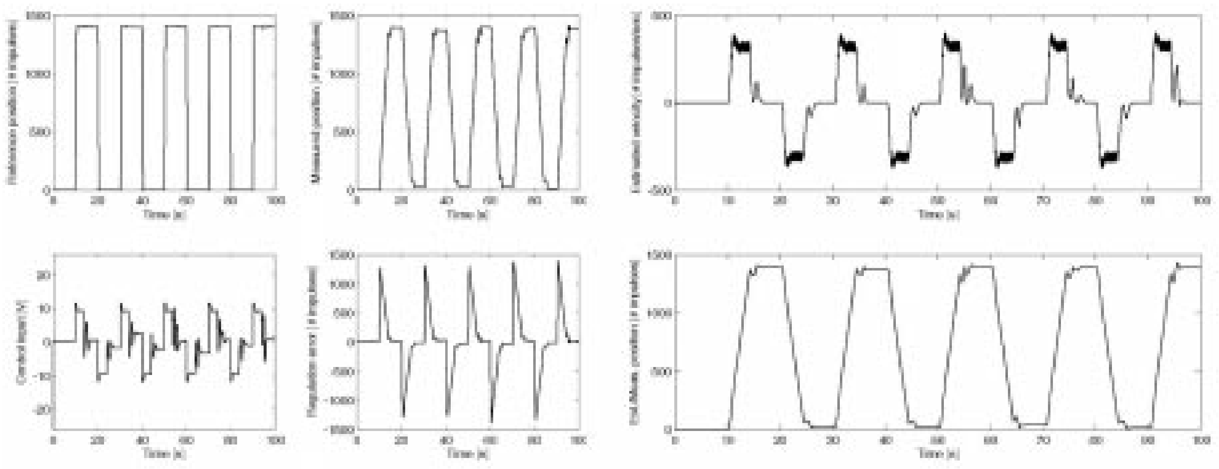
Fig. 27. Proportional/derivative control law (K_D=1).

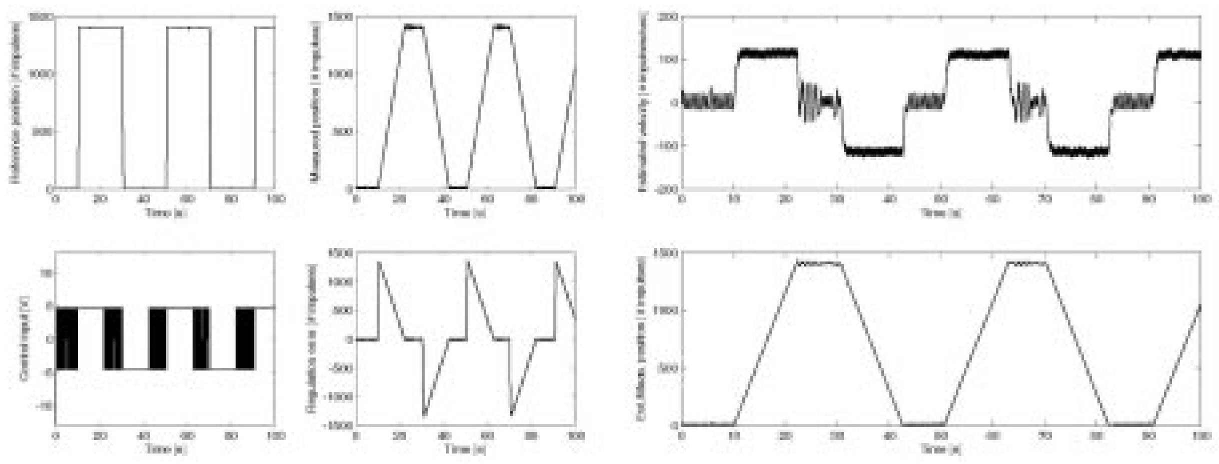
Fig. 28. Sliding mode without dead-zone.

```
   ELSE
  ControlInputOld=KP*(RPosition-
      MVelocityOld)/DKP MAX 255
ENDIF
ControlInputOld=ControlInputOld-
      UIntegral
IF ControlInputOld < 32766 THEN
   Direction=0
   ControlInput=(ControlInputOld MAX
      255)
   SControlInput=300+ControlInput
     ELSE
   Direction=1
   ControlInput=((-ControlInputOld) MAX
      255)
   SControlInput=300-ControlInput
ENDIF
RETURN
```

The Simulink scheme for speed control is shown in Fig. 30.

Students can see the effects of the integral action by comparing the time histories of the same control law, with and without the integral action, which are reported in Fig. 31. Due to the small value of the proportional coefficient, the steady-state values with or without the integral action are very different; the student can thus appreciate how the integral action is useful for obtaining zero steady-state error, for constant values of the reference velocity.

Virtual reality

Finally, Fig. 32 shows the VRML model of the DC motor and of the encoder; the encoder shaft is directly joined to the motor shaft. Such a VRML model can be driven by the data acquired in real time by BS2/Simulink, so that the students can see the reproduced virtual movement in parallel with the real movement of the motor.

DESCRIPTION OF THE DEVICES USED AND OF THE MAIN PROCEDURES

The BASIC Stamp II

The BASIC Stamp II [1] is a hybrid microcontroller designed to be programmed in a version of the BASIC programming language called PBASIC (version 2.5 of the PBASIC language is used, with the freeware BASIC STAMP Editor 2.0 for programming). The BS2 consists of: (1) serial signal conditioning between the PC serial ($\pm$12V) and the BASIC Stamp (+5V); (2) a voltage regulator to +5V with a supply of +5.5V (DC) to +15V (DC); (3) a resonator, which sets the speed at which instructions are processed; (4) EEPROM, which stores the tokenised PBASIC program; (5) an interpreter chip, which reads the BASIC

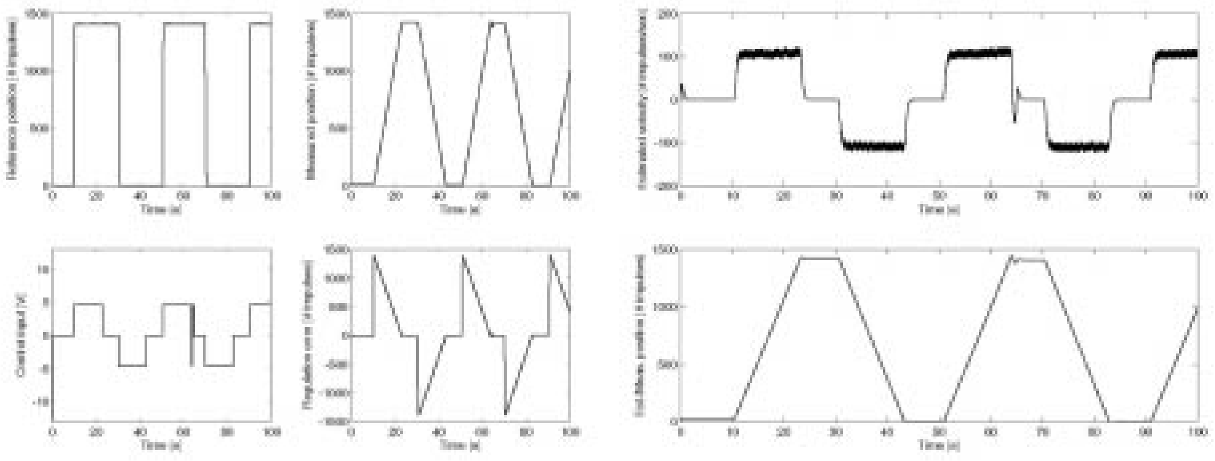

Fig. 29. Sliding mode with dead-zone.

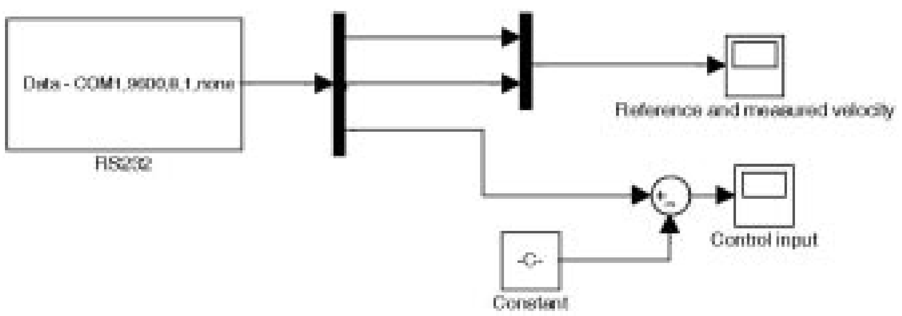

Fig. 30. Simulink interface.

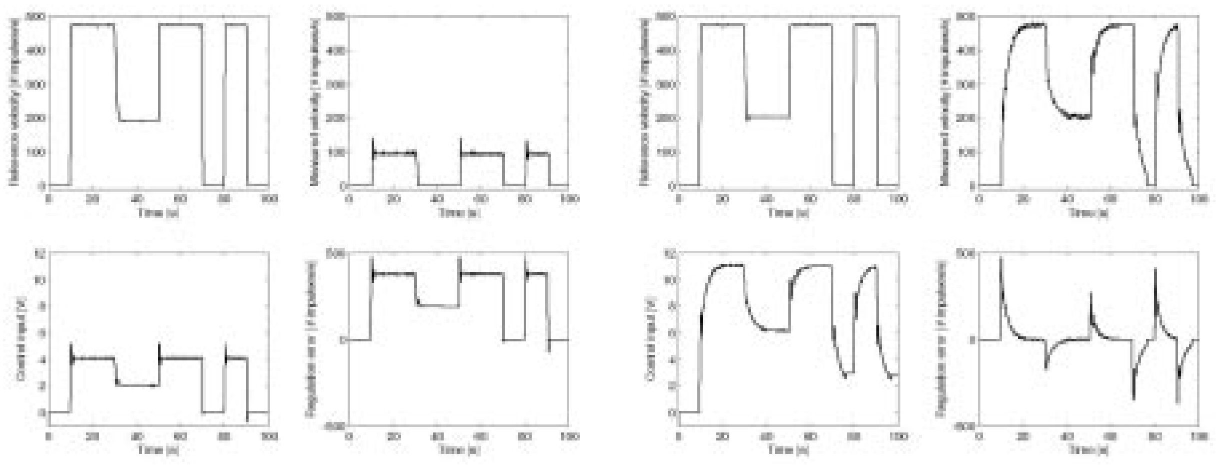

Fig. 31. Speed control: with and without the integral action.

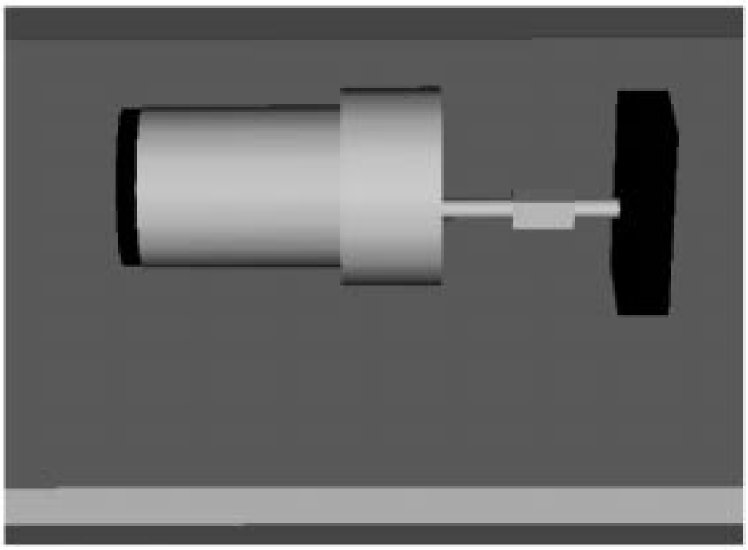

Fig. 32. The VRML 3D model of the DC motor, which can be driven by the real-time data acquired through the BS2/Simulink.

program from the EEPROM and executes the instructions; and (6) the PIC16C57.

The EEPROM memory is 2K bytes, which corresponds to about 500 lines of code. The speed of the clock is 20MHz, which corresponds to 4000 instructions/sec (on average, depending on the type of instruction). The BS2 has an additional 26 bytes of RAM and has 16 input/output (I/O) pins (PIN0 through PIN15), plus two additional pins (SOUT and SIN), which are used respectively for transmitting/receiving serial data during programming through the DEBUG/DEBUGIN instructions.

The digitally controlled potentiometers

The DS1803-010 [2] is an addressable, digitally controlled device which has two 256-position potentiometers ($10k\Omega$). Communication and control of the device is accomplished via a two-wire serial interface having signals SDA and SDL. Device addressing is attained using the device chip select inputs A0, A1, A2 and correct communication protocol over the two-wire serial interface. Each potentiometer is composed of a 256-position resistor array. Two eight-bit registers, each assigned to a respective potentiometer, are used to set the wiper position on the resistor array. The wiper terminal is multiplexed to one of 256 positions on the resistor array, based on its corresponding eight-bit register value. The DS1803-010 is a volatile device that does not maintain the position of the wiper during power-down or loss of power. Communication with the DS1803-010 takes place over the two-wire serial interface, consisting of the bi-directional data terminal, SDA, and the serial clock input, SCL. The pin description of the DS1803-010 is the following: L0, L1 – low end of resistors (connected to ground); H0, H1 – high end of resistors (connected to +5V); W0,W1—wiper terminals of resistors (connected to the input of the wide-band amplifier); VCC – +3V/+5V power supply input (connected to +5V); A0, A1, A2 – chip select inputs (left with no connection); SDA—serial data I/O (connected to the BS2's PIN10, with a 4.7k Ω pull-up resistor); SCL—serial clock input (connected to the BS2's PIN9, with a 4.7k Ω pull-up resistor); GND – ground; and NC – no connection. If V_0 and V_1 are the voltages of the wiper terminals with respect to ground, then the resolution of the voltage $V = V_0 - V_1$ is $5/256 \approx 0.02$ V. With an amplification of 12/5 (realized with the wide-band amplifier), the resolution of the voltage applied to the DC motor is ≈ 0.05 V.

In the following, one can find all the variables used for the communication BS2 ↔ DS1803-010, with a short description of their usage.

```
Pot VAR Bit ' Either potentiometer 0 or 1
Ibyte VAR Byte ' Byte read from the
      potentiometer Pot
Obyte VAR Byte ' Byte sent to the
      potentiometer Pot
Device VAR Byte ' Identification of the
      device 0-7
  ' using the I2C protocol: in our case
  ' the potentiometer
Setting VAR Byte ' Potentiometer setting
SCLPIN CON 9 ' Clock pin (to be connected
      with PIN9 of BS2)
SDAPIN CON 10 ' Data pin (to be connected
      with PIN10 of BS2)
SCLDIR VAR DIR9 ' PIN9 is used as output:
      clock
SDADIR VAR DIR10 ' PIN10 is used either as
      input or as output: data
SDAIN VAR IN10 ' Input on PIN10
SDAOUT VAR OUT10 ' Output on PIN10
SCLOUT VAR OUT9 ' Output on PIN9
IN CON 0 ' LOW
OUT CON 1 ' HIGH
n VAR Byte ' Index
B VAR Bit ' Bit
'------------------
SCLDIR=%1 ' PIN9 is declared as output
```

The following subroutine Writeeeprompot writes the byte specified in the variable Setting to the specified potentiometer, identified by the strings 10101001 when Pot=0 and 10101010 when Pot=1: Writeeeprompot uses the additional subroutines Start, Outbyte, Outone and Nack, whose descriptions follow.

```
Writeeeprompot:
 GOSUB Start
 Obyte=$50 (Device << 1)
 GOSUB Outbyte
 GOSUB Nack
 Obyte= $a9 + Pot
 GOSUB Outbyte
 GOSUB Nack
 Obyte=Setting
 GOSUB Outbyte
 GOSUB Nack
RETURN

Outbyte:
 LOW SDAPIN
 FOR n=0 TO 7
 B= (Obyte >> 7) & 1
 IF (B=1) THEN Outone
 SDADIR=OUT
 clk:
 HIGH SCLPIN
 LOW SCLPIN
 Obyte=Obyte << 1
 NEXT
 SDADIR=IN
RETURN

Outone:
 SDADIR=IN
 GOTO clk
 Getbyte:
 SDADIR=IN ' Input
 Ibyte=0
 FOR n=0 TO 7
 HIGH SCLPIN
 Ibyte=(Ibyte << 1) SDAIN
 LOW SCLPIN
 NEXT
 SDADIR=OUT ' Output
RETURN

Nack:
 SDADIR=IN
 HIGH SCLPIN
 LOW SCLPIN
 SDADIR=OUT
RETURN

Start:
 HIGH SDAPIN
 HIGH SCLPIN
 LOW SDAPIN
 LOW SCLPIN
RETURN
```

The following two variables are used for transmitting to the digitally controlled potentiometers the voltage to be applied to the motor.

```
Direction VAR Bit ' Clockwise (0), Counter
      clockwise (1)
ControlInput VAR Byte ' Absolute value of
      the voltage to be applied to the DC
      motor
  ' (0 to 255, which properly amplified
      correspond to 0V to +/-12V,
  ' depending on the value of Direction)
```

In particular, if V is the voltage that, after proper amplification, must be applied to the DC motor, then it is sufficient to set

$$\text{Direction} = \begin{cases} 0 & \text{if} \quad V < 0 \\ 1 & \text{if} \quad V \geq 0 \end{cases}$$

$$\text{Control\_Input} = abs(V)$$

$$V_0 = \begin{cases} 0 & \text{if} \quad \text{Direction} = 0 \\ \text{Control\_Input} & \text{if} \quad \text{Direction} = 1 \end{cases}$$

$$V_1 = \begin{cases} \text{Control\_Input} & \text{if} \quad \text{Direction} = 0 \\ 0 & \text{if} \quad \text{Direction} = 1 \end{cases}$$

This can be implemented with the following two subroutines:

```
Potentiometer0:
IF (Direction=0) THEN
 Setting=0
```

Table 2.

MR	$\overline{\text{PL}}$	CPU	CPD	MODE
High	Immaterial	Immaterial	Immaterial	Reset (asynchronous)
Low	Low	Immaterial	Immaterial	Parallel load
Low	High	Low → High	High	Count-up
Low	High	High	Low → High	Count-down

```
  ELSE
 Setting=ControlInput
ENDIF
 Pot=0
 GOSUB Writeeeprompot
RETURN

Potentiometer1:
IF (Direction=1) THEN
 Setting=0
  ELSE
 Setting=ControlInput
ENDIF
 Pot=1
 GOSUB Writeeeprompot
RETURN
```

The rotary optical encoder and the synchronous up/down binary counter

The `ENT1J-B28-L00128` [3] is a self-contained rotary optical encoder which produces a two-bit quadrature signal. It produces 128 pulses per shaft revolution, with a resolution of $360/128 \approx 2.8$ degrees. The output is two-bit grey coded: Channel A leads Channel B by 90 (electrical) degrees with clockwise rotation.

The `HEF40193B` [4] is a four-bit synchronous up/down binary counter. The counter has a count-up clock input (`CPU`), a count-down clock input (`CPD`), an asynchronous parallel load input (`PL`), four parallel data inputs (`P0` to `P3`), an asynchronous master reset input (`MR`), four counter outputs (`O0` to `O3`), an active LOW terminal count-up (carry) output (`TCU`) and an active LOW terminal count-down (borrow) output (`TCD`). The pin description of the `HEF40193B` is the following: $\overline{\text{PL}}$ parallel load input (connected to +5V), `P0` to `P3` parallel data inputs (left with no connection), `CPU` count-up clock pulse input (`CPU1` connected to the encoder Channel A, `CPU2` connected to $\overline{\text{TCU1}}$), `CPD` count-down clock pulse input (`CPD1` connected to +5V, `CPD2` connected to $\overline{\text{TPD1}}$), `MR` master reset input (connected to the `BS2`'s `PIN14`), $\overline{\text{TCU}}$ buffered terminal count-up (carry) output ($\overline{\text{TCU1}}$ connected to `CPU2`, $\overline{\text{TCU2}}$ with no connection), $\overline{\text{TCD}}$ buffered terminal count-down (borrow) output ($\overline{\text{TCD1}}$ connected to `CPD2`, $\overline{\text{TCD2}}$ with no connection), `O0` to `O3` buffered counter outputs (`O0` to `O3` of the first counter to the `BS2`'s pins `PIN0` to `PIN3`, `O0` to `O3` of the second counter to the `BS2`'s pins `PIN4` to `PIN7`, so that the `BS2`'s input byte `INL` contains at each time the state of the counter). The function table of the `HEF40193B` is reported in Table 2.

The following describes all the variables and pins used for reading the state of the counter, which is driven by the rotary encoder:

```
MVelocity VAR INL ' Measured velocity: a
     byte (8 bit) is used
MVelocityOld VAR Byte ' Old measured
     velocity,
   ' to be used for possible filtering
MPosition VAR Word ' Measured position
MPositionOld VAR Word ' Old measured
     position,
   ' to be used for possible filtering
Direction VAR Bit ' Clockwise (0), Counter
     clockwise (1)
RES PIN 14 ' Pin used for resetting the
     counter
'--------------------------
DIRL=%00000000 ' PIN0-PIN7 are used to
     read the counter
```

With the following procedure, the `BS2` reads the state of the counter and resets it to zero after reading:

```
Measure:
 MVelocityOld=MVelocity
 IF Direction=1 THEN
  IF MPosition>MVelocityOld THEN
    MPositionOld=MPosition-MVelocityOld
  ELSE
    MPositionOld=0
  ENDIF
 ELSE
    MPositionOld=MPosition+MVelocityOld
 ENDIF
 HIGH RES
 LOW RES
 MPosition=MPositionOld ' Possible
     filtering
RETURN
```

If the motor rotates counter-clockwise, `MPosition` is decreased, whereas, if the motor rotates clockwise, `MPosition` is increased.

With proper amendments, the measured position can be filtered taking into account the previously measured value; e.g. with the following instruction:

```
MPosition = MPosition */ $0040 +
     MPositionOld */ 00C0
```

If the maximum velocity of the encoder shaft is 200 RPM, then the maximum frequency of the impulses to be counted is $426 (200 * 128/60 = 426)$ impulses per second; hence, the eight-bit counter has no overflow up to a maximum sampling time of about 0.5s $(426 * 0.5 = 213 < 255)$.

Reading of reference position

`RCTIME` is used to measure the discharge time of a resistor/capacitor circuit. This allows one to measure the resistance of the 10k Ω potentiometer (the 220 Ω resistor is only used for pin protection when the potentiometer shaft is near the zero position), proportional to the position of its shaft, which is used as an analogue reference value (either for position or velocity) for the motor shaft. First, we bring pin `Potentiometer` (`PIN13`) in state 1; then, when `RCTIME Potentiometer,1, RPosition` is executed, it starts a counter (with a cycle of two μ s). It stops this counter as soon as the pin `Potentiometer` is no longer in state 1 (i.e. as soon as the pin voltage becomes less than +1.5V): `RCTIME` returns in `RPosition` the used number of cycles.

Denoting by $\tau = RC$ the time constant of the RC-circuit constituted by the potentiometer and the 0.1 μF capacitor (see the schematic circuit in Fig. 1), the time t needed for decreasing the voltage from +5V to +1.5V is about $t = 1.2\tau$. Hence, the value returned by `RCTIME` is

$$\text{R\_Position} \approx \frac{1.2\tau}{2 \cdot 10^{-6}}$$

If the maximum $R = 10^4\Omega$ and $C = 10^{-7}$ is F, then the maximum is R_Position ≈ 600. By a scaling of 28/12, the maximum value of the reference input obtained with the potentiometer is 1400 ($600 * 28/12 = 1400$), with a resolution of $1400/600 \approx 2.3$.

The following subroutine reads the value of the manually controlled potentiometer, which is used as a reference value:

```
POTENTIOMETER CON 13 ' Potentiometer
      input pin
RPosition VAR Word  ' Reference position
'------------------
Reading:
 HIGH Potentiometer  ' Pin 13 is High
 PAUSE 1             ' 1 ms pause
 RCTIME Potentiometer,1,RPosition
                   ' Discharge time measurement
 RPosition=(RPosition*25)/12 ' Scaling
RETURN
```

Alternatively, the value of `Rposition` can be received in real time (through a serial communication) by a Simulink program, which transmits the data to the `BS2` with the format "`A%d;`"; in such a case, the subroutine to be used is the following:

```
RxD       CON 15   ' PIN15
Baud96    CON 84   ' 9600-8-N-1
RPosition VAR Word ' Reference position
'------------------
Reading:
 SERIN RxD,Baud96,[WAIT(''A''),SDEC
      RPosition]
RETURN
```

The DC motor and the wide-band amplifier

The DC motor `IG33` is fitted with an all metal gear-head with sintered iron gears running on hardened steel shafts. The output shaft runs in sintered bronze bearings. This permanent magnet motor is wound for ± 12V (DC) operation. The main characteristics of the DC motor are shown in Table 3.

Table 3.

RPM at 12V	200
Max. torque (mNm)	100
Gear-box reduction	20:1
Nominal voltage (V)	12
Nominal no load current (mA)	50
Length (mm)	57.8

Table 4.

Baud rate	9600
Number of data bits	8
Number of stop bits	1
Parity	none

The `WA301` [5] is a 30V pk-pk waveform amplifier (±15V), with a bandwidth from DC to 1MHz; it has a high impedance input, a 0dB to +20dB gain that can be manually set, and two outputs (50 Ω and 600 Ω): the 50 Ω output is used in the present experimental set-up. It has an additional 20dB output attenuator, which, however, is not used in the present experimental set-up.

The RS232 communication

The communication PC $\leftrightarrow$ `BS2` is realized using both serial ports, COM1 (Tx connected to `SOUT` of `BS2`) and `COM2` (Rx connected to `PIN15` of `BS2`), of the PC. For both communications, we have always used the settings shown in Table 4.

CONCLUSIONS

In this paper we have described a simple and inexpensive experimental apparatus that has been developed at Università di Roma Tor Vergata for educational purposes; this tool is presently used in the course "Laboratorio d'Automatica" (Laboratory of Automatic Control), which is offered in the first year of a three-year university programme ("Laurea" degree) in Automatic Control; the students, who are only familiar with the fundamentals of calculus, geometry, physics and computer sciences, thereby have the opportunity to perform several experiments, thus approaching the basics of control theory from a practical point of view first.

Acknowledgements—This work was supported by the ASI (grants I/R/125/02 and I/R/277/02) and MIUR (grants COFIN:-Matrics and FIRB:Tiger). Tel. +39-06-7259 7431. Fax +39-06-7259 7460.

REFERENCES

1. *BASIC Stamp Programming Manual*. Parallax, 2.0c ed., 2003.
2. *DATASHEET: Addressable Dual Digital Potentiometer DS1803*. Dallas Semiconductors.
3. *DATASHEET: Rotary Optical Encoder ENT1J-B28-L00128*. Bourns.
4. *DATASHEET: 4-bit up/down binary counter HEF40193B*. Philips Semiconductors.
5. *DATASHEET: Wideband 30 V pk-pk amplifier for waveform generation WA301*. Thurlby Thandar Instruments.
6. E. Kamen, *Industrial Controls and Manufacturing*, Academic Press, San Diego, CA (1999).
7. W. S. Levine, *Control System Fundamentals*, CRC Press, NY (2002).
8. Y. Dote, *Servo Motor and Motion Control using Digital Signal Processors*, Prentice-Hall, Englewood Cliffs, NJ (1990).
9. A. Tornambè, High-gain observers for non-linear systems, *International Journal of Systems Science*, **23**(9) (1992), pp. 1475–1489.

Antonio Tornambè has been a Professor in the Industrial Robotics, Control Theory and Automatic Control Laboratory at Università di Roma Tor Vergata since 2001 and before that he held various research and teaching positions at Fondazione Ugo Bordoni, Università di Roma Tor Vergata, Politecnico di Torino, Università di Roma Tre, and Università di Siena. His research interests are in the area of control theory and in its application for controlling robotic manipulators and mechanical systems; his research results have been reported in about 200 papers published in international journals and presented at international conferences. He is the author of *Discrete-Event System Theory: An Introduction*, World Scientific Publishing, Singapore, 1995, and co-author of *Mathematical Methods for System Theory*, World Scientific Publishing, Singapore, 1998. He is the co-editor of *Modelling and Control of Mechanisms and Robots*, World Scientific Publishing, Singapore, 1996, *Modelling and Control of Mechanical Systems*, Imperial College Press, London, 1997, *Theory and Practice of Control and Systems*, World Scientific Publishing, Singapore, 1998, and *Modern Control Theory*, Esculapio, Roma, 1999.