UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
INSTITUTO DE FÍSICA
CURSO DE LICENCIATURA

PROJETO DE INSTRUMENTAÇÃO EM ENSINO DE FÍSICA

UTILIZAÇÃO DA PORTA DE JOGOS PARA AQUISIÇÃO DE DADOS

FRANCISCO ANTONIO LOPES LAUDARES

Orientador
CARLOS EDUARDO AGUIAR

Outubro de 2000

À Marcia e Victoria

...ser professor de Ciências implica fazer o
possível para comunicar e argumentar
com base em situações práticas...
Marco Antônio Moreira

Indice

1. Introdução

O computador é um extraordinário instrumento de laboratório. Ele é excelente em experiências que envolvem medidas de tempo, aquisição de grandes quantidades de dados, e tratamento de dados em tempo real. Apesar de todas estas qualidades, o computador é pouco usados nos laboratórios didáticos. E quando isto ocorre, é quase sempre com o auxílio de "kits" pré-fabricados - pacotes de circuitos eletrônicos e programas produzidos por empresas especializadas, vendidos a preços relativamente altos, e que operam como verdadeiras caixas-pretas [1,2]. Em geral é difícil usar estes pacotes para realizar experimentos diferentes daqueles para os quais eles foram projetados, o que diminui muito a sua utilidade didática. Parte importante desta falta de flexibilidade tem origem nos programas de aquisição e tratamento de dados contidos nos kits, que raramente podem ser modificados ou mesmo compreendidos por professores e estudantes.

Neste trabalho discutimos como montar um sistema de aquisição de dados simples, de baixo custo, utilizável em laboratórios didáticos da escola média, e que dá a alunos e professores controle completo sobre o seu uso. O sistema está baseado na substituição do "joystick", em geral usado para controlar jogos no computador, por sensores que podem ser utilizados em uma grande variedade de experiências. A idéia de usar a interface de jogos para aquisição de dados não é nova. Entretanto, propostas anteriores usavam computadores que não existem mais, como o Apple II [3,4], ou programas de aquisição escritos em linguagens pouco usadas hoje, como Pascal e Basic [5] (estes deram origem ao Delphi e VisualBasic, muito populares mas pouco apropriados ao ensino médio). No que se segue descreveremos como montar um sistema de aquisição de dados a partir da porta de jogos de um PC com plataforma Windows, usando programas totalmente escritos em Logo. Esta é uma linguagem simples, desenvolvida para uso didático, e que é freqüentemente ensinada na escola fundamental e média. Com isto *todos* os aspectos do sistema que descrevemos podem ser compreendidos e modificados por estudantes e professores, proporcionando-lhes uma grande flexibilidade no planejamento e execução de experimentos. Uma proposta semelhante, também utilizando o Logo, pode ser encontrada na referência [6]. Duas versões do Logo foram utilizadas neste trabalho: o MSWLogo (em inglês) e o SuperLogo (em português, produzido pelo NIED-UNICAMP). Ambas são gratuitas e podem ser obtidas via internet [7,8].

O motivo principal para se usar a porta de jogos em um laboratório didático é a segurança e facilidade com que podemos conectar sensores à esta interface. Um joystick é essencialmente um conjunto de resistências variáveis cujos valores são lidos a cada momento pelo computador. Substituindo o joystick por um componente eletrônico cuja resistência dependa da grandeza física que queremos medir (temperatura ou intensidade luminosa, por exemplo) podemos monitorar esta grandeza, registrar a sua evolução temporal em intervalos muito pequenos, e manipular quantidades de dados que dificilmente seriam alcançáveis em um laboratório didático convencional.

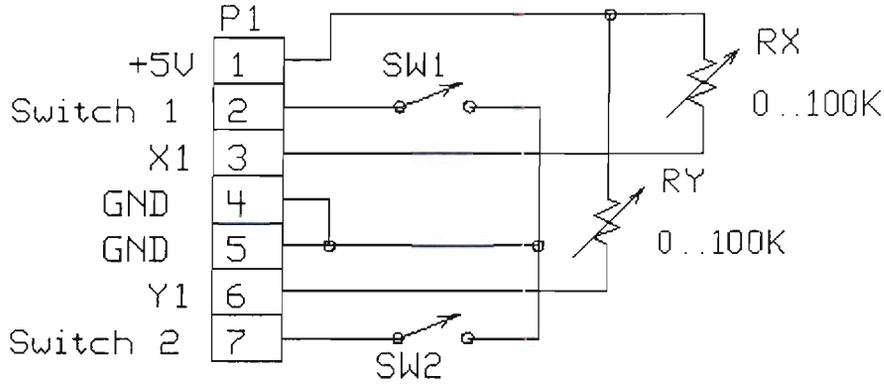
Esta monografia está organizada da seguinte forma. Na seção 2 descrevemos o funcionamento do joystick e da porta de jogos [9-10]. A medida de tempo com o computador é discutida na seção 3, e diferentes métodos de armazenagem dos dados são analisados na seção 4. Na seção

5 discutimos alguns sensores que podem ser conectados à porta de jogos, com ênfase em componentes optoeletrônicos. Nas seções 6 e 7 descrevemos dois experimentos que foram realizados com o sistema. No primeiro estudamos o movimento de um pêndulo, e no segundo a difusão de tinta na água. Alguns comentários e conclusões são apresentados na seção 8.

2. A porta de jogos

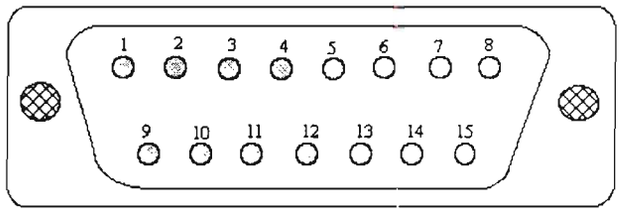
2.1. O joystick

O tipo mais comum de joystick para IBM-PC são os joysticks analógicos. Estes joysticks têm dois botões e dois potenciômetros cujas resistências vão de 0 a 100 kΩ (em alguns casos até 150 kΩ). O movimento esquerda-direita do joystick (eixo X) muda a resistência de um dos potenciômetros, e o movimento frente-trás (eixo Y) muda a resistência do outro. Em geral as resistências são nulas quando o joystick está todo para a esquerda e para a frente. Um esquema do joystick está mostrado abaixo. A tensão de 5 V e o aterramento não são fornecidos pelo joystick, mas pela porta de jogos à qual ele é conectado.



2.2. A porta de jogos

A porta de jogos do IBM-PC é uma interface para dois joysticks analógicos, embora exista a conexão para apenas um deles. O uso simultâneo de dois joysticks só é possível com um cabo especial em forma de 'Y'. O joystick é conectado à parte traseira do computador via um soquete de 15 pinos (DB 15), como mostra a figura abaixo.



A porta de jogos não faz parte da "placa mãe". Em geral ela é implementada em uma placa própria ou, mais freqüentemente, é colocada na placa de som. Neste último caso o conector tem dois dos seus 15 pinos dedicados à porta MIDI (musical instruments digital interface). Os outros pinos dão acesso aos quatro botões e quatro potenciômetros (dois eixos X e dois Y)

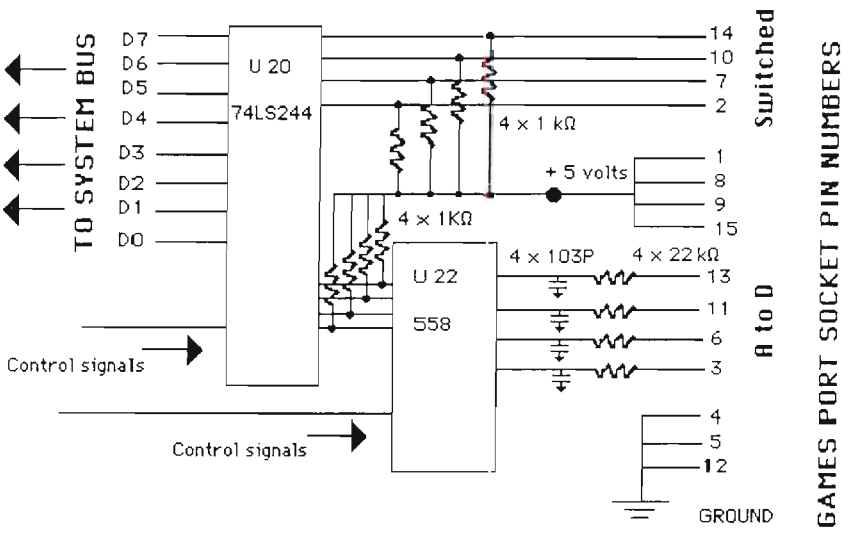
correspondentes aos dois joysticks, a um potencial de +5 Volts, e a um terra. Esta porta é muito segura, pois todos os pinos estão ligados a um circuito integrado que garante qualidade nos sinais e um grau de proteção considerável contra acidentes. Mas mesmo com esta proteção devemos tomar cuidado com curto-circuitos entre o potencial de 5 V e o terra.

A identificação e descrição dos pinos de uma interface de jogos encontram-se na tabela abaixo. Cada um dos joysticks (que chamamos 1 e 2) tem dois botões (A e B) e dois potenciômetros (X e Y). Assim A1 é o botão A do joystick 1, X2 é o potenciômetro X do joystick 2, etc. Muitos joysticks dão acesso aos 4 botões. Nestes casos A1 e B1 referem-se aos botões principais, e A2 e B2 aos secundários.

PINO	FUNÇÃO
1	+5 Volts
2	Botão A1
3	Potenciômetro X1
4	Terra
5	Terra
6	Potenciômetro Y1
7	Botão B1
8	+5 Volts (ou sem uso)
9	+5 Volts
10	Botão A2
11	Potenciômetro X2
12	Terra (ou porta midi)
13	Potenciômetro Y2
14	Botão B2
15	+5 Volts (ou porta midi)

Tabela 1

O esquema abaixo dá mais detalhes sobre os circuitos ligados a cada pino da porta.



2.3. Como a porta de jogos trabalha

A porta de jogos fornece 8 bits (1 byte) para leitura de dados no endereço 513 (decimal). O endereço 512 guarda um byte idêntico. Neste byte, 4 bits informam o estado dos 4 botões, e 4 bits são usados na medida da resistência dos 4 potenciômetros. A tabela 2 mostra como o estado dos joysticks é mapeado no byte fornecido pela porta.

BIT	7	6	5	4	3	2	1	0
ESTADO	B2	A2	B1	A1	Y2	X2	Y1	X1

Tabela 2

Os bits de 0 a 3 informam o estado do circuito que é usado para medir a resistência dos potenciômetros dos joysticks, e os bits de 4 a 7 informam o estado dos botões. Detalhes sobre o significado de cada bit estão dados na tabela 3.

BIT	**SIGNIFICADO**
7	Botão B2 (pino 14), 0=fechado, 1=aberto (default)
6	Botão A2 (pino 10), 0=fechado, 1=aberto (default)
5	Botão B1 (pino 7), 0=fechado, 1=aberto (default)
4	Botão A1 (pino 2), 0=fechado, 1=aberto (default)
3	Potenciômetro Y2 (pino 13), 1=cronometrando, 0=inerte
2	Potenciômetro X2 (pino 11), 1=cronometrando, 0=inerte
1	Potenciômetro Y1 (pino 6), 1=cronometrando, 0=inerte
0	Potenciômetro X1 (pino 3), 1=cronometrando, 0=inerte

Tabela 3

Note que o estado dos botões é dado de forma bastante direta (0 = fechado, 1 = aberto). Por outro lado, a medida da resistência de um potenciômetro é mais complicada, e é feita da seguinte maneira. Os bits associados aos potenciômetros (bits 0...3) têm valor zero até que se escreva algo no endereço da porta (513). Isto dispara um circuito que coloca os bits no estado 1. Cada um destes 4 bits permanece aí por um tempo que depende da resistência do respectivo potenciômetro segundo a relação

$$\text{Tempo } (\mu s) = 24{,}2 + 11 \times \text{Resistência } (k\Omega)$$

Após este tempo no estado 1 o bit volta ao valor 0. Medindo o tempo que o bit permanece no estado 1 obtemos a resistência do potenciômetro. A medida de tempo pode ser feita com um simples programa de contagem, que deve no entanto ser muito rápido [5,6]. Tal programa não pode, portanto, ser escrito em uma linguagem interpretada como Logo. Por isto o Logo tem um comando especial para determinar este tempo, que será discutido mais à frente. Observe que com R = 100 kΩ gastamos aproximadamente 1 ms em uma medida de resistência. Note também que se não houver nada conectado a uma determinada linha (R = ∞) o bit correspondente pode ficar indefinidamente com valor 1.

2.4. Lendo a porta de jogos

Com a linguagem Logo podemos ler facilmente a porta de jogos. No MSWLogo utilizamos o comando **inportb N**, e no SLogo **portaentradab N**. Nos dois casos **N** é o endereço da porta que se quer acessar, que é 513 (ou 512). Se tivermos um joystick conectado à porta e executarmos a instrução

show inportb 513 (ou **mostre portaentradab 513**)

teremos como resultado a *representação decimal* do byte lido (240, por exemplo). Para obter o estado de cada uma das 8 linhas da porta (o valor de cada bit) devemos converter este número para a *representação binária*. Podemos fazer isto usando a operação lógica "E", definida por : 0 E 0 = 0 , 0 E 1 = 0 , 1 E 0 = 0 , 1 E 1 = 1 . O operador **bitand** do MSWLogo (**bite** no SLogo) toma dois inteiros como argumento e aplica a operação "E" a cada par de bits de mesma significância (mesma "posição" no byte). Por exemplo, aplicando **bitand** (ou **bite**) aos números binários 1100 e 1010 obtemos 1000, como mostrado abaixo.

```
   1 1 0 0
E
   1 0 1 0
   -------
   1 0 0 0
```

No Logo, que só usa números decimais, a operação acima é realizado com a instrução

show bitand 12 10 (ou **mostre bite 12 10**)

já que $12_{dec} = 1100_{bin}$ e $10_{dec} = 1010_{bin}$. A resposta que o Logo apresenta é **8**, pois $8_{dec} = 1000_{bin}$.

Então, se quisermos saber qual é o n-ésimo bit (n = 0, 1, 2...) de um número X, basta fazer o **bitand** (ou **bite**) de 2^n com X. Note que todos os bits de 2^n valem 0, com exceção do n-ésimo bit que vale 1. Portanto, se o n-ésimo bit de X for 0 o resultado da operação será 0. Se o n-ésimo bit for 1 o resultado será 2^n. Por exemplo, para saber se o botão A1 (que corresponde ao bit 4) está apertado ou não tomamos o **bitand** de 2^4=16 com a leitura da porta:

```
show bitand 16 (inportb 513)
```
ou
```
mostre bite 16 (portaentradab 513)
```

Se o botão está apertado o resultado será 0, e se ele está livre o resultado será 16. Para saber o valor de cada um dos 8 bits da porta, basta aplicar o procedimento acima para n = 0, 1, 2, 3 ... 7, o que corresponde a usar 1, 2, 4, 8 ... 128 no **bitand** (ou **bite**). O programa **lerpjogos** (em MSWLogo) listado abaixo faz isto e escreve os bits resultantes. A versão em SLogo deste programa pode ser encontrada no Apêndice.

```
to lerpjogos
make "linhas []
make "x (inportb 513)
make "m 1
repeat 8 ~
[
  make "b (bitand :x :m)
  if :b=:m [make "b 1]
  make "linhas fput :b :linhas
  make "m 2*:m
]
show :linhas
end
```

Basta entrar com **lerpjogos** na janela de comandos para que os bits correspondentes a cada linha da porta de jogos sejam mostrados.

Uma forma mais compacta de ler o estado de cada linha da porta de jogos utiliza a função **map** do MSWLogo, ou **mapeie** do SLogo. Para obter um resultado igual ao de **lerpjogos** basta executar a instrução

```
show map [(bitand ? inportb 513)/?] [128 64 32 16 8 4 2 1]
```

no MSWLogo, ou

```
mostre mapeie [(bite ? portaentradab 513)/?] [128 64 32 16 8 4 2 1]
```

no SLogo.

Sabendo como ler o estado dos botões do joystick, podemos usá-los para controlar um programa Logo. Por exemplo, o programa **botao** listado abaixo permite deslocar a tartaruga apertando os botões do joystick. Com um joystick de 4 botões, fazemos a tartaruga se movimentar para frente com o botão A1, girar para a esquerda com o botão A2 e para a

direita com o botão B2 . O botão B1 termina a execução do programa. Escrito em MSWLogo o programa é (a versão em SLogo está no Apêndice):

```
to botao
make "fim 0
do.while [passo] [:fim=0]
end

to passo
make "x (inportb 512)
make "a1 (bitand :x 16)
make "b1 (bitand :x 32)
make "a2 (bitand :x 64)
make "b2 (bitand :x 128)
if :a1=0 [forward 1]
if :b1=0 [make "fim 1]
if :a2=0 [left 1]
if :b2=0 [right 1]
end
```

A figura 1 abaixo mostra como o programa **botao** pode controlar a tartaruga.

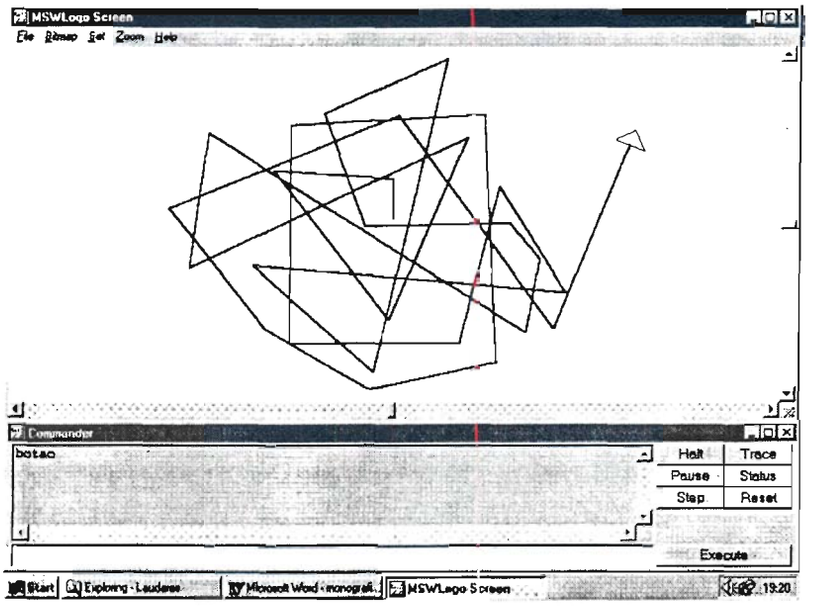


Figura 1

2.5. Medindo resistências com a porta de jogos

Como já comentamos, a resistência de um potenciômetro é obtida medindo-se o tempo que o bit correspondente permanece com valor 1, a partir do instante em que um byte qualquer é escrito no endereço da porta. Este tempo é tipicamente menor que alguns milisegundos. Fazer um programa em Logo para medi-lo não é eficiente, pois tal programa seria muito lento. Para realizar esta medida de tempo o Logo tem um comando especial: o **ingameport M** (no

MSWLogo), ou **portajoystick M** (no SLogo). Este comando fornece o tempo que o bit da porta de jogos, especificado pelo número **M**, permanece no estado 1 (M = 1, 2, 4, 8 corresponde ao bit N = 0, 1, 2, 3, ou seja, $M = 2^N$). O tempo fornecido por **ingameport** ou **portajoystick** não está em unidades predeterminadas. Mas, como já vimos, este tempo varia linearmente com a resistência, o que é suficiente para muitas aplicações. De qualquer forma, usando duas resistências conhecidas é possível calibrar a saída de **ingameport** ou **portajoystick**. Com estes comandos podemos utilizar o "manche" do joystick para controlar programas Logo. E o que é mais importante, podemos monitorar a resistência de sensores externos como fotoresistores (resistores dependentes de luz) e termistores (resistores dependentes da temperatura).

Portanto, para medir a resistência X1 só precisamos executar a instrução

show ingameport 1 (ou **portajoystick 1**),

e para medir Y1,

show ingameport 2 (ou **portajoystick 2**).

Devemos ter algum cuidado ao tentar medir os potenciômetros X2 e Y2. Como eles quase nunca estão ligados à porta (pois o segundo joystick raramente é usado) a resistência que é encontrada é infinita, e o tempo a ser medido também. Neste caso o comando **ingameport** (**portajoystick**) fornece o valor –1, mas pode levar muito tempo (até vários minutos, dependendo do computador) para fazer isto. O programa abaixo mostra como podemos usar o manche do joystick para controlar a posição da tartaruga. Em MSWLogo (veja o Apêndice para a versão SLogo),

```
to manche
make "zerox 400    ;centro da tela
make "zeroy 400
make "escx 0.5     ;escalas
make "escy 0.5
do.while [leitura] [:b1=32]
end

to leitura
make "x1 (ingameport 1)
make "y1 (ingameport 2)
setxy :escx*(:x1-:zerox) :escy*(:zeroy-:y1)
make "b1 bitand 32 (inportb 513)
end
```

A figura 2 mostra um exemplo de como o programa pode controlar a tartaruga.

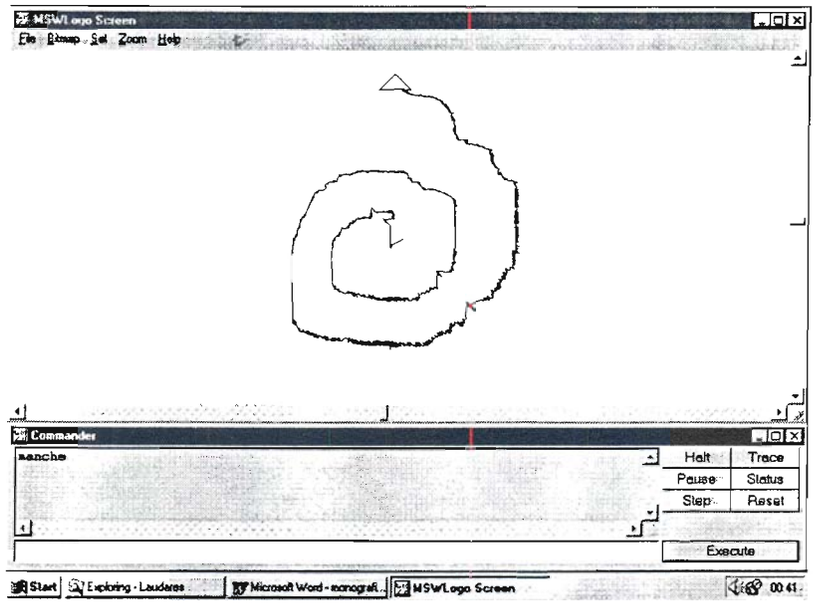


Figura 2

3. A medida de tempo

Agora que já sabemos ler a porta de jogos, o próximo passo é registrar a evolução temporal dos sinais que ela mede. Para isto precisamos também aprender a medir o instante em que o sinal foi lido. No MSWLogo isto é feito com o comando **timemilli**, que fornece o tempo em milisegundos decorrido desde que o sistema operacional (Windows) foi iniciado. No SLogo o comando equivalente chama-se **tempomili**. Executando a instrução

show timemilli (ou **mostre tempomili**)

obtemos a quantos milisegundos o Windows está operando. É importante ter uma estimativa da resolução de **timemilli** (**tempomili**), pois tempo que estes comandos fornecem não é atualizado a cada milisegundo, mas em intervalos cuja magnitude pode depender do computador e da configuração do sistema. O programa **resolucao** mostrado abaixo faz uma estimativa deste intervalo, a partir da diferença entre chamadas sucessivas de **timemilli** (**tempomili**). Em MSWLogo (veja o Apêndice para a versão SLogo),

```
to resolucao
make "n 0
make "soma 0
make "soma2 0
repeat 100000 ~
[
  make "t1 timemilli
  make "t2 timemilli
  make "dt :t2-:t1
  make "soma :soma+:dt
```

```
  make "soma2 :soma2+(:dt*:dt)
  if (:dt>0) [make "n :n+1]
]
make "percent :n/1000
make "resol :soma/:n
make "desvp sqrt (:soma2/:n - :resol*:resol)
(print [Percentagem com dt>0:] :percent "%)
(print [Resolução:] :resol "ms)
(print [Desvio padrão:] :desvp "ms)
end
```

Rodando em vários computadores sob Windows95 o programa acima resultou em resoluções da ordem de 13 ± 2 ms. Apenas ~0,5% da chamadas sucessivas a **timemilli** (**tempomili**) resultaram em valores diferentes. Isto significa que o programa realiza cerca de 200 vezes o ciclo de repetição durante o período em que o "relógio" de **timemilli** (**tempomili**) fica sem atualização. Assim, a resolução de **timemilli** ou **tempomili** sob o Windows95 é de aproximadamente um centésimo de segundo. Sob o Windows98 a resolução mostrou-se bem melhor, da ordem de 5 ± 1 ms.

4. Armazenagem dos dados

Os dados lidos na porta de jogos devem ser armazenados para posterior análise. O tempo gasto na armazenagem pode tornar lenta a aquisição dos dados, e devemos avaliar com cuidado a melhor forma de fazer isto. Com o Logo podemos guardar os dados de duas maneiras:

1. Armazenando-os em listas que ficam na memória.
2. Escrevendo-os diretamente no disco rígido.

A armazenagem em lista pode ser feita colocando-se cada novo dado no início ou final da lista. Há uma diferença significativa entre os dois métodos. O programa abaixo lê durante 10 segundos o estado do botão A1 e do relógio, e escreve cada resultado no início da lista de dados. Junto com cada tempo e sinal, também é escrito o número de leituras já realizadas até este ponto. Após terminada a aquisição dos dados, a lista é gravada em um arquivo no disco rígido.

```
to aquisicaolista
make "tmax 10000
make "dados []
make "n 0
make "t 0
make "t0 timemilli
while [:t < :tmax] ~
[
  make "n :n+1
  make "t timemilli-:t0
  make "a (bitand 16 inportb 512)/16
  make "dados fput (list :n :t :a) :dados
```

```
]
show [Fim da aquisicao]
openwrite "aquisicao.dat
setwrite "aquisicao.dat
make "n count :dados
for [i 1 :n][print (item :i :dados)]
setwrite []
close "aquisicao.dat
show [Dados escritos em aquisicao.dat]
end
```

Executamos o programa acima em um Pentium 166 sob Windows95, com 32M de memória RAM. Apertando e soltando várias vezes o botão A1 do joystick, obtivemos o resultado mostrado na figura 3. Nela vemos como o número de pontos lidos e o estado do botão variam com o tempo. Podemos observar a mudança de estado do botão (a linha denteada que salta entre 0 e 1), mas o importante nesta figura é o crescimento praticamente linear do número de pontos com o tempo. Isto significa que o tamanho da lista não influenciou o tempo necessário para adicionar a ela um novo membro. Note que usamos o comando **fput** para colocar os dados, de modo que cada um deles entrou no início da lista. Se em vez de **fput** usarmos **lput**, ou seja se colocarmos cada novo dado no final da lista, o resultado fica muito diferente. Como está mostrado na figura 4, neste caso a taxa de aquisição diminui à medida que o tempo passa. Em outras palavras, quanto maior é a lista mais demorado torna-se adicionar um dado ao seu final. Note também as escalas diferentes nos dois gráficos. Enquanto no primeiro caso armazenamos cerca de 6000 dados em 10 segundos, no segundo obtivemos apenas uns 2000 pontos no mesmo período. Assim, se desejamos armazenar os dados em uma lista na memória, devemos fazê-lo colocando estes dados no início da lista e não no final.

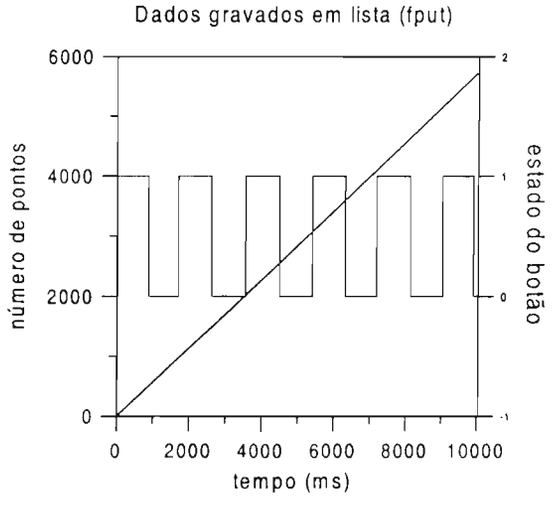


Figura 3

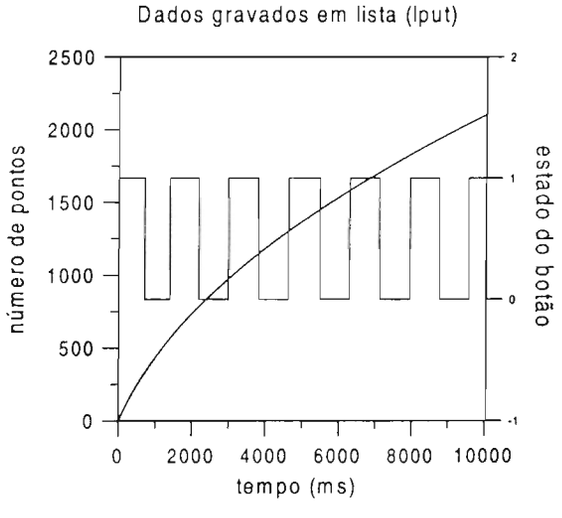


Figura 4

Os dados lidos na porta de jogos também podem ser gravados diretamente no disco rígido. O programa abaixo lê o botão A1 durante 10 segundos e grava no disco o resultado de cada leitura, o tempo correspondente, e o número de pontos já lidos.

```
to aquisicaodisco
make "tmax 10000
make "n 0
make "t 0
openwrite "aquisicao.dat
setwrite "aquisicao.dat
make "t0 timemilli
while [:t < :tmax] ~
[
  make "n :n+1
  make "t timemilli-:t0
  make "a (bitand 16 inportb 512)/16
  (print :n :t :a)
]
show [Fim da aquisicao]
setwrite []
close "aquisicao.dat
show [Dados escritos em aquisicao.dat]
end
```

O resultado da execução de programa no mesmo computador usado anteriormente está mostrado na figura 5.

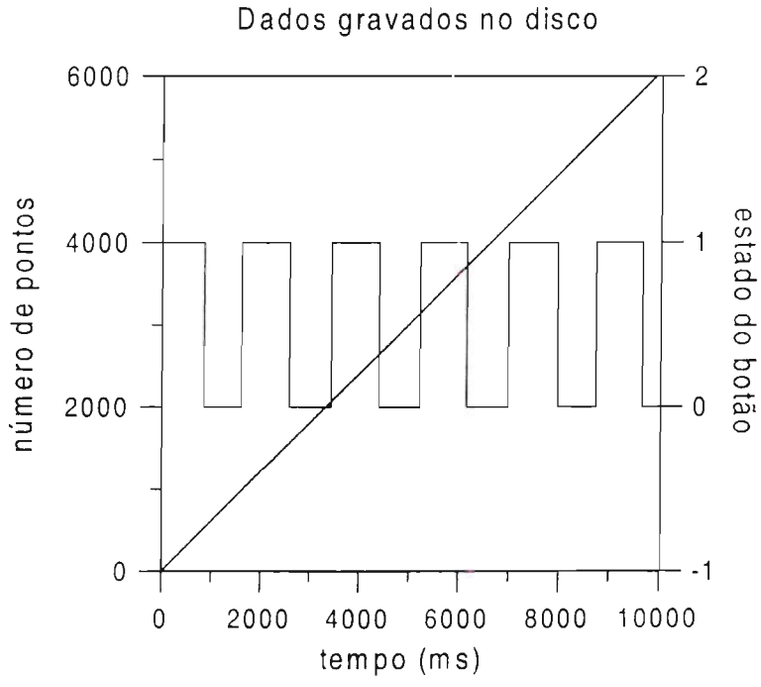


Figura 5

Vemos que taxa de aquisição mantém-se constante com o tempo, como no caso da lista alimentada com **`fput`**. O número de pontos lidos nos dois casos é praticamente o mesmo (~600/segundo), com ligeira vantagem para a armazenagem em disco.

É instrutivo observar detalhadamente como evolui a aquisição dos dados. Isto é feito na figura 6, onde a ampliação de uma pequena parte da figura 5 está mostrada. É possível observar nesta figura o efeito causado pela resolução temporal de ~13 ms que já discutimos. A linha que parecia ser uma reta na figura 5 revela-se uma "escada" quando vista em detalhe – várias medidas sucessivas aparecem com o mesmo registro de tempo. Cerca de 8 ou 9 medidas são feitas e armazenados antes que o relógio consultado pelo Logo atualize o seu "tempo", o que é feito apenas a cada 13 ms.

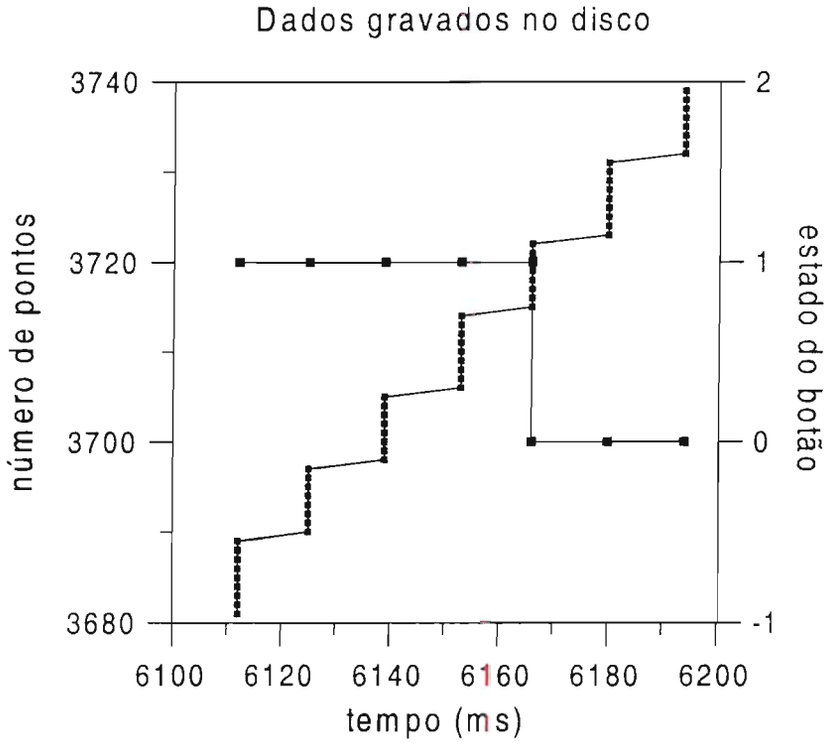


Figura 6

Outro fato importante que pode ser observado na figura 6 é que todos os "tempos" têm o aproximadamente o mesmo número de pontos medidos (8 ou 9), qualquer que seja o sinal do botão (0 ou 1). Em outras palavras, o tempo que a porta de jogos leva para descobrir o estado do botão é sempre igual, quer este esteja aberto ou fechado, e é cerca de 10 vezes menor que o tempo que o sistema leva para atualizar o relógio lido pelo Logo. Isto abre a possibilidade de usarmos o próprio número de medidas como contador de tempo, usando o relógio do sistema, consultado por **`timemilli`** (**`tempomili`**), apenas para calibrar os tempos inicial e final da medida. Desta forma podemos alcançar resoluções temporais da ordem de 1 milisegundo (no Windows95) ou menos (no Windows98).

5. Conectando sensores à porta de jogos

No lugar dos botões e potenciômetros do joystick, podemos conectar à porta de jogos componentes eletrônicos de resistência variável que possam ser usados como sensores. Nesta seção discutiremos rapidamente alguns desses componentes: o termistor NTC, o fotoresistor, o fotodiodo e o fototransistor.

5.1. Termistores

Um termistor é um componente eletrônico cuja resistência varia fortemente com a temperatura. Em geral os termistores são do tipo NTC (de *negative temperature coefficient*), cuja resistência diminui com o aumento da temperatura. Um termistor NTC é feito de material semicondutor, e é tipicamente utilizado na faixa de temperaturas entre –50 °C e 150 °C. A relação entre a resistência e a temperatura absoluta de um NTC (curva característica R/T) é altamente não-linear, e pode ser aproximada por

$$R = \alpha e^{\beta/T}.$$

Medindo a resistência para duas temperaturas diferentes podemos obter as constantes α e β e construir uma curva de calibração razoavelmente precisa. Uma parametrização melhor que a acima é dada pela relação de Steinhart-Hart:

$$1/T = A + B\,(\ln R) + C\,(\ln R)^3$$

onde T é a temperatura em kelvins e R a resistência em ohms. Sabendo a resistência em 3 temperaturas diferentes podemos calcular os coeficientes A, B e C, e obter uma curva de calibração muito precisa. A resistência de um NTC é geralmente especificada a 25 °C, e na maioria dos casos está entre 100 Ω e 100 kΩ. A figura abaixo mostra a variação da resistência com a temperatura para vários NTCs e para um resistor de platina.

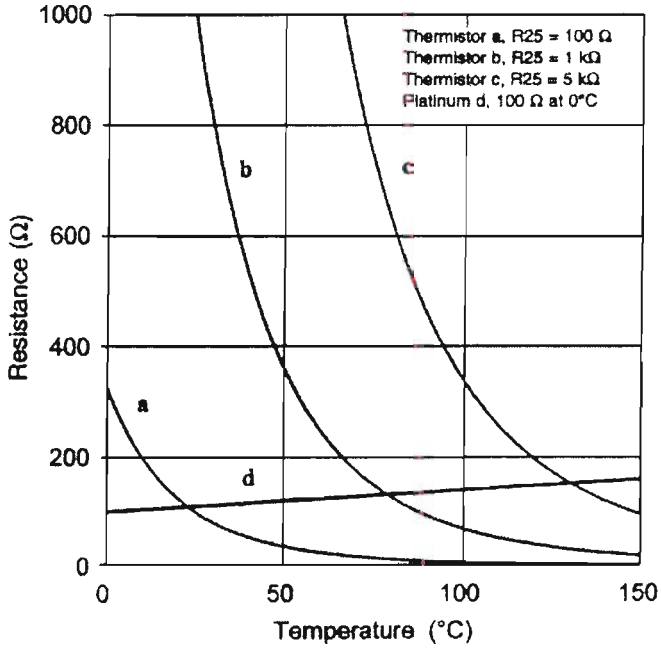


Figura 7

5.2. Fotoresistores

O fotoresistor, também chamado de célula fotocondutora, fotocélula, ou LDR (de *light dependent resistor*) possui uma resistência que depende da quantidade de luz que ele recebe. No escuro os fotoresistores têm resistência elevada, tipicamente entre 100 kΩ e 200 MΩ. Quando são expostos a luz a resistência pode diminuir por várias ordens de magnitude. Sua desvantagem como sensor está na lentidão de resposta, que chega a décimos de segundo - uma fotocélula não percebe o "pisca-pisca" de 60 Hz de uma lâmpada fluorescente. A figura abaixo mostra uma fotocélula de CdS, muito usada por ter uma resposta espectral semelhante à do olho humano.

Figura 8

5.3. Fotodiodos

O fotodiodo (figura 9) é um diodo semicondutor em que a junção está exposta à luz.

Figura 9

A corrente reversa de um fotodiodo aumenta com a incidência de luz, como pode ser visto nas curvas características I/V da figura 10. Observe que as curvas características passam pelo quarto quadrante (V×I<0), de modo que o fotodiodo pode produzir energia elétrica - as células solares são um tipo especial de fotodiodo.

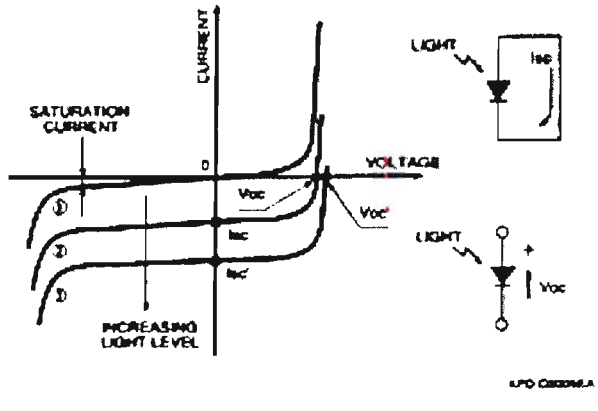


Figura 10

Os fotodiodos são úteis como sensores de radiação visível e infravermelha. A resposta espectral (sensibilidade a diferentes comprimentos de onda) de um fotodiodo de silício está mostrada na figura 11; podemos ver que o máximo encontra-se no infravermelho. Alguns fotodiodos tem uma cobertura que filtra a luz visível, deixando passar apenas a radiação infravermelha. A velocidade de resposta é muito alta nos fotodiodos, tipicamente entre 10^{-10} e 10^{-12} segundos.

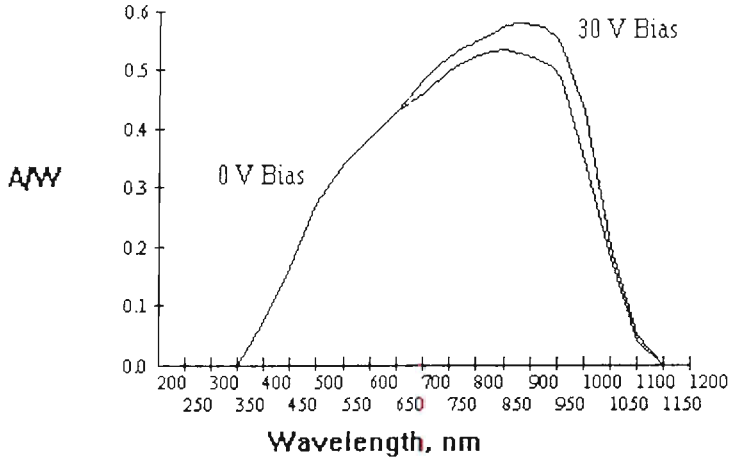


Figura 11

As correntes obtidas com um fotodiodo são baixas, da ordem de 10 μA. Sem amplificação não adianta acoplar um fotodiodo à porta de jogos. O circuito da figura abaixo mostra como a amplificação pode ser feita ligando o fotodiodo a um transistor. O conjunto resultante desempenha um papel semelhante ao de um LDR - em ambos os casos a tensão aplicada entre os pontos A e B resulta em uma corrente que aumenta drásticamente com a iluminação.

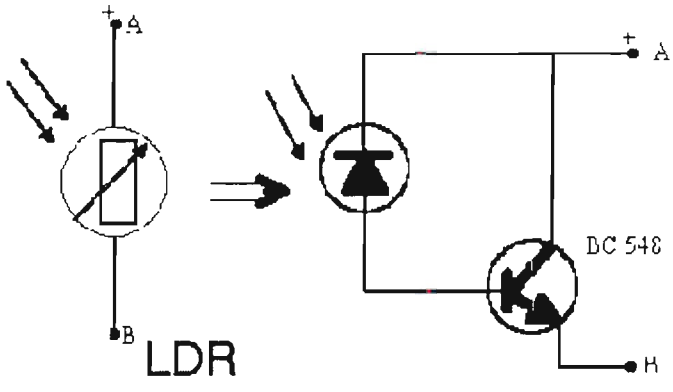


Figura 12

5.4. Fototransistores

Os fototransistores (figura 13) são transistores com a junção coletor-base exposta à luz.

Figura 13

Como podemos ver das curvas características abaixo, fototransistores são muito mais sensíveis que fotodiodos, gerando correntes da ordem de mA. Os fototransistores podem ser conectados diretamente à porta de jogos. O seu tempo de resposta é da ordem de 10^{-7} s (bem maior que o dos fotodiodos).

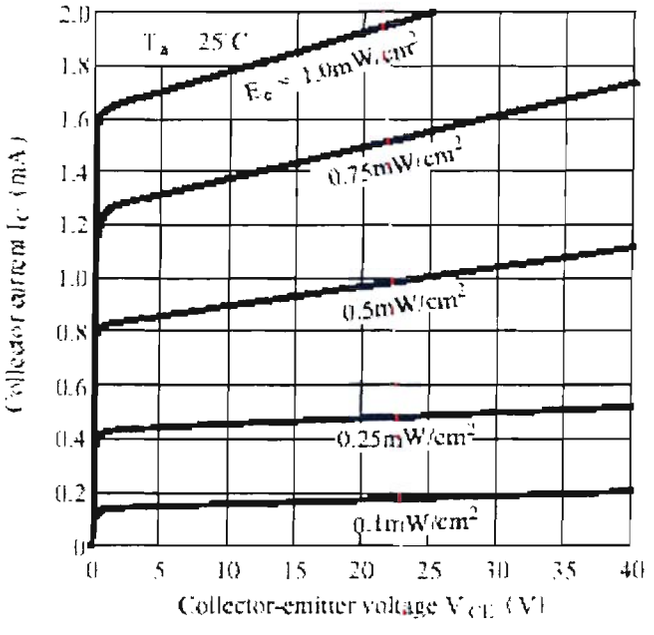


Figura 14

6. O pêndulo simples

6.1. A montagem do experimento

O objetivo da experiência é estudar o movimento de um pêndulo simples. O aparato usado no experimento está esquematizada na figura 15, e as dimensões do pêndulo utilizado estão dadas na figura 16.

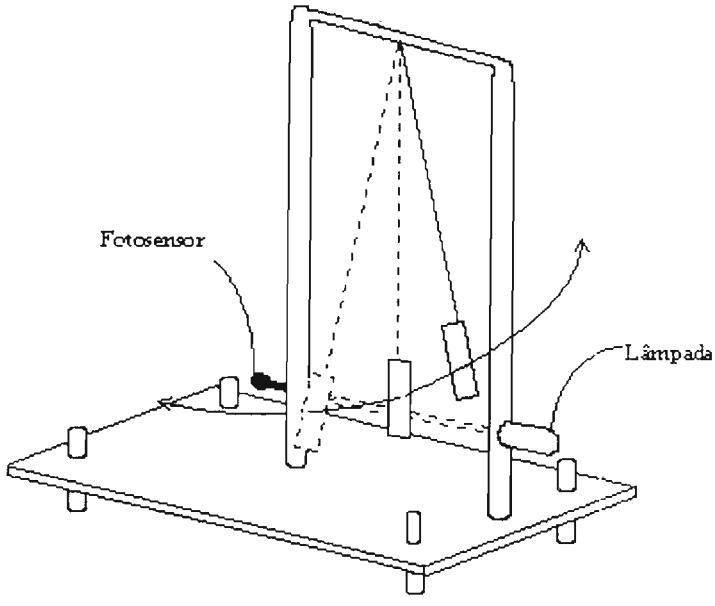


Figura 16

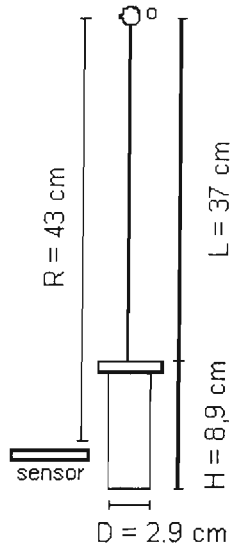


Figura 17

Nosso detetor é um conjunto fotodiodo-transistor, já discutido na seção anterior, iluminado por um LED infravermelho tirado de um controle remoto de TV. A figura 18 mostra o sistema usado. O transistor está ligado à porta de jogos via os pinos 2 e 4 (ou 5) do conector, que correspondem ao botão A1 do joystick e ao terra (veja a seção 2). Assim, quando o fotodiodo estiver no escuro, ou recebendo uma iluminação insuficiente, o transistor se comportará como uma chave aberta. Se o fotodiodo receber um sinal luminoso mais forte o transistor permitirá a passagem de uma corrente alta entre o pino 2 e o terra, realizando algo semelhante ao fechamento de uma chave (ou apertar do botão). Deste modo quando o pêndulo passa entre a lâmpada e o fotodiodo a leitura do bit 4 da porta dará "1", e quando a luz atinge o detetor o resultado será "0".

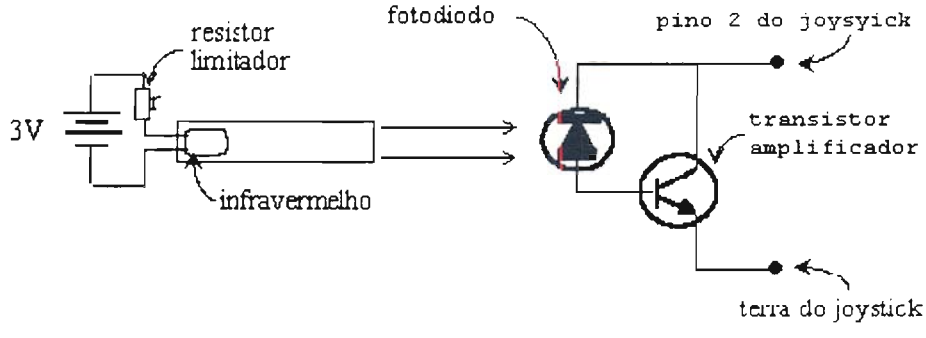


Figura 18

Ao oscilar, o pêndulo gera então um sinal como o mostrado na figura abaixo.

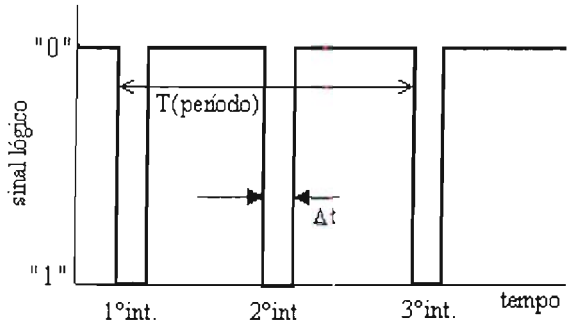


Figura 19

Registrando este sinal podemos obter o período da oscilação, e o tempo Δt durante o qual o pêndulo esteve em frente ao fotosensor. A velocidade do pêndulo neste ponto é dada por

$$V_0 = D / \Delta t$$

onde D é a largura mostrada na figura 11. Como o fotosensor está em frente à posição de equilíbrio do pêndulo, V_0 é a velocidade máxima da oscilação. A velocidade angular correspondente pode ser calculada como

$$\Omega_0 = \frac{V_0}{R} = \frac{D/R}{\Delta t}$$

onde R é a distância do fotosensor ao ponto de fixação do pêndulo (veja a figura 10). A amplitude da oscilação é dada por

$$\cos\Theta_{max} = 1 - (R_G / 2g)\,\Omega_0^{\,2}$$

onde R_G é o raio de giração do pêndulo e g é a aceleração da gravidade.

O pêndulo é composto por um tubo de comprimento H cheio de areia, amarrado a um fio de extensão L. Considerando que o fio não possui massa e o que tubo está homogeneamente ocupado, o raio de giração do sistema será

$$R_G = L\sqrt{1 + \frac{H}{L} + \frac{1}{3}\frac{H^2}{L^2}}$$

Com L = 37 cm e H = 8,9 cm, temos R_G = 41,45 cm. Como o tubo tem uma tampa muito leve de aproximadamente 2 cm, poderíamos desprezar a sua massa e aumentar L para 39 cm, diminuindo H para 6,9 cm. Neste caso encontraríamos R_G = 42,45 cm. Nós adotamos a média dos valores acima como uma boa estimativa do raio de giração, ou seja, tomamos R_G = 42,0 cm.

6.2. O programa de aquisição

O programa de aquisição que usamos, em MSWLogo, está listado abaixo. O programa principal, **aquisicao**, tem um parametro de entrada **taq** que determina o tempo de aquisição em milisegundos. Os dados são gravados no disco rígido, no arquivo **dados.txt**. Para acompanhar o amortecimento da oscilação temos que manter a aquisição por um período longo, o que pode tornar o arquivo de dados muito grande. Uma forma de reduzir a tamanho do arquivo de dados sem diminuir a informação que ele contem é escrever no disco apenas os tempos em que ocorrem os saltos entre "0" e "1" no sinal da porta de jogos (veja a figura 12). Com estes tempos todo o sinal pode ser reconstruído. Nós adotamos esta forma de armazenamento no programa abaixo.

```
to aquisicao :taq
make "lista1 []
make "t 0
make "Tz timemilli
openwrite "dados.txt
setwrite "dados.txt
while [:t < :taq] [Le_Grava]
setwrite []
close "dados.txt
show [RESULTADOS EM DADOS.TXT]
show [FIM]
end

to Le_Grava
make "t timemilli-:tz
test (bitand 16 (inportb 513) 16)=16 ~
iftrue [make "lista1 (fput :t :lista1)]
iffalse ~
[
  test :lista1=[]
  iffalse ~
  [
    (print (last :lista1) (first :lista1))
     make "lista1 []
  ]
]
end
```

Para obter as características de cada oscilação do pêndulo devemos processar os dados contidos no arquivo do programa de aquisição. Isto é feito pelo programa abaixo, **analise**, que calcula o período T e velocidade angular Ω_0 a cada instante t em que ocorre uma passagem do pêndulo pela posição de equilíbrio. Note que a variável **fator** usada para calcular Ω_0 no programa corresponde a D/R. O programa escreve uma tabela com os valores de t, T e Ω_0, no arquivo **tpo.dat**. Todos os tempos nesta tabela estão em segundos.

```
to analise
show [LENDO OS DADOS]
make "dados []
openread "dados.txt
setread "dados.txt
until [eofp] [make "dados lput readlist :dados]
setread []
close "dados.txt
show [ANALISANDO OS DADOS]
make "fator 2.9/43
make "n count :dados
openwrite "tpo.dat
setwrite "tpo.dat
for [i 2 (:n-1) 1] ~
[
  make "t1t2 (item :i-1 :dados)
  make "t1 (first :t1t2)
  make "t2 (last :t1t2)
  make "t3t4 (item :i+1 :dados)
  make "t3 (first :t3t4)
  make "t4 (last :t3t4)
  make "periodo (:t4+:t3-:t2-:t1)/2/1000
  make "tatb (item :i :dados)
  make "ta (first :tatb)
  make "tb (last :tatb)
  test :ta=:tb
  iffalse [make "omega :fator/(:tb-:ta)*1000]
  iftrue [make "omega -1]
  make "t (:ta+:tb)/2/1000
  (print :t :periodo :omega)
]
setwrite []
close "tpo.dat
show [RESULTADOS EM TPO.DAT]
show [FIM]
end
```

6.3. Resultados experimentais

As figuras abaixo mostram o resultado de uma medida em que a oscilação do pêndulo foi acompanhada durante 4 minutos.

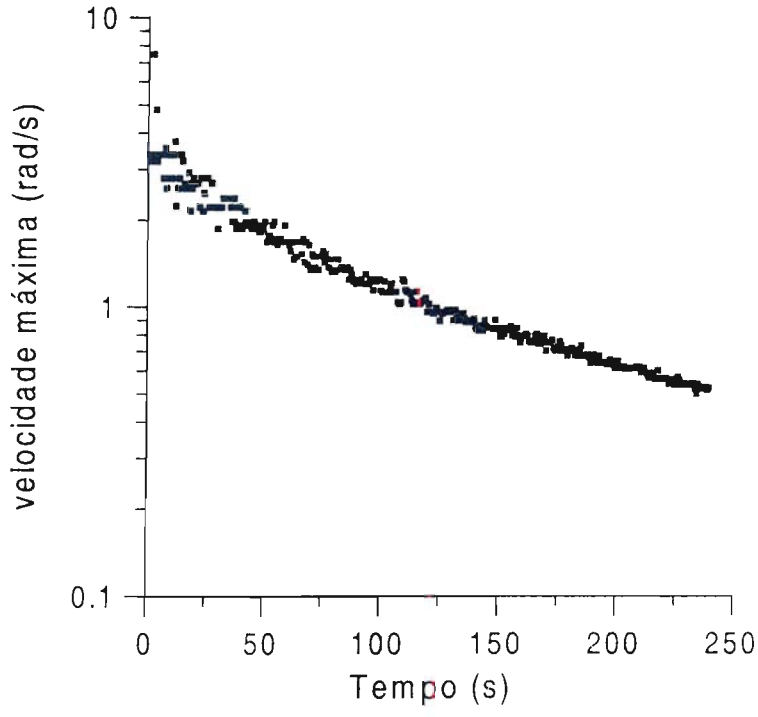


Figura 20

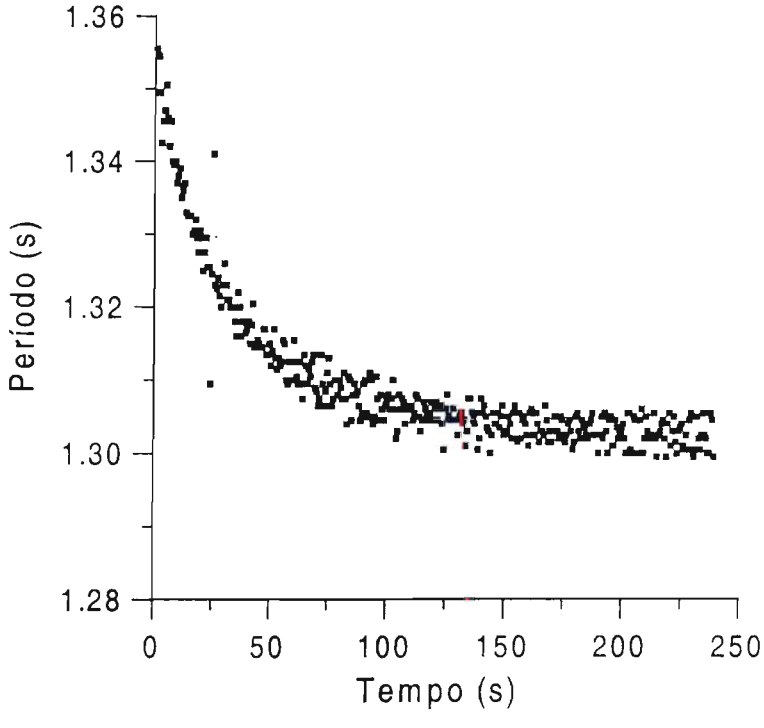


Figura 21

A figura 20 mostra a velocidade angular máxima como função do tempo. A redução da amplitude de oscilação causada pelas forças de atrito é claramente visível.

O período das oscilações como função do tempo está mostrado na figura 21. A diminuição do período com o tempo reflete a sua dependência na amplitude. Observe que a estabilização que ocorre após algum tempo corresponde ao "regime de pequenas oscilações".

A figura 22 mostra o período em função da velocidade angular máxima. Vemos claramente como o período aumenta com a velocidade (amplitude).

Para pequenas oscilações sabemos que o período deve ser dado por

$$T = 2\pi\sqrt{\frac{R_G}{g}}$$

Usando a estimativa de $R_G = 42$ cm para o raio de giração, e $g = 978{,}8$ cm/s^2 para a aceleração gravitacional no Rio de Janeiro[1], obtemos $T = 1{,}302$ s. A figura 22 mostra que este cálculo está em ótimo acordo com o resultado experimental a baixas velocidades.

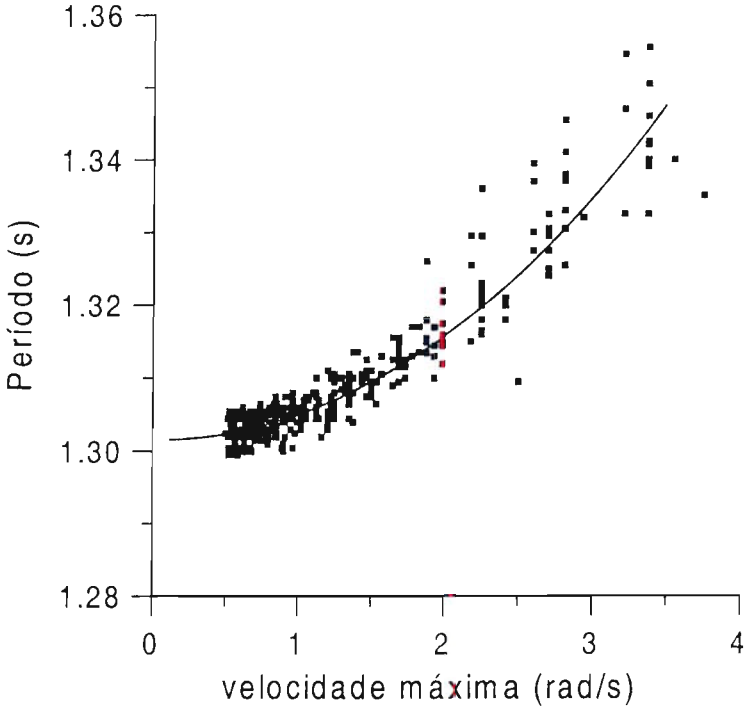


Figura 22

6.4. Período do pêndulo

A figura 22 é interessante por mostrar algo que dificilmente é medido em um laboratório didático convencional: a variação do período com a amplitude da oscilação. Comparar esta medida com a previsão da mecânica é instrutivo. O programa listado a seguir, **`pendulo`**, calcula o período de uma oscilação cuja velocidade angular máxima é dada por **`v0`**.

[1] Fonte: Observatório Nacional

```
to pendulo :v0
;x -> angulo em rad
;v -> velocidade angular
;unidades = cm,s
make "h 0.0001          ;salto no tempo
make "g 978.8           ;aceleracao da gravidade
make "rg 42.0           ;raio de giracao
make "gama 0            ;coeficiente de atrito
make "w0 sqrt :g/:rg
make "t0 2*pi/:w0
make "w02 :w0*:w0
if (:v0=0) [make "v0 0.0001/:w0]
make "t 0
make "x 0
make "v :v0
do.until [passo] [:x<0]
make "periodo 2*(:t-:x/:v)
print (list :v0 :periodo)
end

to passo
make "a -:w02*(radsin :x)-2*:gama*:v
make "v  :v + :a*:h
make "x  :x + :v*:h
make "t  :t + :h
end
```

O programa abaixo executa o procedimento **pendulo** para várias velocidades angulares e grava os resultados em um arquivo chamado **calculo.dat**.

```
to arquivo :v1 :v2 :dv
print [Calculando...]
openwrite "calculo.dat
setwrite "calculo.dat
for [v :v1 :v2 :dv] [pendulo :v]
setwrite []
close "calculo.dat
print [Resultados escritos em calculo.dat]
end
```

A curva que está na figura 22 mostra o resultado obtido. Podemos ver que o cálculo reproduz muito bem o comportamento dos dados experimentais.

Observe que o cálculo do período no programa **pendulo** utiliza apenas conceitos que são familiares no ensino médio (essencialmente: $F = m\,a$, $\Delta x = v\,\Delta t$ e $\Delta v = a\,\Delta t$). Isto ilustra uma característica dos computadores que os torna importantíssimos para o ensino de física: eles facilitam enormemente a modelagem matemática de fenômenos naturais, tornando-a acessível mesmo a quem conhece apenas a matemática pré-cálculo.

Apenas para referência, mostramos abaixo as linhas gerais do cálculo analítico do período do pêndulo. Ignorando as forças de atrito, o período é dado pela integral

$$T = 4\sqrt{\frac{R_G}{2g}} \int_0^{\Theta_{max}} \frac{d\Theta}{\sqrt{\cos\Theta - \cos\Theta_{max}}}$$

que resulta em

$$T = 4\sqrt{\frac{R_G}{g}}\, K\left(sen \frac{\Theta_{max}}{2} \right)$$

onde K(x) é a integral elítica completa. Para x pequeno K(x) pode ser aproximada por

$$K(x) = \frac{\pi}{2}\left(1 + \frac{x^2}{4} + \cdots \right)$$

de modo que

$$T = 2\pi\sqrt{\frac{R_G}{g}}\left(1 + \frac{\Theta_{max}^{\ 2}}{16} + \cdots \right)$$

ou ainda

$$T = 2\pi\sqrt{\frac{R_G}{g}}\left(1 + \frac{1}{16}\frac{R_G}{g}\Omega_0^{\ 2} + \ldots \right)$$

Colocado na figura 22 este resultado daria uma curva praticamente idêntica à obtida numericamente com o programa **pendulo**, que está mostrada. É claro que o cálculo analítico esboçado acima é muito mais difícil de compreender que o cálculo numérico. A comparação entre os dois métodos é reveladora, mostrando como o uso do computador pode tornar acessíveis tópicos tidos como matematicamente complexos.

7. Medida de turbidez da água

7.1. A montagem do experimento

Neste experimento utilizamos um LDR como sensor luminoso, ligado às linhas 3 e 1 da porta de jogos (o potenciômetro X1 e o ponto de +5V, respectivamente). Um tubo contendo água foi coberto com material opaco deixando apenas dois orifícios para a passagem da luz. O nível da água foi mantido bem acima do ponto de passagem da luz. Um esquema do aparato aparece na figura 23. O programa de aquisição é praticamente idêntico ao mostrado na seção 2.5, e não será repetido aqui.

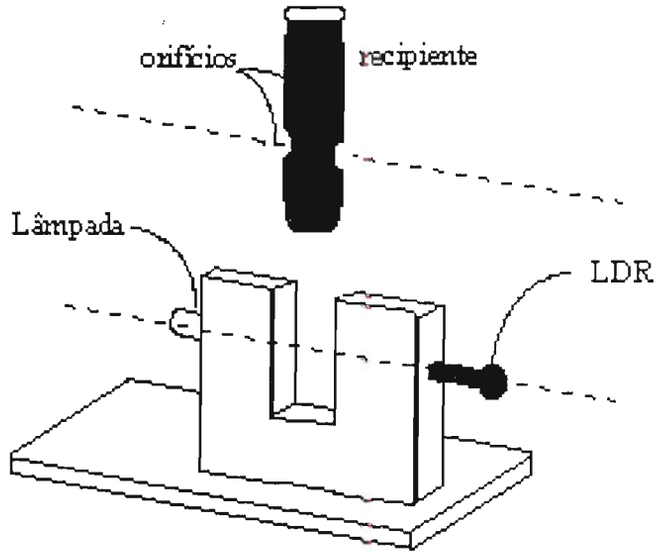


Figura 23

7.2. Resultados

A figura 24 mostra o sinal do LDR como função do tempo, começando com água limpa e máxima luminosidade. Cerca de 5 ml de tinta escura foram derramados na superfície da água aos 10, 50 e 90 segundos, aproximadamente. É possível acompanhar com a figura 23 a difusão da tinta na água. As quedas abruptas correspondem à passagem de grandes quantidades de tinta pela frente do sensor, e ocorrem um pouco depois dos derramamentos. Após um certo tempo a tinta está mais homogeneamente espalhada e um certo equilíbrio é alcançado, só perturbado por uma nova descarga de tinta.

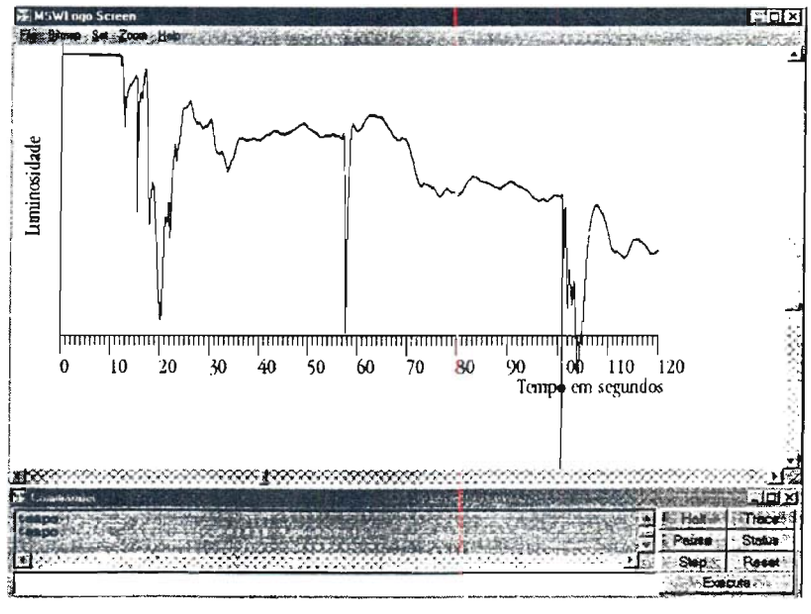


Figura 24

Podemos utilizar este aparato para estudar a evolução temporal de reações químicas, a difusão de partículas em líquidos, e até como medidor da qualidade da água.

8. Comentários finais

Neste trabalho nós discutimos a utilização da porta de jogos de um PC como interface para aquisição de dados. O sistema proposto tem as seguintes características:

1. Pode ser montado sem fazer modificações no computador.
2. Utiliza como sensores componentes eletrônicos simples e fáceis de encontrar.
3. É facil de compreender e implementar.
4. É versátil, podendo ser usado em muitos experimentos diferentes.
5. É bastante rápido, pelo menos para os objetivos de um laboratório didático, alcançando resoluções temporais da ordem de milisegundos.
6. Tem custo próximo de zero se o computador já estiver disponível.
7. Os programas de aquisição e análise de dados sãp todos escritos em Logo.

O sistema pode ser usado em laboratórios didáticos do ensino médio, em condições tais que *tanto alunos quanto professores são capazes de compreender e controlar todas as etapas do processo de aquisição e análise dos dados*. Isto se deve não apenas à simplicidade da interface, mas principalmente ao fato de que os programas de aquisição e análise podem ser inteiramente escritos em Logo.

A aquisição de dados pela porta de jogos não representa, obviamente, a única forma de utilização do computador em um laboratório didático. Além dos kits comerciais mencionados na Introdução (alguns até usam a porta de jogos), inúmeras outras propostas existem, com diferentes graus de sofisticação e custo [11-14]. O denominador comum de todas estas propostas parece ser a idéia de que ao introduzir computadores no laboratório didático passamos a ser capazes de:

1. Observar uma variedade maior de fenômenos.
2. Fazer medidas mais precisas, principalmente as que envolvem tempo.
3. Adquirir dados em maior quantidade.
4. Analisar os dados mais rapidamente.

São estes os motivos que, a nosso ver, tornam os computadores instrumentos tão atraentes nos laboratórios didáticos. Com eles podemos realizar experiências importantes que não seriam viáveis em um laboratório convencional. Análises que costumam ser impraticáveis com lápis e papel, ou mesmo com calculadoras, tornam-se simples e rápidas. Representações gráficas dos resultados experimentais, que em geral são penosamente elaboradas, podem ser criadas rapidamente a partir de arquivos de dados. Hipóteses e modelos podem ser testados com mais facilidade. Em resumo, o papel do computador em um laboratório didático é tornar observáveis fenômenos que normalmente estariam ocultos, e facilitar a sua análise. Ele permite ver mais e melhor.

Agradecimentos

Gostaríamos de agradecer aos Profs. Francisco Artur Chaves e Susana de Souza Barros, e ao Laboratório Didático do Instituto de Física (LADIF), pelo apoio dado à execução deste trabalho.

Apêndice

Este Apêndice contem a versão para SLogo de alguns dos programas em MSWLogo mostrados no texto principal.

```
aprenda lerpjogos
atribua "linhas []
atribua "x (portaentradab 513)
atribua "m 1
repita 8~
[
  atribua "b (bite :x :m)
  se :b=:m [atribua "b 0]
  atribua "linhas ji :b :linhas
  atribua "m 2*:m
]
mostre :linhas
fim
```

```
aprenda botao
atribua "fim 0
façaenquanto [passo] [:fim=0]
fim

aprenda passo
atribua "x (portaentradab 512)
atribua "a1 (bite :x 16)
atribua "b1 (bite :x 32)
atribua "a2 (bite :x 64)
atribua "b2 (bite :x 128)
se :a1=0 [parafrente 1]
se :b1=0 [atribua "fim 1]
se :a2=0 [paraesquerda 1]
se :b2=0 [paradireita 1]
fim
```

```
aprenda manche
atribua "zerox 400    ;centro da tela
atribua "zeroy 400
atribua "escx 0.5     ;escalas
atribua "escy 0.5
```

```
façaenquanto [leitura] [:b1=32]
fim

aprenda leitura
atribua "x1 (portajoystick 1)
atribua "y1 (portajoystick 2)
mudexy :escx*(:x1-:zerox) :escy*(:zeroy-:y1)
atribua "b1 bite 32 (portaentradab 513)
fim
```

```
aprenda resolucao
atribua "n 0
atribua "soma 0
atribua "soma2 0
repita 100000 ~
[
  atribua "t1 tempomili
  atribua "t2 tempomili
  atribua "dt :t2-:t1
  atribua "soma :soma+:dt
  atribua "soma2 :soma2+(:dt*:dt)
  se (:dt>0) [atribua "n :n+1]
]
atribua "percent :n/1000
atribua "resol :soma/:n
atribua "desvp raizq (:soma2/:n - :resol*:resol)
(escreva [Percentagem com dt>0:] :percent "%)
(escreva [Resolução:] :resol "ms)
(escreva [Desvio padrão:] :desvp "ms)
fim
```

Referências

1. *Pasco Scientific*
   http://www.pasco.com
2. *Vernier Software & Technology*
   http://www.vernier.com
3. M.L. de Jong e J.W. Layman
   Using the Apple II as a laboratory instrument
   The Physics Teacher 22 (May 1984) 291
4. W.M. Gonçalvez, A.F. Heinrich e J.C. Sartorelli
   Aquisição de dados com a porta de jogos de computadores Apple
   Revista de Ensino de Física 13 (1991) 63
5. J. Fuller
   Science Experimenters' Kit
   http://www.southwest.com.au/~jfuller/scikit.zip
   Exploradores de la Ciencia Kit
   http://www.teachers.ash.org.au/jfuller/spanish/sciencekit.zip
6. J. Fuller
   Games Port Input
   http://www.southwest.com.au/~jfuller/logotut/games.htm
7. G. Mills e B. Harvey
   MSWLogo
   http://www.softronix.com/logo.html
8. Núcleo de Informática Aplicada à Educação (NIED/UNICAMP)
   SuperLogo
   http://www.nied.unicamp.br/projetos/softw/logow/index.htm
9. Tomi Engdal
   Joysticks and other game controllers
   http://www.hut.fi/Misc/Electronics/docs/joystick/
10. R. Zelenovsky e A. Mendonça
    PC: um guia prático de hardware e interfaceamento
    Editora MZ, segunda edição, 1999.
11. R.D. Peters
    Experimental computational physics using an inexpensive microcomputer
    Computers in Physics (July/Aug 1988) 68
12. P.J. Collings e T.B. Greensdale
    Using the computer as a laboratory instrument
    The Physics Teacher (Feb 1989) 76
13. R.V. Ribas, A.F. Souza e N. Santos
    Um sistema de aquisição de dados de baixo custo para o laboratório didático
    Revista Brasileira de Ensino de Física 20 (1998) 293
14. D.F. Souza, J. Sartori, M.J.V. Bell e L.A.O. Nunes
    Aquisição de dados e aplicações simples usando a porta paralela do micro PC
    Revista Brasileira de Ensino de Física 20 (1998) 413