IOP PUBLISHING EUROPEAN JOURNAL OF PHYSICS

Eur. J. Phys. **30** (2009) 57–67 doi:10.1088/0143-0807/30/1/006

Operating a Geiger–Müller tube using a PC sound card

A A Azooz

Department of Physics, College of Science, Mosul University, Mosul, Iraq

Received 1 June 2008, in final form 1 October 2008
Published 6 November 2008
Online at stacks.iop.org/EJP/30/57

Abstract
In this paper, a simple MATLAB-based PC program that enables the computer to function as a replacement for the electronic scalar-counter system associated with a Geiger–Müller (GM) tube is described. The program utilizes the ability of MATLAB to acquire data directly from the computer sound card. The signal from the GM tube is applied to the computer sound card via the line in port. All standard GM experiments, pulse shape and statistical analysis experiments can be carried out using this system. A new visual demonstration of dead time effects is also presented.

Introduction

The use of a computer sound card as a data acquisition tool has gained increasing interest recently. With two independent analog input channels and the inherited analog to digital (A/D) conversion ability, the sound card is well suited to operate as a data acquisition device. Since all personal computers are equipped with a sound card, it becomes natural to attempt to use this devise as an inexpensive, reasonably fast and handy data acquisition tool. The sampling rate of a typical sound card can be as high as 44 100 kHz. The author has suggested using this device in obtaining Langmuir probe *I*–*V* characteristic data [1] and to demonstrate the Doppler broadening of sound waves [2]. A good review about the sound card advantages and limitation are presented in [3].

A GM tube is a very useful and instructive piece of equipment in most physics undergraduate laboratories. The accessory equipment used in conjunction with a GM tube are an HT power supply, an electronic pulse scalar and a pulse counter. Several basic undergraduate experiments can be setup using a GM counter system. These include determination of a GM plateau voltage range, the detector dead time, radioactivity half life, x-ray diffraction experiment, etc. Although GM tubes and power supplies are relatively cheap and widely available in most labs, the availability of scalar-counter systems may be very much limited especially in developing countries (this is the case with us at least). On the other hand, a large number of P1, P2 and P3 desktop and laptop computers are being abandoned simply because new faster and more capable generations are on the market. Increasing the number of

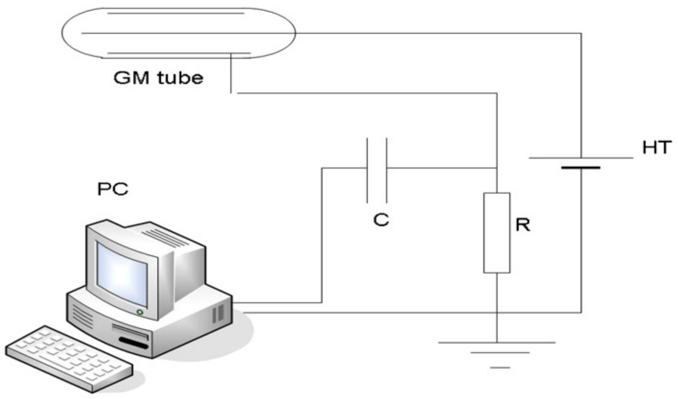

Figure 1. The basic experimental setup.

independently operating GM systems in any undergraduate physics laboratory will certainly give the lab instructor more manoeuvres with the number of experiments using this system that can be installed simultaneously. Many laboratories use computer data acquisition systems to acquire and analyse GM counter data using the LabView software. However, this can only be done in conjunction with some data acquisition hardware. Arqueros *et al* [4] have suggested an interesting method for applying the GM signal into the PC parallel port.

It is our purpose here to present and describe simple MATLAB-based software that uses the PC sound card for data acquisition and analysis of GM tube pulses. The advantages of using the soundcard are related to both its relatively high speed sampling rate as compared to other data acquisition devices and to is easy access availability in all computers.

Experimental setup

The circuit diagram used is shown in figure 1. The circuit produces positive pulses that are applied directly to the input jack of the PC sound card. Alternative circuits with negative pulse configuration can be equally used. This can be done after changing all the $>$ signs to $<$ signs for the three if loops in the software. The only restriction in both cases is that the absolute pulse height is within the 1 V maximum safe limit of most sound cards. Pulses with higher heights will be clipped at 1 V by the sound card. Very high voltages (over 10 V) may cause damage to the sound card. In our case, the values of the resistance and capacitance, $R = 4.7\,\mathrm{k}\Omega, C = 100\,\mathrm{nF}$, produced pulses of about 0.6 V. Lower values of RC can be selected, but at the expense of pulse height. The sound card can easily detect a pulse of about 20 mV with over 3 dB signal-to-noise ratio.

The software

The software uses the MATLAB data acquisition toolbox to acquire analog data from one or both channels of the 'line in' port of the computer sound card. The software is very simple to use. Two independent MATLAB M-files named GM1(N,T) and GM2(N,T) are listed in appendixes A and B. They can be directly copied and pasted to new MATLAB M-files. Saving these file in the MATLAB 'work' directory (this is where MATLAB keeps all user programs and data files by default) will make the system ready. The only prerequisite for

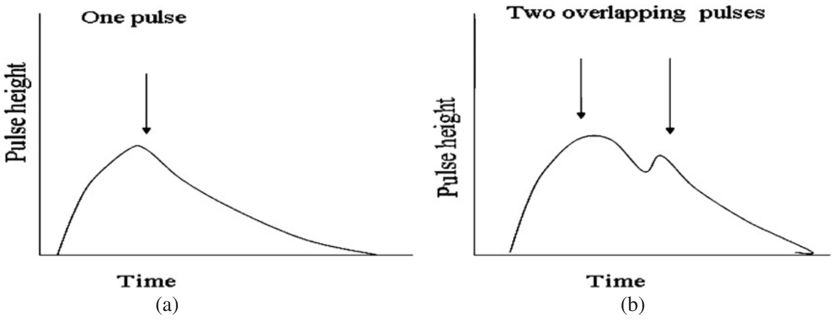

Figure 2. In the pulse identification process, the program counts extreme points as pulses: (a) is counted as one pulse and (b) is counted as two pulses.

running the programs is to have any version of MATLAB software installed on the PC. Typing the statement X = GM1(N,T), where 'N' is the number of times the program will repeat measurements for N separate periods of time of T s each, will get the PC to work as a standard scalar-counter system. The total counts for each period T will be output in X. The program uses two complementary criteria to identify each pulse. The first is that the voltage is above a certain limit. This is necessary to exclude any noise interference and also to drastically reduce the CPU time. This limit is set to 0.4 V by the software default. It can be changed however from the 'if' statements in line 21 in GM1 or GM2. The second is that the values of the two points to the right and to the left of a particular point under consideration should be lower than the voltage value of the considered point. This ensures that for several points constituting a particular pulse, only one extreme point is counted. This also helps reduce system dead time effects. This is because even if two pulses arrive within a time period smaller than the full electronic pulse width resulting in some overlapping, the program has the capability to count both of them. This is demonstrated in figure 2.

The number of counts registered during the run appears on the main MATLAB work space as 'total counts'. It will also give the Geiger pulse width and the area under the pulse. The value for the width obtained represents the pulse width at half maximum. It is evaluated by dividing the integrated area under the pulse by twice the pulse height (triangular approximation). The input variable N can take any desired value enabling the program to run for any length of time even unattended. Due to the fact that the program buffers the data to the RAM, the second input variable T measured in seconds is limited by the size of the RAM of the computer used. Using $T = 50$ s did not produce a memory overflow on a P1 64 MB RAM, an almost obsolete computer. Higher T values can be used on computers having higher RAM. Using $T = 480$ s just came short of producing a memory overflow on a P4 524 MB RAM computer. The total number of counts is the sum of the number of all pulses detected over the full period of time $N \times T$. For example, typing the statement $X =$ GM1(1,1) will make the program run for 1 s only while the statement $X =$ GM1(600,60) will induce the program to run for ten continuous hours divided into periods of 1 min each. Besides the numerical data for the total counts registered, the individual counts obtained over each period T, the mean, the standard deviation of the N measurements over T s each and the pulse width, the program will output a dynamic plot of counts registered over each run for a time period T. This program is suitable for direct use of a PC as a substitution for the standard scalar-counter system and for monitoring radiation over prolonged periods of time. PC loudspeakers or headphones can be turned on to hear the sounds of actual pulses.

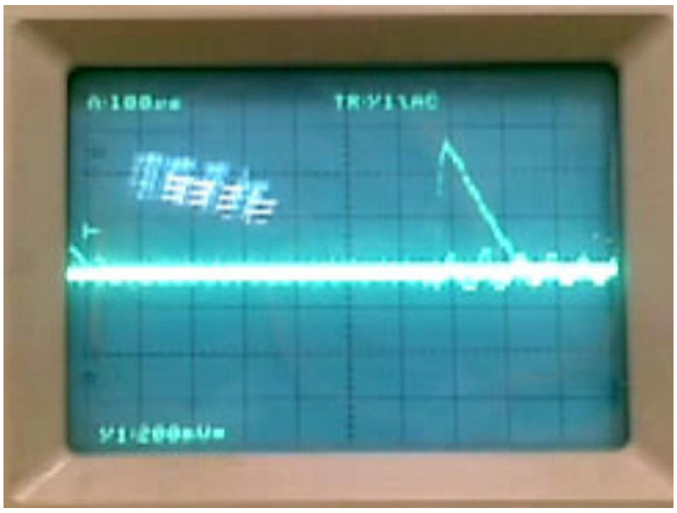

Figure 3. Oscilloscope screen picture of an actual GM pulse.

The second program GM2(N,T) functions in a way very similar to GM1, but it is designed to perform some additional tasks. This program will output the following subplots:

(1) A pulse shape subplot which corresponds to the projection of one typical pulse as detected by the computer. The pulse height is in volts, and the time is in seconds. This is very similar to what one would obtain using an oscilloscope. A typical actual oscilloscope pulse obtained is shown in figure 3. This pulse compares well with that shown in figure 5(a)
(2) A 50 bin histogram representing the time distribution measured in seconds between all successive pulses for all pulses throughout the complete run. This demonstrates the Poisson distribution nature of these times.
(3) A 20 bin histogram of the counts/s averaged over each period of T s throughout the complete run.
(4) A plot of the average counts per second (averaged over T s) against the time in seconds. This will indicate if there is any significant change in the level of radioactivity at any instance in time.

Experimentation

The first experiment carried out using this system is the standard count–voltage dependence. The voltage plateau is determined using a radioactive source. The results are shown in figure 4. The measurements are for 3 min using a Co^{57} radioactive source with activity of about 0.14 μCi. These results are similar to those recorded using the standard electronic scalar-counter system, as shown in the same figure. In both cases the threshold, the plateau and the proportional voltage regions were the same. It is worth mentioning here that the tube is over 30 years old and the plateau region is not that wide. Other experiments using GM counters such as x-ray diffraction, γ ray absorption and determination of counter dead time are currently being run by students in our laboratories using almost obsolete P1 and P2 computers.

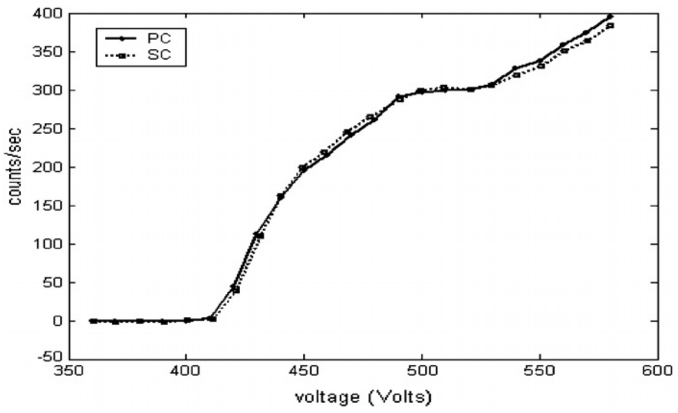

Figure 4. Count–voltage characteristics obtained using a PC and a standard scalar-counter system.

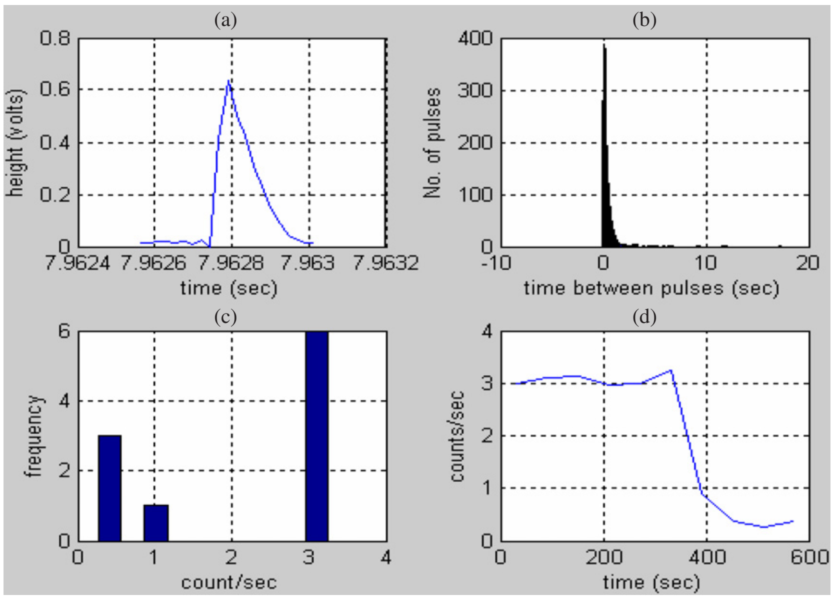

Figure 5. Plots produces by the program GM2 after a 10 min run. The radioactive source is removed during the seventh minute.

Figure 5 shows typical plots produced by the program GM2 after a 10 min run in which the radioactive source was removed during the seventh minute.

The program GM2 is equipped with facilities to allow for a more detailed analysis of data. The program can be activated by typing the statement [$x1$, $x2$, $x3$, $x4$] = GM2(N,T) (or any other desired string variable names for that matter) in a MATLAB work space. The numerical output will include four arrays. The output array $x1$ represents the number of counts registered during each sub-period T. The size of this array is equal to N. The array $x2$ represents the time difference between any two successive pulses registered over the entire run. The size of this array is equal to the total number of counts registered minus one count. The two variables

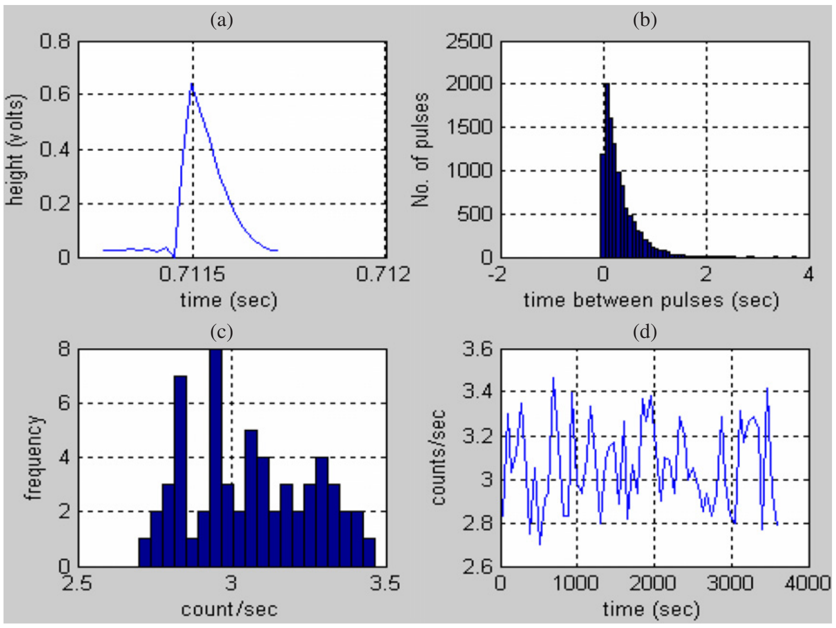

Figure 6. Results of a 1 h run with 60 s sub-periods.

$x3$ and $x4$ are the pulse height–time data corresponding to 20 points distributed on the two sides of the peak of the pulse. All four of these variables can be saved on a hard disk for any further analysis. MATLAB saved data can be directly communicated to the EXCEL Microsoft program.

Radioactive decay is regarded as a good tool for demonstrating the Poisson distribution law. The fact that MATLAB contains both statistical analysis and curve-fitting toolboxes besides the data acquisition one makes the study of the statistical nature of radioactivity all the more handy. By simply selecting the value of the argument N large enough, one can study the distribution of a repeated number of counts. Figure 6 shows the results of running the program for 1 h. For both figures 5 and 6, the plots are presented in the same form the MATLAB produces by default, and no editing is made. As a simple demonstration of how these results can be used for a further analysis, the data of figure 6(b) representing the distribution of time periods between successive pulses are plotted in a magnified form in figure 7.

The solid line represents the result of the fit to the Poisson distribution law of the form

$$P(t)\,\mathrm{d}t = K_1\,\mathrm{e}^{-K_2 t}\,\mathrm{d}t. \tag{1}$$

The data of the first two bins in the histogram are not included in the fit. This is because these two bins correspond to successive pulses separated by very short periods of time where the effect of counter dead time is at work. This point will be further discussed later. The constants K_1 and K_2 are left as free parameters to be determined by the fitting program. Over 95% confidence level fit is obtained with $K_1 = 2587.6$ and $K_2 = 3.1568\ \mathrm{s}^{-1}$. The fitted equation is in good agreement with experimental data. One interesting feature in the histogram is a significant deficiency of events in the first bin of the histogram representing events with a very small time separation between successive pulses. This is a direct manifestation of the counter dead time effects. The system does not have any problem in registering pulses

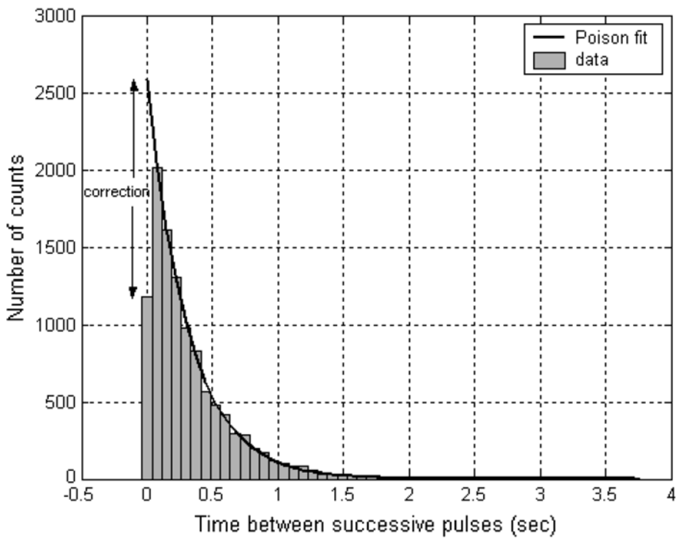

Figure 7. Distribution of time periods between all successive pulses.

separated in time by some fraction of the pulse width. However, pulses arriving at shorter times may not be counted properly. This is a well-known dead time effect, and dead time corrections are customarily applied to counter measurements. The counting rate dead time correction equation is

$$N = \frac{n}{1 - n\tau} \tag{2}$$

$$\tau = \frac{N - n}{nN}, \tag{3}$$

where n is the counting rate measured by the detector, N is the real counting rate and τ is the detecting system dead time.

Figure 7 can lend itself as a tool for an approximate yet demonstrative method for evaluating the dead time. The total number of counts measured on this histogram is $n = 10\,954$. Now if one tries to compensate for the missed events in the first two bins on the histogram by subtracting the fit extrapolation to the first two bins excluded previously, a total of about 1466 events would be needed to bring the data of these two bins in line with the fitted Poisson distribution curve well describing the rest of the data. The above two numbers when added can give us the approximate actual total counts $N = 12\,400$ corrected for dead time effects. Substituting for N and n in equation (3) will give us $\tau = 10.8\ \mu$s. This result is approximately equal to about one fifth of the pulse width calculated by the program. This may be considered as an indication that the program is identifying and consequently counting overlapped pulses in the way described above (figure 1).

Several other experiments can be carried out using this software. For example, students can study the effect of varying the circuit elements R and C on pulse properties. Another experiment is related to the calculation of the number of electron ion pairs produced. This can be obtained from the area under the pulse. Pulse rise and decay times can also be analysed.

Conclusions

The above description represents only samples of the statistical and physical analysis that can be performed using the data output of this system. The main advantage of using this or similar data acquisition systems is the ability not only to count pulses over any desired period of time, but also to record the time between all successive pulses, and pulse shapes. Several regimes for collecting, averaging and analysing groups of data can be designed depending on a particular point of interest.

Experimental details

(1) The line in recording level controls of the audio setting of the PC must be selected.
(2) It is advisable to check the pulse height by an oscilloscope before applying it to a PC.
(3) Before running the program for long periods of time, a test run using GM2 for a few seconds duration is useful for checking the pulse shape. A pulse with a plateau at the top is an indication that the pulses received are higher than 1 V and they have been clipped by the sound card. This can produce incorrect measurements. In such a case, the value of the resistance must be reduced to a proper value

Appendix A

```
function f=GM1(N,T)
k=0;
hold
tic
for j=1:N;
RUN=j
% acquire data
ai=analoginput('winsound');
   addchannel(ai,1:2);
   ai.samplerate=44100;
   ai.samplespertrigger=44100*T;
       ai.triggertype='immediat';
   start(ai);
   [u1,tt]=getdata(ai);
   delete(ai)
clear ai
f(j)=0;
n=max(size(u1));
% Sort out and count pulses
for i=2:n-1;
   if u1(i)>.4
       if u1(i)>u1(i+1)
          if u1(i)>u1(i-1)
          f(j)=f(j)+1;
          k=k+1;
          end
       end
   end
```

```
    end
    ttt(j)=toc;
    plot(ttt,f)
end
'Total counts'
sum(f)
'Average counts per T seconds'
mean(f)
'standard deviation for N readings of duration T each'
std(f)
```

Appendix B

```
function [f,DT,xx,yy]=GM2(N,T)
k=1;
t1=cputime;
for j=1:N;
RUN=j
% acquire data
ai=analoginput('winsound');
    addchannel(ai,1:2);
    ai.samplerate=44100;
    ai.samplespertrigger=44100*T;
        ai.triggertype='immediat';
    start(ai);
    [u1,tt]=getdata(ai);
    delete(ai)
clear ai
f(j)=0;
n=max(size(u1));
% Sourt out and count pulses
tic
for i=2:n-1;
    if u1(i)>.5
        if u1(i)>u1(i+1)
            if u1(i)>u1(i-1)
        f(j)=f(j)+1;
        TT(k)=tt(i)-toc;
            k=k+1;
            end
        end
    end
    end
end
% part one plot the pulse
xx=u1;
[a,m]=max(xx);
xx=tt(m-10:m+10);
yy=u1(m-10:m+10);
```

```
yy=yy-min(yy);
'Area under the puls
Area=trapz(xx,yy)
'Pulse width (sec)'
Area/(2*max(yy))
subplot(2,2,1)
plot(xx,yy)
title('(a)')
xlabel('time (sec)')
ylabel('height (volts)')
grid on
%part two produce the time distribution
DT=diff(TT);
kk=find(DT<0);
DT(kk)=[];
a=max(DT);
b=min(DT);
d=a-b;
c=b:d/50:a;
subplot(2,2,2)
hist(DT,c)
title('(b)')
xlabel('time between pulses (sec)')
ylabel('No. of pulses')
grid on
%part three produce the count distribution
subplot(2,2,3)
hist(f,20);
title('(c)')
xlabel('count/sec')
ylabel('frequency')
grid on
t2=cputime;
z=t2-t1;
zz=z/(2*N):z/N:(z-(z/(2*N)));
subplot(2,2,4)
plot(zz,f/T)
title('(d)')
xlabel('time (sec)')
ylabel('counts/sec')
grid on
'Total counts'
sum(f)
'Mean number of counts per T seconds'
mean(f)
'Standard deviation of N measurements of T second duration each'
std(f)
```

References

[1] Azooz A A 2006 Analog data acquisition for obtaining *I–V* characteristics using sound cards *IEEE–CiSE* pp 50–55

[2] Azooz A A 2007 Experimental demonstration of Doppler spectral broadening using the PC sound card *Am. J. Phys.* **75**

[3] Ternström S 2007 PC's in voice acoustics research http://www.speech.kth.se/voice/white/WhitePaperUsing-PCs.pdf (February 2007)

[4] Arqueros F, Blanco F and Jiménez de Cisneros B 2004 Studying the statistical properties of particle counting with a very simple device *Eur. J. Phys.* **25** 399–407