Race Random Numbers

Physics 434 Final Project Report

BY :

Pratyush Painuly

Kevin Jamison

12/12/2014

Contents -

1. Introduction ….. 3
    - Purpose….. 3
    - Scope….. 3
    - Definitions and Acronyms….. 3
2. System Overview….. 3
3. System Architecture….. 6
    - Architectural Design….. 6
    - Design Rationale….. 7
4. Human Interface Design….. 8

1. Introduction -

a. Purpose

This software design document describes the architecture and system design of and acoustic resonator analyzer.

b. Scope

This software is designed to generate a true random number by using electrons created in the decay of uranium 238, which are measured with a Geiger counter. As a test of the random numbers, the software will play a game with 2 random number generators by summing the random values they generate until a specified number is reached by one of the generators. The user can set the number of bits of the randomly generated numbers and what number the generators should sum to. After the summation is complete, the software displays the sum of each generator, indicates which generator won by turning on a light on the interface, and displays the total amount of numbers generated per generator. The results of partial sums for both generators are also displayed on a plot for the user.

c. Definitions and Acronyms

VI: Virtual Instrument, terminology for a Labview program

2. System Overview -

Our project uses a sample of Uranium-238 dissolved in water. We measure the beta decay of the sample using a Geiger Counter to measure the number of atoms leaving the sample per time interval. The number of atoms per second for the sample is a random set and is analyzed as a poisson distribution. We measure the time period of each data point and then use that data to compute a random binary number. Then we convert that binary number to a decimal number and finally compute the random number. We perform the experiment several times and the verify that the data points are random and follow a poisson distribution.The details about the background science behind the experiment is as follows :

We take 1g of U-238 dissolved in water in a small bottle for using it with the Geiger Counter. U-238 undergoes an alpha decay and has a half-life of $4.468.10^{9}$a [1]. The decay product is Uranium X1(or Protactinium) Pa-234. Nw when Pa-234 undergoes a beta decay, the decay product is ^{234m}Pa. In beta decay, we release an electron, which does not have any nucleons(protons and neutrons). Hence this particular isotope of Uranium has 234 nucleons. This isotope ^{234m}Pa(Uranium X2) with a half life of 1.16 min is exact what we need for our

[1] a : is a unit of time measurement defined to be equal to 365.25 days of 86400 seconds each.

^{234m}Pa —> e- + ^{234}U + antineutrino

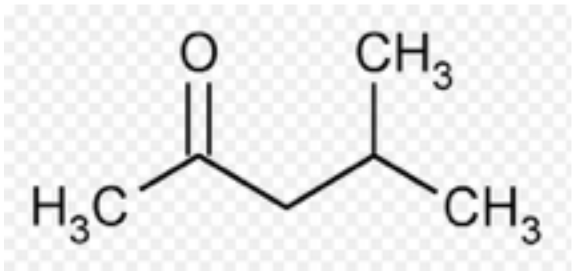
experiment. We want ^{234m}Pa to undergo a beta decay and once it does, the decay product is ^{234}U ^{234}P. Hence, we measure the beta decay of the daughter isotope which has the equation :

We use the Geiger counter to detect the electrons (beta particle) leaving the sample. To be sure that this is the correct choice and the mean lifetime of this ^{234m}Pa is indeed a low number, we verify using the flowing equations :

^{234m}Pa mean lifetime : tau = (1.16) / 0.683 min X 60s/min = 100s [mean lifetime is half-life/ ln(2)]. Hence the average number is approx. 1.3min [2].

Now the question we ask ourselves is that how do we extract Protactinium? The sample was given to us in the lab. The sample was prepared by using the Liquid-liquid extraction also known as solvent extraction and partitioning. “Liquid–liquid extraction also known as solvent extraction and partitioning, is a method to separate compounds based on their relative solubilities in two different immiscible liquids, usually water and an organic solvent. It is an extraction of a substance from one liquid into another liquid phase. Liquid–liquid extraction is a basic technique in chemical laboratories, where it is performed using a variety of apparatus, from separatory funnels to countercurrent distribution equipment.” [3]

For our project we use water and methyl isobutyl ketone ($C_6H_{12}O$), with a density of 0.8. Pa is soluble in the solution and U is soluble in water. Now when we shake the sample for about 30 seconds, it undergoes a beta decay and at the same time we keep it next to the Geiger counter to detect the electrons and run our program which will compute the random numbers and show the result on screen.



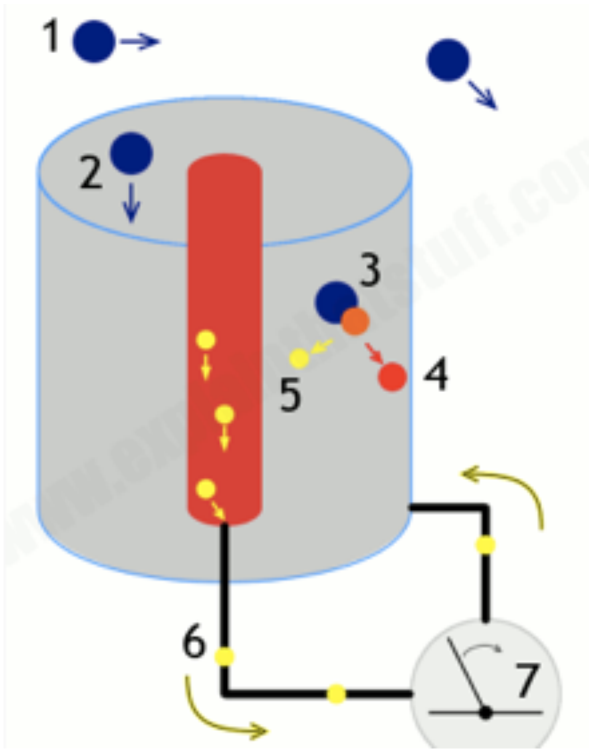
So why does the beta decay occur? It occurs when the nucleus of our isotope, is unstable. It has too many neutrons compared to protons. So one of the neutrons decays into an electron because it is unstable for the simple reason that the neutron/ proton ratio is significantly greater than what is stable for their mass range. So the ratio is away from the stability ratio and this makes the nucleus unstable. Hence the sample containing the isotope emits electrons and since it has a low half-life, it fits very well for our random number generator.

[2] https://github.com/uw-labview/week8/blb/master/week8.pdf

[3] http://en.wikipedia.org/wiki/Liquid%E2%80%93liquid_extraction

Now lets talk about the Geiger counter and its workings. “The Geiger–Müller counter, also called a Geiger counter, is an instrument used for measuring ionizing radiation. It detects radiation such as alpha particles, beta particles and gamma rays using the ionization produced in a Geiger–Müller tube, which gives its name to the instrument.” [4]

From wikipedia, “The G-M tube produces a pulse output which is the same magnitude for all detected radiation, so a Geiger counter with an end window tube cannot distinguish between alpha and beta particles. A skilled operator can use distance to differentiate alpha and high energy beta, but with the detector in close contact with the radiation source the types are indistinguishable. The "pancake" Geiger-Muller detector is a variant of the end window probe, but designed with a larger detection area to make checking quicker. However the pressure of the atmosphere against the low pressure of the fill gas limits the window size due to the limited strength of the window membrane”.



The figure explains the workings of the Geiger counter GM tube. [5]

- The electrons from the sample due to the beta decay ready to flow into the tube.
- The electrons enter the G-M tube.
- The radiation (dark blue) collides with gas molecules in the tube (orange), hence, ionization is caused.Therefore, some of the gas molecules are converted into positive ions (red) and electrons (yellow).
- The positive ions go towards the outer side of the tube.
- The metal wire(red) inside has a high positive voltage. Hence the electrons are attracted to the metal wire running down the tube.
- Electrons in the wire create a current in a circuit connected to it.

[4] http://en.wikipedia.org/wiki/Geiger_counter

[5] http://www.explainthatstuff.com/how-geiger-counters-work.html

- The metal needle(the meter on the counter) gets deflected by the electrons.

The data points which the Geiger counter records follow a Poisson distribution. Poisson distribution is a discrete probability distribution that gives us the probability of a given number of events which occur in a fixed interval of time and if these events occur with a known average rate and independently of the time since the last event. For our experiment, the time interval is 1 second. So we get the number of electron emitted or detected by the counter for each second and the data set follows the poisson distribution. The Poisson distribution can be defined as :

A discrete random variable X follows the poisson distribution with lambda > 0 , if, for k = 0,1,2,3,4,5…., the probability mass function X is given by :

$$f(k;\lambda) = \Pr(X = k) = \frac{\lambda^k e^{-\lambda}}{k!},$$

where

- e is Euler's number ($e = 2.71828...$)
- $k!$ is the factorial of k.

The number generated is a binary number which is converted into a decimal number. For example, if the user selects 3 bits, the maximum possible number generated will have three bits(either 0 or 1 for each bit). The number will be a three bit combination of 0's and 1's. The maximum possible value will be 2^3 which is equal to 7 (111 in binary) and a minimum of 00 i.e 0 in decimal. We will have two geiger counters connected for this project which will give us two different random numbers with the same number of bits.

3. System Architecture -

a. Architectural Design

The software consists of a main VI called Race Random Numbers that performs the sums of the random numbers and displays all of the information to the user and contains the user controls. It also contains 2 sub VI's called Random Number Generator and Random Number Generator 2 that each generates a random number.

Random Number Generator:

This VI accepts a integer number as input which is used to specify the number of digits of the random binary number that it will generate. To generate the number, the VI uses a DAQ Assistant Module in Labview to collect data from a Geiger counter that is sampling electrons from uranium 238 decays. The Geiger counter sends a digital pulse to the DAQ counter when it detects an electron. The DAQ then measures the period from one electron to the next. The software contains a for loop for the DAQ that builds an array of 2 integers that are the values of the periods of electrons from the uranium. The integers are then compared to see which is larger. If the first is larger, then a 0 is given as output, if the first is smaller, then a 1 is given as output, and if they are the same, then a 2 is given as output. This for loop is nested inside a while loop which takes these 0,1,2 values and builds a binary number. The while loop ends when the sum of the number of 0 and 1 in the array is the same as the bit number provided by the user. Next, if any 2 values exist in the array, then they are deleted. Finally the array is converted to a string and then outputted as an integer.

Race Random Numbers:

This VI allows the user to specify the number of bits of the numbers generated and the sum total. Both random number generators are placed inside a while loop that sums their values and checks to see if either has reached the total. When one does reach the total, the while loop ends and the display light for the win is lit on the interface. The while loop also outputs an array of the partial sum values generated and plots it on a graph for the user to see.

b. Design Rationale

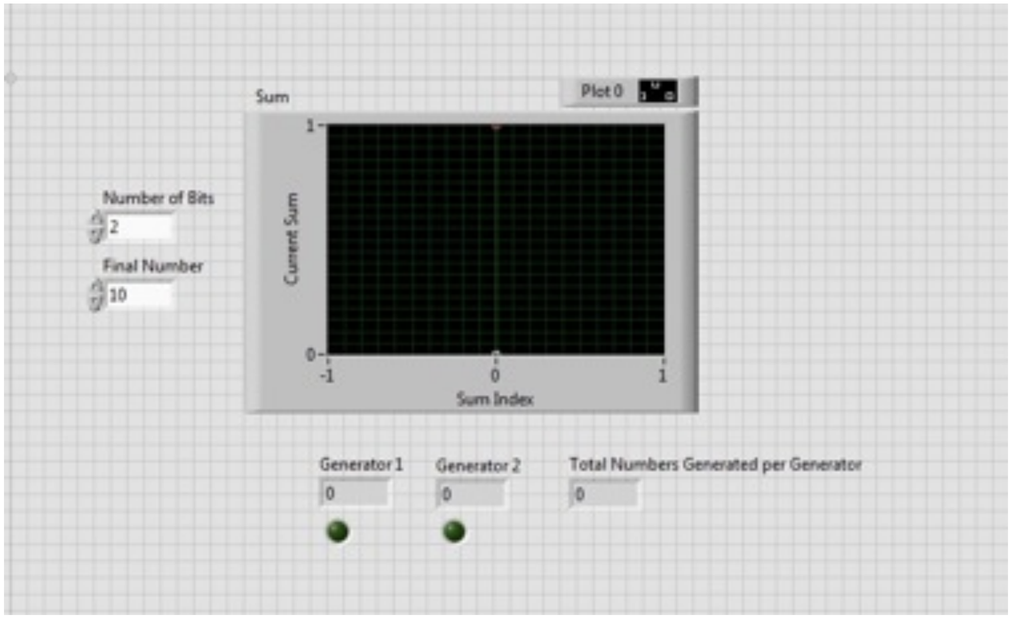
For the Race Random Numbers, it was necessary to create 2 different random number generators because the DAQ module cannot easily be told which counter on the DAQ breakout box the Geiger counter is attached to. To set the counter, the VI must be told this specifically inside the module, so 2 different Vis were created, however, their only difference is the counter the DAQ uses.

We thought that it was best to make the random number separate from the race VI since the application of random number generation is vast, and the Race Random Numbers was only meant as a demonstration. The Random Number Generator VI can easily be used in other Vis by simply specifying how many bits the random number should be.

We chose to measure the random number by comparing the periods because it was the best method we could implement. However, what we would do going forward is to use the Poisson distribution generated by the decays and then convert this to a uniform distribution which we could then use to generate the random numbers. This is better than our current method because our method assumes that the Poisson distribution would be symmetric about it's peak, however, it is not and hence one side is weighted more than the other.

4. Human Interface Design -

The picture below shows the main interface which the user will see the random numbers which are generated -



The VI is called Race Random Numbers. The VI lets the user select how many bits should the binary number should be. In the example above the user has selected two bits for the VI. Hence, the maximum possible binary value that we can get is 11 which is 2+1 = 3 in decimal. Here, we have two geiger counters connected which will give us two separate decimal numbers. These counters are represented by Generator 1 and Generator 2 which shows the current sum of all the numbers computed by the respective Generators. The VI also asks the user to give a final number. This is the number which we want the program to equal or exceed. So when we run the program, the program computes two decimal numbers., and if both of the numbers are lower then the final number, the the program goes on and compute another set of random decimal numbers and add the previous number from generated 1 to this new decimal number from Generator 1. Similarly, it add the new decimal number from Generator 2 to the previous decimal number from Generator 2. Hence, we have two different cumulative sums in the program. As soon as the sum from either of the generator either equals or exceeds the final number, the program stops and the respective Generator wins. All the decimal numbers computed by both the generators will be displayed on the waveform graph.