Microprocessor Lab Project Report

True Random Number Generator using Johnson noise and Chaotic Mapping

By:

Saurabh Mogre

Niladri Chatterji

Kartik Kothari

Guidance:

Prof. Pradeep Sarin

Department of Engineering Physics

Indian Institute of Technology Bombay

November 16, 2012

1 Introduction

The aim of the experiment is to generate truly random number using noise generated in a physical phenomenon. In this case, we have used the thermal noise created in a resistor as the starting point to generate random numbers. We amplify the noise and process it on a microcontroller to obtain strings of random numbers.

2 Thermal noise in a resistor

Johnson–Nyquist noise is the electronic noise generated by the thermal agitation of the charge carriers inside an electrical conductor at equilibrium, which happens regardless of any applied voltage. Thermal noise in an idealistic resistor is approximately white, meaning that the power spectral density is nearly constant throughout the frequency spectrum. Additionally, the amplitude of the signal has very nearly a Gaussian probability density function.

3 Measurement of noise

3.1 Theory

In a typical resistor, the thermal noise has a magnitude in the μvolt range, which needs to be amplified to be detectable on the instruments used. We use a balanced wheatstone bridge as a noise source. The voltage across the balanced arms is measured. In case of a noise free system, the voltage is identically zero. However, if we account for noise, there is a small signal across the arms in addition to a DC component which is caused due to mismatch in the values of resistances, essentially causing an unbalanced wheatsone bridge. The following section describes the implementation of this system.

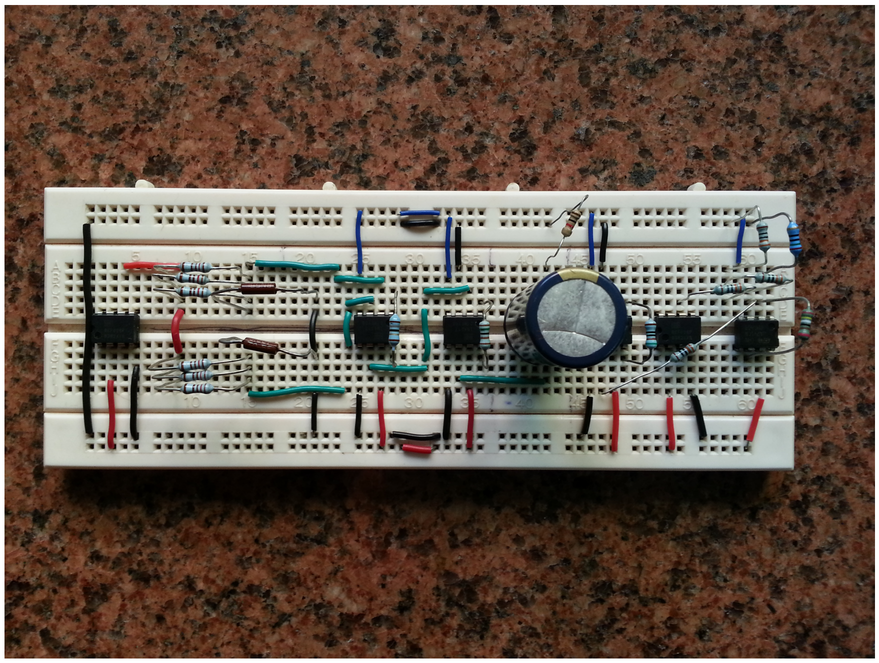
3.2 Circuit

3.2.1 Voltage Supply

The voltage source for the sytem was regulated to 5V using the Texas instruments IC REF02. The voltage regulator provides a steady 5V supply within 0.3% and has a low noise rating of 15μV.

3.2.2 Wheatstone bridge

The wheatstone bridge was designed using precision resistors of tolerance<1%. The values of the resistors were chosen such that they were within the load driving capacity of the Voltage regulator.



3.2.3 Two stage amplification

The signal from the arms of the wheatstone bridge was fed to instrumentation amplifier INA214 by Texas Instruments. Two gain stages were used to reduce bandwidth limiting and to provide enough amplification. The gains in the two stages were 500 and 5 respectively. These values of the gain give a signal in the range of 60mV

3.2.4 Filtering of DC offset

There is a small DC component arising from mismatch in resistor values which gets amplified along with the noise signal. It was filtred out using a high pass filter with a very high Capacitor value. The filter has a cutoff value of 50Hz.

3.2.5 Third stage amplification

The filtered output was then amplified with a gain of 16 to give a signal of 1V pk-pk value.

3.2.6 Noise input to Arduino

The resulting signal from the capacitor could not be fed directly to the microcontroller for processing, as the input range for the Arduino is 0 to 5V. To get the signal in this range, a voltage adder circuit was designed to add a stable 2.5V DC voltage from a REF02 using the opamp LM741.

4 Processing of noise on Arduino

4.1 Description

The Arduino uses the chaotic logistic map to randomise the input values. The logistic map is described by $x_{n+1} = Ax_n(1 - x_n)$. For most values of A>3.45, this system exhibits chaotic behavior. The value of A for our system was chosen to be 3.99
The amplified noise input was fed to the arduino and used as an initial condition for the logistic map. The map was run for for a long time with variable number of iterations depending on the input value.

4.2 Code

This is the code used to generate random numbers using the noise input

```
//This is the code to generate random numbers by running
//a chaotic map on a seed obtained through amplification
//of thermal noise in a resistor
//Saurabh Mogre, Niladri Chatterji, Kartik Kothari
float val=0.0;
float prev=1.0;
float temp=0.0;
int b=0;
void setup(){
  Serial.begin(9600);
}
void loop(){
  val=analogRead(2);//Read value from arduino
  val=val/1023.0;  //Scale value between 0 and 1
  temp=val;
  for(int i=0;i<((prev+1)*1729);i++){  //Run the chaotic map variable number of times
    val=3.99*val*(1.000-val);
    }
  prev=temp;
  b=val*10000;
  if(b<10000 && b>0) Serial.println(b); //Write output to arduino if within limits
}
```

5 Results

Here are some of the results obtained:

5.1 Frequency distribution of noise obtained

The result is close to a gaussian distribution as expected.

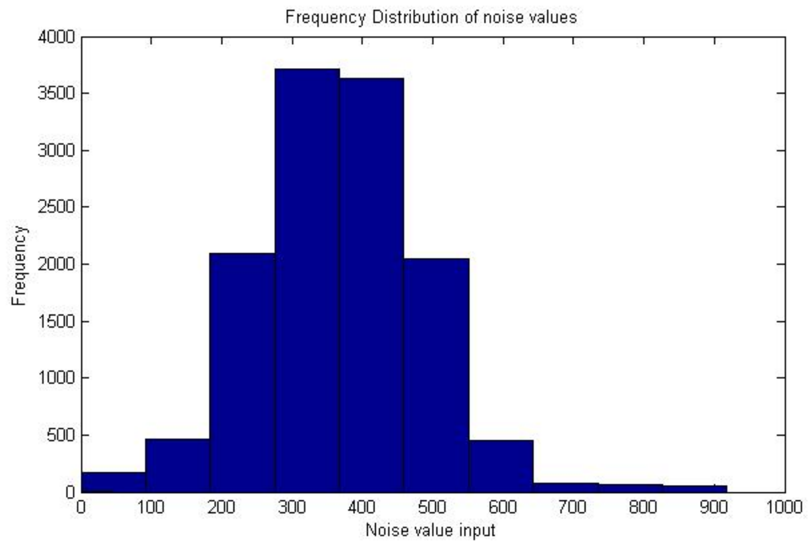


Figure 1: Frequency distribution of noise input

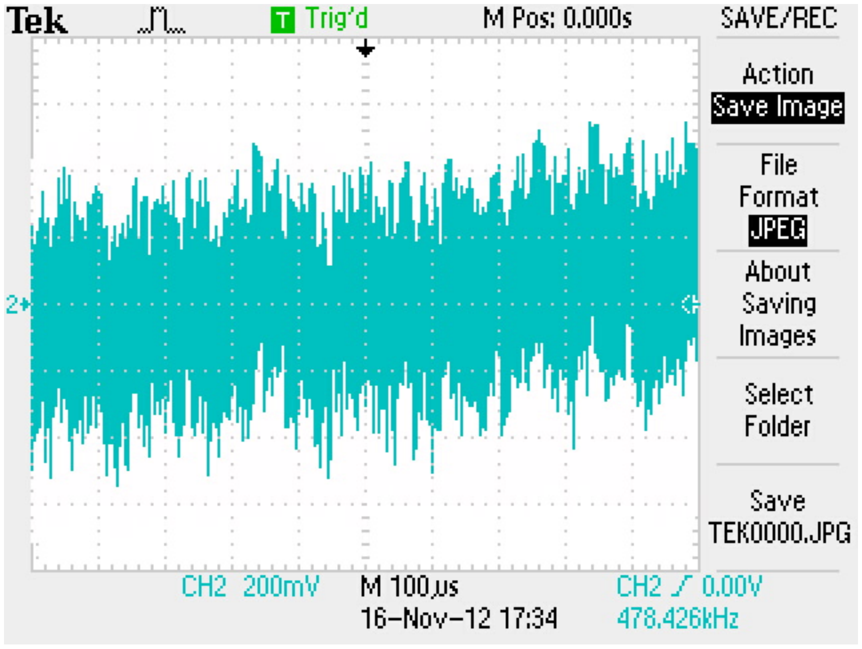


Figure 2: Time variation of noise

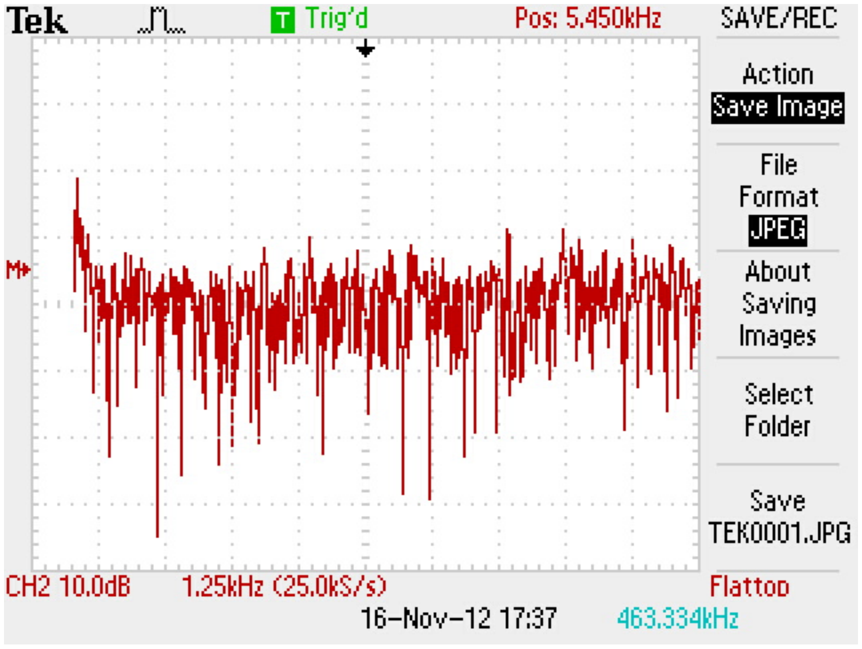


Figure 3: FFT of noise data

5.2 Frequency distribution of stock arduino random() function

The following plot shows the frequency distribution on the output of random(1,10)

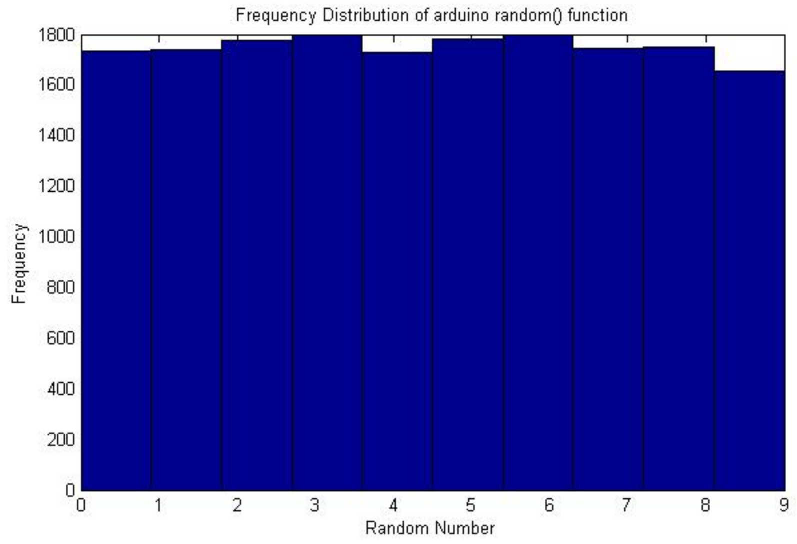


Figure 4: Frequency distribution of values obtained through arduino random number generator

5.3 Frequency distribution of obtained data

The units digit of the number obtained was considered as the random variable. The frequency distribution of the numbers was plotted using MATLAB.

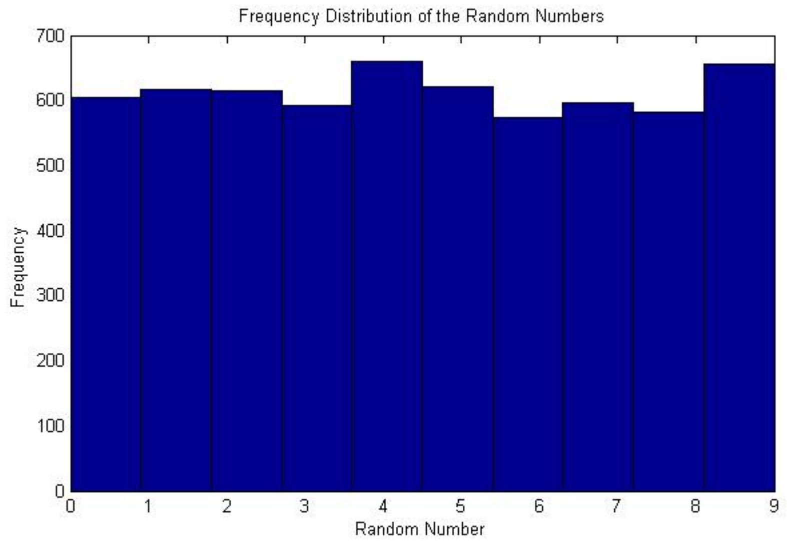


Figure 5: Frequency distribution of values obtained

This figure shows the obtaines time series plot for the same data.

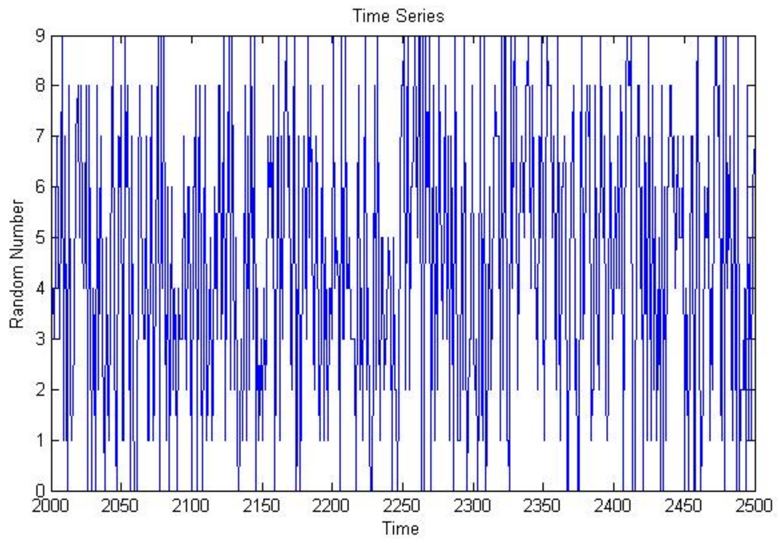


Figure 6: Time series of values obtained

5.4 Numbers obtained without using noise input

The following numbers were obtained by leaving the arduino input floating and were processed similarly. It is seen that the noisy input generates better random numbers.

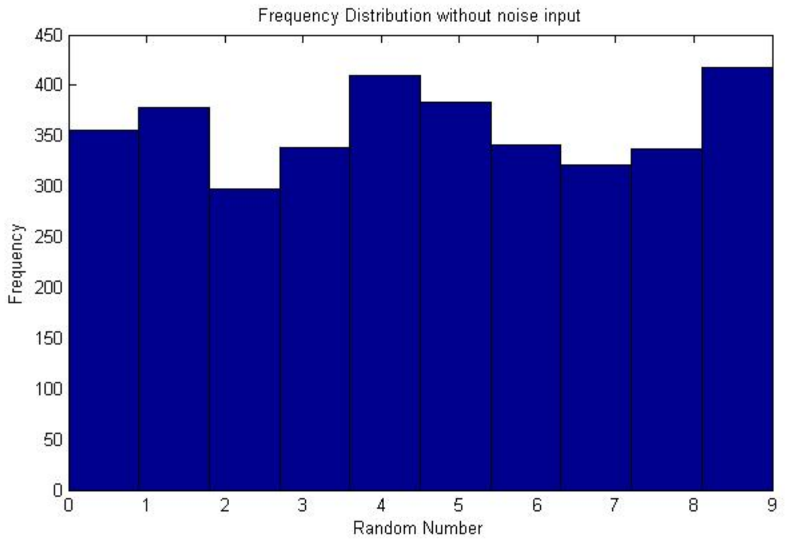


Figure 7: Frequency distribution of values without noise input

This is the time series for the same.

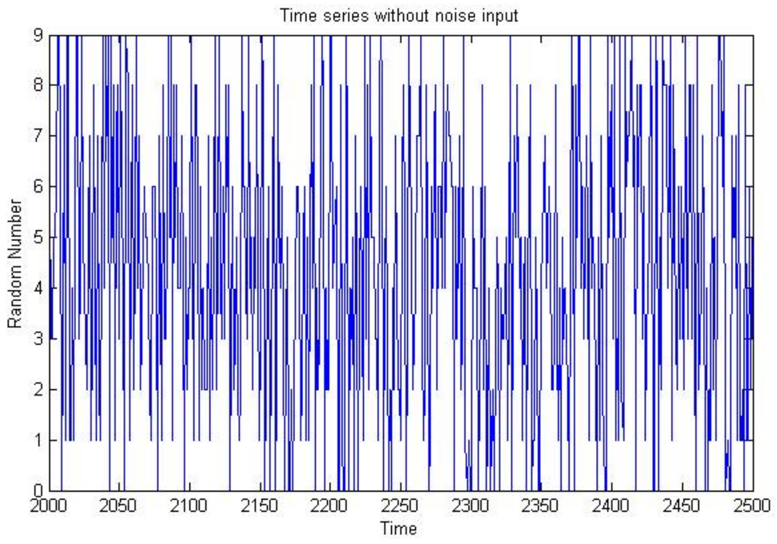


Figure 8: Time series of values without noise input

6 Work Distribution

The work was divided equally between the three members. The division of work was not clearly marked. The main components of the project were:

1. Analog Circuit design
2. Chaotic mapping using Arduino
3. Integration of systems

7 Conclusion

The aim of the project was to generate random numbers for cryptographic usage. We have been able to generate numbers on par with the quality of the random numbers generated by the random() function built in the arduino. The variance for our RNG was found to be 8.29 as compared to 8.13 obtained for the arduino RNG.
Given the need to generate truly random numbers, this experiment tries to approach the problem with a fresh outlook. This system shows how to produce Random numbers with a completely non-deterministic manner, in addition to being portable. The numbers can be used for anything, ranging from Gambling to Statistics.

8 Acknowledgement

We would like to thank Prof. Pradeep Sarin for helping us out and providing us with much required components. We would also like to thank Nitin Pawar sir for his valuable insights. We are grateful to Texas Instruments Inc. for providing us with free samples without which this project would not have been possible.(Who doesn't love free stuff!)