Hardware Random Number Generator Circuit

MTM Scientific, Inc. P.O. Box 522 Clinton, MI 49236

Introduction

The Hardware Random Number Generator (RNG) circuit is designed to create a series of completely random numbers. The numbers are generated from the random noise generated by a reverse-biased semiconductor junction in a transistor. The basic unit of output from the RNG is string of single random bits (e.g. either "1" or "0"). The single random bits can be combined to created any desired type of output, for example numbers, text strings, passwords or ASCII symbols.

The output from the RNG is serial data stream, which can be monitored and recorded by any device with a standard serial port. The output from the RNG is very random, but it is not particularily fast. This RNG design is presented for hobby purposes. We make no warranty about the suitability of the output for applications requiring perfect random numbers. Figure 1 shows the assembled random number generator circuit.

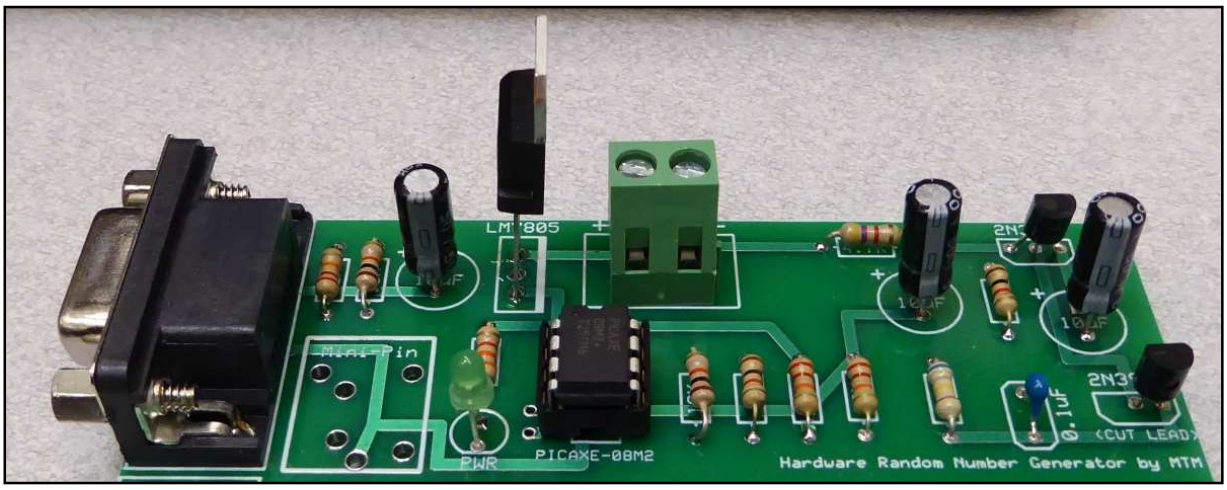

Figure 1. Assembled Hardware Random Number Generator

How the RNG works

Generating truly random numbers continuously turns out to be surprisingly difficult to do. Stated another way, generating numbers with absolutely no traces of a repeating pattern is very challenging. Computers turn out to be notoriously bad at generating random numbers. Which makes sense: Since a computer will only

perform the instructions it is given... exactly and precisely the same way every time. Hardware Random Number Generators (RNG) take a different approach to generating random numbers. The RNG begins with a physical process that is intrinisically random and uses the output of the random process to generate random numbers.

In this RNG design the random process we are using is the noise of reverse-biased semiconductor junction in a small signal transistor. The electrical signal from the transistor source is amplified by a second small signal transistor. The basic outline of this RNG design was created by hobbyist David Eathher about 10 years ago.

Producing the noise signal for our random number generator is not enough. The signal from the circuit is analog. We want a digital signal, and therefore we need to employ a conversion method. The conversion method is called an 'analog-to-digital converter'. Fortunately, this is rather easy to do with the advent of small digital microntrollers, such as the PICAXE-08M2. We perform a 10 bit conversion of the analog signal, but we only use one bit of the conversion for data. We use one of the least significant bits from the conversion, since the higher order bits will be less random.

The Picaxe IC chip is actually a small computer. The Picaxe can be easily programmed with instructions using a computer language known as 'Picaxe Basic'. If you are already familiar with the Basic language you will find it very easy to start programming with this version of the language.

Assembling the RNG Circuit

The RNG circuit must be assembled before it can be used. Assembly of the circuit requires electronic soldering. There is a tutorial available at the MTM website if you are not familiar with soldering. There are several parts that must be assembled with the correct orientation. The parts are the electrolytic capacitors, IC chip, voltage regulator, transistors and LED. Refer to Figure 2 for visual cues for correct placement and assembly of these specific parts.

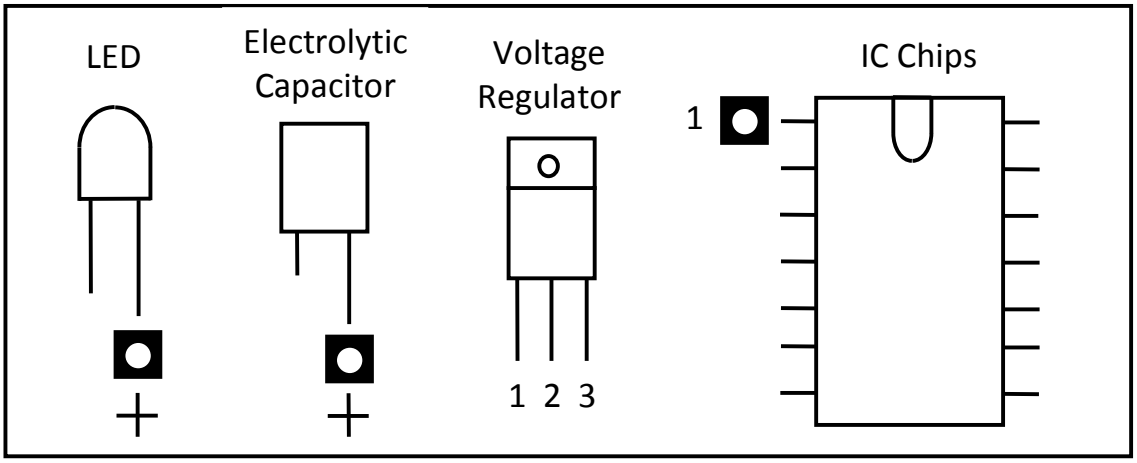

Figure 2. Look for visual clues on the components for proper installation.

Note that one wire lead is removed from the first 2N3904 transistor. That particular transistor is the actual source of the random electrical noise. The photograph in Figure 3 shows the transistor with the lead removed. Cut the lead close to the case, as shown.

Figure 3. Cut the lead on the 2N3904 transistor as shown.

The green LED included in the circuit is for indicating the +5VDC power supply is working. This is convenient when first connecting the power supply. This circuit is designed to use a standard +12VDC power supply input. It is important that the power supply output voltage is regulated. (This circuit was originally made for operation from a regular +9VDC transistor radio battery, but we found that the output become distinctly non-random as the voltage changed during discharge.)

The RNG circuit includes provision for a voltage divider. The voltage divider was included to provide a means for the Picaxe microcontroller to monitor the line voltage. Although it is not used here, you might find this helpful if you decide to perform a check of the RNG operating voltage sometime in the future. (e.g. Stop the output if the voltage level drops too low, for example.)

Programming the PICAXE-08M2

The Picaxe chip must be programmed with software code before it can be used. You can program the Picaxe after the circuit has been assembled. The same serial port used by the circuit to report the random numbers is also used to perform the programming. (Optionally you can use an USB port to communicate with the RNG with the AXE027 cable from Revolution Education. This is the purpose of the 'Mini-Pin' feature on the PCB: Normally empty. More information about the cable is here: http://www.picaxe.com/Software/Drivers/AXE027-USB-Cable-Driver/)

A special software application is required to program the Picaxe IC. The software is provided free by Revolution Education, LLC in the United Kingdom. This freeware program is called the 'Programming Editor' and is available at the Revolution Education website here: http://www.picaxe.com

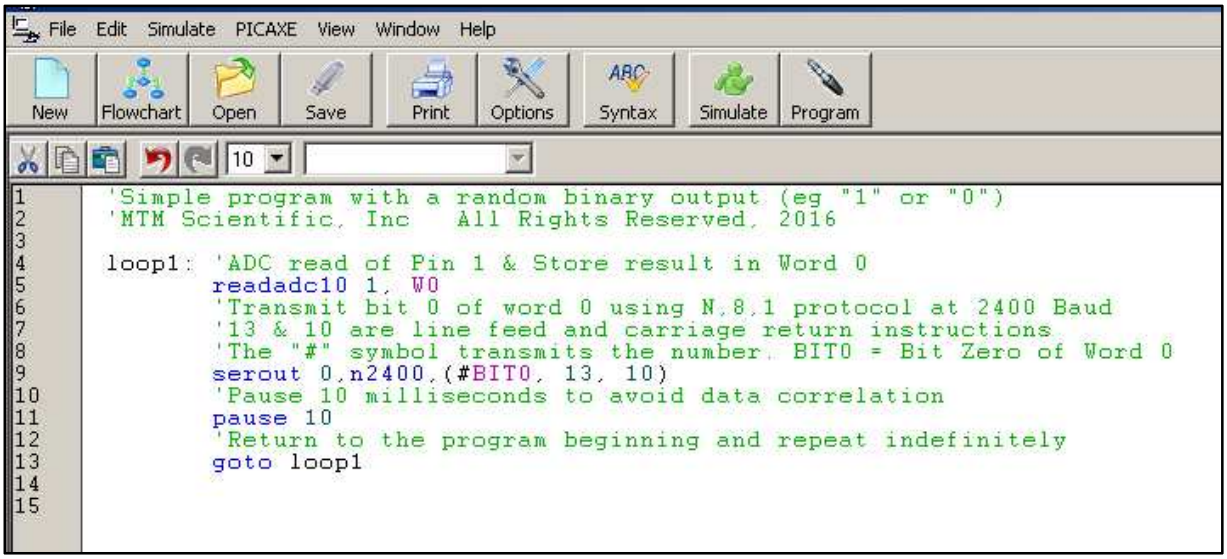

Figure 4. Screenshot of the Picaxe Programming Editor in action.

Collecting and Using the RNG Output

The Random Number Generator has standard serial port output. The output is intended for transmission to a computer for monitoring and subsequent use. It is

most convenient to use a computer which has a standard 9-pin serial input connector. The standard 9-pin connector is directly compatible with the connector on the RNG. There are also numerous free software programs available for monitoring serial ports and collecting the data stream in a file.

One serial port software program that works especially well with Windows is "Termite" from CompuPhase. This freeware program is available at the CompuPhase website here: http://www.compuphase.com A screenshot of the Termite program receiving serial port data from the RNG is shown in Figure 5. The communication settings are shown in the inset in the top left corner of the window.

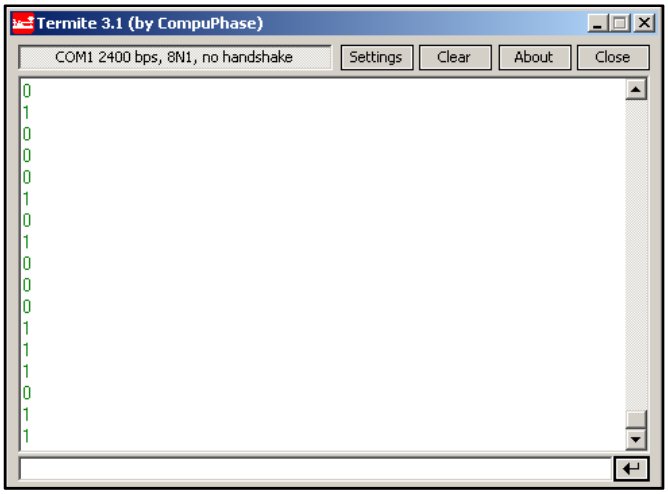

Figure 5. Screenshot of Termite serial port monitor in action.

Analyzing the Output

The output of the RNG is a string of digits: "1" or "0". You might be wondering if testing can be done to determine if the digits are truly random. One very simple test would be to collect a data set and examine if the "1" and "0" population is approximately evenly divided. This analysis was performed as shown in Figure 6. In this example the number of samples was 2335. We see they are almost exactly evenly split between "1" and "0".

All 10 bits of output from the ADC conversion are analyzed in the appendix. We see that many of the lower order bits are random. This suggests that additional bits might be used to increase the data flow output from the RNG. Another option might be to reduce the time delay between consecutive readings. The time delay was included to reduce the chances of correlation between adjacent ADC readings.

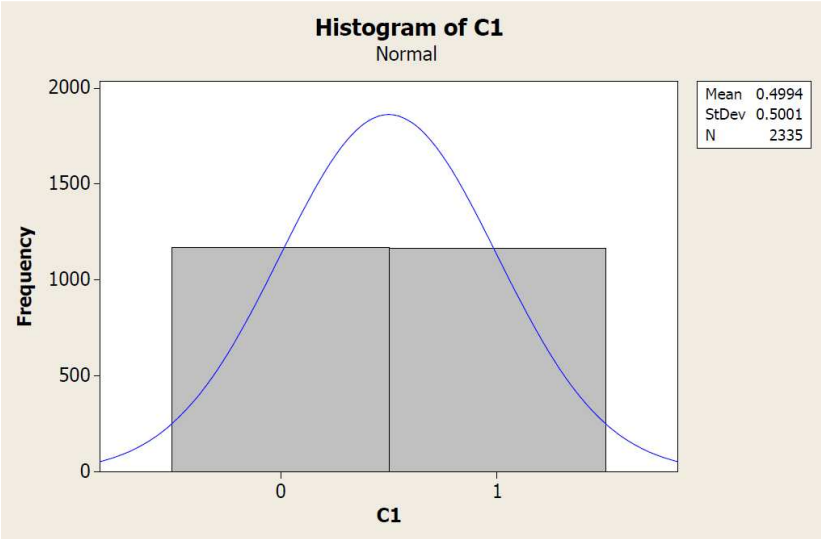

Figure 6. Population of "1" and "0" in a dataset from Bit 0.

Many more tests are available for analyzing the RNG output. If you have an interest in this topic there are many good resources available. A classic set of software tools for performing tests for randomness is "DIEHARD", by George Marsaglia. Although these tests are old, the basic principles they employ are classic and are worthy of study. You can find the Windows software for performing the tests here: http://stat.fsu.edu/pub/diehard/

There are also several methods for processing the random data stream to improve the statistics. One such method is called "WHITENING", as invented by John von Neumann. This method looks for rising and falling edges in the data to eliminate bias. This approach is very easy to implement in software by modifying the Picaxe Basic program, but has the disadvantage of reducing throughput.

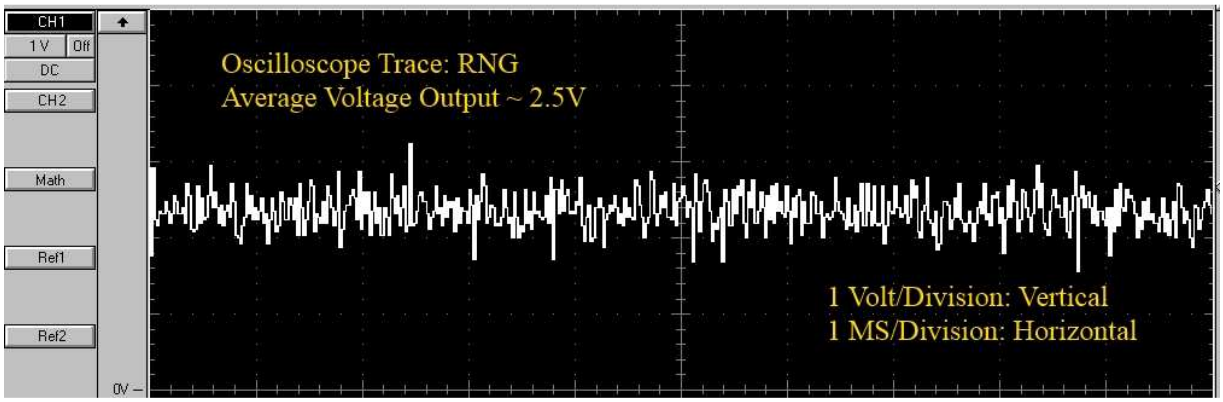

Figure 7. Oscilloscope trace of the RNG output to the Picaxe ADC.

Conditioning the Output

The data output from the RNG is a serial data stream consisting of the digits "1" and "0". You may have an interest in generating random numbers with a different range. This is actually quite easy to do and we will provide an example.

The basic output of the RNG is a bit. We can use each random bit to contruct a computer byte, consisting of 8 bits. A byte is capable of representing numbers in the range of "0" to "255". Therefore we only need to create a short routine to assemble 8 random bits into a random byte. This is easy to implement because Picaxe Basic is capable of working with individual bits. See the example program in the distribution folder called "byte.bas". Larger numbers can be created by combining individual bytes.

Appendix A: Source code listing for the PICAXE-08M2

```
'Simple program with a random binary output (eg "1" or "0")
'MTM Scientific, Inc   All Rights Reserved, 2016

loop1:    'ADC read of Pin 1 & Store result in Word 0
          readadc10 1, W0
          'Transmit bit 0 of word 0 using N,8,1 protocol at 2400 Baud
          '13 & 10 are line feed and carriage return instructions
          'The "#" symbol transmits the number. BIT0 = Bit Zero of Word 0
          serout 0,n2400,(#BIT0, 13, 10)
          'Pause 10 milliseconds to avoid data correlation
          pause 10
          'Return to the program beginning and repeat indefinitely
          goto loop1
```

Appendix B: Hardware Random Number Generator Parts List

-Transistor, 2N3904, Qty:2, Digi-Key, Part #2N3904FS-ND
-PICAXE-08M2, Qty: 1, Available at www.phanderson.com
-Printed Circuit Board, Fabricate at EXPRESSPCB.COM using included artwork file. Qty: 1
-LED, Green, T1, Qty: 1, Marlin P. Jones Associates, Part #15102-OP
-Voltage Regulator, LM7805, Qty: 1, Digi-Key, Part #LM7805CT-ND
-DB9 Connector, Female, Qty: 1, Digi-Key, Part #626-1561-ND
-Green 2 Pin Terminal Connector, Qty: 1, All Electronics, Part #TER-202
-Mini-Pin Connector, Qty: 1, Mouser, Part #161-379XS-E (This part is only needed if you plan to use the Picaxe AXE027 USB cable from Revolution Education, instead of a serial port.)
-Resistors, 680K, 100K, 22K (3), 10K (2), 4.7K, 3.3K, 1K (2)
-Capacitor, 10uF, Qty: 3, Jameco, Part #29891
-Capacitor, 0.1uF, Qty: 1, Digi-Key, Part #445-5258-ND
-DIP Socket, 8 Pin, Qty: 2, Jameco, Part #112206
-12VDC power supply, Qty: 1, This can be any plug style PS with a well regulated output.

Appendix C: Random Number Generator Bit Distributions

Shown in Figure 8 are the bit population profiles for all 10 bits from the A/D conversion in the Picaxe. We see the lower order bits have a nearly evenly split distribution, while the higher order bits tend to show a bias. These distributions are from the raw data. No software whitening has been performed on the data.

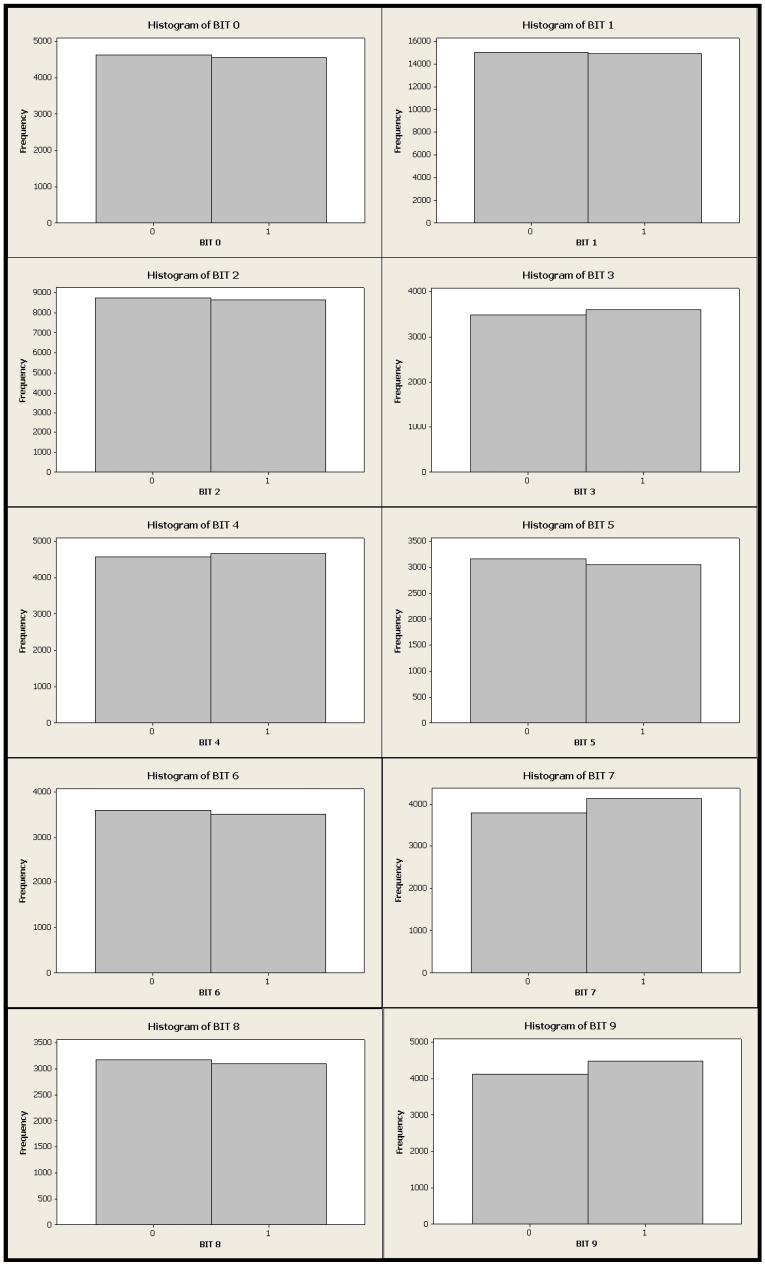

Figure 8. Histograms of the RNG output for Bits 0-9

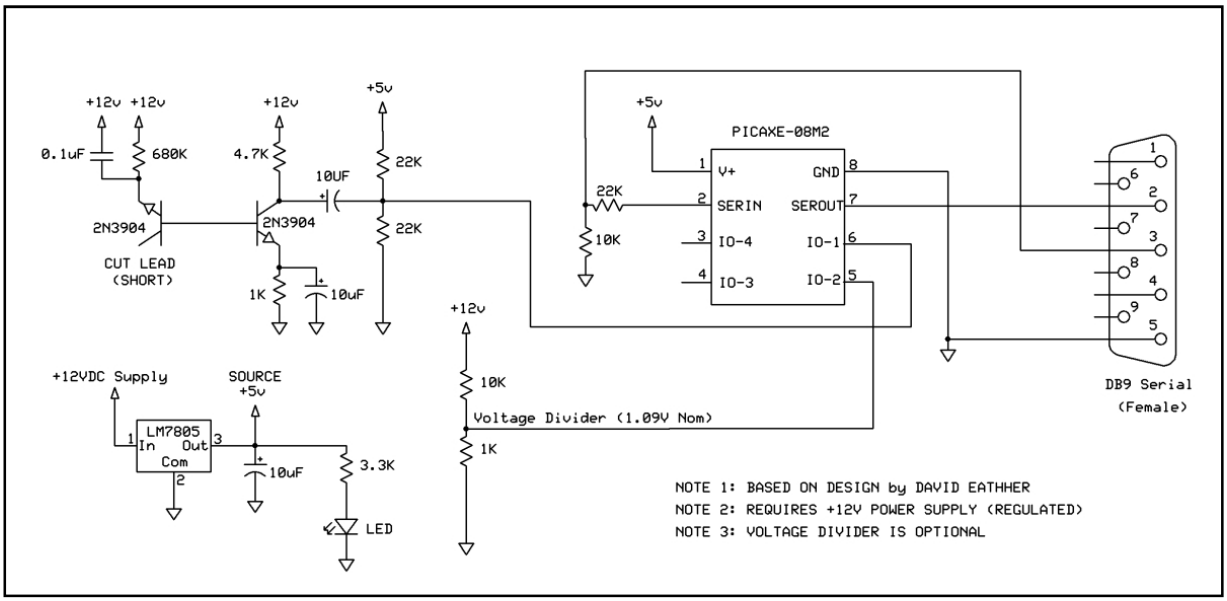

Figure 9. Circuit diagram of the Hardware Random Number Generator

There are many opportunities to improve this basic RNG design. One option would be to use more than 1 bit from the ADC conversion for data. Another option would be to experiment with reducing the time delay between readings to increase the throughput. Please let us know if you come up improvements so we can share the information with other experimenters!

Visit us on the web at: www.mtmscientific.com