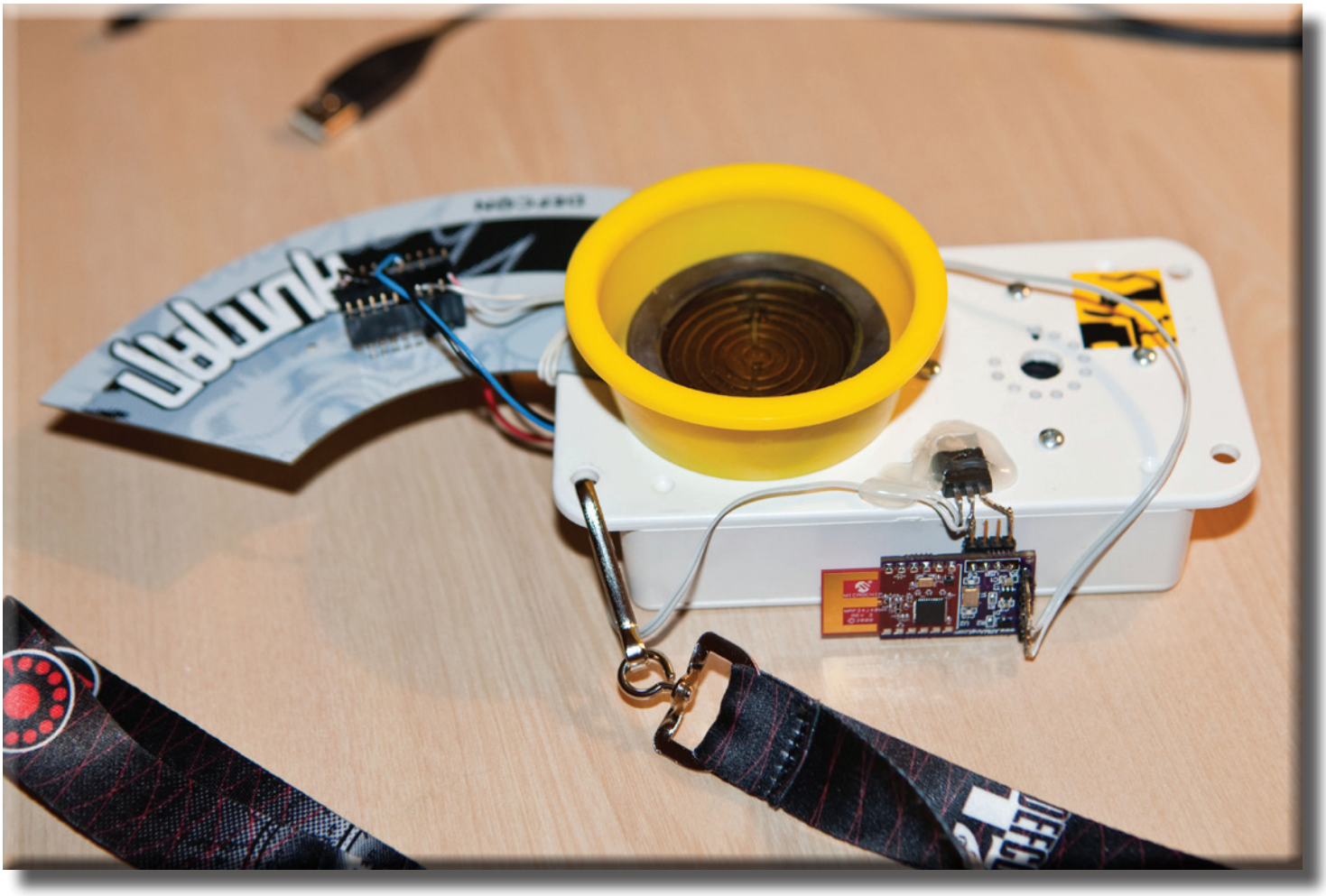
Badge Hacking by Optimized Tomfoolery

Skunkworks, Abraxas3d,
and Firmwarez



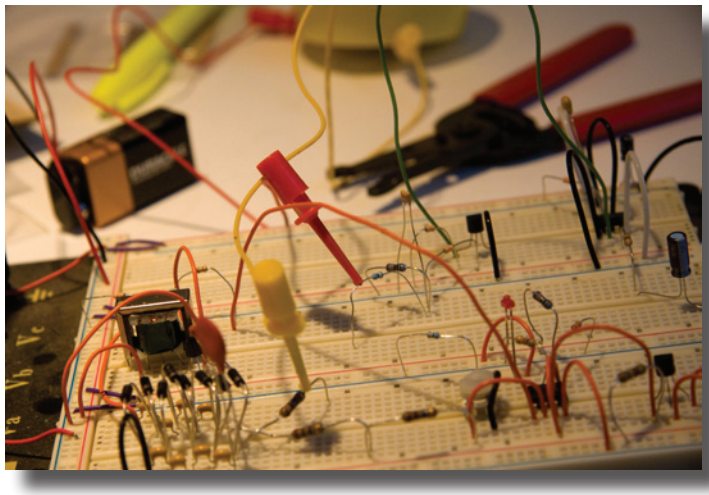
A Geiger-based Random Number Generator with Wireless Link

A narration of events
by Abraxas3d

Geiger Müller Tube Power Supply Construction

On a leisurely tour of the internet one day, a site[1] describing a power supply for "pancake" Geiger Müller tubes was discovered. Since the author of this thoroughly enjoyable site (Alan Yates) had found the pancake Geiger Müller tubes on eBay, I decided to look there too. I found the same type mentioned in the article, and won an auction on 6 July 2009 for a pair from Ural (in the Russian Federation). I then began thinking about what to do with them.

When we (Optimized Tomfoolery[2]) learned more about the badge hacking contest at Defcon[3], it was proposed that we should build up two Geiger counters and bring them in case they could be interfaced to the badge to do something useful, entertaining, or (gasp) obscene. The badge hacking contest challenges all attendees to take their processor-laden badge and do something extraordinary with it. Some helpful information was posted on the Defcon forums on 19 July 2009.



We began to consider a random number generator, as the radiation sources measured by the Geiger counter would be an excellent source of physical randomness. Since random number generation is a central topic in cryptography and information security, we thought it would be an interesting demonstration for a contest at a computer security conference.

1 http://vk2zay.net/article/225

2 http://www.optimizedtomfoolery.com

3 https://forum.defcon.org/showthread.php?t=10655

The Geiger tube package arrived on 21 July 2009.

Several power supply circuits were considered with this one from Galactic Electronics[4] forming the baseline circuit. Electronic parts were ordered from Mouser on 22 July 2009. They shipped on the 23rd, and arrived on the 24th. In the meantime, a trip to Industrial Liquidators in Kearny Mesa for mechanical housings and other necessities was made.

We found a white plastic box with fitted lid for a main housing. The box had four integral standoffs, four screw holes for the lid, and a flange with mounting holes and a place for the egress of wires.

Another find at Industrial Liquidators was a cupcake-shaped yellow plastic bowl with a flat bottom that looked like it might perfectly hold the Geiger tubes.

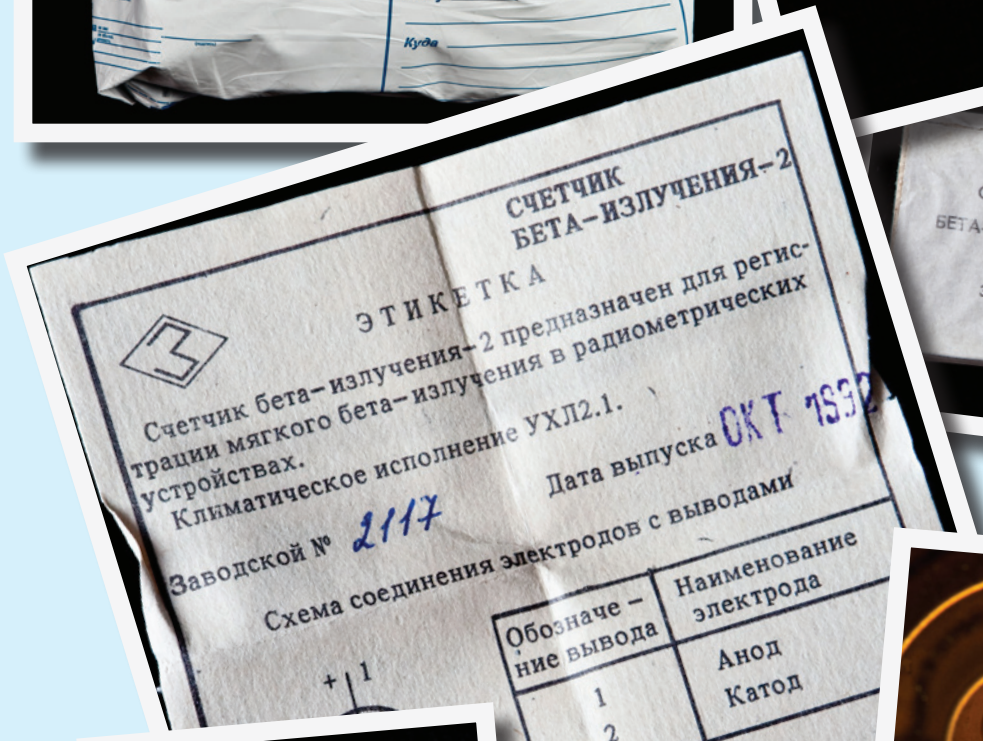
I decided to breadboard the power supply circuit before constructing it on perfboard. One of the first things we noticed from the data sheet included with the Geiger tube (one was in Russian and the other in English) was that the tube required 400 volts instead of the 500 volts in the reference circuit. A resistor value change along with being able to vary the voltage by changing the 50kΩ potentiometer enabled the setting of a range of supply voltages centered around 400 volts.

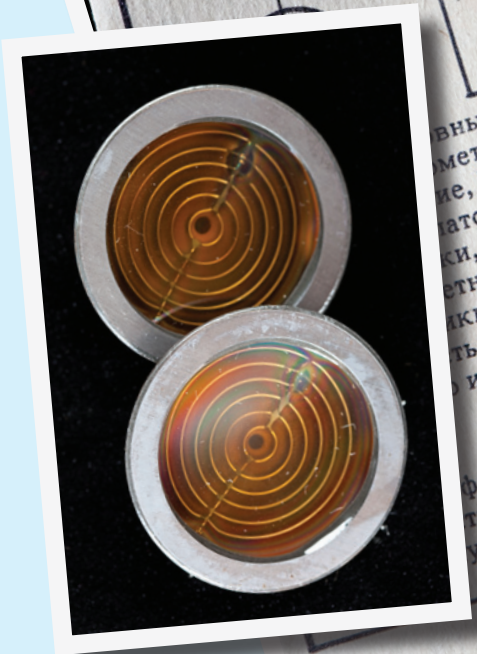
continued on page 4

continued from page 2

4 http://www.galacticelectronics.com/Geiger-Counter.HTML

Geiger Müller Tube Unboxing

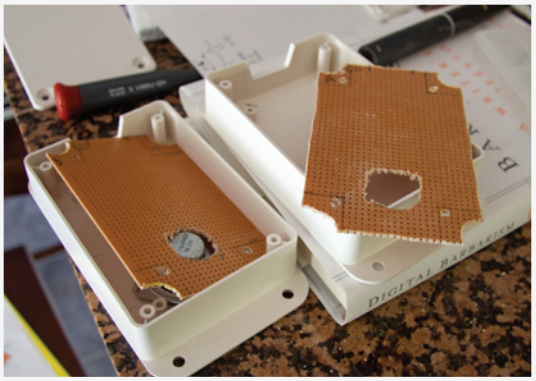




The digital multimeter had an input impedance of 10MΩ which formed a voltage divider with the 10MΩ series resistor in the power supply, which resulted in seeing almost exactly half of the expected voltage. Understanding the characteristics of your test equipment and how those characteristics affect measurement is part of the process of design, build, and test.

A few bugs with the circuit had to be worked out during the breadboard stage. First, we noticed that the low battery light never came on. The root cause turned out to be that I had accidentally ordered the wrong voltage reference. The model circuit used a 1.2 volt reference but I had ordered a 2.5 volt reference. The original circuit designer had selected the 1.2 volt reference because he'd had a bunch on hand. Fixing a mistake in an arbitrary circuit decision involved a bit of irony. A voltage divider swiftly solved the problem. Second, since the voltage reference also affected the oscillator, which affected the transformer, which affected the supply voltage (increasing it to above 600 volts even with the potentiometer at full scale), some reconfiguring had to be done. A larger fixed bias resistor in series with the 50kΩ potentiometer was used. This lowered the supply voltage back to where the range was centered on 400 volts, instead of exceeding the maximum 600 volts that the multimeter could measure.

The Geiger tube was inserted into the circuit, and the detector worked! Meanwhile, a thoriated welding rod (2%)



arrived in the mail (shipped in compliance with the rules of USPS Publication 52 and 49 CFR 173.424. This item falls under the exempt limits as a thorium welding rod as listed in 10 CFR 40.13: "(1) Any quantities of thorium contained in ... (iii) welding rods").

This radioactive source provided appproximately twice as many particle counts as

background radiation alone – enough to be detectable without being too alarming.
The next step was to create the right shape of perfboard. It had to sit down on the standoffs and it had to have the corners cut out in order to fit between the places where the screw holes for the lid were located. The perfboard was shaped with diagonal cutters and a Dremel press.

After the perfboard was properly shaped and test fit, the circuit was transferred to it from the breadboard.

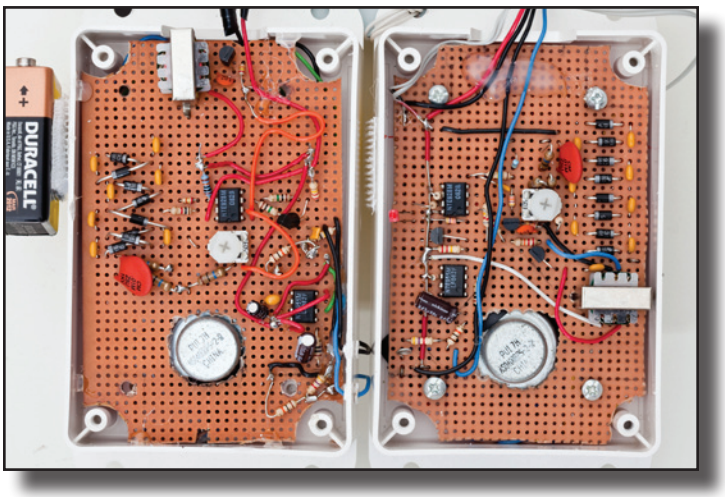
The speaker location was decided, four holes for mounting screws were drilled, and the speaker installed. Instead of concentric rings of holes for the speaker grill, I used designs from the Defcon logo, which can be found on the cover of this document. One of the boxes got a telephone-rotary-dial-shaped speaker grill,

and the other got a happy-face-skull-and-crossbones speaker grill. The Dremel press came in very handy here.

Next, the position for the Geiger tube was decided. Both the speaker and the tube would go on the same (outboard) side, and the badge would be secured to the top or side, depending on the size and shape. The entire contraption could be worn around the neck.

The kitchen table was used to breadboard, the countertop immediately to the right of the stove was used as a Dremel press area, and perfboard construction was done upstairs at the soldering stations in the lab.

Below, one can compare the construction styles of Abraxas3d (left) and Skunkworks (right). This is one side of the completed 400 volt power supply board.



We loaded the software development environment for the badge processor on a couple of laptops and coordinated with a third team member, Firmwarez, from points east, who would be joining us Friday evening. We asked Firmwarez to bring voltage level converters in order to be able to interface the laptops to the badge, since the two systems would have incompatible voltage levels.

We packed for Defcon 17 and drove from San Diego to Las Vegas on Thursday 30 July 2009. We registered and got "permanent" (with the hackable processor onboard) badges. Skunkworks received his Human badge in short order by standing in what seemed to be really long line that moved surprisingly quickly. After some back and forth, a bit of waiting, the employment of a press agreement form as a temporary badge (using a Little Rock Central lanyard) and after producing a particular email (on Skunkworks' unpatched iPhone), I received a permanent Press badge.

The badges each came with a CD containing all sorts of conference files, a sheet of stickers, a CR2032 battery, a lanyard with hacker graphics on it, and a printed program.

I used the temporary paper badge to attend the "Hacking With GNURadio" talk by Videoman, which was one of the talks I wanted to cover as a member of the media. We also listened to "Hacking the Apple TV and Where your Forensic Data Lives" and then "Con Kung-Fu: Defending Yourself @ Defcon". Since the rumor was spreading that they were going to run out of "real" (hard, vs. temporary) badges, I decided to run and get Firmwarez' badge. They were already out, so I got him a plastic temp badge. This would put us one badge down in terms of target hardware, but we figured we could get by with two instead of three. I resolved to try and swap it out in the morning.

We claimed our hotel room and unloaded the truck.

Below, Abraxas3d takes an obligatory MySpace style mirror photo.

Setting Up Shop in a Vegas Hotel Room

We set up the round table near the window as both hardware and laptop station, which was pretty crowded. I made a quick attempt to social engineer another table, but I succeeded only in confusing Maintenance.

The bathroom was proto lab, with the Dremel press stationed to the right of the sink,

and the Dremel tool kit to the left.

The hot glue gun was in here as well, but ended up better stationed right outside the bathroom door on what appeared to be intended as a tiny writing desk/shelf.

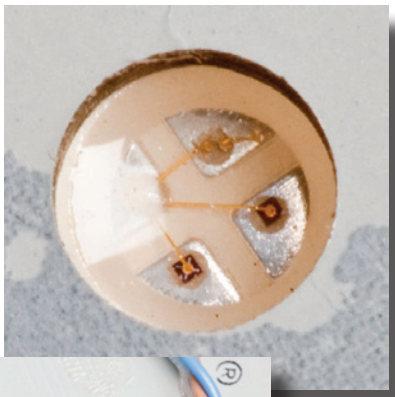
We unplugged the TV almost immediately to plug in the outlet strip. The TV could not easily be removed from the cool pull-out shelf, so I put the adjustable power supply on top of the TV.

Above the TV there was a space to store and sort small items.

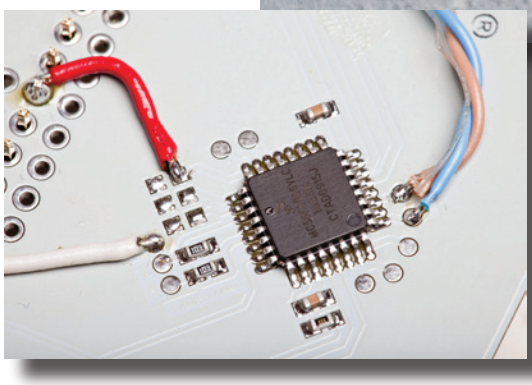
What looked like a cabinet, which housed a refridgerator that we didn't figure out was turned off until we were about to leave, became the oscilloscope, wire, and components workstation. It also held the Merlot and the Bourbon.

The transistor in the detector part of the circuit of one of the two Geiger counters had failed, so I performed some surgery to replace it. The speaker wire also broke, which was a quick repair once it was noticed.

We examined the badges closely and reviewed the schematic, parts list, and other data from the Defcon 17 CD. We determined that the badge had a microphone, a red-green-blue LED (pictured to the right), a Freescale MC56F8006 digital signal controller (pictured below), and... well, that was it. Our initial impression was that the I/O was sparse.



We saw that the processor had three GPIO pins brought out to jumper positions, which on the Human badge were all unpopulated. The Press badge had a zero-ohm resistor installed in the center position. We suspected that these were used to identify the type of badge to the processor.



We saw that there was access to I^2C. There were also pogo pads for JTAG (pins TDI, TDO, TMS, TCK and GND were brought out to test points).

Thursday evening we ate at Kristofer's Steak House at the hotel. It was delicious.

We discovered that the source code to the (unhacked) badge application was on the Defcon CD. We pored over the source code, and determined the various modes of the badge. "Normal" mode flashed the LEDs in proportion to

the amount of noise detected by the microphone. If the badge was exposed to a level of noise (for example, at the Black and White Ball) over a pre-determined threshold, then the LEDs would become more active in "Dance" mode. "Morse Code" mode was triggered by a certain frequency bin being significantly higher in value than others. We weren't able to trigger this mode, but we were able to decipher the morse code message from the source code. We visited the secret webpage[5]. The web page had only a plain text placeholder, which hinted broadly that more information would be posted there on Friday.

We were a bit confused about the way the bins were supposed to behave.

Friday morning, after buffet breakfast, we attended "Welcome to Defcon 17 & the Making and the Hacking of the Badge" talk at 10:00am, where all of the things we'd spent Thursday evening digging up were explained in detail to the

5 http://www.0day.com/n01z/

hundreds of people in the enormous conference room.

The Hardware Hacking Village was high on our list of things to check out. It had opened at 10:00am. We weren't sure if we wanted to work there, or in the hotel room. I was leaning towards the hotel room mainly due to having increased security. We would be able to get up and leave our temporary lab, lock the door, and get something to eat or attend a talk without too much concern over whether or not someone would walk off with a part or tool. Not that anyone would do such a thing at Defcon.

Another reason was that the hotel room was a controlled enviroment, sound-wise. If the Hardware Hacking Village was as popular as the Defcon forums suggested, then it might be crowded, and it might be loud. Depending on the stage of the project, that could either be enjoyable or fatiguing.

We explored the floor, the Capture The Flag contest room, the lockpick village, the vendor area, the PDP-11 room (with quadraphonic stereo playing Pink Floyd's The Wall), some sort of game contest type thing going on, an EFF booth, and the NIST Quantum Crypto Lounge, where we found our first random number generator fellow traveler, some excellent periodic table handouts, and very groovy quantum cryptography exhibits and what appeared to be quantum-related scientific apparatus.

Michael Wayne, a student at University of Illinois at Urbana-Champaign, had a real-live-honest-to-goodness poster session from his paper "Photon Arrival Time Quantum Random Number Generation", which was published in the Journal of Modern Optics in January 2009.
His paper focused on a new method for getting random numbers from quantum events. The "old" method is to force photons (in his case) to take one of two paths. One path is 0 and the other path is 1. The "new" method is to measure time between the events. The "waiting time" distribution for a Poisson process is a decaying exponential. The interval between photons is separated into individual time bins which are then used to create several random bits per detection event. Next, you hash to whiten these results, and you get an entropy of about one random bit per bit of time measurement. The hardware required is a single photon counter and a lot of data processing. The results were 130Mbps random data out. Mr. Wayne was scheduled to present the paper for the first time the following weekend, and we wished him luck and hope it went very well.

This gave us a better theoretical framework for making a random number generator than we had before, so we took notes, talked to Mr. Wayne, and attempted to look up the paper. The journal wanted to charge $37 to download the paper.

Between the price, and the fact that he hadn't published any code, and hadn't used any open source libraries, we decided that we had probably learned as much as we could from the paper under the time constraints of the contest, and continued to explore Defcon.

(Update: Skunkworks found what looks to be the same paper online here[6] on 8 August 2009)

Since Kingpin had cautioned the Hacking the Badge talk attendees about the current draw from the badge being high enough to where a CR2032 might not last all weekend, we decided

6 http://research.physics.illinois.edu/QI/photonics/theses/Wayne-thesis.pdf

to make a battery run to CVS pharmacy across the street from The Riviera hotel. We were able to procure extra batteries for development, along with Atkins-style snack bars, which turned out to be Gut Bombs.

In between lunch and wandering around and dinner, we got the development enviroment up and running, got Firmwarez a permanent badge (we just happened to stumble past registration right after they found one last small box of hard badges) and found a way to interface to the board.

The badge test video showed a header anchored to the decorative holes in the badge that formed the initial of the badge type. Human badges had an "H" and press badges had a "P". We knew we needed a 3V level translator for the serial port, and Firmwarez was bringing it. However, during a trip to the swag booth, we noticed something for sale called a Hardware Hacking Kit. We bought it, and it turned out to contain not only the same sort of headers from the test video, but also a USB serial port ready to hook directly up to the 3V logic signals of the badge via a 4-pin header. Specifically, the serial port was a "Prop Plug" intended to connect to the Propeller microcontroller in the Hardware Hacking Kit.

The item can be found at
http://www.parallax.com

The Prop Plug connected to the asynchronous serial port pins, which is where the provided bootloader expected to get its downloads and

where the provided console output routines sent their output.

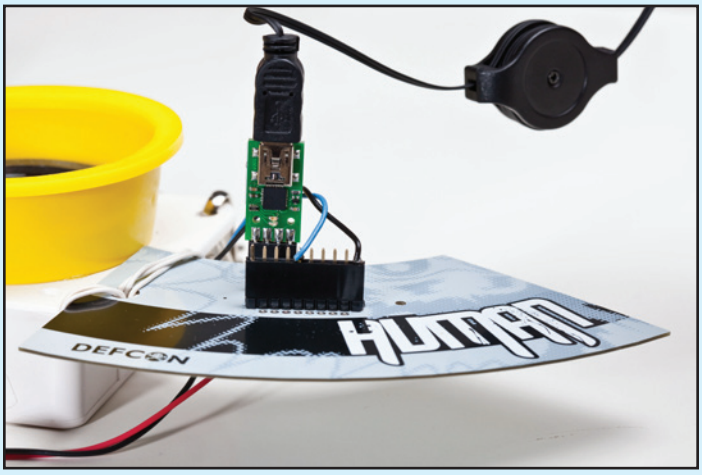
Pictured below is the Prop Plug connected to the badge and receiving serial data. (The LED would be blue if it were sending data.)

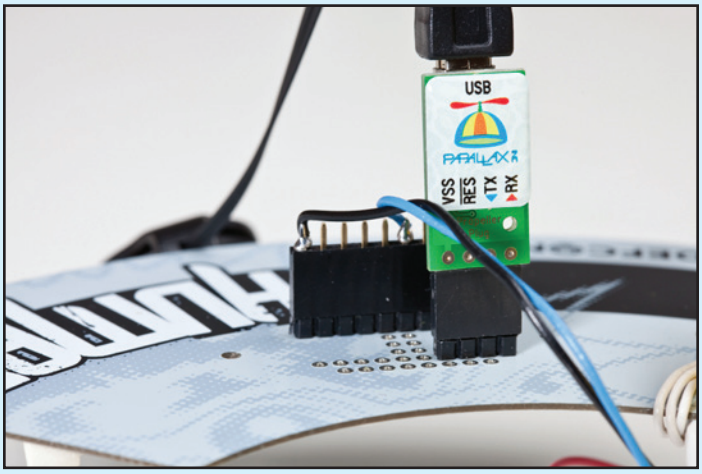
Firmwarez arrived to much fanfare, and we resumed our evaluation of the board. We revisited our decision to use the Geiger counters considering the fact that the badge input and output was not the best match for a counter design. We wanted to leverage as much of the badge as possible, so we talked about what other things we could do for the contest.

Since it had a microphone, and a digital signal controller, and an LED, we discussed a voice stress analyzer, following the dubious research on very low frequency components that allegedly exist in human voice when telling an untruth. Perhaps we could modify the press badge into signalling when the interviewee was telling the truth or not.

In order to do that, the microphone would have to respond to very low frequencies, 12-20Hz. The data sheet for the microphone showed the beginning of a frequency rolloff above this range. While we thought the microphone might detect those frequencies, we weren't able to produce them with the equipment that we had on hand, which included Audacity, laptops, and an assortment of iPods. Since we couldn't reliably characterize the microphone, we resumed brainstorming alternative designs for the contest.

Pictured below are two detail photographs of the serial data connection with the badge.





Seen above, the little red and blue arrows on the RX and TX lines made it easy to hook up correctly the first time.

Skunkworks didn't want to trivialize the badge, in the sense that he wanted the stock hardware to be fully utilized and not overshadowed by multiple external circuits. I didn't feel the same way. Firmwarez was ready to go along with whatever idea reigned supreme. The brainstorming session continued without additional major developments, and I became a bit grumpy.

Firmwarez had been able to procure a nifty custom Zigbee transceiver for use on the badge. He started working on getting the Zigbee up and running in order to provide a wireless link to a nearby PC for any data that our badge project might produce. Not only did he bring the Zigbee transceiver, but he also brought a very small high-contrast display on an evaluation board that he'd been able to procure from his adventures at Microchip Masters earlier in the week. Although we didn't end up using the display, having it as an available option for output broadened the possibilities.

After some more conversation, it was tentatively decided to continue with the Geiger-based random number generator.

Having been brought up to speed, and having flown in directly from Microchip Masters in Phoenix, Firmwarez promptly fell asleep.

The next morning, we got up at the break of dawn, had buffet breakfast, moved from tentative back to definite on using the Geiger counters, confronted our human frailty, then got to work on the badge.

By mid-morning, we were able to connect to the badge using the Codewarrior Integrated Devel-

opment Environment.

We began to think about how to hash the times in order to create random numbers. I couldn't find anything that looked like a hash function in the cryptography libraries in Codewarrior, so I went down the Hardware Hacking Village and asked the engineer from Freescale if it was in the library. After a few minutes of typing and poking and peering, he said that there wasn't a built-in hash function for the device, but that there was an autocorrelation function. Autocorrelation is a statistical test that determines whether a random number generator is producing independent random numbers in a sequence. This would let us test our output, but it wouldn't let us turn timer values into random numbers. However, since I now knew where the advanced mathematics functions were, I was reasonably sure I could start putting together an algorithm that would operate as a hash function.

Skunkworks searched for alternative methods, and found a much simpler method at a site called Cipher Goth that produced random bits by comparing successive intervals of time between Geiger clicks. We read up on it, and kept it on the front burner. The essentials of the method and its advantages are fully described here[7].

Below, Skunkworks successfully communicates with the badge.

One of the caveats of downloading new firmware to the badge is that you must be careful not to overwrite the pointer that points to the boot-

7 http://www.ciphergoth.org/crypto/unbiasing/

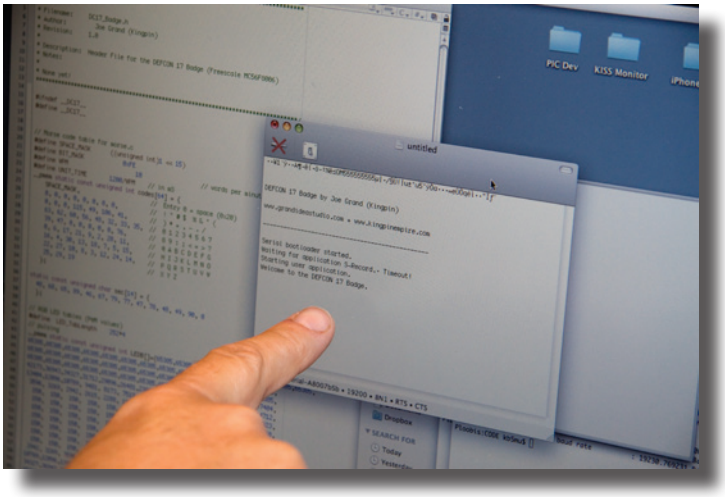
Below and right, Firmwarez works on the Zigbee-based transmitter firmware under the auspices of Kalishnikitty. The AR-15 banner above the EFF booth provided a harmonious counterpoint.

loader. We were able to rely upon the existing arrangement of the project provided on the Defcon CD, that had all the files necessary to recreate an IDE environment with the correct pointer information.

At 12:30pm, we took a break to go listen to "Design and Implementation of a Quantum True Random Number Generator" by Sean Boyce.

This lecture strongly reinforced the idea of comparing the intervals between received particles that we'd researched earlier in the day. Although Sean didn't have a demonstration (it had experienced mechanical difficulties in the journey to Defcon) he did cover almost all the major points of what we would need to know in order to demonstrate a Geiger-based random number generator. We felt we know knew everything that we needed to know to make the implementation work.

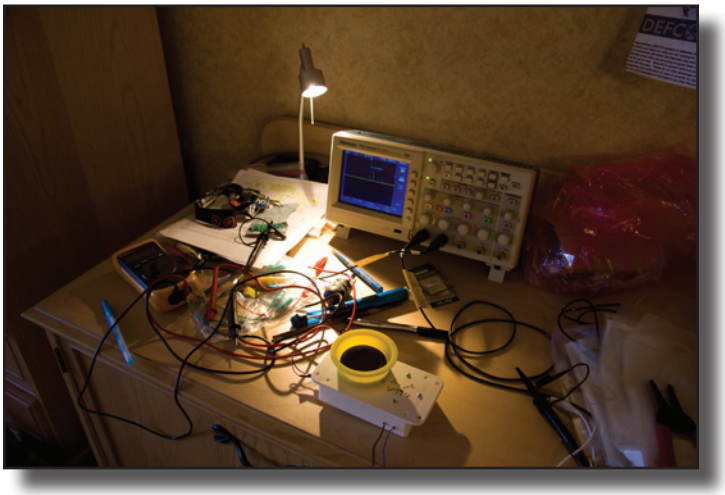
We ate at Circus Circus for dinner, and then got some sleep. Sunday morning, we had Champagne Brunch at the Buffet, and then got back to work.



I made the last few mechanical adjustments and modifications in the bathroom proto lab while Firmwarez and Skunkworks integrated the transmitter into the badge.

The output of the Geiger counter would be taken from the speaker wires. The speaker would continue to output clicks when the Geiger tube fired, but the signal to the speakers would be tapped in order to provide the input to the digital signal controller on the badge.

We used the oscilloscope to characterize the output pulse after the signal conditioning circuitry.



When we connected the Geiger counter up to the oscilloscope, the pulses registered as 80 volts in magnitude. After staring at the display for a few long minutes, we realized that the 10x probes were improperly configured in the menus of the oscilloscope. Remember that part about understanding the characteristics of your test equipment and how those characteristics affect measurement is part of the process of design, build, and test?

The post-menu-adjustment 0-to-8 volt pulse was a consistent 2.5ms in width. We discussed the repercussions. First, the 8 volts would have to be lowered. We decided on 2.5, and calculated a voltage divider circuit that would accomplish that. Second, we talked about what if any loss of randomness would occur if a second pulse was received before the Geiger tube itself was ready, or before the 555 timer had reset. Geiger tubes need, in general, about 1ms to reset. Since we were already in the order of magnitude, we decided this was already optimized.

However, several adjustments to the tomfoolery of the transmitter firmware had to be made, including changing timing, data rates, and removing some unnecessary code.

Gradually, with Firmwarez writing the PIC-based code for the transmitter, and Skunkworks supporting the integration while developing the

interval-compare algorithm for the received Geiger-sourced clicks, the system started to come together.

The 9-volt battery that powered the Geiger counter was attached to the Geiger enclosure by velcro, hot-glued to the side. The battery was expected to last 17 hours, but we swapped out the test battery for a fresh battery well before even the halfway mark, just to be on the safe side.

During a tour of the Hardware Hacking Village, Firmwarez called and made the recommendation that we should go ahead and demonstrate the board earlier rather than at the 2:00pm deadline. People were already checking in with Kingpin, the contest coordinator, and it was going to wrap up at 2:00pm, not start at 2:00pm. We agreed, and began final button-up procedures for the project.

Project Demonstration

Skunkworks donned the Defcon lanyard, which supported the white plastic Geiger enclosure. Firmwarez donned his dark sunglasses and armed himself with a light source. He would carry the PC that would show the transmitted data in graphical form. Whenever the system produced a random digit, it would be transmitted to the PC as either a period or an asterisk. We allowed the system to run long enough to build up a good picture of the generated bits. It looked a little bit like something from the Matrix.

The Geiger enclosure supported the badge, its own 9 volt battery, and the Zigbee transceiver (transmit only for this demonstration). The thoriated welding rod was enclosed in a hard plastic ID case and worn on a second lanyard (emblazoned with Little Rock Central, and recently relieved of its duty to hold the folded-up press application from Thursday). I put on my camera.

Having declared ourselves finished, we started to make our way to the Hardware Hacking Village. In our path, after leaving the elevator, was a frightening sight. There was a blimp, with attached badges! We determined that they were bound for the demonstration as well, and since anything involving flight was a real contender, we endeavored to get in front of them to be absolutely sure of a place in line. Since they had to move relatively slowly through the crowd, it was easy enough to quicken our pace and put some real distance between them and us. We discussed strategy and tactics for the demo, and finally arrived at the Hardware Hacking Village, with all parts of the project intact.

We figured out where the line was, and waited. However, the blimp arrived, and promptly cut in line! After their rollicking demo, we waited some more, while Kingpin's full SD card on his camera was downloaded. Then we waited some more as the projects in line ahead of us were checked in and examined. All of the projects were fun. Some, like the blimp (whose mission in life was to fly towards loud noise), were quite memorable and had obviously taken some time and skill to make.

Skunkworks described this part of the process as the "fun part", but it made me so nervous that I decided to get a tattoo. No, really. I did.

Finally it was our turn. We demonstrated our project to Kingpin. Everything worked perfectly. We opened up the enclosure to show the power supply, and waved the thoriated welding rod in front of the Geiger counter to good effect.

A reporter took some photos and asked us a few questions. We explained the project to him as well.

Here's Skunkworks (and Firmwarez behind him) right after demonstrating the project to Kingpin (seated at the table) and just before giving an interview to reporter Dave Bullock, who included us in his article[8].
After many hours of work and a successful demonstration, we made our way straight over to the NIST Quantum Lounge, in order to demonstrate it to Michael Wayne. Unfortunately, the Lounge was already shut down!

Undeterred, we wore the system around while shopping in the vendor area and visiting at the geocache contest booth. Everywhere we went, we got questions, comments, and photo requests. We enjoyed every minute of it, and answered a lot of questions and hopefully sparked more than a bit of interest in the badge competition.

There was more than enough time to get that tattoo, so I dropped by 3 Lions Tattoo to see if they'd be able to take me as a walk-in. The place was empty, except for a woman behind the counter, who said they were open until something on the order of 2:00am and the schedule was wide open. After a brief pit stop back at the room to (carefully) write down my mitochondrial DNA results, we went to 3 Lions and presented the tattoo request.

Contest results would be announced at 5:00pm, and it was already nearly 3:00pm. Preparation, font selection, and text sizing went quickly, and the work began. Firmwarez headed out for the results and closing talk around 5:00pm, and Skunkworks stayed to document the procedure on his camera. The artist, Nick Jones, did a great job, and was interesting to talk to. The entire experience left a smile on my face and my Cambridge Reference Sequence[9] on my arm, right above an existing blood type tattoo (AB+).

8 http://www.wired.com/threatlevel/2009/08/hacking-the-defcon-17-badges/

9 http://en.wikipedia.org/wiki/Cambridge_Reference_Sequence

We walked over to the enormous conference room, promptly despairing of ever finding Firmwarez. Luckily, we picked the right wall to stand by, because he was sitting 10 feet away from where we stopped.

Announcements, speeches, and contest winners were announced. We laughed, we cried, we cheered. It became part of us.

Finally, the badge hacking winners were announced. Who won? Are you on the edge of your seat? Was it Optimized Tomfoolery? Was it the blimp?

Neither.

The winning entry was truly amazing. It was remarkable. It... blinked LEDs.

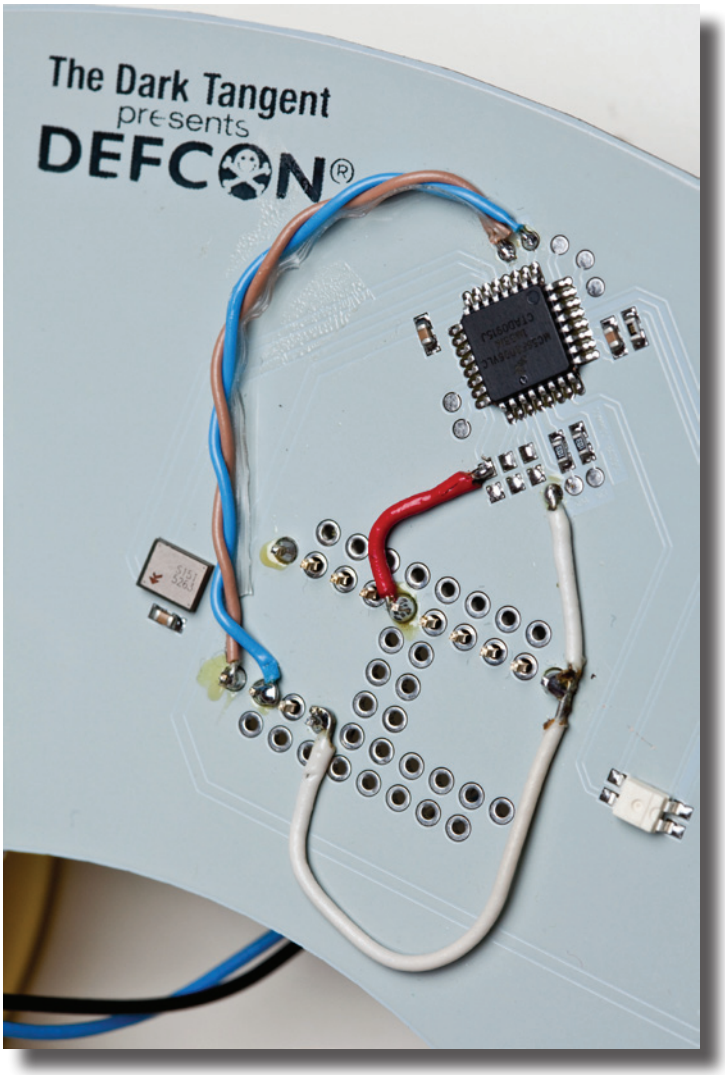
Really. It did. Lots of them. Enough to fill up the underside of the bill of a baseball cap.

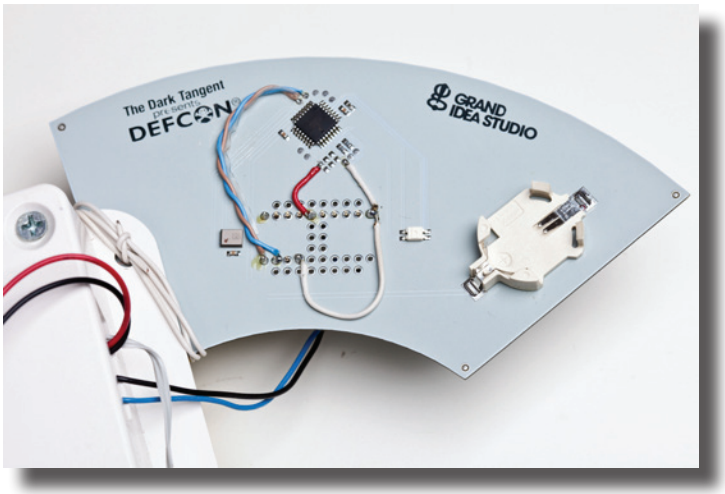
The story that went along with the project was that it would use red LEDs to defeat face recognition systems, putatively protecting Kingpin's cache of beautiful black badges. These black badges were what we were all after, since first place winners of many of the Defcon contests, including the badge hacking contest, win them. The black badge grants lifetime entry into Defcon.

Anyway, a blue box badge project was third place. Second place went to the blimp, which gave a quick demo in the hall. First place was Blinken Lights.

Following the contest winner announcements was a very interesting debriefing about the Defcon network that included descriptions of the setup, usage, and notable network statistics, we dragged ourselves back to Kristofer's Steak House and forced ourselves to eat like adults, including dessert.

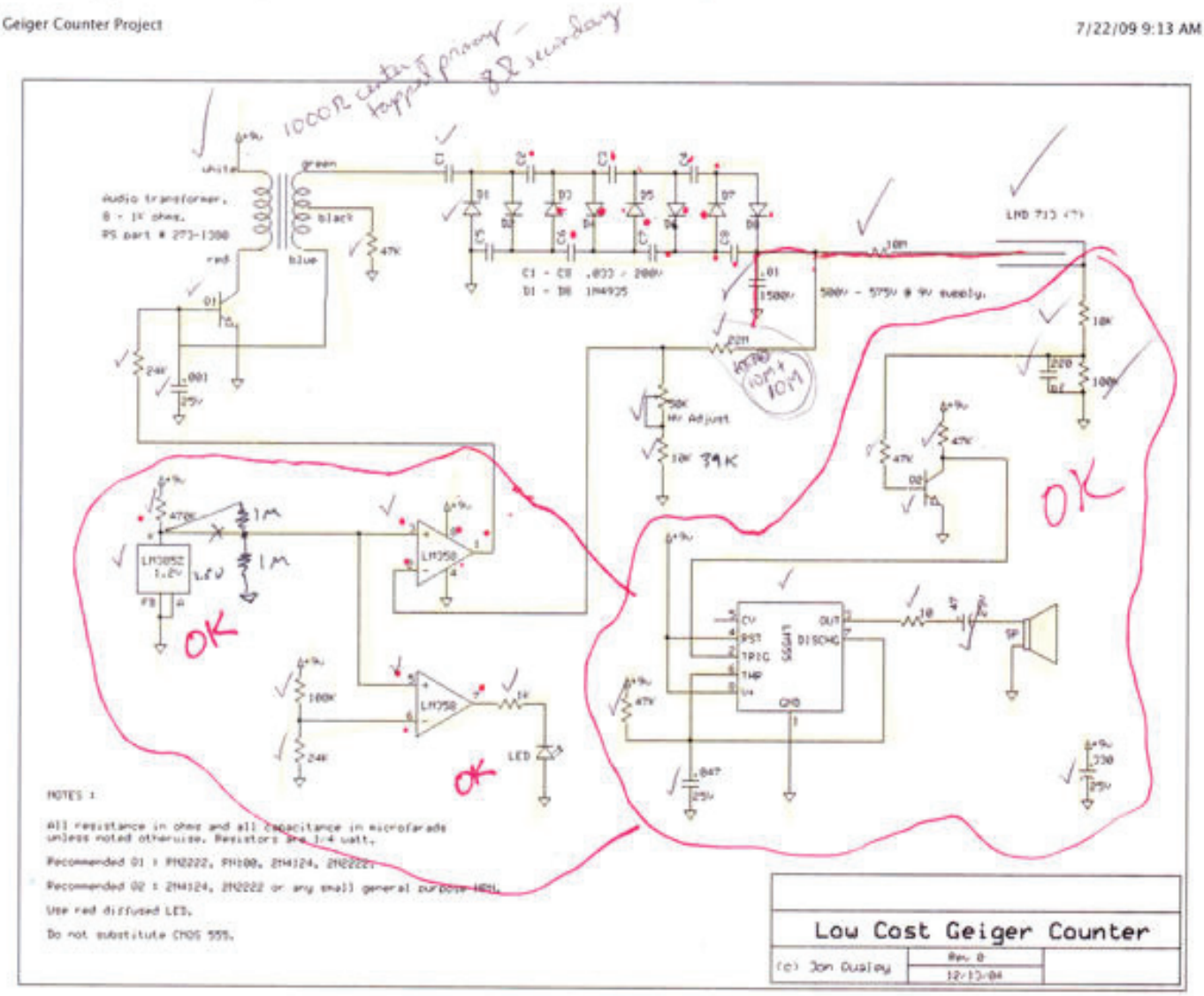
We had a great time, learned a lot, solved many of the world's problems over dinner, and look forward to the next time we have an opportunity to hack up a project at Defcon. ∞





contact us at info@optimizedtomfoolery.com

join our ongoing microwave-band amateur radio project at www.delmarnorth.com/microwave

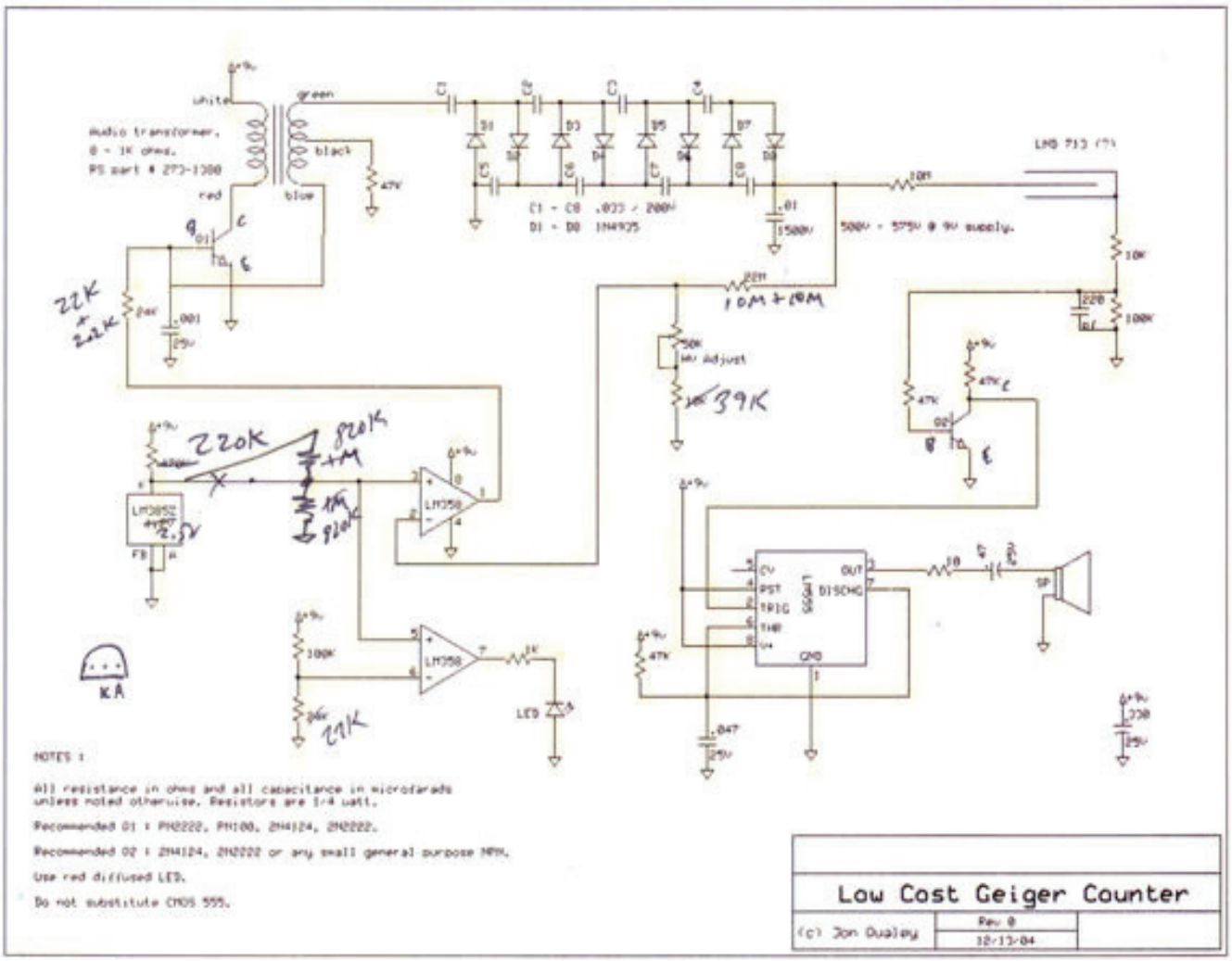


Construction notes -

In this project I took the time to manufacture a printed circuit board using the Ferric Chloride etching technique. I purchased high quality, single sided fiberglass PC board stock from a local surplus outlet (Electronic Surplus, Inc. in Cleveland, OH). I produced the circuit traces (circuit mask) using dry transfer traces I purchased from Radio Shack. I chose fiberglass PC board stock because of it's high resistance which I think is desirable considering the high voltages involved.

The circuit board is pictured below. It is shown upside-down so that the battery does not obscure the circuit board. The voltage multiplier can be seen in the upper left. The LM358 and 555 are seen in the lower right. The Geiger-Mueller tube can be seen in the lower left of the circuit board. I realize now that the GM tube should have been mounted at the bottom of the circuit board (as shown) perpendicular to it's current position. This way the GM tube will be most sensitive to radiation in front of the user, instead of to the left side as the GM tube is currently mounted.

The plastic enclosure used was purchased from Radio Shack. I drilled concentric rings of holes for the speaker outlet and four holes for mounting the circuit board. There is a power switch which is not shown in the schematic.



Construction notes -

(flat)
EBC

In this project I took the time to manufacture a printed circuit board using the Ferric Chloride etching technique. I purchased high quality, single sided fiberglass PC board stock from a local surplus outlet (Electronic Surplus, Inc. in Cleveland, OH). I produced the circuit traces (circuit mask) using dry transfer traces I purchased from Radio Shack. I chose fiberglass PC board stock because of it's high resistance which I think is desirable considering the high voltages involved.

The circuit board is pictured below. It is shown upside-down so that the battery does not obscure the circuit board. The voltage multiplier can be seen in the upper left. The LM358 and 555 are seen in the lower right. The Geiger-Mueller tube can be seen in the lower left of the circuit board. I realize now that the GM tube should have been mounted at the bottom of the circuit board (as shown) perpendicular to it's current position. This way the GM tube will be most sensitive to radiation in front of the user, instead of to the left side as the GM tube is currently mounted.

The plastic enclosure used was purchased from Radio Shack. I drilled concentric rings of holes for the speaker outlet and four holes for mounting the circuit board. There is a power switch which is not shown in the schematic.

Source Code Modifications

```
//#define VERBOSE 1

/*****************************************************************************
*
* DEFCON 17 BADGE  --  Hacked!
*
* Filename:             DC17_Badge.c
* Author:                 Joe Grand (Kingpin)
* Hackers:              Michelle Thompson (Abraxas3D)
*                               Paul Williamson (Little Skunk)
*                               Keith Wheeler
* Revision:             0.001
* Last Updated: July 31, 2009
*
* Description:        Main File for the DEFCON 17 Badge (Freescale MC56F8006)
* Notes:
*
* See DC17_Badge.h for more inf0z...
*****************************************************************************/

/* Including needed modules to compile this module/procedure */
#include "Cpu.h"
#include "Events.h"
#include "PWMRed.h"
#include "PWMGreen.h"
#include "PWMBlue.h"
/* Including shared modules, which are used for whole project */
#include "PE_Types.h"
#include "PE_Error.h"
#include "PE_Const.h"
#include "IO_Map.h"

#include "DC17_Badge.h"

/*****************************************************************************
 ************************** Global variables ********************************
 *****************************************************************************/

// LED
led_state_type LEDstate = RED;
unsigned int LEDptr = 0;

// Make the linker happy
struct packed_flags flags;

// Pass Geiger counter results up from ISR
long    geiger_count;
int             new_geiger_count = 0;
```

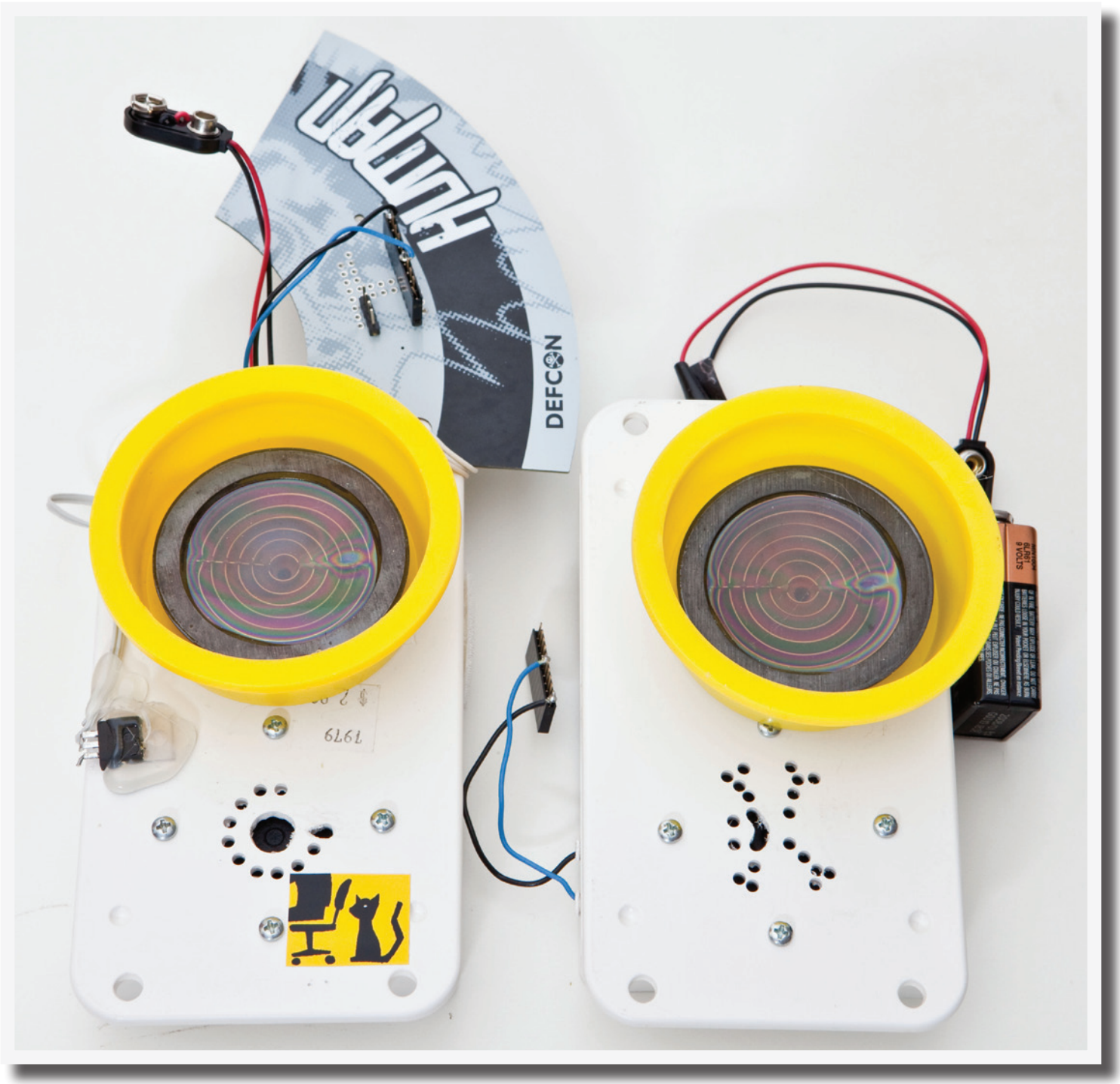
```
#ifdef KINGPIN_BADGE
...

#endif

/*****************************************************************************
 ************************** Functions ***************************************
 *****************************************************************************/

#define GEIGER_0 1
#ifdef GEIGER_0
void dc17_badge(void)

{
        int             rng_state = 0;
        long    rng_first_interval;
        int             rng_bit;
        long    rng_bitcount = 0;
        long    rng_ones = 0;
        long    rng_zeroes = 0;

#define DISPLAY_LINE_LEN        80
        char    display_string[DISPLAY_LINE_LEN + 3];
        char    *display_p = display_string;
        int             display_count = 0;

        geiger_init();

        while(1)
        {
                if (new_geiger_count)
                {

#ifdef VERBOSE
                        Term1_SendStr("Click ");
                        Term1_SendNum(geiger_count);
                        Term1_SendStr("\r\n");

 #endif
                        if (rng_state == 0)
                        {
                                rng_first_interval = geiger_count;
                                rng_state = 1;
                        }

                        else    // we have two new interval measurements. Make a bit!

                        {
                                rng_bit = (geiger_count > rng_first_interval);
```

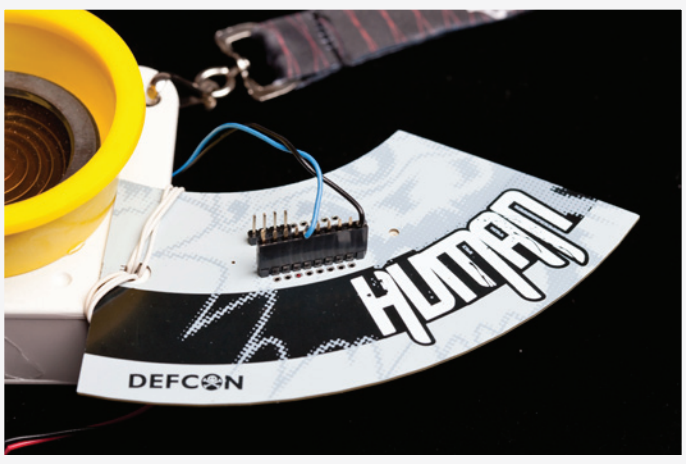
```
#ifdef VERBOSE
                        Term1_SendStr("First interval: ");
                        Term1_SendNum(rng_first_interval);
                        Term1_SendStr("  Second interval: ");
                        Term1_SendNum(geiger_count);
                        Term1_SendStr("  Random bit: ");
                        Term1_SendNum((long)rng_bit);
                        Term1_SendStr("\r\n");

#endif
                        rng_bitcount++;
                        if (rng_bit)
                                rng_ones++;
                        else
                                rng_zeroes++;

#ifdef VERBOSE
                        Term1_SendNum(rng_zeroes);
                        Term1_SendStr(" zeroes, ");
                        Term1_SendNum(rng_ones);
                        Term1_SendStr(" ones. Total bits: ");
                        Term1_SendNum(rng_bitcount);
                        Term1_SendStr("\r\n");

                        *display_p++ = (char)(rng_bit ? '*' : '.');
                        if (++display_count >= DISPLAY_LINE_LEN)
                        {
                                *display_p++ = '\r';
                                *display_p++ = '\n';
                                *display_p++ = 0;

                                Term1_SendStr(display_string);

                                display_count = 0;
                                display_p = display_string;
                        }

                        Term1_SendStr("\r\n");
#else
                        Term1_SendStr(rng_bit ? "*" : ".");
#endif
                        rng_state = 0;
                }

                new_geiger_count = 0;
        }
    }
}
```

```
/*************************************************************/
void geiger_init(void) // badge start-up/initialization
{
        Cpu_DisableInt(); // disable global interrupts
        PWMRed_Enable();  // while we enable all the modules
        PWMGreen_Enable();
        PWMBlue_Enable();
        //MICOUT_Enable();

        Term1_SendStr("Welcome to the Geiger Counter Mark Zero.\n\n\r");

        TI0_Enable();
        TI1_Enable();

        Cpu_EnableInt();  // re-enable global interrupts when we're done and ready for action

        TIPIT_Disable();      // disable PIT to prevent blending

        // initialize global flags

        //!!!

}


/*************************************************************
/* INTERRUPT SERVICE ROUTINES
/*************************************************************/

void dc17_pit_isr(void) // Programmable Interval Timer (PIT): every 8ms, set to operate during Stop
Mode

{
}

void dc17_t0_isr(void) // TMR0: formerly A/D sampling, 8kHz (every 0.125ms)

{
        static int click_stat = 0;
        static long     click_time = 0;

        // for iteration zero, poll the Geiger counter here

        // for iteration zero, just set the LED to match the Geiger counter input

        int geiger = A0_GetVal();
        if (geiger && !click_stat)
        {
                PWMRed_SetRatio16(ON);
```

```
                PWMGreen_SetRatio16(ON);
                PWMBlue_SetRatio16(ON);
                if (!new_geiger_count)
                {
                        geiger_count = click_time;
                        new_geiger_count = 1;
                }
                click_stat = 1;
        }
        else if (!geiger && click_stat)
        {
                PWMRed_SetRatio16(OFF);
                PWMGreen_SetRatio16(OFF);
                PWMBlue_SetRatio16(OFF);
                click_stat = 0;
                click_time = 0;
        }
        else if (!geiger)
        {
                click_time++;
        }

}

void dc17_t1_isr(void) // TMR1: RGB multiplexing, every 1ms
{
                switch (LEDstate)
                {
                        case RED:
                                setRegBit(GPIO_A_PER, PE0);
                clrRegBit(GPIO_A_PER, PE1);
                clrRegBit(GPIO_A_PER, PE2);
                        LEDstate = GREEN;
                                break;
                        case GREEN:
                                clrRegBit(GPIO_A_PER, PE0);
                setRegBit(GPIO_A_PER, PE1);
                clrRegBit(GPIO_A_PER, PE2);
                        LEDstate = BLUE;
                        break;
                        case BLUE:
                                clrRegBit(GPIO_A_PER, PE0);
                clrRegBit(GPIO_A_PER, PE1);
                setRegBit(GPIO_A_PER, PE2);
                        LEDstate = RED;
                                break;
                }
}
```

```
#endif  // GEIGER_0

/*****************************************************************/

#ifdef KINGPIN_BADGE

...

#endif // KINGPIN_BADGE


/****************************** END OF FILE **********************
```





The End