A COORDINATE DECODER

For a while now I've wanted to set up a geocache puzzle which the cacher could only solve by building a simple electronic circuit. This is a brief description of the design, which is now deployed near Cambridge, UK. The cache itself is GC40ZBT but to tackle it, you'll have to solve GC40ZBMfirst.

Both the hardware designs and software are freely available, but the keys used in the caches above aren't included. In other words, nothing in this article will help you solve the actual geocache puzzle!

Desiderata

It's clear what sort of gadget we want:

- It should have flashing lights.
- It should have knobs to tweak, and buttons to push.
- It should be simple to build.

It's also clear that there are some constraints:

- It must not cost the cacher much money.

- The cacher *should not* be able to solve the puzzle other than by building the circuit.

- Most cachers should be able to build the gadget without special training.

After a bit of thinking I decided that a microcontroller running some sort of cryptography software would fit the bill. I reckoned that it would be unreasonable to ask the punter to enter more than 32-bits of data, which gives us an upper bound of $2^{32} \approx 4$ billion possibilities. To get a feel for this, if each test took one second, it would take over 130 years to try them all.

My initial thought was to build something a bit like an electronic safe: once built, the cacher could enter a code by turning a rotary encoder, if he got the combination right the coordinates would be displayed.

However, whilst thinking about this, another idea struck me. Geocachers normally use coordinates specified in thousandths of a minute of arc, so there are 60,000 divisions to each degree. That's a bit less than 2^{16}, so in 32-bits we could easily fit a square degree of locations. So, instead of building a digital safe, it might be better to build a digital scrambler which would assign a random looking 32-bit code to each nearby location, or more usefully convert the code into coordinates.

Specifically, the gadget would accept a 32-bit number as input, decrypt it into another 32-bit number, then use that as a pair of 16-bit N/E offsets. All of this design could be public, save for the key to the scrambling step. Given such a device it would be possible to give the cacher a code to any nearby coordinates, so if the cache was moved the hardware wouldn't need to be changed. It would also be easy to use the same hardware to decode several locations in the puzzle.

Of course, the problem remains of getting the design to the cacher without him being able to simulate it. In the end, a simple solution presented itself: microcontrollers are very cheap, so I could simply give them to the cacher. After all, it's not as though puzzle caches are particularly popular in Cambridge! Giving out pre-programmed microcontrollers solved another problem: how would cachers actually program the chips.

Having started along this path, it became awfully attractive to give away a full kit of parts. That made it easy to make sure that people were using the right components, and solved any worry that people would avoid the cache because it was expensive. On the other hand, I didn't want to bear the cost of boundless components myself. The obvious solution was to loan out kits of components and a breadboard: cachers could build the gadget, use it, then return the kit for someone else.

Obviously there's some risk that people would just walk off with things, but geocachers seem to be a trustworthy tribe.

Source: http://www.mjoldfield.com/atelier/2012/12/coord-decoder.html