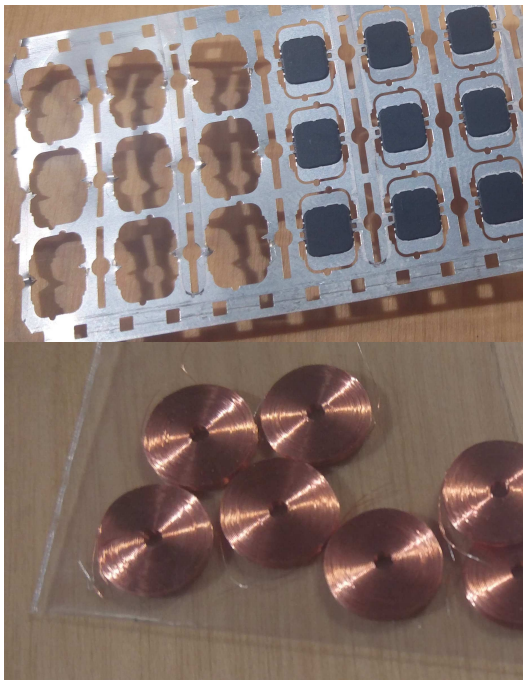
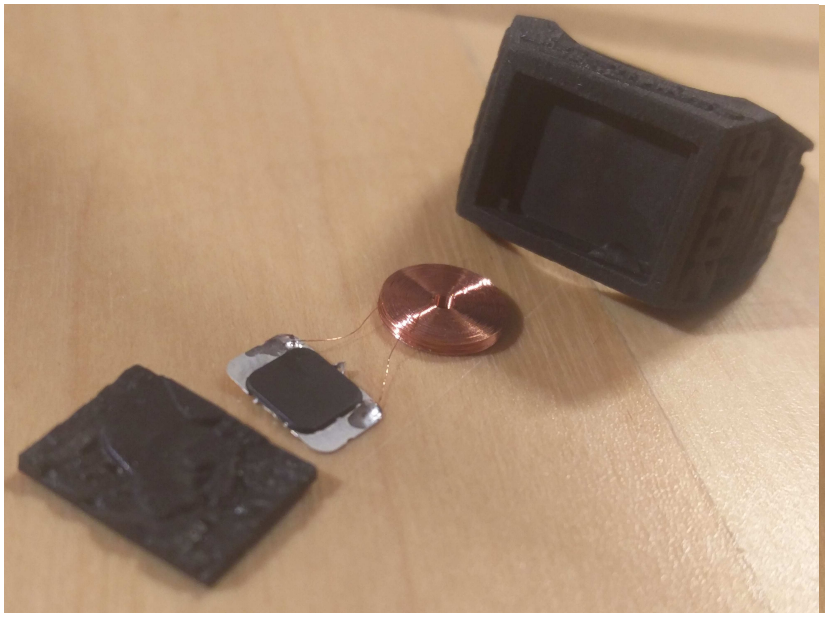
How Your Brass Rat Could Care For You!

Wouldn't it be cool if your brass rat was actually useful for something, in addition to methaphorically opening doors? Proposal:



chip: "ATA5577 compatible" with 330pF cap, $50/100 on Digikey
modulation: f=125KHz, "PSK1" at f/2, data rate f/32, total 224 bits

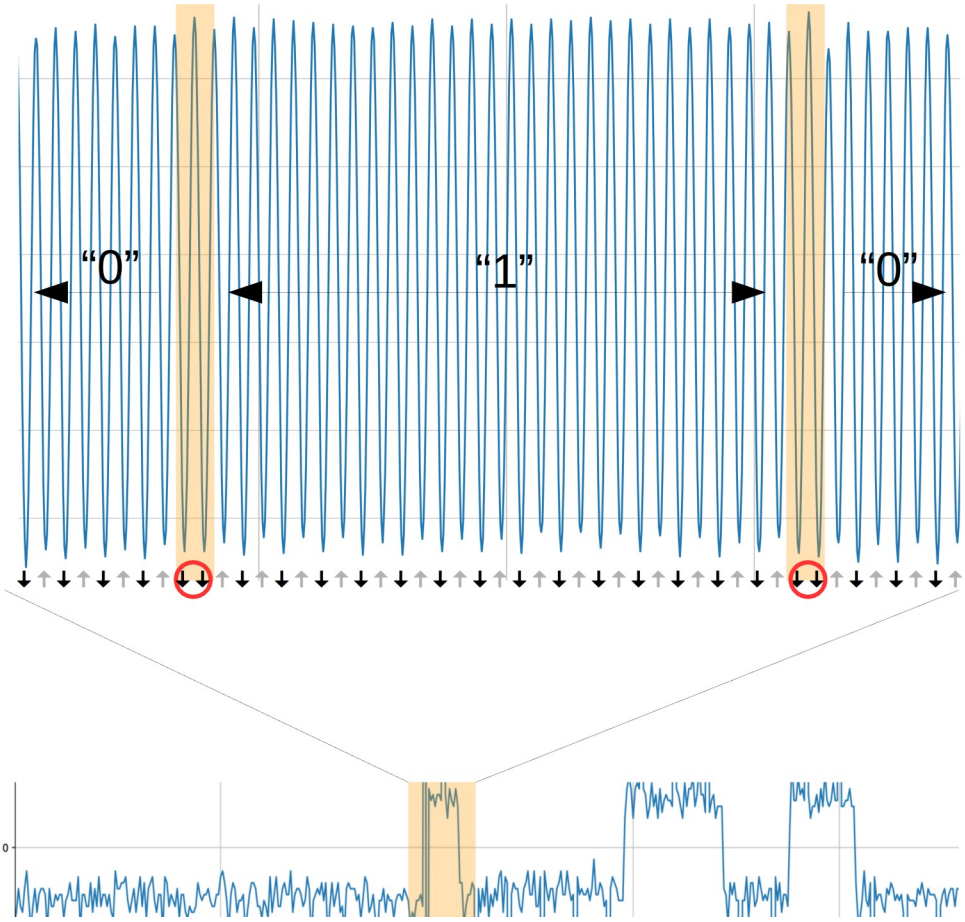
Coil retail varies (AliBaba, ebay), ~$150/200 at SmartPrototyping
"5" ring coil: ⌀ 1.2mm, ∅ 8mm, .9mm, 1.5mH – tiny range, flaky
"8" ring coil: ∅ 9.6mm, 1.5mH – small range, ok



Reading RFID with oscilloscope and eyes

Setup: 125Khz - L - Vout - C - gnd

1. Pick either the top or bottom envelope of the signal (try both).
2. Classify each peak as “high” or “low”, find alternating pattern.
3. A pair of non-alternating bits indicates phase reversal of the half-frequency modulation.
4. Manually note down 224 bits…



Reading RFID in code

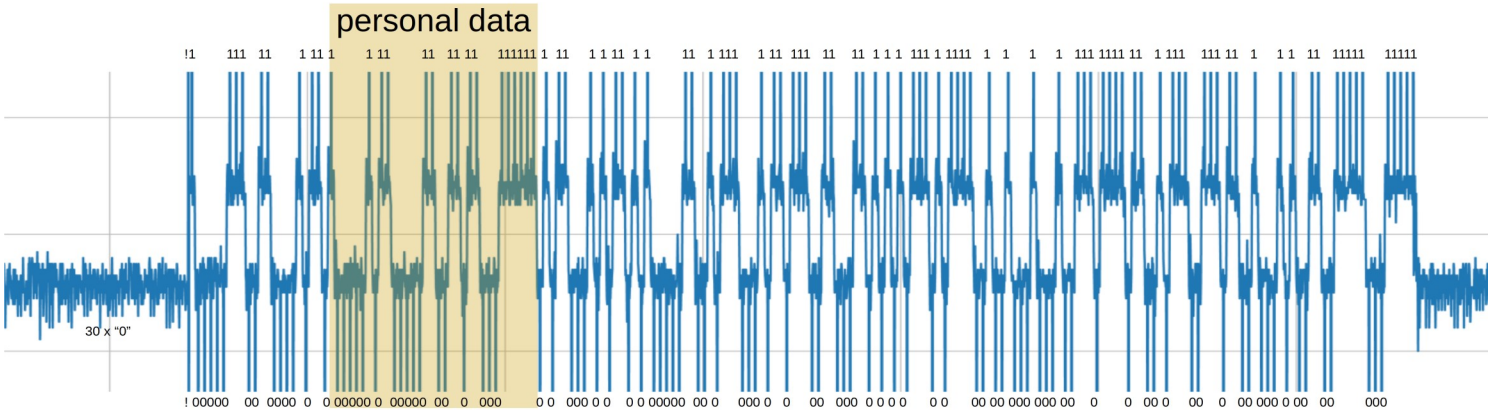
Setup: Vout --> 2MSPS ADC

1. Find the 125KHz peaks on the preferred side of the signal.
2. Plot (odd – even) for each pair of successive peaks, as shown on the plot on the right.
These correspond to data bits:

- negative → “0”
- roughly 0 → transitioning
- positive → “1”

3. The “beginning” of the signal has 32*30 cycles of bit “0”, followed by 32 cycles of “1”.
4. Start sampling 16 into the “1”.
5. Sample 194 times at 32-cycle intervals.
6. Total bits: 30 “0”, 22 const, 33 personal, more const (total 224).

Below, sampling points are marked with vertical jumps:

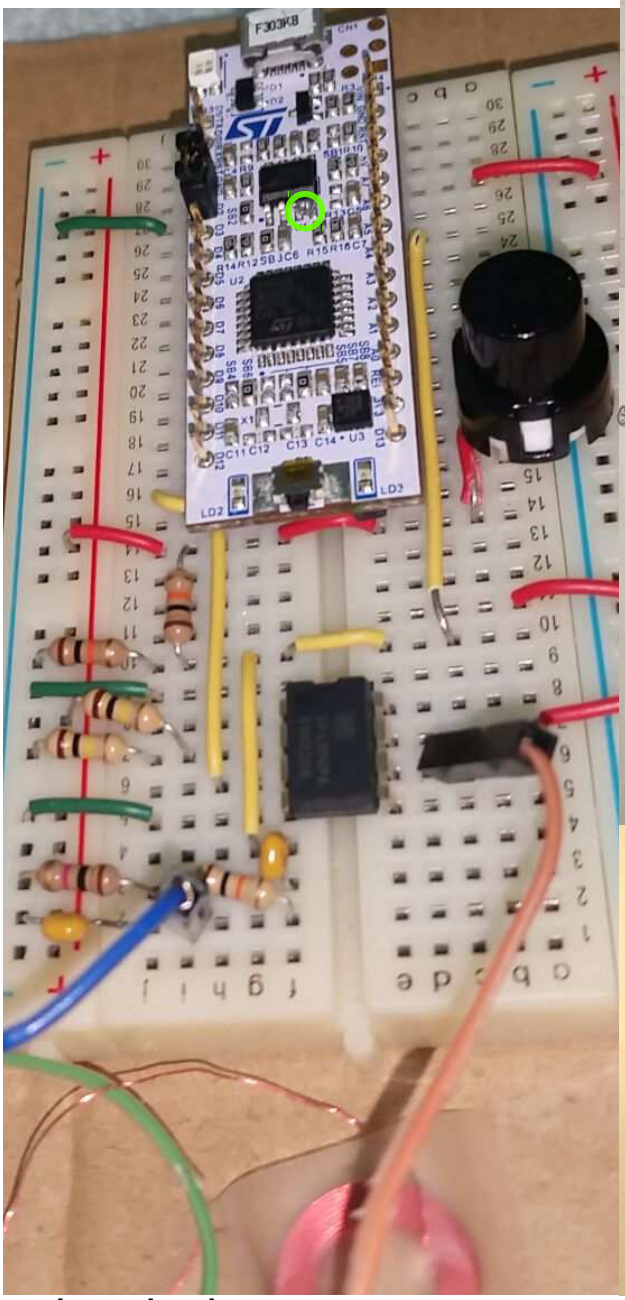


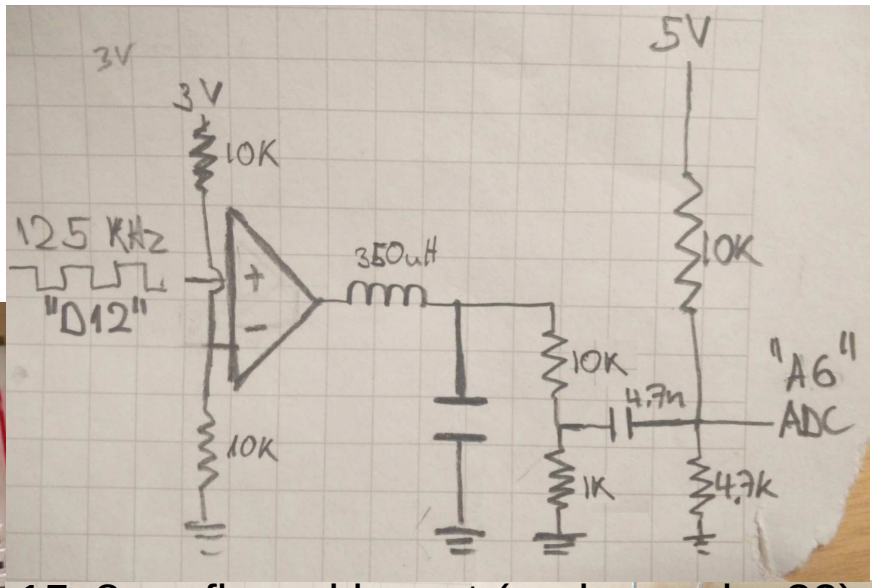
Source materials (mechanical, electrical, software) for everything described in this booklet are available at

`https://github.com/arphid/arphid`

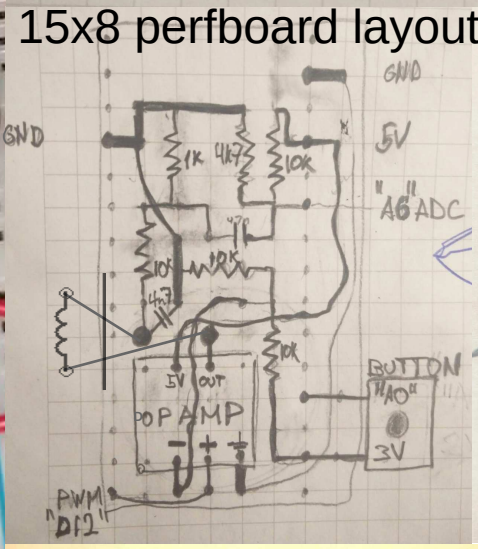
Building a 125KHz Reader+Writer For $15

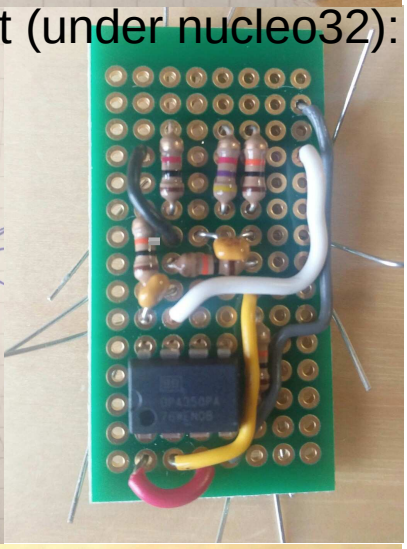
Reader+writer using $15 of standard components on a perfboard or (with less read range) a breadboard. Estimated <1h for build, progam, test. Perfboard: do not cut component leads early, instead bend them to make connections. nucleo32 w/o computer: remove SB9 and solder 2 DC regulator pins as below:

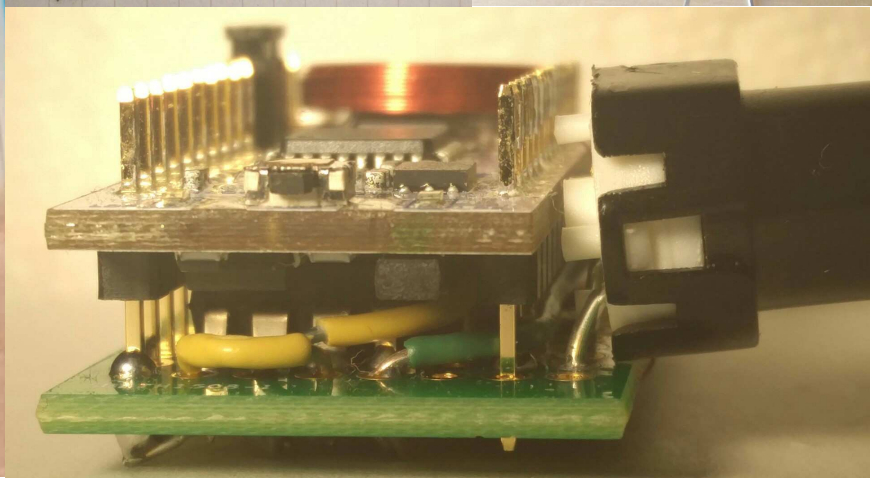




15x8 perfboard layout (under nucleo32):







Electrical components:

STM32F303K8 or similar microcontroller w 2MHz ADC	Nucleo-32 dev board	$11
opamp for 5V 125khz rail-to-rail squarewave output	OPA350PA (overkill)	$ 3
a button for switching between read and write mode	EG4791 (too bulky)	$ 1
capacitor and coil for 125KHz resonance	350uH x 4.7nF	

resistors: 1K, 4K7, 10K, 10K, 10K, 10K; additional dc-blocking cap: 4.7nF

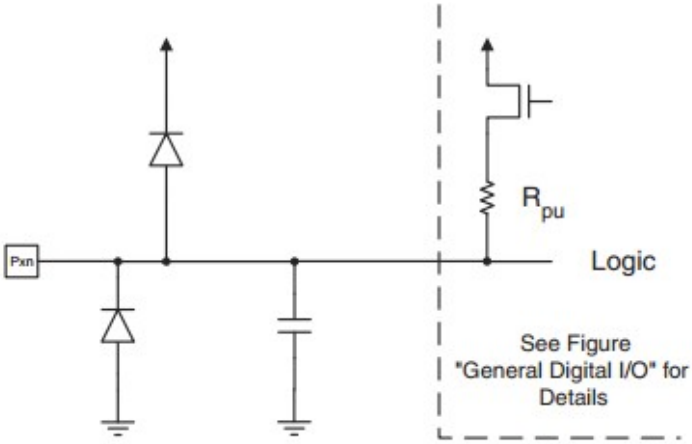
Not Hard Enough Yet? ATTiny85 as a tag!

It is actually possible to make an RFID tag without any RFID parts!

This is not black magic, but rather an inherent property of how RFID works. A good model of the RFID chip's effect on the coil is that on every clock cycle the chip can change its impedance. Thinking of the coils in the chip-coil-air-coil-reader system as transformer, it is obvious that changes in tag chip impedance reach the reader.

To do this, we need microcontroller that can change it's GPIO direction (whether a pin acts as an input or an output) every clock cycle. The overall system will consist of only three components: coil, resonant capacitor for 125KHz, and the microcontroller – all connected in parallel. On the microcontroller, the LC tank will act as:

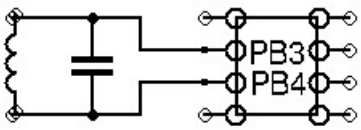
- An external clock – connect between CLK pins
- Output – using GPIO direction registers
- Power and ground – through built-in pin protection diodes diodes as depicted below



ATTiny85 pin protection diodes

```
  out     0x17, %1 // 1 cycle
  out     0x17, %2 // 1 cycle
  out     0x17, %1 // 1 cycle
  out     0x17, %2 // 1 cycle
phaseEvenSel:
  ld  r30, Y+      // 2 cycles
  ijmp // to phaseEven/Odd, 2 cycles
.balign 0x80,,
phaseOdd:
  out     0x17, %2 // 1 cycle
  out     0x17, %1 // 1 cycle
  out     0x17, %2 // 1 cycle
```

Excerpt from cycle-accurate code



Complete circuit diagram of RFID emulator

ATtiny85, capacitor, and a hand-wound coil

The tag code is written in assembly and carefully designed for cycle-accurate timing. Most clock cycles are spent setting coil impedance to send the right bit to the reader, but 4 of every 32 cycles are needed to run control logic. This is acceptable because readers have error correction. However, a *frame* error of even 1/(224*32) breaks reading.

The code has two main sections, "output odd phase" and "output even phase". Each of these sections consists of 28 single-cycle output commands, a 2-cycle load-post-increment command, and a 2-cycle jump to the loaded address. The load reads the next bit of card data (encoded as an address in the assembly code) and the jump resets to the beginning of "output odd/even phase". At the end of 224 bits special control code is run to reset the bit counter in addition to reloading the first bit.

Unrelatedly, EEPROM does work on coil power.

Это самиздат.
€0, $0 USD,£0, 0 RSD, 0 SEK, $50 CAD.
Don't link, mirror! Don't steal, copy!