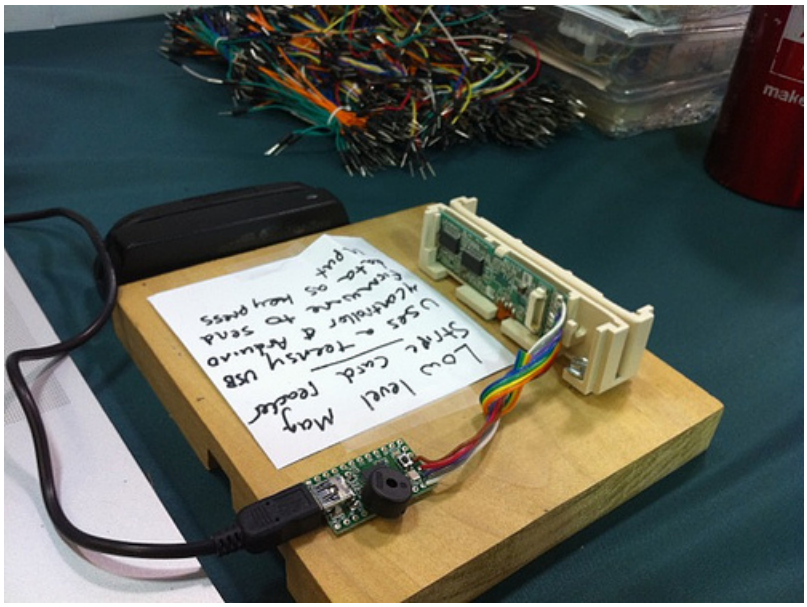
adafruit learning system

Low Level Magstripe Reader

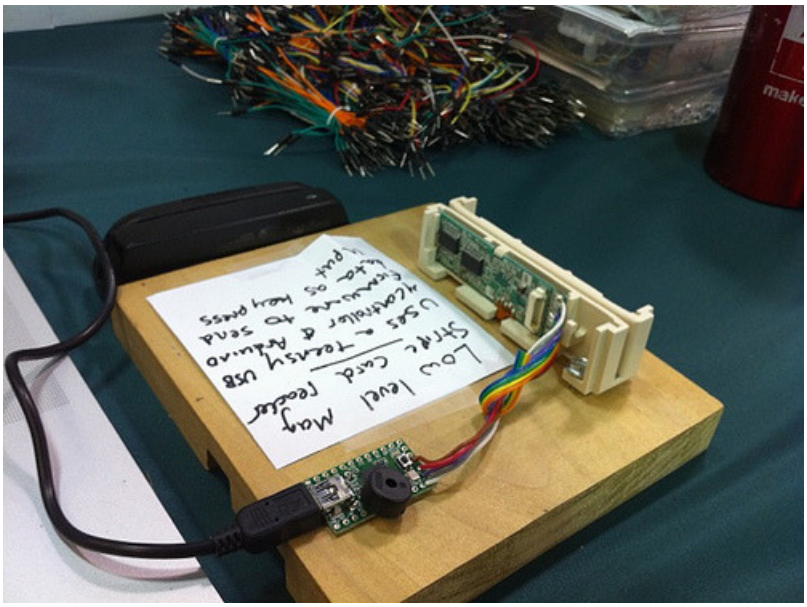
Created by lady ada



Last updated on 2019-04-08 05:02:58 PM UTC

Low Level Magstripe Reading

Swipe swipe...data!



TL;DR

In order to get raw parsed data out of a magstripe reader, we first experiemented with a MAGTEK Centurion Keyboard Encoder (PN-21073062). We found that although we could get all 3 tracks of data, it was not possible to have it parsed out. We then purchased a raw magstripe decoder head with track 1 reading, the Omron V3A-6 (https://adafru.it/c7V) (Datasheet here (https://adafru.it/c7W)). By writing some parity checking code, we were able to read the raw data off of the magstripe, and parse it into output that would be 'typed out' as an emulated keyboard using a USB-enabled Teensy (http://adafru.it/199). An Arduino can also be used, and the data would be output as Serial which may also be useful.

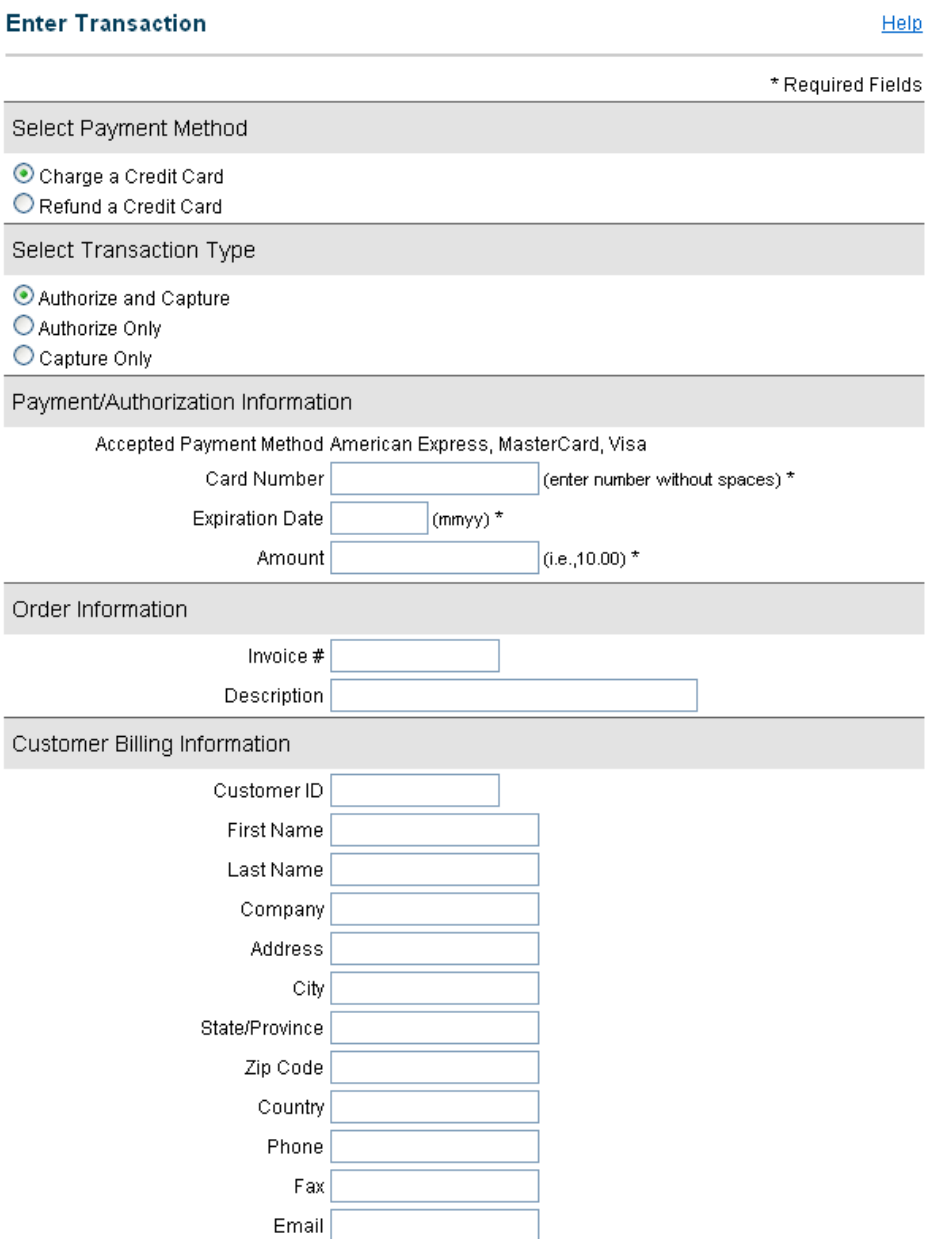
Source code and some handy datasheets are on GitHub. Click Download Source to get the latest version (https://adafru.it/c7X).

We also suggest checking out StripeSnoop (https://adafru.it/c7Z) which despite being a couple years old, was extremely useful!

This project can only be used for card reading, not writing. There is no way to convert a card reader into a card writer.

So there I was...

So there I was, 5 days before HOPE (Hackers on Planet Earth) conference (https://adafru.it/c0W) - getting ready for our booth that would be running all weekend. We had a similar booth at the previous HOPE, and it was pretty successful, one of the only things that hung us up was that we couldn't easily accept credit cards and the ATM had run out of money. We could take credit cards but it took a long time and was typo-ridden because everything had to be entered into a website. See, instead of a boxy terminal, we were using a Virtual Terminal - a website that allows you to enter in all the important data (card number, expiration, name, amount, etc).



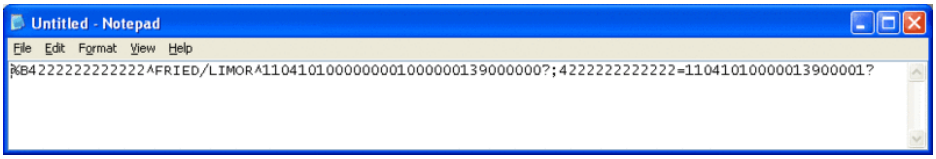
Of course, the virtual terminal was secured with an SSL certificate and we were running it over a cellular connection NOT wifi! But despite being secure, we still had to type all the data into the form and even one tiny mistake means starting over from the beginning!

Parts

Magstripe Reader #1

This year, we wanted to do it much faster, so that people didn't have to stand around for 3 minutes. So I got a magstripe reader! The model I tried first was the MAGTEK Centurion Keyboard Encoder (PN-21073062).

The nice thing about this reader is that it's small, lightweight, very reliable, reads all three tracks, and basically acts like a standard USB keyboard. Other readers can have HID or other drivers that are less trivial to use. With this reader, simply plug it in, start up a text editor and swipe:



The output is the error-corrected data right off the card, all three tracks, with the assumption that it's a proper credit card in the standard layout (https://adafru.it/c80).

The data output is as follows, split up by track.

1. %^^?
2. ;=?
3. nothing

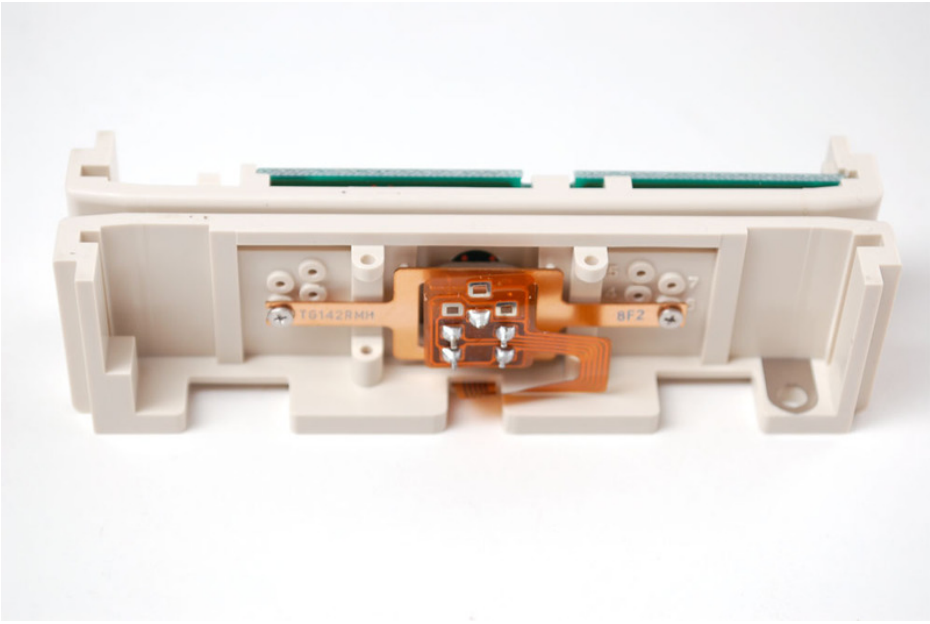
Track one is almost exclusively used for credit card data. Track 2 is used for pretty much anything including older ATM cards, other IDs, laundry, etc. Track 3 data tends to be used by driver licenses. Since I don't have such a license I can't demonstrate track 3.

Anyways, this is all very good and interesting, and I can get all the data I want this way. But what we really wanted was something to actually parse out the name, expiration and PAN (card number) from the data. There didn't seem to be any products that did this for us, so I built one instead!

Raw magstripe reader #2

What I wanted was to get that raw data before it got spit out the USB port. I opened up the Magtek and found that the whole thing is not easily hackable.

So instead I purchased a raw magstripe head from Mouser, remember that the data I want is on track 1 so I got the only head they had with track 1 reading, the Omron V3A-6 (https://adafru.it/c7V) (Datasheet here (https://adafru.it/c7W)). It can also read track 2.



It's a lot more hackable, with just the magnetic flux decoders (the chips) and all the data pins brought out and documented in the datasheet.

Double Track Connector

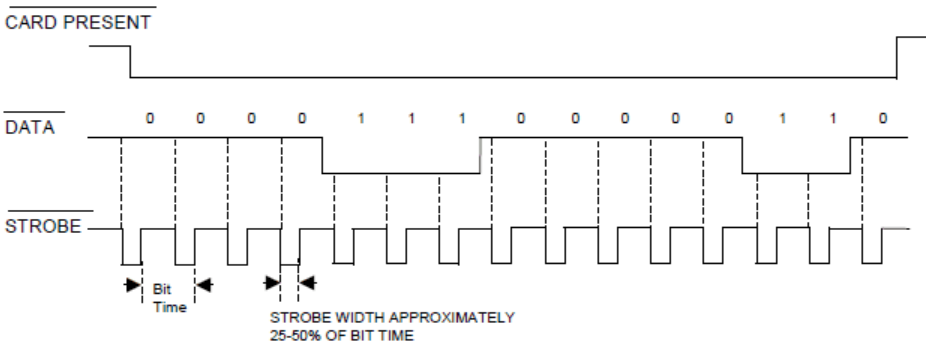
9-pin Molex connector part number 53261-0990. Applicable Molex socket housing part number 51021-0900.

Pin #	Signal	Input/Output	Description
1	0 V	—	—
2	+5 VDC	—	—
3	$\overline{\text{CSV}}$	Input	Current save
4	$\overline{\text{CLS2}}$	Output	Card loaded 2
5	$\overline{\text{RCP2}}$	Output	Read clock 2
6	$\overline{\text{RDP2}}$	Output	Read data 2
7	$\overline{\text{CLS1/3}}$	Output	Card loaded 1/3
8	$\overline{\text{RCP1/3}}$	Output	Read clock 1/3
9	$\overline{\text{RDP1/3}}$	Output	Read data 1/3

Pins 1 and 2 are obvious: power suppy. Pin 3 I didn't actually understand but perhaps it's a sleep mode. Pin 4 is a digital output that goes LOW when a card is being swiped (**CARD PRESENT**). Then the data is clocked in via pins 5 (**STROBE**) and 6 (**DATA**):

TIMING

The timing for Card Present, Strobe, and Data is as shown in Figure 2-3.



Notes:
1. Time out of the CARD PRESENT signal occurs approximately 150 ms after the last strobe transition.
2. DATA is valid 1.0∝sec before the negative edge of STROBE.

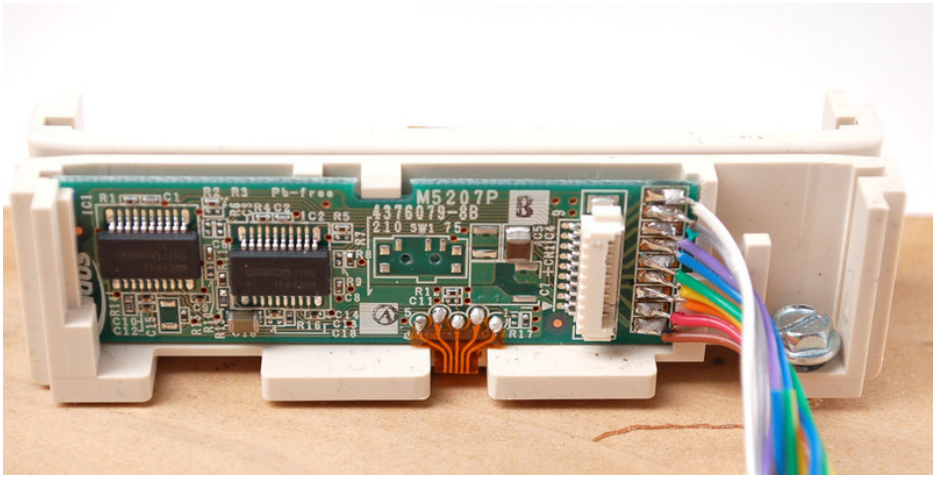
All this great timing data is from a handy app note by magtek (https://adafru.it/c82) (which I strongly recommend to any magstripe enthusiasts!). The data is sampled on the falling edge and is in LSB order - somewhat opposite most SPI implementations.

The data isn't in any 'standard' ASCII representation, it's actually 6 bits per byte with a parity bit to make it 7 bits long. Each byte should be checked for parity and then can be convereted to ASCII by adding 0x20 hex. There is some fun detail in said app note about how to calculate parity bits or you can just read the code. Again, this is only for track 1, track 2 is slightly different.

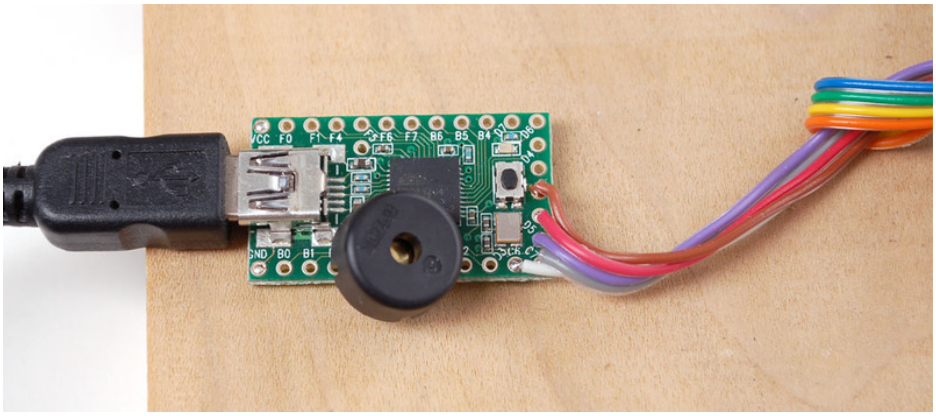
Wire Up

Wire up

Wiring up is really trivial, we simply soldered ribbon cable to each pin (even tho we didn't use them all).



Then connected +5V and ground and the three track 1 pins to the Teensy (http://adafru.it/199). Of course, you can also use an Arduino (or any other microcontroller) and just use the Serial output instead of Keyboard to read the data off.



We also added a piezo buzzer to give us some audible feedback.

Code snippets

This will be a quick review of a few interesting bits in the source which you of course check out on GitHub (https://adafru.it/c7X).

The first part is the data reading loop, which grabs the first ~80 non-zero bytes. We have to toss out the zeros as they are just buffers on the edge of the magstripe. There should be at least 4 but we don't use them so they get ignored in the loop.

Code Snippets

Code Download

The code and documentation is available on GitHub by clicking the green button below. Select Download Zip to get the code and documentation.

https://adafru.it/EtR

https://adafru.it/EtR

Code Review

The first part is the data reading loop, which grabs the first ~80 non-zero bytes. We have to toss out the zeros as they are just buffers on the edge of the magstripe. There should be at least 4 but we don't use them so they get ignored in the loop.

```
// for lots more info, see
// http://stripesnoop.sourceforge.net/devel/magtek-app.pdf

void loop()
{
  // wait till the card ready pin is low
  while (digitalRead(CARD2));

  uint8_t zeros = 0;
  uint8_t parityok = 0;

  // card was swiped!
  // check clocked in data
  for (uint8_t t1 = 0; t1 < TRACK1_LEN; t1++) {
    track1[t1] = 0;
    for (uint8_t b=0; b < BYTELENGTH; b++) {

      // wait while clock is high
      while (digitalRead(CLOCK2) && !digitalRead(CARD2));
      // we sample on the falling edge!
      uint8_t x = digitalRead(DATA2);
      if (!x) {
      // data is LSB and inverted!
        track1[t1] |= _BV(b);
      }
      // heep hanging out while its low
      while (!digitalRead(CLOCK2) && !digitalRead(CARD2));

    }

    if ((t1 == 0) && (track1[t1] == 0)) {
     // get rid of leading 0's
     zeros++;
     t1--;
     continue;
    }

    // we must have some leading zeros!
    if (zeros < 4) {
      t1--;
     continue;
    }

    // if the second byte is a zero, theres noise, backtrack
    if ((t1 == 1) && (track1[t1] == 0)) {
     t1 = -1;
     zeros = 1;
     continue;
    }
  }

  // all the data is read into the track1 buffer

  // shift left until we have no more starting zero bits!
  while ((track1[0] & 0x1) == 0 ) {
    shifttrack(track1, scratch, LEFT);
  }
```

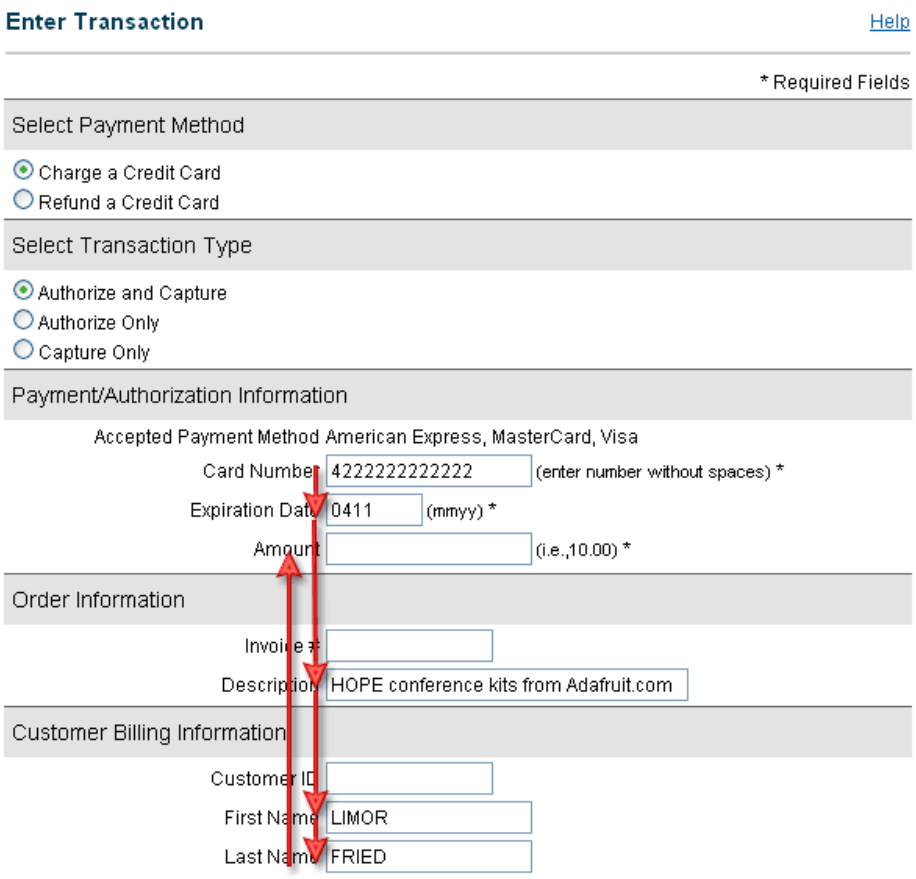
And here is the part after the parsing where we use the Teensy (http://adafru.it/199)'s built in Keyboard emulation to plug the data right into the virtual terminal. Note how it takes the raw data from the track and uses **Keyboard.print()** to output it as well as tab through the browser entries.

```
  // FIND PAN
    uint8_t i=2;
    while ((track1[i] & 0x3F) != 0x3E) {
#ifdef SERIAL
        Serial.print((track1[i] & 0x3F)+0x20, BYTE);
#endif
#ifdef KEYBOARD
        Keyboard.print((track1[i] & 0x3F)+0x20, BYTE);
#endif
      i++;
    }
```

...

```
#ifdef KEYBOARD
      Keyboard.print('\t');
      Keyboard.print(m1, BYTE);
      Keyboard.print(m2, BYTE);
      Keyboard.print(y1, BYTE);
      Keyboard.print(y2, BYTE);

      Keyboard.print('\t'); // tab to amount
      Keyboard.print('\t'); // tab to invoice
      Keyboard.print('\t'); // tab to description
      Keyboard.print("HOPE conference kits from Adafruit.com");
      Keyboard.print('\t'); // tab to customer ID
      Keyboard.print('\t'); // tab to first name
      Keyboard.print(fname);
      Keyboard.print('\t'); // tab to last name
      Keyboard.print(lname);

      for (uint8_t i=0; i<5; i++) {
        Keyboard.set_modifier(MODIFIERKEY_SHIFT);
        Keyboard.set_key1(KEY_TAB);
        Keyboard.send_now();
        Keyboard.set_modifier(0);
        Keyboard.set_key1(0);
        Keyboard.send_now();
      }

#endif

    beep(PIEZO, 4000, 200);
```

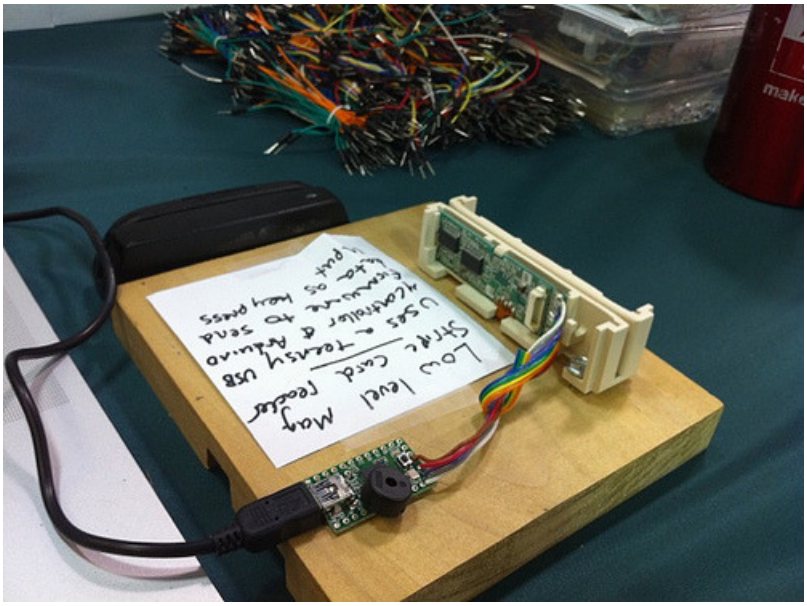
Once it's been swiped, the form is filled automatically, tabbing between the entries and then back-tabbing to the amount box so that can be filled out.


Enter Transaction
Help
* Required Fields
Select Payment Method
Charge a Credit Card
Refund a Credit Card
Select Transaction Type
Authorize and Capture
Authorize Only
Capture Only
Payment/Authorization Information
Accepted Payment Method American Express, MasterCard, Visa
Card Number
4222222222222
(enter number without spaces) *
Expiration Date
0411
(mmyy) *
Amount
(i.e.,10.00) *
Order Information
Invoice #
Description
HOPE conference kits from Adafruit.com
Customer Billing Information
Customer ID
First Name
LIMOR
Last Name
FRIED

Results

And the results?

The swiper worked great, not only was it cool looking but it worked flawlessly all weekend, swiping dozens of cards with almost no problems (one very old card with part of the stripe flaked off gave us trouble but the owner admitted it didn't work anywhere).



We learned a hell of a lot about magstripes and raw data encoding with this project and were informed by many

people that because it doesn't assume the standard formatting like most off-the-shelf swipers, it can be used for reading non-standard cards like Metrocards! We hope others will build this project and use it to decode all sorts of magstripes