TRAVIS GOODSPEED'S BLOG

Showing posts sorted by relevance for query **im-me**. Sort by date Show all posts

TUESDAY, MARCH 9, 2010

IM ME GoodFET Wiring Tutorial

by Travis Goodspeed <travis at radiantmachines.com>
concerning the Girltech IM ME,
with a million thanks to Dave.

WARNING: Reflashing the CC1110 while batteries are low will permanently lock the chip. Either be damned sure to use fresh batteries or leave the batteries out and power the IMME from your GoodFET.

Howdy y'all,

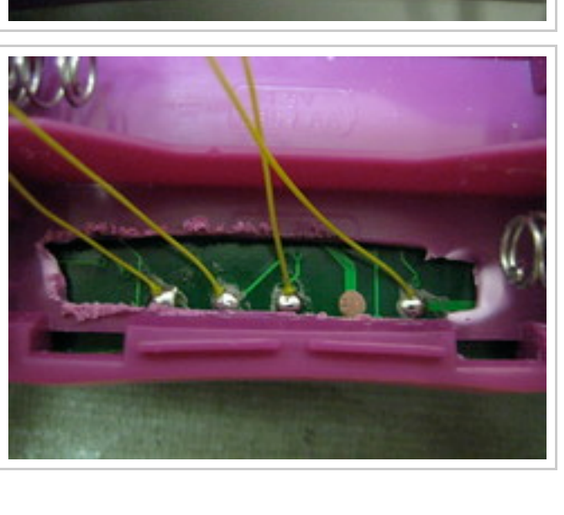
This brief tutorial describes the process of reflashing the Girltech IM ME with custom firmware, so that it may be used as a development platform for the Chipcon CC1110 sub-GHz ISM System-on-Chip. I assume the reader to have an assembled GoodFET with recent firmware, but other programmers may of course be substituted.

You should also read Dave's first article on IM ME hacking, as it describes his method for reprogramming the device. All the pinouts below were taken from his articles, as well as the keyboard and LCD information that he was so neighborly as to publish.

Wiring

First, you'll need to purchase an IM ME, which can be had for $20 USD on a few toy sites while it remains in stock. You'll also need an assembled GoodFET and basic electronics tools.

The testpoints used for programming the IM ME are located behind the batteries in the rear compartment of the device. Ideally, a bed of nails should be used to clip into it, but failing that, just solder on to the Debug Data (DD), Debug Clock (DC), Reset (!RST), and GND pins. Run these to the GoodFET's 14-pin header as shown below.



From left to right on the IM ME, the pins are !RST, DD, DC, +2.5V, and Ground. Because the GoodFET is a low-voltage device, there's no need for the resistor dividers in Dave's article. Use *EITHER* the GoodFET *OR* the batteries for VCC, but not both.

Name	Pin		Name
DD	1	2	Vcc
	3	4	Vcc
RST	5	6	
DC	7	8	
GND	9	10	
	11	12	
	13	14	

Flashing

Once you have the IM ME wired up, you can check its model number and status by running `goodfet.cc status'. This will tell you that the chip is locked, so making a backup of its firmware is non-trivial. If you continue from here, the IM ME will no longer function as an instant messenger.

Erase the chip by 'goodfet.cc erase' then dump an image of RAM as 'goodfet.cc dumpdata immeram.hex' to see if anything neighborly can be found inside.

You now have a blank IM ME, with the LCD most likely showing the last gasping breaths of its firmware. To flash a new firmware image, just grab its ihex file and run 'goodfet.cc flash foo.hex'.

I've placed a few example binaries in the repository of an operating system that I've started for the IM ME called GoodME. To flash Dave's LCD Test, run the following commands.
svn co https://goodfet.svn.sourceforge.net/svnroot/goodme
goodfet.cc flash goodme/bins/dave-lcdtest.hex

For a more functional demo, try *bins/term-morse824mhz.hex*, an ugly hack of an operating system for the IM ME with a Morse code transmitter and random number generator demo. In the Radio demo, holding any of the letter buttons broadcasts on 824MHz. The PRNG demo, shown below, demonstrates the repetition of strings withing the psuedo-random number generator and counts the number of bytes between them. This is sometimes used for key material.

Custom Development

The SDCC compiler is in the package repositories of most civilized operating systems. You might need a more recent version for the cc1110.h header, though building this compiler is a thousand times simpler than GCC. Compiling an example is as simple as *sdcc foo.c; packihx <foo.ihx >foo.hex*, which will produce a suitable Intel Hex file for flashing. The 8051 memory model makes specifying a chip model unnecessary, a handy deviation from those of us with a thousand MSP430 linking scripts.

Within the GoodME repository, you'll find my bastard child of an operating system at /branches/rough/. It was used to make the term-morse824mhz.hex, and its keyboard, font, and LCD drivers are ripe for organ transplants. /trunk/ ought to someday contain a proper operating system for the device, but for now, I haven't the time to complete it.

Have fun, and build something neighborly,
--Travis

POSTED BY TRAVIS GOODSPEED AT 9:11 PM 42 COMMENTS:
LABELS: CC1110, CHIPCON, GOODFET, IMME

MONDAY, DECEMBER 6, 2010

Hacking a Knitting Machine's Keypad

by Travis Goodspeed <travis at radiantmachines.com>
in collaboration with Fabienne Serriere and Arjan Scherpenisse,
at Mediamatic's RFID Devcamp 2010, Amsterdam,
for Multithreaded Banjo Dinosaur Knitting Adventure 2D Extreme,
with kind thanks to David Carne.

Thanks to some extra-neighborly blackmail by Fabienne, I spent last week hacking up a storm in Amsterdam. By extending the work of Steven Conklin, Limor Fried, and Becky Stern, we were able to hack a Brother KH930 knitting machine to print high score panes from a custom video game in yarn. This game was then displayed at Mediamatic for SensorFest, leading all sorts of neighbors to learn that RFID, socializing, and beer can lead to a neighborly time. Never in my life did I expect to get such an adrenaline rush from knitting, much less from from this newfangled social networking nonsense.

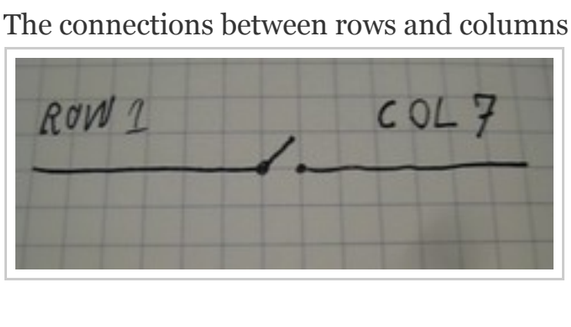
Conklin's technique for loading new patterns into the machine involves using an FTDI chip to simulate a Tandy PDD1 floppy disk drive, then typing "CE 5 5 1 STEP 1 STEP CE STEP STEP CE 9 0 3 STEP STEP" to load a new pattern set from disk and print pattern 3 of the set. (551 is the command to read from disk, and patterns held in RAM begin at 900.) For our exhibit, this sequence was far too complicated to type for every half meter of output, so we hacked the device's keypad for scripting.

This article describes how a matrix keypad works, how we reverse engineered the specific keypad of the Brother KH930, and how to use an Arduino with a handful of transistors to automate the typing of commands necessary for loading a new pattern from an emulated floppy disk. It should be applicable to all sorts of keyboards, but for those with serial protocols there might be a simpler method.

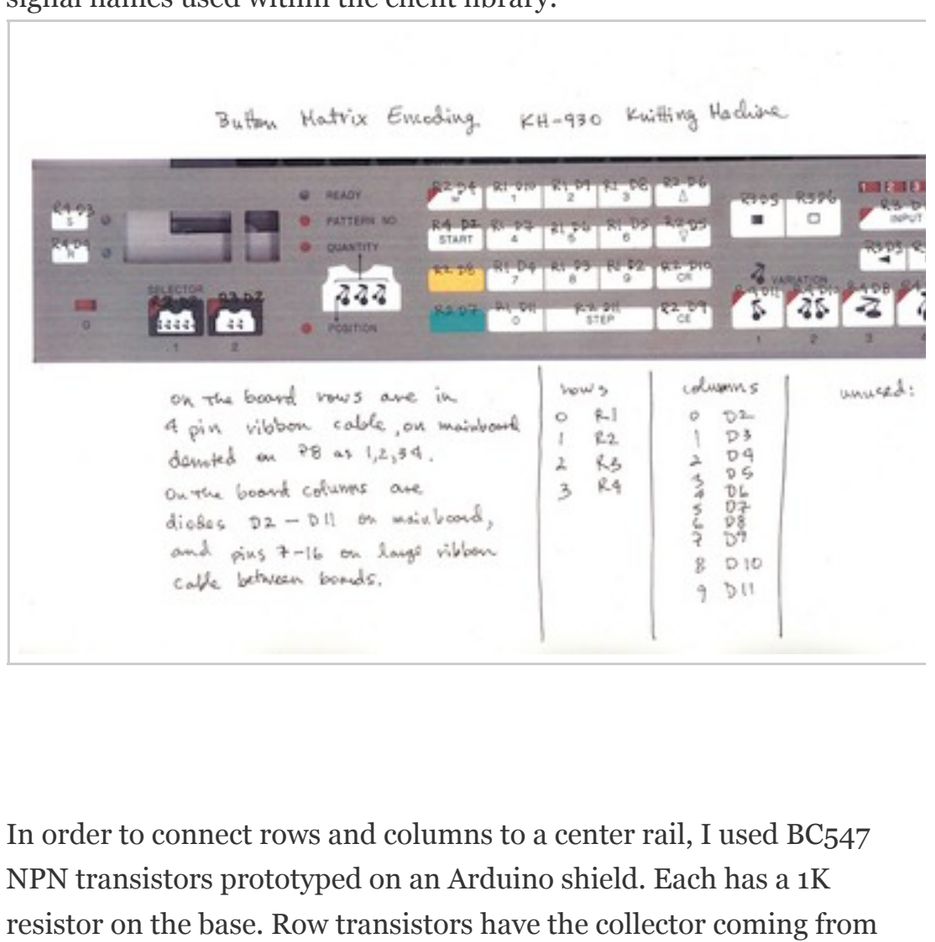
Unfortunately, there won't be room here to describe the first emergency knitting machine purchase in history, our A-Team trip to the far side of Holland in a borrowed van, or the case of Club Mate that we begged from a squatter bar in order to finish the project in the five days allotted.

A keypad matrix is most often built with switches that connect a row wire to a column wire. By keeping every column in a high-impedance state with pull-up resistors, then dropping each row low in sequence, the column inputs can be sensed to determine when a button is pressed. Dave's Hacks' first article on IM ME Hacking describes in detail how the keypad of the GirlTech IMME works, and all other implementations seem to function similarly.

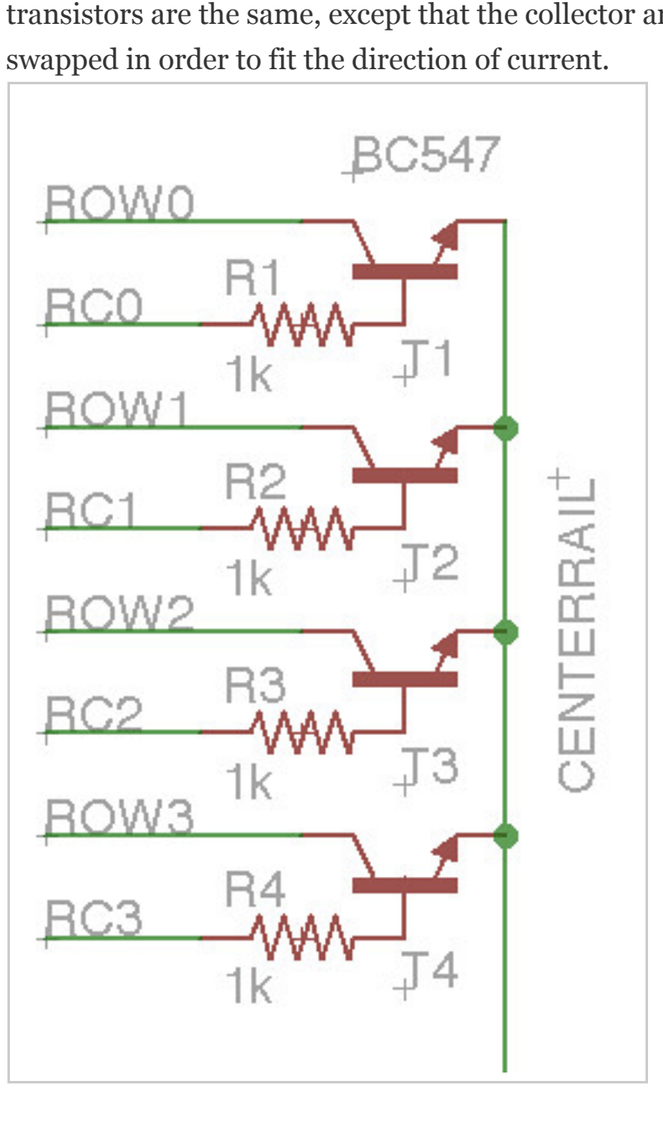
The connections between rows and columns are just switches.



While in a perfect world, rows and columns would be arranged exactly as they appear visually, this is rarely efficient to route in copper for more than twelve buttons. We produced the diagram below by scanning the keyboard membrane, then pushing a button while using the continuity tester to see which row and column wire were connected. The table at the bottom converts from wire signal names to the Arduino signal names used within the client library.

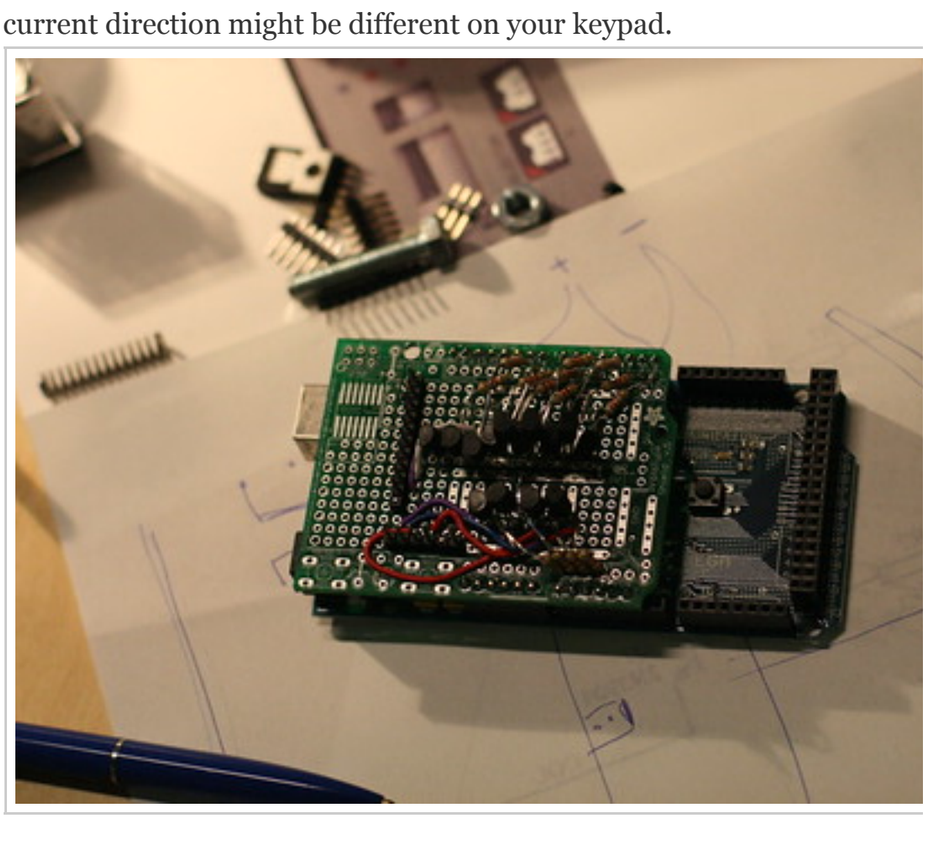


In order to connect rows and columns to a center rail, I used BC547 NPN transistors prototyped on an Arduino shield. Each has a 1K resistor on the base. Row transistors have the collector coming from the row signal and the emitter going to the common rail; column transistors are the same, except that the collector and emitter are swapped in order to fit the direction of current.

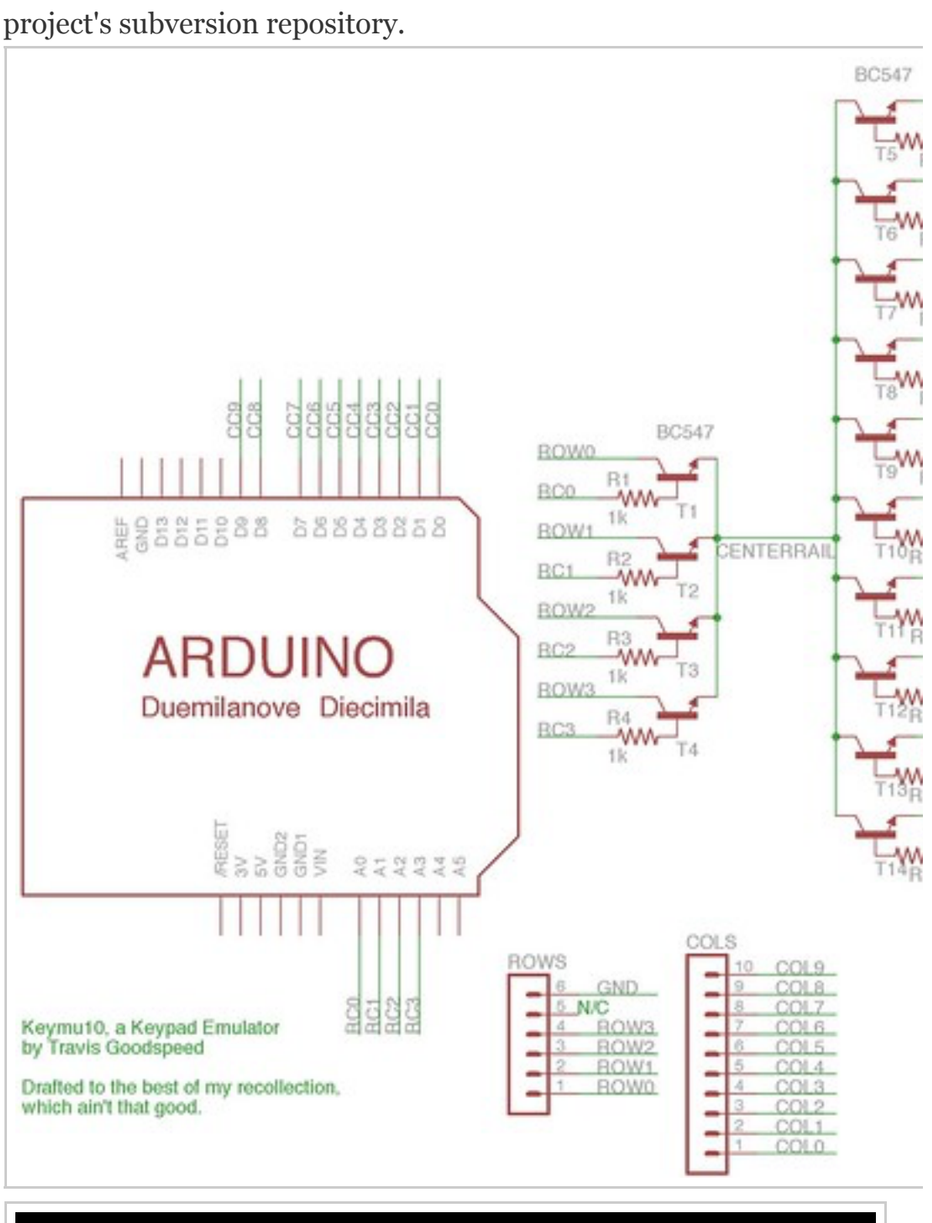


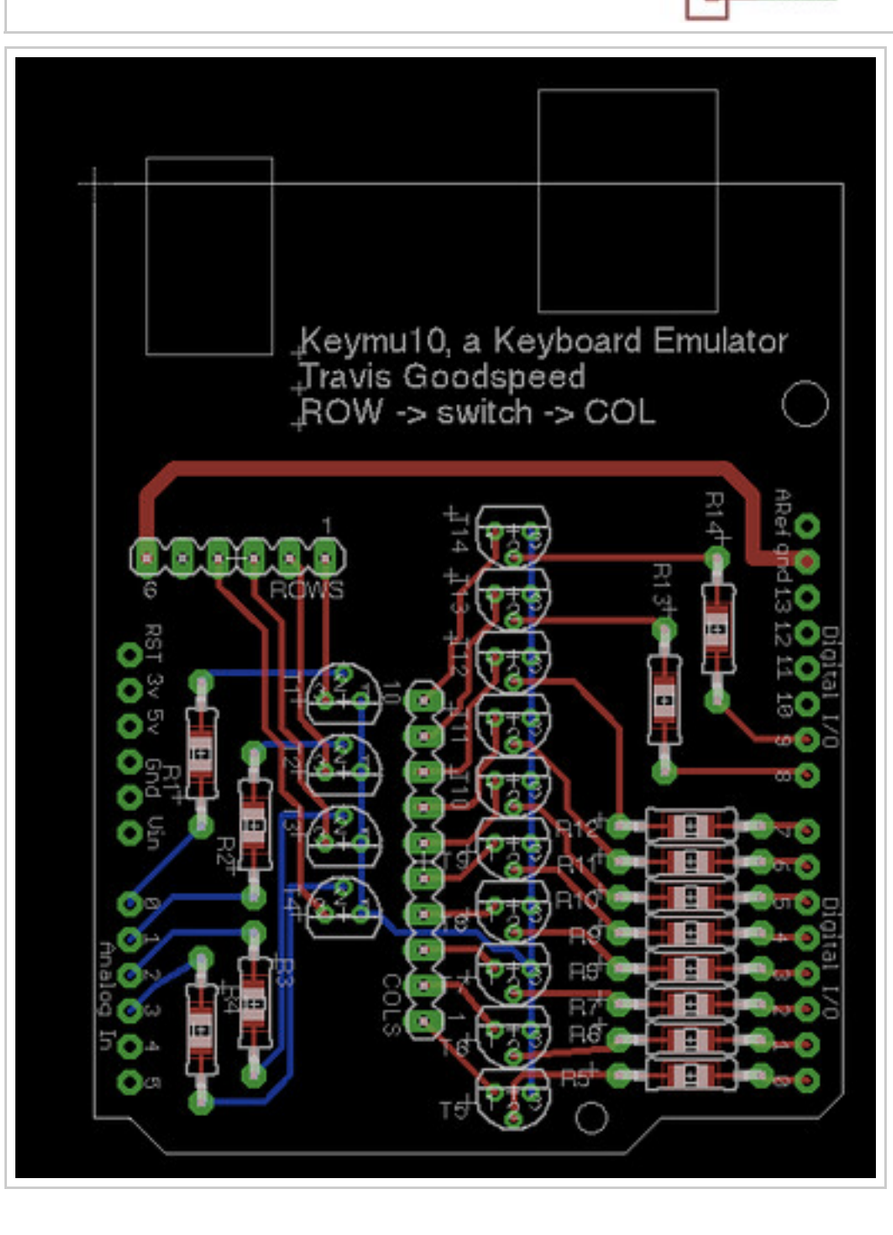
As each transistor conducts when only when its base is high, the Arduino can press a key by first dropping all outputs low, then raising a row control line (RCx) and a column control line (CCx) high. Because of debouncing and a limited scan rate, this must be held for a short amount of time before releasing the key.

Such a design fits perfectly well on a prototyping shield, although the transistors are a tight fit. I recommend first prototyping a single row and column to ensure that the transistors are properly aligned, as current direction might be different on your keypad.



For those who prefer to have a board fabbed, here's a proper schematic and layout. Gerbers and Eagle CAD source files are available in the project's subversion repository.





A more sophisticated software implementation would involve using the Arduino's serial port to communicate with the host, accepting strings in ASCII and translating them to keypresses. We decided against such a complication because we feared relying on a second long USB cable. Further, we thought it handy when necessary to be able to run the device stand-alone with a single (reset) button press from the operator signaling that the next pattern should be loaded.

Full keyboard emulator code is available either by pastebin or from SVN. You'll find it in banjo/code/arduino.
svn checkout svn://svn.mediamatic.nl/devcamps/camp10/banjo

The code works by using simple methods to press a row and column by raising those pin voltages. Higher level methods allow for such functions as loading a pattern from disk or printing a pattern, each by performing multiple key presses and occasionally inserting a delay. By placing this within the Arduino code's *setup()* function, the pattern is loaded whenever the device is reset.

New patterns are loaded from the Banjo Dinosaur Knitting Adventure 2D Extreme game by first starting a floppy disk emulator--as per Steve Conklin's technique--then pulsing the DTR line of the arduino in order to cause a reset. The Perl for that is just a single line,
Device::SerialPort->new("/dev/ttyACM0")->pulse_dtr_on(100)

The end result, with a machine typing by itself, looks a little like this,

Flash out-of-date

That's all there is to it. With fourteen transistors and just as many resistors, you can script your knitting machine--or any other keypad device--from a microcontroller without modifying the underlying firmware.

As a final note, I will give a cookie to the first neighbor who uses this technique to dump all of the power codes from a universal infrared remote control, or to program something long and sophisticated into a graphing calculator that lacks a link port.

POSTED BY TRAVIS GOODSPEED AT 2:16 AM 26 COMMENTS:
LABELS: ARDUINO, BROTHER KH930, DEVCAMP10, KEYPAD, KNITTING

Home

Subscribe to: Posts (Atom)

BLOG ARCHIVE
▼ 2013 (1)
▼ Jul (1)
Hillbilly Tracking of Low Earth Orbit Satellites
► 2012 (3)
► 2011 (7)
► 2010 (12)
► 2009 (29)
► 2008 (39)
► 2007 (5)

ABOUT ME
TRAVIS GOODSPEED
VIEW MY COMPLETE PROFILE