SIP: Why open is better

OpenFortress*
digital signatures

the potential of sip

* IETF: A plethora of options and extensions

* Users: ‘I just want to make phone calls’

* Telcos: ‘Not all phones support feature X’

* Phone manufacturers: ‘Most telcos do not use feature X’

SIP has potential, but nobody is unleashing it!

open implementations are better

* Softphones are full of features (ZRTP, IPv6, . . .)

* Softphones aim to innovate (presence, chat, . . .)

* Joining forces makes *economical* sense

* Open source is great at handling SIP functionality

* But these are all softphones. . .

  . . . present firmware remains *as open as an oyster*

prying open sip firmware

* OpenFortress is now building open source SIP firmware

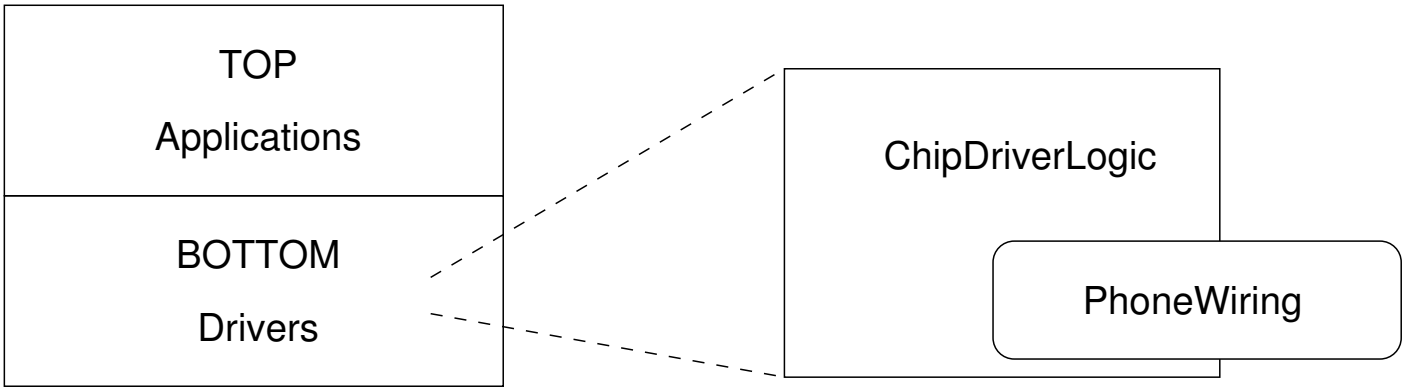
* Thanks to NLnet for partial financial support

* Proof of concept: Grandstream BT200

* Framework designed for portability

* Deliberately GPL-licensed

architecture of 0cpm firmerware



* Linux is too big :'-(

* Some phones have only 256 kB RAM

* The bottom could run on top of Linux, if desired

top application can be varied

The ‘top’ part of the architecture can be one of many:

* SIP phone

* Doorbell

* Alarm clock

* Bootloader

* Mini test-targets to aid porting

ipv6 anywhere

We drop IPv4 and, along with it, NAT.
$\Rightarrow$ direct calls are *always* possible
$\Rightarrow$ ENUM and ITAD et al will work!

Establishing an IPv6 address anywhere?

* First try stateless autoconfiguration

* If so instructed, use DHCPv6

* In absense of IPv6, use DHCPv4

  $\rightarrow$ Setup a device-local tunnel for IPv6 access

network protocols

We use Logical Link Control for local traffic
$\Rightarrow$ flies under the 'Internet radar'
$\Rightarrow$ hard target for remote assaults

DEST	SRC	<1501	LLC	Payload

* type/len $\leq$ 1500 selects IEEE 802.2 instead of 802.3

* LLC1 is a trivial `SOCK_DGRAM` service

* LLC2 is a trivial `SOCK_STREAM` service

console over llc2

* Connect to the console over LLC2 instead of TCP

* Minimal requirements: memory, network, booting code

* Adds about 200 lines of C code

* Shows logs even *before* getting an IP address

* A generally useful tool for reverse engineering

bootloader over llc1

* Access flash memory or its partitions

* Use TFTP over LLC1 instead of over UDP

* Minimal requirements: memory, network, booting code

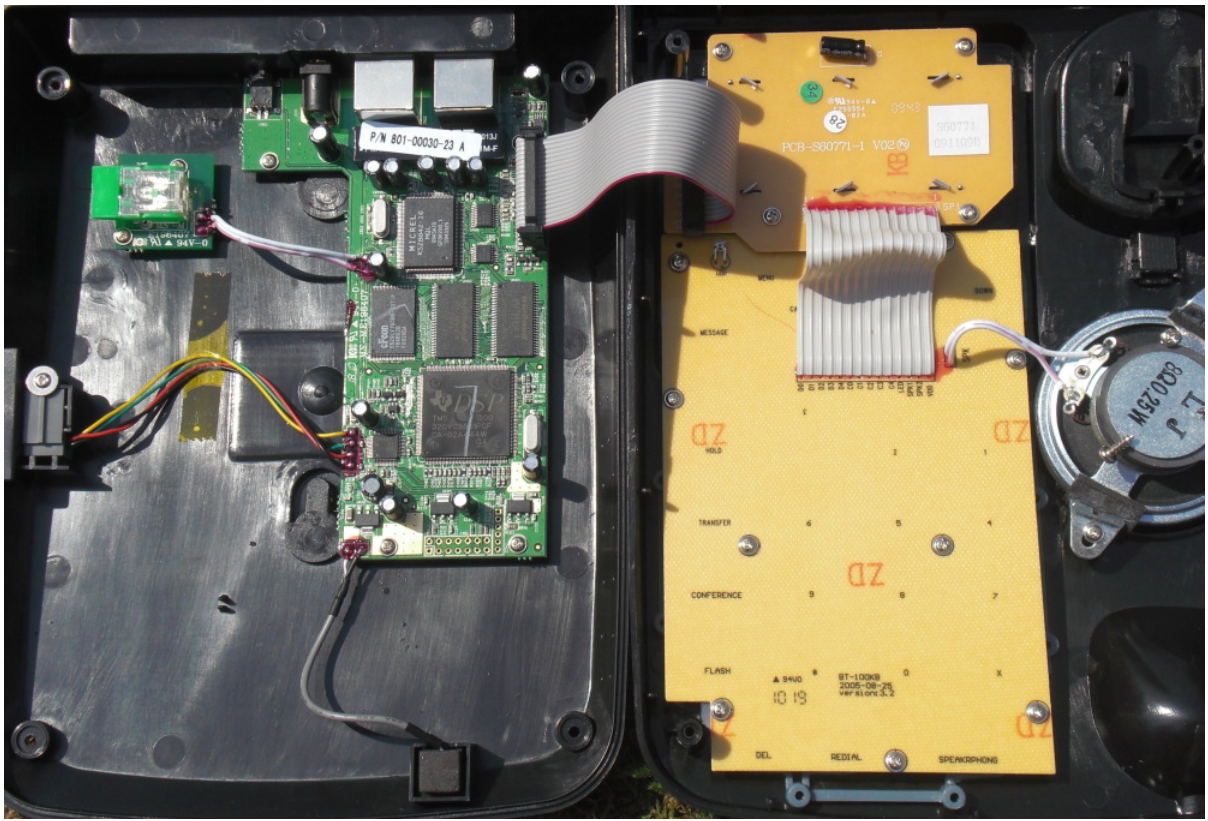
* Stand-alone image is about 10 kB (includes console)

* First program run after reset, exit when phone is on-hook

* A generally useful tool for reverse engineering

phone hardware

Most phones contain: SoC, RAM, Flash, Ethernet, GPIO.



getting started

* Scrape off paint of concealed COTS chips

* Gather datasheets of all components

* Find an open source compiler chain

* Gut feeling: how would you have designed this beast?

* Multimeter: figure out connector capabilities

* Multimeter: trace component connectivity

* Be creative: find a way to launch your own code

ways of launching your own code

* Look for pin headers — they often offer developer access
* Upgrade: new firmware images (little control here!)
* Serial port: often gives access to a boot loader
* JTAG: read/write Flash without host cooperation
* Vendor: booting over serial, I^2C, ISP, . . .

build the porting applications

You can build ‘top’ applications dedicated to driver testing:

* GPIO: hook switch, LED

* Timers, interrupts: flashing LED

* Keys, display: type and show

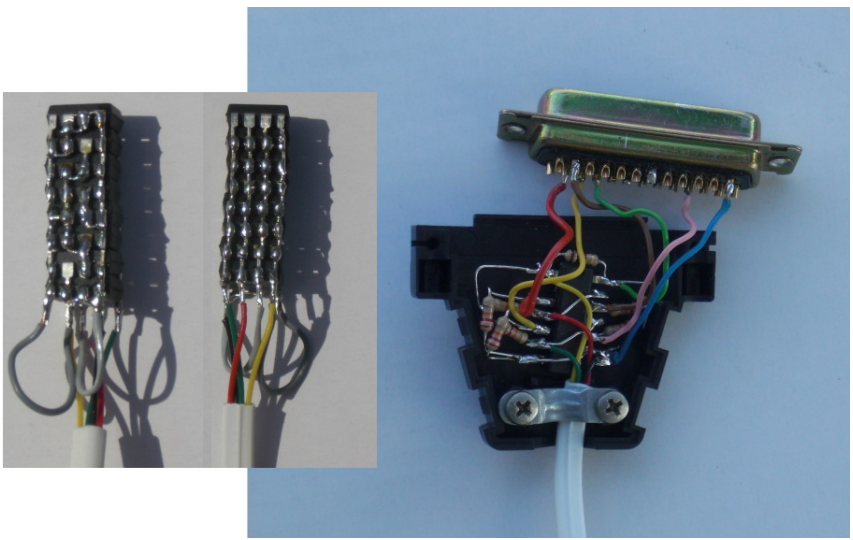
* Netconsole: Ethernet logging of some traffic

* Echo: Sound chip (‘codec’)

This lets you develop and test various ‘bottom’ drivers individually.

use your imagination: boot floppy

* Booting from 8 I^2C EEPROMs of 64 kB
* Gray code, so A0/A1/A2 without crossings
* Use `i2c-parport` as a Linux driver

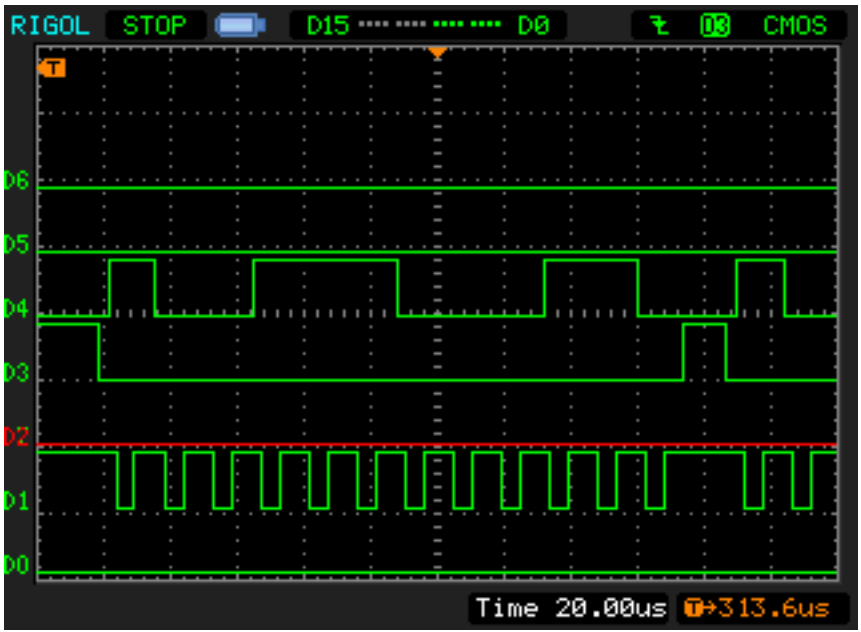


use your imagination: display driver

* Display drivers are often concealed
* BT200 mentioned: WR, RD, CS, DATA
* Based on these names, Holtek HT162x popped up
* Logic analyser + datasheet proved me right
* Logic analyser showed initiation rituals ;-)
* The x in HT162x was determined from LCD size, and no crystal

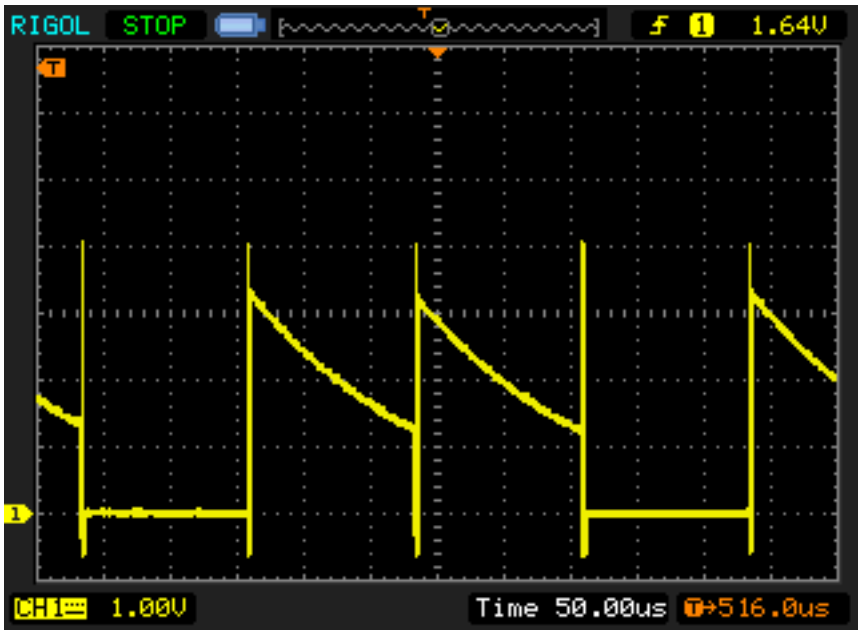
use your imagination: display driver

Shown: D1=WR, D3=CS, D4=DATA



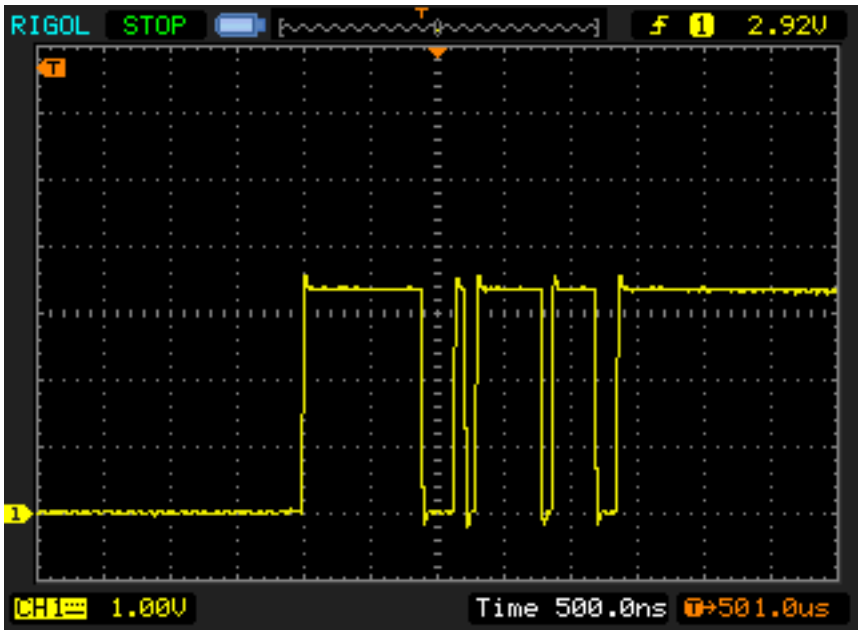
use your imagination: analog effects

What do you think is going on this serial line to the codec?



use your imagination: analog effects

The line floats after sending 16 bits, so others can grab a timeframe:



in summary

* SIP is in desparate need for IPv6 and ZRTP

* Progess in SIP will come from open source firmware

* The 0cpm project is building that

* Current demo-phone is Grandstream BT200

* Anyone is invited to port to their own hardware!

http://reverse.0cpm.org/

info@openfortress.nl

http://openfortress.nl

OpenFortress*

digital signatures