DIY Hardware implant over I2C

Part of the NSA Playset

Josh Datko and Teddy Reed

DEF CON 2Ƨ

August 10, 2014

Outline

NSA Playset Series

What is the NSA Playset?

We hope the NSA Playset will make cutting edge security tools more accessible, easier to understand, and harder to forget.

NSA Playset Talks

RF Retroreflector	Penn & Teller	Friday	12:00
DIY Hardware Implant	**Track 1**	**Sunday**	**11:00**
GSM Sniffing	Track 1	Sunday	12:00
PCIe	Track 2	Sunday	14:00

Inspired by the NSA

The NSA apparently has a hardware hacking catalog.[1]

Flip. . . Flip. . . Flip. . .

> *Oh look honey, there's an I^2C controller board we can get. It attaches to a computer and it's modular, so you can add a GSM cell phone for exfil.*

That's nice dear.

> *I wonder how that works. . .*

[1] like SkyMall for spies and without the Bigfoot.

Requirements for the implant

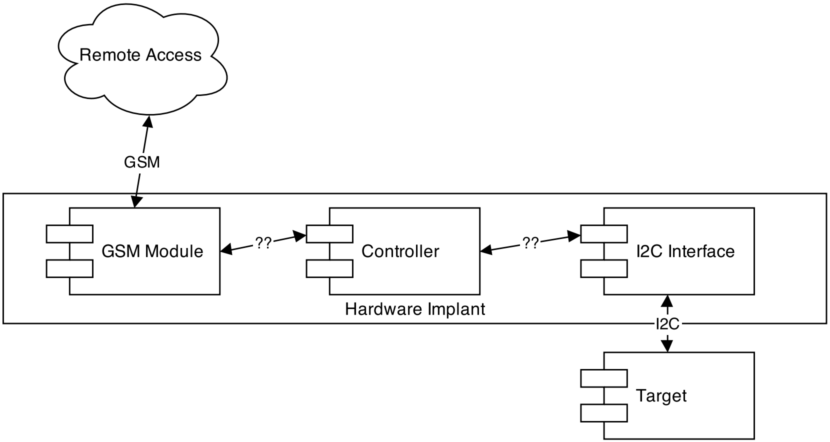
From the docs:

- Must attach over I^2C to the target.
- Must include GSM reachback to the implant.

Our requirements:

- Easy to use.
- Open Source Hardware.
- Flexible: Allow for multiple communication and software protocols.
- Fun. Single chip solutions aren't as fun.

Implant Control Diagram

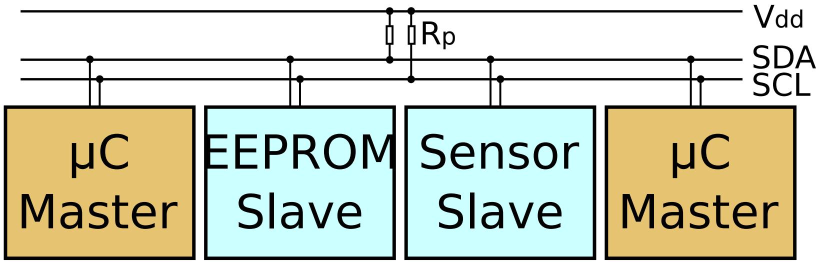


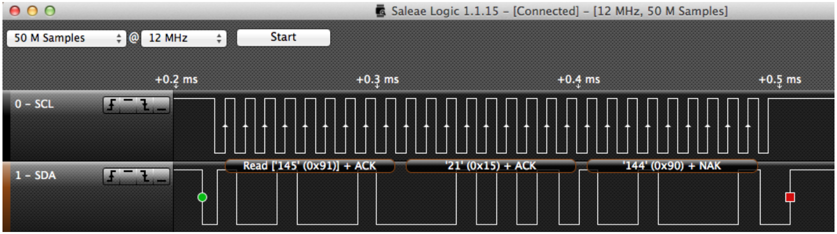
Background: What is I^2C

- Serial bus.
- Two-wires: (plus power and ground).[2]
  - Data: SDA
  - Clock: SCL
- Multi-master.
- Multi-slave.
- Addressable.
- Standard speed is 100kHz (100kbps). High Speed: 3.2Mbps theoretical max.

[2] Typically 5 or 3.3V

Background: I^{2}C in visual form





Read ['145' (0x91)] + ACK
'21' (0x15) + ACK
'144' (0x90) + NAK

I^{2}C attack surfaces

- RAM EEPROMs
- PCI and PCIe
- Battery controllers
- Video . . .

Video I^2C

Why is there I^2C on your monitor adapter?

How does your computer "automatically detect" monitor resolution?

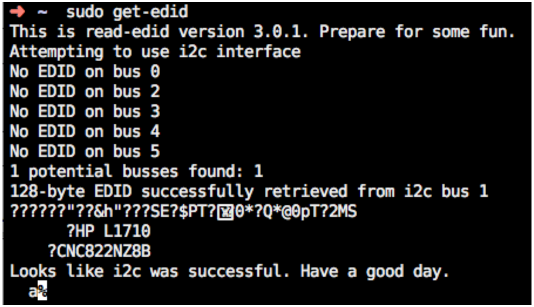
EDID

Extended Display Identification Data

DDC

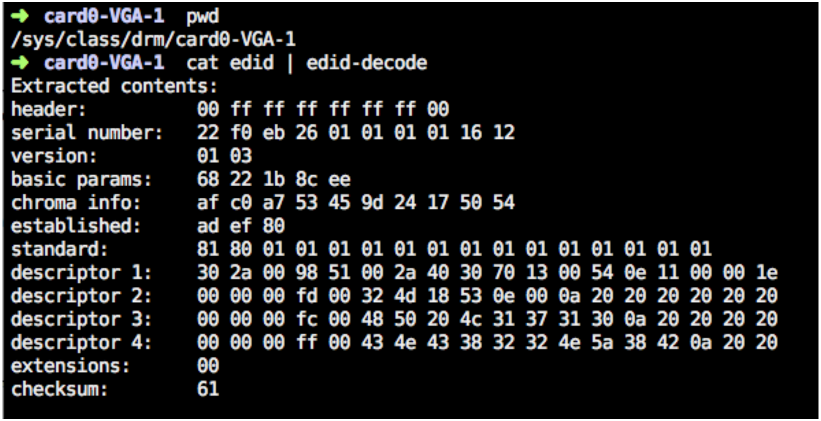
Data Display Channel, a.k.a. 5V I^2C

EDID



```
➜ ~  sudo get-edid
This is read-edid version 3.0.1. Prepare for some fun.
Attempting to use i2c interface
No EDID on bus 0
No EDID on bus 2
No EDID on bus 3
No EDID on bus 4
No EDID on bus 5
1 potential busses found: 1
128-byte EDID successfully retrieved from i2c bus 1
??????"??&h"???SE?$PT?⊠0*?Q*@0pT?2MS
      ?HP L1710
    ?CNC822NZ8B
Looks like i2c was successful. Have a good day.
  a%
```

$ edid-decode



```
➜ card0-VGA-1 pwd
/sys/class/drm/card0-VGA-1
➜ card0-VGA-1 cat edid | edid-decode
Extracted contents:
header:          00 ff ff ff ff ff ff 00
serial number:   22 f0 eb 26 01 01 01 01 16 12
version:         01 03
basic params:    68 22 1b 8c ee
chroma info:     af c0 a7 53 45 9d 24 17 50 54
established:     ad ef 80
standard:        81 80 01 01 01 01 01 01 01 01 01 01 01 01 01 01
descriptor 1:    30 2a 00 98 51 00 2a 40 30 70 13 00 54 0e 11 00 00 1e
descriptor 2:    00 00 00 fd 00 32 4d 18 53 0e 00 0a 20 20 20 20 20 20
descriptor 3:    00 00 00 fc 00 48 50 20 4c 31 37 31 30 0a 20 20 20 20
descriptor 4:    00 00 00 ff 00 43 4e 43 38 32 32 4e 5a 38 42 0a 20 20
extensions:      00
checksum:        61
```

```
ioreg -lw0 -r -c "IODisplayConnect"
```

EDID Extension Blocks

Tag Number	Extension Block Description
00h	Timing Extension
02h	CEA-EXT: CEA 861 Series Extension
10h	VTB-EXT: Video Timing Block Extension
20h	EDID 2.0 Extension
40h	DI-EXT: Display Information Extension
50h	**LS-EXT: Localized String Extension**
60h	DPVL-EXT: Digital Packet Video Link Extension
A7h, AFh, BFh	DTCDB-EXT: Display Transfer Characteristics
F0h	EXTENSION Block Map
FFh	**EXTENSIONS defined by the OEM**

Parsing implemented by the OS-supplied VESA driver or GPU driver manufacturer.

Exploiting EDID/EDID Extension parsing

Hacking Displays Made Interesting
Blackhat EU 2012
Andy Davis - NGS Secure
`https://github.com/nccgroup/EDIDFuzzer`

Simple adaptation for BeagleBone
Implemented in Python (BBIO)
`https://github.com/theopolis/bone-edidfuzzer`

Discover proprietary EDID extensions! Moar fuzzing!
Or assume a-priori software control...

I^2C everywhere IC^3

A video card may have multiple I^2C buses and devices. NVIDIA cards may have I^2C for the following:

- EEPROM for encrypted HDCP keys
- Onboard voltage regulator
- Thermal sensor
- TV decoder chip (older cards)

[3] C'mon, it's punny

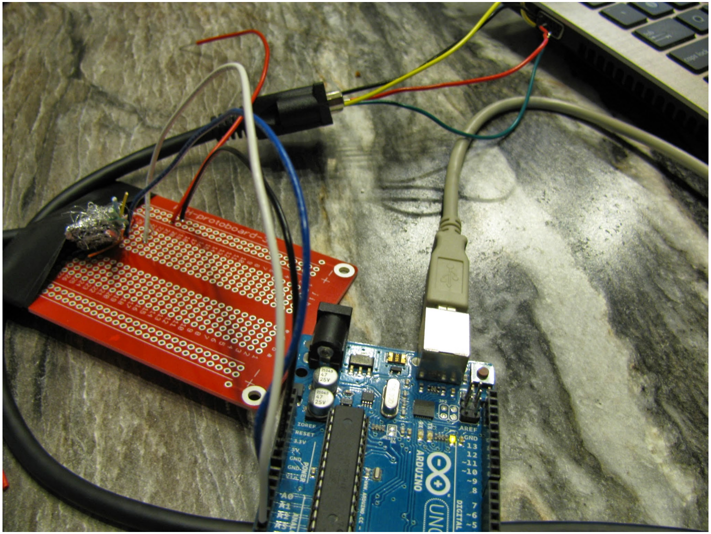
Exploring VGA I^2C

Let's start exploring our attack surface.

Pin	Name	Description
1	RED	Red Video
2	GREEN	Green Video
3	BLUE	Blue Video
⋮	⋮	⋮
5	**GND**	**Ground**
9	**KEY**	**Optional +5V output from graphics card**
12	**SDA**	**I2C data**
15	**SCL**	**I2C data clock**

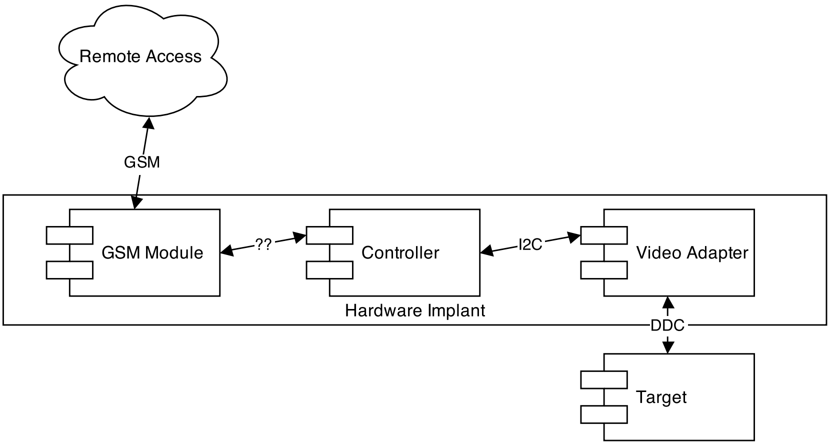
VGA Pinout

I want my I^{2}C [4]



[4]Dire Straights fans, anyone?

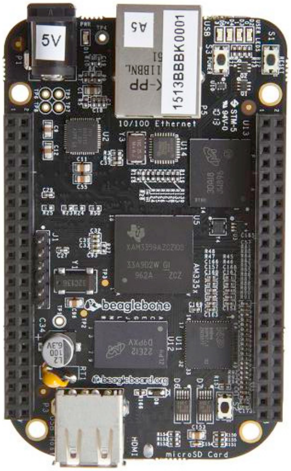
Filling in the details



Controller Selection

BeagleBone Black is the embedded hacker's friend:

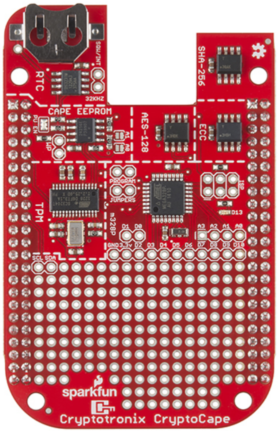
- 1GHz AM3358 ARM® Cortex-A8
- 512MB DDR3 RAM
- Two independent Programmable Real-Time Units (32bit)
- Crypto accelerators for AES, SHA, MD5
- UARTs, PWM, LCD, GPMC, SPI, ADC, CAN, Timers
- **Two I^2C buses**



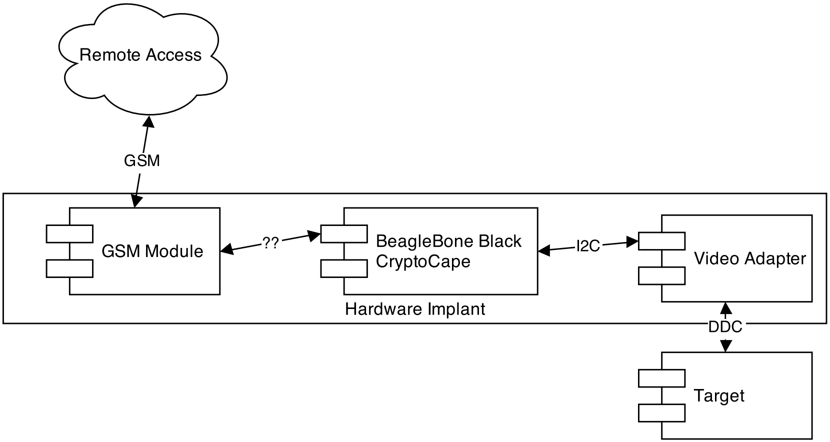
CryptoCape

The BBB ecosystem enables easy hardware expansion with Capes. Let's add some hardware crypto and a micro:

- Authenticators: ECC & MAC (SHA256)
- Encrypted EEPROM (AES-128-CCM)
- Battery backed up Real-time clock
- **Trusted Platform Module**
- **ATmega328p**, all sorts of handy. Plus it's a programmable I^{2}C slave.



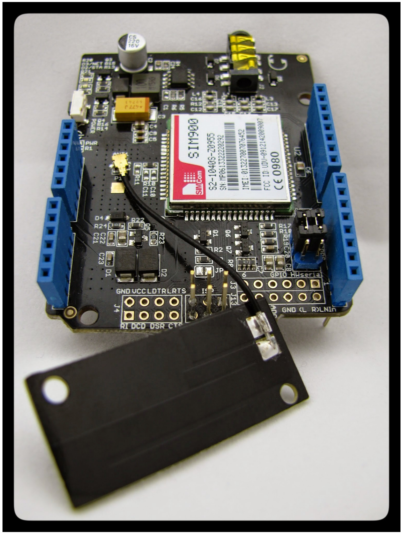
Add the controller



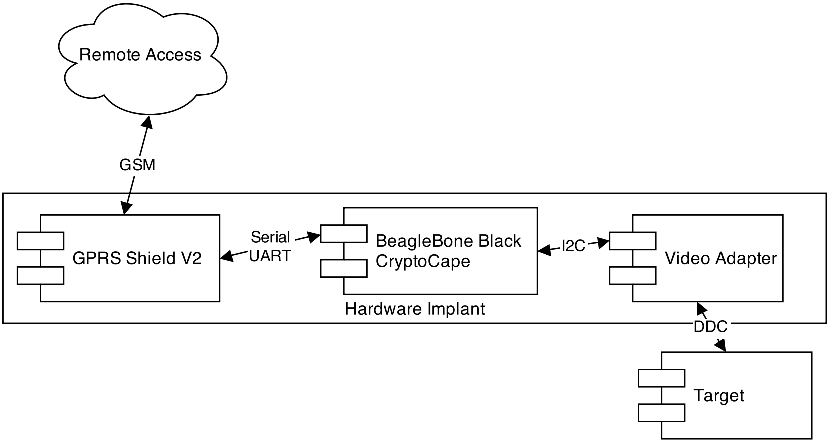
GSM Module

Seeed Studo GPRS Shield v2:

- Arduino form factor
- GSM Quad band support
- TCP support
- SIM card holder
- Works with Tmobile, AT&T
    - You can buy pre-paid SIMs with cash.
    - T-Mobile has unlimited talk & text for 35USD.



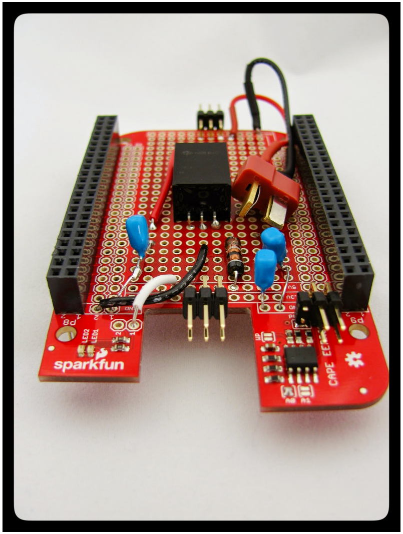
Add the GSM module



Moar Power?

- BBB draws 460mA on boot
- CryptoCape
- GSM Shield draws 300mA on average for "talk", but peak of 2.0 A!?

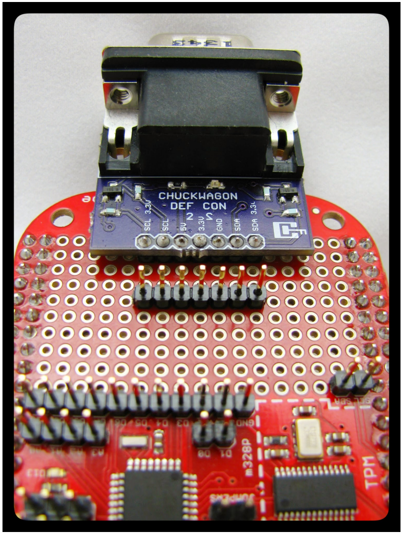
- Meet the *LiPo*WerCape
  - Switching voltage regulator with noise filtering
  - Dual cell LiPo input
  - Output to 5V Power Rail



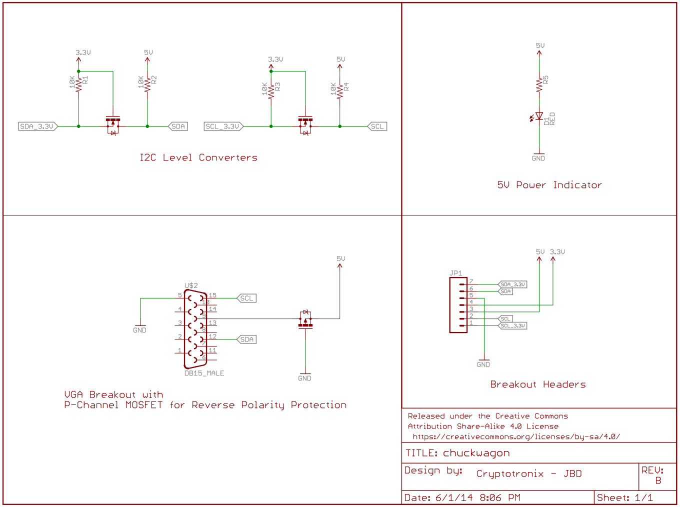
CHUCKWAGON

We still need a way to easily connect to the video adapter. Meet CHUCKWAGON:

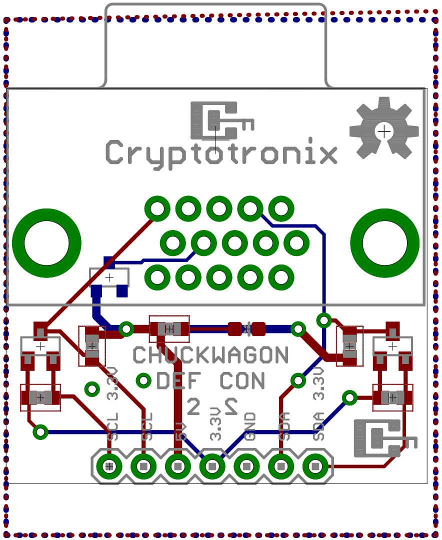
- DDC to I^{2}C converter.
- Breadboard friendly.
- Logic level converters for I^{2}C .
- Supplies 5V from target (not on all VGA connectors).
- Power indicator.



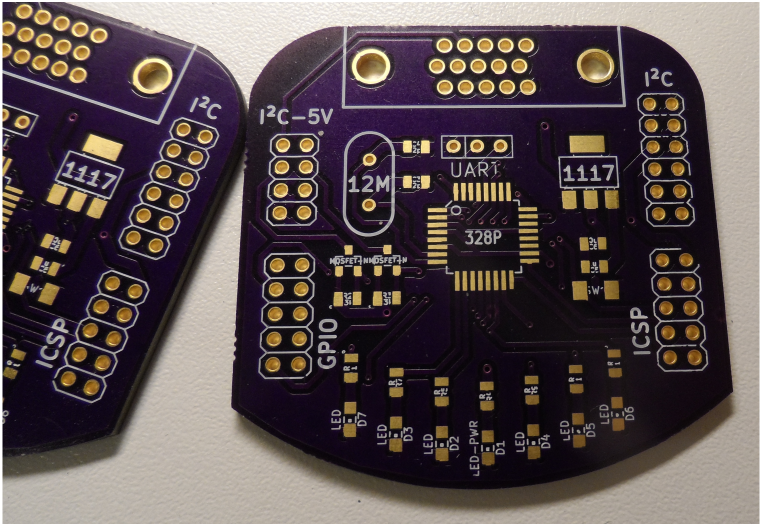
CHUCKWAGON schematic



CHUCKWAGON board

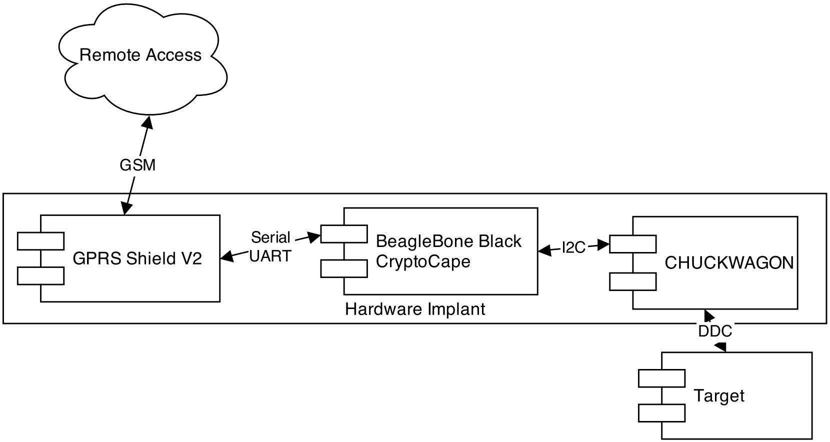


I^{2}C hack not that new...



As seen on Hackaday

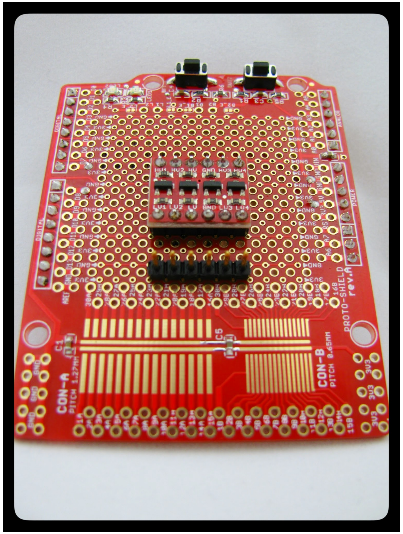
Add the CHUCKWAGON



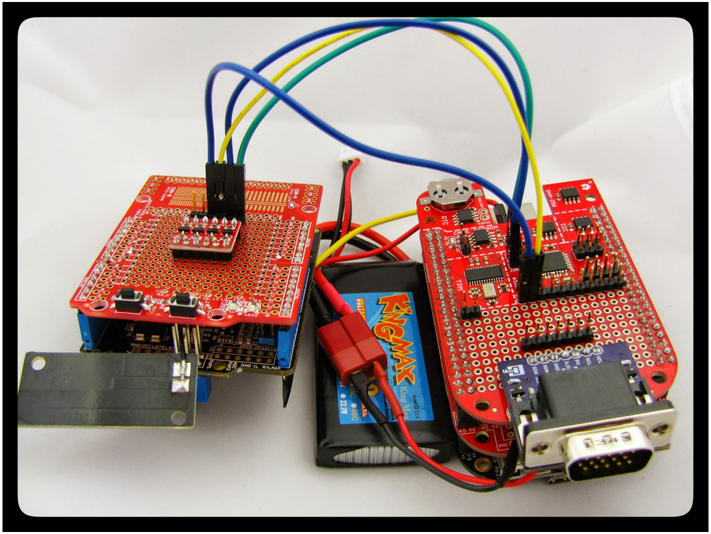
Connect to GSM module

Ok, so let's connect to the GSM Shield from the Beagle!

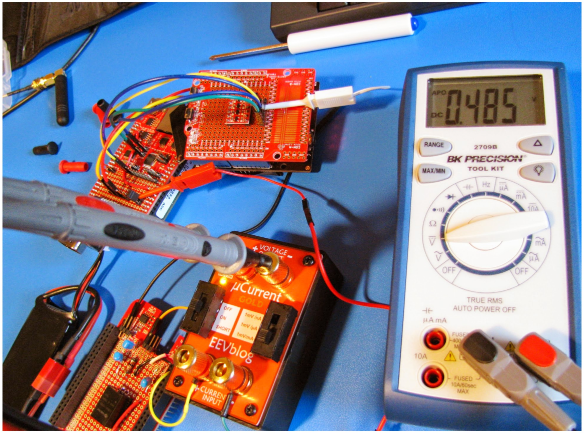
- BBB's UART4, broken-out by ATmega's program jumpers.
- GSM's shield software-serial, D7 and D8
- /me checks datasheet one last time...
- Needs logic level converters!



Completed Hardware with Battery



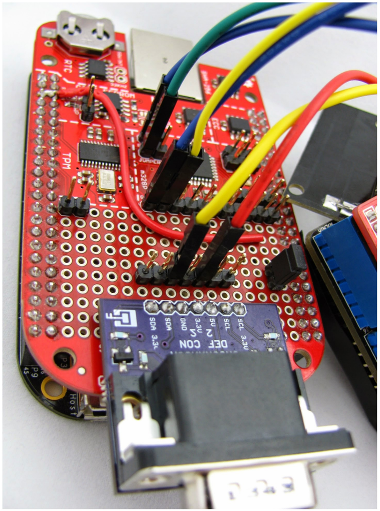
Measuring current

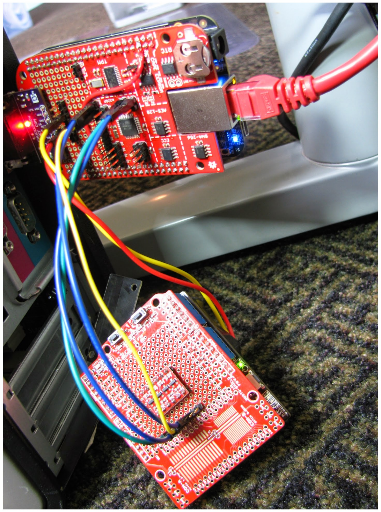


Dave Jones' μCurrent Gold

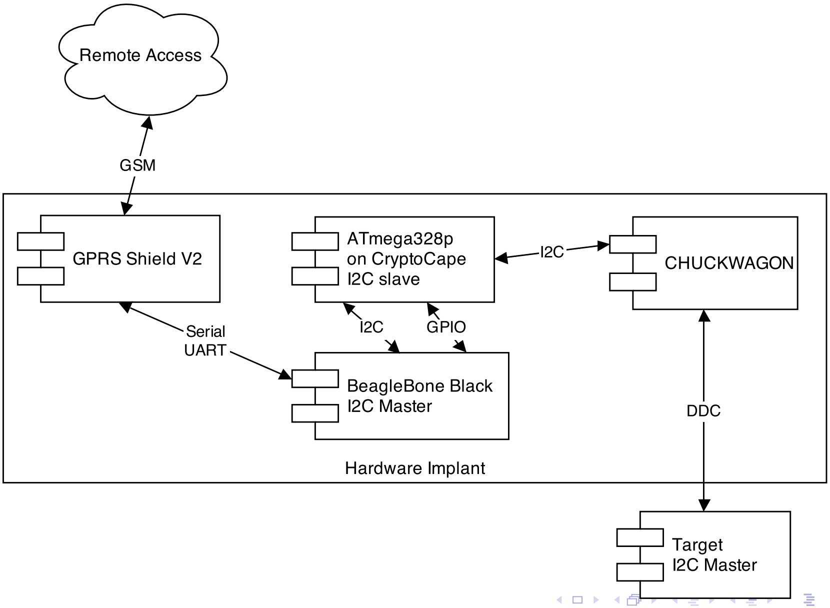
Trusted by hardware implant designers.

Completed Hardware without Battery





Software flow



Usage

1. Get malware on target.
2. Attach CHUCWAGON for exfil or control.

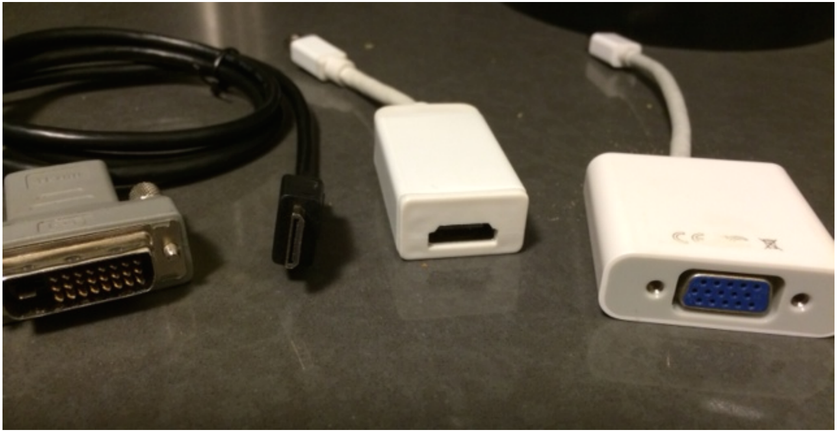
If software on the target can communicate with the implant then:

- Target can exfil out to implant to GSM.
- Target can exfil out to implant for storage.
- Implant can provide code for target to run.
- Control the implant over GSM $\rightarrow$ control the target over GSM

Why is this significant?

I^{2}C via the video adapter is an always on, bi-directional bus on every laptop, PC, or server.

Accessorize!



Prepared for anything or NSA hacking toolkit?

How to improve the CHUCKWAGON

What does CHUCKWAGON rev. B look like?

- Consolidate into one board: *ImplantCape*
- HDMI footprint vs. VGA
- Could all be done from AVR (less power), but BBB is more fun and provides more options.
- **VGA Tap**.
  - Combine with SALSAFLOCK for a implant **plus** RF retroreflector.

Using Crypto for Evil!

Long history of Cryptography and Malware!

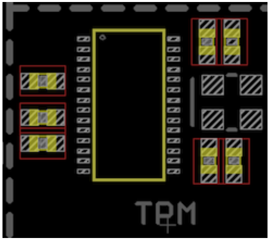
Cryptoviral Extortion:

- 1989 PC Cybord, Joseph Popp
- 1996 Macintosh SE/30 crytovirus PoC, Young and Yung
- 2006 Gpcode.AG/AK, Cryzip
- 2013 CryptoLocker, CryptorBit

Reversing Anti-Analysis:

- Packers, Obfuscator, VM-based JIT
- 2011 TPM "cloaking" malware
- 2014 Uroburos, encrypted VFS
- **2014 TPM-enabled super-targeted malware**

Using Crypto for Evil!



The CryptoCape includes a TPM...

- I^2C friendly
- Protected RSA private key storage
- Windows 8 friendly
- More or less optional, as there is most likely an onboard TPM

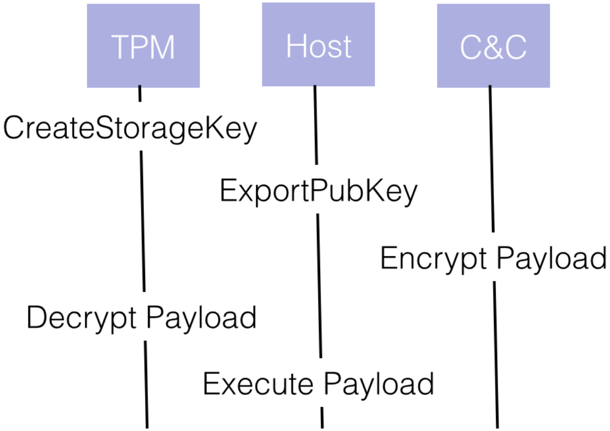
Cloaking Malware with the Trusted Platform Module

2011 USENIX Security
Alan M. Dunn, Owen S. Hofmann, Brent Waters, Emmett Witchel
Summary: Use TPM-protected keys and an Intel TXT PAL to protect malicious code execution from observation, analysis, and tamperment.

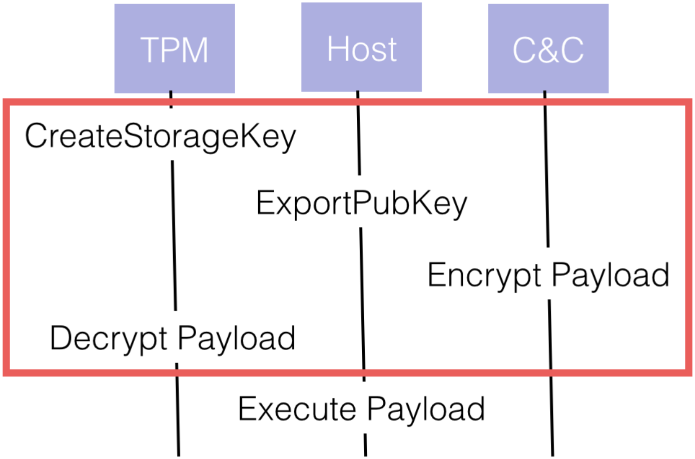
Intel TXT and remote attestation are hard!
But generating a public key on a TPM and using that to encrypt additional payloads is easy...

Put a TPM on your implant and protect against nasty network interception. Also restrict analysis to the target machine upon discovery (or force memory analysis).

TPM-enabled super-targeted Malware



TPM-enabled super-targeted Malware



TPM-enabled super-targeted Malware

Windows 8 automatically enables/initializes a TPM, then creates and manages your owner password. Access to TPM is abstracted through Microsoft CSP.

Windows `PcpTool` Kit:

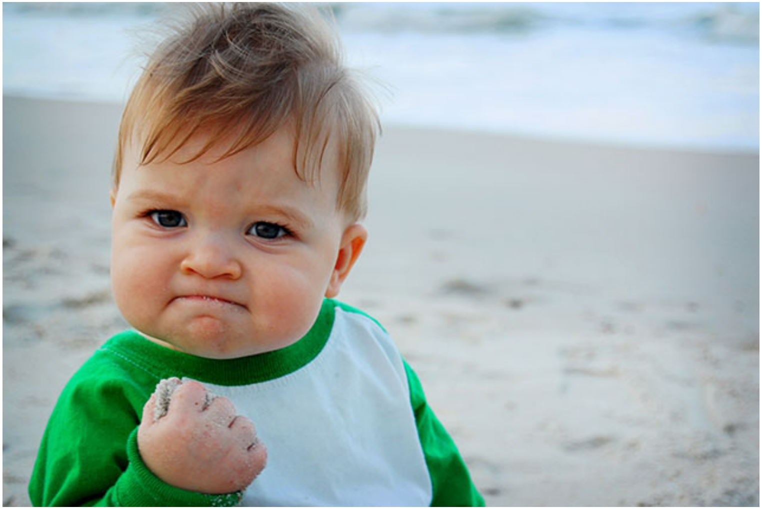
```
NCryptOpenStorageProvider
NCryptCreatePersistedKey
NCryptExportKey
NCryptDecrypt
```

In memory process creation:

```
CreateProcess
ZwUnmapViewOfSection
VirtualAllocEx
WriteProcessMemory
```

Python `pefile` to inject encrypted PE section into a decryption stub.

TPM-enabled super-targeted Malware



tpm-malcrypt

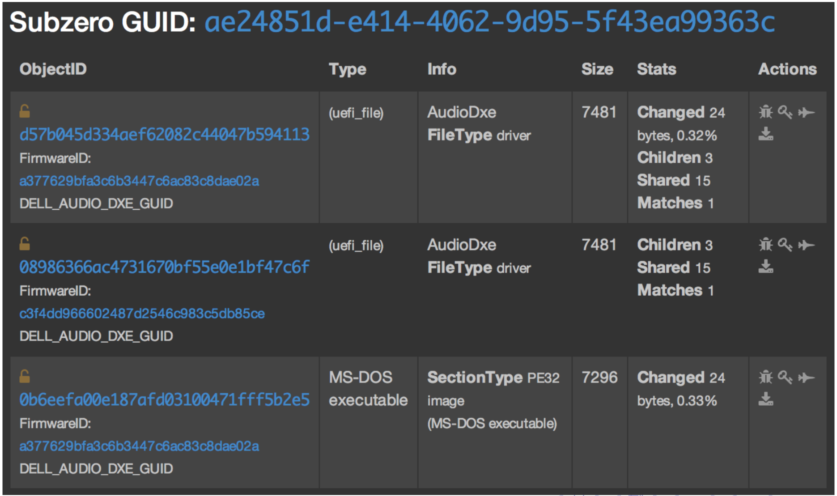
fork tpm-malcrypt!

`https://github.com/theopolis/tpm-malcrypt`

- `tpm-keyextract`, create and exfil a storage public key
- `malcrypter`, encrypt and inject into decryption stub
- `malcrypt`, decryption stub, process creation/injection

Malicious Exfiltration via Audio

Backstory: **#badBIOS** thought to use Audio as an out-of-band exfiltration or C&C mechanism. Dismissed as infeasable by BIOS development SMEs.



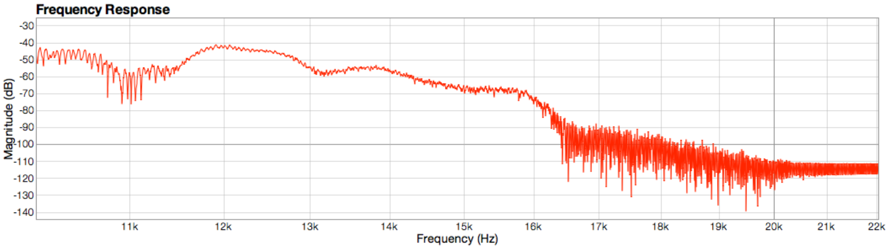
Subzero GUID: ae24851d-e414-4062-9d95-5f43ea99363c

ObjectID	Type	Info	Size	Stats	Actions
d57b045d334aef62082c44047b594113 FirmwareID: a377629bfa3c6b3447c6ac83c8dae02a DELL_AUDIO_DXE_GUID	(uefi_file)	AudioDxe **FileType** driver	7481	**Changed** 24 bytes, 0.32% **Children** 3 **Shared** 15 **Matches** 1	
08986366ac4731670bf55e0e1bf47c6f FirmwareID: c3f4dd966602487d2546c983c5db85ce DELL_AUDIO_DXE_GUID	(uefi_file)	AudioDxe **FileType** driver	7481	**Children** 3 **Shared** 15 **Matches** 1	
0b6eefa00e187afd03100471fff5b2e5 FirmwareID: a377629bfa3c6b3447c6ac83c8dae02a DELL_AUDIO_DXE_GUID	MS-DOS executable	**SectionType** PE32 image (MS-DOS executable)	7296	**Changed** 24 bytes, 0.33%	

Malicious Exfiltration via Audio

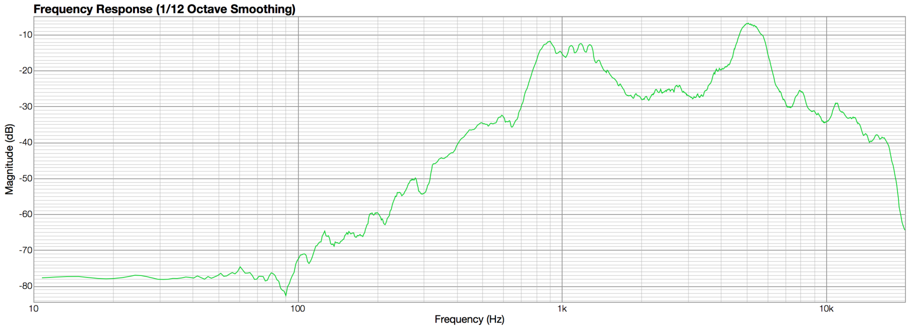
Data of Audio Protocols are very well defined and resiliant.

QPSK10 (10 baud), **QPSK05** (5 baud), quadrature phase shift keying modulation to provide forward error correction.



Malicious Exfiltration via Audio

Possible to "pivot" through colluding machines. Local network exploitation creates a mesh of audio-capable relays such as idle headphones.



Demos, Learning, and Fabulous Prizes

Join us in the HHV for CryptoCape and WAGONBED demos!

Challenge: Solve the puzzle here:
`theopolis.github.io/tpm-malcrypt/challenge.html`

The first 5 correct submissions win a DIY hardware implant kit
(No hardware hacking experience required)

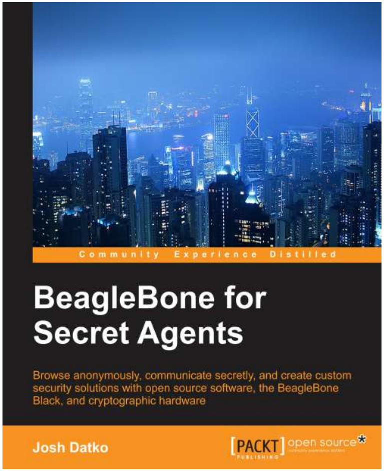
Demos, Learning, and Fabulous Prizes

Thank you!

Upcoming Book

Preorder with code: **BBSAeB** at `packtpub.com`.

- Setting up a Tor bridge and building custom front panel.
- Two factor authentication with a Fingerprint scanner and the CryptoCape
- Using the TPM to protect GPG keys
- Running an IRC gateway with BitlBee, ZNC, and using OTR for protected chat.

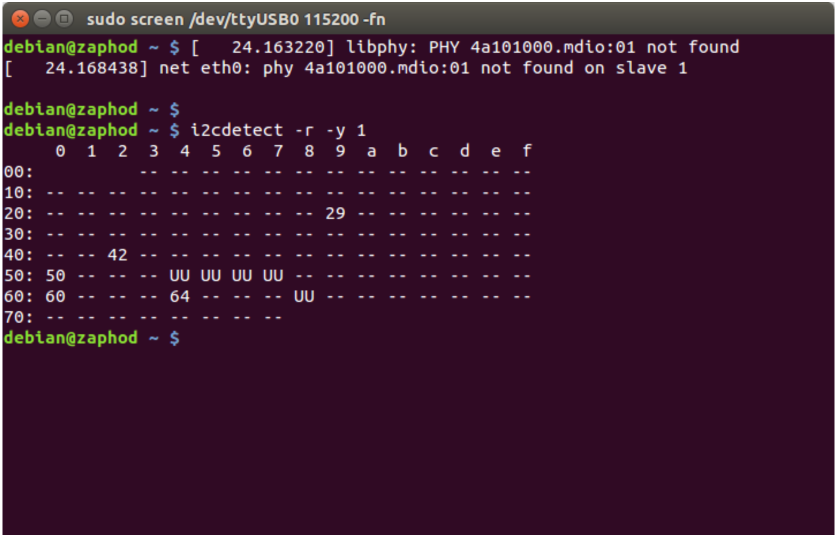

Community Experience Distilled
BeagleBone for Secret Agents
Browse anonymously, communicate secretly, and create custom security solutions with open source software, the BeagleBone Black, and cryptographic hardware
Josh Datko
[PACKT] open source
PUBLISHING

POC Code

CHUCKWAGON sketch and scripts

`https://github.com/NSAPlayset/CHUCKWAGON`

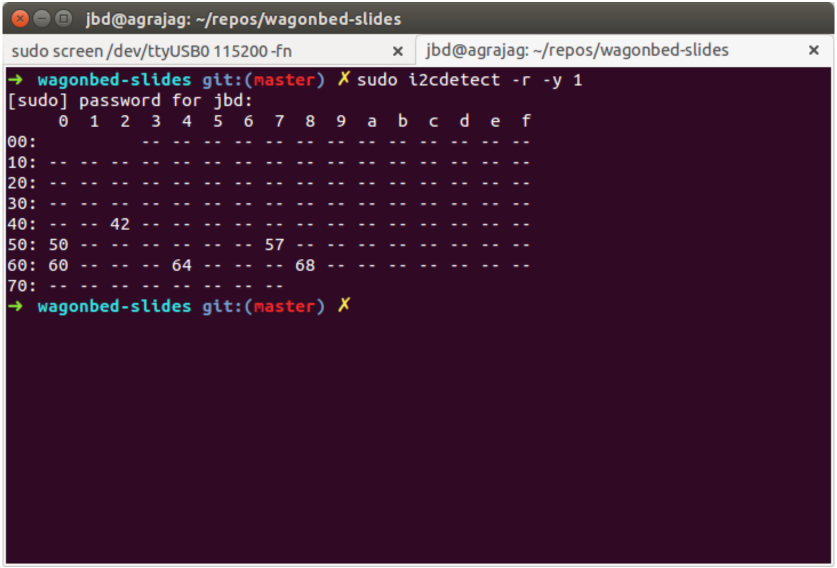
i2cdetect on BBB



sudo screen /dev/ttyUSB0 115200 -fn

```
debian@zaphod ~ $ [   24.163220] libphy: PHY 4a101000.mdio:01 not found
[   24.168438] net eth0: phy 4a101000.mdio:01 not found on slave 1

debian@zaphod ~ $
debian@zaphod ~ $ i2cdetect -r -y 1
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- 29 -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- 42 -- -- -- -- -- -- -- -- -- -- -- -- --
50: 50 -- -- -- UU UU UU UU -- -- -- -- -- -- -- --
60: 60 -- -- -- 64 -- -- -- UU -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
debian@zaphod ~ $
```

i2cdetect on target

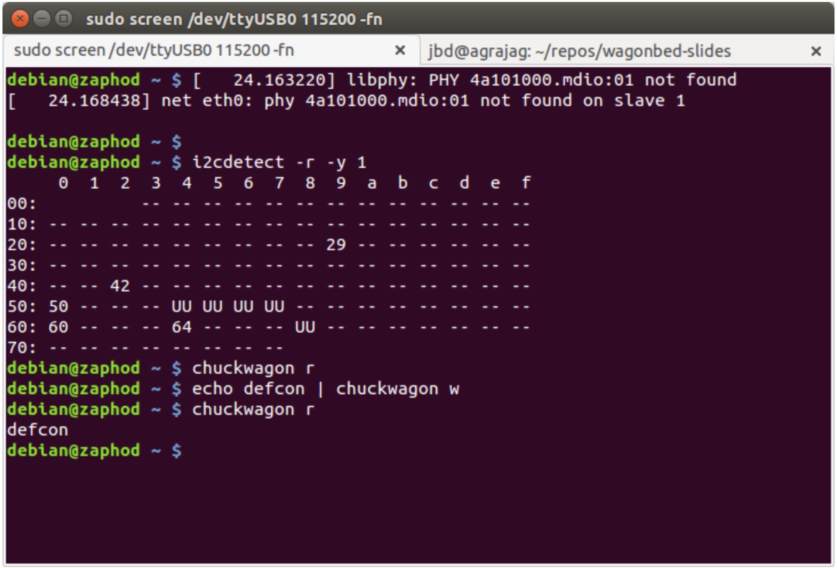


jbd@agrajag: ~/repos/wagonbed-slides

sudo screen /dev/ttyUSB0 115200 -fn | jbd@agrajag: ~/repos/wagonbed-slides

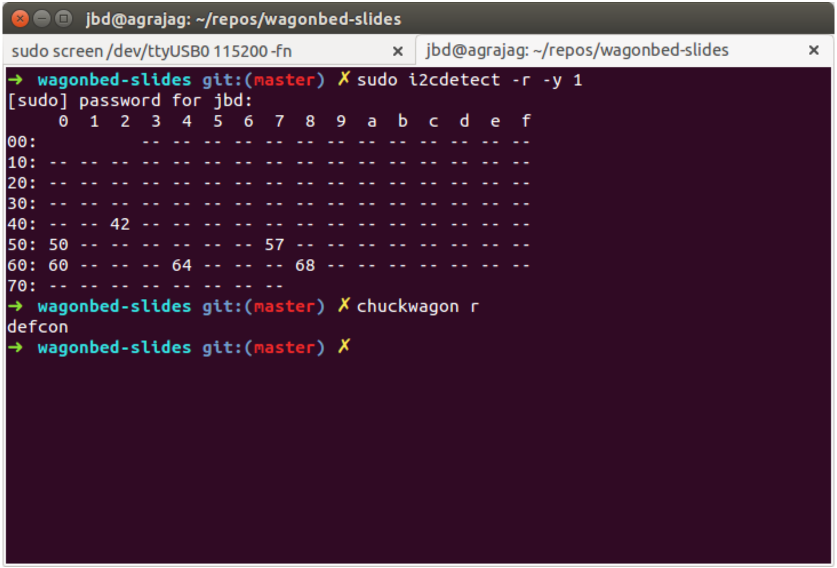
```
→ wagonbed-slides git:(master) ✗ sudo i2cdetect -r -y 1
[sudo] password for jbd:
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- 42 -- -- -- -- -- -- -- -- -- -- -- -- --
50: 50 -- -- -- -- -- -- 57 -- -- -- -- -- -- -- --
60: 60 -- -- -- 64 -- -- -- 68 -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
→ wagonbed-slides git:(master) ✗
```

chuckwagon util on BBB


sudo screen /dev/ttyUSB0 115200 -fn
sudo screen /dev/ttyUSB0 115200 -fn
jbd@agrajag: ~/repos/wagonbed-slides

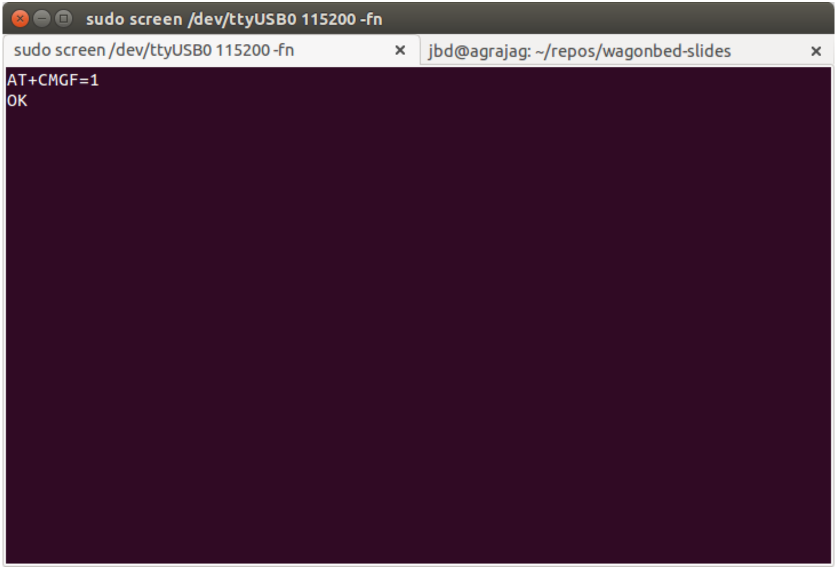
```
debian@zaphod ~ $ [   24.163220] libphy: PHY 4a101000.mdio:01 not found
[   24.168438] net eth0: phy 4a101000.mdio:01 not found on slave 1

debian@zaphod ~ $
debian@zaphod ~ $ i2cdetect -r -y 1
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- 29 -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- 42 -- -- -- -- -- -- -- -- -- -- -- -- --
50: 50 -- -- -- UU UU UU UU -- -- -- -- -- -- -- --
60: 60 -- -- -- 64 -- -- -- UU -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
debian@zaphod ~ $ chuckwagon r
debian@zaphod ~ $ echo defcon | chuckwagon w
debian@zaphod ~ $ chuckwagon r
defcon
debian@zaphod ~ $
```

chuckwagon util on target


jbd@agrajag: ~/repos/wagonbed-slides
sudo screen /dev/ttyUSB0 115200 -fn
jbd@agrajag: ~/repos/wagonbed-slides

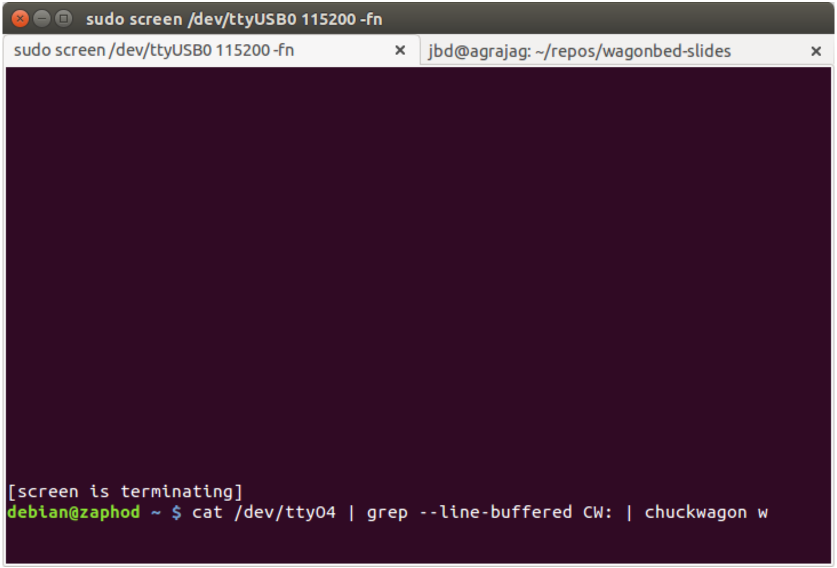
```
➜ wagonbed-slides git:(master) ✗ sudo i2cdetect -r -y 1
[sudo] password for jbd:
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- 42 -- -- -- -- -- -- -- -- -- -- -- -- --
50: 50 -- -- -- -- -- -- 57 -- -- -- -- -- -- -- --
60: 60 -- -- -- 64 -- -- -- 68 -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
➜ wagonbed-slides git:(master) ✗ chuckwagon r
defcon
➜ wagonbed-slides git:(master) ✗
```

BBB starting the GSM module


sudo screen /dev/ttyUSB0 115200 -fn
sudo screen /dev/ttyUSB0 115200 -fn
jbd@agrajag: ~/repos/wagonbed-slides

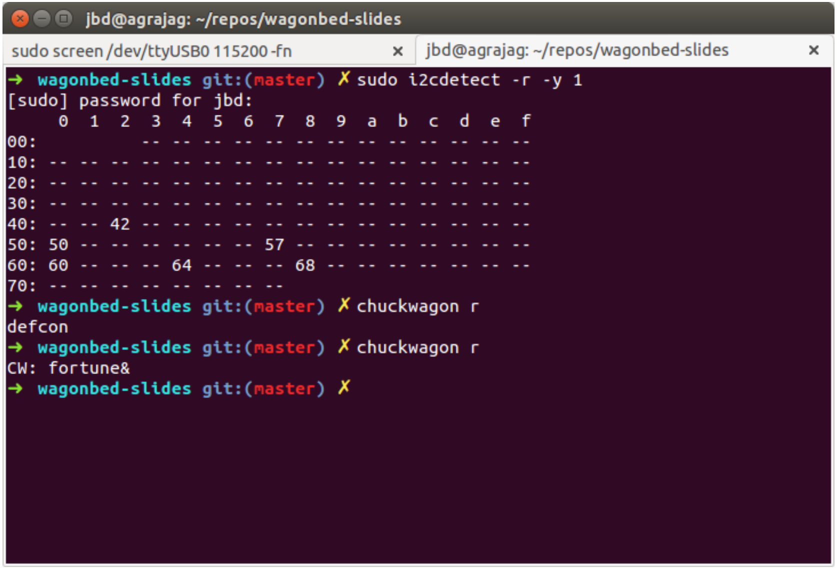
```
AT+CMGF=1
OK
```

BBB waiting on text message


sudo screen /dev/ttyUSB0 115200 -fn
sudo screen /dev/ttyUSB0 115200 -fn
jbd@agrajag: ~/repos/wagonbed-slides

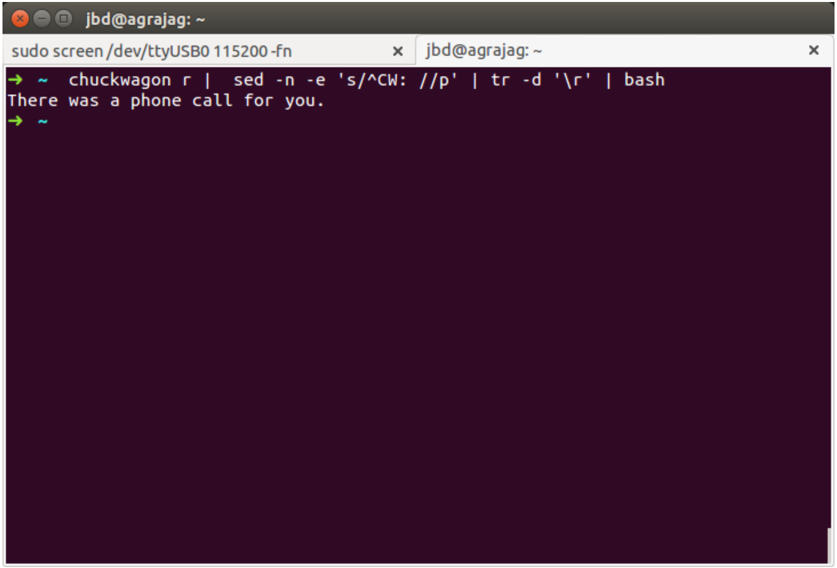
```
[screen is terminating]
debian@zaphod ~ $ cat /dev/tty04 | grep --line-buffered CW: | chuckwagon w
```

Receiving the message on the target


jbd@agrajag: ~/repos/wagonbed-slides
sudo screen /dev/ttyUSB0 115200 -fn
jbd@agrajag: ~/repos/wagonbed-slides

```
➜ wagonbed-slides git:(master) ✗ sudo i2cdetect -r -y 1
[sudo] password for jbd:
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- 42 -- -- -- -- -- -- -- -- -- -- -- -- --
50: 50 -- -- -- -- -- -- 57 -- -- -- -- -- -- -- --
60: 60 -- -- -- 64 -- -- -- 68 -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
➜ wagonbed-slides git:(master) ✗ chuckwagon r
defcon
➜ wagonbed-slides git:(master) ✗ chuckwagon r
CW: fortune&
➜ wagonbed-slides git:(master) ✗
```

Executing the text message



```
→ ~  chuckwagon r |  sed -n -e 's/^CW: //p' | tr -d '\r' | bash
There was a phone call for you.
→ ~
```