Hardware Hacking with the Beaglebone (Bl|H)ack

Joe FitzPatrick & Jeremy Richards

- focus on reverse engineering and exploit development
- 10 years of fun with software
  - vuln research
  - security patch diffing
  - exploit development
  - security training
- Hardware Security:
  - medical devices, soho routers, IoT

Jeremy Richards
@dyngnosis
jeremy@0xtech.com

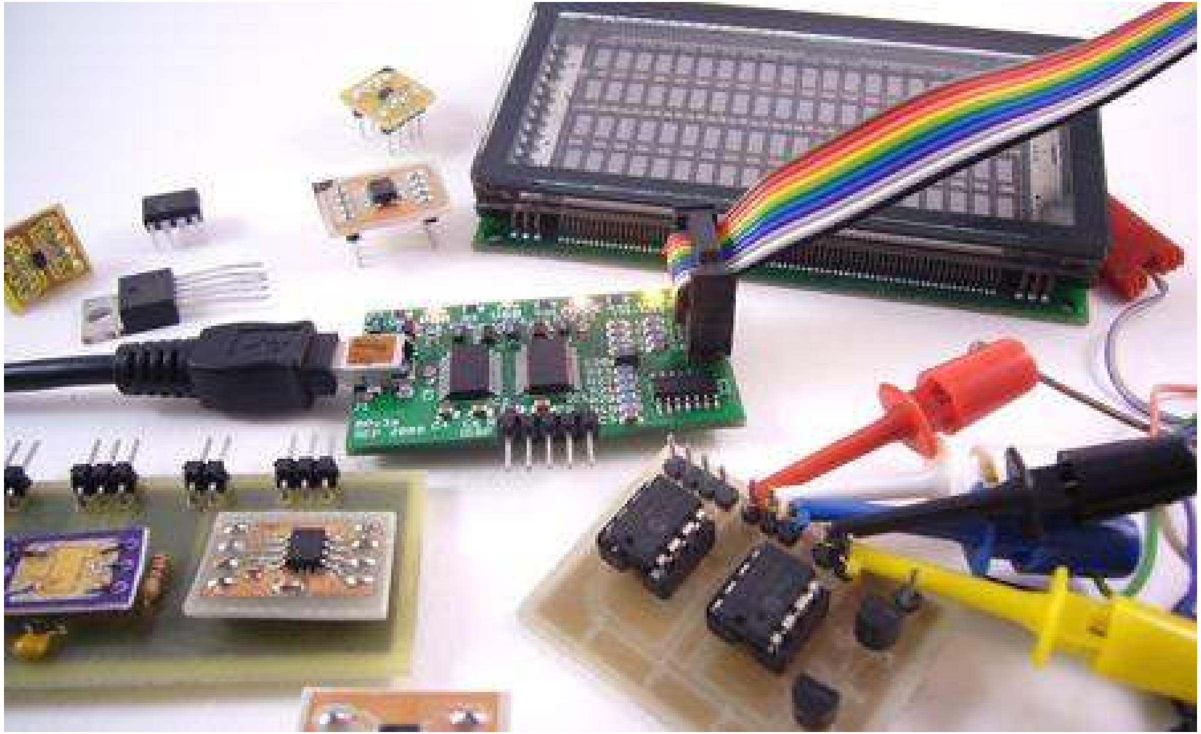
- Electrical Engineering education with focus on CS and Infosec
- 10 years of fun with hardware
  - silicon debug
  - security research
  - pen testing of CPUs
  - security training
- Hardware Security Training:
  - “Applied Physical Attacks on x86 Systems”

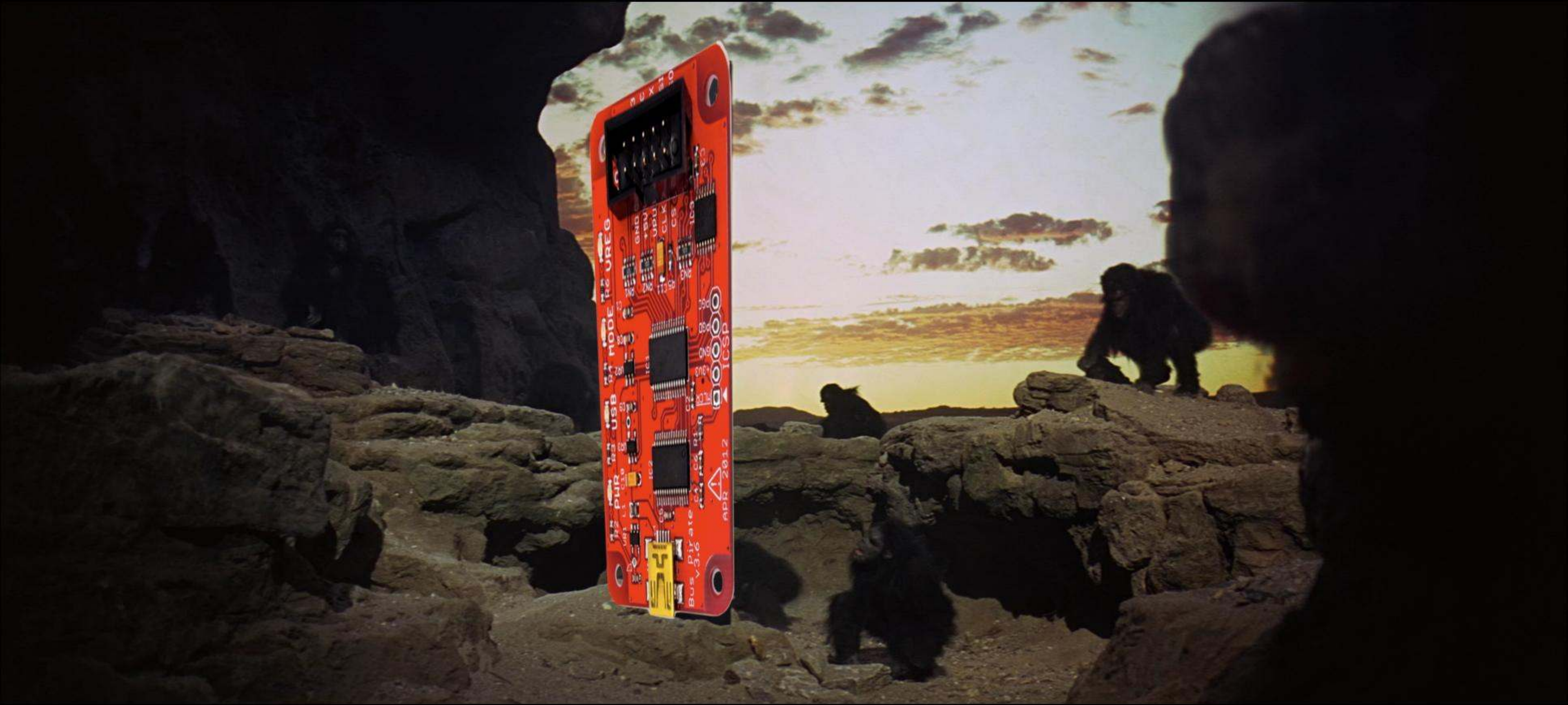
SecuringHardware.com

Joe FitzPatrick
@securelyfitz
joefitz@securinghardware.com

In the beginning, there were Vendor-supplied Proprietary tools.

Then, everyone said:
“Let’s make a low-cost, general purpose serial interface tool”



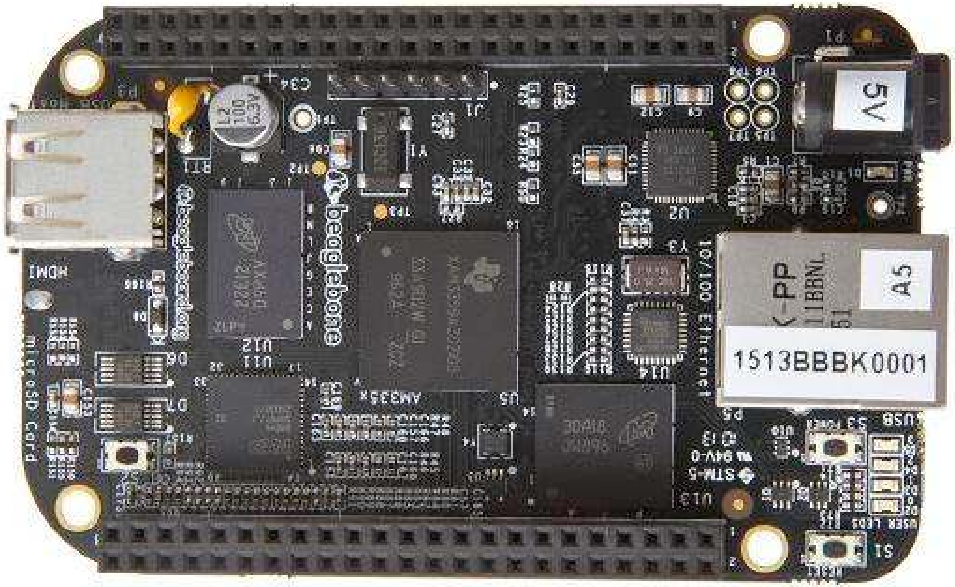

VREG
MODE
USB
PWR
GND
+5V
VPU
CLK
CS
ICSP
APR 2012
Bus Pirate
v3.6

Then, everyone looked at what they had made, and it was good

But technology moves on, and there are better specialized tools for many things...

Too many are single purpose tools (also I'm messy)

How about a new all-purpose hardware hacking tool?



Why the Beaglebone Black?

- It’s cheap!
- It’s readily available
- It runs it’s own software
- It has hardware ports for:
  - UART
  - SPI
  - I2C
  - CAN
  - and more….
- It has GPIO’s and is easy to program

Task	Pre-BusPirate $$$$, 🕒🕒🕒	Bus Pirate $$, 🕒🕒🕒🕒	Post-BusPirate $$$, 🕒🕒	Beaglebone Hack $$, 🕒
Talk UART				
Interface I2C				
Dump SPI Flash				
Analyze Logic				
JTAG				

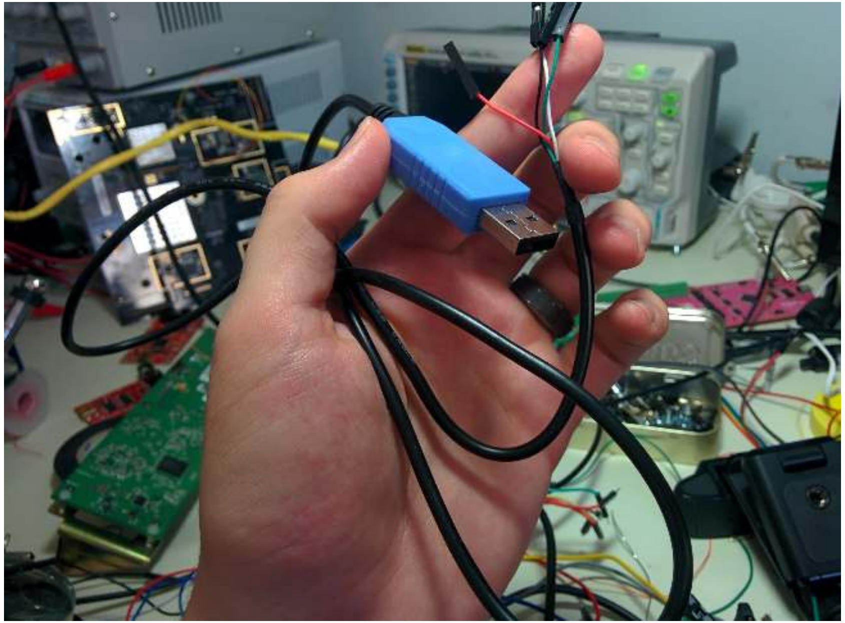
Howtos

UART FTDI Cable

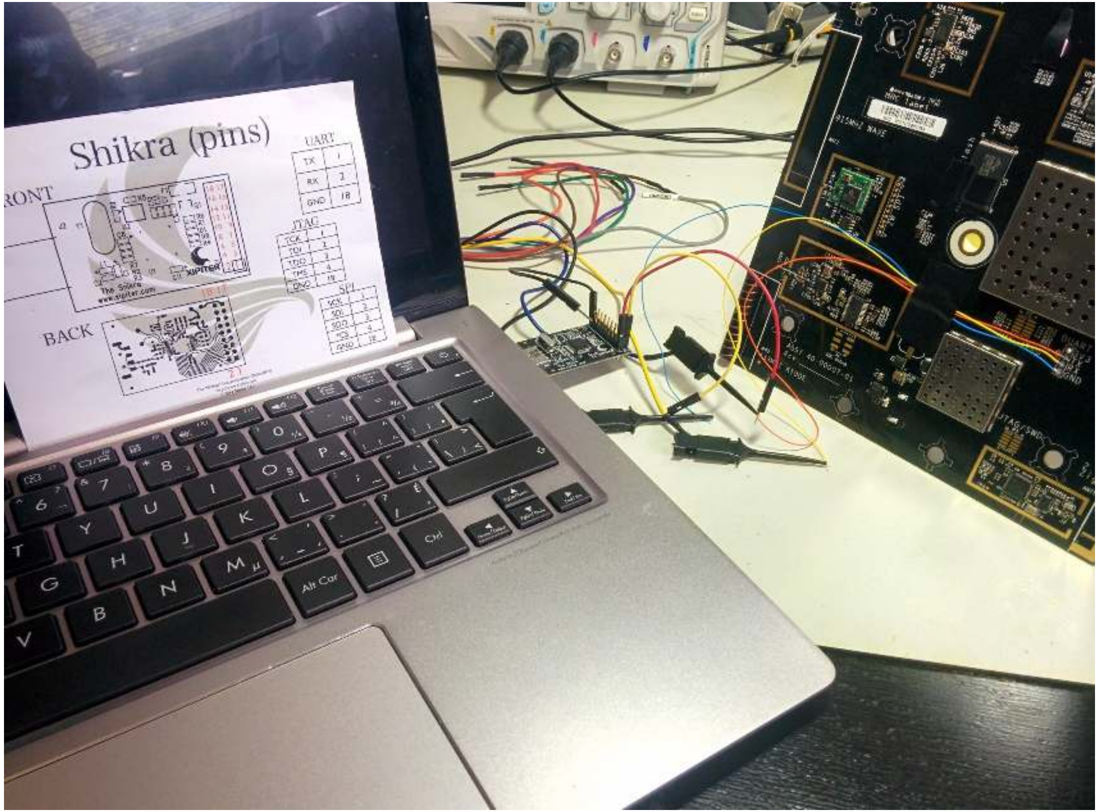
One of the the lightest weight method of getting a UART console is the FTDI Cable.

The cable requires drivers to be installed (windows) and creates a com port a terminal program is used to connect to the device.

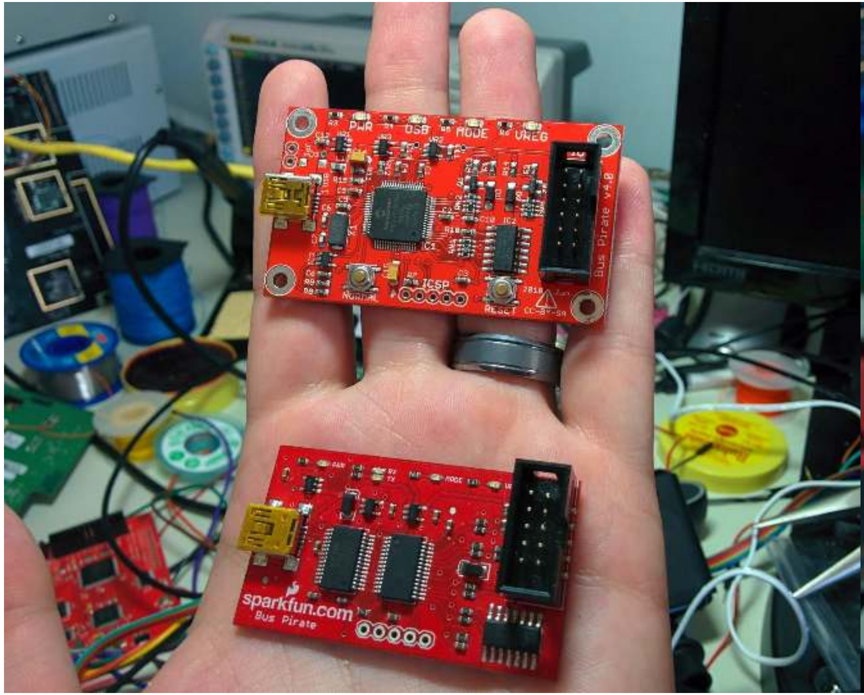
Careful not to hook that red wire up to anything important ;) (RIP)

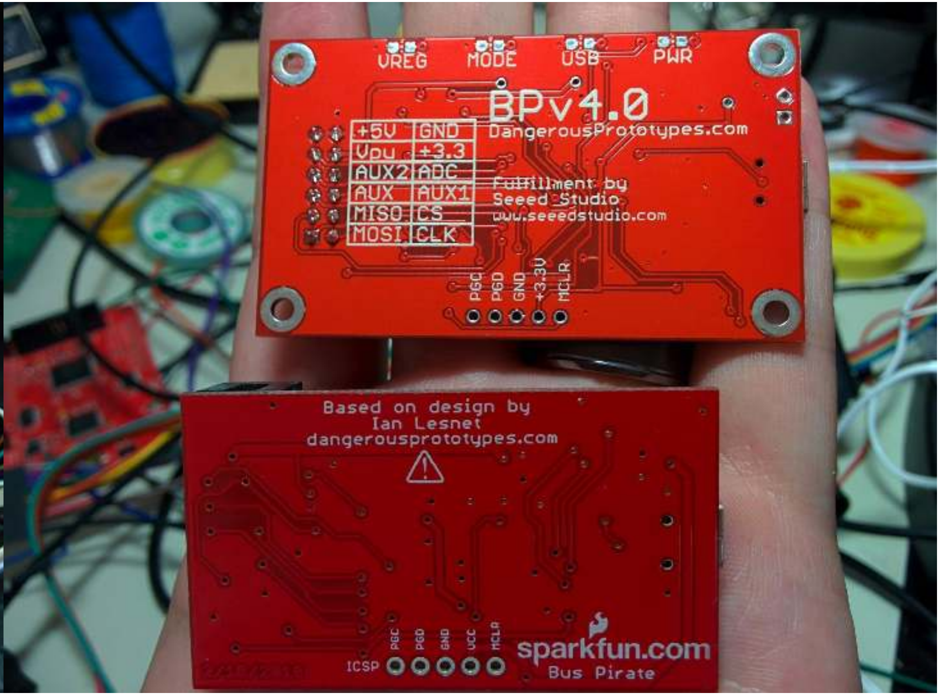


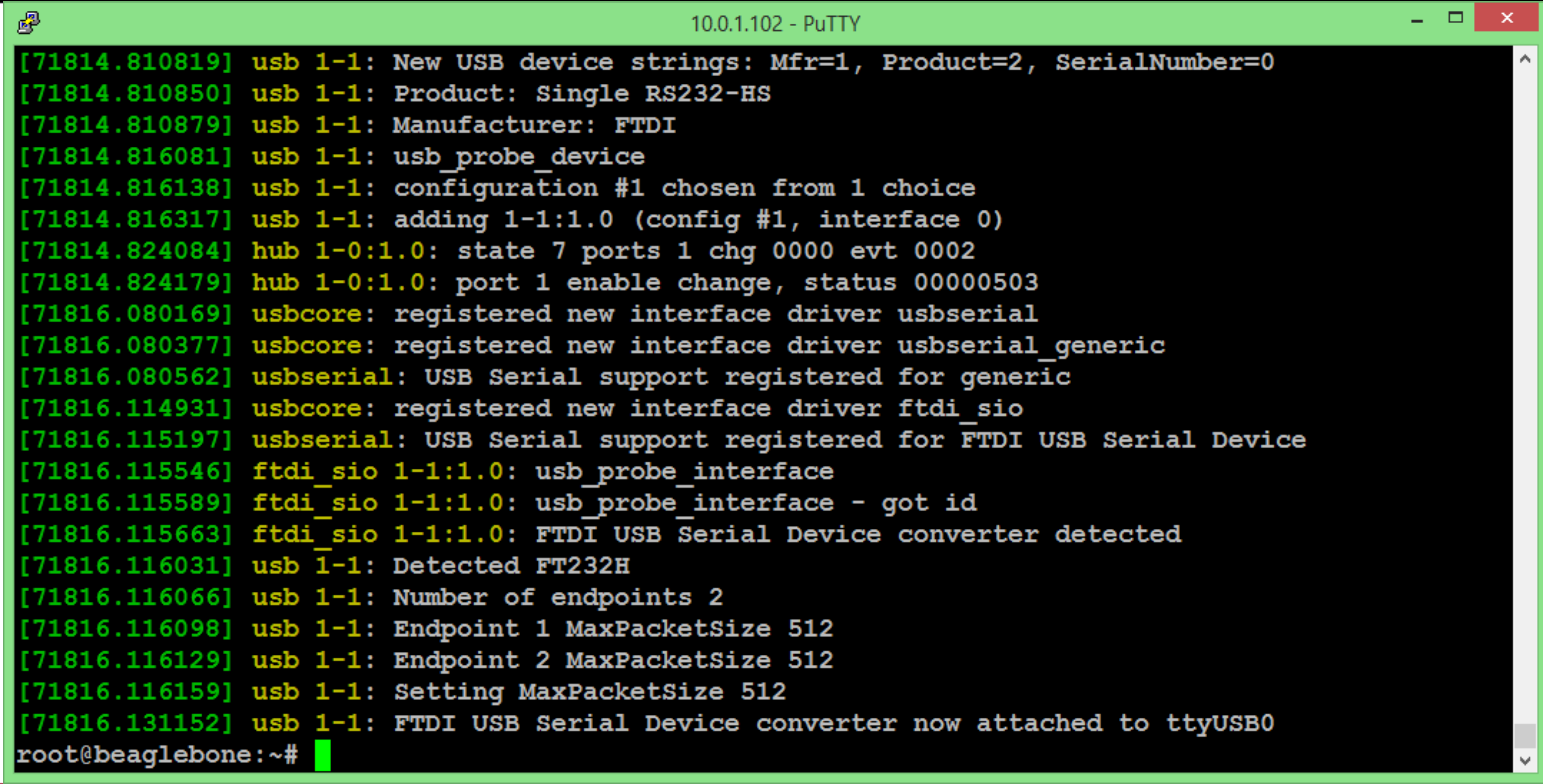
UART - Shikra & other FTDI based devices



UART - BusPirate







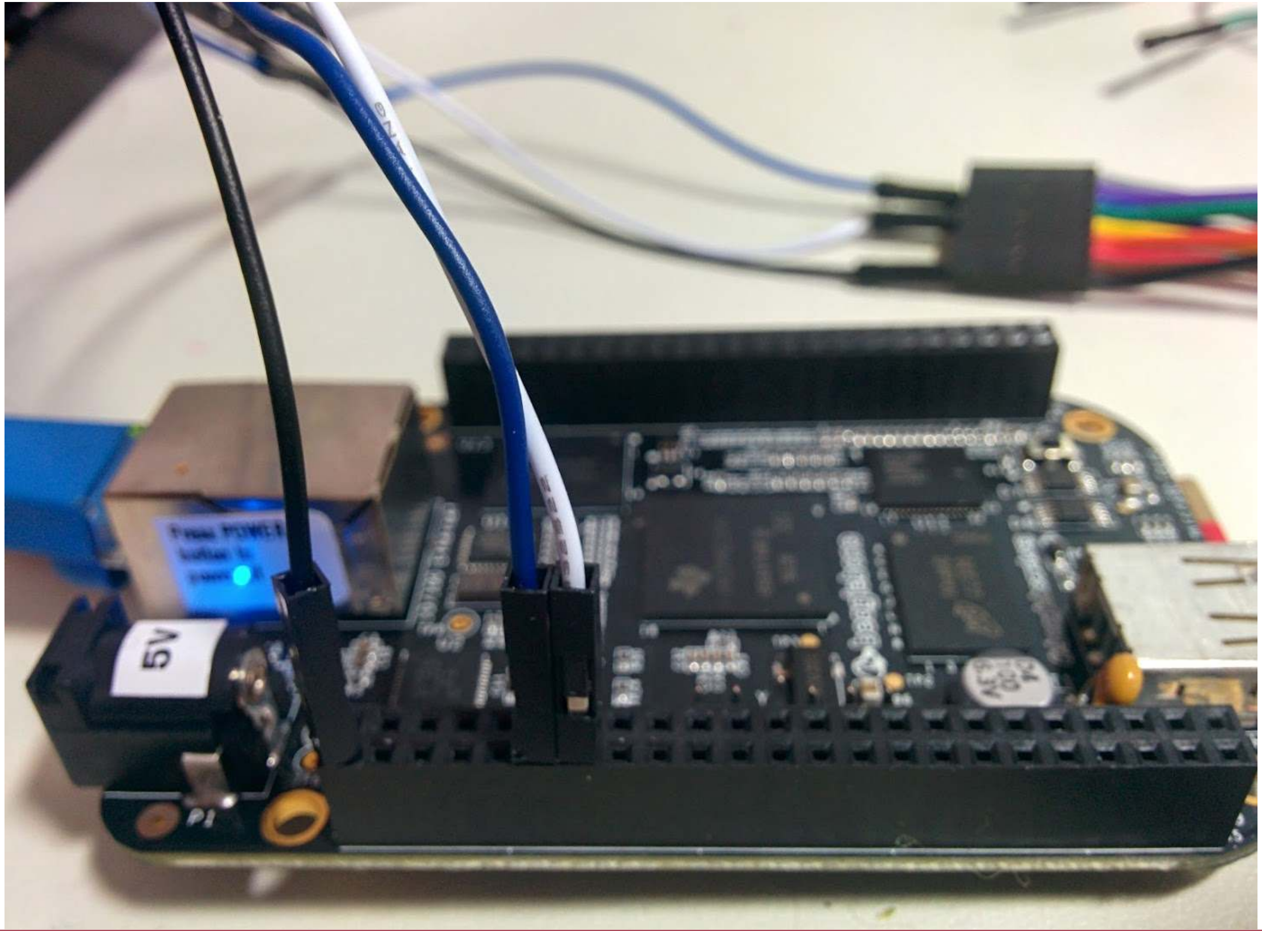
```
[71814.810819] usb 1-1: New USB device strings: Mfr=1, Product=2, SerialNumber=0
[71814.810850] usb 1-1: Product: Single RS232-HS
[71814.810879] usb 1-1: Manufacturer: FTDI
[71814.816081] usb 1-1: usb_probe_device
[71814.816138] usb 1-1: configuration #1 chosen from 1 choice
[71814.816317] usb 1-1: adding 1-1:1.0 (config #1, interface 0)
[71814.824084] hub 1-0:1.0: state 7 ports 1 chg 0000 evt 0002
[71814.824179] hub 1-0:1.0: port 1 enable change, status 00000503
[71816.080169] usbcore: registered new interface driver usbserial
[71816.080377] usbcore: registered new interface driver usbserial_generic
[71816.080562] usbserial: USB Serial support registered for generic
[71816.114931] usbcore: registered new interface driver ftdi_sio
[71816.115197] usbserial: USB Serial support registered for FTDI USB Serial Device
[71816.115546] ftdi_sio 1-1:1.0: usb_probe_interface
[71816.115589] ftdi_sio 1-1:1.0: usb_probe_interface - got id
[71816.115663] ftdi_sio 1-1:1.0: FTDI USB Serial Device converter detected
[71816.116031] usb 1-1: Detected FT232H
[71816.116066] usb 1-1: Number of endpoints 2
[71816.116098] usb 1-1: Endpoint 1 MaxPacketSize 512
[71816.116129] usb 1-1: Endpoint 2 MaxPacketSize 512
[71816.116159] usb 1-1: Setting MaxPacketSize 512
[71816.131152] usb 1-1: FTDI USB Serial Device converter now attached to ttyUSB0
root@beaglebone:~#
```

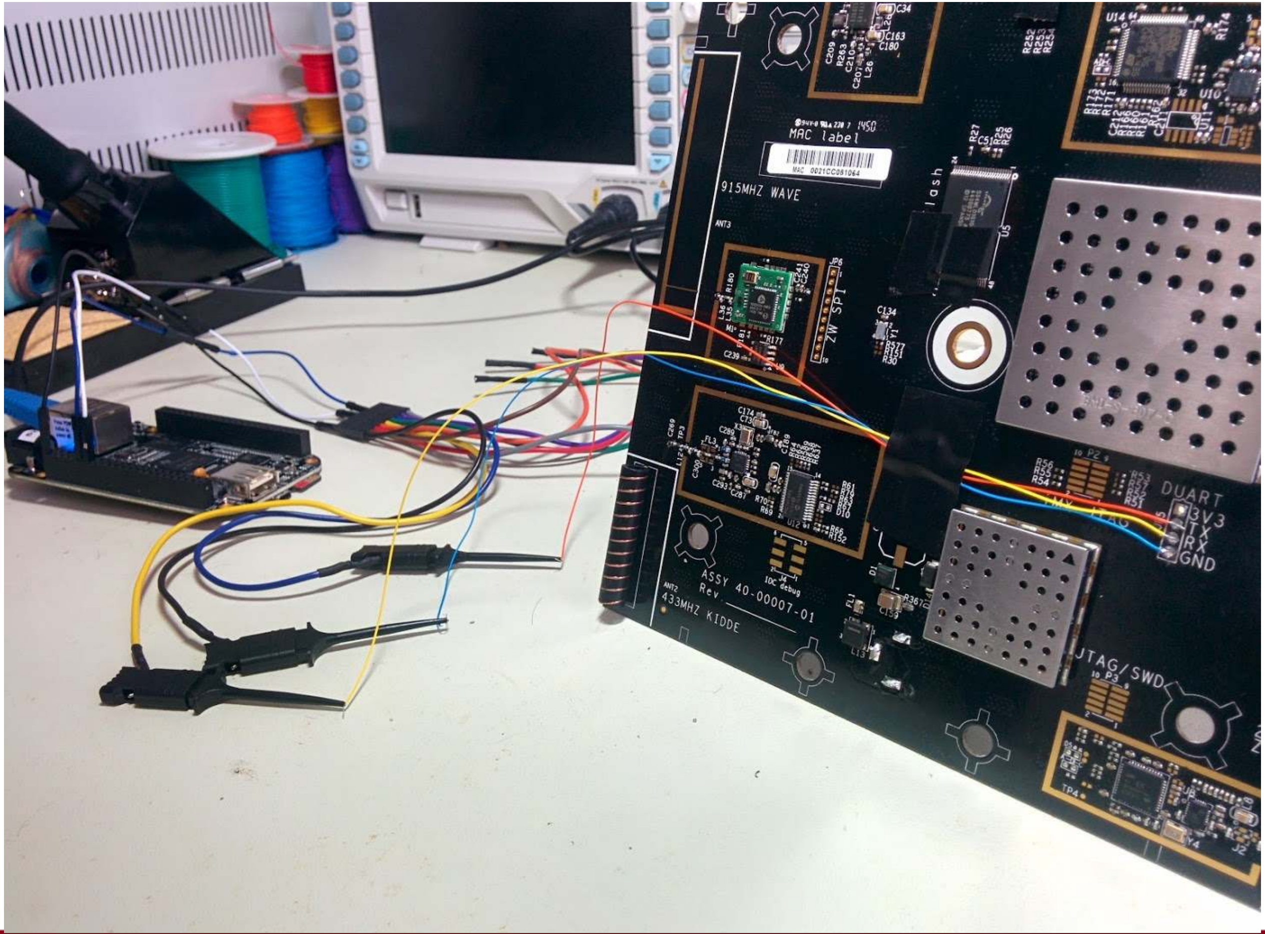
UART with BBH

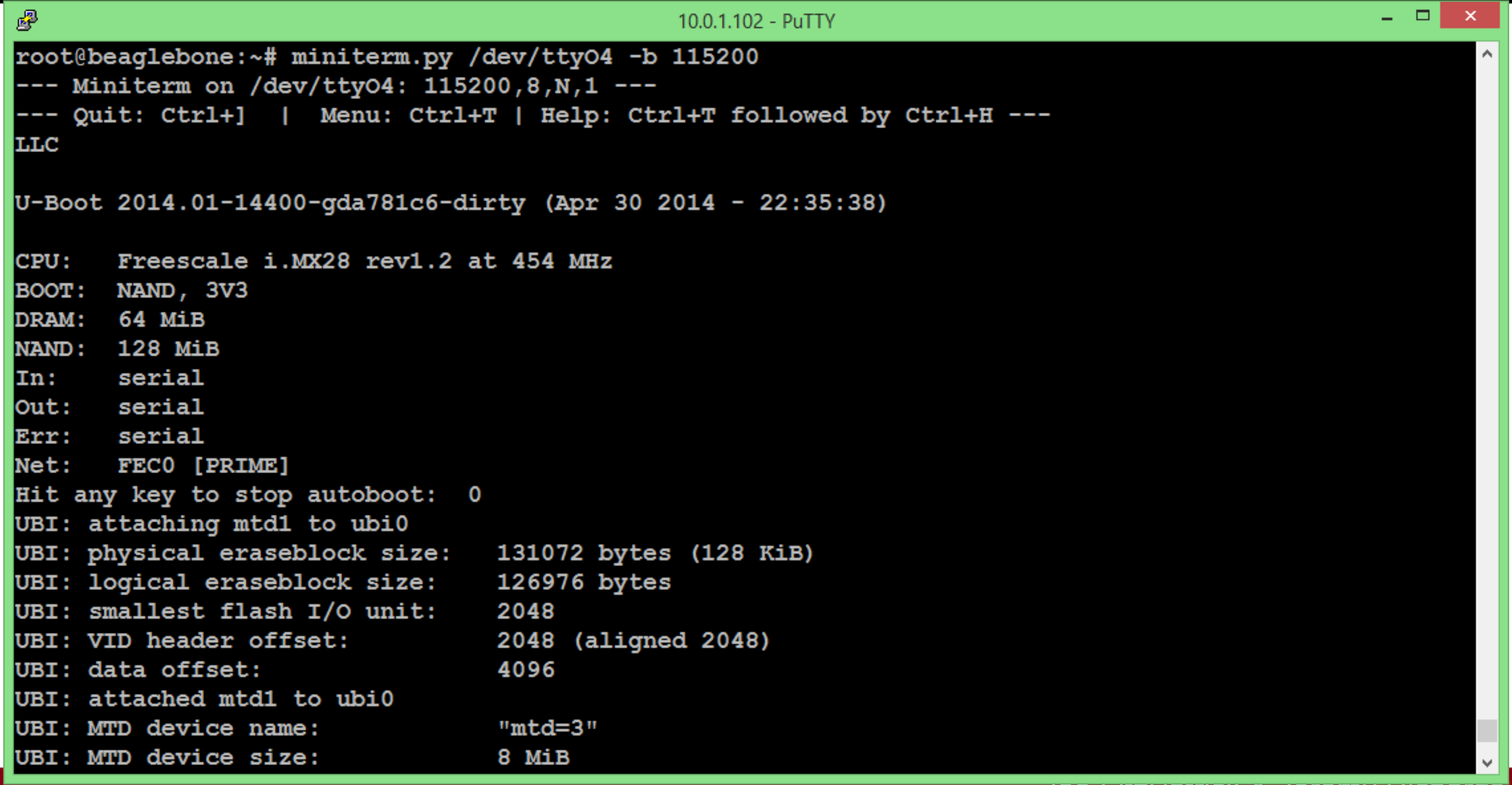
```
>echo BB-UART4 > /sys/devices/bone_capemgr.*/slots
```

UART4:

RX	P9_11
TX	P9_13
CTS	P8_35
RTS	P8_33

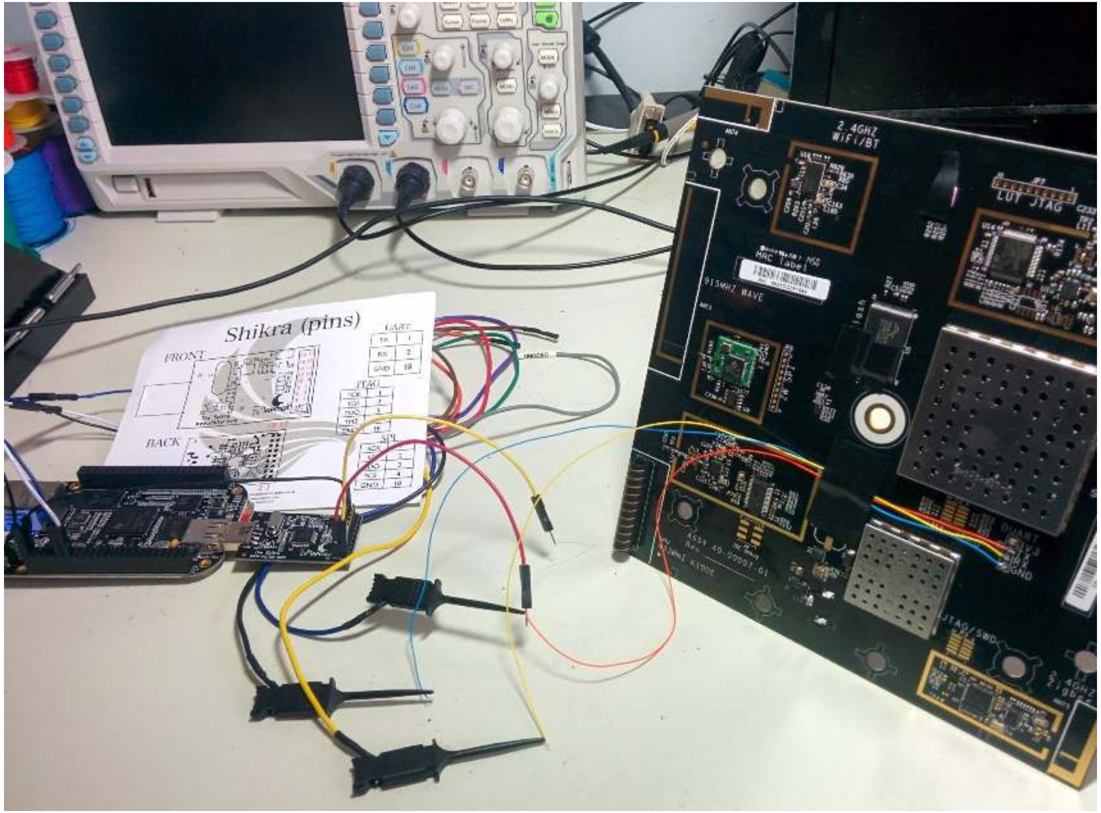

5V
P1

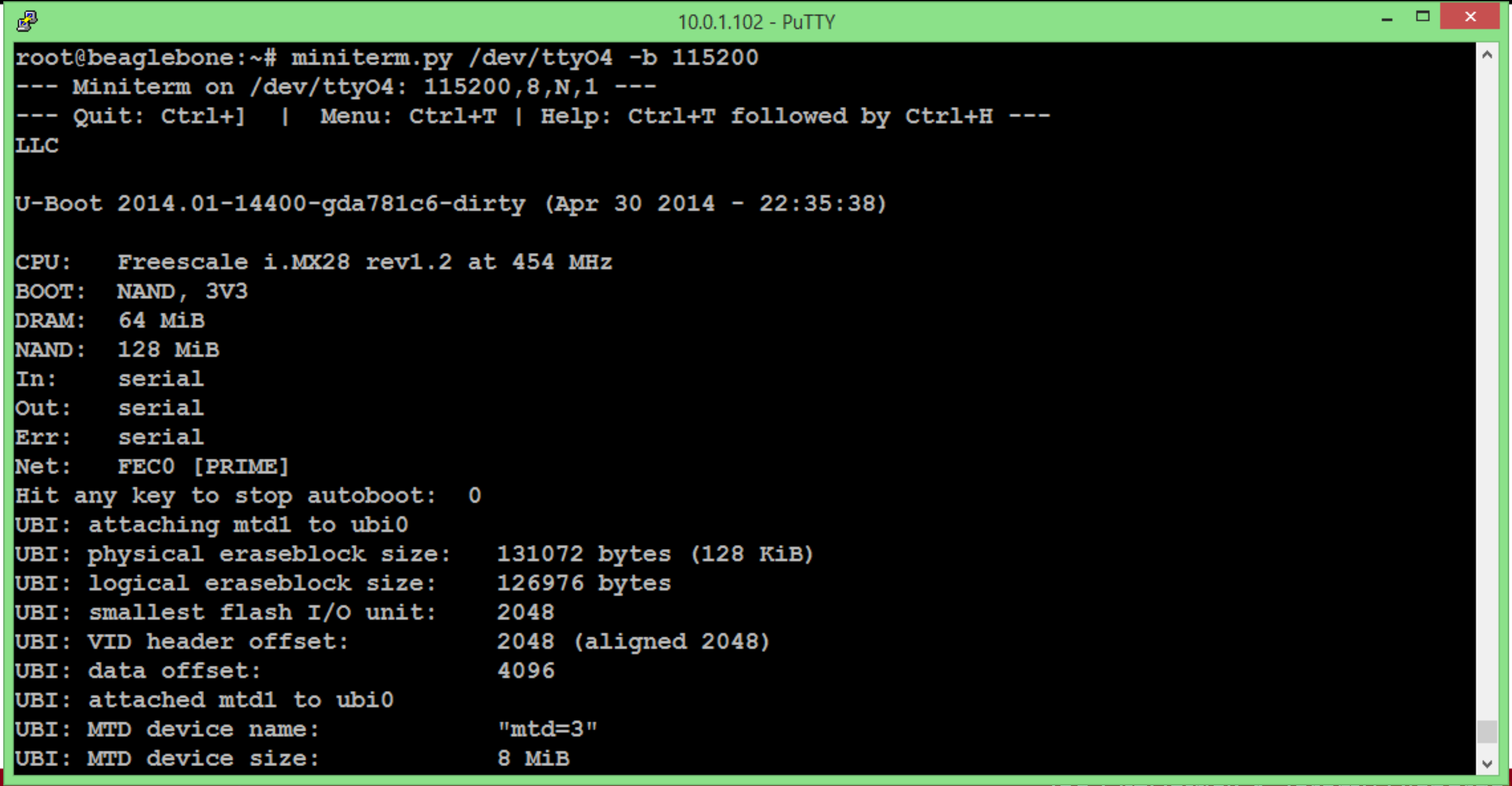

MAC label
915MHZ WAVE
ANT3
ZW SPI
JP6
DUART
3V3
TX
RX
GND
JTAG/SWD
P3
ASSY 40-00007-01
Rev.
ANT2
433MHZ KIDDE
IDC debug
U14
U5
C134
TP4



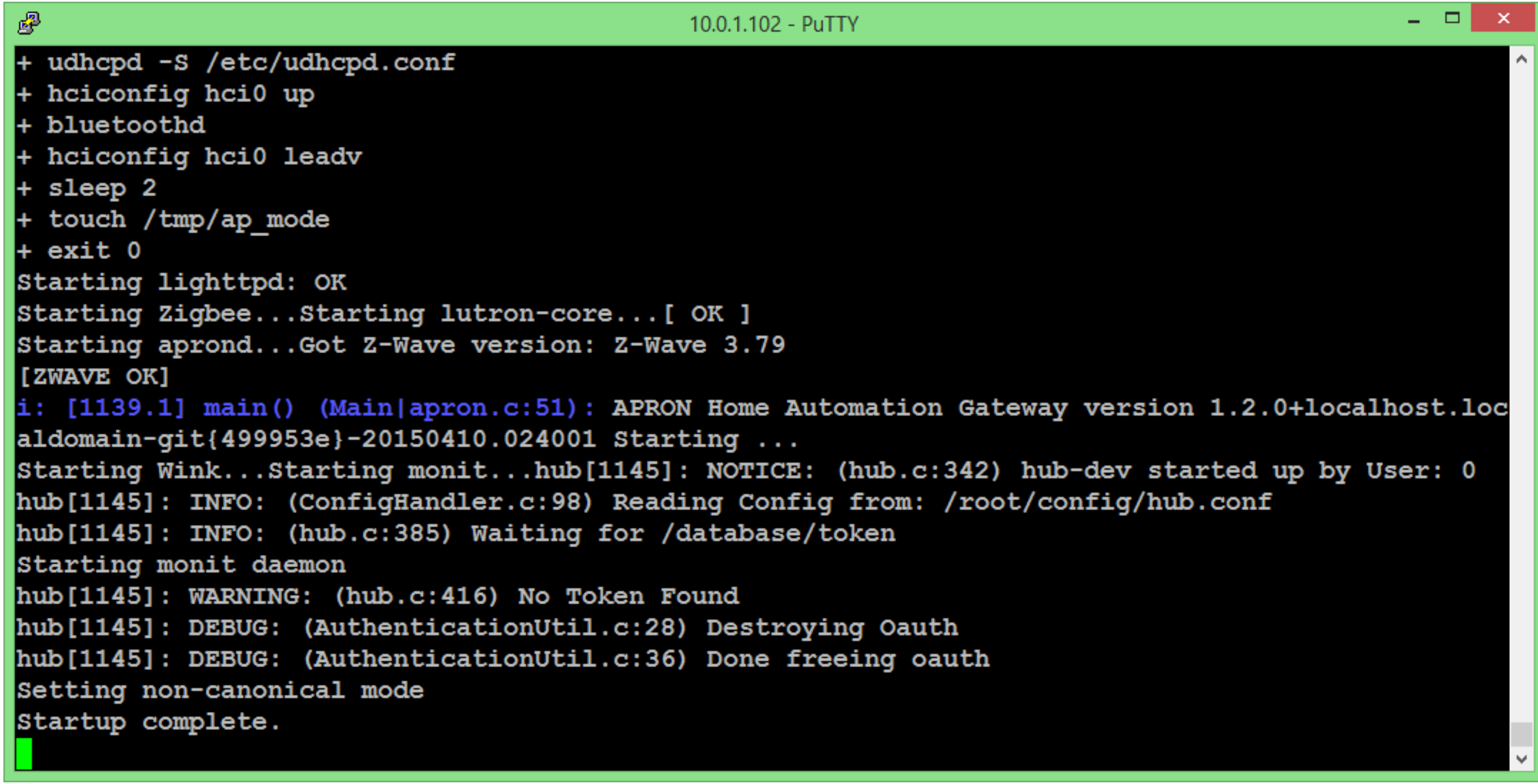
```
root@beaglebone:~# miniterm.py /dev/ttyO4 -b 115200
--- Miniterm on /dev/ttyO4: 115200,8,N,1 ---
--- Quit: Ctrl+]  |  Menu: Ctrl+T | Help: Ctrl+T followed by Ctrl+H ---
LLC

U-Boot 2014.01-14400-gda781c6-dirty (Apr 30 2014 - 22:35:38)

CPU:   Freescale i.MX28 rev1.2 at 454 MHz
BOOT:  NAND, 3V3
DRAM:  64 MiB
NAND:  128 MiB
In:    serial
Out:   serial
Err:   serial
Net:   FEC0 [PRIME]
Hit any key to stop autoboot:  0
UBI: attaching mtd1 to ubi0
UBI: physical eraseblock size:   131072 bytes (128 KiB)
UBI: logical eraseblock size:    126976 bytes
UBI: smallest flash I/O unit:    2048
UBI: VID header offset:          2048 (aligned 2048)
UBI: data offset:                4096
UBI: attached mtd1 to ubi0
UBI: MTD device name:            "mtd=3"
UBI: MTD device size:            8 MiB
```

UART - Need GPIO pins for something else?

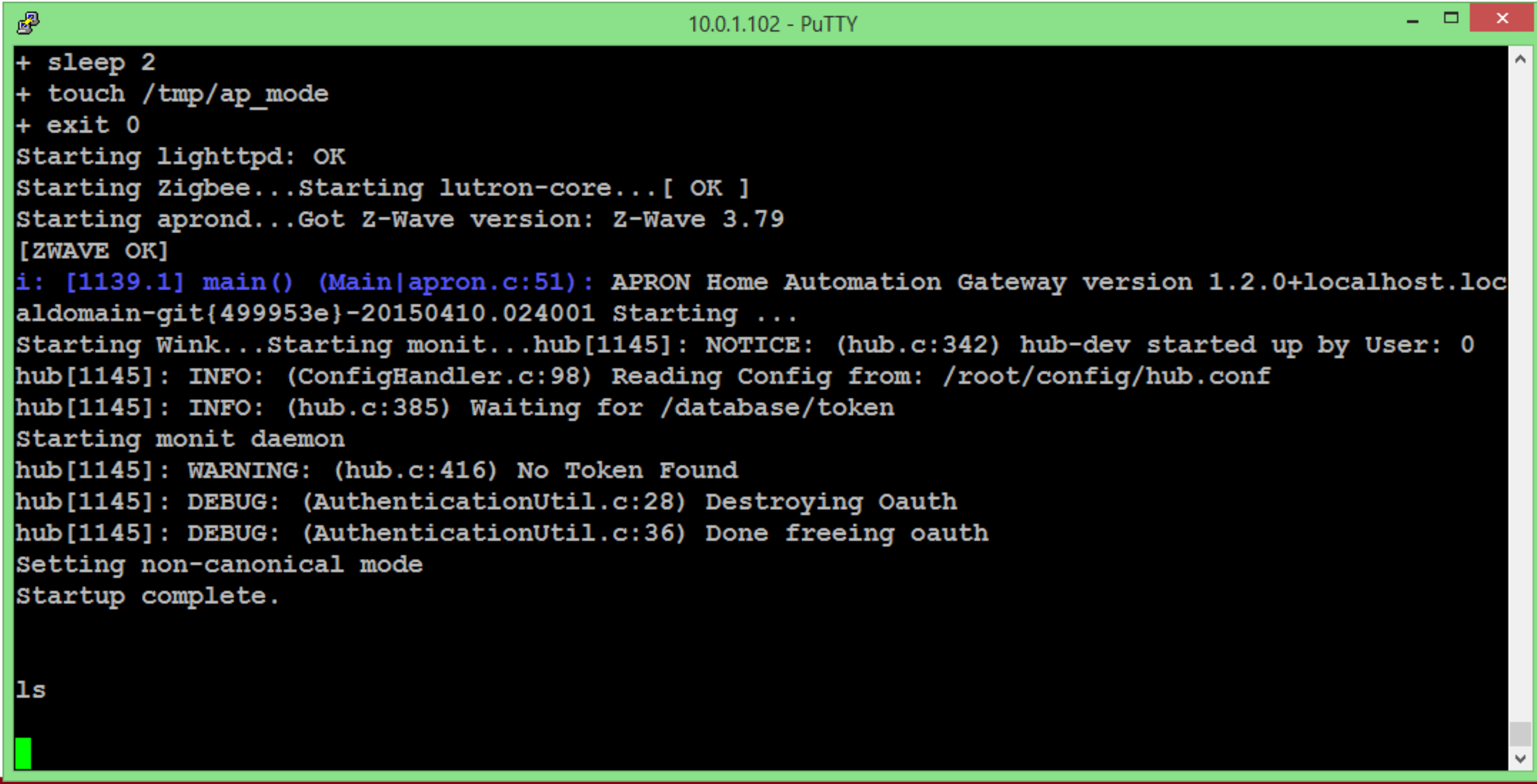




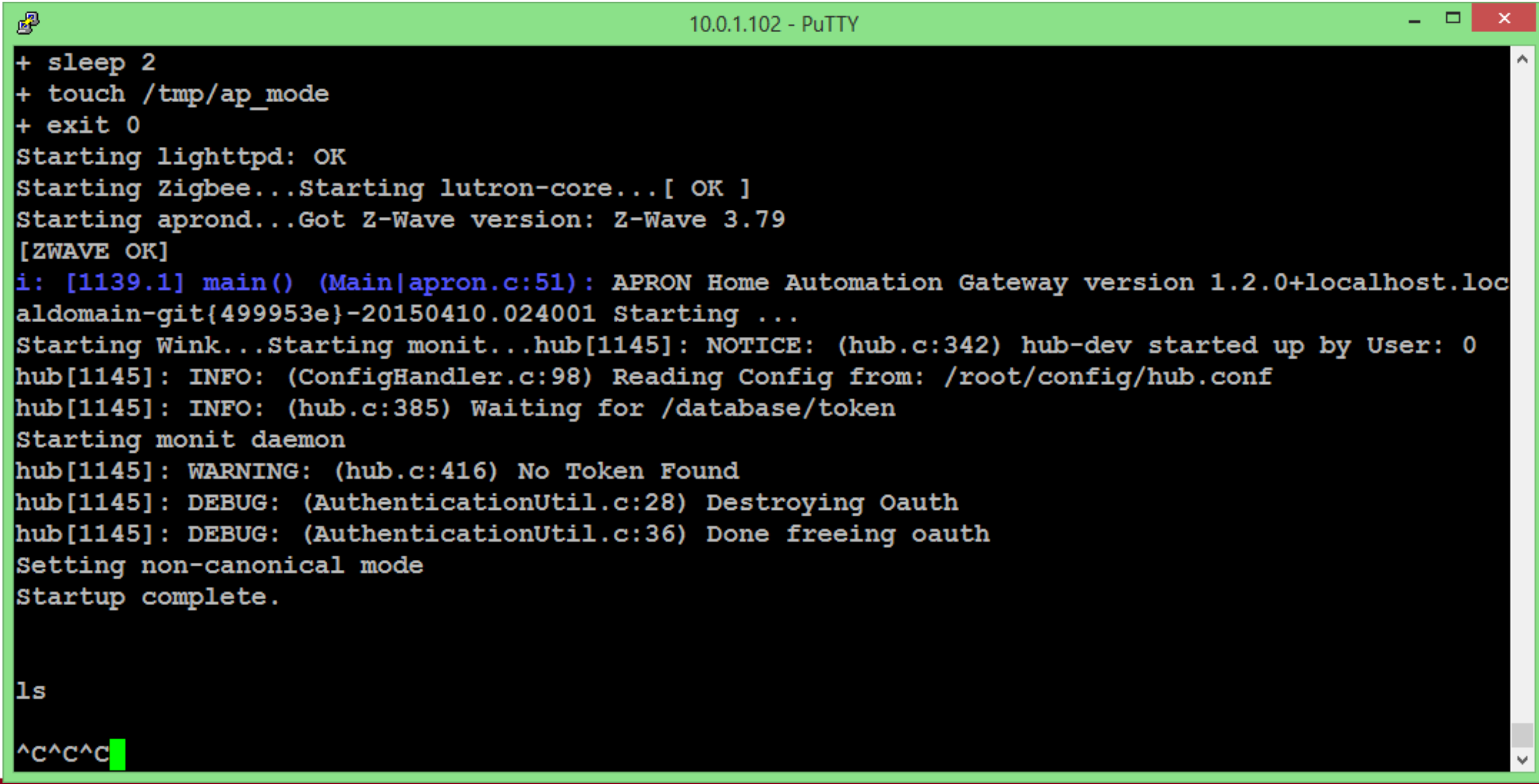
```
root@beaglebone:~# miniterm.py /dev/ttyO4 -b 115200
--- Miniterm on /dev/ttyO4: 115200,8,N,1 ---
--- Quit: Ctrl+]  |  Menu: Ctrl+T | Help: Ctrl+T followed by Ctrl+H ---
LLC

U-Boot 2014.01-14400-gda781c6-dirty (Apr 30 2014 - 22:35:38)

CPU:   Freescale i.MX28 rev1.2 at 454 MHz
BOOT:  NAND, 3V3
DRAM:  64 MiB
NAND:  128 MiB
In:    serial
Out:   serial
Err:   serial
Net:   FEC0 [PRIME]
Hit any key to stop autoboot:  0
UBI: attaching mtd1 to ubi0
UBI: physical eraseblock size:   131072 bytes (128 KiB)
UBI: logical eraseblock size:    126976 bytes
UBI: smallest flash I/O unit:    2048
UBI: VID header offset:          2048 (aligned 2048)
UBI: data offset:                4096
UBI: attached mtd1 to ubi0
UBI: MTD device name:            "mtd=3"
UBI: MTD device size:            8 MiB
```



```
+ udhcpd -S /etc/udhcpd.conf
+ hciconfig hci0 up
+ bluetoothd
+ hciconfig hci0 leadv
+ sleep 2
+ touch /tmp/ap_mode
+ exit 0
Starting lighttpd: OK
Starting Zigbee...Starting lutron-core...[ OK ]
Starting aprond...Got Z-Wave version: Z-Wave 3.79
[ZWAVE OK]
i: [1139.1] main() (Main|apron.c:51): APRON Home Automation Gateway version 1.2.0+localhost.loc
aldomain-git{499953e}-20150410.024001 Starting ...
Starting Wink...Starting monit...hub[1145]: NOTICE: (hub.c:342) hub-dev started up by User: 0
hub[1145]: INFO: (ConfigHandler.c:98) Reading Config from: /root/config/hub.conf
hub[1145]: INFO: (hub.c:385) Waiting for /database/token
Starting monit daemon
hub[1145]: WARNING: (hub.c:416) No Token Found
hub[1145]: DEBUG: (AuthenticationUtil.c:28) Destroying Oauth
hub[1145]: DEBUG: (AuthenticationUtil.c:36) Done freeing oauth
Setting non-canonical mode
Startup complete.
```

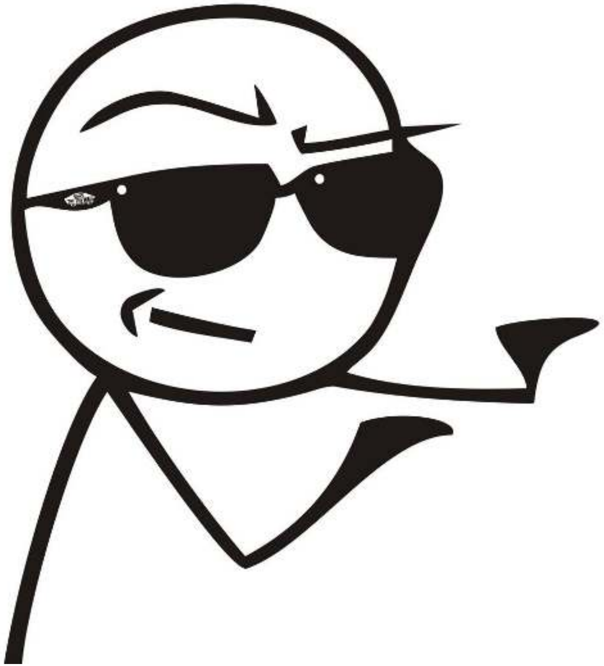


```
+ sleep 2
+ touch /tmp/ap_mode
+ exit 0
Starting lighttpd: OK
Starting Zigbee...Starting lutron-core...[ OK ]
Starting aprond...Got Z-Wave version: Z-Wave 3.79
[ZWAVE OK]
i: [1139.1] main() (Main|apron.c:51): APRON Home Automation Gateway version 1.2.0+localhost.loc
aldomain-git{499953e}-20150410.024001 Starting ...
Starting Wink...Starting monit...hub[1145]: NOTICE: (hub.c:342) hub-dev started up by User: 0
hub[1145]: INFO: (ConfigHandler.c:98) Reading Config from: /root/config/hub.conf
hub[1145]: INFO: (hub.c:385) Waiting for /database/token
Starting monit daemon
hub[1145]: WARNING: (hub.c:416) No Token Found
hub[1145]: DEBUG: (AuthenticationUtil.c:28) Destroying Oauth
hub[1145]: DEBUG: (AuthenticationUtil.c:36) Done freeing oauth
Setting non-canonical mode
Startup complete.


ls

```



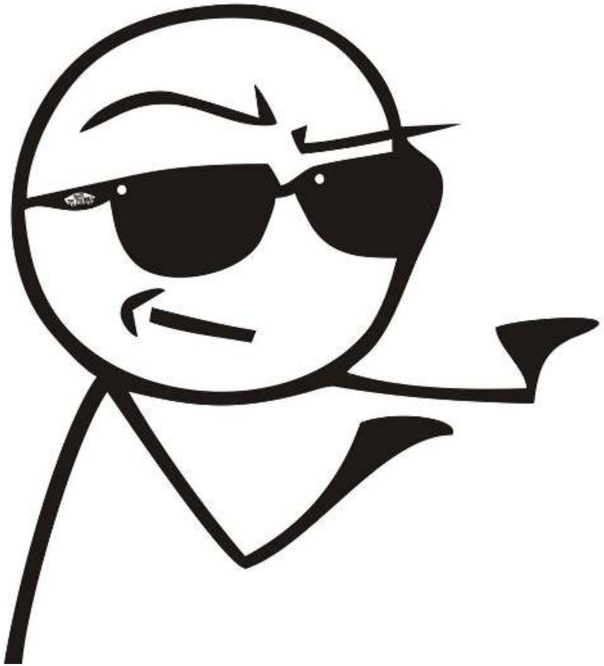
```
+ sleep 2
+ touch /tmp/ap_mode
+ exit 0
Starting lighttpd: OK
Starting Zigbee...Starting lutron-core...[ OK ]
Starting aprond...Got Z-Wave version: Z-Wave 3.79
[ZWAVE OK]
i: [1139.1] main() (Main|apron.c:51): APRON Home Automation Gateway version 1.2.0+localhost.loc
aldomain-git{499953e}-20150410.024001 Starting ...
Starting Wink...Starting monit...hub[1145]: NOTICE: (hub.c:342) hub-dev started up by User: 0
hub[1145]: INFO: (ConfigHandler.c:98) Reading Config from: /root/config/hub.conf
hub[1145]: INFO: (hub.c:385) Waiting for /database/token
Starting monit daemon
hub[1145]: WARNING: (hub.c:416) No Token Found
hub[1145]: DEBUG: (AuthenticationUtil.c:28) Destroying Oauth
hub[1145]: DEBUG: (AuthenticationUtil.c:36) Done freeing oauth
Setting non-canonical mode
Startup complete.


ls

^C^C^C
```

UART - Wink Hub root

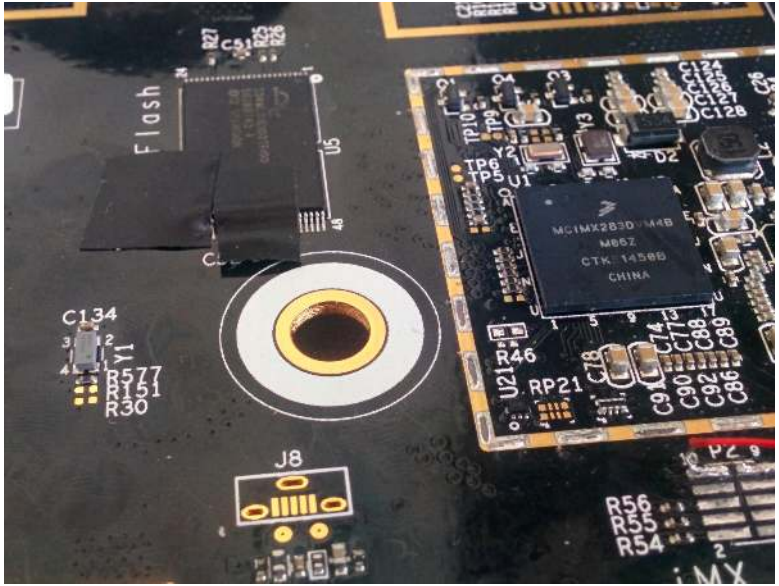


Method: Get Uboot to freak out by glitching NAND RAM. We will make the NAND flash available at first check then short it to cause the kernel image load to fail… and then drop into an interactive shell that lets us define environment variable. Copy existing and then add init=/bin/sh

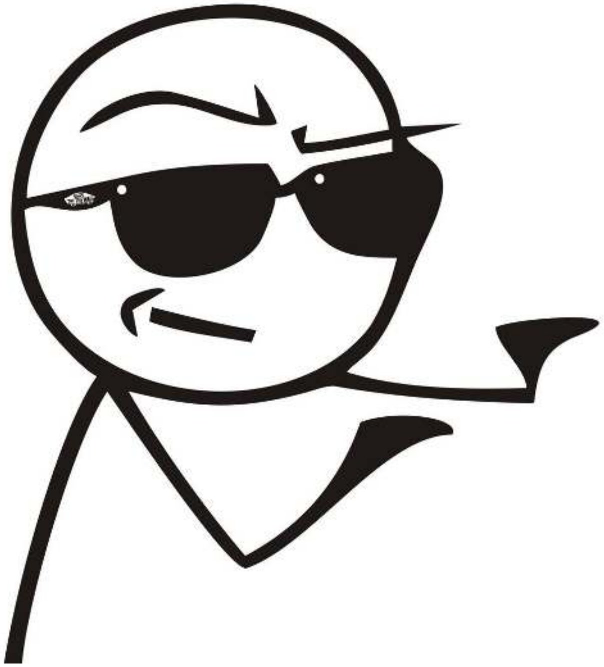
UART - Wink Hub root



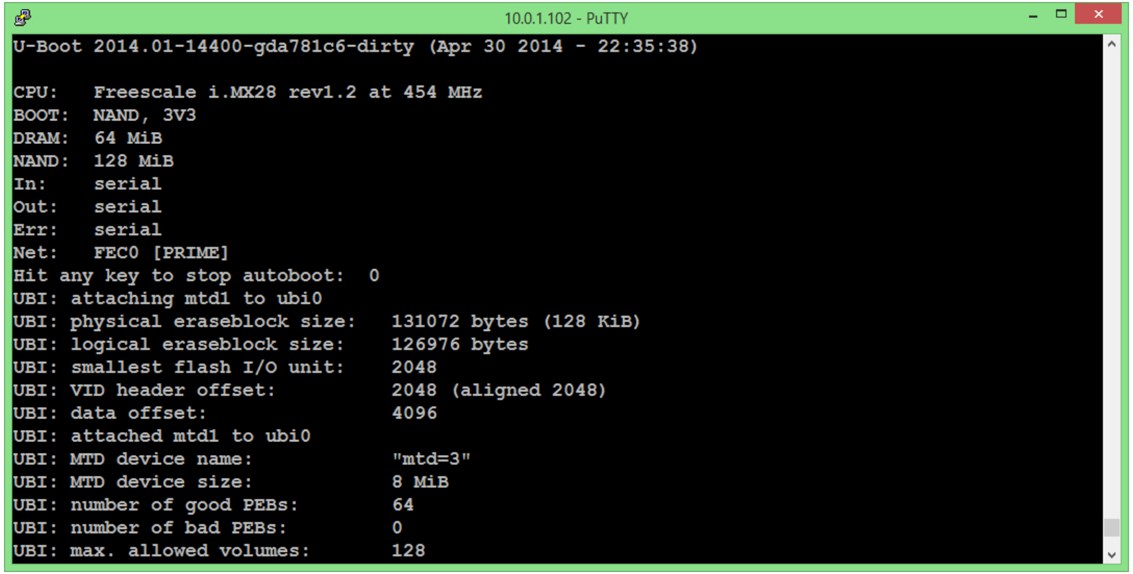
Method: We will make the NAND flash available at first check



UART - Wink Hub root

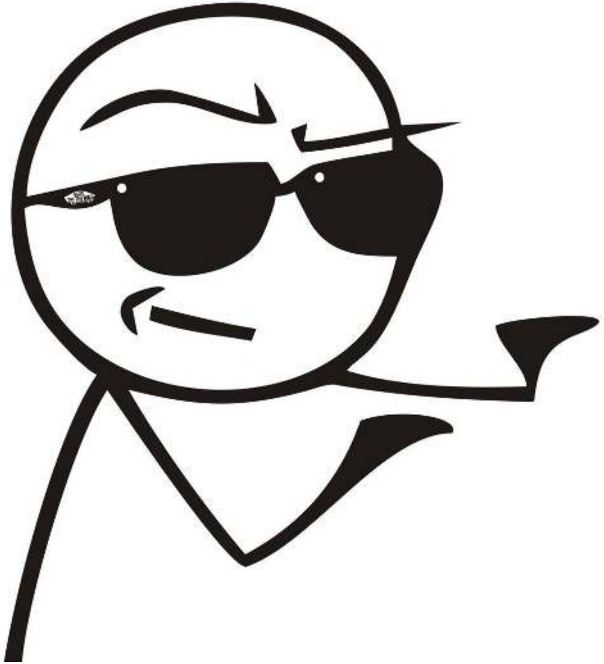


Method: We will make the NAND flash available at first check...

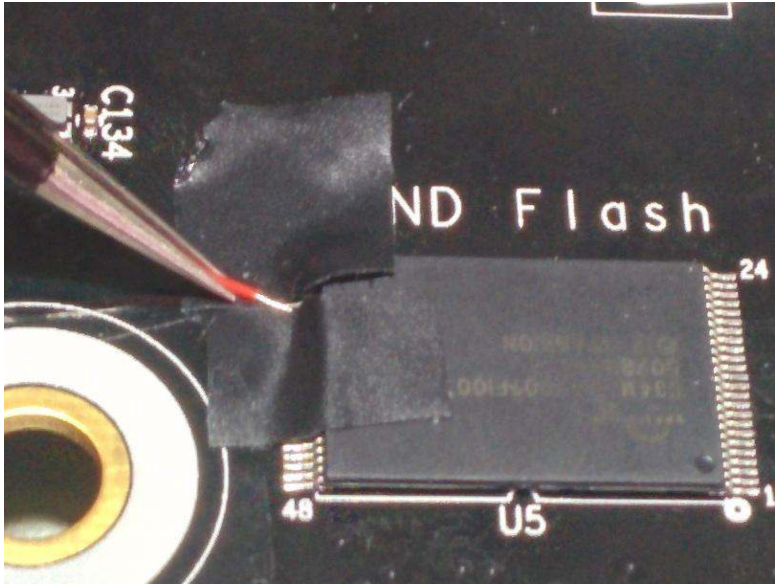


```
U-Boot 2014.01-14400-gda781c6-dirty (Apr 30 2014 - 22:35:38)

CPU:   Freescale i.MX28 rev1.2 at 454 MHz
BOOT:  NAND, 3V3
DRAM:  64 MiB
NAND:  128 MiB
In:    serial
Out:   serial
Err:   serial
Net:   FEC0 [PRIME]
Hit any key to stop autoboot:  0
UBI: attaching mtd1 to ubi0
UBI: physical eraseblock size:   131072 bytes (128 KiB)
UBI: logical eraseblock size:    126976 bytes
UBI: smallest flash I/O unit:    2048
UBI: VID header offset:          2048 (aligned 2048)
UBI: data offset:                4096
UBI: attached mtd1 to ubi0
UBI: MTD device name:            "mtd=3"
UBI: MTD device size:            8 MiB
UBI: number of good PEBs:        64
UBI: number of bad PEBs:         0
UBI: max. allowed volumes:       128
```

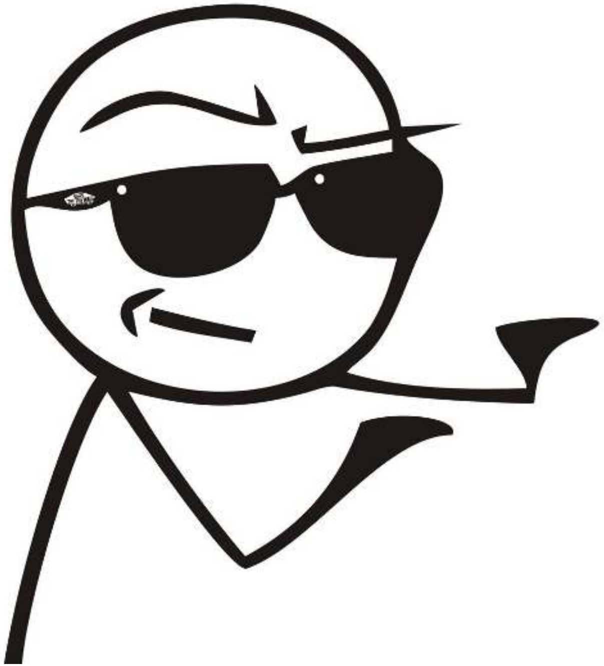
UART - Wink Hub root



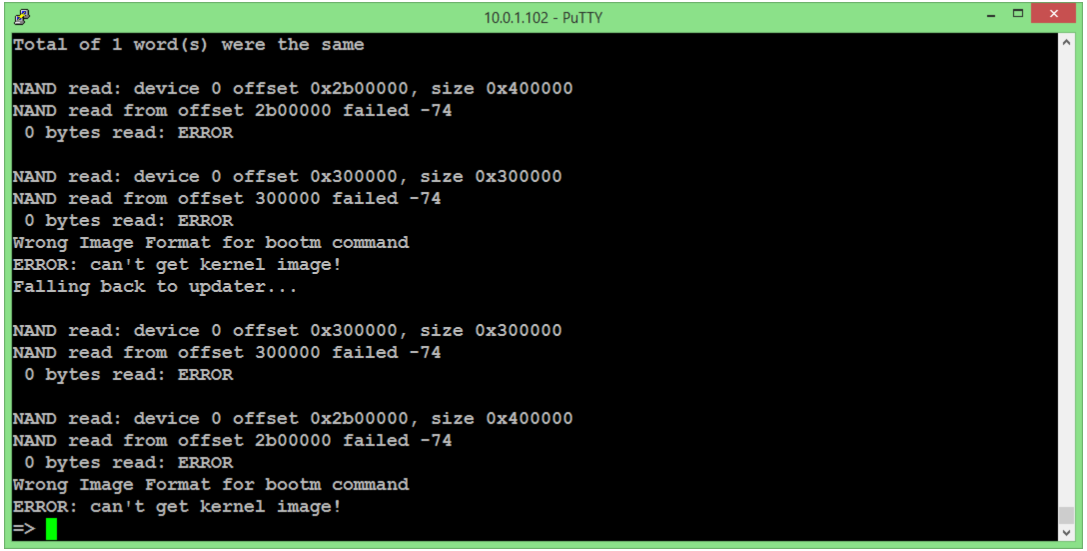
then short it to cause the kernel image load to fail…



UART - Wink Hub root

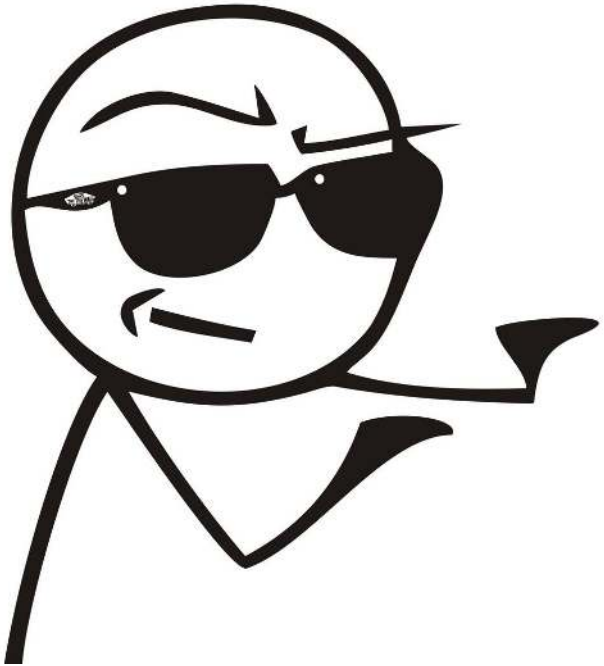


...and then drop into an interactive shell that lets us define environment variable.

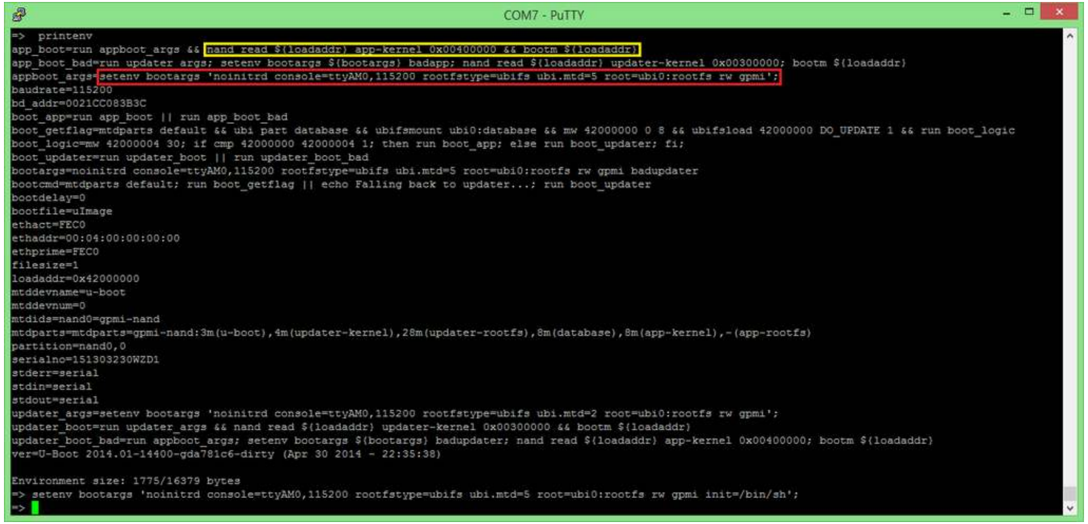


```
Total of 1 word(s) were the same

NAND read: device 0 offset 0x2b00000, size 0x400000
NAND read from offset 2b00000 failed -74
 0 bytes read: ERROR

NAND read: device 0 offset 0x300000, size 0x300000
NAND read from offset 300000 failed -74
 0 bytes read: ERROR
Wrong Image Format for bootm command
ERROR: can't get kernel image!
Falling back to updater...

NAND read: device 0 offset 0x300000, size 0x300000
NAND read from offset 300000 failed -74
 0 bytes read: ERROR

NAND read: device 0 offset 0x2b00000, size 0x400000
NAND read from offset 2b00000 failed -74
 0 bytes read: ERROR
Wrong Image Format for bootm command
ERROR: can't get kernel image!
=>
```

UART - Wink Hub root

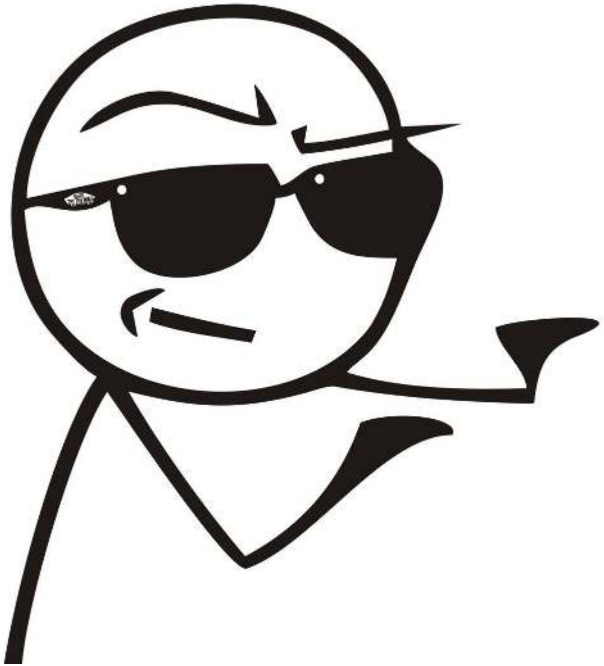


Method: Copy existing app_bootargs=and then add init=/bin/sh. Finally run the app_boot (yellow)

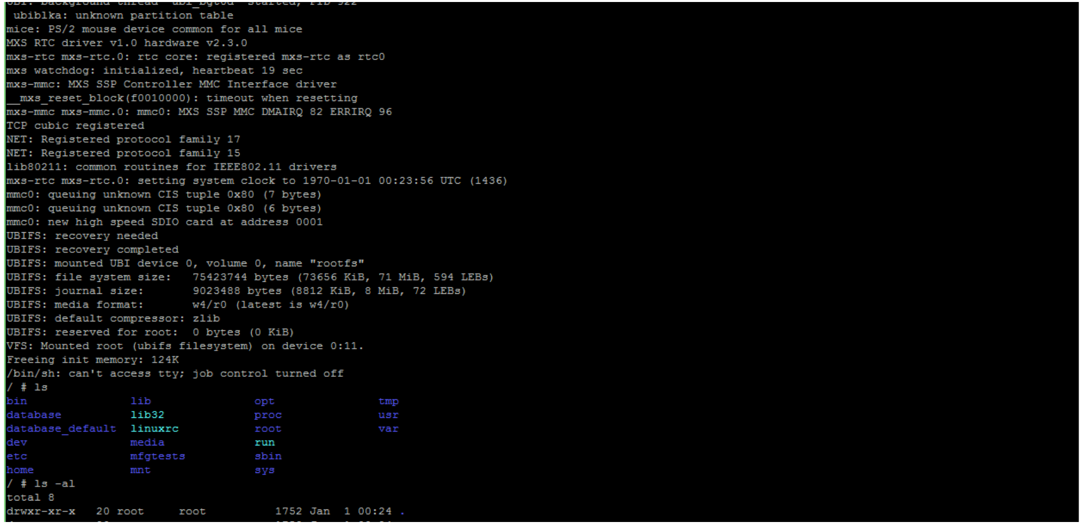


```
=>  printenv
app_boot=run appboot_args && nand read ${loadaddr} app-kernel 0x00400000 && bootm ${loadaddr}
app_boot_bad=run updater args; setenv bootargs ${bootargs} badapp; nand read ${loadaddr} updater-kernel 0x00300000; bootm ${loadaddr}
appboot_args=setenv bootargs 'noinitrd console=ttyAM0,115200 rootfstype=ubifs ubi.mtd=5 root=ubi0:rootfs rw gpmi';
baudrate=115200
bd_addr=0021CC083B3C
boot_app=run app_boot || run app_boot_bad
boot_getflag=mtdparts default && ubi part database && ubifsmount ubi0:database && mw 42000000 0 8 && ubifsload 42000000 DO_UPDATE 1 && run boot_logic
boot_logic=mw 42000004 30; if cmp 42000000 42000004 1; then run boot_app; else run boot_updater; fi;
boot_updater=run updater_boot || run updater_boot_bad
bootargs=noinitrd console=ttyAM0,115200 rootfstype=ubifs ubi.mtd=5 root=ubi0:rootfs rw gpmi badupdater
bootcmd=mtdparts default; run boot_getflag || echo Falling back to updater...; run boot_updater
bootdelay=0
bootfile=uImage
ethact=FEC0
ethaddr=00:04:00:00:00:00
ethprime=FEC0
filesize=1
loadaddr=0x42000000
mtddevname=u-boot
mtddevnum=0
mtdids=nand0=gpmi-nand
mtdparts=mtdparts=gpmi-nand:3m(u-boot),4m(updater-kernel),28m(updater-rootfs),8m(database),8m(app-kernel),-(app-rootfs)
partition=nand0,0
serialno=151303230WZD1
stderr=serial
stdin=serial
stdout=serial
updater_args=setenv bootargs 'noinitrd console=ttyAM0,115200 rootfstype=ubifs ubi.mtd=2 root=ubi0:rootfs rw gpmi';
updater_boot=run updater_args && nand read ${loadaddr} updater-kernel 0x00300000 && bootm ${loadaddr}
updater_boot_bad=run appboot_args; setenv bootargs ${bootargs} badupdater; nand read ${loadaddr} app-kernel 0x00400000; bootm ${loadaddr}
ver=U-Boot 2014.01-14400-gda781c6-dirty (Apr 30 2014 - 22:35:38)

Environment size: 1775/16379 bytes
=> setenv bootargs 'noinitrd console=ttyAM0,115200 rootfstype=ubifs ubi.mtd=5 root=ubi0:rootfs rw gpmi init=/bin/sh';
=> 
```

UART - Wink Hub root

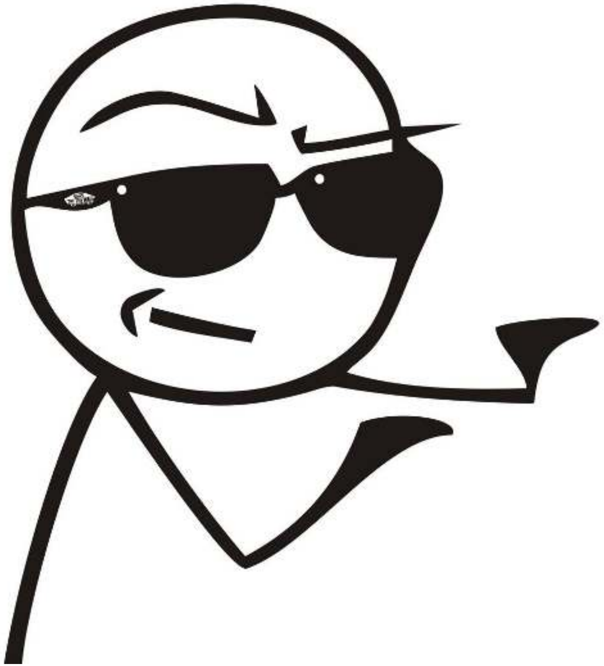


Method: Copy existing app_bootargs=and then add init=/bin/sh. Finally run the app_boot (yellow)

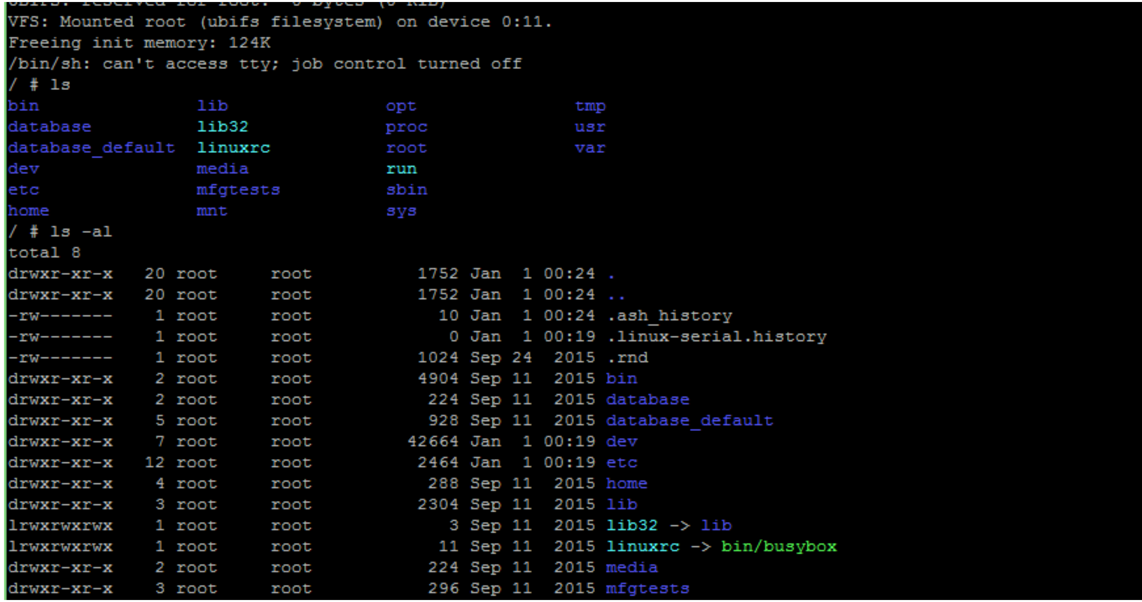


```
UBI: background thread "ubi_bgt0d" started, PID 522
 ubiblka: unknown partition table
mice: PS/2 mouse device common for all mice
MXS RTC driver v1.0 hardware v2.3.0
mxs-rtc mxs-rtc.0: rtc core: registered mxs-rtc as rtc0
mxs watchdog: initialized, heartbeat 19 sec
mxs-mmc: MXS SSP Controller MMC Interface driver
__mxs_reset_block(f0010000): timeout when resetting
mxs-mmc mxs-mmc.0: mmc0: MXS SSP MMC DMAIRQ 82 ERRIRQ 96
TCP cubic registered
NET: Registered protocol family 17
NET: Registered protocol family 15
lib80211: common routines for IEEE802.11 drivers
mxs-rtc mxs-rtc.0: setting system clock to 1970-01-01 00:23:56 UTC (1436)
mmc0: queuing unknown CIS tuple 0x80 (7 bytes)
mmc0: queuing unknown CIS tuple 0x80 (6 bytes)
mmc0: new high speed SDIO card at address 0001
UBIFS: recovery needed
UBIFS: recovery completed
UBIFS: mounted UBI device 0, volume 0, name "rootfs"
UBIFS: file system size:   75423744 bytes (73656 KiB, 71 MiB, 594 LEBs)
UBIFS: journal size:       9023488 bytes (8812 KiB, 8 MiB, 72 LEBs)
UBIFS: media format:       w4/r0 (latest is w4/r0)
UBIFS: default compressor: zlib
UBIFS: reserved for root:  0 bytes (0 KiB)
VFS: Mounted root (ubifs filesystem) on device 0:11.
Freeing init memory: 124K
/bin/sh: can't access tty; job control turned off
/ # ls
bin               lib               opt               tmp
database          lib32             proc              usr
database_default  linuxrc           root              var
dev               media             run
etc               mfgtests          sbin
home              mnt               sys
/ # ls -al
total 8
drwxr-xr-x   20 root     root          1752 Jan  1 00:24 .
```

UART - Wink Hub root



Method: Copy existing app_bootargs=and then add init=/bin/sh. Finally run the app_boot (yellow)



```
UBIFS: reserved for root: 0 bytes (0 KiB)
VFS: Mounted root (ubifs filesystem) on device 0:11.
Freeing init memory: 124K
/bin/sh: can't access tty; job control turned off
/ # ls
bin               lib               opt               tmp
database          lib32             proc              usr
database_default  linuxrc           root              var
dev               media             run
etc               mfgtests          sbin
home              mnt               sys
/ # ls -al
total 8
drwxr-xr-x   20 root     root          1752 Jan  1 00:24 .
drwxr-xr-x   20 root     root          1752 Jan  1 00:24 ..
-rw-------    1 root     root            10 Jan  1 00:24 .ash_history
-rw-------    1 root     root             0 Jan  1 00:19 .linux-serial.history
-rw-------    1 root     root          1024 Sep 24  2015 .rnd
drwxr-xr-x    2 root     root          4904 Sep 11  2015 bin
drwxr-xr-x    2 root     root           224 Sep 11  2015 database
drwxr-xr-x    5 root     root           928 Sep 11  2015 database_default
drwxr-xr-x    7 root     root         42664 Jan  1 00:19 dev
drwxr-xr-x   12 root     root          2464 Jan  1 00:19 etc
drwxr-xr-x    4 root     root           288 Sep 11  2015 home
drwxr-xr-x    3 root     root          2304 Sep 11  2015 lib
lrwxrwxrwx    1 root     root             3 Sep 11  2015 lib32 -> lib
lrwxrwxrwx    1 root     root            11 Sep 11  2015 linuxrc -> bin/busybox
drwxr-xr-x    2 root     root           224 Sep 11  2015 media
drwxr-xr-x    3 root     root           296 Sep 11  2015 mfgtests
```

Task	Pre-BusPirate $$$$, 🕓🕓🕓	Bus Pirate $$, 🕓🕓🕓🕓	Post-BusPirate $$$, 🕓🕓	Beaglebone Hack $$, 🕓
Talk UART	**RS232 hardware + level shifting**	**narrow tolerance**	**FT232R, just works, $$**	**native hardware**
Interface I2C				
Dump SPI Flash				
Analyze Logic				
JTAG				

I2C

I2C

AKA IIC

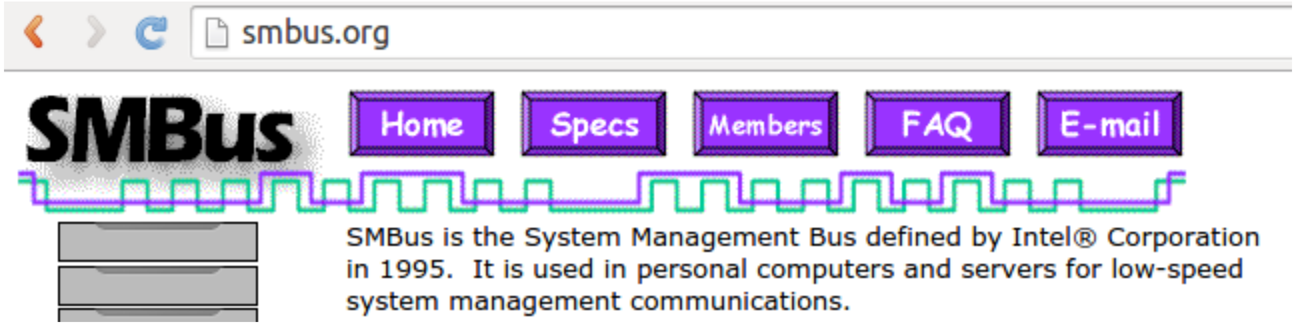
AKA eye-squared-see

AKA I^2C

AKA eye-two-see

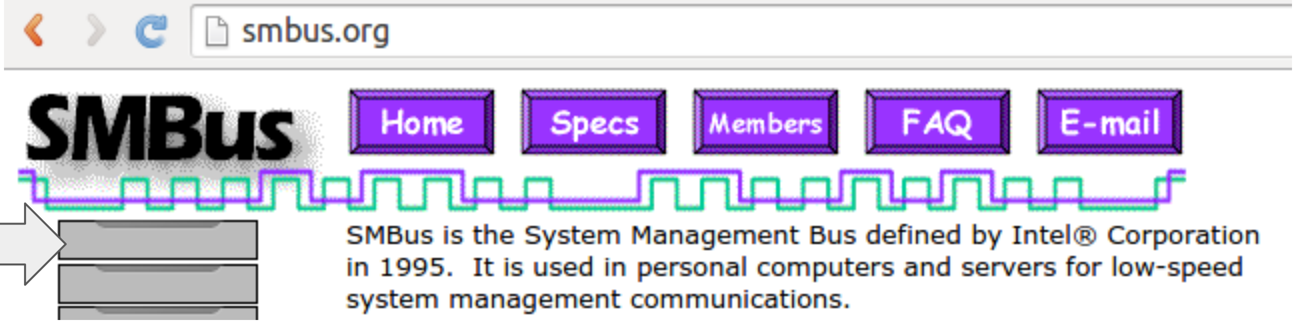
AKA aye-eye-see

Also, SMBus, 2-wire, and much more are similar in concept and often compatible...



SMBus is the System Management Bus defined by Intel® Corporation in 1995. It is used in personal computers and servers for low-speed system management communications.

And hasn't updated their website since...

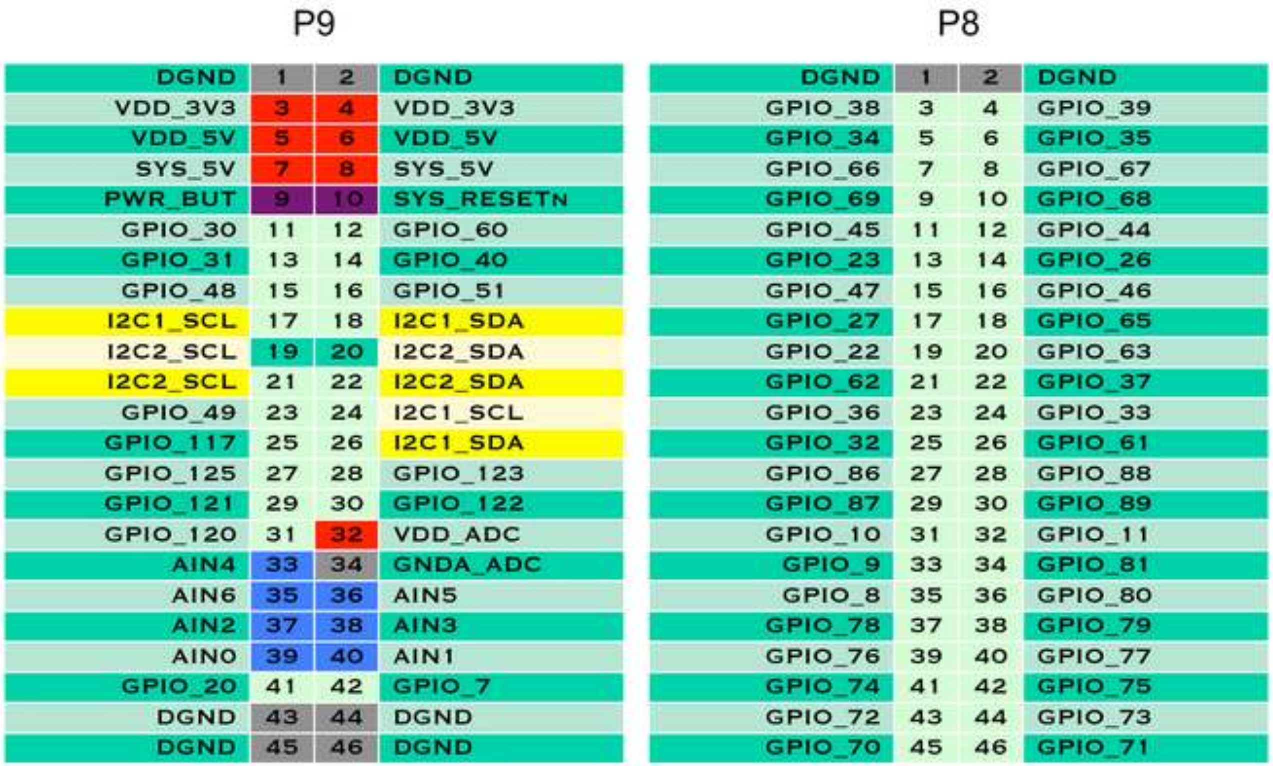


SMBus is the System Management Bus defined by Intel® Corporation in 1995. It is used in personal computers and servers for low-speed system management communications.

Note:
Java Applet

And hasn't updated their website since...

2 I2C ports

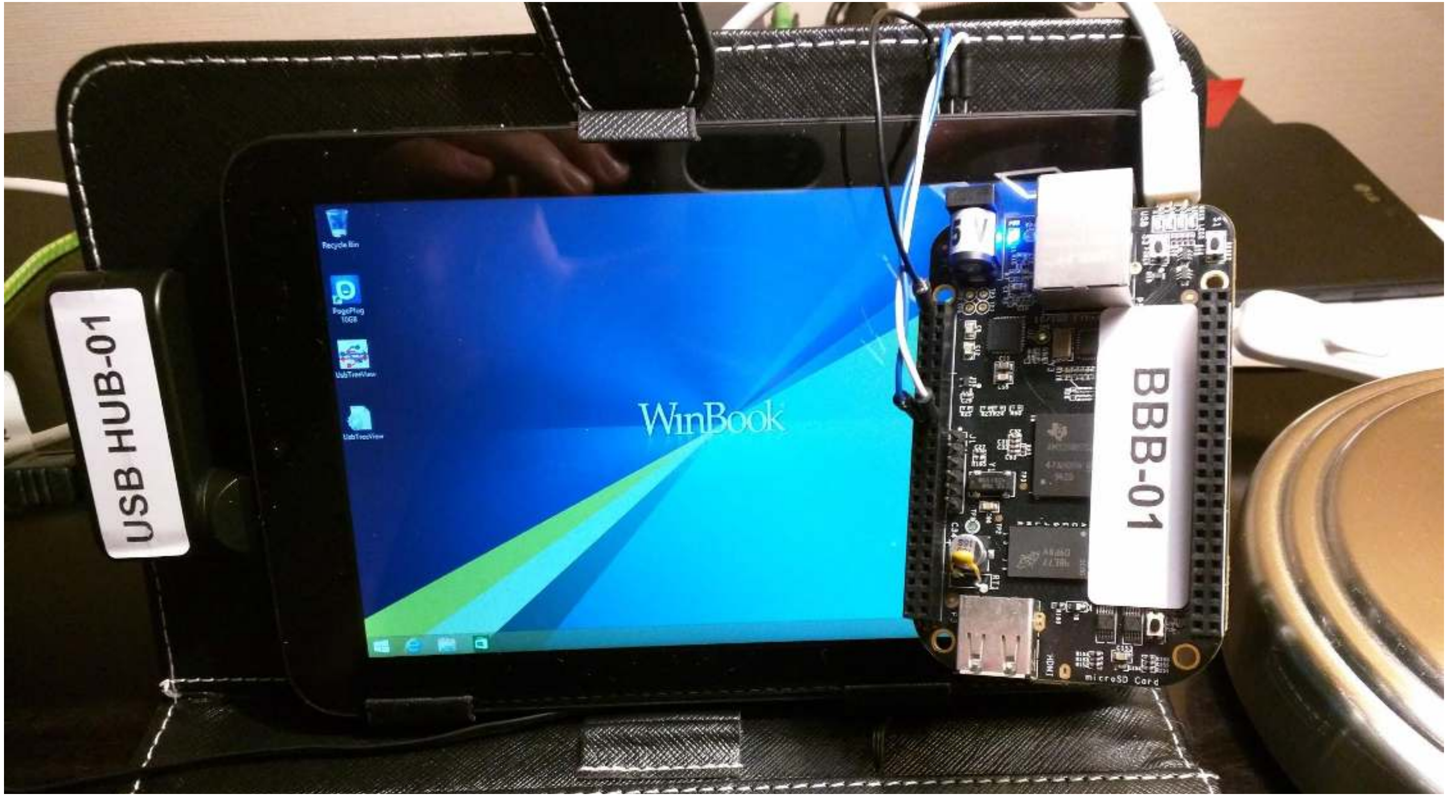


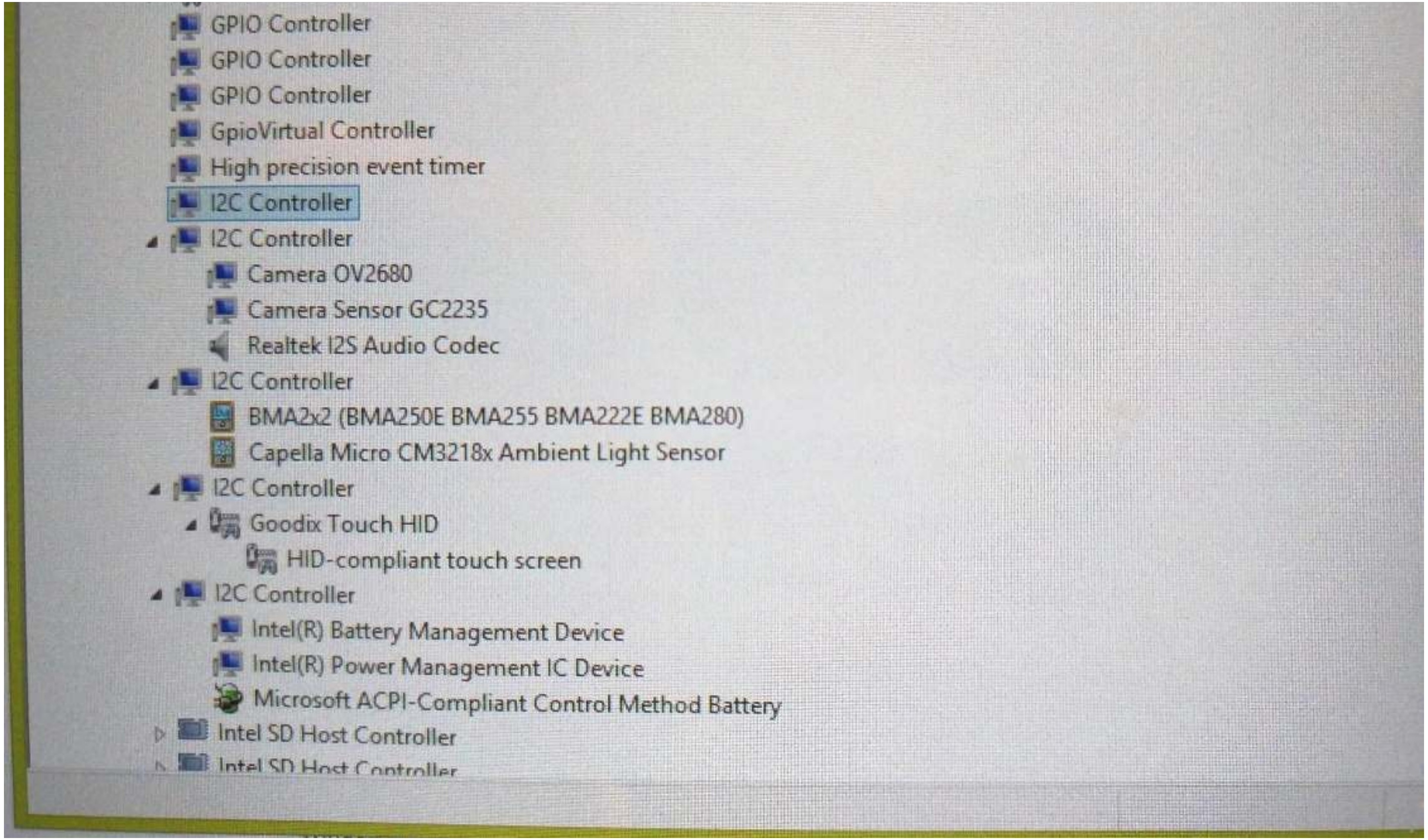
P9

DGND	1	2	DGND
VDD_3V3	3	4	VDD_3V3
VDD_5V	5	6	VDD_5V
SYS_5V	7	8	SYS_5V
PWR_BUT	9	10	SYS_RESETN
GPIO_30	11	12	GPIO_60
GPIO_31	13	14	GPIO_40
GPIO_48	15	16	GPIO_51
I2C1_SCL	17	18	I2C1_SDA
I2C2_SCL	19	20	I2C2_SDA
I2C2_SCL	21	22	I2C2_SDA
GPIO_49	23	24	I2C1_SCL
GPIO_117	25	26	I2C1_SDA
GPIO_125	27	28	GPIO_123
GPIO_121	29	30	GPIO_122
GPIO_120	31	32	VDD_ADC
AIN4	33	34	GNDA_ADC
AIN6	35	36	AIN5
AIN2	37	38	AIN3
AIN0	39	40	AIN1
GPIO_20	41	42	GPIO_7
DGND	43	44	DGND
DGND	45	46	DGND

P8

DGND	1	2	DGND
GPIO_38	3	4	GPIO_39
GPIO_34	5	6	GPIO_35
GPIO_66	7	8	GPIO_67
GPIO_69	9	10	GPIO_68
GPIO_45	11	12	GPIO_44
GPIO_23	13	14	GPIO_26
GPIO_47	15	16	GPIO_46
GPIO_27	17	18	GPIO_65
GPIO_22	19	20	GPIO_63
GPIO_62	21	22	GPIO_37
GPIO_36	23	24	GPIO_33
GPIO_32	25	26	GPIO_61
GPIO_86	27	28	GPIO_88
GPIO_87	29	30	GPIO_89
GPIO_10	31	32	GPIO_11
GPIO_9	33	34	GPIO_81
GPIO_8	35	36	GPIO_80
GPIO_78	37	38	GPIO_79
GPIO_76	39	40	GPIO_77
GPIO_74	41	42	GPIO_75
GPIO_72	43	44	GPIO_73
GPIO_70	45	46	GPIO_71


USB HUB-01
WinBook
BBB-01
Recycle Bin
PogoPlug 10GB
UsbTreeView
microSD Card
HDMI



```
   GPIO Controller
   GPIO Controller
   GPIO Controller
   GpioVirtual Controller
   High precision event timer
   I2C Controller
 ◢ I2C Controller
       Camera OV2680
       Camera Sensor GC2235
       Realtek I2S Audio Codec
 ◢ I2C Controller
       BMA2x2 (BMA250E BMA255 BMA222E BMA280)
       Capella Micro CM3218x Ambient Light Sensor
 ◢ I2C Controller
     ◢ Goodix Touch HID
           HID-compliant touch screen
 ◢ I2C Controller
       Intel(R) Battery Management Device
       Intel(R) Power Management IC Device
       Microsoft ACPI-Compliant Control Method Battery
 ▷ Intel SD Host Controller
 ▷ Intel SD Host Controller
```

```
root@beaglebone:/home/debian# echo BB-I2C1 > /sys/devices/bone_capemgr.*/slots
root@beaglebone:/home/debian# i2cdetect -l
i2c-0    i2c           OMAP I2C adapter                 I2C adapter
i2c-1    i2c           OMAP I2C adapter                 I2C adapter
i2c-2    i2c           OMAP I2C adapter                 I2C adapter
root@beaglebone:/home/debian# i2cdetect -r 2WARNING! This program can confuse your
I2C bus, cause data loss and worse!
I will probe file /dev/i2c-2 using read byte commands.
I will probe address range 0x03-0x77.
Continue? [Y/n]
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- 14 -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
```

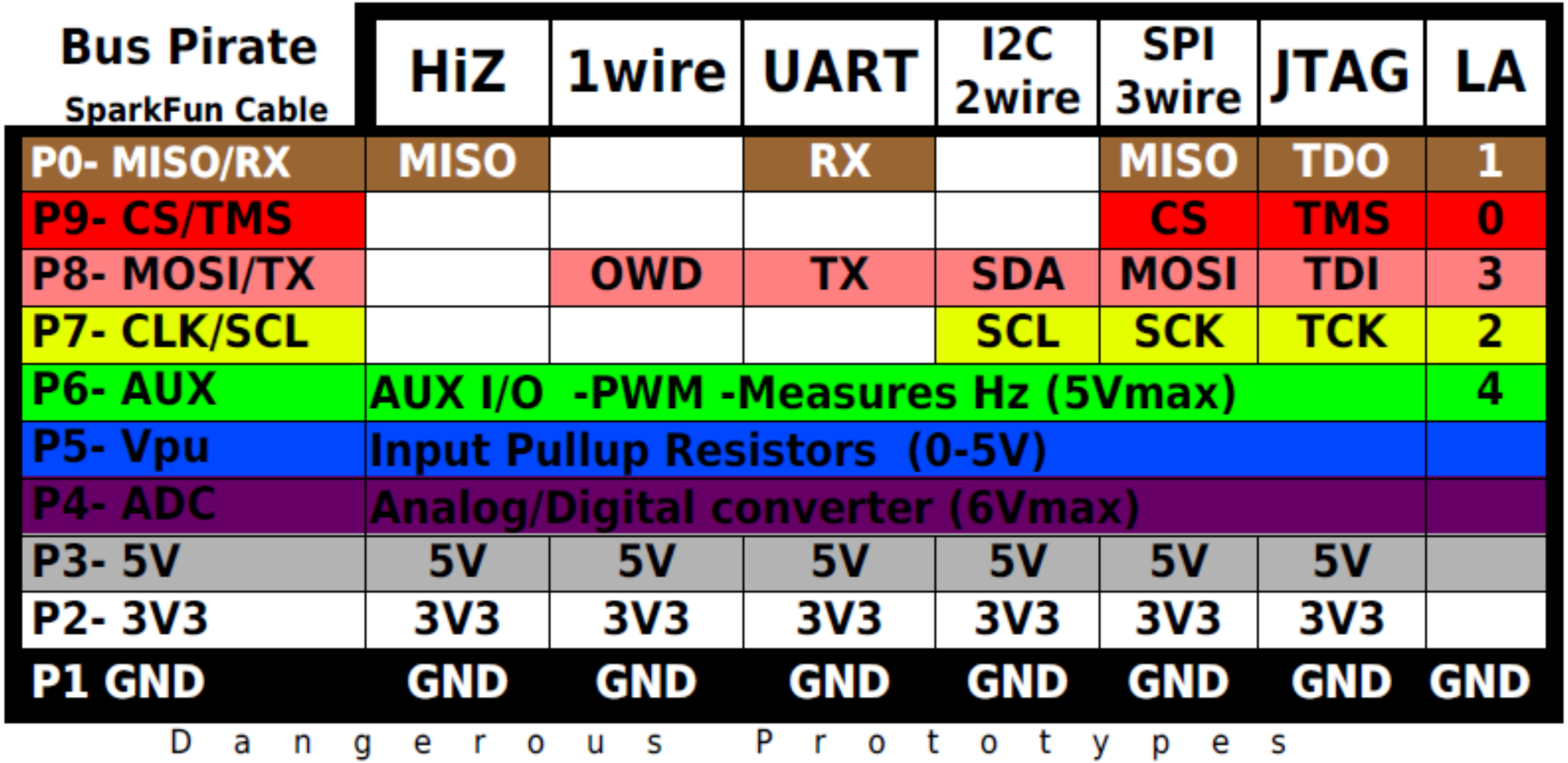
Task	Pre-BusPirate $$$$, 🕒🕒🕒	Bus Pirate $$, 🕒🕒🕒🕒	Post-BusPirate $$$, 🕒🕒	Beaglebone Hack $$, 🕒
Talk UART	RS232 hardware + level shifting	narrow tolerance	FT232R, just works, $$	native hardware
Interface I2C	**?**	**passable**	**Aardvark/Beagle - $$$**	**native hardware**
Dump SPI Flash				
Analyze Logic				
JTAG				

SPI

Serial Peripheral Interface

- FLASH chips
- SD Cards
- Sensors
- Displays
- more...

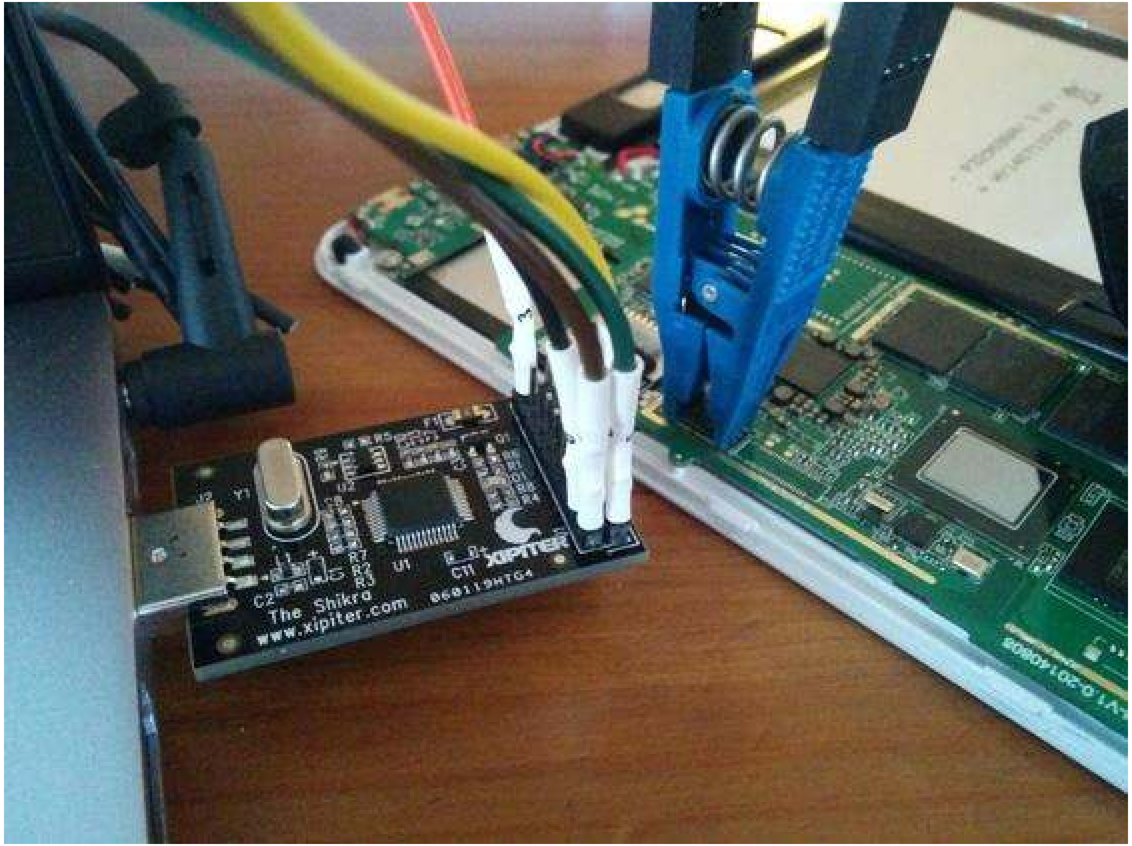
```
>echo BB-SPIDEV0 > /sys/devices/bone_capemgr.*/slots

>flashrom -p linux_spi:dev=/dev/spidev1.0 -r dumpfile.bin
```



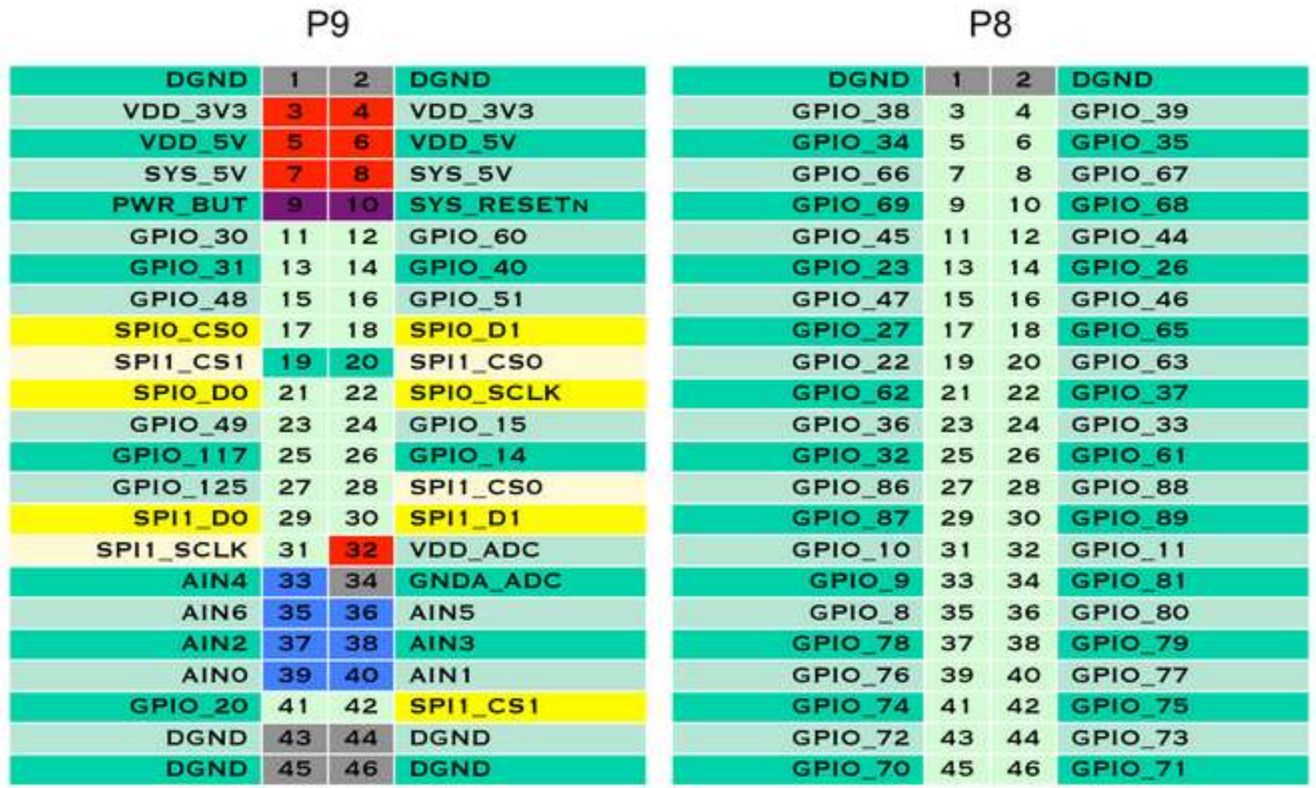
Bus Pirate SparkFun Cable	HiZ	1wire	UART	I2C 2wire	SPI 3wire	JTAG	LA
P0- MISO/RX	MISO		RX		MISO	TDO	1
P9- CS/TMS					CS	TMS	0
P8- MOSI/TX		OWD	TX	SDA	MOSI	TDI	3
P7- CLK/SCL				SCL	SCK	TCK	2
P6- AUX	AUX I/O -PWM -Measures Hz (5Vmax)						4
P5- Vpu	Input Pullup Resistors (0-5V)						
P4- ADC	Analog/Digital converter (6Vmax)						
P3- 5V	5V	5V	5V	5V	5V	5V	
P2- 3V3	3V3	3V3	3V3	3V3	3V3	3V3	
P1 GND	GND	GND	GND	GND	GND	GND	GND

Dangerous Prototypes

```
$ flashrom -p buspirate_spi:dev=/dev/ttyUSB0,spispeed=1M
```


The Shikra
www.xipiter.com
XIPITER

2 SPI ports



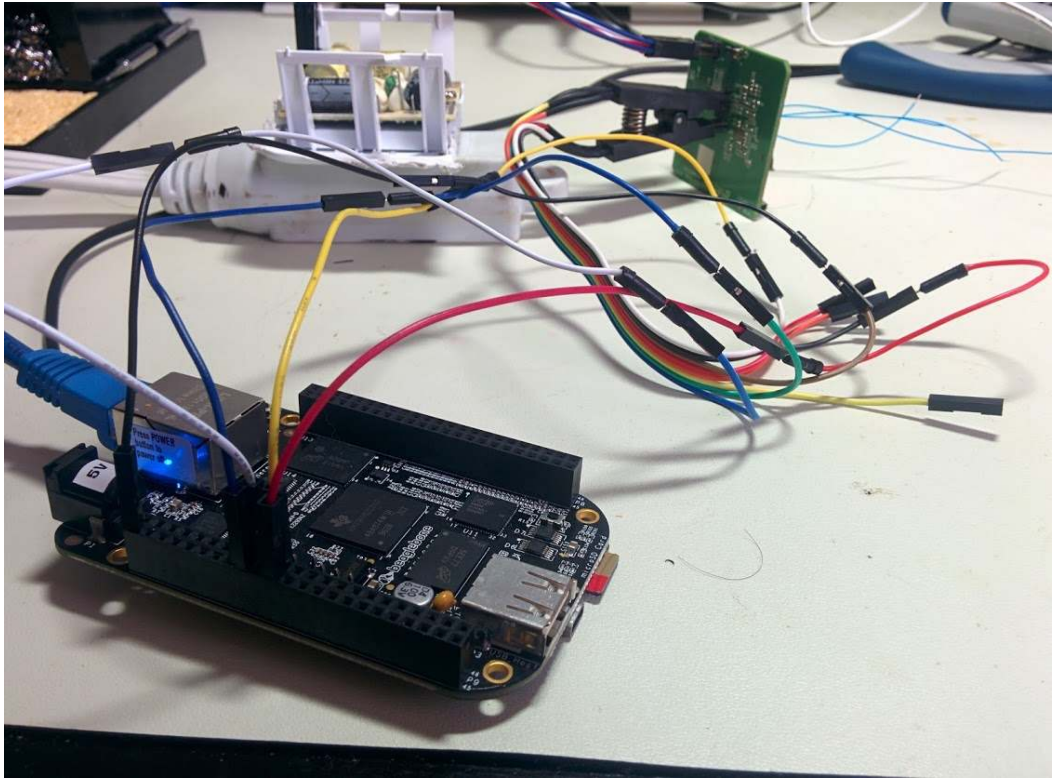
P9

DGND	1	2	DGND
VDD_3V3	3	4	VDD_3V3
VDD_5V	5	6	VDD_5V
SYS_5V	7	8	SYS_5V
PWR_BUT	9	10	SYS_RESETN
GPIO_30	11	12	GPIO_60
GPIO_31	13	14	GPIO_40
GPIO_48	15	16	GPIO_51
SPI0_CS0	17	18	SPI0_D1
SPI1_CS1	19	20	SPI1_CS0
SPI0_D0	21	22	SPI0_SCLK
GPIO_49	23	24	GPIO_15
GPIO_117	25	26	GPIO_14
GPIO_125	27	28	SPI1_CS0
SPI1_D0	29	30	SPI1_D1
SPI1_SCLK	31	32	VDD_ADC
AIN4	33	34	GNDA_ADC
AIN6	35	36	AIN5
AIN2	37	38	AIN3
AIN0	39	40	AIN1
GPIO_20	41	42	SPI1_CS1
DGND	43	44	DGND
DGND	45	46	DGND

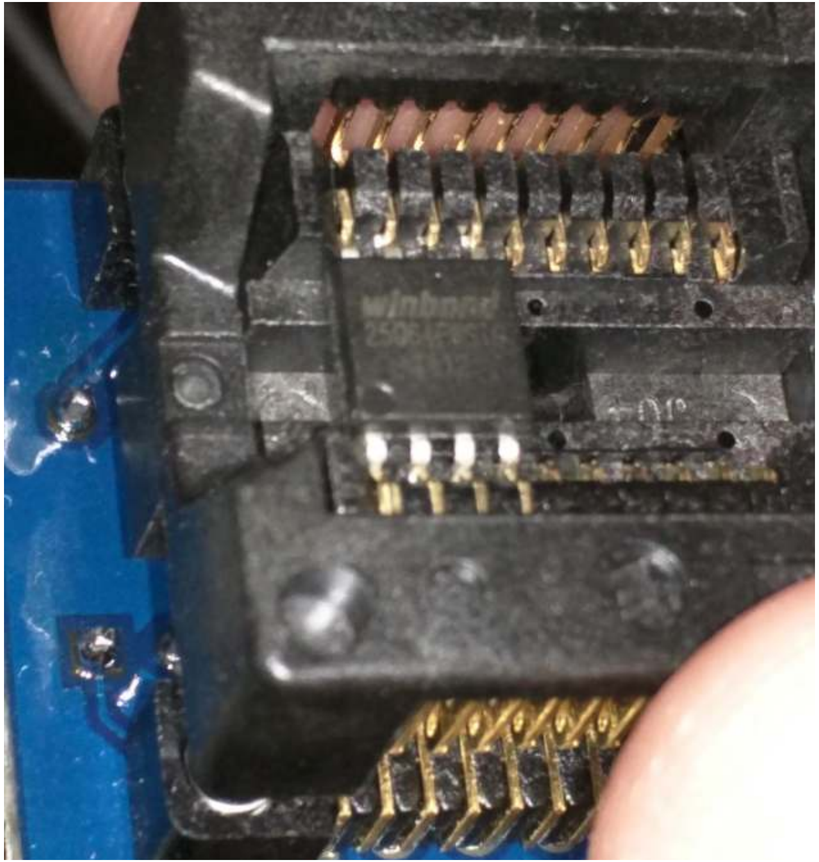
P8

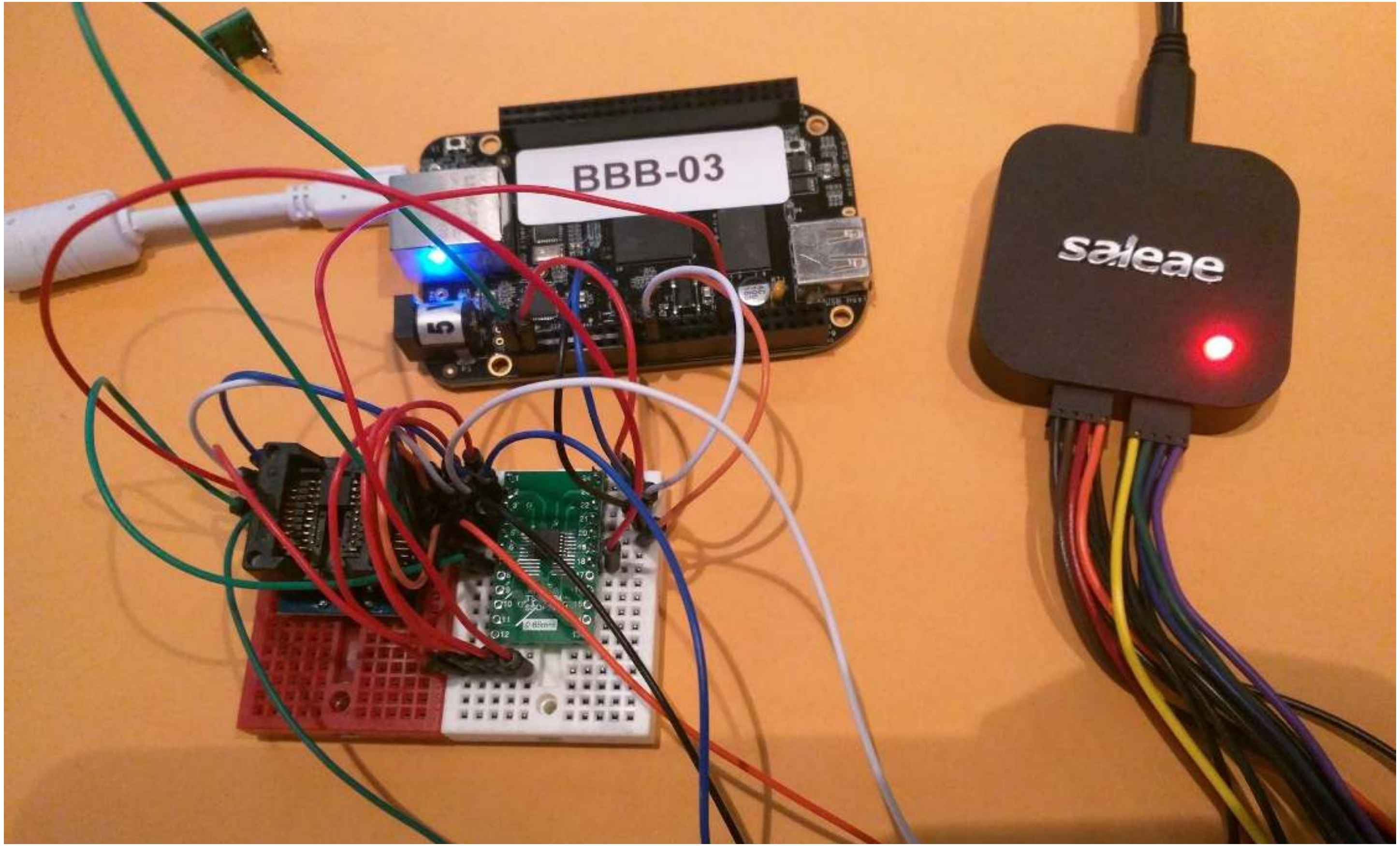
DGND	1	2	DGND
GPIO_38	3	4	GPIO_39
GPIO_34	5	6	GPIO_35
GPIO_66	7	8	GPIO_67
GPIO_69	9	10	GPIO_68
GPIO_45	11	12	GPIO_44
GPIO_23	13	14	GPIO_26
GPIO_47	15	16	GPIO_46
GPIO_27	17	18	GPIO_65
GPIO_22	19	20	GPIO_63
GPIO_62	21	22	GPIO_37
GPIO_36	23	24	GPIO_33
GPIO_32	25	26	GPIO_61
GPIO_86	27	28	GPIO_88
GPIO_87	29	30	GPIO_89
GPIO_10	31	32	GPIO_11
GPIO_9	33	34	GPIO_81
GPIO_8	35	36	GPIO_80
GPIO_78	37	38	GPIO_79
GPIO_76	39	40	GPIO_77
GPIO_74	41	42	GPIO_75
GPIO_72	43	44	GPIO_73
GPIO_70	45	46	GPIO_71

spi on the BBH



winbond
1430
U5
U8

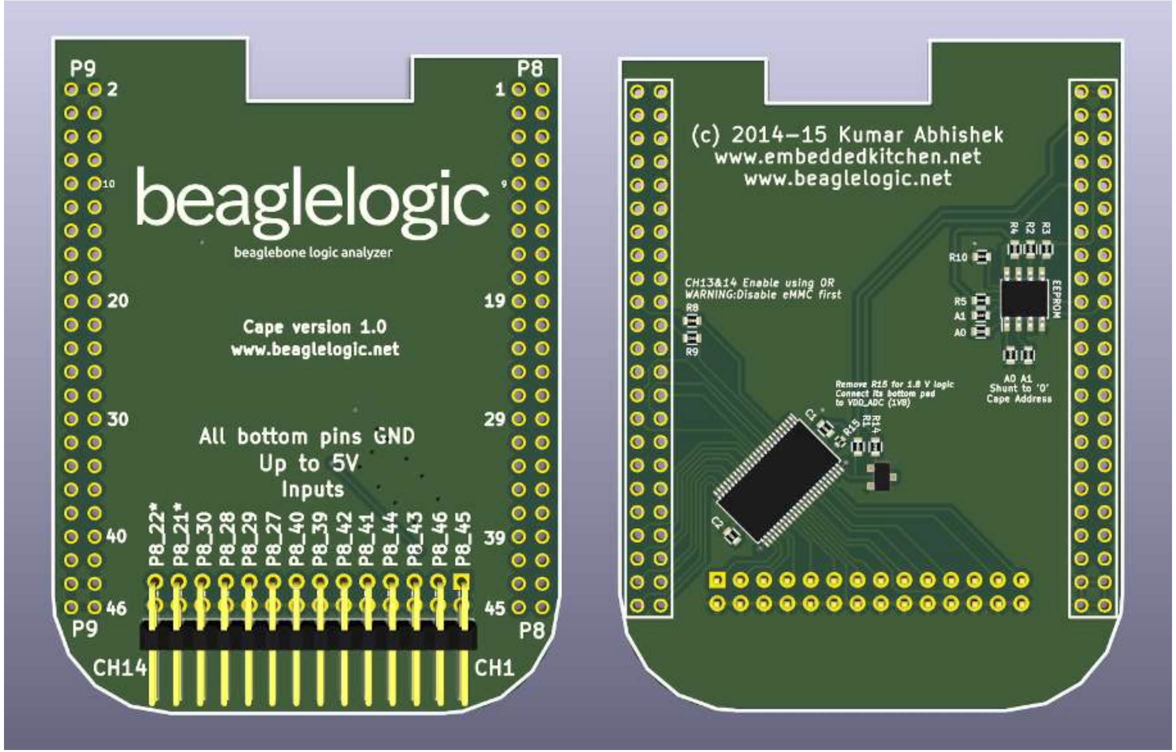

winbond


BBB-03
saleae

```
# echo BB-SPIDEV0 > /sys/devices/bone_capemgr.*/slots
# time flashrom -p linux_spi:dev=/dev/spidev1.0 -r dumpfile.bin
flashrom v0.9.8-r1888 on Linux 3.8.13-bone47 (armv7l)
flashrom is free software, get the source code at http://www.
flashrom.org

Calibrating delay loop... OK.
Found Spansion flash chip "S25FL208K" (1024 kB, SPI) on linux_spi.
===
<...snip...>
Reading flash... done.

real   0m2.616s
user   0m0.900s
sys 0m0.168s
```

Task	Pre-BusPirate $$$$, 🕓🕓🕓	Bus Pirate $$, 🕓🕓🕓🕓	Post-BusPirate $$$, 🕓🕓	Beaglebone Hack $$, 🕓
Talk UART	RS232 hardware + level shifting	narrow tolerance	FT232R, just works, $$	native hardware
Interface I2C	?	passable	Aardvark/Beagle - $$$	native hardware
Dump SPI Flash	**Universal Programmer $$$$**	**slow** 🕓🕓🕓🕓🕓🕓🕓🕓 🕓🕓	**ft232H, $$** **teensy/arduino $$**	**native hardware insanely fast ~~🕓~~**
Analyze Logic				
JTAG				

logic analyzer - beaglelogic



logic analyzer - beaglelogic

BeagleLogic turns your BeagleBone [Black] into a 14-channel, 100Msps Logic Analyzer. Once loaded, it presents itself as a character device node /dev/beaglelogic.

- 'beaglelogic' kernel module
- two Programmable Real-Time Units (PRUs)
- works with the sigrok library

https://github.com/abhishek-kakkar/BeagleLogic

logic analyzer - sigrok

```
>echo BB-BEAGLELOGIC > /sys/devices/bone_capemgr.*/slots

>modprobe beagelogic

>echo 33554432 > /sys/devices/virtual/misc/beaglelogic/memalloc
```

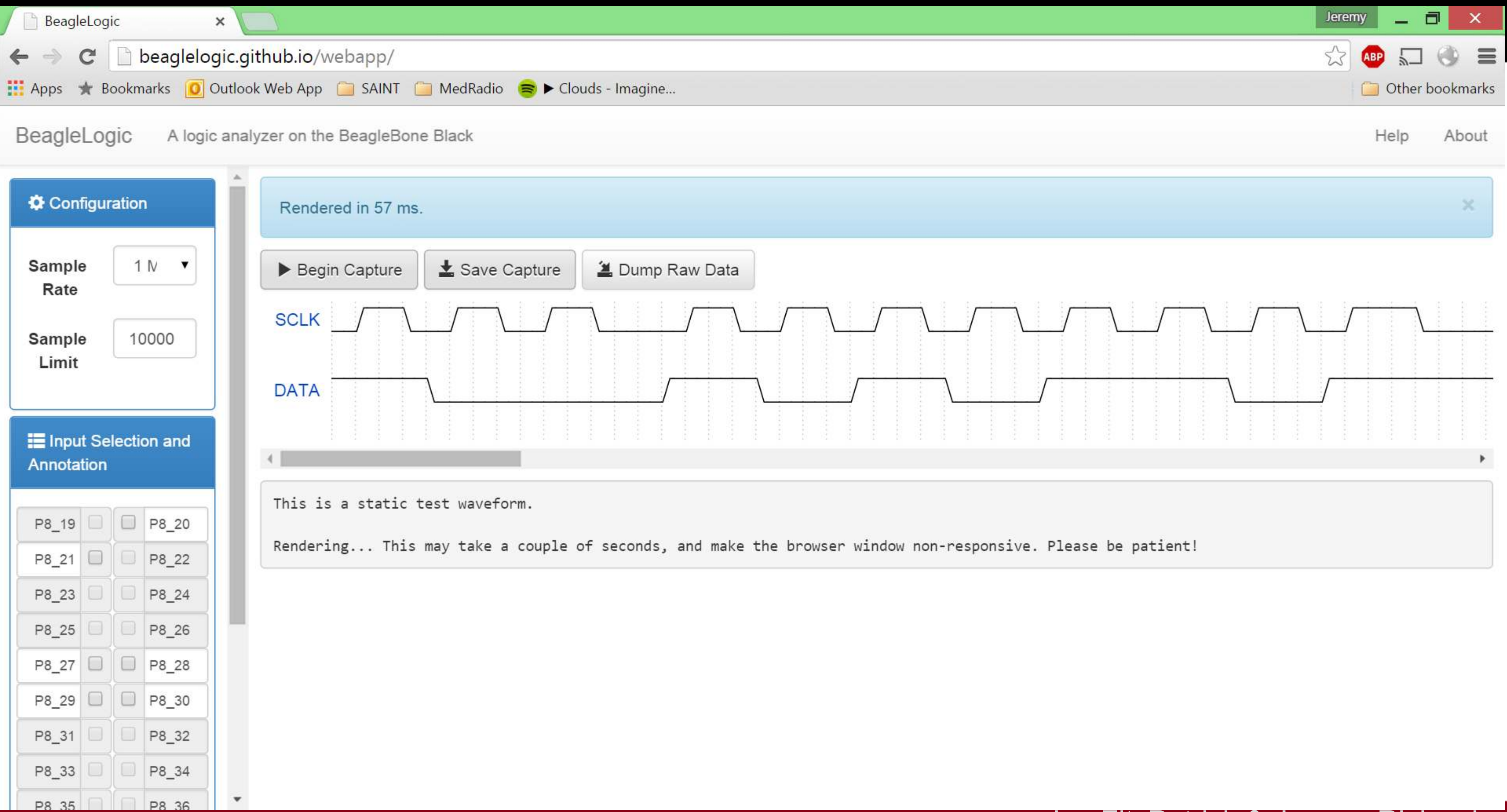
logic analyzer - sigrok

Basic raw captures with dd

```
>dd if=/dev/beaglelogic of=mydump bs=1M count=1
```

sigrok support

```
>sigrok-cli --time 10s -o test-capture-1.sr  -d beaglelogic  -c samplerate=500khz  --channels P8_45,P8_46
```



Sample Rate: 1 M

Sample Limit: 10000

Rendered in 57 ms.

```
This is a static test waveform.

Rendering... This may take a couple of seconds, and make the browser window non-responsive. Please be patient!
```

logic analyzer - 12 (+2) chan

```
exclusive-use =
//    "P8.20",    /* pru1: pr1_pru1_pru_r31_13 */
//    "P8.21",    /* pru1: pr1_pru1_pru_r31_12 */
      "P8.27",    /* pru1: pr1_pru1_pru_r31_8  */
      "P8.28",    /* pru1: pr1_pru1_pru_r31_10 */
      "P8.29",    /* pru1: pr1_pru1_pru_r31_9  */
      "P8.30",    /* pru1: pr1_pru1_pru_r31_11 */
      "P8.39",    /* pru1: pr1_pru1_pru_r31_6  */
      "P8.40",    /* pru1: pr1_pru1_pru_r31_7  */
      "P8.41",    /* pru1: pr1_pru1_pru_r31_4  */
      "P8.42",    /* pru1: pr1_pru1_pru_r31_5  */
      "P8.43",    /* pru1: pr1_pru1_pru_r31_2  */
      "P8.44",    /* pru1: pr1_pru1_pru_r31_3  */
      "P8.45",    /* pru1: pr1_pru1_pru_r31_0  */
      "P8.46",    /* pru1: pr1_pru1_pru_r31_1
```

logic analyzer - sigrok protocol decoders

```
> sigrok-cli -i test-capture-2.sr  -P uart:baudrate=115200:parity_type=none -B uart
```

Above is a UART example. sigrok can also decode CAN (automotive), i2c, JTAG, modbus, 1wire, parallel, sdcard spi, spi flash, SWD, USB packet

A full list of protocols with decoders is available here:

http://sigrok.org/wiki/Protocol_decoders

Task	Pre-BusPirate $$$$, 🕓🕓🕓	Bus Pirate $$, 🕓🕓🕓🕓	Post-BusPirate $$$, 🕓🕓	Beaglebone Hack $$, 🕓
Talk UART	RS232 hardware + level shifting	narrow tolerance	FT232R, just works, $$	native hardware
Interface I2C	?	passable	Aardvark/Beagle - $$$	native hardware
Dump SPI Flash	Universal Programmer $$$$	slow 🕓🕓🕓🕓🕓🕓🕓🕓🕓🕓	ft232H, $$ teensy/arduino $$	native hardware insanely fast ~~🕓~~
Analyze Logic	Benchtop equipment $$$$	limited capture	saleae $$$	native hardware
JTAG				

JTAG - Work in Progress

OpenOCD has a driver for toggling GPIO via Sysfs:

```
interface sysfsgpio
# Each of the JTAG lines need a gpio number set: tck tms tdi tdo
# Header pin numbers: ## ## ## ##

sysfsgpio_jtag_nums ## ## ## ##
# At least one of srst or trst needs to be specified

# Header pin numbers: TRST - ##, SRST - ##
sysfsgpio_trst_num ##
sysfsgpio_srst_num ##
```

JTAG - Work in Progress

To use it:

```
# echo BB-JTAG > /sys/devices/bone_capemgr.*/slots
# openocd -f sysfsgpio-bbb.cfg -f target.cfg
```

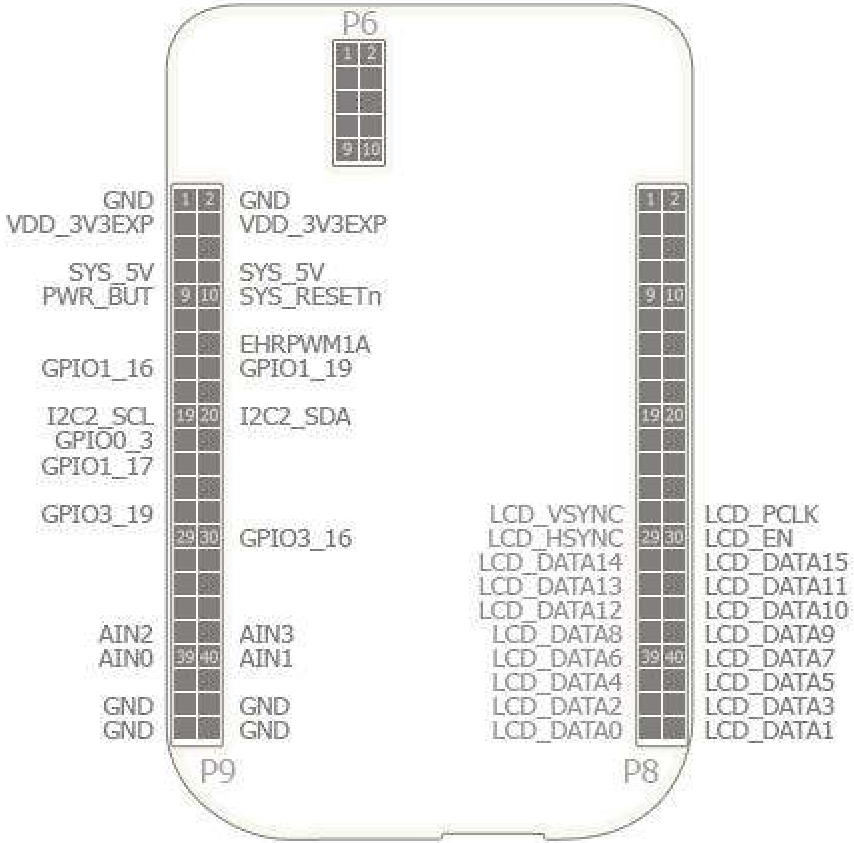
Task	Pre-BusPirate $$$$, 🕓🕓🕓	Bus Pirate $$, 🕓🕓🕓🕓	Post-BusPirate $$$, 🕓🕓	Beaglebone Hack $$, 🕓
Talk UART	RS232 hardware + level shifting	narrow tolerance	FT232R, just works, $$	native hardware
Interface I2C	?	passable	Aardvark/Beagle - $$$	native hardware
Dump SPI Flash	Universal Programmer $$$$	slow 🕓🕓🕓🕓🕓🕓🕓🕓🕓🕓	ft232H, $$ teensy/arduino $$	native hardware insanely fast ~~🕓~~
Analyze Logic	Benchtop equipment $$$$	limited capture	saleae $$$	native hardware
JTAG	**Vendor-supplied $$$$$**	**flakey**	**ft232h $$**	**GPIO via sysfs perf. like ft232h**

Beaglebone Capes

Allow expandability onto the BBB

Have an EEPROM so they're auto-detected

GPIOs are configured automatically!



Why a cape?

It’s nice to have clearly labeled headers for UART, SPI, JTAG, etc…

It’s nice to buffer your I/O so you don’t kill your BBB

It’s really nice to have level shifting to let us use 1.8 to 5.5 on our pins!

Design Decisions

BBB I/O is 3.3v

It’s NOT 5v tolerant - wires we poke around with should be

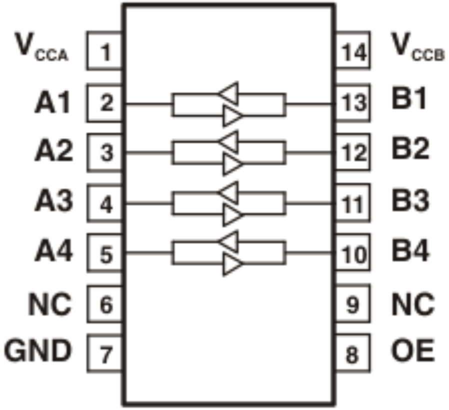
Level Shifting up OR down is pretty straightforward

But for this part (and MANY others):

VCCA ≤ VCCB

We can’t have VCCA=3.3V and VCCB=1.8V to 5V

D OR PW PACKAGE
(TOP VIEW)

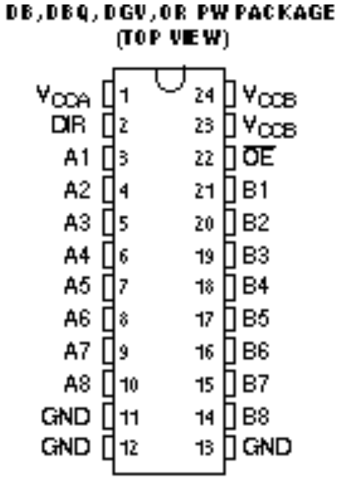


Suitable Parts

SN74LVC8T245 and SN74LVCH16T245:

V_{CCA}: A-port supply voltage. 1.65 V ≤ V_{CCA} ≤ 5.5 V

V_{CCB}: B-port supply voltage. 1.65 V ≤ V_{CCB} ≤ 5.5 V



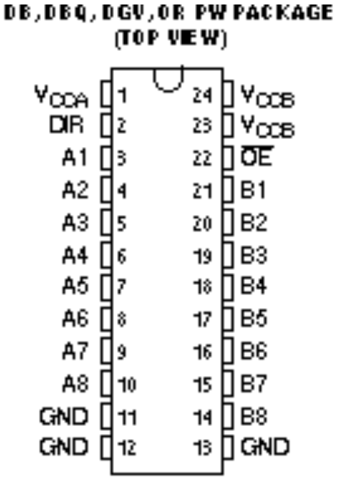
Suitable Parts

SN74LVC8T245 and SN74LVCH16T245:

V_{CCA}: A-port supply voltage. 1.65 V ≤ V_{CCA} ≤ 5.5 V

V_{CCB}: B-port supply voltage. 1.65 V ≤ V_{CCB} ≤ 5.5 V

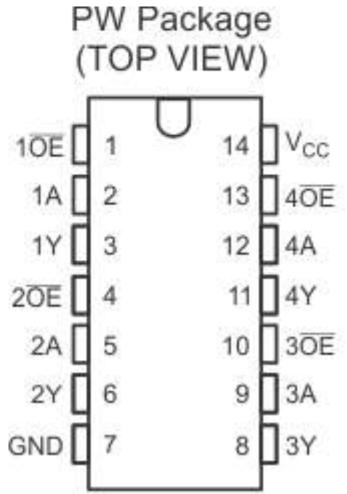
Bidirectional - but direction has to be set



Suitable Parts

SN74LV4T125:

- Up Translation
  - 1.2 V(1) to 1.8 V at 1.8-V Vcc
  - 1.5 V(1) to 2.5 V at 2.5-V Vcc
  - **1.8 V(1) to 3.3 V at 3.3-V Vcc**
  - 3.3 V to 5.0 V at 5.0-V Vcc
- Down Translation
  - 3.3 V to 1.8 V at 1.8-V Vcc
  - 3.3 V to 2.5 V at 2.5-V Vcc
  - **5.0 V to 3.3 V at 3.3-V Vcc**


PW Package
(TOP VIEW)
$1\overline{\text{OE}}$ 1
1A 2
1Y 3
$2\overline{\text{OE}}$ 4
2A 5
2Y 6
GND 7
14 V_{CC}
13 $4\overline{\text{OE}}$
12 4A
11 4Y
10 $3\overline{\text{OE}}$
9 3A
8 3Y

Unidirectional - but single supply!

Suitable Parts

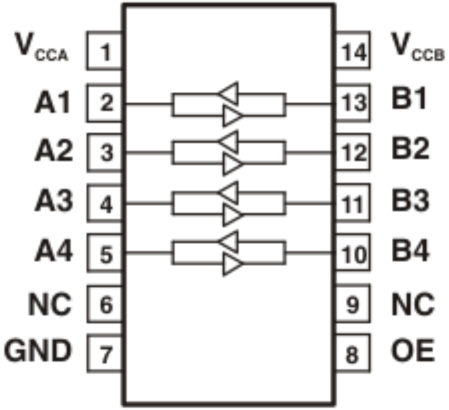
Bidirectional up and down solution?

Does not seem to exist :(

We could:

- Shift up to 5v, then down to 1.8-5
- Have separate Up and Down translation
- Translate down, and protect the inputs with zener diodes

D OR PW PACKAGE
(TOP VIEW)



Design in Development

SN74LVCH16T245

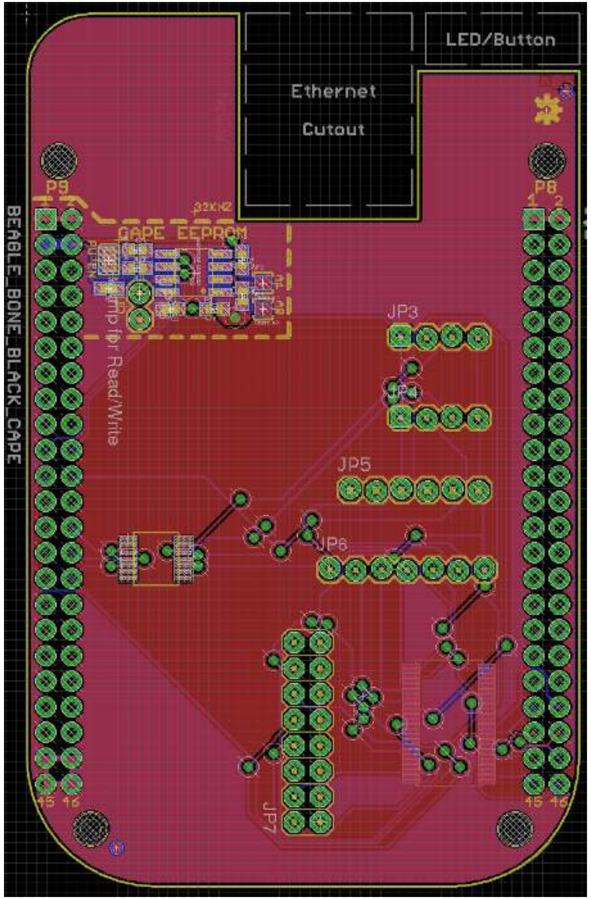
for Beaglelogic and all output-only signals

SN74LV4T125

for input-only signals

TXS0102

for bidirectional signals, with zener diodes



Task	Pre-BusPirate $$$$, 🕓🕓🕓	Bus Pirate $$, 🕓🕓🕓🕓	Post-BusPirate $$$, 🕓🕓	Beaglebone Hack $$, 🕓
Talk UART	RS232 hardware + level shifting	narrow tolerance	FT232R, just works, $$	native hardware
Interface I2C	?	passable	Aardvark/Beagle - $$$	native hardware
Dump SPI Flash	Universal Programmer $$$$	slow 🕓🕓🕓🕓🕓🕓🕓🕓🕓🕓	ft232H, $$ teensy/arduino $$	native hardware insanely fast ~~🕓~~
Analyze Logic	Benchtop equipment $$$$	limited capture	saleae $$$	native hardware
JTAG	Vendor-supplied $$$$$	flakey	ft232h $$	GPIO via sysfs perf. like ft232h

Future Ideas

Facedancer functionality (USB MITM) - This is partially working as part of the USB proxy project (https://github.com/dominicgs/USBProxy)

BusPirate emulation over GPIO

Sigrok cloud decoding (REST web service)

JTAG identification via GPIO + Logic Analyzer/Decoders

Final Tips

If you’re getting unexpected output you may not have enough power. The BeagleBone can be powered by 5v power adapter if its not getting enough power over USB.

If your wires are long you might get some strange results when dumping (eg spi). You can increase stability by reducing the speed.

```
... -p linux_spi:dev=/dev/spidev1.0,spispeed=1000 ...
```

spispeed is in khz, so 1000 =1mhz

Final Tips

Know what bone firmware you are running. Configuring the tools differs between versions.

Have a second one… or 10. It is the best way to test/debug/do a sanity check.

Q&A