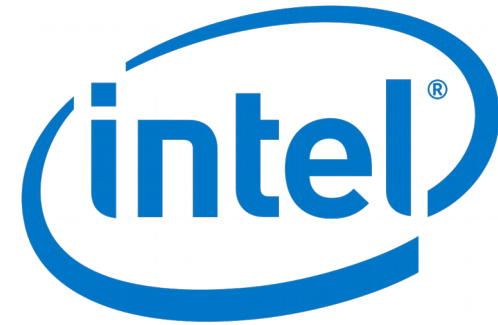


A close-up photograph of an embedded system board, likely a Raspberry Pi or similar single-board computer. The board is populated with various components, including several LEDs (red, green, yellow) and a blue microSD card. The text "Embedded Apprentice Linux Engineer" is overlaid in white, bold, sans-serif font across the center of the image. The background is dark and slightly blurred, emphasizing the components in the foreground.

Embedded Apprentice Linux Engineer

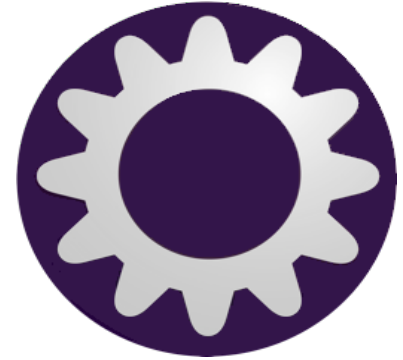
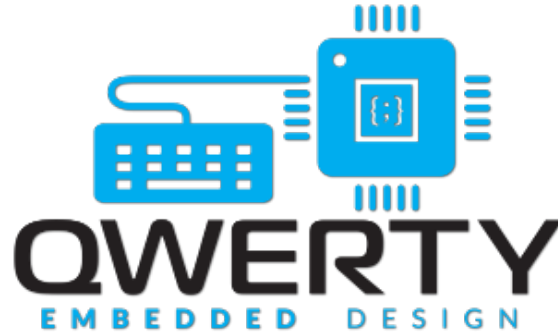
Our Conference Sponsors



Our Hardware Sponsors



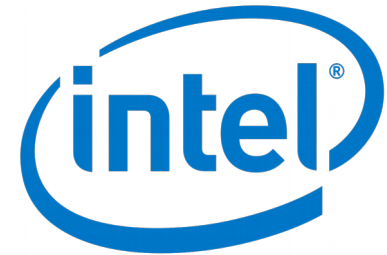
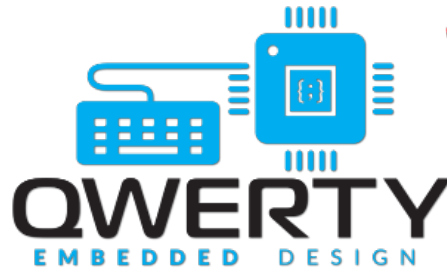
electronics



OSH Park



Our Training Sponsors



Before we do anything....

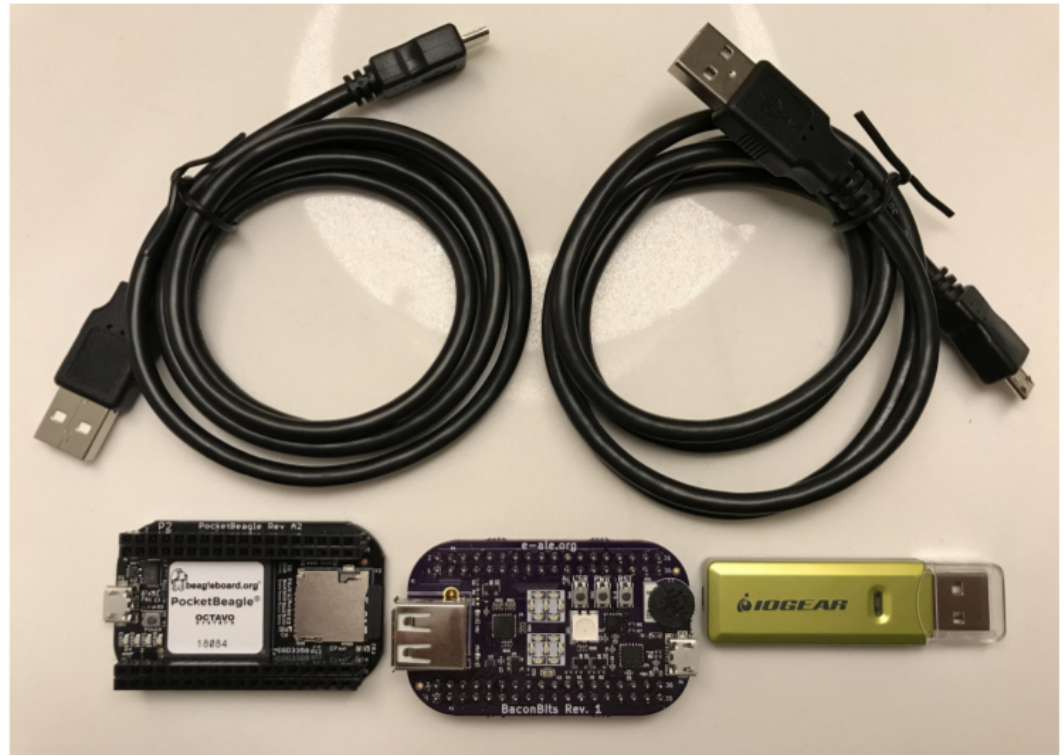
```
# This is going to take a little bit and we need to start now.  
wget https://downloads.toganlabs.com/e-ale/e-ale-setup.sh .  
chmod +x e-ale-setup.sh  
./e-ale-setup.sh
```

Quick Introduction

- Beth 'pidge' Flanagan
- CTO of Togán Labs (www.toganlabs.com)
- Former release engineer for the Yocto Project
- SW dev for 25 years, embedded for 10
- If anything goes wrong with this talk, it's Behan's fault.

Now we can open things up

- 2x microUSB cables
- 1 uSD reader/writer
- 1 pocketbeagle
- 1 BaconBits cape
- 1 uSD card
- 1 uSD to SD adapter



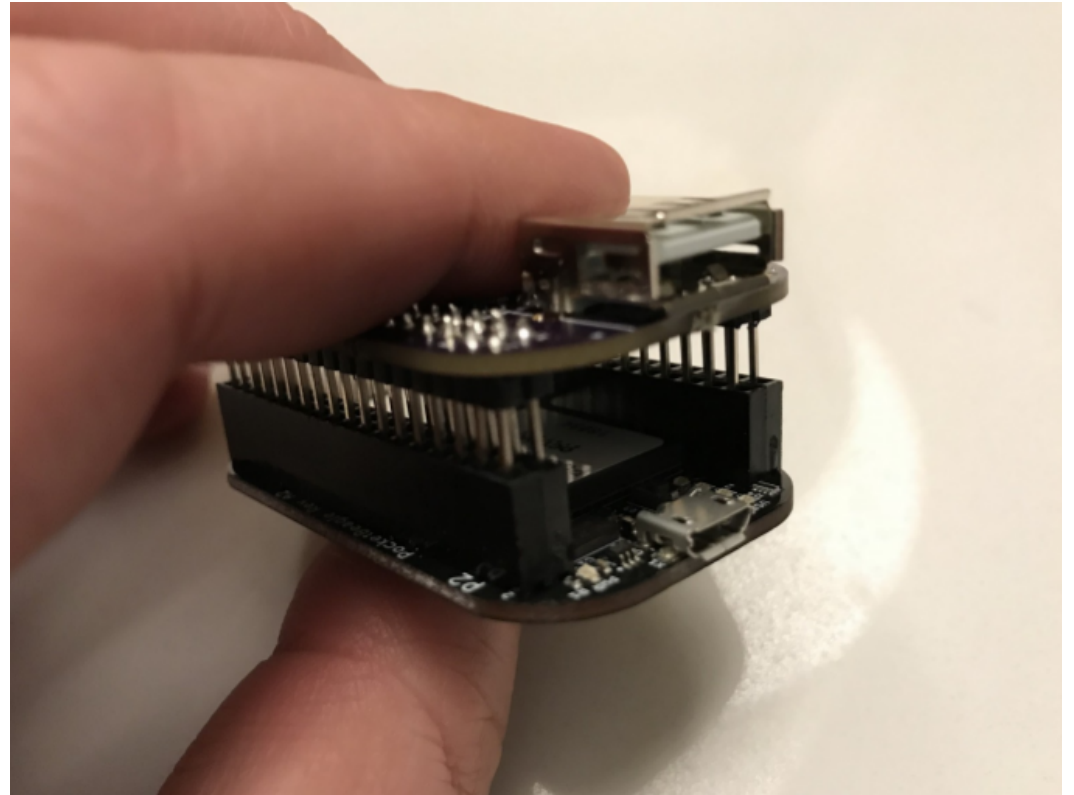
The hardware

- BaconBits on the top
- Pocketbeagle on the bottom



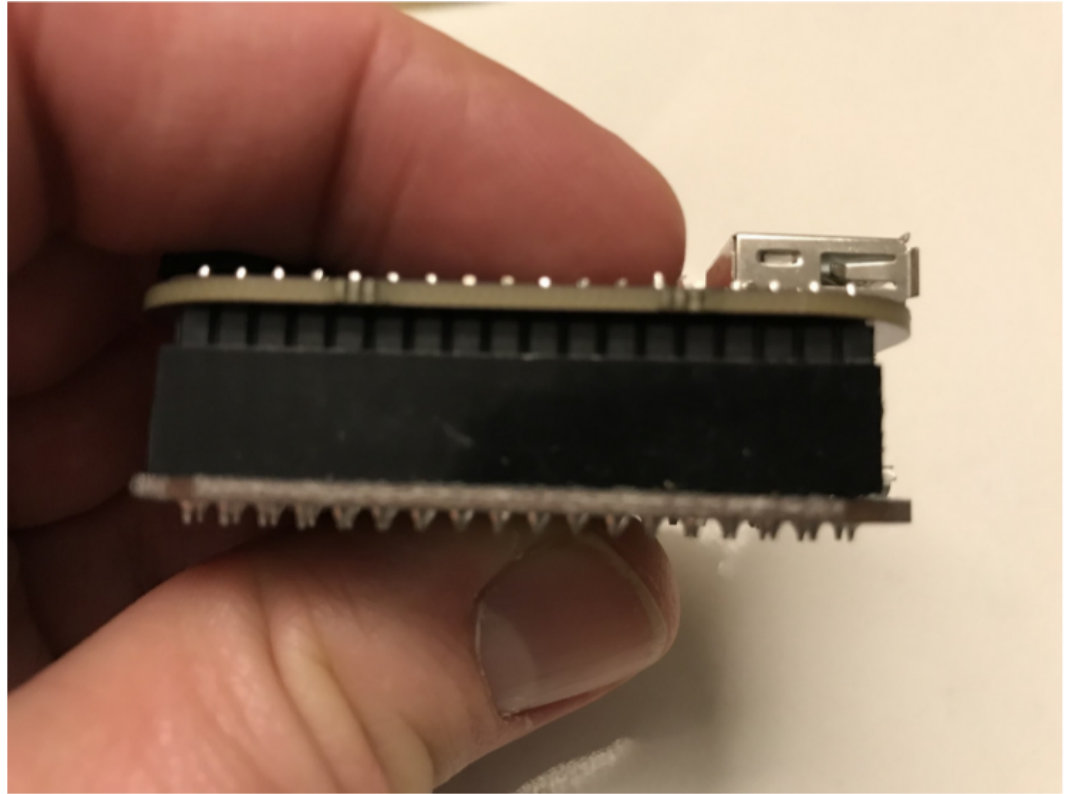
Putting the hardware together

- USB type-A (the big one) on the cape is on the same end as the micro-USB (the wee one) on the pocketbeagle
- If you get this wrong they're difficult to get apart.



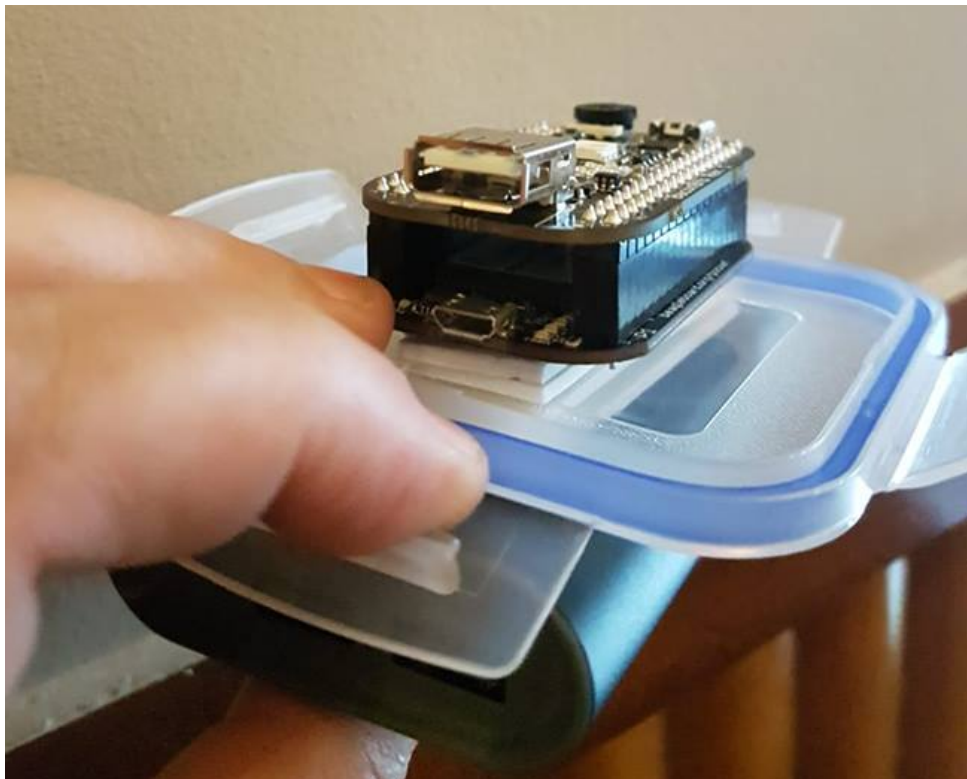
What it should look like now.

- CAREFULLY align the pins, push them together.



What mine looks like

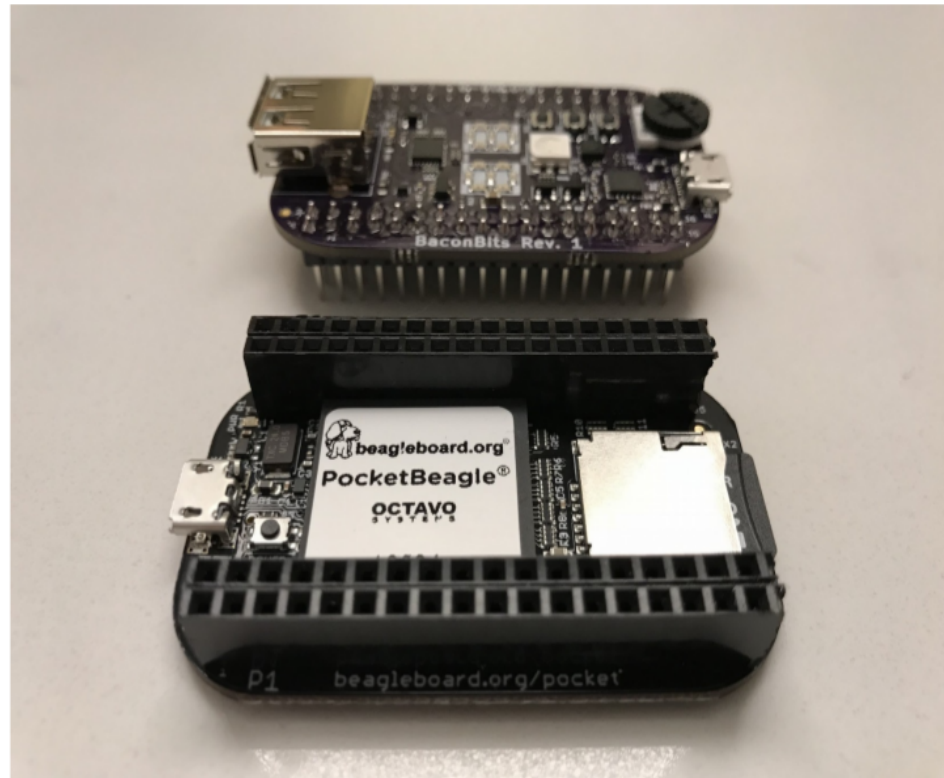
- I dislike Altoids tins
- I also approve of waterproofing things
- And dedicated USB batteries
- Provides opportunities to explain embedded linux to airport security.



Comms

- BaconBits provides serial console (we'll be using this)
- Pocketbeagle provides Ethernet over USB
- Both ports can provide power.

18



Let's blast some things to the uSD! ^{*e-ale*}

Why I use dd but I'm going to ask *e-ale* you to use etcher.io

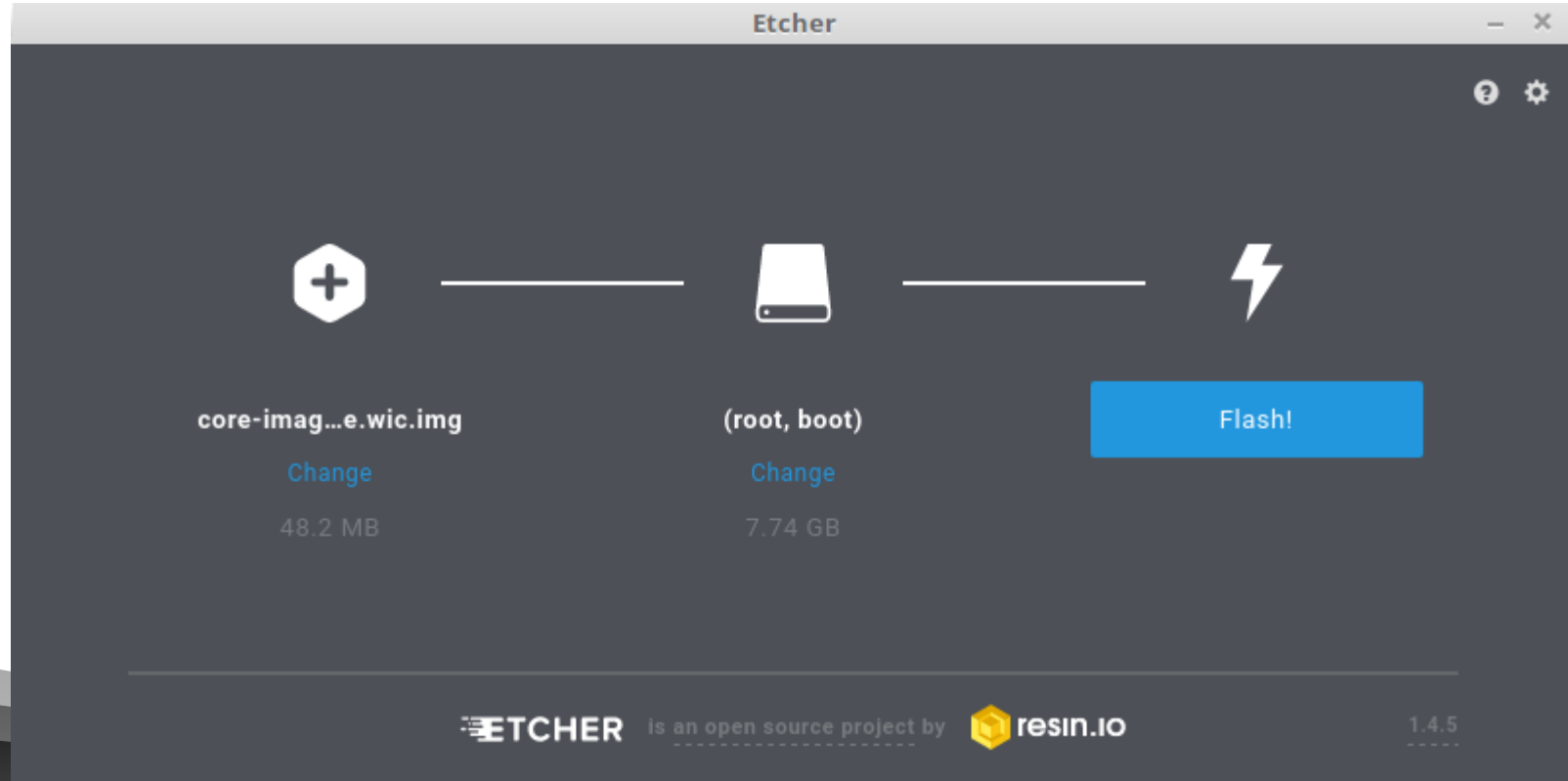
- dd is a wonderful command line tool to create sd cards. BUT.
- Long ago, I use to sync `.bash_history` between my dev machines
- laptop's usb was listed as `/dev/sdb`.
- Desktops / was `/dev/sdb`.
- Ctrl+shift+r for dd and BOOM!
- Hybrid SuSE/poky generic-x86-64 OS until I could reinstall.

Conclusion. Use Etcher.io today EVEN IF YOU KNOW dd.

Let's blast some things to the uSD!

- Insert sd card reader + sd card
- From terminal: `~/e-ale-intro/etcher-electron-1.4.5-x86_64.AppImage`

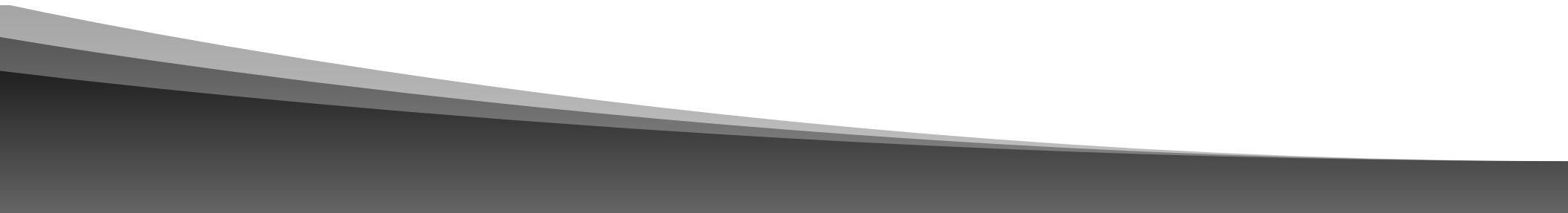
Let's blast some things to the uSD!



Let's blast some things to the uSD!

- Insert sd card reader + sd card
- From terminal: `~/e-ale-intro/etcher-electron-1.4.5-x86_64.AppImage`

And now we wait.



When etcher is done

- Remove uSD card from reader/writer
- Insert into the pocketbeagle uSD slot gold pads down.
- You should feel a click.
- Take one USB cable and insert into your laptop and the other end into the microusb of the BaconBits (the top board).
- This will give us a serial console

BOOT

- I'm going to teach minicom and moving files manually:
 - There are a bunch of ways to do this (kermit, screen, ethernet over USB, uboot, nfs, tftp, etc. etc.)
 - But this is the dirt simple thing that doesn't require setup and just about every board in the world with an SD card also supports a serial console.
 - But it also annoys Behan, which is a bonus
- Run "sudo minicom" in a terminal

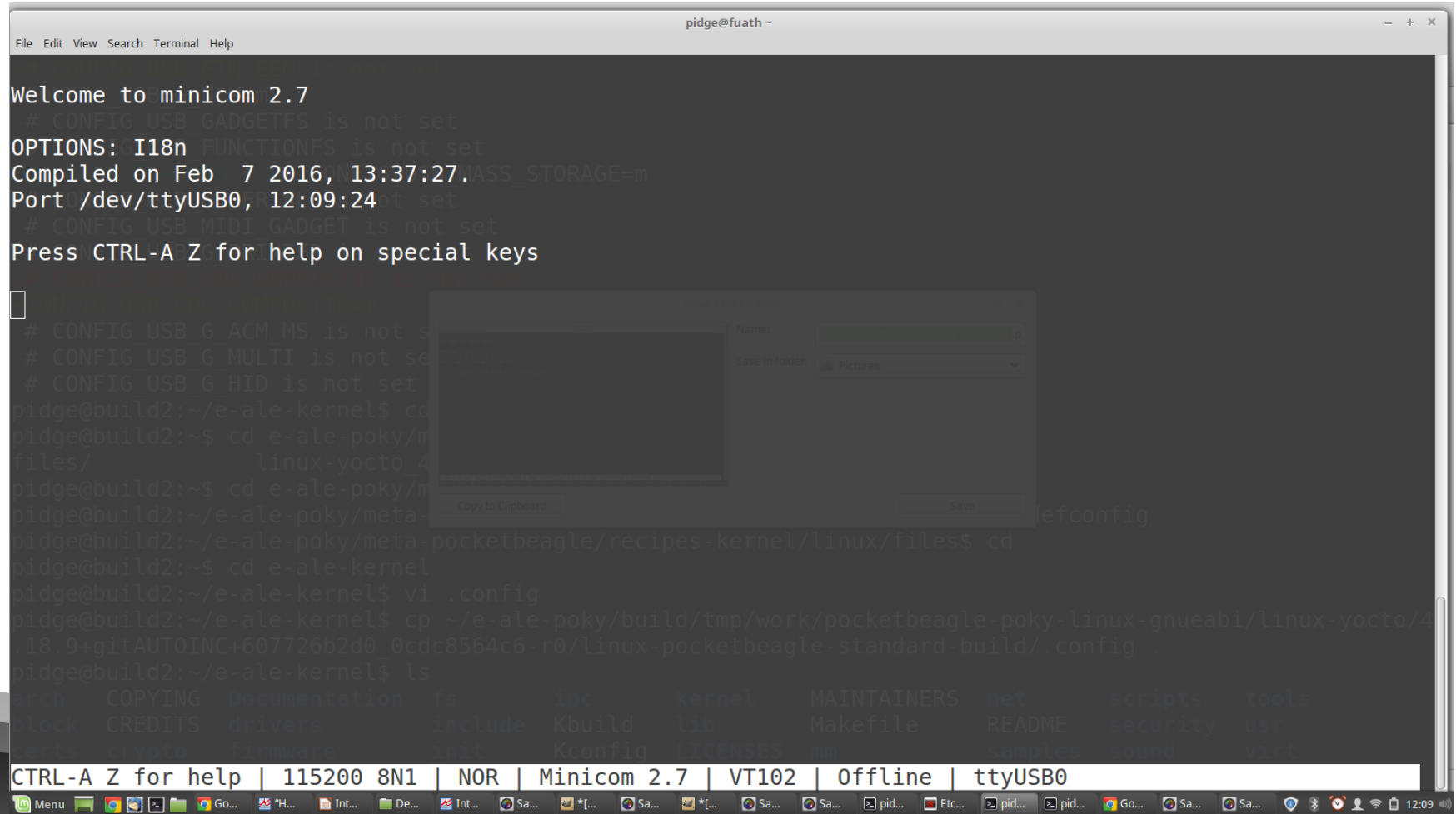
File Edit View Search Terminal Help

pidge@fuath ~

```
Welcome to minicom 2.7
# CONFIG_USB_GADGETFS is not set
OPTIONS: I18n
FUNCTIONFS is not set
Compiled on Feb  7 2016, 13:37:27.
ASS_STORAGE=m
Port /dev/ttyUSB0, 12:09:24
# CONFIG_USB_MIDI_GADGET is not set
Press CTRL-A Z for help on special keys

# CONFIG_USB_G_ACM_MS is not set
# CONFIG_USB_G_MULTI is not set
# CONFIG_USB_G_HID is not set
pidge@build2:~/e-ale-kernel$ cd
pidge@build2:~$ cd e-ale-poky/meta-
files/
linux-yocto-4
pidge@build2:~$ cd e-ale-poky/meta-
pidge@build2:~/e-ale-poky/meta-pocketbeagle/recipes-kernel/linux/files$ cd
pidge@build2:~$ cd e-ale-kernel
pidge@build2:~/e-ale-kernel$ vi .config
pidge@build2:~/e-ale-kernel$ cp ~/e-ale-poky/build/tmp/work/pocketbeagle-poky-linux-gnueabi/linux-yocto/4
.18.9+gitAUTOINC+607726b2d0_0cdc8564c6-r0/linux-pocketbeagle-standard-build/.config .
pidge@build2:~/e-ale-kernel$ ls
arch  COPYING  Documentation  fs  ipc  kernel  MAINTAINERS  net  scripts  tools
block  CREDITS  drivers  include  Kbuild  lib  Makefile  README  security  usr
firmware  init  Kconfig  LICENSES  mm  samples  sound  virt

CTRL-A Z for help | 115200 8N1 | NOR | Minicom 2.7 | VT102 | Offline | ttyUSB0
```



The image shows a terminal window titled 'pidge@fuath ~'. The terminal displays the output of a minicom session, including a welcome message and configuration options. A file save dialog is overlaid on the terminal, showing a file name field and a 'Save in folder' dropdown menu set to 'Pictures'. The dialog has 'Copy to Clipboard' and 'Save' buttons. The terminal text is partially obscured by the dialog. At the bottom of the terminal, a status bar shows 'CTRL-A Z for help | 115200 8N1 | NOR | Minicom 2.7 | VT102 | Offline | ttyUSB0'. The desktop environment at the bottom of the image shows a taskbar with various application icons and a system tray with a clock showing 12:09.

BOOT

- Take the second USB cable and plug it into the pocketbeagle and your laptop.
- You should see a login prompt!
- Username: root
- Password: <empty>

It's ALIVE!

Now, let's do something with it

Setting up the toolchain

```
cd ~/e-ale-intro
tar xvf gcc-linaro-7.3.1-2018.05-x86_64_arm-linux-gnueabi.tar.xz
export PATH=~/e-ale-intro/gcc-linaro-7.3.1-2018.05-x86_64_arm-linux-gnueabi/bin:$PATH
arm-linux-gnueabi-gcc -v
```

Let's make a kernel module

```
cd ~/e-ale-intro/basic-kernel-module;  
make KERNEL=~/.e-ale-intro/linux-kernel/ CROSS=~/.e-ale-intro/gcc-linaro-7.3.1-2018.05-  
x86_64_arm-linux-gnueabi/bin/arm-linux-gnueabi-  
ls
```

Looking deeper

```
PWD := $(shell pwd)
obj-m += hello.o
```

```
all:
```

```
    make ARCH=arm CROSS_COMPILE=$(CROSS) -C $(KERNEL) M=$(PWD) modules
```

```
clean:
```

```
    make -C $(KERNEL) M=$(PWD) clean
```

Looking deeper

```
PWD := $(shell pwd)
obj-m += hello.o
```

all:

```
make ARCH=arm CROSS_COMPILE=<location of toolchain and prefix> -C <location of
kernel source (change directory, pull in that Makefile)> M=<location of external module>
modules
```

clean:

```
make -C $(KERNEL) M=$(PWD) clean
```

Getting it onto the device the lazy *e-ale* way

Unplug **just** the pocketbeagle (Leave the baconbits cape plugged in)
Eject uSD card and put it into your laptop

```
cp ~/e-ale-intro/basic-kernel-module/hello.ko <whereever your laptop mounted the sd card>/tmp
```

```
#very important!
```

```
#
```

```
# See: Eat My Data: How Everybody gets File IO Wrong
```

```
# by Stewart Smith
```

```
sync
```

Getting it onto the device the lazy way

Put SD card back in. Plugin the pocketbeagle. Wait for everything to boot and then log in

```
cd /tmp
# hello.ko should be here in tmp
ls
# hello shouldn't be here. If so, you are MAGIC or I did something wrong
lsmod |grep hello
# Let's install it
insmod /tmp/hello.ko
# hello should be here. If not, either you or I did something wrong
lsmod |grep hello
# Let's remove it
rmmod hello; lsmod |grep hello
```

More Resources

<https://github.com/e-ale/meta-pocketbeagle>

<https://github.com/e-ale/basic-kernel-module>

<https://github.com/e-ale/linux-kernel>

Special thanks for Jason Kridner's images (and hands)

(One of the best talks I've ever seen)

https://www.flamingspork.com/talks/2007/06/eat_my_data.odp