Beaglebone: The Perfect Telemetry Platform?

Matt Ranostay
ELC 2013
Ranostay Industries
http://ranostay.org
mranostay@gmail.com

What is the Beaglebone?

- Low cost AM335x SoC platform
  - $89 from various source (Mouser, Digikey, Amazon)
- 3.3v I/O can interface with many ICs and sensors out of the box, or with cheap 3.3v to 1.8v/5v logic converter
  - I2C
  - SPI
  - GPIO
- 1.8v I/O Analog In (ADC) pins

Introduction to Capes

- Daughter cards that connect to the expansion headers
- Examples of ones that currently exists and provide addition functionality
  - Audio Cape
  - DVI Cape
  - Camera Cape
  - Weather Cape
- Expansion header allows easy breadboard access

Starting a Beaglebone Telemetry Project

- What to design?
- How to design it?
  - Fritzing – Open Source circuit design
    - More hobbyist + breadboard friendly
  - Eagle PCB – Not free but affordable for hobbyists
    - Larger learning curve + schematic first designing.
- Materials and hardware skills required
  - Soldering Iron + Minimal electrical engineering knowledge
- Software skills
  - Any language for reporting

Practical Applications

- Weather Reporting Station
- Radiation Monitoring
- Earthquake Detection Mesh Network
- Home Security System
- Entropy Pool Generation

What kind of data can we report?

- Barometric
- Temperature
- Radiation Exposure (Counts Per Minute)
- Earthquakes
- GPS + Orientation + Compass Heading
- Ambient Light

How can we share data?

- Cosm (Formerly Pachube)
  - Free for typical usage
- Allows reporting of almost any sensor possible via feeds
- Simple JSON or EEML interface
- Handles graphing all datasets and points
  - Allows settings triggers when thresholds are peaked (Twitter or HTTP POST)

Sample Reporting Snippet

```
def read_bmp085_pressure():
    f = open("/sys/bus/i2c/drivers/bmp085/1-0077/pressure0_input")
    return "%.2f" % (int(f.read().strip()) / 100.0)
….
def start_reporting(pac, w1_serial_id):
    while True:
            if w1_serial_id:
                pac.update([eeml.Data("w1-temp", read_w1_temp(w1_serial_id))])
            bmp085_pressure = read_bmp085_pressure()
            pac.update([eeml.Data("bmp085-temp", read_bmp085_temp())])
            pac.update([eeml.Data("bmp085-pressure", bmp085_pressure)])
            pac.update([eeml.Data("sht21-humidity", read_sht21_humidity())])
            pac.update([eeml.Data("sht21-temp", read_sht21_temp())])
            pac.update([eeml.Data("tsl2250-lux", read_tsl2550_lux_value())])
            pac.put()
            sleep(1)

if __name__ == "__main__":
    API_KEY = get_env_value("COSM_KEY")
    …
    API_URL = "/v2/feeds/%s.xml" % feed
    ...
        pac = eeml.Cosm(API_URL, API_KEY, use_https = False)
        start_reporting(pac, w1_serial_id))
```

Sample COSM Data

```
<eeml xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://
    www.eeml.org/xsd/0.5.1" xsi:schemaLocation="http://www.eeml.org/xsd/0.5.1 http://
    www.eeml.org/xsd/0.5.1/0.5.1.xsd" version="0.5.1">
     <environment>
          <data id="w1-temp"><current_value>25.00</current_value></data>
          <data id="bmp085-pressure"><current_value>1004.00</current_value></data>
          <data id="sht21-humidity"><current_value>39.74</current_value></data>
          <data id="tsl2250-lux"><current_value>0</current_value></data>
          <data id="sht21-temp"><current_value>2.45</current_value></data>
          <data id="bmp085-temp"><current_value>25.50</current_value></data>
     </environment>
</eeml>
```

Developing Telemetry Platform

- How are you going to place everything?
  - SMD, THT, etc
- Cost of sensors and how important accuracy and precision are
  - You get what you pay for…
- Sensors interfaces to use
  - I2C, SPI, Analog In, etc, etc
- Pins to use and what other capes you many want to use

DT Overlays/Not Capebus

- Consists of DT fragments that modify the live tree
  - Each fragment can target a DT node to add/remove/ override settings
- Geiger Cape for instance has fragments that setup various subsystems
  - PWM – Duty period and percent
  - GPIO + LEDS – Events and status
  - Analog In – Voltage reading across Geiger tube

Cape Manager

- Extends DT overlays to implement “firmware” for each cape design + version
  - Example -> firmware/capes/cape-bone-geiger-00A0.dts
- Allow override for prototype capes (those without an identification EEPROM)
- Doesn’t require rebuilding all of the device tree for changes in one cape
  - Parsing of version information that can toggle features and fix erratas

Cape Manager + DT Overlays

```
...
fragment@0 {
        target = <&am33xx_pinmux>;
        __overlay__ {
            bone_geiger_cape_led_pins: pinmux_bone_geiger_cape_led_pins {
                pinctrl-single,pins = <
                    0xe4 0x07       /* lcd_hsync.gpio2_23, OUTPUT | MODE7 */
                    0xec 0x07       /* lcd_ac_bias_en.gpio2_25, OUTPUT | MODE7 */
                >;
            };

            bone_geiger_cape_pins: pinmux_bone_geiger_cape_pins {
                pinctrl-single,pins = <
                    0x48 0x06           /* gpmc_a2.ehrpwm1a, OMAP_MUX_MODE6 | AM33XX_PIN_OUTPUT */
                    /* 0x19c 0x34 */  /* mcasp0_ahclkr.eCAP2_in_PWM2_out, OMAP_MUX_MODE4 | INPUT_PULLUP */
                    0x19c 0x37         /* mcasp0_ahclkr.gpio3_17, OMAP_MUX_MODE4 | INPUT_PULLUP */
                >;
            };
        };
    };
```

... Continued on next page ...

Cape Manager + DT Overlays Cont.

```
...
 bone-cape-geiger {
                        compatible = "bone-cape-geiger";
                        status = "okay";

                        pinctrl-names = "default";
                        pinctrl-0 = <&bone_geiger_cape_pins>;

                        pwms = <&ehrpwm1 0 500000 0>;
                        pwm-names = "bone-geiger-cape";

                        pwm-frequency  = <20000>;       /* 20KHz */
                        pwm-duty-cycle = <60>;          /* 60% - 500V LND712 */
                        event-blink-delay = <30>;       /* 30ms */

                        gpios = <&gpio4 17 0>;          /* pulse */

                        vsense-name = "AIN5";           /* analog vsense */
                        vsense-scale = <37325>;         /* scaling */
                };
```

... Continued on next page ...

Cape Manager (Working Example)

Confirm Cape Manger picked up device

 # cd /sys/devices/ocp.2/bone-cape-geiger.11

 # ls

counter modalias run uevent

driver power subsystem vsense

(start running)

 # echo 1 > run

(power LED turns on and the event LED lights up on a “click”)

Display counts:

 # cat counter

4344

Display VSENSE (voltage feedback loop) in Millivolts:

 # cat vsense

538004

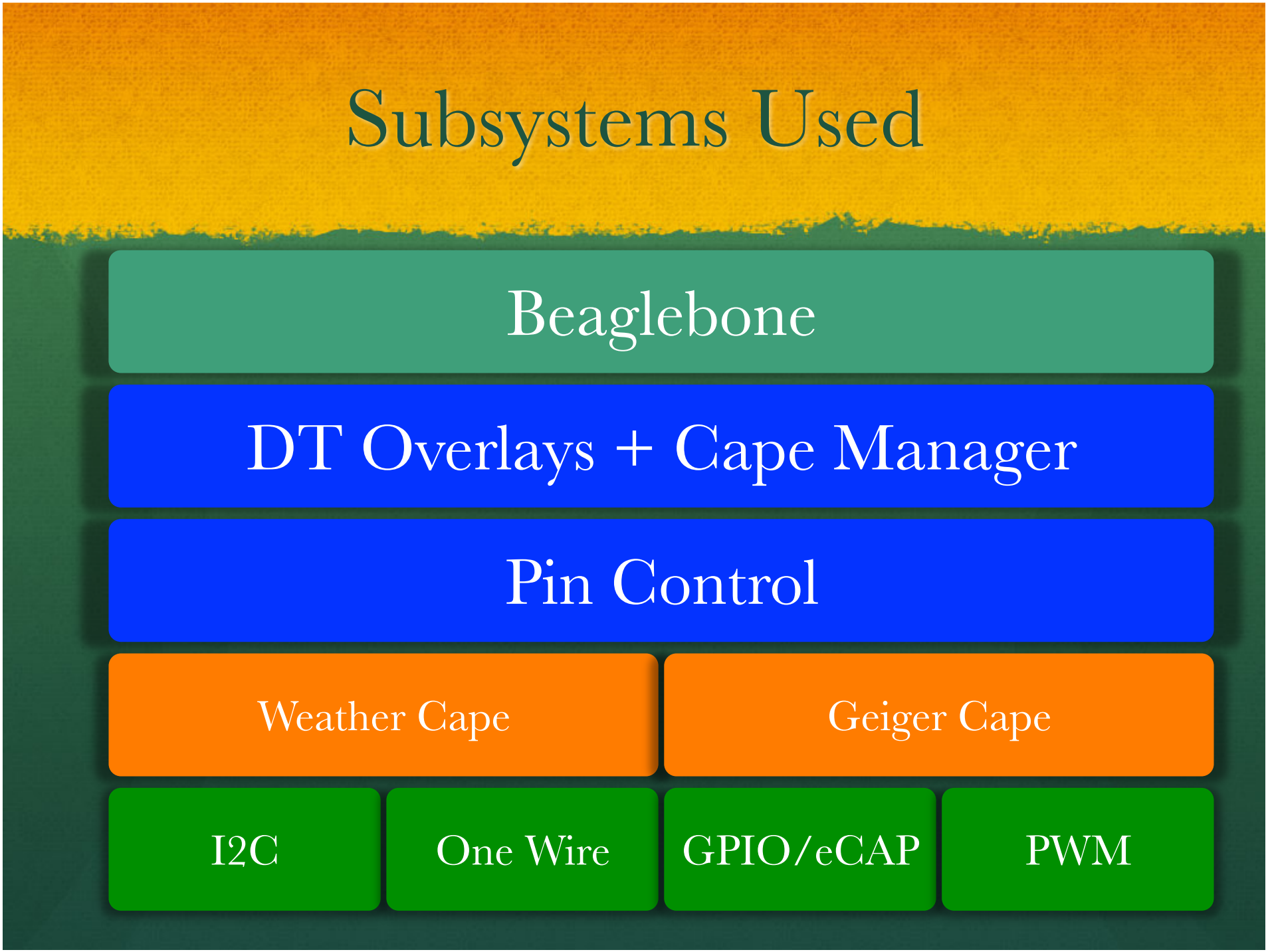
Geiger Counter

- Reason for picking this project
  - Simple + Fun
  - Practical purpose
  - Nature gives a perfect test source since atoms are always decaying
  - Excuse to test out various consumer and scientific items for radioactivity

Geiger Counter Continued

- Data points provided are simple “clicks” in time
- All the magic happens in how you report and display the data
- Remote stations need to take in account for power usage
  - CPUFreq ‘powersave’ governor *e.g. cpufreq-set –g powersave*
  - Adjust sample update rate
  - Offload any data processing upstream off the device.
  - Note that AM335x is not low power.. About 2.5 watts on the Beaglebone

Subsystems Used



Geiger Counter Reporting Flow

- GPIO Event - ~1 millisecond
- eCAP Event - ~36 microseconds

Geiger Cape Pinout

Table 11. Expansion Header P9 Pinout

SIGNAL NAME	PIN	CONN		PIN	SIGNAL NAME
	GND	1	2	GND	
	VDD_3V3EXP	3	4	VDD_3V3EXP	
	VDD_5V	5	6	VDD_5V	
	SYS_5V	7	8	SYS_5V	
PWR_BUT*		9	10	A10	SYS_RESETn
UART4_RXD	T17	11	12	U18	GPIO1_28
UART4_TXD	U17	13	14	U14	EHRPWM1A
GPIO1_16	R13	15	16	T14	EHRPWM1B
I2C1_SCL	A16	17	18	B16	I2C1_SDA
I2C2_SCL	D17	19	20	D18	I2C2_SDA
UART2_TXD	B17	21	22	A17	UART2_RXD
GPIO1_17	V14	23	24	D15	UART1_TXD
GPIO3_21	A14	25	26	D16	UART1_RXD
GPIO3_19	C13	27	28	C12	SPI1_CS0
SPI1_D0	B13	29	30	D12	SPI1_D1
SPI1_SCLK	A13	31	32	VDD_ADC(1.8V)	
AIN4	C8	33	34	GNDA_ADC	
AIN6	A5	35	36	A5	AIN5
AIN2	B7	37	38	A7	AIN3
AIN0	B6	39	40	C7	AIN1
CLKOUT2	D14	41	42	C18	GPIO0_7
	GND	43	44	GND	
	GND	45	46	GND	

- Green – Ground
- Red – Power Supplies
- Blue – PWM
- Yellow – GPIO/eCAP event
- Purple- I2C
- Black – Analog In

Sensor Selection (Geiger Counter)

- Geiger tubes have various features of quality
  - Dead time
  - Sensitivity
  - Type of radiation it can detect
    - LND-712 was selected since it can detect all three main types of radiation (alpha + gamma + beta). Most expensive tube of it's type.
    - SBT-9 (Soviet-Era tube) was a close second but not as sensitive as the LND or available in North America
    - SBM-20 (Soviet surplus) is an good low end choice (gamma + beta only)
- Voltage needed to register counts vary greatly
  - LND-712 – 500 volts
  - SBT-9 and SBM-20 (and most Soviet tubes) – 300 to 400 volts

Geiger Counter Circuit

- Tube is a large capacitor (in size only, and is only ~3 picofarads) that gets charged up, and when hit by ionizing radiation causes an “overcharge”. Excess charge has to go somewhere and that causes a “click”
- Logic level and pulse shifter design
  - GPIO version extends pulse to ~1 millisecond
  - eCAP version only level shifts

Safety (Geiger Counter)

- High voltage is sourced to the tube. Current boost converter can provide about 600V
  - Although almost no current is flowing. It still can be dangerous…
  - Analog-in pin used for feedback loop to measure voltage
  - Keep HV traces short as possible
- MOSFET can be asserted ON at 100% which is very bad
  - Never trust expansion header pin states…
  - Don't trust software…
- Various ways of solving this in hardware
  - I2C GPIO expander to toggle states
  - Op-amp + low-pass filter to trigger board reset (SYS_RESETN pin to GND)

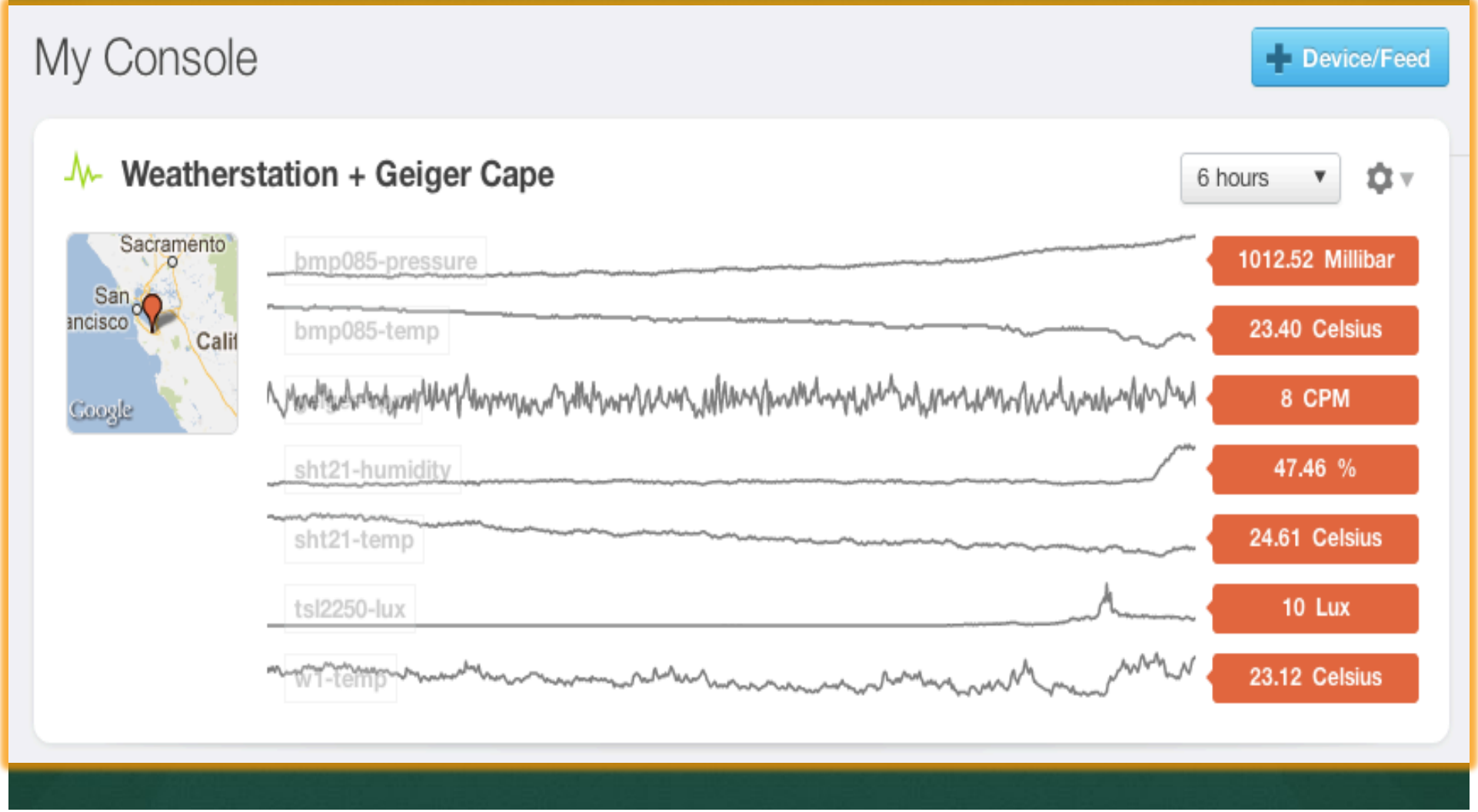
ELC Demo Setup (Geiger Counter)

- Geiger Cape (Prototype Rev 8) + LND-712
- 1.8” TFT Adafruit SPI interface display
- Beaglebone Rev A6

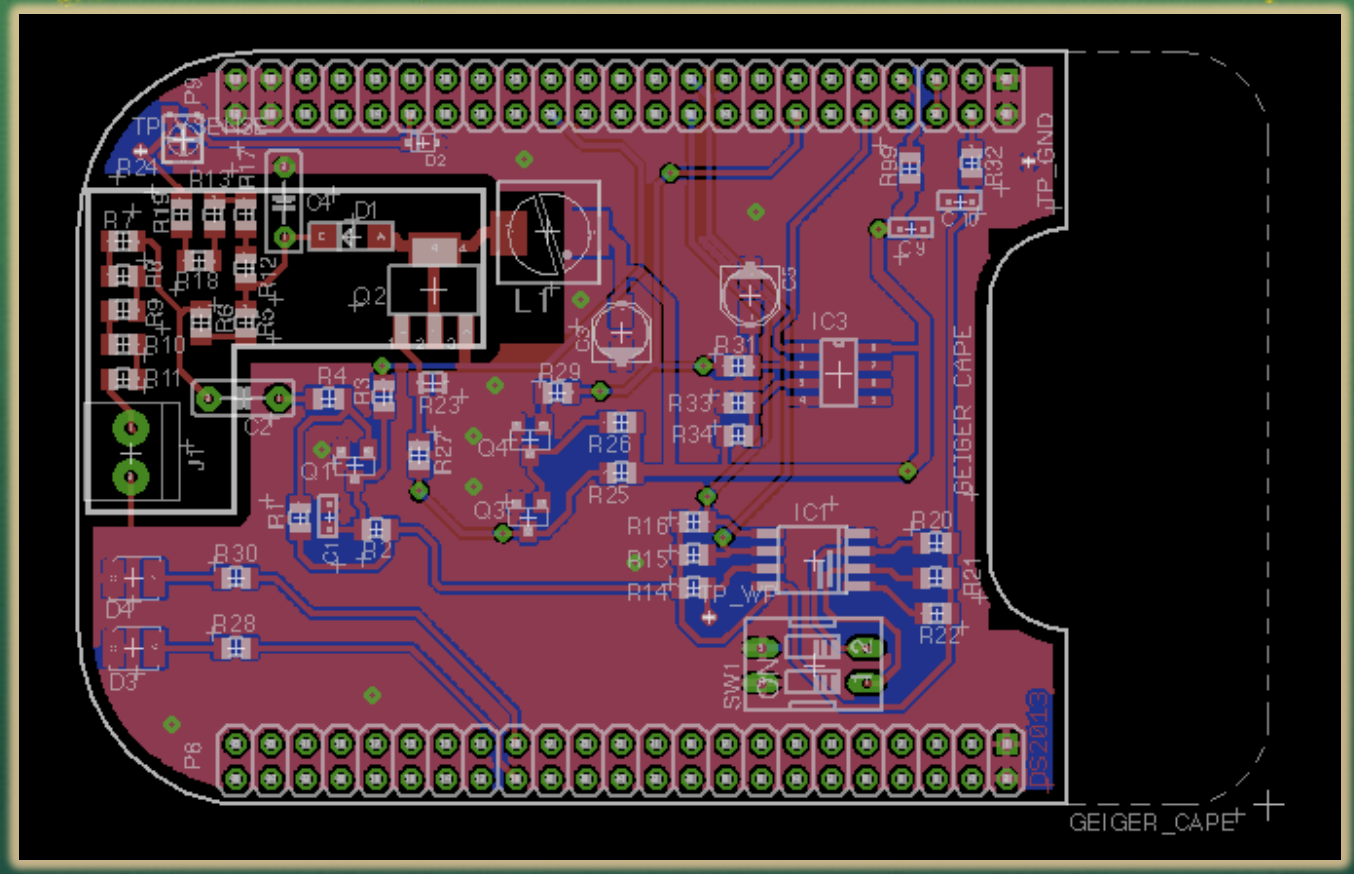
Weather + Radiation Station

- Geiger Cape (Prototype Rev 8) + SBT-9
- CircuitCo Weather Cape
- Beaglebone Rev A6

Weather + Radiation Station Continued



PCB Design



- Geiger Cape Prototype Rev 10

Lessons Learned

- Know what functionality your device has and use it
- Avoid bitbanging interfaces that already available
- Microcontroller Unit != Microprocessor Unit treat it as such
- Watch initial states
- Example of a bad states would be a PWM pin that is shared with a GPIO that gets asserted on…
- Watch GPIO states on pins…
- Cape Manager + DT Overlays will not save you every-time…

Lessons Learned Continued

- Watch the logic voltage level
- Watch the pin muxing…
- Really easy to conflict with another cape.
  - DT Overlays + Cape Manager holds the hope for the future
  - Possible damage if you aren't careful
- Test on breadboard first
- Analog In (ADC) safety. Absolute 1.8v LIMIT!
  - Invest in some 1.8V Zener diodes, especially if you use sensors that may output higher voltage
  - For THT applications two standard rectifier diodes will give you a equivalent 1.4V Zener diode
- Show Demo! No magic smoke, no whammy!

Questions?

References

- Beaglebone SRM
  - http:/beagleboard.org/static/beaglebone/latest/Docs/Hardware/BONE_SRM.pdf
- Cadsoft Eagle PCB
  - http://www.cadsoftusa.com
- Adafruit Eagle Library (Beaglebone Cape Part)
  - https://github.com/adafruit/Adafruit-Eagle-Library
- Fritzing
  - http://fritzing.org
- Fritzling Parts (Beaglebone Cape Part)
  - https://github.com/ohporter/fritzing-parts

References Continued

- Cupertino (Indoors) Radiation + Weather
  - https://cosm.com/feeds/73056
- Slides + Demo Source Code
  - https://github.com/mranostay/beaglebone-telemetry-presentation
  - https://github.com/mranostay/cosm-analog
- Thanks to other Geiger Cape team members
  - Dimitris Sapountzakis (Hardware)
  - Pantelis Antoniou (Software + DT Overlay patchset)
  - Koen Kooi (Design Advice)

References Continued

- DT Overlays + Capebus patchset branch
  - https://github.com/koenkooi/linux/tree/3.8-for-panto-rebase
- LND-712 End Window Alpha + Beta + Gamma detector datasheet
  - http://www.lndinc.com/products/pdf/711/
- SBT-9 Soviet-era Alpha + Beta + Gamma tube
  - http://gstube.com/data/3004/ (Russian)