The E-ALE (Embedded Apprentice Linux Engineer) is a series of seminars held at existing conferences covering topics which are fundamental to a Linux professional in the field of Embedded Linux.

This seminar will spend equal time on lecture and hands on labs at the end of each seminar which allow you to practice the material you’ve learned.

This material makes the assumption that you have minimal experience with using Linux in general, and a basic understanding of general industry terms. The assumption is also made that you have access to your own computers upon which to practice this material.

More information can be found at https://e-ale.org/

This material is licensed under CC-BY SA4
# Contents

1 Preliminaries ......................................................... 1
   1.1 Introductions .................................................. 2
   1.2 Getting Started ................................................ 4

2 Spelunking Baconbits .............................................. 6
   2.1 BaconBits Hardware .......................................... 7
   2.2 PocketBeagle Hardware ..................................... 11
   2.3 Summary ......................................................... 12
   2.4 Device Tree .................................................... 13

3 Spelunking TechLab .................................................. 21
   3.1 TechLab Hardware ............................................. 22
   3.2 PocketBeagle Hardware .................................... 26
   3.3 MMA8453 ....................................................... 27
   3.4 Summary ........................................................ 28
   3.5 Device Tree .................................................... 29
## Contents

4 Labs

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>Preparation</td>
<td>42</td>
</tr>
<tr>
<td>4.2</td>
<td>Lab 1</td>
<td>43</td>
</tr>
<tr>
<td>4.3</td>
<td>Lab 2</td>
<td>44</td>
</tr>
<tr>
<td>4.4</td>
<td>Lab 3</td>
<td>45</td>
</tr>
</tbody>
</table>
Chapter 1

Preliminaries
1.1 Introductions

About Me

- CTO at Konsulko Group
- Using Linux since 1992
- Professional embedded Linux engineer since 1998
- Previously maintained kernel support for embedded PPC platforms, RapidIO subsystem, and Broadcom Mobile SoCs
- Various contributions around the kernel
About Konsulko Group

- Konsulko Group is a services company founded by embedded Linux veterans
- Community and commercial embedded, Linux, and Open Source Software development
- Linux Foundation training partners
- See https://www.konsulko.com for more information
1.2 Getting Started

- Download the slides for local reference
What Are We Going To Do?

Using real world examples of a paddle-style joystick on the BaconBits cape and an accelerometer-based joystick on the TechLab cape we will learn the following:

- How to read schematics
- How to follow schematic entities to datasheets
- How to convert datasheet info to Device Tree or driver data
- Practice our new skills with some lab exercises!
Chapter 2

Spelunking Baconbits
2.1 BaconBits Hardware

Component Placement

Figure 2.1: BaconBits Component Identification

- RV1 is the thumbwheel device
2.1. BACONBITS HARDWARE

BaconBits Schematic Overview


Figure 2.2: BaconBits Schematic
2.1. BACONBITS HARDWARE

BaconBits Thumbwheel

Figure 2.3: BaconBits Thumbwheel

- Signals:
  - ADC_GND
  - ADC_PWR
  - ANALOG_IN
2.1. BACONBITS HARDWARE

BaconBits P1 Connector

Figure 2.4: BaconBits P1 Connector

- Pins:
  - ADC_GND : P1-17
  - ADC_PWR : P1-18
  - ANALOG_IN : P1-19
2.2 PocketBeagle Hardware

PocketBeagle Pinout

Figure 2.5: PocketBeagle Expansion Header
2.3 Summary

Hardware Investigation Results

- Thumbwheel:
  - Connected to analog input 0 (\texttt{AIN0})

- User Button:
  - Connected to \texttt{GPMC\_AD13} which can be muxed as \texttt{GPIO1\_13}
  - Active low
2.4 Device Tree

What is Needed?

- Mux the `GPMC_AD13` pin as `GPIO1_13`
- Create a paddle device with a compatible string
- Link to the GPIO pinmux node
- Link to ADC channel 0 for the thumbwheel
- Link to `GPIO1_13` for the button
Note that **GPIO1.13** is at offset **0x834**


<table>
<thead>
<tr>
<th>Address</th>
<th>Offset</th>
<th>Description</th>
<th>Section</th>
</tr>
</thead>
<tbody>
<tr>
<td>800h</td>
<td>conf_gPMC_ad0</td>
<td>See the device datasheet for information on default pin mux configurations. Note that the device ROM may change the default pin mux for certain pins based on the SYSBOOT mode settings.</td>
<td>9.3.1.50</td>
</tr>
<tr>
<td>804h</td>
<td>conf_gPMC_ad1</td>
<td></td>
<td>9.3.1.50</td>
</tr>
<tr>
<td>808h</td>
<td>conf_gPMC_ad2</td>
<td></td>
<td>9.3.1.50</td>
</tr>
<tr>
<td>80Ch</td>
<td>conf_gPMC_ad3</td>
<td></td>
<td>9.3.1.50</td>
</tr>
<tr>
<td>810h</td>
<td>conf_gPMC_ad4</td>
<td></td>
<td>9.3.1.50</td>
</tr>
<tr>
<td>814h</td>
<td>conf_gPMC_ad5</td>
<td></td>
<td>9.3.1.50</td>
</tr>
<tr>
<td>818h</td>
<td>conf_gPMC_ad6</td>
<td></td>
<td>9.3.1.50</td>
</tr>
<tr>
<td>81Ch</td>
<td>conf_gPMC_ad7</td>
<td></td>
<td>9.3.1.50</td>
</tr>
<tr>
<td>820h</td>
<td>conf_gPMC_ad8</td>
<td></td>
<td>9.3.1.50</td>
</tr>
<tr>
<td>824h</td>
<td>conf_gPMC_ad9</td>
<td></td>
<td>9.3.1.50</td>
</tr>
<tr>
<td>828h</td>
<td>conf_gPMC_ad10</td>
<td></td>
<td>9.3.1.50</td>
</tr>
<tr>
<td>82Ch</td>
<td>conf_gPMC_ad11</td>
<td></td>
<td>9.3.1.50</td>
</tr>
<tr>
<td>830h</td>
<td>conf_gPMC_ad12</td>
<td></td>
<td>9.3.1.50</td>
</tr>
<tr>
<td>834h</td>
<td>conf_gPMC_ad13</td>
<td></td>
<td>9.3.1.50</td>
</tr>
<tr>
<td>838h</td>
<td>conf_gPMC_ad14</td>
<td></td>
<td>9.3.1.50</td>
</tr>
<tr>
<td>83Ch</td>
<td>conf_gPMC_ad15</td>
<td></td>
<td>9.3.1.50</td>
</tr>
<tr>
<td>840h</td>
<td>conf_gPMC_ad0</td>
<td></td>
<td>9.3.1.50</td>
</tr>
</tbody>
</table>

Figure 2.6: **AM335x Pin Mux Registers**
IIO provider binding

Documentation/devicetree/bindings/iio/iio-bindings.txt:

==IIO providers==

Required properties:
#io-channel-cells: Number of cells in an IIO specifier; Typically 0 for nodes with a single IIO output and 1 for nodes with multiple IIO outputs.

Example for a simple configuration with no trigger:

    adc: voltage-sensor@35 {
        compatible = "maxim,max1139";
        reg = <0x35>;
        #io-channel-cells = <1>;
    };

    .
    .
    .
IIO consumer binding

Documentation/devicetree/bindings/iio/iio-bindings.txt:

==IIO consumers==

Required properties:
io-channels: List of phandle and IIO specifier pairs, one pair for each IIO input to the device. Note: if the IIO provider specifies '0' for #io-channel-cells, then only the phandle portion of the pair will appear.

Optional properties:
io-channel-names:
List of IIO input name strings sorted in the same order as the io-channels property. Consumers drivers will use io-channel-names to match IIO input names with IIO specifiers.

For example:

device {
  io-channels = <&adc 1>, <&ref 0>;
  io-channel-names = "vcc", "vdd";
};
TI TSC ADC binding

Documentation/devicetree/bindings/input/touchscreen/ti-tsc-adc.txt:

- child "adc"
  compatible: Should be
  "ti,am3359-adc" for AM335x/AM437x SoCs
  "ti,am654-adc", "ti,am3359-adc" for AM654 SoCs
  ti,adc-channels: List of analog inputs available for ADC.
    AIN0 = 0, AIN1 = 1 and so on till AIN7 = 7.
Pinctrl client binding

Documentation/devicetree/bindings/pinctrl/pinctrl-bindings.txt:

Required properties:
pinctrl-0: List of phandles, each pointing at a pin configuration node. These referenced pin configuration nodes must be child nodes of the pin controller that they configure.

Optional properties:
pinctrl-1: List of phandles, each pointing at a pin configuration node within a pin controller.

For example:

```c
/* For a client device requiring named states */
device {
    pinctrl-names = "active", "idle";
    pinctrl-0 = <&state_0_node_a>;
    pinctrl-1 = <&state_1_node_a &state_1_node_b>;
}
```
2.4. DEVICE TREE

GPIO consumer binding

Documentation/devicetree/bindings/gpio/gpio.txt:

- GPIO properties should be named "[<name>]-gpios", with <name> being the purpose of this GPIO for the device.

Example of a node using GPIOs:

```
node {
    enable-gpios = <&qe_pio_e 18 GPIO_ACTIVE_HIGH>;
}
```

GPIO_ACTIVE_HIGH is 0, so in this example gpio-specifier is "18 0" and encodes GPIO pin number, and GPIO flags as accepted by the "qe_pio_e" gpio-controller.
DT changes shown against mainline kernel am335x_pocketbeagle.dts

- User button GPIO pinmux configuration:

```c
  gpio1_13_pin: pinmux-gpio1-13-pin {
    pinctrl-single,pins = <
      AM33XX_IOPAD(0x0834, PIN_INPUT | MUX_MODE7)
    >;
  };
```

- Paddle device node:

```c
  paddle {
    compatible = "e-ale,baconbits-paddle";
    pinctrl-0 = &gpio1_13_pin;
    io-channels = &am335x_adc 0;
    io-channel-names = "thumbwheel";
    button-gpios = &gpio1 13 GPIO_ACTIVE_LOW;
  };
```
Chapter 3

Spelunking TechLab
3.1 TechLab Hardware

- The TechLab Cape has an accelerometer and a couple buttons that can be used as the basis for a joystick device.

Figure 3.1: TechLab Cape
TechLab Schematic Overview


Figure 3.2: TechLab Schematic
TechLab Accelerometer

- **Signals:**
  - P1.28(I2C2.SCL)
  - P1.26(I2C2.SDA)
- **I2C address 0x1C**

Figure 3.3: TechLab Accelerometer
3.1. TECHLAB HARDWARE

TechLab Buttons

Figure 3.4: TechLab Buttons

- Signals:
  - P1.29/PRU0.7(R BTN)
  - P2.33/GPIO45(L BTN)
3.2 PocketBeagle Hardware

PocketBeagle Pinout

Figure 3.5: PocketBeagle Expansion Header
3.3 MMA8453

MMA8453 Resolution


5.2 8-bit or 10-bit data

The measured acceleration data is stored in the OUT_X_MSB, OUT_X_LSB, OUT_Y_MSB, OUT_Y_LSB, OUT_Z_MSB, and OUT_Z_LSB registers as 2's complement 10-bit numbers. The most significant 8-bits of each axis are stored in OUT_X (Y, Z)_MSB, so applications needing only 8-bit results can use these three registers and ignore OUT_X,Y,Z_LSB. To do this, the F_READ bit in CTRL_REG1 must be set. When the F_READ bit is cleared, the fast read mode is disabled.

When the full-scale is set to 2 g, the measurement range is −2 g to +1.9961 g, and each count corresponds to 1 g/256 (3.9 mg) at 10-bits resolution. When the full-scale is set to 8 g, the measurement range is 8 g to +7.9844 g, and each count corresponds to 1 g/64 (15.6 mg) at 10-bits resolution. The resolution is reduced by a factor of 4 if only the 8-bit results are used. For more information on the data manipulation between data formats and modes, refer to NXP application note AN4076. There is a device driver available that can be used with the Sensor Toolbox demo board (LFSTBEB8451, 2, 3Q).

Figure 3.6: MMA8453 Resolution

- 10-bit samples (verified by inspection of mma8452.c kernel driver)
- In full resolution 2g mode, this means a range of -255 to 256
3.4 Summary

Hardware Investigation Results

- Accelerometer:
  - I2C SCL (P1.28(I2C2.SCL))
  - I2C SDA (P1.26(I2C2.SDA))

- Buttons:
  - Left (GPIO pull-up) (P2.33/GPIO45(L.BTN))
  - Right (GPIO pull-up) (P1.29/PRU0.7(R.BTN))
IIO provider binding

Documentation/devicetree/bindings/iio/iio-bindings.txt:

```plaintext
==IIO providers==

Required properties:
#io-channel-cells: Number of cells in an IIO specifier; Typically 0 for nodes with a single IIO output and 1 for nodes with multiple IIO outputs.

Example for a simple configuration with no trigger:

```plaintext
adc: voltage-sensor@35 {
    compatible = "maxim,max1139";
    reg = <0x35>;
    #io-channel-cells = <1>;
};
```

...
3.5. DEVICE TREE

IIO consumer binding

Documentation/devicetree/bindings/iio/iio-bindings.txt:

==IIO consumers==

Required properties:
io-channels: List of phandle and IIO specifier pairs, one pair for each IIO input to the device. Note: if the IIO provider specifies ’0’ for #io-channel-cells, then only the phandle portion of the pair will appear.

Optional properties:
io-channel-names: List of IIO input name strings sorted in the same order as the io-channels property. Consumers drivers will use io-channel-names to match IIO input names with IIO specifiers.

For example:

```c
device {
    io-channels = <&adc 1>, <&ref 0>;
    io-channel-names = "vcc", "vdd";
};
```
3.5. DEVICE TREE

MMA8453 binding

Documentation/devicetree/bindings/iio/accel/mma8452.txt:

Freescale MMA8451Q, MMA8452Q, MMA8453Q, MMA8652FC, MMA8653FC or FXLS8471Q triaxial accelerometer

Required properties:

- compatible: should contain one of
  * "fsl,mma8451"
  * "fsl,mma8452"
  * "fsl,mma8453"
  * "fsl,mma8652"
  * "fsl,mma8653"
  * "fsl,fxls8471"

- reg: the I2C address of the chip

Optional properties:

- interrupts: interrupt mapping for GPIO IRQ

- interrupt-names: should contain "INT1" and/or "INT2", the accelerometer’s interrupt line in use.

Example:

    mma8453fc01d {
        compatible = "fsl,mma8453";
        reg = <0x1d>;
        interrupt-parent = <&gpio1>;
        interrupts = <5 0>;
        interrupt-names = "INT2";
Pinctl client binding

Documentation/devicetree/bindings/pinctrl/pinctrl-bindings.txt:

Required properties:
pinctrl-0: List of phandles, each pointing at a pin configuration node. These referenced pin configuration nodes must be child nodes of the pin controller that they configure.

Optional properties:
pinctrl-1: List of phandles, each pointing at a pin configuration node within a pin controller.

For example:

    /* For a client device requiring named states */
    device {
        pinctrl-names = "active", "idle";
        pinctrl-0 = <&state_0_node_a>;
        pinctrl-1 = <&state_1_node_a &state_1_node_b>;
    };

© CC-BY SA4
3.5. DEVICE TREE

GPIO consumer binding

Documentation/devicetree/bindings/gpio/gpio.txt:

- 
- 
- 

GPIO properties should be named "[<name>-]gpios", with <name> being the purpose of this GPIO for the device.

- 
- 
- 

Example of a node using GPIOs:

```plaintext
node {
    enable-gpios = <&qe_pio_e 18 GPIO_ACTIVE_HIGH>;
}
```

GPIO_ACTIVE_HIGH is 0, so in this example gpio-specifier is "18 0" and encodes GPIO pin number, and GPIO flags as accepted by the "qe_pio_e" gpio-controller.

- 
- 
- 
/* P1_29 (ZCZ ball A14) pruin */
P1_29_pinmux {
    compatible = "bone-pinmux-helper";
    status = "okay";
    pinctrl-names = "default", "gpio", "gpio_pu", "gpio_pd",
                   "gpio_input", "qep", "pruout", "pruin";
    pinctrl-0 = <&P1_29_default_pin>;
    pinctrl-1 = <&P1_29_gpio_pin>;
    pinctrl-2 = <&P1_29_gpio_pu_pin>;
    pinctrl-3 = <&P1_29_gpio_pd_pin>;
    pinctrl-4 = <&P1_29_gpio_input_pin>;
    pinctrl-5 = <&P1_29_qep_pin>;
    pinctrl-6 = <&P1_29_pruout_pin>;
    pinctrl-7 = <&P1_29_pruin_pin>;
};
/* P2_33 (ZCZ ball R12) */
P2_33_pinmux {
    compatible = "bone-pinmux-helper";
    status = "okay";
    pinctrl-names = "default", "gpio", "gpio Pu", "gpio PD", "gpio input", "qep", "pruout";
    pinctrl-0 = <&P2_33_default_pin>;
    pinctrl-1 = <&P2_33_gpio_pin>;
    pinctrl-2 = <&P2_33_gpioPu_pin>;
    pinctrl-3 = <&P2_33_gpioPD_pin>;
    pinctrl-4 = <&P2_33_gpioInput_pin>;
    pinctrl-5 = <&P2_33_qep_pin>;
    pinctrl-6 = <&P2_33_pruout_pin>;
};
### P1.29 GPIO resource


<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A14</td>
<td>MCASP0_AHCLKX</td>
<td>mcasp0_ahclkx</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>eQEP0_strobe</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>mcasp0_axr3</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>mcasp1_axr1</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EMU4</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>pr1_pru0_pru_r30_7</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>pr1_pru0_pru_r31_7</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>gpio3_21</td>
<td>7</td>
</tr>
</tbody>
</table>

Figure 3.7: P1.29 ZCZ A14 (gpio3_21)
P2.33 GPIO resource


<table>
<thead>
<tr>
<th>R12</th>
<th>GPMC_AD13</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>gpmc_ad13</td>
<td>0</td>
</tr>
<tr>
<td>lcd_data18</td>
<td>1</td>
</tr>
<tr>
<td>mmc1_dat5</td>
<td>2</td>
</tr>
<tr>
<td>mmc2_dat1</td>
<td>3</td>
</tr>
<tr>
<td>eQEP2B_in</td>
<td>4</td>
</tr>
<tr>
<td>pr1_mii0_txd1</td>
<td>5</td>
</tr>
<tr>
<td>pr1_pr0_pru_r30_15</td>
<td>6</td>
</tr>
<tr>
<td>gpio1_13</td>
<td>7</td>
</tr>
</tbody>
</table>

Figure 3.8: **P2.33 ZCZ R12 (gpio1.13)**
Describing the Joystick Hardware

- Specify the MMA8453 accelerometer
- Specify the Left and Right buttons used by joystick
- Specify the MMA8453 channels used by joystick
- Specify pinmux settings needed for buttons
- Specify GPIO resources needed for buttons
Implementation

Shown as a modification of the beagleboard.org Linux kernel am335x_pocketbeagle.dts

```dts
&i2c2 {
    #address-cells = <1>;
    #size-cells = <0>;
    mma8453: mma8453@1c {
        #io-channel-cells = <1>;
        status = "okay";
        compatible = "fsl,mma8453";
        reg = <0x1c>;
    }
};
};
```
Implementation (continued)

```c
P2_33_pinmux { status = "disabled"; }; /* Left - gpio3_21 */
P1_29_pinmux { status = "disabled"; }; /* Right - gpio1_13 */
cape-universal { status = "disabled"; };

joystick {
    compatible = "bborg,techjoy";
    pinctrl-0 = <&P2_33_gpio_input_pin>,
                <&P1_29_gpio_input_pin>;
    io-channels = <&mma8453 0>, <&mma8453 1>;
    io-channel-names = "accel_x", "accel_y";
    button-gpios = <&gpio3 21 GPIO_ACTIVE_LOW>,
                   <&gpio1 13 GPIO_ACTIVE_LOW>;

};
```
Chapter 4

Labs
4.1 Preparation

Documentation Required

- https://www.ti.com/lit/ug/spruh73q/spruh73q.pdf
- Figure 3.5: Pocketbeagle Expansion Header
Do the following:

- Find the Light Sensor on the TechLab
- Document hardware signal(s) it uses
- Document the DT property and value that expresses the ADC channel used by this sensor
4.3 Lab 2

TechLab Buzzer

Do the following:

- Find the Buzzer on the TechLab
- Document the hardware signal(s) it uses
- Document the DT property and value that expresses the GPIO used by the buzzer
4.4 Lab 3

TechLab Multi-colored LED

Do the following:

- Find the Multi-colored LED on the TechLab
- Document the hardware signal(s) it uses for RGB.
- Document the DT property and value that expresses the PWMs used by the LED

Hint: see Documentation/devicetree/bindings/pwm/pwm.txt (or .yaml) and pwm-tiehrpwm.txt for PWM binding properties. Also consult the AM335x TRM for hardware details on the Enhanced PWM peripheral.