iioinput-user

Introduction to Userspace IIO

Matt Porter <mporter@konsulko.com>

e-ale

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**/

Contents

Chapter 1

Preliminaries

1.1 Introductions

About Me

- CTO at Konsulko Group
- Using Linux since 1992
- Professional embedded Linux engineer since 1998
- Previously maintained embedded PPC platforms, RapidIO subsystem, and Broadcom Mobile SoCs in the kernel
- Various small 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
- See `https://www.konsulko.com` for more information
- Linux Foundation training partners
  - Use code **Konsulko.10.ATP.kr** for a 10% discount on any Linux Foundation training course.

1.2 Project Plan

Slides

- Download the slides for local reference
- https://cm.e-ale.org/2019/SCaLE17x/iio/iioinput-user-SLIDES.pdf

What To Do?

- Check out that cool thumbwheel on the BaconBits MiniCape.
- Let's do something with it!
- We'll learn how to read the position of the thumbwheel.
- With that data, we have the foundation for a userspace single axis joystick driver.

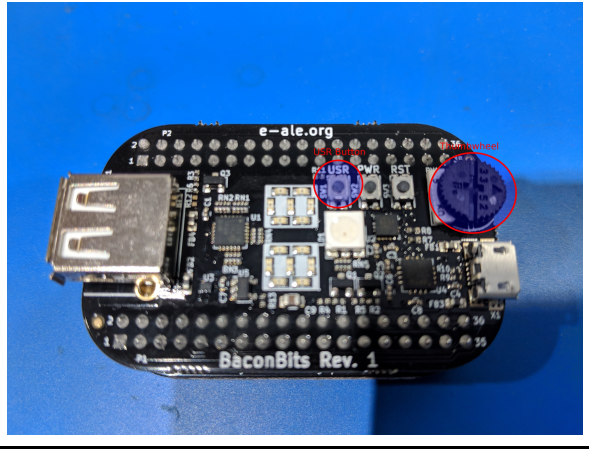

Figure 1.1: **BaconBits MiniCape**

Exact Steps

- Understand how the thumbwheel is interfaced in hardware
- Understand IIO and its purpose in the universe
- Learn how IIO exposes the thumbwheel's ADC to userspace
- Learn how to use libiio for userspace IIO interaction
- Write a thumbwheel-based userspace joystick driver and test it

Chapter 2

Hardware

2.1 BaconBits Hardware

Component Placement

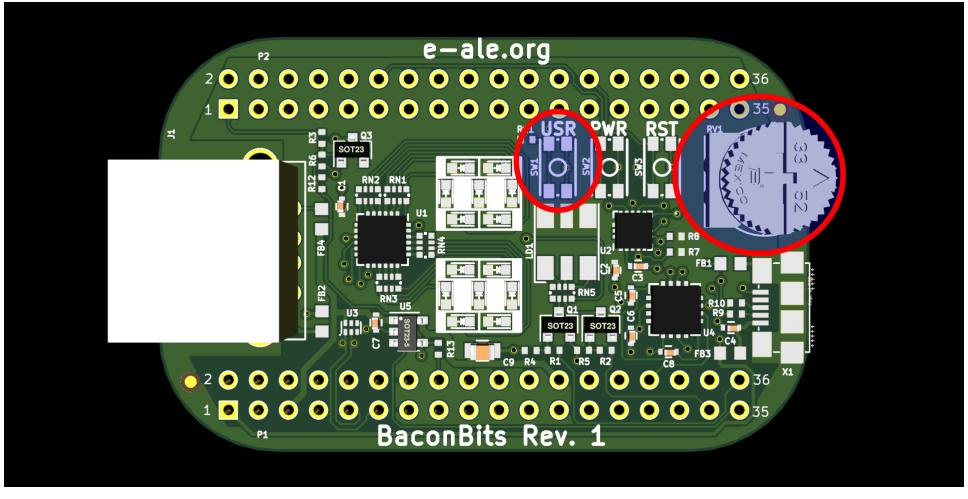

Figure 2.1: **BaconBits Component Identification**

- **RV1** is the thumbwheel device

BaconBits Schematic Overview

- https://github.com/e-ale/BaconBitsCapeHW/blob/master/baconbits.pdf

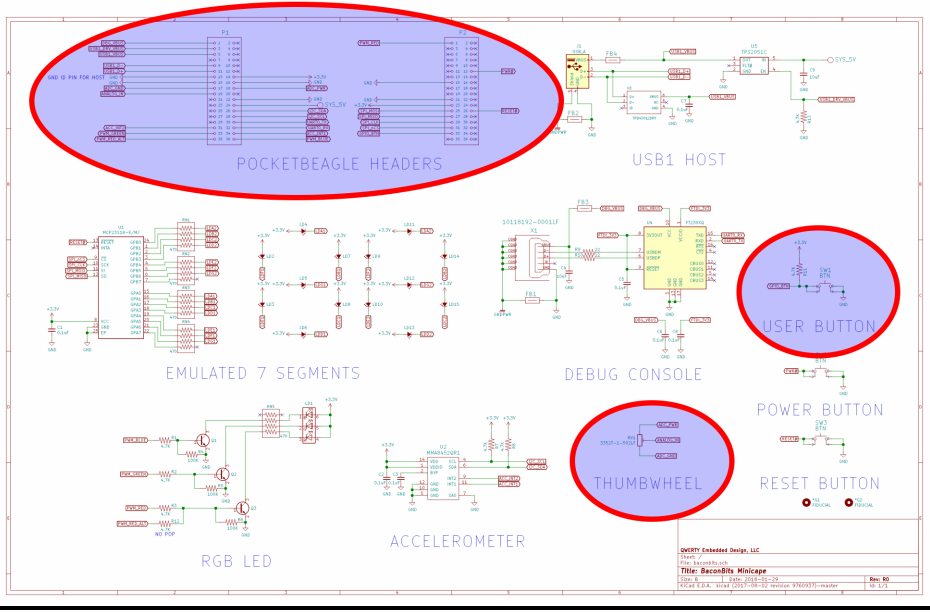

Figure 2.2: **BaconBits Schematic**

BaconBits Thumbwheel

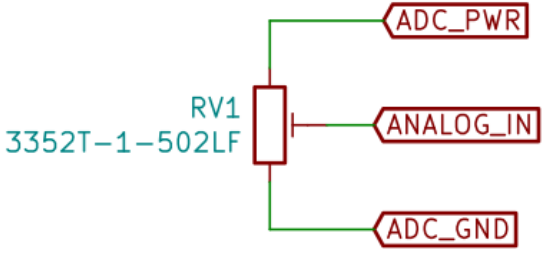

Figure 2.3: **BaconBits Thumbwheel**

- Signals:
  - **ADC_GND**
  - **ADC_PWR**
  - **ANALOG_IN**

BaconBits P1 Connector

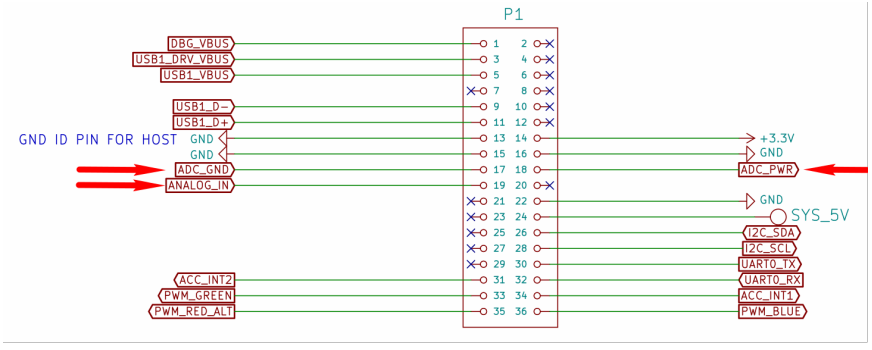

Figure 2.4: **BaconBits P1 Connector**

- Pins:
  - **ADC_GND : P1-17**
  - **ADC_PWR : P1-18**
  - **ANALOG_IN : P1-19**

2.2 PocketBeagle Hardware

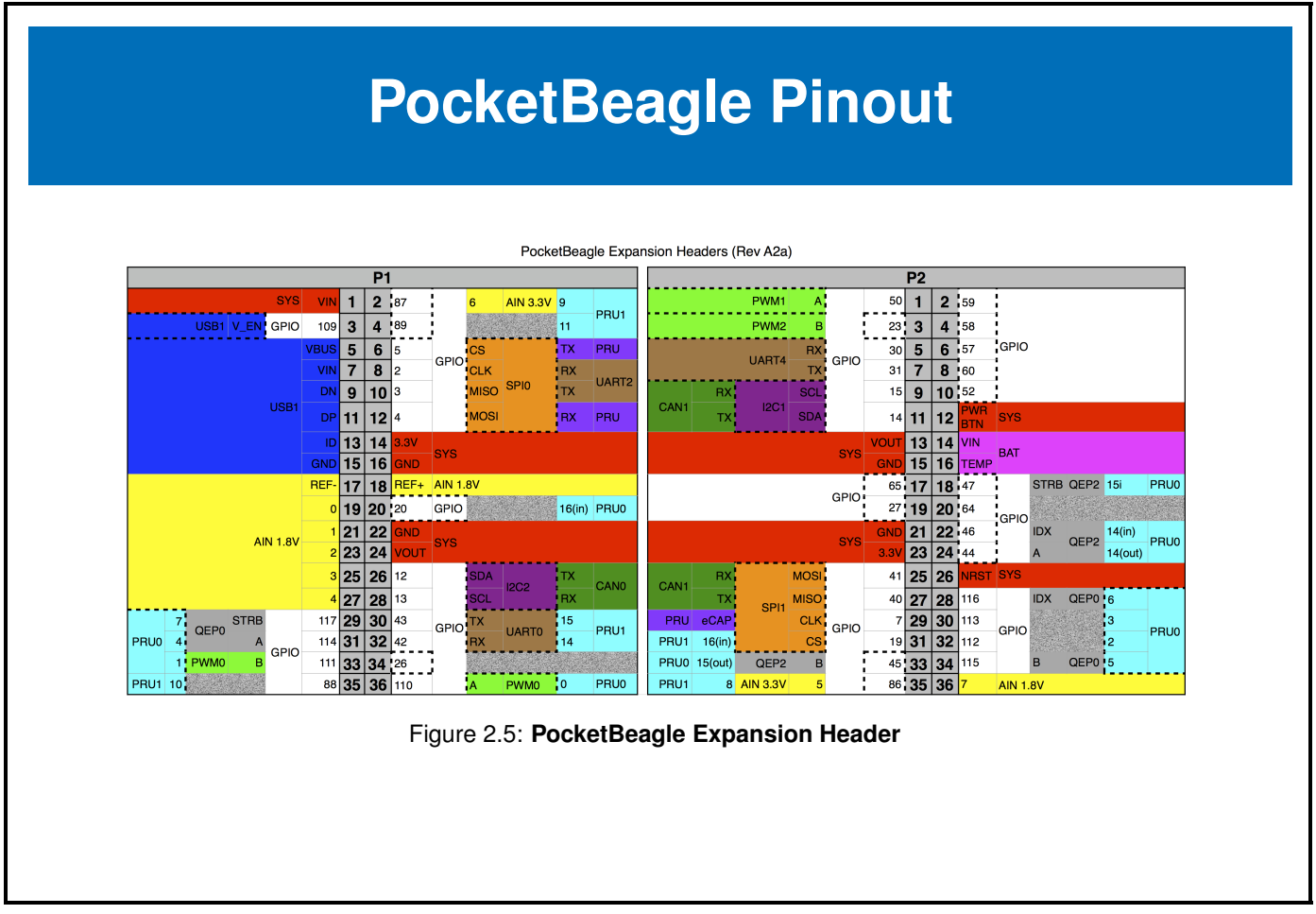

Figure 2.5: **PocketBeagle Expansion Header**

PocketBeagle Schematic Headers

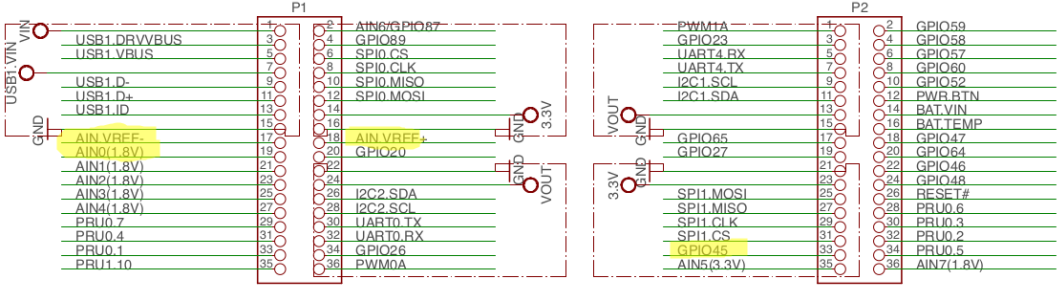

Figure 2.6: **PocketBeagle Schematic Headers**

PocketBeagle Schematic Analog

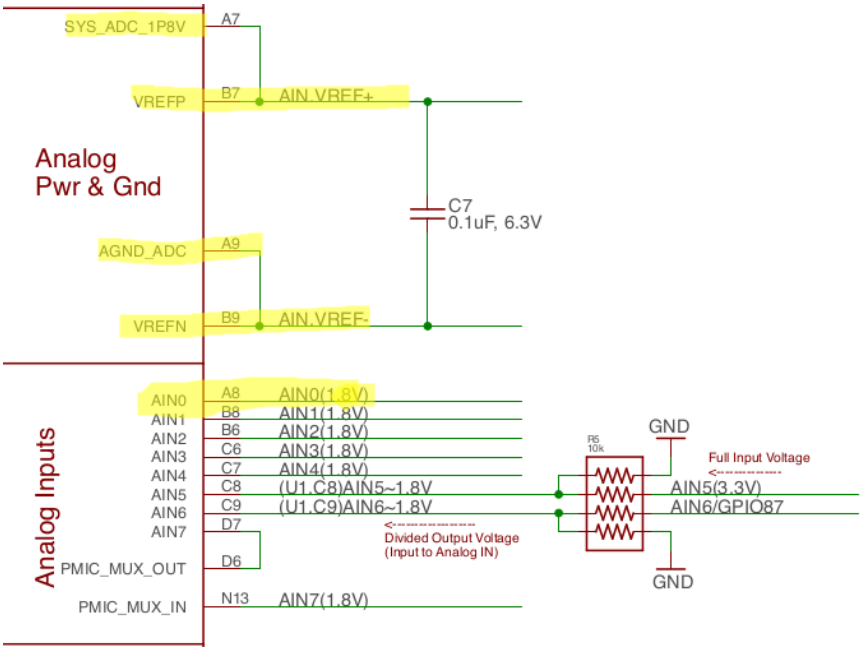

Figure 2.7: **PocketBeagle Schematic Analog**

2.3 Summary

Hardware Investigation Results

- Thumbwheel:
  - Connected to analog input 0 (**AIN0**)

2.4 Devicetree

Devicetree Configuration

No need to do any pinmuxing since the analog pins are not muxed.

It is necessary to enable the AM335x ADC control, however, am335x-pocketbeagle.dts includes am335x-pocketbeagle-common.dtsi which contains:

```
&tscadc {
        status = "okay";
        adc {
                ti,adc-channels = <0 1 2 3 4 5 6 7>;
                ti,chan-step-avg = <0x16 0x16 0x16 0x16 0x16 0x16 0x16 0x16>;
                ti,chan-step-opendelay = <0x98 0x98 0x98 0x98 0x98 0x98 0x98 0x98>;
                ti,chan-step-sampledelay = <0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0>;
        };
};
```

https://github.com/beagleboard/linux/blob/4.14/arch/arm/boot/dts/am335x-pocketbeagle-common.dtsi#L1100

As a result, the platform is configured and enabled by default to use all 8 channels of the ADC at boot.

Chapter 3

Linux Subsystems

e-ale

3.1 IIO Subsystem

Overview

- The Linux IIO subsystem is a framework to support sensors and any type of device with ADCs or DACs
- https://www.kernel.org/doc/html/v5.0/driver-api/iio/intro.html
- IIO provides a core framework to support device drivers in a common manner
- IIO also provides an interface for in-kernel users of IIO devices
- IIO provides a high-latency userspace interface via sysfs
- IIO provides an efficient buffered interface via character device

IIO architecture

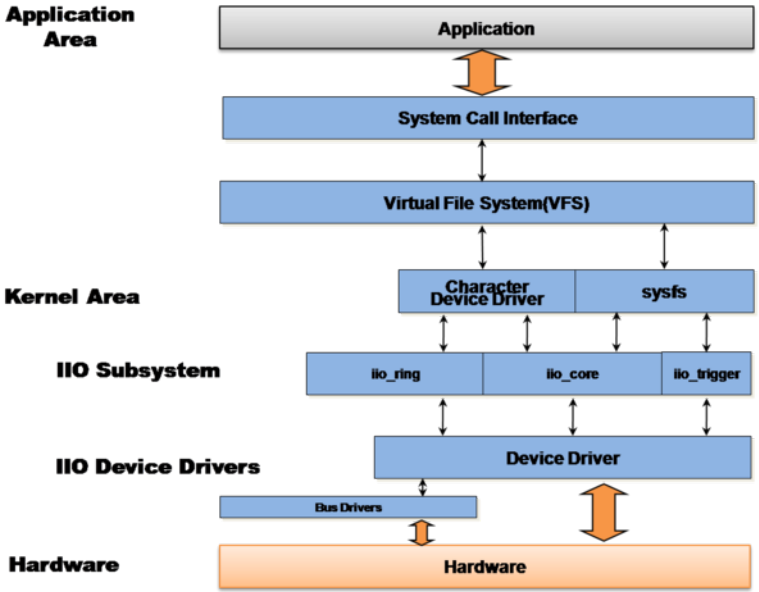

Figure 3.1: **IIO architecture**

IIO concepts

- Producer/Consumer model
  - Sensor/ADC drivers are producers of samples.
  - In-kernel drivers or userspace clients are consumers of samples.
- Ring buffer (efficient buffered access to samples)
  - Hardware and software circular buffer support for producers to place samples.
- Triggers (external events to **trigger** capture of a sample)
  - GPIO
  - RTC
  - sysfs file
- Scaled samples
  - Ability to provide both raw samples and scaled (in units relevant to the sensor e.g. mA, mV, lumens, foot pounds per fortnight, etc.)

IIO userspace ABI

The canonical ABI documentation is here https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/tree/Documentation/ABI/testing/sysfs-bus-iio

Focusing on the polled sysfs interface:

- IIO device nodes
  - /sys/bus/iio/devices/iio:deviceN
- IIO device attribute nodes
  - /sys/bus/iio/devices/iio:deviceN/in_*_raw
  - /sys/bus/iio/devices/iio:deviceN/in_*_scale
    - Variations for each type of sample (voltage, current, power, temperature

IIO temp sensor example

Example graciously provided by Matt Ranostay from his https://elinux.org/images/b/ba/ELC_2017_-_Industrial_IO_and_You-_Nonsense_Hacks!.pdf presentation.

```
$ cd /sys/bus/iio/devices/iio:device0
$ cat in_temp_raw
98
$ cat in_temp_ambient_raw
416
$ cat in_temp_scale
250
$ cat in_temp_ambient_scale
62.500000
```

libiio

libiio is a higher level userspace library for working with IIO devices. The official documentation is at `http://analogdevicesinc.github.io/libiio/`. It was created in response to the difficulty of working with the low-level IIO userspace ABI for buffered I/O management.

libiio provides high level APIs to manage

- polled sysfs read/write attributes
- trigger events
- buffered samples

libiio is also used by the **iiod** server to provide networked access to IIO data and provides several utilities like **iio_info** and **iio_attr** for retrieving data from IIO.

3.2 Input Subsystem

Overview

- The Linux Input subsystem is a framework to support all types of input devices
- https://www.kernel.org/doc/html/v5.0/input/input.html
- Consists of the core **input module**, **device drivers**, and **event handlers**

Device Drivers

- **Device drivers** interface with hardware and provide events to the **input module**
- Examples are:
  - **gpio_keys**
  - **hid-generic**
  - **usbmouse**

Event Handlers

- **Event handlers** interface with the **input module** and pass events to other kernel subsystems or userspace
  - **evdev** passes generic input events to userspace. Devices are in **/dev/input**:

```
crw-r--r--   1 root     root      13,  64 Apr  1 10:49 event0
crw-r--r--   1 root     root      13,  65 Apr  1 10:50 event1
crw-r--r--   1 root     root      13,  66 Apr  1 10:50 event2
crw-r--r--   1 root     root      13,  67 Apr  1 10:50 event3
```

  - **joydev** passes joystick events to userspace. Devices are in **/dev/input**:

```
crw-r--r--   1 root     root      13,   0 Apr  1 10:50 js0
crw-r--r--   1 root     root      13,   1 Apr  1 10:50 js1
crw-r--r--   1 root     root      13,   2 Apr  1 10:50 js2
crw-r--r--   1 root     root      13,   3 Apr  1 10:50 js3
```

evdev

- **evdev** nodes support blocking/non-blocking **read** and **select**
- Reading an **evdev** node returns a **struct input_event**:

```
struct input_event
{
        struct timeval time;
        unsigned short type;
        unsigned short code;
        unsigned int value;
};
```

evtest

- **evtest** can be used to test evdev events at the command line
- Example:

```
$ evtest /dev/input/event0
Input driver version is 1.0.1
Input device ID: bus 0x3 vendor 0x46d product 0x1028 version 0x111
Input device name: "Logitech M570"
Supported events:
Event type 0 (EV_SYN)
Event type 1 (EV_KEY)
Event code 272 (BTN_LEFT)
...
Event type 4 (EV_MSC)
Event code 4 (MSC_SCAN)
Properties:
Testing ... (interrupt to exit)
Event: time 1540159516.312712, type 4 (EV_MSC), code 4 (MSC_SCAN), value 90001
Event: time 1540159516.312712, type 1 (EV_KEY), code 272 (BTN_LEFT), value 1
Event: time 1540159516.312712, -------------- SYN_REPORT ------------
```

Chapter 4

Labs

4.1 Lab 1

IIO sysfs interface

In Lab1, we will accomplish the following:

- Within sysfs, locate the IIO device corresponding to the thumbwheel's ADC channel
- Read the position of the thumbwheel using a sysfs file

Find our ADC IIO device

Confirm that the IIO subsystem has exported a device sysfs:

```
root@beaglebone:~# ls /sys/bus/iio/devices/
iio:device0
```

Is that our device? Let's check the actual Pocketbeagle/AM335x TSCADC device:

```
root@beaglebone:~# ls /sys/bus/platform/devices/44e0d000.tscadc\:adc
driver           iio:device0  of_node  subsystem
driver_override  modalias     power    uevent
```

Sure enough, there's a link to that same IIO device node.

Read the ADC channel 0 value

Find the file corresponding to ADC channel 0:

```
root@beaglebone:~# ls /sys/bus/iio/devices/iio\:device0
buffer           in_voltage2_raw  in_voltage6_raw  power
dev              in_voltage3_raw  in_voltage7_raw  scan_elements
in_voltage0_raw  in_voltage4_raw  name             subsystem
in_voltage1_raw  in_voltage5_raw  of_node          uevent
```

Read the raw ADC channel 0 value:

```
root@beaglebone:~# cat /sys/bus/iio/devices/iio\:device0/in_voltage0_raw
4095
```

Try turning the thumbwheel and reading the position again, what are the results?

4.2 Lab 2

Using libiio utilities

libiio is a helper library that abstracts away some of the complications of the raw IIO userspace ABI.

In Lab2, we will accomplish the following:

- Using libiio, discover all devices
- Using libiio, read the current position of the thumbwheel

Preparation

Install a newer version of the **libiio** and **libiio-utils** packages from sid:

```
$ sudo apt update
$ sudo apt install -t sid libiio libiio-utils
```

Find our ADC IIO devices

Example which devices the IIO subsystem has exported:

```
root@beaglebone:/# iio_info
                Library version: 0.16 (git tag: v0.16)
                Compiled with backends: local xml ip usb serial
                IIO context created with local backend.
                Backend version: 0.16 (git tag: v0.16)
                Backend description string: Linux beaglebone 4.14.71-ti-r80 #1 SMP PREEMPT Fri Oct 5 23:50:11 UTC 2018 armv7l
                IIO context has 1 attributes:
                        local,kernel: 4.14.71-ti-r80
                IIO context has 1 devices:
                        iio:device0: 44e0d000.tscadc:adc (buffer capable)
                                8 channels found:
                                        voltage0:  (input, index: 0, format: le:u12/16>>0)
                                        1 channel-specific attributes found:
                                                attr  0: raw value: 4095
                                        voltage1:  (input, index: 1, format: le:u12/16>>0)
                                        1 channel-specific attributes found:
                                                attr  0: raw value: 3680
                                        voltage2:  (input, index: 2, format: le:u12/16>>0)
                                        1 channel-specific attributes found:
                                                attr  0: raw value: 3979
                .
                .
                .
```

We should be able to identify the ADC channel that represents AIN0

Read the ADC channel 0 value with libiio

Using **44e0d000.tscadc:adc** and **voltage0** we can read the position the thumbwheel: Read the raw ADC channel 0 value:

```
root@beaglebone:/# iio_attr -c 44e0d000.tscadc:adc voltage0
dev '44e0d000.tscadc:adc', channel 'voltage0' (input), attr 'raw', value '4095'
```

or for terser output:

```
root@beaglebone:~# iio_attr -c 44e0d000.tscadc:adc voltage0 | cut -d ' ' -f 9
'4095'
```

Try turning the thumbwheel and reading the position again, what are the results?

Read ADC values continuously

```
#!/bin/bash
while [ 1 ]; do
        echo `iio_attr -c 44e0d000.tscadc:adc voltage0 | cut -d ' ' -f 9`
done
```

4.3 Lab 3

Userspace Input Driver

In Lab3, we will accomplish the following:

- Learn about the kernel uinput module and python-uinput
- Create a userspace joystick driver
- Test the joystick driver

Preparation

Install the python uinput module and evtest:

```
# pip3 install python-uinput
# apt install evtest
```

uinput drivers

Linux userspace input driver provides methods to:

- Register input devices with the kernel
- Register types of events the userspace driver may generate (and ranges if applicable. e.g. BTN, KEY, ABS
- Trigger events from userspace

  The uinput driver interface and C APIs are described in detail at https://www.kernel.org/doc/html/v5.0/input/uinput.html

Write a uinput joystick driver in Python

python-uinput provides easy to use Python support for uinput drivers. The following is a complete joystick driver:

```
#!/usr/bin/python3

from subprocess import Popen, PIPE, STDOUT
import uinput

events = (
        uinput.ABS_X + (0, 4095, 0, 0),
        )

with uinput.Device(events) as device:
    cmd = 'iio_attr -c 44e0d000.tscadc:adc voltage0'
    while (1):
        p = Popen(cmd, shell=True, stdout=PIPE, close_fds=True)
        output = p.stdout.read()
        string = output.decode("utf-8")
        cols = string.split(' ');
        value = cols[8][1:-2]
        device.emit(uinput.ABS_X, int(value))
```

Test the joystick driver

```
# modprobe uinput
# ./joystick.py &
# evtest /dev/input/event1
Event: time 1538932264.800879, -------------- SYN_REPORT ------------
Event: time 1538932264.853011, type 3 (EV_ABS), code 0 (ABS_X), value 1790
Event: time 1538932264.853011, -------------- SYN_REPORT ------------
Event: time 1538932265.006461, type 3 (EV_ABS), code 0 (ABS_X), value 1795
Event: time 1538932265.006461, -------------- SYN_REPORT ------------
Event: time 1538932265.058354, type 3 (EV_ABS), code 0 (ABS_X), value 2153
Event: time 1538932265.058354, -------------- SYN_REPORT ------------
.
.
.
```

Try turning the thumbwheel, what are the results?

Extra credit: play Tetris

- Copy https://www.victornils.net/tetris/vitetris-0.57.tar.gz to the PocketBeagle
- ssh to the PocketBeagle for better performance

```
# tar jxf vitetris-0.57.tar.gz
# cd vitetris-0.57
# ./configure
# make
# ./tetris
```

4.4 Extra Credit

Write a kernel joystick driver

https://cm.e-ale.org/2018/iioinput-drivers/iioinput-SLIDES.pdf

