# Blinking an LED

….The hard way.

# In Linux, everything is a file

Learning about Linux through SYSFS

Thanks to Bill Gatliff

# What is SYSFS?

- Virtual file system that exposes drivers to userspace
- **`/sys/devices`** ← driver hierarchy
- **`/sys/class`** ← common interfaces

- Let's go thru some examples…

# What is SYSFS?

- Virtual file system that exposes drivers to userspace
- bone$ **cd /sys/class**
- bone$ **ls**

```
backlight   firmware    lcd       net           scsi_device   tty
bdi         gpio        leds      power_supply  scsi_disk     udc
block       graphics    mbox      pwm           scsi_generic  usb_device
bluetooth   hwmon       mdio_bus  regulator     scsi_host     vc
bsg         i2c-adapter mem       rfkill        sound         video4linux
devfreq     i2c-dev     misc      rtc           spi_master    vtconsole
display     input       mmc_host  scsi_changer  spidev
```

- Let's go through some examples…

# Blinking an LED

- Everything is a file in Linux

```
$ cd /sys/class/leds
$ ls -F
bat100  beaglebone:green:usr0  green
bat25   beaglebone:green:usr1  red
bat50   beaglebone:green:usr2  wifi
bat75   beaglebone:green:usr3  wl18xx_bt_en
$ cd beaglebone\:green\:usr0
$ ls
brightness  max_brightness  subsystem  uevent
device      power           trigger
```

# Blinking an LED

```
$ cat trigger
none nand-disk mmc0 timer oneshot [heartbeat]
     backlight gpio cpu0 default-on transient
$ echo none > trigger
$ echo 1 > brightness
$ echo 0 > brightness
```

# Blinking an External LED

- The gpio pins are accessed through **`/sys/class/gpio`**
- Earlier we used gpio **`P9_14`**
- The table shows which gpio pin it's assigned to

# P9 Pin Header Table

| Pin | $PINS | ADDR | GPIO | Name | Mode7 | Mode6 |
|---|---|---|---|---|---|---|
| P9_01 | | 44e10000 | | GND | | |
| P9_02 | | Offset from: | | GND | | |
| P9_03 | | 44e10800 | | DC_3.3V | | |
| P9_04 | | | | DC_3.3V | | |
| P9_05 | | | | VDD_5V | | |
| P9_06 | | | | VDD_5V | | |
| P9_07 | | | | SYS_5V | | |
| P9_08 | | | | SYS_5V | | |
| P9_09 | | | | PWR_BUT | | |
| P9_10 | | | | SYS_RESETn | | |
| P9_11 | 28 | 0x870/070 | 30 | UART4_RXD | gpio0[30] | uart4_rxd_mux2 |
| P9_12 | 30 | 0x878/078 | 60 | GPIO1_28 | gpio1[28] | mcasp0_aclkr_mux3 |
| P9_13 | 29 | 0x874/074 | 31 | UART4_TXD | gpio0[31] | uart4_txd_mux2 |
| P9_14 | 18 | 0x848/048 | 50 | EHRPWM1A | gpio1[18] | ehrpwm1A_mux1 |
| P9_15 | 16 | 0x840/040 | 48 | GPIO1_16 | gpio1[16] | ehrpwm1_tripzone_input |
| P9_16 | 19 | 0x84c/04c | 51 | EHRPWM1B | gpio1[19] | ehrpwm1B_mux1 |

# Blinking an External LED

- Here's how you turn it on

```
$ cd /sys/class/gpio
$ ls
export gpiochip0 gpiochip32 gpiochip64 gpiochip96
unexport
```

- If no gpio pins are visible. Use P9_14

```
$ echo 50 > export
$ ls
export gpio50 gpiochip0 gpiochip32 gpiochip64 …
```

- Notice **gpio40** has appeared

# Blinking an External LED

- Go in a take control

```
bone$ cd gpio50
bone$ ls
active_low  direction  edge  power  subsystem  uevent
value
bone$ echo out > direction
bone$ echo 1 > value
```

- Your LED should be on

# Reading a switch

- Once you know how to control an LED, reading a switch is easy
- A switch is wired to **P9_42**. Which gpio is this?

```
$ cd /sys/class/gpio
$ echo 7 > export
$ cd gpio7
$ echo in > direction
```

# Reading a Switch

- Button not pushed

```
$ cat value
0
```

- Button pushed

```
$ cat value
1
```

# Read in a Loop

- You ca over and over

Spaces are important

```
#!/bin/ba
cd /sys/ lass/gpio
while [ 1 ]
do
    cat gpio57/value
    sleep 0.25
done
```

```
tr '\n' '\r' < gpio57/value
```

# Analog In

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_50 |
| GPIO_48 | 15 | 16 | GPIO_51 |
| GPIO_5 | 17 | 18 | GPIO_4 |
| I2C2_SCL | 19 | 20 | I2C2_SDA |
| GPIO_3 | 21 | 22 | GPIO_2 |
| GPIO_49 | 23 | 24 | GPIO_15 |
| GPIO_117 | 25 | 26 | GPIO_14 |
| GPIO_115 | 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 |

# Analog In

- Input voltage range is 0 to 1.8V.
- These are accessed much link the gpio

```
$ export SLOTS="/sys/devices/platform/bone_capemgr/slots"
$ echo BB-ADC > $SLOTS
$ cd /sys/bus/iio/devices/iio:device0
$ ls -F
buffer/          in_voltage1_raw  in_voltage4_raw  name      scan_elements/
dev              in_voltage2_raw  in_voltage5_raw  of_node@  subsystem@
in_voltage0_raw  in_voltage3_raw  in_voltage6_raw  power/    uevent
$ cat in_voltage0_raw
3936
```

# Analog In - Explore

- How did I figure this out?
- The variable NODE_PATH tells where the node modules are kept

```
bone$ echo $NODE_PATH
/usr/local/lib/node_modules
```

- See what's there

```
bone$ ls $NODE_PATH
async       i2c       node-red-node-bb-upm      npm         serialport
blessed     mraa      node-red-node-beaglebone  request     socket.io
bonescript  node-red  node-red-node-mongodb     sensortag   winston
```

# Analog In

- You can keep reading the input using

```
while [ 1 ]
do
 tr '\n' '\r' < in_voltage0_raw
done
```