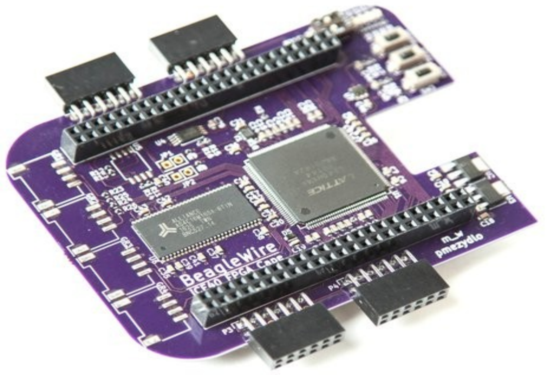
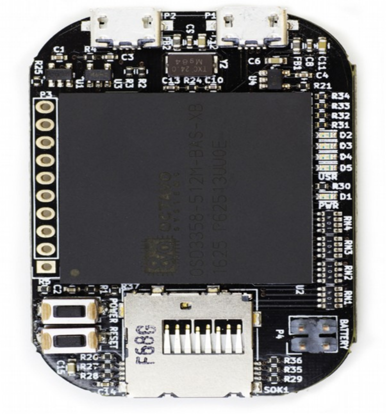
Introduction to the LED subsystem

Michael Welling (m_w)
mwelling@ieee.org

whoami

Michael Welling
Founder, QWERTY Embedded Design, LLC





PCB Design
Linux Board Support
Firmware

Community Efforts
- Beagleboard.org GSoC Mentor
- 96Boards.org Community Mezzanine Initiative
- E-ale.org Instructor
- KiCAD Training

QWERTY
EMBEDDED DESIGN

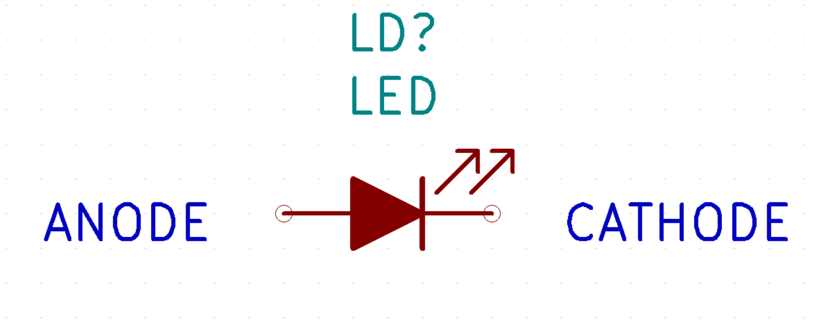
Overview

- What’s an LED?
- History of LEDs
- History of the LED subsystem
- LED class interface
- LED triggers
- LED devicetree binding
- LED driver API
- Userspace LEDs driver
- Labs

What’s an LED?

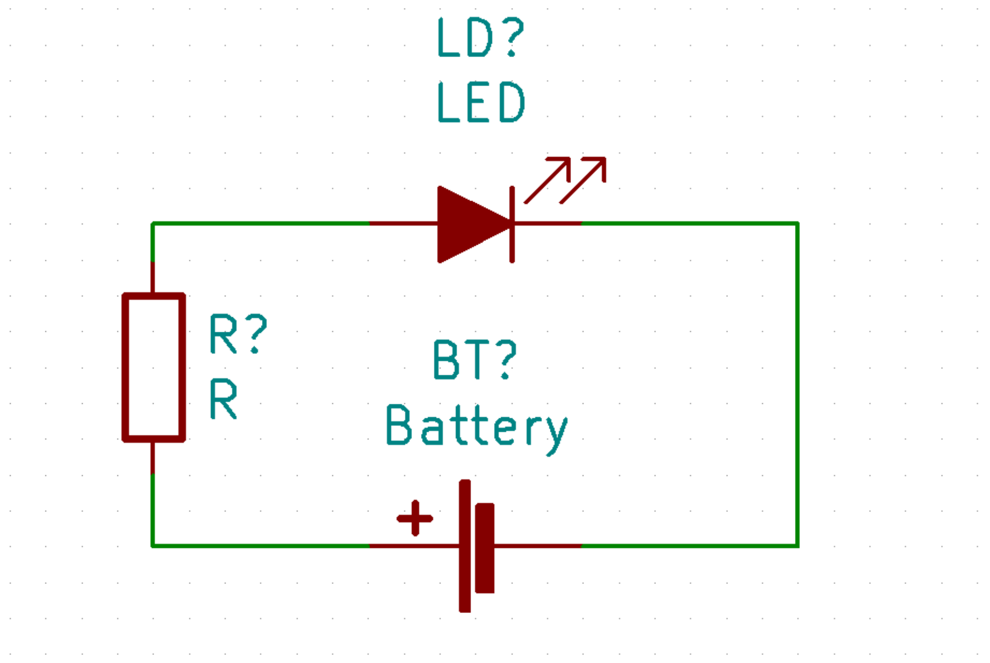
Light emitting diode or LED is a semiconductor device that emits light when current passes through it.

This is due to a phenomenon called electroluminescence which occurs when an electron combines with a hole emitting a photon at the P-N junction.

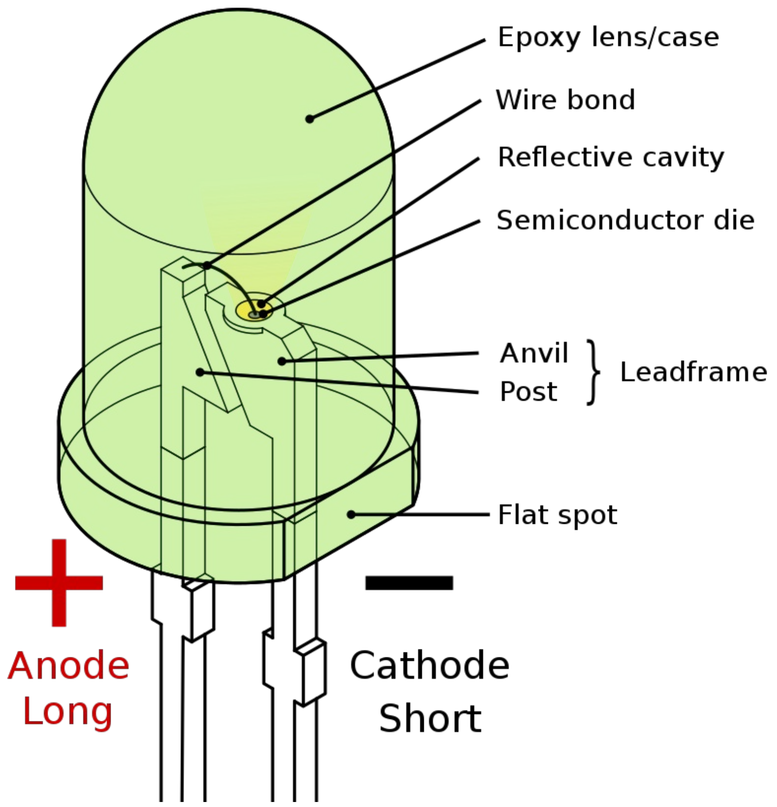


What’s an LED?

$R_s = (V_{supply} - V_f) / I_f$



What’s an LED?


Epoxy lens/case
Wire bond
Reflective cavity
Semiconductor die
Anvil
Post
Leadframe
Flat spot
+
Anode
Long
−
Cathode
Short

LED History

- This effect was first observed in 1907 by H.J. Round while experimenting with cat whisker detectors on various crystalline substances.
- Similar observation were made by Oleg Losev in 1927 at point contacts of silicon carbide. Losev went on to study the phenomenon in great detail and published several articles on the topic.
- In 1961, James R. Biard pioneered work on the first GaAs based infrared LED. TI received a patent for this invention and this would provide the foundation for the earliest visible spectrum LED.
- In 1962, Nick Holonyak invented the first visible light red GaAsP LED while working at GE.

  His work led to the commercialization of LEDs.
- In the 1970s, blue and green LEDs were invented through use of GaN. Herb Maruska, Wally Rhines, and Jacques Pankove pioneered this work and their work was the basis of high brightness blue LEDs developed in the early 1990s. White LEDs followed shortly thereafter.

LED subsystem history

First introduced to the mainline kernel in 2006 by Richard Purdie extending the work of John Lenz.

https://lwn.net/Articles/169919/

Previous maintainers included Richard and Bryan Wu.

Currently co-maintained by Jacek Anaszewski and Pavel Machek.

LED subsystem

```
LED SUBSYSTEM
M:	Jacek Anaszewski <jacek.anaszewski@gmail.com>
M:	Pavel Machek <pavel@ucw.cz>
R:	Dan Murphy <dmurphy@ti.com>
L:	linux-leds@vger.kernel.org
T:	git git://git.kernel.org/pub/scm/linux/kernel/git/j.anaszewski/linux-leds.git
S:	Maintained
F:	Documentation/devicetree/bindings/leds/
F:	drivers/leds/
F:	include/linux/leds.h
```

LED subsystem

List: linux-leds

Info:

This is the mailing list for Linux LEDS development

Archives:

http://www.spinics.net/lists/linux-leds/

LED class interface

LEDs can be controlled from the Linux userspace using the sysfs interface provided in /sys/class/leds.

Each registered LED will be given a folder named with the following convention:

"devicename:color:function"

The folder will contains follow attribute files to control the LED:

- brightness – LED brightness
- max_brightness – Maximum available brightness
- trigger – Trigger for blinking the LED
- pattern – Allows user to apply a blinking pattern to the LED.

Complex triggers will expose additional attributes related to the trigger.

LED triggers

heartbeat – triggers an LED to blink like a heartbeat

timer – periodically triggers LEDs on and off

cpu* – triggers an LED to blink on cpu activity

ide-disk – triggers an on IDE disk activity

usb-host – triggers an LED on USB activity

mmc* – triggers an LED on MMC activity

LED triggers (cont)

gpio – triggers on GPIO events

oneshot – trigger on arbitrary event and stay on for a specified period of time

rfkill* – trigger on wireless power on/off

kbd* – trigger on keyboard event

Devicetree binding (GPIO)

LEDs connected to GPIO lines

Required properties:

- compatible : should be "gpio-leds".

Each LED is represented as a sub-node of the gpio-leds device. Each node's name represents the name of the corresponding LED.

Devicetree binding (GPIO)

LED sub-node properties:

- gpios : Should specify the LED's GPIO, see "gpios property" in
  Documentation/devicetree/bindings/gpio/gpio.txt. Active low LEDs should be
  indicated using flags in the GPIO specifier.
- label : (optional)
  see Documentation/devicetree/bindings/leds/common.txt
- linux,default-trigger : (optional)
  see Documentation/devicetree/bindings/leds/common.txt

Devicetree binding (GPIO)

LED sub-node properties (cont):

- default-state: (optional) The initial state of the LED.
  see Documentation/devicetree/bindings/leds/common.txt
- retain-state-suspended: (optional) The suspend state can be retained.Such
  as charge-led gpio.
- retain-state-shutdown: (optional) Retain the state of the LED on shutdown.
  Useful in BMC systems, for example when the BMC is rebooted while the host
  remains up.
- panic-indicator : (optional)
  see Documentation/devicetree/bindings/leds/common.txt

Devicetree binding (GPIO)

```
#include <dt-bindings/gpio/gpio.h>

leds {
	compatible = "gpio-leds";
	hdd {
		label = "Disk Activity";
		gpios = <&mcu_pio 0 GPIO_ACTIVE_LOW>;
		linux,default-trigger = "disk-activity";
	};
};
```

Devicetree binding (PWM)

LED connected to PWM

Required properties:

- compatible : should be "pwm-leds".

Each LED is represented as a sub-node of the pwm-leds device. Each node's name represents the name of the corresponding LED.

Devicetree binding (PWM)

LED sub-node properties:

- pwms : PWM property to point to the PWM device (phandle)/port (id) and to
  specify the period time to be used: <&phandle id period_ns>;
- pwm-names : (optional) Name to be used by the PWM subsystem for the PWM device

  For the pwms and pwm-names property please refer to:

  Documentation/devicetree/bindings/pwm/pwm.txt
- max-brightness : Maximum brightness possible for the LED

Devicetree binding (PWM)

LED sub-node properties (cont):

- active-low : (optional) For PWMs where the LED is wired to supply
  rather than ground.
- label : (optional)
  see Documentation/devicetree/bindings/leds/common.txt
- linux,default-trigger : (optional)
  see Documentation/devicetree/bindings/leds/common.txt

Devicetree binding (PWM)

```
pwmleds {
        compatible = "pwm-leds";
        kpad {
                label = "omap4::keypad";
                pwms = <&twl_pwm 0 7812500>;
                max-brightness = <127>;
        };
};
```

LED driver API

A driver wanting to register a LED class device for use by other drivers / userspace needs to allocate and fill a ***led_classdev*** struct and then call ***led_classdev_register*** or ***devm_led_classdev_register***.

If the non devm version is used the driver must call ***led_classdev_unregister*** from its remove function before free-ing the ***led_classdev*** struct.

LED driver API

led_set_brightness:

it is guaranteed not to sleep, passing ***LED_OFF*** stops blinking.

led_set_brightness_sync:

for use cases when immediate effect is desired - it can block the caller for the time required for accessing device registers and can sleep, passing ***LED_OFF*** stops hardware blinking, returns ***-EBUSY*** if software blink fallback is enabled.

LED driver API

led_get_brightness:

Get the current LED brightness level.

blink_set:

Hardware accelerated blinking set. delay_on and delay_off passed and matched as closely as possible.

Userspace LEDs driver

Userspace LEDs driver allows a user to instantiate driver by opening ***/dev/uleds*** and writing a ***uleds_user_dev*** struct.

This can be useful for testing triggers and can also be used to implement virtual LEDs.

https://elixir.bootlin.com/linux/latest/source/tools/leds/uledmon.c

LAB

- Investigate LED devicetree bindings and corresponding sysfs entries.
- Use GPIO trigger to turn on the RBG’s Green LED when L button is pressed and the red when R button is pressed.
- Write a program to write hexadecimal codes to the seven segment LEDs.
- Write a program that reads the ADC from the light sensor and changes the brightness of the RGB’s Blue LED.