ELEC 424/553

Mobile & Embedded Systems

Lecture 15

Device Trees

Google Slides Link:
https://docs.google.com/presentation/d/12NhqB2oumZLMAdyMIhZ__mOPM4p-D0Lp8QoXPU2QQSg/edit?usp=sharing

Picture URL: https://charlesandhudson.com/flamethrower-drone/

Housekeeping

- Assignment 2 due Saturday
- Project 2 to be posted Monday
  - gpiod - welcome to the driver (dark) side
- The one and only exam/midterm will be posted on Mon Oct 30
  - Canvas
  - Online
  - Take home
  - Open class notes and lecture slides
  - Open assignments/projects/exercises
- Final project details to come
  - Start forming teams of 4
  - Must be strictly undergrad team or graduate team

2022: Hope Accelerates

The New York Times

Tesla Sold Record 343,000 Vehicles in Third Quarter

The increase from the same period last year comes as competition grows among automakers producing electric vehicles in the United States.

Give this article

A Tesla charging station in Santa Monica, Calif. Allison Dinner/Getty Images

By Neal E. Boudette

Oct. 2, 2022

Tesla said on Sunday that it sold 343,830 electric vehicles worldwide in the third quarter, a record number even as the automaker faces ongoing production and supply chain challenges.

The company's sales figures were a 42 percent increase from the 241,391 vehicles it sold in the third quarter of 2021. Tesla also said it produced 365,923 cars, compared with a year-ago total of 237,823. The automaker opened two major factories earlier this year — one in Austin, Texas, and the other in Germany.

https://www.nytimes.com/2022/10/02/business/tesla-sales.html

electrek

Exclusives | Autos | Alt. Transport | Autonomy | Energy | Tesla Shop

Ads by Google

Send feedback | Why this ad?

OCTOBER 20

Tesla is aiming to ramp up to 50,000 Tesla Semi electric trucks per year

Fred Lambert - Oct. 20th 2022 7:37 am PT @FredericLambert

273 Comments | Facebook | Twitter | Pinterest | LinkedIn | Reddit

Tesla is aiming to ramp up Tesla Semi production to 50,000 electric trucks per year – as soon as 2024. It would make Tesla one of the largest class 8 truck manufacturers.

https://electrek.co/2022/10/20/tesla-semi-ramp-50000-electric-trucks-per-year/

2023: Reality Hits

INSIDER

HOME > TRANSPORTATION

Tesla's Cybertruck is still MIA

Nora Naughton Oct 2, 2023, 12:58 PM CDT

Lots of questions remain about Elon Musk's polarizing Cybertruck. Photo by Frederic J. BROWN / AFP via Getty Images

- Elon Musk is still making complex demands about the appearance of the Cybertruck.
- Details on Tesla's futuristic-looking Cybertruck are still sparse.
- No date has been set for a delivery event this month.

Screenshot of article from *Insider*. URL: https://www.businessinsider.com/tesla-cybertruck-release-date-delivery-delay-signs-2023-9

THE WALL STREET JOURNAL.

SUBSCRIBE | SIGN IN

BUSINESS | AUTOS

Automakers Have Big Hopes for EVs; Buyers Aren't Cooperating

Sales growth has slowed in the U.S. as car companies are finding a limited pool of consumers willing to pay more for these models

By Sean McLain Follow
Oct. 15, 2023 5:30 am ET

Share | Resize | 2747 | Listen (1 min)

Ford says supplies for the Mustang Mach-E have steadily increased since June after a planned factory shutdown. PHOTO: NICK HAGEN FOR THE WALL STREET JOURNAL

The auto industry's push to boost sales of electric vehicles is running into a cold, hard reality: Buyers' interest in these models is proving shallower than expected.

While EV sales continue to grow—rising 51% this year through September—the rate has slowed from a year earlier and unsold inventory is starting to pile up for some brands.

Some car companies, such as Ford Motor F -1.50% ▼ and Toyota Motor TM -1.17% ▼, are tempering their expectations for EVs and shifting more resources into hybrids, which have been drawing consumers at a faster clip.

Screenshot of article from *The Wall Street Journal*. URL: https://www.wsj.com/business/autos/electric-vehicle-buyer-interest-67b407cb

Accessing GPIO

- **Kernel space**
  - `gpio`
  - **`gpiod`**
- **User space**
  - `sysfs`
  - `Char dev` [Kernel 4.8]
    - **`libgpiod`**
      - Command line & C program

Accessing GPIO

- **Kernel space**
  - `gpio`
  - `gpiod`
- **User space**
  - `sysfs`
  - `Char dev` [Kernel 4.8]
    - `libgpiod`
      - Command line & C program

What Does `gpiod` Solve?

Text from this commit on kernel.org (2013):

This patch exports the gpiod_* family of API functions, a safer alternative to the legacy GPIO interface. Differences between the gpiod and legacy gpio APIs are:

- **gpio works with integers**, whereas **gpiod operates on opaque handlers** which cannot be forged or used before proper acquisition
- gpiod get/set functions are **aware of the active low state of a GPIO**
- gpio consumers should now include **<linux/gpio/consumer.h>** to access the new interface, whereas chips drivers will use <linux/gpio/driver.h>

The legacy gpio API is now built as inline functions on top of gpiod.

Signed-off-by: Alexandre Courbot <acourbot@nvidia.com>
Signed-off-by: Linus Walleij <linus.walleij@linaro.org>

Kernel Space: `gpio`

- Old functions (**`gpio`**) were **gpio_*** (**`gpio.h`**) - link to code shown in Bootlin Elixir screenshot below. Deprecated now.

/ include / linux / gpio.h All syml Search Identifier

```
/* SPDX-License-Identifier: GPL-2.0 */
/*
 * <linux/gpio.h>
 *
 * This is the LEGACY GPIO bulk include file, including legacy APIs. It is
 * used for GPIO drivers still referencing the global GPIO numberspace,
 * and should not be included in new code.
 *
 * If you're implementing a GPIO driver, only include <linux/gpio/driver.h>
 * If you're implementing a GPIO consumer, only include <linux/gpio/consumer.h>
 */
#ifndef __LINUX_GPIO_H
#define __LINUX_GPIO_H

#include <linux/errno.h>

/* see Documentation/driver-api/gpio/legacy.rst */
```

Kernel Space: gpiod

- New functions (**gpiod**) are **gpiod_*** (**consumer.h**) - link to code shown in Bootlin Elixir screenshot below.

/ include / linux / gpio / consumer.h All syml Search Identifier

```
/* Acquire and dispose GPIOs */
struct gpio_desc *__must_check gpiod_get(struct device *dev,
                                         const char *con_id,
                                         enum gpiod_flags flags);
struct gpio_desc *__must_check gpiod_get_index(struct device *dev,
                                               const char *con_id,
                                               unsigned int idx,
                                               enum gpiod_flags flags);
struct gpio_desc *__must_check gpiod_get_optional(struct device *dev,
                                                  const char *con_id,
                                                  enum gpiod_flags flags);
struct gpio_desc *__must_check gpiod_get_index_optional(struct device *dev,
                                                        const char *con_id,
                                                        unsigned int index,
                                                        enum gpiod_flags flags);
struct gpio_descs *__must_check gpiod_get_array(struct device *dev,
                                                const char *con_id,
                                                enum gpiod_flags flags);
struct gpio_descs *__must_check gpiod_get_array_optional(struct device *dev,
                                                         const char *con_id,
                                                         enum gpiod_flags flags);
void gpiod_put(struct gpio_desc *desc);
void gpiod_put_array(struct gpio_descs *descs);

struct gpio_desc *__must_check devm_gpiod_get(struct device *dev,
                                              const char *con_id,
```

First Consider struct gpio_desc

/ include / linux / gpio / consumer.h All syml Search Identi

```
/* Acquire and dispose GPIOs */
struct gpio_desc *__must_check gpiod_get(struct device *dev,
                                         const char *con_id,
                                         enum gpiod_flags flags);
struct gpio_desc *__must_check gpiod_get_index(struct device *dev,
                                               const char *con_id,
                                               unsigned int idx,
                                               enum gpiod_flags flags);
struct gpio_desc *__must_check gpiod_get_optional(struct device *dev,
                                                  const char *con_id,
                                                  enum gpiod_flags flags);
struct gpio_desc *__must_check gpiod_get_index_optional(struct device *dev,
                                                        const char *con_id,
                                                        unsigned int index,
                                                        enum gpiod_flags flags);
struct gpio_descs *__must_check gpiod_get_array(struct device *dev,
                                                const char *con_id,
                                                enum gpiod_flags flags);
struct gpio_descs *__must_check gpiod_get_array_optional(struct device *dev,
                                                         const char *con_id,
                                                         enum gpiod_flags flags);
void gpiod_put(struct gpio_desc *desc);
void gpiod_put_array(struct gpio_descs *descs);

struct gpio_desc *__must_check devm_gpiod_get(struct device *dev,
                                              const char *con_id,
```

```
/**
 * Opaque descriptor for a GPIO. These are obtained using gpiod_get() and are
 * preferable to the old integer-based handles.
 *
 * Contrary to integers, a pointer to a gpio_desc is guaranteed to be valid
 * until the GPIO is released.
 */
struct gpio_desc;
```

Bootlin Elixir Screenshots. URL:
https://elixir.bootlin.com/linux/latest/source/include/linux/gpio/consumer.h

Asking The Spooky Questions

- What is `static device *dev`?
- What is `const char *con_id`?

/ include / linux / gpio / consumer.h All symb Search Identifier

```
/* Acquire and dispose GPIOs */
struct gpio_desc *__must_check gpiod_get(struct device *dev,
                                         const char *con_id,
                                         enum gpiod_flags flags);
struct gpio_desc *__must_check gpiod_get_index(struct device *dev,
                                               const char *con_id,
                                               unsigned int idx,
                                               enum gpiod_flags flags);
struct gpio_desc *__must_check gpiod_get_optional(struct device *dev,
                                                  const char *con_id,
                                                  enum gpiod_flags flags);
struct gpio_desc *__must_check gpiod_get_index_optional(struct device *dev,
                                                        const char *con_id,
                                                        unsigned int index,
                                                        enum gpiod_flags flags);
struct gpio_descs *__must_check gpiod_get_array(struct device *dev,
                                                const char *con_id,
                                                enum gpiod_flags flags);
struct gpio_descs *__must_check gpiod_get_array_optional(struct device *dev,
                                                         const char *con_id,
                                                         enum gpiod_flags flags);
void gpiod_put(struct gpio_desc *desc);
void gpiod_put_array(struct gpio_descs *descs);

struct gpio_desc *__must_check devm_gpiod_get(struct device *dev,
                                              const char *con_id,
```

Bootlin Elixir Screenshot. URL:
https://elixir.bootlin.com/linux/latest/source/include/linux/gpio/consumer.h

Casually Browsing Through The Documentation: consumer.txt

- You need to `#include <linux/gpio/consumer.h>`
- “gpiod_get() takes the **device** that will use the GPIO and the **function** the requested GPIO is supposed to fulfill”
  - `struct gpio_desc *gpiod_get(struct device *dev, const char *con_id, enum gpiod_flags flags)`
- What is the **device**?
- What is the **function**?
- “For a more detailed description of the **con_id** parameter in the **DeviceTree** case see Documentation/gpio/**board.txt**”

Quotes & info from consumer.txt. URL: https://www.kernel.org/doc/Documentation/gpio/consumer.txt

Casually Browsing Through The Documentation: board.txt

- “GPIOs can ***easily*** be mapped to devices and functions in the ***device tree***”
- Easily is an overstatement
  - Elon Musk: ‘Designing a rocket is “a piece of cake”’
- What is the device tree? We’ll see an excerpt below
- “GPIOs mappings are defined in the **consumer device's node**, in a property named <**function**>-**gpios**, where <**function**> is the function the driver will request through gpiod_get(). For example:

```
foo_device {
   compatible = "acme,foo";
   ...
   led-gpios = <&gpio 15 GPIO_ACTIVE_HIGH>, /* red */
               <&gpio 16 GPIO_ACTIVE_HIGH>, /* green */
               <&gpio 17 GPIO_ACTIVE_HIGH>; /* blue */

   power-gpios = <&gpio 1 GPIO_ACTIVE_LOW>;
};”
```

Info from board.txt. URL: https://www.kernel.org/doc/Documentation/gpio/board.txt
Elon quote: https://twitter.com/joroulette/status/1233427150675509249

Casually Browsing Through The Documentation: board.txt

‘This property will make GPIOs 15, 16 and 17 available to the driver under the "led" function, and GPIO 1 as the "power" GPIO:

```
struct gpio_desc *red, *green, *blue, *power;

red = gpiod_get_index(dev, "led", 0, GPIOD_OUT_HIGH);
green = gpiod_get_index(dev, "led", 1, GPIOD_OUT_HIGH);
blue = gpiod_get_index(dev, "led", 2, GPIOD_OUT_HIGH);

power = gpiod_get(dev, "power", GPIOD_OUT_HIGH);’
```

Then you can just user other gpiod functions to turn on and off the LEDs
Problem: How do we get **dev**? We’ll talk about that in just a moment

Info from board.txt. URL: https://www.kernel.org/doc/Documentation/gpio/board.txt

Third Argument of `gpiod_get()`: `enum gpiod_flags flags`

Below copied from consumer.txt (ref at bottom)

GPIOD_ASIS or 0 to not initialize the GPIO at all. The direction must be set later with one of the dedicated functions.
GPIOD_IN to initialize the GPIO as input.
GPIOD_OUT_LOW to initialize the GPIO as output with a value of 0.
GPIOD_OUT_HIGH to initialize the GPIO as output with a value of 1.
GPIOD_OUT_LOW_OPEN_DRAIN same as GPIOD_OUT_LOW but also enforce the line to be electrically used with open drain.
GPIOD_OUT_HIGH_OPEN_DRAIN same as GPIOD_OUT_HIGH but also enforce the line to be electrically used with open drain.

consumer.txt. URL: https://www.kernel.org/doc/Documentation/gpio/consumer.txt

Casually Browsing Through The Documentation: consumer.txt

- You need to `#include <linux/gpio/consumer.h>`
- "gpiod_get() takes the **device** that will use the GPIO and the **function** the requested GPIO is supposed to fulfill"
  - `struct gpio_desc *gpiod_get(struct device *dev, const char *con_id, enum gpiod_flags flags)`
- What is the **device**?
- What is the **function**?
- "For a more detailed description of the **con_id** parameter in the **DeviceTree** case see Documentation/gpio/**board.txt**"

Oh dev, where art thou?

Quotes & info from consumer.txt. URL: https://www.kernel.org/doc/Documentation/gpio/consumer.txt

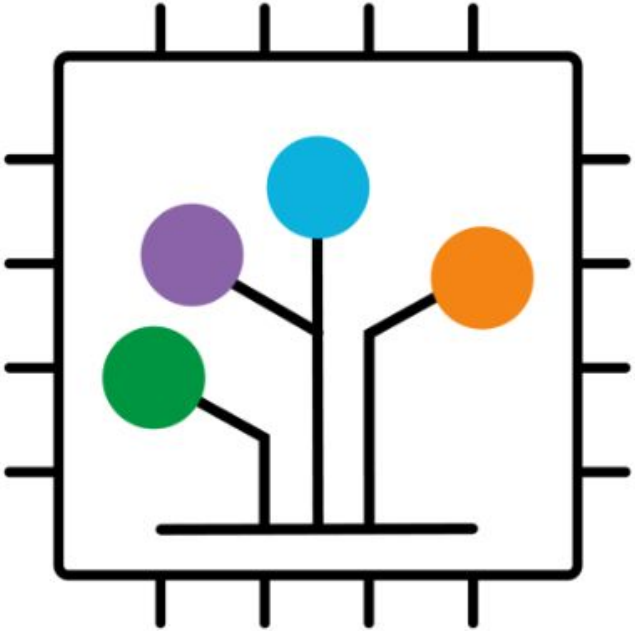
The Device Tree

"The **devicetree** is a **data structure** for **describing hardware**. Rather than hard coding every detail of a device into an operating system, many aspects of the hardware can be described in a data structure that is **passed to the operating system at boot time**."

- https://www.devicetree.org

"can now configure his platform without having to recompile the kernel" - https://docs.google.com/document/u/1/d/17P54kZkZO_-JtTjrFuVz-Cp_RMMg7GB_8W9JK9sLKfA/pub <- entertaining

Another helpful resource: https://bootlin.com/pub/conferences/2020/lee/petazzoni-dt-hw-description-everybody/petazzoni-dt-hw-description-everybody.pdf



Logo from Devicetree Specification v0.3. URL: https://github.com/devicetree-org/devicetree-specification/releases/tag/v0.3

How Do You Boot Up a Computer

- You press the power button

Image by: Penguin, Boot. Modified by Opensource.com. CC BY-SA 4.0. URL: https://opensource.com/article/18/1/analyzing-linux-boot-process

How Do You Boot Up a Computer

- ~~You press the power button~~
- Bootstrapping - hence the term boot

Image by: Penguin, Boot. Modified by Opensource.com. CC BY-SA 4.0. URL: https://opensource.com/article/18/1/analyzing-linux-boot-process

How Do You Boot Up a Computer

- ~~You press the power button~~
- Bootstrapping - hence the term boot
- We launch larger and larger software
- Eventually get to user space

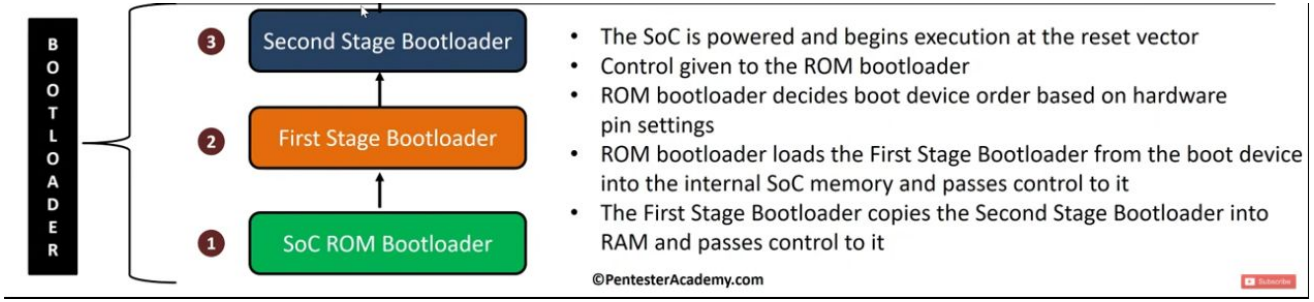
Image by: Penguin, Boot. Modified by Opensource.com. CC BY-SA 4.0. URL: https://opensource.com/article/18/1/analyzing-linux-boot-process

How Do You Boot Up a Computer

- ~~You press the power button~~
- Bootstrapping - hence the term boot
- We launch larger and larger software
- Eventually get to user space
- Back in the old days, kernel had full hardware details

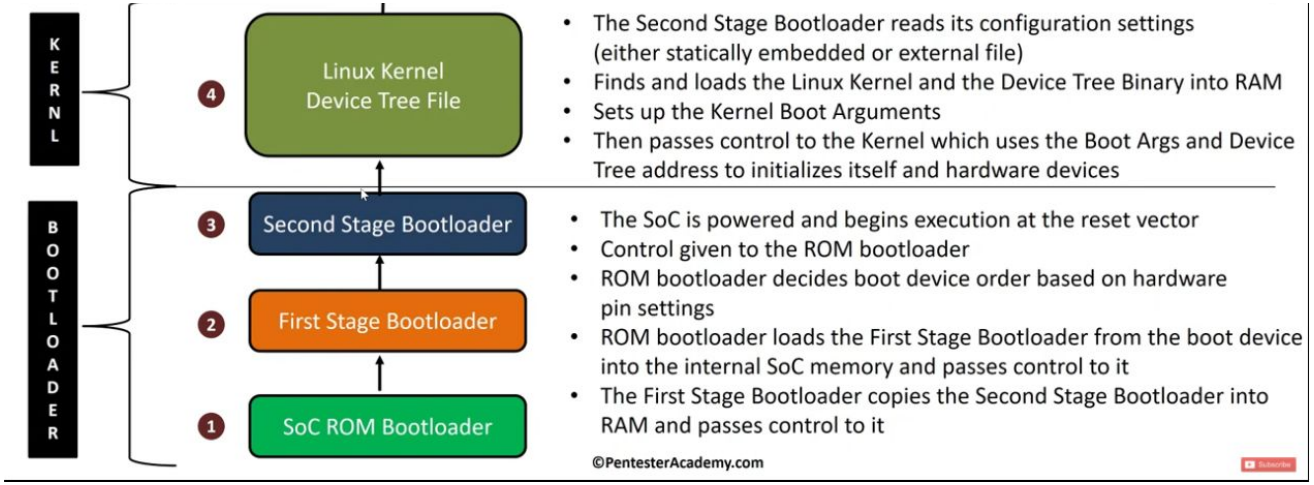
Image by: Penguin, Boot. Modified by Opensource.com. CC BY-SA 4.0. URL: https://opensource.com/article/18/1/analyzing-linux-boot-process

Boot Process (Slide From Pentester Academy TV)



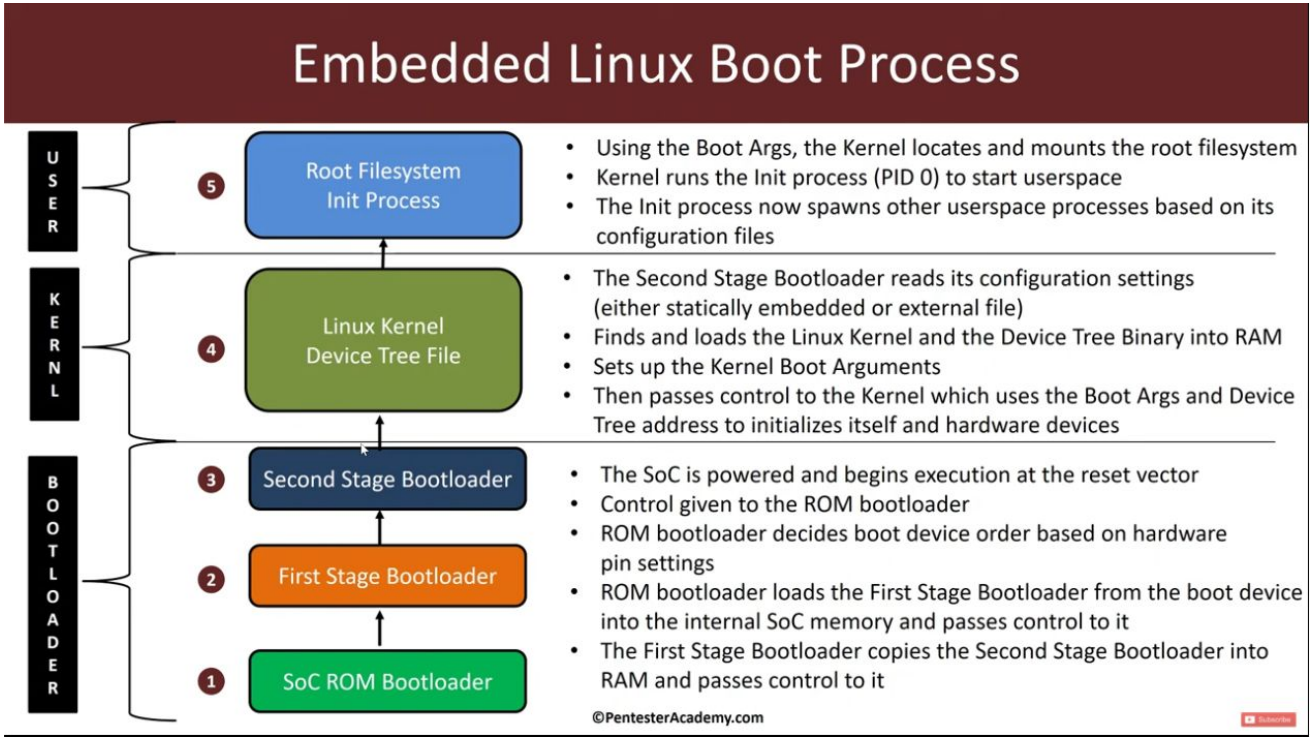
Slide from: Pentester Academy TV, “Embedded Linux Booting Process (Multi-Stage Bootloaders, Kernel, Filesystem)”. (2018) https://www.youtube.com/watch?v=DV5S_ZSdK0s

Boot Process (Slide From Pentester Academy TV)



Slide from: Pentester Academy TV, “Embedded Linux Booting Process (Multi-Stage Bootloaders, Kernel, Filesystem)”. (2018) https://www.youtube.com/watch?v=DV5S_ZSdK0s

Boot Process (Slide From Pentester Academy TV)

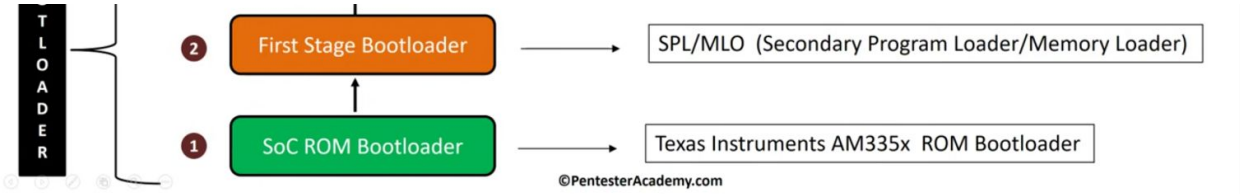


Slide from: Pentester Academy TV, “Embedded Linux Booting Process (Multi-Stage Bootloaders, Kernel, Filesystem)”. (2018) https://www.youtube.com/watch?v=DV5S_ZSdK0s

BBB Boot Process (Slide From Pentester Academy TV)

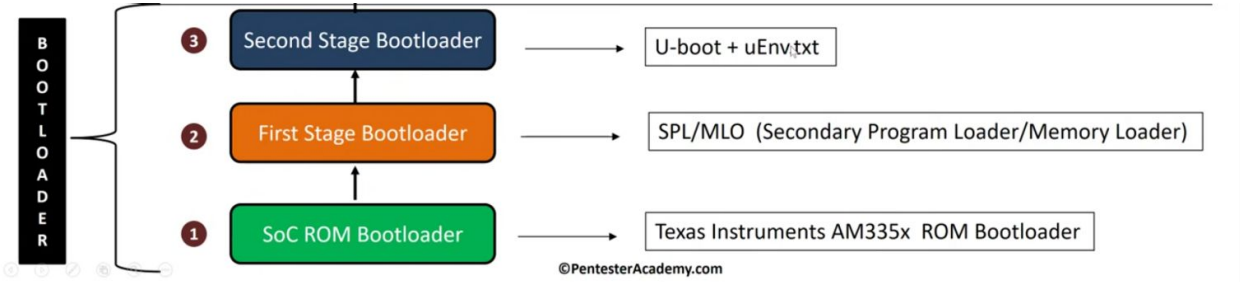
Slide from: Pentester Academy TV, “Embedded Linux Booting Process (Multi-Stage Bootloaders, Kernel, Filesystem)”. (2018)
https://www.youtube.com/watch?v=DV5S_ZSdK0s

BBB Boot Process (Slide From Pentester Academy TV)



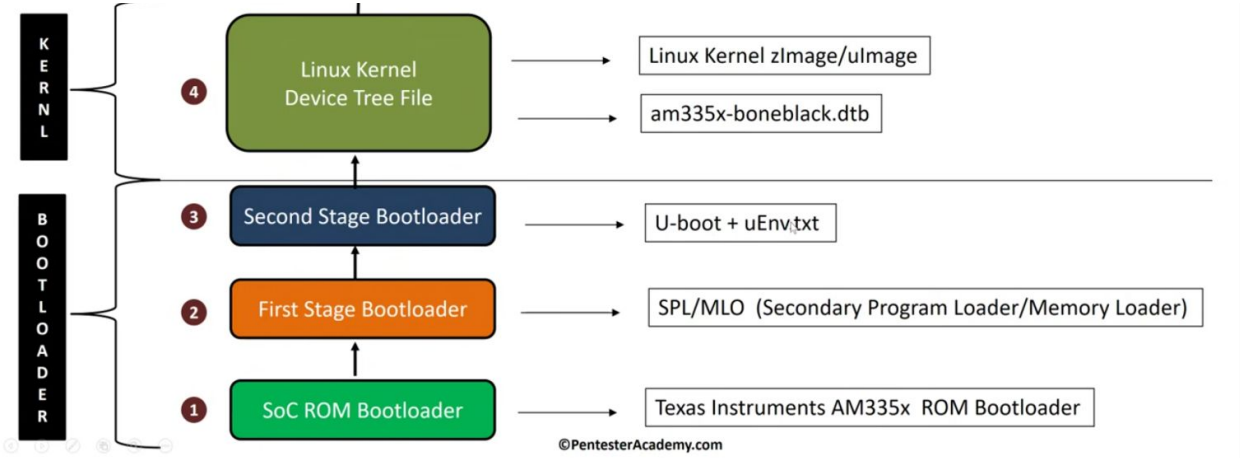
Slide from: Pentester Academy TV, “Embedded Linux Booting Process (Multi-Stage Bootloaders, Kernel, Filesystem)”. (2018)
https://www.youtube.com/watch?v=DV5S_ZSdK0s

BBB Boot Process (Slide From Pentester Academy TV)



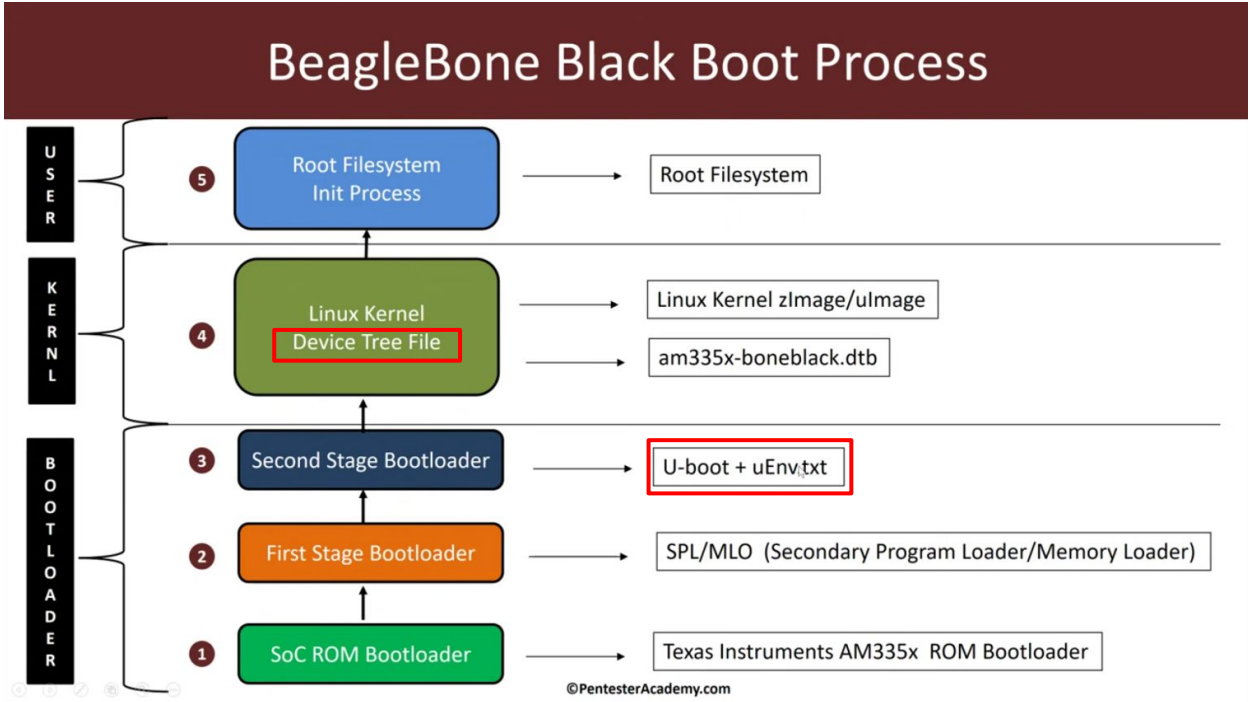
Slide from: Pentester Academy TV, “Embedded Linux Booting Process (Multi-Stage Bootloaders, Kernel, Filesystem)”. (2018)
https://www.youtube.com/watch?v=DV5S_ZSdK0s

BBB Boot Process (Slide From Pentester Academy TV)



Slide from: Pentester Academy TV, “Embedded Linux Booting Process (Multi-Stage Bootloaders, Kernel, Filesystem)”. (2018)
https://www.youtube.com/watch?v=DV5S_ZSdK0s

BBB Boot Process (Slide From Pentester Academy TV)



Slide from: Pentester Academy TV, “Embedded Linux Booting Process (Multi-Stage Bootloaders, Kernel, Filesystem)”. (2018)
https://www.youtube.com/watch?v=DV5S_ZSdK0s

Compiling The Devicetree On Your Pi

- [shouldn't need to do this: `sudo apt install device-tree-compiler -y`]
- `man dtc`
- `dtc -I fs /sys/firmware/devicetree/base`
- This will print out a lot
- Let's save it to a file
- `dtc -I fs /sys/firmware/devicetree/base > ~/device_tree.txt`
- Let's copy it back to our computer for easy viewing
- `scp pi@raspberrypi.local:~/device_tree.txt .`
- Look at a few things in text editor

Devicetree Syntax

Node name

Unit address

Property name

Property value

Properties of node@0

```
/ {
    node@0 {
        a-string-property = "A string";
        a-string-list-property = "first string", "second string";
        a-byte-data-property = [0x01 0x23 0x34 0x56];

        child-node@0 {
            first-child-property;
            second-child-property = <1>;
            a-reference-to-something = <&node1>;
        };

        child-node@1 {
        };
    };

    node1: node@1 {
        an-empty-property;
        a-cell-property = <1 2 3 4>;

        child-node@0 {
        };
    };
};
```

Bytestring

A phandle
(reference to another node)

Label

Four cells (32 bits values)

Free Electrons. Kernel, drivers and embedded Linux development, consulting, training and support. http://free-electrons.com 10/47

Screenshot of slide from: Thomas Petazzoni, “Device Tree For Dummies”. URL: https://elinux.org/images/f/f9/Petazzoni-device-tree-dummies_0.pdf

Observations At Top of File - Pi

```
/dts-v1/;

/ {
    compatible = "raspberrypi,model-zero-w\0brcm,bcm2835";
    serial-number = "00000000064e4bf5";
    model = "Raspberry Pi Zero W Rev 1.1";
    memreserve = <0x1c000000 0x4000000>;
    interrupt-parent = <0x01>;
    #address-cells = <0x01>;
    #size-cells = <0x01>;

    reserved-memory {
        ranges;
        #address-cells = <0x01>;
        #size-cells = <0x01>;
        phandle = <0x2f>;
    ...
```

Observations At Top of File - BeagleBone Black

```
/dts-v1/;

/ {
    compatible = "ti,am335x-bone-black\0ti,am335x-bone\0ti,am33xx";
    serial-number = "2125SBB05081";
    model = "TI AM335x BeagleBone Black";
    interrupt-parent = < 0x01 >;
    #address-cells = < 0x01 >;
    #size-cells = < 0x01 >;

    clk_mcasp0_fixed {
      compatible = "fixed-clock";
      #clock-cells = < 0x00 >;
      phandle = < 0x2ca >;
      clock-frequency = < 0x1770000 >;
    };

    ...
```

Observations Later In File - Pi

```
leds {
    compatible = "gpio-leds";
    phandle = <0x89>;

    led-act {
        gpios = <0x07 0x2f 0x01>;
        label = "ACT";
        phandle = <0x2e>;
        default-state = "off";
        linux,default-trigger = "actpwr";
    };
};
```

Observations Later In File - BeagleBone Black

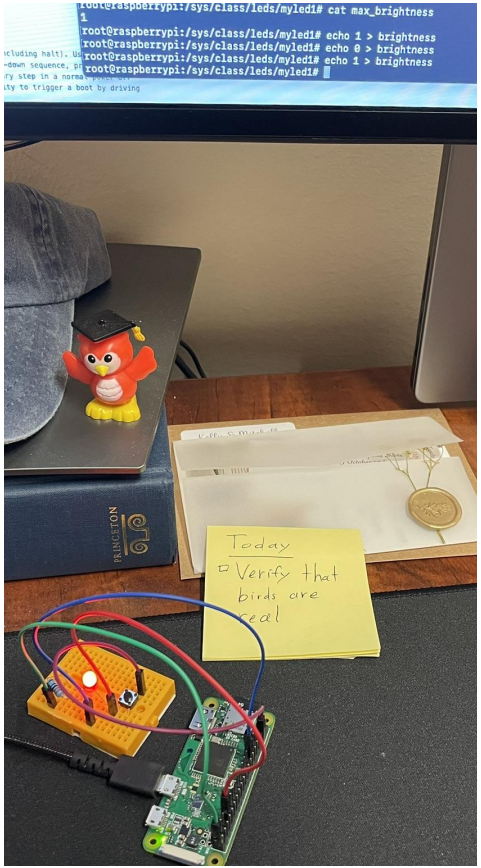
```
leds {
    compatible = "gpio-leds";
    pinctrl-0 = < 0x20c >;
    pinctrl-names = "default";

      ...

    led4 {
         gpios = < 0x58 0x17 0x00 >;
         label = "beaglebone:green:usr2";
         default-state = "off";
         linux,default-trigger = "cpu0";
     };
```

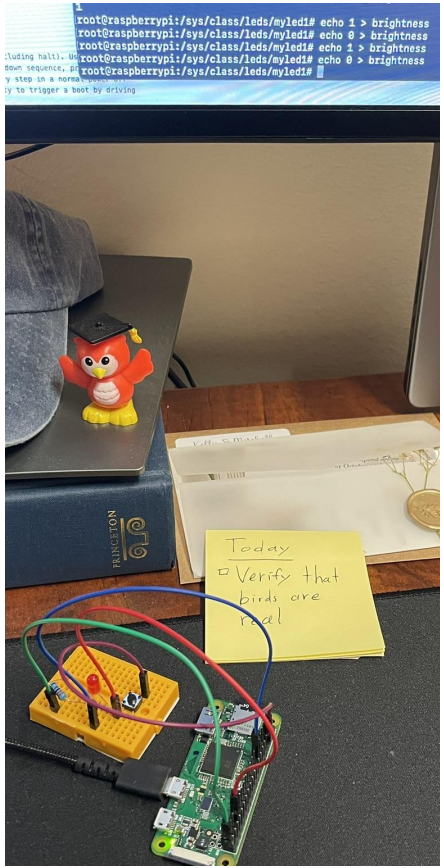
Project 2 Goal: Modify Devicetree to Use `gpiod` To Toggle GPIO

- Devicetree (DT) overlays
- Just like C files, we can have multiple files for the DT
- **Overlaying**: Tree of file with includes will overlay included file trees

Pi Overlay Fun

1. Go to /boot/config.txt
2. https://github.com/raspberrypi/firmware/tree/master/boot/overlays
3. Don’t do the following subpoints
   a. *Uncomment #dtoverlay=gpio-ir,gpio_pin=17*
   b. *Reboot, try using default Johannes4Linux code*
   c. *Shouldn’t allow you to get gpio17*
4. sudo apt install gpiod
5. gpioinfo
6. Let’s try to get gpio17 now
7. Back up any assignment code - this can break Pi’s!
8. Edit boot file: dtoverlay=gpio-led,gpio=17 [reboot after]
9. gpioinfo, then sudo su -, then ls /sys/class/leds
10. cd /sys/class/leds/myled1
11. Change circuit so gpio17 is connected to LED
12. echo 1 > brightness
13. echo 0 > brightness





“[BeagleBone Black] How to set gpio pin when booting”. Digi-Key TechForum. https://forum.digikey.com/t/beaglebone-black-how-to-set-gpio-pin-when-booting/6588/3