SMILIES
reSilient coMputer archItectures
and LIfE Sciences

Politecnico
di Torino

Department of Control and
Computer Engineering

DAUIN

CUSTOM LINUX SYSTEMS

STEFANO DI CARLO

WHAT IS CUSTOM EMBEDDED LINUX ?

- A **tailored** Linux OS specifically designed for embedded systems
- Optimized for efficiency and built for **purpose**

- Key Characteristics
  - **Optimized Footprint**
    (Minimal resource usage)
  - **Hardware-Specific**
    (Built for the target hardware architecture like ARM)
  - **Feature-Centric**
    (Contains only the drivers, libraries, and apps needed)
  - **Secure and Stable**
    (Designed to enhance security and ensure reliability)

REAL-WORLD EXAMPLES

- IoT (Internet of Things)
- Automotive
- Industrial Automation
- Consumer Electronics
- Healthcare

- Why Use Linux for These Devices?
  - **Cost Efficiency**: Open-source nature lowers development costs
  - **Customizability**: Tailored for each use case.
  - **Hardware Optimization**: Ensures efficient use of system resources.
  - **Security**: Configurable to meet industry-specific compliance standards.

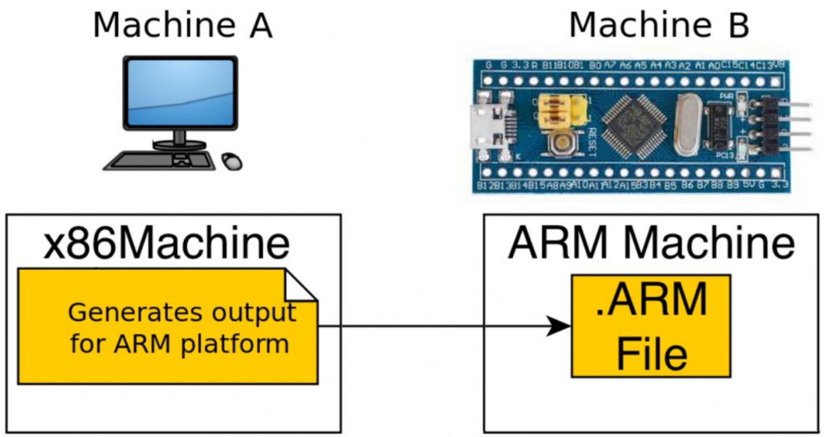
KEY ELEMENTS OF EMBEDDED LINUX SYSTEMS

- An embedded Linux system is built from several essential components, each playing a critical role in **booting**, **running**, and enabling the specific functionality of the device.

- The 4 Core Elements of Embedded Linux
  - Toolchain
  - bootloader
  - kernel
  - Root files
- one more element can be domain specific application

TOOLCHAIN

- A **toolchain** is a set of tools that allows developers to build, debug, and deploy software for embedded Linux systems. It is essential for creating binaries that run on the target device.



- Key Components of a Toolchain
  - Cross-Compiler

    Converts source code into executable binaries for the target hardware architecture (e.g., ARM, MIPS, x86).
  - Linker
  - Assembler
  - Libraries
  - Debugger
  - Build System

    Automates the process of compiling and linking.

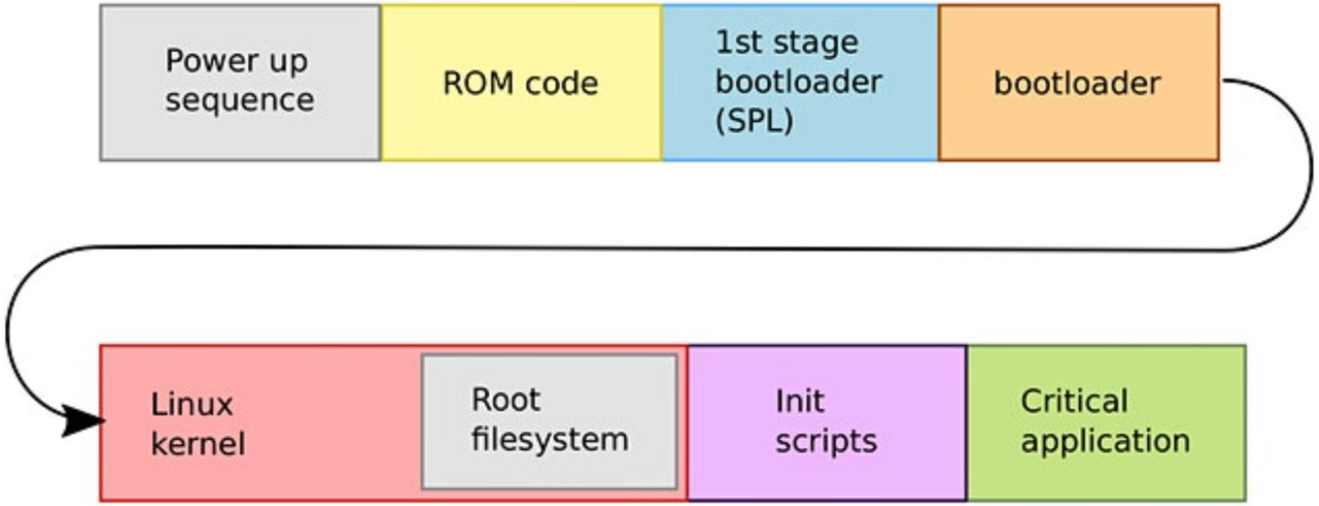
BOOTLOADER

- A **bootloader** is a small program that runs before the operating system kernel and is responsible for initializing hardware and loading the kernel into memory.
- Key functions
  - Hardware Initialization
  - Kernel Loading
  - System Configuration
  - Optional Features

- Examples of Common Bootloaders
  - U-Boot (Universal Bootloader)
    - Most widely used in embedded Linux.
    - Highly customizable and supports a wide range of architectures.
  - GRUB (GRand Unified Bootloader)
    - More common in desktops but supports embedded use cases.
    - Advanced configuration options.
  - Barebox
    - Lightweight and fast bootloader for embedded systems.

BOOTLOADER (CONT.)

- Bootloader Phases
  - Stage 1 (Primary Bootloader)
    - Minimal functionality to initialize hardware and load the secondary bootloader.
  - Stage 2 (Secondary Bootloader)
    - Loads and executes the Linux kernel with a full set of configurations.



KERNEL

- The **kernel** is the heart of an embedded Linux system, acting as the **bridge** between the hardware and software. It manages resources and ensures that the system runs smoothly and efficiently.
  - System Resource Management
  - Hardware Abstraction
  - Process Scheduling

- Key Components of a kernel
  - Device Drivers

    Drivers for various hardware components (e.g., sensors, displays, network interfaces).Linker
  - Networking Stack
  - File System Support
  - Customizability

    Non-essential features and drivers can be removed to optimize size and performance.

ROOT FILESYSTEM

- Directory structure that contains all the files, libraries, binaries, and configuration files needed for the system to function.
- Typically mounted at the root (/) of the device and is the foundation for the Linux operating system.
- contains the files necessary for booting the system and providing user-level functions.

- Key Components of Root Filesystem
  - Libraries
  - Binaries
  - Configuration Files
- The root filesystem can be customized to include only the **necessary components** for the embedded device, optimizing **memory** and **storage** usage.

DOMAIN-SPECIFIC APPLICATIONS (OPTIONAL ELEMENT)

- Applications designed and developed for the specific function or industry of an embedded device.
- Purpose
  - Implement Device Functionality
  - Optimized for Performance

- Key Components
  - Tailored to Specific Needs
    - highly customized to suit the exact requirements of the industry, ensuring optimal performance and reliability.
  - Integration with Kernel
    - integrated directly with kernel-level features, such as device drivers and network stacks.

EMBEDDED LINUX BUILD TOOLS

- Building a custom embedded Linux system requires robust tools that **simplify** the process of **configuring**, **building**, and **deploying** the system.
- Buildroot is a popular tool in this space, offering simplicity and flexibility for embedded Linux development.

- Key features
  - Minimalistic Approach
  - Fast Build Times
  - Cross-Compilation Support
  - Extensive Configuration Options

HOW BUILDROOT WORKS

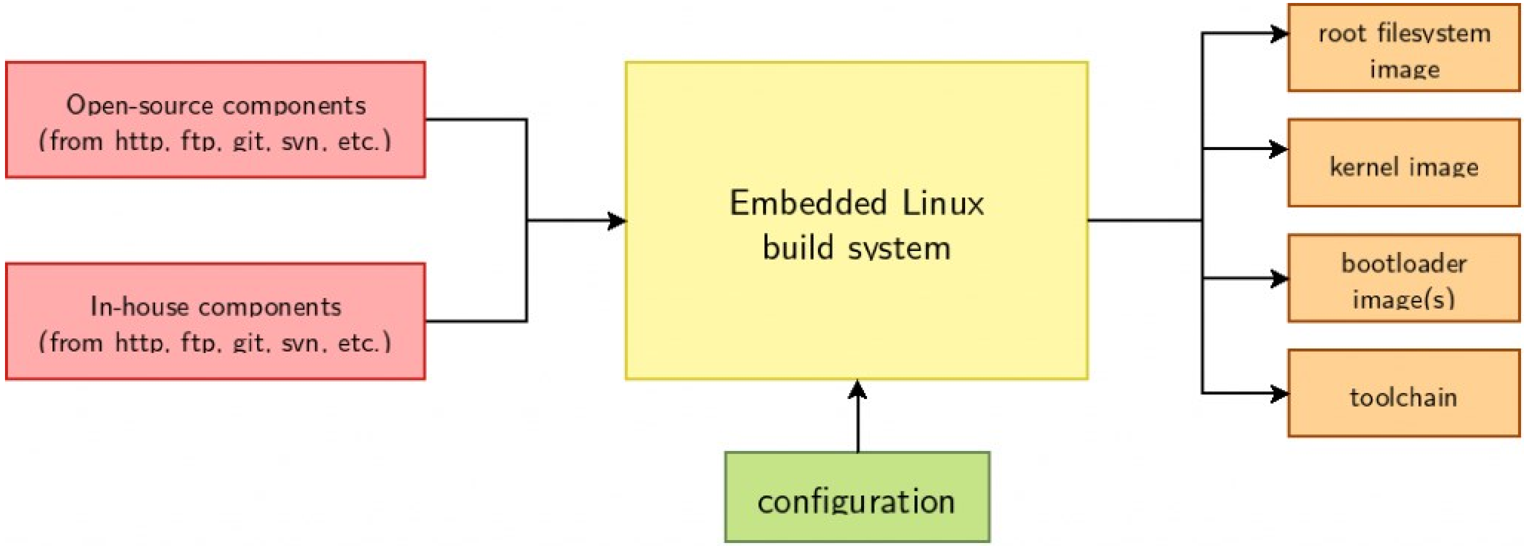
- Buildroot is a set of **Makefiles** designed to simplify the process of building custom Linux-based embedded systems. It automates the downloading, configuration, compilation, and **integration** of software packages into a functional system image.

- Each directory in Buildroot contains at least two files:
  - **something.mk:** The Makefile that downloads, configures, compiles, and installs a package.
  - **Config.in:** Describes configuration options for the package.

- Key Components/Directories of Buildroot
  - **Toolchain:** Makefiles for building the cross-compilation toolchain
  - **Arch:** definitions for processor architectures supported by Buildroot (e.g., ARM)
  - **Package:** Makefiles for user-space tools and libraries to be compiled and added to the target root filesystem
  - **Boot:** Manages the Makefiles and files for the supported bootloaders (e.g., U-Boot)
  - **System:** Provides support for system integration (e.g., target filesystem skeleton, init system)
  - **Fs:** files related to the generation of the target root filesystem image

BUILDROOT WORKFLOW

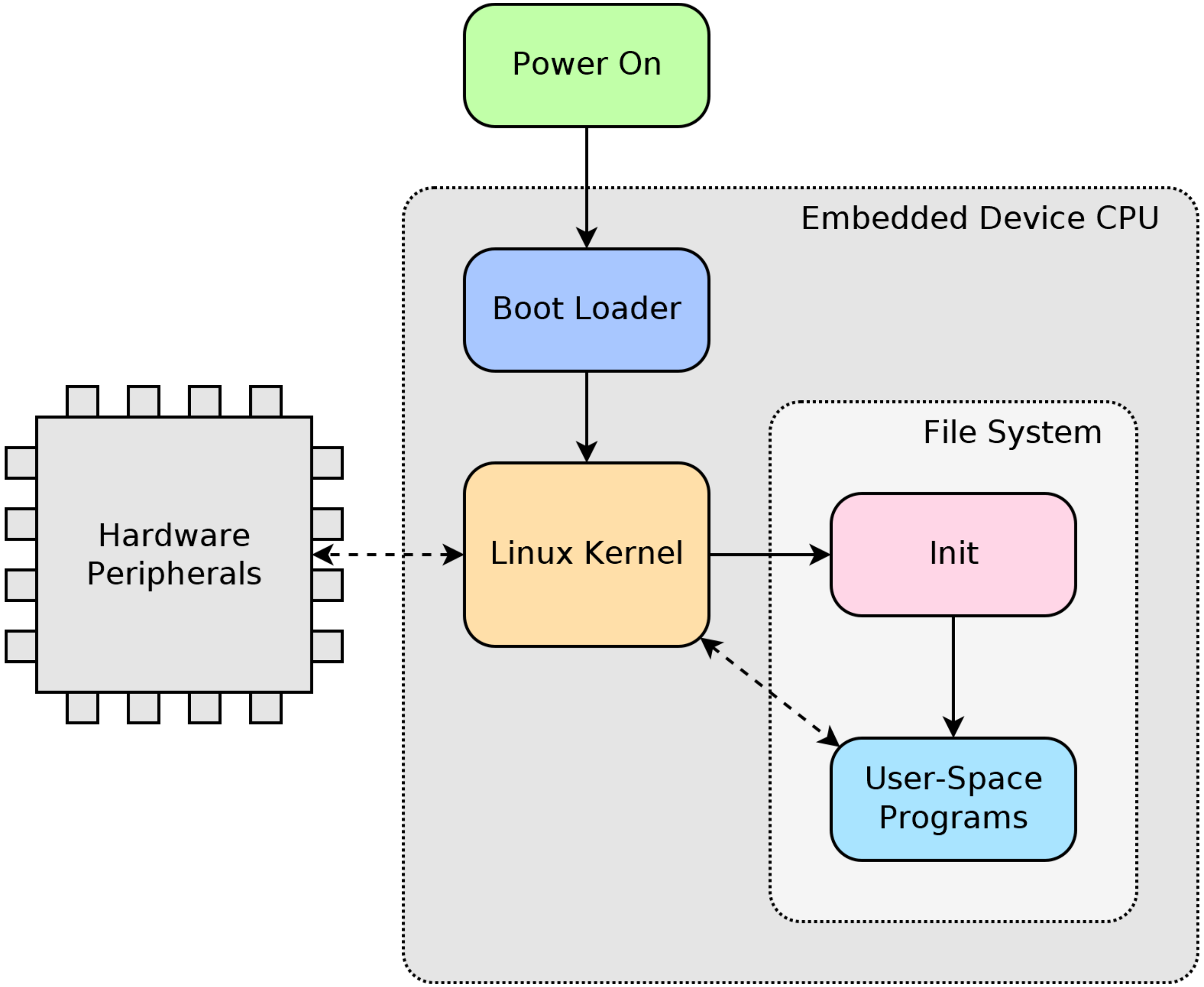
- **Input Data:** Configuration files and package selections.
- **Output Product:** A fully built Linux-based embedded system, including the Linux kernel, root filesystem, bootloader, and device tree, ready for deployment on the target hardware.



Open-source components (from http, ftp, git, svn, etc.)
In-house components (from http, ftp, git, svn, etc.)
Embedded Linux build system
configuration
root filesystem image
kernel image
bootloader image(s)
toolchain

GOAL

- A fully functional, self-contained embedded Linux operating system
- Running domain-specific applications
- Including all essential components required for the device to operate effectively.



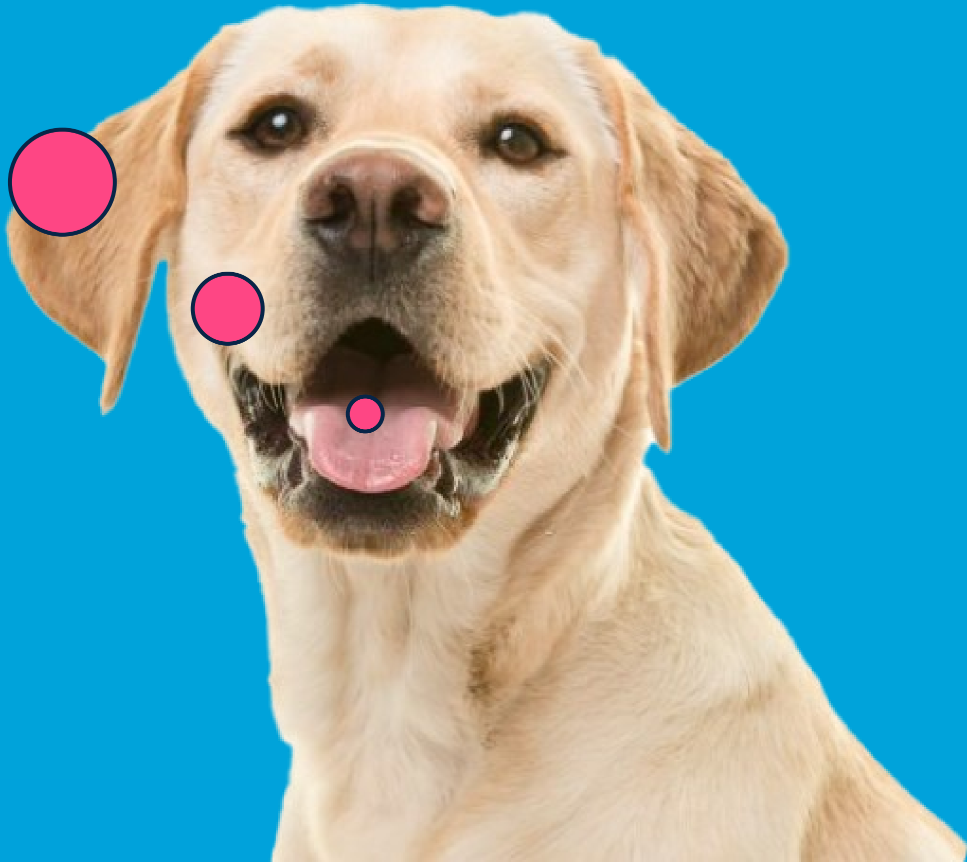
smilies
reSilient coMputer archItectures
and LIfE Sciences

Politecnico
di Torino

Department of Control and
Computer Engineering

DAUIN

QUESTIONS?



THANK YOU!