EMBEDDED SYSTEMS
2020/2021

TARGETS

HARDWARE SCHEMATIC DESCRIPTION

- Our target hardware can be described hierarchically as:
  - A board physically carrying the main system parts that include:
    - A system on a Chip (SoC)
    - On-board devices
      - Memory
      - Storage
      - I/O peripherals
      - Wired and wireless networking
    - I/O Connectors
    - Power electronics
    - In-system programming and debugging circuitry

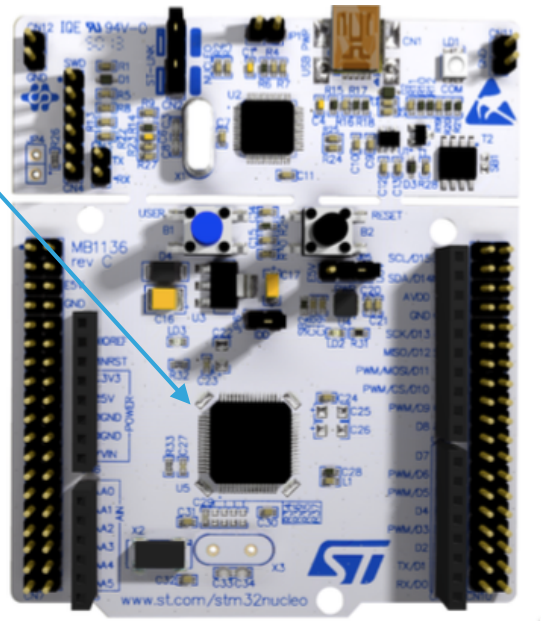
Raspberry Pi 3 B+

Nucleo-64 STM32F446

HARDWARE SCHEMATIC DESCRIPTION

- A system on a Chip (SoC) contains
  - One single-core or multi-core CPU
  - I/O peripherals
- It may also optionally integrate
  - Specialized processors
    - DSPs
    - CODECs
    - GPUs
  - Memory
  - Storage

Raspberry Pi 3 B+

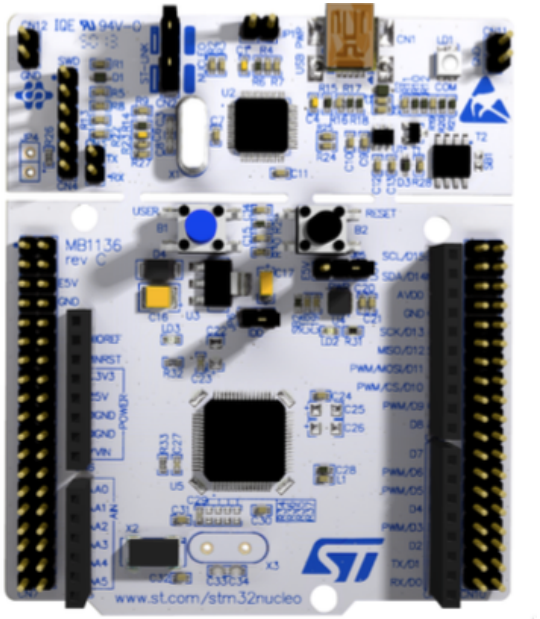


Nucleo-64 STM32F446

HARDWARE SYSTEMS

- Our targets are two embedded systems
  - A general-purpose Single-Board Computer (SBCs)
    - General Purpose embedded CPU (ARM Cortex A - 64/32 bits)
    - Desktop/Mobile class I/O (audio/video subsystems with hardware codecs and display/camera interfaces)
    - Wired and Wireless networking
    - Abundant RAM and storage (through SD-cards)
    - Few Embedded I/O peripherals and ports
  - A microcontroller based system (STM32F446)
    - MCU (ARM Cortex M - 32 bits)
    - Small RAM and storage
    - Many Embedded I/O peripherals and ports

Raspberry Pi 3 B+



Nucleo-64 STM32F446

HOW DO WE DEVELOP FOR OUR TARGETS?

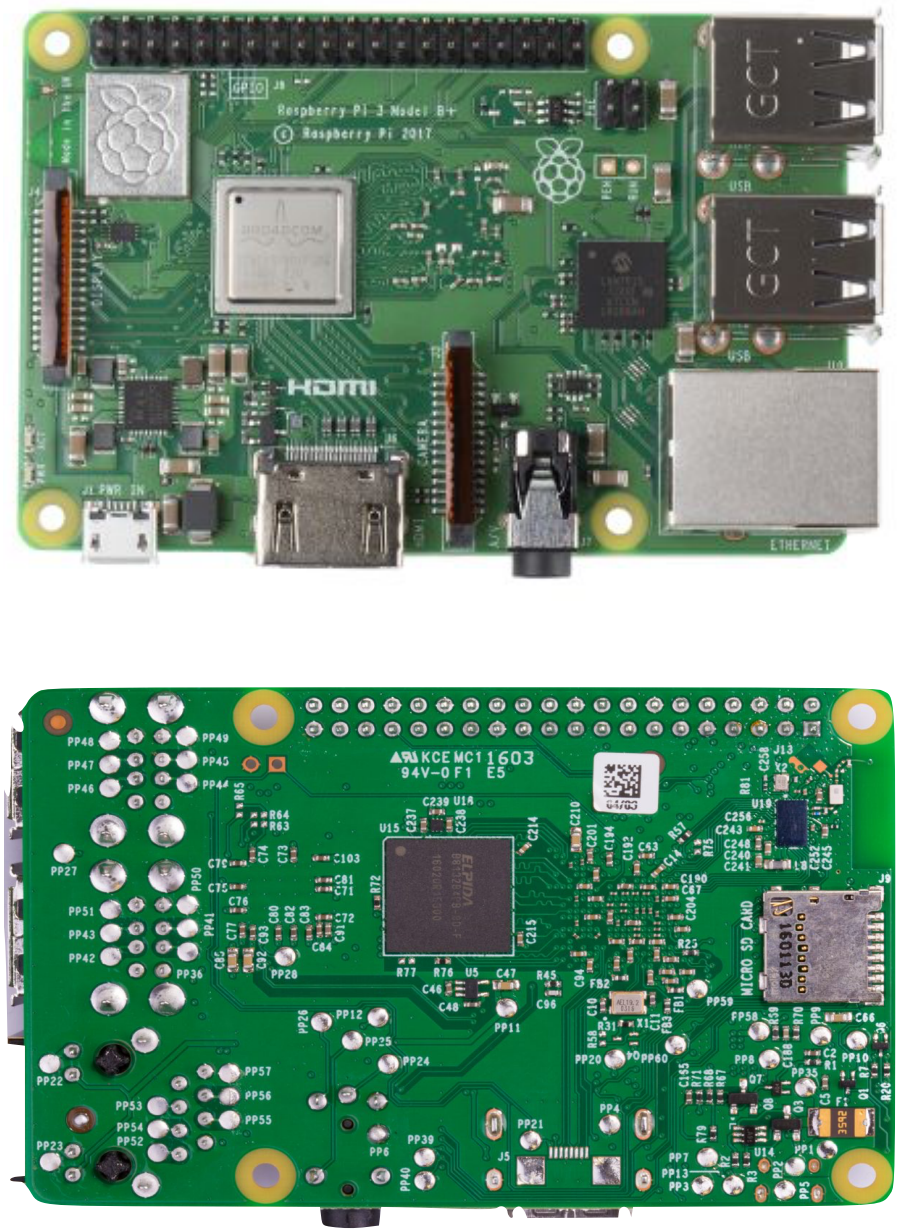
- Three models
  - Native compilation
  - Cross-compilation
  - Interactive programming
- As products we can have both:
  - High-level applications abstracted from the hardware through software layers such as OSs, Hardware Abstraction Layers (HALs), and runtimes
  - Low-level, hardware-tied system code (baremetal)

Raspberry Pi 3 B+

Nucleo-64 STM32F446

HOW DO WE DEVELOP FOR OUR TARGETS?

- Native compilation
  - The most powerful systems can host a full server/desktop OS and a development environment including editors, compilers and debuggers

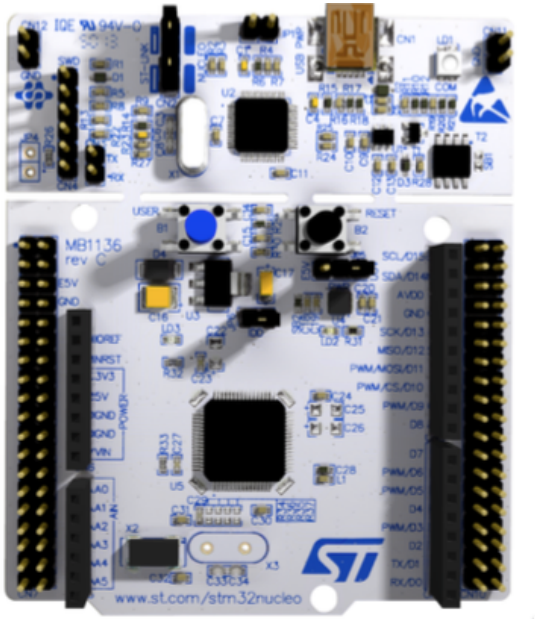


Raspberry Pi 3 B+

HOW DO WE DEVELOP FOR OUR TARGETS?

- Cross compilation
  - A software toolchain (e.g. GCC) is used on a development machine to produce code for the target machine
  - The executable is loaded on the target machine and runs there. OSs are often used but are not mandatory (bare metal)
  - Emulators can be used on the development machine to run the target code

Raspberry Pi 3 B+

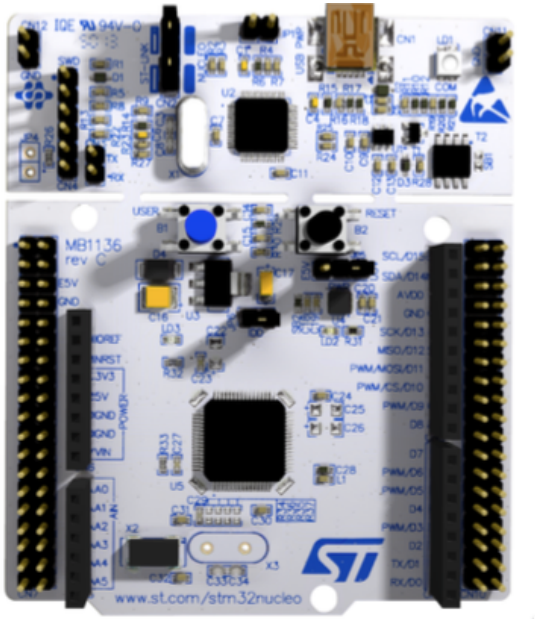


Nucleo-64 STM32F446

HOW DO WE DEVELOP FOR OUR TARGETS?

- Interactive environment on the target machine
  - An interpreter is executed on the target machine: code can be easily tested on the target, experimental programming is feasible
  - Depending on the complexity of the interpreter (e.g. Python), this model may not be suitable for resource-constrained machines
  - Forth interpreters can be implemented easily for resource-constrained machines using no OS (**bare metal**)

Raspberry Pi 3 B+



Nucleo-64 STM32F446