ECE 471 – Embedded Systems
Lecture 4

Vince Weaver

`https://web.eece.maine.edu/~vweaver`

`vincent.weaver@maine.edu`

10 September 2025

Announcements

- Any questions on HW#1?
- HW#2 will be posted Friday
- Let me know if you don't have a Raspberry Pi yet

Raspberry Pi

Note there are two separate (but related) organizations:

- Raspberry Pi Foundation – charitable group to encourage computer science education
- Raspberry Pi, Ltd – company that makes and sells Raspberry Pi Boards
- Their goals don't always line up
- During parts shortage extremely limited board availability, Ltd was prioritizing businesses over education institutions

What is a Raspberry Pi?

- Raspberry Pi Foundation wanted small board to encourage CS in schools
- Easy to use and cheap enough that students can experiment without worrying too much about bricking it
- Back in the day small micro-computers encouraged hacking, modern Windows systems not so much

Why use a Raspberry Pi?

- There are other small embedded boards (Beaglebone, etc.) but Pi has many nice features
  - high performance (especially pi4/pi5)
  - low cost (relatively, with less RAM)
  - using Linux so no software-lock-in (STM hal?)
  - relatively well documented (but still not great)
  - available software/support (this is big!)
    Other ARM boards give kernel blob with no support and quickly gets out of date / no commits upstream

Raspberry Pi Models

- Model Names originally from BBC Micro
- Up through pi4 all have more or less same SoC. VideoCore IV GPU runs show (VI pi4, VII pi5)
- First released in 2012
- They like to release new models just after I've bought the older models for my cluster

BCM2835/BCM2708 – ARM1176 (ARMv6)

- Single core, slow ethernet
- **Model 1B** – 700MHz, 512MB RAM, SD, USB hub+USB Ethernet
- **Model 1B+** – like B but micro-SD, composite video-out inside of audio jack, 4 USB ports, longer GPIO header, re-arranged outputs, more mounting holes, fewer LEDs, lower power
- **Model 1A / Model 1A+** – less RAM

(256MB/512MB), no Ethernet, no USB hub, cheaper, less power

- **Zero** – 1GHz, 512MB, smaller, cheaper, $5
- **Zero W** – 1GHz, has wireless, $10
- **Compute Node** – like B but on SO-DIMM backplane, eMMC

BCM2836/BCM2709 – ARM Cortex A7 (ARMv7)

- **Model 2B (original)** – like 1B+ but with 1GB RAM, 900MHz Quad-core Cortex A7

BCM2837/BCM2710 – ARM Cortex A53 (ARMv8)

- **Model 3B** – 4-core 64-bit, 1.2GHz, wireless Ethernet, bluetooth (crash on OpenBLAS Linpack)
- **Model 2B (v1.2)** – update with Cortex A53
- **Model 3B+** – better thermal, faster Ethernet (1GB but maxes at 300MB), power over Ethernet header. Still only 1GB RAM
- **Model3 A+**, **Compute 3**

BCM2711 – ARM Cortex A72 (ARMv8)

- **Model 4B**
- 1.5GHz, Videocore VI at 500MHz
- USB-C power connector
- 1, 2, 4 or 8GB RAM
- USB3, microHDMI*2
- PCIe if you de-solder USB chip
- Real gigabit Ethernet
- GPIO header has more i2c/spi etc options
- **pi400**: built into keyboard (4GB 1.8GHz)

BCM2712 – ARM Cortex A76 (ARMv8.2)

- **Model 5**: 2GB / 4GB / 8GB / 16GB(?) RAM
- Power button!
- Videocore VII
- USB-C power (wants 5V at 5A if possible)
- Official PCIe support
- Drop headphone jack (composite video via header)
- Move peripherals to separate chip built with older process
- Real time clock (no battery by default)
- PIO (programmable I/O), on-board Cortex M3?

- **pi500** built into keyboard, 8GB

Pi Pico - RP2040

- Pi Pico
  - Can't run Linux
  - Completely new design, custom SoC
  - 133MHz Dual core ARM-cortex M0+
  - 264k SRAM / 2MB Flash / $4
- Pi Pico2
  - ARM Cortex M33 and RISC-V processors
  - 520k RAM, optional wifi/bluetooth

Software/Programming the Pi

- Many, many options
- Can even write your own on bare metal (see ECE531)
- We'll use C on Linux

Why Linux?

- Open source
- Free (no cost, but also freedom)
- Widely used for ARM-based embedded systems
- I like Linux.

Brief Linux History

- UNIX: OS by bell labs from 1970s
  Spread widely because AT&T couldn't due to antitrust
- BSD (Berkeley) version made in California
- Linus Torvalds (from Finland) gets a 386
- No free UNIX? FreeBSD caught up in AT&T lawsuit
- Linus announces his custom OS in 1991
- Open-source development by many all over the world
- Don't be afraid of Linus (or open-source projects in general)

The media over-hypes how angry some developers get.

Why C?

- System Programming Language (OS/embedded)
- Portability (sort of) (i.e. how big is an `int`)
- Higher than assembly (barely)

  Pearce: “all the power of assembly with all the ease-of-use of assembly”

Why C over Java or C++?

- They can hide what is actually going on.
- C statements map more or less directly to assembly.
- With things like operator overload, templates, exceptions, garbage collection, extra layers are added.
- This can matter for both size, speed, determinism, and real time.
- On embedded might be restricted to a C++ subset

Why C over Python?

- Python is interpreted
- Mostly speed. (although you can JIT)
- Also if accessing low level hardware, in general you are calling libraries from python that are written in C anyway.

Why C over Rust? (or Go or Zig)

- Don't overlook momentum of an old platform, sample code, libraries, etc.
- Rust still a moving target, needs to settle down a bit
- Maybe in the future

Downsides of C? – Undefined Behavior

- Compiler is allowed to do anything it wants (including dropping code) if it encounters something undefined by the standard.
- This can be something as simple as just overflowing a signed integer or shifting by more than 32.
- People joke of “nasal demons” i.e. standard says anything can happen here, even demons flying out of your nose

Downsides of C? – Too Much Trust

- “Enough rope to shoot yourself in the foot”.
- C gives a lot of power, especially with pointers.
- It assumes you know what you are doing though.
- With great power comes great responsibility.

Downsides of C – Security

- Biggest issue is memory handling (lack thereof)
- Buffer overflows

```
int a[5];
a[0]=1;             // fine
a[10000000]=1;      // obviously bad
a[5]=1;             // subtly bad
```

- How can that go wrong? Crash? Corrupt Memory? Wrong results? Total system compromise?
- Overwriting stack can be bad, as return address there

- Especially if user input going into the variable

Downsides of C – Security

- Why can't the compiler stop you? Maybe it can in above example.
- What if the offset read from user input instead?

```
scanf("%d",&i);
a[i]=1;
```

- Could still maybe detect this, but would need to add extra code which might be slow?
- Problem is arrays and pointers are same in C

Downsides of C – Pointers

- Wouldn't we be better off w/o pointers?
- Philosophy of safer languages (can even be faster! Can avoid aliasing pessimizations!)
- The actual processor does pointers a lot though
- We are close to hardware
- In embedded we might need to poke to exact memory accesses (MMIO and such)

Program in C (song/video)

- `https://www.youtube.com/watch?v=tas0O586t80`