Embedded Systems and Microcontrollers

What is an embedded system?

Definition

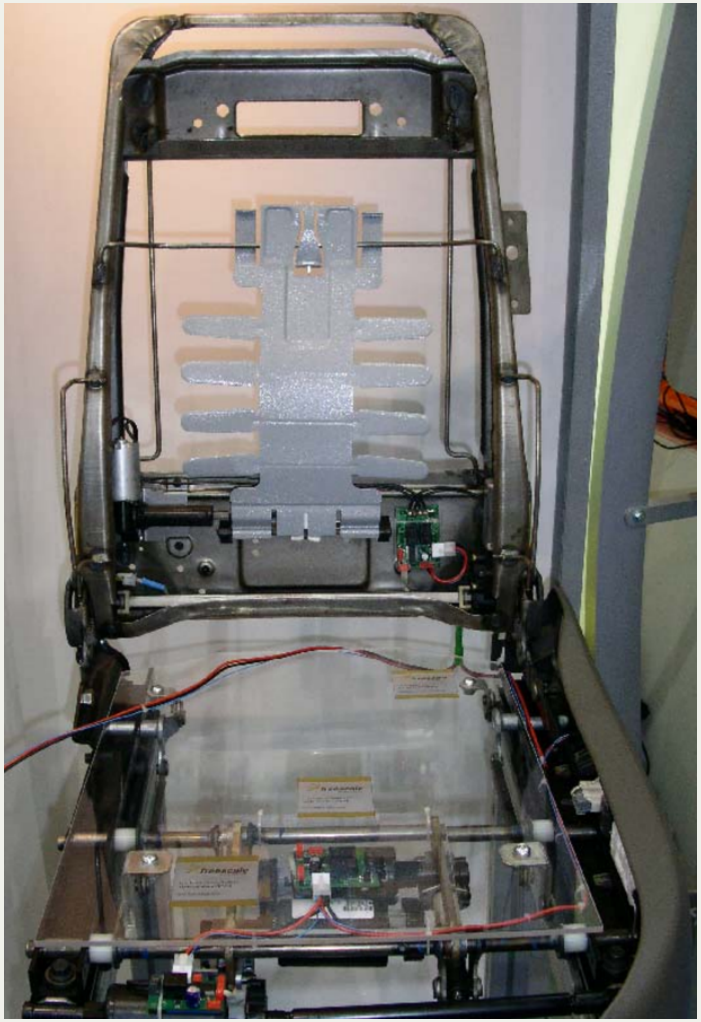
An **embedded system** is a computer **system** with a dedicated function within a larger mechanical or electrical **system**, often with real-time computing constraints. It is **embedded** as part of a complete device often including hardware and mechanical parts. **Embedded systems** control many devices in common use today.

(https://en.wikipedia.org/wiki/Embedded_system)

Embedded systems control many devices in common use today...

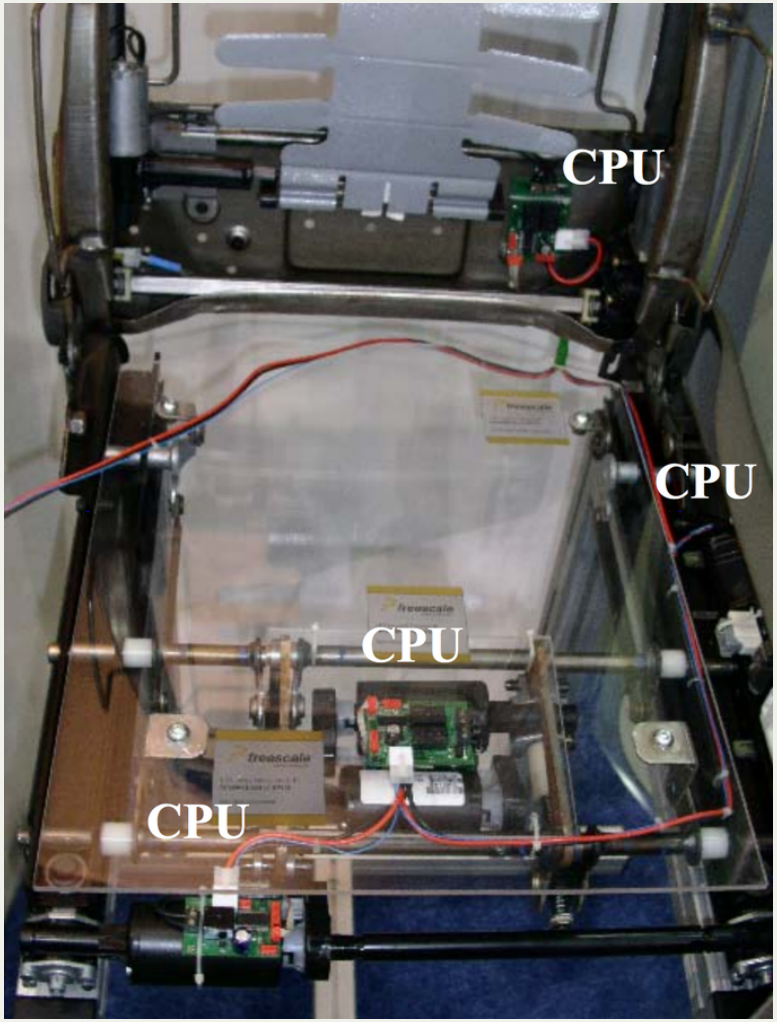
How many CPU's in this picture?

Car seat from a
2004 trade show



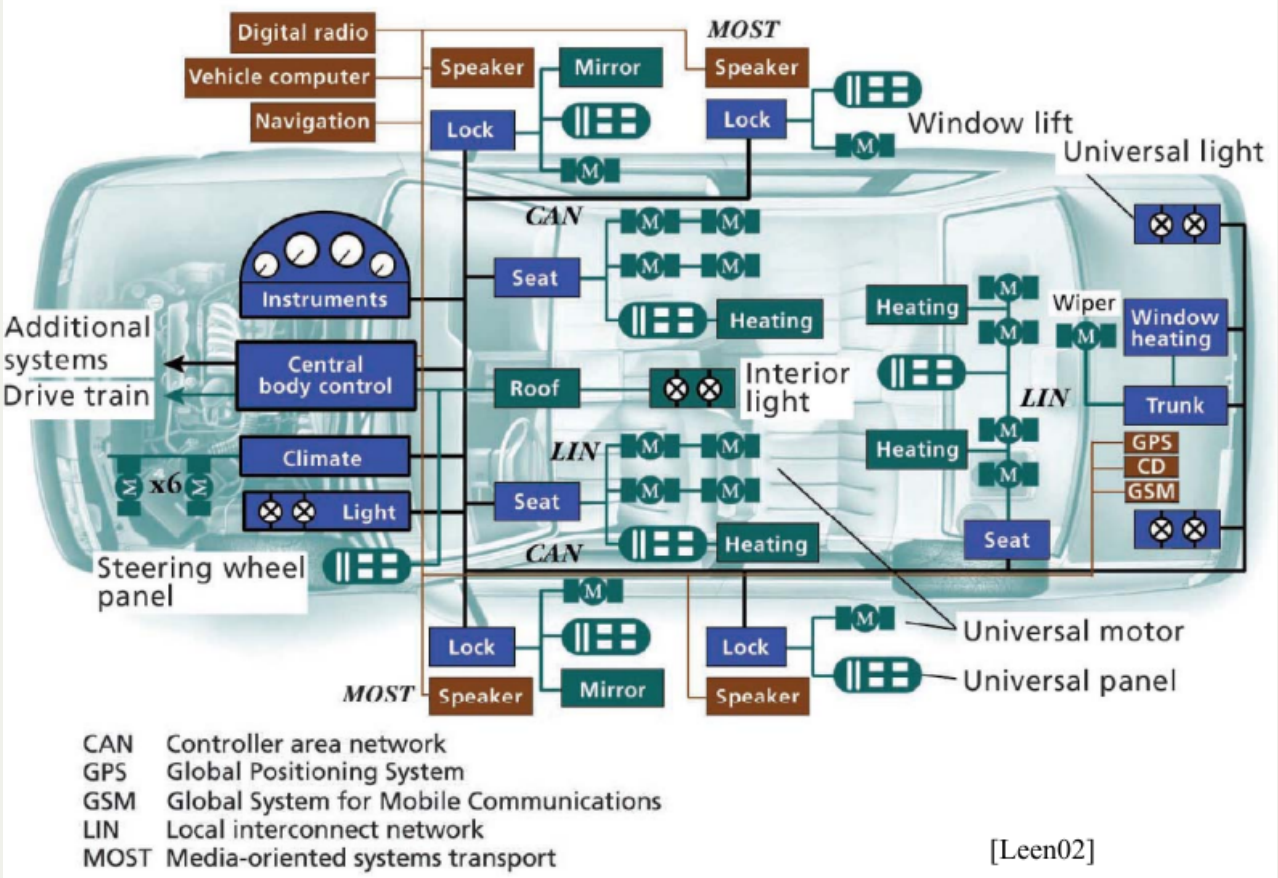
How many CPU's in this picture?

Car seat from a 2004 trade show



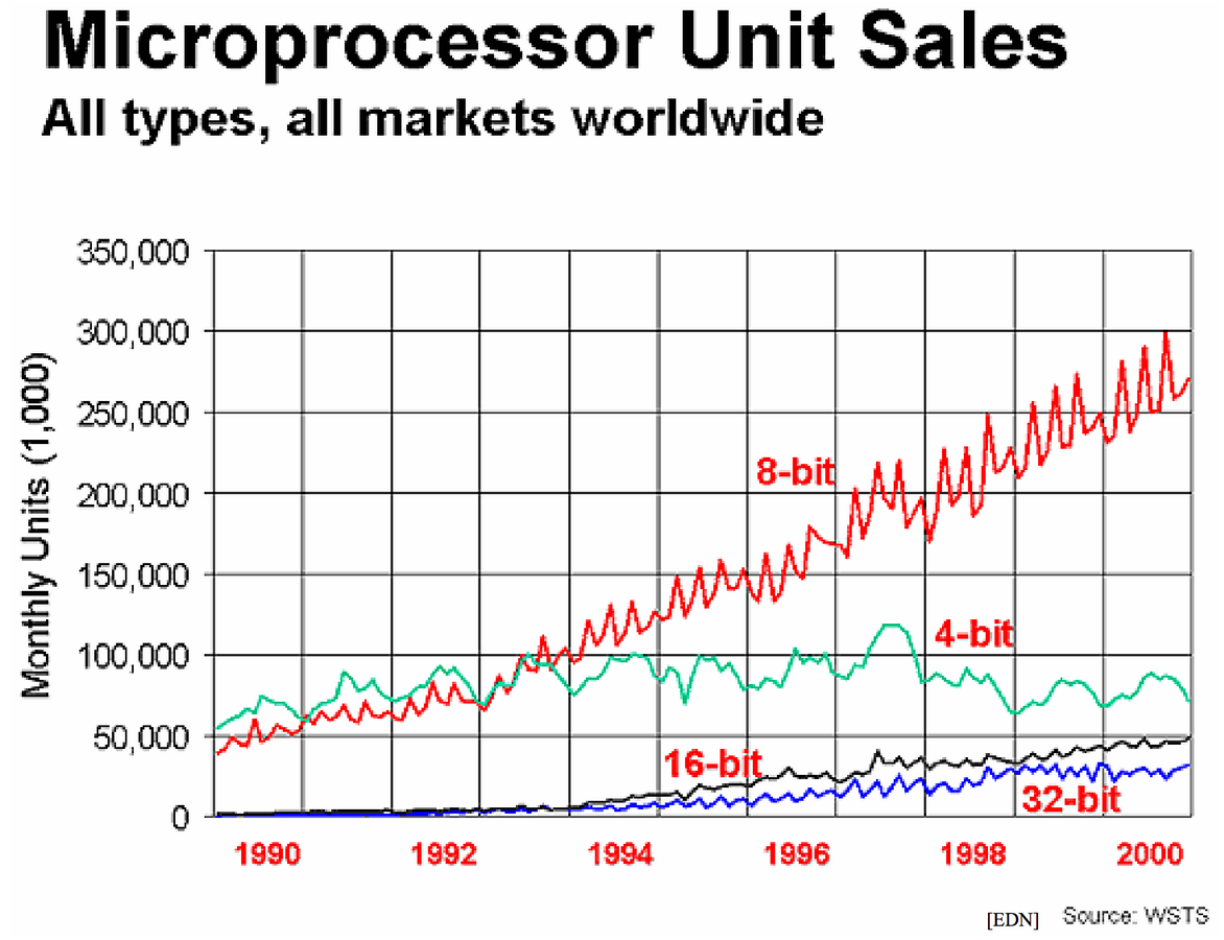
And this is just the seat adjustment...

Example car network:



CAN Controller area network
GPS Global Positioning System
GSM Global System for Mobile Communications
LIN Local interconnect network
MOST Media-oriented systems transport

[Leen02]



2013:

8 bit: 6.7 B

16 bit: 7.8 B

32 bit: 4.2 B

Why should you care?

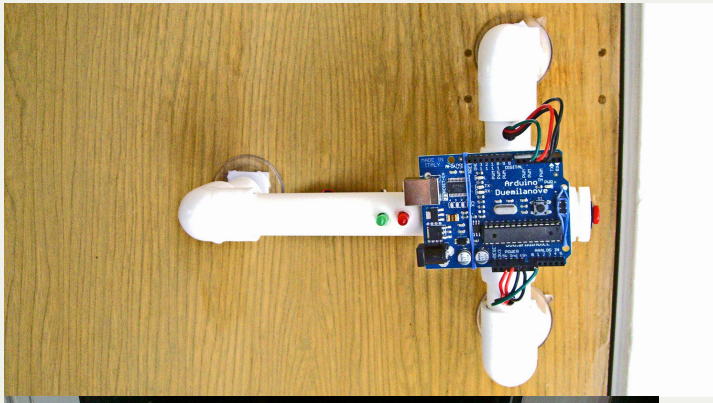
A growing embedded world

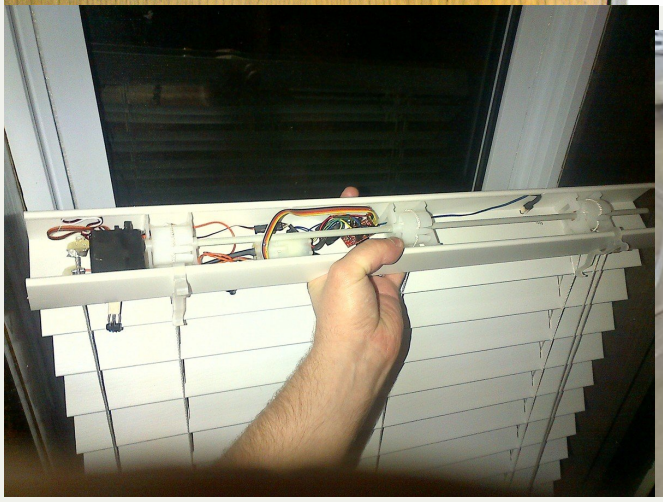
More products are becoming "smart" and "connected" and all of the companies developing them are looking for people who know how to develop for embedded systems.

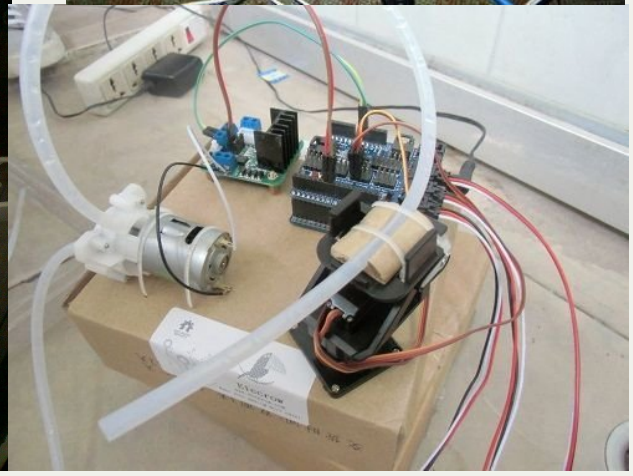
Green Hills®
SOFTWARE

A practical skill

DIY projects usually aren't done with a laptop attached to everything.

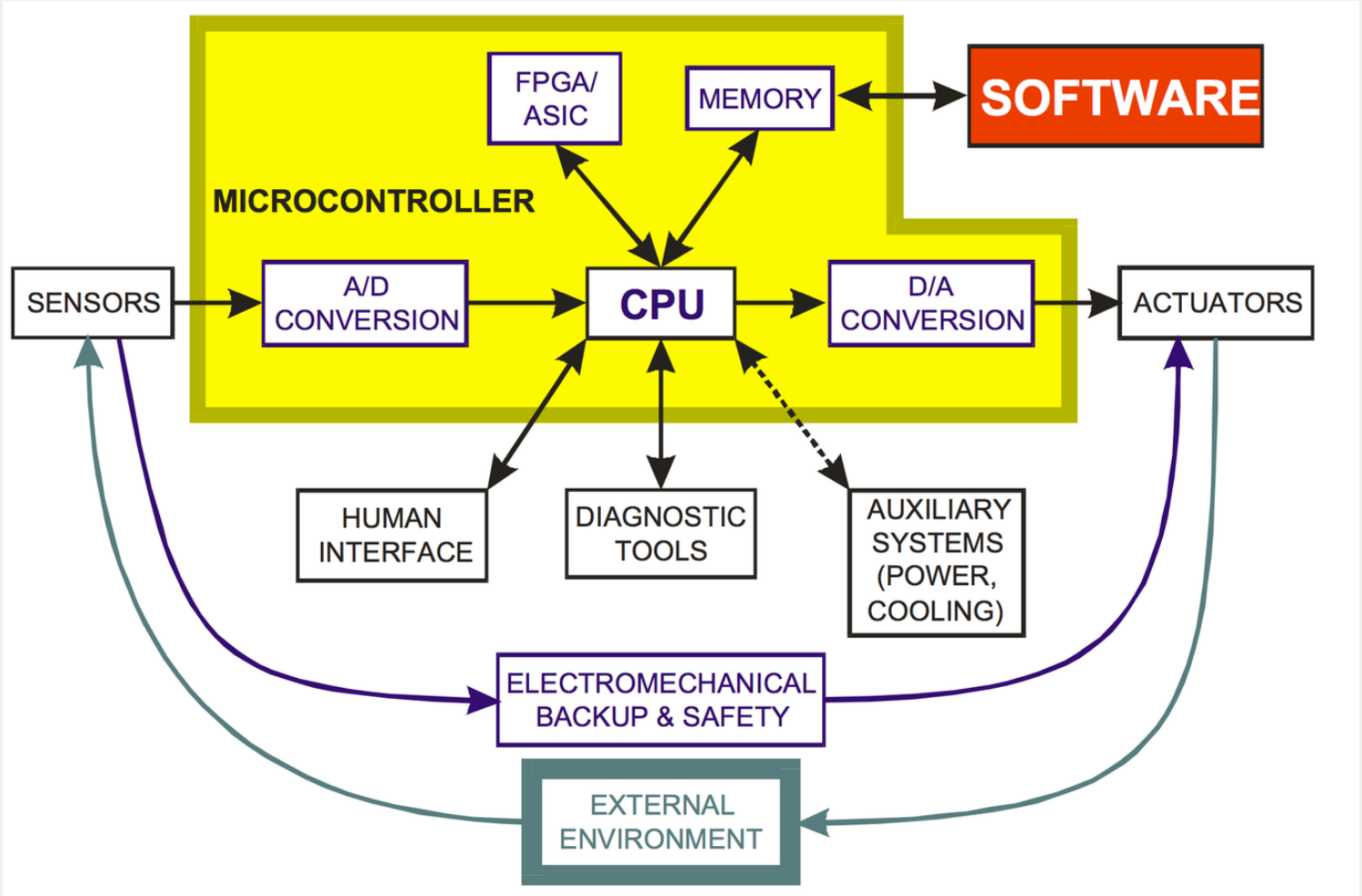






Parts of an embedded system

One look at an embedded system



Importantly

- CPU
- Input
- Output
- Memory

CPUs

A quick tour of popular microcontrollers and other CPUs

Bare 8 bit CPUs

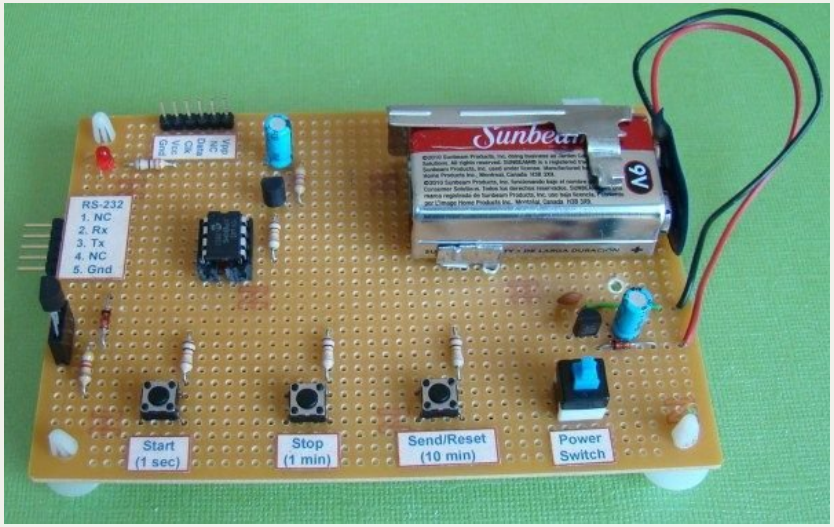
Many many many options....

Pros:

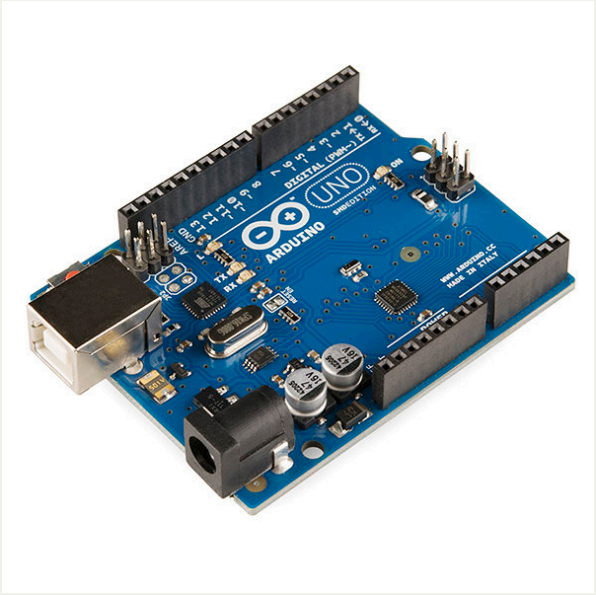
- Dirt cheap ($2-6)
- Scales well
- Small form factor

Cons:

- Need some EE knowledge to set up.
- Usually need additional parts for minimum product
- Often need special programming tools/software



Microcontroller Boards



Pros:

- A lot of documentation
- Easy to get started
- Plenty of 3rd party add-ons

Cons:

- More costly ($15-25)
- Still somewhat limiting in certain applications

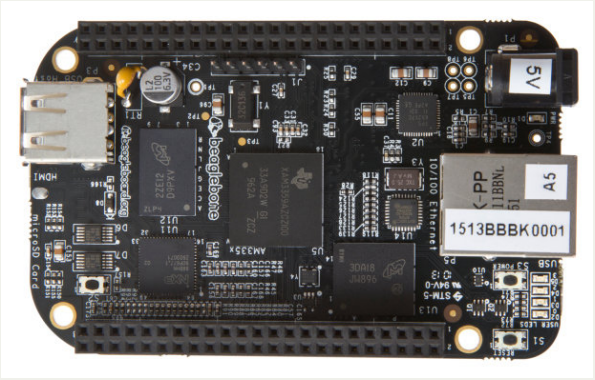
Computer on a board

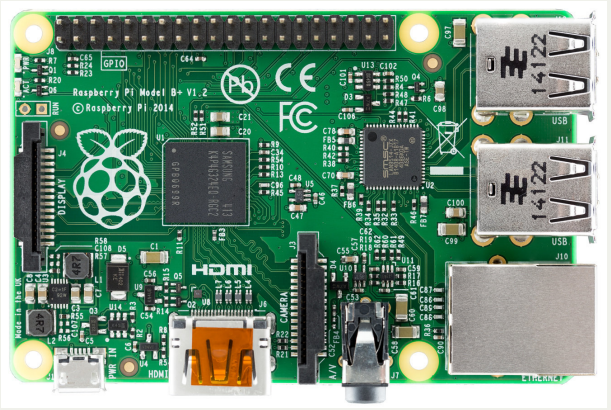
Pros:

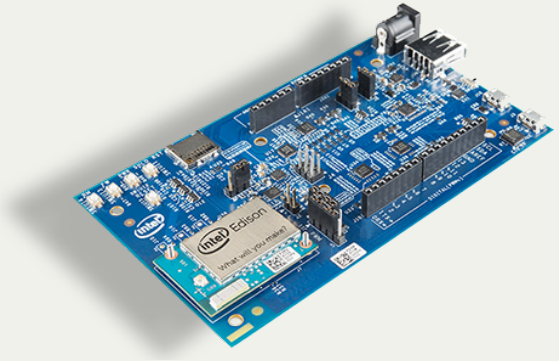
- Powerful
- Similar to writing software most people are used to
- Comprehensive inputs/outputs

Cons:

- Much more expensive ($40-$70)
- Has similar problems to a computer
- Larger power requirements
- Usually can't be "Programmed"







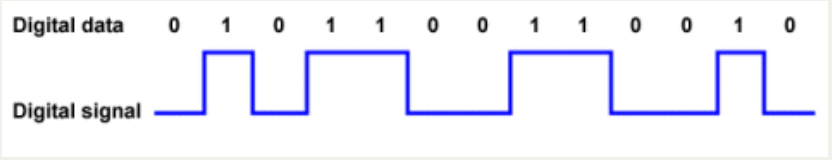
Input/Output

How to get your data in or out of the real world

Basic Digital

The most basic way to exchange information.

Nearly all CPUs have accessible input/output pins and are able to use those pins to output a 1 or 0. (or listen for a 1 or 0)



Basic Analog

Easy way to interface a CPU with "The real world"

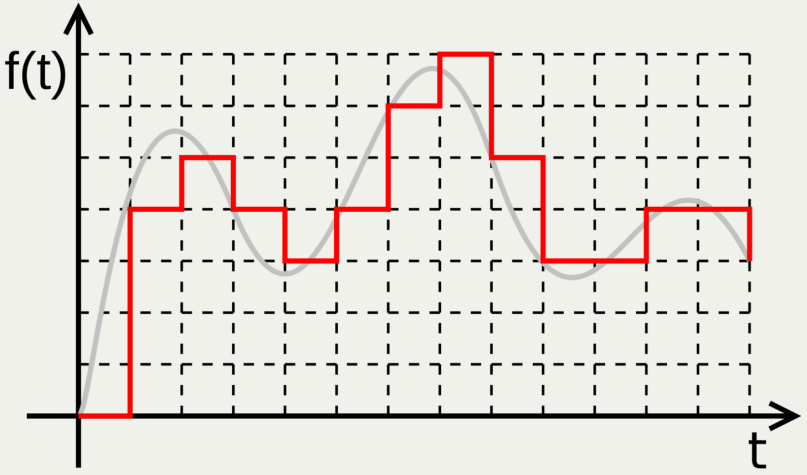
Many things don't just output ones and zeros.

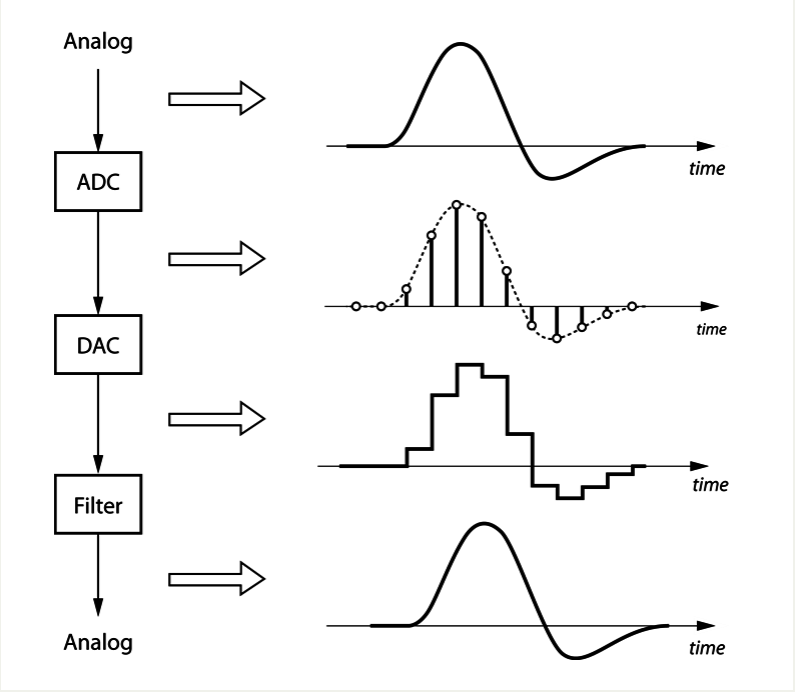
Solution: Use converters!

Digital to analog converter (DAC)

Analog to digital converter (ADC)

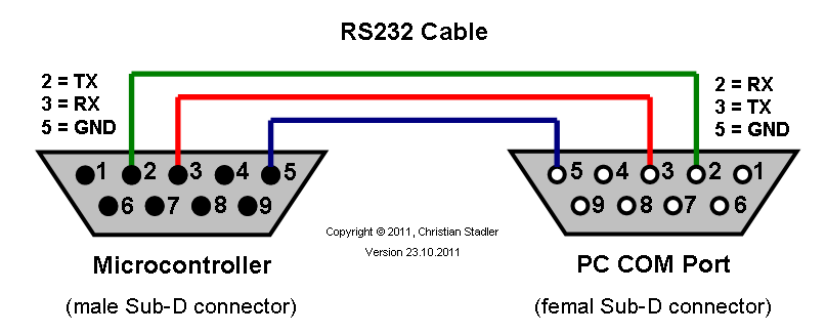
Data is represented as a value from a range.





Serial

The basis of a simple two way connection since 1962. (RS232)

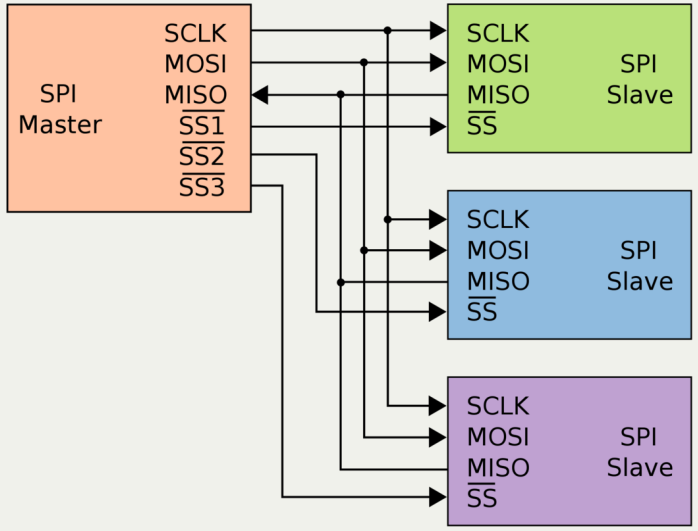


- Needs 2 wires to implement
  - Transmit/Receive (Tx/Rx)
  - Often a 3rd ground wire
- One to one communication
- Included on most devices

SPI

Like serial, but more listeners!

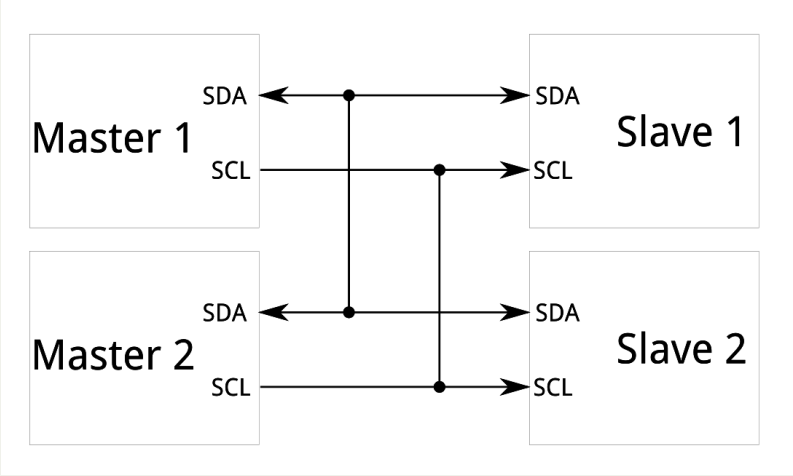
- Needs 3+n wires to implement n listeners
  - Clock, Tx, Rx, and a select for each listener
- One to many communication
- Also included on most devices
- With many listeners, the select lines can become annoying
- Used in many commerical devices today



I^2C "I squared C"

Flexible and featureful

- Needs 2 wires to implement
  - SDA & SCL
  - Often a 3rd power wire
- Many to many communication
- Pretty complicated, outside the scope of this presentation.



Input

- Light sensor (Analog)
- Force/Pressure (Analog)
- Accelerometer (Serial)
- Gyroscope (Serial)
- Potentiometer (Analog)
- RFID (Serial)
- Distance (I2C)
- pH (I2C)
- Temperature (I2C)
- Motion (Serial)
- Geiger counter (Serial)
- Heart rate (Serial)
- Alcohol Sensor (I2C)
- Vibration Sensor (Analog)
- Camera (Video codec)
- GPS (Serial)

Output

- Speakers (Analog)
- LEDs (Analog)
  - Many possibilities
- Screens (Parallel/Serial)
- Motors (Analog)
- Servos (Digital)
- Other devices!

Embedded coding issues/tricks

Some stuff you might run into

Coding tricks

Simple 8 and 16 bit CPU's aren't always the most featureful, and not every system has a nice C compiler. (Thankfully most do)

- What do you do if your chip only has an 8 bit multiply?
  - Shift and Add
- What you only have 3 registers?
  - Have to put the return elsewhere

^ Real problems addressed in CMU's embedded courses

Not the same old x86

- What is an "int"
  - How many bits?
- What exactly does one do with a triple de-refrence and add operator?

Thankfully, most widely used chips today use instruction sets that have C or other language compilers.

Robustness and Safety

Would you trust the person sitting next to you to design your car's breaking system?

What about the an airplane flight control system?

Robustness and Safety

Some important concerns for safe embedded systems:

- Floating point
  - Imprecise, not safe to accumulate values in
    - Patriot Missile incident
- Redundant systems!
  - There shouldn't be only one set of wires between the pilot and the engines!
- Watchdog and rebooting
  - Mars rover "freeze"
  - Multiple reboot levels
  - Is it safe to reset?
- Failsafe vs Faildeadly

Robustness and Safety

Most importantly:

"If safety is not baked into the recipe in the process of creating the product, it cannot be added later."

-Philip Koopman, Embedded Professor at CMU

I want more?

18-348/349

More hardware/More software

Thanks for coming!