Ch. 1: Introduction to Computer Architecture (of Embedded Systems)

Embedded Systems

vs. General-Purpose systems

- Functional
  - GP: Load diff. programs; OS matters
  - ES: usually runs one app all the time; OS usually doesn't matter
- Nonfunctional
  - GP: desktop, laptop, no strict constraints
  - ES: size, shape, battery life, weight, heat, noise, ...

Concepts in System Architecture

Application	Application(s)	application
OS	OS	(system software)
firmware	firmware	firmware
hardware	hardware	hardware
General-Purpose Computers	Complex Embedded Systems	Simple Embedded Systems

Firmware vs. Software

- Firmware
  - program code already resident in hardware
  - upgradable if stored in nonvolatile memory (flash or EEPROM)
- Software
  - program code (OS) loaded in by firmware (bootloader) into RAM to run
  - program code (application) loaded by OS (loader) into RAM to run

Firmware in Embedded Systems

- Firmware can contain all program code
  - Bootloader, OS, application
  - OS may be optional
- Two styles
  - Execute in place (XIP - from NV mem)
  - Boot loading: copy firmware from NV mem to RAM and run from RAM

Processor
(or Microprocessor or CPU)

- Executes *instructions* from program memory
- Instruction: opcode, operands
  - Operate instructions (+, - , *, /, shift..)
  - Control instructions: (branch, jump, ...)
  - Load/store: moving data to/from data memory
- Program and data may occupy shared or separate memory

Microcontroller Unit (MCU)

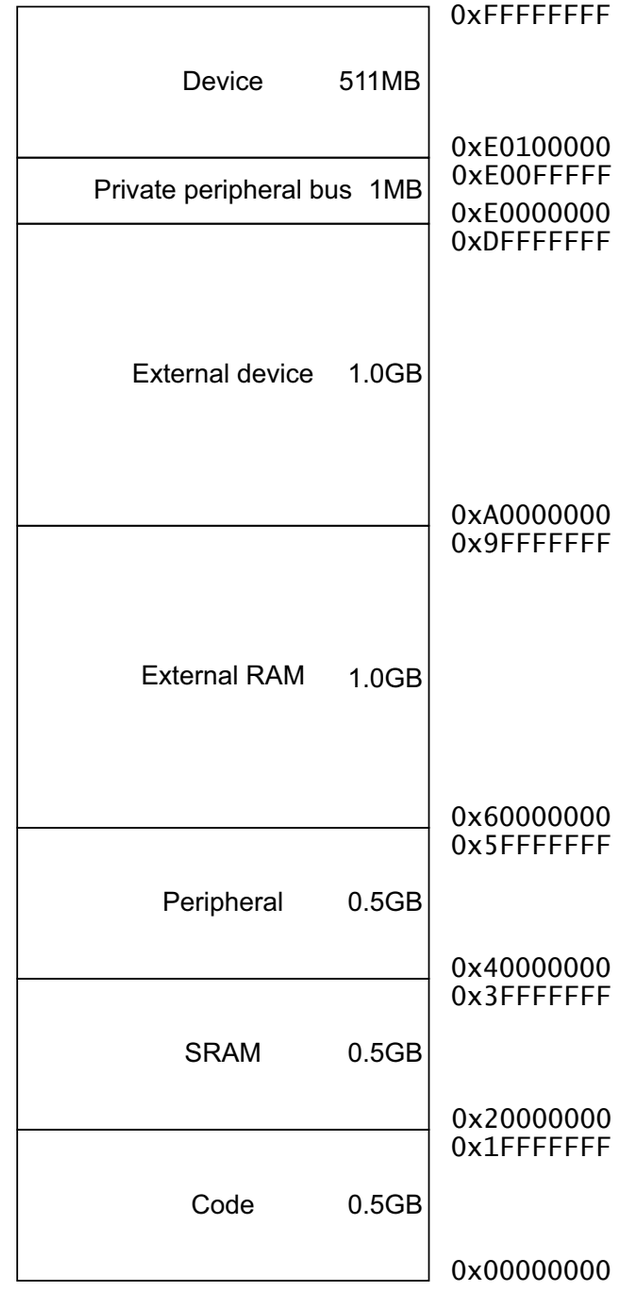
- Microprocessor + other features on chip
  - Input/output (I/O) interfaces
  - Memory: SRAM, flash
  - ADC, DAC, voltage comparator, ...
  - Network interface, radio transceiver
- Central component of most embedded systems

How MCU performs I/O

- Memory-mapped I/O
  - Certain memory addresses selects an I/O controller
  - Load/store to I/O address triggers I/O action
- Ported I/O
  - Separate address space for special function registers (SFR)
  - Read/Write SFR triggers I/O action

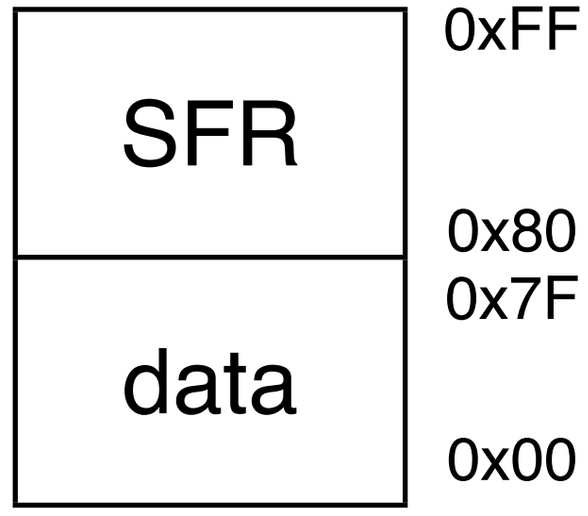
Examples of MMIO: ARM Cortex M0

- ARM Cortex-M0:
  Memory-mapped (4GB total)
- Built-in:
  - 0.5G each for code, SRAM peripherals
- External:
  - 1G each for RAM, devices
  - 0.5G for private bus & device



Example Ported I/O: Intel 8051

- Internal RAM (256 bytes)
  - Lower 128 bytes for data
  - Upper 128 bytes for SFR
  - Use MOV instruction
- External data (64 KB)
  - Separate from internal RAM
  - Use MOVX instruction

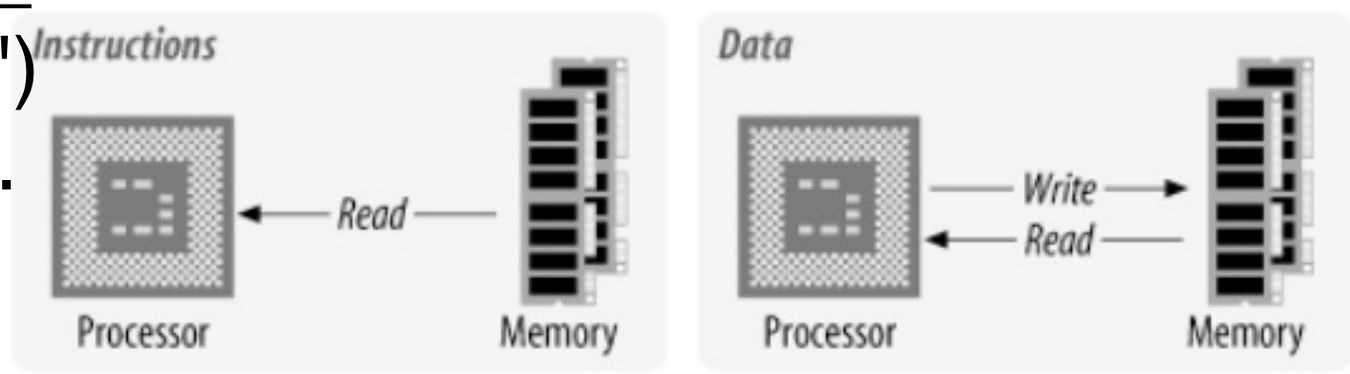


Types of memory for MCU

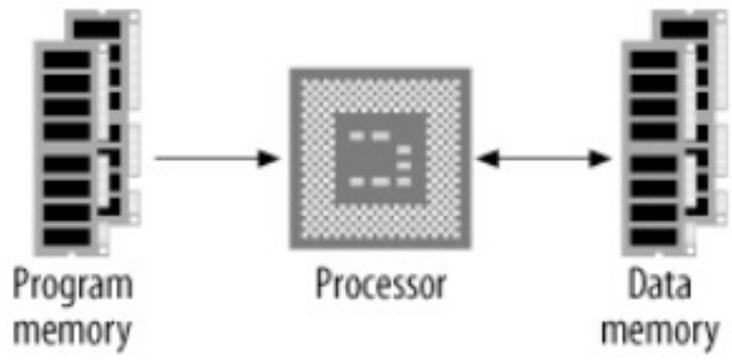
- Inside processor:
  - registers: 4-32 "words" addressable directly by instruction
  - internal memory: 128-256 words, possibly overlaid with registers (not on all MCUs)
- External to processor core (but may be on-chip)
  - SRAM - for data
  - Flash - for program code and constant data
- Off-chip memory

Address Spaces for Code and Data

- Princeton architecture:
  (AKA "von Neumann")
  Memory shared betw.
  instruction and data
  (most CPUs)



- Harvard architecture:
  Separate memories
  for instructions and
  data
  (most MCUs)



Von Neumann vs Harvard Architecture

- Most MCUs: Harvard architecture
  - Separate space for code flash from data SRAM => Can fetch code and access data simultaneously
    - e.g., 8051, AVR,
- More flexible: von Neumann
  - Self-modifying code, bootloading, interpreter
  - e.g., ARM, MSP430,
- Harvard ones may be wired as von Neumann

DSP vs MCU

- DSP = digital signal processors
  - instructions for numeric operations
    e.g., multiply-accumulate
  - operation on data arrays and banking
  - repeated operation (hardware looping)
  - mostly Harvard architectures
- MCU: more general purpose
  - better at "control" than "data" operations

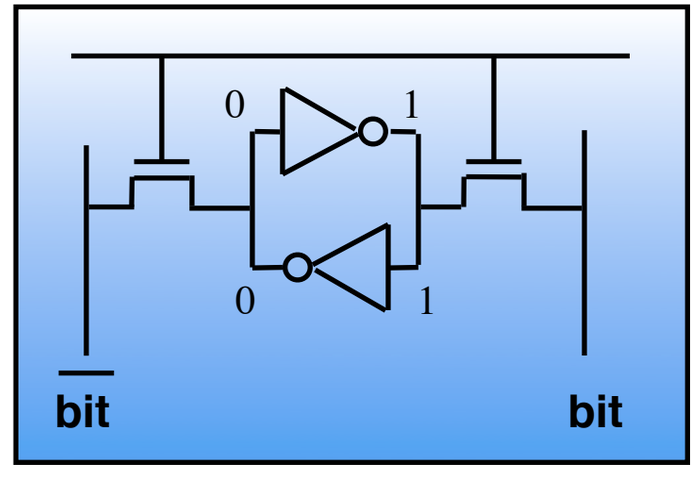
Memory

- Volatile Memory (RAM: random access memory)
  - cut power => lose content
  - randomly access (read/write) each location, as opposed to a stack or queue ("structured")
  - give an explicit address, read/write the data
- Nonvolatile Memory (ROM: read-only memory)
  - keeps content even without power
  - also randomly addressable

RAM

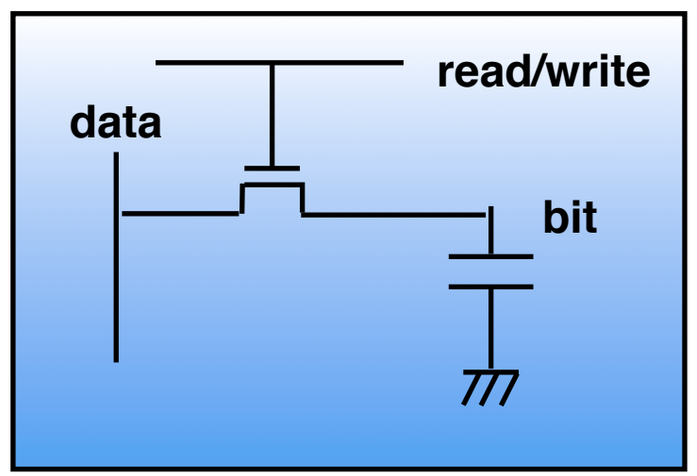
- SRAM: static RAM
  - Used in most embedded systems on-chip
  - Simpler to interface, low power, costlier
- DRAM: dynamic RAM
  - Used in most off-chip memory for computers
  - More complex, but cheap per bit

SRAM: Static RAM



- Static: (2 steady-states)
  - Cross-coupled inverters
    - Simpler to use, doesn't need refreshing
- Disadvantage: higher cost
  - six transistors per bit (vs. one for DRAM)
  - could get down to 5 or 4, but still costly

DRAM: Dynamic RAM



- Cell = switch to capacitor
  - Capacitor stores charge
  - Switch connects/disconnects capacitor
- Advantage: high density (1 transistor / bit)
- Disadvantage: more complex
  - row-column access
  - needs refresh periodically and reading

ROM: "Read-only memory"

- More accurately "Nonvolatile Memory"
  - content is retained after power off
  - some could actually be written into! But not as convenient as RAM (page access)
- Types of ROM
  - PROM, OTP, EPROM, UV-EPROM, EEPROM, Flash, Mask ROM, ...

PROM (OTP)

- PROM = Programmable ROM
  - programmable => by the user
- OTP = One-time Programmable ROM
  - Originally, each bit has a fuse
  - to program, blow the fuse with high current (to permanently make a bit)
    => "burning" ROM, use a "burner"
    => permanent, can't be re-programmed!!

EPROM, UV-EPROM

- EPROM: Erasable PROM
  - Electrically programmable (~12+ volts) w/out burning fuse; use a programmer
- UV-EPROM: Ultraviolet-Erasable PROM
  - after programming, cover window w/tape
  - to erase, expose to UV light (10-20 min) in an eraser

EEPROM

- Electrically Erasable PROM
  - electrically programmable and erasable
- Advantages over UV-PROM
  - UV one requires erasing entire ROM, but EEPROM can erase one byte at a time
  - Actually, the MCU can erase and write it!
    => strictly speaking, not a "ROM"!

Flash Memory

- A kind of EEPROM
  - Organized in bulk for better density
  - Page = smallest *erase* unit
  - Sector = smallest *write* unit.
    (Multiple sectors per page)
- Bits per cell
  - single-level cell (SLC: 1-bit/cell, faster)
  - Multi-level (MLC: multi-bit/cell, cheaper)

Types of flash memory

- NAND-flash
  - cheaper: page erase, sector read/write
  - good for data (e.g., digital camera)
- NOR-flash
  - page erase, sector write, word read
  - good for MCU program (firmware) XIP
- Limited number of rewrite cycles (10,000)

Mask ROM

- Fabricated ("fabbed") by IC foundry
  - not programmable => cheaper for high volume, after firmware is finalized
- Copy protection
  - other PROMs content may be read out
  - Mask ROM => hard or impossible to copy!

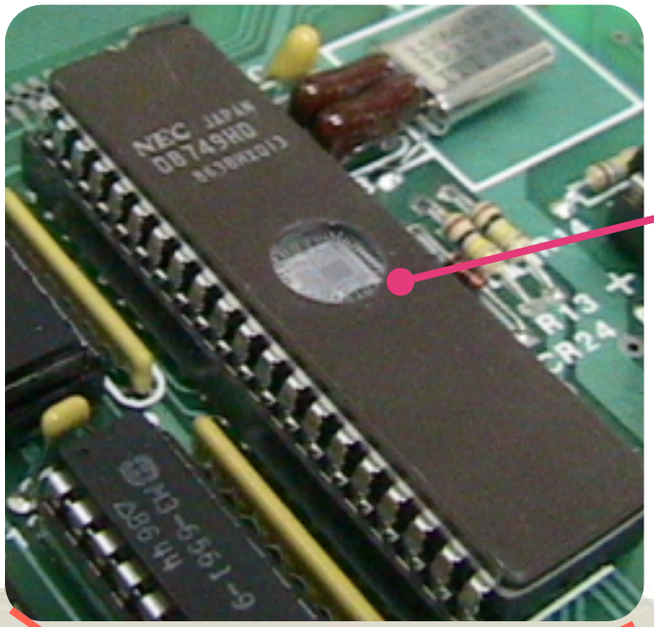
Question: how to program?

- Make (mask) ROM (cheap if large quantity)
  - Problem: not changeable
- Write to it
  - PROM/EPROM: (P=programmable) write-once
  - UV-EPROM: erasable w/(UV) light
  - EEPROM: electrically erasable PROM
  - Flash: a type of EEPROM

How to write to nonvolatile memory?

- Hardware
  - External programmer/writer (hw)
  - In-system (on-board) programmer
- Software/firmware
  - Bootloader (on startup)
  - Application code (writes to flash/ EEPROM like an I/O device)

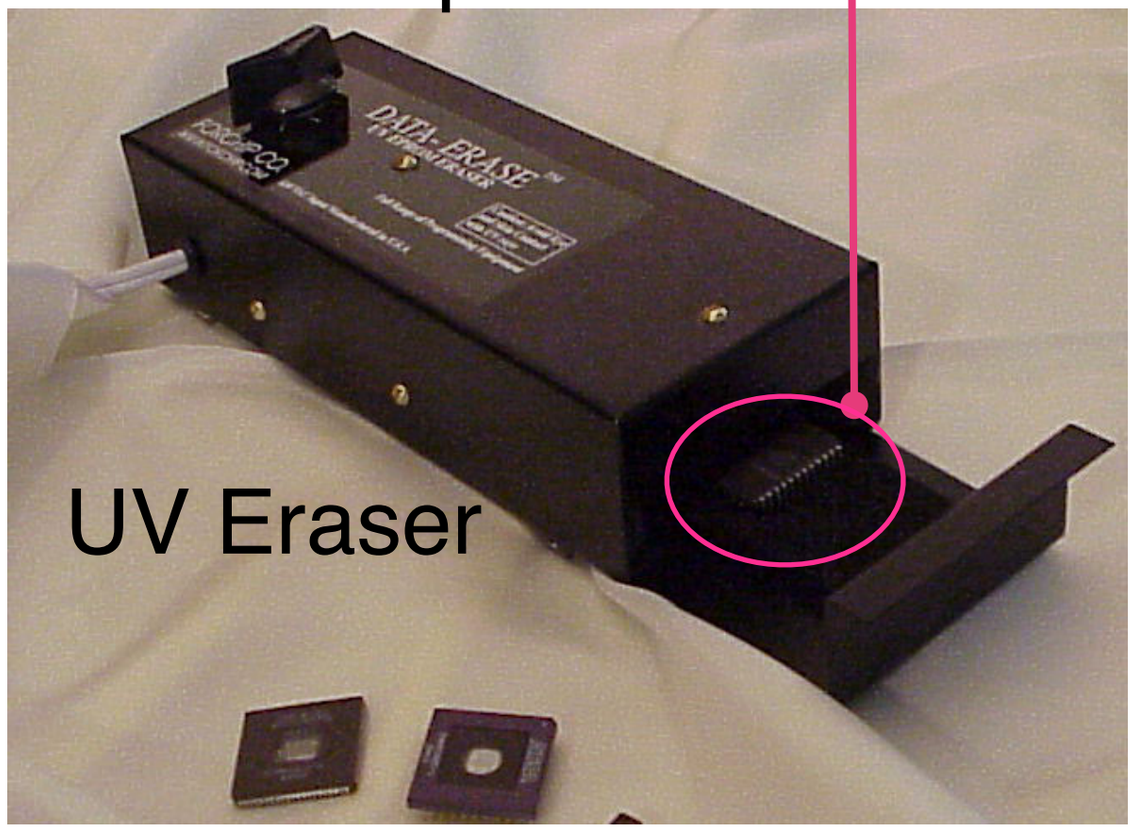
EPROM Programmer and UV-Eraser



Chip with
UV-erase window

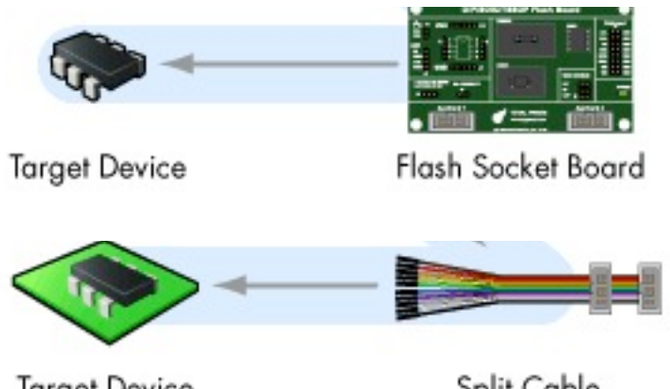
Programmer

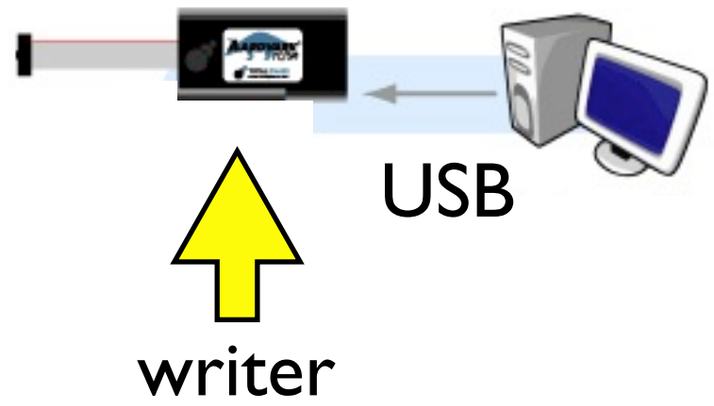
put chip inside drawer to expose to UV



UV Eraser

Serial flash/EEPROM writer





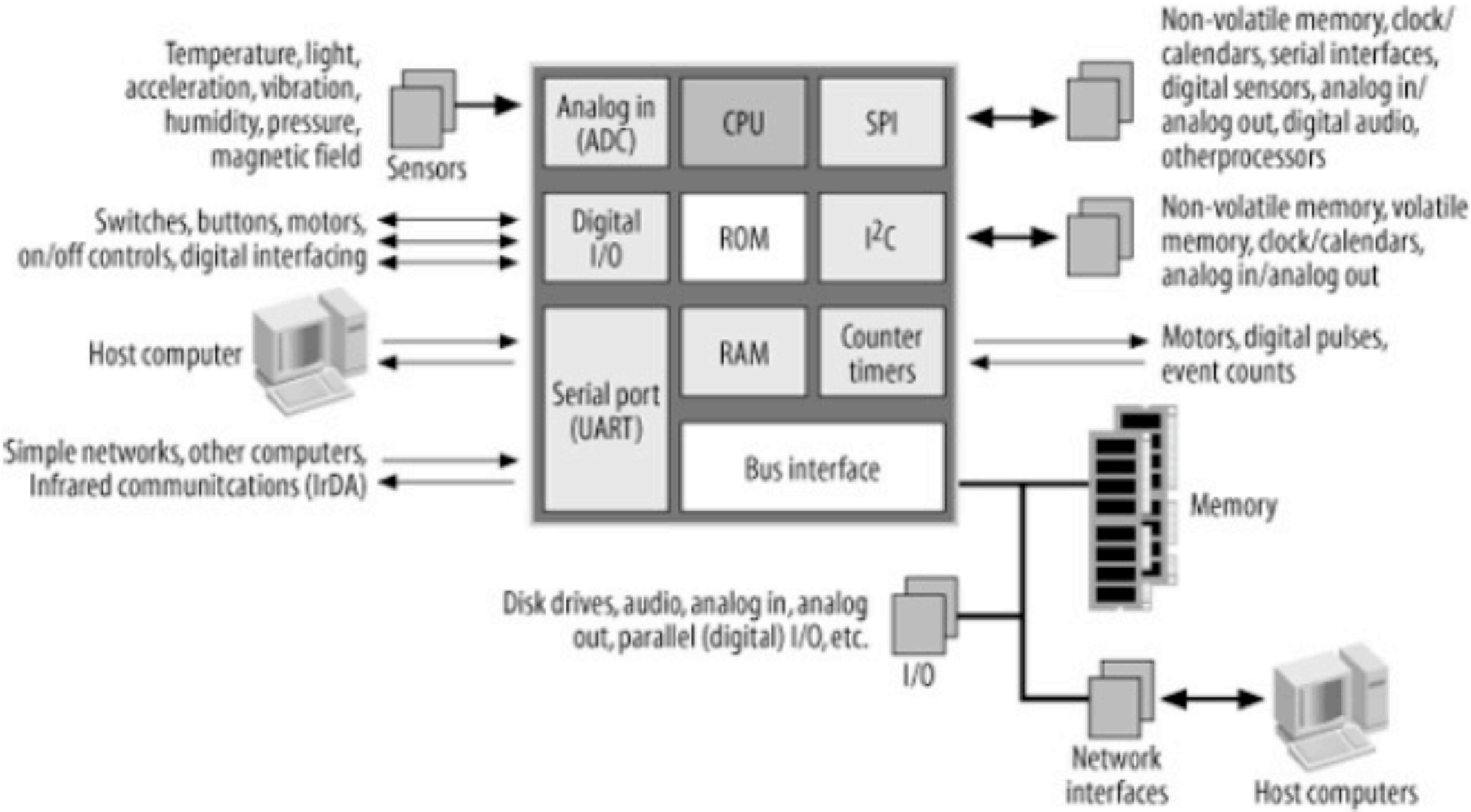
- The target device (serial flash, EEPROM, etc) may be built-in to MCU or an external chip
- Verification after writing (important!)

Credit: http://www.totalphase.com/solutions/apps/flash_eeprom_guide/?gclid=CIC4jf2io6YCFRiAgwodXgznpQ

Self-programming software/firmware

- Bootloader
  - startup firmware that may offer option to run normally or write new firmware
- Self-modifying code
  - application code on MCU writes to its own EEPROM/flash as it runs
  - Can work over any interface (serial, USB, wireless, Ethernet, any device)

Block Diagram of an Embedded System



Platform-Based Design

- Use a platform as a starting point
  - customize later if necessary
- Platform = reusable system design
  - not just hardware parts and board
  - also software library, OS, driver,

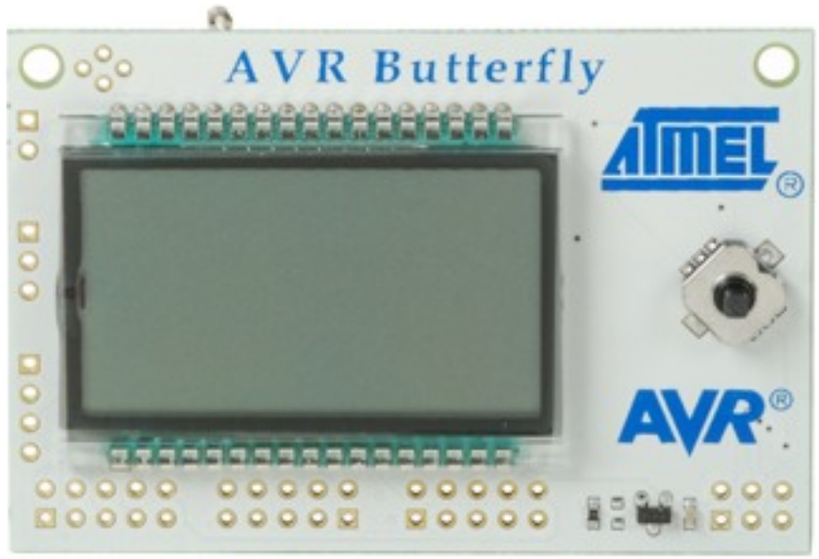
MCU platforms

- Simulator program
  - EdSim51 (Java app for 8051+I/O board)
- Boards
  - Evaluation board: MCU on a board
    - may come with peripherals, programmer
    - Low end: ~$20; Avg: $50; High: >>$100s
  - Provided by class: EcoBT

Evaluation Boards

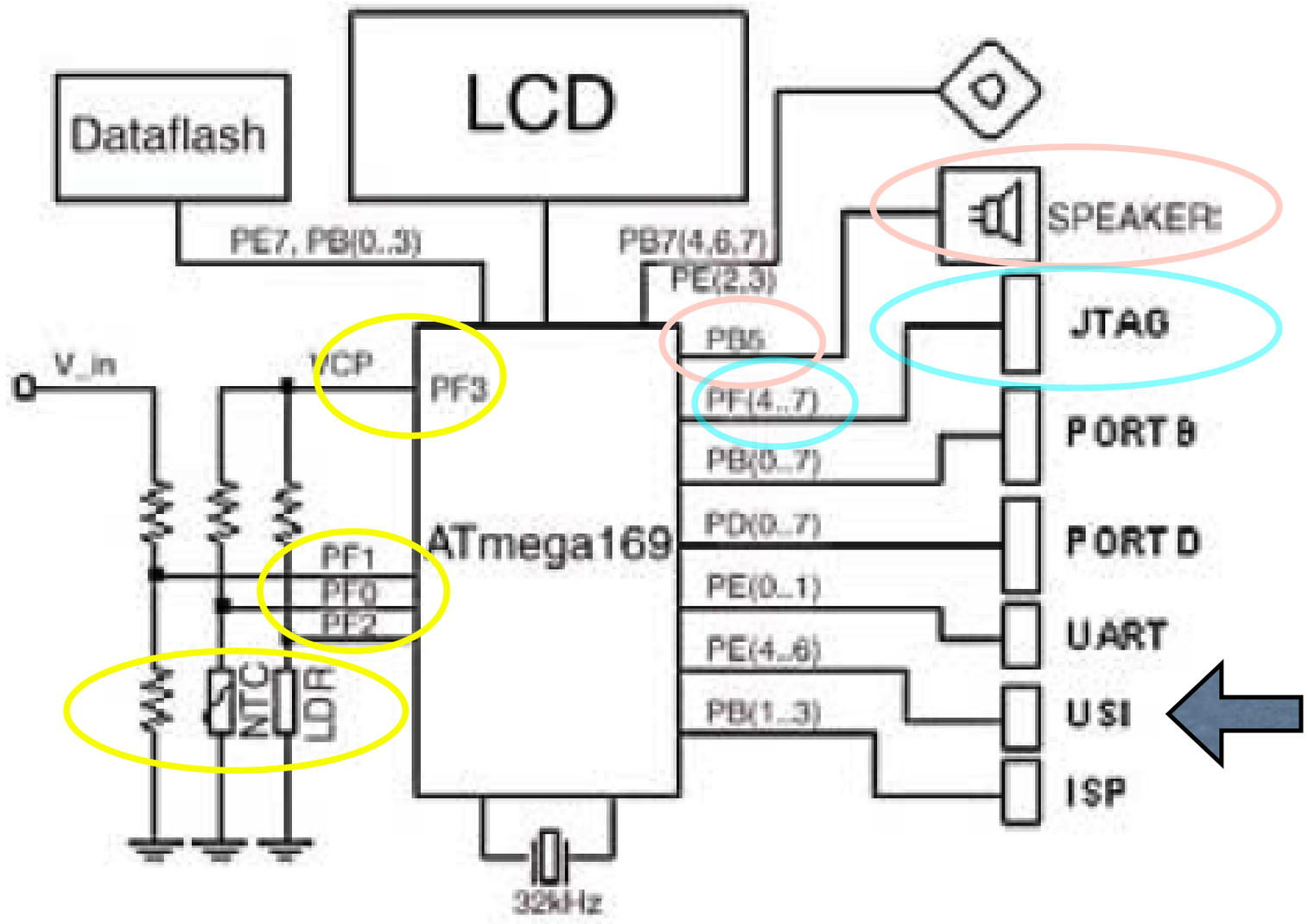
- board with MCU and some peripherals
  - power regulation, I/O prototype area, possibly pushbuttons, LCD/LED, etc
- Reasons:
  - convenient, ready to use, flexible
- Issues:
  - platform-specific, not portable; compiler

Example 1: AVR Butterfly



- $20, MCU with LCD, temp/light sensor, tone gen.
- 512K flash over SPI
- Comes w/ a *bootloader* over serial port
- ATMega 169 MCU:
  - UART, SPI, USI, PWM, 8ch/10-bit ADC, Timer/counter, built-in LCD driver, GPIO
- Link: http://search.digikey.com/scripts/DkSearch/dksus.dll?Detail&name=ATAVRBFLY-ND

Port usage by the AVR Butterfly



This includes SPI

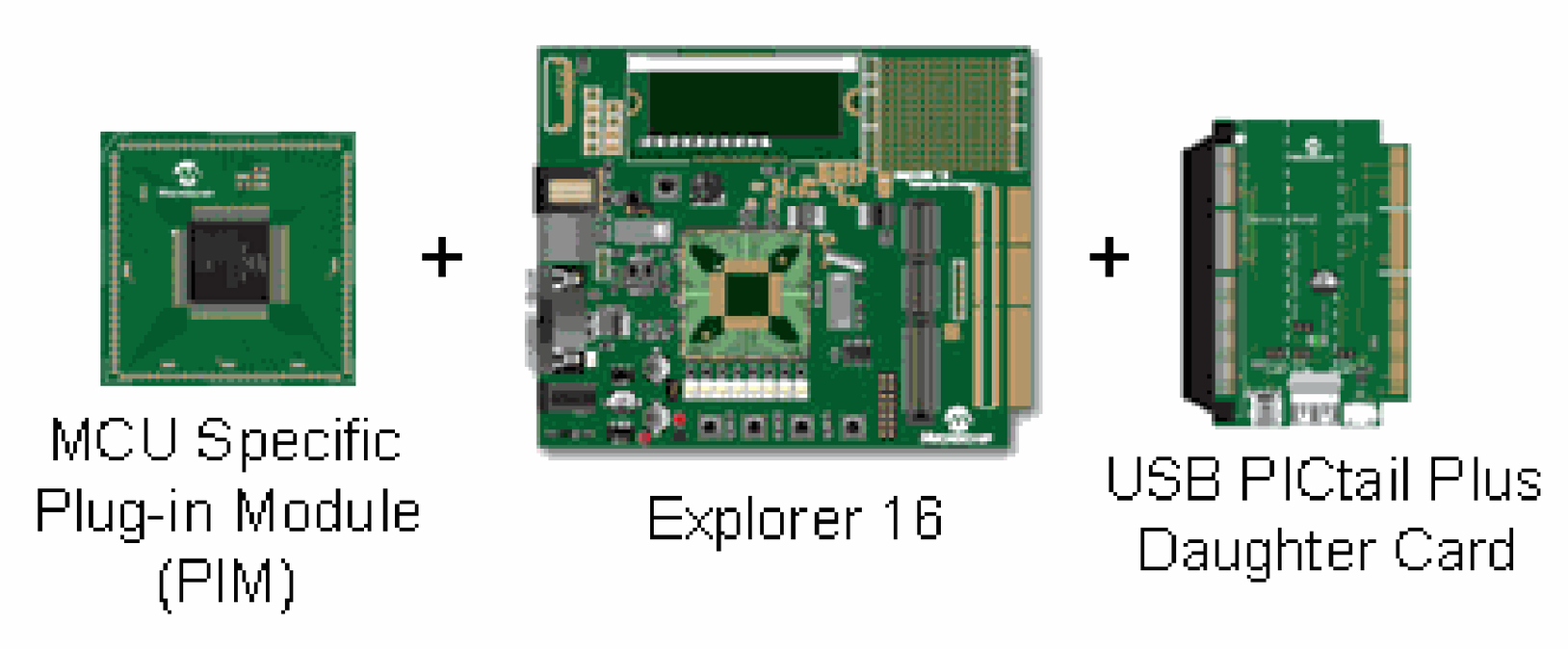
2. MSP 430 boards

- LaunchPad ($5)
- Touch Sensor
- Experimenter board
- Chronos Wireless Watch ($50)

http://www.ti.com/mcu/docs/mcusplash.tsp?contentId=128825#eZ430

3: Microchip PIC24 USB Development Kit

- PIC24FJ256GB100 with USB On-The-Go
- PIM + "Explore 16" + UCB PICtail
- More info at http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=2651&param=en534494

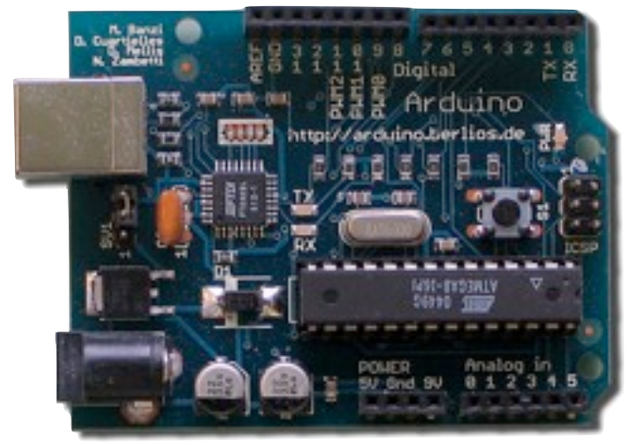


3. PICtail boards

- Audio, speech playback
- CAN/LIN (local interconn. network)
- SD & MMC Cards, Smart Card
- Ethernet, USB, IrDA, WiFi, GPS
- Motor interface
- Touchpad Color LCD
- Sensor: temperature, light, humidity

http://www.microchip.com/pictail

4. Arduino



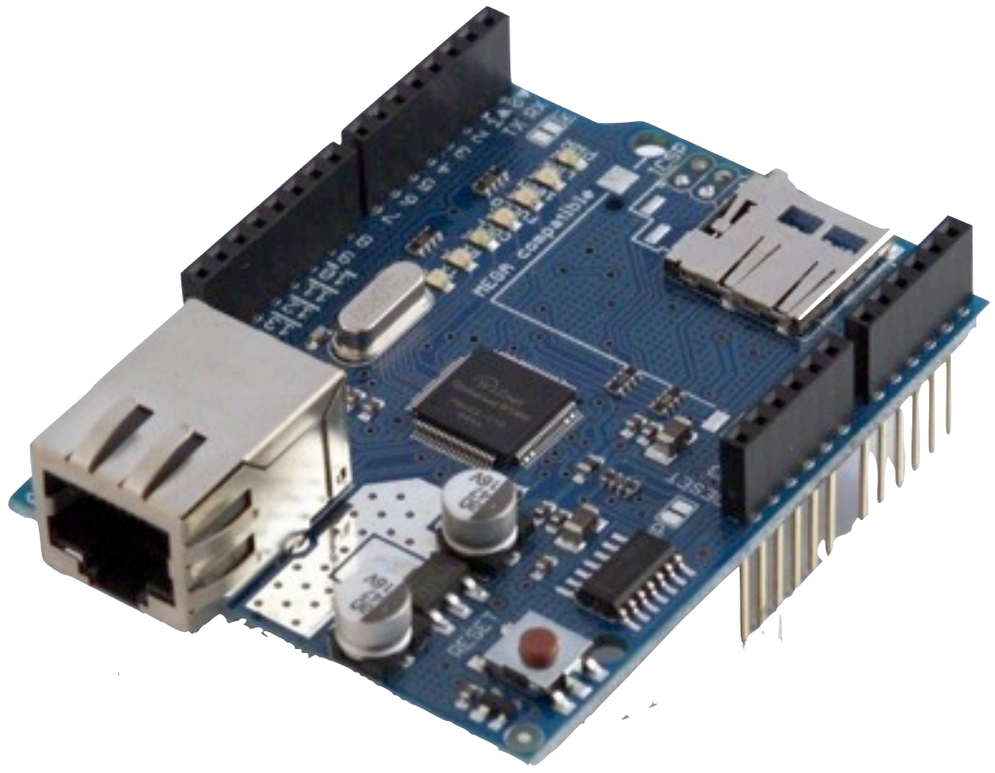
- Popular with robotic robotic art
- Open-source Hardware (board) design
  - MCU: Atmel ATMega168
  - Newer MCUs are also used
- Software
  - programming language (Wiring) + IDE (Processing)
  - works with Flash, MaxMSP

Photograph by SparkFun Electronics. Used under the Creative Commons Attribution Share-Alike 3.0 license.

http://www.arduino.cc/en/

4. Arduino Shield

- Expansion Boards
  - Standard connector for signals and power
  - Stackable
- Several official shields
- Many available from 3rd party
  - WiFi, Ethernet, sensors, storage, ...



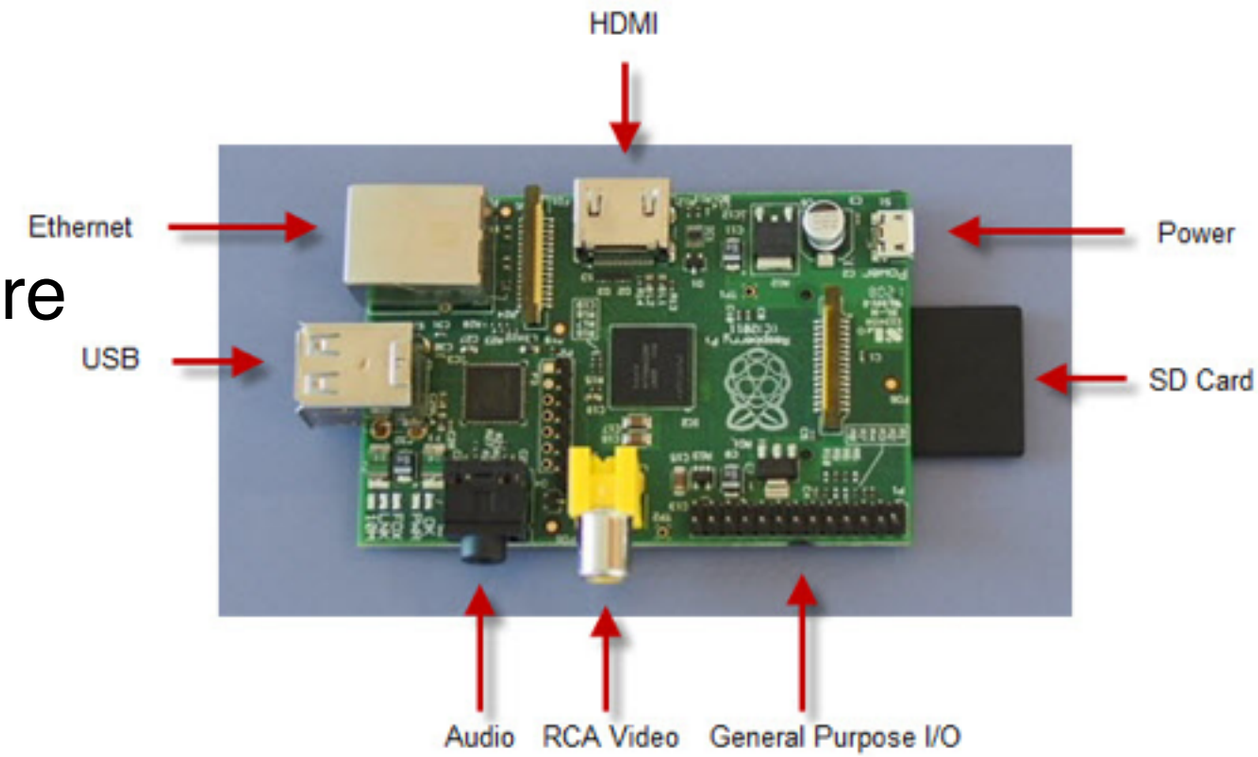
Ethernet Shield

5. TMS570LS20SUSB

- MCU: Dual-Core Cortex-R4F ARM
  - running in lockstep!
  - 2MB on-chip flash
  - 160KB RAM
- Support:
  - JTAG/USB, GUI/sim/asm, CCStudio
  - http://focus.ti.com/docs/toolsw/folders/print/tmdx570ls20susb.html

6. Raspberry Pi

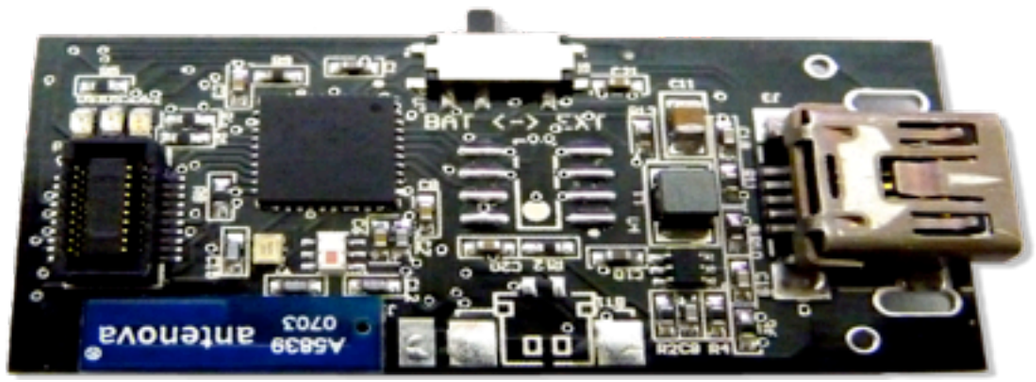
- $35 Credit-card-sized computer
  - HDMI out, USB, Ethernet
- Broadcom BCM2835 chip
  - 700 MHz ARM1176JFS core w/ floating point
  - Videocore 4 GPU
- 512MB RAM Package-on-package
- Boots Linux from SD Card

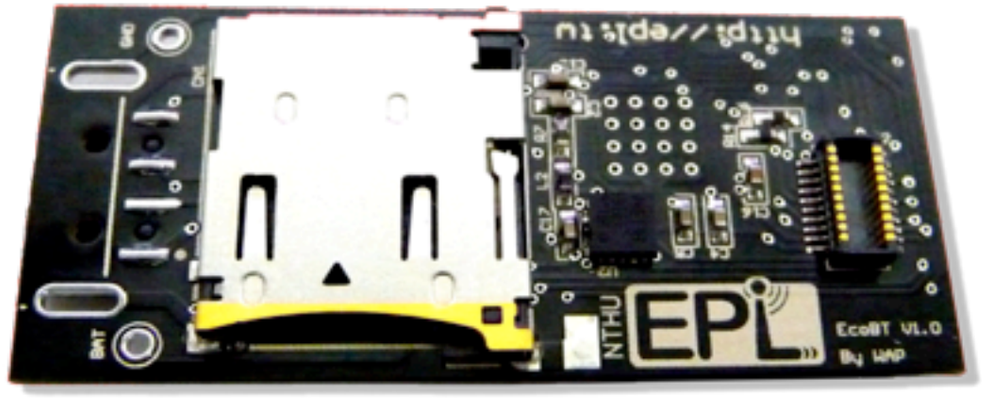


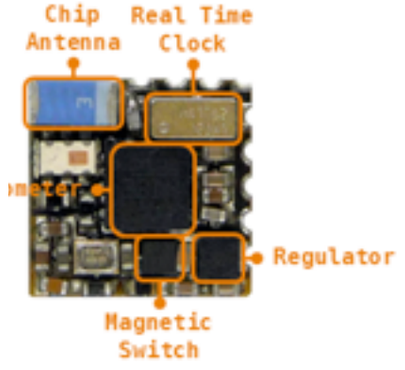
http://www.raspberrypi.org

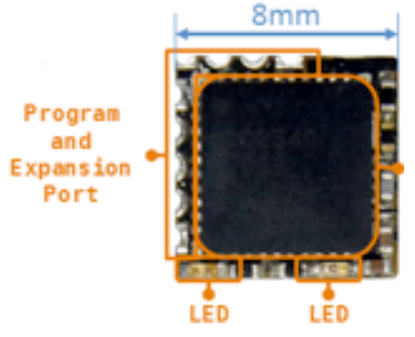
7. EcoBT

- Wireless sensor nodes
  - Bluetooth 4.0 Low Energy
    - Sizes:4 x 2 cm, 8 x 8 mm
- TI CC2540 MCU
  - 8051 core, 8 KB RAM, 256 KB flash
  - accelerometer, RTC









Summary of Platforms

Platform	MCU	I/O
AVR Butterfly	8-bit (AVR)	LCD, light, toneGen, joystick
MSP kit	16-bit (430)	USB (slave), add-on board
PIC24 kit	16-bit (PIC24)	USB (master or slave), add-on board
Arduino	8-bit (AVR)	Serial or USB (slave), Shield boards
TMS570	dual 32-bit (ARM)	USB (master or slave)
RaspberryPi	32-bit (ARM)	USB master, HDMI out analog A/V, Ethernet, MicroSD card,
EcoBT	8-bit (8051)	USB (slave), triaxial accelerometer, RTC, magnetic switch, add-on boards

there are many others!

Examples of Embedded Systems

- Sprinkler controller
- Wireless ID card reader

1. Sprinkler controller

- Controls sprinklers for watering plants
  - During certain time of day, day of week, watering duration
- Different watering zones

1. Sprinkler controller: Base functionality

- Real-time clock
  - needs to know day of week, hour, minute
  - could extend to day, month, year, ...
- Scheduler
  - On/off time, day of week for each zone
  - Manual override
- User interface for display and input
- Electrical interface for on/off control

1. Sprinkler Controller: Feature Enhancements

- Add a rain or moisture sensor
  - automatically disable watering on rainy days
  - increase watering time on dry days?
- Network Connection
  - Log in to weather service for forecast?
  - Synchronize time
- Battery backup option

2. Wireless ID-Card Reader

- Motivation: Undergrad classes
  - Taking attendance by swiping ID card
- Improve over commercial solution
  - Real-time logging into database if possible
  - Store locally if disconnected
  - Multiple readers, synchronized operation

Commercial Solution: PDA + Card Reader

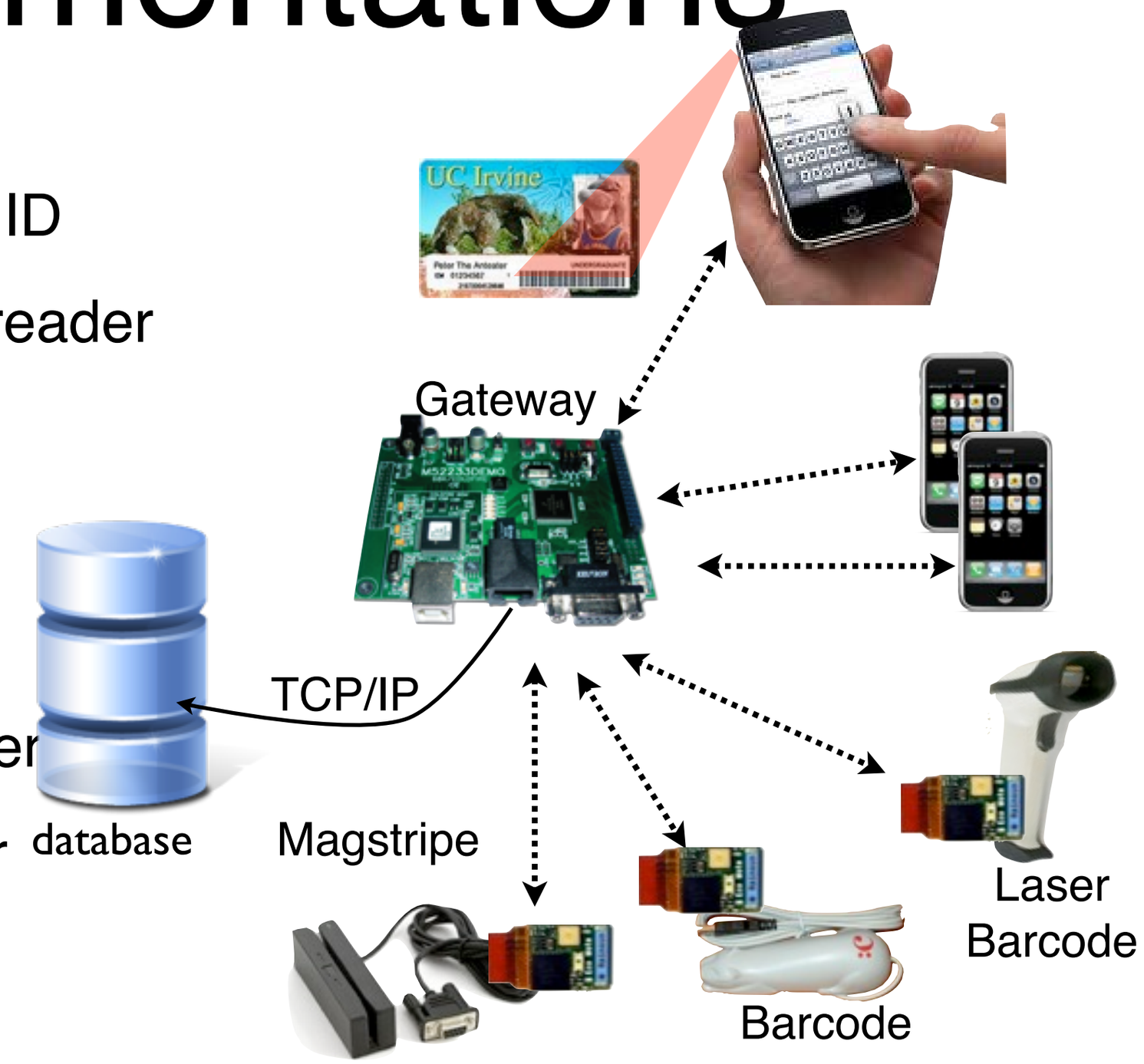
- PDA (Windows Mobile)
- Card reader in CF-II expansion slot
- Price: $330
- Problems:
  - bulky, expensive
  - sync with PC, not real-time sync

Organization

- Gateway Unit
  - Similar to Wi-Fi access points, but for handheld scanners
    - Responsible for TCP/IP uplink to DB
- Handheld Units
  - mobile, low cost; need not speak TCP/IP

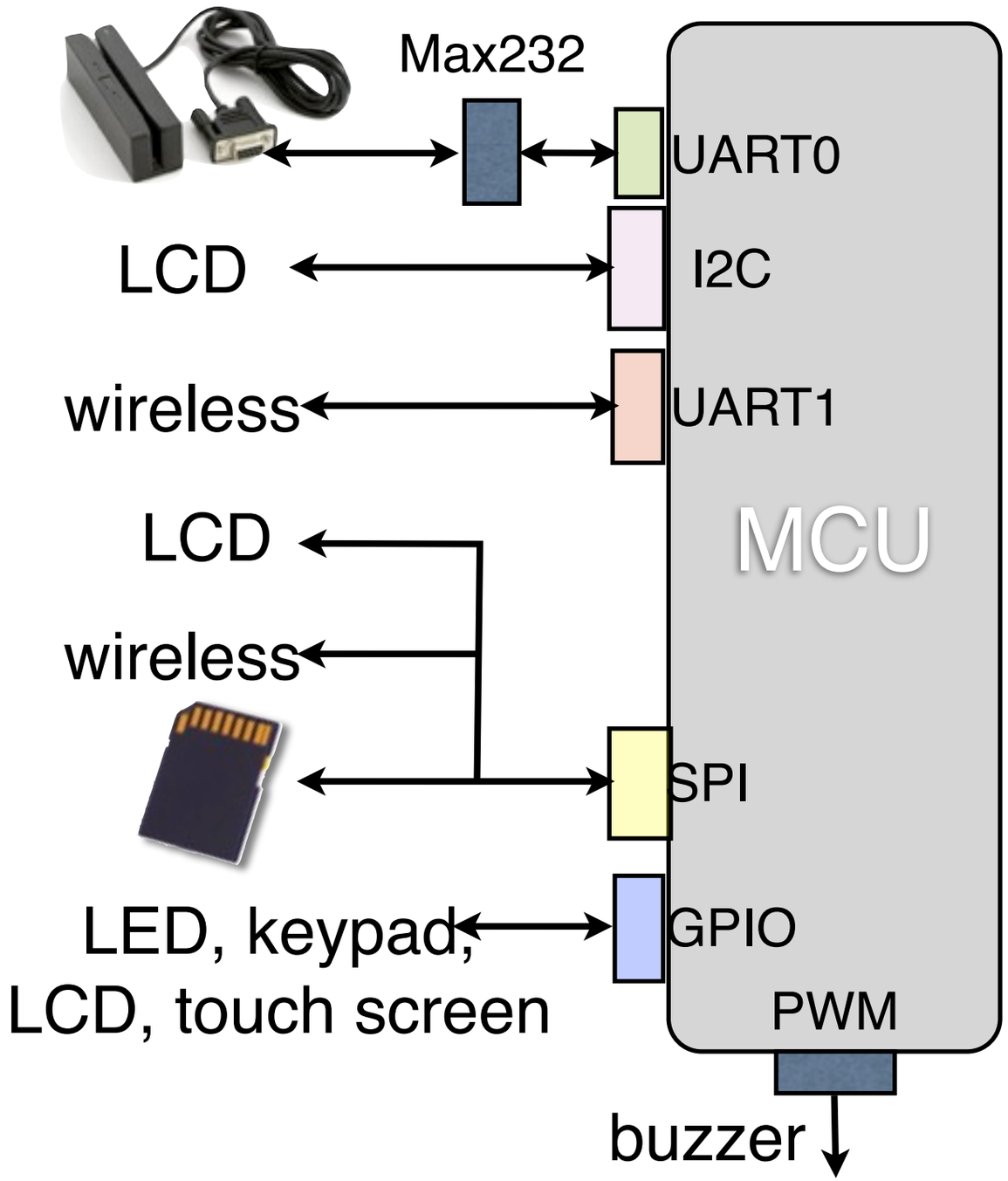
Variety of Handheld Implementations

- Also ID entry
  - Manual Keying ID
  - Cam. barcode reader
- Surrogate GUI for
  - Gateway
  - Card reader
  - Barcode scanner
- Database browser



Interfacing Details

- Wireless module
- LCD
- SD Card or USB stick (data logging)
- LED, keypad, buzzer
- Magstripe Card Reader
- plus Ethernet (built-in)



3. Internet-enabled media bridge

- Playing digital or analog src
- Could control a radio
  - volume, station, band...