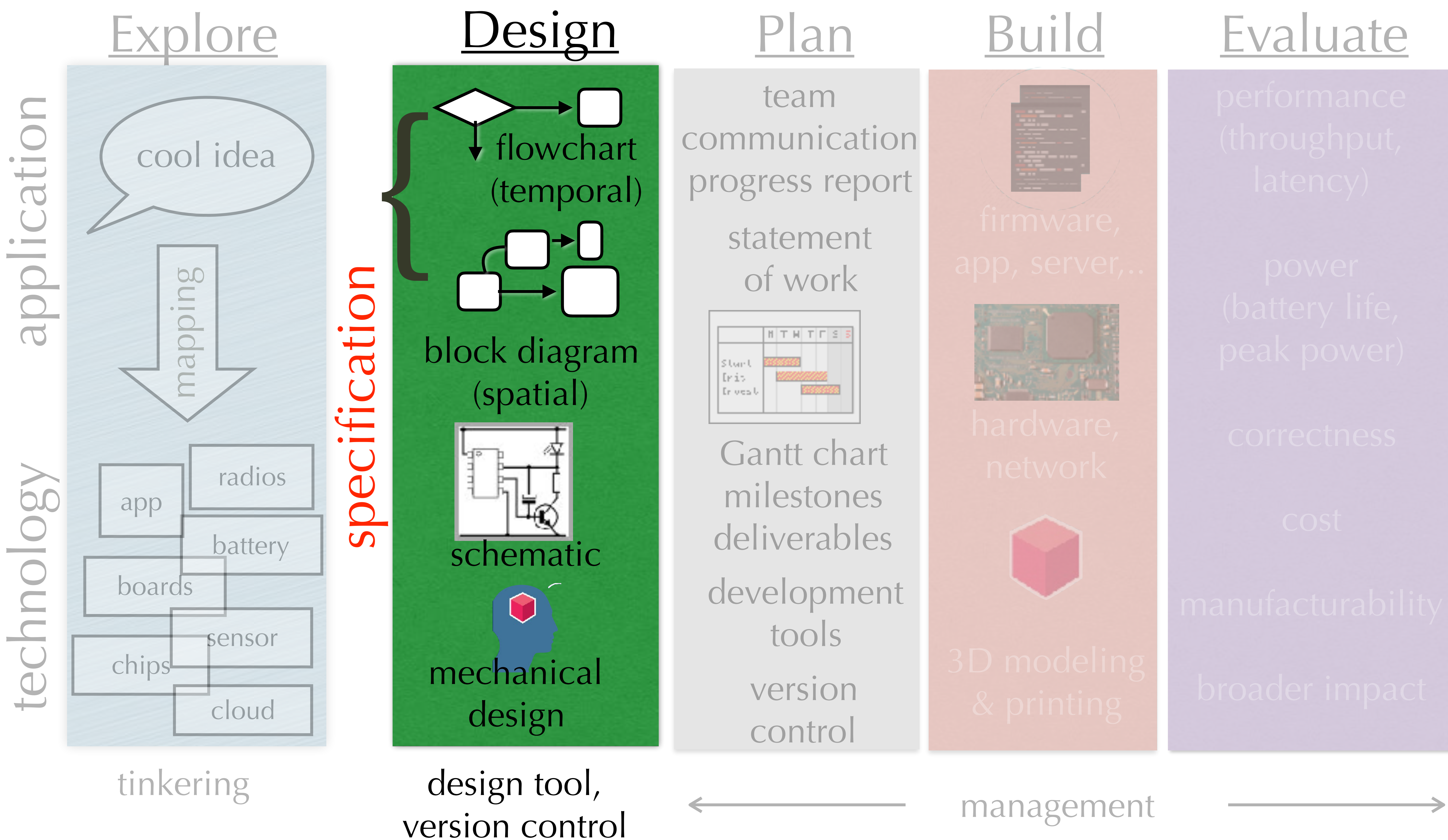
EECS 159A/CSE 181A
MCU-based
System Design

From Spec to Design



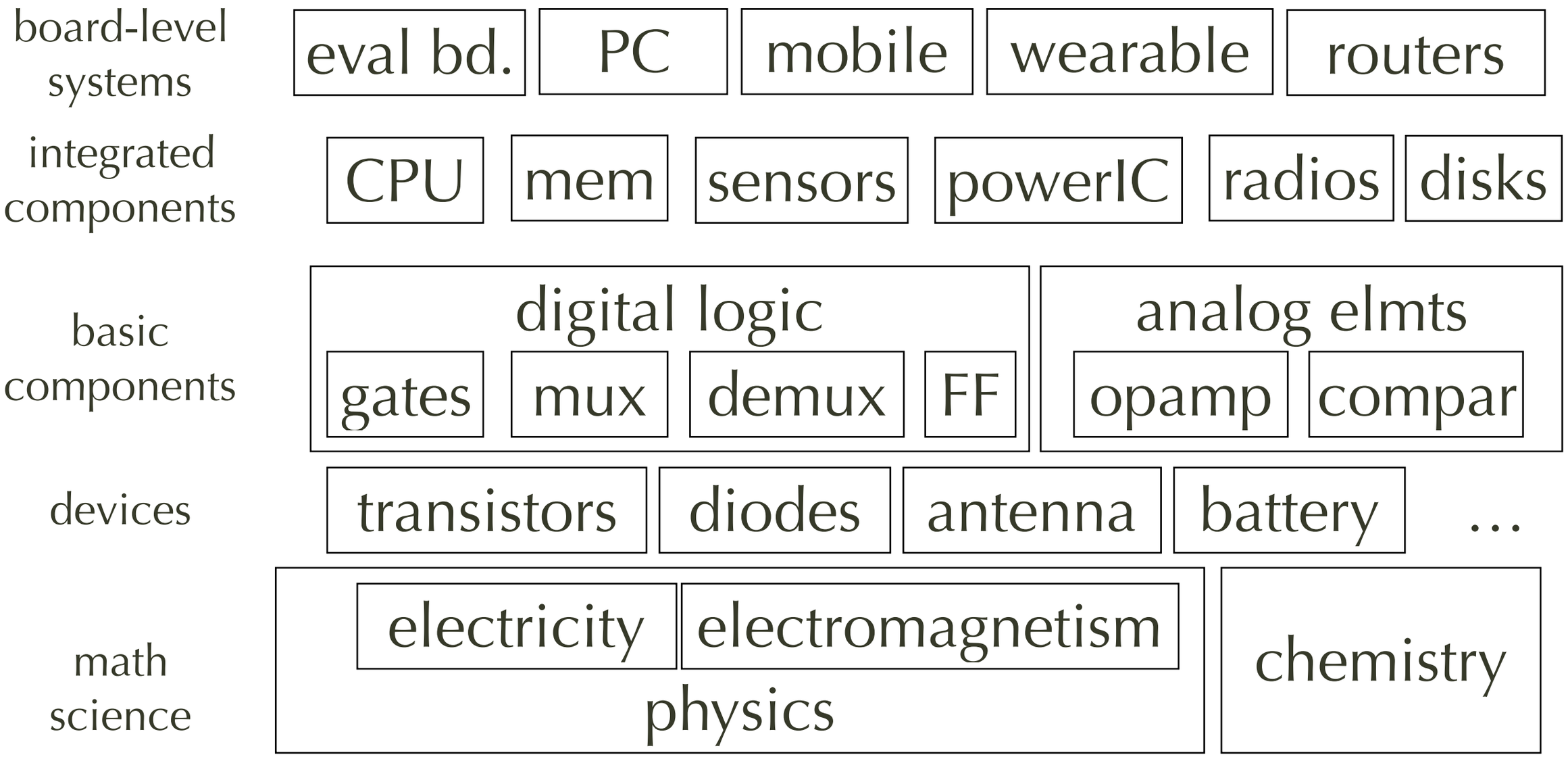
Abstraction-based Design

- Identify the level of abstraction for a design task
  - Abstraction means hiding details
- Why abstraction?
  - Separation of concerns
  - When designing at one level, don't want to have to worry about details at another level
- Why not abstraction?
  - Potentially less efficient, but could be improved by "cross-layer optimization"

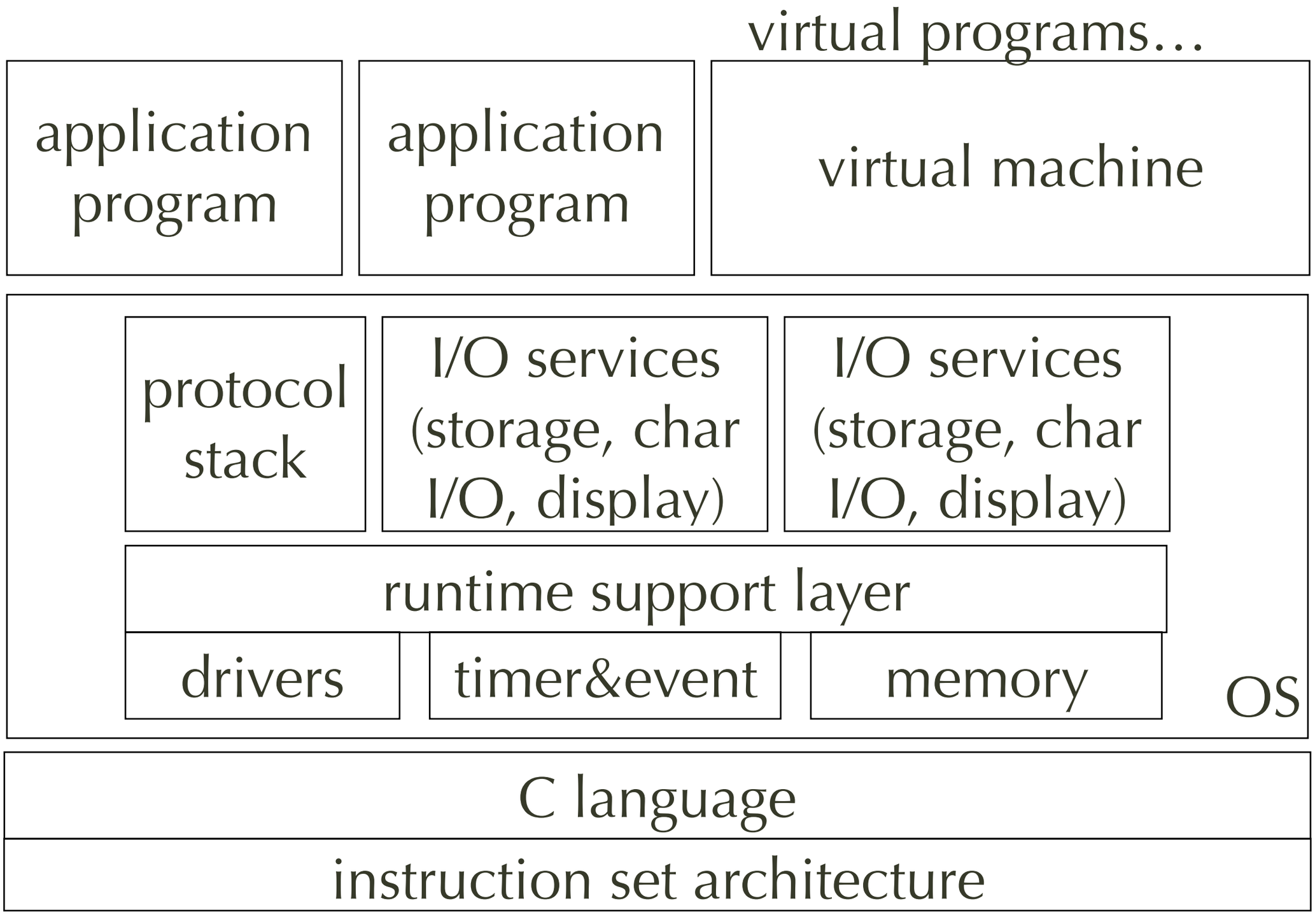
Layers of Concern

- Abstraction layers
  - Application
  - Protocol Stack
  - Runtime
  - Driver
  - Hardware
  - Electrical
- Domains
  - networking
  - communication
  - software architecture
  - hardware architecture
  - power and energy

Layers of Abstraction: hardware



Levels of Abstraction: software



Abstraction Layers in your design?

- Hardware
  - Existing board with add-on peripherals
  - Custom board, custom circuit
- Software
  - Existing operating system vs. “bare-metal”
  - Existing software library vs. own code
  - Existing virtual machine vs. own interpreter

MCU-centric design

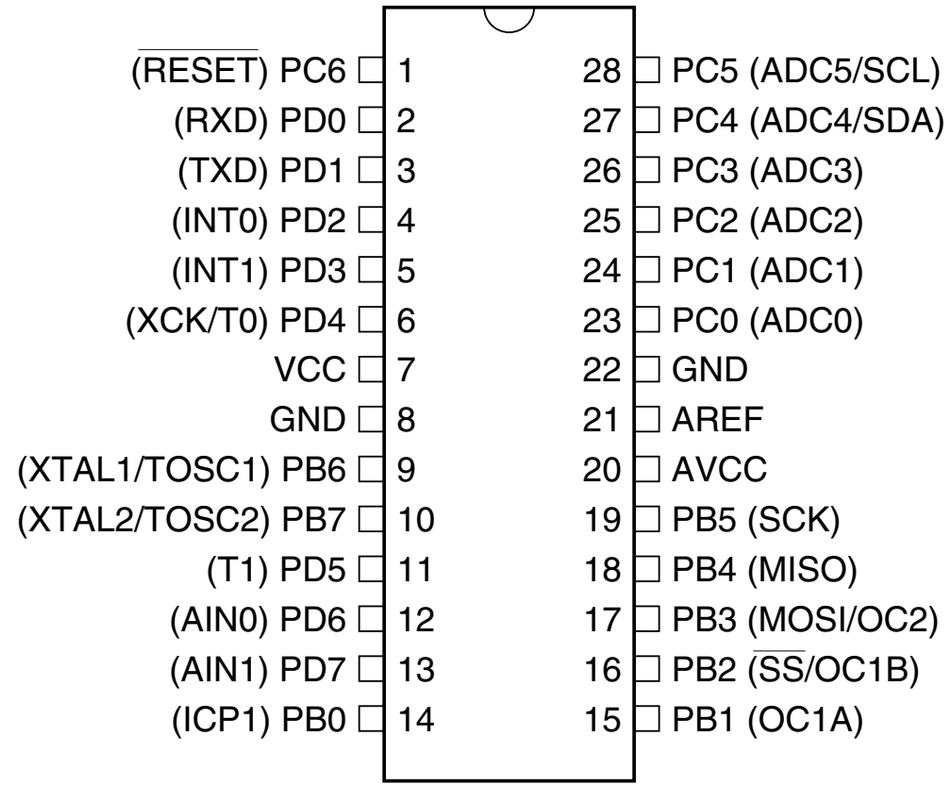
- A convenient starting level of abstraction
  - Centered on a microcontroller unit (MCU)
  - Down to some electrical level
  - Up to several layers of software
- Why MCU?
  - Lets you write software to control hardware
  - Networking or communication capability
  - Possible code reuse

CPU vs. MCU

- CPU ("Central Processing Unit")
  - Processor; main memory is external
  - Bus interface: address, data, control (incl. interrupt)
- MCU (Microcontroller unit)
  - Processor + memory + I/O controllers
  - May have built-in I2C, SPI, UART, USB, Ethernet, DMA, LCD controller, PWM, RF transceivers, ...

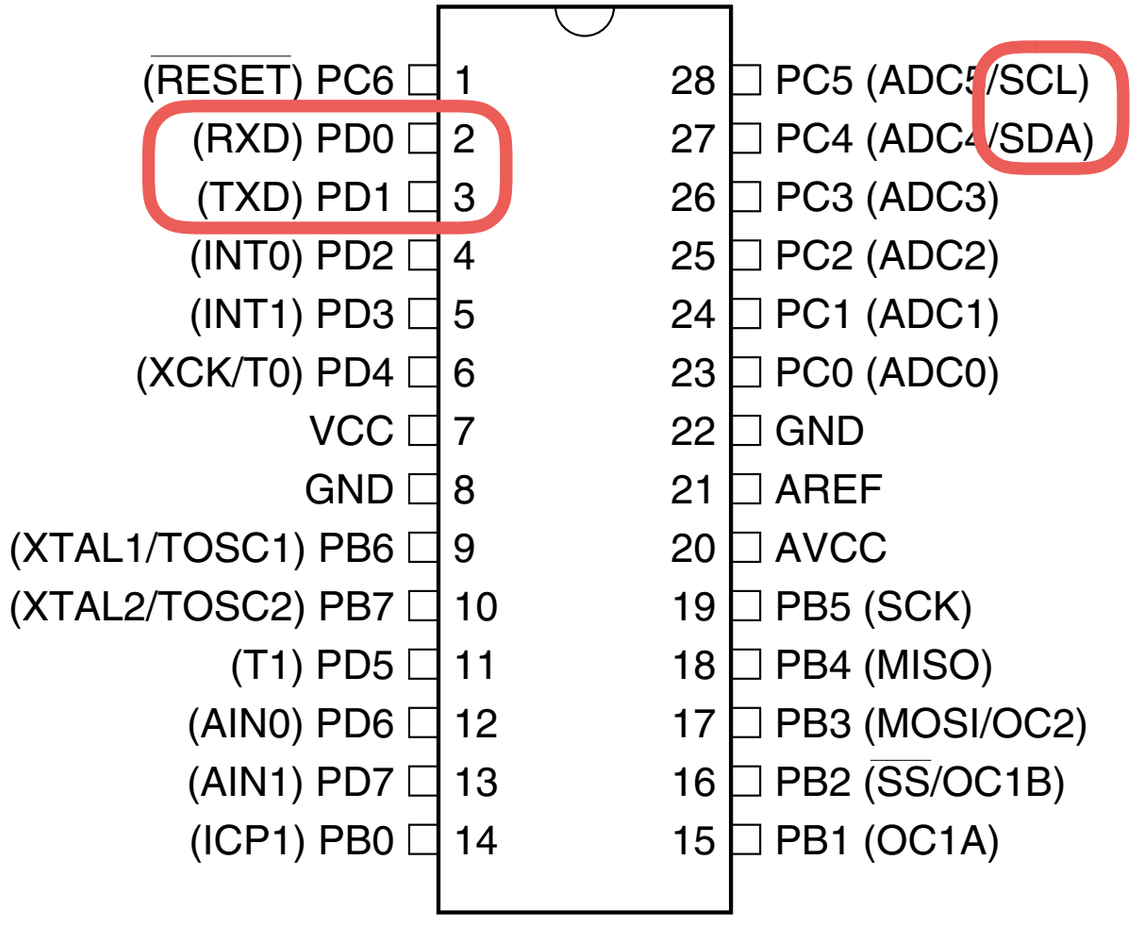
Case Study MCU: ATmega8

- 8-bit MCU by ATMEL
  - 8 KB flash, 512B EEPROM, 1KB SRAM
  - 3 PWM channels
  - 8 channels of 10-bit ADC
  - UART, SPI, I2C, watchdog timer, 23 GPIO pins
  - 0-16 MHz, 4.5-5.5V supply
  - Power: Active 3.6 mA, Idle 1.0 mA, off .5µA
- used in original Arduino board



Resource Allocation on MCU

- Certain pins are overloaded
  - Need to select one out of multiple functions
  - e.g., using serial port (RxD, TxD) => can't use PD0, PD1 as GPIO pins
- Others can be shared
  - e.g., I2C (SCL, SDA) can be shared by multiple I2C devices

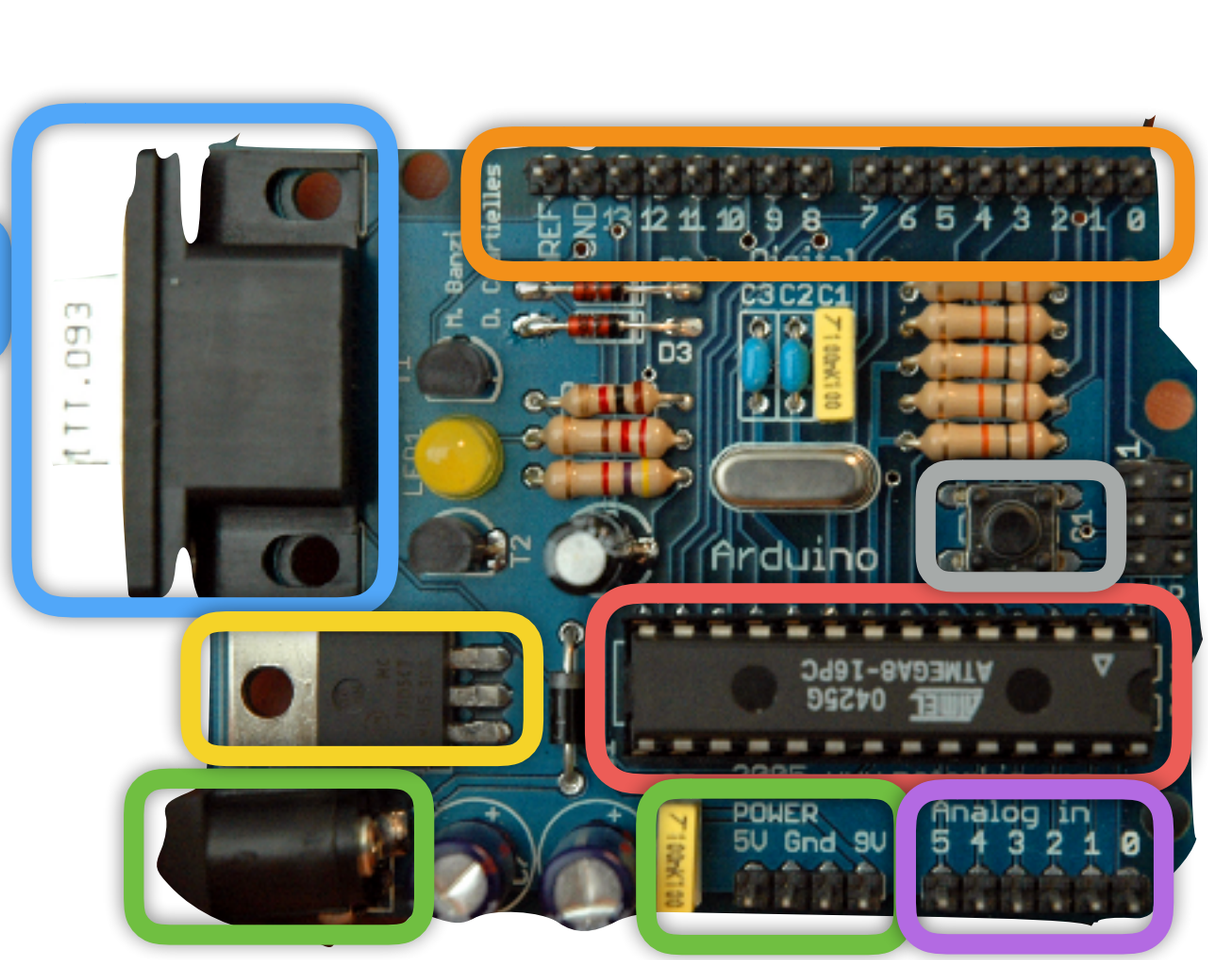


Communication Interfaces

- "Low-level" interfaces
  - UART, I2C, SPI, I2S, ...
  - transfers bytes or sequences of bytes
- "Multi-level" interfaces
  - USB, FireWire, Ethernet, WiFi, Bluetooth, ...
  - Why? because of protocol stack to deal with
    - plug-and-play, connection or pairing,
    - scheduling, routing, arbitration, ...

MCU vs. Board

- MCU is almost always on a board
- example: early Arduino board
  - ATmega8 MCU
  - Connectors for
    - power, ADC, serial, I/O expansion
    - voltage regulator
    - button, and more…



Connecting Peripherals to an MCU

- MCU on a board is just the beginning
- Usually need additional peripherals, if not already on-chip or on-board
  - User input and output (e.g., keypad, display)
  - Sensor (e.g., barcode reader, magstripe reader)
  - Storage (e.g., SD card, USB flash drive)
  - Communication (e.g., WiFi, Bluetooth, Ethernet, serial port, USB, …)
  - Real-time clock

Card Readers

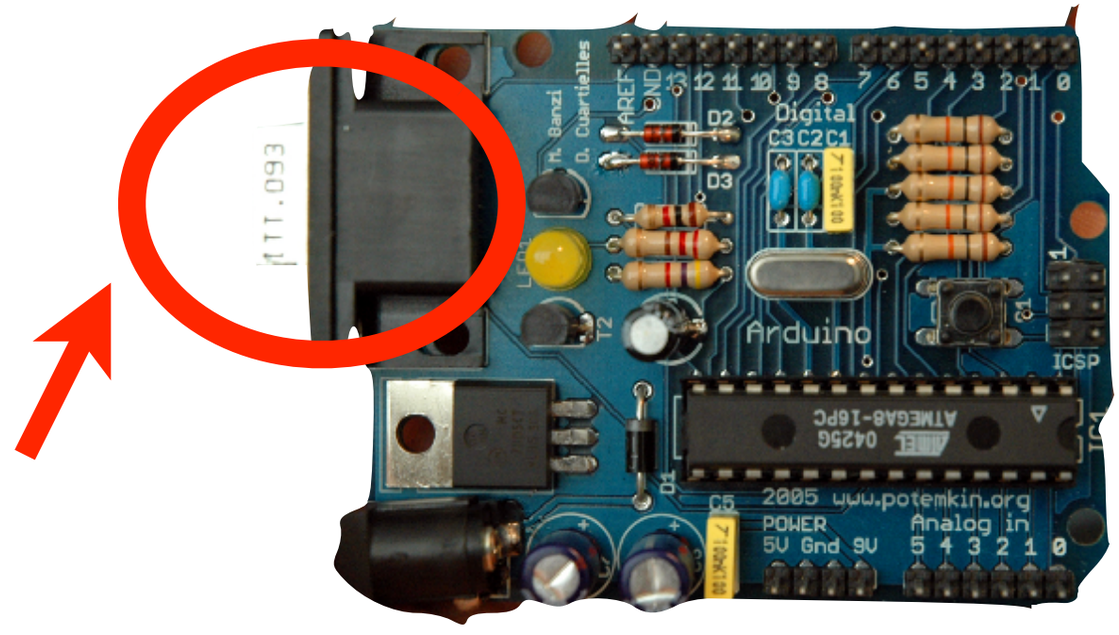
- Card interface: barcode, magstripe, RFID
  - How it decodes the data isn't important, unless you are decoding it yourself
- Computer interface: RS-232 or USB? others?
  - This is what matters the most in MCU
  - Choose MCU with the interfacing capability

Example card reader to MCU interface

a magstripe reader

DB-9 connector
for RS-232 port

an evaluation board
for the M52259 MCU



can the target board
handle the interface?
what baud rate?
what data or command format?

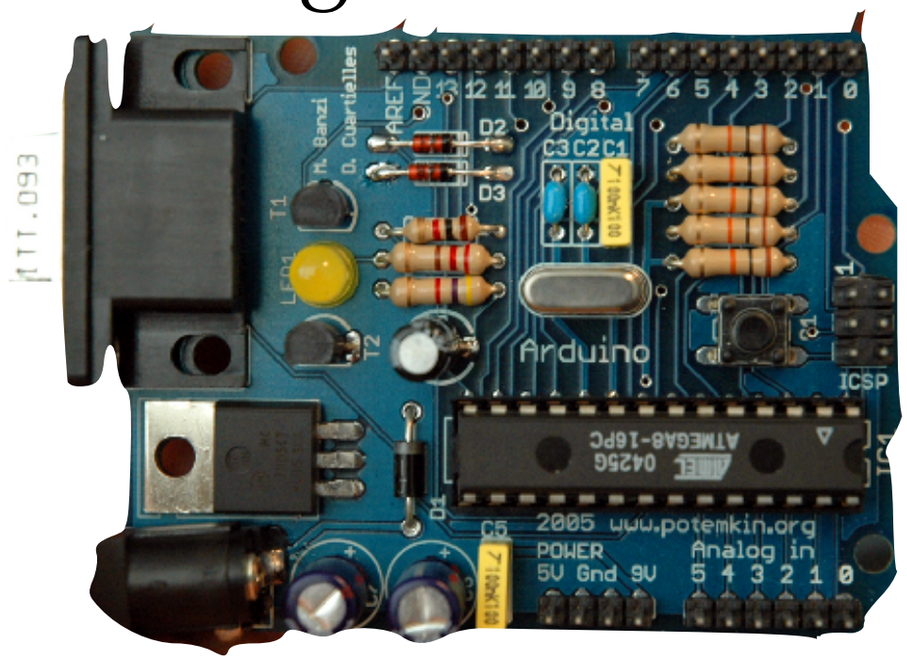
Example: ID card scanner — Sensor

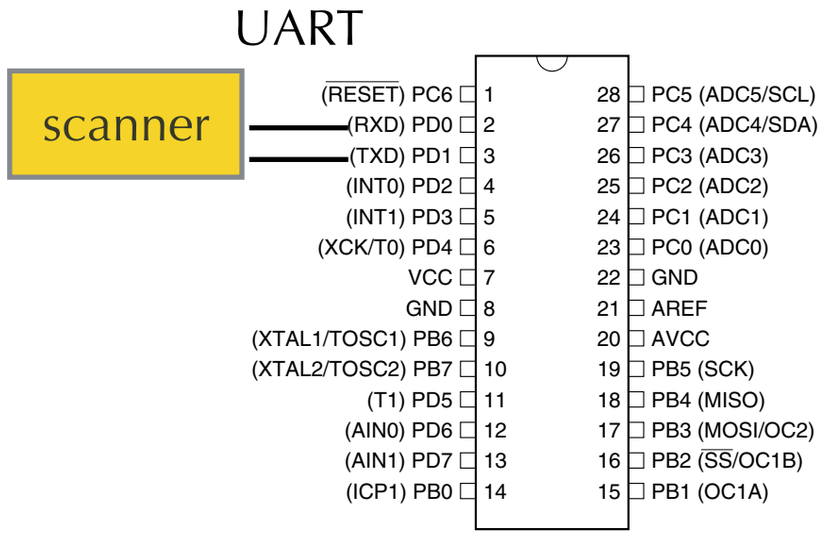
Card Reader

USB

how to connect when the target board doesn't have USB?

target board

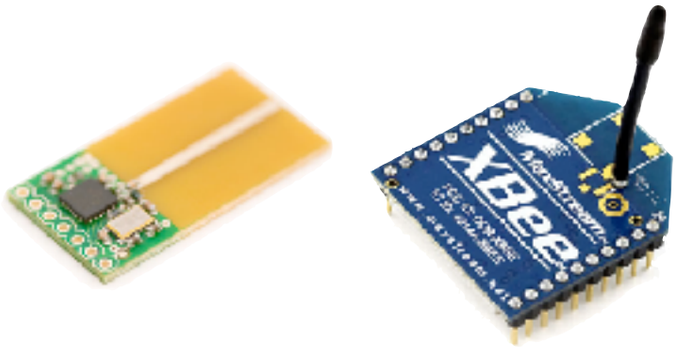




1) serial to USB converter (e.g., FTDI)
2) use another MCU board with built-in USB (master) controller

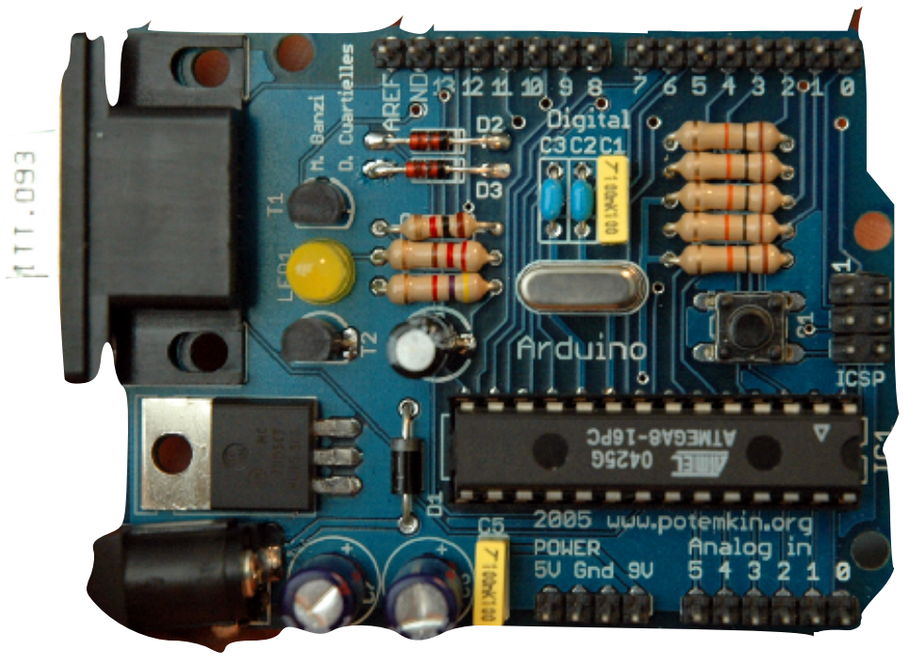
Example: ID card scanner — Communication

Wireless Interface:
on-chip, on-board,
or add-on?

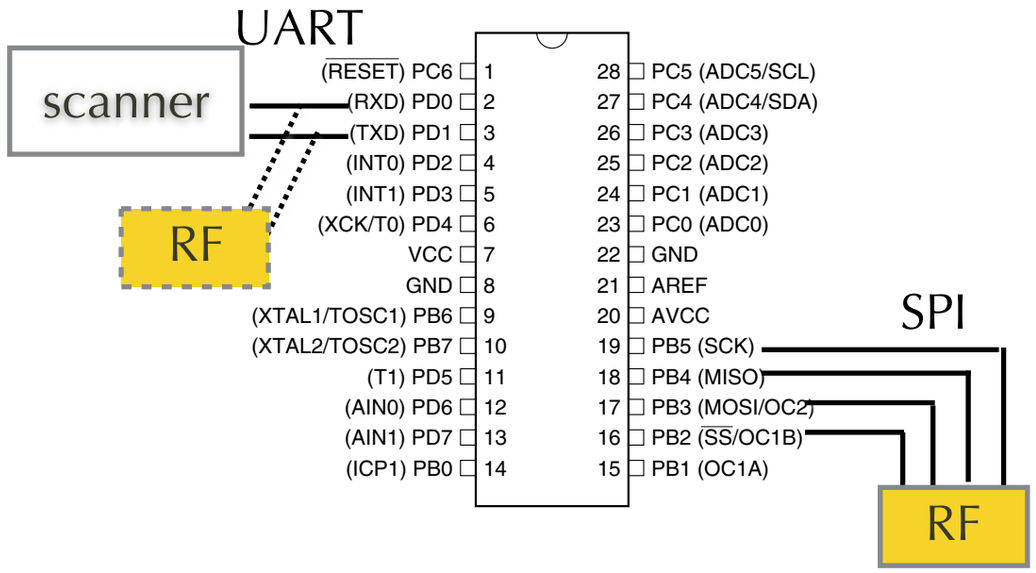


SPI?
UART?

target board



UART => already used by scanner!

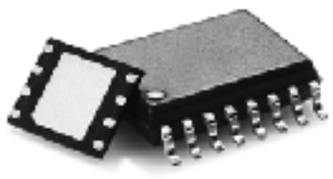


Do you have another "port" on the MCU to connect to this comm. module?

Example: ID card scanner — Flash memory

SD card

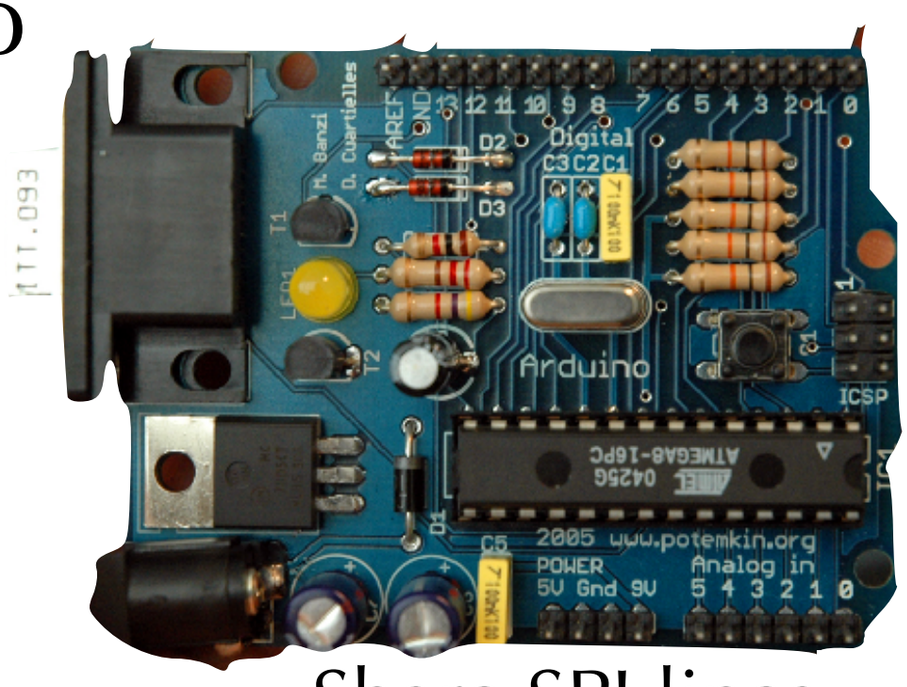
- no SD port on Arduino
- could use SPI mode



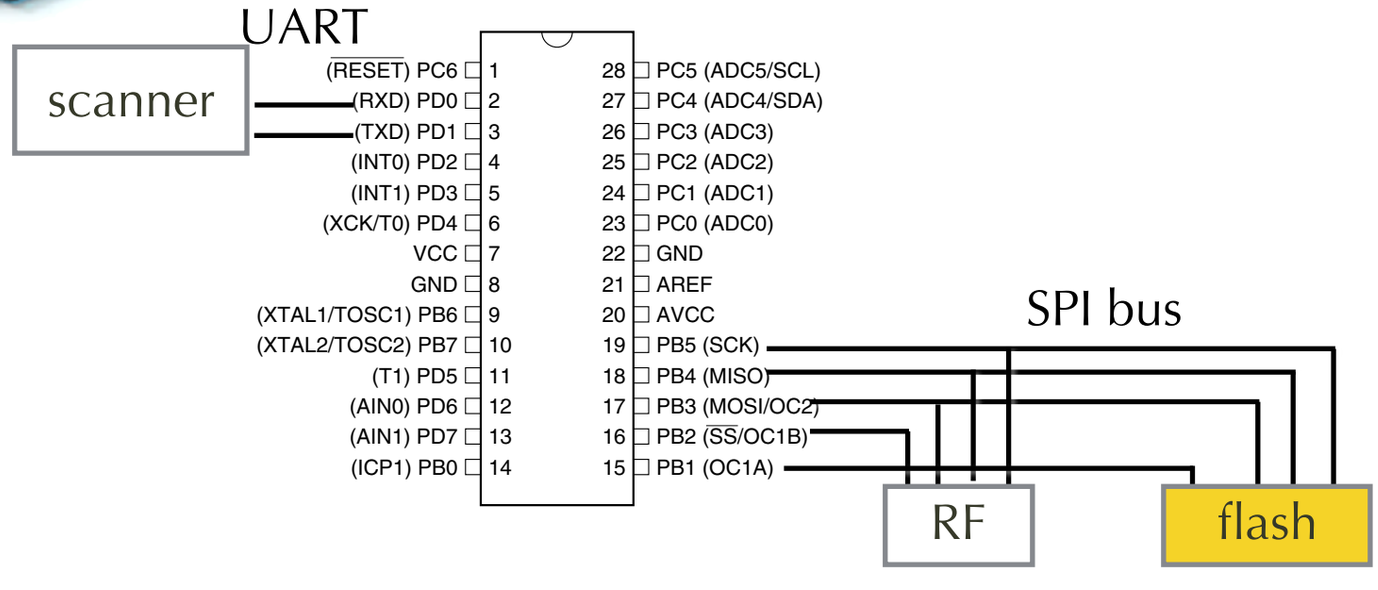
Serial flash (SPI)

USB? probably not

target board



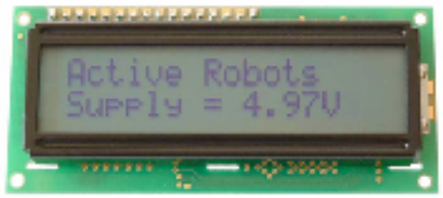
Share SPI lines between RF & flash, separate chip-select

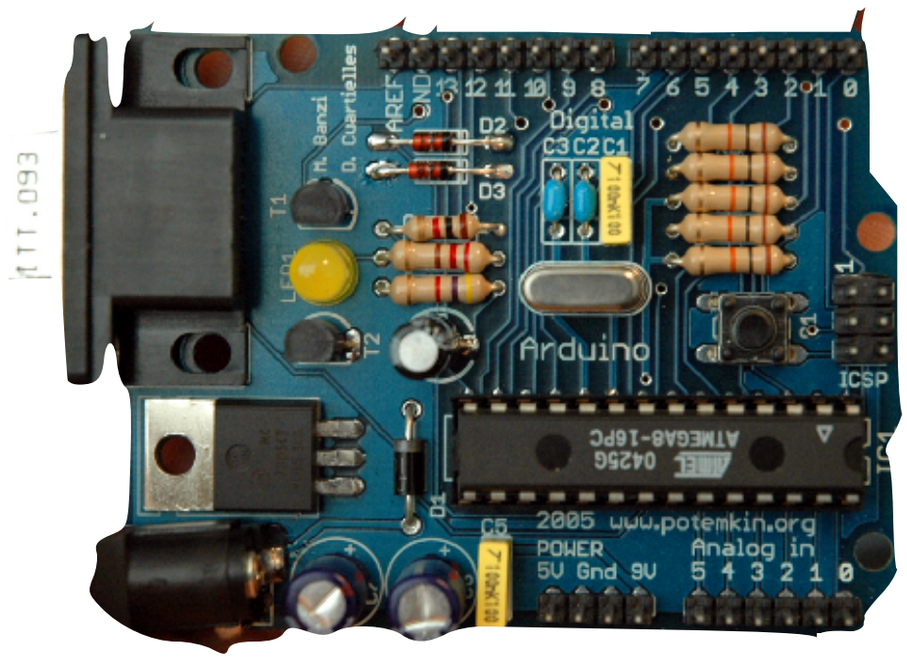


Example: ID card scanner — User I/O

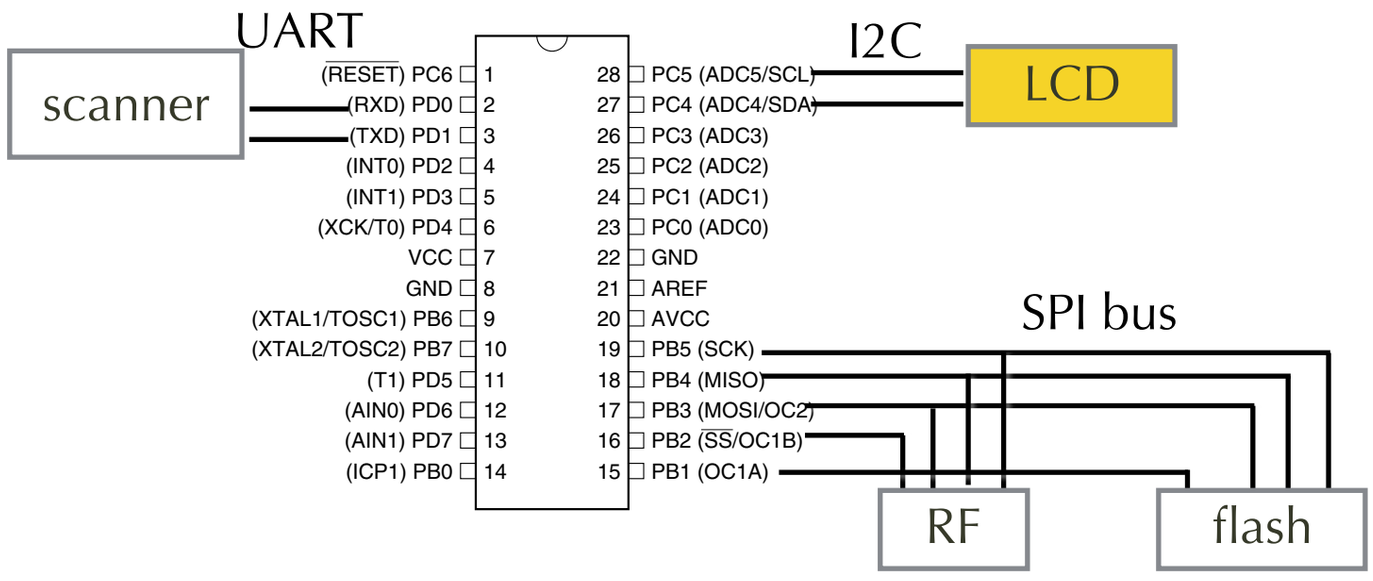
LCD, Keypad

target board





GPIO, I2C

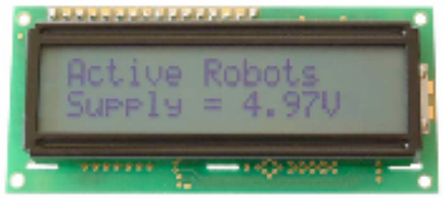


LCD:
Choice of I2C (2 pins)
vs. GPIO (4 or 7 wires)

Example: ID card scanner — User I/O

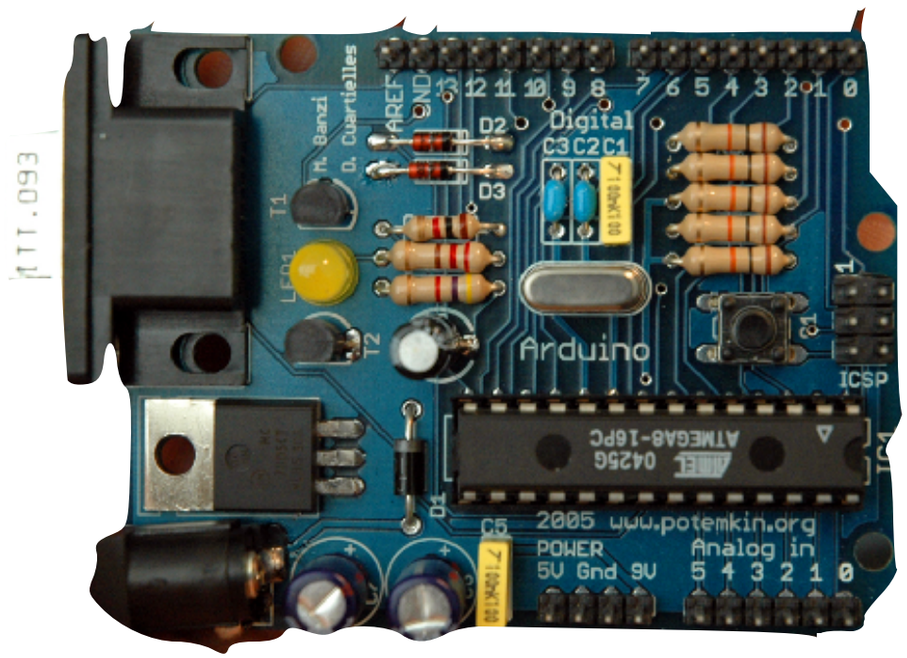
LCD, Keypad

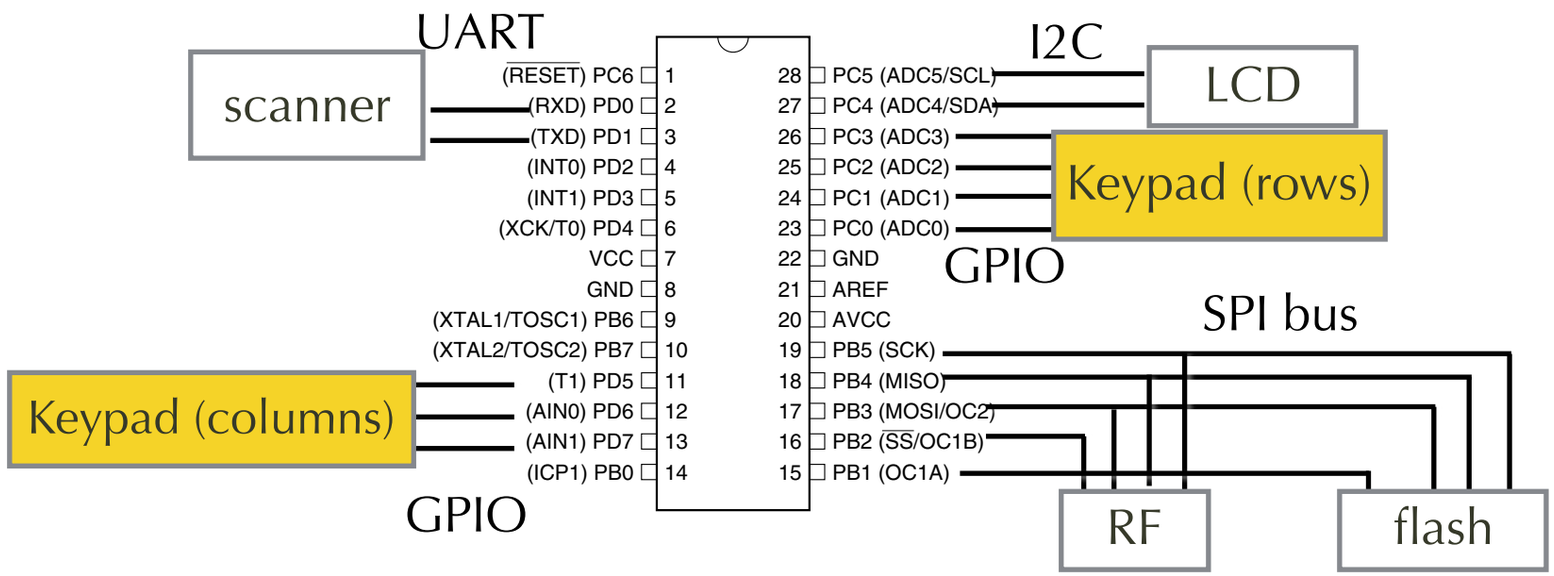
target board



GPIO, I2C

4+3 GPIO



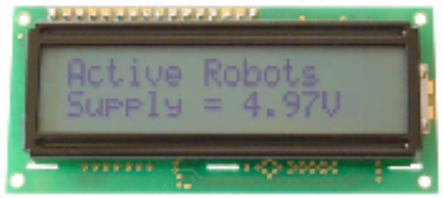


row-column scanning

Example: ID card scanner — User I/O

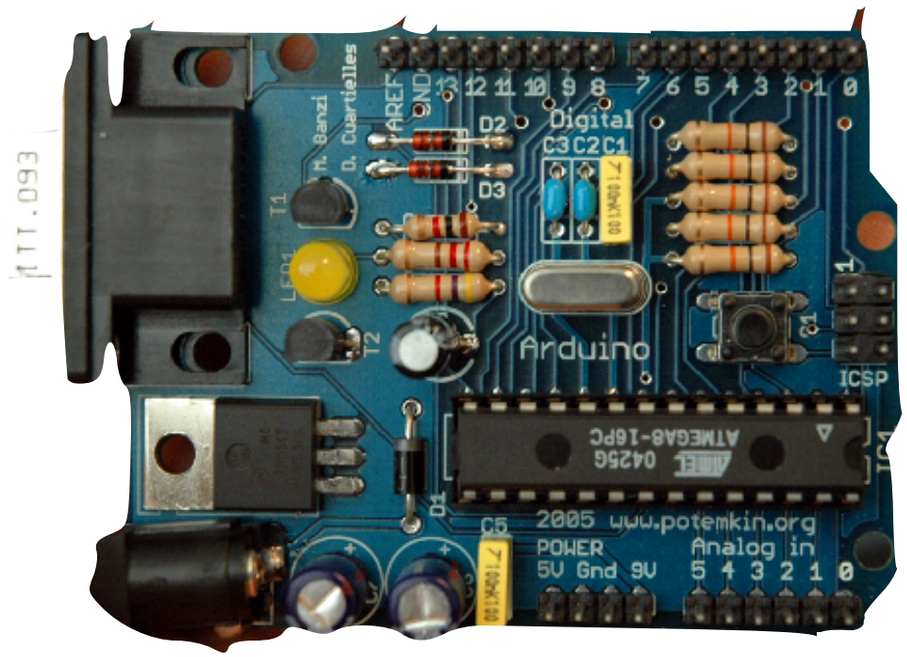
LCD, Keypad

target board

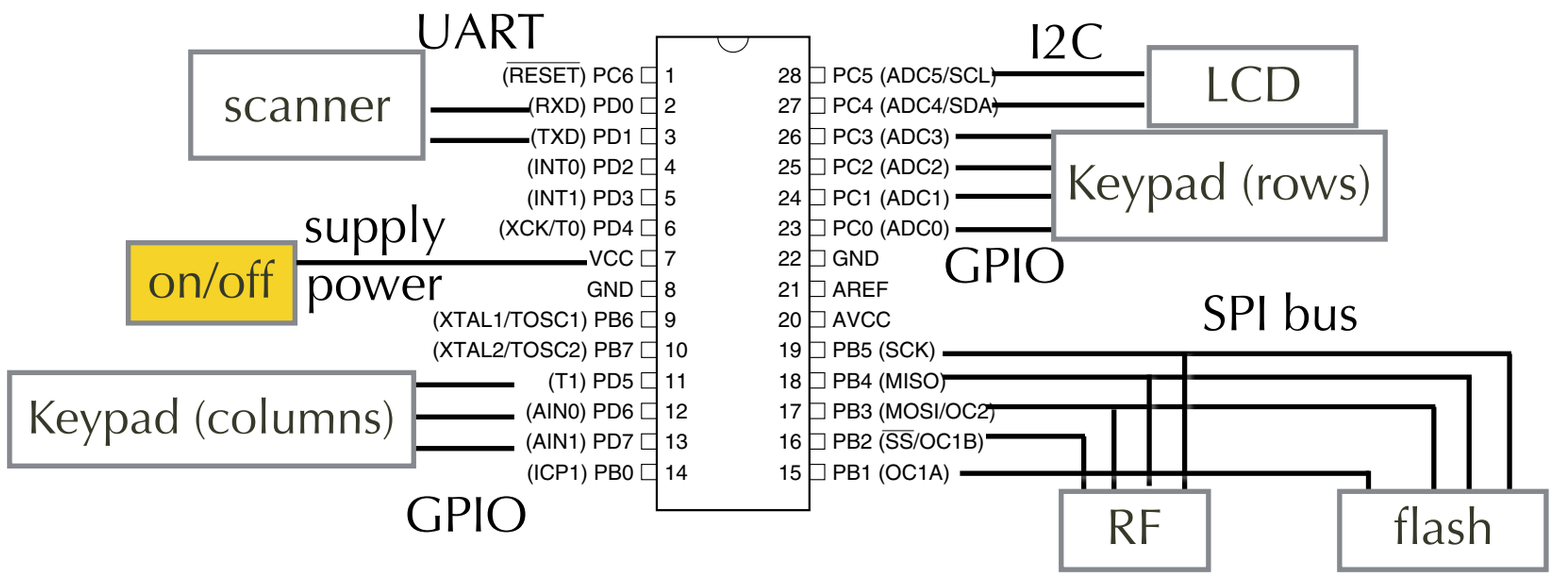


GPIO, I2C

4+3 GPIO



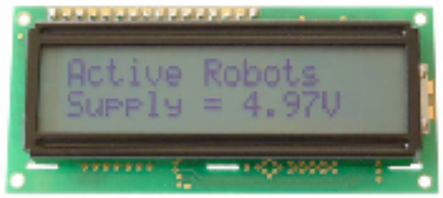
power on/off



Could be a SPST switch, cuts off power completely; or could be a triac, long-hold to power on/off. (e.g., LTC2954)

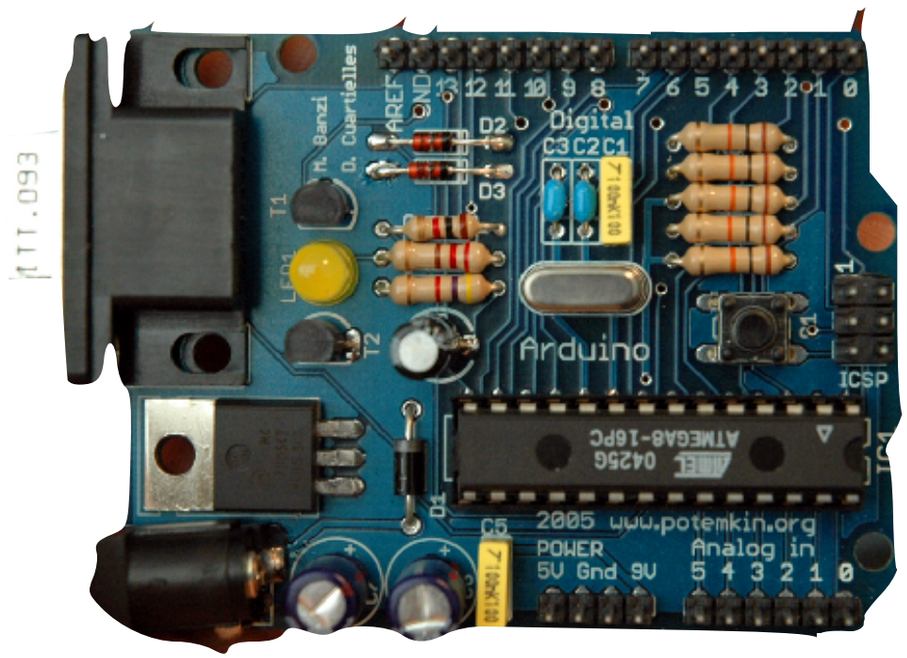
Example: ID card scanner — User I/O

LCD, Keypad

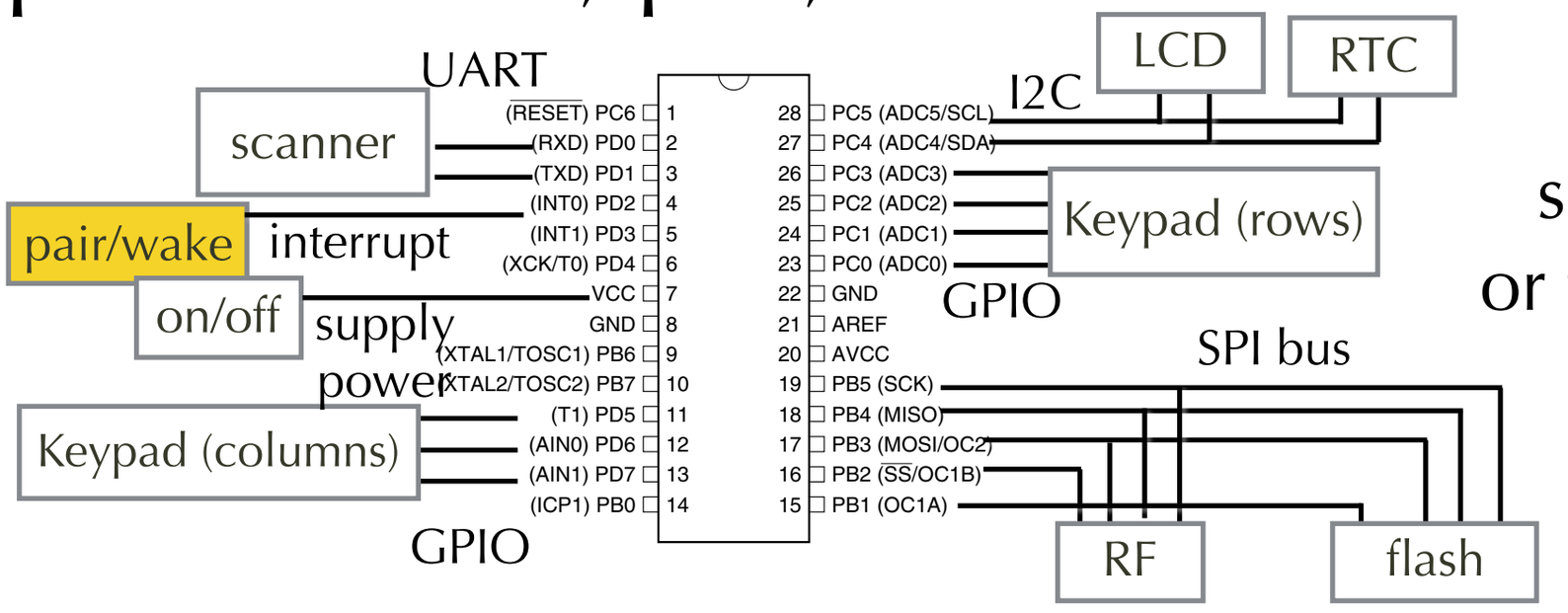


GPIO, I2C 4+3 GPIO

target board



power on/off, pair, wake



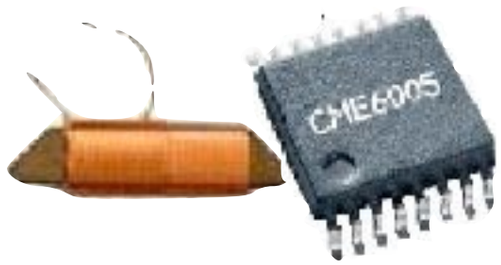
should be interrupt driven,
or the RF module may already
handle button & LED

Example: ID card scanner — Real-time Clock

Battery-backed RTC

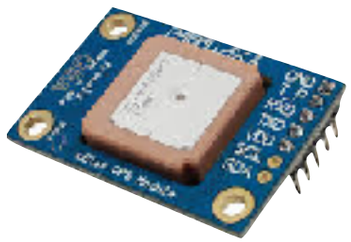
I2C

atomic time (WWVB) broadcast



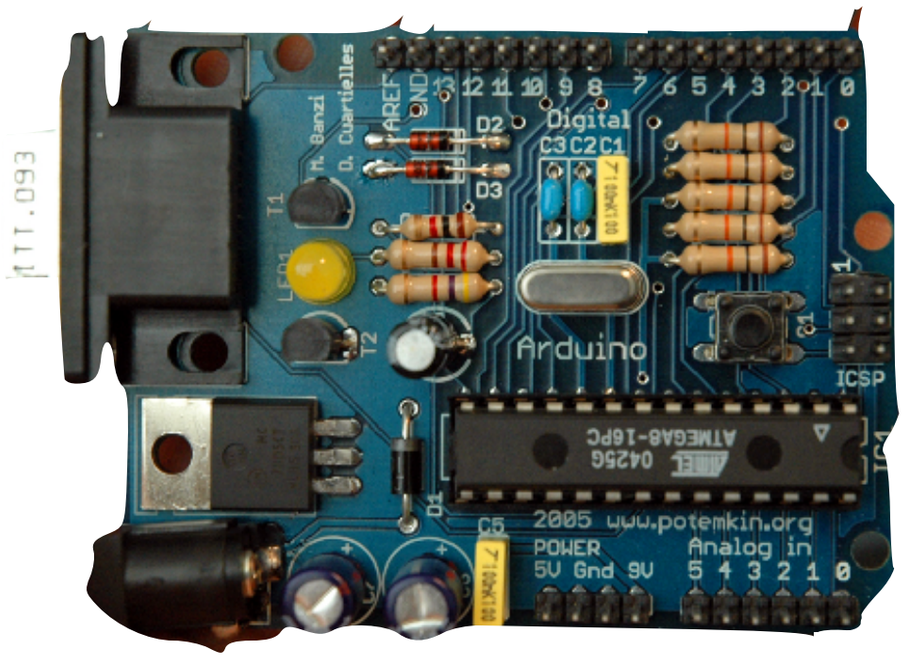
bit-serial time code

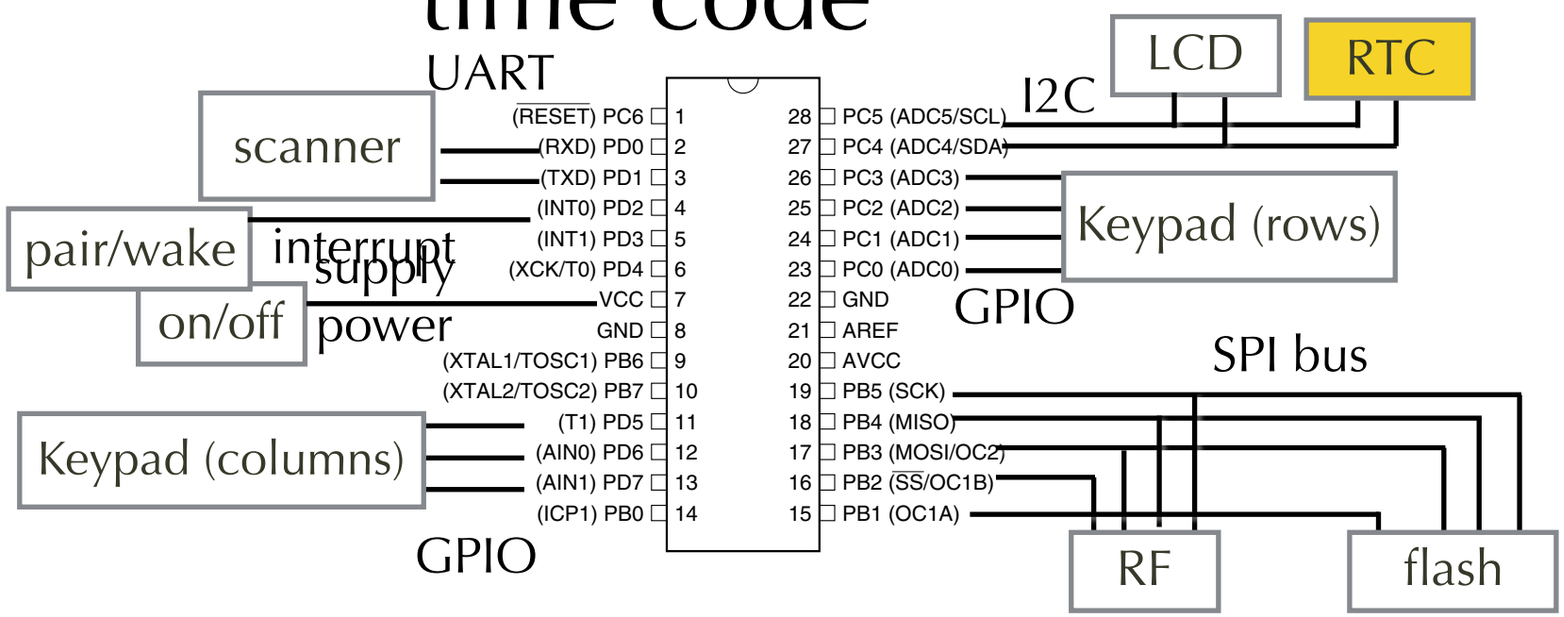
GPS receiver



UART, I2C

target board



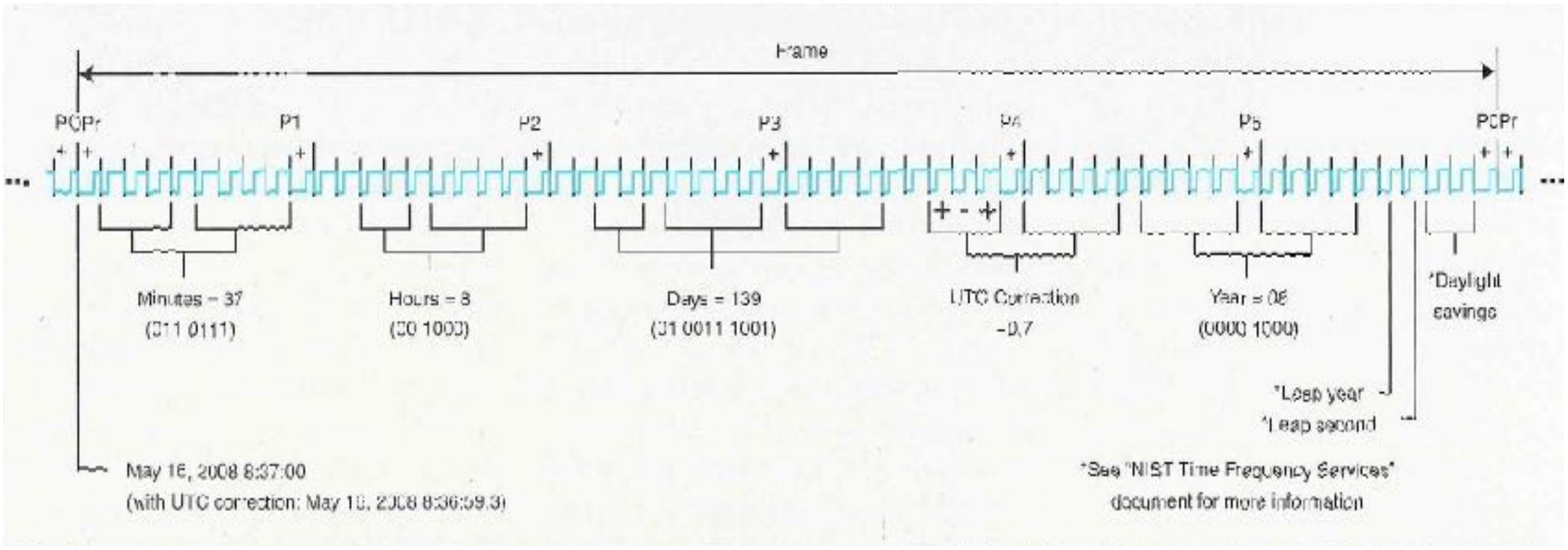


Atomic time broadcast

[Time encoding of WWVB]

- Binary “0” : 0.2s (Low), 0.8 s(High)
- Binary “1” : 0.5s (Low), 0.5s (High)
- Position Maker : 0.8s (Low), 0.2s(High) – indicate the start of frame

=> it takes 60 sec to send of get one frame using falling edge capture of MCU.



Example: ID card scanner — Power Source

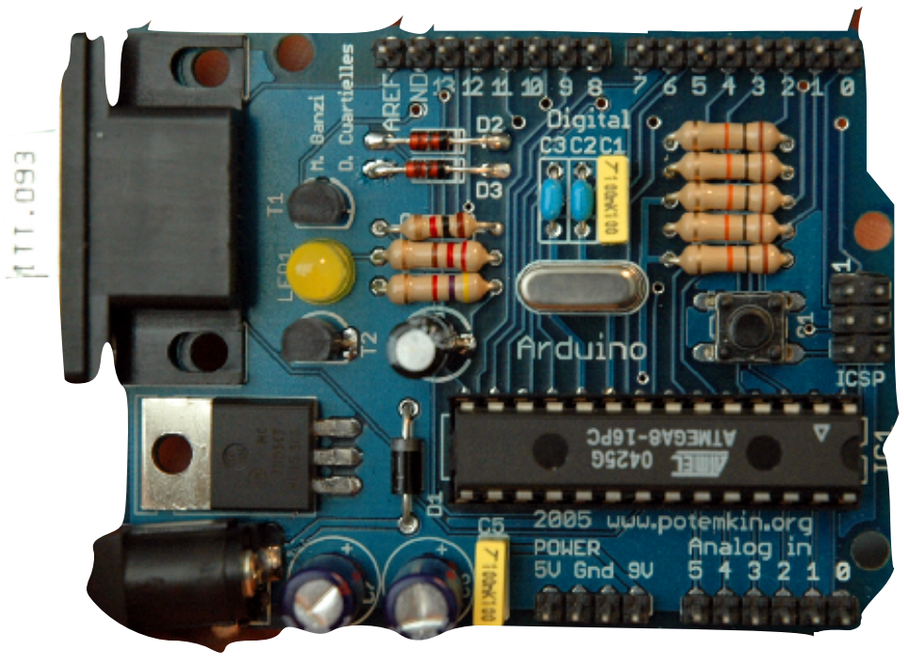
AC adapter?
battery pack?
Rechargeable?

4 AA =>
6V or 4.8V

Li-Po =>
4.2V

Coin-cell
=> 3V

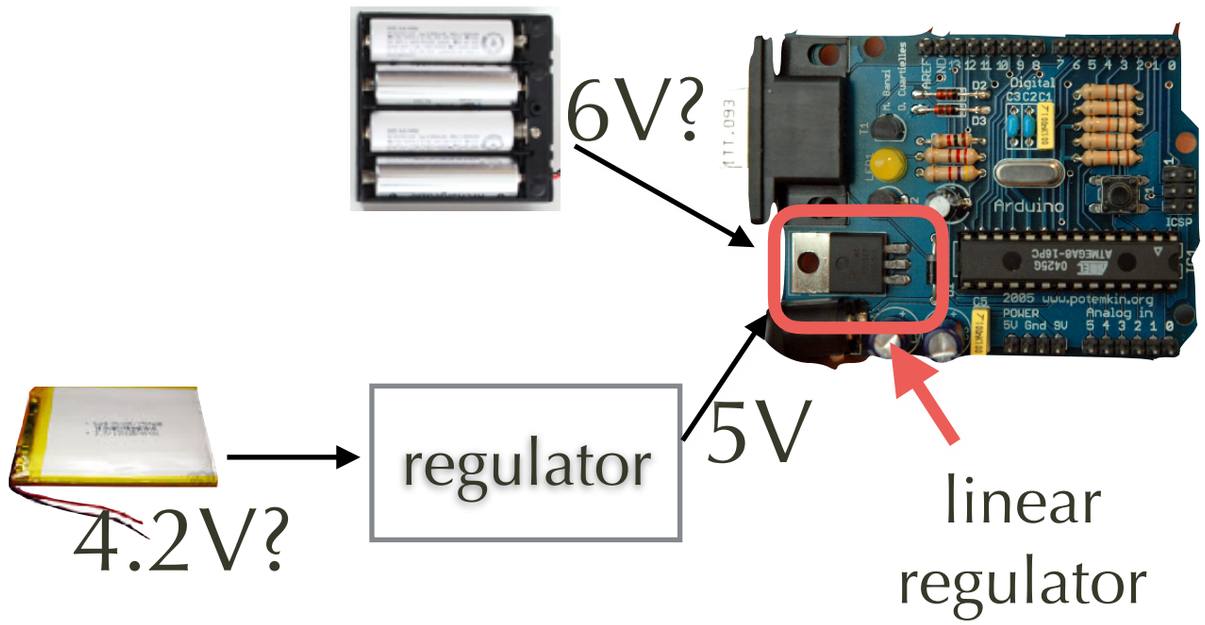
target board



Does the supply support the required voltage and can output sufficient current?
How many mAh if battery?

Resolving Differences in Interfaces: Supply Power

- Linear regulators
  - caps voltage at a max
  - very stable power
  - lower efficiency (~60%)

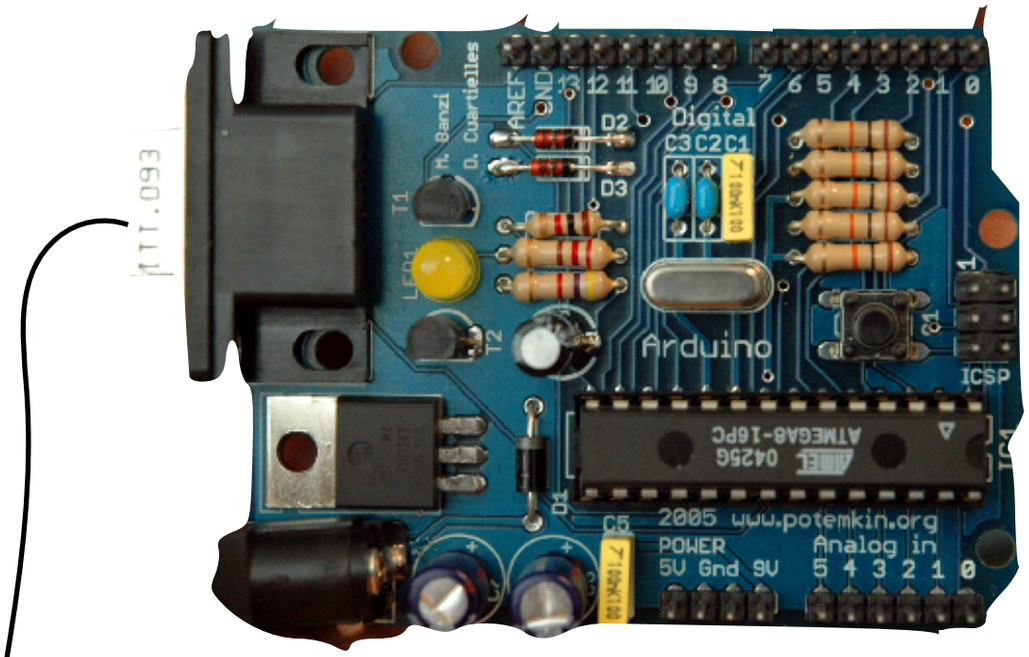


- Switching regulators, aka DC-DC converters
  - buck, boost, or buck-boost converters
  - higher efficiency (>90%)
  - slightly noisy power (ripples)

Example: ID card scanner — Firmware & Debugging Interface

- Usually JTAG
  - flashing, stepping, …
- Bootloader
  - serial port (ISP)?
  - USB? SD Card?
  - over wireless?
- Debugging
  - printf? LED?

target board



Summary of peripheral interfaces

Card Reader, RFID?

RS232?
USB?
PS2?

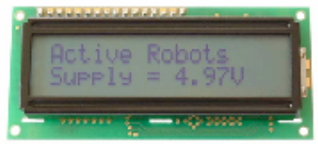
MCU

UART?
SPI?

Wireless: ext or built-in?

SD Card, USB stick?

SPI?



Keypad? LCD?

SPI? I2C? GPIO?

Battery?
On-off switch?

Other ports for debugging use? buzzer? LED?

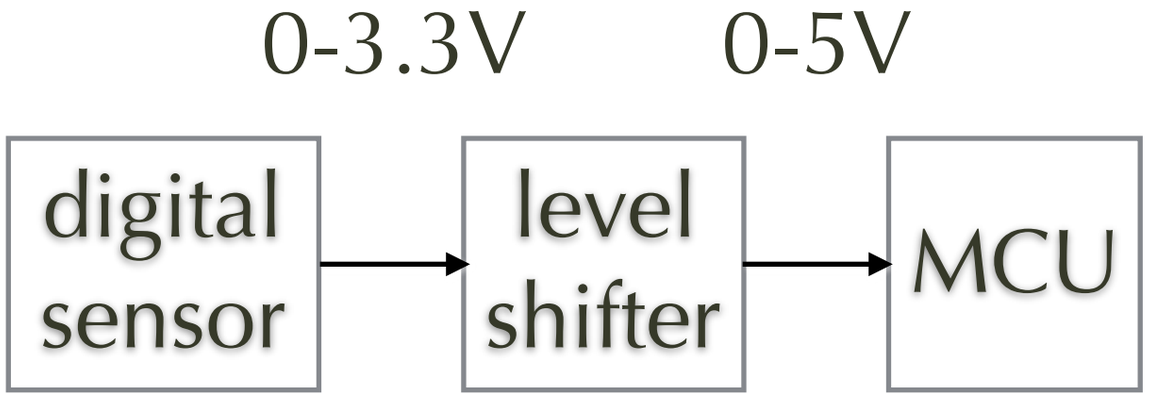
Other buttons? Reset? Wake up?

Feasibility

- Enough I/O resources (pins)?
  - digital? analog? serial port? bus? interrupt?
- Enough power? (voltage, current)
  - Active mode, sleep mode
- Enough memory?
  - Code flash?
  - Data RAM?

Resolving differences in signal voltage & current

- Voltage level translator
  - unidirectional
  - bidirectional
  - direction-controlled
    - logic level
- Buffer driver
  - driving voltage and current on a (bus) line or load (e.g., LED)
  - Inverting or noninverting
  - UART: CMOS to RS-232 (-25V to +25V)



What if there are not enough GPIO pins?

- Use a multiplexer or demultiplexer
  - Select 1 out of *N* signals to input or output
  - Requires $\log_2 N$ signals to select
    => not always feasible
  - Can be Digital or Analog Mux or Demux
- Or just pick another MCU w/more pins
  - Could increase size, cost, power consumption
  - May need to switch to another MCU family

Requirements vs. Constraints

- Your (hardware) design may fulfill the required functions, but…
- Does it satisfy all constraints?
  - Enough energy? (battery life)
  - System weight? Size?
  - Latency? Performance?
  - BOM Cost?
- If not, you may need to start over with another MCU

Consider other MCU boards when

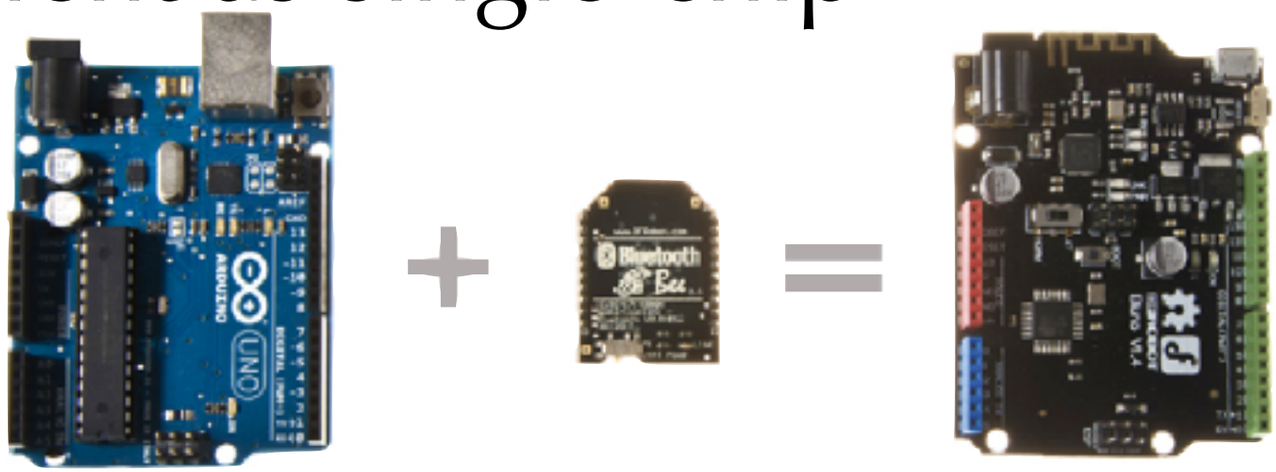
- Need to meet requirements
  - I/O and communication capabilities
  - processing capabilities
  - memory capacities
  - power consumption
- Consider other constraints
  - cost of board
  - physical size and weight

Existing MCU Boards

- Reusable hardware and software
  - Hardware: circuit board with MCU, sensor, actuator,
  - Software: runtime support, library, tools
  - Reusable over a range of applications
- Why existing MCU boards ("platform")?
  - Avoid re-inventing the wheel by customizing
  - A lot of details to get right (e.g., RF, power)
  - Better reuse of code and modules

Arduino - Bluno

- Arduino Uno plus BLE coprocessor
  - MCU sends AT commands to control BLE
  - Uses Arduino software development environment
  - Can add other Arduino shields
- Not as power-efficient as single-chip solution



BeagleBoard

http://beagleboard.org

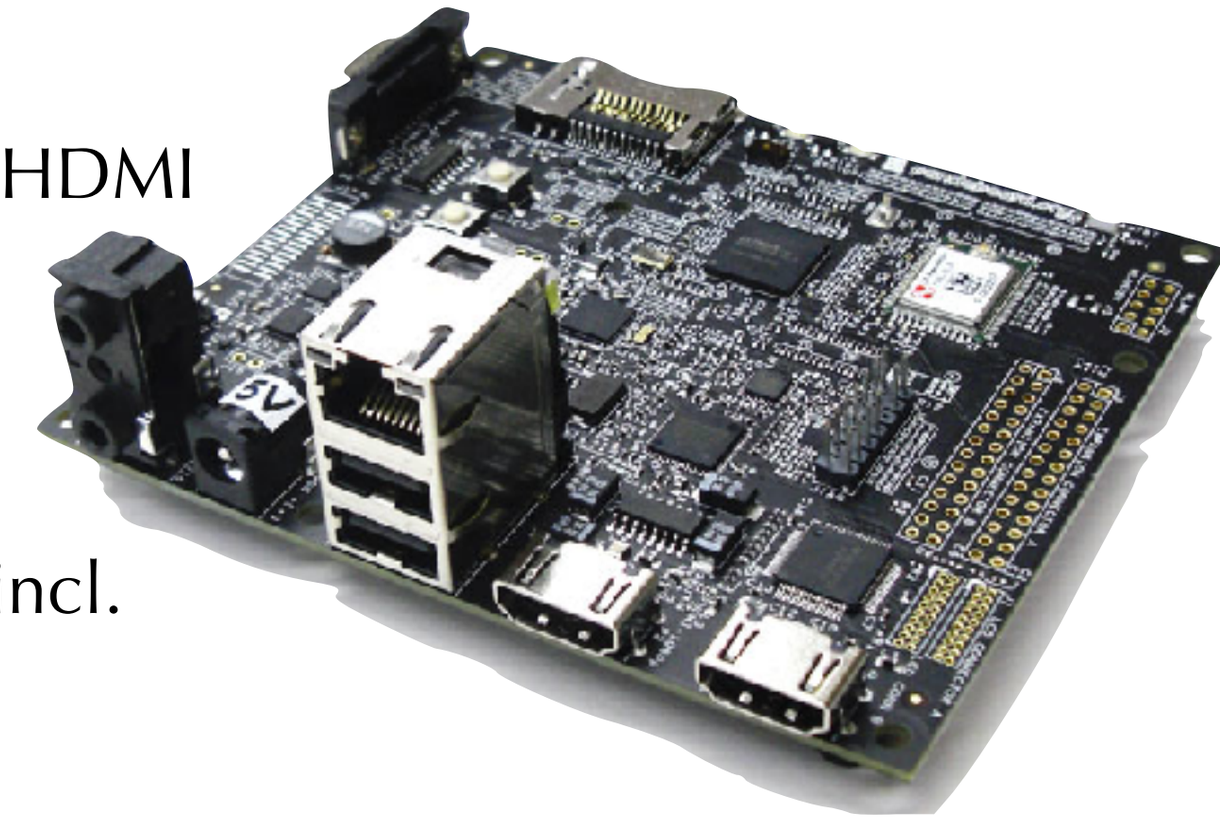
- Original BeagleBoard
  - TI OMAP3530 (ARM Cortex A8)@720MHz
  - Ethernet, USB, HDMI, SD card

- BeagleBoard Black (~$55)
  - TI AM3359 (Cortex A8)@1 GHz
    512MB DDR3 RAM, 4GB flash storage
    600-1000 mW active, 5-7mW sleep
  - Ethernet, µUSB, µHDMI, µSD

PandaBoard

http://pandaboard.org

- OMAP 4 or OMAP 5
  - Dual-core ARM Cortex A9 @1GHz or 1.2 GHz
  - 1GB RAM, Full HD video, 2 HDMI
- Network interfaces
  - Wi-Fi: 802.11b/g/n
  - Bluetooth BR/EDR and BT4 (incl. BLE)
  - Fast Ethernet
- SD card



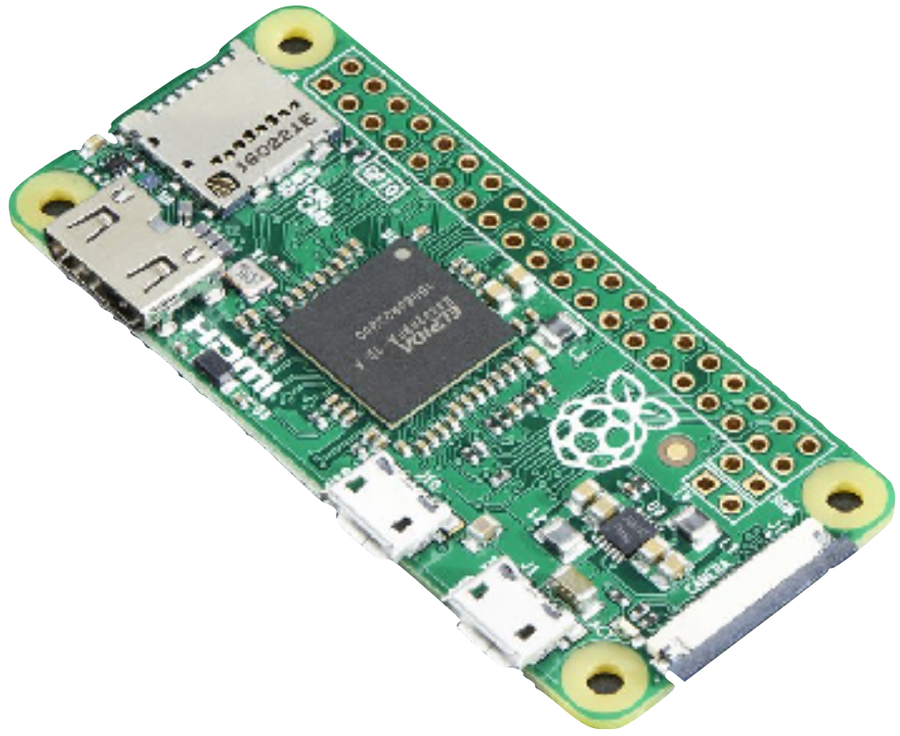
Raspberry Pi

http://www.raspberrypi.org

- MCU: Broadcom BCM2835
- RAM
  - 256MB SDRAM (earlier),
    512MB in Model B (2014)
- Ethernet, 2 USB
- HDMI out, audio jack
- 4GB eMMC flash

Raspberry Pi Zero

- Single-core 1 GHz processor chip
- Micro SD card slot
- Mini HDMI port
- Two micro USB ports (one for power, one for USB),
- 512MB of RAM
- 65 x 31 x 5 mm^3
- 9 grams

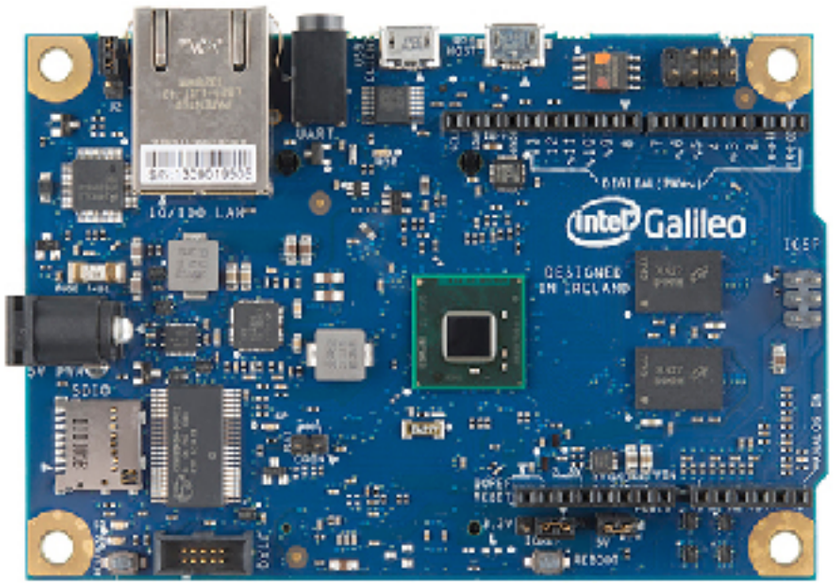


~$5

Intel Galileo

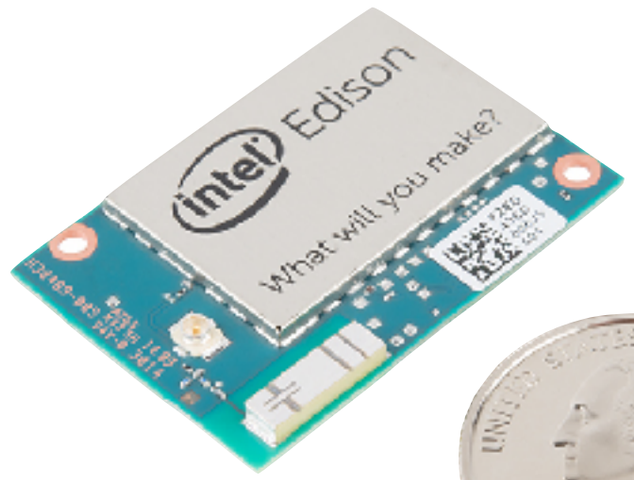
http://www.intel.com/content/www/us/en/do-it-yourself/galileo-maker-quark-board.html

- Intel's response to Raspberry Pi
- Arduino Certified
  - Can use Arduino Shield 3
  - Supports Arduino SDK
- MCU: Quark X1000 SoC
  - essentially Pentium core, 400 MHz
  - 10/100 Ethernet, MicroSD, USB, RS-232
- 8MB NOR flash, 256 MB DDR3 RAM
- No HDMI…

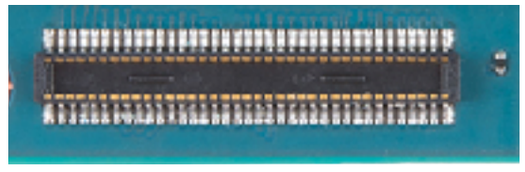


Intel Edison

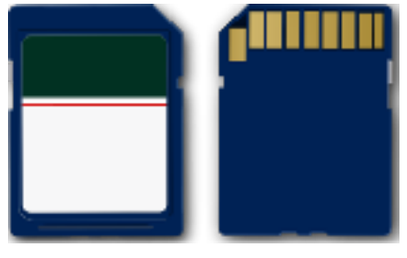
http://www.intel.com/content/www/us/en/do-it-yourself/edison.html



- Processors
  - Quark MCU @100 MHz
  - Dual-core ATOM CPU @500MHz
  - 4 GB NAND flash (eMMC), 1GB DDR3 RAM
- Wi-Fi and BLE
- 70-pin wired connector on back



- Dimensions
  - 35.5 x 25.0 x 3.9 mm^3 — just slightly larger than SD card @ 32 x 24



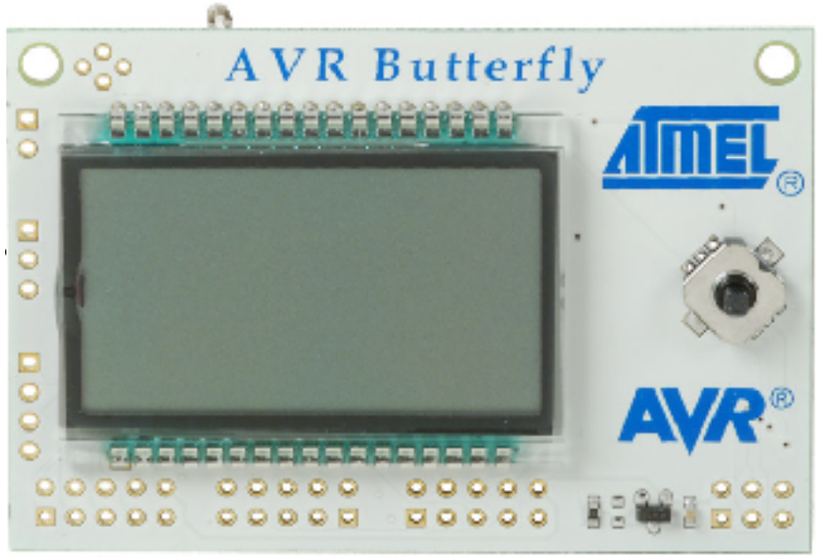
TI Sensor Tag

http://www.ti.com/sensortag

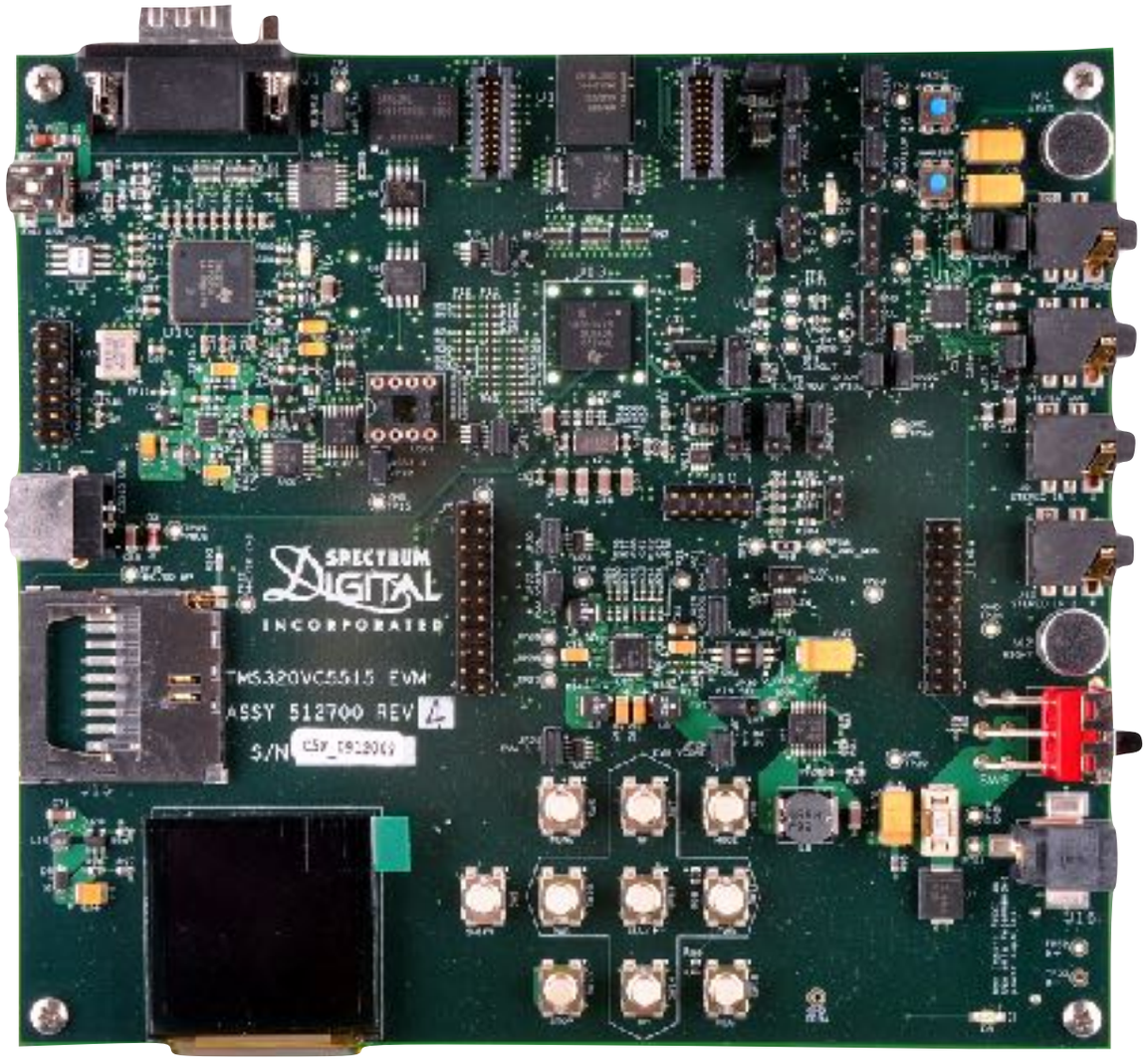
- Based on CC2541
- Rich sensors
  - temperature, humidity, pressure
  - acceleration, gyro, magnetometer
- Powered by CR2032 coin cell
- Over-the-air download (OAD)
  - can download iBeacon profile
- Programmable or ready-to-use firmware
- Limited expandability

AVR Butterfly

- $20, MCU with LCD, temp/light sensor, tone gen, joystick, prototyping area
- 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



Evaluation Options for TI C5515 fixed-point DSP



$395

$79

Can’t find what you need?

- Build your own board!
  - but this may be the more expensive option
- Start with evaluation kit if possible
  - Chip on a board with connectors and prototyping area
  - Connect boards to test out functionality
- Expect to iterate a few times to work out all issues