introducing physical computing

jacopo (b2p)

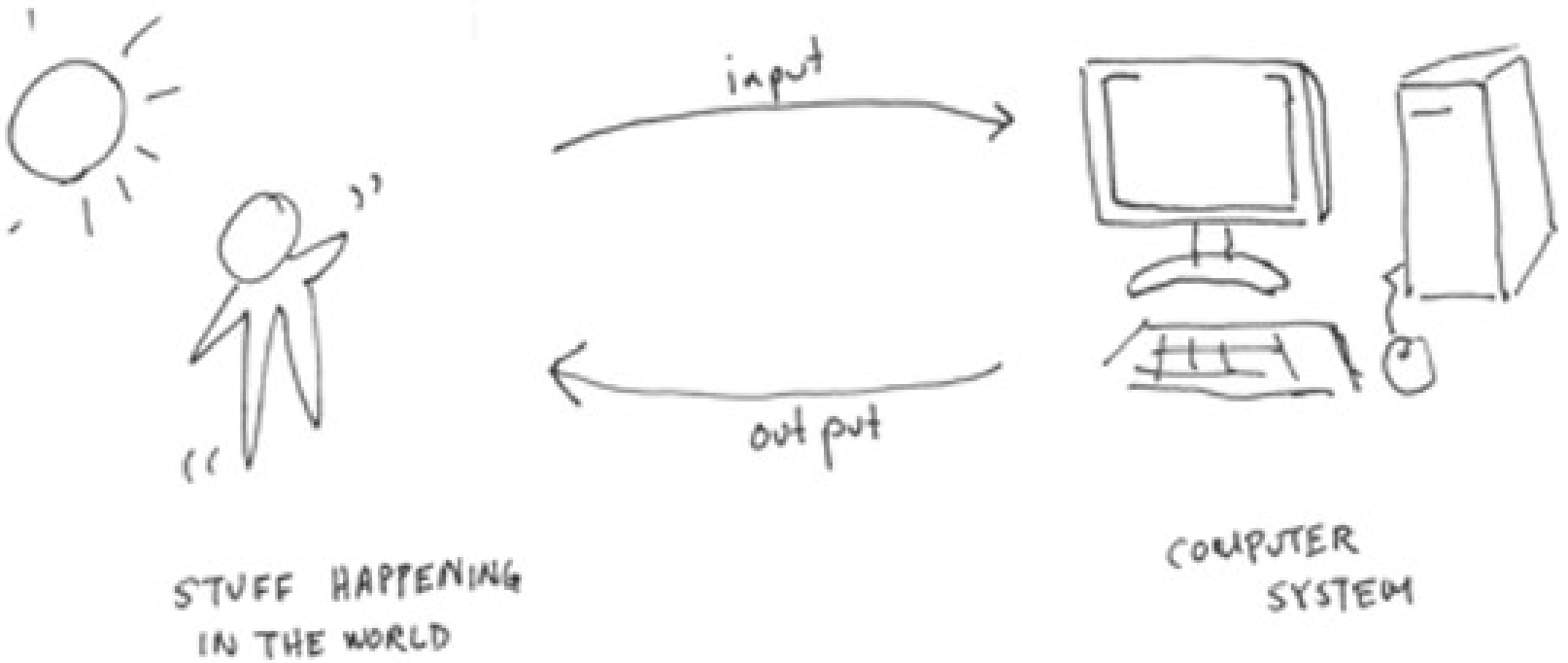
Summary

- The real world interface
- Input/Output, Comunication and powering
- What's a uController
- Powering/programming/toolchain
- The Arduino hw platform
- The Arduino software
- Arduino-Processing interfacing

The real world interface

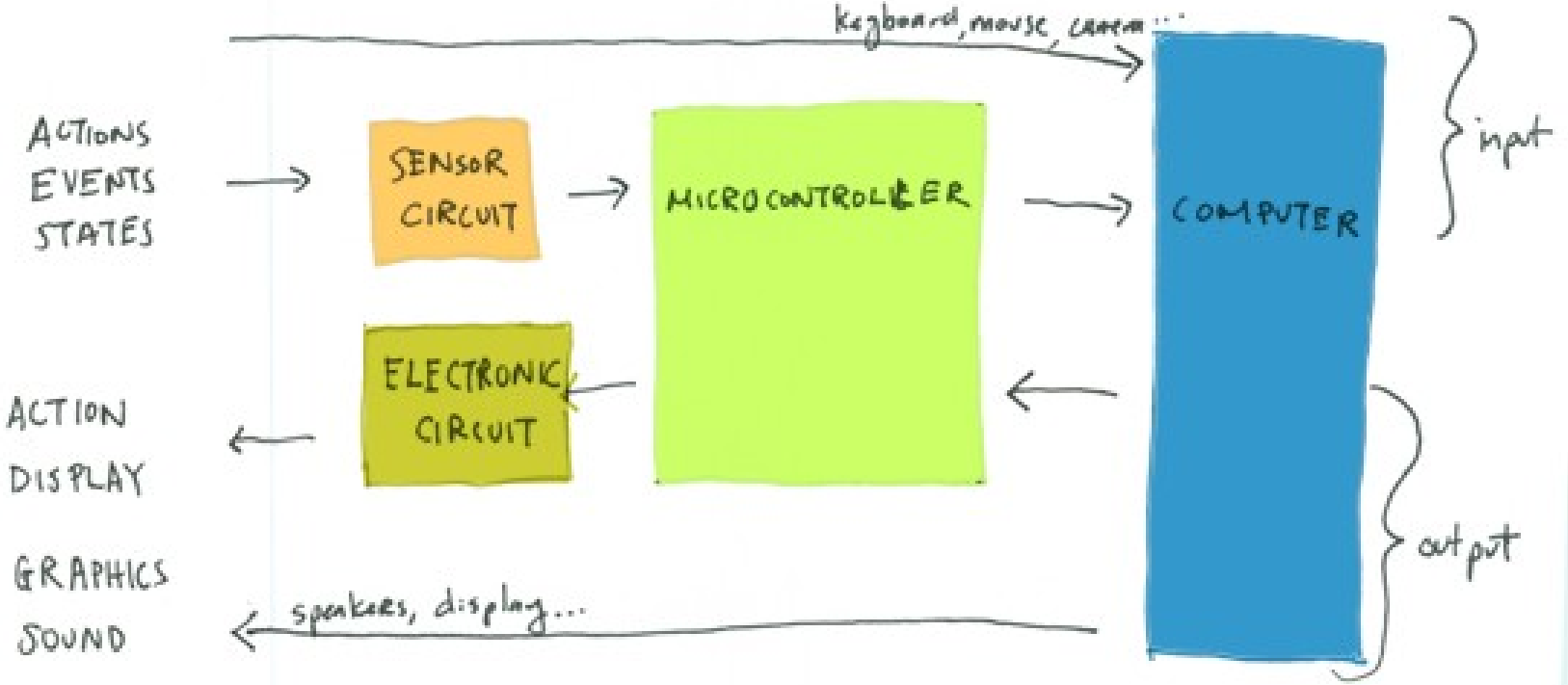
- Il 'sistema mondo' cosi' come lo conosciamo ha una “interfaccia” con il quale e' possibile rapportarsi ed interagire.
- Fenomeni fisici, elettrici, visivi ed anche sensoriali possono essere tradotti in modo da potere interagire con un sistema elettronico.

The world physical interface



From ccrma.standfor.edu

How world talks to machines

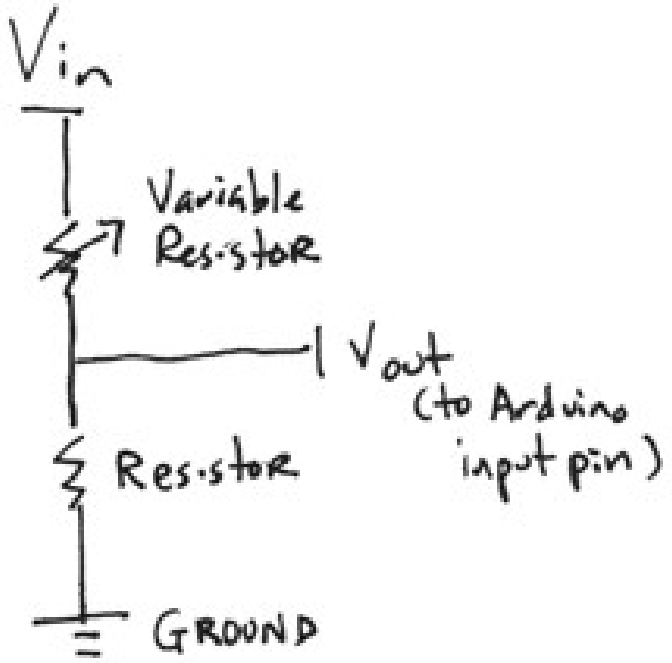


From ccrma.standfor.edu

Sensori elettronici

- Sensoristica elettronica: traduzione dal fenomeno di interesse ad un sistema analogico-digitale
- La variazione di un fenomeno fisico di interesse viene “tradotta” in un corrispondente fenomeno elettrico
- Tipici esempi di sono: fotoresistori, potenziometri, termocoppie, touchpad

Esempio: un semplice sensore



From ccrma.standfor.edu

Input/Ouput

- Per rapportarsi con il mondo circostante sono necessari almeno due tipi di possibile interfacciamento
- IO Analogici: il valore di ingresso e' un valore definito in un intervallo continuo
- IO Digitali: i possibili valori di ingresso logici sono semplici 1/0, acceso/spento, on/off

IO analogico

- Risultato della trasduzione da un fenomeno fisico (distanza, pressione, calore, luminosita' etc) in un parametro elettrico
- Tipicamente si misura la variazione di DDP o tensione (espressa in Volt)
- ADC = Analog to Digital Converter: ingresso analogico diviene un valore discreto comprensibile al uC digitale (eg 5V=1024)

IO Digitale

- Valori di tensione “soglia” si traducono in letture di 1 o 0 sul pin di ingresso
- 0V - ~1.3V applicati = 0 logico
- ~3.7V - 5V applicati = 1 logico
- Utilizzo base: GPIO (general purpose IO)
- Avanzato: protocol mimic (simulate a specific protocol when not available)

What is an MCU

- A microcontroller is small, power saving, programmable input-output processor
- Semplificando (molto) si applica un input e si ottiene un output.
- L'input puo' essere analogico (tensione) o digitale (gpio o spedito tramite un bus)
- Altrettanto vale per l'ouptut.

- A microcontroller is essentially a small computer on a chip. Like any computer, it has memory, and can be programmed to do calculations, receive input, and generate output
- Unlike a PC, it incorporates memory, a CPU, peripherals and I/O interfaces into a single chip.
- Microcontrollers are ubiquitous, found in almost every electronic device from ovens to iPods.

From ccrma.standfor.edu

Where are MCUs?

- If everywhere seems a little overestimating to you:
- Any kind of domotic applications (ever heard about smart fridges?)...
- Automatization, automobiles (~150MCU in a 2010 built car), remote controller, sensors networks, even in toys..

Programming an MCU

- We said programmable how???
- MCUs run standard C code (everything runs standard C code, anyway...)
- Toolchain, compiler, IDE are heavily dependent on MCU family (eg. Microchip proprietary tools/avr-gcc)

Cross-compiling

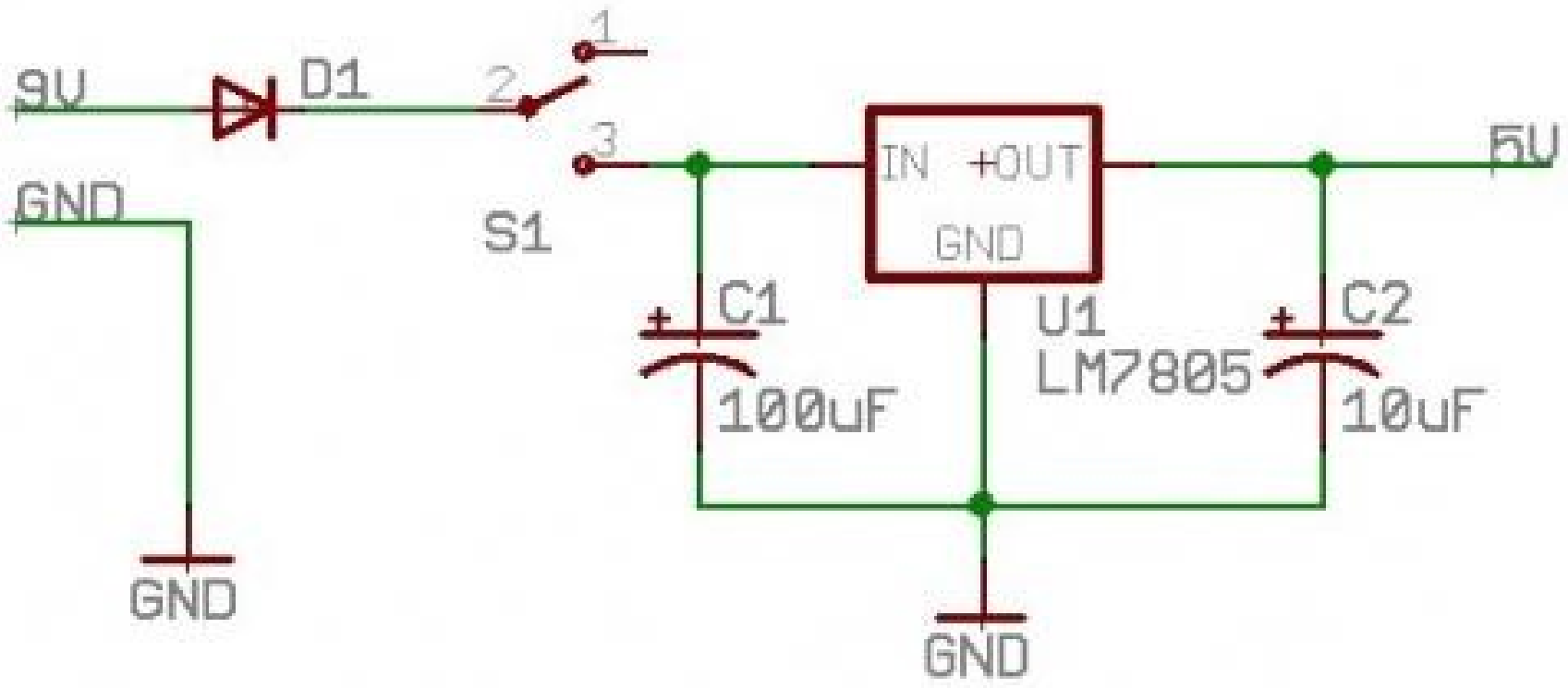
- The process of cross-compiling is what enables you to write code on your pc and run it on an external device
- Compiled code (object code) is transferred on the MCU with physical tools such as external programmer, JTAG debugger, and sometimes with software emulated programmer (ie. Arduino bootloader)

Powering an MCU

- MCU are usually powered at 3.3V or 5V
- “regolatori lineari” come il 7805 o 7833: dispositivi che data in ingresso una tensione compresa tra 7V e 12V restituiscono 5 o 3.3V rispettivamente...
- Tipical scenario:

  220V -> trasformatore AC-DC -> 12/9V -> 7805 -> 5V -> MCU

Simple(st) Voltage Regulation



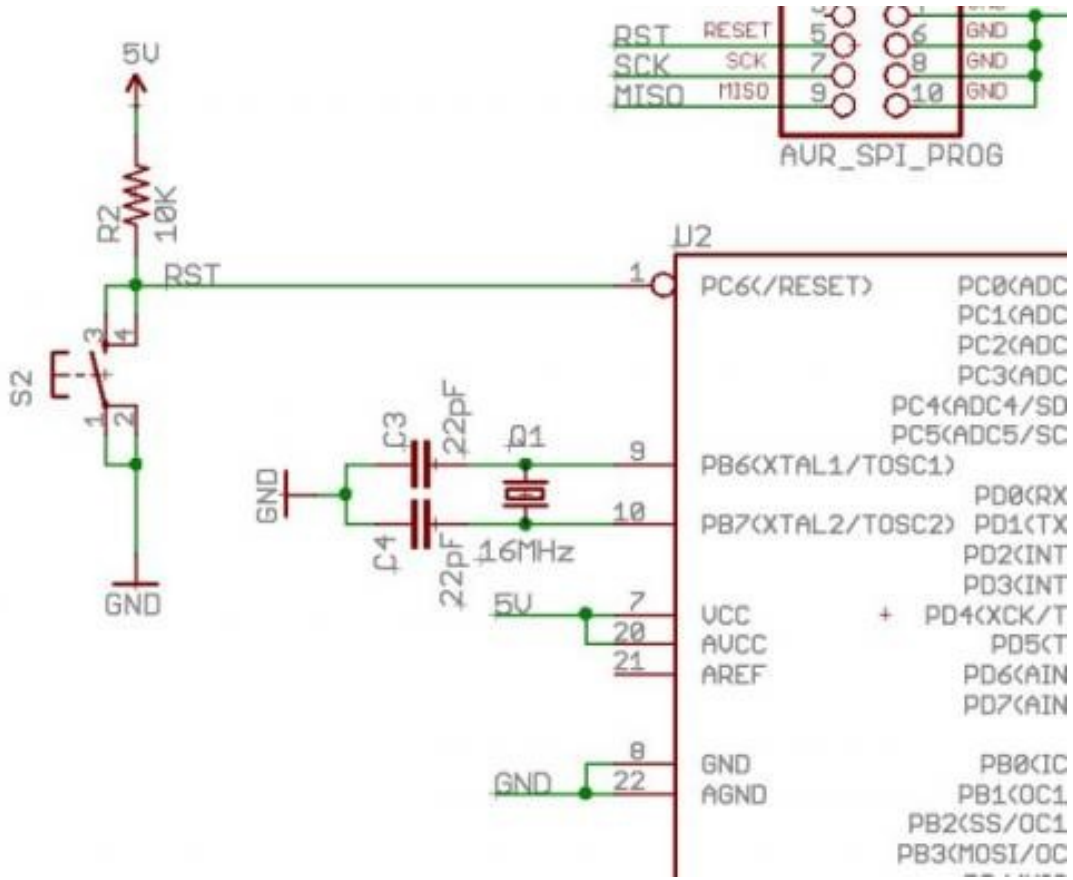
From
sparkfun.com

Clocking an MCU

- A tutti i dispositivi elettronici serve una temporizzazione, qualcosa che sincronizzi le operazioni interne al processore.
- Viene generalmente viene detto “clock”, e non e' altro che una onda quadra (presumibilmente) precisa con una determinata frequenza
- PC: >= 3Ghz (per singolo core)
- MCU: 16Mhz (atmega), ~40Mhz PIC
- Faster clock does not mean faster execution

- Different typologies of clock are available:
- Internal RC: provided by the MCU (processor clocks itself) (cheap)
- External RC: a 2 pin crystal oscillator that provides clock to the device (standard)
- External Clock: a multi-device powered clock (high end)

Integrating an External XTAL



Bus di comunicazione

- Sono nativamente disponibili (nativamente = supporto nell'ISA del processore) alcuni bus di comunicazione per il collegamento MCU-MCU, MCU-PC, MCU-Whatsoaever
- La loro presenza e il tipo dipende dal microcontrollore scelto
- SPI, i2c, USART (Low End)
- USB, CAN bus (High End)

UART and serial communication

- Why USART is important for us?
- Si tratta di uno dei bus piu' comuni, e anche se nei pc di nuova generazione non e' presente una porta dedicata, il protocollo puo' facilmente essere convertito verso il piu' comune USB.
- L'interfacciamento seriale e' il piu' comune metodo di comunicazione MCU-PC
- Utile per debug, controllo, comunicazione

USART: standards

- PC usually employs RS232 standard, DB9 connector, +12/-12V (o +-5V, +-9V, +-15V)
- MCU employ TTL logic, that ranges between 0/3.3V (and usually has only rx-tx pins)
- Conversion is needed to communicate to MCU: MAX232 is the standard solution
- MCU output could then be translated to USB using a UART-USB cable that integrate a FTDI (or PL232) chip

Let's get dirty: a real MCU

- Esistono diverse famiglie di microcontrollori, prodotti e progettati da diversi OEM.
- PIC: Microchip, proprietary IDE, compiler and toolchain.
- TI's MSP430 16 bit, low power
- Freescale: proprietary development tools
- Atmel Atmega serie: our choice :)

- Why Atmel Atmega:
  free development tools (gcc, avr-libc, avrdude etc)
- Atmega AVR series ranges from 8 bit low power uC to 32bit industrial grade processors
- Active and extended community

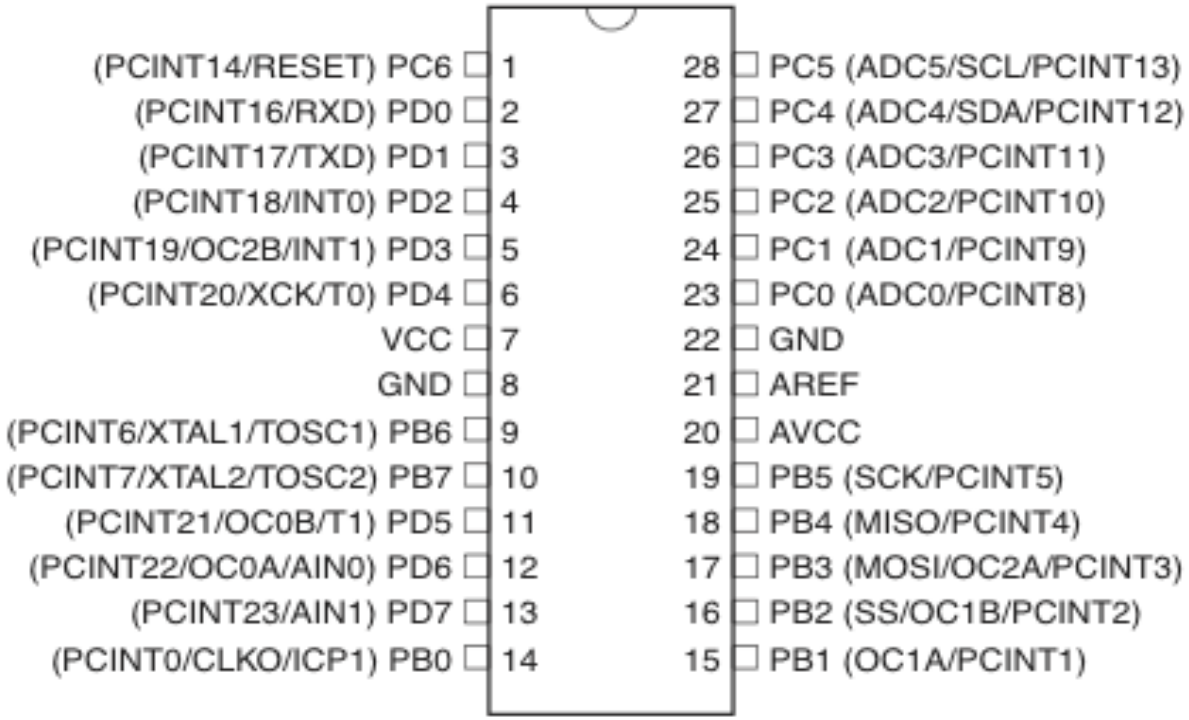
  (avrfreaks.com, arduino.cc, wiring etc)

Our target: avr328p

- Arduino (2009, uno etc) base processor
- Low cost (~4 euro)
- Easy integration (DIY)

- Schematics: what's on board?
- Pin mode selection: a 4-line example

What's available?



Each pin provides multiple functionalities...

How do we select the one we aim to use??

Even More Dirty: let's blink

- Blinking a led is the classic “Hello World” for uControllers.
- Learn how to:
- Read datasheets
- write code
- Compile
- transfer code on the machine

Some details

- Compiler: avr-gcc transforms human readable code into machine runnable one
- Avr-libc: provides a subset of standard C library to easy the developing process (_delay_ms, math.h, interrupt.h)
- Avrdude: Upload the code to the programmer, that programs the MCU

Arduino: the platform

- Abbiamo tra le righe nominato arduino svariate volte.
- Perche' e' cosi' interessante arduino, e per che tipo di applicazioni si usa solitamente?

- Prima cosa: Arduino e' COMODO
- Non tutti hanno tempo/voglia/interesse a disegnare e saldare la propria board di sviluppo
- Il che significherebbe: fare un prototipo, saldarlo, disegnare gli schematici, stamparli al contrario, preparare un bagno di acidi di borio, immergere una piastra di rame con attaccato sopra lo stampato girato al contrario, estrarre dal bagno di acido, ripassare tutte le connessioni con un pennarello, controllare con il tester che non si siano 'mangiate' nel bagno di acido etc etc
- Arduino funziona 'out of the box'

Comodo, si ma perche'?

- Oltre a risparmiare ore per disegnare il circuito etc etc.. Arduino esce nativamente con il supporto USB.
- L'USB permette di alimentare il 328p nativamente a 5V
- L'USB permette (mediante un chip FTDI) di 'parlare' direttamente dal pc con arduino
- Due dei principali 'problemi' nell'approccio ai microcontrollori sono risolti

A voodoo magic bootloader

- Inoltre (e forse e' la cosa migliore) arduino puo' essere programmato direttamente da USB
- Pre: cos'e' un bootloader, e quando esegue
- Il bootloader di arduino emula un programmatore fisico, e fa in modo che “il programmatore programmi se stesso” (wow)

Arduino: il software

- Se non bastasse arduino distribuisce anche un proprio IDE (GPLv3) basato su processing, con proprie librerie che forniscono un accesso piu' “morbido” alle interfaccie di input/output del uControllore.
- Cosi' come processing rende facile a chi non ha esperienza di programmazione sviluppare elaborazioni grafiche e animazioni, lo stesso fa arduino per il physical computing.

Exploit the power

- Rifacciamo cio' che abbiamo fatto prima (blink a led) ma con arduino (sia l'hardware che il software)
- Non c'e' necessita' di un programmatore esterno
- L'IDE integra librerie ed esempi di ogni tipo
- Il debug puo' avvenire via seriale (esempio con e senza arduino)

Arduino: per fare cosa?

- Domotica: home automatization, remote gardening, domestic access control
- Gaming: open source gameboy, DIY guitar hero controller, wii-mote hacking, retro gaming
- Automatization: servo controlled web cameras, DIY junkbots, sensorial devices
- Art: Live vjing, audio-video infrastructure, pure data, max SP, processing interfacing etc

Resume

- Per cio' che abbiamo visto fino ad ora, abbiamo a disposizione una piattaforma semplice, economica e flessibile per costruire piccoli dispositivi elettronici 'intelligenti'
- Un po' di risorse:

  La starting guide di Arduino stesso

  “Making things talk” (T.Igoe)

  Arduino playground (arduino.cc)

Interfacciarsi con il PC

- Abbiamo parlato in precedenza di UART e comunicazione seriale, e di come la presenza di un chip FTDI su arduino permetta la comunicazione diretta via USB con un qualsiasi computer.
- Mediante questa interfaccia si possono scambiare ovviamente dati ed informazioni in modo bidirezionale

Comunicare con Arduino

- Avendo un canale hw gia' instaurato e funzionante, possiamo definire tutti i 'protocolli' che desideriamo per comunicare con il nostro device
- Un protocollo e' semplicemente una forma di accordo sulla struttura dei messaggi che vengono trasferiti da una entita' ad un'altra
- Eg: TCP/IP per le reti, HTTP per il web, gli stessi bus di comunicazione sono basati su protocolli condivisi

- Possiamo quindi implementare (sui due end device) software appositi per scambiarsi dati ed agire di conseguenza.
- Oppure non reinventare la ruota e sfruttare cio' che altri hanno gia' sviluppato (in particolar modo per arduino).

Firmata Library

- Firmata e' un protocollo di comunicazione specifico per arduino
- Basato sul formato MIDI (con il quale puo' coesistere), permette di accedere mediante una definita interfaccia software ai pin e all'hardware di arduino.
- La comunicazione e' bidirezionale: dal pc posso sapere che tensione ho in ingresso su un ADC, ma posso altresi decidere di scrivere 0 su un output digitale.

Firmata: protocol specification

- Due tipologie di messaggio:
- Control messages:
  set a pin state, report a pin state
- Sys-ex messages:
  enables exchanging of longer messages (such as strings or data sets)

Firmata: extended

- Firmata supporta altri tipi di comunicazione che non sono limitati a “semplici” scambi di bit
- Supporta protocolli come i2c, permettendo di gestire dal pc una connessione i2c tra arduino e altri device
- Supporta la gestione e la comunicazione con servo motori e motori stepper (per applicazioni di robotica e automazione)

Processing

- Processing, il cui IDE e' la base di quello di arduino, e' un linguaggio di programmazione basato su un subset di Java, che permette in poche righe di codice di realizzare animazioni grafiche.
- Il target sono animazioni di forme geometriche, editing dinamico di immagini, animazioni temporizzate
- Dal sito: “creative coding and computational art”

Processing

- Processing e' orientato agli oggetti
- Processing astrae dalla complessita' di un vero linguaggio di programmazione come Java tutti i dettagli relativi all'utilizzo di librerie grafiche e di disegno
- Processing fornisce uno strumento 'ready-to-use' comprensibile anche a chi non conosce java e magari “non sa programmare”
- Risorse:
  processing.org (ufficiale)
  openprocessing.org (esempi e tutorial)

Processing-Arduino

- Processing puo' “parlare” con arduino..
- Come? Utilizzando le Firmata library di cui abbiamo discusso in precedenza.
- E' stata sviluppata una “estensione” di processing (un oggetto Java) che permette di controllare da processing una board arduino, e viceversa, di modificare il comportamento di uno sketch processing in base a cio' che accade all'arduino

Arduino-Processing interfacing

- What do we need to do so?
- Arduino side:
  standard firmata library, waiting for command from processing;
  answer back reporting pin status or changes pin status depending on the command issued from the processing side

Arduino-Processing Interfacing

- Processing Side:
  On the processing side, we simply include the “Arduino Library” (easy from the IDE), and use it's API to communicate with the external device.
  The library translate our requests into firmata messages and, on the sketch side, we act accordingly to the messages sent or received to or from the arduino.

Arduino-Processing: examples

- The standard library provides a set of examples to show how is possible to interact from processing with arduino.
- arduino_inputp.pde:
  graphically represents the actual status of arduino pins
- arduino_output.pde:
  provides a graphical user interface to controll arduino pins.

Arduino-Processing: other examples

- Show som examples of interaction:
- The grid:
  use a slider (potentiometer) to dynamically modify the dimension of a grid
- Distance indicator:
  using a distance sensor we graphically represent the distance between arduino and an object. (Try do that without arduino, but with a led)