
Arduino

e l'acquisizione dati nel laboratorio didattico

Carmelo Sgrò (carmelo.sgro@pi.infn.it)

Dipartimento di Fisica E. Fermi, Università di Pisa
Piano Nazionale Lauree Scientifiche

Febbraio 2017

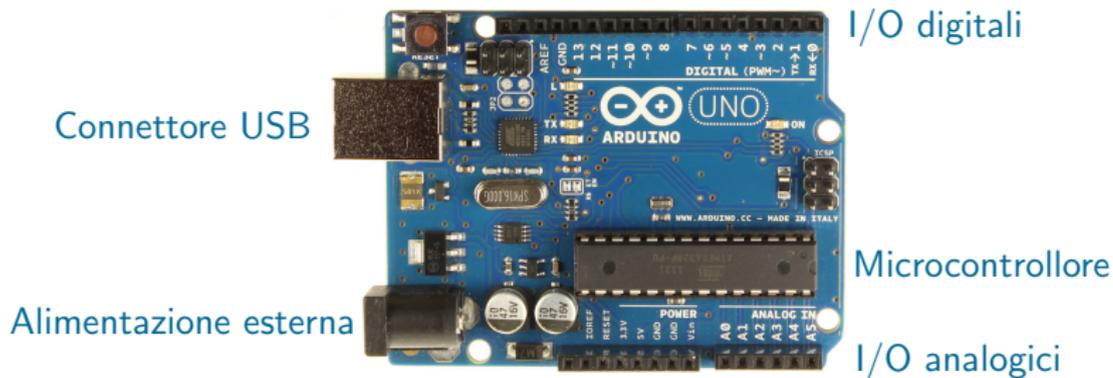
- ▶ Arduino: cos'è e come funziona
- ▶ Ambiente di sviluppo di Arduino e le sue librerie
- ▶ Alcuni semplici esempi
 - ▶ Accendere un LED
 - ▶ Scrivere sulla seriale

- ▶ Il concetto di Acquisizione Dati (DAQ)
- ▶ La porta seriale: salvare dati su disco
- ▶ Il convertitore analogico digitale
- ▶ Un sensore di temperatura

- ▶ La misura di tempo: funzione micros e millis
- ▶ L'interrupt
- ▶ Un esempio: il pendolo

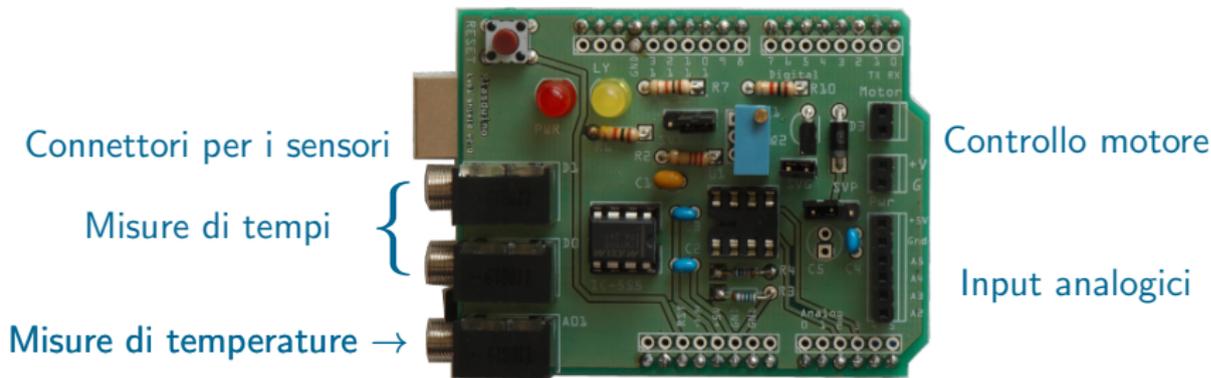
INTRODUZIONE AD ARDUINO

[HTTP://WWW.ARDUINO.CC/](http://www.arduino.cc/)



- ▶ **Arduino: una piattaforma *open* di prototipizzazione elettronica:**
 - ▶ Una scheda elettronica basata su un microcontrollore
 - ▶ Un ambiente software ad alto livello, con un'ampia libreria di funzioni;
- ▶ **Una tipica scheda di Arduino offre:**
 - ▶ Un microcontrollore
 - ▶ I pin per le varie periferiche
 - ▶ Un'interfaccia seriale via USB
 - ▶ Alimentazione tramite USB o indipendente

GLI “SHIELD”



- ▶ Uno *shield* è un circuito stampato da connettere *sopra* la scheda di Arduino
 - ▶ Viene utilizzata dagli utenti per implementare le proprie idee.
 - ▶ Si possono comprare per scopi specifici o farselo da solo
 - ▶ Esempio preso dal Laboratorio di Fisica I:
 - ▶ connettori per i sensori
 - ▶ condizionamento dei segnali (temperatura)
 - ▶ LED
 - ▶ calibrazione interna

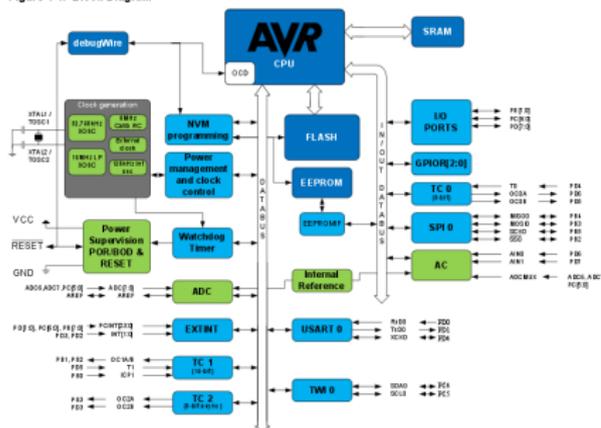
COS'E' UN MICROCONTROLLORE

- ▶ Un dispositivo che mette insieme una piccola CPU ed alcune periferiche:

- ▶ Pin di input/output (ad esempio per accendere/spegnere un LED)
- ▶ Contatori e timer
- ▶ Convertitori Analogico/Digitali (ADC)
- ▶ Pulse Width Modulation (PWM)
- ▶ Interfacce seriali (per la comunicazione con dispositivi esterni)
- ▶ Memoria interna
- ▶ ...

Block Diagram

Figure 4-1. Block Diagram



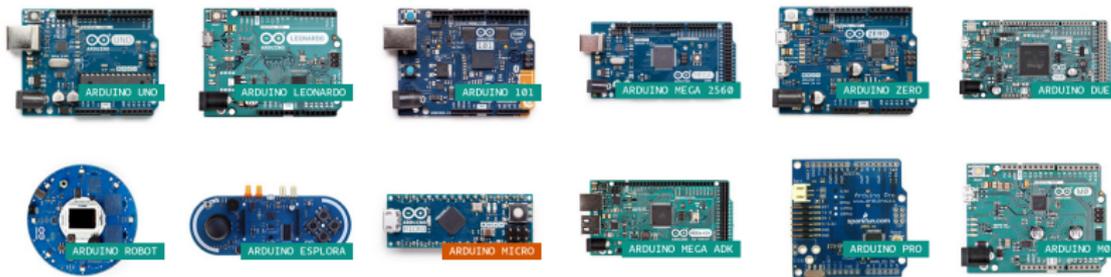
- ▶ Per i nostri scopi utilizzeremo un Arduino UNO
- ▶ Basato su Atmel ATmega328

- ▶ I coraggiosi possono andare a curiosare nel manuale:
<http://www.atmel.com/Images/doc8161.pdf>

QUANTI TIPI DI ARDUINO CI SONO?

[HTTPS://WWW.ARDUINO.CC/EN/MAIN/PRODUCTS](https://www.arduino.cc/en/Main/Products)

- ▶ Tanti e con caratteristiche diverse
- ▶ Entry level: per iniziare, ma non solo...
- ▶ Enhanced Features: per progetti complessi e prestazioni elevate

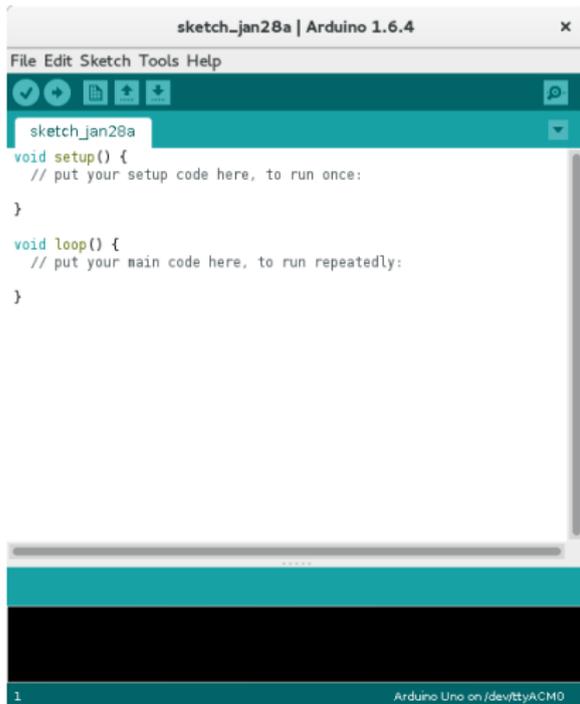


- ▶ Kit con tutorials e componenti: per imparare e “smanettare”



IL SOFTWARE DI SVILUPPO: ARDUINO IDE

[HTTPS://WWW.ARDUINO.CC/EN/MAIN/SOFTWARE](https://www.arduino.cc/en/Main/Software)



```
sketch_jan28a
void setup() {
  // put your setup code here, to run once:
}

void loop() {
  // put your main code here, to run repeatedly:
}

1 Arduino Uno on /dev/ttyACM0
```

Potete scaricarlo ed installarlo seguendo le istruzioni nel link in alto

- ▶ Parte fondamentale del successo di Arduino
- ▶ Consente di programmare il microcontrollore facilmente
- ▶ Esempi per quasi tutto
- ▶ Librerie per quasi tutto
- ▶ Vasta comunità di sviluppatori e forum di discussione
- ▶ Liberamente disponibile per vari OS (ed anche il codice sorgente)
- ▶ Attenzione però:
la generalità e facilità di utilizzo può andare a discapito delle prestazioni

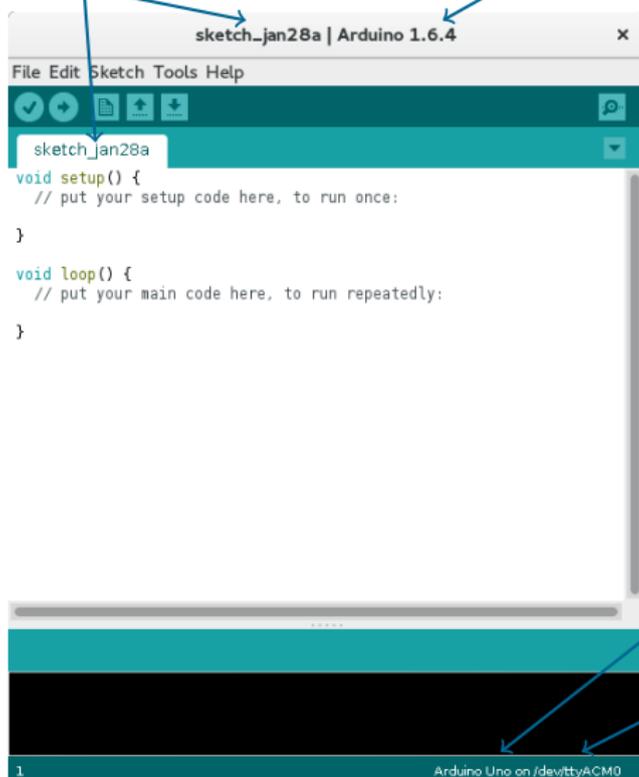
ARDUINO IDE

Nome del programma (Sketch)

Versione del software

Tasti per compilare e caricare lo sketch

Funzioni fondamentali
setup:
eseguita solo all'inizio.
loop:
eseguita in continuazione.

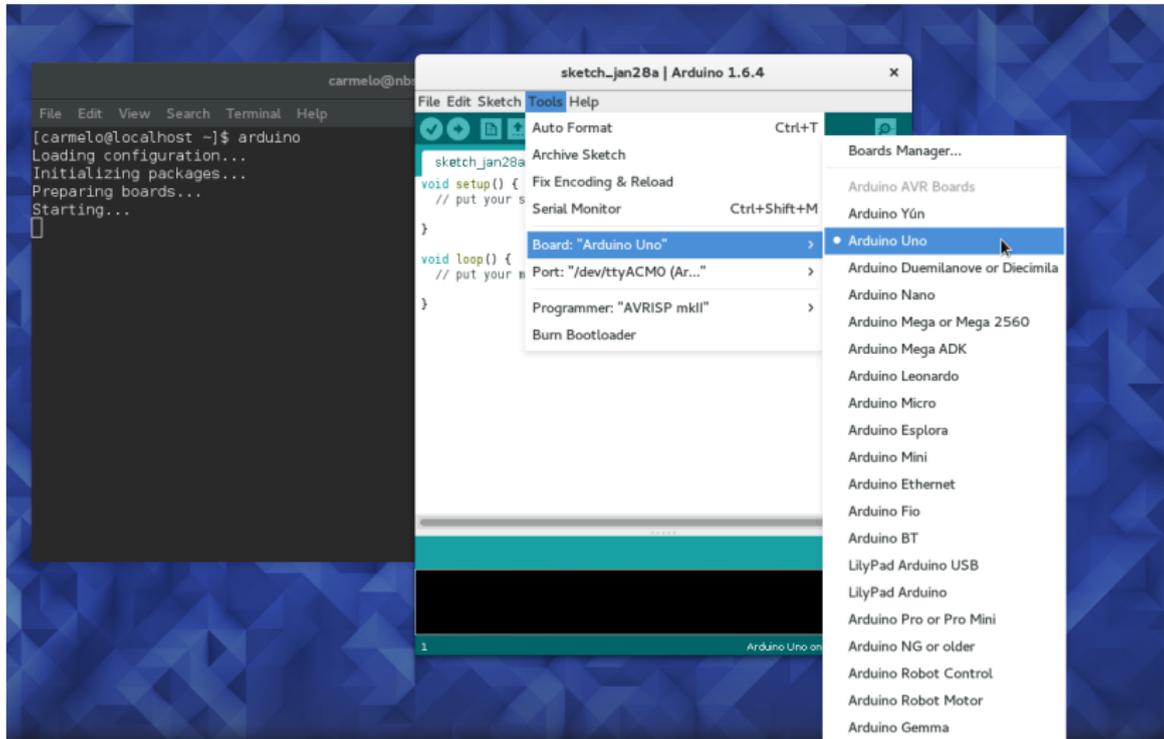


Serial Monitor

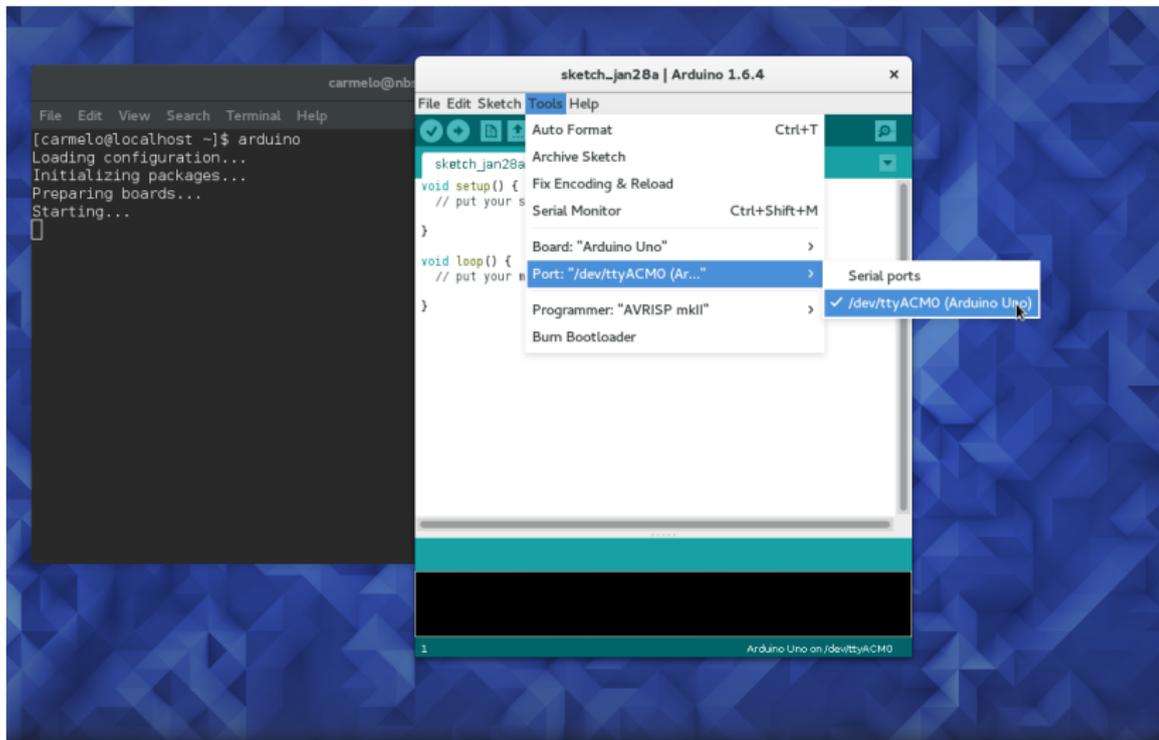
Modello di Arduino

Porta USB (seriale)

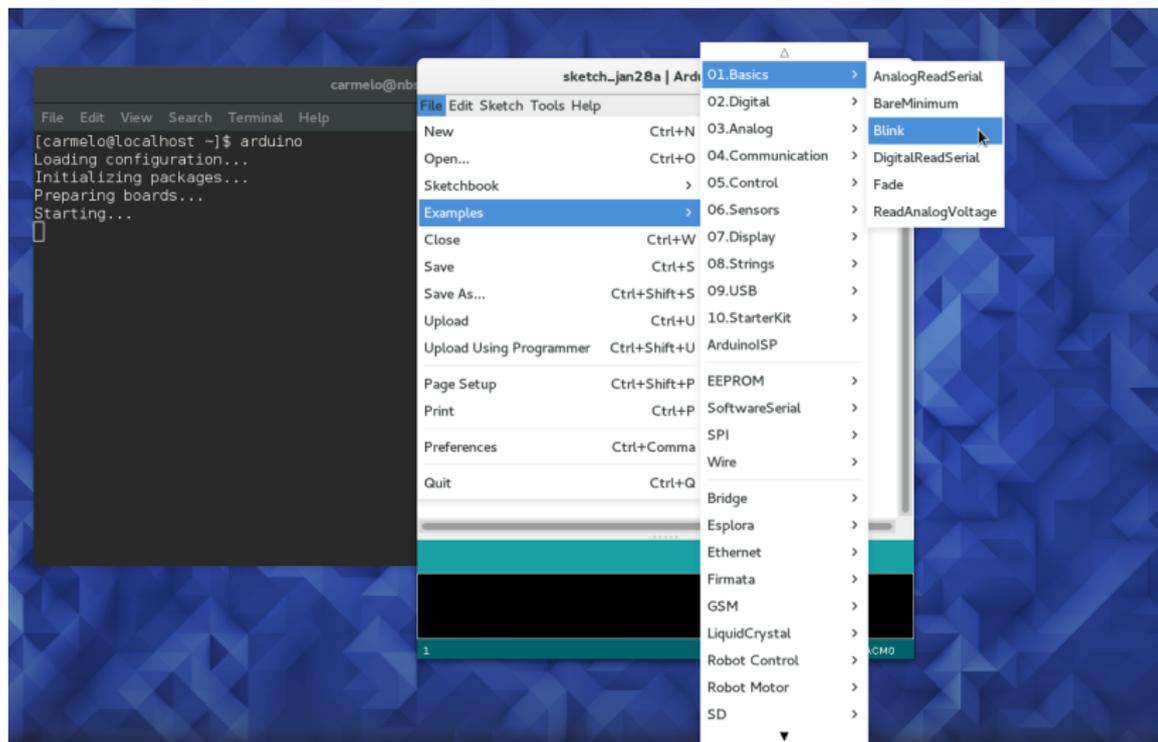
ARDUINO IDE: SELEZIONARE IL TIPO DI ARDUINO



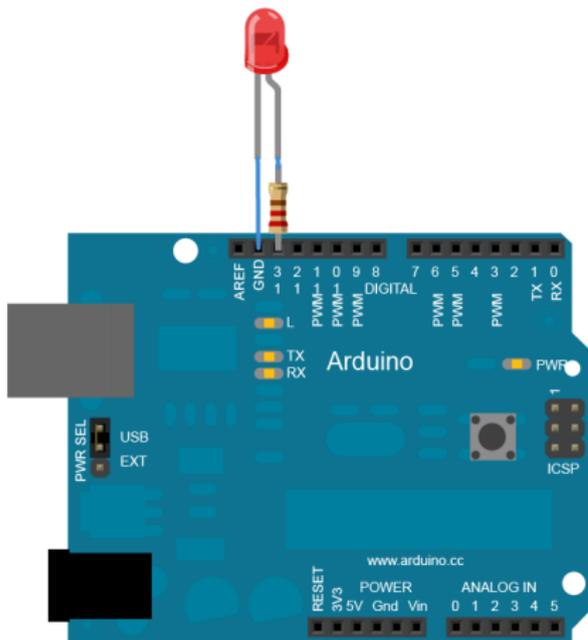
ARDUINO IDE: SELEZIONARE LA PORTA USB



COMINCIAMO CON UN ESEMPIO SEMPLICE: ACCENDERE UN LED



BLINK A LED: NOTA SUL CIRCUITO



- ▶ Esempio base su come si accendono e spengono i pin
- ▶ Arduino è alimentato a 5 Volt tramite la porta USB
 - ▶ Un pin “alto” è quindi a 5 V
 - ▶ Un pin “basso” è quindi a 0 V (GND)
 - ▶ Queste tensioni di riferimento sono disponibili sui pin esterni
- ▶ Un LED è già connesso sulla board al pin 13
- ▶ In generale potete connettere un LED and un Pin con lo schema qui a sinistra
 - ▶ Usare sempre una resistenza di limitazione della corrente!

BLINK A LED

[HTTPS://WWW.ARDUINO.CC/EN/TUTORIAL/BLINK](https://www.arduino.cc/en/tutorial/blink)

```
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin 13 as an output.
  pinMode(13, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(13, HIGH); // turn the LED on (HIGH is the voltage level)
  delay(1000);           // wait for a second
  digitalWrite(13, LOW); // turn the LED off by making the voltage LOW
  delay(1000);           // wait for a second
}
```

- ▶ Nel *setup* selezioniamo il pin 13 come pin di output
- ▶ Nel *loop* accendiamo (HIGH) e spegnamo (LOW) il pin 13, con un ritardo di 1000 ms (delay)
- ▶ La sintassi è praticamente C
- ▶ Cliccare su “Verify” per compilare e “Upload” per caricare su Arduino

1. Cambiare i ritardi nella funzione *delay*
2. Definire una variabile per il pin 13:

```
const int ledPin = 13;  
// The const keyword make the variable "read-only"
```

3. Definire una stringa per il pin 13:

```
#define ledPin 13  
// The compiler will replace any mention of ledPin  
// with the value 13 at compile time.  
// No semicolon after #define
```

4. ...

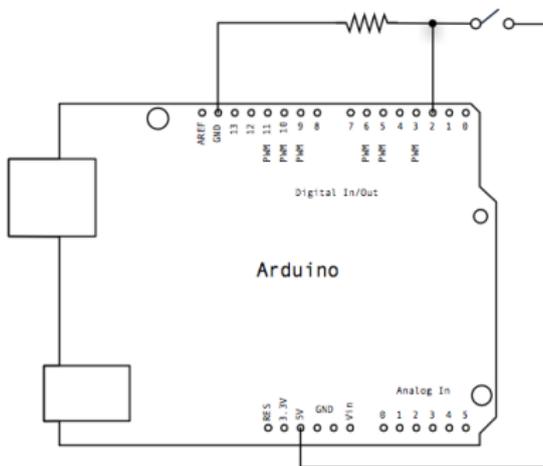
- ▶ Un altro esempio che potete studiare è "BlinkWithoutDelay"
 - ▶ Utilizza la funzione *millis* per il tracciare il tempo
- ▶ Guida completa alle istruzioni fondamentali in:
<https://www.arduino.cc/en/Reference/HomePage>

- ▶ In Arduino alcuni pin sono riservati a funzioni specifiche
 - ▶ Visibili sulla serigrafia della scheda
- ▶ In UNO i pin 0 (RX) e 1 (TX) sono utilizzati per la comunicazione seriale (TTL), verso il chip che gestisce l'USB (ATmega8U2 USB-to-TTL Serial chip)
- ▶ La comunicazione è gestita ad alto livello tramite la funzione *Serial*
 - ▶ <https://www.arduino.cc/en/Reference/Serial>
 - ▶ Si può emulare una porta seriale su altri pin tramite la libreria `SoftwareSerial`, ma non ne parleremo qui...
- ▶ Studiamo il funzionamento della porta seriale tramite l'esempio `DigitalSerialRead`

ESEMPIO DIGITALSERIALREAD

[HTTPS://WWW.ARDUINO.CC/EN/TUTORIAL/DIGITALREADSERIAL](https://www.arduino.cc/en/tutorial/digitalreadserial)

- ▶ Lo scopo è scrivere sulla seriale lo stato di un pin (il 2)
- ▶ Schema elettrico: connettere il pin 2 a GND (tramite una resistenza) o a 5V tramite un interruttore



```
// digital pin 2 has a pushbutton attached to it. Give it a name:
int pushButton = 2;

// the setup routine runs once when you press reset:
void setup() {
  // initialize serial communication at 9600 bits per second:
  Serial.begin(9600);
  // make the pushbutton's pin an input:
  pinMode(pushButton, INPUT);
}

// the loop routine runs over and over again forever:
void loop() {
  // read the input pin:
  int buttonState = digitalRead(pushButton);
  // print out the state of the button:
  Serial.println(buttonState);
  delay(1);          // delay in between reads for stability
}
```

- ▶ Porta seriale inizializzata nel setup con velocità di 9600 baud
- ▶ Il pin 2 settato con “input” e letto periodicamente
- ▶ Lo stato del pin 2 (0 o 1) ritrasmesso sulla seriale

- ▶ Aprite il Serial Monitor di Arduino e osservate l'output mentre cambiate lo stato del pin 2

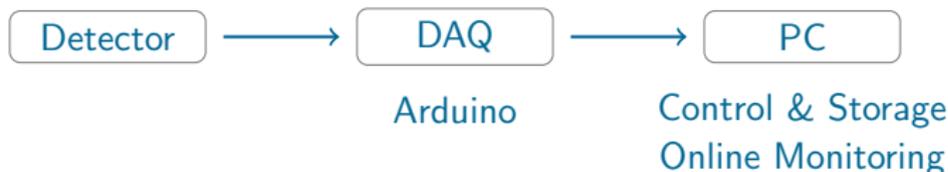


- ▶ Tip & Tricks:
 - ▶ La velocità della porta seriale in Arduino deve essere la stessa del vostro monitor sul pc
 - ▶ La programmazione di Arduino avviene sulla stessa porta seriale: chiudetela prima di caricare un nuovo sketch

- ▶ Scrivete un programma che legge una carattere dalla seriale e lo ritrasmette
- ▶ oppure accende e spegne il LED in base al un carattere che trasmettete sulla seriale
- ▶ Suggerimento:

```
if (Serial.available() > 0) {  
  inByte = Serial.read();  
  ...  
}
```

ACQUISIZIONE DATI (DAQ)



Lo scopo è leggere i sensori e scrivere i dati su disco in maniera più efficiente possibile. In un tipico esperimento troviamo:

1. I sensori dedicati
2. Un apparato hardware per gestire e leggere i vari sensori
 - ▶ Arduino nel nostro caso
 - ▶ Con eventualmente circuiteria per il condizionamento dei segnali
3. Un sistema di controllo (su PC) per gestire l'acquisizione e salvare i dati su disco
 - ▶ Questo sistema dipende dall'infrastruttura (hardware e software) che avete a disposizione "in casa"
 - ▶ Qui ci limiteremo alla parte di salvataggio dei dati

Attenzione! Altre funzioni ausiliarie (analisi online, visualizzazione, etc.) possono ridurre l'efficienza. Valutate sempre cosa è meglio fare off-line

- ▶ Modo “quick and dirty” per reindirizzare l’output della seriale su file
- ▶ Il linguaggio python contiene una libreria pyserial
 - ▶ Con dei tool pronti per le operazioni più comuni
 - ▶ <http://pyserial.readthedocs.io/en/latest/tools.html>
- ▶ Per vedere i dispositivi connessi:

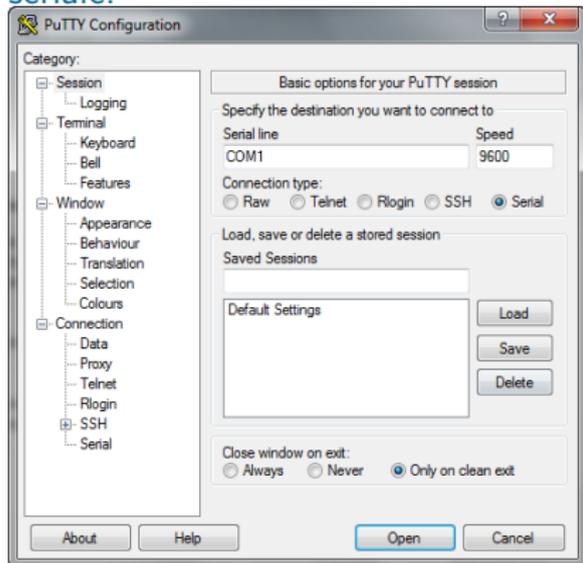
```
$ python -m serial.tools.list_ports -v/dev/ttyACM0
  desc: ttyACM0
  hwid: USB VID:PID=2341:0043 SER=A4139363831351013102 LOCATION=2-1.3
1 ports found
```

- ▶ Aprire una console e reindirizzare l’output su un file “logfile.txt” (Linux only)

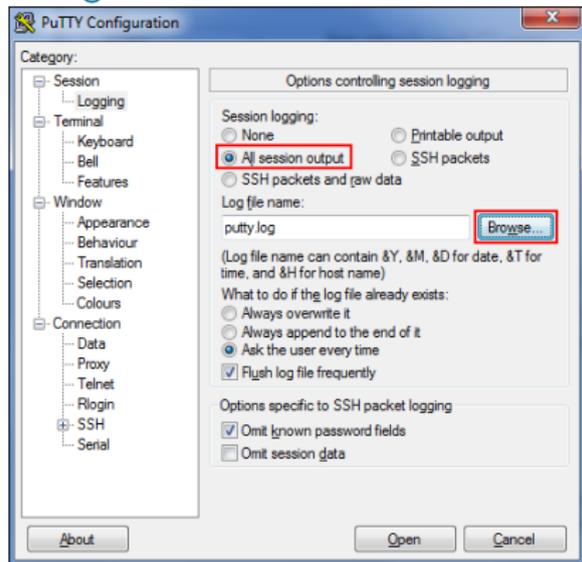
```
$ python -m serial.tools.miniterm /dev/ttyACM0 9600 | tee logfile.txt
--- Miniterm on /dev/ttyACM0 9600,8,N,1 ---
--- Quit: Ctrl+] | Menu: Ctrl+T | Help: Ctrl+T followed by Ctrl+H ---
```

TERMINALE SERIALE CON PUTTY (WINDOWS)

Per far partire una connessione seriale:



Per salvare tutta la comunicazione su log file:



IL CONVERTITORE ANALOGICO/DIGITALE (ADC)

- ▶ Arduino utilizza un convertitore Analogico Digitale a 10 bit ad approssimazioni successive:
 - ▶ 10 bit: 2^{10} passi da 0 a 5 V (0–1023 adc counts)
 - ▶ approssimazioni successive: una tensione di riferimento viene confrontata con il segnale da digitalizzare, un bit alla volta a partire dal più significativo
- ▶ Ci sono 6 canali analogici, ma in realta' c'e' solo un ADC
 - ▶ Si legge un canale alla volta
- ▶ L'ADC si legge con la funzione `analogRead()`:
<https://www.arduino.cc/en/Reference/AnalogRead>
- ▶ Il codice di seguito é adattato dall'esempio `AnalogSerialRead`

```
void setup() {  
  Serial.begin(9600); // initialize serial communication  
}  
  
// the loop routine runs over and over again forever:  
void loop() {  
  int sensorValue = analogRead(A0); // read the input on analog pin 0  
  Serial.println(sensorValue);      // print out the value you read  
  delay(1000);                      // wait 1 second till next acquisition  
}
```

- ▶ I termistori possono essere visti come resistenze il cui valore cambia con la temperatura:
<https://it.wikipedia.org/wiki/Termistore>
- ▶ NTC: Negative Temperature Coefficient, resistenza che decresce con l'aumentare della temperatura
- ▶ La relazione che lega temperatura e resistenza è parametrizzata dall'equazione di "Steinhart-Hart":

$$\frac{1}{T} = A + B \cdot \ln(R_T) + C \cdot (\ln(R_T))^3 \quad (1)$$

- ▶ I parametri A, B e C sono le costanti di calibrazione e dipendono dal modello di sensore che avete a disposizione
 - ▶ Il produttore vi da i parametri nominali
 - ▶ Ma ve li potete ricalibrare da soli: quanti punti vi servono?

- ▶ Per ricavare la resistenza utilizziamo lo schema elettrico di un partitore resistivo
 - ▶ Dobbiamo leggere la tensione V e ricavare R_T

$$R_T = R(V_{CC}/V - 1) \quad (2)$$

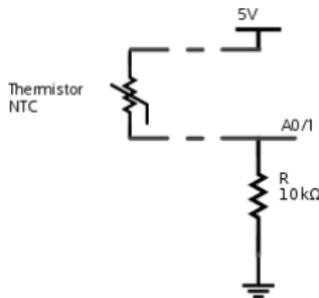
- ▶ Esercizio: convertire la lettura dell'ADC in temperatura in Arduino
- ▶ Con $R = 10000 \Omega$, $A = 8.65 \cdot 10^{-4}$ e $B = 2.55 \cdot 10^{-4}$
 $C = 1.75 \cdot 10^{-7}$, suggerimenti:

```
#define SHUNT_RESISTOR 10000.  
#define A 8.65082e-04  
#define B 2.55459e-04  
#define C 1.75e-07
```

```
int sensorValue;  
float logR;  
float Temp;
```

```
[...]
```

```
logR = log(SHUNT_RESISTOR*(1023./adcValue - 1)) ;  
Temp = 1./(A + B*logR + C*logR*logR*logR) - 273.15 ;
```



- ▶ Calibrazione dello “zero” ($0\text{ }^{\circ}\text{C}$), con acqua distillata in equilibrio tra fase liquida e solida
 - ▶ e.g. <https://www.youtube.com/watch?v=KY0JayWqB3g>
 - ▶ c'è un “offset” sistematico nelle misure?
- ▶ Quali altri riferimenti “sicuri” si possono prendere per la calibrazione?
- ▶ Tempo di rilassamento: il sensore deve andare all'equilibrio termico con il mezzo, $T(t) \sim T_0 e^{t/\tau} + T_{fin}$ dove τ dipende dalla sua capacità termica
 - ▶ Siete in grado di misurare τ dai dati delle vostre esperienze?
- ▶ Contributi alla risoluzione:
 - ▶ Precisione dei componenti – effetti sistematici
 - ▶ Granularità dell'ADC – risoluzione strumentale
 - ▶ Se fate i conti bene dovrete trovare una risoluzione $\Delta T \sim 0.1 - 0.2\text{ }^{\circ}\text{C}$.

ESERCIZIO (PER I VOLENTEROSI)

RICAVARE IL CONTRIBUTO ALLA RISOLUZIONE IN T LEGATO ALL'ADC

Sappiamo che:

- ▶ L'ADC legge una tensione da 0 a V_{cc} (5V) in 1024 passi
- ▶ La resistenza del termistore si ricava quindi come:

$$R_T = R(1023/x - 1) \quad (3)$$

con $R = 10000 \Omega$ e $\Delta x = 1$

- ▶ La temperatura (in gradi kelvin) si ottiene con:

$$T = \frac{1}{A + B \cdot \ln(R_T) + C \cdot (\ln(R_T))^3} \quad (4)$$

con $A = 8.65 \cdot 10^{-4}$, $B = 2.55 \cdot 10^{-4}$ e $C = 1.75 \cdot 10^{-7}$

Siamo in grado di propagare l'incertezza ΔT al variare di T ?

- ▶ Usando le formule sacre $\Delta T = |dT/dx| \cdot \Delta x$
- ▶ Studiando $T(x+1) - T(x)$ in maniera numerica

- ▶ Tutte le operazioni interne del micro sono sincronizzate da un oscillatore interno: il *clock*
 - ▶ Con un periodo nominale di 16 MHz
 - ▶ Ovvero un “segnale” ogni $0.0625 \mu s (=1/16)$
- ▶ Un *timer* è un contatore che viene incrementato dal clock e può essere utilizzato per misurare i tempi
- ▶ La libreria di Arduino fornisce le funzioni per ottenere il tempo relativo dall'accensione (reset) del dispositivo:
 - ▶ “`millis()`”: <https://www.arduino.cc/en/Reference/Millis>
 - ▶ restituisce un *unsigned long*
 - ▶ overflow dopo circa 50 giorni
 - ▶ “`micros()`”: <https://www.arduino.cc/en/Reference/Micros>
 - ▶ restituisce un *unsigned long*
 - ▶ granularità nominale di $4 \mu s$
 - ▶ overflow dopo circa 70 minuti
- ▶ **Esercizio:** insieme alla misura di temperatura, fatevi restituire anche il tempo di acquisizione
 - ▶ Notare che il tempo di acquisizione ed il momento in cui il dato arriva al PC possono essere diversi

MISURE DI TEMPO: INTERRUPT

- ▶ Un “interrupt” è un meccanismo dei processori per reagire ad un input esterno asincrono (ed interrompere la normale attività)
 - ▶ Si può attivare, ad esempio, quando un pin cambia stato
 - ▶ 2 pins disponibili in Arduino UNO per questa funzione
 - ▶ Si può programmare con la funzione “attachInterrupt()”:
<https://www.arduino.cc/en/Reference/AttachInterrupt>
- ▶ Esercizio: accendere un LED con un interrupt sul pin 2:

```
const byte LED = 13;
const byte BUTTON = 2;

void switchPressed () // Interrupt Service Routine (ISR)
{
  if (digitalRead (BUTTON) == HIGH)
    digitalWrite (LED, HIGH);
  else
    digitalWrite (LED, LOW);
} // end of switchPressed

void setup ()
{
  pinMode (LED, OUTPUT); // so we can update the LED
  pinMode(BUTTON, INPUT_PULLUP); // internal pull-up resistor
  attachInterrupt (digitalPinToInterrupt (BUTTON), switchPressed, CHANGE); // attach inter
} // end of setup

void loop () {} // loop doing nothing
```

MISURE DI TEMPO: INTERRUPT

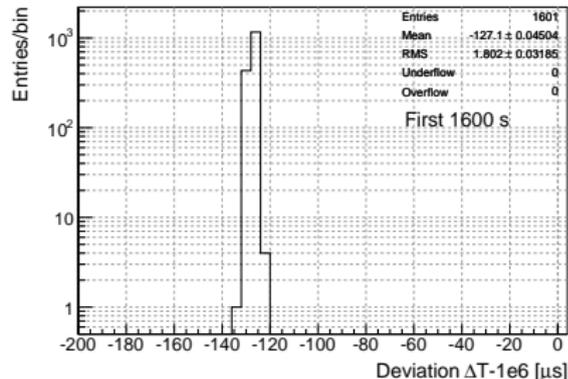
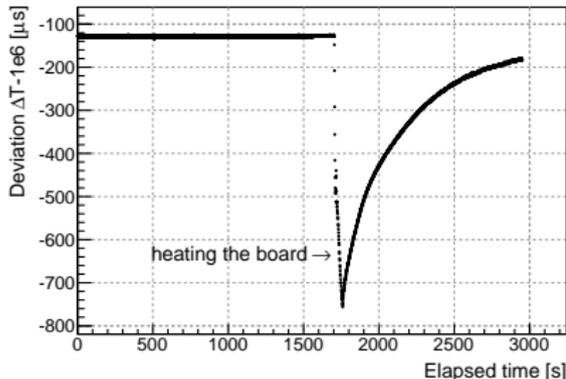
- ▶ Gli interrupt sono facili da usare, ma possono essere complicati da gestire
 - ▶ `delay()` non funziona all'interno di un interrupt
 - ▶ `millis()` non si incrementa
 - ▶ Le variabili utilizzate nell'interrupt devono essere dichiarate *volatile*
- ▶ Esercizio: leggere un tempo quando il pin 2 cambia stato.

```
volatile unsigned long t;  
volatile int state;  
const byte BUTTON = 2;
```

```
void switchPressed () // Interrupt Service Routine (ISR)  
{  
    t = micros();  
    state = digitalRead(BUTTON);  
    Serial.print(t); //prints time since program started  
    Serial.print(" "); // separator  
    Serial.println(state); // Pin state just after the trigger  
} // end of switchPressed
```

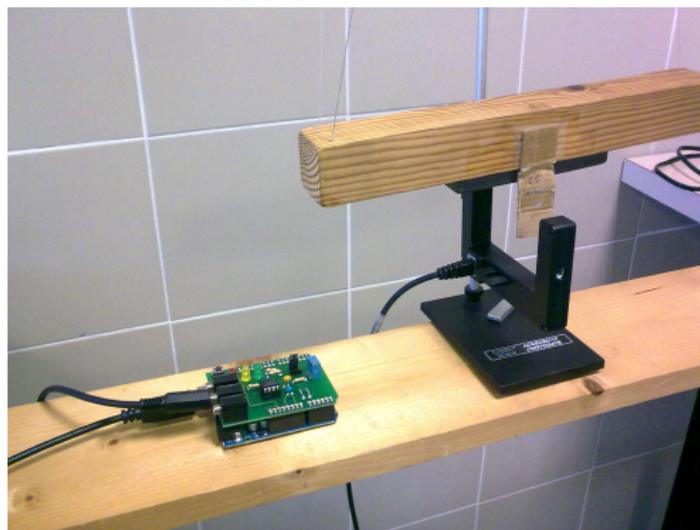
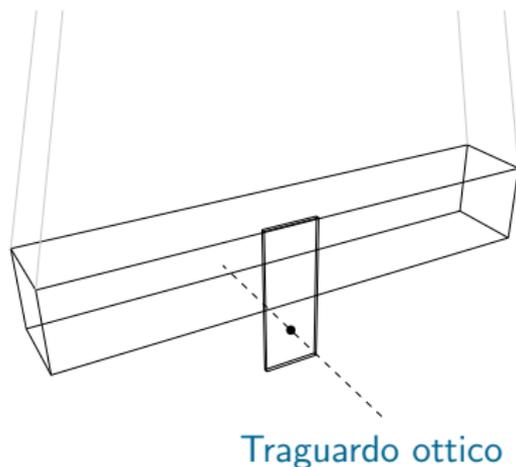
```
void setup ()  
{  
    Serial.begin(9600); // initialize serial communication  
    pinMode(BUTTON, INPUT_PULLUP);  
    attachInterrupt (digitalPinToInterrupt (BUTTON), switchPressed, CHANGE); // attach inter  
} // end of setup  
void loop () {} // loop doing nothing
```

MISURE DI TEMPO: PERFORMANCE



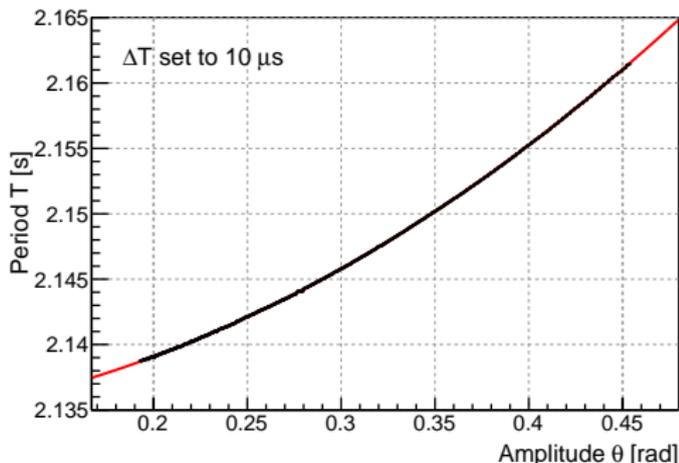
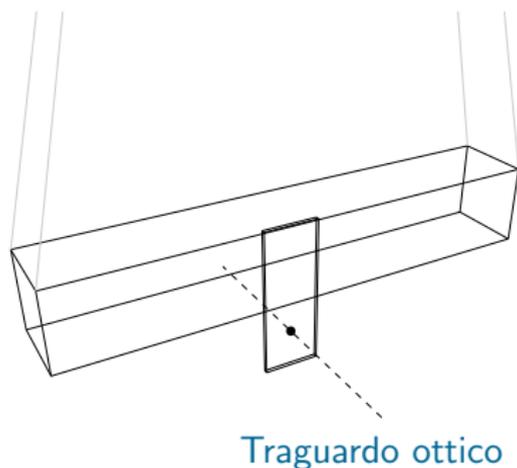
- ▶ Testato in laboratorio il 1PPS (1 pulse-per-second) di un GPS:
 - ▶ RMS dell'intervallo misurato tra due 1PPS successivi di 1.8 μs , non lontano da $4/\sqrt{12}$ μs .
 - ▶ Deviazione media dal valore nominale di ~ 100 μs (su 1 s) a temperatura ambiente.
- ▶ La granularità di 4 μs è vera
- ▶ Qual'è l'incertezza sulla singola misure di tempo?
 - ▶ Incertezza strumentale più piccola delle fluttuazioni statistiche tipiche
 - ▶ Fate tante misure; studiate media e RMS...

UN ESEMPIO: IL PENDOLO “DIGITALE”



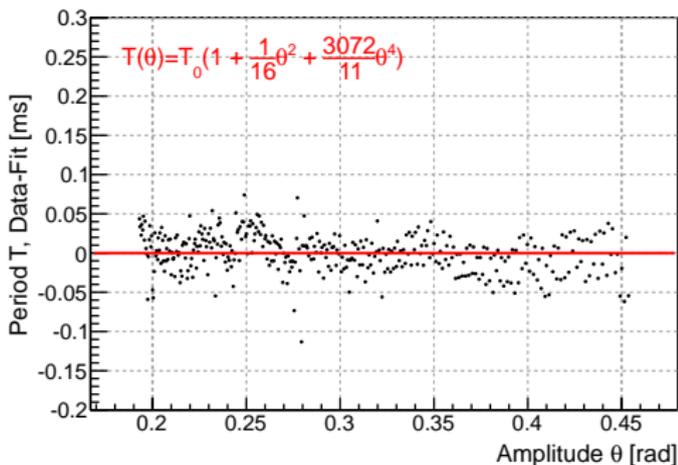
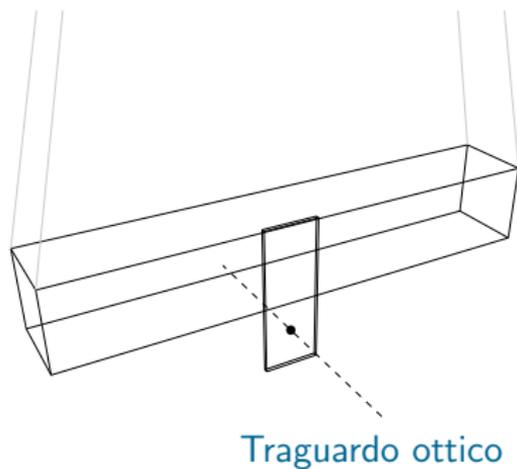
- ▶ Misura del periodo T e del tempo di transito t_T di una bandierina (di larghezza misurabile) nel punto più basso.
 - ▶ Misura dello smorzamento esponenziale (?)
- ▶ Trascurando le perdite di energia in una oscillazione possiamo stimare l'ampiezza θ_{\max} .
 - ▶ Misura dell'anarmonicità del pendolo.
 - ▶ Si apprezza chiaramente il termine in θ^4 !

UN ESEMPIO: IL PENDOLO “DIGITALE”



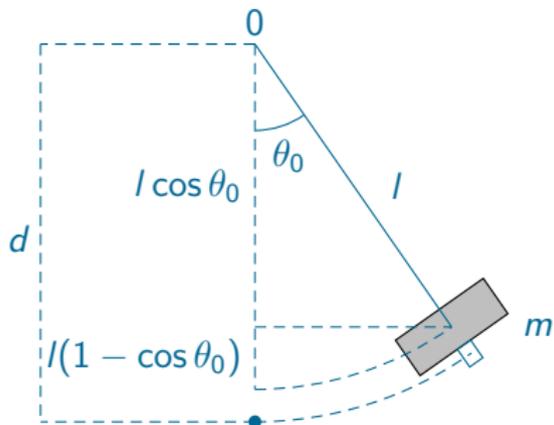
- ▶ Misura del periodo T e del tempo di transito t_T di una bandierina (di larghezza misurabile) nel punto più basso.
 - ▶ Misura dello smorzamento esponenziale (?)
- ▶ Trascurando le perdite di energia in una oscillazione possiamo stimare l'ampiezza θ_{\max} .
 - ▶ Misura dell'anarmonicità del pendolo.
 - ▶ Si apprezza chiaramente il termine in θ^4 !

UN ESEMPIO: IL PENDOLO “DIGITALE”



- ▶ Misura del periodo T e del tempo di transito t_T di una bandierina (di larghezza misurabile) nel punto più basso.
 - ▶ Misura dello smorzamento esponenziale (?)
- ▶ Trascurando le perdite di energia in una oscillazione possiamo stimare l'ampiezza θ_{\max} .
 - ▶ Misura dell'anarmonicità del pendolo.
 - ▶ Si apprezza chiaramente il termine in θ^4 !

IL PENDOLO QUADRIFILARE IN LABORATORIO 1



Formule utili:

$$v = (w/t_T) * (l/d)$$

conservazione energia:

$$mgl(1 - \cos \theta_0) = \frac{1}{2}mv_0^2$$

da cui:

$$\theta_0 = \arccos \left(1 - \frac{v_0^2}{2gl} \right)$$

- ▶ L'ambiente "plasduino": il nostro DAQ per gli studenti
 - ▶ Modulo "pendulum"
- ▶ Per questa esperienza salva su file di testo:
 1. indice progressivo della transizione
 2. flag del tipo di transizione: 0/1 se la bandierina entra/esce
 3. tempo assoluto della transizione t
- ▶ Da questi dati si possono ricavare il periodo di oscillazione T , ed il tempo di transito t_T .
 - ▶ Dopo aver misurato la larghezza w della bandierina
 - ▶ Per i da 3, in passi di 4:
$$t_{\text{abs}} = (t_i + t_{i-1})/2$$
$$t_T = (t_i - t_{i-1})$$
$$T = (t_{i+1} - t_{i-3})$$
 (ci sono altri modi?)