L'ambiente Arduino

From Arduino Home Page (***www.arduino.cc/***):

- ✓ *Arduino is an open-source electronics platform based on easy-to-use hardware and software. It's intended for anyone making interactive projects.*

From Arduino Home Page (***www.arduino.cc/***):

- ✓ *Arduino is an open-source electronics platform based on easy-to-use hardware and software. It's intended for anyone making interactive projects.*

Schede basate su micro-controllori in grado di:

- ✓ ricevere segnali analogici (ad esempio, da sensori esterni);
- ✓ monitorare (controllare) dispositivi esterni attraverso segnali digitali di ingresso (uscita);
- ✓ implementare contatori e timer;
- ✓ comunicare con dispositivi esterni attraverso diversi tipi di interfaccia (ad esempio, seriale o Ethernet);
- ✓ essere alimentanti da dispositivi esterni, USB o Ethernet
- ✓ …

From Arduino Home Page (***www.arduino.cc/***):

- ✓ *Arduino is an open-source electronics platform based on easy-to-use hardware and software. It's intended for anyone making interactive projects.*

Il micro-controllore può essere programmato in un ambiente *software* che:

- ✓ è basato sul linguaggio C++ (il compilatore non è competo);
- ✓ include il programmatore;
- ✓ offre un controllo della sintassi;
- ✓ include un *serial monitor* attraverso il bus USB;
- ✓ …

Arduino hardware

- La scheda usata in laboratorio è il modello **Arduino Uno** (*entry level*)
  - utilizza il micro-controllore ATmega328
  - 6 ingressi analogici
  - un sensore di temperatura interno
  - 14 linee digitali configurabili come input o output (6 possono essere usati come uscita PWM)
  - oscillatore interno a 16 MHz (*internal clock*)
  - un connettore per l'alimentazione
  - un connettore USB tipo B (*peripheral device*)
  - un LED collegato alla linea digitale 13
  - ...

Arduino hardware

- La scheda usata in laboratorio è il modello **Arduino Uno** (*entry level*)
  - utilizza il micro-controllore ATmega328

- Low-power 8-bit micro-controller
- Nominal operating voltage: 5 V
- Flash Memory (for program): 32 kByte
- SRAM (volatile memory): 2 kByte
- EEPROM (nonvolatile memory): 1 kByte
- Pre-burned with a boot-loader (0.5 kByte of flash memory used) that allows programming without external hardware (alternatively, ICSP header can be used)
- …

Arduino hardware

- La scheda usata in laboratorio è il modello **Arduino Uno** (*entry level*)
  - utilizza il micro-controllore ATmega328

- Convertitore Analogico/digitale (ADC) interno
  - ✓ Sampling frequency: max 76.9 kSa/s
  - ✓ Resolution: 10 bit (8 bit for f_S > 15 kSa/s)
  - ✓ INL: 0.5 LSB; absolute uncertainty: ± 2 LSB
  - ✓ Selectable external (V_{CC}) or internal (1.1 V) voltage reference
  - ✓ Free-running or single conversion mode
  - ✓ Interrupt on ADC conversion complete
  - ✓ …

Arduino hardware

- La scheda usata in laboratorio è il modello **Arduino Uno** (*entry level*)
  - utilizza il micro-controllore ATmega328

- Comparatore analogico interno
  - ✓ Compares input values AIN0 (positive) and AIN1 (negative)
  - ✓ Analog Comparator Output (ACO) is set when AIN0 > AIN1
  - ✓ ACO can be used to trigger a Timer/Counter
  - ✓ Other analogue inputs can be selected as negative inputs (only with the ADC switched off)
  - ✓ …

Arduino hardware

- La scheda usata in laboratorio è il modello **Arduino Uno** (*entry level*)
  - utilizza il micro-controllore ATmega328
  - 6 ingressi analogici

- Input range: 0 – V_{REF}
- Bandwidth: 38.5 kHz
- Input resistance: 100 MΩ
- 6 channels single-ended multiplexer (MUX)
  - ✓ Once AD conversion starts, channel selection is locked to ensure a correct ADC conversion time
  - ✓ ADC single conversion mode: always select the channel before starting the conversion
  - ✓ ADC free-running mode: wait for the first conversion to complete, then change the channel selection

Arduino hardware

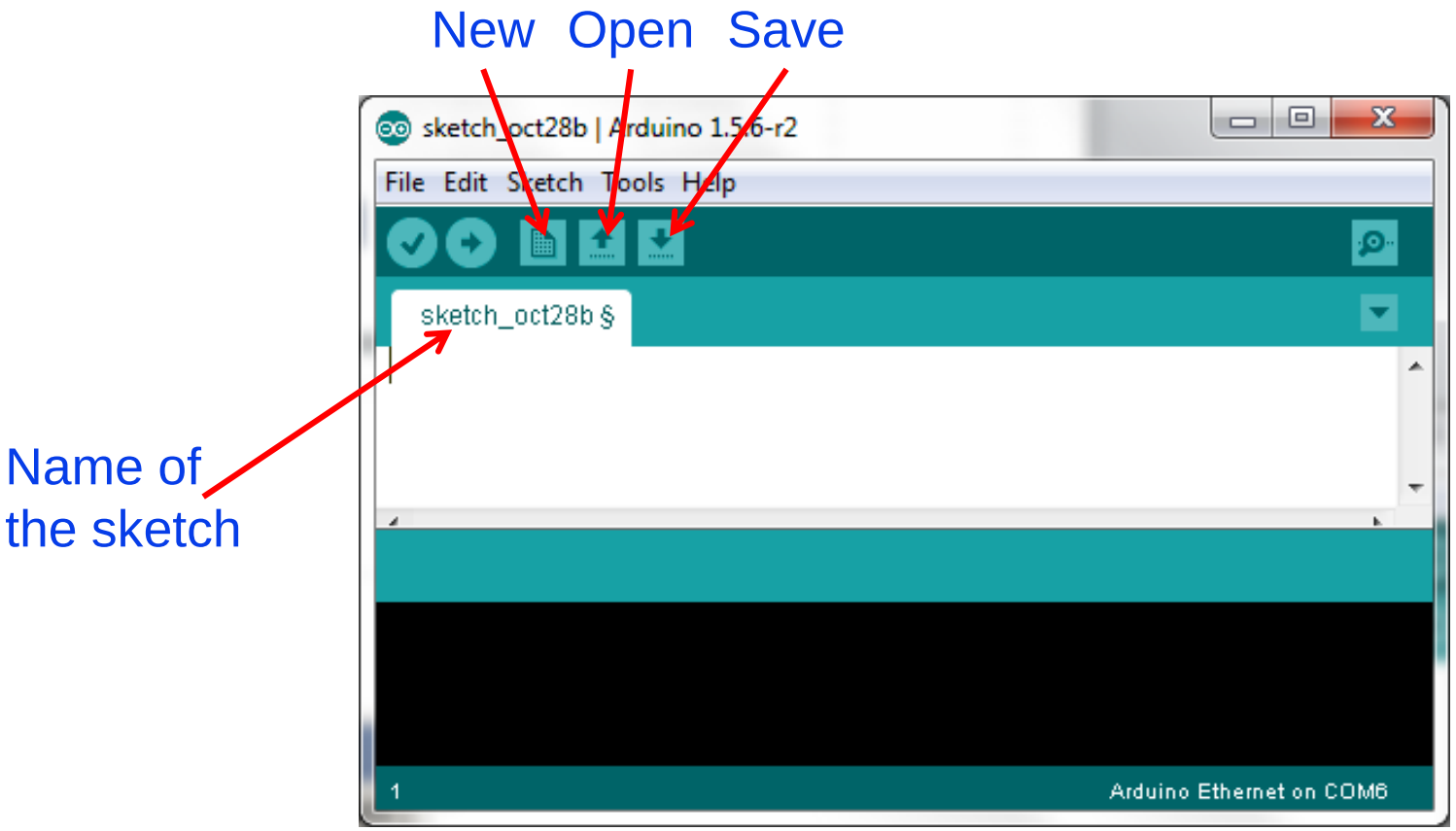
Arduino hardware

Arduino software

- Un programma Arduino (***sketch***) può essere creato, compilato e "scaricato" sul micro-controllore usando l'ambiente di sviluppo integrato

Arduino software

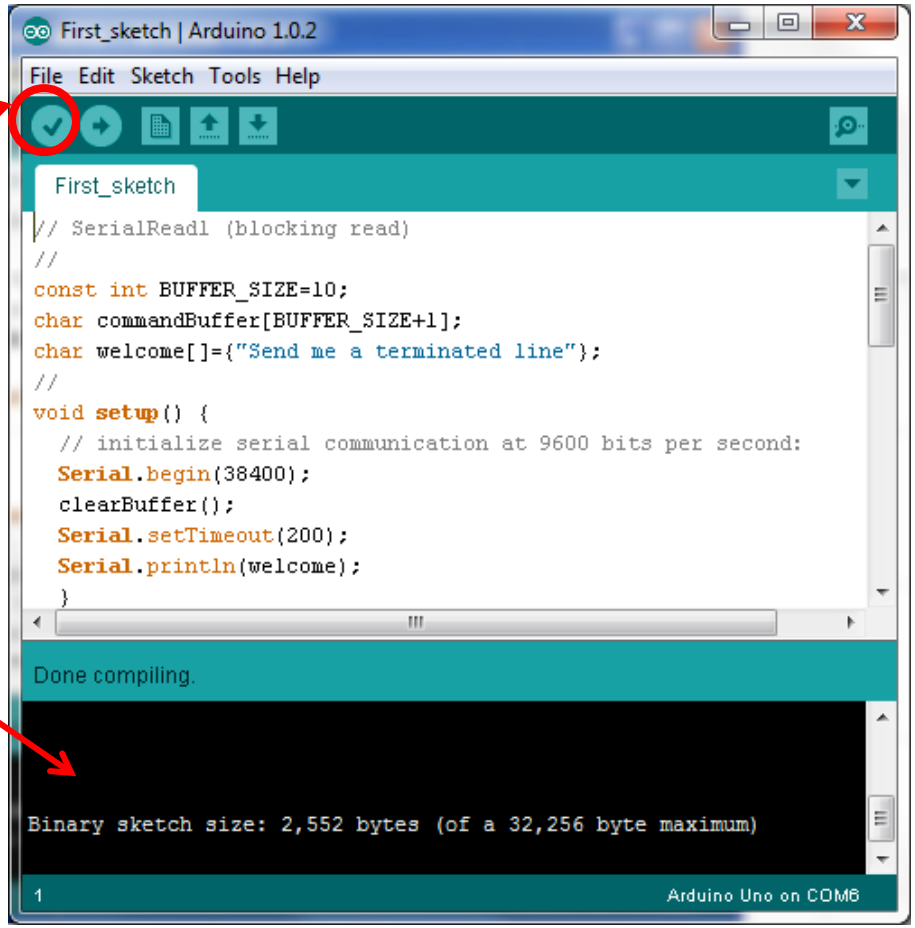
- L'ambiente di sviluppo



Arduino software

- L'ambiente di sviluppo

Questo bottone avvia il compilatore



Questa finestra mostra gli eventuali errori, la dimensione dello *sketch* e la memoria disponibile per il programma (32 kByte)

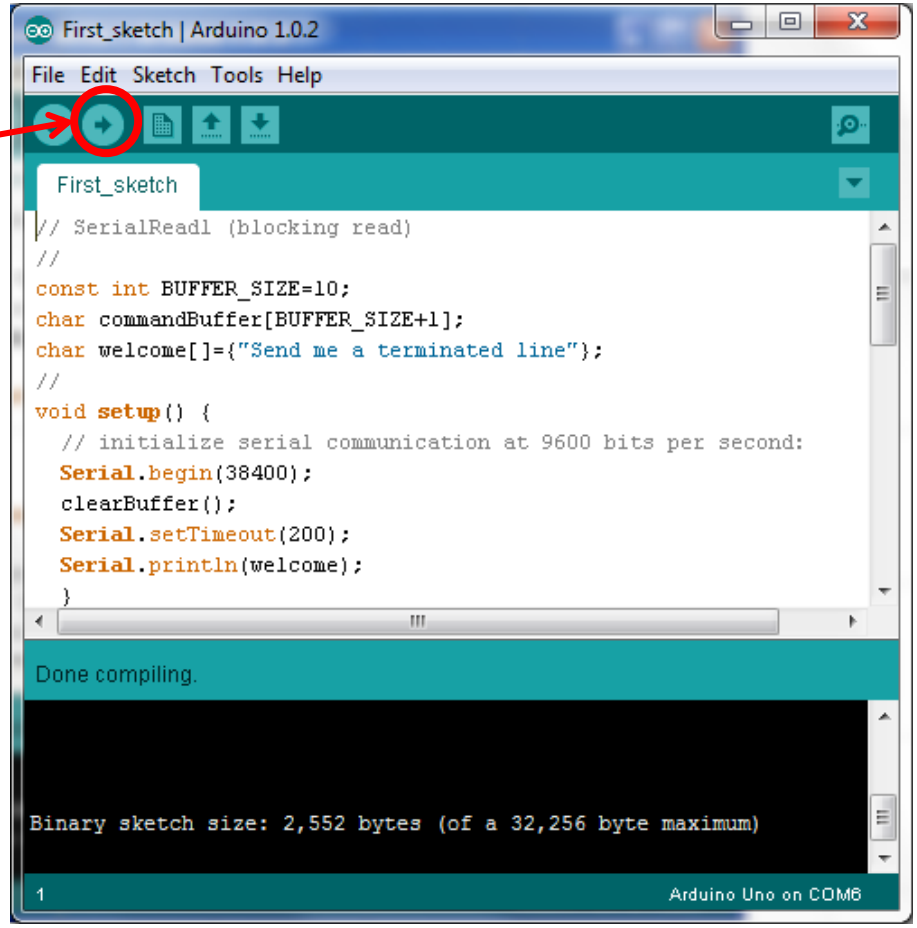
Arduino software

- L'ambiente di sviluppo

Questo bottone avvia il compilatore e, in assenza di errori, invia il codice al micro-controllore

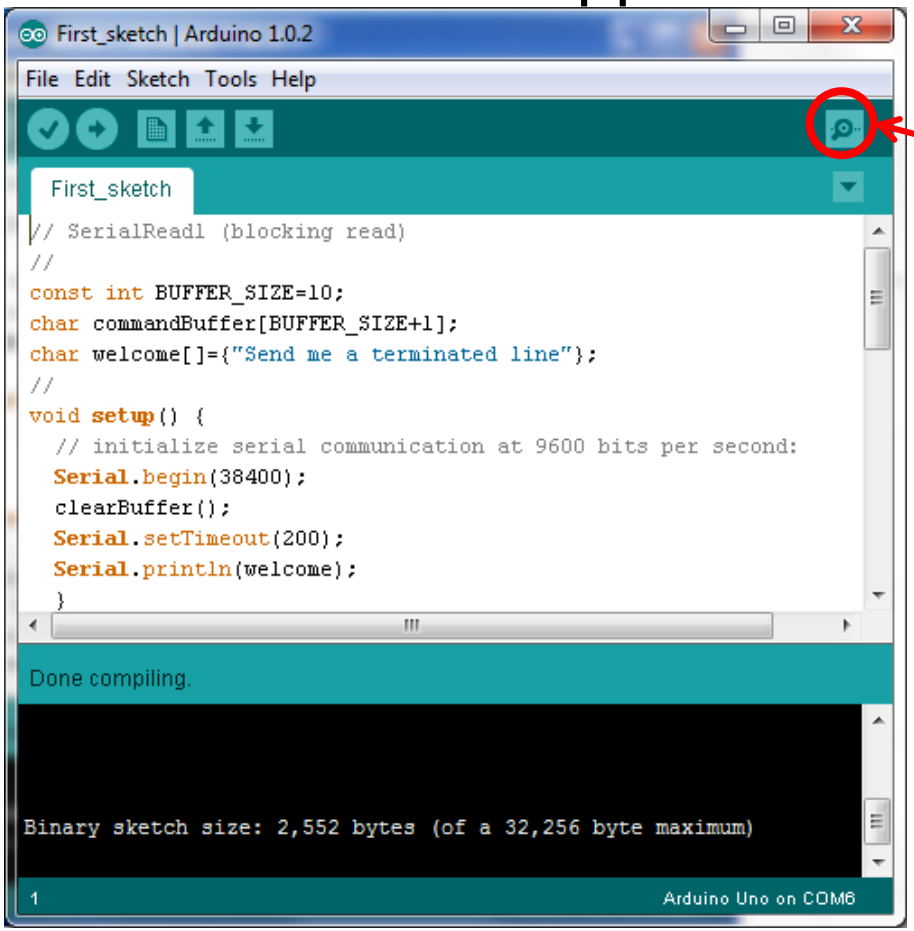
- WARNING: lo sketch non è salvato

Il micro-controller è programmato attraverso l'interfaccia USB



Arduino software

- L'ambiente di sviluppo



Questo bottone apre il "*serial monitor*" permettendo la comunicazione tra PC e micro-controller via USB

- ↳ Avvia anche l'esecuzione del programma sul micro-controllore

Arduino software

- L'ambiente di sviluppo
  - Struttura di uno *sketch*

```
/*   start comment
*/   end comment

header

void setup() {
                    function body }

//    comment line

void loop() {
                    function body }
```

Arduino software

- L'ambiente di sviluppo
  - Struttura di uno *sketch*

```
/*   start comment
*/   end comment

header

void setup() {
                function body }

//   comment line

void loop() {
                function body }
```

Due modi per inserire I commenti:

- ✓ Multiline (/* ... */)
- ✓ Single line (//)

Arduino software

- L'ambiente di sviluppo
  - Struttura di uno *sketch*

```
/*   start comment
*/   end comment

header

void setup() {
                    function body }

//    comment line

void loop() {
                    function body }
```

Sezione dove le variabili globali possono essere definite e inizializzate

- ✓ Esempi:
  - int sensorPin = A0;
  - float voltage = 0;

Arduino software

- L'ambiente di sviluppo
  - Struttura di uno *sketch*

```
/*   start comment
*/   end comment

header

void setup() {
                function body }

//   comment line

void loop() {
                function body }
```

Funzioni **sempre** presenti in uno *sketch*

- Entrambe le funzioni devono essere incluse, anche se rimangono vuote

Arduino software

- L'ambiente di sviluppo
  - Struttura di uno *sketch*

```
/*   start comment
*/   end comment

header

void setup() {
                    function body }

//    comment line

void loop() {
                    function body }
```

Il codice nella funzione ***setup*** è eseguito solo una volta all'avvio del programma (dopo l'accensione o il *reset*)

- ✓ Esempi d'uso:
  - impostare le linee di I/O
  - inizializzare porte di comunicazione
  - inizializzare variabili

Arduino software

- L'ambiente di sviluppo
  - Struttura di uno *sketch*

```
/*   start comment
*/   end comment

header

void setup() {
                    function body }

//    comment line

void loop() {
                    function body }
```

Il codice nella funzione ***loop*** è ripetuto continuamente

- ✓ è il codice che permette di controllare la scheda

Arduino software

- L'ambiente di sviluppo
  - Struttura di uno *sketch*

```
//    comment line

void loop() {
                    function body }

void serialEvent() {
                    function body }

void clearBuffer() {
                    function body }
```

È possibile aggiungere altre funzioni in base al codice da eseguire.

Arduino software

- Un esempio di *sketch*

```
void setup() {
        // initialize serial communication
        Serial.begin(9600);
}
void loop() {
        // read the input on analog pin 0:
        int sensorValue = analogRead(A0);
        // print out the read value:
        Serial.println(sensorValue);
        delay(1000); // stop 1 s and repeat
}
```

Arduino software

- L'ambiente di sviluppo
  - Versione base: uso delle *built-in functions*, ma con alcune limitazioni
    - ✓ analogReference(INTERNAL)
    - ✓ analogRead(A0)
    - ✓ EEPROM.write(address,calconst) → EEPROM library
    - ✓ myFile.write(voltarray,100) → SD library
    - ✓ ...
  - Controllo completo delle funzionalità del micro-controller agendo sui suoi registri interni

- Riferimenti utili
  - ***Using Arduino boards in measurements for dummies*** (M. Parvis)
    - Disponibile sulla pagina web del corso
  - Sezione **LEARNING > REFERENCE** sulla *home page* Arduino
    - www.arduino.cc/

Misura di temperatura con sensore LM335 e scheda Arduino Uno

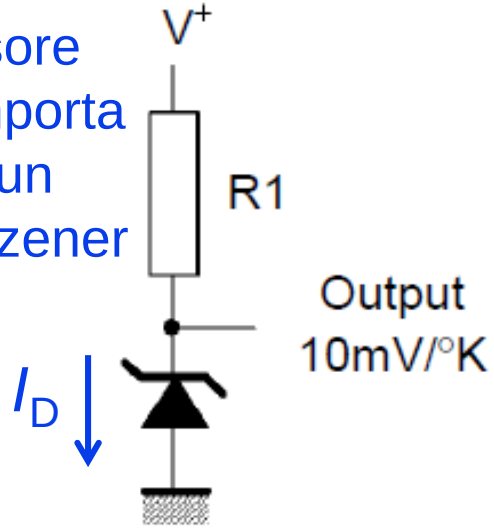
Obiettivo dell'esperienza

- Sviluppare un termometro digitale usando
  - un sensore elettronico di temperatura
    - ✓ LM335 (costo ≈ 1 EUR)
  - una scheda Arduino
    - ✓ Arduino UNO (costo ≈ 20 EUR)
- Progettare il circuito di condizionamento del sensore
- Stimare l'incertezza attesa

Il sensore

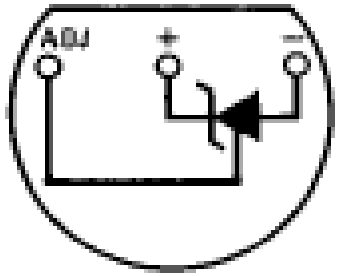
- National Semiconductor modello LM335
  - Sensibilità nominale S = 10 mV/K
  - Uscita riferita allo zero assoluto
    - V_{out} = 0 V @ 0 K ≡ -273.15 °C
    - $\delta T = \pm 1$°C @ 25°C (typ.)
    - Funz. Tra -40°C e +100°C

Il sensore si comporta come un diodo zener



La corrente I_D deve essere inclusa nel campo (0.4 ÷ 5) mA

TO-92
Plastic Package



Bottom view

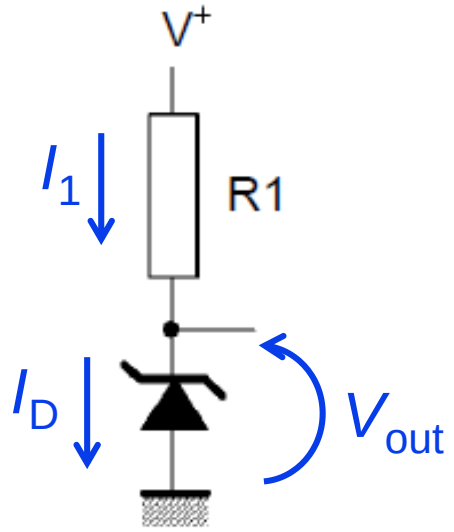
Temperatura di funzionamento tra -40°C a 100°C

Circuito di condizionamento

- Tensione di alimentazione fornita dalla scheda Arduino (uscita 5 V)
  - Verificare se l'uscita 5 V è in grado di alimentare il sensore
- Stimare il valore di R_1

$$I_D \approx I_1 = \frac{V^+ - V_{out}}{R_1}$$

I_D dipende da V_{out}, che a sua volta dipende dalla temperatura in misura



Circuito di condizionamento

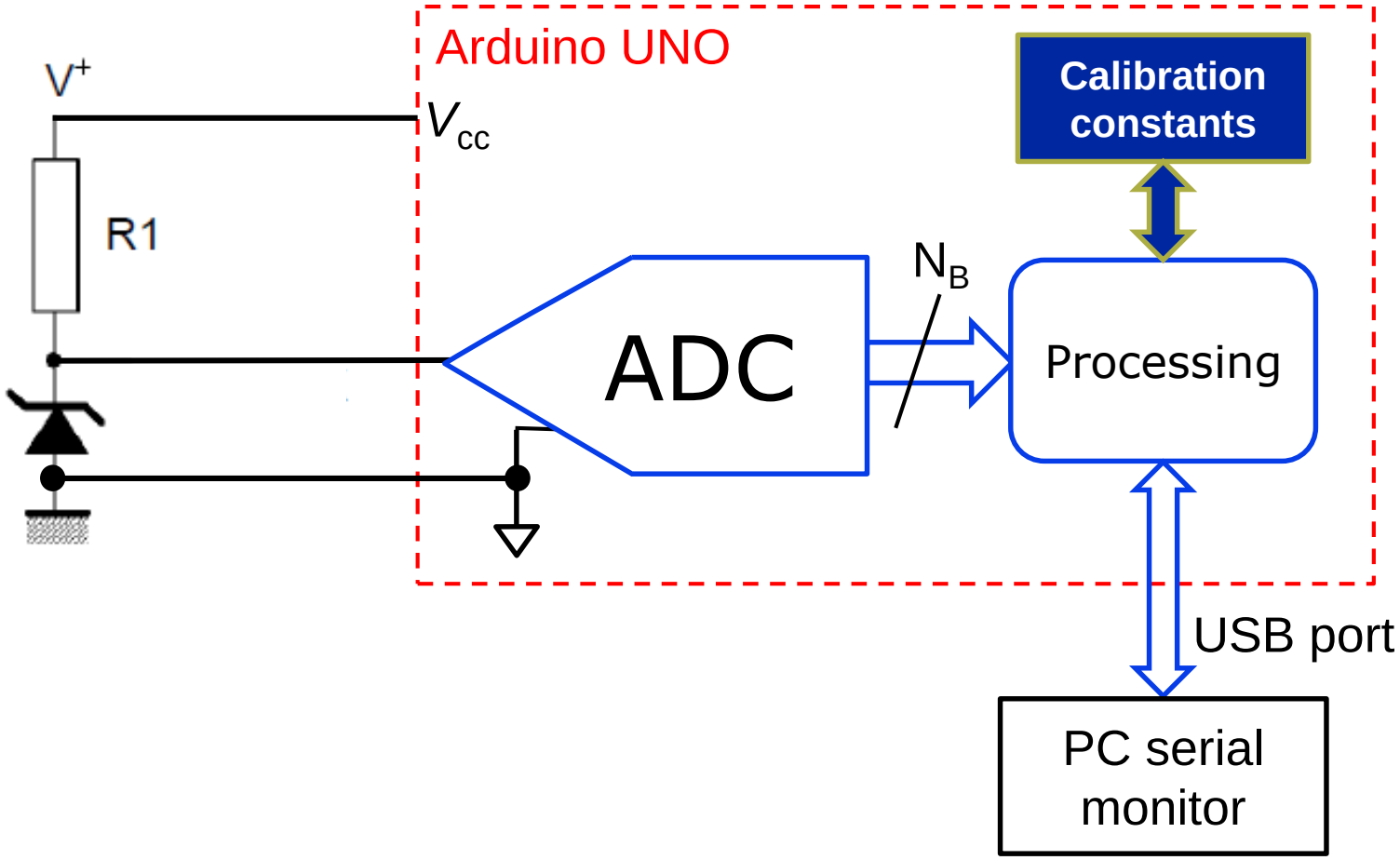
- Stimare il valore di R_1
  - $V^+ = 5$ V
  - Campo di temperatura: (5 ÷ 75) °C

$$I_{\mathrm{D,min}} = \frac{V^+ - V_{\mathrm{out,max}}}{R_1} > 0.4\ \mathrm{mA}$$

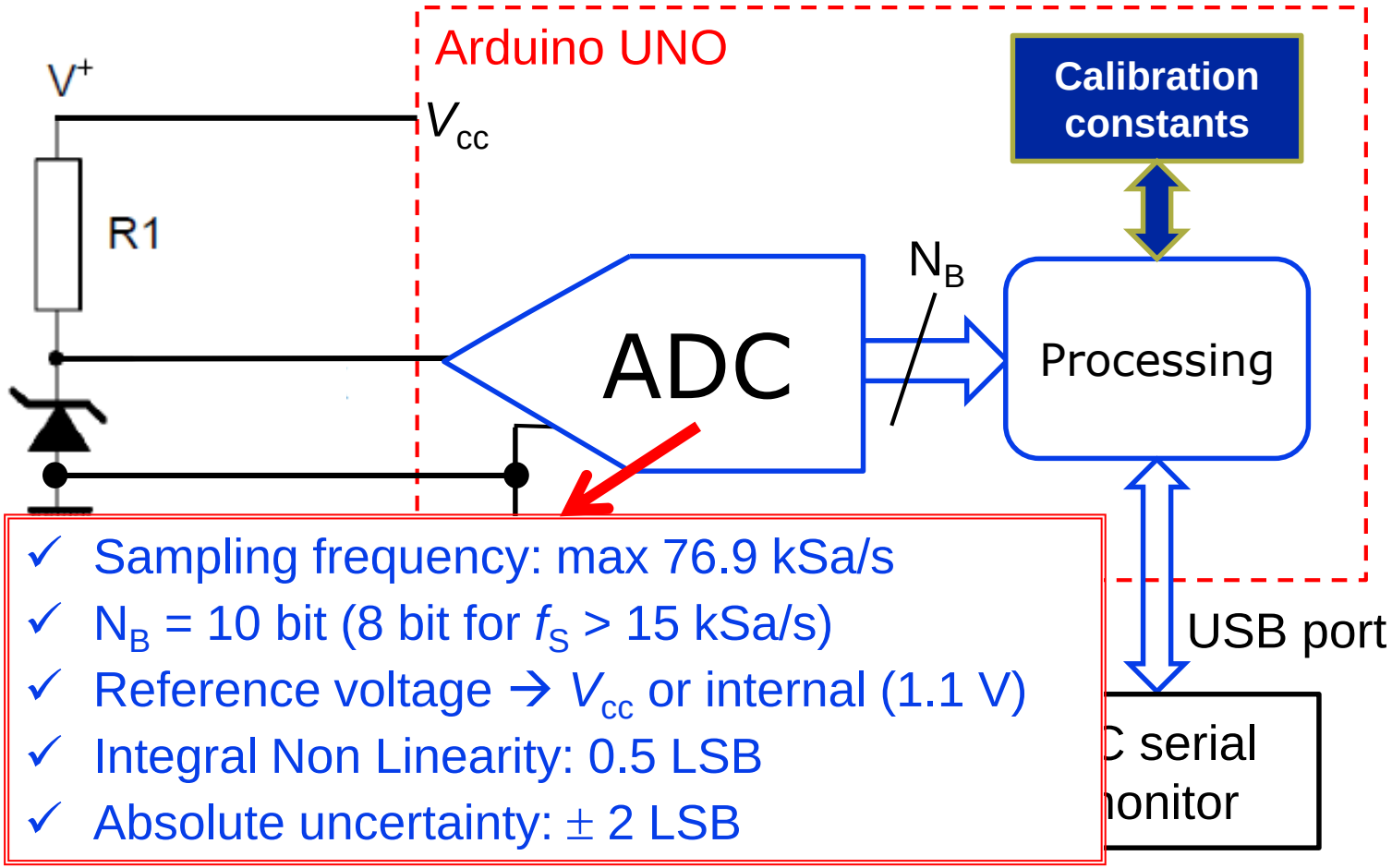
$$I_{\mathrm{D,max}} = \frac{V^+ - V_{\mathrm{out,min}}}{R_1} < 5\ \mathrm{mA}$$

- …

Schema di principio



Schema di principio



- ✓ Sampling frequency: max 76.9 kSa/s
- ✓ N_B = 10 bit (8 bit for f_S > 15 kSa/s)
- ✓ Reference voltage → V_{cc} or internal (1.1 V)
- ✓ Integral Non Linearity: 0.5 LSB
- ✓ Absolute uncertainty: ± 2 LSB

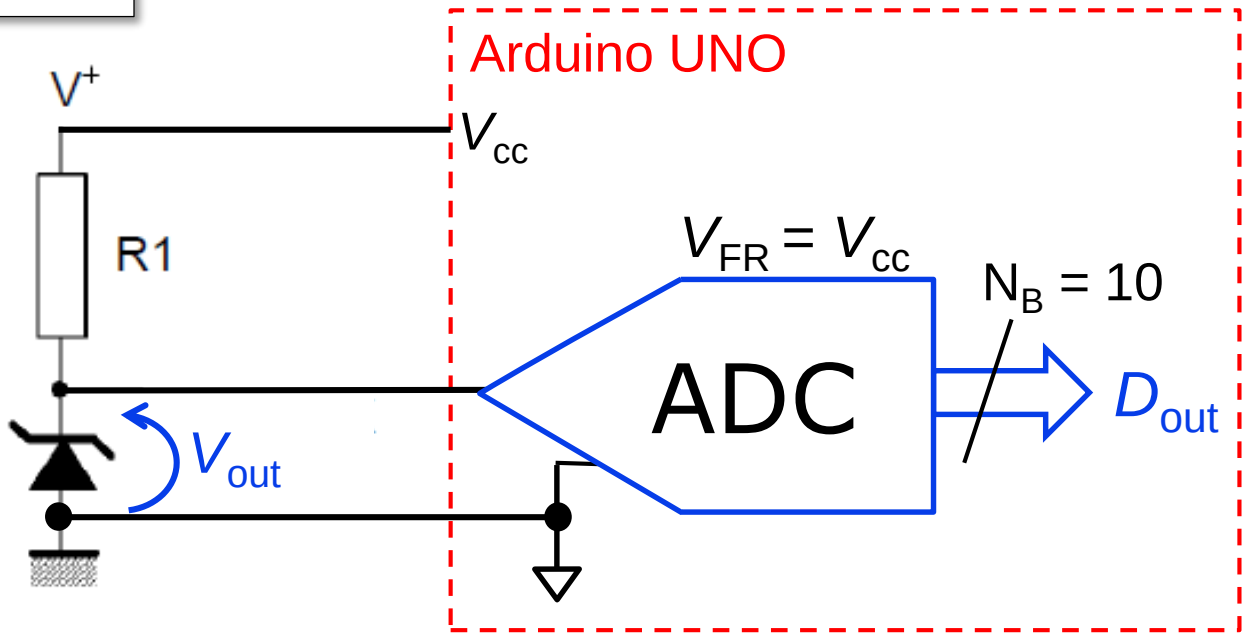
Schema di principio

- Campo di temperatura: (5 ÷ 75) °C
  - ≈ (278 ÷ 348) K
    - V_{out}: (2.78 ÷ 3.48) V
- Tensione di riferimento: V_{cc}
  - V_{cc}: dalla porta USB
    - USB 2.0 → V_{cc} = (5 ± 0.25) V
    - USB 3.0 → V_{cc} tra 4.45 V e 5.25 V

Catena di misura

$$D_{out} = \frac{V_{out}}{V_q} = \frac{V_{out}}{V_{FR}} \cdot 2^{N_B}$$

$$D_{out} = \frac{S \cdot T}{V_{FR}} \cdot 2^{N_B}$$



$$T = D_{out} \cdot \frac{V_{FR}}{2^{N_B}} \cdot \frac{1}{S}$$

FUNZIONE DI TARATURA

Catena di misura

$$T = D_{\text{out}} \cdot \frac{V_{\text{FR}}}{2^{N_B}} \cdot \frac{1}{S}$$

- Stima dell'incertezza (modello deterministico)

$$\delta T = \left|\frac{\partial T}{\partial D_{\text{out}}}\right| \cdot \delta D_{\text{out}} + \left|\frac{\partial T}{\partial V_{\text{FR}}}\right| \cdot \delta V_{\text{FR}} + \left|\frac{\partial T}{\partial S}\right| \cdot \delta S$$

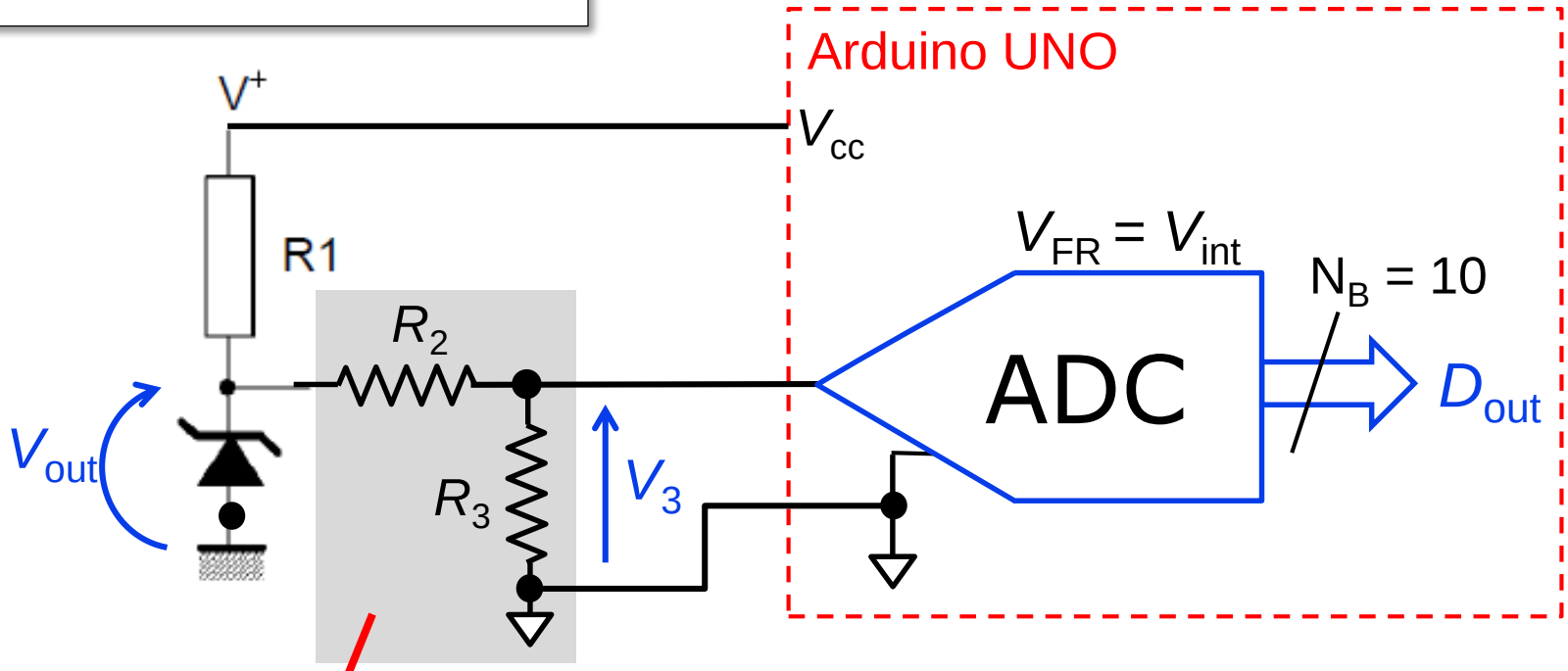
$$\delta T = \frac{V_{\text{FR}}}{S \cdot 2^{N_B}} \cdot \delta D_{\text{out}} + \frac{D_{\text{out}}}{S \cdot 2^{N_B}} \cdot \delta V_{\text{FR}} + \delta T^{\text{sensor}}$$

$\delta T^{V_{\text{FR}}}$ → 17.4 K !!! (per $D_{\text{out,max}}$ = 713)

Catena di misura

- Possibili interventi per ridurre l'incertezza
  - ✓ Usare il riferimento interno dell'ADC (1.1 V)
    - ↳ Necessario attenuare il segnale di uscita del sensore
    - ↳ V_{int} = (1.1 ± 0.1) V ☹☹☹
  - ✓ Misurare il riferimento di tensione dell'ADC
    - ↳ V_{cc} oppure V_{int}?
    - ↳ La prima scelta non richiede un partitore di tensione, **ma la costante di taratura è legata al PC usato**

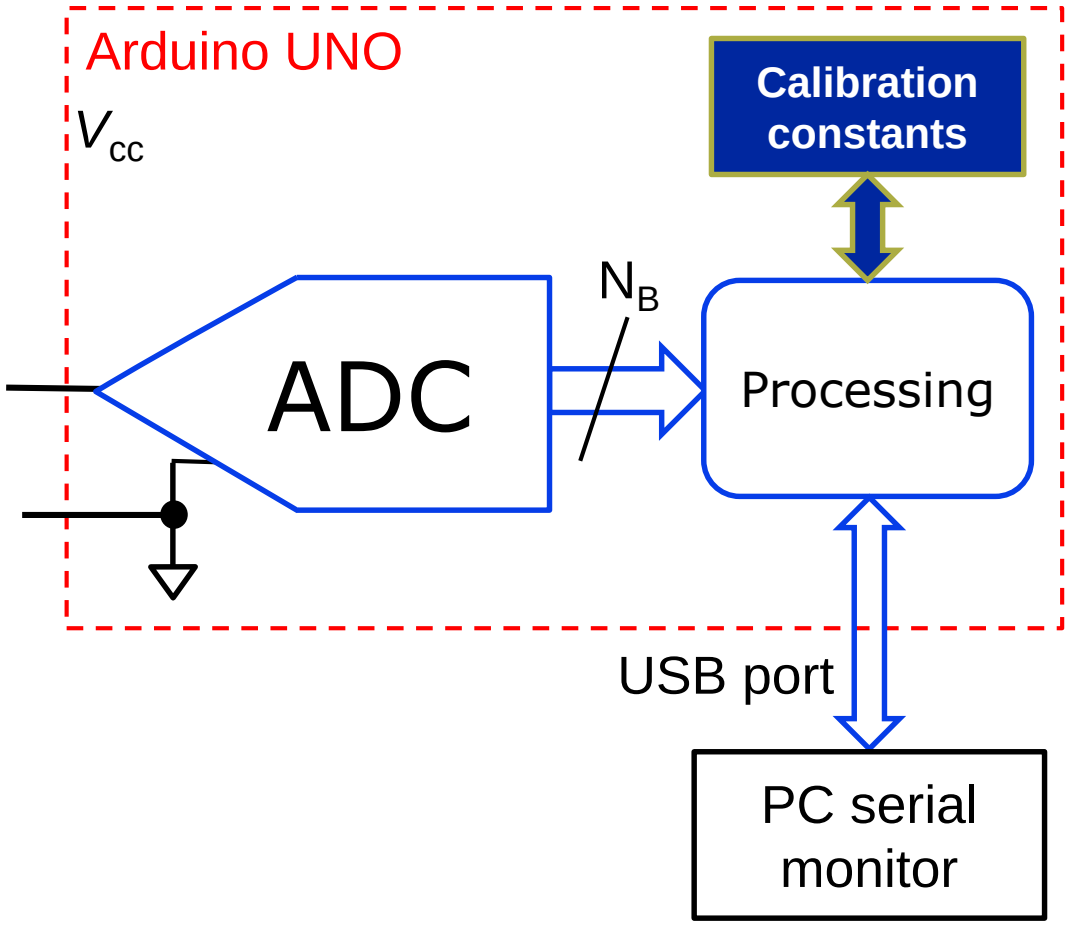
Schema modificato



Partitore di tensione

- ✓ Minimizzare l'effetto di carico
- ✓ Caratterizzare il suo fattore di attenuazione

Firmware del micro-controllore



- Gestire la comunicazione seriale con il PC
- Configurare l'ADC
- Acquisire il segnale di tensione
- Implementare la funzione di taratura
- Fornire le misure di temperatura

Firmware del micro-controllore

- Gestire la comunicazione seriale
  - Fissare il *baud rate*, usare la *built-in function* serialEvent(), …
    - Vedere capitoli 3, 4 e 5 della guida *Using Arduino boards in Measurements for dummies*
- Configurare l'ADC
- Acquisire il segnale di tensione
  - *built-in functions* analogReference() e analogRead()
    - Vedere capitolo 6 della guida

Firmware del micro-controllore

- Implementare la funzione di taratura
  - Convertire il codice di uscita dell'ADC in misura di temperatura
    - Usare le opportune costanti di taratura (V_{cc} o V_{int}, fattore di attenuazione)
    - Stimare l'incertezza di misura
- Fornire le misure di temperatura
  - Inviare i risultati al *serial monitor* via USB

Caratterizzazione del sistema

- Specificare:
  - ✓ Campo di temperatura
  - ✓ Risoluzione
  - ✓ Incertezza
    - *unadjusted*
    - *adjusted* (ADC *offset* e *gain error*)
  - ✓ Dimensione del *firmware*