Laboratorio di Segnali e Sistemi
- Esercitazione -8 -
Familiarizzazione con arduino

Claudio Luci

SAPIENZA
UNIVERSITÀ DI ROMA

last update : 070117

Argomenti dell'esercitazione:

- Familiarizzazione con Arduino:
  - blinking LED
- Familiarizzazione porta seriale:
  - modulazione PWM
- Familiarizzazione DAC:
  - Calibrazione DAC
  - Circuito adattatore segnale d'uscita
  - Generatore di rumore (facoltativo)
- Familiarizzazione ADC
  - Calibrazione ADC
  - Campionamento di una forma d'onda

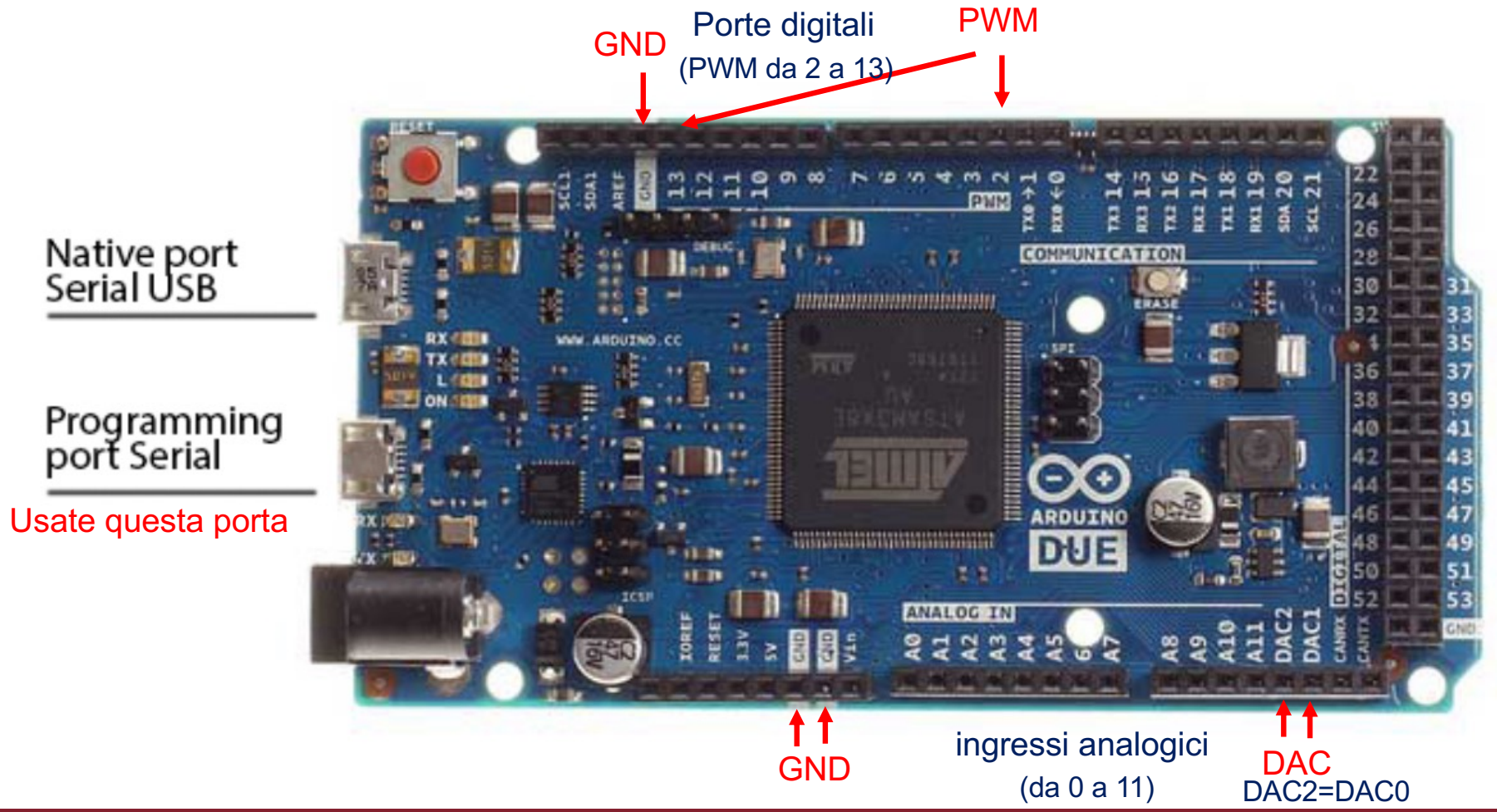
Presa dalla relazione del gruppo 26 del 2020

Figura 1: Arduino re d'Italia dal 1002 al 1014.

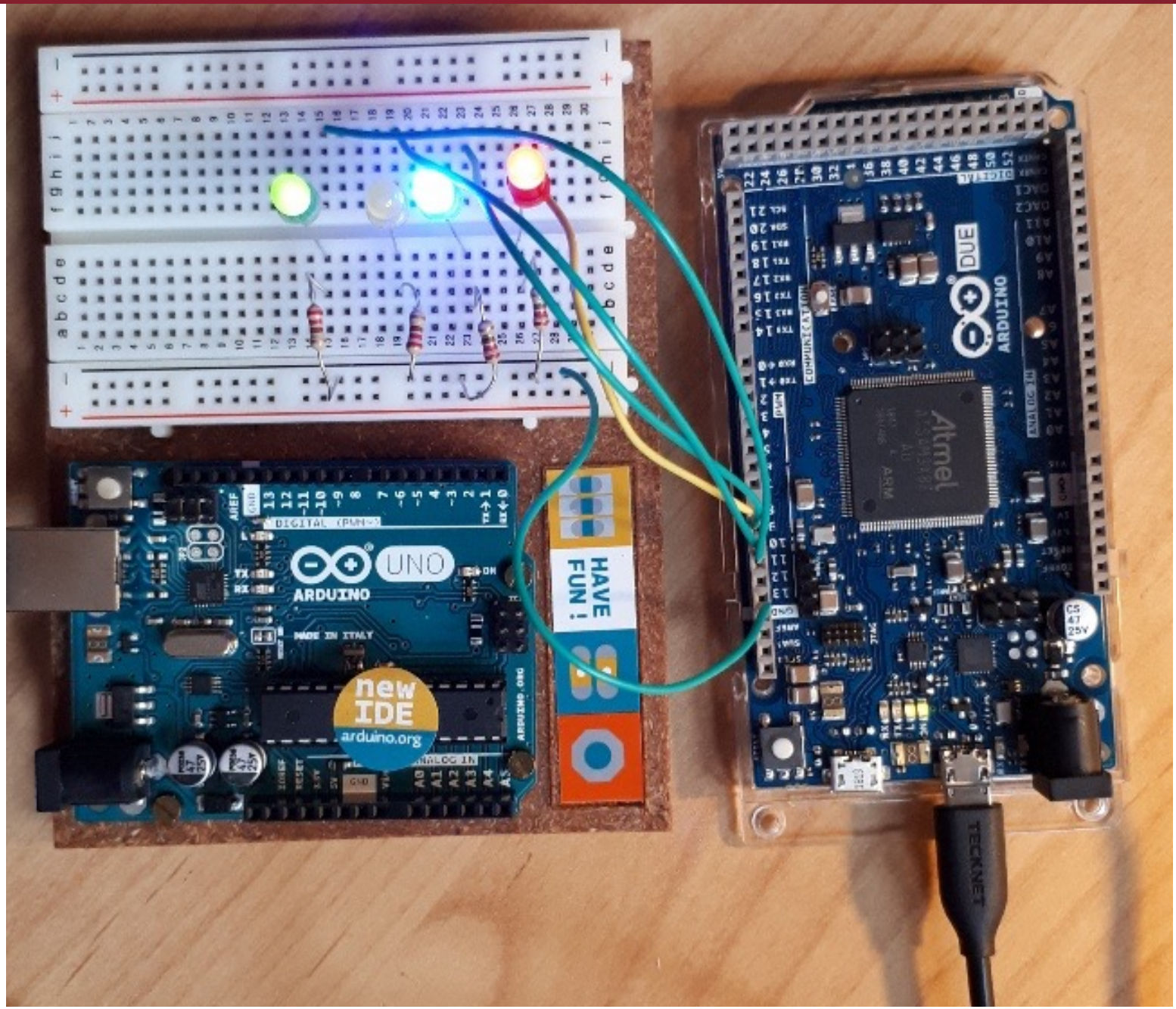
Riportate nella relazione gli sketch che avete utilizzato

Arduino

- L'obiettivo di questa esperienza è di familizzare con la scheda Arduino Due, con il microcontrollore ATMEL SAM3X8E e con il relativo software.
- È bene avere sempre disponibile per consultazione il sito internet di Arduino (https://www.arduino.cc)
- I circuiti da utilizzare possono essere montati sulla consueta scheda sperimentale, connettendola ad Arduino Due mediante opportuni ponticelli.
- Potete trovare le varie funzioni di Arduino Due su: https://www.arduino.cc/reference/en/

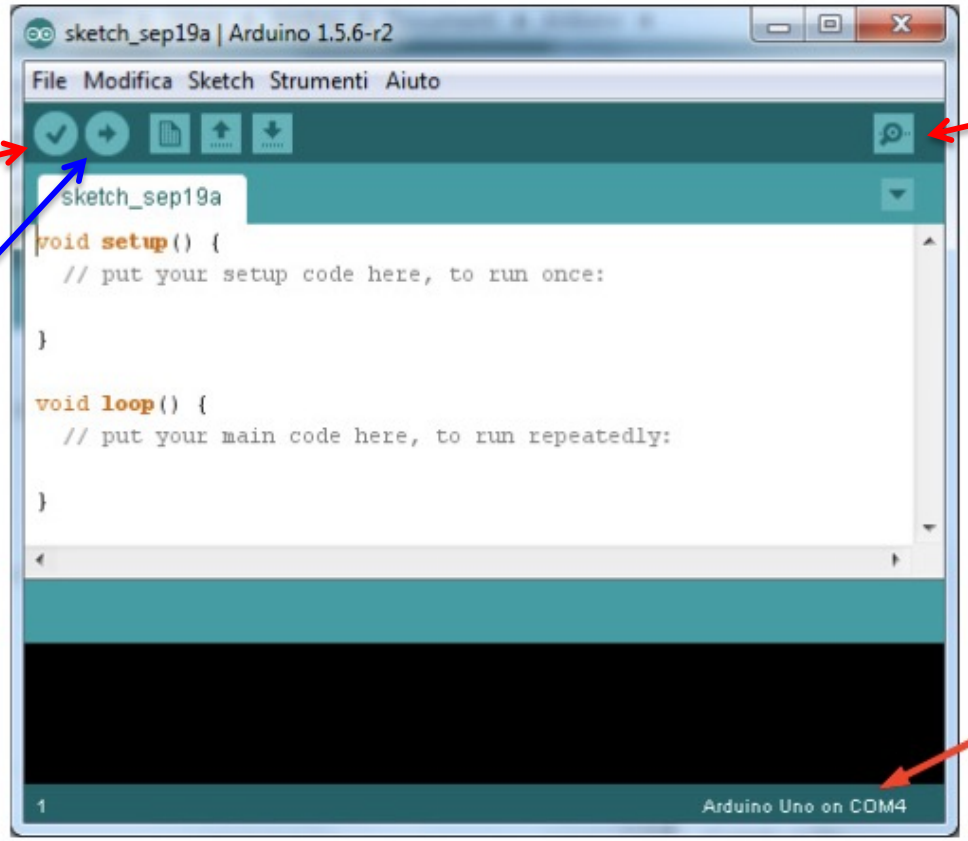


arduino uno vs arduino due



Come iniziare

- La scheda Arduino DUE viene gestita dall'applicazione Arduino, già istallata sul PC. Questa applicazione consente di scrivere un programma in linguaggio C, compilarlo, scaricarlo sul microcontrollore e mandarlo in esecuzione.
- All'avvio dell'applicazione (dopo aver collegato la scheda al PC tramite una delle porte USB) si apre una finestra che offre un *framework* per la scrittura dello *sketch* (come è chiamato nel gergo di Arduino).
- Se non avete il collegamento sul desktop potete avviare da *C:/Programmi/Arduino/arduino*



compila

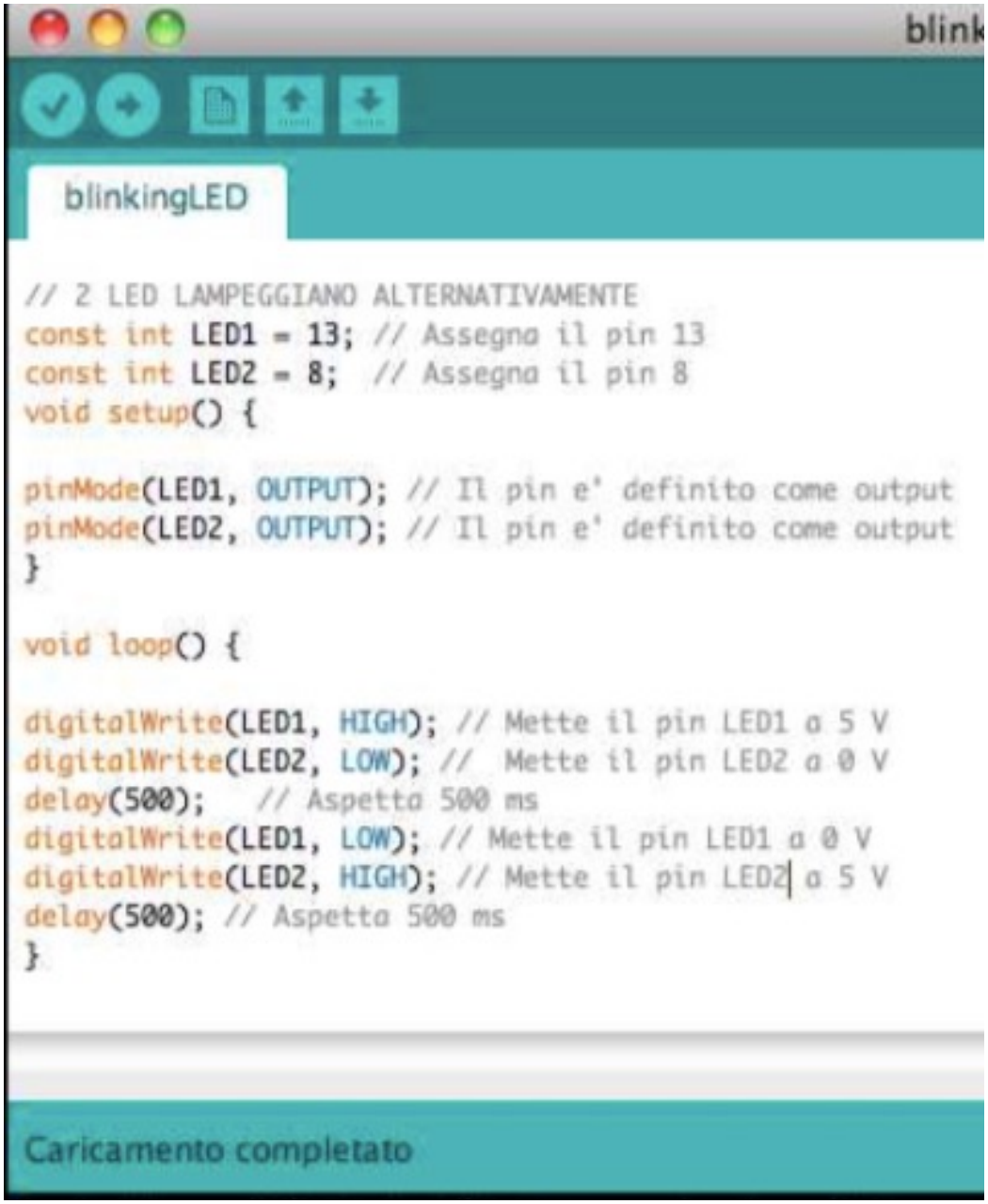
Compila, carica e run

Apre il serial monitor per eseguire operazioni di input/output

Dopo aver scritto il programma esso può essere compilato utilizzando il pulsante apposito (il primo in alto a sinistra):
il risultato della compilazione viene riportato nella piccola finestra nera, dove vengono anche segnalati gli eventuali errori presenti.

Porta USB utilizzata

Esempio di programma: blinking led



Traduzione in C dello sketch

```
#include "Wprogram.h";
void setup();
void loop();
void setup() {
pinMode(LED1, OUTPUT); // Il pin e' definito come output
pinMode(LED2, OUTPUT); // Il pin e' definito come output
}
void loop() {
digitalWrite(LED1, HIGH); // Mette il pin LED1 a 5 V
digitalWrite(LED2, LOW); //  Mette il pin LED2 a 0 V
delay(500);   // Aspetta 500 ms
digitalWrite(LED1, LOW); // Mette il pin LED1 a 0 V
digitalWrite(LED2, HIGH); // Mette il pin LED2 a 5 V
delay(500); // Aspetta 500 ms
}
int main() {
setup();
for(;;) { loop();}
return 0;
}
```

La function setup() viene eseguita solo una volta all'inizio mentre la function loop() viene ripetuta all'infinito

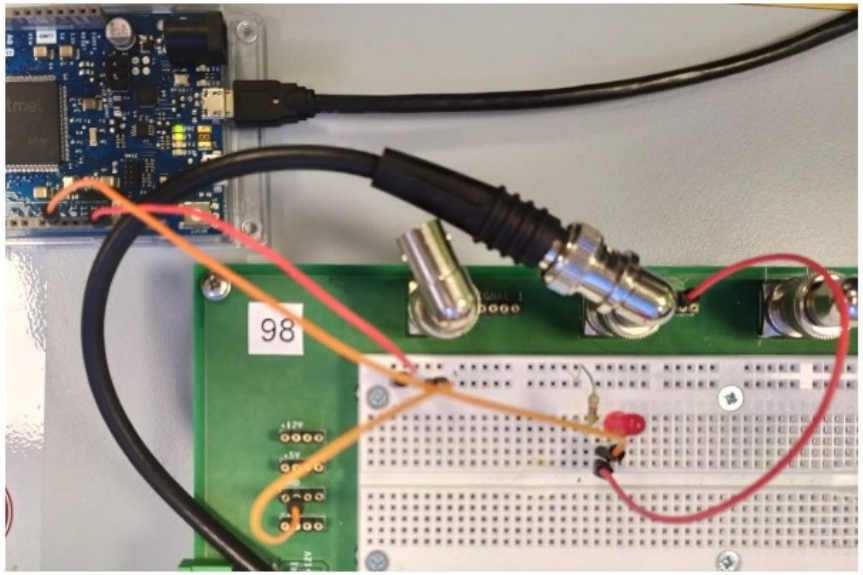
Procedura completa

- La procedura completa quindi è la seguente:
  1. Collegare la scheda Arduino al PC tramite la porta USB;
  2. Avviare l'applicazione Arduino sul PC;
  3. Scrivere il programma;
  4. Salvarlo in una cartella nel proprio spazio disco (Menu: File → Salva con nome …);
  5. Selezionare la porta USB a cui è collegata la scheda (Menu: Strumenti → Porta);
  6. Compilare ed eseguire il programma
- Dal mio sito potete scaricare alcuni esempi di sketch, utili come fonte di ispirazione, che potete copiare nella vostra cartella:
  a. *Led_Rampa.ino*
  b. *Analog_input.ino*
  c. *Analog_write.ino*
  d. *DAC_noise.ino*
  e. *Esempio_Waweform.ino*
  f. *adc_read_2019.ino* (zippato)
  g. *adc_arduino_5.pde* (zippato) Questo serve per Processing
  h. Libreria *controlP5* necessaria per Processing (zippato)
- Notare che ogni sketch è contenuto in una cartella con lo stesso nome: questa struttura deve essere necessariamente mantenuta.
- Nell'ultima esercitazione dovrete utilizzare lo sketch già preparato *adc_read_5_2019.ino* da utilizzare in combinazione con il software Processing.

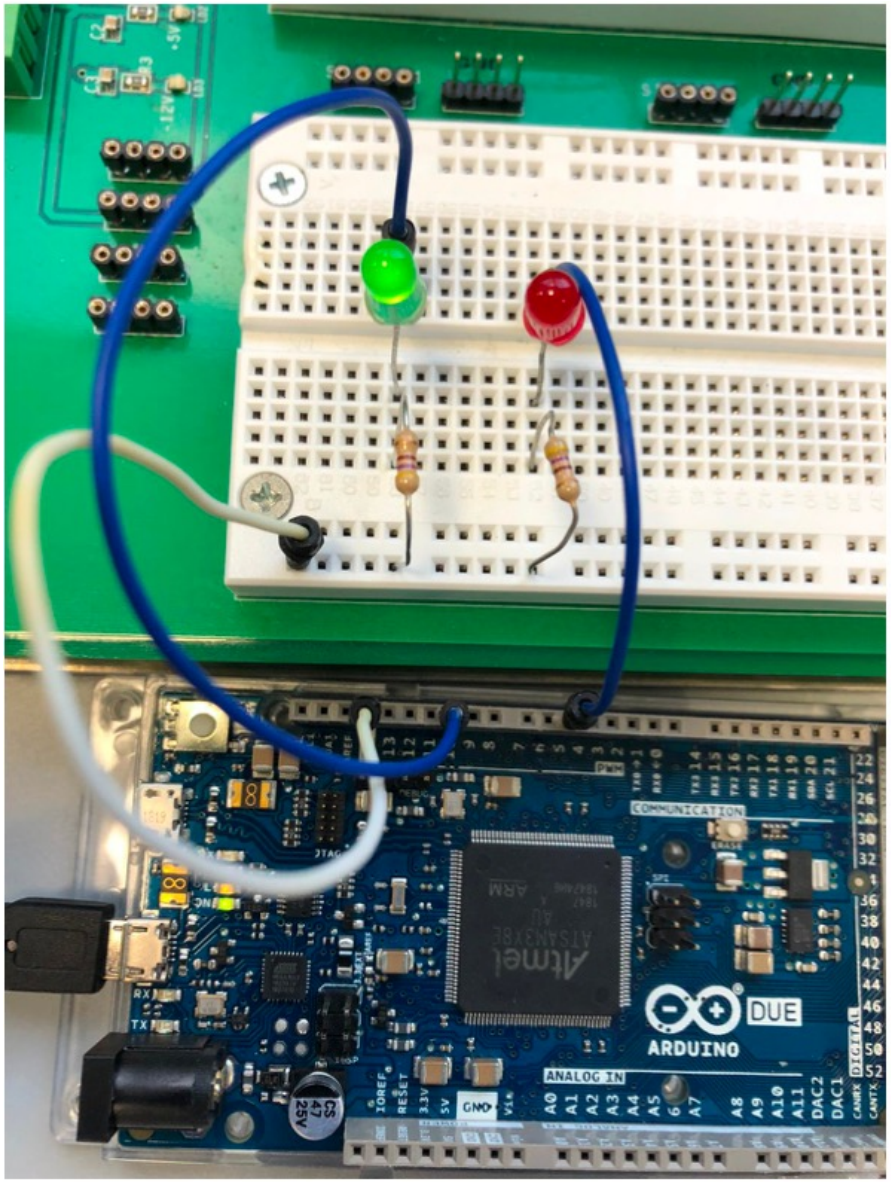
Esempio di montaggio

Notare bene:

Quando si usano degli strumenti (generatore di segnali e/o oscilloscopio), la massa di Arduino Due va collegata alla massa della basetta.



LED rampa



Blinking LED

Esempio: led rampa

Esempio di *sketch* che accende gradualmente il LED e poi lo spegne.

Questo è solo un altro esempio dell'uso della PWM, non dovete implementarlo in laboratorio

Led_Rampa

```
int ledPin11 = 11;
int ledPin10 = 10;
int ledPin9 = 9;
int ledPin8 = 8;
int val1 = 0;
int val2 = 0;
int i = 0;
int millisecondi = 4;
void setup() {
  // put your setup code here, to run once:
  pinMode(ledPin11, OUTPUT);
   pinMode(ledPin10, OUTPUT);
  pinMode(ledPin9, OUTPUT);
  pinMode(ledPin8, OUTPUT);
  Serial.begin(9600);
  Serial.println("This is an Arduino-based dimmer...");
  Serial.println(" ");
}
```

```
void loop() {
  // put your main code here, to run repeatedly:
  for (i=0;i<255;i++) {
    val1=i;
    val2=255-i;
    analogWrite(ledPin11, val1);
    analogWrite(ledPin10, val2);
    analogWrite(ledPin9, val1);
    analogWrite(ledPin8, val2);
    delay(millisecondi);
  }
   for (i=0;i<255;i++) {
    val2=i;
    val1=255-i;
    analogWrite(ledPin11, val1);
    analogWrite(ledPin10, val2);
    analogWrite(ledPin9, val1);
    analogWrite(ledPin8, val2);
    delay(millisecondi);
  }
}
```

Comunicazione seriale

- Programmi di I/O verso la finestra di monitor.
- Misurare la velocita' di esecuzione del μC per varie istruzioni: operazioni aritmetiche, funzioni, operazioni di Input/Output.
- Funzioni di timer: `millis()` e `micros()`
- `delay()` in millisecondi

Esempio

serial_io §

```
long int data=0;
long unsigned t0, t1; //time...

void setup() {
  // Open serial communications and wait for port to open:
  Serial.begin(9600);
  Serial.println("Here we go!");
  Serial.println(" ");
}

void loop() { // run over and over
  if (Serial.available() > 0) {
    data = Serial.parseInt();
    t0=micros();
//  | Serial.print("Hai scritto il numero ");
    Serial.println(data);
//    Serial.print("E questo e' moltiplicato per 10: ");
    Serial.println(data*10);
    t1=micros();
    Serial.print("durata: ");
    Serial.print(t0);
    Serial.print(" ");
    Serial.print(t1);
    Serial.print(" ");
    Serial.println(t1-t0);
    delay(100);
  }
}
```

Scrive una linea vuota

Comunicazione seriale: output del programma

COM5 (Arduino/Genuino Uno)

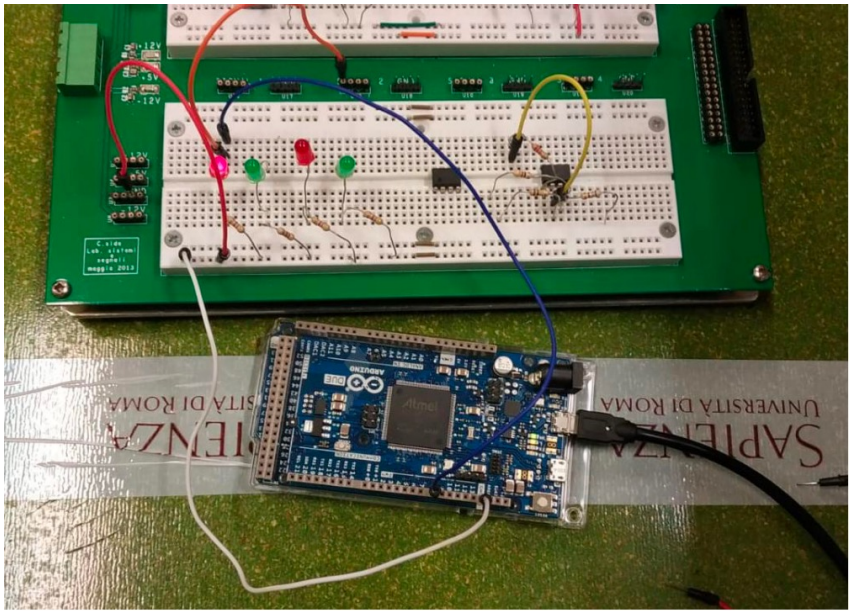
```
tempo esecuzione moltiplicazione

Hai scritto il numero 1
Moltiplicato per 10 = 10
durata: 512 micro S

Hai scritto il numero 10
Moltiplicato per 10 = 100
durata: 596 micro S

Hai scritto il numero 100
Moltiplicato per 10 = 1000
durata: 684 micro S

Hai scritto il numero 1000
Moltiplicato per 10 = 10000
durata: 776 micro S
```

```
Hai scritto il numero 10000
Moltiplicato per 10 = 100000
durata: 864 micro S

Hai scritto il numero 100000
Moltiplicato per 10 = 1000000
durata: 956 micro S

Hai scritto il numero 1000000
Moltiplicato per 10 = 10000000
durata: 1052 micro S

Hai scritto il numero 10000000
Moltiplicato per 10 = 100000000
durata: 1152 micro S
```

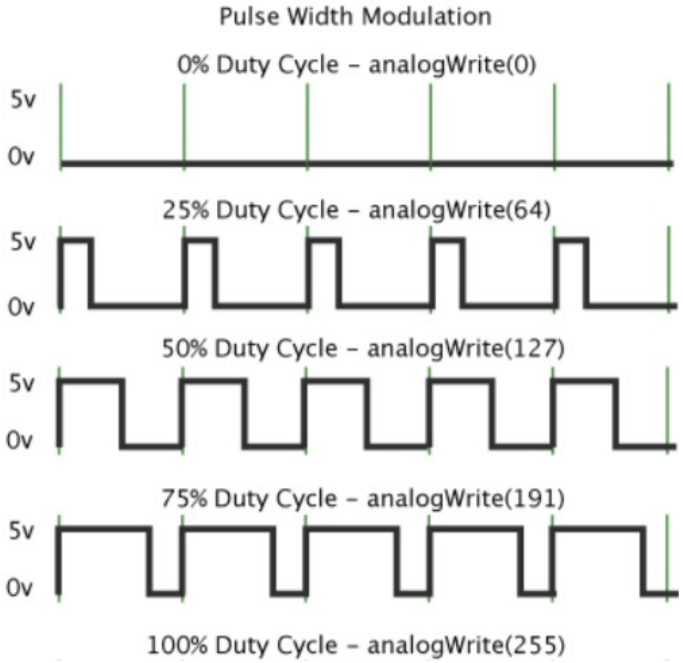
Scorrimento automatico Hai premuto Invia ma non ..

Volendo, se avete tempo e voglia, potete vedere che tipo di correlazione c'è tra il tempo di esecuzione e il numero impostato. Ma non perdete tempo su questa cosa, ci sono tante cose da fare

Operazioni in Input/Output



- Leds per visualizzazione degli outputs (protetti da una resistenza)
- `analogRead(analogPin)`, `analogWrite(pin, value)`
- analogWrite usa modulazione PWM (*Pulse Width Modulation*)



I pin da 2 a 13 possono essere usati in uscita in modo "analogico" (PWM).

0 = 0% ; 255 = 100%

Frequenza 1 kHz

In aggiunta alla PWM sui pin da 2 a 13, Arduino DUE ha anche una vera uscita analogica quando si usa la stessa funzione analogWrite() sui pin DAC0 e DAC1.

FYI: con analogWriteResolution() potete cambiare la risoluzione da 8 bit fino a 12 bit (max 4095)

Esempio: utilizzo di analogWrite()

La luminosita' del Led dipende dal numero scritto sulla riga di input

analog_write

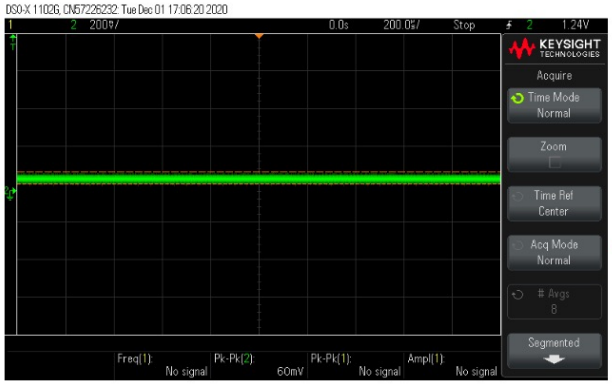
```
int ledPin = 9;
int val = 128;

void setup() {
pinMode(ledPin, OUTPUT);
Serial.begin(9600);
Serial.println("Light your fire...");
Serial.println(" ");
}

void loop() {
  if (Serial.available() > 0) {
    val = Serial.parseInt();
 // check value in
    if (val >= 0 && val <= 255) {
     analogWrite(ledPin, val);
     Serial.print("Scritto valore :");
     Serial.println(val);
     }
    else {
     Serial.println("Valore fuori range (0:255)!!!");
     }
  }
}
```

- Mettete il Led (protetto da una resistenza da un centinaio di Ohm) su uno dei pin da 2 a 13 (evitate i pin 5 e 6); scegliete 4-5 valori del duty cycle (0 → 255)
- Leggete l'output del pin anche con l'oscilloscopio e riportate sulla relazione gli screenshot
- Verificate che la frequenza sia di 1 kHz

Questo e' il valore che avete scritto con la tastiera del computer sulla porta seriale di input di arduino

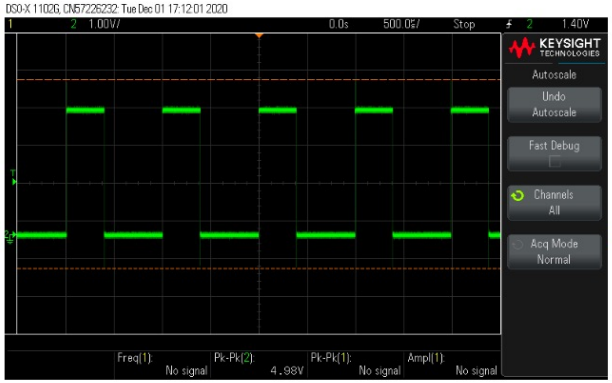
Attenzione: c'è un carattere di controllo a fine linea che viene interpretato come uno zero!

Dovreste ottenere questo

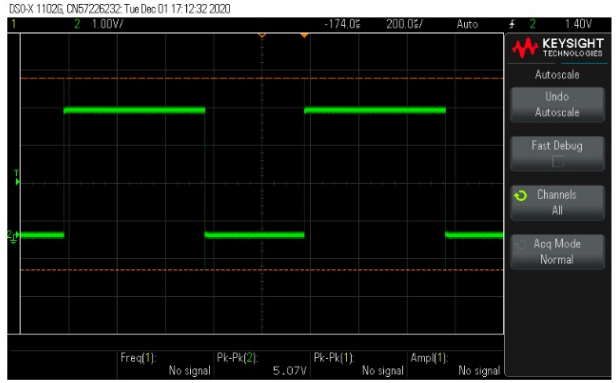


(a) 0

(b) 64

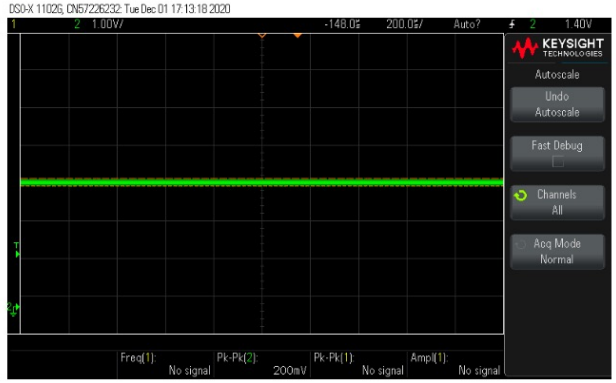


(c) 100



(d)150

200



255

Studio del DAC

Scrivere ed eseguire un programma per effettuare un test di linerità, dinamica di uscita ed offset del DAC di Arduino. Utilizzando la risoluzione a 12 bit configurate diversi valori di tensione (10-20) nell'uscita di DAC di Arduino prescelta. Connettere l'uscita all'oscilloscopio o al multimetro tramite la basetta sperimentale e misurare i valori delle tensioni ottenute costruendo la curva di linearità del DAC in tutto il suo range dinamico. Commentate i risultati ottenuti.

$$V = \frac{V_{max}}{2^n - 1} \sum_{0}^{n-1} 2^i a_i$$

$$V = \frac{V_{REF}}{4095} \cdot N \quad (0 \le N \le 4095)$$

12 bit: N=4095
10 bit: N=1023
8 bit: N=255

V_{REF} è la tensione di riferimento di arduino Due (3.3 V)

Arduino Due does not have an analog output voltage from 0 V to Vref, but from 1/6 to 5/6 of the reference voltage, that is, 0.55 V and 2.75V with Vref = 3.3 V.

The output voltage range of the DAC is only 2.75-0.55 = 2.2 V, with a resolution of 2.2/4095 = 0.5372 mV

$$V = \frac{1}{6} \cdot V_{REF} + \frac{4}{6} \cdot \frac{V_{REF}}{4095} \cdot N \quad (0 \le N \le 4095)$$

Esempio di un programma che usa (anche) il DAC

Esempio_DAC

```
int val = 128;
int analogPin = A3;
int RdVal = 0;
float Vx = 0.0;
float Vref = 3.3;
int val_A = 0;

void setup() {
  // put your setup code here, to run once:
  analogReference(AR_DEFAULT); // Arduino Due 3.3 V
  analogWriteResolution(12);  // set the analog output resolution to 12 bit (4096 levels)
  analogReadResolution(12);   // set the analog input resolution to 12 bit

  Serial.begin(9600);
  Serial.println("Write the number to be converted (< 4096) ");
  Serial.println(" ");
}
```

analogWrite() di arduino Uno ha una risoluzione di 8 bit, che è anche il default di arduino Due, ma può essere portata a 12 bit con analogWriteResolution(bit)

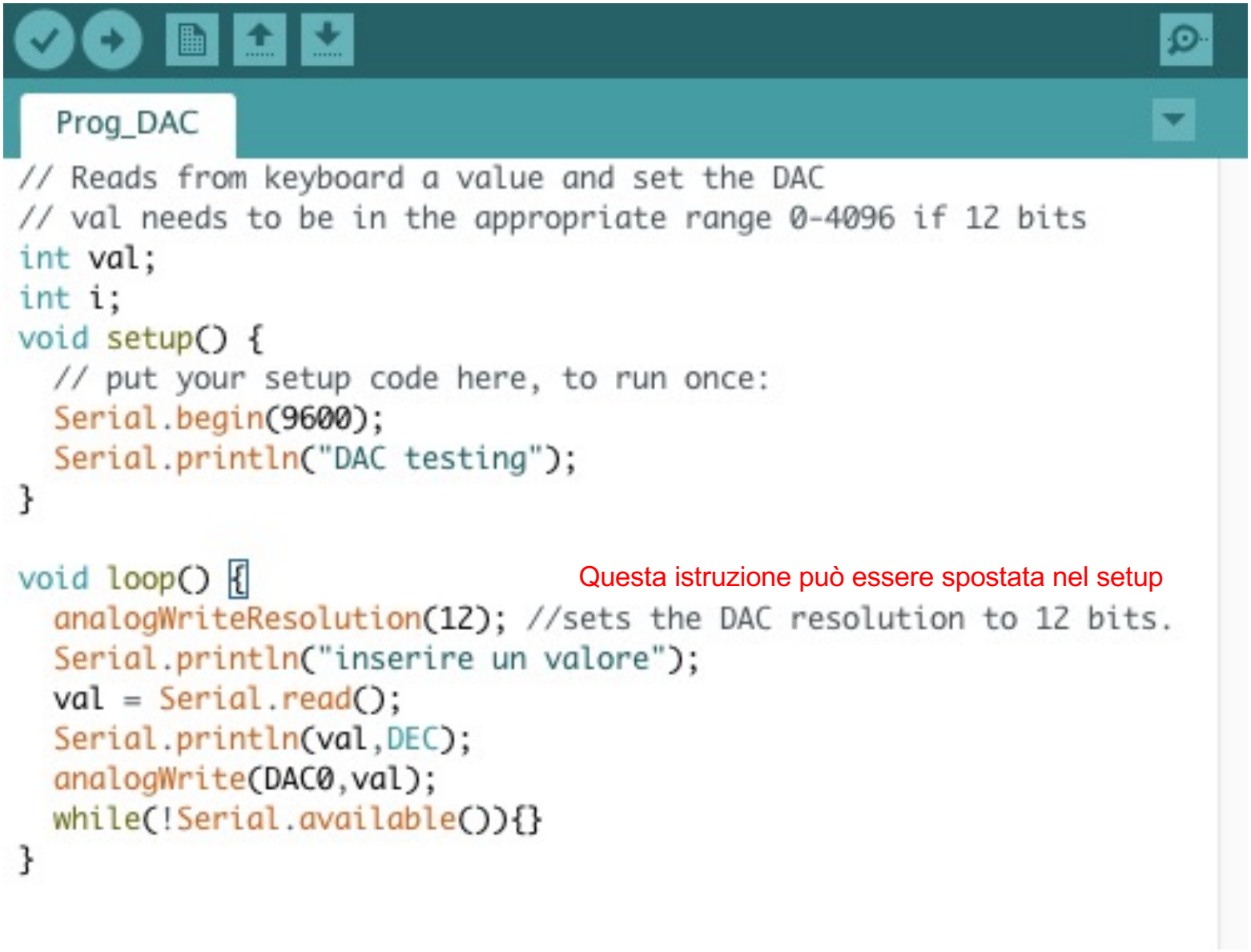
L'ADC di Arduino Uno è a 10 bit che è anche il default di Arduino Due. Può essere portato a 12 bit con analogReadResolution(12)

```
void loop() {
  // put your main code here, to run repeatedly:
 if (Serial.available() > 0) {
    val=Serial.parseInt();
  // check value in
     if (val>=0 && val <=4095) {
      analogWrite(DAC1, val);  // write the digital value on DAC1
      Serial.print("Scritto valore :");
      Serial.println(val);
// read from the analog pin the voltage written by the DAC
      delay(200);
        RdVal = analogRead(analogPin);
        Vx = Vref*(float(RdVal)/float(4095));
        val_A = (Vx-Vref/float(6))*1.5*float(4095)/Vref;
        delay(200);
        Serial.print(" Numero letto sul pin analogico: ");
        Serial.print(RdVal);
        Serial.print(" Tensione letta: ");
        Serial.println(Vx);
        Serial.print(" N ricostruito: ");
        Serial.println(val_A);
     }
     else {
      Serial.println("Valore fuori range (0:4095)!!!");
     }
  }

}
```

Questo è un programma che scrive sul DAC1, che poi viene collegato sul pin analogico A3, dal quale viene riletto il valore di tensione convertito in un numero dall'ADC, il quale deve essere riconvertito nel numero di partenza per fare il confronto.
(c'è sempre una differenza di 10-14 unita).

Voi dovrete leggere l'uscita di DAC1 con il multimetro e/o l'oscilloscopio

Programma basico di uso del DAC

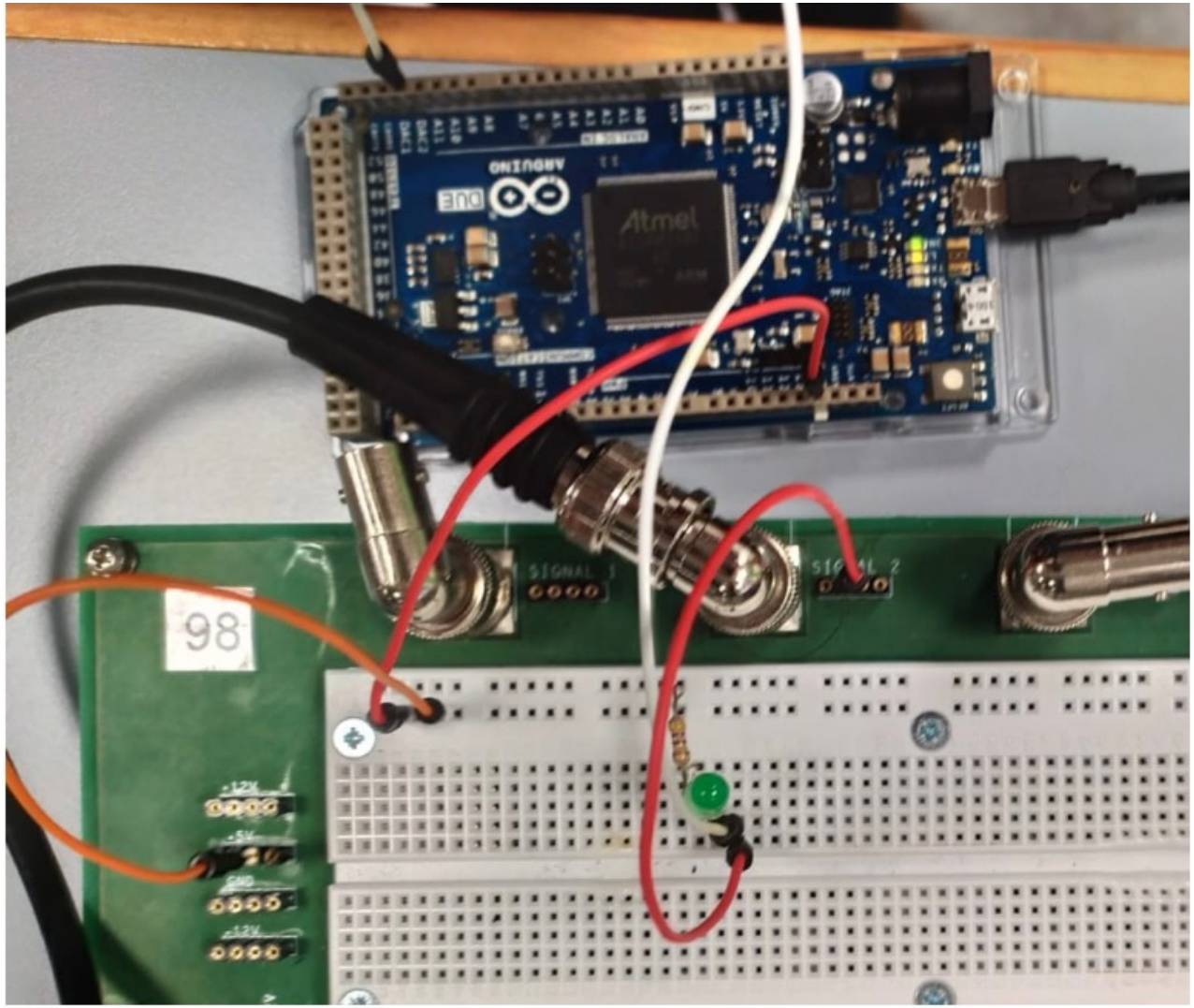


Prog_DAC

```
// Reads from keyboard a value and set the DAC
// val needs to be in the appropriate range 0-4096 if 12 bits
int val;
int i;
void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  Serial.println("DAC testing");
}

void loop() {
  analogWriteResolution(12); //sets the DAC resolution to 12 bits.
  Serial.println("inserire un valore");
  val = Serial.read();
  Serial.println(val,DEC);
  analogWrite(DAC0,val);
  while(!Serial.available()){}
}
```

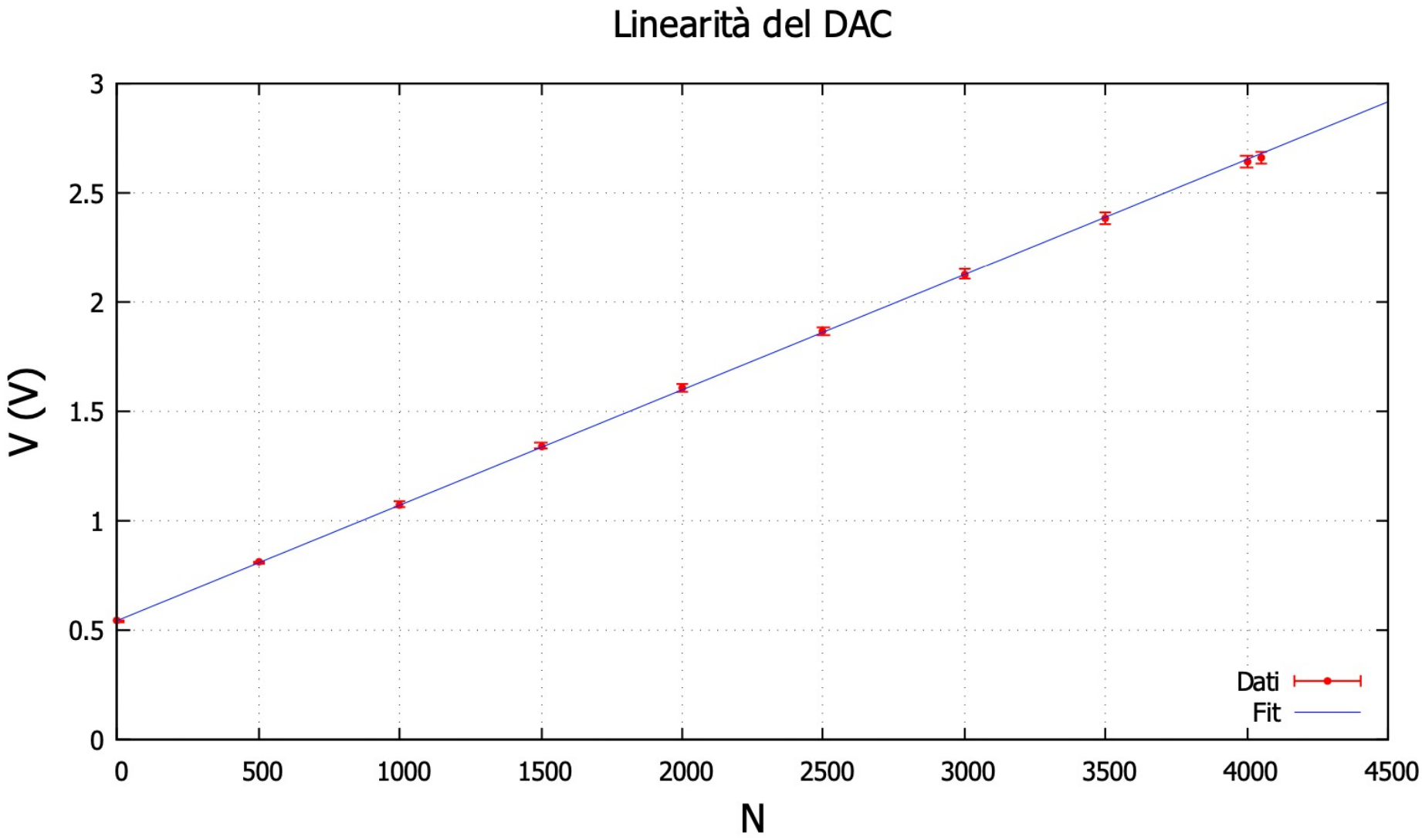
Questa istruzione può essere spostata nel setup

Esempio di montaggio del DAC



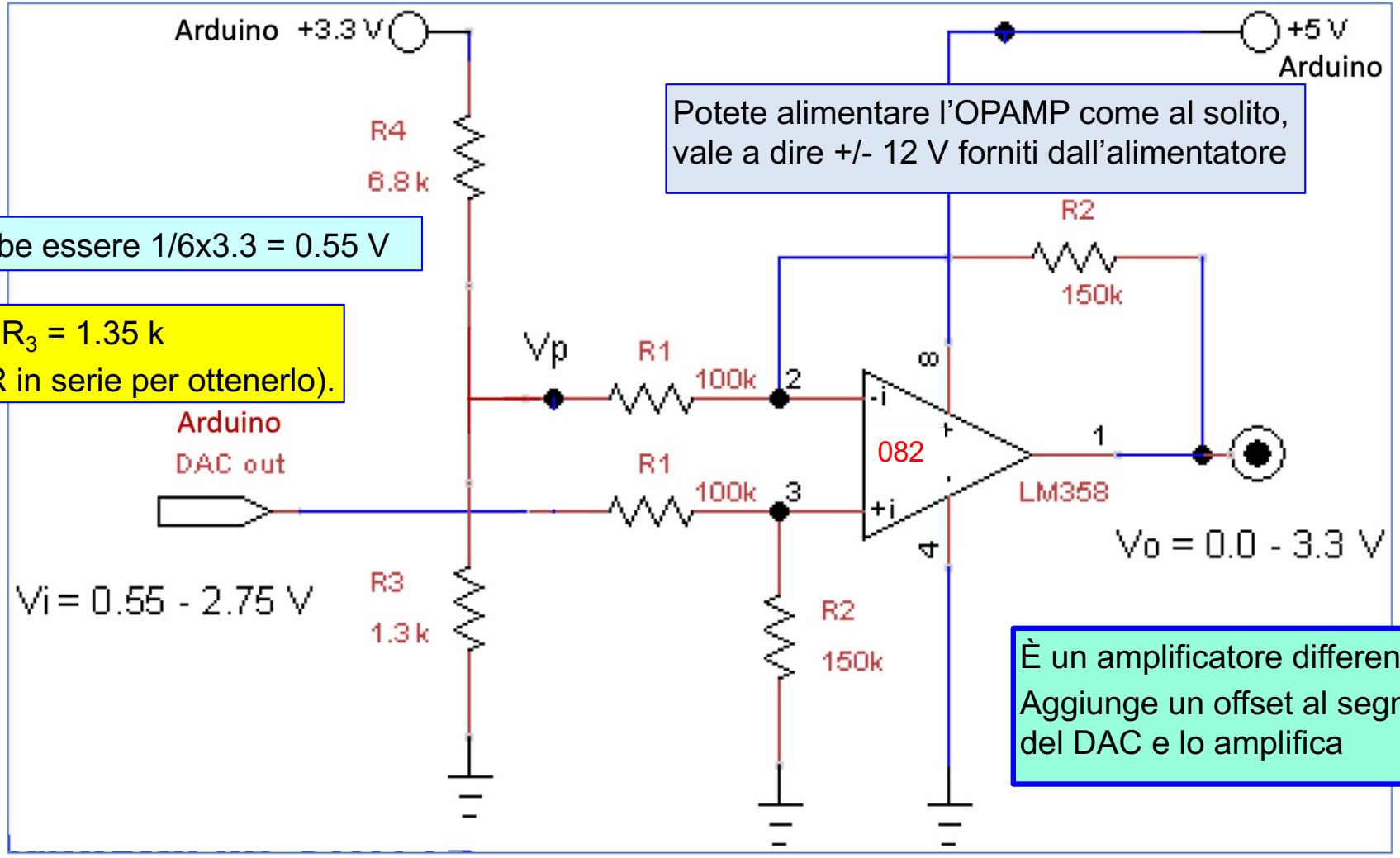
Verificate se il segnale di uscita del DAC cambia se collegate LED e resistenza oppure no.
Nel caso in cui dovesse cambiare fate le misure a “vuoto”

Grafico di linearità del DAC



Qui potete anche fare il test di linearità che tanto amate!

Adattore del livello di uscita del DAC



Potete alimentare l'OPAMP come al solito, vale a dire +/- 12 V forniti dall'alimentatore

Vp dovrebbe essere 1/6x3.3 = 0.55 V

R_3 = 1.35 k
(usate R in serie per ottenerlo).

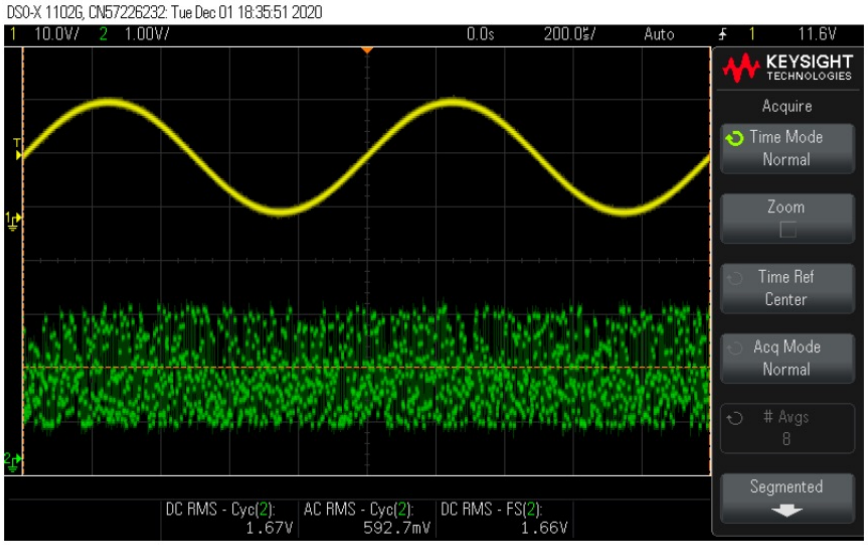
È un amplificatore differenziale.
Aggiunge un offset al segnale del DAC e lo amplifica

Rifate il grafico di linearità che avete fatto nel punto precedente

Facoltativo: generatore di rumore

Scrivere ed eseguire un programma che utilizzi uno dei pin di DAC di Arduino per generare un rumore bianco di ampiezza ~50 mV RMS. Tale risultato può essere ottenuto generando valori di tensione random e programmando con essi la porta di DAC. Collegare l'uscita DAC prescelta all'oscilloscopio e produrre uno screenshot che mostri il risultato ottenuto.

```
long randNumber;
int i;

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  randomSeed(analogRead(0));
}  Mettete la risoluzione di analogWrite() a 12 bit

void loop() {
  // put your main code here, to run repeatedly:
  randNumber = random(0,200);

  analogWrite(DAC0,randNumber);
  i++;
  if(i<15) Serial.println(randNumber);
}
```



Con la risoluzione a 12 bit e numeri random tra 0 e 200, la tensione di uscita del DAC varia tra 3.3/6 =0.55 V e $0.55 + \frac{4}{6} \times 3.3 \times \frac{200}{4095}$=0.55+0.107 V

Input analogico: calibrazione ADC

- Scrivete ed eseguite un programma per effettuare la calibrazione dell'ADC del microcontrollore, scegliendo uno degli ingressi analogici.
- Utilizzate come Analog Reference il default (3.3 V).
- Come tensione da convertire utilizzate l'uscita dell'adattore del DAC che avete appena costruito
- Dovete scrivere uno *sketch* che riceva in input alla porta seriale il numero da dare al DAC e scriva sulla porta seriale il numero risultante dalla conversione dell'ADC (RdVal)
- Costruire il grafico della calibrazione dell'ADC misurando dieci-venti punti a diverse tensioni (la tensione in ingresso all'ADC va misurata con il multimetro).

ATTENZIONE: gli input analogici di Arduino sono molto delicati, non bisogna mai violare queste regole:

1. Mai connettere la tensione agli ADC quando Arduino è spento
2. Mai superare la tensione di 3.3 V in input sui pin ADC di Arduino

Memento: l'ADC di Arduino non legge valori negativi di tensione. Aggiungete un offset ai segnali del generatore per avere sempre una tensione nel range 0 - 3.3 V

Esempio di analogRead()

analog_read

```
int analogPin = A3; // analog pin A3
int RdVal = 0;
float Vx = 0.0;

void setup() {
  analogReference(AR_DEFAULT); // Arduino Due 3.3 V
//  analogReadResolution(12);  // set the ADC to 12 bit
  Serial.begin(9600);
  Serial.println(" Arduino Voltage meter ");
  Serial.println(" ");
}
void loop() {
  RdVal = analogRead(analogPin); // ADC value
  Vx = float(3.3)*(float(RdVal)/float(1023)); // 10 bit ADC
// Vx = float(3.3)*(float(RdVal)/float(4095)); // 12 bit ADC
   delay(1000);
   Serial.print(" Numero letto sul pin analogico: ");
   Serial.print(RdVal);
   Serial.print(" Tensione letta: ");
   Serial.println(Vx);
}
```

- Mandate sul pin 3 di ingresso analogico una tensione continua compresa tra 0 e 3.3 V costruita con il DAC e l'adattatore;
- Misurate la tensione d'ingresso con il multimetro
- Confrontate il valore misurato con quello scritto da Arduino sulla porta seriale.

$$\text{Output} = \frac{2^{10}-1}{V_{ref}} V_{in}$$ (Oppure 2^{12})

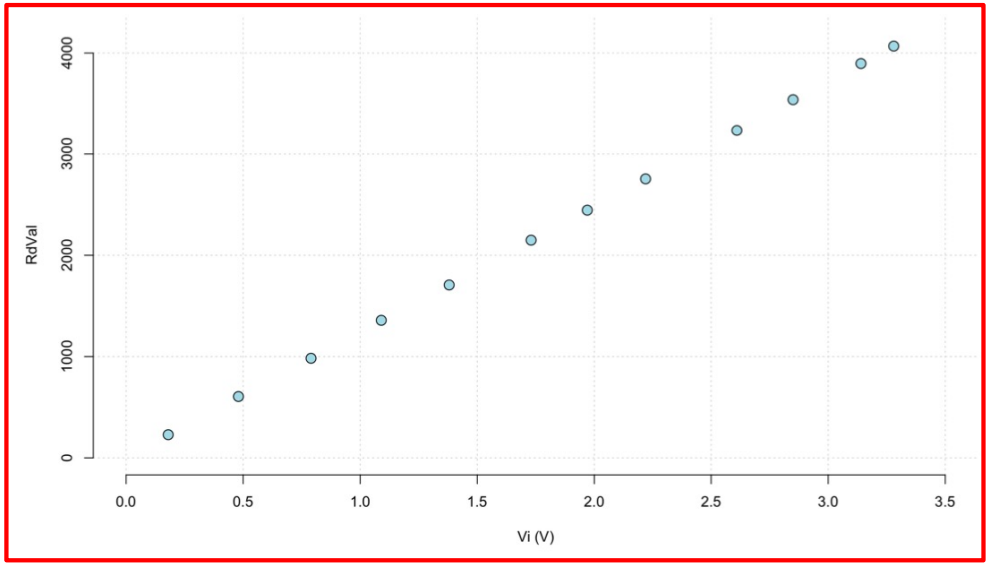
Calibrazione dell'ADC di Arduino

- Modificate il programma in modo da scrivere il valore di RdVal (e non Vx)
- Mandate in input una decina di valori di tensione continua compresi tra 0 e 3.3 V
- Costruite il grafico tra il valore dell'ADC e la tensione inviata
- Ricavate il valore della calibrazione

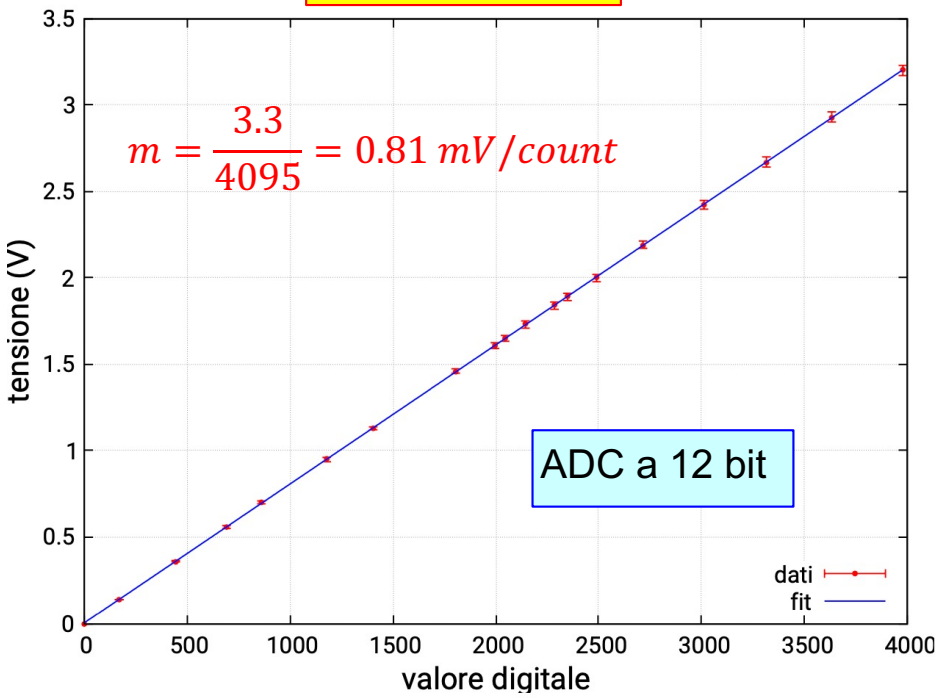
Potete lasciare il valore di default di 10 bit oppure usare il range completo di 12 bit

Dovreste ottenere questo

ADC versus V_{input}



V_{input} versus ADC



- Voi avete come variabile indipendente la tensione di input e come variabile dipendente il valore dell'ADC, quindi logicamente dovreste costruire il primo grafico
- Però voi avete bisogno di una costante di calibrazione che converta il numero letto dall'ADC in un valore di tensione, quindi dovete costruire il secondo grafico; la costante di calibrazione che cercate è il coefficiente angolare di questa retta.
- La costante di calibrazione vi servirà per la misura successiva
- Verificate anche che l'*offset* sia effettivamente zero.

Input analogico

Conversione Analogico – Digitale di una forma d'onda

Ovvero campionamento digitale di una forma d'onda

- Fate ora la conversione di un segnale sinusoidale e di un'onda triangolare forniti dal generatore di segnali.
- Scegliete un valore picco-picco di 1.0 – 1.5 V e aggiungete un offset in modo che il segnale sia sempre compreso tra 0 e 3.3 V (Ovviamente sarebbe meglio non "toccare" questi due estremi).
- Scrivete uno *sketch* che campioni diversi punti della forma d'onda e scriva in una tabella il valore della tensione e il corrispondente valore dell'ADC.
- Ad ogni campione corrisponde un indice progressivo e, se siete capaci, anche un tempo.
- Scrivete tutti i punti sulla porta seriale e poi trasferiteli su un file sul PC (e/o sul vostro laptop) facendo *cut and paste* (o usando qualche altro sistema più furbo)
- Usando open office (o qualunque altro programma grafico) ricostruite la forma d'onda di partenza
- Se avete usato le costanti di calibrazione appropriate, dovreste essere in grado di ricavare la tensione picco-picco e il periodo del segnale
- Confrontate questi valori con l'ampiezza picco-picco e il periodo misurati con l'oscilloscopio

Avete costruito il vostro oscilloscopio personale utilizzando Arduino

N.B. Nella prossima esercitazione utilizzeremo il software processing per acquisire la forma d'onda e costruire i grafici

Programma lettura ADC (waweform)

```
// The program samples a waweform and send it to the serial port.
// You can set the sampling frequency by introducing a delay T
// in between each sample (take into account that analogRead()
// function takes about 5 microsecond.
// You can define the integration window by setting NSample

int analogPin = A3; // tensione da misurare
float FullScale = 3.3; // default reference value for arduino Due
int data[1000];  // array to store the sampled counting

int NSamples = 1000; // non superate le dimensioni del vettore data
int T = 0; // in microsecondi; puo' essere usata per cambiare
           // la frequenza di acquisizione.
           // AnalogRead() impiega circa 5 microsecondi

int i = 0;
long unsigned t0, t1; // time
float deltaT; // intervallo di tempo tra una lettura e l'altra (us)


void setup() {
  // put your setup code here, to run once:
  analogReference(AR_DEFAULT); // Arduino Due 3.3 V
// analogReadResolution(12);   // set the analog input to 12 bit

  Serial.begin(9600);
  Serial.println(" Waveform Sampling ");
  Serial.println(" ");

}
```

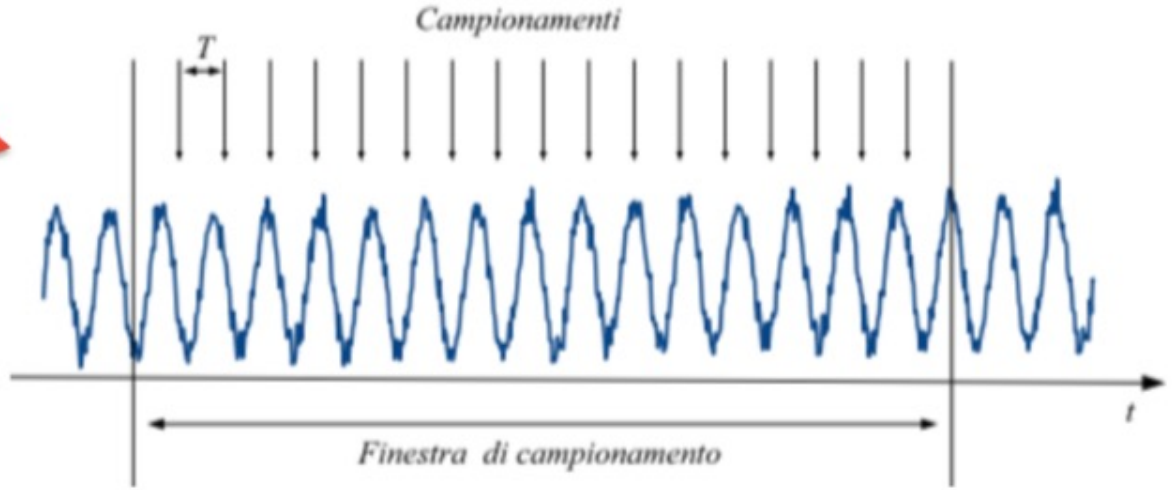
segnale sul pin analogico 3

Fate attenzione al numero di bit che scegliete per l'ADC.
Se usate 12 bit allora dovete dividere per 4095
(oppure usate la costante di calibrazione che avete ricavato nel punto precedente)

```
void loop() {
// start the loop to sample the waweform

  t0=micros();
  for (i=0 ; i<NSamples; i++) {  // data acquisition
    data[i] = analogRead(analogPin); // this operation
                                     // takes abpit 5 us
    delayMicroseconds(T);
  }

  t1=micros();
  Serial.print(" Durata: ");
  Serial.println(t1-t0);
  Serial.print(" Numero di punti ");
  Serial.println(NSamples);
  Serial.print(" Tempo di campionamento (in microsecondi ");
  deltaT = float(t1-t0)/NSamples;
  Serial.println(deltaT);
  Serial.println(" ");

// We start to send data
  for (i=0 ; i<NSamples; i++) {  // send the buffer to the PC
    Serial.print(i);
    Serial.print("  conteggi: ");
    Serial.print(data[i]);
    Serial.print("  V = ");
    float temp=float(data[i])/1023*FullScale; // 10 bit ADC
//    float temp=float(data[i])/4095*FullScale; // 12 bit ADC
    Serial.print(temp);
    Serial.print("  tempo = ");
    Serial.println(i*deltaT);
  }
//
  while(1){} // per fermare il programma
// per effettuare una nuova acquisizione premere
// il pulsante reset su arduino
}
```

Frequenza di campionamento, etc…

$f_c = \frac{1}{T}$



Frequenza di Nyquist

$f_N = \frac{f_c}{2}$

T = 5 µs → f_c = 200 kHz	→ f_N = 100 kHz	Finestra = NSamples x T = 1000 x 5 µs = 5 ms

Massima frequenza del segnale da visualizzare = f_N

Tuttavia, per avere una buona visualizzazione, dovremmo avere all'interno della finestra di campionamento minimo un periodo e massimo due o tre periodi.

f_{min} = 1/finestra = 1/ 5 ms = 200 Hz

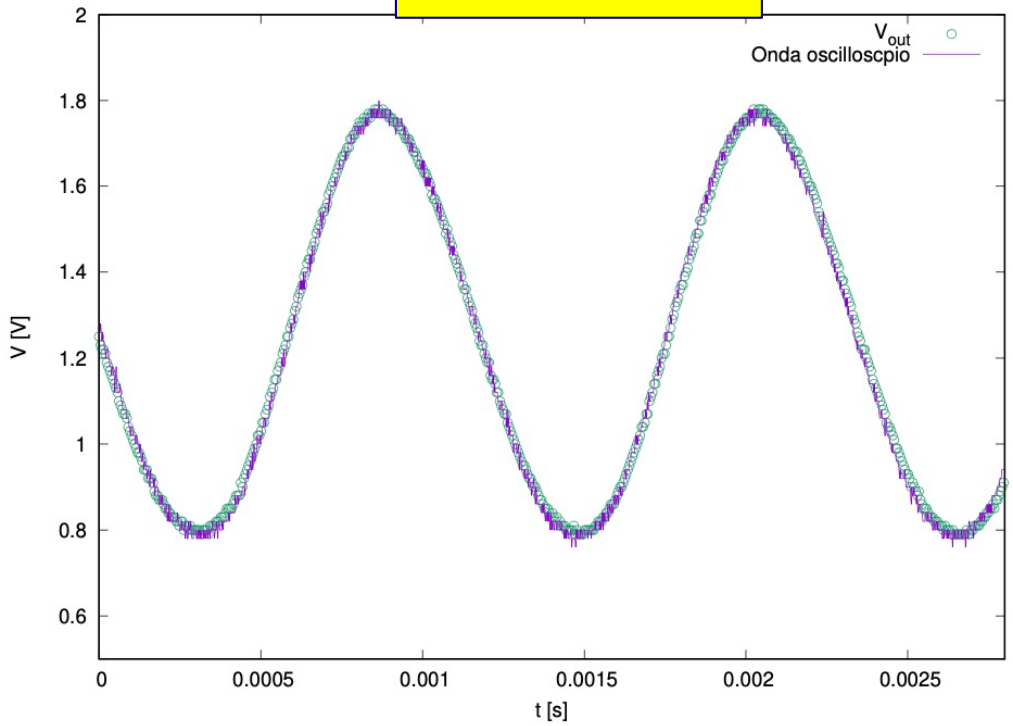
Cambiando T oppure NSample potete variare le varie frequenze. Fate alcune prove.

Provate anche con un'onda triangolare.

Importante: aggiungete un offset in modo che la tensione sia sempre positiva.
Il suo valore massimo deve essere minore di 3.3 V, altrimenti verra' tagliato.

Dovreste ottenere questa cosa

Onda sinusoidale



Cosa mettiamo sull'asse dei tempi?

Forse ms sarebbe stato meglio

Figura 8: Tensione letta dall'ADC di Arduino, in ingresso un segnale sinusoidale di ampiezza picco-picco $V = 1.04\,V$, frequenza $\nu = 860\,Hz$ e offset di $1.30\,V$.

- Fate un confronto con uno screenshot dello stesso segnale mandato sull'oscilloscopio. In fondo avete anche voi realizzato un'oscilloscopio digitale (di bassa qualità)
- Se riuscite a fare questi plot in tempo reale durante l'esercitazione e non a casa, potreste avere il tempo di fare delle modifiche (numeri di punti, frequenza, valor medio, etc…)
- Fuori programma: provate a scrivere un programma che faccia la DFT per vedere se funziona.

Dovreste ottenere questa cosa

Onda triangolare

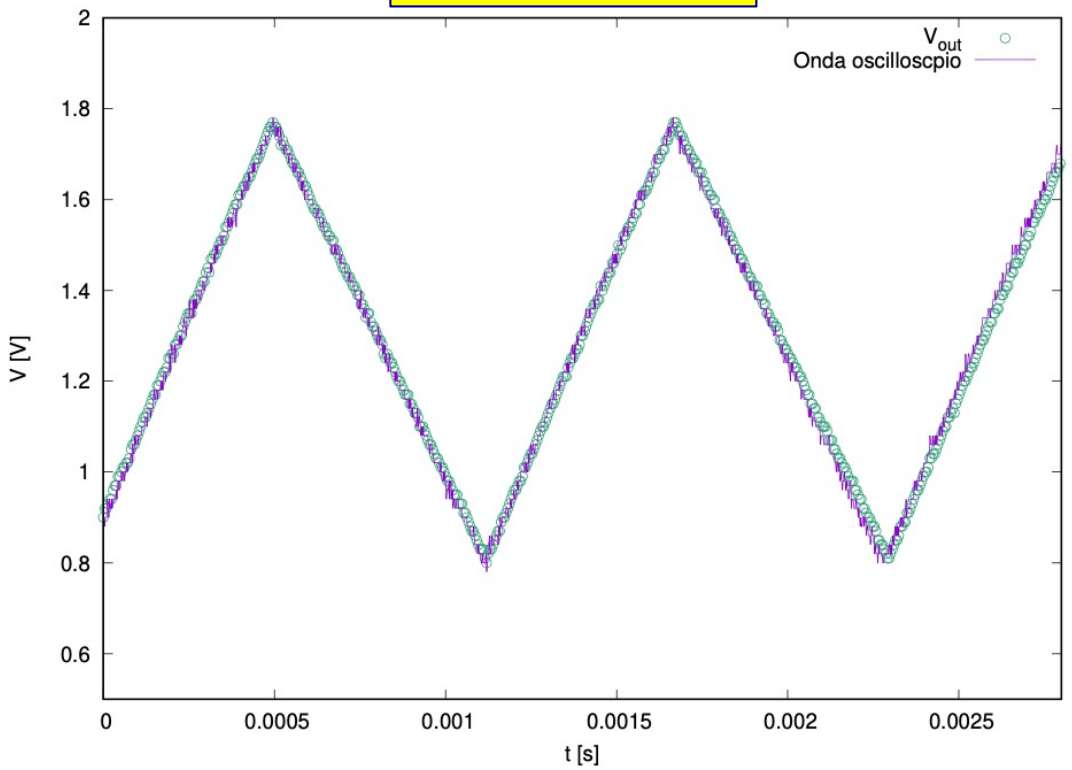


Figura 9: Tensione letta dall'ADC di Arduino, in ingresso un segnale triangolare di ampiezza picco-picco $V = 1.04\,V$, frequenza $\nu = 860\,Hz$ e offset di $1.30\,V$.

- Fate un confronto con uno screenshot dello stesso segnale mandato sull'oscilloscopio. In fondo avete anche voi realizzato un'oscilloscopio digitale (di bassa qualità)
- Se riuscite a fare questi plot in tempo reale durante l'esercitazione e non a casa, potreste avere il tempo di fare delle modifiche (numeri di punti, frequenza, valor medio, etc…)

SAPIENZA
UNIVERSITÀ DI ROMA

Fine esercitazione 8