

Laboratorio di Segnali e Sistemi

- Esercitazione -8 -

Familiarizzazione con arduino

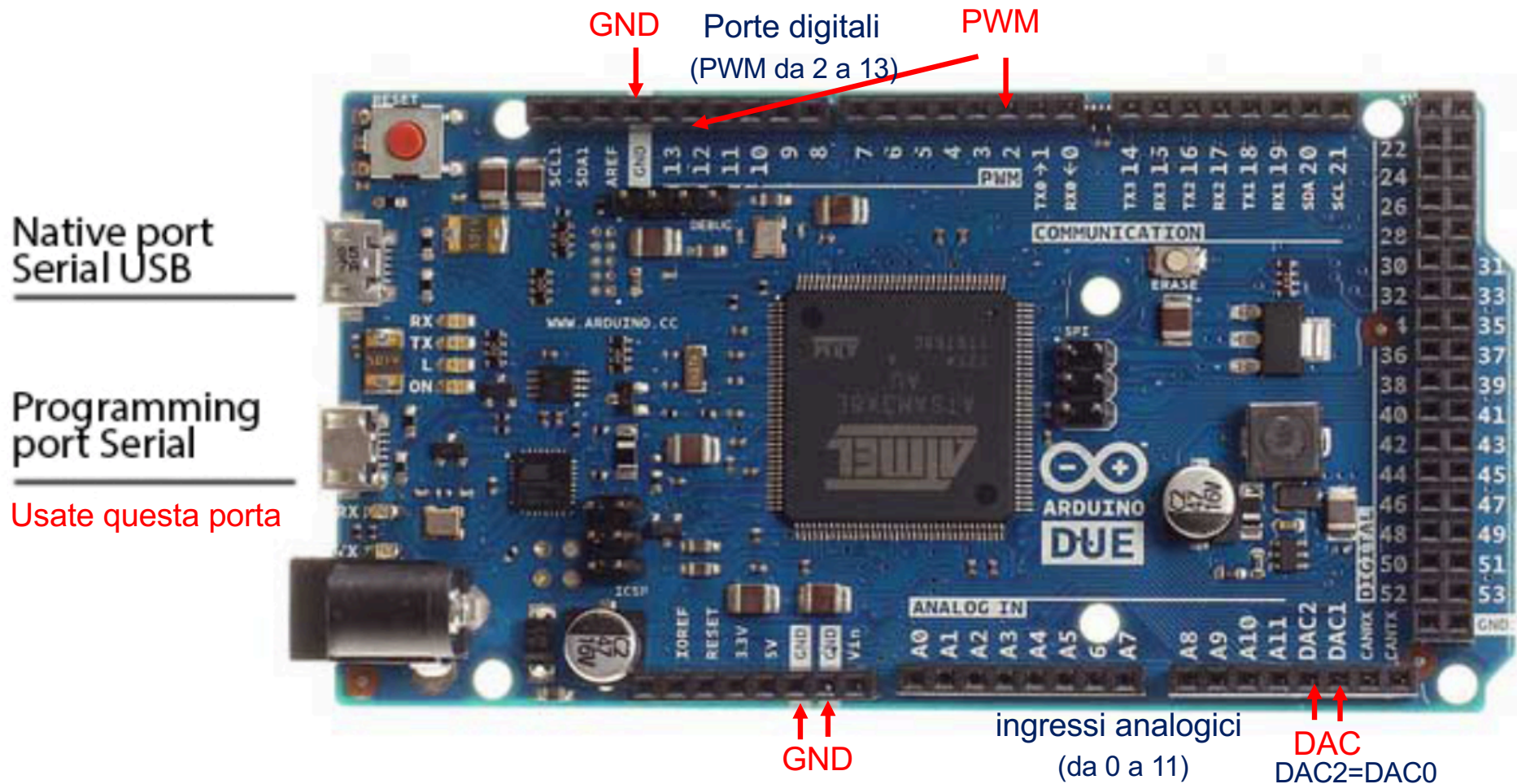


Claudio Luci
SAPIENZA
UNIVERSITÀ DI ROMA

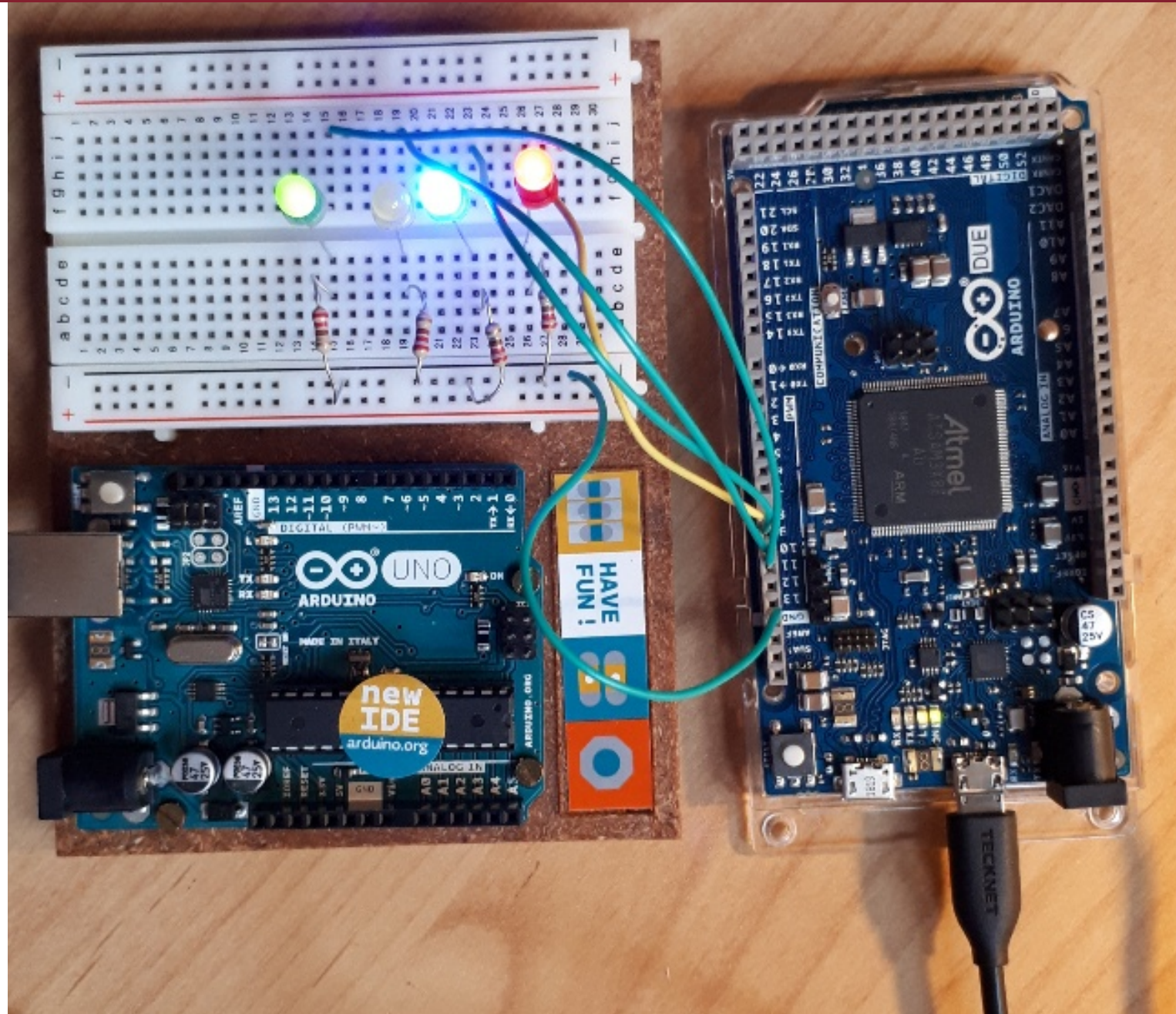
last update : 070117

Arduino

L'obiettivo di questa esperienza e' di familiarizzarsi con la scheda Arduino Due, con il microcontrollore ATMEL SAM3X8E e con il relativo software (vedi Appendice per maggiori dettagli). E' bene avere sempre disponibile per consultazione il sito internet di Arduino (<http://www.arduino.cc>). <https://www.arduino.cc/reference/en/> (qui trovate le varie funzioni)
I circuiti da utilizzare possono essere montati sulla consueta scheda sperimentale, connettendola ad Arduino Due mediante opportuni ponticelli.



arduino uno vs arduino due

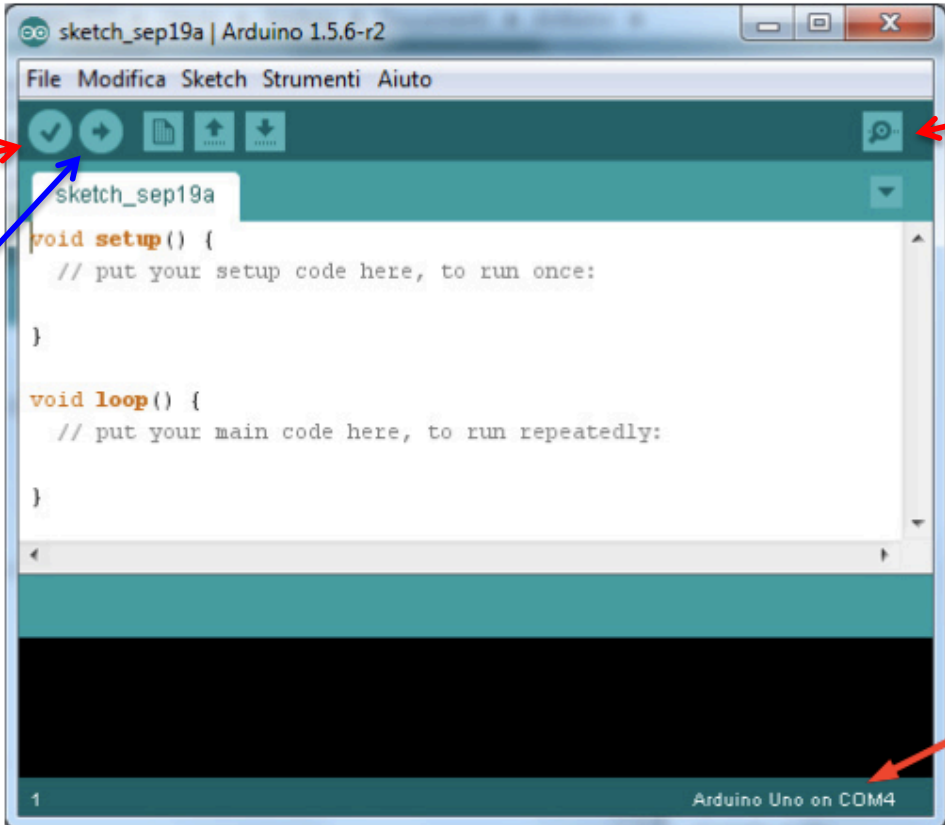


Come iniziare

La scheda Arduino **DUE** viene gestita dall'applicazione Arduino, già installata sul PC. Questa applicazione consente di scrivere un programma in linguaggio C, compilarlo, scaricarlo sul microcontrollore e mandarlo in esecuzione.

All'avvio dell'applicazione² (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), vedi Fig 9.4.

²Se non avete il collegamento sul desktop potete avviare da C:/Programmi/Arduino/arduino.



The screenshot shows the Arduino IDE window titled "sketch_sep19a | Arduino 1.5.6-r2". The menu bar includes "File", "Modifica", "Sketch", "Strumenti", and "Aiuto". The toolbar contains icons for "Verify" (a checkmark), "Upload" (a right-pointing arrow), "New" (a document icon), "Open" (a folder icon), "Save" (a floppy disk icon), and "Serial Monitor" (a monitor icon). The main text area contains the following code:

```
sketch_sep19a
void setup() {
  // put your setup code here, to run once:
}

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

At the bottom of the window, a status bar indicates "1" and "Arduino Uno on COM4".

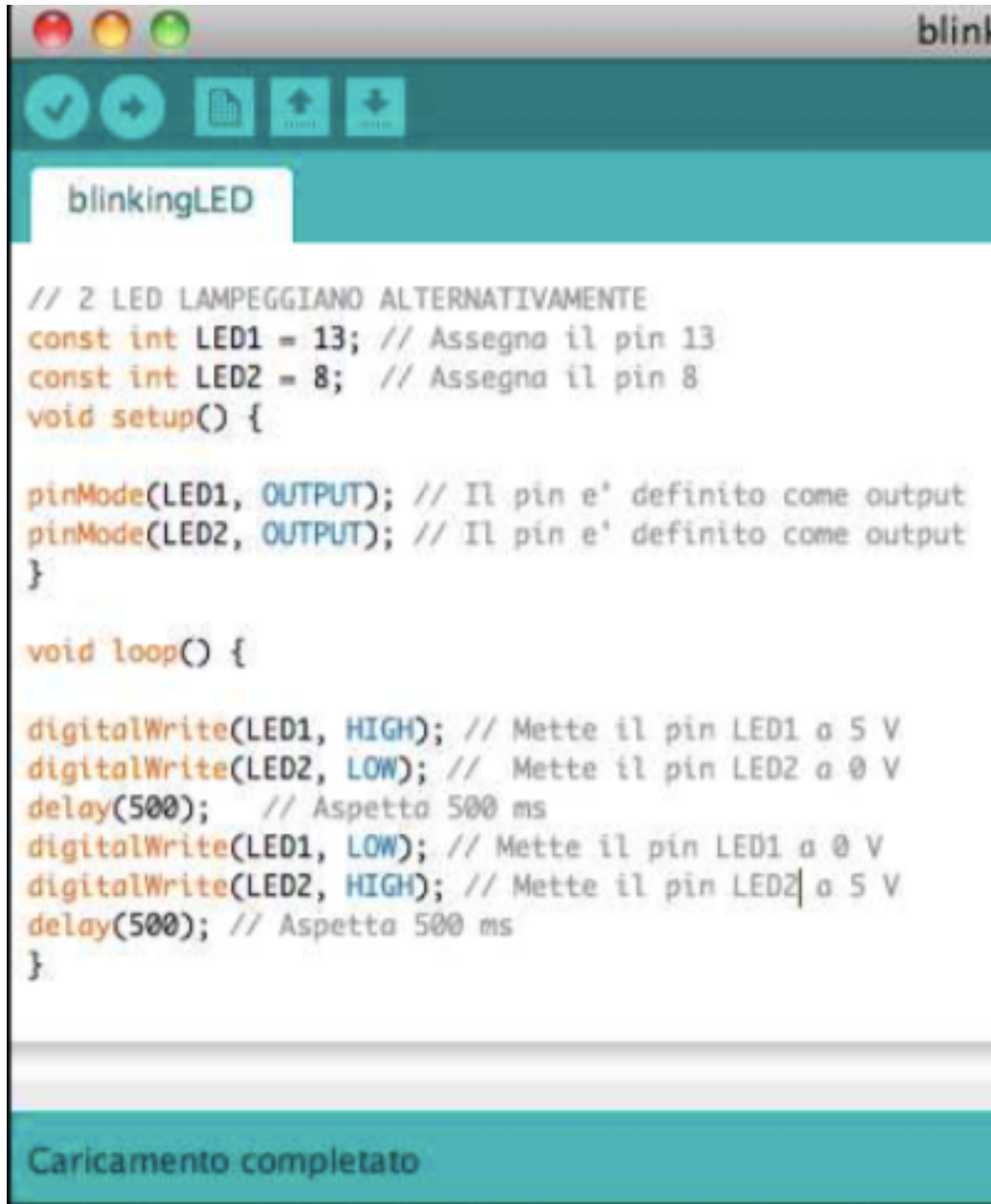
Annotations:

- A red arrow points to the "Verify" icon with the text "compila".
- A blue box with the text "Compila, carica e run" has a blue arrow pointing to the "Verify" icon.
- A red arrow points to the "Serial Monitor" icon with the text "Apri il serial monitor per eseguire operazioni di input/output".
- A red arrow points to the "COM4" text in the status bar with the text "Porta USB utilizzata".

A text box on the right contains the following text:

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.

Esempio di programma: blinking led



```
// 2 LED LAMPEGGIANO ALTERNATIVAMENTE
const int LED1 = 13; // Assegna il pin 13
const int LED2 = 8; // Assegna il pin 8
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
}
```

Caricamento completato

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:

- Collegare la scheda Arduino tramite porta USB;
- Avviare l'applicazione Arduino;
- Scrivere il programma;
- Salvarlo in una cartella nel proprio spazio disco (Menu: File -> Salva con nome....);
- Selezionare la porta USB a cui è collegata la scheda (Menu: Strumenti -> Porta);
- Compilare ed eseguire il programma.

Nigro (Non piu', trovate degli esempi da scaricare sul mio sito)

Dal sito del docente potete prelevare una raccolta di alcuni semplici sketch (file *esempi-arduino.zip*), utili come fonte di ispirazione, che potete copiare nella vostra cartella. Notare che ogni sketch e' contenuto in una cartella con lo stesso nome: questa struttura deve essere obbligatoriamente mantenuta

Nell'ultima esercitazione dovrete utilizzare lo sketch gia' preparato *adc_read_5*, contenuto nella stessa raccolta.



Quest'anno il nome e' diverso:
adc_read_5_2019.ino
Lo potete copiare dal mio sito web
Vedere la prossima esercitazione

Esempio: led rampa

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 velocità di esecuzione del μC per varie istruzioni: operazioni aritmetiche, funzioni, operazioni di Input/Output.
- Funzioni di timer: `millis()` e `micros()`
- `delay()` in millisecondi

```
serial_io § Esempio
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

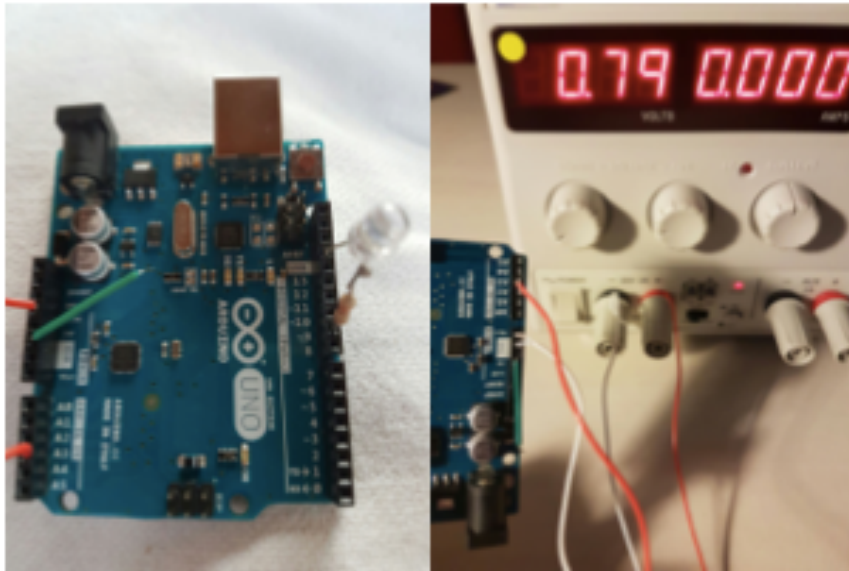
Hai scritto il numero 1000000
Moltiplicato per 10 = 10000000
durata: 1052 micro S

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

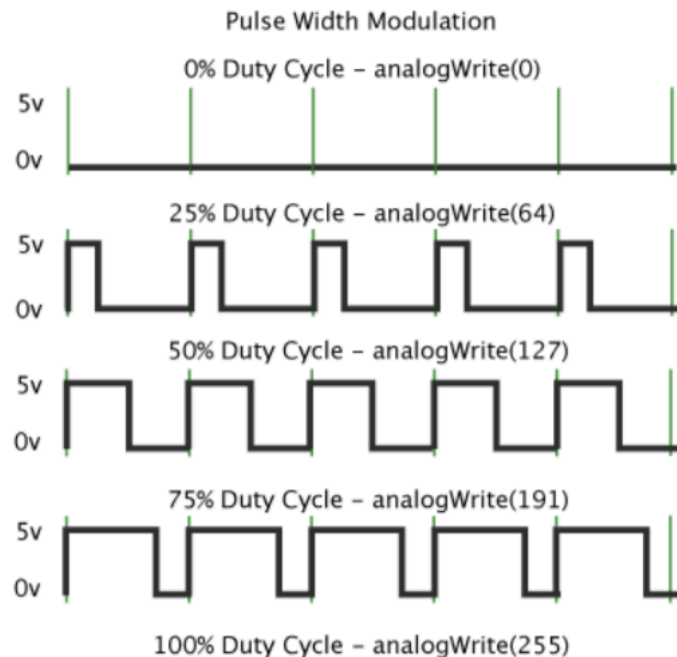
 Scorrimento automatico Hai premuto Invia ma non ..
```

Volendo, se avete tempo e voglia, potete vedere che tipo di correlazione c'è tra il tempo di esecuzione e il numero impostato

Operazioni in Input/Output



- Leds per visualizzazione degli outputs (protetti da una resistenza)
- Input forniti da generatore triplo o dai 5v (3.3V) di Arduino
- `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 addition to PWM capabilities on the pins noted above, the Due has true analog output when using `analogWrite()` on pins DAC0 and 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)!!!");
    }
  }
}
```

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

- ❖ 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

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

Dovreste ottenere questo (output arduino uno)

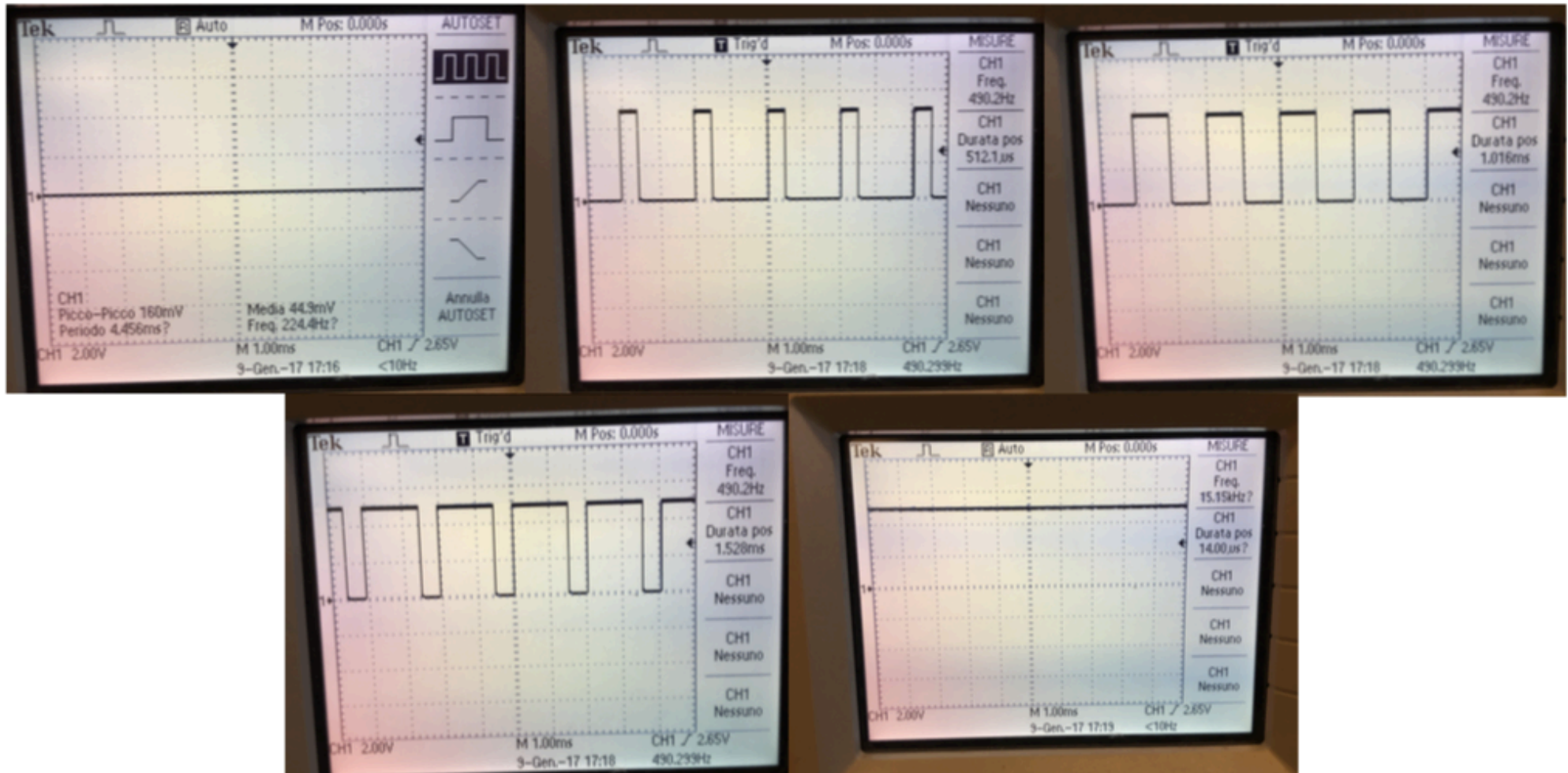


Immagine 5: Tensione Pin 9 a circa 490 Hz con duty cycle rispettivamente da sinistra a destra e da sopra a sotto pari a 0, 64, 127, 191, 255

Esempio: altro utilizzo di analogWrite()

La luminosita' del Led varia linearmente in maniera periodica

```
Led_Rampa
int ledPin = 9;          // LED connected to digital pin 9
int val = 0;            // val: 0-255 (duty cycle)
int i = 0;
int millisecondi = 10;
void setup()
{
  Serial.begin(9600);
  Serial.println("This is an Arduino-based dimmer...");
  pinMode(ledPin, OUTPUT);
}
void loop()
{
  for (i=0;i<255;i++)
  {
    val = i;
    analogWrite(ledPin, val);
    delay(millisecondi);
  }
  for (i=0;i<255;i++)
  {
    val = 255-i;
    analogWrite(ledPin, val);
    delay(millisecondi);
  }
}
```

- ❑ Nell'esempio e' riportato un onda triangolare, ma potete scrivere anche altri tipi di variazione, ad esempio un dente di sega
- ❑ Visualizzate con l'oscilloscopio l'uscita del pin 9, dovrete vedere un andamento a "fisarmonica". Se ci riuscite, riportate alcuni screenshot sulla relazione

**NON DOVETE FARE QUESTA PARTE
NELLA RELAZIONE DI QUEST'ANNO**

Dovreste ottenere questo

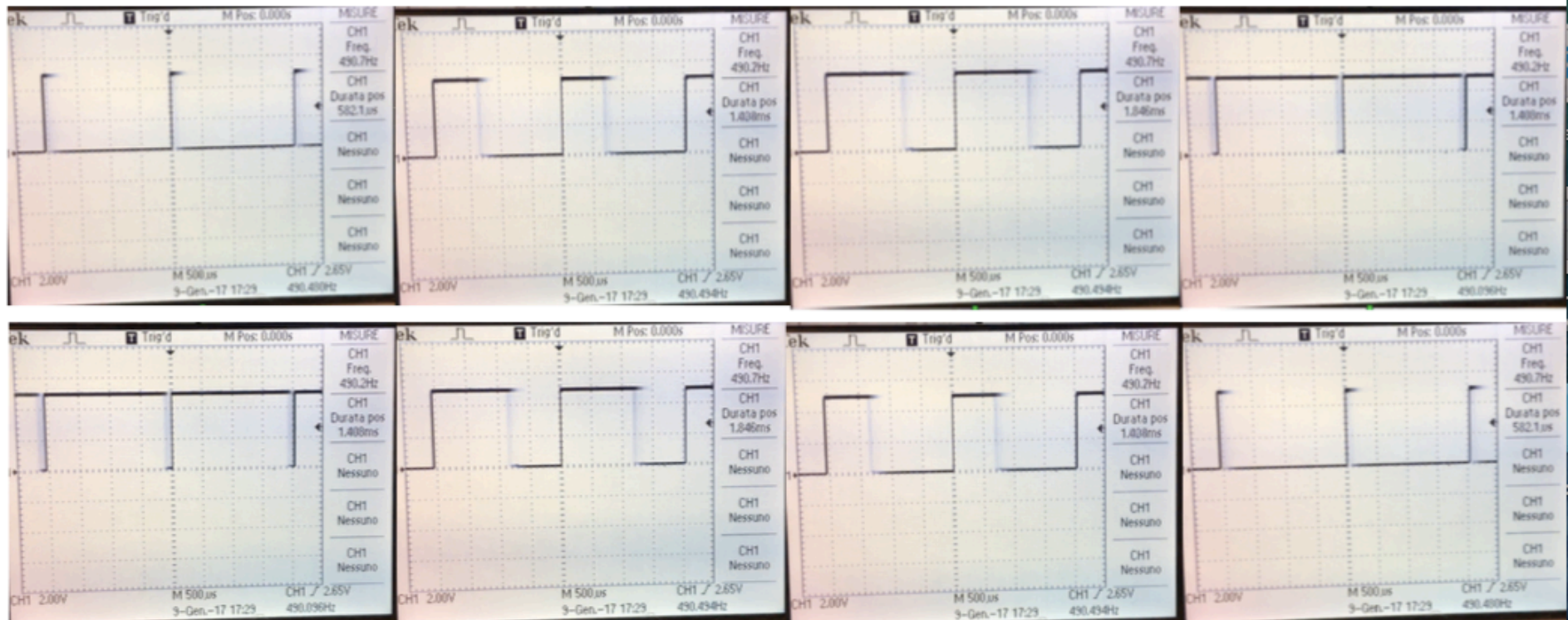


Immagine 6: Fotosequenza del video all'oscilloscopio. Da sopra a sotto e sinistra a destra.

Studio del DAC

Scrivere ed eseguire un programma per effettuare un test di linearità, 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.

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



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

V_{REF} e' 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

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

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(" ");
}
```

Questo e' 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'e' sempre una differenza di 10-14 unita).

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

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

L'ADC di Arduino Uno e' a 10 bit che e' anche il default di Arduino Due. Puo' 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)!!!");
    }
  }
}
```


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()){
  }
}
```

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));
}

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

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

The `max` parameter should be chosen according to the data type of the variable in which the value is stored. In any case, the absolute maximum is bound to the `long` nature of the value generated (32 bit - 2,147,483,647). Setting `max` to a higher value won't generate an error during compilation, but during sketch execution the numbers generated will not be as expected.

Input analogico: calibrazione ADC

Scrivere ed eseguire un programma per effettuare la calibrazione dell'ADC del μC , scegliendo uno degli ingressi analogici. Utilizzare come Analog Reference il default (3.3 V) e come tensione da convertire l'uscita dell'alimentatore triplo ($0 \div 3 V$) opportunamente collegata all'ADC di Arduino tramite la basetta sperimentale. Costruire il grafico della calibrazione misurando dieci-venti punti a diverse tensioni.

ATTENZIONE violare queste regole danneggia l'ADC di Arduino:

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

Provare la conversione utilizzando come tensione d'ingresso una tensione sinusoidale di ampiezza 1V tra 0 e 1V. Copiate i valori delle letture di Arduino in una tabella in modo da poter poi fare un grafico con open Office e verificare la fedeltà della forma d'onda ottenuta rispetto a quella di partenza.

Memento: l'ADC di arduino non legge valori negativi di tensione. Aggiungete un offset 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);
}
```

Potete lasciare il valore di default di 10 bit

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

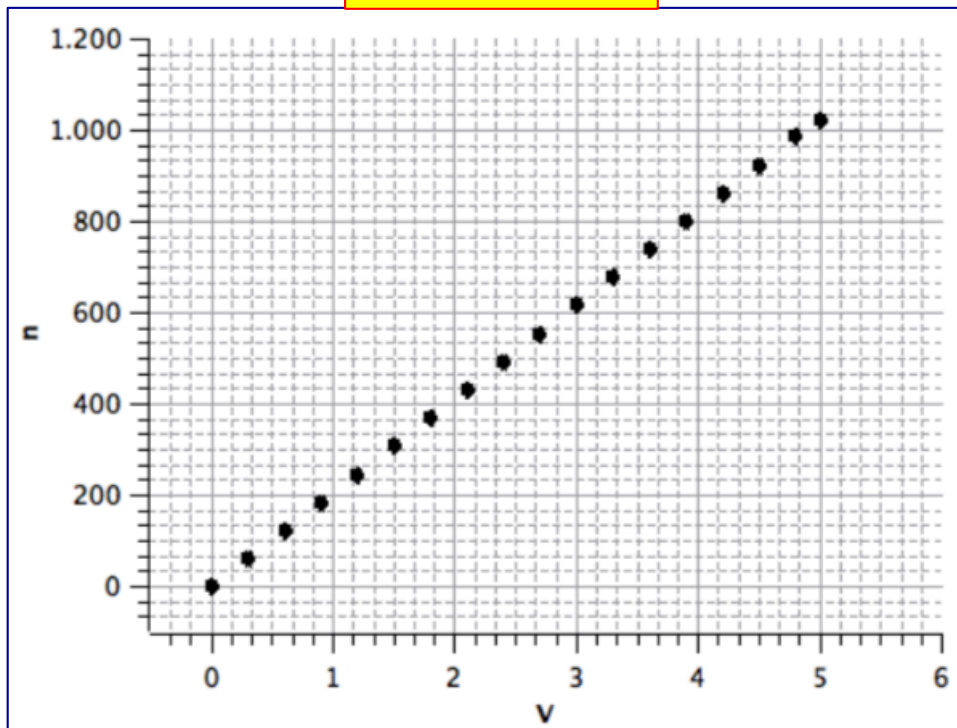
$$\text{Output} = \frac{2^{10} - 1}{V_{ref}} V_{in}$$

Calibrazione dell'ADC di arduino

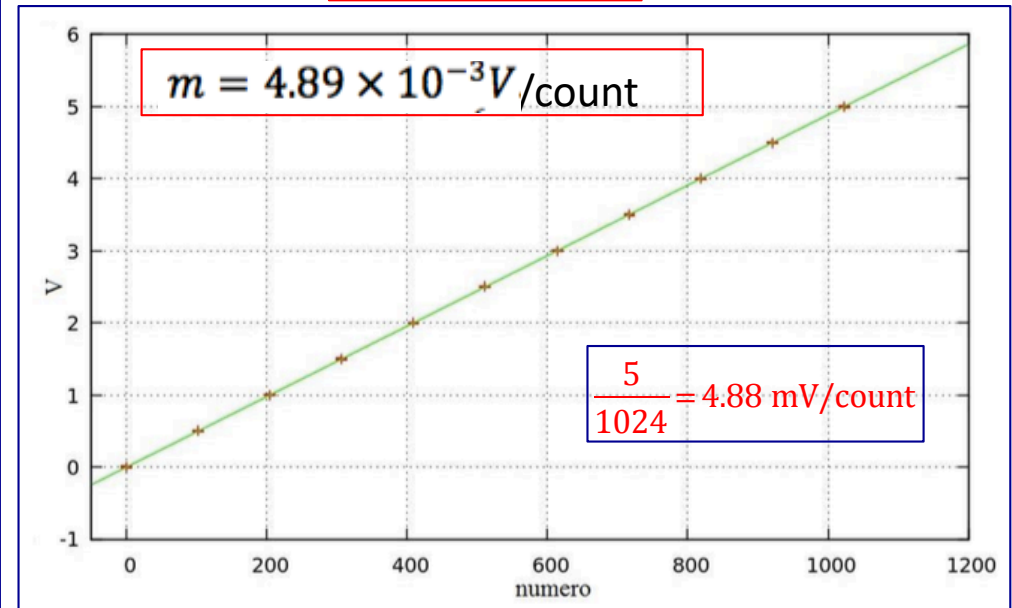
- Modificate il programma in modo da scrivere il valore di RdVal (e non Vx)
- Mandate in input una decina di valori di tensione continua compresi tra 0 e 3.3 V
- Costruite il grafico tra il valore dell'ADC e la tensione inviata
- Ricavate il valore della calibrazione

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 dovrete costruire il primo grafico
- Pero' voi avete bisogno di una costante di calibrazione che converta il numero letto dall'ADC in un valore di tensione, quindi dovrete costruire il secondo grafico; la costante di calibrazione che cercate e' il coefficiente angolare di questa retta.
- La costante di calibrazione vi servira' per la misura successiva

Input analogico

Conversione Analogico – Digitale di una forma d'onda

Ovvero campionamento digitale di una forma d'onda

Scrivere ed eseguire un programma per effettuare la calibrazione dell'ADC del μC , prendendo uno degli ingressi analogici. Utilizzare come Analog Reference il default (3.3 V) e come tensione da convertire l'uscita dell'alimentatore triplo (0 ÷ 3.3 V). Costruire il grafico della calibrazione.

Provare la conversione utilizzando come tensione d'ingresso una tensione sinusoidale o un'onda triangolare. Scrivere i risultati su una tabella in modo da poter poi fare un grafico con open Office e verificare la fedeltà del risultato ottenuto rispetto all'onda di partenza.

il punto 1) l'abbiamo fatto, andiamo al punto 2)

- 1 Calibrazione dell'ADC del μC , usando un ingresso analogico.
 - Utilizzare come Analog Reference il default (3.3 V) e come tensione da convertire l'uscita dell'alimentatore triplo (0 - 3.3 V).
 - Costruire il grafico della calibrazione.
- 2 Tensione sinusoidale e un'onda triangolare in ingresso.
 - Scrivere i risultati su una tabella in modo da poter poi fare un grafico con open Office e verificare la fedeltà del risultato ottenuto rispetto all'onda di partenza

Programma lettura ADC (waweform)

```
// The program samples a waveform 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 segnale sul pin analogico 3
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(" ");
}

void loop() {
  // start the loop to sample the waveform

  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
}
```

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

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

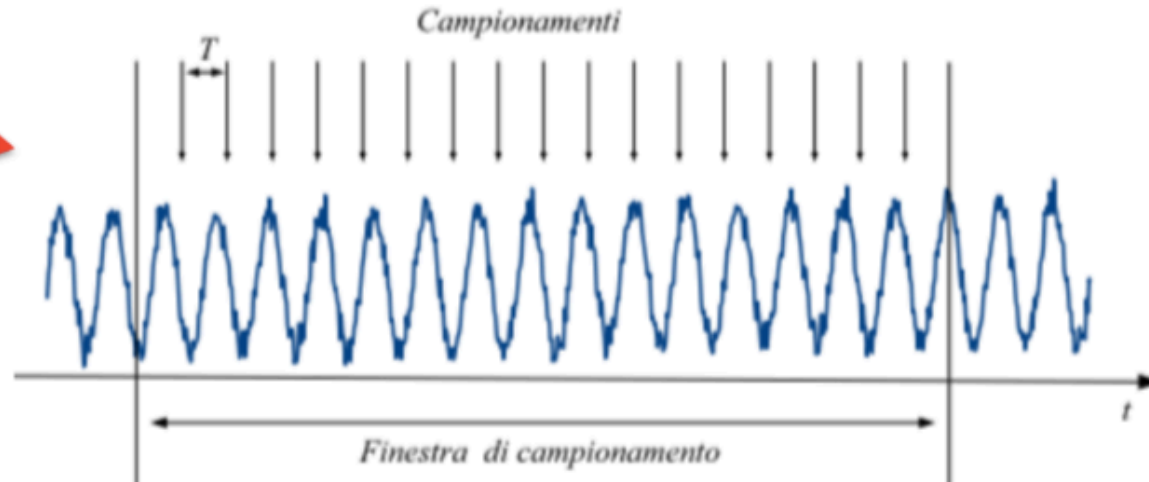
    delayMicroseconds(T);
  }

  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

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

$T = 5 \mu\text{s} \rightarrow f_c = 200 \text{ kHz}$	$\rightarrow f_N = 100 \text{ kHz}$	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.

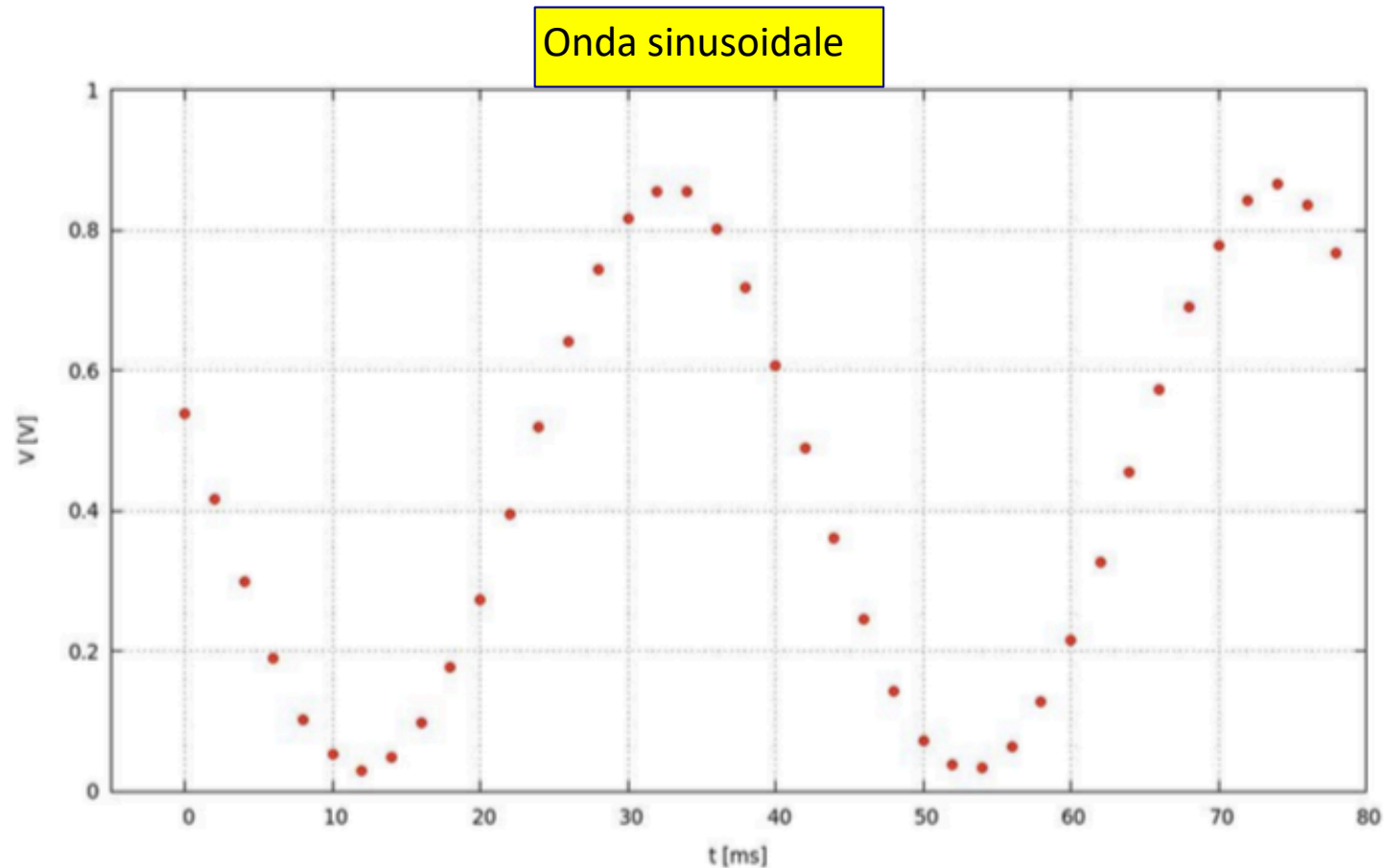
$$f_{\min} = 1/\text{finestra} = 1/5 \text{ ms} = 200 \text{ Hz}$$

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 verrebbe tagliato.

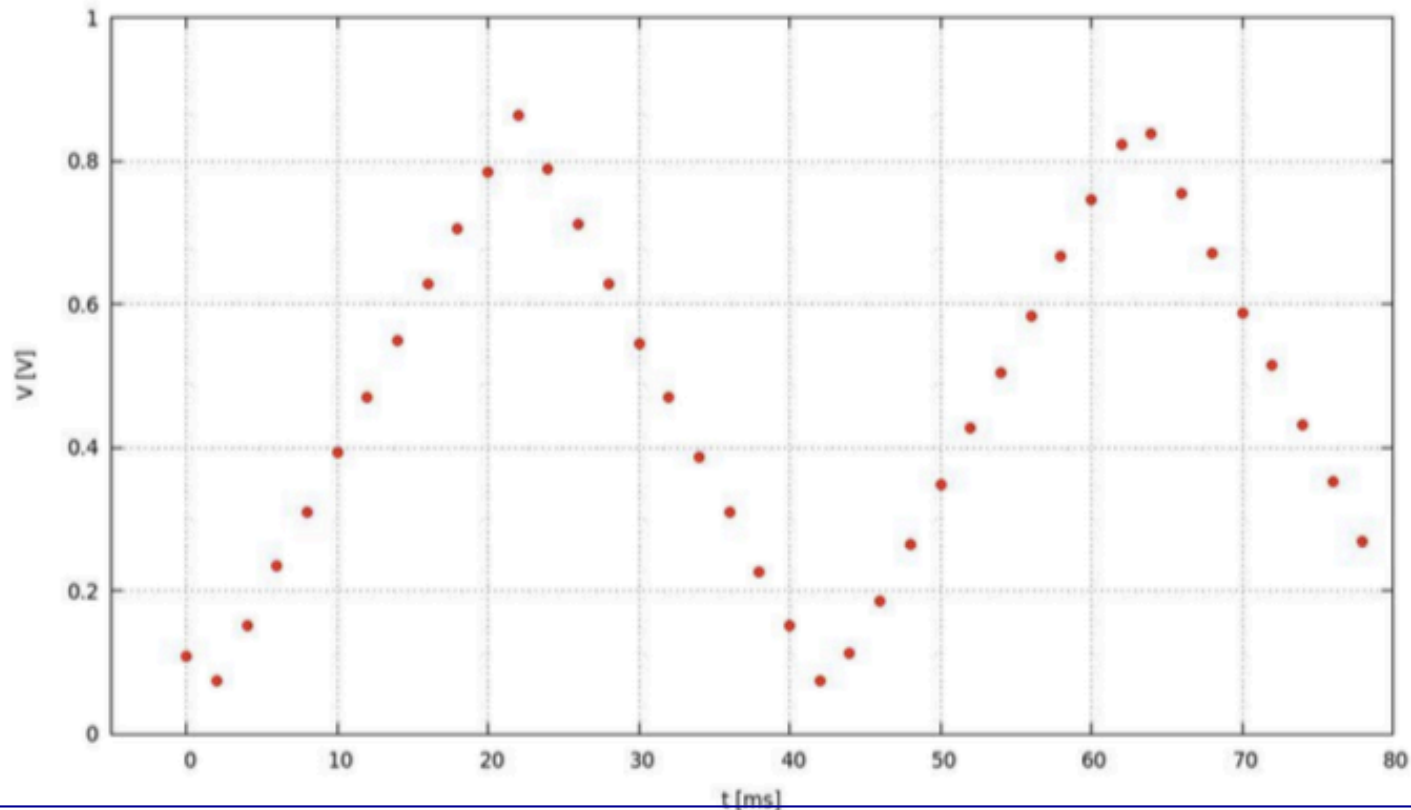
Dovreste ottenere questa cosa



- 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 fa la DFT per vedere se funziona.

Dovreste ottenere questa cosa

Onda triangolare



- 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