Multimetro digitale basato su microcontrollore STM32 con rilevamento della forma d'onda

Marco Cipriani

SAPIENZA
UNIVERSITÀ DI ROMA

Facoltà di Ingegneria dell'informazione, informatica e statistica

Corso di laurea in Ingegneria Elettronica

Relatore: Prof. Emanuele Piuzzi

A.A. 2023-2024

Sommario

- **Introduzione**
- **Funzionamento dei multimetri digitali**
- **Architettura del multimetro proposto**
- **Firmware e algoritmi DSP**
- **Prove svolte**
- **Conclusioni**

Introduzione

Motivazione e obbiettivo della tesi

I multimetri attualmente in commercio non sono in grado di rilevare la forma d'onda della tensione in ingresso, né fornire maggiori informazioni utili alla diagnostica di guasti elettrici, se non le semplici misure di tensione, corrente e resistenza.

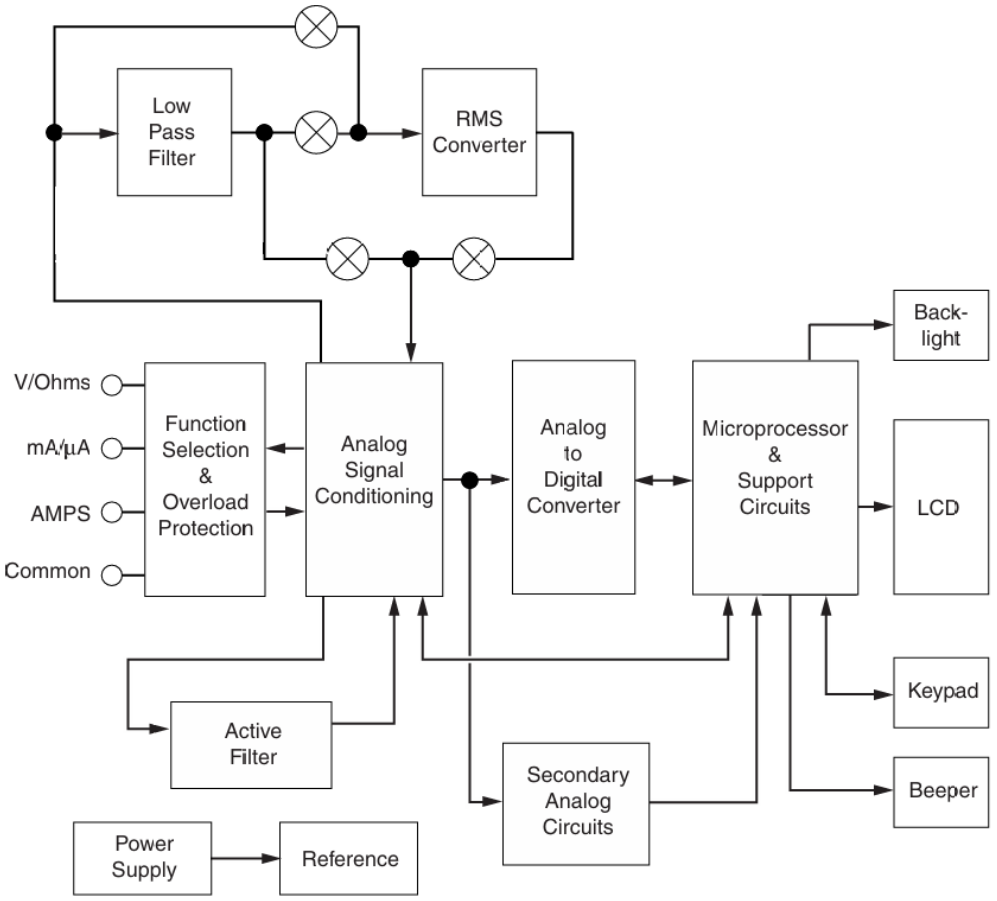
I microprocessori moderni hanno grande potenziale nell'elaborazione digitale dei segnali. È possibile fare di meglio?

Funzionamento dei multimetri digitali

Caso studio: *Fluke® 87V*

I blocchi circuitali principali nel circuito del Fluke® 87V sono:

- Protezione ingressi
- Condizionamento del segnale
- Convertitore RMS
- Convertitore analogico-digitale
- Logica di controllo

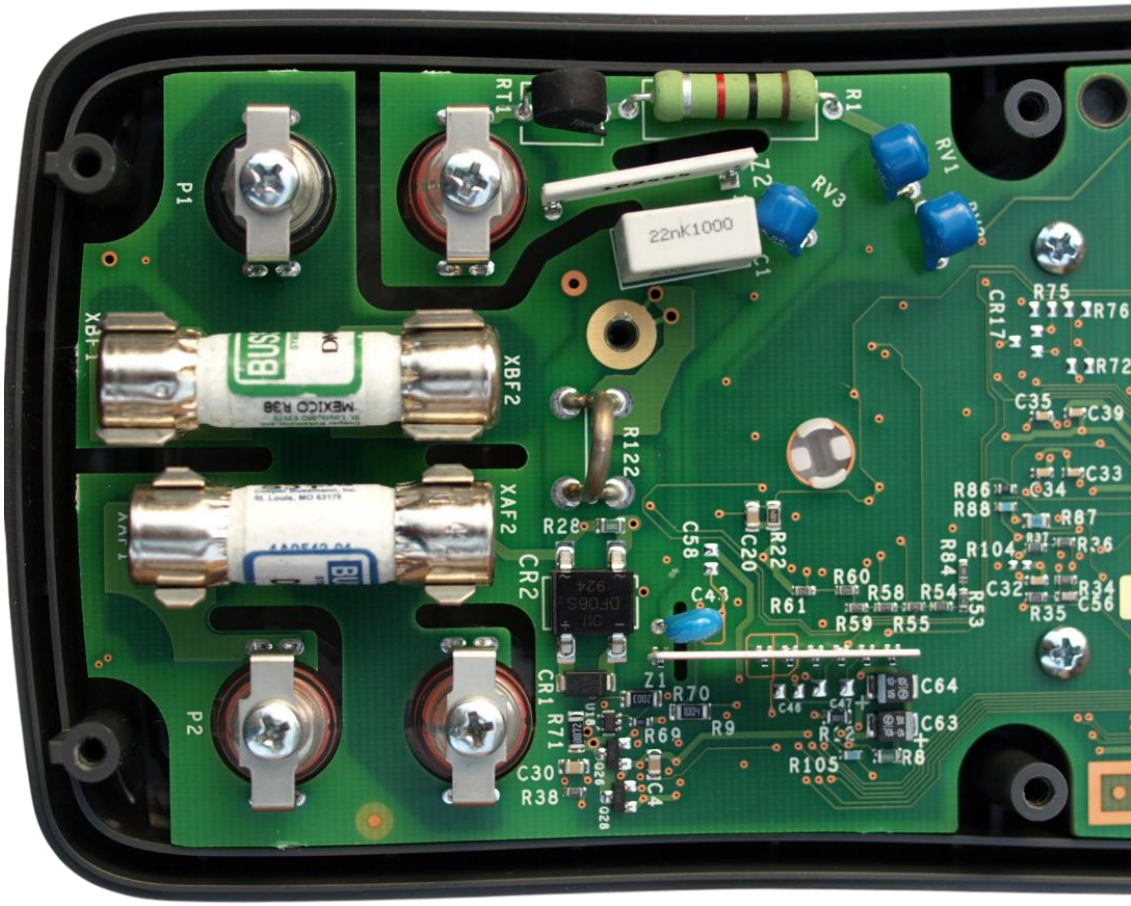


Funzionamento dei multimetri digitali

Protezione ingressi

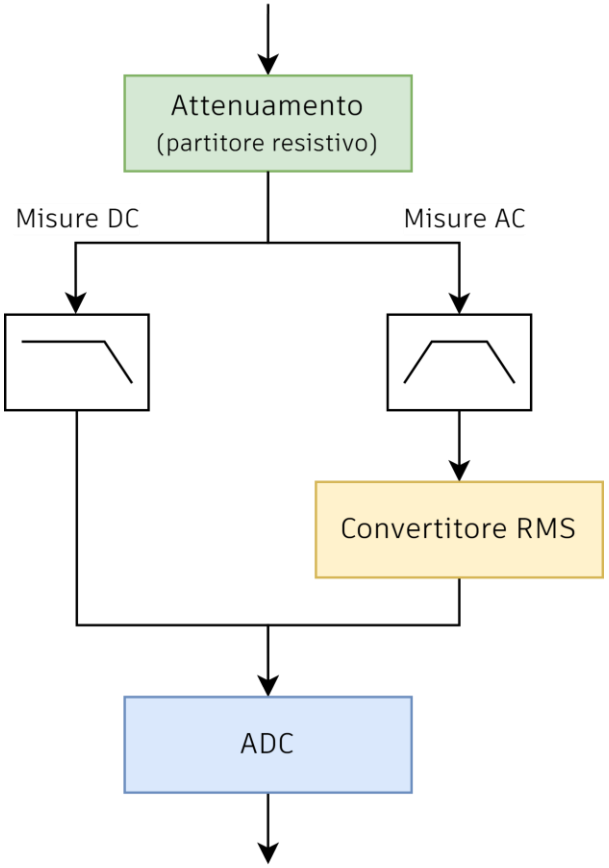
Possibili componenti:

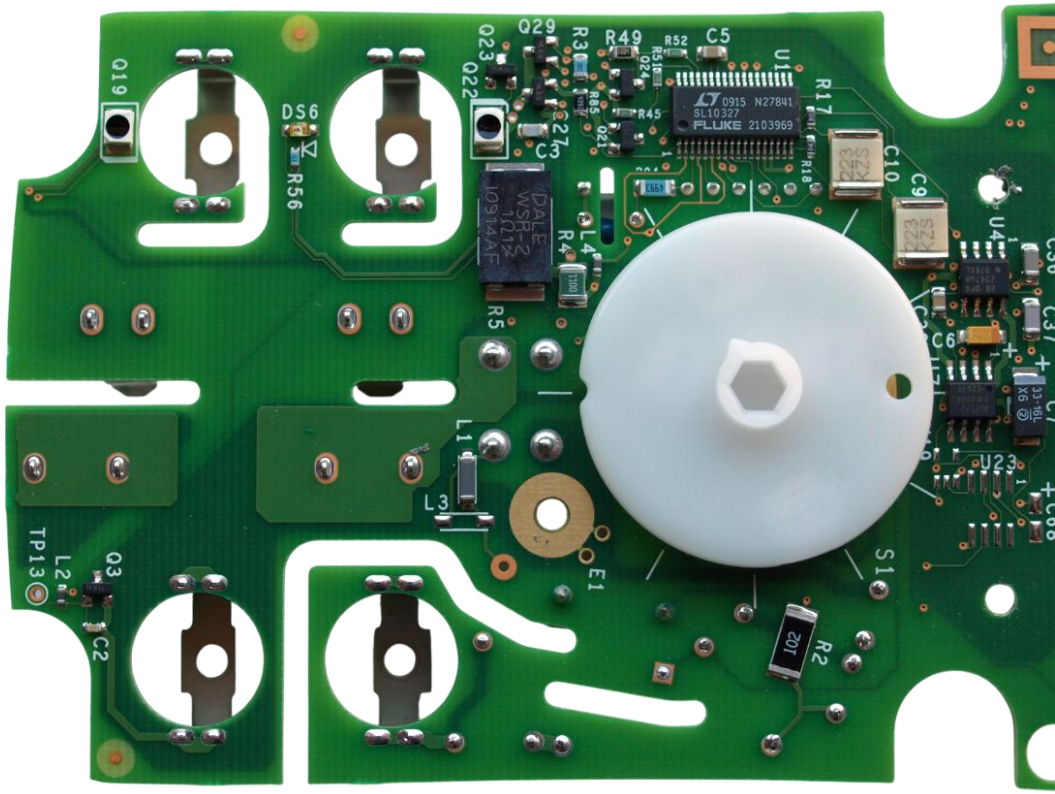
- *Metal-Oxide Varistor*
- Fusibili *PPTC*
- Fusibili *HRC*
- Ponte di diodi con *clamping*
- Diodi *TVS*



Funzionamento dei multimetri digitali

Selezione, filtraggio e condizionamento del segnale



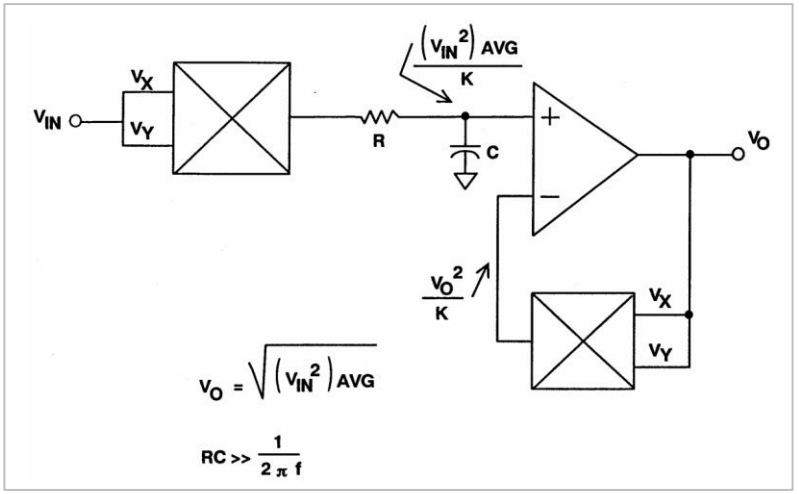


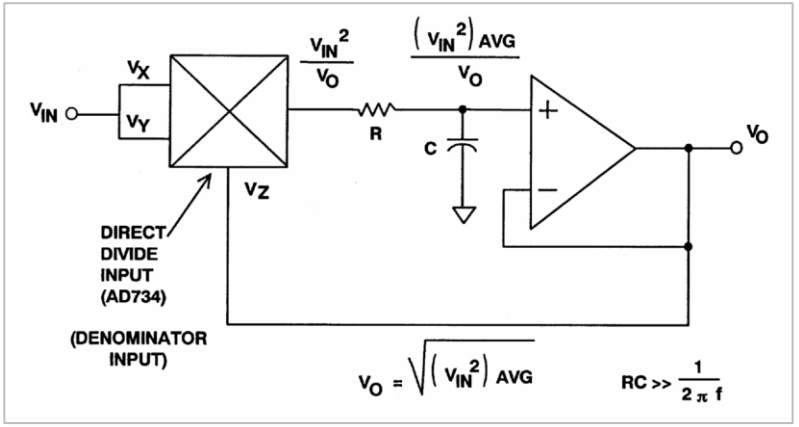
Funzionamento dei multimetri digitali

Convertitore *RMS*

Tipologie:

- A valore di picco
- A valore medio
- *True RMS* esplicito (figura superiore)
- *True RMS* implicito (figura inferiore)





Funzionamento dei multimetri digitali

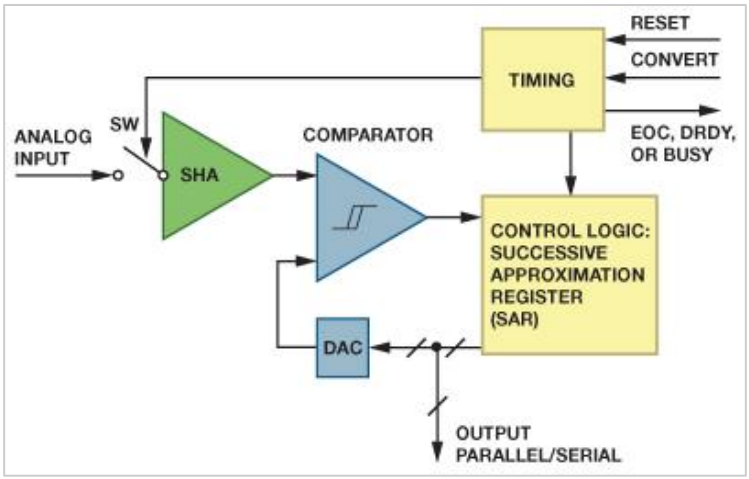
Convertitore analogico-digitale

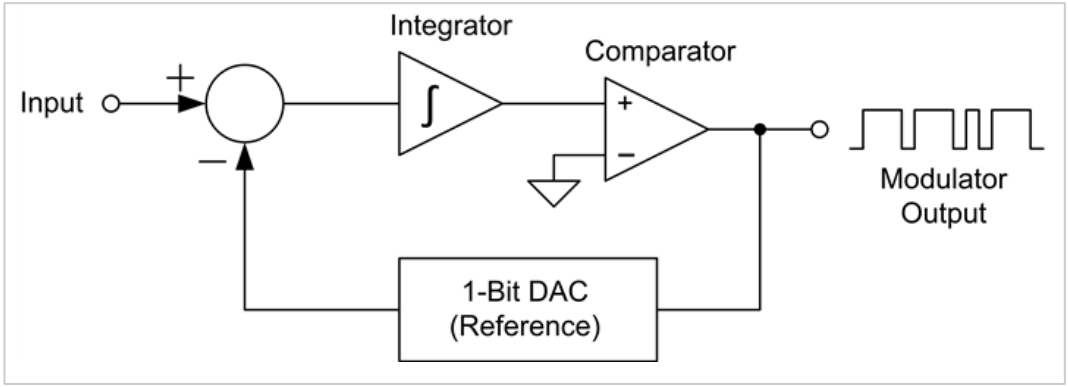
Tipologie:

- A doppia rampa
- Ad approssimazioni successive (*SAR*)
- Delta-sigma (ΔΣ)

Interfacce:

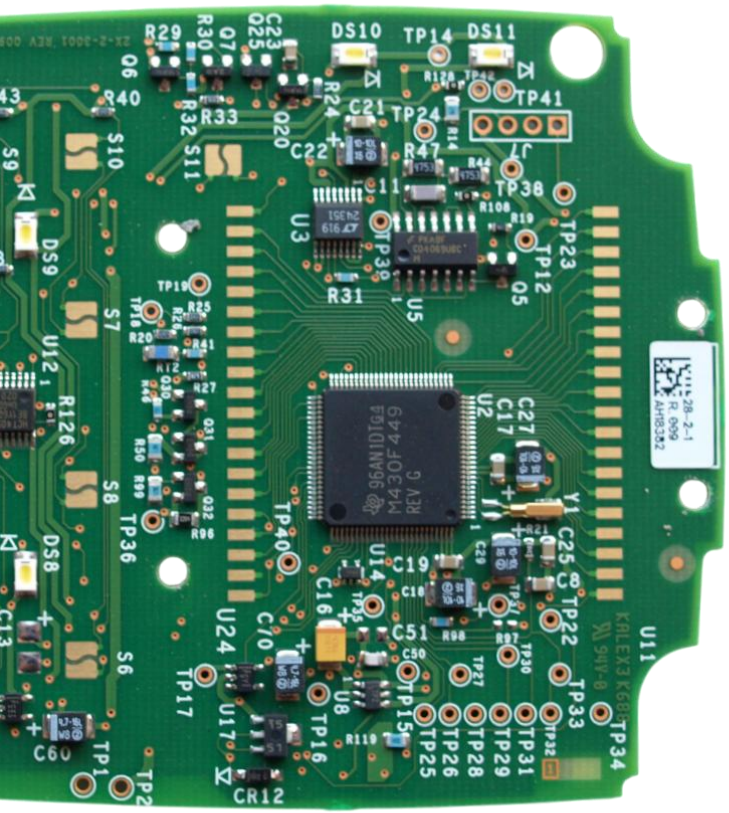
- *I²C*
- *SPI*





Funzionamento dei multimetri digitali

Logica di controllo



Funzioni:

- selezionare il tipo di misura in base all'input proveniente dall'interruttore rotativo e dai pulsanti
- effettuare l'*auto-ranging* se supportato
- comandare l'acquisizione
- leggere i dati sul *bus*
- applicare fattori di conversione e calibrazione
- visualizzare la misura sul *display*

Architettura del multimetro proposto

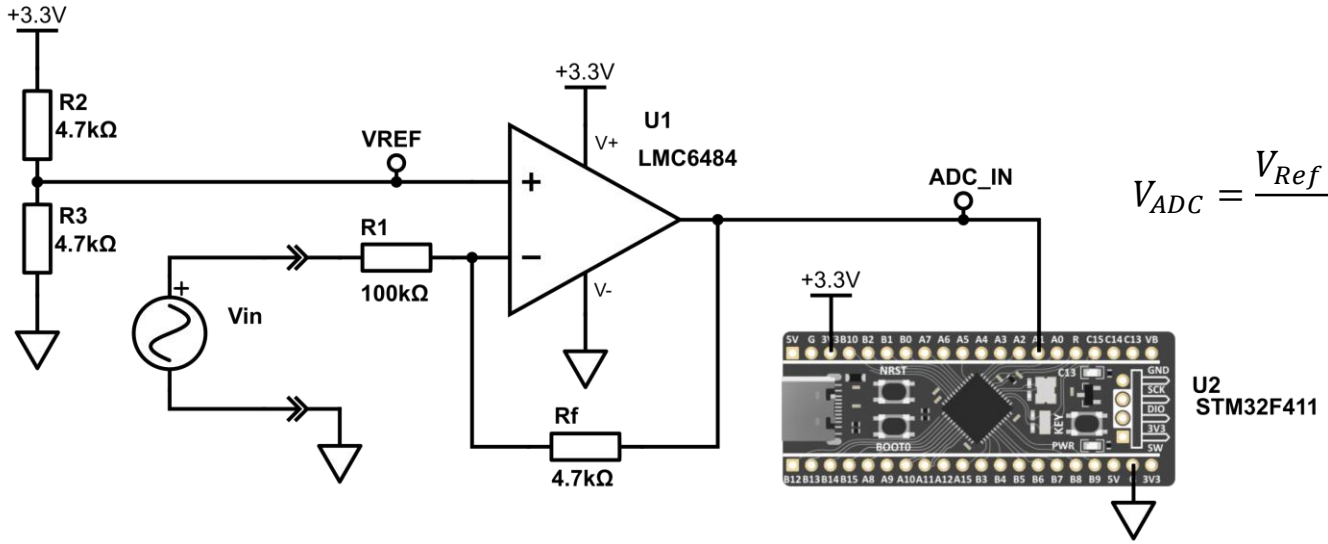
Requisiti e funzionalità

- Rilevamento della forma d'onda del segnale in ingresso, tra cui:
  - Sinusoidale
  - Quadra
  - Triangolare
  - Dente di sega
- Valore medio di tensione, ossia la componente *DC*
- Valore efficace di tensione (*RMS*)
- Valori minimo, massimo e picco-picco
- Frequenza e periodo
- Dinamica d'ingresso di $\pm 30\ V$

Architettura del multimetro proposto

Front-end analogico e microprocessore

- Arm Cortex-M4 con FPU e istruzioni SIMD: **STM32F411**
- Amplificatore operazionale per scalare $\pm 30\ V$ in $0 - 3.3\ V$

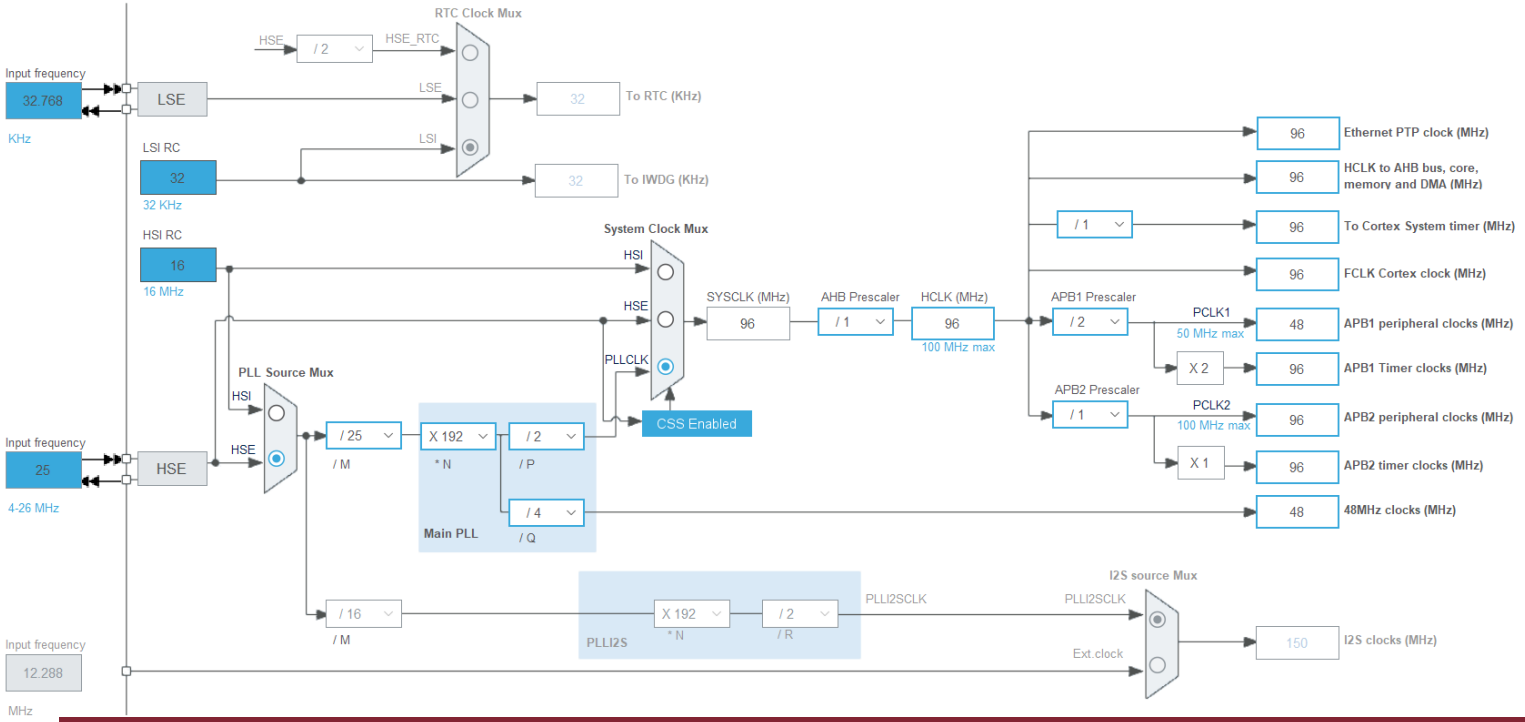


$$V_{ADC} = \frac{V_{Ref} \cdot (R_1 + R_f) - V_{IN} \cdot R_f}{R_1}$$

Architettura del multimetro proposto

Configurazione dell'ambiente di sviluppo

- Creazione del progetto in *STM32CubeIDE*
- *Clock tree* e *PLL*



Architettura del multimetro proposto

FPU, *SIMD* e libreria *Arm® CMSIS DSP*

- Per accelerare le operazioni di *digital signal processing*, è necessario abilitare l'unità di calcolo in virgola mobile
- Le operazioni *SIMD* (*Single Instruction/Multiple Data*) accelerano il calcolo della trasformata di Fourier discreta
- La libreria CMSIS DSP è ottimizzata per l'elaborazione di segnali su microcontrollori con memoria limitata

```
square:
  vmul.f32 s0, s0, s0
  bx lr
```

```
square:
  mov r1, r0
  push {r3, lr}
  bl __aeabi_fmul
  pop {r3, pc}
```

Architettura del multimetro proposto

Convertitore analogico-digitale *SAR*, *DMA* e *timer trigger*

$$f_{Nyquist} = 2 \cdot f_{max} = 200\ kHz$$

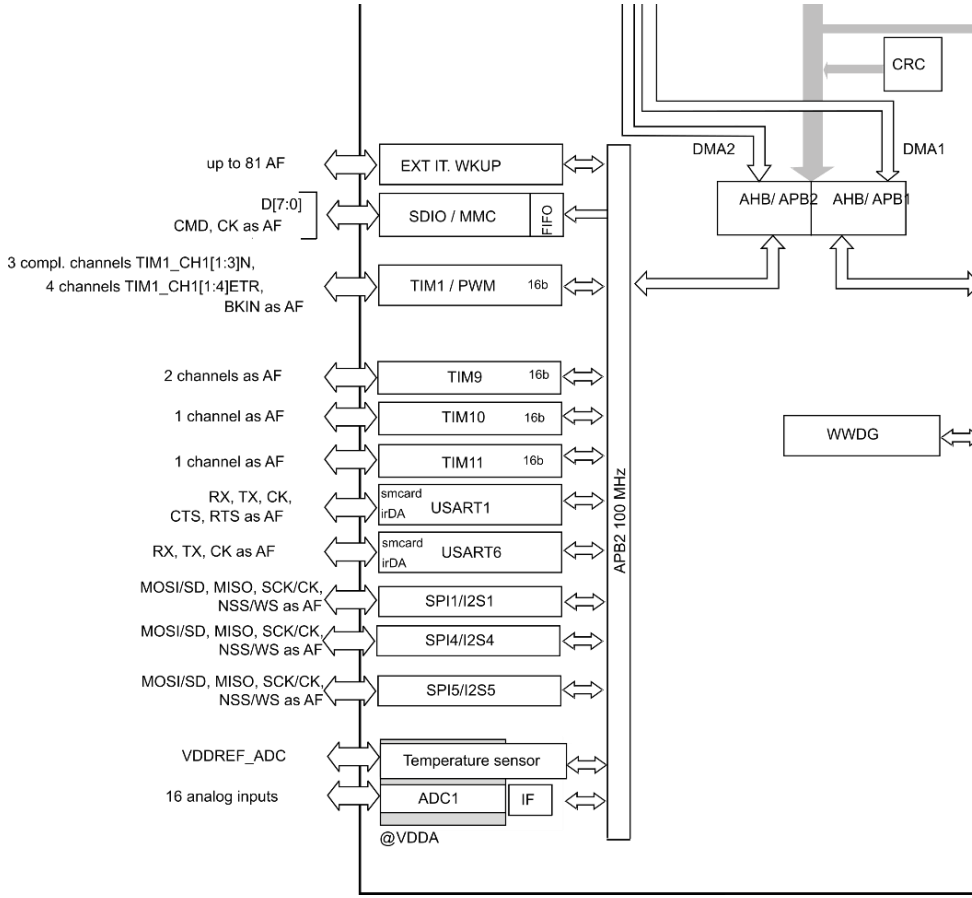
$$ADC\ \text{sampling rate} = \frac{1}{T_{CONV}} = \frac{1}{T_{SMPL} + T_{SAR}}$$

$$T_{SAR} = 15\ ADC\ \text{clock cycles @12-bit resolution}$$

$$T_{CONV} = T_{SMPL} + T_{SAR} < \frac{1}{f_{Nyquist}}$$

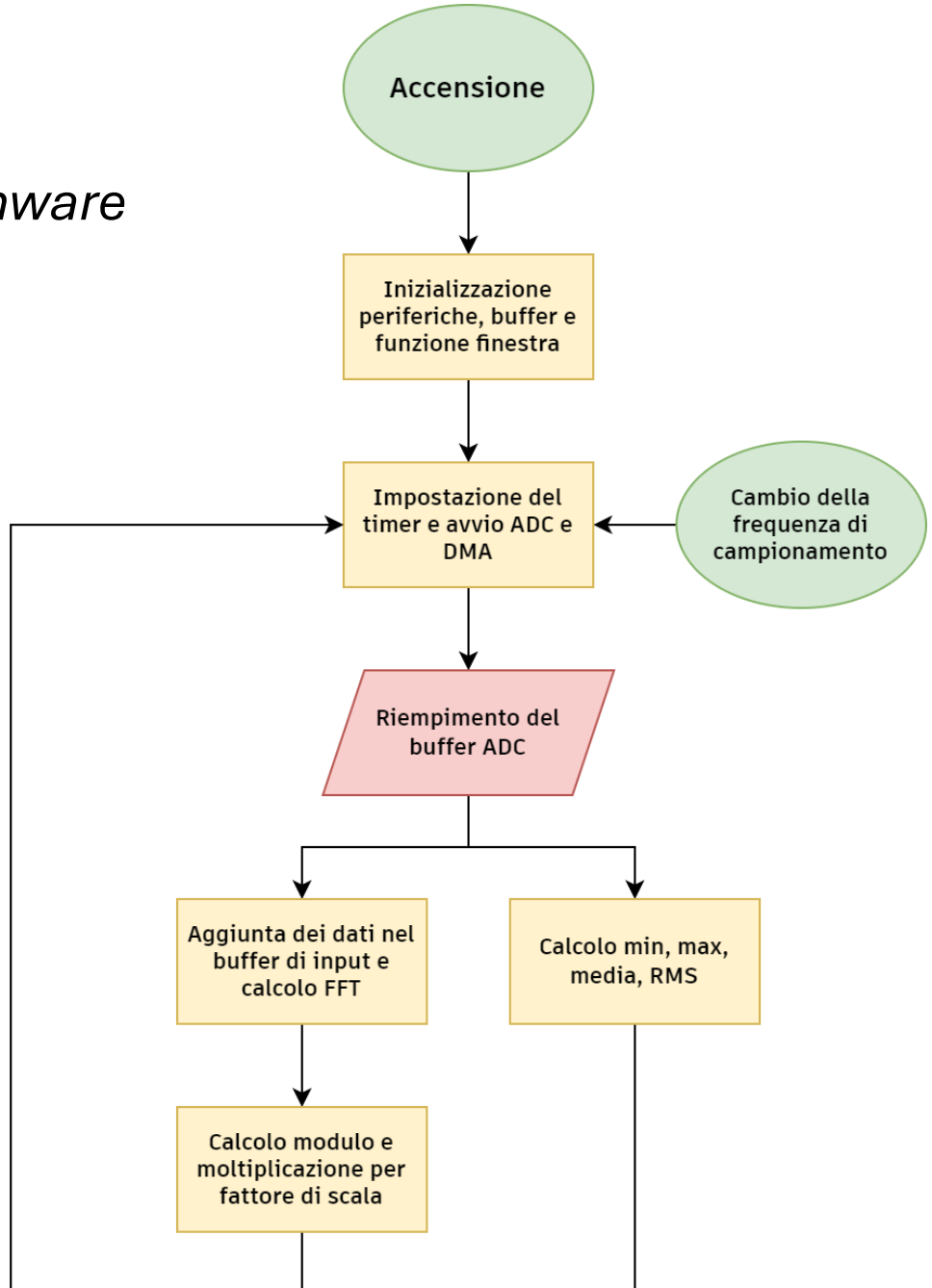
$$\Rightarrow \frac{15\ ADC\ \text{clock cycles} + T_{SAR}}{f_{ADC}} < \frac{1}{f_{Nyquist}}$$

$$\Rightarrow T_{SMPL} < 105\ ADC\ clock\ cycles$$

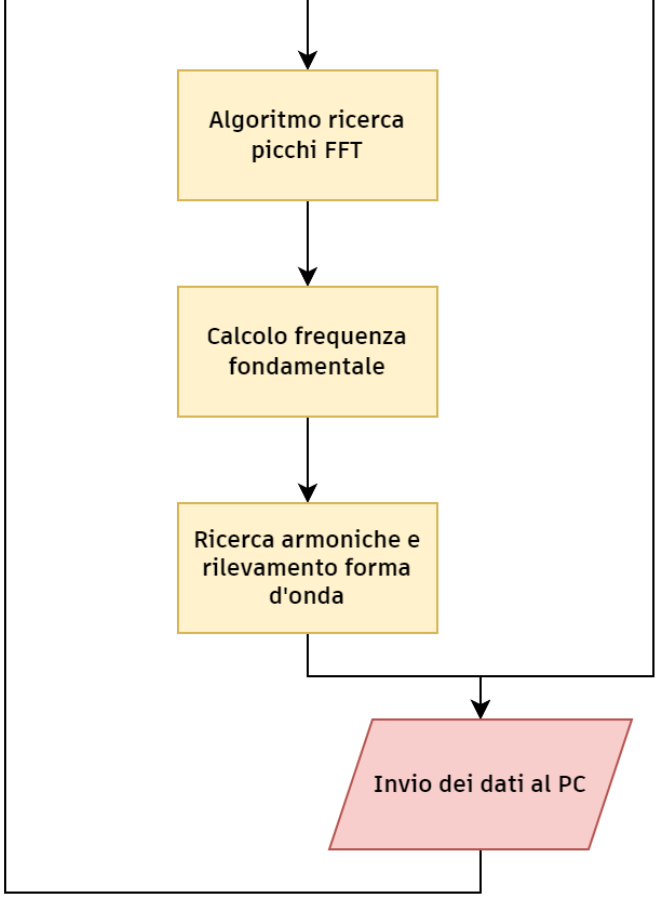


Algoritmi e codice DSP

Diagramma di flusso del *firmware*



$$k = k_1 \cdot k_2 = \frac{2}{N} \cdot \frac{\sum_{n=0}^{4095} \mathrm{Hann}_n}{N}$$



Algoritmi e codice DSP

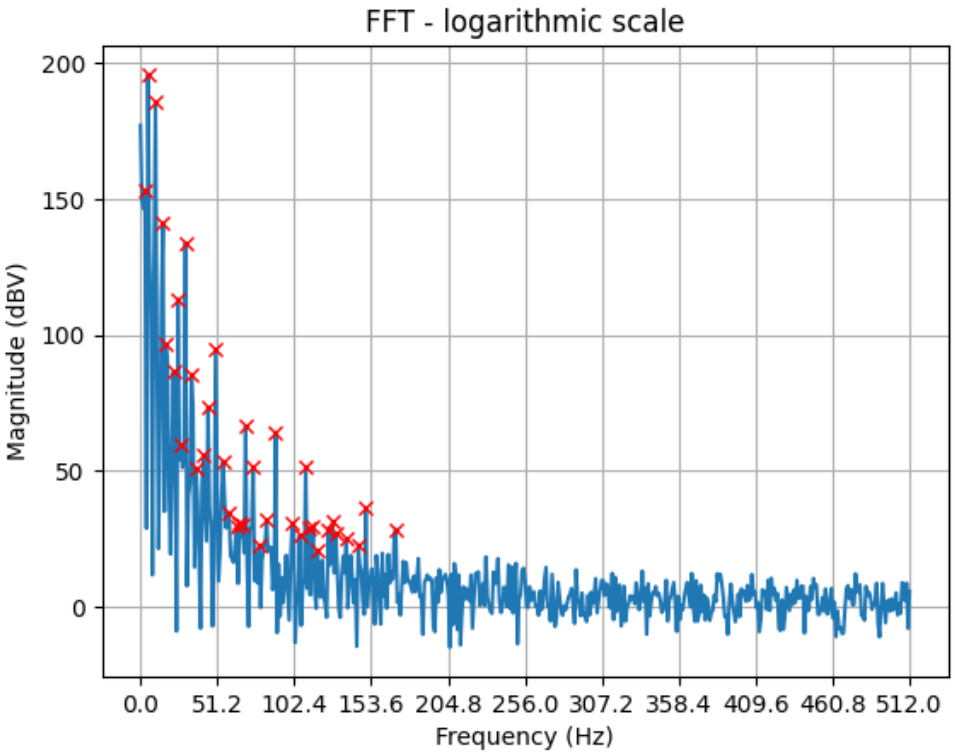
Diagramma di flusso del *firmware*

Algoritmi e codice DSP

Ricerca della fondamentale e delle armoniche

È stato implementato un algoritmo di ricerca picchi, che determina tutti i massimi locali sopra una soglia di rumore nel *buffer* di *output* dell'*FFT*.
Il picco più alto è assunto come frequenza fondamentale.
Le armoniche vengono cercate tra i picchi a frequenza multipla della fondamentale.

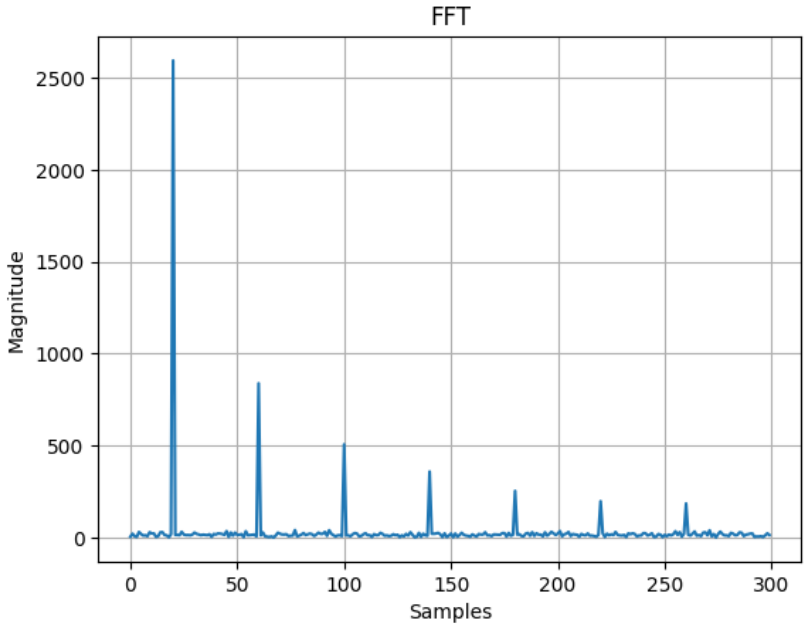
$$f = \text{index} \cdot f_{sampling}/N\,, \quad N = 4096$$



Algoritmi e codice DSP

Rilevamento della forma d'onda

L'algoritmo confronta le ampiezze relative dei picchi alla frequenza fondamentale e alle sue armoniche. L'assenza di armoniche indica la presenza di un segnale sinusoidale.



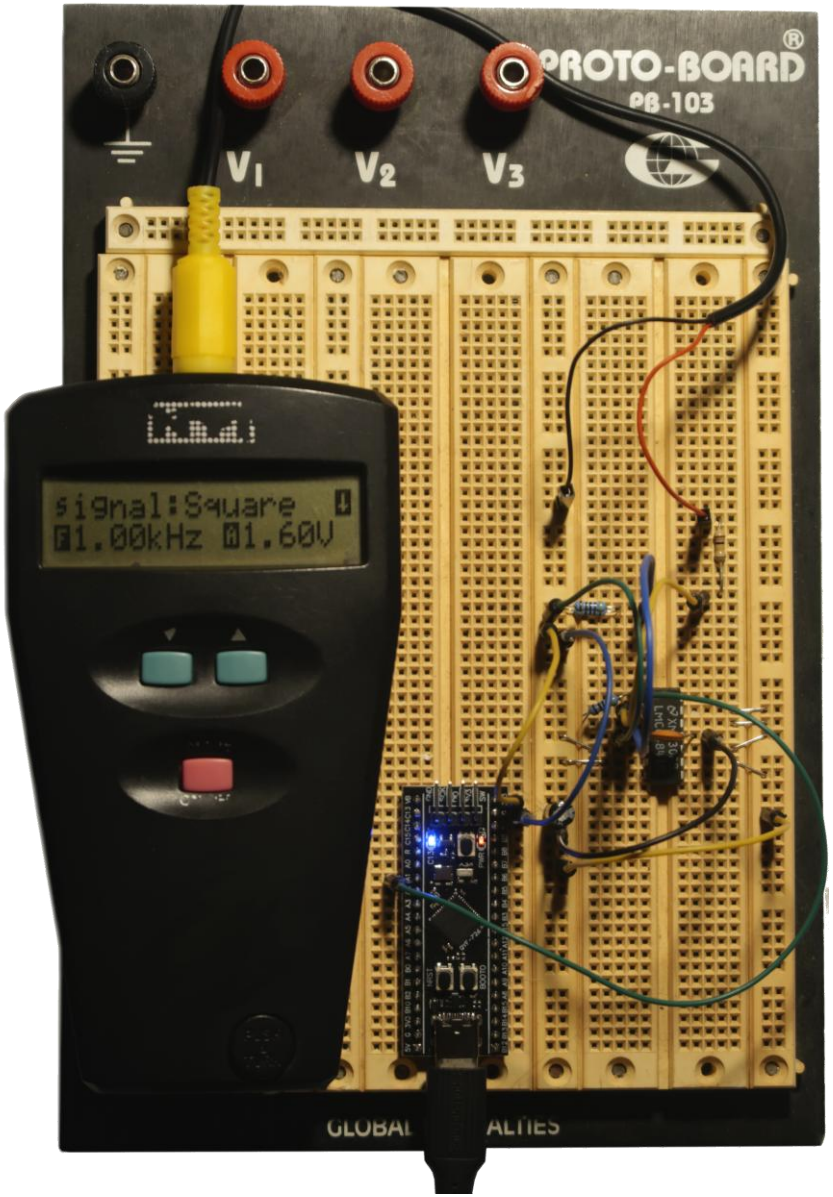
Tipo di onda	Serie di Fourier	R_2	R_3
Quadra	$x(t) = \frac{4}{\pi}\sum_{k=1}^{+\infty}\frac{1}{2k-1}\sin(2\pi(2k-1)\cdot t)$	0	1/3
Dente di sega	$x(t) = -\frac{2}{\pi}\sum_{k=1}^{+\infty}\frac{(-1)^k}{k}\sin(2\pi k\cdot t)$	1/2	1/3
Triangolare	$x(t) = -\frac{8}{\pi^2}\sum_{k=1}^{+\infty}\frac{(-1)^k}{(2k-1)^2}\sin(2\pi(2k-1)\cdot t)$	0	1/9

Con R_n rapporto tra l'ampiezza dell'n-esima armonica e l'ampiezza della componente fondamentale

Prove svolte

Hardware di *testing*

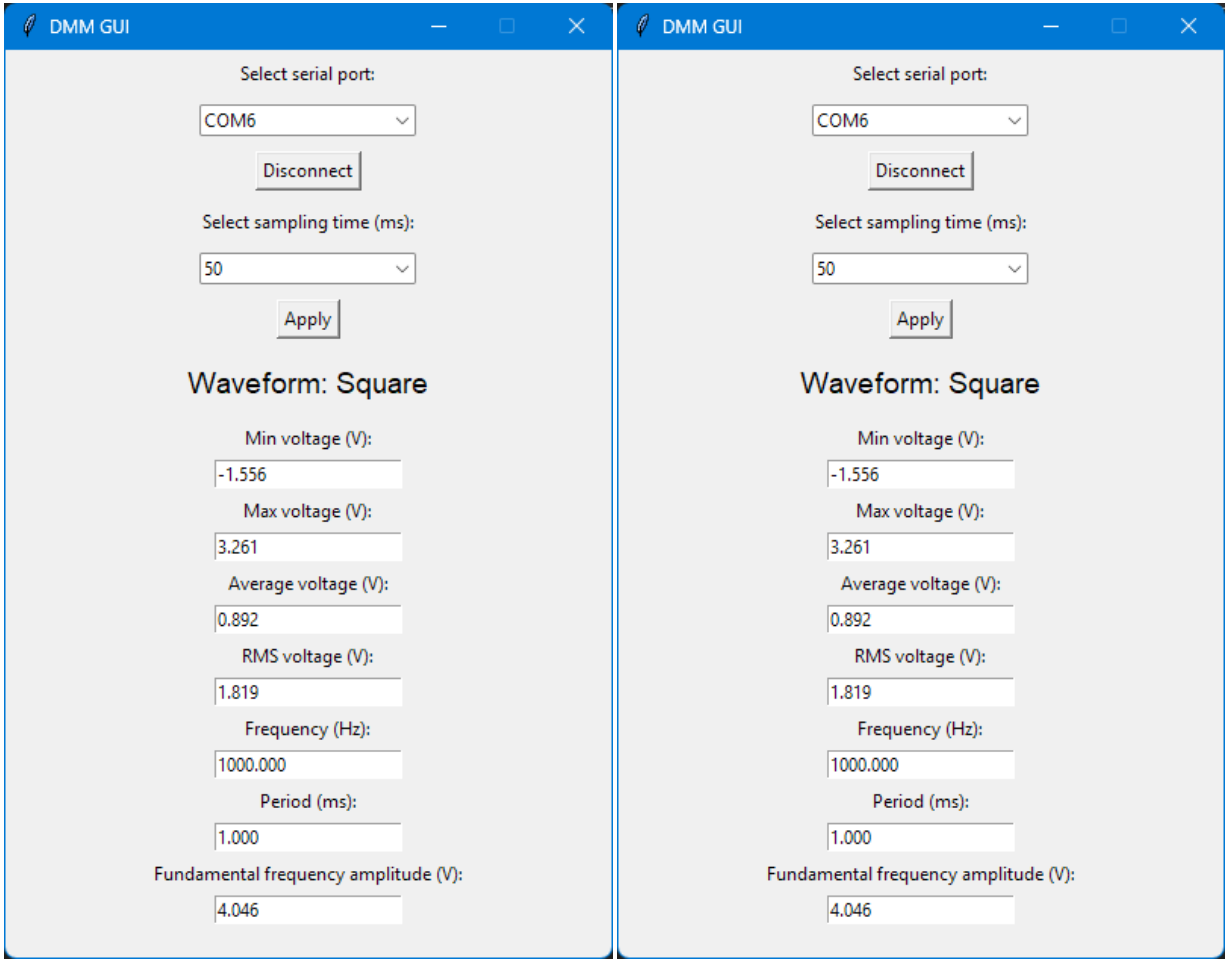
- Circuito su *breadboard*
- Test con:
  - Generatore di segnali audio
  - Trasformatore
  - Batterie



Prove svolte

Misure effettuate

- Interfaccia grafica scritta in *Python*
- Forme d'onda rilevate con successo
- Tempo di calcolo di soli 12 ms



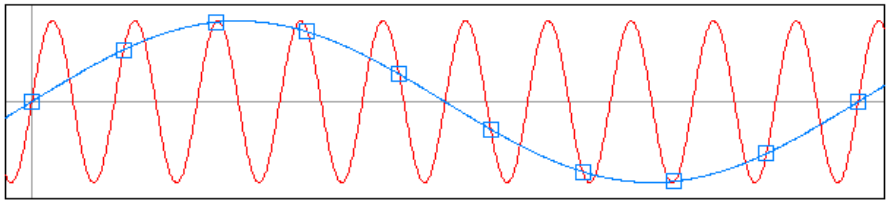
Prove svolte

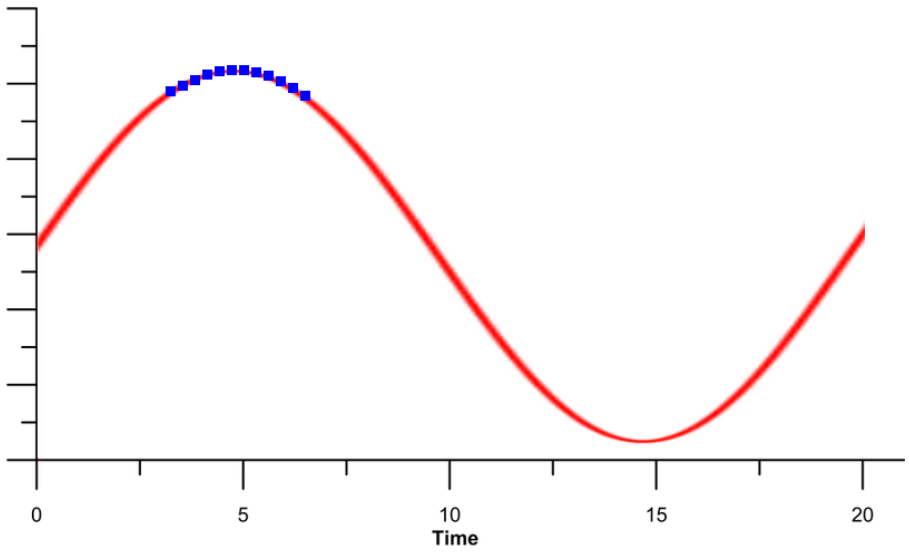
Problemi nelle misure di frequenza

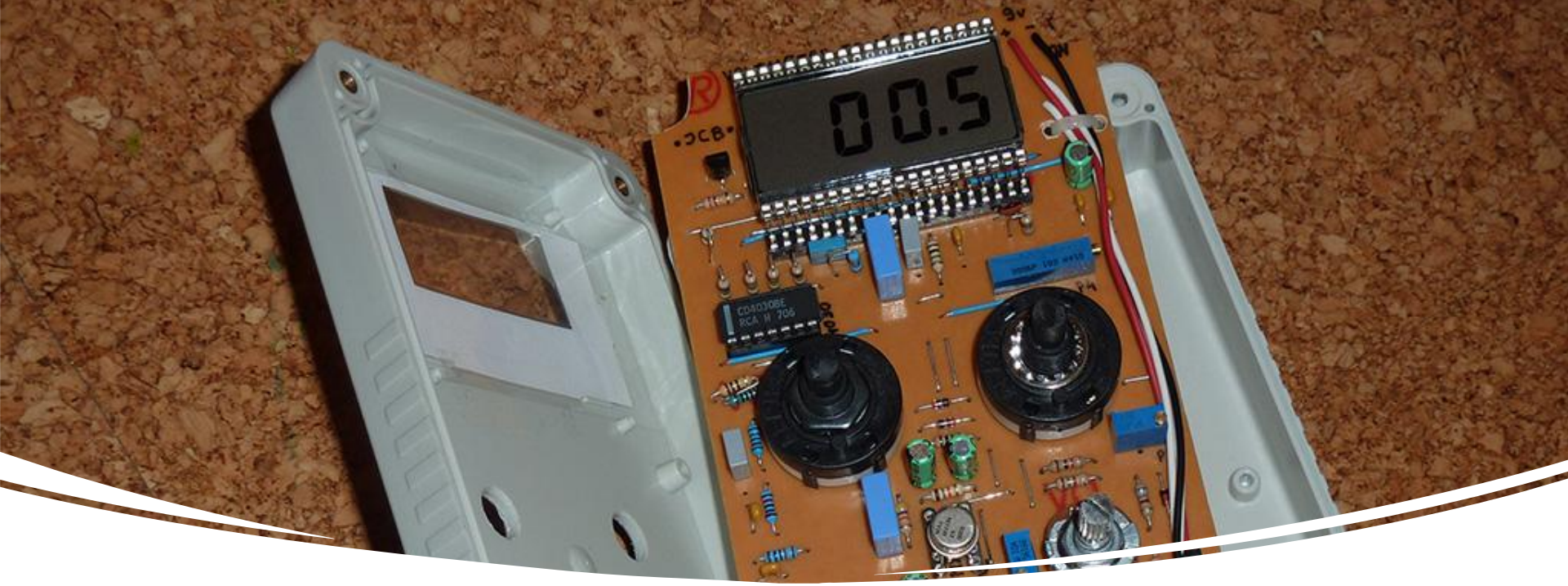
Il numero di campioni della trasformata di Fourier è risultato essere il fattore limitante nella risoluzione in frequenza:

$$R = f_S/N\,, \quad N = 4096$$

Il problema dell'*aliasing* ha forte importanza in un dispositivo che deve lavorare con un ampio spettro di frequenze. Anche un campionamento troppo veloce ha però delle problematiche ad esso associate.







Conclusioni

Possibili miglioramenti

- Front-end analogico con protezioni in ingresso e auto-ranging
- Frequenzimetro integrato e selezione automatica della frequenza di campionamento
- Riservare parte della memoria *SRAM* ai *buffer*
- Ottimizzazione del codice
- Alimentazione a batteria e display