

Nome Cognome:	<b>Naoki Pross</b>
Professione:	<b>Elettronico</b>
Titolo del progetto:	<b>Spectrum Analyzer</b>

Azienda:	<b>CPT Bellinzona</b> Centro Professionale Tecnico Viale S. Franscini 25 6500 Bellinzona		
Formazione approfondita:	<b>S.2 Sviluppare prototipi</b>		
Formatore:	<b>Rinaldo Geiler, Daniele Kamm</b>		
Data d'inizio:	<b>12.04.2018</b>	Ore a disposizione:	<b>83 UD</b>
Data fine lavoro:	<b>15.05.2018</b>	Ore effettive:	<b>– UD</b>



# Indice

<b>1</b>	<b>Introduzione</b>	<b>3</b>
1.1	Contesto . . . . .	3
1.2	Requisiti . . . . .	3
1.3	Concetti matematici . . . . .	3
1.4	Norme di progetto . . . . .	4
1.4.1	Software . . . . .	4
1.4.2	Hardware . . . . .	4
1.4.3	Programmazione . . . . .	4
<b>2</b>	<b>Hardware</b>	<b>5</b>
2.1	Schema a blocchi . . . . .	5
2.2	Selezione delle entrate . . . . .	5
2.3	Circuito di entrata . . . . .	6
2.4	Microcontroller . . . . .	7
2.5	Visualizzazione . . . . .	7
<b>3</b>	<b>Software</b>	<b>8</b>
3.1	Campionamento . . . . .	8
3.2	Trasferimento dei dati . . . . .	8
3.3	Interfaccia al Computer . . . . .	9
3.3.1	Librerie e codice di terzi . . . . .	9
3.3.2	Qt Framework . . . . .	9
3.3.3	Compilazione sotto Linux . . . . .	9
3.3.4	Compilazione manuale sotto Linux . . . . .	10
3.3.5	Compilazione sotto Windows . . . . .	11
3.3.6	Architettura . . . . .	11
3.3.7	Interfaccia utente . . . . .	13
3.4	Interfaccia al Display . . . . .	14
3.5	Fast Fourier Transform . . . . .	14
<b>4</b>	<b>Conclusioni</b>	<b>15</b>
4.1	Problemi riscontrati . . . . .	15
4.1.1	Errore nella scelta dell'opamp . . . . .	15
4.1.2	Errore nel dimensionamento del filtro attivo . . . . .	15
4.1.3	Sincronizzazione dei threads . . . . .	15
4.2	Commento . . . . .	16
4.3	Ringraziamenti . . . . .	17
4.4	Certificazione . . . . .	17
<b>5</b>	<b>Trasformata di Fourier</b>	<b>18</b>
5.1	Nozioni preliminarie . . . . .	18
5.1.1	Regressione lineare con il metodo dei minimi quadrati . . . . .	18
5.1.2	Funzione armonica . . . . .	18
5.1.3	Proprietà di ortogonalità del seno e del coseno . . . . .	19
5.2	Polinomio Trigonometrico . . . . .	21
5.2.1	Polinomio Trigonometrico . . . . .	21
5.2.2	Polinomio Trigonometrico Reale . . . . .	21
5.3	Serie di Fourier . . . . .	22
5.3.1	Convergenza della serie . . . . .	22
5.4	Trasformata di Fourier discreta . . . . .	23
5.4.1	Derivazione della DFT . . . . .	23

5.5	Trasformata di Fourier . . . . .	25
5.6	Interpretazione geometrica . . . . .	26
5.6.1	Spazi funzionali . . . . .	26
5.6.2	Prodotto interno . . . . .	26
5.6.3	Metodo dei minimi quadrati . . . . .	26
5.7	Fast Fourier Transform . . . . .	26
5.7.1	Motivazioni e Complessità temporale . . . . .	26
5.7.2	Proprietà dei numeri complessi . . . . .	26

# 1 Introduzione

## 1.1 Contesto

Per portare a termine il percorso formativo per un attestato di capacità federale presso la Scuola Arti e Mestieri di Bellinzona è richiesto lo sviluppo individuale di un progetto di produzione di un prodotto. Per interesse personale nella matematica della trasformata di Fourier mi è stato assegnato di sviluppare un analizzatore spettrale.

## 1.2 Requisiti

È richiesto di sviluppare circuito per analizzare lo spettro dei segnali di frequenza fino a 10 kHz. Il dispositivo dovrà avere 3 possibili sorgenti: RCA/Cinch e 2 Audio Jack per un microfono e per una sorgente di audio generica. È inoltre richiesto che il calcolo dei dati dello spettrogramma sia eseguito da un microcontroller della Microchip, collegato a due altri dispositivi quali, un display e ad un computer in RS232, per poter visualizzare lo spettrogramma computato.

## 1.3 Concetti matematici

Il circuito realizzato si appoggia sul concetto matematico di importanza fondamentale, nelle discipline come la fisica e l'elettrotecnica, della *Trasformata di Fourier*. Questa operazione matematica è fondata su un principio dimostrato da Joseph Fourier che asserisce che è possibile rappresentare una qualsiasi funzione periodica, in alcuni casi anche non periodica, con una serie di sinusoidi di frequenze multiple ad una di base. L'operazione di *Trasformata* dunque è uno strumento per osservare le frequenze di queste armoniche, esso trasforma una funzione in funzione del tempo  $f(t)$  in una funzione rispetto alla frequenza o alla pulsazione  $\hat{f}(\omega)$ , che restituisce ad ogni  $\omega$  l'ampiezza e la fase dell'armonica.

Secondariamente, il progetto usufruisce anche di un altro strumento chiamato *Fast Fourier Transform* (FFT) scoperto inizialmente nel 1965 dai matematici J. Cooley e J. Tukey. La FFT è un algoritmo con molte implementazioni che riduce la complessità computazionale della trasformata di Fourier discreta da  $\mathcal{O}(n^2)$  a  $\mathcal{O}(n \log n)$ . Questo è necessario perché le operazioni matematiche da eseguire sono dei prodotti tra numeri complessi, i quali impiegano molto tempo per essere computati.

Tutti i concetti descritti saranno approfonditi nei capitoli seguenti.

## 1.4 Norme di progetto

### 1.4.1 Software

Componente	Software
Version control	Git
Documentazione	L <sup>A</sup> T <sub>E</sub> X
Diario di lavoro	L <sup>A</sup> T <sub>E</sub> X
Pianificazione	MS Excel 2016
L <sup>A</sup> T <sub>E</sub> X engine	X <sub>Y</sub> L <sup>A</sup> T <sub>E</sub> X
ECAD	Altium Designer 2017
Embedded toolchain	Microchip XC, MPLabX
Desktop Toolchain	QtCreator, g++, MinGW

### 1.4.2 Hardware

Per i valori non specificati sono utilizzati i predefiniti del software ECAD.

Regola	Valore	Unità
Number of Layers	2	–
Silkscreen / Overlay	No	–
Minimum trace width	30	mil
Maximum trace width	60	mil
Minimum trace clearance	20	mil
Minimum power rail width	50	mil

### 1.4.3 Programmazione

Regole per programmazione embedded	
Paradigma	Imperativo sequenziale
Convenzione per i nomi	snake_case, sempre minuscolo
Tabulatore	4 spazi
Tabulato con gli spazi	Sì
Regole per programmazione desktop	
Paradigma	Imperativo ad oggetti (OOP)
Convenzione per i nomi	Convenzioni di Qt
Tabulatore	4 spazi
Tabulato con gli spazi	Sì

## 2 Hardware

### 2.1 Schema a blocchi

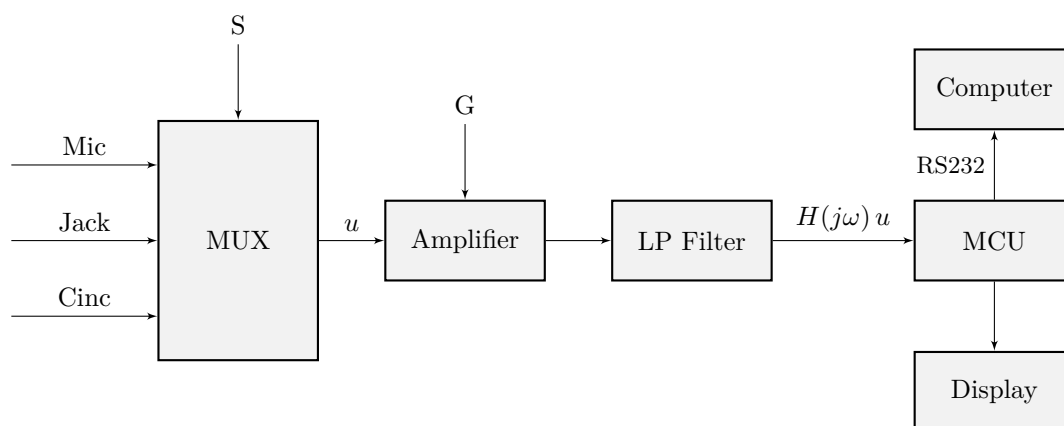


Figura 2.1: Schema a blocchi

### 2.2 Selezione delle entrate

Essendo richiesta dai requisiti la possibilità di selezione tra 3 entrate, è stato utilizzando un semplice multiplexer controllato direttamente dal microcontroller. Per la sua semplicità non sono necessari particolari osservazioni.

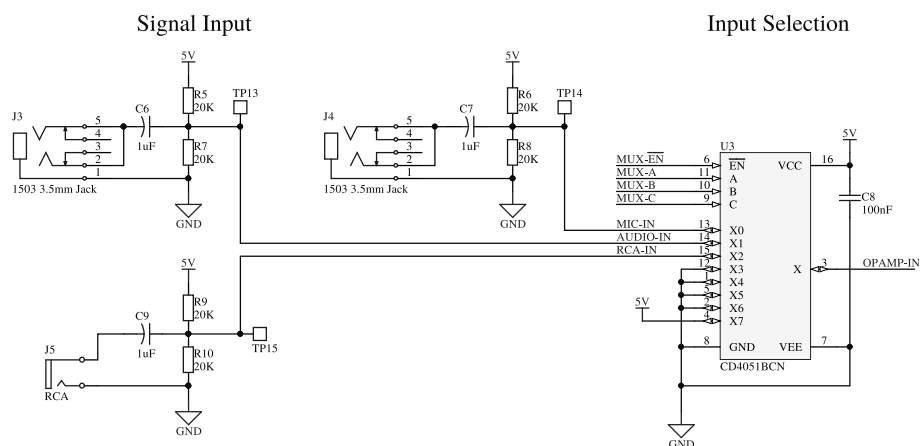


Figura 2.2: Circuito di selezione delle entrate

Tutte le entrate dispongono di un condensatore di disaccoppiamento seguito da un partitore di tensione simmetrico per aggiungere un offset pari a metà dell'alimentazione. Il valore delle resistenze di 20 k $\Omega$  è scelto per avere un'impedenza rispetto al connettore uguale all'impedenza caratteristica dei cavi audio di 10 k $\Omega$ .

## 2.3 Circuito di entrata

Il segnale di cui si analizza lo spettro, prima di essere campionato, viene adattato mediante un circuito di amplificazione e filtraggio. Esso è necessario per due ragioni. Il circuito di amplificazione è presente per poter regolare il circuito nel caso in cui si dovesse avere in entrata un segnale di ampiezza molto piccola. Il secondo circuito invece, di filtraggio, è necessario per rimuovere disturbi di alta frequenza che potrebbero introdurre disturbi nel campionamento. Questa è una tipica configurazione prima di un circuito di conversione AD (analogico - digitale), ed è conosciuto anche come circuito di filtraggio *anti-alias*.

Signal Adapter

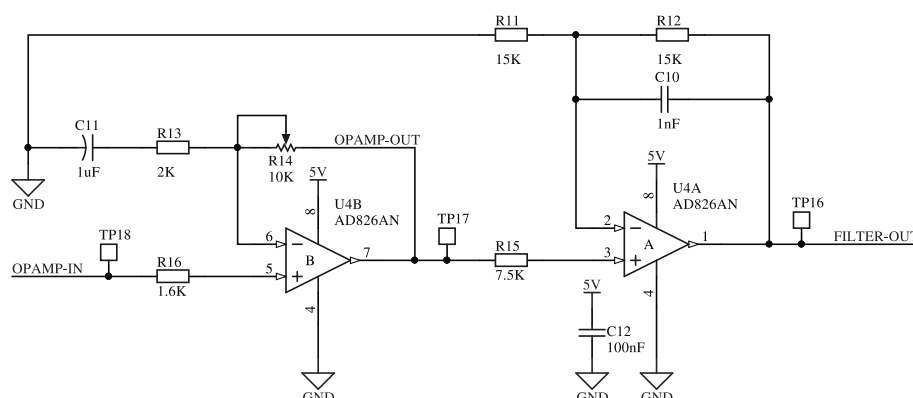


Figura 2.3: Circuito di adattamento del segnale in entrata.

È importante notare che per questa applicazione si è scelto utilizzare degli opamp *rail to rail*<sup>1</sup>, che hanno una tensione di saturazione vicina a quella di alimentazione. Essi sono necessari per poter raggiungere tensioni vicino allo 0 V, che non sarebbero possibili con un opamp normale siccome l'alimentazione del circuito è asimmetrica tra 0 e 12 V.

**Amplificatore.** Come si può notare a sinistra nella figura 2.3, il circuito di amplificazione non ha una configurazione tipica. Esso è basato su una configurazione non invertente ma dispone di un condensatore (C11) che modifica la retroazione in modo da reagire unicamente alla componente AC del segnale. Questo permette di amplificare la componente alternata ignorando l'offset del segnale, perciò di *non* utilizzare un'alimentazione simmetrica  $\pm 5$  V.

L'amplificazione di questo amplificatore è comunque data dal rapporto  $1 + R_{14}/R_{13}$  che permette un un guadagno fino a 6 oppure 15,5 dB.

**Filtro.** A destra della figura 2.3 vi è il circuito di filtraggio, realizzato utilizzando un tipico filtro passa basso attivo di primo ordine. Esso è dimensionato con una frequenza di taglio di 10 kHz poichè quest'ultimo è il limite di Nyquist, conosciuto anche dal teorema di Shannon, il quale stata che la frequenza di campionamento deve essere almeno doppia della frequenza dell'armonica di frequenza maggiore.

Anche il filtro essendo non invertente ha un rapporto di amplificazione dato da  $1 + R_{12}/R_{11}$ . Nella figura il valore della resistenza  $R_{11}$  è **incorretto** (Vedi sezione 4.1.2). Dopo la correzione ( $R_{11} \geq 750$  k $\Omega$ ) il filtro ha un amplificazione approssimativamente unitaria ( $\approx 1.02$ ).

<sup>1</sup>Vedi sezione 4.1.1



## 2.4 Microcontroller

Per questa applicazione è stato deciso di utilizzare il microcontroller a 8 bit di Microchip PIC18F45K22, principalmente per la sua frequenza di lavoro. Questo PIC senza oscillatori esterni dispone di un clock a 16 MHz che grazie ad un PLL interno può essere aumentata fino a ad un massimo di 64 MHz.

Inoltre questo microcontroller dispone di un moltiplicatore hardware 8x8 che impiega un solo ciclo, risparmiando la difficoltà di dover ottimizzare le computazioni della Fast Fourier Transform.

Un ultima ragione importante per la scelta di questo componente è data dalla disponibilità di una libreria per controllare la matrice LED, utilizzata per la visualizzazione, adattata da Arduino da P. Randjelovic in un LPI precedente.

In allegato è presente una pagina riassuntiva dal datasheet.

## 2.5 Visualizzazione

## 3 Software

### 3.1 Campionamento

Per campionare il segnale è stato scelto di utilizzare il TIMER2, sia per la sua semplicità che per la granularità offerta dal registro di comparazione. Il campionamento è eseguito ad una frequenza di 20 kHz, poco sotto al valore massimo possibile che si può ottenere considerando il tempo di conversione dell'ADC.

**FIGURE 13-1: TIMER2/4/6 BLOCK DIAGRAM**

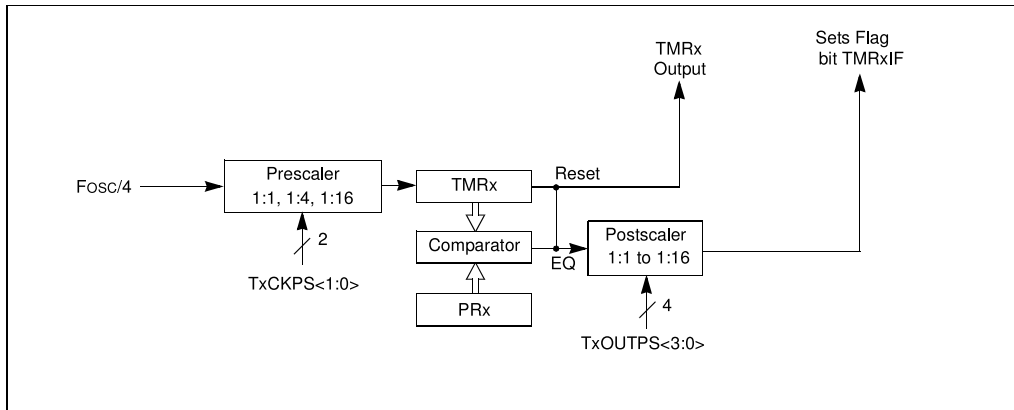


Figura 3.1: Schema a blocchi del TIMER2. Fonte: Microchip PIC18F2X/4XK22 datasheet

Dallo schema a blocchi nella figura 3.1, si può osservare che la frequenza degli interrupt, ossia di campionamento, è data dalla seguente relazione.

$$f = \frac{F_{osc}}{4} \cdot \frac{1}{\text{prescaler}} \cdot \frac{1}{\text{comparator}} \cdot \frac{1}{\text{postscaler}}$$

Per il questo progetto il PIC18F45K22 è configurato con un postscaler 1:1 ed un prescaler 1:16, per far corrispondere un unità del comparatore ad 1 microsecondo. Perciò il comparatore è impostato a 50, poichè  $1/50 \mu s = 20 \text{ kHz}$ .

$$f = \frac{64 \text{ MHz}}{4} \cdot \frac{1}{1} \cdot \frac{1}{\text{comparator}} \cdot \frac{1}{16} = \frac{1 \text{ MHz}}{\text{comparator}}$$

### 3.2 Trasferimento dei dati

Per trasferire i dati campionati era inizialmente stato scelto di utilizzare una struttura dati binaria. In seguito però si è deciso di utilizzare un formato interamente ASCII per semplificare il debugging.

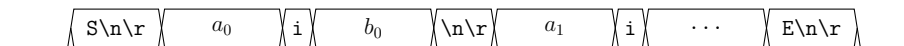


Figura 3.2: Protocollo di trasmissione dei dati, il primo dato mandato è  $a_0 + ib_0$

I dati sono trasferiti con un semplice protocollo illustrato nella figura 3.2. Un frame di dati incomincia con il carattere ASCII maiuscolo 'S' seguito da un line break (`\n`). Ogni riga seguente è un numero complesso a 4 cifre con il carattere 'i' come separatore tra la parte immaginaria e complessa (Es: 0140i0670). Il frame termina con il carattere ASCII maiuscolo 'E'.

## 3.3 Interfaccia al Computer

Per preferenze principalmente personali è stato scelto di realizzare l'interfaccia al computer utilizzando il moderno linguaggio di programmazione C++ (versione  $\geq 11$ ) utilizzando un framework (Qt) che sarà descritto successivamente. I vantaggi dati dall'utilizzo del C++ anziché linguaggi interpretati come il Python, linguaggi compilati in bytecode come Java, o con runtime particolari come LabView / CVI sono molteplici. Innanzitutto tutti gli strumenti necessari per lo sviluppo hanno mezzi o varianti *open source / libre*, di conseguenza gratuiti e in molti casi multiplatforma. Al contrario invece dei sistemi proprietari come quelli offerti da National Instruments che sono estremamente costosi e possono essere utilizzati unicamente sulle piattaforme con un supporto ufficiale. Tra i vari linguaggi di programmazione non proprietari il C++ è comunque in posizione di vantaggio siccome è tra i più performanti in quanto non richiede alcun interpreter (bytecode o non) o nessuna runtime, stando quindi alla pari con il linguaggio C guadagnando però i vantaggi dell'astrazione data dalla programmazione ad oggetti.

### 3.3.1 Librerie e codice di terzi

Per ridurre i tempi dedicati alla realizzazione del programma, seppur mantenendo una buona qualità, è stato scelto di utilizzare le seguenti librerie.

- Serial (<http://wjwwood.io/serial/>): Utilizzata come interfaccia multiplatforma per l'accesso di basso livello alla porta seriale del sistema operativo sottostante. Descrizione dal sito:

This is a cross-platform library for interfacing with rs-232 serial like ports written in C++. It provides a modern C++ interface with a workflow designed to look and feel like PySerial, but with the speed and control provided by C++.

- QCustomPlot (<http://qcustomplot.com>): Utilizzata per produrre il grafico all'interno del software, per visualizzare i dati dal microcontroller. Descrizione dal sito:

QCustomPlot is a Qt C++ widget for plotting and data visualization. It has no further dependencies and is well documented. This plotting library focuses on making good looking, publication quality 2D plots, graphs and charts, as well as offering high performance for realtime visualization applications.

### 3.3.2 Qt Framework

La dipendenza principale utilizzata per realizzare la grafica è il framework di Qt. Oggi Qt è un'azienda indipendente che vende un supporto commerciale per lo sviluppo di applicazioni su praticamente ogni piattaforma. Qt è un framework maturo che esiste oramai da 22 anni ed è disponibile sia con una licenza commerciale che con le licenze open source LGPL e GPL.

La toolchain di Qt aggiunge al normale sviluppo un preprocessore speciale chiamato MOC che genera in automatico il codice dalle strutture grafiche realizzate con QtDesigner. Il resto della toolchain è composta da componenti tipici che possono essere intercambiati liberamente poichè Qt supporta gcc/g++, clang, MSVC e MinGW. Per compilare il codice è dunque necessario un compiler qualsiasi di C++ e l'IDE QtCreator, oppure qmake. Poichè questi pacchetti offrono il preprocessore MOC. Per progetti open source entrambi sono offerti gratuitamente dal sito ufficiale [www.qt.io](http://www.qt.io).

### 3.3.3 Compilazione sotto Linux

Il programma è stato realizzato in parte sotto Debian 9.4 Stretch ed in parte sotto Fedora 27. Per entrambi i sistemi sono necessarie le dipendenze per lo sviluppo in Qt5.

Una volta installate le dipendenze dalla cartella di progetto è possibile utilizzare il makefile per compilare le dipendenze e il codice.

```
$ make build-deps    # dependencies
$ make               # spectrum analyzer code
```

Purtroppo la libreria QCustomPlot utilizza un sistema di build molto particolare che richiede molte dipendenze. Perciò in alcuni casi è preferibile scaricare dal seguente link l'ultima versione dei due files `qcustomplot.cpp` e `qcustomplot.hpp` ed immetterli manualmente nella cartella `lib/qcustomplot/`.

<http://qcustomplot.com/index.php/download>

Ed infine si compila con

```
$ make serial        # build only Serial library dep
$ make               # build spectrum analyzer code
```

### 3.3.4 Compilazione manuale sotto Linux

Per compilare manualmente il progetto sono necessari pochi steps grazie a qmake. Come per il caso precedente la libreria QCustomPlot può essere scaricata dal sito.

1. Scaricare le dipendenze.

```
$ git submodule init
$ git submodule update
```

2. Compilare la libreria Serial

```
$ mkdir -p lib/build-serial
$ qmake -makefile -o Makefile -Wall "CONFIG+=release" \
    -o lib/build-serial/ lib/serial
$ make -C lib/build-serial/
```

3. Compilare la libreria QCustomPlot

```
$ cd lib/qcustomplot/src
$ sed -i -e 's/qmake474/qmake/' release.py
$ ./release.py
$ cd ../../ # go back to project root dir
```

4. Compilare il progetto

```
$ mkdir -p build-desktop
$ qmake -makefile -o Makefile -Wall "CONFIG+=release" \
    -o build-desktop/ src-desktop
$ make -C build-desktop
```

### 3.3.5 Compilazione sotto Windows

Per compilare il progetto in Windows è necessario installare QtCreator dal sito ufficiale [www.qt.io](http://www.qt.io).

1. Installare QtCreator, Qt  $\geq$  5.0 (consigliato 5.10.0)
2. Installare MinGW oppure MSVC + Visual Studio (consigliato MinGW)
3. Inizializzare ed aggiornare i submoduli di Git oppure clonare recursivamente il progetto.
4. Scaricare dal sito <http://qcustomplot.com/index.php/download> i documenti `qcustomplot.cpp` `qcustomplot.hpp` della libreria QCustomPlots ed immetterli nella cartella `lib/qcustomplots/`
5. Aprire il progetto `lib/serial/serial.pro` ed impostare la build directory sia per release che per debug in `lib/build-serial`.
6. Compilare la libreria Serial come release.
7. Aprire il progetto `src-desktop/SpectrumAnalyzer.pro` ed impostare la cartella di build sia per release che debug nella cartella `build-desktop/`
8. Compilare il progetto SpectrumAnalyzer come release
9. Controllare che lo script `deploy-desktop.cmd` abbia le variabili `QT_PATH` e `QT_VERSION` che corrispondano con la vostra l'installazione.
10. Eseguire lo script `deploy-desktop.cmd`.

Nella cartella `build-desktop` sarà pronto l'eseguibile con tutte le librerie dinamiche (DLL) necessarie.

### 3.3.6 Architettura

Il programma desktop è programmato per agire ad eventi asincroni dati dalla porta seriale del sistema operativo e dalle interazioni dell'utente. La gestione degli eventi grafici è gestita interamente da Qt, perciò sono stati scritti solamente i metodi che vengono attivati in funzione degli eventi dall'utente.

Per la porta seriale invece il compito della gestione è stato delegato ad una classe `SerialWorker` che ha un rapporto di *composizione* con la classe `MainWindow` in quanto essa esiste unicamente quando esiste `MainWindow`. Nella figura 3.3 è mostrato un diagramma delle sequenze che mostra il flusso dei dati attraverso le componenti del programma.

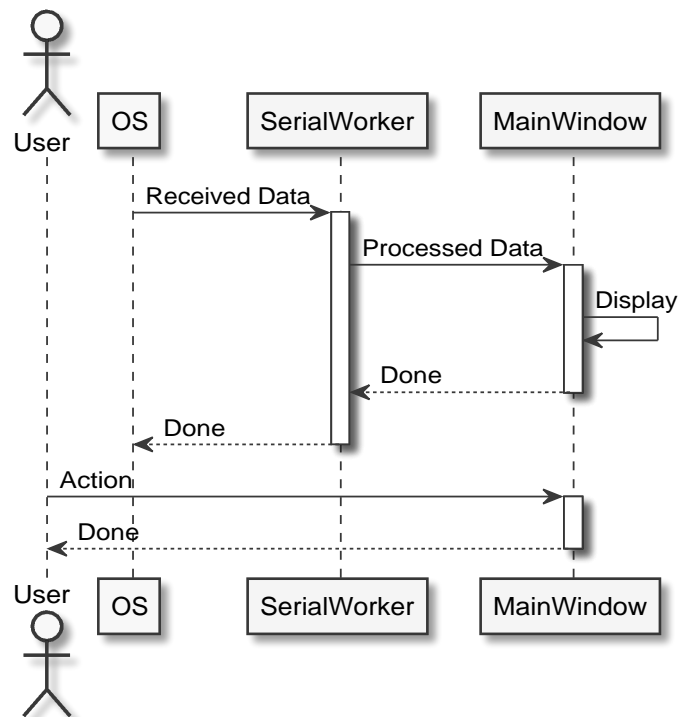


Figura 3.3: Diagramma delle sequenze

Per l'implementazione nella figura 3.4 è possibile osservare il diagramma UML delle classi. È importante notare che Qt introduce dei nuovi tipi di membri chiamati *slots* e *signals*. Gli slots sono dei normali metodi che rispondono ai signals. I signals invece sono delle funzioni prive di implementazione che possono essere *emesse*.

Quando viene realizzato un modello ad oggetti in Qt è possibile collegare degli slots a dei segnali per poter gestire delle azioni asincrone. Nel progetto dello SpectrumAnalyzer il segnale `receivedData` della classe `SerialWorker` viene messo quando sono stati ricevuti dei dati validi dalla seriale. Il segnale ha come argomento un vettore di numeri complessi interi.

Nella classe `MainWindow` il segnale di `receivedData` è associato allo slot `serialDataReceiver` con argomento uguale al segnale, dunque un vettore di numeri complessi interi, che processa i dati e li mostra nell'interfaccia utente.

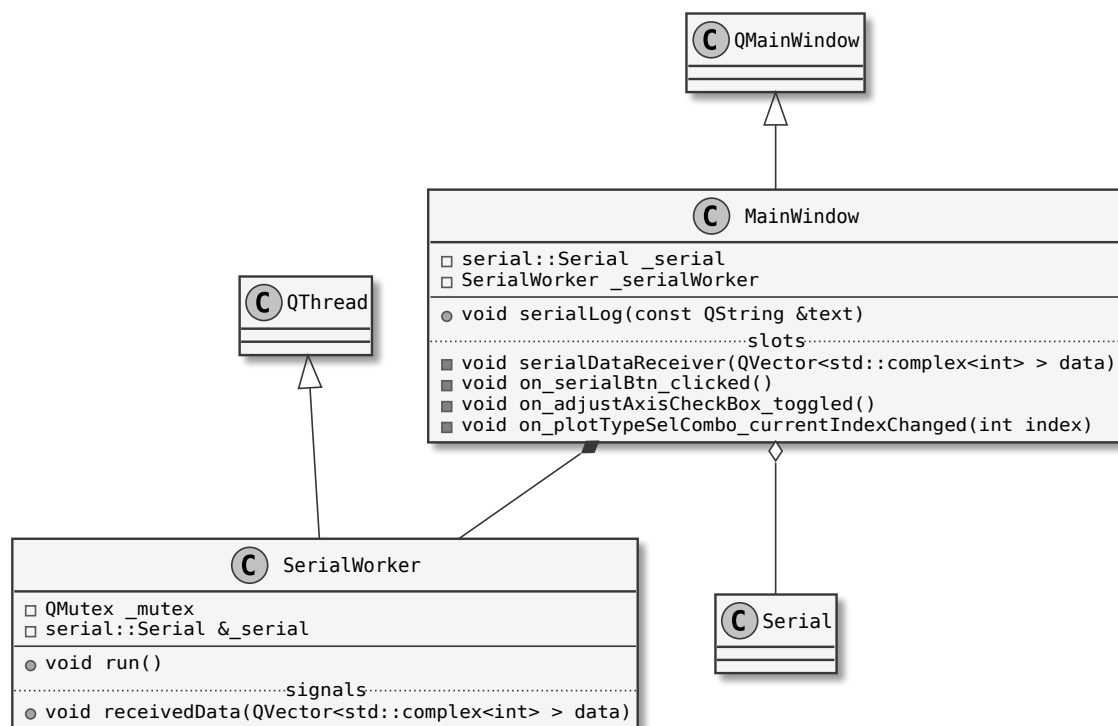


Figura 3.4: Diagramma delle classi

### 3.3.7 Interfaccia utente

L'interfaccia utente, realizzata con Qt è molto semplice e non dovrebbe richiedere spiegazioni.

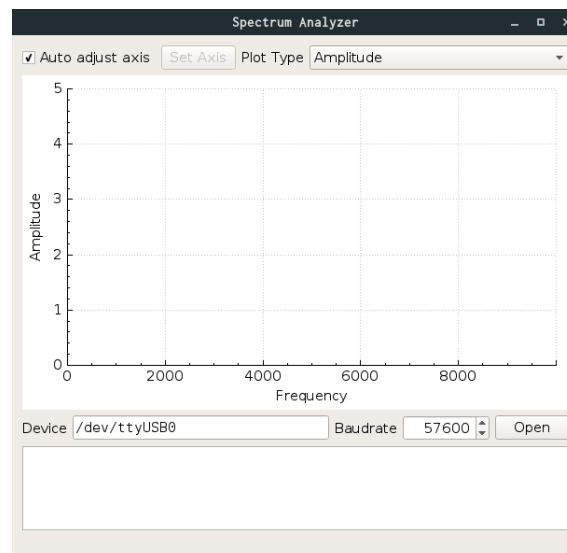


Figura 3.5: L'applicativo sotto Debian 9.4 Stretch

## 3.4 Interfaccia al Display

## 3.5 Fast Fourier Transform

Il codice della FFT è stato preso dal dominio pubblico. Originalmente scritto da Tom Roberts (8.11.1989), successivamente adattato da Malcom Stanley (15.12.1994), Dimitrios P. Buras (14.6.2006) ed infine da Simon Inns (4.1.2011) [1] non ha praticamente necessitato modifiche. L'algoritmo implementato è chiamato "Fixed-Point in-place Fast Fourier"



## 4 Conclusioni

### 4.1 Problemi riscontrati

#### 4.1.1 Errore nella scelta dell'opamp

Durante la fase di test, dopo l'assemblaggio, si è notato che l'amplificatore operazionale AD826 non è un opamp rail to rail come l'AD820, utilizzato durante la fase di sviluppo su piastra sperimentale. Ciò limita l'escursione del segnale amplificato di quasi 1 V, riducendo la precisione del campionamento.

Fortunatamente il pinout DIP8 degli operazionali dual package è standard, perciò non vi sono eccessive difficoltà nella sostituzione del componente. Purtroppo però non sarà possibile acquistare in tempo un opamp sostitutivo.

#### 4.1.2 Errore nel dimensionamento del filtro attivo

Durante la fase di test è stato inoltre rilevato che il filtro attivo (vedi figura 2.3) di anti-alias aveva un'amplificazione non unitaria, dunque incorretta. La tensione di offset di 2.5 V veniva amplificata ed in molti punti l'operazionale entrava in saturazione. Ciò era causato dal valore incorretto di  $R_{11}$  poichè l'amplificazione, data dalla relazione sottostante, era di 2.

$$A_v = 1 + R_{12}/R_{11} = 1 + 15 \text{ k}\Omega/15 \text{ k}\Omega = 2$$

Il valore corretto per il resistore  $R_{11}$  è  $\geq 750 \text{ k}\Omega$ , in modo da ottenere un'amplificazione quasi unitaria, con un errore di +0.02, che corrisponde ad un incremento di 50 mV della tensione di offset di 2.5 V. Valori di ordini di grandezza maggiori sono preferibili, infatti dopo aver rilevato il guasto a  $R_{11}$  è stato assegnato un valore di 910 k $\Omega$ .

$$A_v = 1 + R_{12}/R_{11} = 1 + 15 \text{ k}\Omega/910 \text{ k}\Omega \approx 1.016$$

#### 4.1.3 Sincronizzazione dei threads

Durante lo sviluppo del software desktop è stato riscontrato un unico problema riguardante la sincronizzazione dei threads. La risorsa `serial::Serial MainWindow::_serial` come implica il nome è istanziata nella classe `MainWindow`, ma essa è gestita anche dalla classe `SerialWorker` siccome è suo compito leggere i dati.

Perciò la risorsa deve essere protetta da un `QMutex` e la sua *lifetime* (ciclo di vita) deve essere gestita tenendo in considerazione il thread parallelo. Il bug era causato da una chiusura della risorsa seriale mentre il thread era ancora attivo. Chiudendo la risorsa mentre il thread del `SerialWorker` è attivo, al prossimo tentativo di lettura il metodo `serial::Serial::read()` causa una `serial::IOException` che fa crashare il programma.

La ragione per cui il thread non veniva fermato, era da da utilizzo incorretto dell'API dei `QThread`. Per chiudere correttamente un thread secondo il framework di Qt si richiede un'interruzione con `QThread::requestInterruption()`. Invece nel codice veniva utilizzato `QThread::quit()`, che se non in condizioni particolari non chiude il processo parallelo.

Il diff sottostante mostra il commit in cui il problema viene risolto.

```
1 diff --git a/src-desktop/mainwindow.cpp b/src-desktop/mainwindow.cpp
2 index 73c416e..15c7d77 100644
3 --- a/src-desktop/mainwindow.cpp
4 +++ b/src-desktop/mainwindow.cpp
```

```

5 @@ -42,8 +42,12 @@ void MainWindow::serialDataReceiver(const QString &data)
6 void MainWindow::on_serialBtn_clicked()
7 {
8     if (_serial.isOpen()) {
9 -         _serialWorker.quit();
10 -         while (_serialWorker.isRunning());
11 +         // _serialWorker.quit();
12 +         // while (_serialWorker.isRunning());
13 +         if (_serialWorker.isRunning()) {
14 +             _serialWorker.requestInterruption();
15 +             _serialWorker.wait();
16 +         }
17
18         _serial.close();
19         serialLog("Serial device closed");
20 diff --git a/src-desktop/serialworker.cpp b/src-desktop/serialworker.cpp
21 index 15301cb..f70bce8 100644
22 --- a/src-desktop/serialworker.cpp
23 +++ b/src-desktop/serialworker.cpp
24 @@ -1,6 +1,7 @@
25 #include "serialworker.h"
26
27 #include <QMutexLocker>
28 +#include <string>
29
30 SerialWorker::SerialWorker(serial::Serial &serial) :
31     _mutex(), _serial(serial)
32 @@ -15,12 +16,17 @@ SerialWorker::~SerialWorker()
33
34 void SerialWorker::run()
35 {
36 -     while (isRunning()) {
37 +     while (!isInterruptionRequested()) {
38         QMutexLocker locker(&_mutex);
39 -         while (!_serial.available() && isRunning());
40 +         while (!_serial.available() && !isInterruptionRequested());
41
42 -         QString data = QString::fromStdString(_serial.readline());
43 -         emit receivedData(data);
44 +         for (std::string line : _serial.readlines()) {
45 +             QString data = QString::fromStdString(line);
46 +             emit receivedData(data);
47 +         }
48 +
49 +         // QString data = QString::fromStdString(_serial.readline());
50 +         // emit receivedData(data);
51         _serial.flushOutput();
52     }
53 }

```

## 4.2 Commento

Personalmente ho trovato il progetto molto interessante e coinvolgente. Malgrado la complessità dell'argomento trattato, grazie al supporto di docenti ed amici, sono riuscito ad avere una comprensione tutto sommato abbastanza completa del funzionamento del principio matematico dell'analisi spettrale.

## 4.3 Ringraziamenti

Vorrei ringraziare Eduardo Cima: professore di elettrotecnica alla SAM e Raffaele Ancarola: studente di fisica del primo anno al Politecnico Federale di Losanna (EPFL), per il grande supporto attraverso spiegazioni e chiarimenti degli strumenti matematici della trasformata di Fourier; ed infine vorrei ringraziare anche il professor Emidio Planamente per l'aiuto a risolvere il bug di sincronizzazione (4.1.3).

## 4.4 Certificazione

Il sottoscritto dichiara di aver redatto e prodotto individualmente il lavoro di produzione.

Data: \_\_\_\_\_

Firma: \_\_\_\_\_  
Naoki Pross

# 5 Trasformata di Fourier

## 5.1 Nozioni preliminarie

### 5.1.1 Regressione lineare con il metodo dei minimi quadrati

La regressione lineare è un'approssimazione di una serie di dati ad una funzione lineare. Questa retta di approssimazione può essere calcolata in molteplici modi, per questo progetto è di interesse utilizzare il *metodo dei minimi quadrati*. Sarà dunque spiegato come trovare i coefficienti di una retta a  $m + 1$  termini interpolando  $N$  punti.

$$r(x, a_0, \dots, a_m) = a_0 + x \sum_{i=1}^m a_i \quad (5.1)$$

Consideriamo di avere gli insiemi  $X$  e  $Y$  entrambi con  $N$  termini di cui si prende le coppie ordinate di valori, ossia i punti da interpolare. Il metodo dei minimi quadrati trova i coefficienti della retta minimizzando il quadrato della differenza tra il valore stimato dalla retta  $y = r(x_k)$  e il valore reale  $y_k$ .

$$\min((r(x_k) - y_k)^2) \quad \forall x_k \in X, y_k \in Y$$

Definiamo quindi la funzione da minimizzare  $\varepsilon$

$$\varepsilon(a_0, \dots, a_m) = \sum_{k=1}^N \left[ r(x_k, a_0, \dots, a_m) - y_k \right]^2 \quad (5.2)$$

Da cui si computa le derivati parziali rispetto ai coefficienti ricercati, ottenendo un sistema di equazioni lineare poichè si cerca per ogni derivata quando essa equivale a 0. Ciò corrisponde anche ad affermare che il *gradiente* di  $\varepsilon$  è un vettore  $\in \mathbb{R}^{m+1}$  con tutte le componenti a 0.

$$\nabla \varepsilon = (0, \dots, 0)$$

A questo punto si può procedere risolvendo il sistema con l'algebra lineare definendo la matrice di trasformazione  $\mathbf{A}$  e il vettore dei termini noti  $\vec{u}$

$$\nabla \varepsilon = \mathbf{A} \langle a_0, \dots, a_m \rangle + \vec{u} \iff \langle a_0, \dots, a_m \rangle = \mathbf{A}^{-1}(-\vec{u})$$

### 5.1.2 Funzione armonica

Una funzione armonica, sinusoidale, può essere descritta in molteplici modi.

$$\begin{aligned} f(x) &= a \cdot \sin(\omega x + \varphi) \\ f(x) &= b \cdot \cos(\omega x + \vartheta) \end{aligned}$$

Conoscendo la formula di Eulero

$$e^{i\varphi} = \cos(\varphi) + i \cdot \sin(\varphi)$$

possiamo riscrivere  $f(x)$  utilizzando la forma complessa del seno e del coseno.

$$\begin{aligned} f(x) &= \frac{a}{2i} \cdot (e^{i(x\omega+\varphi)} - e^{-i(x\omega+\varphi)}) \\ f(x) &= \frac{b}{2} \cdot (e^{i(x\omega+\vartheta)} + e^{-i(x\omega+\vartheta)}) \end{aligned}$$

### 5.1.3 Proprietà di ortogonalità del seno e del coseno

Per avere delle fondamenta solide prima dell'introduzione dell'argomento principale, saranno dimostrate le proprietà di ortogonalità del seno e coseno considerando il periodo  $T$  in uno spazio euclideo.

#### Intuizione geometrica

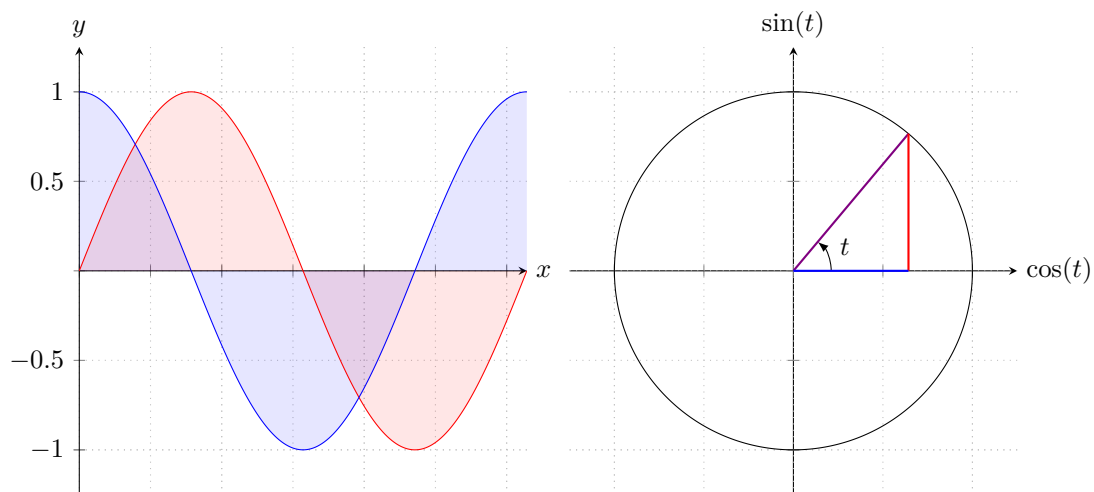


Figura 5.1: Funzioni sin e cos

Si può osservare intuitivamente dal cerchio unitario nella figura 5.1 che le funzioni seno e coseno sono ortogonali tra loro. Dal grafico a sinistra possiamo inoltre osservare che l'area (integrale) di un periodo è sempre nulla.

#### Dimostrazioni algebriche

1. Area di sin in un periodo.

$$\int_0^T \sin\left(\frac{m2\pi x}{T}\right) dx = 0 \quad \forall m \in \mathbb{Z}$$

$$\begin{aligned} \int_0^T \sin\left(\frac{m2\pi x}{T}\right) dx &= \left[ -\frac{T}{2\pi m} \cdot \cos\left(\frac{2\pi}{T}mx\right) \right]_0^T \\ &= -\frac{T}{2\pi m} \cdot \cos(2\pi m) + \frac{T}{2\pi m} \cdot \cos(0) \\ &= 0 \end{aligned}$$

2. Area di cos in un periodo.

$$\int_0^T \cos\left(\frac{m2\pi x}{T}\right) dx = 0 \quad \forall m \in \mathbb{Z}^*$$

$$\begin{aligned} \int_0^T \cos\left(\frac{m2\pi x}{T}\right) dx &= \left[ \frac{T}{2\pi m} \cdot \sin\left(\frac{2\pi}{T} mx\right) \right]_0^T \\ &= \frac{T}{2\pi m} \cdot \sin(2\pi m) + \frac{T}{2\pi m} \cdot \sin(0) \\ &= \begin{cases} 0 & \iff m \neq 0 \\ T & \iff m = 0 \end{cases} \end{aligned}$$

3. Prodotto tra sin e cos.

$$\int_0^T \sin\left(\frac{m2\pi x}{T}\right) \cos\left(\frac{n2\pi x}{T}\right) dx = 0 \quad \forall m, n \in \mathbb{Z}$$

$$\begin{aligned} \int_0^T \sin\left(\frac{m2\pi x}{T}\right) \cos\left(\frac{n2\pi x}{T}\right) dx &= \underbrace{\frac{1}{2} \int_0^T \sin\left[\frac{2\pi}{T}(n+m)x\right] dx}_0 - \underbrace{\frac{1}{2} \int_0^T \sin\left[\frac{2\pi}{T}(n-m)x\right] dx}_0 \\ &= 0 \end{aligned}$$

4. Prodotto tra due sin di frequenze diverse.

$$\int_0^T \sin\left(\frac{m2\pi x}{T}\right) \sin\left(\frac{n2\pi x}{T}\right) dx = 0 \quad \forall m, n \in \mathbb{Z} \mid m \neq \pm n$$

$$\begin{aligned} \int_0^T \sin\left(\frac{m2\pi x}{T}\right) \sin\left(\frac{n2\pi x}{T}\right) dx &= \underbrace{\frac{1}{2} \int_0^T \cos\left[\frac{2\pi}{T}(n-m)x\right] dx}_{m-n \neq 0 \implies 0} - \underbrace{\frac{1}{2} \int_0^T \cos\left[\frac{2\pi}{T}(n+m)x\right] dx}_{m+n \neq 0 \implies 0} \\ &= \begin{cases} 0 & \iff n \neq \pm m \\ T/2 & \iff n = m \\ -T/2 & \iff n = -m \end{cases} \end{aligned}$$

5. Prodotto tra due cos di frequenze diverse.

$$\int_0^T \cos\left(\frac{m2\pi x}{T}\right) \cos\left(\frac{n2\pi x}{T}\right) dx = 0 \quad \forall m, n \in \mathbb{Z}^* \mid m \neq \pm n$$

$$\begin{aligned} \int_0^T \cos\left(\frac{m2\pi x}{T}\right) \cos\left(\frac{n2\pi x}{T}\right) dx &= \underbrace{\frac{1}{2} \int_0^T \cos\left[\frac{2\pi}{T}(n+m)x\right] dx}_{m+n \neq 0 \implies 0} + \underbrace{\frac{1}{2} \int_0^T \cos\left[\frac{2\pi}{T}(n-m)x\right] dx}_{m-n \neq 0 \implies 0} \\ &= \begin{cases} 0 & \iff n \neq \pm m \\ T/2 & \iff n = \pm m \end{cases} \end{aligned}$$

6. sin e cos raggruppate nella forma complessa con la formula di Eulero.

$$\int_0^T e^{i2\pi kx/T} dx = 0 \quad \forall k \in \mathbb{Z}^*$$

$$\int_0^T e^{i2\pi kx/T} dx = \underbrace{\int_0^T \cos\left(\frac{2\pi}{T} kx\right) dx}_{k \neq 0 \Rightarrow 0} + i \underbrace{\int_0^T \sin\left(\frac{2\pi}{T} kx\right) dx}_0$$

$$= \begin{cases} 0 & \iff k \neq 0 \\ T & \iff k = 0 \end{cases}$$

## 5.2 Polinomio Trigonometrico

### 5.2.1 Polinomio Trigonometrico

Analogamente a come è definito un polinomio  $P$  “normale”, una somma di termini dal grado 0 fino al grado  $N$ , è possibile definire anche un polinomio trigonometrico  $T$ .

$$P_N(x) = \sum_{n=0}^N a_n x^n \quad a_n \in \mathbb{R}, a_N \neq 0$$

$$T_N(x) = \sum_{n=0}^N c_n e^{i\omega n x} \quad c_n \in \mathbb{C}, \omega \in \mathbb{R}, c_N \neq 0$$

Questo polinomio è detto *trigonometrico* perchè utilizzando la formula di eulero  $e^{i\varphi} = \cos(\varphi) + i \sin(\varphi)$  si può espandere nel seguente modo.

$$T_N(x) = \sum_{n=0}^N [a_n \cdot \cos(\omega n x) + i b_n \cdot \sin(\omega n x)] \quad a_n, b_n \in \mathbb{C}$$

### 5.2.2 Polinomio Trigonometrico Reale

È definito inoltre il polinomio trigonometrico *reale* come

$$T_N(x) = \sum_{n=0}^N [a_n \cdot \cos(\omega n x) + b_n \cdot \sin(\omega n x)] \quad a_n, b_n \in \mathbb{R}$$

Mediante delle identità trigonometriche può essere riscritto anche nel modo seguente.

$$T_N(x) = \sum_{n=0}^N A_n \cdot \cos(\omega n x - \varphi)$$

Oppure nella sua forma complessa in cui  $c_k = \frac{a_k}{2} + \frac{b_k}{2i}$ .

$$T_N(x) = \sum_{n=0}^N c_n e^{i\omega n x} + \bar{c}_n e^{-i\omega n x} \quad c_n \in \mathbb{C}$$

Definendo  $c_{-n} = \bar{c}_n$  è possibile ridurre la notazione.

$$T_N(x) = \sum_{n=-N}^N c_n e^{i\omega n x}$$

In tutti i casi possiamo osservare che il polinomio trigonometrico è una somma di sinusoidi di frequenze multiple ad una base  $f = \omega/2\pi$ . Se descritto mediante la terminologia dell'algebra lineare, si può anche osservare che un polinomio trigonometrico è una combinazione lineare nello spazio funzionale ortonormato dalle basi  $\sqrt{\frac{2}{T}} \cdot \sin(\omega x)$  e  $\sqrt{\frac{2}{T}} \cos(\omega x)$ .

## 5.3 Serie di Fourier

Un polinomio trigonometrico reale di infiniti termini è una serie di Fourier, in onore al matematico francese J. B. Fourier.

$$S(x) = \sum_{n=-\infty}^{\infty} c_n e^{i\omega n x} = \sum_{n=0}^{\infty} a_n \cdot \cos(\omega n x) + b_n \cdot \sin(\omega n x)$$

### 5.3.1 Convergenza della serie

La convergenza puntuale della serie di Fourier è dimostrabile dalle condizioni del teorema di Dirichlet. Purtroppo questa dimostrazione richiede una grande quantità di nozionistica matematica che non può essere riassunta in pochi paragrafi. In allegato e nelle referenze sono presenti dei documenti che analizzano e dimostrano questa proprietà.

Da questo punto è da dare per assunto che è dimostrata la convergenza puntuale della serie di Fourier di una funzione reale a condizione che:

1.  $|f(x)|$  (valore assoluto) è integrabile in un periodo.
2. La funzione deve essere di variazione limitata, ossia devono esserci un numero finito di minimi e massimi in un qualsiasi intervallo.
3. La funzione deve avere un numero finito di discontinuità in un qualsiasi intervallo chiuso e le discontinuità non devono essere infinite.



## 5.4 Trasformata di Fourier discreta

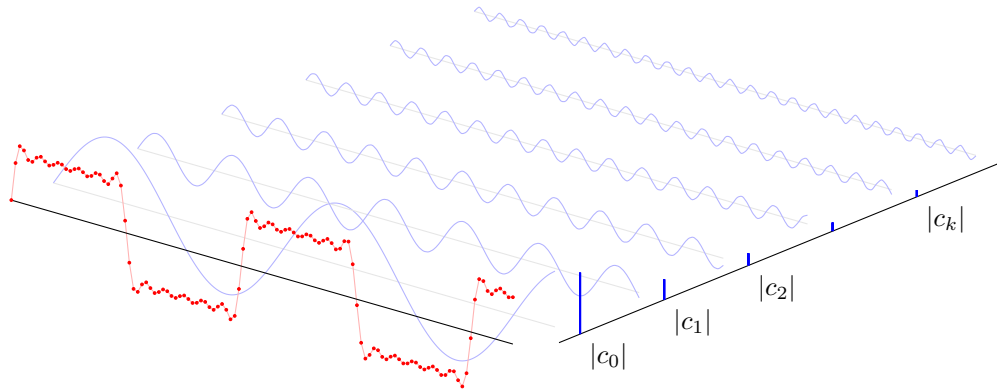


Figura 5.2: Rappresentazione grafica della trasformata di Fourier discreta [8]

La Trasformata di Fourier Discreta (DFT) è l'operazione matematica che permette di trovare i coefficienti della serie di Fourier o di un polinomio trigonometrico, per approssimare al meglio una funzione. Se presa a se stante però essa essendo una *trasformata* può essere osservata anche come una funzione che da uno spettro *discreto* di una funzione. La densità spettrale ottenuta dalla DFT dipende dalla frequenza di base scelta e nel caso di un polinomio trigonometrico, anche dal numero di termini di quest'ultimo.

$$X_k = c_k \cdot T = \int_0^T f(x) \cdot e^{-ik2\pi f x} dx$$

Nella figura 5.2, possiamo osservare sul grafico  $x \perp z$  in rosso la funzione campionata, ed il suo spettro discreto sull'asse  $y \perp z$  in blu. Più lo spettro è denso, ossia più campioni, più è precisa l'approssimazione.

### 5.4.1 Derivazione della DFT

Per trovare la DFT, supponiamo di voler approssimare una funzione, utilizzando il metodo dei minimi quadrati, con un polinomio trigonometrico reale.

$$T_N(x) = \sum_{n=0}^N A_n \cdot \cos(\omega n x - \varphi) \quad \omega = 2\pi f$$

Sappiamo che questo può essere scritto anche nel seguente modo.

$$T_N(x) = \sum_{n=0}^N a_n \cdot \cos(\omega n x) + b_n \cdot \sin(\omega n x)$$

Dunque per trovare i termini  $a_0, \dots, a_N$  e  $b_0, \dots, b_N$  definiamo una funzione  $\varepsilon$  da minimizzare con il metodo dei minimi quadrati.

$$\varepsilon = \int_0^T [T_N(x) - f(x)]^2 dx$$

Per generalizzare sarà dimostrato come trovare il  $k$ -esimo termine.

### Termini dei coseni

I termini dei coseni  $a_k$  sono ottenuti eguagliando la derivata parziale di  $\varepsilon$  a zero.

$$\begin{aligned}
 \frac{\partial \varepsilon}{\partial a_k} = 0 &= \frac{\partial}{\partial a_k} \int_0^T [T_N(x) - f(x)]^2 dx \\
 0 &= \int_0^T \frac{\partial}{\partial a_k} \left[ \sum_{n=0}^N a_n \cos(\omega n x) + b_n \sin(\omega n x) - f(x) \right]^2 dx \\
 0 &= \int_0^T 2 \left[ \sum_{n=0}^N a_n \cos(\omega n x) + b_n \sin(\omega n x) - f(x) \right] \cdot a_k \cos(\omega k x) dx \\
 0 &= \underbrace{\int_0^T \cos(\omega k x) \sum_{n=0}^N a_n \cos(\omega n x) dx}_{k \neq n \Rightarrow 0} + \underbrace{\int_0^T \cos(\omega k x) \sum_{n=0}^N b_n \sin(\omega n x) dx}_0 - \int_0^T \cos(\omega k x) \cdot f(x) dx \\
 0 &= a_k \cdot \frac{T}{2} - \int_0^T \cos(\omega k x) \cdot f(x) dx \\
 \frac{a_k}{2} &= \frac{1}{T} \int_0^T \cos(\omega k x) \cdot f(x) dx
 \end{aligned}$$

### Termini dei seni

I termini dei seni  $b_k$  sono ottenuti eguagliando la derivata parziale di  $\varepsilon$  a zero.

$$\begin{aligned}
 \frac{\partial \varepsilon}{\partial b_k} = 0 &= \frac{\partial}{\partial b_k} \int_0^T [T_N(x) - f(x)]^2 dx \\
 0 &= \int_0^T \frac{\partial}{\partial b_k} \left[ \sum_{n=0}^N a_n \cos(\omega n x) + b_n \sin(\omega n x) - f(x) \right]^2 dx \\
 0 &= \int_0^T 2 \left[ \sum_{n=0}^N a_n \cos(\omega n x) + b_n \sin(\omega n x) - f(x) \right] \cdot b_k \sin(\omega k x) dx \\
 0 &= \underbrace{\int_0^T \sin(\omega k x) \sum_{n=0}^N a_n \cos(\omega n x) dx}_0 + \underbrace{\int_0^T \sin(\omega k x) \sum_{n=0}^N b_n \sin(\omega n x) dx}_{k \neq n \Rightarrow 0} - \int_0^T \sin(\omega k x) \cdot f(x) dx \\
 0 &= b_k \cdot \frac{T}{2} - \int_0^T \sin(\omega k x) \cdot f(x) dx \\
 \frac{b_k}{2} &= \frac{1}{T} \int_0^T \sin(\omega k x) \cdot f(x) dx
 \end{aligned}$$

### Termine complesso

A questo punto è possibile raggruppare i termini  $a_k$  e  $b_k$  in un unico valore complesso  $c_k$ , come descritto nella sezione 5.2.2.

$$\begin{aligned}
 c_k &= \frac{a_k}{2} + \frac{b_k}{2i} \\
 c_k &= \frac{1}{T} \int_0^T \cos(\omega k x) \cdot f(x) \, dx + \frac{1}{Ti} \int_0^T \sin(\omega k x) \cdot f(x) \, dx \\
 c_k &= \frac{1}{T} \int_0^T f(x) \cdot [\cos(\omega k x) - i \sin(\omega k x)] \, dx \\
 c_k &= \frac{1}{T} \int_0^T f(x) \cdot e^{-i\omega k x} \, dx
 \end{aligned}$$

## 5.5 Trasformata di Fourier

Dalla DFT si è sviluppato lo strumento matematico per ottenere lo spettro discreto di una funzione. Il risultato della DFT però è una funzione discontinua con valori spettrali unicamente multipli della base.

La Trasformata di Fourier estende ulteriormente producendo una funzione di spettro *continua*. Un modo intuitivo per ottenere questo requisito, è di far tendere il limite della densità spettrale, data dal periodo, verso l'infinito. Per i seguenti passaggi sarà utilizzata la funzione  $s$  per indicare la funzione reale di cui si osserva lo spettro. Partendo dall'approssimazione data dalla DFT possiamo asserire che

$$\begin{aligned}
 s(x) &\approx \sum_{n=-\infty}^{\infty} \frac{X_n}{T} \cdot e^{i2\pi x n/T} \\
 s(x) &\approx \sum_{n=-\infty}^{\infty} \frac{1}{T} \underbrace{\int_{-T/2}^{T/2} s(x) \cdot e^{-i2\pi x n/T} \, dx}_{X_n = c_n \cdot T} \cdot e^{i2\pi x n/T} \\
 s(x) &= \lim_{T \rightarrow \infty} \sum_{n=-\infty}^{\infty} \frac{1}{T} \int_{-T/2}^{T/2} s(x) \cdot e^{-i2\pi x n/T} \, dx \cdot e^{i2\pi x n/T}
 \end{aligned}$$

A questo punto osserviamo che

$$\lim_{T \rightarrow \infty} \sum_{n=-\infty}^{\infty} \frac{1}{T} = \int_{-\infty}^{\infty} df \qquad \lim_{T \rightarrow \infty} \frac{n}{T} = f \quad (\text{variabile continua})$$

Dunque otteniamo

$$s(x) = \int_{-\infty}^{\infty} df \underbrace{\int_{-\infty}^{\infty} s(x) \cdot e^{-i2\pi f x} \, dx}_{\mathcal{F}\{s\}} \cdot e^{i2\pi f x}$$

Si nota a questo punto che il termine centrale è la trasformata di Fourier ed è una funzione dalla variabile continua  $f$  (frequenza). Si osservi inoltre che sono stati cambiati i limiti di integrazione del periodo. Definendo entrambi i limiti in funzione di  $T$  e facendo tendere  $T$  all'infinito, si ottiene come conseguenza secondaria che la trasformata non è più limitata ad una funzione periodica.

Formalmente dunque la Trasformata di Fourier, tipicamente notata con  $\mathcal{F}$  oppure con l'accento circonflesso  $\hat{f}$  sul nome della funzione, è definita come segue.

$$\mathcal{F}\{f\} = \hat{f}(\omega) = \int_{-\infty}^{\infty} f(x) \cdot e^{-i\omega x} \, dx$$

## **5.6 Interpretazione geometrica**

### **5.6.1 Spazi funzionali**

### **5.6.2 Prodotto interno**

### **5.6.3 Metodo dei minimi quadrati**

## **5.7 Fast Fourier Transform**

### **5.7.1 Motivazioni e Complessità temporale**

### **5.7.2 Proprietà dei numeri complessi**

# Elenco delle figure

2.1	Schema a blocchi . . . . .	5
2.2	Circuito di selezione delle entrate . . . . .	5
2.3	Circuito di adattamento del segnale . . . . .	6
3.1	Schema a blocchi del TIMER2 . . . . .	8
3.2	Protocollo di trasmissione dei dati . . . . .	8
3.3	Diagramma delle sequenze . . . . .	12
3.4	Diagramma delle classi . . . . .	13
5.1	Funzioni sin e cos . . . . .	19
5.2	Rappresentazione grafica della trasformata di Fourier discreta [8] . . . . .	23

# Elenco delle tabelle

# Bibliografia

- [1] REAL TIME AUDIO SPECTRUM ANALYZER. (online) Simon Inns. Jan 8 2011.  
<https://www.waitingforfriday.com/?p=325>  
<http://archive.is/IJeAe> (archived)
- [2] DIVIDE AND COUQUER: FFT. (online, video)  
Erik Demaine, MIT OpenCourseWare. MIT 6.046J Design and Analysis of Algorithms, Spring 2015  
<https://youtu.be/iTMn0Kt18tg>
- [3] BUT WHAT IS THE FOURIER TRANSFORM? A VISUAL INTRODUCTION. (online, video)  
Grant Sanderson. Jan 26, 2018. YouTube.  
<https://youtu.be/spUNpyF58BY>
- [4] THÉORIE ET TRAITEMENT DES SIGNAUX. (Pag. 70 – 72)  
Coulon, Frédéric de (1996). Lausanne: Presses polytechniques et universitaires romandes, 1996  
(Traité d'électricité de l'Ecole polytechnique fédérale de Lausanne; vol. 6)
- [5] ALGÈBRE LINÉAIRE. AIDE-MÉMOIRE, EXERCICES ET APPLICATIONS. (Pag. 64 Moindres carrées)  
Dalang, Robert C. Chaabouni, Amel (2001). Lausanne: Presses polytechniques et universitaires romandes, 2001
- [6] ALGÈBRE LINÉAIRE. (Pag 63 – 65)  
Cairolì, R. (1991). Lausanne: Presses polytechniques et universitaires romandes, 1991
- [7] LINEAR ALGEBRA. AN INTRODUCTORY APPROACH.  
Curtis, Charles W. (2000). New York: Springer, 2000
- [8] EXAMPLE: FOURIER TRANSFORM. (online) Jake. TeX.SE  
<http://www.pgfpplots.net/tikz/examples/fourier-transform/>