
Nome Cognome: **Naoki Pross**

Professione: **Elettronico**

Titolo del progetto: **Spectrum Analyzer**

Azienda: **CPT Bellinzona**
Centro Professionale Tecnico
Viale S. Franscini 25
6500 Bellinzona

Formazione approfondita: **S.2 Sviluppare prototipi**

Formatore: **Rinaldo Geiler, Daniele Kamm**

Data d'inizio: **12.04.2018** Ore a disposizione: **83 UD**

Data fine lavoro: **15.05.2018** Ore effettive: **– UD**

Indice

1	Introduzione	2
1.1	Contesto	2
1.2	Requisiti	2
1.3	Concetti matematici	2
1.4	Norme di progetto	3
2	Hardware	4
2.1	Schema a blocchi	4
2.2	Selezione delle entrate	4
2.3	Circuito di entrata	5
2.4	Microcontroller	6
2.5	Visualizzazione	6
3	Software	7
3.1	Campionamento	7
3.2	Trasferimento dei dati	7
3.3	Interfaccia al Computer	8
3.3.1	Librerie e codice di terzi	8
3.3.2	Qt Framework	8
3.3.3	Compilazione sotto Linux	8
3.3.4	Compilazione manuale sotto Linux	9
3.3.5	Compilazione sotto Windows	10
3.3.6	Architettura	10
3.3.7	Interfaccia utente	12
3.4	Interfaccia al Display	13
3.5	Fast Fourier Transform	13
4	Conclusioni	14
4.1	Risultati	14
4.2	Problemi riscontrati	14
4.2.1	Errore nella scelta dell'opamp	14
4.2.2	Errore nel dimensionamento del filtro attivo	14
4.2.3	Sincronizzazione dei threads	14
4.3	Commento	16
4.4	Ringraziamenti	16
4.5	Certificazione	16
5	Trasformata di Fourier	17
5.1	Nozioni preliminarie	17
5.1.1	Regressione lineare con il metodo dei minimi quadrati	17
5.1.2	Funzione armonica	17
5.1.3	Proprietà di ortogonalità del seno e del coseno	18
5.2	Polinomio Trigonometrico	20
5.2.1	Polinomio Trigonometrico	20
5.2.2	Polinomio Trigonometrico Reale	20
5.3	Serie di Fourier	21
5.3.1	Convergenza della serie	21
5.4	Trasformata di Fourier discreta	22
5.4.1	Derivazione della DFT	22
5.5	Trasformata di Fourier	24
5.6	Interpretazione geometrica	25

5.6.1	Spazi funzionali	25
5.6.2	Prodotto interno	25
5.6.3	Metodo dei minimi quadrati	25
5.7	Fast Fourier Transform	25
5.7.1	Motivazioni e Complessità temporale	25
5.7.2	Proprietà dei numeri complessi	25
Allegati		28
	Pianificazione del progetto	28
	Diario giornaliero	29
	Schema Elettrico (V.1)	33
	Schema Elettrico (V.2)	35
	Design del PCB (V.1)	37

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 ω 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

Tabella 1.1: Norme di progetto: Software

Componente	Software
Version control	Git
Documentazione	L ^A T _E X
Diario di lavoro	L ^A T _E X
Pianificazione	MS Excel 2016
L ^A T _E X engine	X _Y L ^A T _E X
ECAD	Altium Designer 2017
Embedded toolchain	Microchip XC, MPLabX
Desktop Toolchain	QtCreator, g++, MinGW

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

Tabella 1.2: Norme di progetto: Hardware

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

Tabella 1.3: Norme di progetto: 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

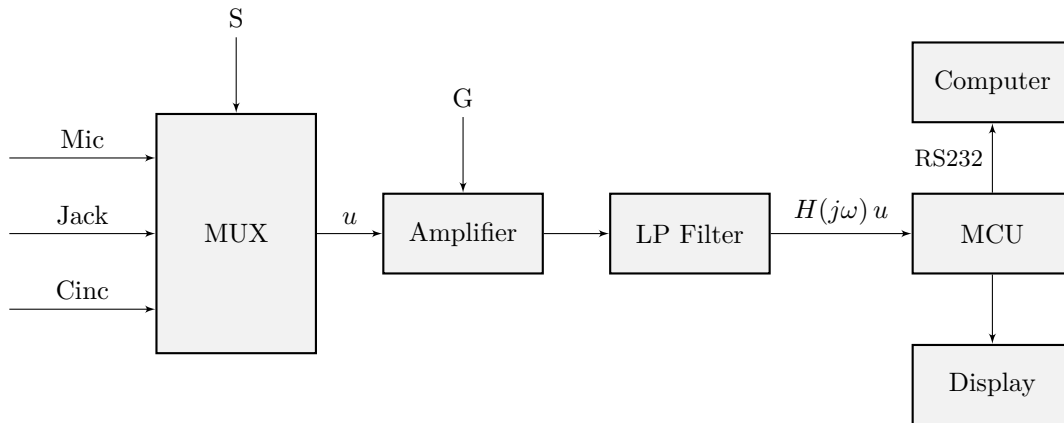


Figura 2.1: Schema a blocchi

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.

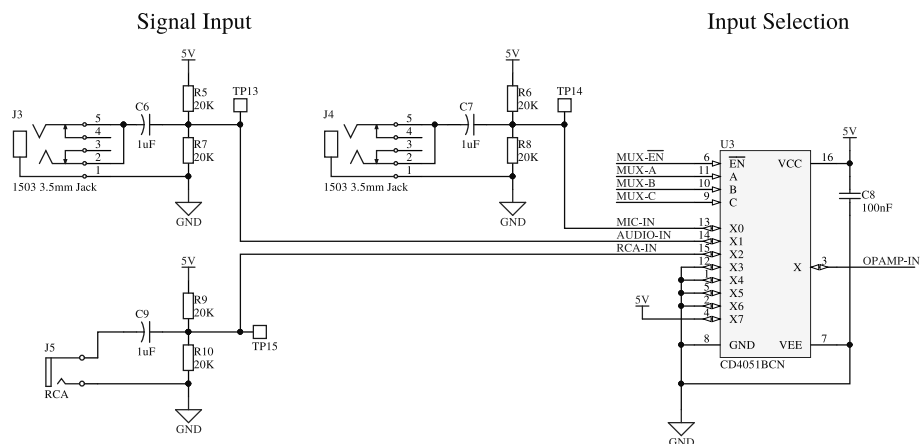


Figura 2.2: Circuito di selezione delle entrate

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 Ω è scelto per avere un'impedenza rispetto al connettore uguale all'impedenza caratteristica dei cavi audio di 10 k Ω .

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

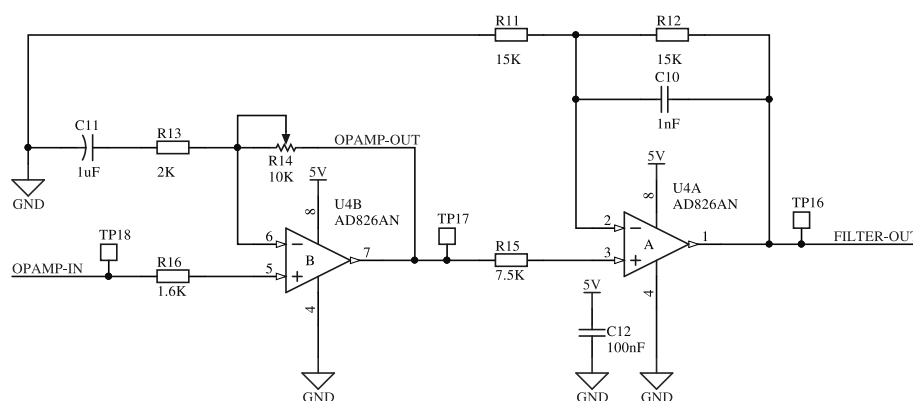


Figura 2.3: Circuito di adattamento del segnale in entrata.

È importante notare che per questa applicazione si è scelto utilizzare degli opamp *rail to rail*¹, 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 ± 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 4.2.2). Dopo la correzione ($R_{11} \geq 750$ k Ω) il filtro ha un amplificazione approssimativamente unitaria (≈ 1.02).

¹Vedi sezione 4.2.1

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

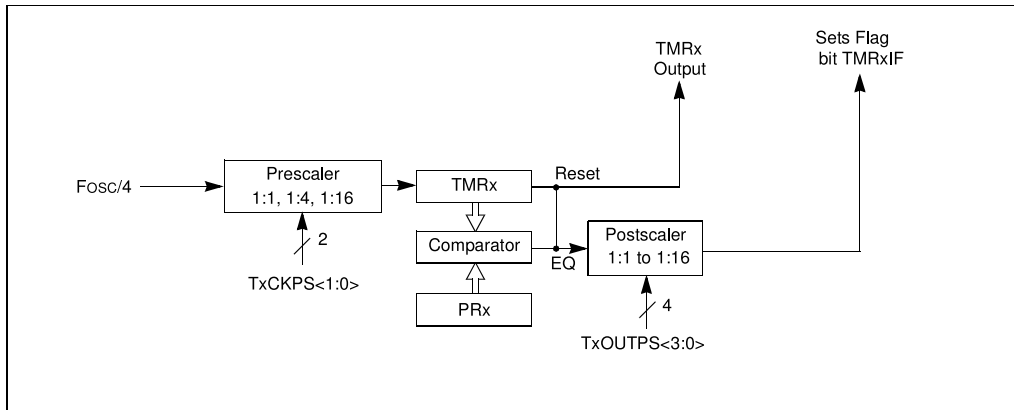


Figura 3.1: Schema a blocchi del TIMER2. Fonte: Microchip PIC18F2X/4XK22 datasheet

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.

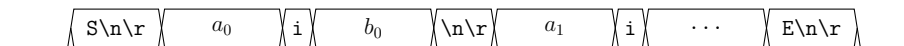


Figura 3.2: Protocollo di trasmissione dei dati, il primo dato mandato è $a_0 + ib_0$

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 ($\backslash n \backslash r$). 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 ≥ 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.

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.

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.

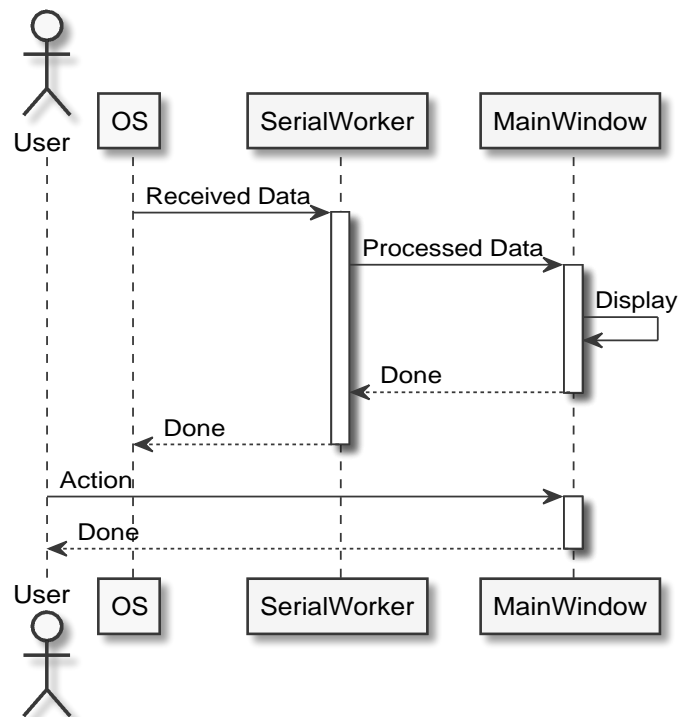


Figura 3.3: Diagramma delle sequenze

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.

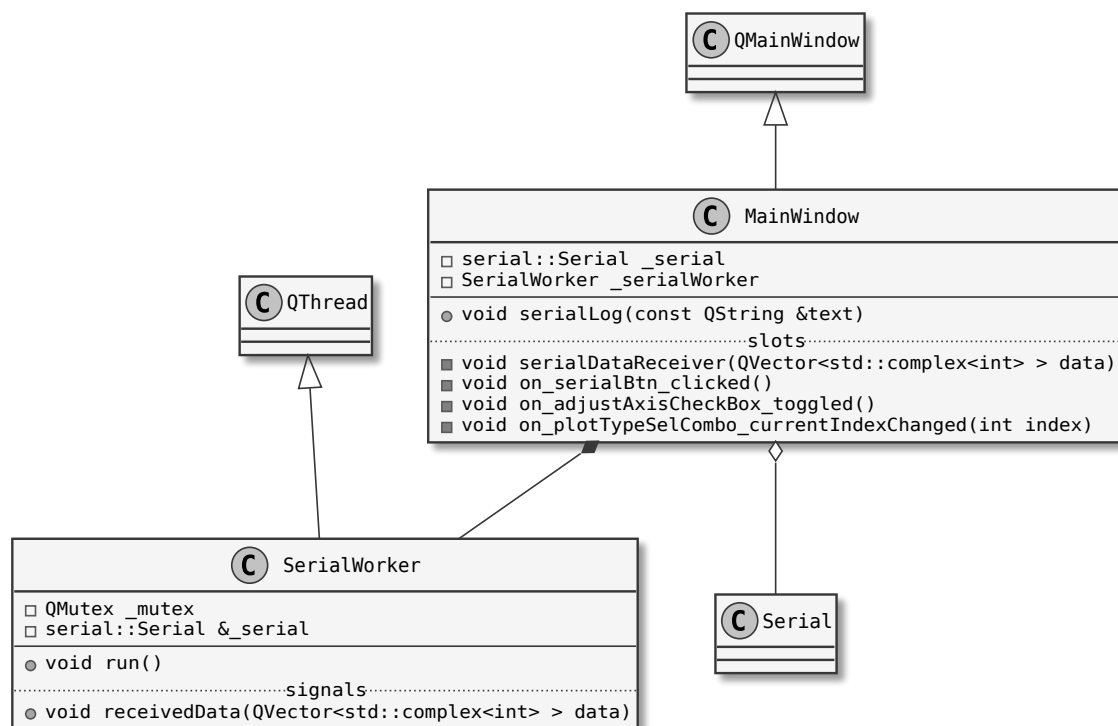


Figura 3.4: Diagramma delle classi

3.3.7 Interfaccia utente

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

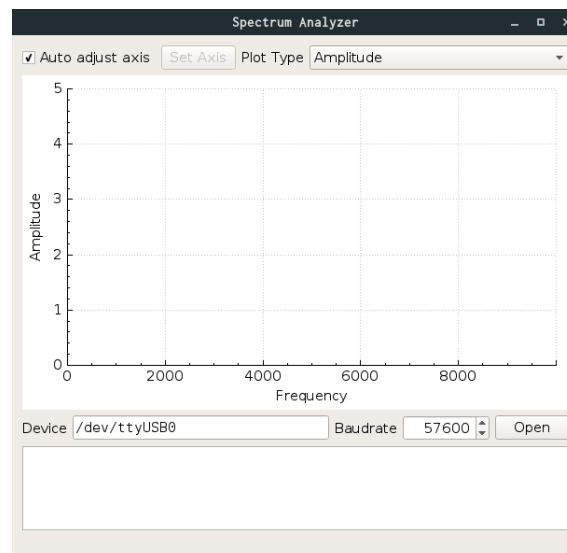


Figura 3.5: L'applicativo sotto Debian 9.4 Stretch

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 Risultati

4.2 Problemi riscontrati

4.2.1 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.

4.2.2 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 Ω .

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

4.2.3 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 }
```

4.3 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.

4.4 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 (4.2.3).

4.5 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 ε

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

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 ε 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

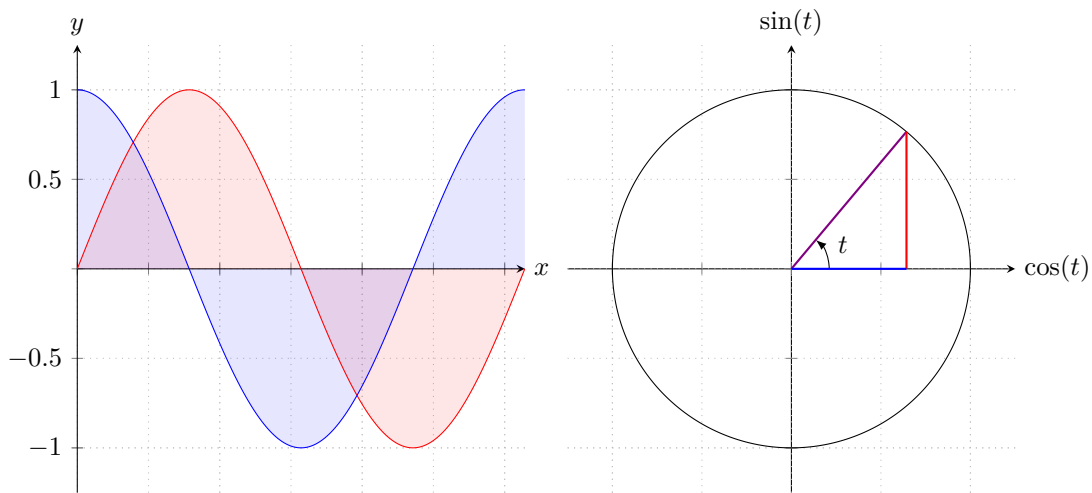


Figura 5.1: Funzioni sin e cos

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

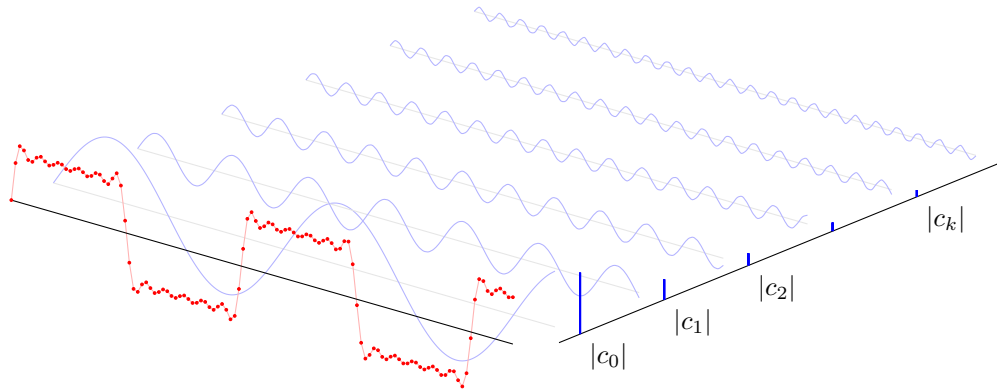


Figura 5.2: Rappresentazione grafica della trasformata di Fourier [8]

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 è dipendente 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 ε 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 ε 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 ε 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.

$$c_k = \frac{a_k}{2} + \frac{b_k}{2i}$$

$$c_k = \frac{1}{T} \int_0^T \cos(\omega k x) \cdot f(x) dx + \frac{1}{Ti} \int_0^T \sin(\omega k x) \cdot f(x) dx$$

$$c_k = \frac{1}{T} \int_0^T f(x) \cdot [\cos(\omega k x) - i \sin(\omega k x)] dx$$

$$c_k = \frac{1}{T} \int_0^T f(x) \cdot e^{-i\omega k x} dx$$

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

$$s(x) \approx \sum_{n=-\infty}^{\infty} \frac{X_n}{T} \cdot e^{i2\pi x n/T}$$

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

$$s(x) = \lim_{T \rightarrow \infty} \sum_{n=-\infty}^{\infty} \frac{1}{T} \int_{-T/2}^{T/2} s(x) \cdot e^{-i2\pi x n/T} dx \cdot e^{i2\pi x n/T}$$

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	4
2.2	Circuito di selezione delle entrate	4
2.3	Circuito di adattamento del segnale	5
3.1	Schema a blocchi del TIMER2	7
3.2	Protocollo di trasmissione dei dati	7
3.3	Diagramma delle sequenze	11
3.4	Diagramma delle classi	12
3.5	L'applicativo sotto Debian 9.4 Stretch	12
5.1	Funzioni sin e cos	18
5.2	Rappresentazione grafica della trasformata di Fourier [8]	22
6.1	Diagramma di flusso del codice del microcontrollore	40

Elenco delle tabelle

1.1	Norme di progetto: Software	3
1.2	Norme di progetto: Hardware	3
1.3	Norme di progetto: Programmazione	3

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.pgplots.net/tikz/examples/fourier-transform/>

Inizio: 12.04.2018

Fine: 15.05.2018

Pianificazione.xlsx

DIARIO GIORNALIERO

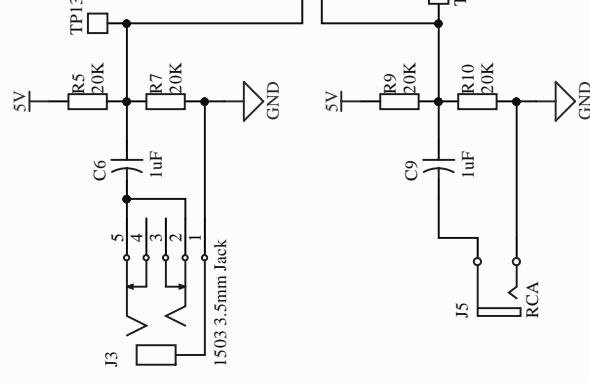
Candidato: Naoki Pross	Progetto: Spectrum Analyzer
Formatore: Rinaldo Geiler, Daniele Kamm	Periodo: 12.04.2018 – 15.04.2018

Giorno	Data	Ore	Descrizione attività (Attività eseguite, metodi adottati, decisioni prese, dimostrazioni effettuate, ecc.)	Osservazioni
Gio	12.04.2018	2	Preparazione della documentazione, della pianifica ed organizzazione generale del progetto.	
Gio	12.04.2018	5	Analisi e studio del concetto matematico.	
Gio	12.04.2018	1	Raccolto informazioni e librerie software per i componenti utilizzati dal progetto. Inoltre è stata preparata una struttura per la documentazione.	
Ve	13.04.2018	2	Scelto i componenti analogici e passivi e preparato una BOM (Bill Of Materials). Progettato uno schema elettrico.	È stato scelto di utilizzare un unico amplificatore di qualità migliore (per audio) con un multiplexer invece di un package con più amplificatori di precisione inferiore.
Ve	13.04.2018	3	Realizzato la parte centrale dello schema elettrico in formato ECAD (Altium Designer). Ossia i circuiti di multiplexing, di adattamento del segnale e di filtraggio delle frequenze indesiderate.	Manca il jack di alimentazione (non ancora necessaria su tavola sperimentale) e il circuito di adattamento di tensione da 12V a 5V.
Ve	13.04.2018	2	Modificato il circuito progettato per utilizzare un filtro attivo anziché passivo dopo aver osservato sperimentalmente l'attenuazione dal filtro passivo.	Il circuito su piastra sperimentale non è ancora funzionante per sviluppare i primi programmi di test.
Ve	13.04.2018	2	Studiato il concetto matematico della Fourier transform con il professor Edoardo Cima.	L'attività non era programmata poiché la disponibilità dei docenti è limitata.
Lu	16.04.2018	1	Realizzato un circuito di prova su piastra sperimentale.	Si è osservato che il circuito progettato non era idoneo. L'amplificatore scelto in precedenza (TL071) non è in grado di lavorare come necessario nel margine da 0V a 5V. Dunque è stato cambiato in un OPAMP Rail-to-Rail AD820 sotto consiglio di D. Kamm.
Lu	16.04.2018	2	Corretto il circuito di amplificazione e risolto imperfezioni minori.	L'amplificatore combinato con il filtro è stato separato in due stadi, di amplificazione e di filtraggio.

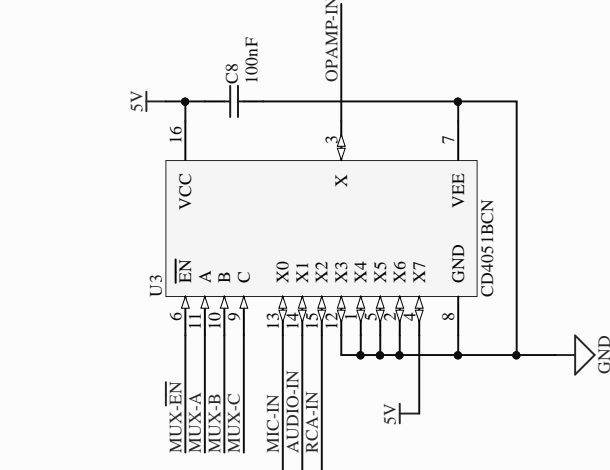
Lu	16.04.2018	1	Corretto il montaggio sulla piastra sperimentale.	Il filtro attivo non è presente sulla tavola perché si deve aspettare la comanda del componente.
Lu	16.04.2018	5	Implementato il codice del microcontroller per configurare l'ADC ed un timer per un campionamento regolare.	Si possono osservare i dettagli su Git nei seguenti commit: 7730a96 , b836638 , b482c7e , aa19054 , 2996f65 .
Lu	16.04.2018	1	Riordinato lo schema elettrico. Aggiunto il circuito di regolazione della tensione in entrata con un MC7805.	Manca ancora il connettore principale dell'alimentazione, rimane da decidere se utilizzare dei morsetti o un power jack.
Ma	17.04.2018	2	Diviso lo schema elettrico su più fogli per rendere il tutto più ordinato. Preparato il footprint del Jack Audio.	
Ma	17.04.2018	1	Analizzato un problema inerente alla programmazione dell'interfaccia software per il computer con il professor Emidio Planamente. È presente un errore nella gestione delle risorse nel thread parallelo di gestione del seriale. Il thread della classe <code>SerialWorker</code> deve essere terminato per rilasciare la risorsa <code>MainWindow::_serial</code> , ma ciò non accade e il programma crasha. Il problema è ancora irrisolto. Riportato lo stato e discusso del progetto con Marco Bertoz (Perito).	Dettagli tecnici: 8ba16b0
Ma	17.04.2018	2	Terminato lo schema elettrico e controllato tutti i footprints. Richiesto una revisione al professor Rinaldo Geiler prima di procedere al PCB.	Se non vi sono errori si potrà iniziare il design del PCB.
Gio	26.04.2018	4	Integrato dei consigli dal feedback da Geiler, ossia correzioni minori e l'aggiunta di un bottone di reset manuale. Iniziato il design del PCB.	Presentato il progetto al capo perito.
Gio	26.04.2018	4	Risolto un il bug dell'interfaccia software desktop con il professor E. Planamente. Il thread di lettura del seriale adesso viene chiuso correttamente. Implementato la rappresentazione grafica dei segnali campionati dal microcontroller ricevuti attraverso la seriale RS232.	Vedi 69d5d42 , 2791cdd
Ve	27.04.2018	5	Terminato il routing del PCB. Risolto i problemi indicati dal DRC.	Il footprint del potenziometro R14 è sbagliato. Il footprint del connettore RCA non può essere controllato poiché il componente non è ancora arrivato. Le correzioni del componente R14 e di eventuali altri saranno eseguite una volta ottenuti tutti i componenti prima della stampa.

Ve	27.04.2018	4	Iniziato ad implementare un protocollo migliore per mandare i dati dal microcontroller al PC. Risolto un bug minore dell'applicativo desktop che causava un malfunzionamento sotto Windows. In dettaglio: la funzione <code>_serial.waitReadable()</code> ; emetteva un <code>IOException</code> causando la chiusura del thread di lettura del seriale.	
Lu	30.04.2018	2	Terminato l'implementazione del protocollo per mandare i dati. È ora possibile mandare numeri complessi interi sia positivi che negativi.	
Lu	30.04.2018	4	Implementato il calcolo della FFT sul microcontroller e la corrispondente visualizzazione sul PC.	Commits: 8adeaa8 , bec4185 , dd19e0d
Lu	30.04.2018	1	Corretto i footprints, preparato i lucidi per la stampa.	
Lu	30.04.2018	2	Cambiato il baudrate della trasmissione a 57.6k e raddoppiato il numero di campioni. Modificato l'implementazione del PC per utilizzare le strutture <code>std::complex</code> invece della mia implementazione <code>sam::complex_int16_t</code> poichè sono standard ed hanno già tutte le operazioni matematiche definite.	Commits: d34ffc6 , 41dae5e
Lu	30.04.2018	1	Continuato la documentazione.	
Ve	11.05.2018	6	Raccolto i componenti necessari ed assemblato la scheda.	Alcuni connettori sono stati saldati in maniera rialzata per poter saldare sul lato superiore, poichè la stampa (realizzata a scuola) non collega i due layers nei fori / vias.
Ve	11.05.2018	2	Allestito test hardware. È stato trovato un errore nel dimensionamento della resistenza R11 del filtro attivo di anti alias. Il circuito aveva un rapporto di amplificazione di 2 anzichè di 1. L'errore di dimensionamento è stato temporaneamente risolto sostituendo R11 con una resistenza da 910k portando il rapporto di amplificazione a circa 1.016.	
Lu	14.05.2018	2	Corretto delle imperfezioni nella stampa, quali piccolo corticircuiti e piste strappate. Saldato il connettore RCA.	
Lu	14.05.2018	2	Implementato il codice per selezionare l'entrata. Iniziato ad analizzare l'implementazione della libreria HT1632 per la matrice LED.	
Lu	14.05.2018	1	Testato la compilazione del software desktop sotto windows 7. Aggiornato lo script di deployment (rilascio, pubblicazione).	

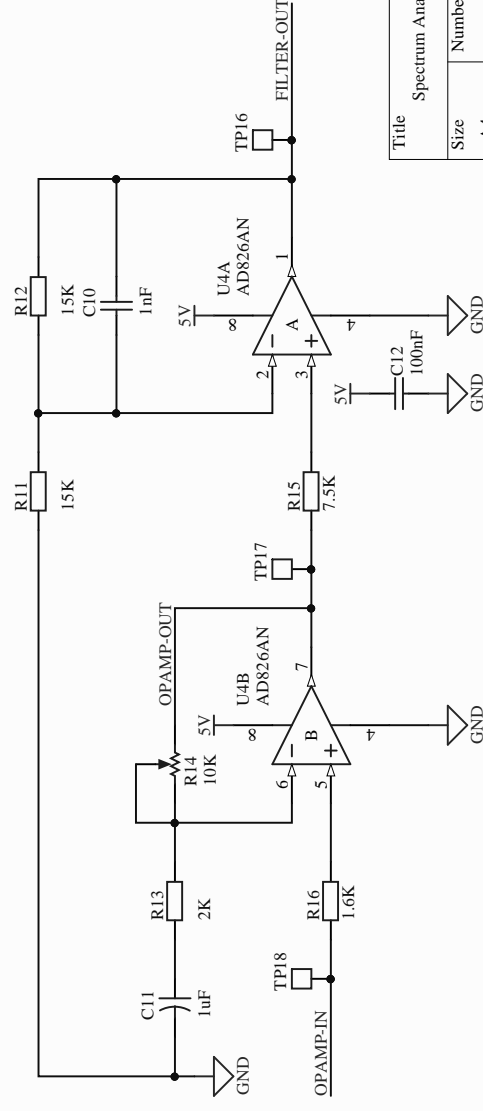
Lu	14.05.2018	2	Apportato le correzioni nello schema elettrico per il prossimo prototipo.	
----	------------	---	---	--



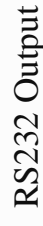
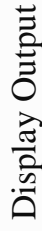
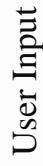
Input Selection



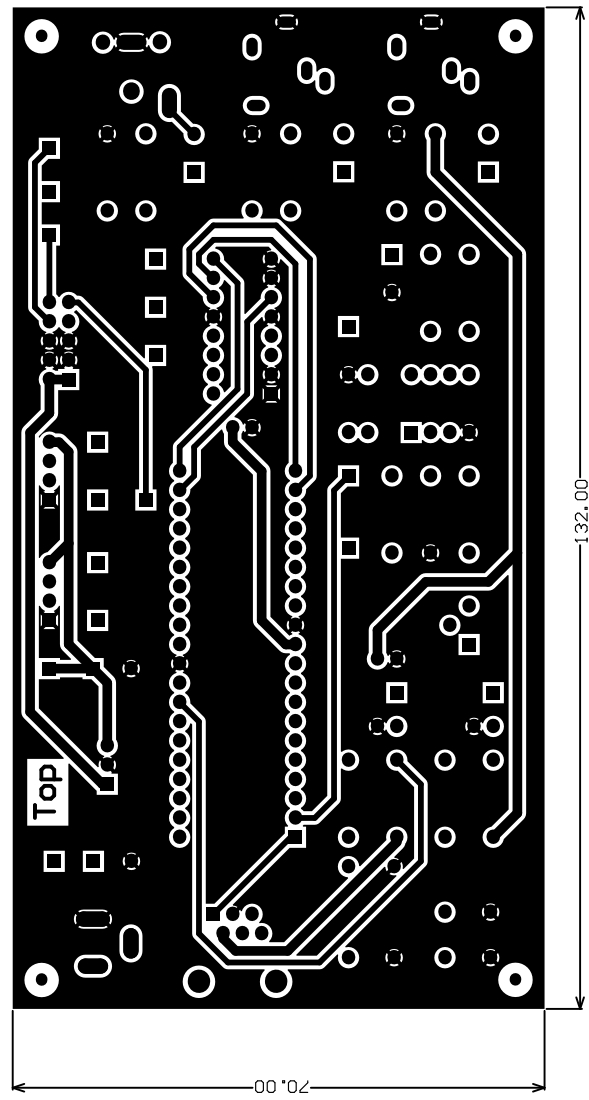
Signal Adapter

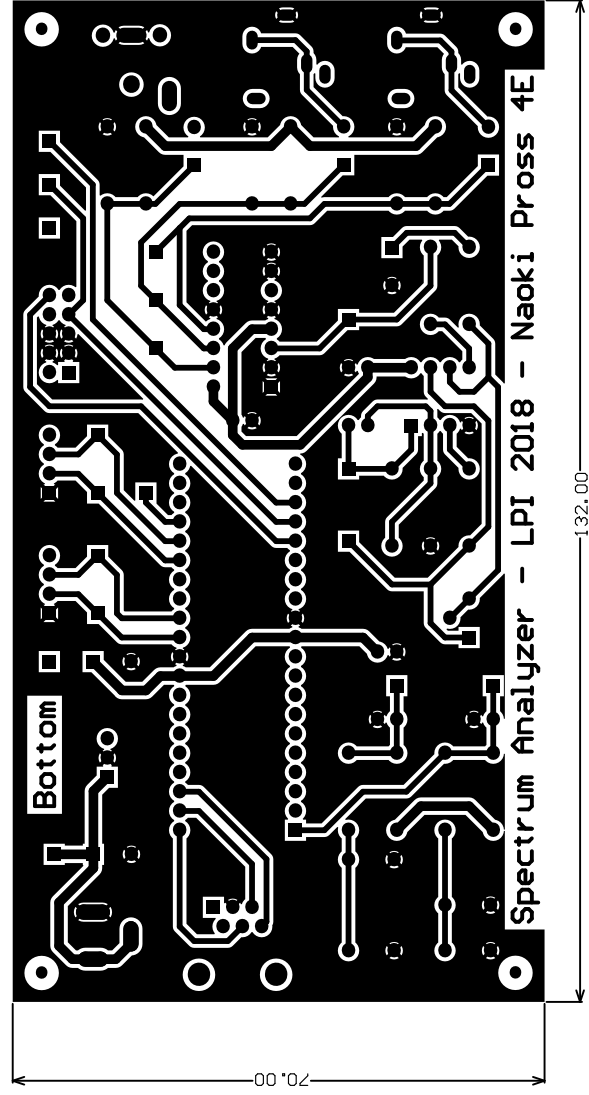


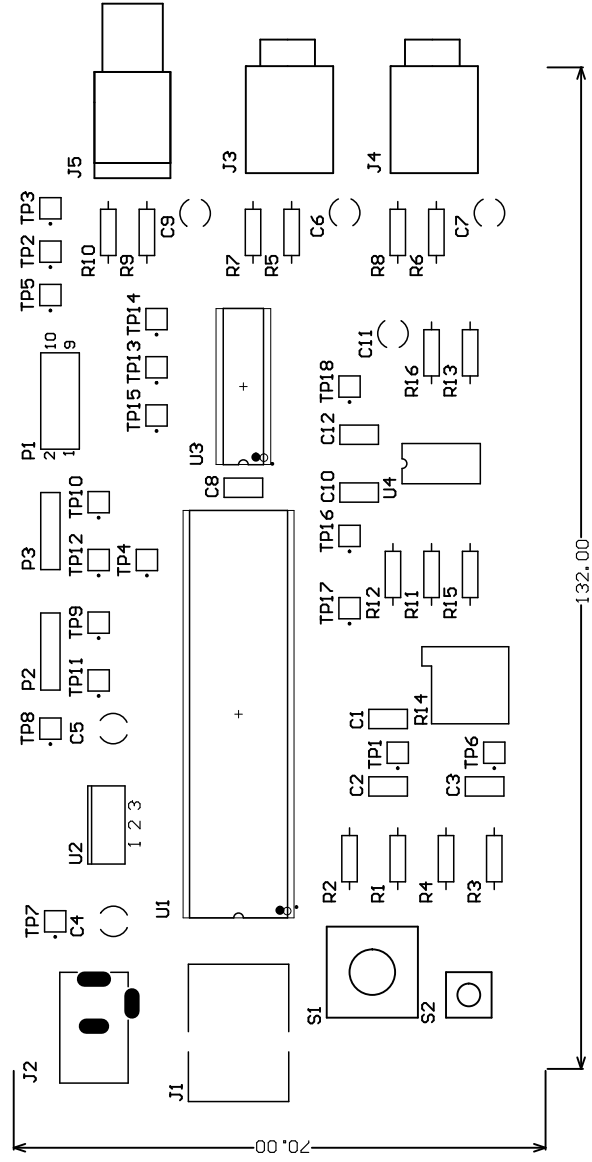
Spectrum Analyzer: Signal Adapter		
Size	Number	Revision
A4		
Date:	27.04.2018	Sheet 2 of 2
File:	Z:\SAMB 4\..SignalAdapter.SchDoc	Drawn By: Naoki Pross



Spectrum Analyzer: Signal Processing		
Size	Number	Revision
A4		V 2.0
Date:	14.05.2018	Sheet 1 of 2
File:	Z:\SAMB 4\...SignalProcessing_SchDoc	Drawn By: Naoki Pross







Codice sorgente

Codice del microcontroller

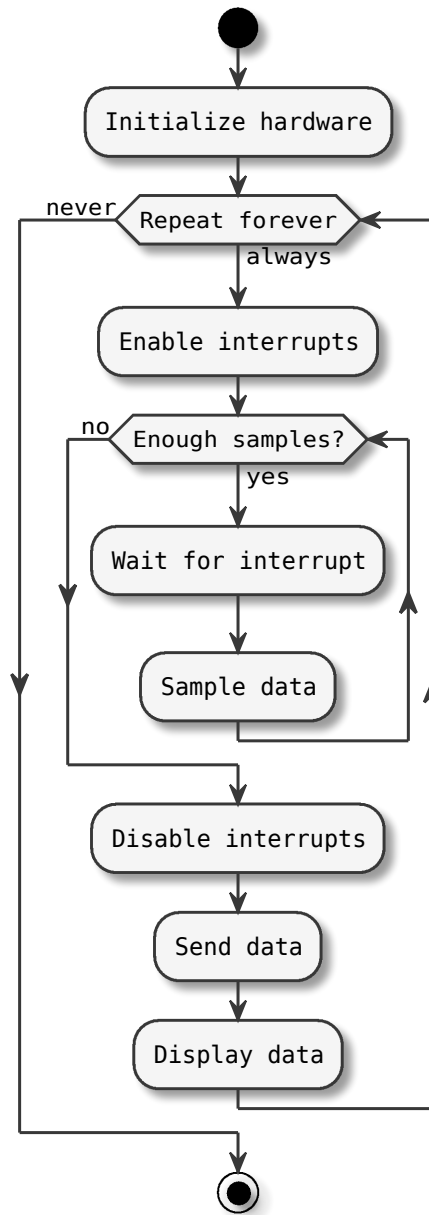


Figura 6.1: Diagramma di flusso del codice del microcontrollore

```
1 /*
2  * File:    main.c
3  * Author:  Naoki Pross
4  * Project: SAM Spectrum Analyzer
5  *
6  * Created on 12.04.2018, 11:57
7  */
8
9 #define DEBUG
10
11 #include "hwconfig.h"
12 #include "rs232.h"
13 #include "ht1632.h"
14
15 #include "fft.h"
16
17 #include <xc.h>
```

```

18 #include <stdio.h>
19 #include <stdbool.h>
20 #include <stdint.h>
21 #include <string.h>
22
23
24 // number of samples to collect
25 #define SAMPLES_SIZE_POW    7
26 #define SAMPLES_SIZE      (1<<SAMPLES_SIZE_POW)
27
28
29
30 volatile uint8_t samples_count;
31 volatile bool start_sample;
32 volatile uint8_t mux_channel = 0;
33
34
35 interrupt void isr(void)
36 {
37     if (INTCONbits.PEIE && PIR1bits.TMR2IF && samples_count < SAMPLES_SIZE) {
38         // start sampling
39         start_sample = true;
40
41         // reset interrupt flag
42         PIR1bits.TMR2IF = 0;
43     }
44
45     if (INTCONbits.INTOIE && INTCONbits.INTOIF) {
46         mux_channel = (mux_channel < 3) ? mux_channel + 1 : 0;
47
48         PORTD &= ~(0x07);
49         PORTD |= mux_channel & 0x07;
50
51         INTCONbits.INTOIF = 0;
52     }
53 }
54 }
55
56
57 /* hardware initialization routine */
58 inline void init_hw()
59 {
60     // disable interrupts
61     di();
62
63     /* PLL / FOSC configuration */
64     // enable PLL
65     OSCTUNEbits.PLEN = 1;
66     // set FOSC to HFINTOSC (max frequency)
67     OSCTUNEbits.TUN = 0b011111;
68     // set 16 MHz oscillator, datasheet p.30
69     OSCCONbits.IRCF = 0b111;
70     // select primary clock (with PLL)
71     OSCCONbits.SCS = 0b00;
72
73     /* initialize digital i/o */
74     // signal input
75     TRISAbits.RAO = 1;
76     ANSELAbits.ANSAO = 1;
77
78     // input selection button
79     TRISBbits.TRISB0 = 1;
80     ANSELBbits.ANSB0 = 0;
81     // interrupt on PBO
82     INTCONbits.RBIE = 1;
83     INTCONbits.RBIF = 0;
84     // interrupt on falling edge
85     INTCON2bits.INTEDG0 = 0;
86

```

```

87     INTCONbits.INTOIE = 1;
88     INTCONbits.INTOIF = 0;
89
90     // eusart and mssp port
91     ANSELC = 0x00;
92     TRISC = 0x00;
93
94     // mux selection port
95     TRISD = 0x00; // 0xF8;
96     ANSELD = 0x00; // 0xF8;
97
98     /* initialize ADC i/o */
99     // set ADC clock period to Fosc/64 = 1us
100    // (maximum without violating Tadmin)
101    ADCON2bits.ADCS = 0b110;
102    // set ADC acquisition time to 2*Tad)
103    ADCON2bits.ACQT = 0b001;
104    // set result format 2 bits on ADREH and 8 bits on ADRESL
105    ADCON2bits.ADFM = 1;
106    // set internal reference
107    ADCON1bits.PVCFG = 0b00;
108    // set internal negative reference
109    ADCON1bits.NVCFG = 0b00;
110    // select channel 0 (ANO)
111    ADCON0bits.CHS = 0b000000;
112    // enable ADC
113    ADCON0bits.ADON = 1;
114
115    /* interrupts initialization */
116    // timer 2 comp value
117    // with post 1:1 and pre 1:16, 1 unit is 1 us
118    PR2 = 50;
119    // postscaler 1:1
120    T2CONbits.T2OUTPS = 0b0000;
121    // prescaler 1:16
122    T2CONbits.T2CKPS = 0b11;
123    // start timer
124    T2CONbits.TMR2ON = 1;
125    // timer 2 interrupts
126    PIE1bits.TMR2IE = 1;
127    PIR1bits.TMR2IF = 0;
128    // enable peripheral interrupts
129    INTCONbits.PEIE = 1;
130
131    /* initialize serial devices */
132    eusart1_init();
133    eusart2_init();
134
135    // enable global interrupts
136    ei();
137 }
138
139
140 void main(void)
141 {
142     unsigned int i;
143
144     short real[SAMPLES_SIZE];
145     short imag[SAMPLES_SIZE];
146
147     // set samples to zero
148     memset(real, 0u, SAMPLES_SIZE * sizeof(real[0]));
149     memset(imag, 0u, SAMPLES_SIZE * sizeof(imag[0]));
150
151
152     // initialize hardware
153     init_hw();
154     PORTB = 0x00;
155     PORTD = 0x00;

```

```

156
157
158     while (true) {
159         // reset samples count
160         samples_count = 0;
161         start_sample = false;
162
163 #ifdef DEBUG
164     PORTDbits.RD4 = 0;
165 #endif
166
167         // sample signal
168         PIE1bits.TMR2IE = 1;
169         while (samples_count < SAMPLES_SIZE){
170             while (!start_sample);
171
172             ADCON0bits.GO = 1;
173             while (ADCON0bits.ndone);
174
175             real[samples_count] = ADRESH<<8 | ADRESL;
176
177             samples_count++;
178             start_sample = false;
179         }
180         PIE1bits.TMR2IE = 0;
181
182 #ifdef DEBUG
183     PORTDbits.RD4 = 1;
184 #endif
185         // compute FFT
186         fix_fft(real, imag, SAMPLES_SIZE_POW);
187
188         // send data
189         printf("S\n\r");
190         for (i = 0; i < SAMPLES_SIZE; i++) {
191             printf("%04di%04d\n\r", real[i], imag[i]);
192         }
193         printf("E\n\r");
194
195
196 #ifdef DEBUG
197     // wait
198     // __delay_ms(500);
199 #endif
200     }
201
202     return;
203 }

1 /*
2  * File:      hwconfig.h
3  * Author:    Naoki Pross
4  * Project:   SAM Spectrum Analyzer
5  *
6  * Created on 16.04.2018
7  */
8
9 #ifndef HWCONFIG_H
10 #define HWCONFIG_H
11
12 // PIC18F44K22 Configuration Bit Settings
13 // 'C' source line config statements
14
15 // CONFIG1H
16 // Oscillator Selection bits (Internal oscillator block)
17 #pragma config FOSC = INTIO67
18 // 4X PLL Enable (Oscillator multiplied by 4)
19 #pragma config PLLCFG = ON
20 // Primary clock enable bit (Primary clock is always enabled)

```

```

21 #pragma config PRICLKEN = ON
22 // Fail-Safe Clock Monitor Enable bit (Fail-Safe Clock Monitor disabled)
23 #pragma config FCMEN = OFF
24 // Internal/External Oscillator Switchover bit (Oscillator Switchover mode disabled)
25 #pragma config IESO = OFF
26
27 // CONFIG2L
28 // Power-up Timer Enable bit (Power up timer disabled)
29 #pragma config PWRTEN = OFF
30 // Brown-out Reset Enable bits (Brown-out Reset enabled in hardware only (SBOREN is disabled
   ))
31 #pragma config BOREN = SBORDIS
32 // Brown Out Reset Voltage bits (VBOR set to 1.90 V nominal)
33 #pragma config BORV = 190
34
35 // CONFIG2H
36 // Watchdog Timer Enable bits (WDT is always enabled. SWDTEN bit has no effect)
37 #pragma config WDTEN = OFF
38 // Watchdog Timer Postscale Select bits (1:32768)
39 #pragma config WDTPS = 32768
40
41 // CONFIG3H
42 // CCP2 MUX bit (CCP2 input/output is multiplexed with RC1)
43 #pragma config CCP2MX = PORTC1
44 // PORTB A/D Enable bit (PORTB<5:0> pins are configured as analog input channels on Reset)
45 #pragma config PBADEN = ON
46 // P3A/CCP3 Mux bit (P3A/CCP3 input/output is multiplexed with RB5)
47 #pragma config CCP3MX = PORTB5
48 // HFINTOSC Fast Start-up (HFINTOSC output and ready status are not delayed by the
   oscillator stable status)
49 #pragma config HFOFST = ON
50 // Timer3 Clock input mux bit (T3CKI is on RC0)
51 #pragma config T3CMX = PORTC0
52 // ECCP2 B output mux bit (P2B is on RD2)
53 #pragma config P2BMX = PORTD2
54 // MCLR Pin Enable bit (MCLR pin enabled, RE3 input pin disabled)
55 #pragma config MCLRE = EXTMCLR
56
57 // CONFIG4L
58 // Stack Full/Underflow Reset Enable bit (Stack full/underflow will cause Reset)
59 #pragma config STVREN = ON
60 // Single-Supply ICSP Enable bit (Single-Supply ICSP enabled if MCLRE is also 1)
61 #pragma config LVP = ON
62 // Extended Instruction Set Enable bit (Instruction set extension and Indexed Addressing
   mode disabled (Legacy mode))
63 #pragma config XINST = OFF
64
65 // CONFIG5L
66 // Code Protection Block 0 (Block 0 (000800-001FFFh) not code-protected)
67 #pragma config CP0 = OFF
68 // Code Protection Block 1 (Block 1 (002000-003FFFh) not code-protected)
69 #pragma config CP1 = OFF
70 // Code Protection Block 2 (Block 2 (004000-005FFFh) not code-protected)
71 #pragma config CP2 = OFF
72 // Code Protection Block 3 (Block 3 (006000-007FFFh) not code-protected)
73 #pragma config CP3 = OFF
74
75 // CONFIG5H
76 // Boot Block Code Protection bit (Boot block (000000-0007FFFh) not code-protected)
77 #pragma config CPB = OFF
78 // Data EEPROM Code Protection bit (Data EEPROM not code-protected)
79 #pragma config CPD = OFF
80
81 // CONFIG6L
82 // Write Protection Block 0 (Block 0 (000800-001FFFh) not write-protected)
83 #pragma config WRT0 = OFF
84 // Write Protection Block 1 (Block 1 (002000-003FFFh) not write-protected)
85 #pragma config WRT1 = OFF
86 // Write Protection Block 2 (Block 2 (004000-005FFFh) not write-protected)

```

```

87 #pragma config WRT2 = OFF
88 // Write Protection Block 3 (Block 3 (006000-007FFFh) not write-protected)
89 #pragma config WRT3 = OFF
90
91 // CONFIG6H
92 // Configuration Register Write Protection bit (Configuration registers (300000-3000FFFh) not
    write-protected)
93 #pragma config WRTC = OFF
94 // Boot Block Write Protection bit (Boot Block (000000-0007FFFh) not write-protected)
95 #pragma config WRTB = OFF
96 // Data EEPROM Write Protection bit (Data EEPROM not write-protected)
97 #pragma config WRTD = OFF
98
99 // CONFIG7L
100 // Table Read Protection Block 0 (Block 0 (000800-001FFFh) not protected from table reads
    executed in other blocks)
101 #pragma config EBTR0 = OFF
102 // Table Read Protection Block 1 (Block 1 (002000-003FFFh) not protected from table reads
    executed in other blocks)
103 #pragma config EBTR1 = OFF
104 // Table Read Protection Block 2 (Block 2 (004000-005FFFh) not protected from table reads
    executed in other blocks)
105 #pragma config EBTR2 = OFF
106 // Table Read Protection Block 3 (Block 3 (006000-007FFFh) not protected from table reads
    executed in other blocks)
107 #pragma config EBTR3 = OFF
108
109 // CONFIG7H
110 // Boot Block Table Read Protection bit (Boot Block (000000-0007FFFh) not protected from
    table reads executed in other blocks)
111 #pragma config EBTRB = OFF
112
113 // #pragma config statements should precede project file includes.
114 // Use project enums instead of #define for ON and OFF.
115
116 #define _XTAL_FREQ 64000000
117
118 #endif /* HWCONFIG_H */

```

Fast Fourier Transform

```

1  /*****
2      fft.h
3
4      FFT Audio Analysis
5      Copyright (C) 2011 Simon Inns
6
7      This program is free software: you can redistribute it and/or modify
8      it under the terms of the GNU General Public License as published by
9      the Free Software Foundation, either version 3 of the License, or
10     (at your option) any later version.
11
12     This program is distributed in the hope that it will be useful,
13     but WITHOUT ANY WARRANTY; without even the implied warranty of
14     MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
15     GNU General Public License for more details.
16
17     You should have received a copy of the GNU General Public License
18     along with this program. If not, see <http://www.gnu.org/licenses/>.
19
20     Email: simon.inns@gmail.com
21
22     *****/
23
24 #ifndef _FFT_H
25 #define _FFT_H
26
27 // Definitions
28 #define N_WAVE      1024      // full length of Sinewave[]

```



```

29 #define LOG2_N_WAVE 10          // log2(N_WAVE)
30
31 // Since we only use 3/4 of N_WAVE, we define only
32 // this many samples, in order to conserve data space.
33 const short sinewave[N_WAVE-N_WAVE/4] = {
34     0,      201,    402,    603,    804,    1005,    1206,    1406,
35     1607,    1808,    2009,    2209,    2410,    2610,    2811,    3011,
36     3211,    3411,    3611,    3811,    4011,    4210,    4409,    4608,
37     4807,    5006,    5205,    5403,    5601,    5799,    5997,    6195,
38     6392,    6589,    6786,    6982,    7179,    7375,    7571,    7766,
39     7961,    8156,    8351,    8545,    8739,    8932,    9126,    9319,
40     9511,    9703,    9895,    10087,    10278,    10469,    10659,    10849,
41     11038,    11227,    11416,    11604,    11792,    11980,    12166,    12353,
42     12539,    12724,    12909,    13094,    13278,    13462,    13645,    13827,
43     14009,    14191,    14372,    14552,    14732,    14911,    15090,    15268,
44     15446,    15623,    15799,    15975,    16150,    16325,    16499,    16672,
45     16845,    17017,    17189,    17360,    17530,    17699,    17868,    18036,
46     18204,    18371,    18537,    18702,    18867,    19031,    19194,    19357,
47     19519,    19680,    19840,    20000,    20159,    20317,    20474,    20631,
48     20787,    20942,    21096,    21249,    21402,    21554,    21705,    21855,
49     22004,    22153,    22301,    22448,    22594,    22739,    22883,    23027,
50     23169,    23311,    23452,    23592,    23731,    23869,    24006,    24143,
51     24278,    24413,    24546,    24679,    24811,    24942,    25072,    25201,
52     25329,    25456,    25582,    25707,    25831,    25954,    26077,    26198,
53     26318,    26437,    26556,    26673,    26789,    26905,    27019,    27132,
54     27244,    27355,    27466,    27575,    27683,    27790,    27896,    28001,
55     28105,    28208,    28309,    28410,    28510,    28608,    28706,    28802,
56     28897,    28992,    29085,    29177,    29268,    29358,    29446,    29534,
57     29621,    29706,    29790,    29873,    29955,    30036,    30116,    30195,
58     30272,    30349,    30424,    30498,    30571,    30643,    30713,    30783,
59     30851,    30918,    30984,    31049,    31113,    31175,    31236,    31297,
60     31356,    31413,    31470,    31525,    31580,    31633,    31684,    31735,
61     31785,    31833,    31880,    31926,    31970,    32014,    32056,    32097,
62     32137,    32176,    32213,    32249,    32284,    32318,    32350,    32382,
63     32412,    32441,    32468,    32495,    32520,    32544,    32567,    32588,
64     32609,    32628,    32646,    32662,    32678,    32692,    32705,    32717,
65     32727,    32736,    32744,    32751,    32757,    32761,    32764,    32766,
66     32767,    32766,    32764,    32761,    32757,    32751,    32744,    32736,
67     32727,    32717,    32705,    32692,    32678,    32662,    32646,    32628,
68     32609,    32588,    32567,    32544,    32520,    32495,    32468,    32441,
69     32412,    32382,    32350,    32318,    32284,    32249,    32213,    32176,
70     32137,    32097,    32056,    32014,    31970,    31926,    31880,    31833,
71     31785,    31735,    31684,    31633,    31580,    31525,    31470,    31413,
72     31356,    31297,    31236,    31175,    31113,    31049,    30984,    30918,
73     30851,    30783,    30713,    30643,    30571,    30498,    30424,    30349,
74     30272,    30195,    30116,    30036,    29955,    29873,    29790,    29706,
75     29621,    29534,    29446,    29358,    29268,    29177,    29085,    28992,
76     28897,    28802,    28706,    28608,    28510,    28410,    28309,    28208,
77     28105,    28001,    27896,    27790,    27683,    27575,    27466,    27355,
78     27244,    27132,    27019,    26905,    26789,    26673,    26556,    26437,
79     26318,    26198,    26077,    25954,    25831,    25707,    25582,    25456,
80     25329,    25201,    25072,    24942,    24811,    24679,    24546,    24413,
81     24278,    24143,    24006,    23869,    23731,    23592,    23452,    23311,
82     23169,    23027,    22883,    22739,    22594,    22448,    22301,    22153,
83     22004,    21855,    21705,    21554,    21402,    21249,    21096,    20942,
84     20787,    20631,    20474,    20317,    20159,    20000,    19840,    19680,
85     19519,    19357,    19194,    19031,    18867,    18702,    18537,    18371,
86     18204,    18036,    17868,    17699,    17530,    17360,    17189,    17017,
87     16845,    16672,    16499,    16325,    16150,    15975,    15799,    15623,
88     15446,    15268,    15090,    14911,    14732,    14552,    14372,    14191,
89     14009,    13827,    13645,    13462,    13278,    13094,    12909,    12724,
90     12539,    12353,    12166,    11980,    11792,    11604,    11416,    11227,
91     11038,    10849,    10659,    10469,    10278,    10087,    9895,    9703,
92     9511,    9319,    9126,    8932,    8739,    8545,    8351,    8156,
93     7961,    7766,    7571,    7375,    7179,    6982,    6786,    6589,
94     6392,    6195,    5997,    5799,    5601,    5403,    5205,    5006,
95     4807,    4608,    4409,    4210,    4011,    3811,    3611,    3411,
96     3211,    3011,    2811,    2610,    2410,    2209,    2009,    1808,
97     1607,    1406,    1206,    1005,    804,    603,    402,    201,

```

```

98     0,    -201,    -402,    -603,    -804,   -1005,   -1206,   -1406,
99   -1607,   -1808,   -2009,   -2209,   -2410,   -2610,   -2811,   -3011,
100  -3211,   -3411,   -3611,   -3811,   -4011,   -4210,   -4409,   -4608,
101  -4807,   -5006,   -5205,   -5403,   -5601,   -5799,   -5997,   -6195,
102  -6392,   -6589,   -6786,   -6982,   -7179,   -7375,   -7571,   -7766,
103  -7961,   -8156,   -8351,   -8545,   -8739,   -8932,   -9126,   -9319,
104  -9511,   -9703,   -9895,  -10087,  -10278,  -10469,  -10659,  -10849,
105 -11038,  -11227,  -11416,  -11604,  -11792,  -11980,  -12166,  -12353,
106 -12539,  -12724,  -12909,  -13094,  -13278,  -13462,  -13645,  -13827,
107 -14009,  -14191,  -14372,  -14552,  -14732,  -14911,  -15090,  -15268,
108 -15446,  -15623,  -15799,  -15975,  -16150,  -16325,  -16499,  -16672,
109 -16845,  -17017,  -17189,  -17360,  -17530,  -17699,  -17868,  -18036,
110 -18204,  -18371,  -18537,  -18702,  -18867,  -19031,  -19194,  -19357,
111 -19519,  -19680,  -19840,  -20000,  -20159,  -20317,  -20474,  -20631,
112 -20787,  -20942,  -21096,  -21249,  -21402,  -21554,  -21705,  -21855,
113 -22004,  -22153,  -22301,  -22448,  -22594,  -22739,  -22883,  -23027,
114 -23169,  -23311,  -23452,  -23592,  -23731,  -23869,  -24006,  -24143,
115 -24278,  -24413,  -24546,  -24679,  -24811,  -24942,  -25072,  -25201,
116 -25329,  -25456,  -25582,  -25707,  -25831,  -25954,  -26077,  -26198,
117 -26318,  -26437,  -26556,  -26673,  -26789,  -26905,  -27019,  -27132,
118 -27244,  -27355,  -27466,  -27575,  -27683,  -27790,  -27896,  -28001,
119 -28105,  -28208,  -28309,  -28410,  -28510,  -28608,  -28706,  -28802,
120 -28897,  -28992,  -29085,  -29177,  -29268,  -29358,  -29446,  -29534,
121 -29621,  -29706,  -29790,  -29873,  -29955,  -30036,  -30116,  -30195,
122 -30272,  -30349,  -30424,  -30498,  -30571,  -30643,  -30713,  -30783,
123 -30851,  -30918,  -30984,  -31049,  -31113,  -31175,  -31236,  -31297,
124 -31356,  -31413,  -31470,  -31525,  -31580,  -31633,  -31684,  -31735,
125 -31785,  -31833,  -31880,  -31926,  -31970,  -32014,  -32056,  -32097,
126 -32137,  -32176,  -32213,  -32249,  -32284,  -32318,  -32350,  -32382,
127 -32412,  -32441,  -32468,  -32495,  -32520,  -32544,  -32567,  -32588,
128 -32609,  -32628,  -32646,  -32662,  -32678,  -32692,  -32705,  -32717,
129 -32727,  -32736,  -32744,  -32751,  -32757,  -32761,  -32764,  -32766,
130 };
131
132 // Function prototypes
133 void fix_fft(short fr[], short fi[], short m);
134
135 #endif

1 /*****
2     fft.c
3
4     FFT Audio Analysis
5     Copyright (C) 2011 Simon Inns
6
7     This program is free software: you can redistribute it and/or modify
8     it under the terms of the GNU General Public License as published by
9     the Free Software Foundation, either version 3 of the License, or
10    (at your option) any later version.
11
12    This program is distributed in the hope that it will be useful,
13    but WITHOUT ANY WARRANTY; without even the implied warranty of
14    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
15    GNU General Public License for more details.
16
17    You should have received a copy of the GNU General Public License
18    along with this program. If not, see <http://www.gnu.org/licenses/>.
19
20    Email: simon.inns@gmail.com
21
22    *****/
23
24 #ifndef FFT_C
25 #define FFT_C
26
27 #include <htc.h>
28 #include "fft.h"
29
30 // fix_fft.c - Fixed-point in-place Fast Fourier Transform

```

```

31
32 // All data are fixed-point short integers, in which -32768
33 // to +32768 represent -1.0 to +1.0 respectively. Integer
34 // arithmetic is used for speed, instead of the more natural
35 // floating-point.
36 //
37 // For the forward FFT (time -> freq), fixed scaling is
38 // performed to prevent arithmetic overflow, and to map a 0dB
39 // sine/cosine wave (i.e. amplitude = 32767) to two -6dB freq
40 // coefficients.
41 //
42 // Written by: Tom Roberts 11/8/89
43 // Made portable: Malcolm Slaney 12/15/94 malcolm@interval.com
44 // Enhanced: Dimitrios P. Bouras 14 Jun 2006 dbouras@ieee.org
45 // Ported to PIC18F: Simon Inns 20110104
46
47 /*
48  fix_fft() - perform forward fast Fourier transform.
49  fr[n],fi[n] are real and imaginary arrays, both INPUT AND
50  RESULT (in-place FFT), with 0 <= n < 2**m
51 */
52 void fix_fft(short fr[], short fi[], short m)
53 {
54     long int mr = 0, nn, i, j, l, k, istep, n, shift;
55     short qr, qi, tr, ti, wr, wi;
56
57     n = 1 << m;
58     nn = n - 1;
59
60     /* max FFT size = N_WAVE */
61     //if (n > N_WAVE) return -1;
62
63     /* decimation in time - re-order data */
64     for (m=1; m<=nn; ++m)
65     {
66         l = n;
67         do
68         {
69             l >>= 1;
70         } while (mr+l > nn);
71
72         mr = (mr & (l-1)) + l;
73         if (mr <= m) continue;
74
75         tr = fr[m];
76         fr[m] = fr[mr];
77         fr[mr] = tr;
78         ti = fi[m];
79         fi[m] = fi[mr];
80         fi[mr] = ti;
81     }
82
83     l = 1;
84     k = LOG2_N_WAVE-1;
85
86     while (l < n)
87     {
88         /*
89          fixed scaling, for proper normalization --
90          there will be log2(n) passes, so this results
91          in an overall factor of 1/n, distributed to
92          maximize arithmetic accuracy.
93
94          It may not be obvious, but the shift will be
95          performed on each data point exactly once,
96          during this pass.
97          */
98
99         // Variables for multiplication code

```

```

100     long int c;
101     short b;
102
103     istep = 1 << 1;
104     for (m=0; m<l; ++m)
105     {
106         j = m << k;
107         /* 0 <= j < N_WAVE/2 */
108         wr = sinewave[j+N_WAVE/4];
109         wi = -sinewave[j];
110
111         wr >>= 1;
112         wi >>= 1;
113
114         for (i=m; i<n; i+=istep)
115         {
116             j = i + 1;
117
118             // Here I unrolled the multiplications to prevent overhead
119             // for procedural calls (we don't need to be clever about
120             // the actual multiplications since the pic has an onboard
121             // 8x8 multiplier in the ALU):
122
123             // tr = FIX_MPY(wr,fr[j]) - FIX_MPY(wi,fi[j]);
124             c = ((long int)wr * (long int)fr[j]);
125             c = c >> 14;
126             b = c & 0x01;
127             tr = (c >> 1) + b;
128
129             c = ((long int)wi * (long int)fi[j]);
130             c = c >> 14;
131             b = c & 0x01;
132             tr = tr - ((c >> 1) + b);
133
134             // ti = FIX_MPY(wr,fi[j]) + FIX_MPY(wi,fr[j]);
135             c = ((long int)wr * (long int)fi[j]);
136             c = c >> 14;
137             b = c & 0x01;
138             ti = (c >> 1) + b;
139
140             c = ((long int)wi * (long int)fr[j]);
141             c = c >> 14;
142             b = c & 0x01;
143             ti = ti + ((c >> 1) + b);
144
145             qr = fr[i];
146             qi = fi[i];
147             qr >>= 1;
148             qi >>= 1;
149
150             fr[j] = qr - tr;
151             fi[j] = qi - ti;
152             fr[i] = qr + tr;
153             fi[i] = qi + ti;
154         }
155     }
156
157     --k;
158     l = istep;
159 }
160 }
161
162 #endif

```

Libreria RS232

```

1 /*
2  * File:    rs232.h
3  * Author:  Naoki Pross

```

```

4  *
5  * Created on 7.12.2017
6  */
7
8  #ifndef RS232_H
9  #define RS232_H
10
11 #include <stddef.h>
12
13 // defines which eusart device is used with printf()
14 #define EUSART_1_PRINTF 1
15
16 extern void eusart1_init(void);
17 extern void eusart2_init(void);
18
19 extern void eusart1_putch(char c);
20 extern void eusart2_putch(char c);
21
22 extern char eusart1_getch(void);
23 extern char eusart2_getch(void);
24
25 extern char eusart1_getche(void);
26 extern char eusart2_getche(void);
27
28 extern void eusart1_write(void *data, size_t len);
29 extern void eusart2_write(void *data, size_t len);
30
31
32 #endif /* RS232_H */

```



```

1  #include "rs232.h"
2
3  #include <xc.h>
4
5  #ifdef EUSART_1_PRINTF
6  void putch(char c) {
7      #if EUSART_1_PRINTF
8          eusart1_putch(c);
9      #else
10         eusart2_putch(c);
11     #endif
12 }
13 #endif
14
15 void eusart1_init(void)
16 {
17     // set Async and 8 bits frame
18     TXSTA1bits.SYNC = 0;
19     TXSTA1bits.TX9 = 0;
20
21     // baud prescaler
22     RCSTA1bits.SPEN = 1;
23     // SPBRG1 = 103;          // 9600 bps
24     SPBRG1 = 16;            // 57600 bps
25     SPBRGH1 = 0;
26     TXSTA1bits.BRGH = 0;
27     BAUDCON1bits.BRG16 = 0;
28
29     // set up TX / RX pins
30     TRISCbits.TRISC7 = 1;
31     TRISCbits.TRISC6 = 1;
32     // enable continuous reception
33     RCSTA1bits.CREN = 1;
34     TXSTA1bits.TXEN = 1;
35 }
36
37 void eusart2_init(void)
38 {
39     // set Async and 8 bits frame

```

```

40     TXSTA2bits.SYNC = 0;
41     TXSTA2bits.TX9 = 0;
42
43     // baud prescaler
44     RCSTA2bits.SPEN = 1;
45     SPBRG2 = 103;                      // 9600 bps
46     SPBRGH2 = 0;
47     TXSTA2bits.BRGH = 0;
48     BAUDCON2bits.BRG16 = 0;
49
50     // set up TX / RX pins
51     TRISDbits.TRISD7 = 1;
52     TRISDbits.TRISD6 = 1;
53     // enable continuous reception
54     RCSTA2bits.CREN = 1;
55     TXSTA2bits.TXEN = 1;
56 }
57
58 void eusart1_putch(char c)
59 {
60     while (!TX1IF);
61     TX1REG = c;
62 }
63
64 void eusart2_putch(char c)
65 {
66     while (!TX2IF);
67     TX2REG = c;
68 }
69
70 char eusart1_getch(void)
71 {
72     while (!RC1IF);
73     return RC1REG;
74 }
75
76 char eusart2_getch(void)
77 {
78     while (!RC2IF);
79     return RC2REG;
80 }
81
82 char eusart1_getche(void)
83 {
84     char c = eusart1_getch();
85     eusart1_putch(c); // echo
86
87     return c;
88 }
89
90 char eusart2_getche(void)
91 {
92     char c = eusart2_getch();
93     eusart2_putch(c); // echo
94
95     return c;
96 }
97
98 void eusart1_write(void *data, size_t len)
99 {
100     char *dptr = (char *) data;
101
102     while (len--) {
103         eusart1_putch(*(dptr++));
104     }
105 }
106
107 void eusart2_write(void *data, size_t len)
108 {

```

```

109     char *dptr = (char *) data;
110
111     while (len--) {
112         eusart2_putchar(*(dptr++));
113     }
114 }

```

Codice dell'applicativo desktop

```

1 #include "mainwindow.h"
2
3 #include <QApplication>
4
5 int main(int argc, char *argv[])
6 {
7     QApplication a(argc, argv);
8     MainWindow w;
9     w.show();
10
11     return a.exec();
12 }

```

Finestra principale

```

1 #ifndef MAINWINDOW_H
2 #define MAINWINDOW_H
3
4 #include <QMainWindow>
5 #include <QThread>
6 #include <QVector>
7
8 #include <complex>
9
10 #include "qcustomplot.h"
11 #include "serialworker.h"
12 #include "serial/serial.h"
13
14 namespace Ui {
15 class MainWindow;
16 }
17
18 class MainWindow : public QMainWindow
19 {
20     Q_OBJECT
21
22 public:
23     explicit MainWindow(QWidget *parent = 0);
24     ~MainWindow();
25
26     void serialLog(const QString &text);
27
28 private slots:
29     void serialDataReceiver(QVector<std::complex<int>> data);
30
31     void on_serialBtn_clicked();
32     void on_adjustAxisCheckBox_toggled(bool checked);
33     void on_logFreqAxisCheckBox_toggled(bool checked);
34     void on_plotTypeSelCombo_currentIndexChanged(int index);
35
36     void on_actionQuit_triggered();
37     void on_actionSave_data_triggered();
38     void on_actionExport_image_triggered();
39
40
41 private:
42     Ui::MainWindow *_ui;
43     serial::Serial _serial;
44

```

```

45     SerialWorker _serialWorker;
46     QVector<double> _xsamples;
47     QVector<double> _ysamples;
48
49     QSharedPointer<QCPAxisTicker> _defaultTicker;
50     QSharedPointer<QCPAxisTicker> _logTicker;
51 };
52
53 #endif // MAINWINDOW_H

1 #include "mainwindow.h"
2 #include "exportimagedialog.h"
3
4 #include "ui_mainwindow.h"
5
6 #include <QMetaType>
7 #include <QFileDialog>
8 #include <QFileInfo>
9 #include <QDir>
10
11 #include <exception>
12 #include <algorithm>
13 #include <iostream>
14
15 #define _USE_MATH_DEFINES
16 #include <math.h>
17
18 MainWindow::MainWindow(QWidget *parent) :
19     QMainWindow(parent),
20     _ui(new Ui::MainWindow),
21     _serial(),
22     _serialWorker(_serial)
23 {
24     // qt setup ui
25     _ui->setupUi(this);
26
27     // set default serial device
28     _ui->serialDevice->setText(
29 #ifdef _WIN32
30     "COM1"
31 #elif defined(__linux__)
32     "/dev/ttyUSB0"
33 #endif
34     );
35
36     // enable automatic axis scaling
37     _ui->adjustAxisCheckBox->setChecked(true);
38     // _ui->setAxisBtn->setEnabled(false);
39
40     // graph 0 on bottom and left axis
41     _ui->plot->addGraph(_ui->plot->xAxis, _ui->plot->yAxis);
42     // graph 1 on bottom and left axis
43     _ui->plot->addGraph(_ui->plot->xAxis, _ui->plot->yAxis);
44
45     // set graph data
46     _xsamples.clear();
47     _ysamples.clear();
48
49     // graph 1 on top and right axis
50     _ui->plot->addGraph(_ui->plot->xAxis2, _ui->plot->yAxis2);
51
52     // primary axis
53     _ui->plot->xAxis->setLabel("Frequency");
54     _ui->plot->yAxis->setLabel("Amplitude");
55
56     _ui->plot->xAxis->setRange(0, 10000);
57     _ui->plot->yAxis->setRange(0, 5);
58
59     _ui->plot->xAxis->setNumberFormat("f");

```



```

60     _ui->plot->yAxis->setNumberFormat("gb");
61
62     _ui->plot->xAxis->setNumberPrecision(0);
63     _ui->plot->yAxis->setNumberPrecision(0);
64
65     _defaultTicker = _ui->plot->xAxis->ticker();
66     _logTicker = QSharedPointer<QCPAxisTickerLog>(new QCPAxisTickerLog);
67
68     // secondary axis
69     // _ui->plot->xAxis2->setLabel("");
70     // _ui->plot->yAxis2->setLabel("");
71
72     // _ui->plot->xAxis2->setRange(0, 1000);
73     // _ui->plot->yAxis2->setRange(0, 5);
74
75     // show seconday axis
76     // _ui->plot->xAxis2->setVisible(true);
77     // _ui->plot->yAxis2->setVisible(true);
78
79     // register metatype to use on qt events (signals)
80     qRegisterMetaType<QVector<std::complex<int>>>("QVector<std::complex<int>>");
81
82     // serial device received data callback
83     connect(
84         &_serialWorker, SIGNAL(receivedData(QVector<std::complex<int>>)),
85         this, SLOT(serialDataReceiver(QVector<std::complex<int>>))
86     );
87
88     // connect serial exception to close button
89     // TODO: build a slot to close the serial device
90     connect(
91         &_serialWorker, SIGNAL(exception()),
92         this, SLOT(on_serialBtn_clicked())
93     );
94
95     // combobox changed index callback
96     connect(_ui->plotTypeSelCombo,
97         QOverload<int>::of(&QComboBox::currentIndexChanged),
98         [=](int index){on_plotTypeSelCombo_currentIndexChanged(index);}
99     );
100 }
101
102 MainWindow::~MainWindow()
103 {
104     if (_serialWorker.isRunning()) {
105         _serialWorker.requestInterruption();
106         _serialWorker.wait();
107     }
108
109     if (_serial.isOpen()) {
110         _serial.close();
111     }
112
113     delete _ui;
114 }
115
116 void MainWindow::serialLog(const QString &text)
117 {
118     _ui->serialDisplay->append(text);
119 }
120
121 void MainWindow::serialDataReceiver(QVector<std::complex<int>> data)
122 {
123     // constants to convert from integer data to double
124     //
125     const double xConvert = 20000.0/128.0;
126     double yConvert;
127     //
128     const double yAdjustLowerFactor = 1.0;

```

```

129     const double yAdjustUpperFactor = 1.1;
130
131     // reset data
132     _xsamples.clear();
133     _ysamples.clear();
134
135     switch (_ui->plotTypeSelCombo->currentIndex()) {
136     case 0: // amplitude
137     case 2: // complex
138     case 3: // real
139     case 4: // imaginary
140         _ui->plot->yAxis->setRange(0, 5);
141         yConvert = 5.0/1024.0;
142         break;
143
144     case 1: // phase
145         _ui->plot->yAxis->setRange(-M_PI, M_PI);
146         yConvert = 2.0 * M_PI;
147         break;
148     }
149
150
151
152     // add data to plot
153     // the first frequency bucket and the second half cannot be used
154     for (int i = 1; i < data.size() / 2; i++) {
155         // log data, disabled for better performance
156         // serialLog(QString::number(data[i].real()) + "i" + QString::number(data[i].imag())
157             );
158
159         double yvalue = 0;
160         switch (_ui->plotTypeSelCombo->currentIndex()) {
161         case 0:
162             // amplitude
163             yvalue = std::abs(data[i]);
164             break;
165         case 1:
166             // phase
167             yvalue = std::arg(data[i]);
168             break;
169         case 2:
170             // complex
171             // TODO
172             break;
173         case 3:
174             // real
175             yvalue = data[i].real();
176             break;
177         case 4:
178             // imaginary
179             yvalue = data[i].imag();
180             break;
181         }
182
183         _ysamples.append(static_cast<double>(yvalue * yConvert));
184         _xsamples.append(static_cast<double>((_ysamples.size() + 1) * xConvert));
185     }
186
187     // set scale axis
188     if (_ui->adjustAxisCheckBox->isChecked()) {
189         // only y axis needs adjustments
190         _ui->plot->yAxis->setRangeLower((
191             *std::min_element(_ysamples.begin(), _ysamples.end())
192             ) * yAdjustLowerFactor);
193
194         _ui->plot->yAxis->setRangeUpper((
195             *std::max_element(_ysamples.begin(), _ysamples.end())
196             ) * yAdjustUpperFactor);
197     }

```

```

197
198 // plot data
199 _ui->plot->graph(0)->setData(_xsamples, _ysamples, true);
200 _ui->plot->replot();
201 }
202
203 void MainWindow::on_serialBtn_clicked()
204 {
205     if (_serial.isOpen()) {
206         // close serial thread
207         if (_serialWorker.isRunning()) {
208             _serialWorker.requestInterruption();
209             _serialWorker.wait();
210         }
211
212         // close serial device
213         _serial.close();
214         serialLog("Serial device closed");
215
216         // change text and enable widgets
217         _ui->serialBtn->setText("Open");
218         _ui->serialDevice->setEnabled(true);
219         _ui->baudSpinBox->setEnabled(true);
220
221         return;
222     }
223
224     // open serial device
225     try {
226         _serial.setPort(_ui->serialDevice->text().toStdString());
227         _serial.setBaudrate(_ui->baudSpinBox->value());
228         _serial.open();
229         _serialWorker.start();
230
231         serialLog("Serial device opened");
232
233         // change text and disable widgets
234         _ui->serialBtn->setText("Close");
235         _ui->serialDevice->setEnabled(false);
236         _ui->baudSpinBox->setEnabled(false);
237     } catch (const serial::IOException &e) {
238         serialLog("Failed to open serial device");
239         serialLog("IOException:");
240         serialLog(e.what());
241     } catch (const std::exception &e) {
242         serialLog("Exception:");
243         serialLog(e.what());
244     }
245 }
246
247 void MainWindow::on_adjustAxisCheckBox_toggled(bool checked)
248 {
249     _ui->plot->replot();
250 }
251
252
253 void MainWindow::on_logFreqAxisCheckBox_toggled(bool checked)
254 {
255     if (checked) {
256         // set log axis
257         _ui->plot->xAxis->setTicker(_logTicker);
258         _ui->plot->xAxis->setScaleType(QCPAxis::stLogarithmic);
259         _ui->plot->xAxis->setNumberFormat("ebc");
260     } else {
261         _ui->plot->xAxis->setTicker(_defaultTicker);
262         _ui->plot->xAxis->setScaleType(QCPAxis::stLinear);
263         _ui->plot->xAxis->setNumberFormat("f");
264     }
265 }

```

```

266     _ui->plot->replot();
267 }
268
269 void MainWindow::on_plotTypeSelCombo_currentIndexChanged(int index)
270 {
271     switch (index) {
272     case 0: // amplitude
273     case 2: // complex
274     case 3: // real
275     case 4: // imaginary
276     {
277         QSharedPointer<QCPAxisTickerFixed> fixedTicker(new QCPAxisTickerFixed);
278         _ui->plot->yAxis->setTicker(fixedTicker);
279     }
280     break;
281
282     case 1: // phase
283     {
284         QSharedPointer<QCPAxisTickerPi> piTicker(new QCPAxisTickerPi);
285         _ui->plot->yAxis->setTicker(piTicker);
286     }
287     break;
288     }
289
290     _ui->plot->replot();
291 }
292
293 void MainWindow::on_actionQuit_triggered()
294 {
295     QApplication::quit();
296 }
297
298 void MainWindow::on_actionSave_data_triggered()
299 {
300     QString filename = QFileDialog::getSaveFileName(this,
301         "Save Data", QDir::homePath(),
302         "Comma Separated Values (*.csv)");
303
304     if (filename.isNull())
305         return;
306
307     // check if there is any data to write
308     if (_xsamples.empty() || _ysamples.empty()) {
309         serialLog("No data to write");
310         return;
311     }
312
313     // check that file has an extensions
314     QFileInfo fileInfo(filename);
315     if (fileInfo.suffix() == "")
316         filename += ".csv";
317
318     QFile file(filename);
319     if (!file.open(QIODevice::WriteOnly | QIODevice::Text)) {
320         serialLog("Failed to open file:");
321         serialLog(filename);
322         return;
323     }
324
325 #ifdef _WIN32
326     const QString eol = "\n\r";
327 #else
328     const QString eol = "\n";
329 #endif
330
331     QTextStream out(&file);
332
333     // header
334     out << "time, magnitude" << eol;

```

```

335
336     const int samples_size = (_xsamples.size() > _ysamples.size())
337         ? _xsamples.size() : _ysamples.size();
338
339     for (int i = 0; i < samples_size; i++) {
340         out << QString::number(_xsamples[i]) << ", "
341         << QString::number(_ysamples[i]) << eol;
342     }
343
344     file.close();
345 }
346
347 void MainWindow::on_actionExport_image_triggered()
348 {
349     // ask output image size
350     ExportImageDialog dialog(this);
351     if (dialog.exec() != QDialog::Accepted) {
352         return;
353     }
354
355     QString selectedFilter;
356     QString filename = QFileDialog::getSaveFileName(this,
357         "Save plot image", QDir::homePath(),
358         "Vector Image (*.pdf);;Bitmap Image (*.bmp);;Compressed Image (*.png *.jpg)",
359         &selectedFilter
360     );
361
362     // the user did not select a file
363     if (filename.isNull())
364         return;
365
366     QFileInfo fileInfo(filename);
367
368     // if the user has not specified an extension
369     if (fileInfo.suffix() == "") {
370         if (selectedFilter == "Vector Image (*.pdf)")
371             filename += ".pdf";
372         else if (selectedFilter == "Bitmap Image (*.bmp)")
373             filename += ".bmp";
374         else if (selectedFilter == "Compressed Image (*.png *.jpg)")
375             filename += ".png";
376
377         // update FileInfo
378         fileInfo = QFileInfo(filename);
379     }
380
381     // write data
382     if (fileInfo.suffix() == "pdf")
383         _ui->plot->savePdf(filename,
384             dialog.getImageWidth(),
385             dialog.getImageHeight());
386     else if (fileInfo.suffix() == "bmp")
387         _ui->plot->saveBmp(filename,
388             dialog.getImageWidth(),
389             dialog.getImageHeight());
390     else if (fileInfo.suffix() == "png")
391         _ui->plot->savePng(filename,
392             dialog.getImageWidth(),
393             dialog.getImageHeight());
394     else if (fileInfo.suffix() == "jpg" || fileInfo.suffix() == "jpeg")
395         _ui->plot->saveJpg(filename,
396             dialog.getImageWidth(),
397             dialog.getImageHeight());
398     else {
399         serialLog("Invalid file type: ");
400         serialLog(filename);
401     }
402 }

```

Dialog per esportare le immagini

```
1 #ifndef EXPORTIMAGEDIALOG_H
2 #define EXPORTIMAGEDIALOG_H
3
4 #include <QDialog>
5
6 namespace Ui {
7 class ExportImageDialog;
8 }
9
10 class ExportImageDialog : public QDialog
11 {
12     Q_OBJECT
13
14 public:
15     explicit ExportImageDialog(QWidget *parent = 0);
16     ~ExportImageDialog();
17
18     int getImageWidth() const;
19     int getImageHeight() const;
20
21 private:
22     Ui::ExportImageDialog *_ui;
23 };
24
25 #endif // EXPORTIMAGEDIALOG_H

```



```
1 #include "exportimagedialog.h"
2 #include "ui_exportimagedialog.h"
3
4 ExportImageDialog::ExportImageDialog(QWidget *parent) :
5     QDialog(parent),
6     _ui(new Ui::ExportImageDialog)
7 {
8     _ui->setupUi(this);
9 }
10
11 ExportImageDialog::~ExportImageDialog()
12 {
13     delete _ui;
14 }
15
16
17 int ExportImageDialog::getImageWidth() const
18 {
19     return _ui->widthSpinBox->value();
20 }
21
22 int ExportImageDialog::getImageHeight() const
23 {
24     return _ui->heightSpinBox->value();
25 }
```

Gestione della risorsa seriale

```
1 #ifndef SERIALWORKER_H
2 #define SERIALWORKER_H
3
4 #include <QObject>
5 #include <QThread>
6 #include <QMutex>
7 #include <QVector>
8
9 #include <complex>
10
11 #include "serial/serial.h"
12
```

```

13 class SerialWorker : public QThread
14 {
15     Q_OBJECT
16 public:
17     SerialWorker() = delete;
18     explicit SerialWorker(serial::Serial &serial);
19     ~SerialWorker();
20
21     void run() override;
22
23 signals:
24     void receivedData(QVector<std::complex<int>> data);
25     void exception();
26
27 private:
28     QMutex _mutex;
29     serial::Serial &_serial;
30 };
31
32 #endif // SERIALWORKER_H

1 #include "serialworker.h"
2
3 #include <QMutexLocker>
4
5 #include <string>
6 #include <complex>
7
8
9 SerialWorker::SerialWorker(serial::Serial &serial) :
10     _mutex(), _serial(serial)
11 {
12
13 }
14
15 SerialWorker::~~SerialWorker()
16 {
17
18 }
19
20 void SerialWorker::run()
21 {
22     while (!isInterruptionRequested()) {
23         QMutexLocker locker(&_mutex);
24         QVector<std::complex<int>> data;
25
26         while (!isInterruptionRequested()) {
27             QString str;
28
29             try {
30                 // wait for serial buffer to accumulate at least 12 bytes
31                 while (_serial.available() < 14)
32                     if (isInterruptionRequested())
33                         return;
34
35                 // read data
36                 str = QString::fromStdString(_serial.readline());
37             } catch (serial::IOException e) {
38                 emit exception();
39                 return;
40             }
41
42
43             // start of data
44             if (str.trimmed() == "S") {
45                 data.clear();
46             }
47             // end of data
48             else if (str.trimmed() == "E") {

```

```

49         break;
50     }
51     // data
52     else {
53         QStringList valueStr = str.trimmed().split('i');
54         if (valueStr.size() < 2)
55             continue;
56
57         bool isInt[2];
58         std::complex<int> value(
59             valueStr.at(0).toInt(&isInt[0]),
60             valueStr.at(1).toInt(&isInt[1])
61         );
62
63         // if (!isInt[0] || !isInt[1])
64         //     continue
65
66         data.push_back(value);
67     }
68 }
69
70 emit receivedData(data);
71 }
72 }

```