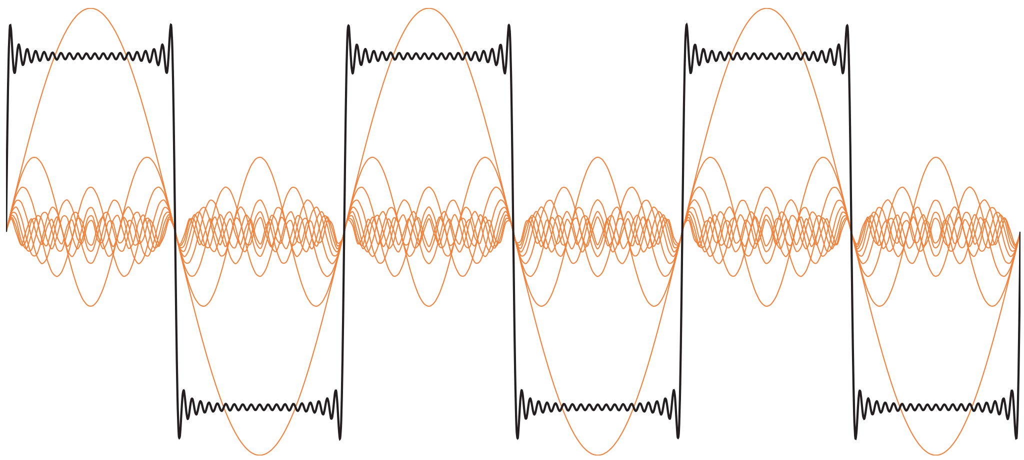
Anno scolastico 2017-18
Lavoro Professionale Individuale

Nome e 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 di fine lavoro:	**15.05.2018**	Ore effettive:	**– UD**

Abstract

Per completare un percorso formativo alla SAM è richiesto lo sviluppo di un progetto. Questa documentazione descrive e documenta lo sviluppo di un prototipo di un circuito di analisi spettrale a 3 entrate: 2 jack ed un Cinc/RCA, in grado di offrire uno spettro fino a 10 kHz. Realizzato intorno ad un PIC18F45K22 con l'utilizzo dell'algoritmo della Fast Fourier Transform (FFT), è possibile osservare i valori complessi dello spettro mediante la porta seriale RS232 oppure utilizzando il software sviluppato in C++ per Windows e Linux (probabilmente è possibile compilarlo anche sotto MacOS).

Il progetto si è concluso con lo sviluppo di un primo design per un PCB, funzionante, ed uno schema elettrico revisionato.

Indice

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 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 perchè 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	LaTeX
Diario di lavoro	LaTeX
Pianificazione	MS Excel 2016
LaTeX engine	XeLaTeX
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

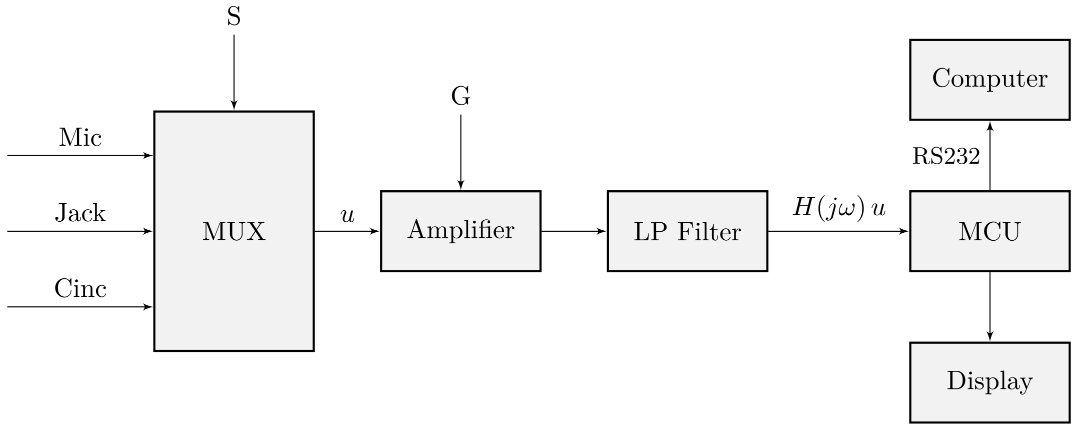


Figura 2.1: Schema a blocchi

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 osservarzioni.

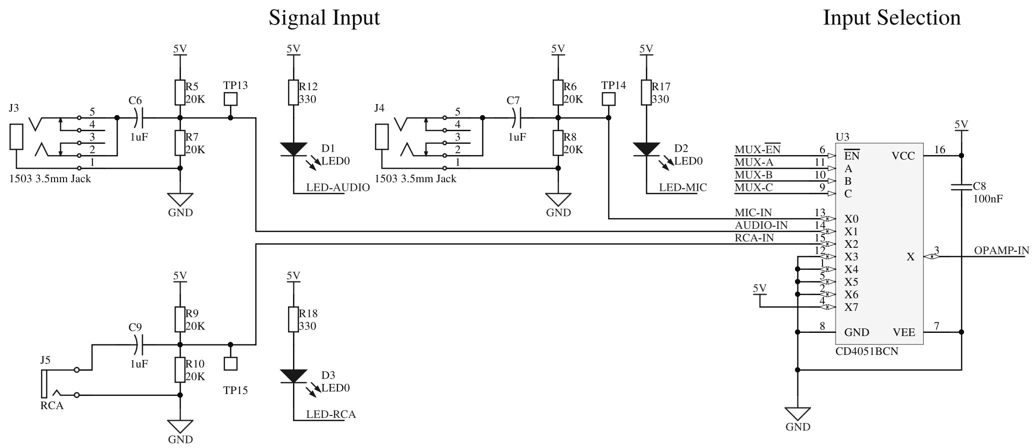


Figura 2.2: Circuito di selezione delle entrate

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

2.3 Circuito di entrata

Il segnale di cui si analizza lo spettro, prima di essere campionato, viene adattato mediate 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*.

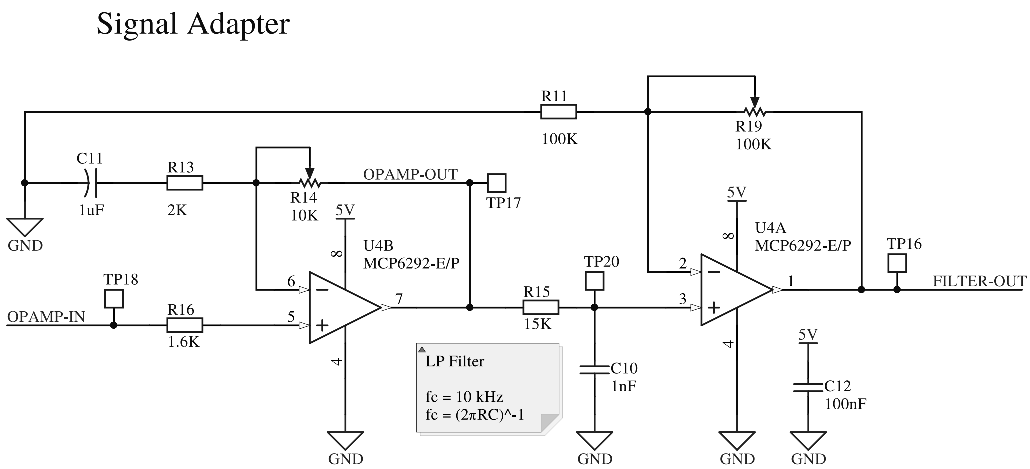


Figura 2.3: Circuito di adattamento del segnale in entrata.

È importante notare che per questa applicazione si è scelto utilizzare degli opamp *rail to rail*, 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 2.5, il circuito di amplificazione non ha una configurazione tipica. Esso è basato su una configurazione non invertente ma dispone di un consensatore (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 2.5 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.

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 Tranform.

Un ultima ragione importante per la scelta di questo componente è data dalla disponibilità di una libreria per controllare il 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.

Tabella 2.1: Sommario della configurazione utilizzata

Componente utilizzato	**Valore**	**Osservazioni**
Oscillatore Interno	16 MHz	Questa *non* è la frequenza di lavoro
PLL	×4	La frequenza di lavoro è di 16×4 MHz = 64 MHz
ADC	`ANSA0`	Utilizzato per il campionamento
I/O ports	`A, B, C, D`	
Timer 2	Vedi 3.1	Utilizzato per il campionamento
I/O interrupts	`INT0 (RB0)`	Utilizzato per la selezione del canale
EUSART 1		Utilizzato per trasferire i campioni della FFT.
EUSART 2		Porta di debugging

2.5 Schemi originali

Dopo la realizzazione del primo prototipo, gli schemi sono stati revisionati, apportando delle correzioni. In questa sezione sono presenti gli schemi originali utilizzati per produrre il (primo) prototipo.

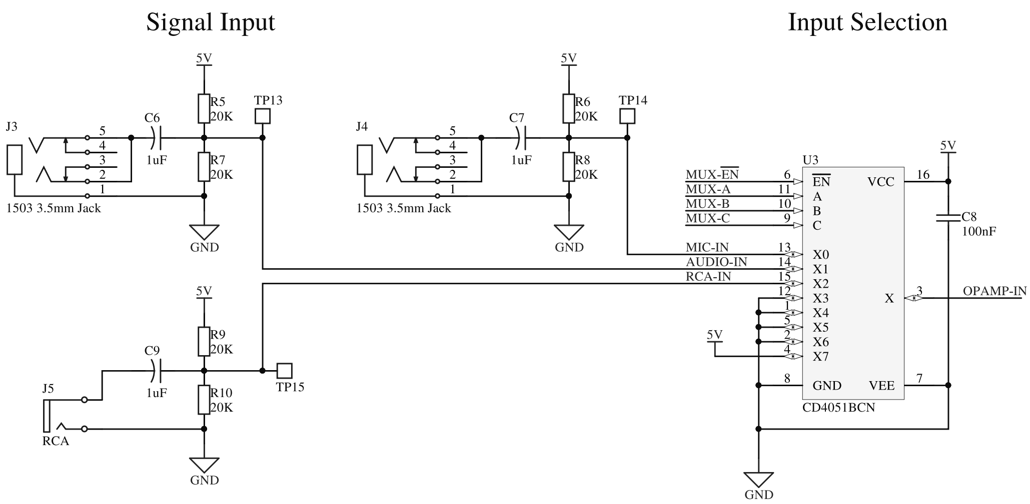


Figura 2.4: Circuito di selezione delle entrate (prima versione)

Signal Adapter

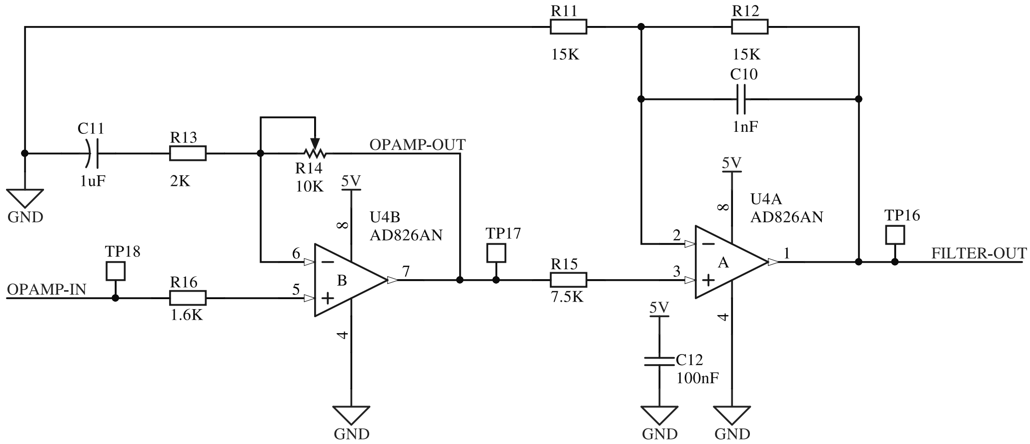


Figura 2.5: Circuito di adattamento del segnale in entrata (prima versione)

3 Software

3.1 Campionamento

Per campionare il segnale è stato scelto di utilizzare il TIMER2, sia per la sua semplicità che per la granulatià 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

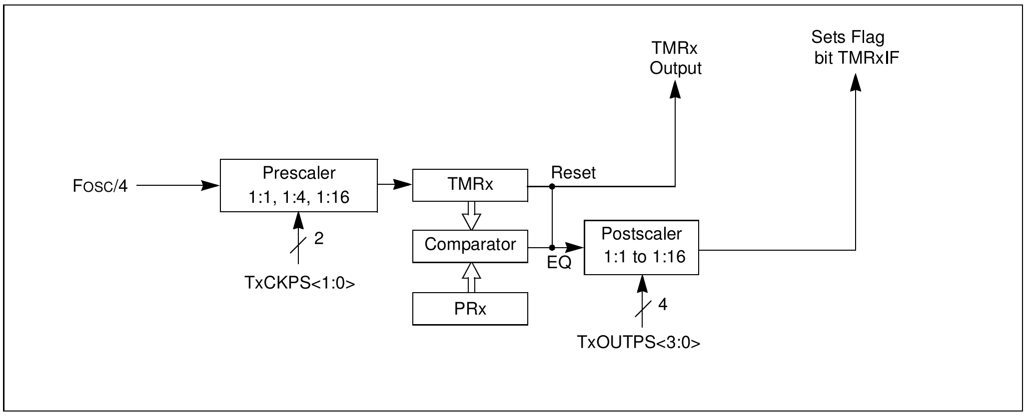


Figura 3.1: Schema a blocchi del TIMER2. Fonte: Microchip PIC18F2X/4XK22 datasheet

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 $1/50\,\mu\text{s} = 20\,\text{kHz}$.

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

3.2 Trasferimento dei dati

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

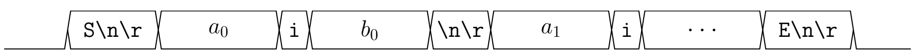


Figura 3.2: Protocollo di trasmissione dei dati, il primo dato mandato è $a_0 + ib_0$

I dati sono trasferiti con un semplice protocollo illustrato nella figura 3.2. Un frame di dati incomincia con il carattere ASCII maiuscolo 'S' seguito da un line break (`\n\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++ anzichè 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 multipiattaforma. 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 multipiattaforma per l'accesso di basso livello alla porta seriale del sistema operativo sottostatante. 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. Pochè 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 liberia 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+=releae" \
    -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-deskop
$ 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.

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 rispondo 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.

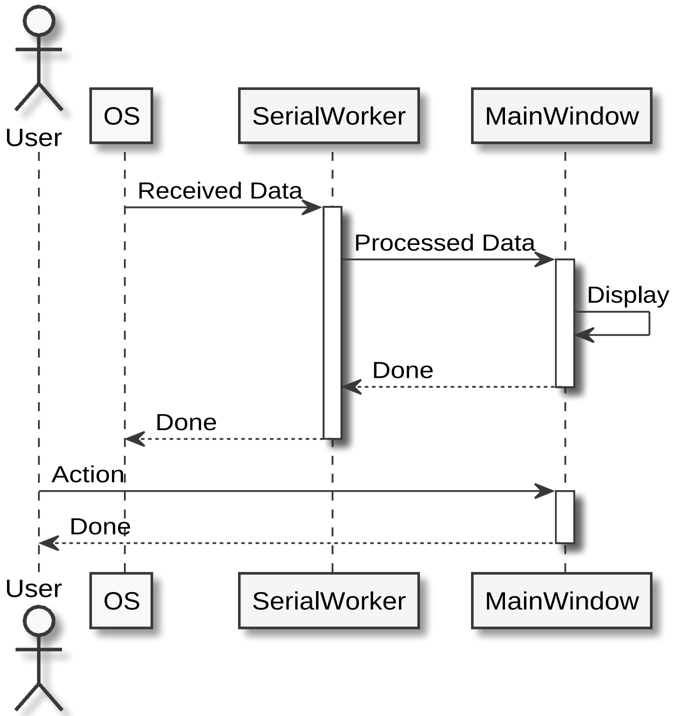


Figura 3.3: Diagramma delle sequenze

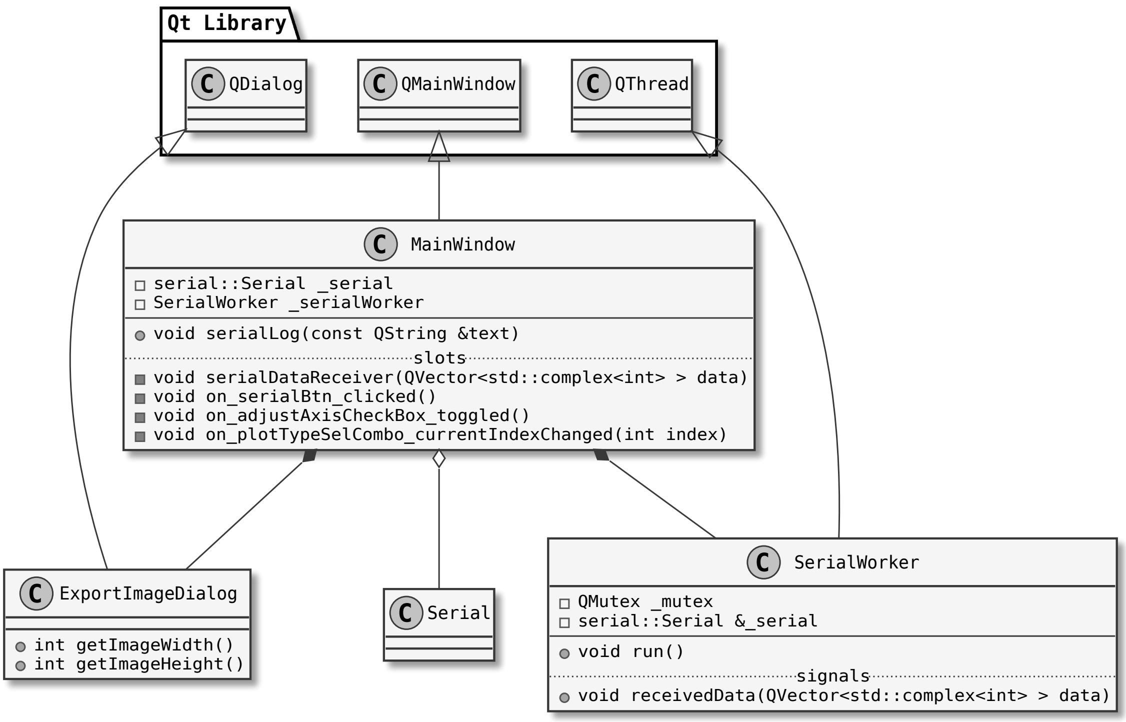


Figura 3.4: Diagramma delle classi

3.3.7 Interfaccia utente

L'interfaccia utente, realizzata con Qt è molto semplice e non dovrebbe richiedere delle istruzioni d'utilizzo. Dal software è possibile esportare i dati sia in formato CSV sia in un immagine del grafico in formato vettoriale o bitmap / compresso (png, jpg).

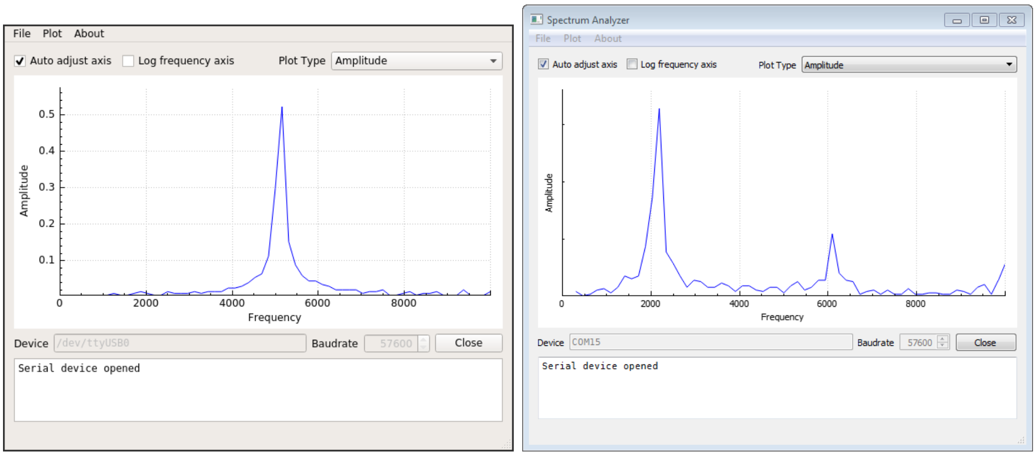


Figura 3.5: L'applicativo sotto Fedora 27 (sinistra) e Windows 7 (destra)

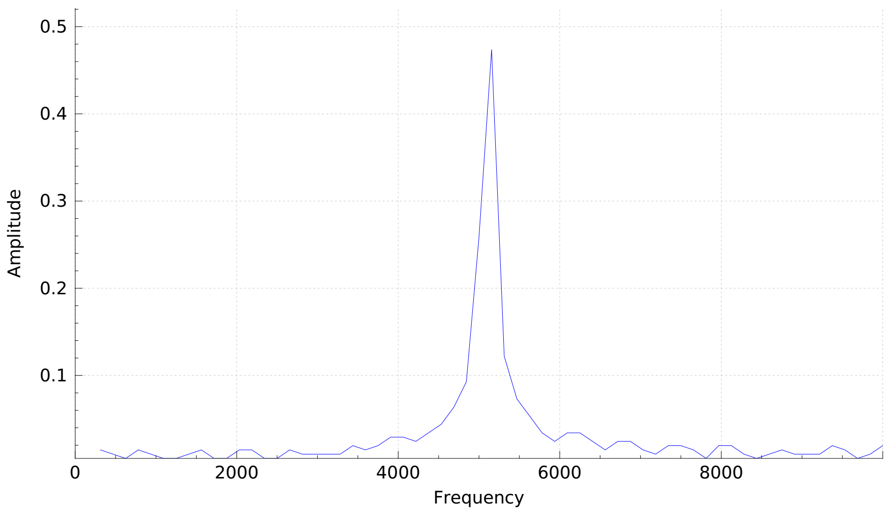


Figura 3.6: Esempio di immagine esportata dal software

3.4 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

4.1.1 Esempi di misurazioni

I dati illustrati nella figura 4.1 sono esportati dal programma desktop.

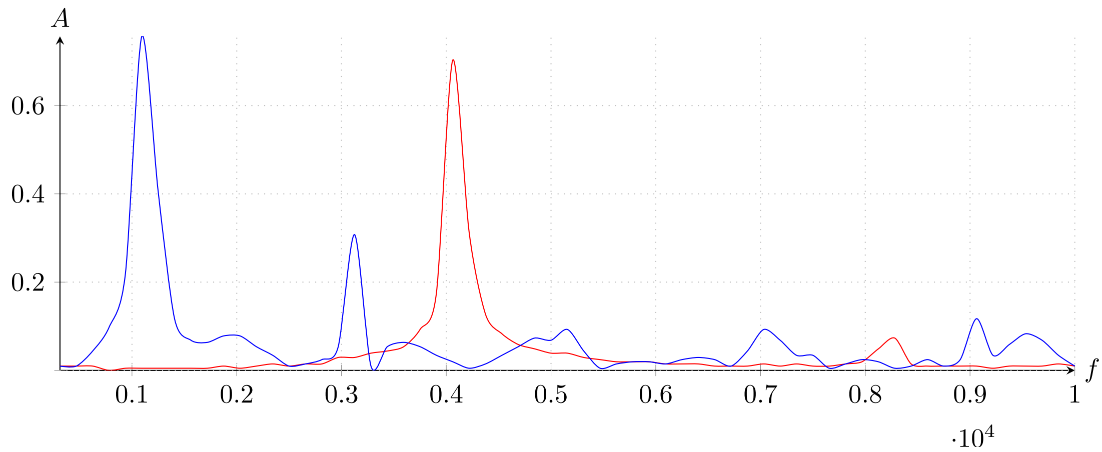


Figura 4.1: Spettro di un'onda sinusoidale (rosso) e di un segnale quadrato (blu)

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. Un possibile sostituto, utilizzato nella seconda revisione dello schema, potrebbe essere l'operazionale MCP6292 (in alternativa: MCP6002, TI OPA2340, ICL7621DCPAZ).

4.2.2 Errore nel dimensionamento del filtro attivo

Durante la fase di test è stato inoltre rilevato che il filtro attivo (vedi figura 2.5) 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.

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

Un valore sostitutivo per il resistore R_{11} è $\geq 750\,\mathrm{k\Omega}$, 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 $910\,\mathrm{k\Omega}$.

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

Così facendo però è stata modificata la frequenza di taglio del filtro. Perciò si è concluso che il circuito scelto non è utilizzabile e nella seconda revisione la configurazione è stata sostituita da una cella RC con un amplificatore non invertente.

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 instanziata 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 tentativo di lettura seguente il metodo `serial::Serial::read()` causa una `serial::IOException` che fa crashare il programma.

La ragione per cui il thread non veniva fermato, era l'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.

```diff
diff --git a/src-desktop/mainwindow.cpp b/src-desktop/mainwindow.cpp
index 73c416e..15c7d77 100644
--- a/src-desktop/mainwindow.cpp
+++ b/src-desktop/mainwindow.cpp
@@ -42,8 +42,12 @@ void MainWindow::serialDataReceiver(const QString &data)
 void MainWindow::on_serialBtn_clicked()
 {
     if (_serial.isOpen()) {
-        _serialWorker.quit();
-        while (_serialWorker.isRunning());
+        // _serialWorker.quit();
+        // while (_serialWorker.isRunning());
+        if (_serialWorker.isRunning()) {
+            _serialWorker.requestInterruption();
+            _serialWorker.wait();
+        }

         _serial.close();
         serialLog("Serial device closed");
diff --git a/src-desktop/serialworker.cpp b/src-desktop/serialworker.cpp
index 15301cb..f70bce8 100644
--- a/src-desktop/serialworker.cpp
+++ b/src-desktop/serialworker.cpp
@@ -1,6 +1,7 @@
 #include "serialworker.h"

 #include <QMutexLocker>
+#include <string>

```

```
 SerialWorker::SerialWorker(serial::Serial &serial) :
     _mutex(), _serial(serial)
@@ -15,12 +16,17 @@ SerialWorker::~SerialWorker()

 void SerialWorker::run()
 {
-    while (isRunning()) {
+    while (!isInterruptionRequested()) {
         QMutexLocker locker(&_mutex);
-        while (!_serial.available() && isRunning());
+        while (!_serial.available() && !isInterruptionRequested());

-        QString data = QString::fromStdString(_serial.readline());
-        emit receivedData(data);
+        for (std::string line : _serial.readlines()) {
+            QString data = QString::fromStdString(line);
+            emit receivedData(data);
+        }
+
+        // QString data = QString::fromStdString(_serial.readline());
+        // emit receivedData(data);
         _serial.flushOutput();
     }
 }
```

4.3 Possibili miglioramenti e sviluppi futuri

4.3.1 Filtro

Il circuito di filtraggio del rumore di alta frequenza potrebbe essere sostituito con un filtro di secondo o di terzo ordine per migliorarne le prestazioni.

4.3.2 Circuito di selezione dell'entrata

Il multiplexer utilizzato per l'entrata ha un valore d'impedenza non particolarmente eccezionale, per migliorare le prestazioni si dovrebbe scegliere un multiplexer differente.

4.3.3 Supporto per MacOS

Il software dell'analisi spettrale attualmente non utilizza dipendenze che richiedono una piattaforma particolare. Portare il software per MacOS non dovrebbe essere difficile.

4.4 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.

4.5 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).

4.6 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 esplicato come trovare i coefficienti di una retta a $m+1$ termini interpolando N punti.

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

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 interopolare. 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

$$\varepsilon(a_0, \ldots, a_m) = \sum_{k=1}^{N} \Big[r(x_k, a_0, \ldots, a_m) - y_k \Big]^2 \tag{5.2}$$

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.

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

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 $\vec{u}$

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

5.1.2 Funzione armonica

Una funzione armonica, sinusoidale, può essere descritta in molteplici modi. Nella forma trigonometrica, data la pulsazione $\omega = 2\pi f = \dfrac{2\pi}{T}$ e la fase $\varphi = \dfrac{\pi}{2} - \vartheta$.

$$f(x) = a \cdot \sin(\omega x + \varphi)$$
$$f(x) = b \cdot \cos(\omega x + \vartheta)$$

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.

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

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

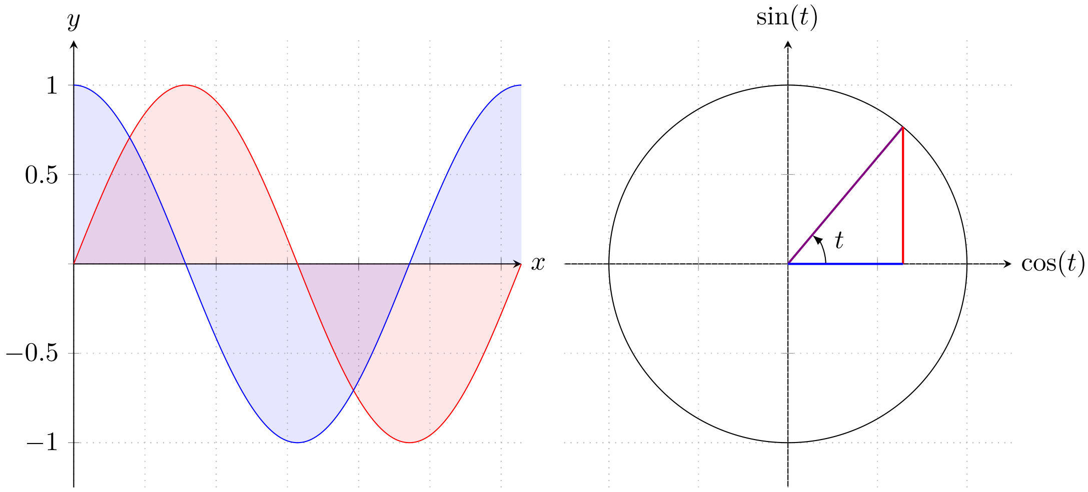


Figura 5.1: Funzioni $\sin$ e $\cos$

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.

Dimostrzioni algebriche

1. Area di $\sin$ in un periodo.

$$\int_0^T \sin(\frac{m2\pi x}{T})\,\mathrm{d}x = 0 \quad \forall m \in \mathbb{Z}$$

$$\begin{aligned}\int_0^T \sin(\frac{m2\pi x}{T})\,\mathrm{d}x &= \Big[-\frac{T}{2\pi m}\cdot\cos\big(\frac{2\pi}{T}mx\big)\Big]_0^T \\ &= -\frac{T}{2\pi m}\cdot\cos\big(2\pi m\big) + \frac{T}{2\pi m}\cdot\cos\big(0\big) \\ &= 0\end{aligned}$$

2. Area di $\cos$ in un periodo.

$$\int_0^T \cos(\frac{m2\pi x}{T})\,\mathrm{d}x = 0 \quad \forall m \in \mathbb{Z}^*$$

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

3. Prodotto tra $\sin$ e $\cos$.

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

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

4. Prodotto tra due $\sin$ di frequenze diverse.

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

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

5. Prodotto tra due $\cos$ di frequenze diverse.

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

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

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

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

$$\begin{aligned}\int_0^T e^{i2\pi kx/T}\,\mathrm{d}x &= \underbrace{\int_0^T \cos(\frac{2\pi}{T}kx)\,\mathrm{d}x}_{k\neq 0 \implies 0} + i\underbrace{\int_0^T \sin(\frac{2\pi}{T}kx)\,\mathrm{d}x}_{0} \\ &= \begin{cases} 0 & \iff k \neq 0 \\ T & \iff k = 0 \end{cases}\end{aligned}$$

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.

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

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

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.

$$T_N(x) = \sum_{n=0}^{N} \left[a_n \cdot \cos(\omega nx) + ib_n \cdot \sin(\omega nx)\right] \qquad a_n, b_n \in \mathbb{C}$$

5.2.2 Polinomio Trigonometrico Reale

È definito inoltre il polinomio trogonometrico *reale* come

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

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

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

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

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

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 trogonometrico 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 dalle basi ortonormate $\sin(\omega x)$ e $\cos(\omega x)$.

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 discontinutià in un qualsiasi intervallo chiuso e le discontinuità non devono essere infinite.

5.4 Trasformata di Fourier discreta

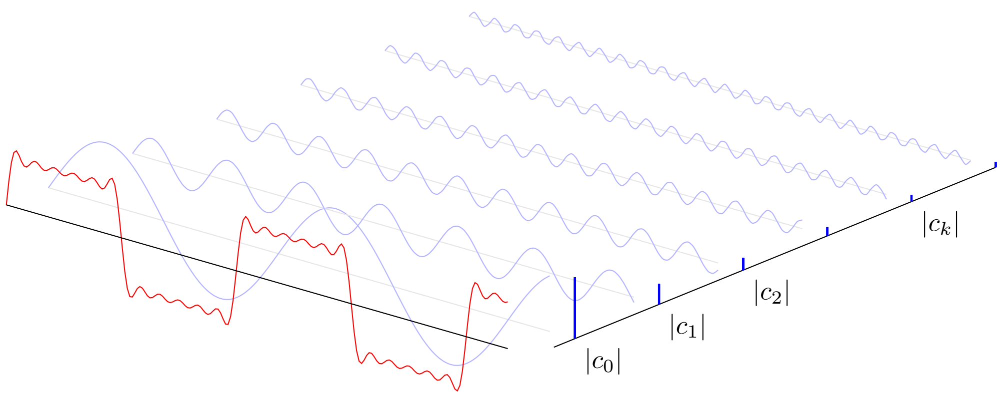


Figura 5.2: Rappresentazione grafica della trasformata di Fourier [8]

La Trasformata di Fouerier 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 sestante 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 dipende dalla frequenza di base scelta e nel caso di un polinomio trigonometrico, anche dal numero di termini di quest'ultimo.

$$\boxed{X_k = c_k \cdot T = \int_0^T f(x) \cdot e^{-i\omega kx} \, \mathrm{d}x}$$

Nella figura 5.2, possiamo osservare sul grafico $x \perp z$ in rosso una funzione, ed il suo spettro sull'asse $y \perp z$ in blu. Si può intuire che più

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.

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

Sappiamo che questo può essere scritto anche nel seguente modo.

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

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.

$$\varepsilon = \int_0^T \big[T_N(x) - f(x)\big]^2 \, \mathrm{d}x$$

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.

$$\frac{\partial \varepsilon}{\partial a_k} = 0 = \frac{\partial}{\partial a_k} \int_0^T \big[T_N(x) - f(x)\big]^2 \, \mathrm{d}x$$

$$0 = \int_0^T \frac{\partial}{\partial a_k} \left[\sum_{n=0}^{N} a_n \cos(\omega nx) + b_n \sin(\omega nx) - f(x)\right]^2 \mathrm{d}x$$

$$0 = \int_0^T 2\left[\sum_{n=0}^{N} a_n \cos(\omega nx) + b_n \sin(\omega nx) - f(x)\right] \cdot a_k \cos(\omega kx) \, \mathrm{d}x$$

$$0 = \underbrace{\int_0^T \cos(\omega kx) \sum_{n=0}^{N} a_n \cos(\omega nx) \, \mathrm{d}x}_{k \neq n \implies 0} + \underbrace{\int_0^T \cos(\omega kx) \sum_{n=0}^{N} b_n \sin(\omega nx) \, \mathrm{d}x}_{0} - \int_0^T \cos(\omega kx) \cdot f(x) \, \mathrm{d}x$$

$$0 = a_k \cdot \frac{T}{2} - \int_0^T \cos(\omega kx) \cdot f(x) \, \mathrm{d}x$$

$$\frac{a_k}{2} = \frac{1}{T} \int_0^T \cos(\omega kx) \cdot f(x) \, \mathrm{d}x$$

Termini dei seni

I termini dei seni b_k sono ottenuti eguagliando la derivata parziale di ε a zero.

$$\begin{aligned}
\frac{\partial \varepsilon}{\partial b_k} = 0 = \frac{\partial}{\partial b_k} \int_0^T & \big[T_N(x) - f(x)\big]^2 \,\mathrm{d}x \\
0 &= \int_0^T \frac{\partial}{\partial b_k} \bigg[\sum_{n=0}^{N} a_n \cos(\omega n x) + b_n \sin(\omega n x) - f(x) \bigg]^2 \,\mathrm{d}x \\
0 &= \int_0^T 2 \bigg[\sum_{n=0}^{N} a_n \cos(\omega n x) + b_n \sin(\omega n x) - f(x) \bigg] \cdot b_k \sin(\omega k x) \,\mathrm{d}x \\
0 &= \underbrace{\int_0^T \sin(\omega k x) \sum_{n=0}^{N} a_n \cos(\omega n x) \,\mathrm{d}x}_{0} + \underbrace{\int_0^T \sin(\omega k x) \sum_{n=0}^{N} b_n \sin(\omega n x) \,\mathrm{d}x}_{k \neq n \implies 0} - \int_0^T \sin(\omega k x) \cdot f(x) \,\mathrm{d}x \\
0 &= b_k \cdot \frac{T}{2} - \int_0^T \sin(\omega k x) \cdot f(x) \,\mathrm{d}x
\end{aligned}$$

$$\frac{b_k}{2} = \frac{1}{T} \int_0^T \sin(\omega k x) \cdot f(x) \,\mathrm{d}x$$

Termine complesso

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

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

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

5.4.2 DFT da un insieme di campioni discreti

L'operazione della DFT può essere applicata anche ad un insieme di N valori discreti, che per esempio potrebbero originare da un campionamento di un segnale elettrico. In tal caso la trasformata discreta di Fourier è la seguente.

$$\boxed{X_k = c_k \cdot T = \sum_{n=0}^{N-1} x_n \cdot e^{-i\omega k n}}$$

Utilizzando questa notazione con gli indici è possibile anche notare che la TFD può essere espressa come una matrice $\mathbf{V}$. Definendo il vettore $\vec{x} = (x_0, x_1, \ldots, x_{N-1})$ e $\vec{X} = (X_0, \ldots, X_{N-1})$ si ottiene

$$\boxed{\vec{X} = \mathbf{V} \cdot \vec{x}}$$

Dove

$$[\mathbf{V}]_{kn} = (e^{i\omega})^{-kn}$$

ossia, se rappresentata come tabella

$$\mathbf{V} = \begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & (e^{i\omega})^{-1} & (e^{i\omega})^{-2} & \dots & (e^{i\omega})^{-(N-1)} \\ 1 & (e^{i\omega})^{-2} & (e^{i\omega})^{-4} & \dots & (e^{i\omega})^{-2(N-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & (e^{i\omega})^{-(N-1)} & (e^{i\omega})^{-2(N-1)} & \dots & (e^{i\omega})^{-(N-1)^2} \end{pmatrix}$$

Osservando la matrice si nota subito una simmetria diagonale, che è data dalla progressione geometrica $[\mathbf{V}]_{ij} = [\mathbf{V}]_i^{j-1}$. Questa matrice è detta una matrice di Vandermonde, da cui è stato scelto il nome $\mathbf{V}$.

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

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

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

$$s(x) = \lim_{T\to\infty} \sum_{n=-\infty}^{\infty} \frac{1}{T} \int_{-T/2}^{T/2} s(x) \cdot e^{-i2\pi xn/T}\,\mathrm{d}x \cdot e^{i2\pi xn/T}$$

Osserviamo che

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

Dunque otteniamo

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

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.

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

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

Elenco delle tabelle

Bibliografia

[1] Real Time Audio Spectrum Analyzer. (online) Simon Inns. Jan 8 2011.
https://www.waitingforfriday.com/?p=325
http://archive.is/IJeAe (archived)

[2] Divide and Couquer: FFT. (online, video)
Erik Demaine, MIT OpenCourseWare. MIT 6.046J Design and Analysis of Algorithms, Spring 2015
https://youtu.be/iTMn0Kt18tg

[3] But what is the Fourier Transform? A visual introduction. (online, video)
Grant Sanderson. Jan 26, 2018. YouTube.
https://youtu.be/spUNpyF58BY

[4] Théorie et traitement des signaux. (Pag. 70 – 72)
Coulon, Frédéric de (1996). Lausanne: Presses polytechniques et universitaires romandes, 1996 (Traité d'électricité de l'Ecole polytechnique fédérale de Lausanne; vol. 6)

[5] Algèbre linéaire. Aide-mémoire, exercices et applications. (Pag. 64 Moindres carrées)
Dalang, Robert C. Chaabouni, Amel (2001). Lausanne: Presses polytechniques et universitaires romandes, 2001

[6] Algèbre linéaire. (Pag 63 – 65)
Cairoli, 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.pgfplots.net/tikz/examples/fourier-transform/

Progetto: *Spectrum Analyzer* Inizio: 12.04.2018

Persona in formazione: *Naoki Pross* Fine: 15.05.2018

		Data	12.04.2018	13.04.2018	16.04.2018	17.04.2018	26.04.2018	27.04.2018	30.04.2018	11.05.2018	14.05.2018	15.05.2018
		Giorno	Gio	Ve	Lu	Ma	Gio	Ve	Lu	Ve	Lu	Ma
			M / P	M / P	M / P	M / P	M / P	M / P	M / P	M / P	M / P	M / P
	ATTIVITÀ	Tempo										
0	Analisi e comprensione della FT, DFT, FFT	10										
		7										
1	Preparazione di un prototipo su piastra sperimentale	9										
		3										
2	Progettazione dello schema elettrico e del PCB	18										
		26										
3	Assemblaggio del PCB	4										
		6										
4	Programmazione del microcontroller	11										
		16										
5	Programmazione dell'interfaccia desktop	11										
		9										
6	Allestimento ed esecuzione di test hardware e software	6										
		4										
7		0										
		0										
8	Stesura documentazione e del diario giornaliero	14										
		3										
	Totale preventivo	83										
	Totale consuntivo	74										

DIARIO GIORNALIERO

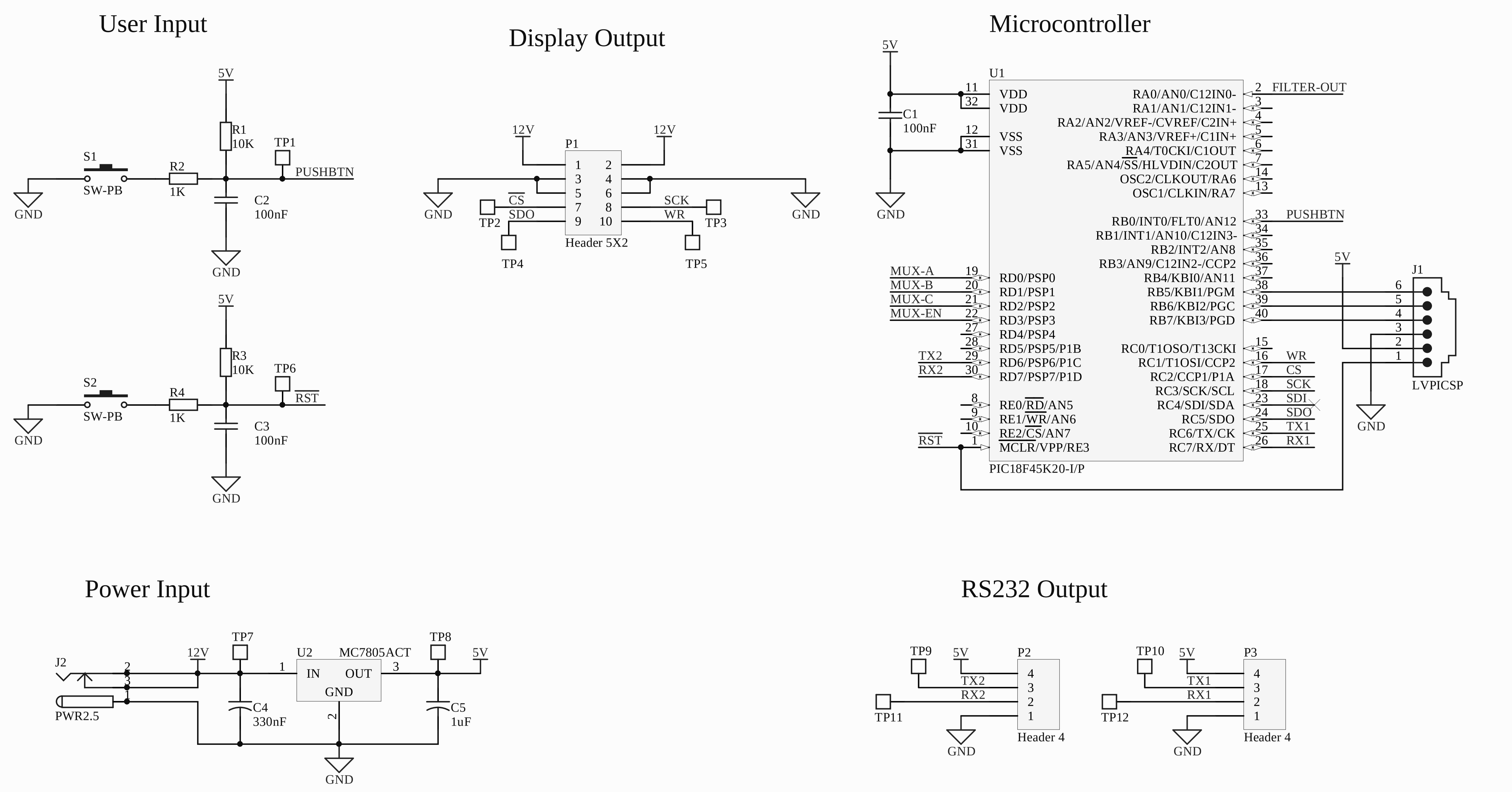
Candidato:	Naoki Pross	**Progetto:**	Spectrum Analyzer
Formatore:	Rinaldo Geiler, Daniele Kamm	**Periodo:**	12.04.2018 – 15.04.2018

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 12 V a 5 V.
Ve	13.04.2018	2	Modificato il circuito progettato per utilizzare un filtro attivo anzichè passivo dopo aver osservato sperimentalmente l'attenuazione dal filtro passivo.	Il circuto 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 poichè 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 precedenta (TL071) non è in grado di lavorare come necessario nel margine da 0 V a 5 V. 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.

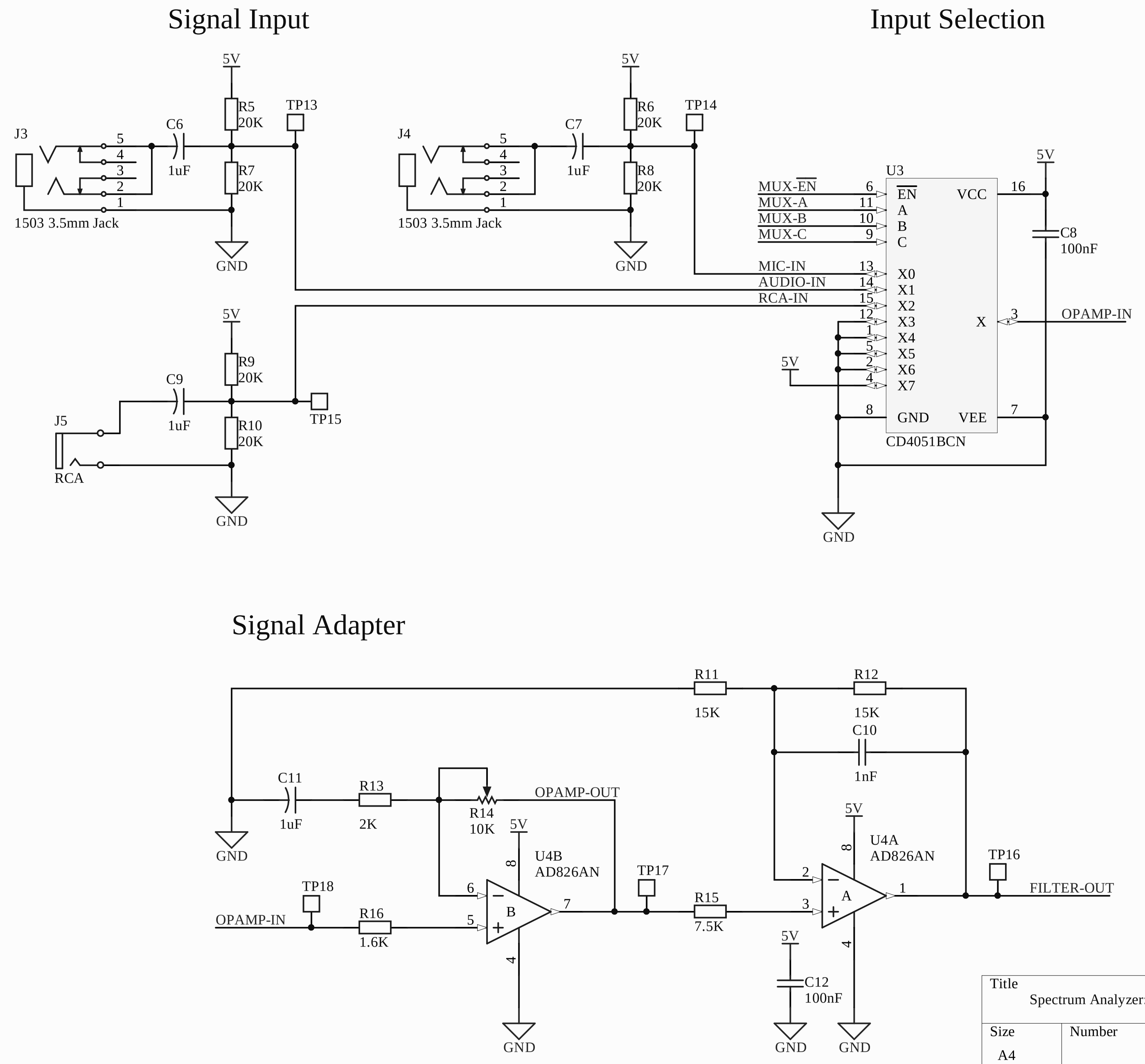
Lu	16.04.2018	1	Corretto il montaggio sulla piastra sperimentale.	Il filtro attivo non è presente sulla tavola perché si deve aspettare la comanda del componente.
Lu	16.04.2018	5	Implementato il codice del microcontroller per configurare l'ADC ed un timer per un campionamento regolare.	Si possono osservare i dettagli su Git nei seguenti commit: `7730a96`, `b836638`, `b482c7e`, `aa19054`, `2996f65`.
Lu	16.04.2018	1	Riordinato lo schema elettrico. Aggiunto il ciruito 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 dendere 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 `SerialWorker` deve essere terminato per rilasciare la risorsa `MainWindow::_serial`, 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 miniori 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 poichè 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 `_serial.waitReadable();` emetteva un `IOException` 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 57.6 k e raddoppiato il numero di campioni. Modificato l'implementazione del PC per utilizzare le strutture `std::complex` invece della mia implementazione `sam::complex_int16_t` 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 910 k 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).	

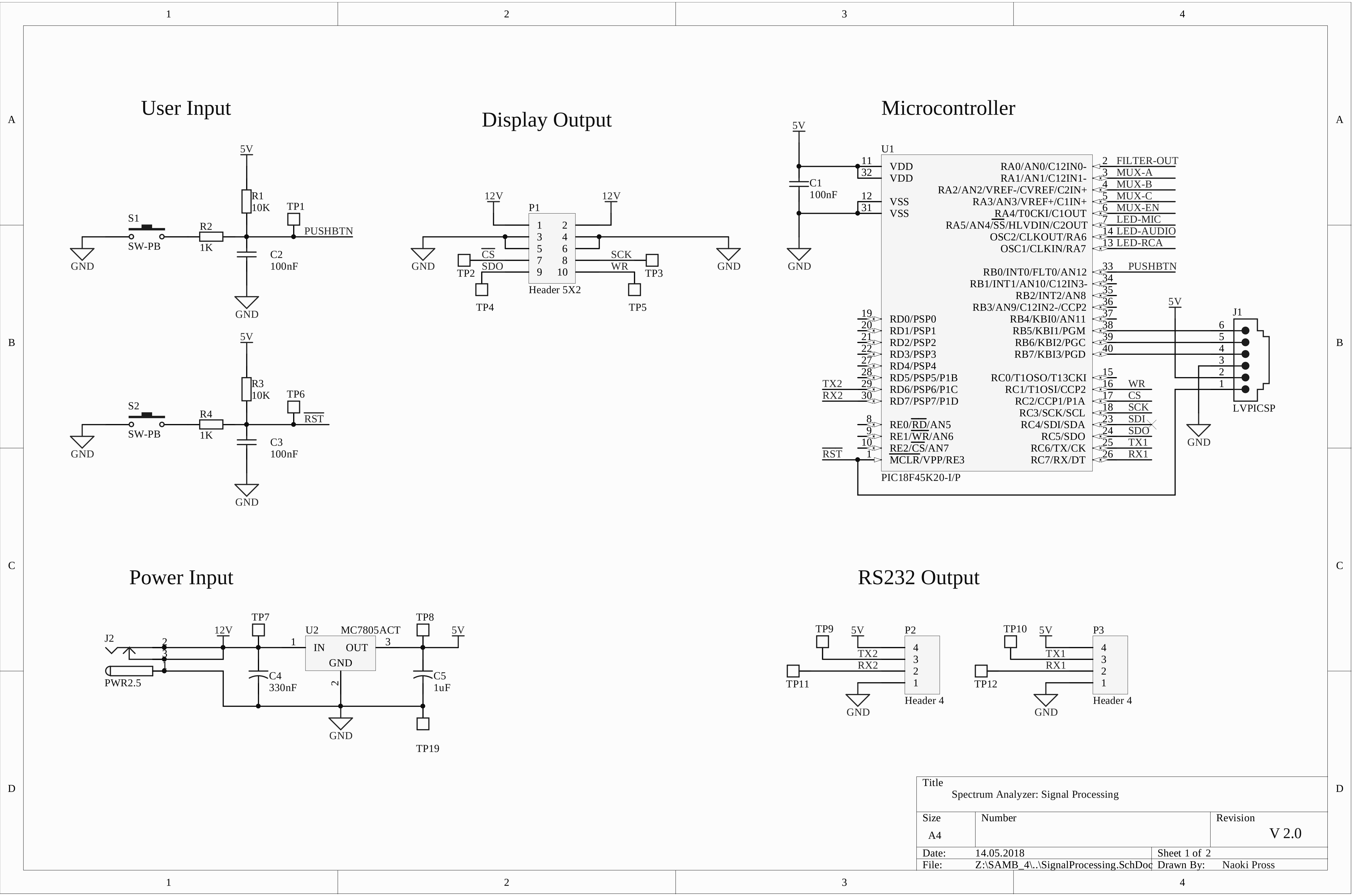
Lu	14.05.2018	2	Apportato le correzioni nello schema elettrico per il prossimo prototipo.	
Lu	14.05.2018	3	Continuato la documentazione.	



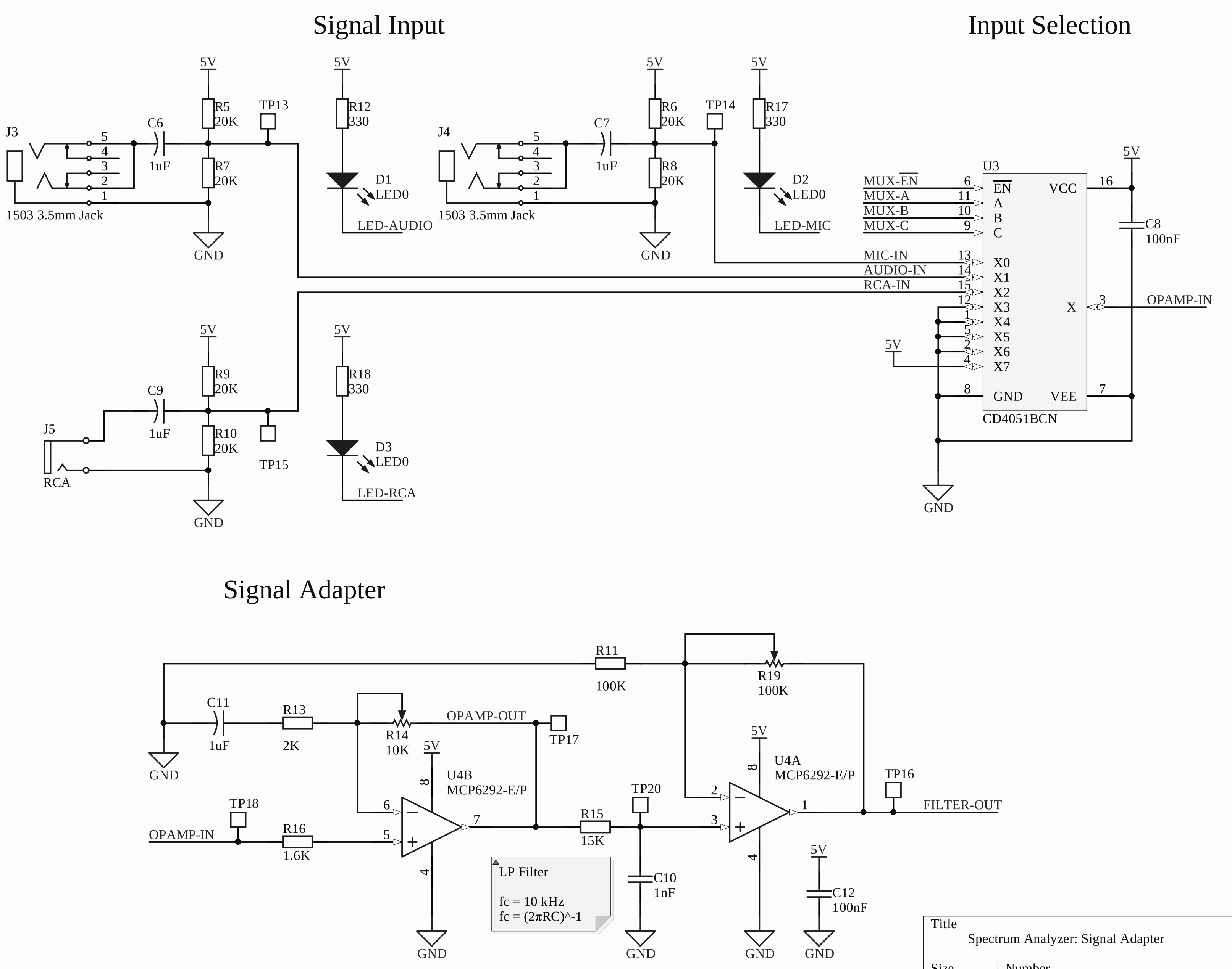
Title	Spectrum Analyzer: Signal Processing		
Size	Number		Revision
A4			
Date:	27.04.2018	Sheet 1 of 2	
File:	Z:\SAMB_4\..\SignalProcessing.SchDoc	Drawn By:	Naoki Pross



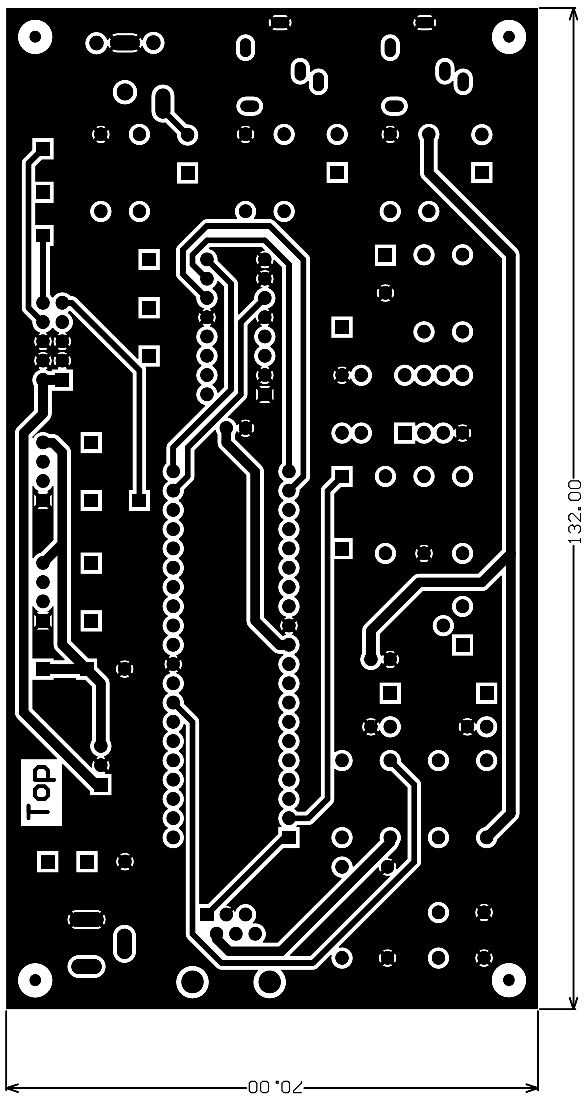
Title: Spectrum Analyzer: Signal Adapter		
Size: A4	Number	Revision
Date: 27.04.2018		Sheet 2 of 2
File: Z:\SAMB_4\..\SignalAdapter.SchDoc		Drawn By: Naoki Pross



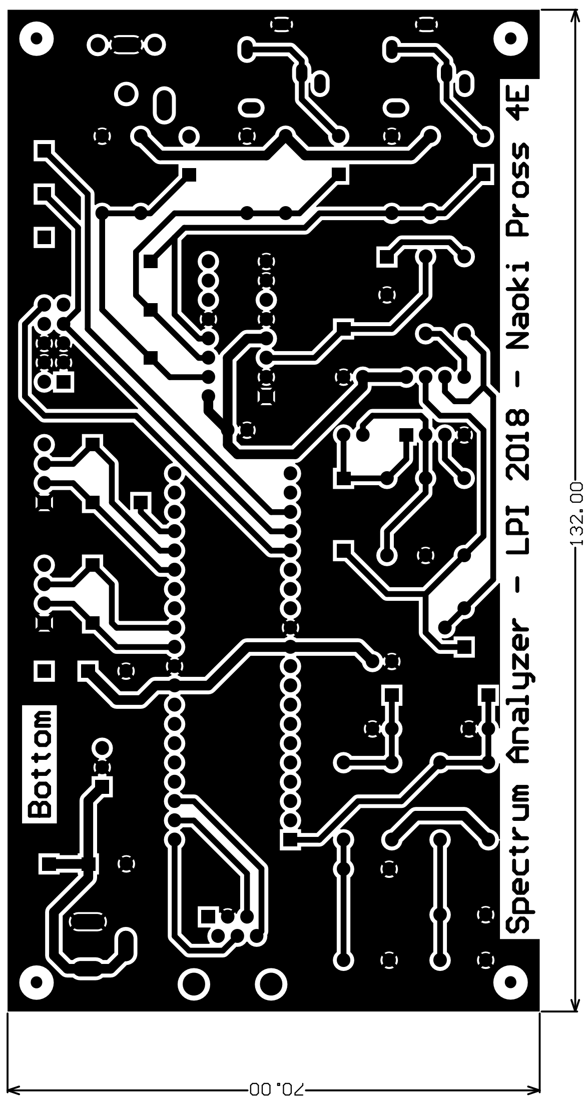
User Input
Display Output
Microcontroller
Power Input
RS232 Output
5V
R1 10K
R2 1K
S1 SW-PB
TP1
PUSHBTN
C2 100nF
GND
R3 10K
R4 1K
S2 SW-PB
TP6
RST
C3 100nF
12V
P1
Header 5X2
CS
SDO
SCK
WR
TP2
TP3
TP4
TP5
U1
C1 100nF
VDD
VSS
RA0/AN0/C12IN0-
RA1/AN1/C12IN1-
RA2/AN2/VREF-/CVREF/C2IN+
RA3/AN3/VREF+/C1IN+
RA4/T0CKI/C1OUT
RA5/AN4/SS/HLVDIN/C2OUT
OSC2/CLKOUT/RA6
OSC1/CLKIN/RA7
RB0/INT0/FLT0/AN12
RB1/INT1/AN10/C12IN3-
RB2/INT2/AN8
RB3/AN9/C12IN2-/CCP2
RB4/KBI0/AN11
RB5/KBI1/PGM
RB6/KBI2/PGC
RB7/KBI3/PGD
RC0/T1OSO/T13CKI
RC1/T1OSI/CCP2
RC2/CCP1/P1A
RC3/SCK/SCL
RC4/SDI/SDA
RC5/SDO
RC6/TX/CK
RC7/RX/DT
RD0/PSP0
RD1/PSP1
RD2/PSP2
RD3/PSP3
RD4/PSP4
RD5/PSP5/P1B
RD6/PSP6/P1C
RD7/PSP7/P1D
RE0/RD/AN5
RE1/WR/AN6
RE2/CS/AN7
MCLR/VPP/RE3
PIC18F45K20-I/P
FILTER-OUT
MUX-A
MUX-B
MUX-C
MUX-EN
LED-MIC
LED-AUDIO
LED-RCA
SDI
TX1
RX1
TX2
RX2
J1
LVPICSP
J2
PWR2.5
TP7
C4 330nF
U2
MC7805ACT
IN
OUT
TP8
C5 1uF
TP19
TP9
TP10
TP11
TP12
P2
P3
Header 4
Title
Spectrum Analyzer: Signal Processing
Size
A4
Number
Revision
V 2.0
Date: 14.05.2018
Sheet 1 of 2
File: Z:\SAMB_4\..\SignalProcessing.SchDoc
Drawn By: Naoki Pross

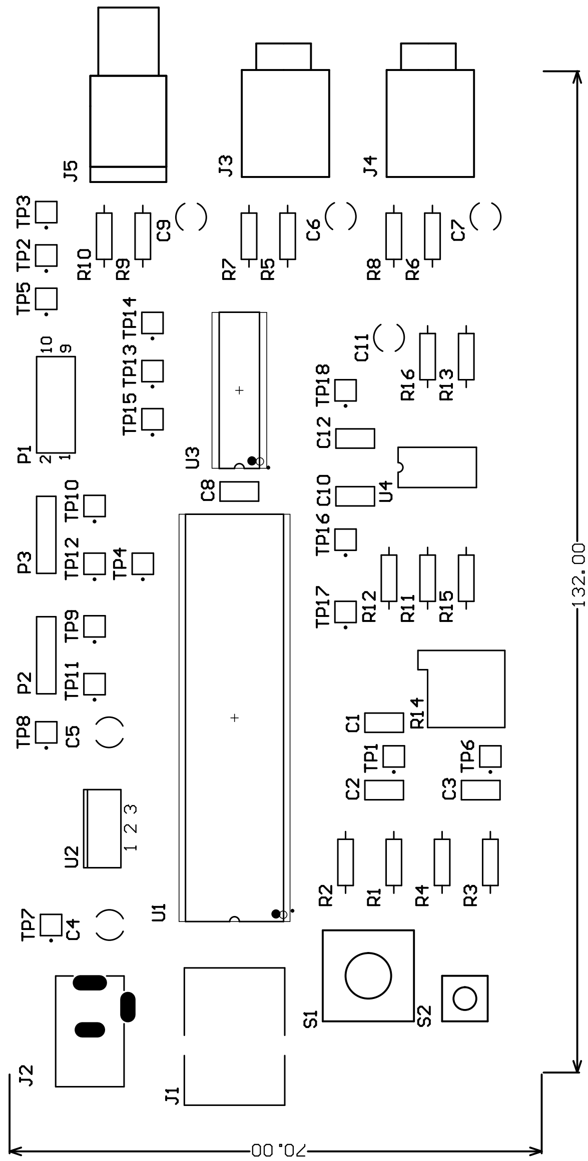


Title		
Spectrum Analyzer: Signal Adapter		
Size: A4	Number	Revision: V 2.0
Date: 14.05.2018		Sheet 2 of 2
File: Z:\SAMB_4\..\SignalAdapter.SchDoc		Drawn By: Naoki Pross



Top
132.00
70.00


Spectrum Analyzer - LPI 2018 - Naoki Pross 4E
Bottom
132.00
70.00



J5
J3
J4
TP3
TP2
TP5
R10
R9
C9
R7
R5
C6
R8
R6
C7
TP14
TP13
TP15
C11
R16
R13
TP18
C12
U3
P1
10
9
2
1
C8
C10
U4
TP10
P3
TP12
TP4
TP16
TP17
R12
R11
R15
TP9
P2
TP11
R14
C1
TP8
C5
TP1
TP6
C2
C3
U2
1 2 3
U1
R2
R1
R4
R3
TP7
C4
S1
S2
J2
J1
132.00
70.00

Codice sorgente

Codice del microcontroller

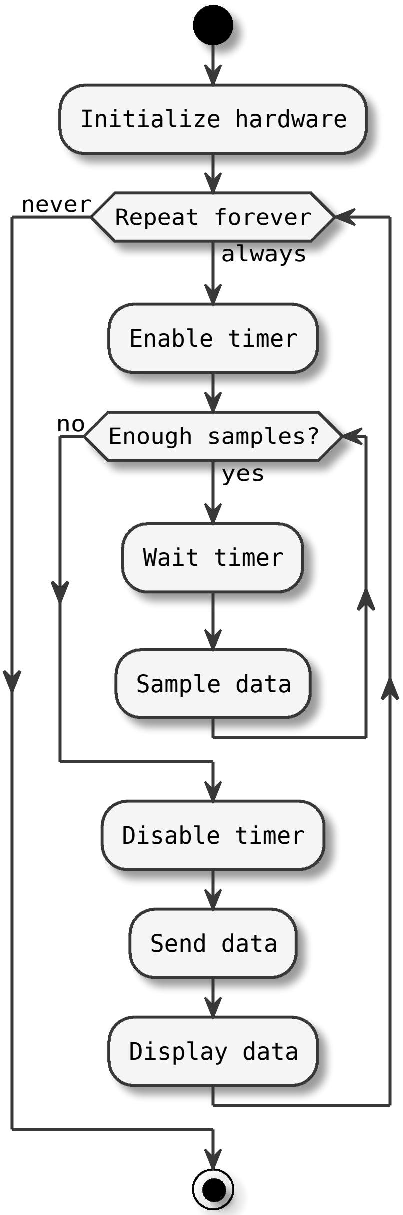


Figura 6.1: Diagramma di flusso del codice del microcontrollore