INFORMATICA MUSICALE

UNIVERSITA' DEGLI STUDI DI CATANIA
DIPARTIMENTO DI MATEMATICA E INFORMATICA
LAUREA TRIENNALE IN INFORMATICA
A.A. 2020/21
Prof. Filippo L.M. Milotta

ID PROGETTO: 0C

TITOLO PROGETTO: Spectrum Analyzer

AUTORE 1: Belfiore Massimo

AUTORE 2: Freni Davide Giovanni

AUTORE 3: Selgi Giovanni

Indice

1. Obiettivi del progetto

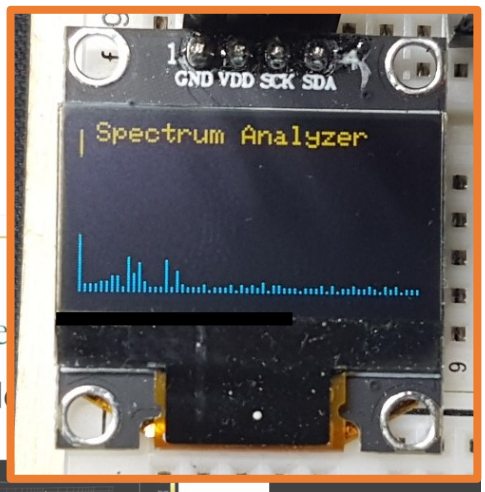
Quando inizialmente abbiamo deciso di realizzare questo progetto avevamo in mente di sfruttare l'occasione per approfondire la nostra conoscenza sul mondo dei microcontrollori e di Arduino, ovvero il futuro dell'IoT, mettendo tra l'altro in pratica le conoscenze precedentemente acquisite in altri corsi di studio, come quelle nel campo dell'informatica e nel campo dell'elettronica.

Dopo esserci procurati una scheda Arduino Uno, un display OLED, un microfono a elettrete e una breadboard, abbiamo iniziato ad immergerci nella programmazione e nella realizzazione di un circuito che ci potesse permettere di raggiungere il risultato previsto.

Fondamentale per ottenere un analizzatore di spettro è stato implementare nel codice l'utilizzo di una specifica libreria che esegua l'algoritmo FFT, il quale ci ha consentito di ottenere la corrispondente rappresentazione di un segnale sonoro nel dominio della frequenza.

Successivamente lo spettro così ottenuto verrà visualizzato in tempo reale sul nostro display, pur tenendo in considerazione i ritardi di acquisizione e di elaborazione.

Spettro audio di una tipica lezione di Informatica Musicale



2. Metodo Proposto

Vediamo tutto quello che c'è dietro il funzionamento di un'analizzatore di spettro.

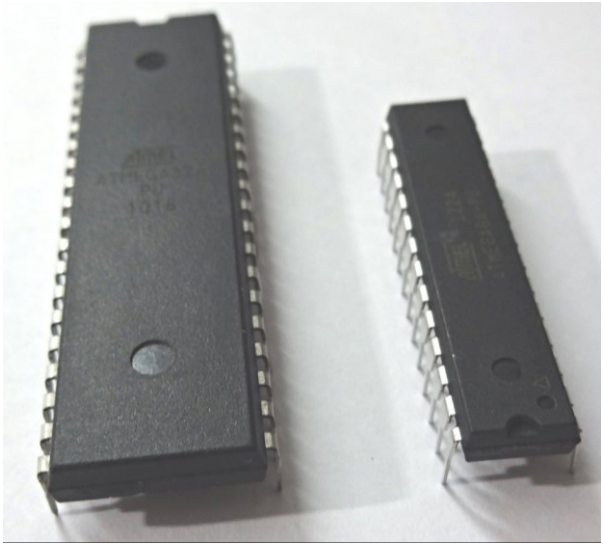
Prima di ogni cosa è opportuno dire cos'è un **microcontrollore** (**MCU**).

Questo risulta essere un piccolo dispositivo dotato di CPU (Central Processing Unit), memorie, ingressi ed uscite digitali, convertitori ADC e periferiche varie, il tutto racchiuso in un unico package, comportando una forte riduzione nelle dimensioni dell'intero sistema.

Tra i notevoli vantaggi ottenuti dal loro utilizzo si annovera un'ottima efficienza energetica, anche a fronte di frequenze di clock che possono essere abbastanza elevate, una riduzione delle dimensioni del sistema ed una maggiore affidabilità.

Altra caratteristica dei MCU è la grande flessibilità di utilizzo. Grazie agli ingressi di cui sono dotati, possono interfacciarsi con il mondo esterno mediante specifici sensori, che forniscono i dati più disparati.

Dalle condizioni meteo con valori di temperatura, luminosità, umidità, presenza di gas alla rilevazione di terremoti, possono aiutare l'uomo in molteplici modi. Le uscite disponibili inoltre consentono di comandare luci e interruttori, o di impostare i settaggi preferiti dall'utilizzatore di ogni elettrodomestico a piacere del programmatore, perché sì, sono ormai onnipresenti.

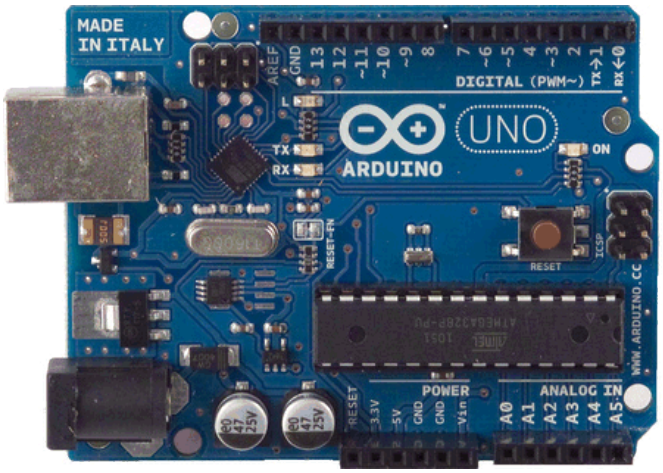


Un decisivo impulso alla diffusione dei microcontrollori ad una vasta gamma di utenti è da ricondursi ad alcuni membri dell'Interaction Design Institute di Ivrea, che hanno dato vita al progetto Arduino, tutto italiano, copiato ormai da molte aziende grazie alla natura open-source del progetto. Nasce come strumento di prototipazione rapida e per scopi hobbisti, didattici e professionali, e promette di aiutare chiunque a sviluppare le proprie idee, programmando in C dall'apposito Arduino IDE.

Link al sito ufficiale di Arduino, in cui è disponibile lo store, una sezione per il software, un forum dedicato e molto altro: https://www.arduino.cc.

Esistono differenti versioni in base alle esigenze e a ciò che si intende realizzare. Per il nostro progetto abbiamo usato **Arduino UNO**, un microfono **Electret** ed un **display OLED**, sviluppando il software che gestisce il tutto.

La scheda Arduino che abbiamo scelto per il nostro progetto è dotata di ATMEGA328, un microcontrollore ad 8 bit ad alte prestazioni prodotto dalla Atmel con architettura di tipo RISC (acronimo di Reduced Instruction Set Computer).



Essa dispone di 14 ingressi/uscita digitali (di cui 6 possono essere utilizzate come uscite PWM), 6 ingressi analogici, etichettati da A0 ad A5, ognuno dei quali fornisce 10 bit di risoluzione (in pratica 1024 valori diversi).

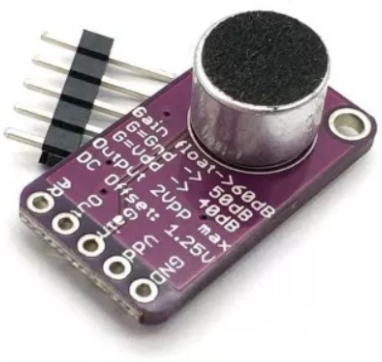
Per quanto riguarda l'oscillatore invece, è integrato un risuonatore ceramico a 16 MHz.

In più sono disponibili anche un collegamento USB, un jack di alimentazione, un header ICSP, un LED pilotabile dall'utente e un pulsante di reset.

MICROFONO MAX9814

Un microfono a elettrete è un tipo di microfono a condensatore che viene fornito con una carica "permanente" incorporata.

Come tutti i microfoni a condensatore, anche questi necessitano di una coppia di piastre cariche (piastre positive e negative) per poter funzionare e registrare il suono.



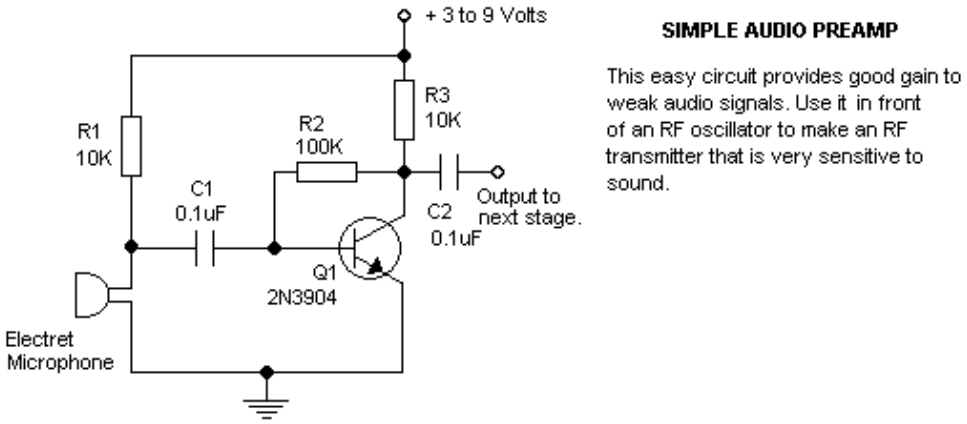
Un microfono a elettrete è una variazione del microfono a condensatore, ovvero un tipo di microfono che sfrutta l'effetto di variazione capacitiva, captando come le piastre sono sollecitate dalle variazioni di pressione che un suono provoca nel mezzo di propagazione. Invece di richiedere una fonte di tensione esterna per caricare il

diaframma, un microfono a elettrete utilizza un elemento di plastica caricato in modo permanente (elettrete) posizionato in parallelo con una piastra posteriore metallica conduttiva.

Anche se i microfoni a elettrete sono precaricati e non necessitano di tensione per caricarli, richiedono comunque tensione per funzionare. La tensione non carica il diaframma come i microfoni a condensatore, ma è necessaria per alimentare l'amplificatore, con cui il nostro microfono viene fornito.

La maggior parte dei microfoni a elettrete ha un piccolo amplificatore FET integrato nelle loro custodie. Questo amplificatore richiede alimentazione per funzionare. Questa tensione viene immessa nel microfono tramite una resistenza (da 1 KΩ a 10 KΩ). I microfoni a elettrete rispondono meglio alle frequenze comprese tra le frequenze medio-basse e alte, mentre non rispondono bene alle basse frequenze. Per questo motivo, tendono a essere limitati alle comunicazioni vocali.

Uno svantaggio dei microfoni a elettrete è che le loro prestazioni diminuiscono nel corso degli anni in quanto, col passare del tempo, la carica sull'elettrete viene persa.



Link alla fonte: http://www.reconnsworld.com/audio_simplepreamp.html

Ad occuparsi della **conversione analogico-digitale** è l'ADC integrato nell'Arduino UNO

Sappiamo che il suono è un segnale analogico che, se pensato come tono puro, può essere interpretato come una funzione sinusoidale dotata di una certa intensità e di una certa frequenza.

Un dispositivo di trasduzione analogico riceve in ingresso un suono come una variazione di pressione su una superficie nel tempo e la elabora al fine di trasmettere in uscita un segnale continuo d'intensità variabile fra un valore minimo ed un valore massimo, ovvero un segnale analogico in tensione.

Nella fase di acquisizione di un segnale sonoro giocano un ruolo fondamentale i circuiti elettronici ADC. I convertitori analogico-digitale sono di fondamentale importanza ogni quando si ha la necessità di convertire un segnale analogico con andamento continuo, quindi una tensione analogica ricevuta in ingresso dal circuito, in un segnale a tempo discreto e a valori discreti restituito in output da esso (rappresentato da un numero binario).

Oggigiorno la maggior parte dei dispositivi lavora con segnali digitali.

È questo il motivo per cui è necessario convertire i segnali che di per sé sono analogici, in segnali digitali.

Il primo passo da compiere per ottenere questa conversione prende il nome di nome di **campionamento**, il secondo di **quantizzazione,** mentre il terzo di **codifica**.

Campionamento

L'ADC riceve in ingresso una tensione generalmente compresa fra un valore minimo e un valore massimo, espressa in Volt, derivante da un segnale analogico continuo nel tempo.

Attraverso il circuito campionatore, l'ADC avvia un'operazione di campionamento in cui il segnale viene suddiviso in tanti piccoli campioni che lo rappresentano. Essendo il valore in ogni punto del segnale un numero reale, bisogna discretizzare questi valori per rappresentarlo correttamente. Il segnale campionato è una sequenza di impulsi di tensione distanziati nel tempo di un intervallo costante Ts, cioè il periodo di campionamento. Dopo ogni periodo di campionamento si preleva un campione. L'ampiezza dei campioni dipende dal segnale d'ingresso v(t) secondo la relazione $v_S(t_K) = v(t_K)$, dove t_K è il generico istante di campionamento. Il numero di volte al secondo con cui i campioni vengono riportati in uscita è detto frequenza di campionamento indicata con f_S, è legata al tempo di campionamento dalla relazione $f_S=1/T_K$ e quindi è espressa in Hertz. La frequenza è l'inverso dell'intervallo T tra due istanti che prende il nome di periodo.

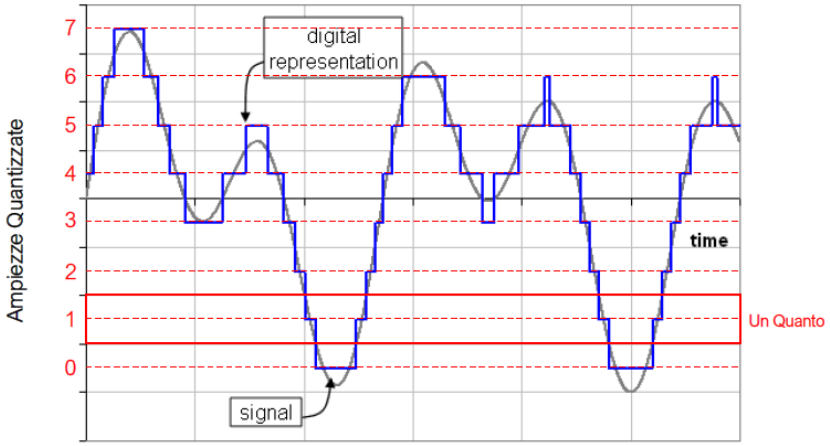
Questa parte risulta delicata perché per realizzare un campionamento corretto e quindi per poi in futuro ricostruire fedelmente il segnale in ogni suo punto, evitando spiacevoli conseguenze, come l'effetto Aliasing, è fondamentale campionare ad almeno il doppio della frequenza di Nyquist.

Per evitare effetti indesiderati e per sistemare quindi gli errori di campionamento viene posto all' ingresso del convertitore analogico-digitale un filtro passa-basso di anti-Aliasing, che lascia passare le basse frequenze, limitando il contenuto spettrale del segnale ricostruito.

Quantizzazione

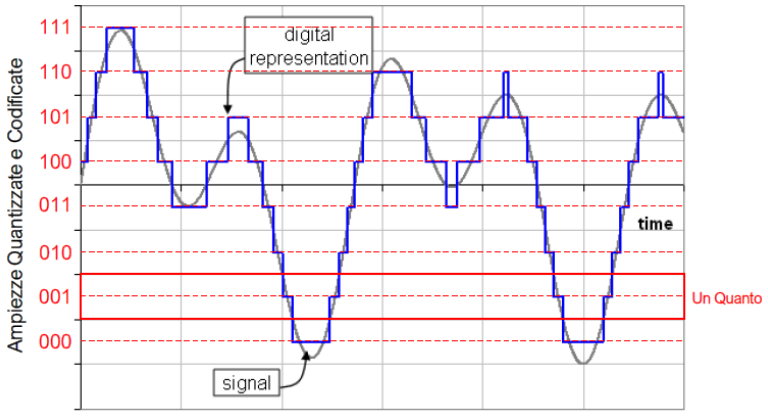
Una volta che il segnale è stato suddiviso in campioni, è necessario avviare un processo attraverso cui i valori vengono approssimati in numeri interi per essere mantenuti all'interno di un certo range.

Tale processo prende il nome di quantizzazione ed è un'operazione che avviene nel dominio delle ampiezze ed è di tipo lossy, ovvero con perdita di informazione. Quindi, in un segnale campionato in uscita ad ogni campione corrisponderà un livello di tensione che permetterà ai campioni stessi di essere codificati digitalmente con n bit. La quantizzazione non è un'operazione reversibile e introduce rumore. Nel particolare, applichiamo una quantizzazione uniforme quando si ha che gli intervalli in cui sta l'insieme dei valori dei campioni sono equispaziati, mentre applichiamo una quantizzazione non-uniforme quando vogliamo ridurre l'errore di quantizzazione prestando più attenzione ai valori più frequenti.



Fase di codifica

Dopo la fase di campionamento di un segnale analogico e quella di quantizzazione segue la fase di codifica della grandezza prima campionata e successivamente quantizzata. La grandezza quindi, rappresentata ancora come tensione, viene trasformata in un numero rappresentato in forma di codifica binaria. Dunque, dati dei livelli di quantizzazione n, saranno riportati in uscita n intervalli, in modo da ottenere in output dall' ADC un codice binario. In particolare, dai contenuti studiati durante il corso, spicca la tecnica di codifica Pulse Code Modulation (PCM) di un audio digitale, famosa per la sua semplicità, che effettuata su un segnale analogico lo codifica in digitale. Essa considera tutti i singoli campioni di un segnale campionato come un impulso, associando ad essi una parola binaria che ne rappresenta l'ampiezza. La codifica PCM viene effettuata su un segnale analogico avente una banda di frequenze inferiore a 4KHz e la lunghezza delle parole binarie è dipendente dai bit di quantizzazione lineare utilizzati.



L' Importanza dei Parametri di un ADC in uno spectrum analyzer

Nel nostro progetto, ovvero in un analizzatore di spettro digitale, ci sono due parametri principali di un ADC che hanno più importanza rispetto agli altri.

Stiamo parlando della risoluzione di misura e dell'accuratezza e precisione.

L'accuratezza caratterizza la dispersione dei valori compresi attorno ad un valore numerico centrale e ci fornisce la misura dello scarto fra due valori: il valore analogico acquisito da uno o più ADC in input e il valore analogico restituito/generato in output. Essa tiene conto degli errori dell'ADC, come ad esempio l'errore di quantizzazione, e viene misurata in Volt o in frazioni LSB (Least significant Bit). Da un'elevata accuratezza segue sempre una elevata precisione ma non è vero il viceversa.

Grazie alla precisione, ovvero il parametro dato da quell'insieme di caratteristiche che assicurano la ripetibilità del valore convertito, un dispositivo è in grado di fornire in output valori il più possibile vicini fra essi.

In un ADC la risoluzione è definita come la più piccola variazione del segnale d'ingresso che il dispositivo è in grado di rilevare in modo affidabile. Essa dipende da il numero di bit B a disposizione dell'ADC per la quantizzazione, dal numero di cifre decimali sul display a disposizione per la visualizzazione dei codici digitali in uscita e dal rumore generato internamente dall'ADC o dal sistema ospitante l'ADC.

È giusto dare ampio spazio anche alla **trasformata di Fourier,** che è risultata di fondamentale importanza per la riuscita del progetto, in quanto è un operatore che ci permette di scrivere una funzione dipendente dal tempo nel dominio delle frequenze.

Detto x(t) il segnale dato nel dominio del tempo, l'operazione di trasformazione si indica con la simbologia

$$X(f) = \mathcal{F}\{x(t)\},$$

e da un punto di vista analitico la trasformata di Fourier si definisce come:

$$X(f) = \int_{-\infty}^{\infty} x(t) e^{-j2\pi ft}\, dt\ .$$

L'esistenza di questa trasformata è garantita per segnali x(t) per i quali risulta

$$\int_{-\infty}^{\infty} |x(t)|\, dt < \infty\ ,$$

cioè per x(t) che sono assolutamente integrabili.

Ad ogni modo nella pratica si preferisce utilizzare la trasformata discreta di Fourier o del Coseno (DCT).

Difatti per i nostri scopi abbiamo fatto ricorso alla trasformata di Fourier veloce (FFT), cioè a un algoritmo che permette di calcolare la trasformata discreta di Fourier (DFT) o la sua inversa.

L'algoritmo FFT più utilizzato è il cosiddetto algoritmo di Cooley-Tukey, il quale riesce a ridurre la complessità computazionale della DFT da $O(N^2)$ (complessità che si viene ad avere quando si applica normalmente la definizione di DFT) a $O(N\ log_2 N)$.

Discrete Fourier transform	$O\left(N^2\right)$
Fast Fourier transform	$O\left(Nlog_2 N\right)$

È proprio per questo che la trasformata di Fourier veloce trova svariate applicazioni nel mondo dell'ingegneria, della musica e della matematica.

La libreria che abbiamo adottato per il calcolo della FFT è reperibile qui → https://github.com/kosme/fix_fft

Grazie a questi strumenti siamo riusciti a realizzare il nostro **analizzatore di spettro**, che fornisce una rappresentazione del segnale in ingresso nel dominio della frequenza.

Però è bene precisare quelli che sono i limiti che presenta il nostro dispositivo.

Nonostante il microfono possa captare tutto il range delle frequenze udibili, il microcontrollore non risulta essere sufficientemente potente da riuscire ad analizzarle tutte senza comprometterne la precisione.

La frequenza di campionamento si ottiene dal clock dell'ADC dividendo per il numero di cicli necessari per effettuare il campionamento, che per l'ATmega328 (che si trova sulla piattaforma Arduino Uno) è pari a 13. La frequenza di clock dell'ADC a sua volta si ottiene dividendo la frequenza di clock del microcontrollore scelto (nel nostro caso 16 Mhz) per il prescaler. Per motivi legati alle prestazioni del dispositivo si è scelto il prescaler di default, ottenendo il valore di 125kHz per il clock dell'ADC e 9,615kHz per la frequenza di campionamento.

Quindi la massima frequenza utilizzabile è pari a circa 4.8kHz, in virtù del teorema di Nyquist. La rappresentazione dello spettro è suddivisa in 64 intervalli, quindi ciascun intervallo coprirà un range di frequenza di ampiezza pari a circa 70Hz.

3. Risultati Ottenuti

“Eppur si muove!!!”.

Con lo sviluppo del seguente codice siamo riusciti a far funzionare il nostro analizzatore di spettro:

```
/* Librerie */
#include "fix_fft.h"                    // Libreria Fast Fourier Transform
#include <Wire.h>                       // Libreria I2C
#include <Adafruit_GFX.h>               // Libreria grafica Display OLED
#include <Adafruit_SSD1306.h>           // Driver Display OLED

/* Definizione costanti */
// Display
#define OLED_ADDR 0x3C                  // Indirizzo I2C Display OLED
#define SCREEN_WIDTH 128                // Width Display OLED
#define SCREEN_HEIGHT 64                // Height Display OLED
// Ingresso campionamento segnale
#define audioIn A0                      // Pin di ingresso segnale audio
```

Queste sono tutte le librerie e le costanti che ci sono tornate utili.

```
/* Istanza del display */
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT);

/* Dichiarazione variabili */
// char - tipo con segno - codifica numeri da -128 a 127
// byte - tipo senza segno a 8 bit - codifica numeri da 0 a 255
const int FFT_N =128;                   // Numero campioni FFT
char re[FFT_N], im[FFT_N];              // array per parte reale ed immaginaria della FFT
byte ylim = 60;                         // Limite di colonne (asse y) del display OLED per titolo

void setup() {
  //Inizializzazione il display OLED con loop in caso di fallita inizializzazione
  if (!display.begin( SSD1306_SWITCHCAPVCC, OLED_ADDR)) {
    while (true);
  }

  // Pulizia del buffer del Display per evitare visualizzazione logo Adafruit e settaggio testo
  display.clearDisplay();
  display.setTextSize(1);
  display.setTextColor(WHITE);
  // Invio comandi al Display OLED
  display.display();
}
```

Nella funzione setup(), che viene eseguita una sola volta, andiamo a inserire la parte “statica” del programma, cioè forniamo alla scheda tutti i dati di cui ha bisogno per operare correttamente.

È in essa che inizializziamo il display, servendoci delle funzioni begin(), clearDisplay() e display(), le quali appartengono alle due librerie Adafruit.

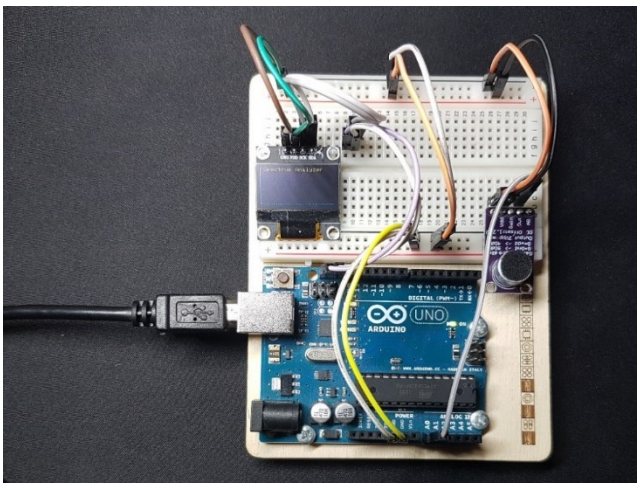
```
void loop() {
   // I dati della FFT vengono immagazzinati in una variabile char con valori da -128 a 127
   // L'ADC (10 bit) restituisce valori da 0 a 1023
   // Per adattare i valori a char si divide il valore di ADC per 4, range (0,255), e si sottrae 128, range (-128,127)

  for (byte i = 0; i < FFT_N; i++) {                  // Legge FFT_N campioni
    int sample = analogRead(audioIn);                 // Lettura dei campioni da ADC
    re[i] = sample / 4 - 128;                         // Adattamento dei campioni al tipo char
    im[i] = 0;                                        // Parte immaginaria dell'array per la FFT posta a 0
  }

  fix_fft(re, im, 7, 0);                              // FFT dei campioni audio - restituisce i valori negli stessi array

  display.clearDisplay();                             // Pulizia del buffer del Display
  display.setCursor(0, 0);                            // Posiziona il cursore nel punto in alto a sinistra del Display
  display.print(" Spectrum Analyzer");                    // Stampa la scritta nella riga superiore del display

  for (byte i = 0; i < FFT_N/2; i++) {
    int dat = sqrt(re[i] * re[i] + im[i] * im[i]);           // L'ampiezza è la radice quadrata della somma dei quadrati di parte reale ed immaginaria
    display.drawLine(i * 2, ylim, i * 2, ylim - dat, WHITE);  // Disegna la barre in corrispondenza a ciascuno dei 64 bin (2 pixel)
  }

  display.display();                                  // Aggiorna display
}
```

La funzione loop(), invece, ci consente di fornire alla scheda Arduino tutte le informazione inerenti all'esecuzione del programma. Ciò che contraddistingue il loop() è che le istruzioni vengono eseguite ripetutamente, proprio come se fosse un while(1) (C-like). È qui che viene dato ampio spazio alla FFT.
Soffermiamoci sulla funzione fix_fft (char fr[], char fi[], int m, int inverse),
inclusa nella libreria fix_fft.h, che prende in input i seguenti parametri:

- fr: dati parte reale
- fi: dati parte immaginaria
- m: indica 2^m 'bins', con m = 7,128 bin di cui solo meta' (64) contengono valori utilizzabili (Nyquist)
- 0: per ottenere la FFT diretta; 1: per ottenere la FFT INVERSA.

e restituisce la FFT dei campioni audio.

In questo modo siamo quindi riusciti a creare a basso prezzo un dispositivo generalmente non alla portata di tutti, in quanto parecchio costoso.
Basterà infatti generare una sorgente sonora nelle vicinanze del nostro Spectrum Analyzer perché esso sarà in grado di captarla, analizzarla e visualizzare a schermo il corrispondente spettro.
Pertanto, avendo a disposizione tutto l'occorrente necessario e il codice sorgente sopra visto, sarà possibile costruire questo analizzatore di spettro in pochi step, ottenendo il seguente risultato:



Riferimenti Bibliografici:

«La Conversione Analogico/Digitale con Arduino» Giulio Tamberi

«Microelettronica» Jaeger-Blalock.

«Elaborazione delle immagini digitali» F.Stanco-S.Battiato.

«I moderni sistemi operativi» Andrew S. Tanenbaum

Sitografia:

https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf

https://en.wikipedia.org/wiki/Fast_Fourier_transform

https://teoriadeisegnali.it/libro/html/html/libro-3.1.html

https://www.circuito.io/