Esame di Stato 2020/2021

Christopher Selvaggio

Tune Tilt:

Accordatore Elettronico per Chitarra con Sistema Robotico di Puntamento Laser Comandato da Note Musicali

Introduzione

Il Tune Tilt è un pedale per chitarra che comanda un braccio robotico. Il suo funzionamento si basa sulla comunicazione fra lo strumento e arduino, al fine di far interagire in modo non banale le diverse componenti del progetto. Il Tune Tilt ha, dunque, la doppia valenza di pedale accordatore comunicante con app e di rilevatore di frequenza in grado di indicare la nota eseguita tramite un braccio meccanico.

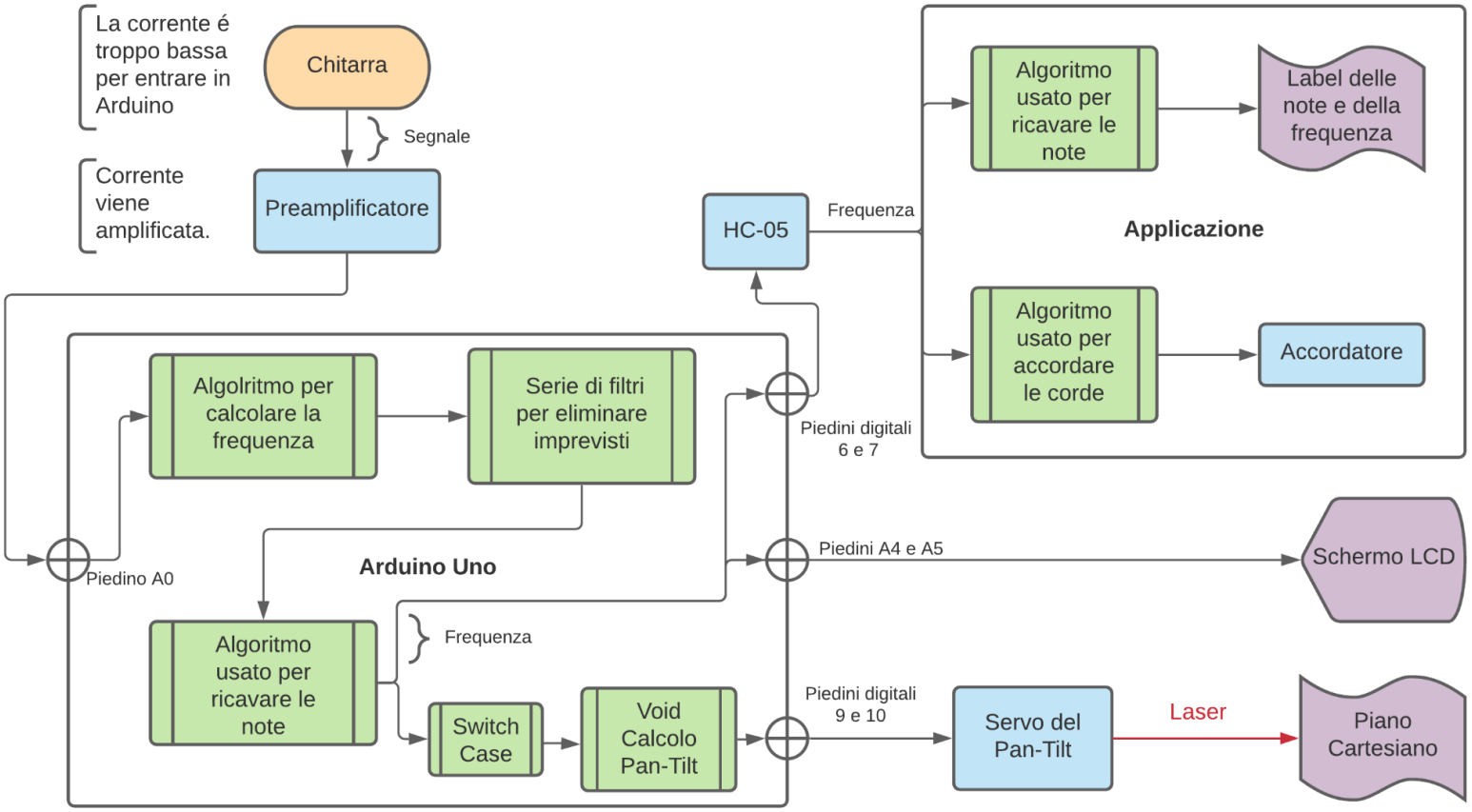
Si illustra ora schematicamente la logica di funzionamento del Tune Tilt e dell'app ad esso collegata. L'input della chitarra viene trasferito al microcontrollore mediante un jack audio per far passare il segnale, questo segnale poi passa attraverso un preamplificatore e un circuito che amplifica il segnale, in modo tale da renderlo acquisibile da Arduino.

Entra in Arduino un segnale analogico, che poi viene trasformato in digitale attraverso l'ADC interno al microcontrollore, da qui viene campionato e si ricava il periodo del segnale. Dal periodo si ricava la frequenza e dunque le note. La frequenza viene elaborata in una serie di if, che mandano ad uno schermo LCD configurato I2C i valori in Hz e le note.

Si puó trovare ogni nota che eseguibile sul manico della chitarra, le note che rientrano nella sensibilità dello strumento dunque vanno da Mi2 a Mi6. Per ogni nota disponibile si impone una variabile char uguale ad una lettera, ogni lettera corrisponde ad un puntamento del pan tilt, ottenuto mediante uno switch case, e ad una casella illuminata dal laser. La frequenza è anche mandata ad un HC-05, che la manda ad una applicazione che quindi ricava i valori della frequenza del segnale della chitarra.

In questa applicazione c'è una funzione di accordatore, dove il segnale passa attraverso una serie di if e se la chitarra è scordata, si vede su una trackbar il valore della frequenza e come accordarla. Nell'applicazione è anche disponibile una versione in Cinese, che traduce tutte le scritte e anche le note.

Schema a blocchi



Lo schema a blocchi è una rappresentazione grafica intuitiva che mostra senza perdita di generalità il funzionamento del Tune Tilt, mostrando la logica della sua progettazione e le caratteristiche tecniche del robot.

Si possono agevolmente notare le diverse componenti del Tune Tilt, dall'ingresso del segnale fino all'elaborazione e alla rappresentazione grafica dello stesso e quella mediante il laser.

Componenti

Il Tune Tilt usa diversi tipi di componenti per realizzare la sua funzione, ecco una lista di tutti i componenti necessari per costruirlo.
Ci sono 3 usi diversi per i componenti:
-Componenti usati per l'alimentazione del circuito preamplificatore e arduino.
-Componenti usati per amplificare il segnale della chitarra in entrata rendendolo idoneo per Arduino
-Apparati di arduino

Alimentazione:

-Spina da pannello
-Trasformatore di tensione doppio secondario
-2 Condensatori elettrolitici da 2200uF
-2 Condensatori elettrolitici da 220uF
-Regolatori di tensione LM7812 e LM7912
-Ponte di diodi
-LM2596 DC-DC Step Down

Preamplificatore:

-2 Jack audio mono
-Amplificatore operazionale TL082CP
-3 resistenze da 100k
-Potenziometro da 10k
-Condensatore elettrolitico da 10uF
-Condensatore ceramico da 47nF
-Zoccolo dip 8 per il TL082CP

Arduino:

-Arduino uno
-2 Servomotori
-HC-05
-Schermo LCD con IIC I2C

Altri Componenti:

-Saldatore
-Cavi elettrici
-Millefori
-Guaine termorestringenti
-Biadesivo
-Nastro isolante

Spiegazione Componenti

Condensatore

Il **condensatore** è un componente elettrico che ha la capacità di immagazzinare la corrente, e ha diversi usi, come ad esempio quello di batteria.

Nella teoria dei circuiti il condensatore è un componente *ideale* che può mantenere la carica e l'energia accumulata all'infinito.

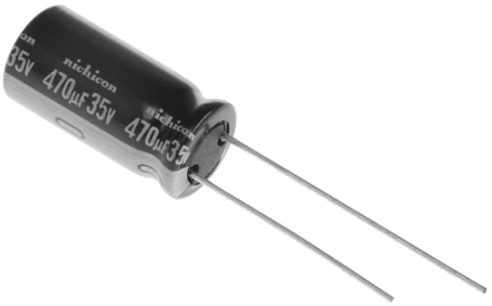
Vengono usati principalmente due tipi di condensatori:

Dielettrico solido

Il condensatori a dielettrico solido possono essere di diversi tipi; i piú popolari sono quelli ceramici.

I condensatori ceramici sono usati per la loro bassa induttanza parassita e per via delle ridotte dimensioni.

Elettrolitico



Il condensatore elettrolitico è il condensatore piú usato nei circuiti.

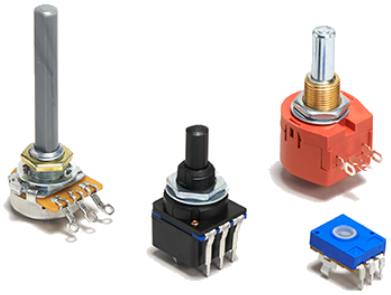
Nei condensatori elettrolitici l'isolamento è dovuto alla formazione di un sottilissimo strato di ossido metallico sulla superficie di una armatura a contatto con una soluzione chimica.

I condensatori elettrolitici possono essere di diversi materiali, come ad esempio, di alluminio, tantalio e anche di aerogel di carbonio.

Resistenza

La **resistenza elettrica** è una grandezza fisica scalare che misura la tendenza di un corpo ad opporsi al passaggio di una corrente elettrica, quando sottoposto ad una tensione elettrica. Questa opposizione dipende dal materiale con cui è realizzato, dalle sue dimensioni e dalla sua temperatura.

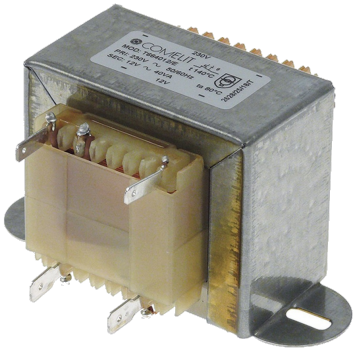
Potenziometro



Il **potenziometro** è un dispositivo elettrico equivalente ad un partitore di tensione resistivo variabile, infatti una sua parte viene disposta in parallelo al carico utilizzatore. Sostanzialmente è una resistenza al quale si puó variare il valore.

É costituito da un cilindro isolante su cui è fittamente avvolto un filo metallico con resistività opportuna, mentre le due estremità sono connesse a due morsetti. Scorre un cursore sul cilindro da un'estremità all'altra, recante un contatto strisciante sul filo, a sua volta collegato ad un morsetto.

Trasformatore



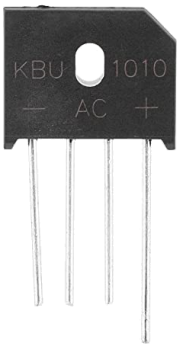
Il **trasformatore** è una macchina elettrica statica alimentata a corrente alternata, basata sul fenomeno dell'induzione elettromagnetica, destinata a trasformare, tra il circuito primario e il circuito secondario del trasformatore, i fattori tensione e corrente della potenza elettrica. Trasferisce quindi energia elettrica da un circuito a un altro con una tensione diversa, accoppiandoli induttivamente.

Un trasformatore a corrente elettrica, nell'avvolgimento primario genera un flusso magnetico variabile, e di conseguenza un altro campo magnetico variabile attraverso l'avvolgimento secondario, questo succede attraverso la legge di Faraday-Neumann. Questo campo magnetico variabile induce una tensione nell'avvolgimento secondario. Questo effetto è chiamato mutua induzione.

Il trasformatore è una macchina in grado di operare essenzialmente in corrente alternata, perché in genere sfrutta i principi dell'elettromagnetismo legati ai flussi variabili. Questo crea un componente con un rendimento molto alto e con perdite sono molto basse.

Se un carico elettrico è collegato all'avvolgimento secondario, una corrente elettrica vi scorre e l'energia, tramite il trasformatore, viene trasferita dal circuito primario al carico. In un trasformatore ideale, la tensione indotta nell'avvolgimento secondario è proporzionale alla tensione primaria, ed è data dal rapporto fra il numero delle spire dell'avvolgimento primario e il numero di spire dell'avvolgimento secondario.

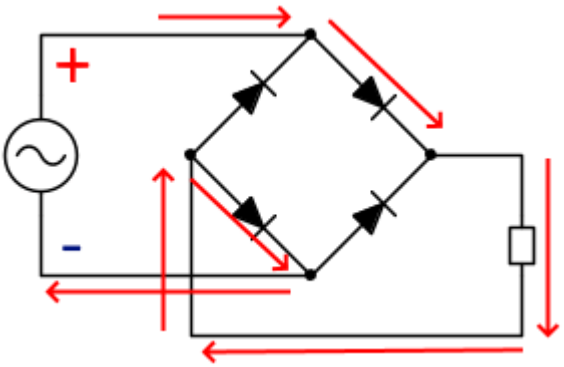
Ponte di diodi

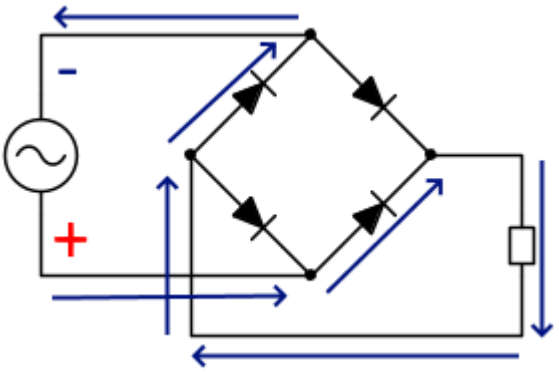


Il **raddrizzatore** è un dispositivo che serve a trasformare un segnale alternato in uno unidirezionale. Questa operazione si chiama *raddrizzamento*.

Il suo funzionamento deriva da quello del diodo. Nei circuiti alimentati a tensione alternata, esso consente il passaggio della corrente dall'anodo verso il catodo con una piccola caduta di tensione, mentre blocca il passaggio della corrente in senso inverso. Si può quindi usare il diodo a circuito chiuso nel caso delle correnti dirette, e ad un circuito aperto per le correnti inverse.

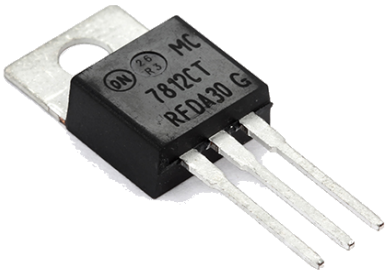
In elettronica trova applicazione pratica come trasformatore di forma d'onda: è infatti usato per trasformare la corrente elettrica da alternata in continua.





Qui vediamo il funzionamento del ponte di diodi con una corrente alternata, come cambia il passaggio della corrente attraverso i diodi e appunto come riesce a raddrizzare la sinusoide.

LM78XX e LM79XX



Un regolatore di tensione positiva, anche conosciuto come stabilizzatore di tensione, della serie 78xx, più precisamente 7812, è un componente elettronico integrato in grado di convertire una tensione continua in un'altra di più basso valore. Tutto questo vale anche per gli stabilizzatori di tensione negativi, gli LM79XX.

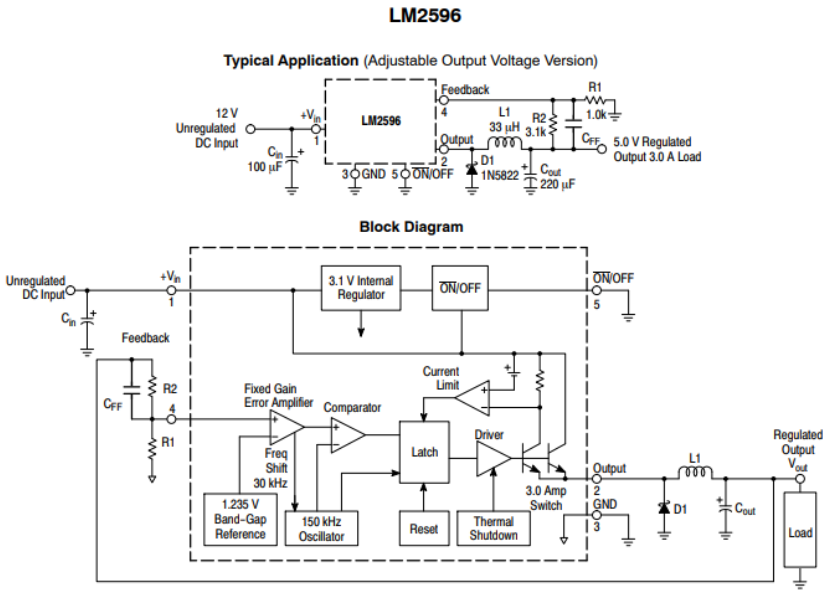
La loro nomenclatura è molto semplice. LM78xx sono per tensioni positive mentre LM79xx sono per tensioni negative; i numeri al posto delle “x” sono il valore della tensione che dará in uscita. Per esempio se abbiamo un LM7809 avremo una tensione di uscita pari a +9V.

DC-DC Step-Down

Questo è un modulo regolatore di tensione switching step-down con chip LM2596s.

Consente di regolare con precisione una tensione in ingresso variabile da 3 a 40V ad una in uscita da un minimo di 1.25V ad un massimo di circa 37V.

La tensione erogata puó essere regolata attraverso il trimmer, un potenziometro, che a seconda della posizione modifica il valore dell’uscita.



Sopra si vede il circuito del DC-DC Step-Down che regola la corrente e il chip che amplifica e regola le tensioni. Vediamo che sono usati dei comparatori e degli amplificatori operazionali.

Amplificatore Operazionale

In elettronica, un **amplificatore operazionale**, in inglese *operational amplifier* oppure *op-amp*, è un amplificatore differenziale con una o piú uscite. Grazie anche alla sua versatilità, è uno dei dispositivi più vastamente utilizzati sia in ambito commerciale,che scientifico, in particolare nei circuiti analogici e in circuiti riguardanti segnali acustici.

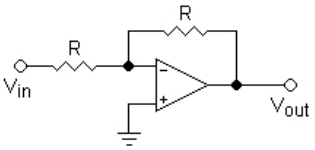
La maggior parte degli amplificatori operazionali è progettata per lavorare con una tensione di alimentazione duale, cioè con un valore positivo e uno negativo, simmetrici rispetto a ground. Nell'alimentazione duale, il livello del segnale in uscita può spaziare tra i due valori di tensione d'alimentazione a meno di un piccolo margine, che può variare a seconda del tipo di operazionale adottato.

L'amplificatore operazionale è un dispositivo a elevato guadagno il cui utilizzo è permesso dall'aggiunta di una rete di retroazione, che collega l'uscita all'ingresso in modo tale da diminuire o aumentare il valore del segnale in entrata, in funzione della configurazione dell’amplificatore. In questo modo il comportamento del dispositivo non dipende dal particolare valore del suo guadagno, ovvero il guadagno ad anello aperto, ma soltanto dalle caratteristiche della rete di retroazione.

Negli amplificatori un ingresso è detto *invertente* ed è indicato con il simbolo −, l'altro è detto *non invertente* ed è indicato con il simbolo +. Utilizzando un'opportuna configurazione, inoltre, l'operazionale viene utilizzato anche come amplificatore di corrente.

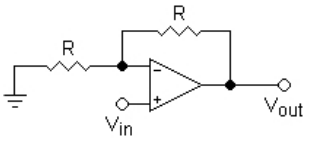
Il funzionamento dell'amplificatore operazionale si basa sull'elevato valore del guadagno ad anello aperto. In altre parole, l'amplificatore fa in modo che la rete di retroazione mantenga minima la differenza di potenziale tra i due ingressi, così da avere un guadagno limitato e rendere utilizzabile il dispositivo. Ci sono diversi tipi di configurazioni per gli amplificatori operazionali, eccone alcune.

Invertente e Non invertente



Configurazione Invertente

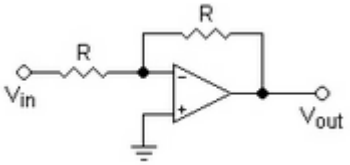
$G=-\frac{R2}{R1}$



Configurazione Non Intvertente

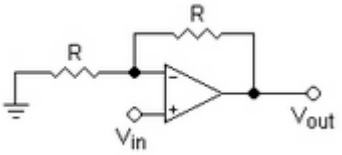
$G=1+\frac{R2}{R1}$

La configurazione invertente si ottiene applicando la tensione di ingresso V in sul morsetto invertente e mantenendo l'altro morsetto al ground, come nella figura a sinistra.



In questa configurazione la retroazione negativa diminuisce il valore della corrente in ingresso, e pertanto l'impedenza in ingresso, ovvero misurata tra i due ingressi, è data dall'impedenza dell'amplificatore senza retroazione divisa per un fattore di correzione, che motiva l'assunzione di un impedenza in ingresso infinita nel caso ideale, in cui il guadagno è infinito. Allo stesso modo, il valore dell'impedenza in uscita dipende dal fatto che la retroazione riporta all'ingresso una parte della corrente in uscita.

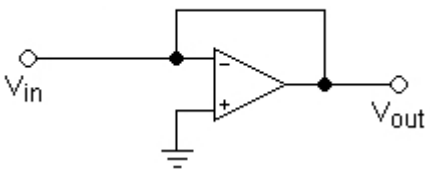
Un amplificatore non invertente si ottiene applicando la tensione d'ingresso V in sul morsetto non invertente e mantenendo l'altro morsetto a massa attraverso una resistenza. L'uscita è inoltre in fase con l'ingresso. Le due resistenze formano un partitore di tensione.



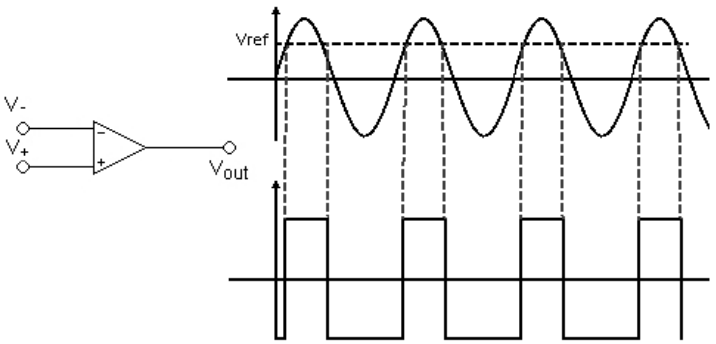
In tale configurazione la retroazione negativa diminuisce il valore della tensione in ingresso e pertanto l'impedenza in ingresso è data dall'impedenza dell'amplificatore senza retroazione, che motiva l'assunzione di un impedenza in ingresso infinita nel caso ideale, in cui anche il guadagno è infinito.

Il buffer

Ci sono anche diverse configurazioni usate di meno, come ad esempio **il Buffer**, che a causa dell'elevata impedenza di ingresso e la piccola impedenza in uscita viene spesso impiegato come disaccoppiatore di ingresso e uscita tra circuiti.



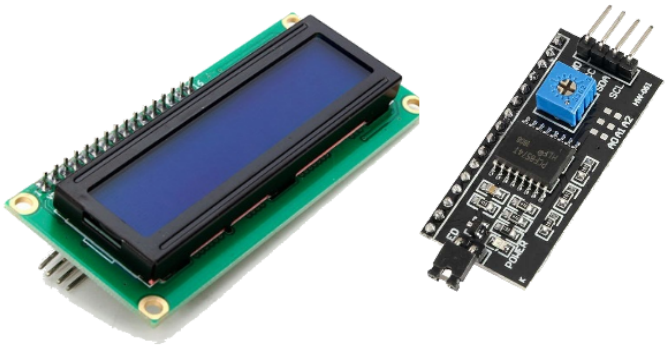
Il comparatore



Un altro operazionale che abbiamo visto è stato il **Comparatore.** Il comparatore è un circuito capace di fornire in uscita una tensione continua, che può assumere soltanto due livelli, e segnalare così una particolare condizione di disuguaglianza tra le due tensioni di ingresso.

Quando si applicano due segnali da confrontare ai due ingressi, l'uscita assumerà un valore di tensione prossimo alla tensione positiva di alimentazione, mandandolo in saturazione, se l'ingresso non invertente ha tensione maggiore dell'ingresso invertente. Nel caso opposto l'uscita presenterà una tensione prossima all'alimentazione negativa.

Display LCD con interfaccia seriale IIC I2C



I **display a cristalli liquidi** è un tipo di display che utilizza le proprietà di modulazione della luce dei cristalli liquidi.

Piccoli LCD sono comuni anche in diversi dispositivi portatili di consumo, per esempio in fotocamere digitali, orologi e smartphone. I display a cristalli liquidi possono avere dimensioni che variano da poche decine di millimetri a oltre 100 pollici. Si utilizza in questo progetto un display 16x2, ossia avente due righe per 16 colonne, dove ogni colonna puó mostrare un carattere.

Per configurare piú facilmente lo schermo lcd usiamo un Adattatore IIC I2C.
Si tratta di un adattatore dal bus seriale I2C al bus parallelo utilizzato dai Display LCD con matrice 16x2. Ad esempio un display LCD 16x2 impegna per il suo controllo, almeno 6 porte del Microcontrollore. Questo modulo permette di comunicare con un display LCD mediante il protocollo I2C che impegna solo due porte.

Servomotore Digitale

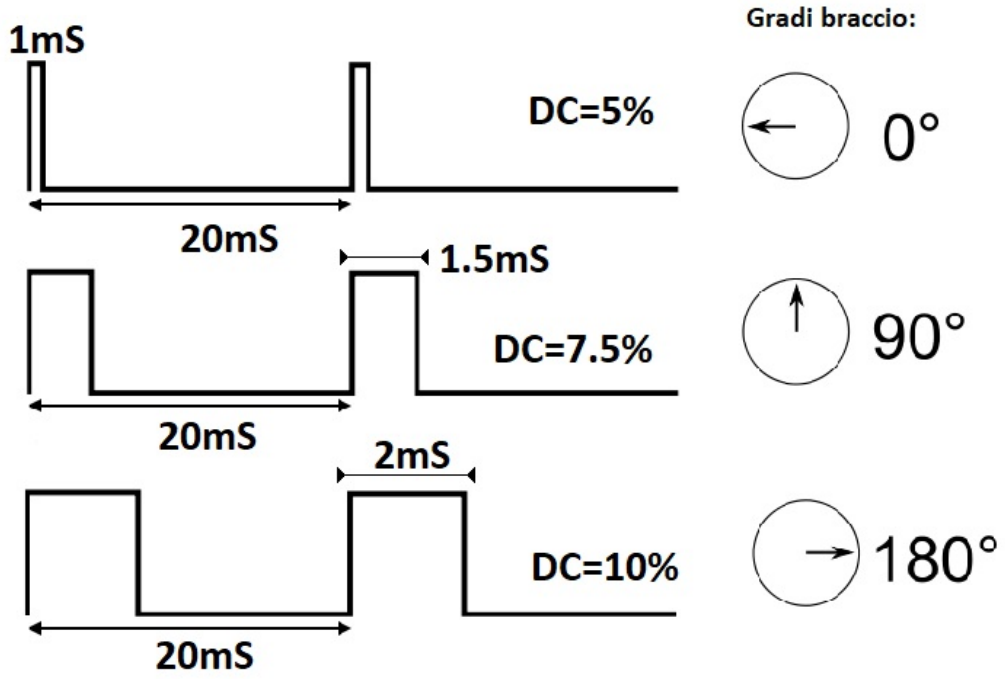
Un **servomotore** è un particolare tipo di motore che si differenzia dai motori tradizionali in quanto le sue condizioni operative sono soggette ad ampie e spesso repentine variazioni. Il **servomotore** si deve adattare a queste variazioni con la massima rapidità e precisione. Può essere elettrico, pneumatico, idraulico.

I **servomotori** trovano applicazione nei controlli di posizione, nelle macchine a controllo numerico, nei sistemi automatici di regolazione e nelle periferiche di sistema, come stampanti.

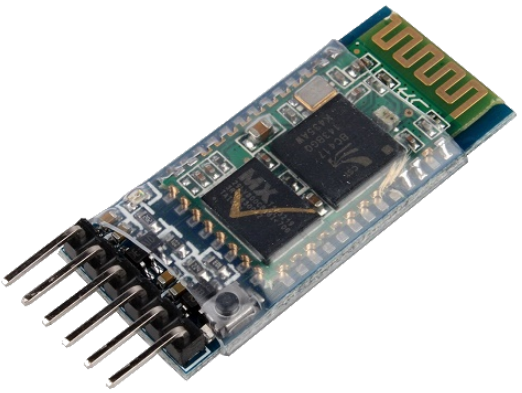
Si presentano come piccoli contenitori di plastica da cui fuoriesce un perno in grado di ruotare in un angolo compreso tra 0 e 180° mantenendo stabilmente la posizione raggiunta. La rotazione del motore è effettuata tramite un circuito di controllo interno in grado di rilevare l'angolo di rotazione raggiunto dal perno tramite un potenziometro resistivo e bloccare il motore sul punto desiderato.

Sotto si puó vedere come funzionano questi tipo di servomotori, ovvero attraverso il duty cycle del segnale PWM. Il duty cycle è la frazione di tempo che un'entità passa in uno stato attivo in proporzione al tempo totale considerato.

La formula per il Duty Cycle è: $$DC\% = \frac{Ton}{T} * 100$$



HC-05



Il modulo **HC-05** è un modulo che permette di convertire una **porta seriale UART** in una **porta Bluetooth**; questo dispositivo può essere utilizzato per permettere la comunicazione tra un microprocessore, montato ad esempio su una scheda Arduino, ed un dispositivo dotato di comunicazione Bluetooth, come ad esempio uno smartphone.

L'HC-05 usa i piedini **TX ed RX** per mandare e ricevere informazioni dalla seriale bluetooth creata nel codice.

La caratteristica principale del modulo HC-05 è la possibilità di essere impostato, tramite i comandi AT, in due modalità: come dispositivo Master o dispositivo Slave; questo permetterà di utilizzare lo stesso tipo di modulo per creare una rete di dispositivi.

Per impostare il modulo **HC-05** come dispositivo Master o Slave e per effettuare altre operazioni come variare il nome o il baud rate, è necessario attivare la **modalità AT**; questo è possibile attraverso un programma apposito per la configurazione dell'**HC-05**

```
#include <SoftwareSerial.h>
SoftwareSerial BTSerial(10, 11); // RX | TX
void setup()
{
 pinMode(9, OUTPUT);  // this pin will pull the HC-05 pin 34
 digitalWrite(9, HIGH);
 Serial.begin(9600);
 Serial.println("Enter AT commands:");
 BTSerial.begin(38400);  // HC-05 default speed in AT command more
}
void loop()
{
 // Keep reading from HC-05 and send to Arduino Serial Monitor
 if (BTSerial.available())
 Serial.write(BTSerial.read());
 // Keep reading from Arduino Serial Monitor and send to HC-05
 if (Serial.available())
 BTSerial.write(Serial.read());
}
```

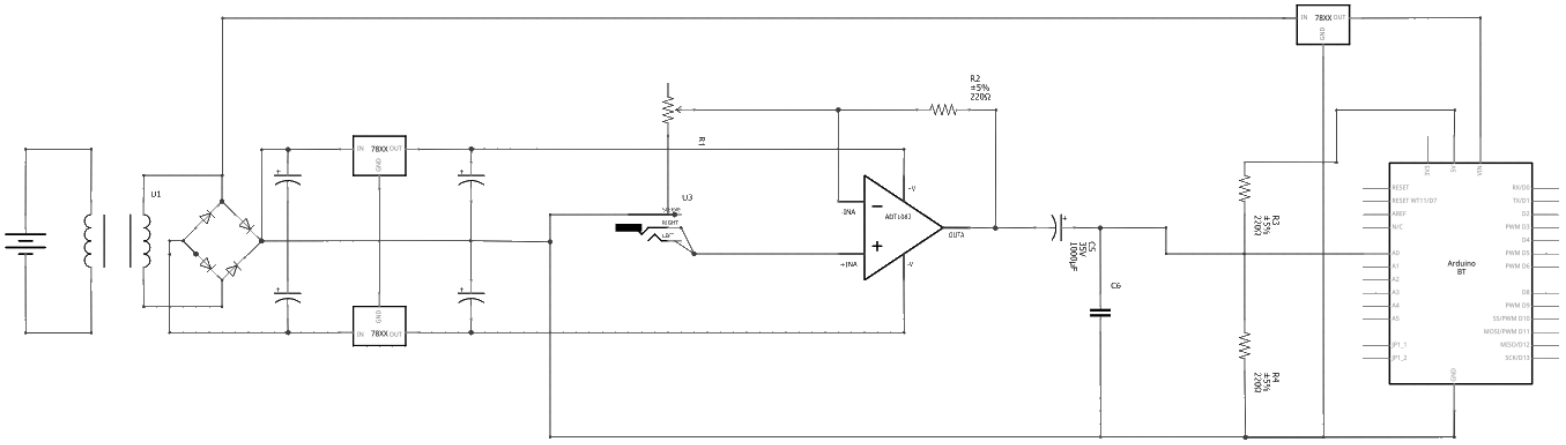
Ci sono diversi **comandi AT,** come ad esempio:

AT+VERSION, un comando che mostra la versione della scheda.

AT+NAME=..., comando usato per cambiare il nome all'HC-05.

AT+UART=..., comando usato per cambiare il Baud Rate della scheda.

Circuito



Il circuito per il Tune Tilt è abbastanza complesso, si divide in 3 parti diverse le quali hanno ognuna un funzionamento diverso.
Ogni parte del circuito ha un compito specifico ed essenziale per il funzionamento del progetto.

- Trasformatore

Il circuito del Trasformatore è il cuore del progetto, è una parte essenziale del circuito poiché è lui ad alimentare ogni altro apparato del Tune Tilt. Senza di esso il preamplificatore non avrebbe abbastanza corrente per funzionare. Il suo ruolo fondamentale è quello di supplire alla mancanza di batterie: il Tune Tilt non ha bisogno di essere ricaricato ma è necessario solo collegarlo alla corrente elettrica mediante una presa.

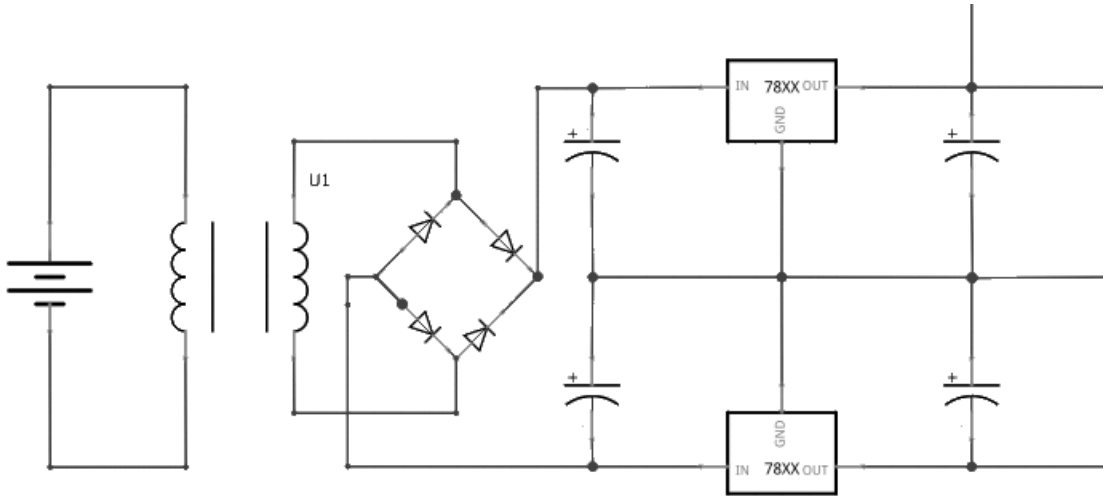
- Preamplificatore

Il preamplificatore è la parte del circuito che consente al segnale in entrata di essere amplificato, analizzato e trasformato in un valore utilizzabile da arduino: altrimenti la tensione sarebbe troppo bassa per essere letta dal microprocessore. A tale scopo il preamplificatore fornisce al segnale un offset di 2.5 V.

- Connessioni ad Arduino

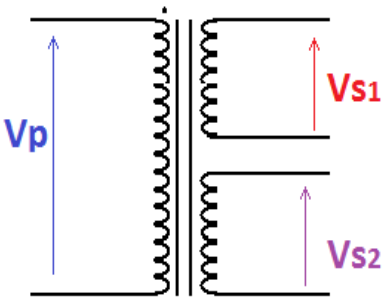
Le diverse connessioni ad arduino controllano le diverse parti del circuito, come ad esempio i piedini al modulo bluetooth, allo schermo e al braccio meccanico. In questo modo si garantisce l’efficacia del Tune Tilt a svolgere il suo compito.

Trasformatore

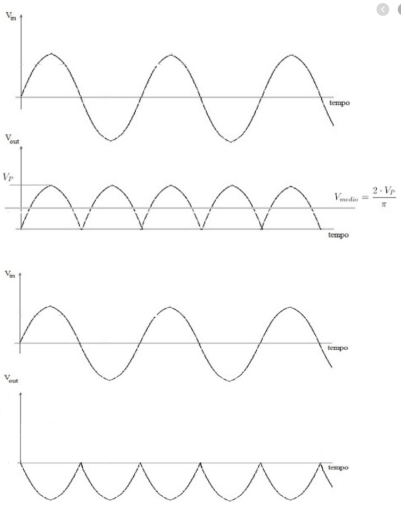


Il circuito del preamplificatore originariamente era alimentato usando due pile da 9V in serie dando cosí una tensione da 18Vpp. Per rendere il circuito piú uniforme e compatto si è deciso di sostituire le due batterie usando un trasformatore duale doppio secondario, che emula le due pile creando due “loop” di tensione diversi.

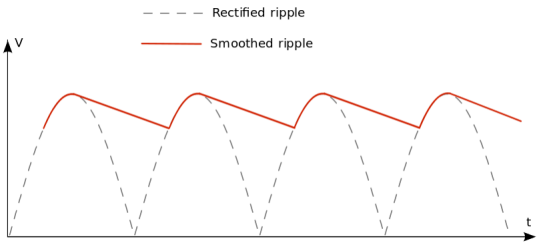
Il trasformatore usato ha 3 bobine; una in entrata dove passano i 230V, e 2 in uscita dove escono 15V di alternata in ognuna. La bobina iniziale crea un campo magnetico che viene percepito dalle due bobine in uscita interagiscono con il campo magnetico del primo avvolgimento inducendo una uscita in tensione alternata. La quantitá di avvolgimenti determina la quantitá di tensione in uscita, grazie a questo poi si possono regolare i valori del trasformatore.



Dopo essere uscita dal trasformatore, entra in un ponte raddrizzatori a diodi. Un diodo elimina la parte positiva o negativa di una semionda, in questo caso è presente un ponte raddrizzatore che prende la corrente alternata che esce dal trasformatore e la trasforma in due modi, in uscita dal piedino “+” la semionda negativa viene ribaltata all’insú eliminando la parte negativa e sul piedino “-” la parte positiva viene ribaltata in giú, visto nelle figure a destra.



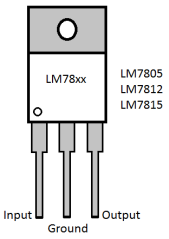
Le due correnti poi passano attraverso dei condensatori da 2200uF in parallelo che servono per ridurre il ripple. Senza i condensatori, dentro ai due regolatori di voltaggio entrerebbe una tensione intermittente, essendo “alta” soltanto la semionda raggiunge il massimo del segnale.
Il condensatore fa sì che quando la semionda va ad alimentare il circuito e la semionda è “giú” usa la carica accumulata dentro di esso per caricare il circuito, e crea una sorta di corrente in continua.

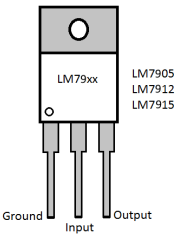


Il ripple

「I **ripple** è considerato come l'ampiezza del segnale successivo ad un raddrizzamento all'interno dell'alimentatore. Questo segnale è sovrapposto alla tensione continua desiderata, per cui si cerca di eliminarlo, in modo da ottenere una tensione con un andamento più lineare possibile. Questo segnale non è mai del tutto eliminabile, ma è possibile costruire dei filtri in grado di ridurne l’effetto.
Negli alimentatori il **ripple** è ridotto in gran parte dal condensatore di filtro. Infatti, teoricamente, per ottenere un ripple zero da questo semplice circuito sarebbe necessario un condensatore di capacità infinita, per cui il ripple è considerato un difetto dal quale è impossibile liberarsi. 」

Dopo aver ridotto il ripple, la tensione positiva e quella negativa entrano rispettivamente dentro a dei regolatori di tensione LM7812 e LM7912, che abbassano la tensione a +12V e -12V portandola ad un valore adatto per alimentare il circuito preamplificatore.



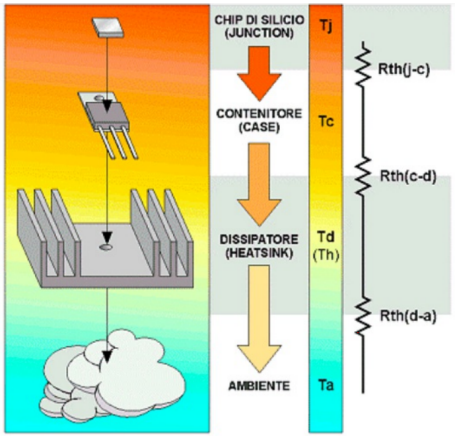


Dissipatore

「Solitamente quando passa corrente attraverso i regolatori di tensione, tendono a riscaldarsi, per questo si è attaccato un dissipatore a quello integrato. Un **dissipatore**, è un dispositivo che consente l'abbassamento della temperatura dei componenti elettronici che emanano calore per effetto Joule, come transistor e processori. La temperatura limite dei dispositivi elettronici si aggira attorno ai 150°C e senza dissipatori potrebbe essere raggiunta molto prima del previsto.

$$P_D = \frac{1}{R_{th\text{-}ja}} \cdot (T_j - T_a)$$

La formula per trovare la potenza dissipata di un dissipatore è questa, si relaziona la differenza della temperatura del componente con la temperatura d’ambiente, con il valore della resistenza termica del dissipatore.



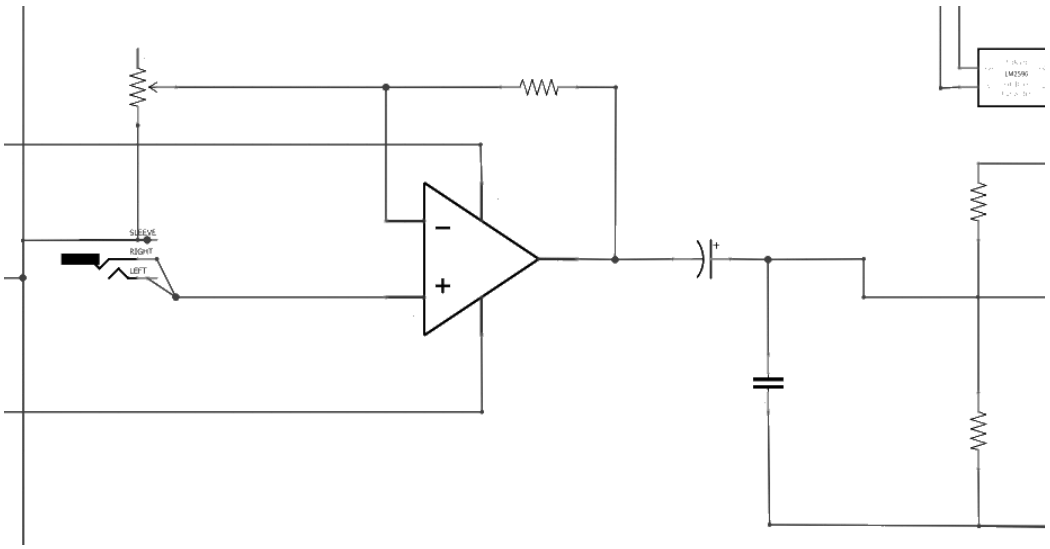
Per calcolare il valore della resistenza termica, quando si applica un dissipatore, si deve tenere conto di tutte le giunzioni che il calore deve attraversare per essere dissipato.

Quando la giunzione non è connessa a nessun tipo di dissipatore, la resistenza termica è (j-a) ovvero Junction-Air, mentre in questo circuito, se si dovesse calcolare la potenza dissipata, si dovrebbe fare la somma di tutte le resistenze termiche e poi calcolare la potenza dissipata da essa. 」

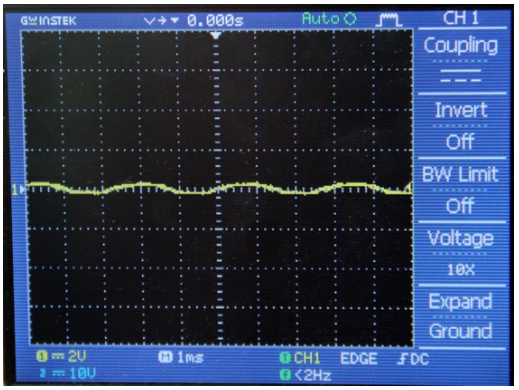
La corrente poi passa ad altri due condensatori da 220uF in parallelo, questi condensatori sono piú piccoli di quelli usati per eliminare il ripple, e hanno il compito di proteggere il circuito nel caso se ci fosse una scossa di corrente imprevista che potrebbe andare a danneggiare lo stesso.

Questo circuito emula due batterie in serie, dove la prima batteria è tra il **+** sono i +12V in uscita dal 7812 e il **-** la linea centrale collegata a ground, mentre per la seconda batteria il **+** è il ground e il **-** sono i -12V in uscita dal 7912. Questo verrá poi usato per alimentare il nostro amplificatore operazionale TL082CP, e, attraverso un DC-DC step down che ridurrá la corrente a 9V, anche arduino.

Preamplificatore



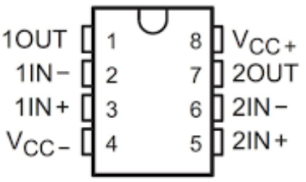
Il preamplificatore è la parte piú importante per l'acquisizione del segnale, ed ecco perché:



Il segnale di una chitarra elettrica non amplificata è intorno ai 100 mV, un segnale estremamente basso che non verrebbe percepito da arduino se fosse mandato in un pin analogico.

Normalmente il segnale di una chitarra viene amplificato da un amplificatore, che da molte piú opzioni un semplice op-amp.

Piedinatura TL082CP

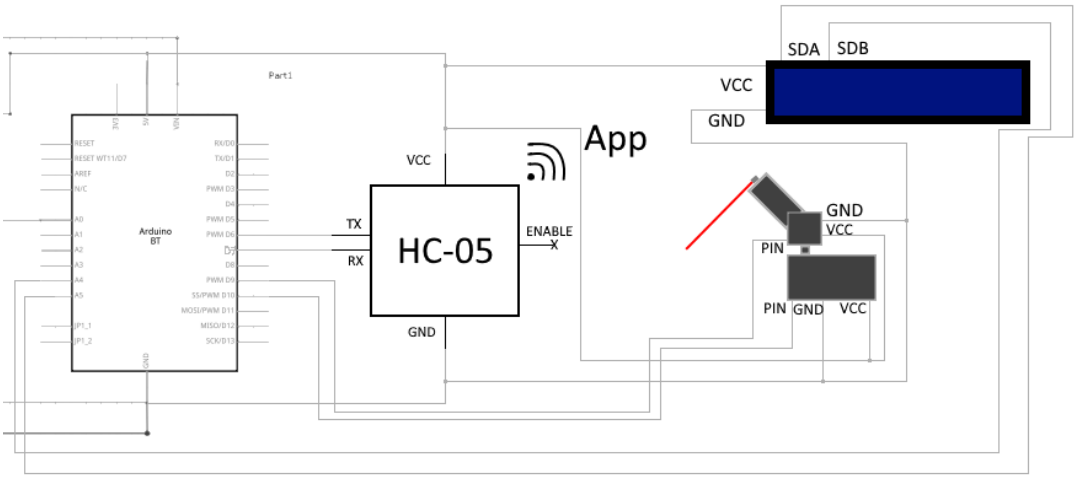


Per questo usiamo l'amplificatore operazionale TL082CP, che è alimentato al piedino 8 e 4, dai +12V e -12V in uscita dal circuito del trasformatore, in configurazione non invertente per amplificare il segnale.

Dopo essere uscito dall'operazionale, il segnale attraversa dei condensatori che sono messi per ridurre il DC bias e ridurre il rumore del segnale amplificato. Da qui il segnale entra in un partitore di tensione tra due resistenze di 100k danno ad esso un offset di 2.5V, abbastanza per rimuovere la parte negativa del segnale rendendolo accettabile da arduino.

Dopo essere passato attraverso i vari componenti del circuito il segnale va nel piedino A0, dove verrá poi analizzato attraverso una serie di formule e di operazioni che verranno spiegate poi nel codice.

Connessioni ad Arduino



Dopo che il segnale è stato analizzato e trasformato da arduino, viene mandato a 3 diversi apparati esterni, un HC-05, uno schermo LCD e un pan tilt che punterá su una griglia incollata sulla scatola.

Bluetooth

L'HC-05 è connesso ai piedini digitali 6 e 7 **il TX e l'RX** della seriale bluetooth che è stata dichiarata ed è alimentata con i 5V e GND di arduino.
Il Tune Tilt invia tramite bluetooth, dal microcontrollore, all'applicazione la frequenza processata, e l'applicazione non solo sarà in grado di individuare la frequenza, ma anche di informare lo strumentista su quale corda a vuoto sta suonando e sul modo opportuno di correggere l'accordatura per intonare lo strumento.

Pan Tilt

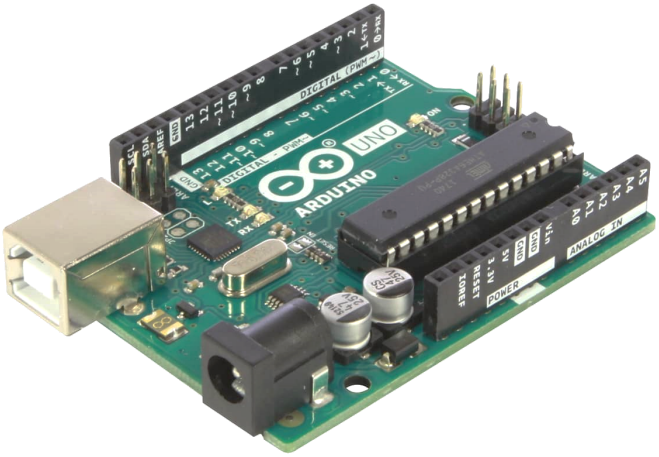
Il pan e il tilt sono connessi rispettivamente ai piedini digitali 9 e 10 per il loro controllo, e sono alimentati attraverso un LM7805 con in parallelo un condensatore per compensare il calo di tensione che si ha sui servo ogni volta che si accendono.
Il pan tilt punta su una griglia 3 x 4 che contiene tutte le note, e a seconda della nota punta su una casella diversa. Il raggio laser sarà dunque in grado di identificare la casella corrispondente alla nota suonata ed illuminarla in un tempo idoneo per accordare lo strumento o individuare la nota eseguita.

Display LCD

E il display LCD è collegato ai piedini analogici A5 e A4, rispettivamente i piedini SDB e SDA ovvero i due bus seriali che mandano le informazioni allo schermo. Qui vengono mandati i dati del segnale della chitarra, Il valore della nota in Hertz e la nota in sé.

Sketch Arduino

Cos’è arduino?

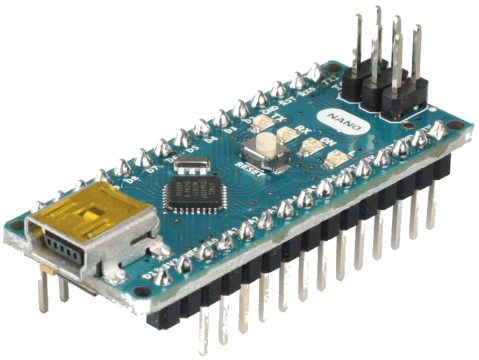


Arduino è una scheda programmabile che viene usata per scrivere e caricare codice informatico in linguaggio “C” nella scheda stessa.

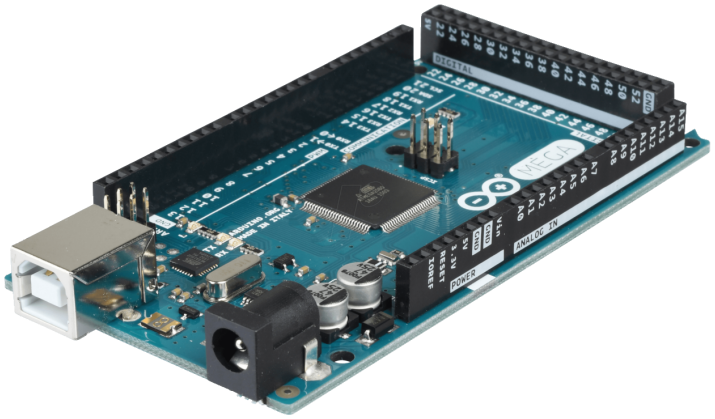
È un dispositivo che da adito a migliaia di creazioni, e in questo progetto è il cuore di tutto.

In questo progetto viene usato Arduino Uno, ma ci sono diverse versioni di Arduino, ognuna con diverse utilità.

Come ad esempio Arduino Nano, una scheda arduino che ha delle dimensioni ridotte ma che ritiene le stesse qualitá, rendendolo piú adatto per progetti compatti e leggeri.



Arduino Nano aumenta la quantitá di piedini da dall’Uno e al posto della porta USB di tipo B, usa una porta di tipo micro-B, che è piú moderna.

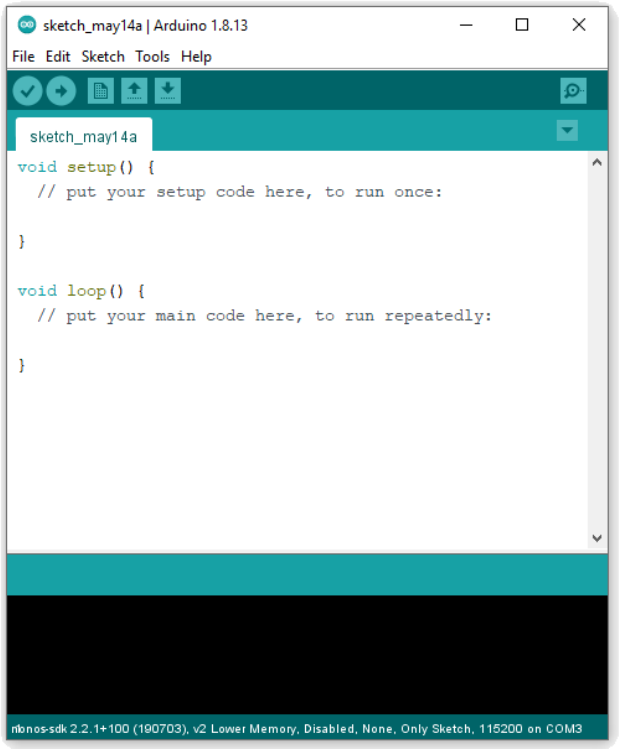


Nell’altra direzione abbiamo Arduino Mega, una versione piú grande e con piú potenza rispetto alla versione Uno.
Arduino mega usa un chip diverso che rende la computazione piú veloce, ha una memoria 8 volte piú grande di Arduino Uno. Vanta anche di 54 pin di ingresso e di uscita di cui 16 analogici e 15 dei quali possono gestire la PWM. Ritiene dal fratello piú piccolo peró la porta USB, di tipo B.

Come si programma arduino?

Arduino puó essere programmato attraverso l'IDE di arduino.

L'Ambiente di Sviluppo Integrato, o IDE, di Arduino è un'applicazione multipiattaforma scritta in Java, dove si scrivono programmi per il microcontrollore nella lingua "C++". Il software è in grado di compilare e caricare il programma funzionante sulla scheda.



I programmi scritti utilizzando il software di Arduino sono chiamati sketch.
Nella barra nera in basso viene dato del "feedback" su cosa sta succedendo e vengono segnalati gli errori.
In alto a sinistra si puó aprire la seriale, dove si possono vedere diverse informazioni.
I pulsanti della barra degli strumenti consentono di verificare e caricare programmi, creare, aprire e salvare schizzi e aprire il monitor seriale.

Nel maggior parte dei casi ci sono due void principali nell'IDE di arduino: Il void loop e il void setup.

- Nel **void setup** si da alla scheda tutte le informazioni necessarie per l'esecuzione del programma, come ad esempio impostare come output o input diversi piedini oppure inizializzare una seriale.

- Nel **void loop** si fornisce alla scheda tutte le informazioni per l'esecuzione del programma. La particolarità del **void loop** è che i comandi vengono ripetuti sempre in ordine, e quando il codice termina, il **void loop** riparte da capo.

Per caricare un programma basta cliccare sul bottone "Upload" in alto a sinistra oppure fare "Ctrl + U".

Ci sono diversi comandi e funzioni usati nella programmazione del codice:

If

La funzione If(), controlla e verifica una condizione impostata da noi ed esegue una formula dichiarata sotto se la condizione viene soddisfatta.

```
if (condition) {
  //statement(s)
}
```

Quando la condizione viene soddisfatta il codice impostato sotto viene eseguito.

Ciclo For

La funzione for(); viene usata per ripetere una serie di formule per una quantitá di volte determinata dal programmatore.
Solitamente viene usata una variabile per incrementare e terminare il ciclo.

```
for (initialization; condition; increment) {
  // statement(s);
}
```

Initialization solitamente è la variabile che viene usata per incrementare il conto del loop, come per esempio ***int I = 0;***
riprodotto il ciclo, la variabile viene controllata: se è una variabile accettabile, allora viene incrementata, e il codice viene eseguito. Questo finché non è piú una variabile
Condition è la condizione che impostiamo per il nostro loop. Ogni volta che viene accettabile, cioè quando il loop finisce.

Ciclo While

La funzione while(); permette di creare un ciclo continuo e infinito finché la condizione dichiarata dentro ad esso diventa falsa.

```
while (condition) {
  // statement(s)
}
```

La condizione solitamente puó essere o vera o falsa, essendo una funzione booleana.

Dichiarazione Variabili e Librerie:

```
#include <LiquidCrystal_I2C.h>     //Librerie da inserire
#include <SoftwareSerial.h>
#include <Wire.h>
#include <Servo.h>
LiquidCrystal_I2C lcd(0x27, 16, 2);
SoftwareSerial Bluetooth(6,7);           //il tx di arduino va sul 6, e l'rx va sul 7,
Servo pan;                               //essendo rispettivamente il 6 rx e 7 tx di arduino
Servo tilt;                              //variabili per il funzionamento del pantilt
char n;
String nota = "?";
double c, gamma, a=3, alfarad, alfagrad;
double potenzatilt, potenzapan,tgfi, x, y;
double mt = -12.2, mp = -11, qt = 1675, qp = 1375, firad;
double figrad, impfi, d,l1, l2 = 6;
double gammarad,gammagrad,impgamma;
float frequenza[3];
float misura[5];
float temporary;
boolean clipping = 0;      //clipping indicator variables
                           //variabili per la memoria
byte newData = 0;
byte prevData = 0;
unsigned int time = 0;     //tiene il tempo e manda valori al timer[]
int timer[10];             //timer
int slope[10];             //valore del cambiamento dello slope
unsigned int totalTimer;   //usato per calcolare il periodo
unsigned int period;       //memoria per il periodo della fase
byte index = 0;            //elenco della memoria
float frequency;
float frequencyMeasured;
int maxSlope = 0;
int newSlope;
                           //variabili che determinano quando puó misurare la frequenza
byte noMatch = 0;          //conta quante volte il valore non era accettabile e lo resetta
byte slopeTol = 3;
int timerTol = 10;
unsigned int ampTimer = 0;
byte maxAmp = 0;
byte checkMaxAmp;
byte ampThreshold = 75;                  //alza se il segnale ha molto noise
size_t misuraLength = sizeof(misura) / sizeof(misura[0]);
```

Incominciamo con la dichiarazione delle librerie:

```
#include <LiquidCrystal_I2C.h>
#include <SoftwareSerial.h>
#include <Wire.h>
#include <Servo.h>
```

- *Liquid Crystal IC2*, che serve per il funzionamento dello schermo lcd.
- La libreria *Softwareserial*, che permette la dichiarazione di una fake serial che verrá usata per la comunicazione bluetooth tramite l'HC-05.
- La libreria *Wire*, che permette la comunicazione tra Arduino e i dispositivi I2C.
- E infine la libreria *Servo*, che serve per la dichiarazione dei **servomotori** e il funzionamento di essi.

Le librerie cosa sono? Le librerie sono dei pacchetti di codice creati con l'idea di essere usati in diversi programmi.
Ci sono due tipi di librerie, statiche e dinamiche:

-Le librerie statiche, anche conosciute come archivi, consiste in routine che vengono compilate e collegate direttamente nel programma. Quando si compila un programma che utilizza una libreria statica, tutte le funzionalità della libreria statica che il programma utilizza diventano parte dell' eseguibile.

-Le librerie dinamiche, chiamate anche librerie condivise, al contrario delle librerie statiche vengono applicate soltanto quando si inizializza il codice.
Anche se in questo caso si usano librerie statiche.

Dopo di questo si procede con la dichiarazione dello **schermo LCD**, dei **due servo** del pan tilt, e della **software serial** che è usata per il bluetooth dove dichiariamo i piedini 6 e 7 come RX e TX.

```
LiquidCrystal_I2C lcd(0x27, 16, 2);
SoftwareSerial Bluetooth(6,7);
Servo pan;
Servo tilt;
```

Si dichiarano poi le diverse variabili usate per **il puntamento del pan tilt.**
Ad esempio il char n, che viene usato come oggetto nello switch case per comunicare con l'applicazione bluetooth e i diversi puntamenti delle note.

```
char n;
String nota = "?";
double c, gamma, a=3, alfarad, alfagrad;
double potenzatilt, potenzapan,tgfi, x, y;
double mt = -12.2, mp = -11, qt = 1675, qp = 1375, firad;
double figrad, impfi, d,l1, l2 = 6;
double gammarad,gammagrad,impgamma;
```

Un'altra parte importante dichiarata qui sono tutte le variabili di tipo double che verranno usati e spiegati piú tardi nella formula del pan tilt. Le variabili double sono del tipo di variabili che possono tenere la massima quantitá di byte possibili.

```
float frequenza[3];
float misura[5];
float temporary;
boolean clipping = 0;     //clipping indicator variables
                          //variabili per la memoria
byte newData = 0;
byte prevData = 0;
unsigned int time = 0;    //tiene il tempo e manda valori al timer[]
int timer[10];            //timer
int slope[10];            //valore del cambiamento dello slope
unsigned int totalTimer;  //usato per calcolare il periodo
unsigned int period;      //memoria per il periodo della fase
byte index = 0;           //elenco della memoria
float frequency;
float frequencyMeasured;
int maxSlope = 0;
int newSlope;
```

Sono dichiarati gli array frequenza, da 3 posti, e misura, da 5 posti, questi verranno usati per lo smistamento della frequenza dopo nel void loop.

Vengono dichiarate le diverse variabili per la **memoria** del codice.
Queste variabili verranno usate per il memorizzare dei diversi dati che poi verranno usate per il calcolo frequenza.
Variabili **booleane**, che possono essere o 1 o 0, variabili **int** che possono contenere numeri interi, e variabili **float** che possono contenere numeri decimali, al contrario degli **int**.

```
byte noMatch = 0;         //variabili che determinano quando puó misurare la frequenza
byte slopeTol = 5;        //conta quante volte il valore non era accettabile e lo resetta
int timerTol = 15;

unsigned int ampTimer = 0;
byte maxAmp = 0;
byte checkMaxAmp;
byte ampThreshold = 50;   //alza se il segnale ha molto noise
```

Le diverse variabili che serviranno per ricavare il **periodo** e poi la **frequenza**.
Ci sono le variabili **byte** che possono contenere dati binari e il loro valore predefinito è 0.

```
size_t misuraLength = sizeof(misura) / sizeof(misura[0]);
```

La variabile “misuraLength” è la grandezza dell’array, questo sará usato dopo per l’organizzazione dell’array nel bubble sort.

Void Setup e settaggio dell'ADC

```
void setup(){
Serial.begin(9600);
Bluetooth.begin(9600);                  //incominciamo al fake serial con baudrate a 36500
pan.attach(9);                           //attacchiamo pan e tilt al 9 e 10
tilt.attach(10);
lcd.begin();
lcd.backlight();
lcd.clear();
lcd.setCursor(1,0);
lcd.print("Benvenuto al");
lcd.setCursor(3,1);
lcd.print("Tune Tilt!");
cli();
                                        //abilitiamo il campionamento costante del pin A0 a 38.5kHz
  ADCSRA = 0;                            //resettiamo i registri ADCSRA

  ADMUX |= (1 << REFS0);                //settiamo il voltaggio di riferimento
  ADMUX |= (1 << ADLAR);                //allineiamo a sinistra la fase cosí si leggono solo i primi 8 bit

  ADCSRA |= (1 << ADPS2) | (1 << ADPS0);   //settiamo il ADC clock con un prescaler di 16mhz
  ADCSRA |= (1 << ADATE);               //abilitiamo l'auto trigger
  ADCSRA |= (1 << ADIE);                //abilitiamo gli interrupt
  ADCSRA |= (1 << ADEN);                //abilitiamo ADC
  ADCSRA |= (1 << ADSC);                //incominciamo il misuramento dell'adc

  sei();                                //abilitiamo gli interrupts
}
```

Void Setup

Solitamente in un programma in c++ il void setup è dove si vanno a definire i collegamenti dei diversi apparati ai piedini di arduino.

Vengono inizializzate le due seriali con un baud rate di 9600, dove dalle quali possiamo scrivere e leggere valori. Di seguito si settano i piedini 9 e 10 ai due servo, il pan e il tilt, che verranno seguitamente comandati, e viene impostato il piedino 12 come output.

```
Serial.begin(9600);
pinMode(12,OUTPUT);
Bluetooth.begin(9600);
pan.attach(9);
tilt.attach(10);
```

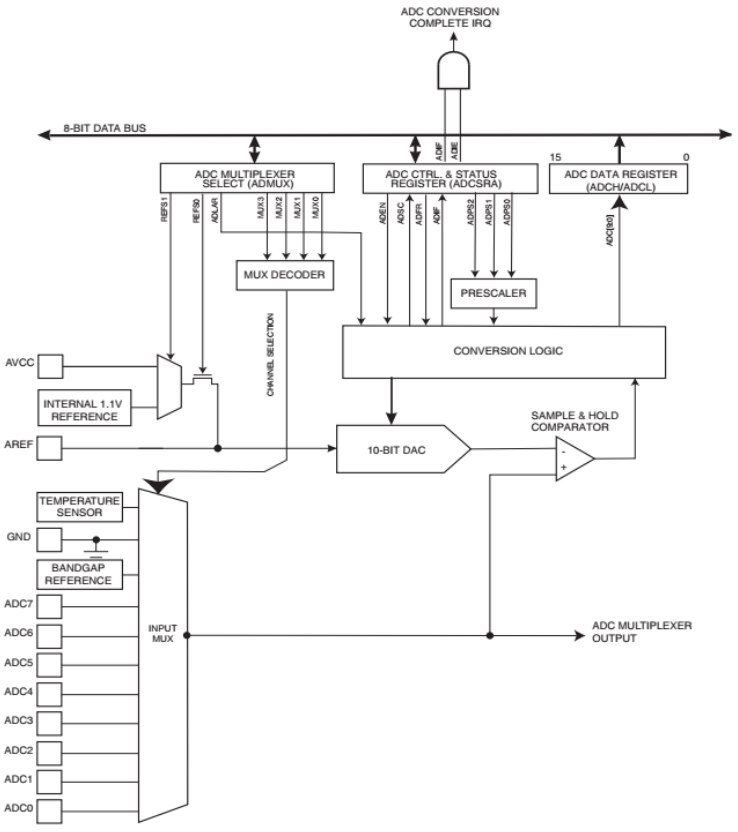
```
lcd.begin();
lcd.backlight();
lcd.clear();
lcd.setCursor(1,0);
lcd.print("Benvenuto al");
lcd.setCursor(3,1);
lcd.print("Tune Tilt!");
```

Qui viene impostato lo schermo lcd, dove usiamo comandi come lcd.begin(); per inizializzarlo e seguitamente comandi come lcd.backlight(); per accendere la luce di sfondo, un led dietro allo schermo usato per renderlo piú visibile. Cancellato lo schermo con lcd.clear();. Seguitamente si stampano sullo schermo le parole "Benvenuto al Tune Tilt!" attraverso i comandi lcd.setCursor(1,0); e lcd.print("Benvenuto al"); che scrive sulla prima riga dello schermo partendo dalla prima colonna le parole "Benvenuto al", e successivamente lcd.setCursor(3,1); e lcd.print("Tune Tilt!"); che scrive sulla seconda riga partendo dalla terza colonna le parole "Tune Tilt!".

Settaggio dell'ADC

```
cli();
                                        //abilitiamo il campionamento costante del pin A0 a 38.5kHz
  ADCSRA = 0;                            //resettiamo i registri ADCSRA e ADCSRB
  ADCSRB = 0;

  ADMUX |= (1 << REFS0);                //settiamo il voltaggio di riferimento
  ADMUX |= (1 << ADLAR);                //allineiamo a sinistra la fase cosí si leggono solo i primi 8 bit

  ADCSRA |= (1 << ADPS2) | (1 << ADPS0);    //settiamo il ADC clock con un prescaler di 16mhz
  ADCSRA |= (1 << ADATE);                   //abilitiamo l'auto trigger
  ADCSRA |= (1 << ADIE);                    //abilitiamo gli interrupt
  ADCSRA |= (1 << ADEN);                    //abilitiamo ADC
  ADCSRA |= (1 << ADSC);                    //incominciamo il misuramento dell'adc

  sei();                                    //abilitiamo gli interrupts
}
```

Prima di poter spiegare questa serie di comandi bisogna sapere cos'è e cosa fa l'ADC converter di Arduino:

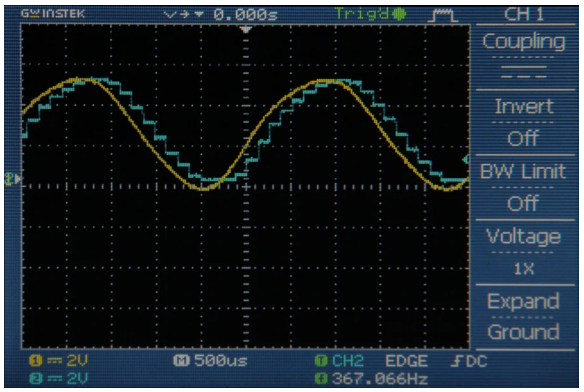


Un convertitore analogico-digitale (ADC) viene utilizzato per convertire un segnale analogico, come la tensione, in una forma digitale in modo che possa essere letto ed elaborato da un microcontrollore. La maggior parte dei microcontrollori al giorno d'oggi ha convertitori ADC incorporati.

Il processo di conversione ADC deve essere avviato dal programma e il completamento di una conversione può richiedere diverse centinaia di microsecondi. I convertitori ADC sono molto utili nelle applicazioni di controllo e monitoraggio poiché la maggior parte dei sensori producono segnali in uscita analogici.

Per spiegare in modo dettagliato tutte le funzioni dell'Analog-Digital Converter servirebbe scrivere una intera tesina, ma siccome tutte le operazioni utilizzate sono dentro l'interrupt dell'ADC, si tratterà solo quello.

Prima di tutto, perché usare l'ADC al posto di un semplice "analogRead();"?

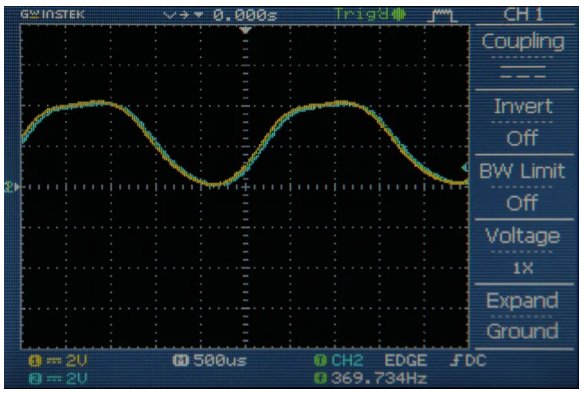


Il "sampling rate" piú alto del comando "analogRead();" è di intorno ai 8kHz, un valore accettabile in molti casi, ma non in questo caso. Per avere una misura accurata del segnale in entrata, il la frequenza di campionamento dev'essere almeno di 38.5kHz, simile a quella di arduino, che è 40kHz. Per fare questo bisogna bypassare il comando "analogRead();" ed utilizzare l'ADC.

Per avere una frequenza di campionamento di 38.5 kHz si imposta il counter interno dell'ADC di arduino manualmente a 500 kHz con questo comando:

```
ADCSRA |= (1 << ADPS2) | (1 << ADPS0);
```

Viene usato il prescale per impostare il clock a 500 kHz.



Si è messo il counter a 500 kHz perché, per leggere un nuovo valore analogico, l'ADC impiega 13 cicli del clock. In questo modo si ha 500 / 13 che da una frequenza di campionamento di circa 38.5kHz, questo mi da un campionamento ogni 26µs, mentre prima un campionamento era soltanto ogni 125µs.

Calcolo della Frequenza

```
ISR(ADC_vect) {                          //quando il valore dell' ADC é pronto |
                                         //                                    V
  prevData = newData;                    //immagazzina i valori dell'ADC
  newData = ADCH;                        //prende i valori di A0
  if (prevData < 127 && newData >=127){  //se va oltre 127
   newSlope = newData - prevData;        //calcola la pendenza e
   if (abs(newSlope-maxSlope)<slopeTol){ //se le pendenze sono uguali
                                         //registra il nuovo valore e resetta il tempo
      slope[index] = newSlope;
      timer[index] = time;
      time = 0;
      if (index == 0){

        noMatch = 0;
        index++;                         //incrementa l'indice
      }
      else if (abs(timer[0]-timer[index])<timerTol && abs(slope[0]-newSlope)<slopeTol){
                                         //se la durata del timer e delle pendenze sono uguali
        totalTimer = 0;
        for (byte i=0;i<index;i++){      //somma i valori del timer
          totalTimer+=timer[i];
        }
        period = totalTimer;             //definisce il periodo

        timer[0] = timer[index];
        slope[0] = slope[index];
        index = 1;                       //mette l'indice a 1
        noMatch = 0;
      }
      else{                              //se va oltre 127 ma non sono uguali
        index++;                         //incrementa l'indice e se supera 9 si resetta
        if (index > 9){
          reset();
        }
      }
    }
    else if (newSlope>maxSlope){         //se la pendenza nuova é molto piú grande di quella massima
      maxSlope = newSlope;
      time = 0;                          //resetta il clock
      noMatch = 0;
      index = 0;                         //resetta l'indice
    }
        index = 0;
        noMatch = 0;
        maxSlope = 0;
      }

  void reset(){                            //resetta alcune variabili
    index = 0;
    noMatch = 0;
    maxSlope = 0;
  }
```

Il calcolo per la frequenza è la parte piú importante del programma e ogni passaggio fatto è essenziale nel funzionamento del Tune Tilt.

```
ISR(ADC_vect) {                          //quando il valore dell' ADC é pronto |
                                         //                                    V
  prevData = newData;                    //immagazzina i valori dell'ADC
  newData = ADCH;                        //prende i valori di A0
  if (prevData < 127 && newData >=127){  //se va oltre 127
   newSlope = newData - prevData;        //calcola la pendenza e
```

Come prima si è ipotizzato che si potrebbero ottenere dei valori piú precisi della frequenza tenendo conto delle intersezioni con un valore costante di 2.5V al posto di analizzare ogni picco.

Questo perché misurando agli estremi del segnale sarebbe presente una distorsione molto alta e questo potrebbe disturbare l'analisi del segnale. Calcolandola a 2.5V, dove c'è meno distorsione, si ottiene un risultato piú preciso.

```
if (abs(newSlope-maxSlope)<slopeTol){    //se le pendenze sono uguali
                                         //registra il nuovo valore e resetta il temp
    slope[index] = newSlope;
    timer[index] = time;
    time = 0;
    if (index == 0){

      noMatch = 0;
      index++;                           //incrementa l'indice
    }
```

Siccome il segnale viene misurato ad una frequenza di A0 con una precisione di 8 bit, ovvero da 0 a 255, il punto di media a 2.5V, avrá un valore di 127. Quindi se il nuovo valore preso dall'ADCH è maggiore a 127 e il valore vecchio è meno di 127,il risultato è accettabile.

```
else if (abs(timer[0]-timer[index])<timerTol && abs(slope[0]-newSlope)<slopeTol){
                                    //se la durata del timer e delle pendenze sono uguali
  totalTimer = 0;
  for (byte i=0;i<index;i++){       //somma i valori del timer
    totalTimer+=timer[i];
  }
  period = totalTimer;              //definisce il periodo

  timer[0] = timer[index];
  slope[0] = slope[index];
  index = 1;                        //mette l'indice a 1
  noMatch = 0;
}
else{                               //se va oltre 127 ma non sono uguali
  index++;                          //incrementa l'indice e se supera 9 si resetta
  if (index > 9){
    reset();
```

Per misurare il periodo di un segnale viene impostato un “timer” dentro l’ADC che incrementa ogni volta che l’interrupt viene eseguito.

Ogni volta che il segnale passa i 2.5V si assegna il valore del timer ad una variabile chiamata “period” e si resetta il timer a 0. Poi nel void loop si calcola la frequenza dividendo la velocitá del timer, che è 38462, per il “period” trovato.

Per riuscire a calcolare piú tipi di onde e di fasi si decide di creare degli eventi chiamati eventi di soglia, che vengono dichiarati ogni volta che il segnale passa i 2.5V. Se questo succede piú volte in un ciclo, viene selezionato il primo segnale ricavato piú grande.

Dopo aver dichiarato una variabile “time”, incrementata a 38.5kHz, che misura il tempo tra due eventi di soglia, la si immagazzina in un array “timer[]”.
Dopo di questo si calcola anche la pendenza che viene salvata in un array chiamato “slope[]”.

Poi si confrontano i valori di “timer[]” e “slope[]” , dunque si ricava dove coincidono i due array, da questo si sommano gli elementi dell'array "timer[]" per ricavare la durata del ciclo e mandare questo valore alla variabile "period".

```
else if (newSlope>maxSlope){          //se la pendenza nuova é molto piú grande di quella massima
  maxSlope = newSlope;
  time = 0;                           //resetta il clock
  noMatch = 0;
  index = 0;                          //resetta l'indice
}
else{
  noMatch++;
  if (noMatch>9){
    reset();
  }
}
```

Tutto questo accade nell'interrupt dell'ADC. Dopo di questo nel void loop si calcola la frequenza con la solita formula. Nel codice c'è anche una variabile "noMatch" che aiuta a stabilire se coincidono i due array "slope[]" e "timer[]". Quando non coincidono i due array per 9 volte di fila, si attiva il void reset.

Per rendere l'acquisizione del segnale piú efficace cerchiamo di eliminare i casi dove c'è un clipping del segnale, ovvero quando il segnale massimo, o minimo sono uguali al minimo o massimo del segnale, ovvero i valori di 0 e 1023.

```
 if (newData == 0 || newData == 1023){     //se clippa
   clipping = 1;
 }
 time++;                                   //incrementa timer alla velocitá di 38.5kHz
 ampTimer++;                               //incrementa l'ampiezza del timer
 if (abs(127-ADCH)>maxAmp){
   maxAmp = abs(127-ADCH);
 }
 if (ampTimer==1000){
   ampTimer = 0;
   checkMaxAmp = maxAmp;
   maxAmp = 0;
 }
}
```

Se il segnale in entrata dall'ADCH, ovvero new data è uguale a 0 oppure a 1023, metto la variabile booleana clipping a 1. Dopo di questo, quando il timer dell'amplificazione raggiunge 1000, si resetta il timer e si pone il massimo dell'amplificazione a 0.

```
void reset(){                             //resetta alcune variabili
  index = 0;
  noMatch = 0;
  maxSlope = 0;
}
```

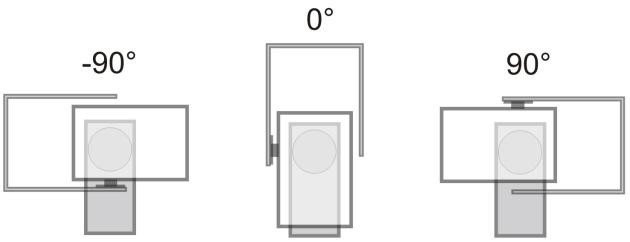
Il void reset è un semplice void che resetta l'indice dei due array, la variabile "noMatch" e il valore massimo della pendenza.

Void Puntamento Pan Tilt

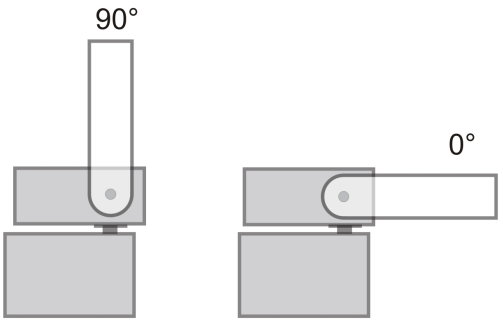
```
void terra (double x, double y){
        y=y+5.6;
        firad = atan(x/y);
        figrad = (180 * firad) / PI;
        impfi = mp * figrad + qp;
        d = sqrt((x * x) + (y * y));
        c = sqrt((d * d) + (a * a));
        l1 = sqrt((c * c) - (l2 * l2));
        gammarad = atan(l1 / l2);
        gammagrad = (gammarad * 180) / PI;
        alfarad = atan(a / d);
        alfagrad = ((180 * alfarad) / PI);
        gamma = gammagrad - alfagrad;
        impgamma = mt * gamma + qt;
        pan.writeMicroseconds(impfi);
        tilt.writeMicroseconds(impgamma);
}
```

Il void per il puntamento del pan tilt fa uso della cinematica inversa per ricavare un valore in impulsi da mandare ai due servo per farli puntare in delle coordinate scelte da noi.

Come prima cosa dobbiamo tarare i servo, mandando impulsi per capire il valore degli impulsi per farli puntare a diversi gradi.



Il pan si tara a -90°, 0°, 90°, trovano i valori degli impulsi in *µs* a quei valori in gradi.
Questo perché il valore della X puó essere sia negativo che positivo.



Il tilt si tara soltanto quando il servo è a 0° e 90°. Questo perché non serve puntare sotto agli 0° perché il valore della Y sará sempre con un valore positivo.

Dopo aver trovato i valori degli impulsi dei due servi, ricaviamo le “m” di entrambi i servi. La “m” è il coefficiente angolare della “retta” raffigurativa dei valori degli impulsi in *µs* del servo.

La formula della retta è: $y = mx + q$

E dunque la formula per trovare la m è: $m = (y-q)/x$

Dove la ***y*** è il valore degli impulsi a 90°, ***q*** è il valore in impulsi a 0°, e la ***x*** è 90.
In questo modo troviamo i valori dei due coefficienti angolari dei due servi, che ci serviranno poi dopo per calcolare gli impulsi finali.

Dopo aver tarato entrambi i motori si deve calcolare l'angolo φ° con: questo ci dá il valore in radianti dell'angolo Phi.

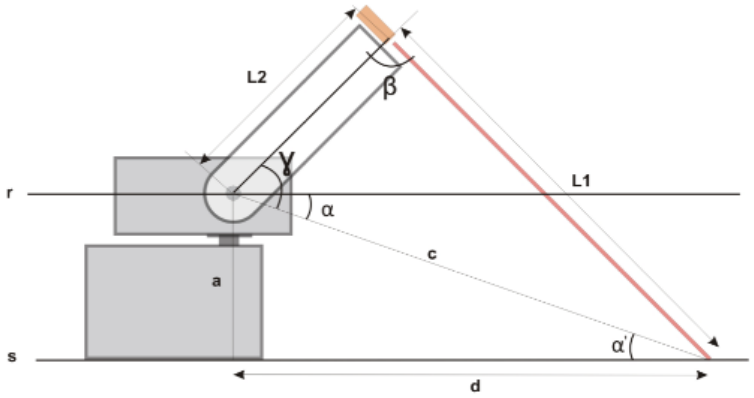
```
firad = atan(x/y);
```

```
figrad = (180 * firad) / PI;
```

Lo trasformiamo in gradi con la formula: dandoci il valore in gradi dell'angolo.

Calcoliamo il valore degli impulsi di φ° mettendo i valori trovati nella formula della retta.

```
impfi = mp * figrad + qp;
```



Calcoliamo le diverse ipotenuse e per il calcolo degli angoli attraverso il teorema di pitagora.

```
d = sqrt((x * x) + (y * y));
c = sqrt((d * d) + (a * a));
l1 = sqrt((c * c) - (l2 * l2));
```

Calcoliamo la "d", ovvero la lunghezza della distanza tra il servo e il punto da trovare.
La "c", ovvero l'ipotenusa del triangolo creato tra "a", ovvero la distanza dalla vite a terra, e la "d".
Infine calcoliamo "L1", che è la lunghezza della proiezione sul piano del laser, dove utilizziamo la "c" e "L2", ovvero la lunghezza del tilt.

```
gammarad = atan(l1 / l2);
gammagrad = (gammarad * 180) / PI;
alfarad = atan(a / d);
alfagrad = ((180 * alfarad) / PI);
gamma = gammagrad - alfagrad;
impgamma = mt * gamma + qt;
```

Innanzitutto calcoliamo l'angolo γ.

Per fare questo dobbiamo prima calcolare il suo valore in radianti facendo:

```
gammarad = atan(l1 / l2);
```

Poi bisogna calcolare `gammagrad = (gammarad * 180) / PI;` l'angolo in gradi facendo la stessa formula che viene usata per calcolare φ, ovvero:

L'angolo gamma che è calcolato comprende ancora l'angolo α, quindi dobbiamo calcolarlo e sottrarlo ad esso.

Per calcolare l'angolo α, dobbiamo prima fare: Calcolando il suo valore in radianti e poi convertendolo in gradi.

```
alfarad = atan(a / d);
alfagrad = ((180 * alfarad) / PI);
```

Infine troviamo il vero valore dell'angolo ɣ infine facendo:

```
gamma = gammagrad - alfagrad;
```

Con questo angolo possiamo trovare il valore in impulsi dell'angolo gamma mettendolo nella formula della retta.

```
impgamma = mt * gamma + qt;
```

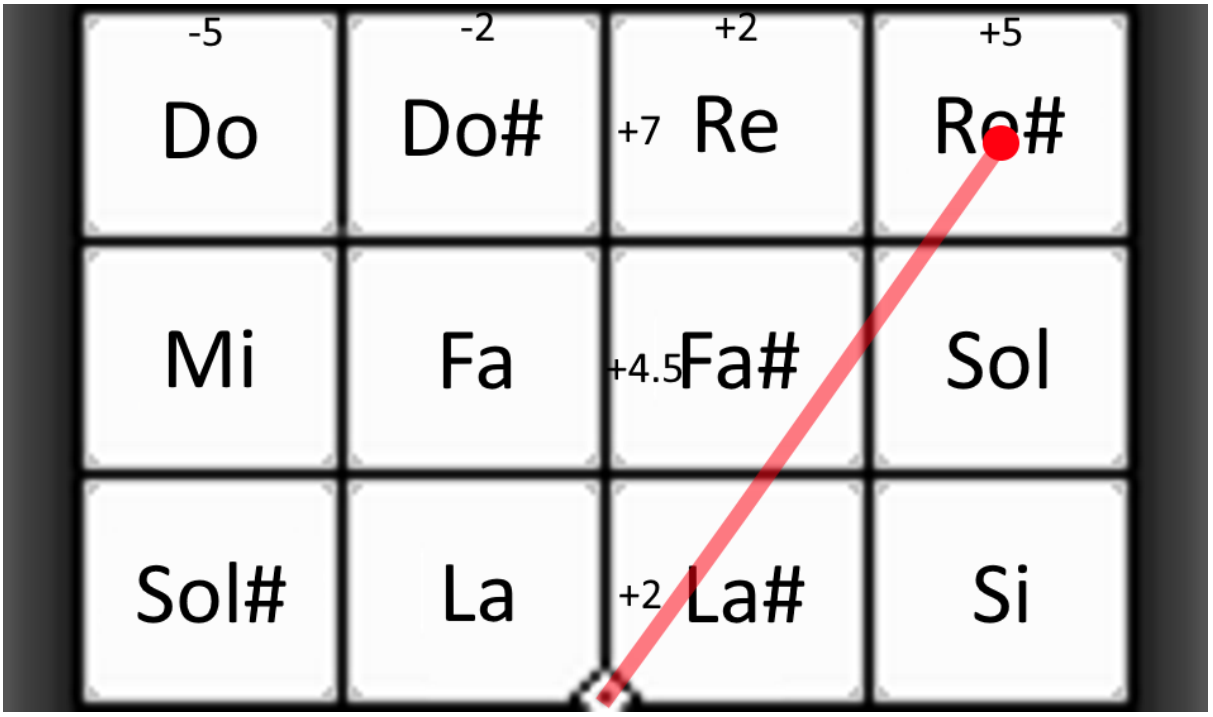
```
pan.writeMicroseconds(impfi);
tilt.writeMicroseconds(impgamma);
```

Infine viene mandato il valore degli impulsi di ***φ*** e ɣ, rispettivamente al pan e il tilt.

Questo void verrá poi usato nello switch case dove potrá essere usato per modificare il puntamento del Pan-Tilt.

Mandando quando vengono messe le coordinate ***x*** = 5, ***y*** = 6.5, Il void terra elabora gli impulsi da mandare per fare puntare il laser sulla tabella in alto a destra, ovvero quella del Re#.

Sulla tabella si vedono anche le diverse coordinate di ogni casella da mettere nel void terra per avere un puntamento corretto sulla griglia.



Void Bubble Sort

```
void sort(float a[], float size) {
    for(int i=0; i<(size-1); i++) {
        for(int o=0; o<(size-(i+1)); o++) {
                if(a[o] > a[o+1]) {
                    int t = a[o];
                    a[o] = a[o+1];
                    a[o+1] = t;
                }
        }
    }
}
```

Il void sort sono una serie di cicli **for** che servono per riorganizzare e riarrangiare i valori dell'array. Al posto di "float a[ ]" si mette l'array che si vuole ordinare, e al posto di "float size" si mette la grandezza dell'array.

Per questo è stata preparata la variabile "misuraLength" che calcola la grandezza dell'array "misura[ ]".

In informatica il **Bubble sort** è un semplice algoritmo di ordinamento di una lista di dati. Ogni coppia di elementi adiacenti viene comparata e invertita di posizione se sono nell'ordine sbagliato. L'algoritmo continua nuovamente a ri-eseguire questi passaggi per tutta la lista finché non vengono più eseguiti scambi, ció che indica che la lista è ordinata.

Switch Case

```
switch(n){
    case 'A': //Do
    terra(-4.2,7.2);

    break;

    case'B': //Do#
    terra (-1.4,7.1);

    break;

    case'C': //Re
    terra (2,7.1);

    break;

    case'D': //Re#
    terra (4.7,6.5);

    break;

    case'E': //Mi
    terra (-4.5,4.7);

    break;

    case'F': //Fa
    terra (-1.3,4.5);

    break;
```

In programmazione lo switch-case, è una struttura di controllo che permette la verifica del valore di una variabile.

Una variabile viene valutata e ne viene fatto un confronto di uguaglianza rispetto a valori definiti dal programmatore. Tali valori sono suddivisi in casi, che vengono specificati utilizzando la parola case. Se l'espressione ha un valore uguale a quello di uno dei casi previsti, viene eseguito il corrispondente codice.

In questo caso la nostra variabile è il "char n", e a seconda della lettera che "n" è si hanno delle coordinate diverse nel void "terra", risultando in un movimento diverso del pan-tilt.

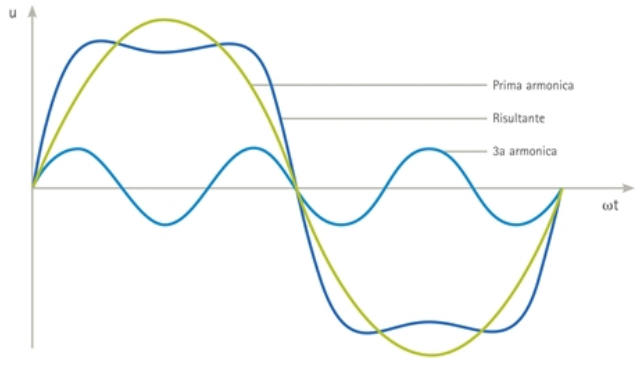
Void Loop e Note

```
void loop(){
  if (checkMaxAmp>ampThreshold){
    frequencyMeasured = 38462/float(period);                          //calcola la frequenza facendo il timer rate / periodo

      for (int i=0;i<4;i++){
      misura[i]=frequencyMeasured;
    }
    sort(misura,misuraLength);
    for(int i = 0; i < 2; i++)
{
  frequenza[i] = misura[i+1];
  delay(50);
}
    if(frequenza[0]<=frequenza[1]+10 && frequenza[0]>=frequenza[1]-10||frequenza[0]<=frequenza[2]+10 && frequenza[0]>=frequenza[2]-10){
      frequency=frequenza[0];
      delay(50);
    }
    if(frequenza[1]<=frequenza[0]+10 && frequenza[1]>=frequenza[0]-10||frequenza[1]<=frequenza[2]+10 && frequenza[1]>=frequenza[2]-10){
      frequency=frequenza[1];
      delay(50);
    }
    if(frequenza[2]<=frequenza[0]+10 && frequenza[2]>=frequenza[0]-10||frequenza[2]<=frequenza[1]+10 && frequenza[2]>=frequenza[1]-10){
      frequency=frequenza[2];
      delay(50);
    }
  }
    if(frequency < 80 || frequency > 1326){
    lcd.clear();
    lcd.setCursor(2,0);
    lcd.print("Calcolando...");
    delay(500);
 }
    else {
    Serial.print(frequencyMeasured);
    Serial.println("Freq: ");
    Serial.print(frequency);
    Serial.println(" Hz");
    lcd.clear();
    lcd.setCursor(1,0);
    lcd.print("Freq = ");
    lcd.print(frequency);
    lcd.print("Hz");
    Bluetooth.print(frequency);
    Bluetooth.println("hz;");

}

if(frequency > 80 && frequency < 84){
nota = "Mi 2   ";
lcd.setCursor(2,1);
lcd.print("Nota = ");
lcd.print(nota);

delay(500);
n='E';}

if(frequency > 84 && frequency < 89){
nota = "Fa 2   ";
lcd.setCursor(2,1);
lcd.print("Nota = ");
lcd.print(nota);

delay(500);
n='F';}
```

Solitamente nel void loop viene scritto cosa il programma deve fare e in molti casi è la parte piú importante di un programma scritto su arduino.

In questo pezzo di programma si mette la frequenza acquisita attraverso diversi filtri per avere un segnale in entrata piú costante e consistente, questo perché?

La chitarra è uno strumento che produce un segnale molto caotico e svariato, per questo l'algoritmo che ho usato nel 20% dei casi ha ricavato il valore di frequenza sbagliato. Questo perché?



Un qualunque segnale periodico produce una serie di armoniche che vanno ad alterare il segnale originale.

Gli **armonici naturali** sono una successione di suoni le cui frequenze sono multipli di una nota di base, chiamata fondamentale.

Un suono prodotto da un corpo vibrante non è mai puro, ma è costituito da più suoni che si differenziano fra loro in frequenza.

Al suono fondamentale, quindi, se ne aggiungono altri: questi sono gli armonici, che vanno ad alterare la fondamentale rendendolo un segnale impuro, con variazioni. Ad esempio, se una corda di lunghezza L emette un Mi come primo armonico, la stessa corda vibra con meno intensità anche a frequenza doppia, emettendo un Mi all'ottava superiore.

Per eliminare questi imprevisti ho fatto passare la frequenza ricavata attraverso diversi filtri.

```
    for (int i=0;i<4;i++){
    misura[i]=frequencyMeasured;
  }
  sort(misura,misuraLength);
  for(int i = 0; i < 2; i++)
{
  frequenza[i] = misura[i+1];
  delay(50);
}
```

La frequenza calcolata viene campionata e 5 valori vengono messi dentro ad un array, chiamato "misura[]".

L'array "misura[]" viene riarrangiato attraverso la funzione sort, la funzione del bubble sort dichiarata prima, dal valore piú basso al valore piú alto.

Dopo che è stato organizzato, vengono scartati i valori piú estremi, mettendo i 3 valori in mezzo dentro un altro array, "frequenza[]".

```
if(frequenza[0]<=frequenza[1]+5 && frequenza[0]>=frequenza[1]-5||frequenza[0]<=frequenza[2]+5 && frequenza[0]>=frequenza[2]-5){
  frequency=frequenza[0];
  delay(50);
}
 else if(frequenza[1]<=frequenza[0]+5 && frequenza[1]>=frequenza[0]-5||frequenza[1]<=frequenza[2]+5 && frequenza[1]>=frequenza[2]-5){
  frequency=frequenza[1];
  delay(50);
}
 else if(frequenza[2]<=frequenza[0]+5 && frequenza[2]>=frequenza[0]-5||frequenza[2]<=frequenza[1]+5 && frequenza[2]>=frequenza[1]-5){
  frequency=frequenza[2];
  delay(50);
}
```

Vengono seguitamente organizzati ancora attraverso diversi “if”, che comparano i 3 valori dell'array “frequenza[]”, e se due degli array hanno un valore simile a l’un l'altro, viene messo quel valore nella variabile “frequency”, che poi verrá elaborata in futuro.

```
if(frequency < 80 || frequency > 1326){
lcd.clear();
lcd.setCursor(2,0);
lcd.print("Calcolando...");
delay(500);
```

Se il valore della frequenza esce fuori dal range di note di una chitarra, in questo caso meno di 80 Hz e piú di 1326 Hz, scrive sullo schermo “Calcolando...”

Invece se il valore rientra nel range di note di una chitarra, prima scrive sulla seriale la frequenza misurata. Dopo di questo scrive sulla prima riga dello schermo LCD il valore in frequenza con i comandi “lcd.print(frequency);”, e infine manda il valore in frequenza alla seriale bluetooth che verrá analizzata dall’applicazione.

```
else {
Serial.print(frequencyMeasured);
Serial.println("Freq: ");
Serial.print(frequency);
Serial.println(" Hz");
lcd.clear();
lcd.setCursor(1,0);
lcd.print("Freq = ");
lcd.print(frequency);
lcd.print("Hz");
Bluetooth.print(frequency);
Bluetooth.println("hz;");
```

```
if(frequency > 80 && frequency < 84){
nota = "Mi 2   ";
lcd.setCursor(2,1);
lcd.print("Nota = ");
lcd.print(nota);

delay(500);
n='E';}
```

Infine la frequenza viene analizzata un’ultima volta dove, quando rientra nella gamma di una nota, viene analizzata e viene scritto sullo schermo lcd la nota corrispondente. Infine viene impostata la variabile “char n” ad un carattere, questa variabile determinerá dove il braccio robotico si muoverá attraverso lo switch case. Questo if verrá poi ripetuto per ogni nota da “Mi 2” a “Mi 6”.

Codice Applicazione

Per la creazione di questa applicazione è stato usato App Inventor, un programma open source che permette di creare e customizzare applicazioni programmandole attraverso dei blocchi.

App Inventor è un programma molto semplice, che solitamente viene usato per progettare app con scopi semplici con codici di poche righe; questo ha reso il processo di programmazione molto lungo e tedioso, essendo un'applicazione con un codice abbastanza complicato.

Cos'è App inventor?

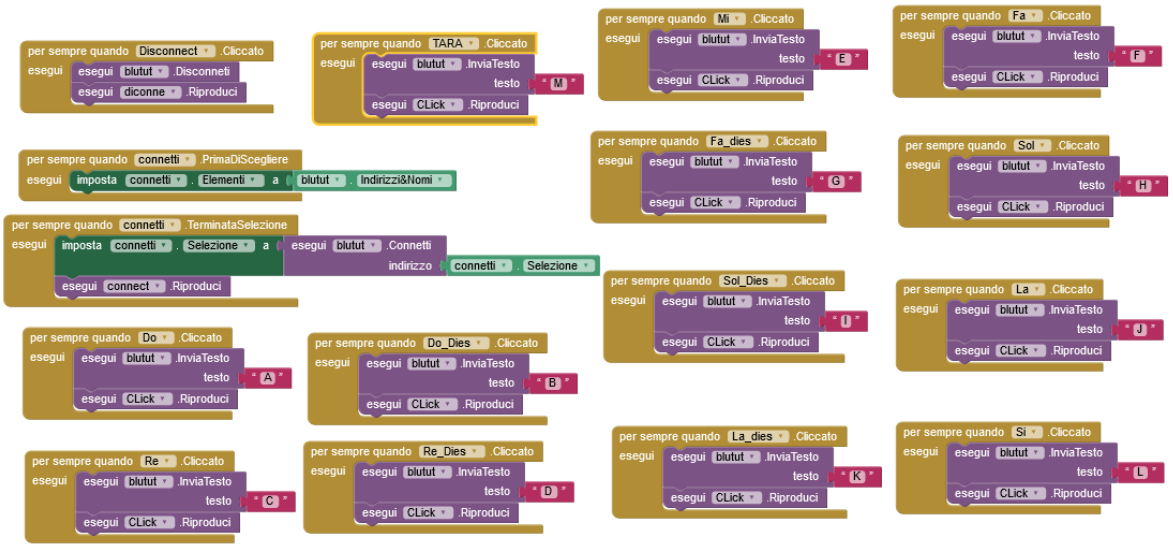
App Inventor è un semplice ambiente di sviluppo per applicazioni Android.

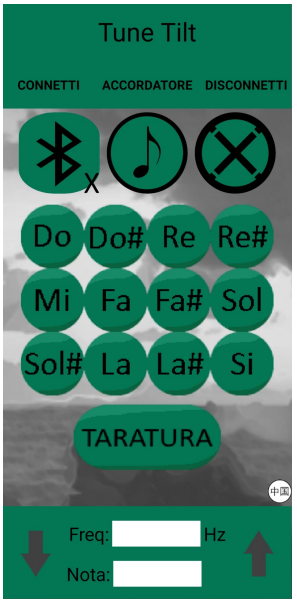
Questo ambiente di sviluppo fu creato soprattutto per persone che volessero programmare semplici applicazioni per android ad uso personale.

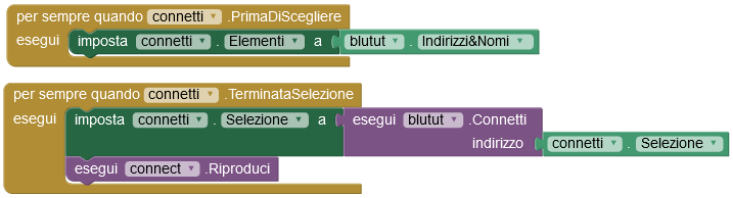
Tali applicazioni possono essere installate direttamente sul cellulare tramite WiFi o USB, provate su un emulatore Android per PC.

La grafica dell'interfaccia è molto semplice ed intuitiva, grazie al drag-and-drop ed è molto simile ad altri semplici ambienti di programmazione come Scratch.

Comandi per i Bottoni e Listpicker





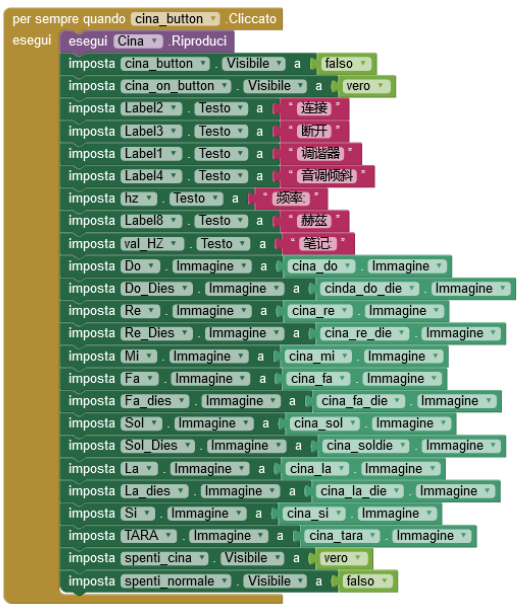


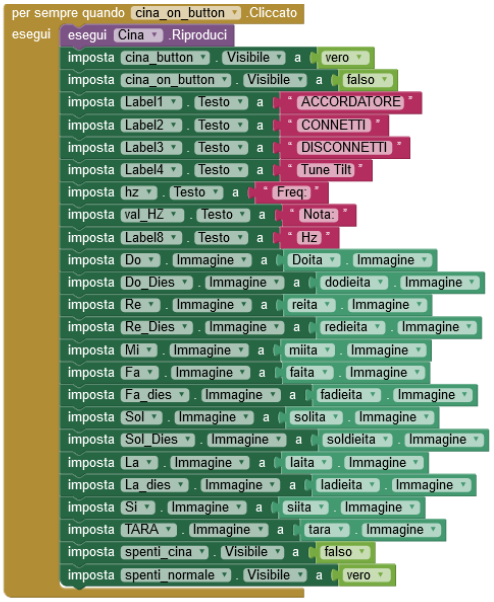
I due comandi necessari per il funzionamento dell'applicazione sono quelli usati per la connessione al modulo **HC-05**, cioé dove viene inizializzata la seriale e viene usato come indirizzo di esso quello del dispositivo bluetooth scelto nella lista del **listpicker**. Quando poi si ha una connessione con il dispositivo corretto, viene riprodotto un suono.

Il bottone disconnetti viene usato per disconnettere il dispositivo bluetooth collegato. Dopo lo scollegamento viene riprodotto un suono che indica la sua riuscita.

Dopo di esso ci sono i bottoni che mandano un carattere ad arduino attraverso il modulo **HC-05**. Questi caratteri verranno analizzati e utilizzati dentro uno switch case che andrá a comandare il servo, se si vuole usare il pan tilt normalmente. In questo caso il carattere A corrisponde a una coppia di valori diversi che sono stati calcolati apposta per puntare su un punto predefinito.

Programma per Traduzione in Cinese

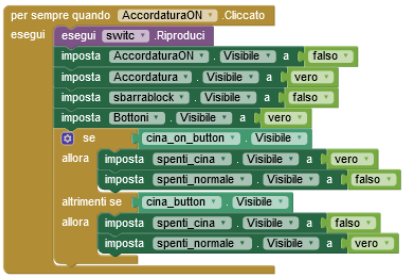


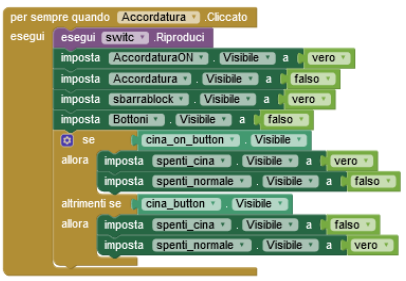


I due bottoni servono a cambiare la lingua dell'applicazione traducendo tutte le scritte in cinese, comprese quelle riferite alle note le note.
Ci sono due bottoni perché quando un bottone viene premuto diventa invisibile, e rende l'altro visibile.
Avere due bottoni diversi da piú libertá e possibilitá per risolvere diversi problemi.
Quando viene premuto il bottone per tradurre in cinese:

- Cambia l'immagine dei bottoni delle note con le scritte cinesi.
- Rende visibili i bottoni cinesi per l'accordatura in cinese.
- Traduce tutte le scritte.
- Infine riproducono un motivo musicale suonato con strumenti orientali.

Programma per la Modalitá Accordatura

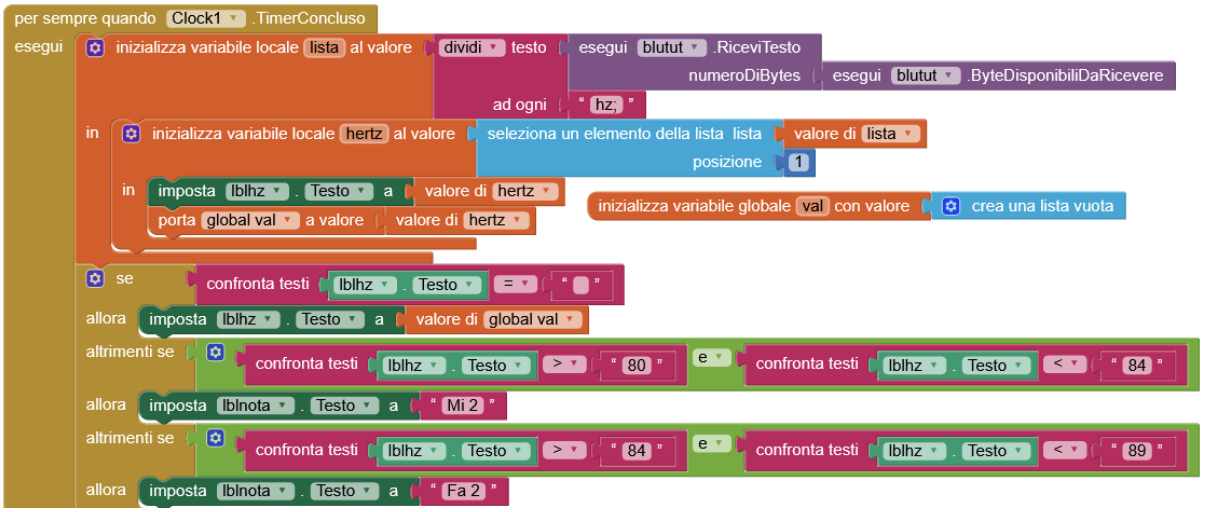




I bottoni per impostare la modalitá "Accordatura" sono simili a quelli per la traduzione in cinese, anche se fortunatamente sono meno comandi.

Hanno una funzionalitá principale, quella di rendere visibile la scrollbar e il blocco per accordare la chitarra se scordata. Ha dentro di esso anche un "if" che rende alcuni bottoni visibili e non analizzando se la lingua è in italiano o in cinese. Infine viene riprodotto un suono che indica la pressione del bottone.

Acquisizione Hz da Arduino e Dichiarazione Note



Acquisizione Hz

Per ricevere i valori in hertz mandati dall'HC-05 dichiariamo un Clock, che ha un tick di 500ms. Ogni tick inizializza una lista locale che prende i valori della bluetooth serial con i comandi ".RiceviTestonumeroDiBytes" e ".ByteDisponibiliDaRicevere", poi letti fino al "hz;" con il comando "dividi testo ad ogni …".
Dentro a questa variabile locale viene inizializzata un'altra variabile locale "hertz" che seleziona il primo valore della lista locale, nel quale viene immagazzinato il valore numerico degli Hertz. Questo valore viene stampato in un'etichetta che verrá analizzata in futuro.
Fuori dal clock viene inizializzata una variabile globale "val" che, dentro alla variabile locale "hertz", viene settata al valore di numerico della frequenza preso dall'HC-05.

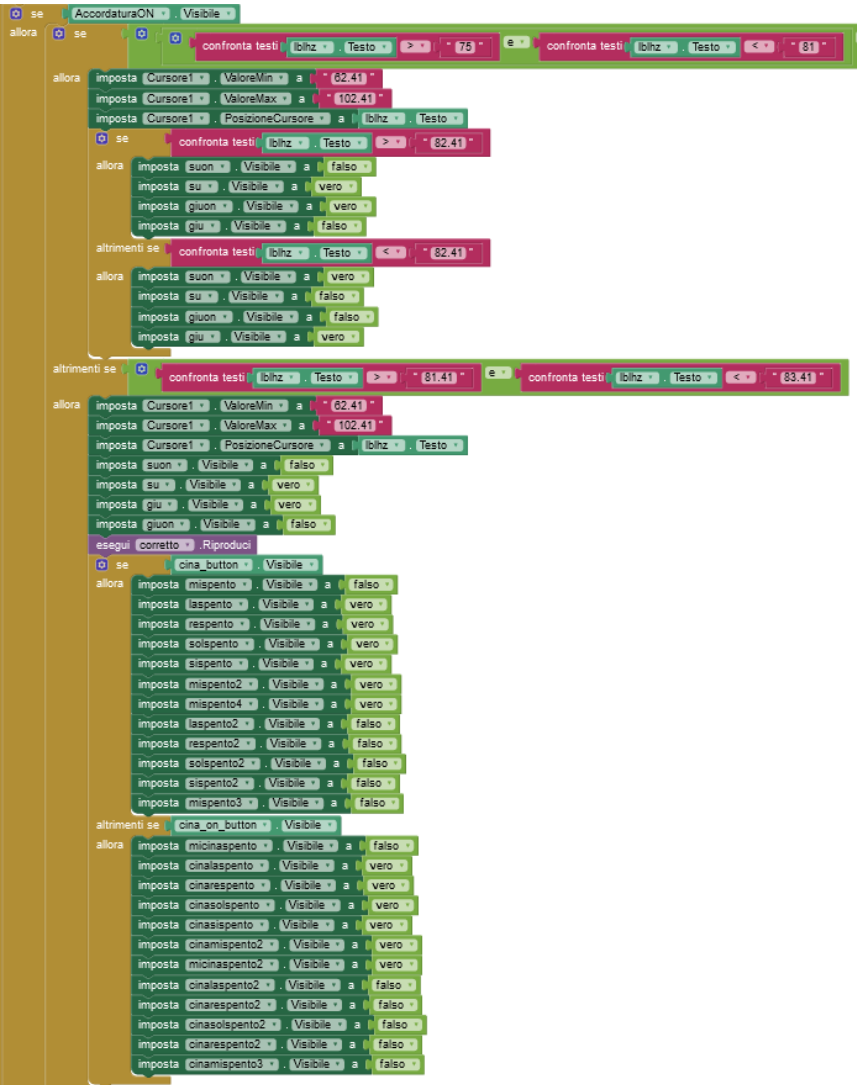
Dichiarazione Note

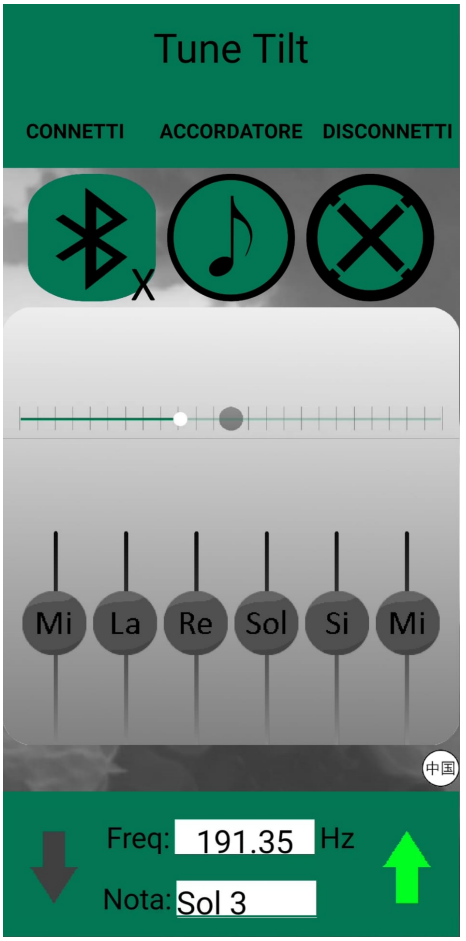
Ancora dentro al clock viene impostato un "if" nel quale:

Se la casella di testo degli hertz è vuota, la imposta al valore in hertz della variabile globale "val".

Invece se il valore nella casella di testo è compreso tra due valori, allora in un altra casella di testo, viene dichiarata la nota in un'altra etichetta.
I due valori che vengono usati per trovare la nota sono i valori poco meno e poco piú del valore in Hertz della nota, ad esempio la nota Mi 2 ha un valore di "82.41 Hz" noi settiamo il valore minimo e massimo da 80 a 84. Se il valore rientra in questi due valori allora il programma riconosce che la nota suonata è il Mi 2.

Programma per Accordare le Corde Aperte





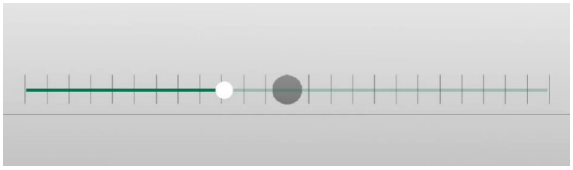
Questa lunga serie di comandi sono il cervello dell'accordatore dell'applicazione:

Tutte queste serie di "if" fanno sia che la frequenza venga analizzata in modo da essere elaborata e tradotta in elementi grafici che ci aiutano ad accordare la corda che vogliamo.

Quando suoniamo una corda scordata, come in questo esempio, la corda del Sol, vediamo che l'app ci avverte della scordatura della corda e ci indica visivamente come accordarla.

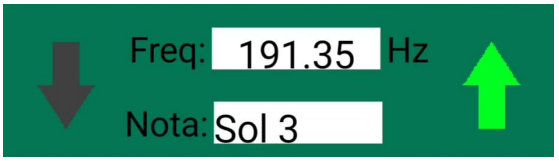
Innanzitutto si controlla se il bottone dell'accordatura è stato premuto, volendo dire che l'app è in modalitá accordatura, prima di fare qualunque tipo di operazione.

Se è in modalitá accordatura si controlla se il valore dell'etichetta dove viene messa la frequenza è un po' prima o un po' dopo della frequenza della corda. Ad esempio, la frequenza della corda Mi 2 è 82.41 Hz, quindi se la frequenza é o tra i 75 e 81 Hz o tra 84 e 95 Hz.

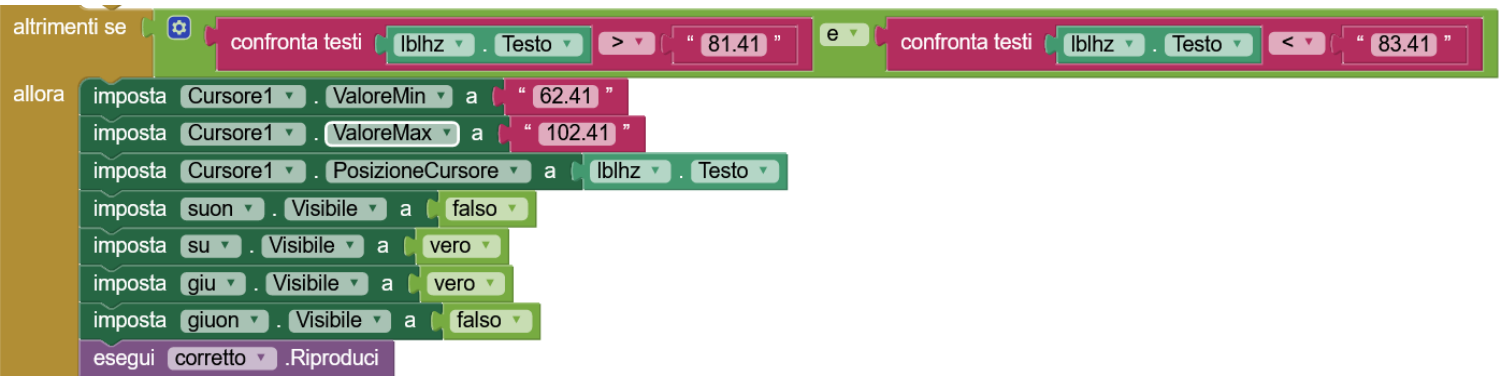


Se le condizioni dell'if sono soddisfatte, si imposta il valore minimo a 20 valori prima della frequenza accordata, il valore massimo a 20 valori prima della frequenza accordata e il cursore di essa al valore della frequenza che sta producendo la chitarra, che potrebbe essere o meno o piú di quella ideale.

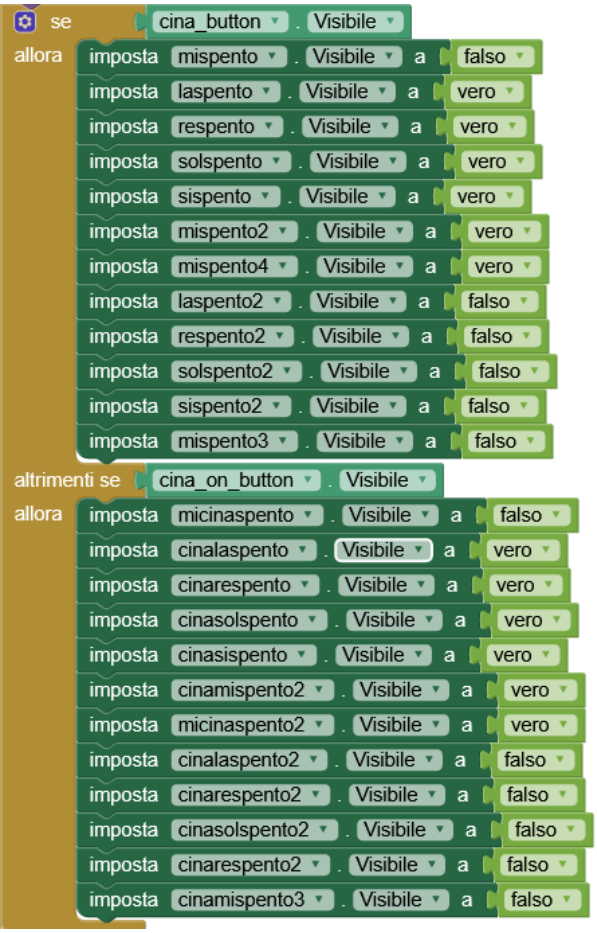
Se la frequenza è minore della frequenza ideale, si accende una freccia che indica di accordare in su la corda, mentre se la frequenza è maggiore della ideale, si accende una freccia che indica di accordare in giú la corda.



La frequenza di Sol 3 è 196 Hz.



Altrimenti se la frequenza acquisita rientra tra piú o meno 1 Hz del valore corretto, si spengono le frecce, si imposta la trackbar come è stato fatto precedentemente e si riproduce un suono che ti avverte di essere accordato.



Se l'applicazione è in modalitá italiana, si accende la foto della nota che stiamo accordando e si spengono tutti gli altri, in questo caso si accende la foto per la corda Mi 2, e si spengono tutte le altre. Mostrandosi cosí:

Invece se l'applicazione è in modalitá cinese, si accende la foto della corda che stiamo accordando e si spengono tutte le altre, questa volta con il carattere cinese della nota, in questo caso il Mi 2. Mostrandosi cosí:

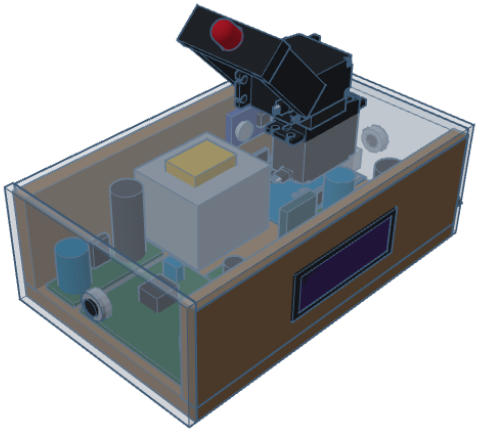
Costruzione

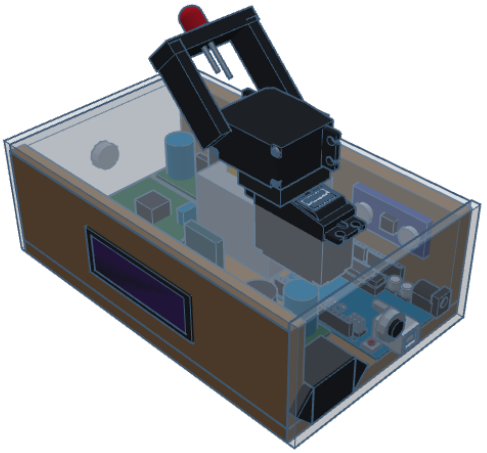
Qui si analizza il processo della progettazione del progetto in 3D e la costruzione della scatola che incapsula i componenti.

Dopo aver realizzato tutto il circuito, compreso il trasformatore e tutti i suoi componenti, e aver saldato vari elementi elettronici è sorto un problema:

Avendo a disposizione una scatola abbastanza piccola non c'era molto spazio per tutti i componenti, quindi è stato necessario risistemare tutti i componenti in modo da ottimizzare lo spazio.

TinkerCad e Modello 3D





Fortunatamente Tinkercad, un sito internet che consente di simulare circuiti elettronici con arduino e diversi componenti, ha un creatore di modelli 3D.

Tinkercad è stato molto comodo, essendo che la maggior parte dei componenti piú complicati su un livello visivo ha giá modelli 3D fatti dai creatori del sito.

In tal modo è stato possibile, senza un eccessivo dispendio di fatiche, costruire un modellino fedele al risultato finale del robot che ne mostrasse in maniera esaustiva tutte le caratteristiche costruttive e tecniche dello stesso gratuitamente.

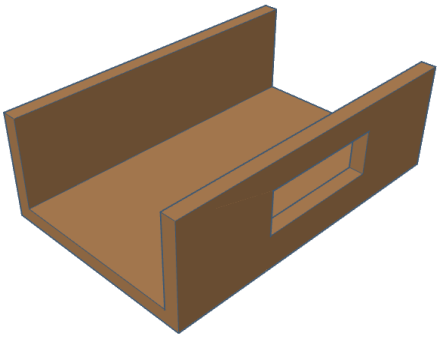
Materiali Usati

Nella creazione del tune tilt sono stati usati diversi materiali per creare la custodia dei componenti elettronici, ovvero:

- Pannello di legno 120mm x 6.5mm x 180mm.
- 2 Pannelli di legno 6.5mm x 55mm x 180mm.
- Pannello di plexiglass 180mm x 4mm x 120mm.
- 2 Pannelli di plexiglass 4mm x 65mm x 120mm.
- 10 viti a croce.

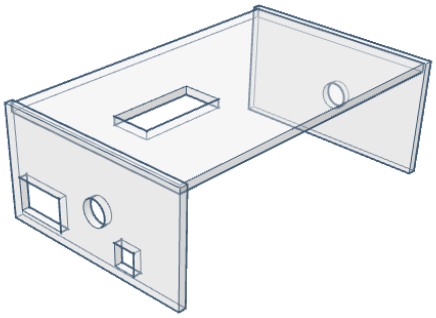
Assemblamento

Come prima cosa, sono state assemblate le parti in legno, formando la base con i due lati esterni.



Dopo questo si è tagliata una finestra da 70mm x 25mm per inserire il display LCD.

Infine è stata trattata la scatola di legno con olio di lino, per fare risaltare le venature, e per renderla esteticamente piú gradevole.



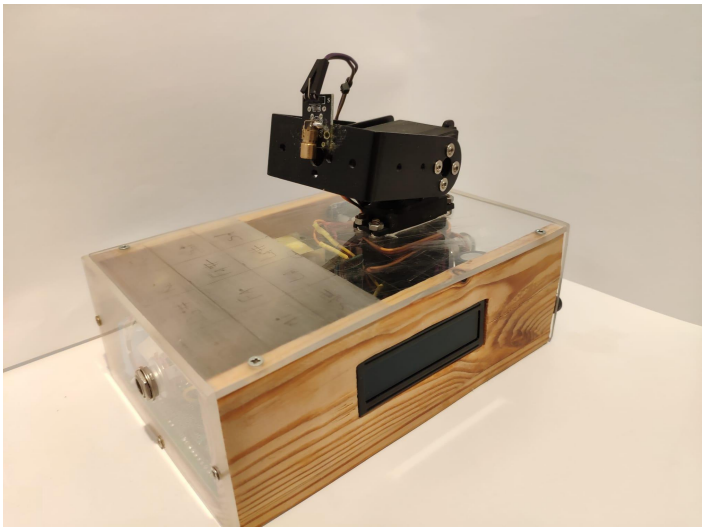
Dopo di questo sono state sagomate le parti in plexiglass, e ricavato i fori per i connettori e le prese. Compreso il foro dove monteró sopra il pan-tilt.

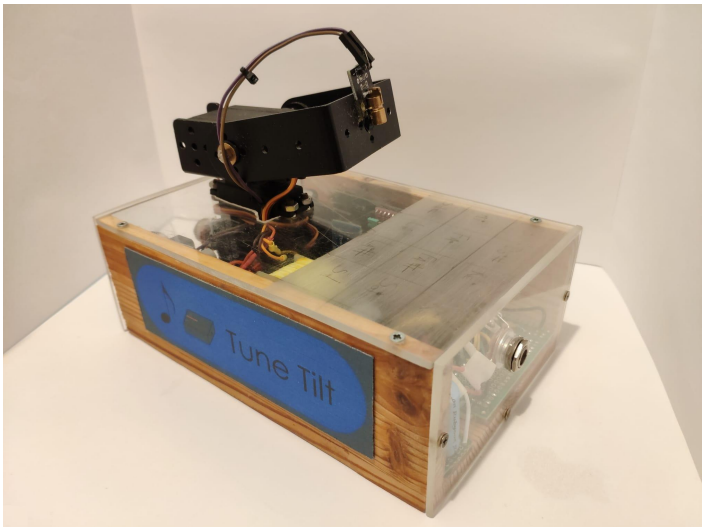
Infine sono stati aperti dei piccoli fori agli estremi del plexiglass per le viti.

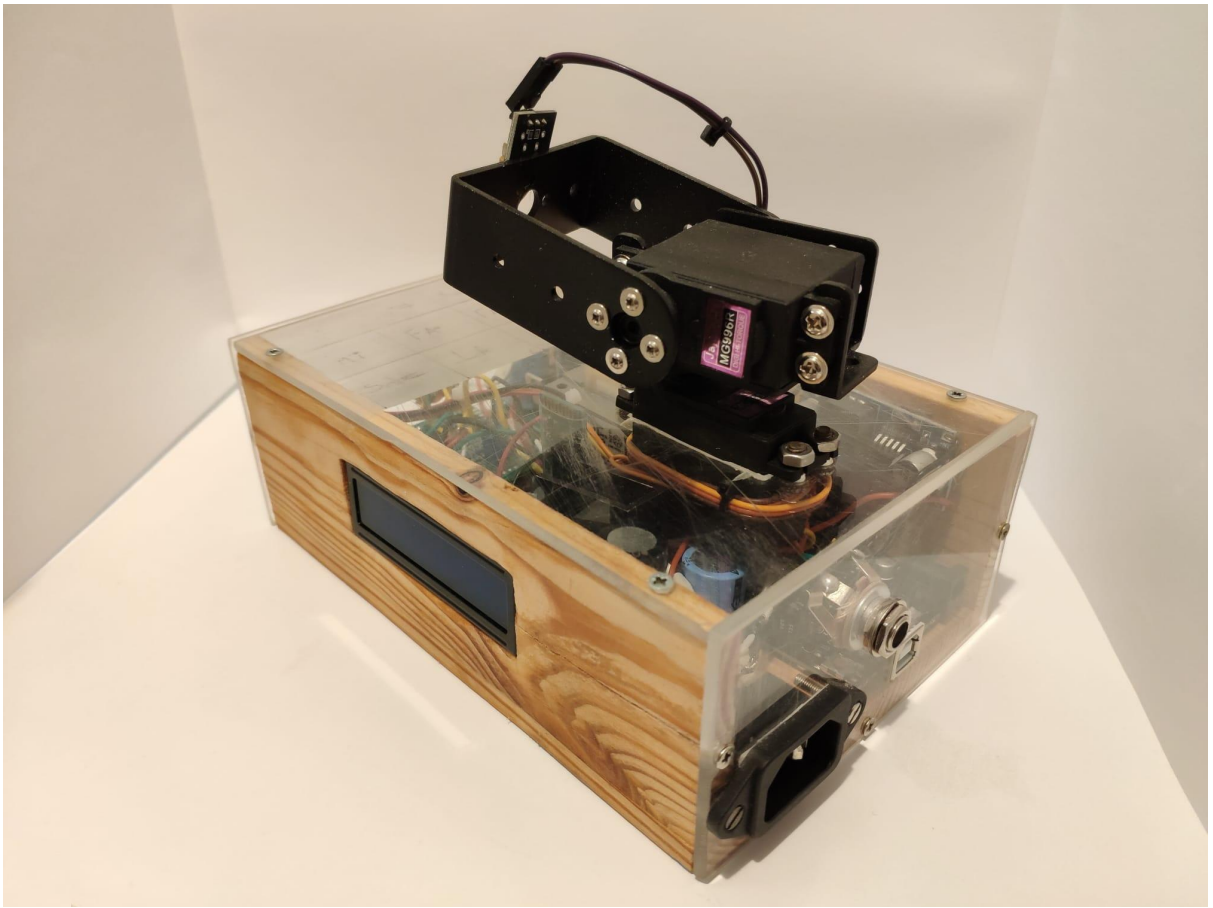
Dopo di che si sono cablati e assemblati tutti i componenti elettronici come progettato su TinkerCad, creando un prodotto finale soddisfacente.

Per abbellire ho creato un adesivo con un logo disegnato da me, che ho attaccato sul lato opposto al display.

Foto







Conclusione e Ringraziamenti

La realizzazione di questo progetto è stata molto complessa e ha messo alla prova diverse parti della mia formazione, dalla progettazione fino alla realizzazione ultima del robot. Le difficoltà incorse in tutte le fasi sono state notevoli, così come gli incidenti di percorso che hanno portato continui adattamenti, modifiche e ripensamenti del progetto stesso.

Il risultato è decisamente soddisfacente sia per il progetto in sé sia per le opportunità formative che ha portato la sua realizzazione, sia sul piano dell'utilizzo delle conoscenze acquisite sia sul piano delle competenze e delle abilità di problem-solving. Adattarlo inoltre ad una passione che sento forte, come quella per la musica, ha reso ancor più stimolante lo sviluppo del progetto, dalla fase embrionale fino alla sua completa realizzazione.

Questo periodo è stato denso e ricco di esperienze, ma anche di avversità. Sarebbe stato impossibile affrontarlo così positivamente senza l'aiuto, il supporto e lo stimolo continuo di molte persone che adesso è giunto il momento di ringraziare.

In primo luogo ci tengo a ringraziare sentitamente tutti i docenti che mi hanno accompagnato in questi anni, mettendo a disposizione le loro preziose conoscenze con pazienza e devozione. In particolare ringrazio il Prof Arco che mi ha aiutato a completare questo progetto.

Un ringraziamento alla famiglia che mi ha aiutato moralmente nello spronarmi a sviluppare le mie passioni e anche per avermi aiutato in aspetti pragmatici nello svolgimento di questo percorso, anche in aspetti pragmatici come un supporto nell 'assemblamento del progetto.

Un sentito ringraziamento va ai miei amici che hanno saputo credere nelle mie potenzialità, in maniera mai banale e dai quali ho tratto insegnamenti preziosi.

In particolare preziosi sono stati i miei compagni di classe, senza i quali questi anni non sarebbero stati così intensi e stimolanti. Senza il loro supporto emotivo e senza i loro preziosi consigli, e le loro critiche, sicuramente la mia formazione sarebbe stata mutilata e avrei avuto orizzonti molto più annebbiati rispetto al mondo. L'amicizia e la condivisione sono state un cardine fondante di questi anni di cui conserverò gelosamente non solo il ricordo, ma anche l'insegnamento.

Sitografia

Per la realizzazione e la progettazione del Tune Tilt sono stati consultati i seguenti siti:

Per la piattaforma Arduino i siti di riferimento sono stati:

- https://www.instructables.com/Arduino-Frequency-Detection/
- https://www.instructables.com/Arduino-Audio-Input/
- Schede Pan-Tilt del Professor Arco
- https://www.arnabkumardas.com/arduino-tutorial/adc-register-description/

Per la creazione dell'app è stato usato:

- https://appinventor.mit.edu/

Per i modelli 3D è stato usato:

- https://www.tinkercad.com/

Altri siti che sono stati utili sono:

- https://www.wikipedia.com
- https://www.treccani.it/
- https://scuolaelettrica.it/elettrotecnica/elettro6.php?imposto=SI
- https://hwhacks.com/2016/05/03/bubble-sorting-with-an-arduinoc-application/
- https://www.ti.com › lit › symlink › tl082
- https://www.alldatasheet.com/datasheet/pdf
- https://datasheetspdf.com › pdf
- Schede di TPS del Professor Porzio
- https://lucid.app/
- https://fritzing.org/