Università degli Studi
dell'Aquila

Dipartimento di
Ingegneria e Scienze
dell'Informazione e
Matematica

Corso di Laurea
Triennale in
Ingegneria Informatica

OSCILLOSCOPIO DIGITALE CON ARDUINO E ANDROID

Relatore
Prof. Vincenzo Stornelli
Correlatore
Dott. Leonardo Pantoli

Studente
Luca Di Vita
Matricola
210430

A.A. 2015/2016

Un ringraziamento speciale va ai miei famigliari: mia madre,
mio padre e mio fratello che hanno reso possibile tutto questo.
Al mio relatore, il Prof. Vincenzo Stornelli, nonché al correlatore
Prof. Leonardo Pantoli ed a tutti i tecnici del laboratorio di
elettronica per i preziosi consigli e suggerimenti.
Ed infine uno speciale a tutti gli amici e compagni
che mi hanno accompagnato e sostenuto in questo
percorso rendendolo fantastico. E in ultimo,
ma non per importanza a Francesca
che mi ha sopportato e supportato.

Indice

Introduzione

In questa tesi è trattato l'oscilloscopio digitale illustrandone tipologie, modalità di utilizzo e fine ultimo. In particolare sarà presa in considerazione una sua "versione tascabile" ottenuta con l'ausilio di recenti tecnologie, (sia hardware che software, quali Arduino) linguaggi di programmazione di attuale utilizzo e protocolli di comunicazione.

L'argomento sarà discusso toccando tutti i rami propri dell'ingegneria dell'informazione quali l'elettronica, l'informatica e le telecomunicazioni.

Il tutto può concettualmente essere diviso in due macro argomenti: l'Hardware e il Software, dai quali si scenderà nel particolare per trattare le singole tecnologie.

Nello specifico si tratterà nel primo capitolo lo strumento nelle sue generalità e l'idea di una sua versione tascabile, nel successivo secondo capitolo l'hardware della versione dello strumento concettualizzata in questa sede, in una sua versione preliminare, e una descrizione delle tecnologie software utilizzate per la programmazione e per la comunicazione. Nel terzo capitolo invece, si discute della realizzazione pratica dello strumento, e implementazione del software.

Introduction

The topic discussed, in this thesis, is the digital oscilloscope with its different types, mode of use and, at the end, the purpose. In particular I will considering the “mobile version” of digital oscilloscope, obtained with the aid of recent technology (both hardware and software) such that Arduino, programming languages and communication protocols. The topic it will be treaty throught all engineering brunch, as elettronics, informatics and telecomunications. the entire treatment can be divided into two macro subjects: Hardware and software, in general, and in particular, the singular technologies. The first chapter begins with an overview of the instrument discussion and the idea of its mobile version; followed a chapter about hardware version of the tool discussed here, in its preliminary version, with a description of software technologies used for programming and for comunication. The last chapter it is discussed the practical realization of the instrument and its software implementation.

1. Generalità sull'oscilloscopio

L'Oscilloscopio è uno strumento di misura elettronico che consente di visualizzare, su un riferimento cartesiano bidimensionale, l'andamento nel tempo di segnali di tensione consentendo di eseguire misure nel tempo quali: periodo, frequenza ecc. e in ampiezza come: valor medio, valore efficace ecc. La versatilità dello strumento permette, attraverso l'uso di opportuni trasduttori, di visualizzare l'andamento temporale di qualsiasi grandezza fisica. Il suo utilizzo è, generalmente, rivolto all'analisi di segnali periodici le quali frequenze e risoluzioni temporali sono dipendenti dalla banda passante dello strumento stesso a sua volta legata alla qualità dei materiali e, in definitiva, al costo dello strumento. Si va dalle decine di MHz ai diversi GHz.

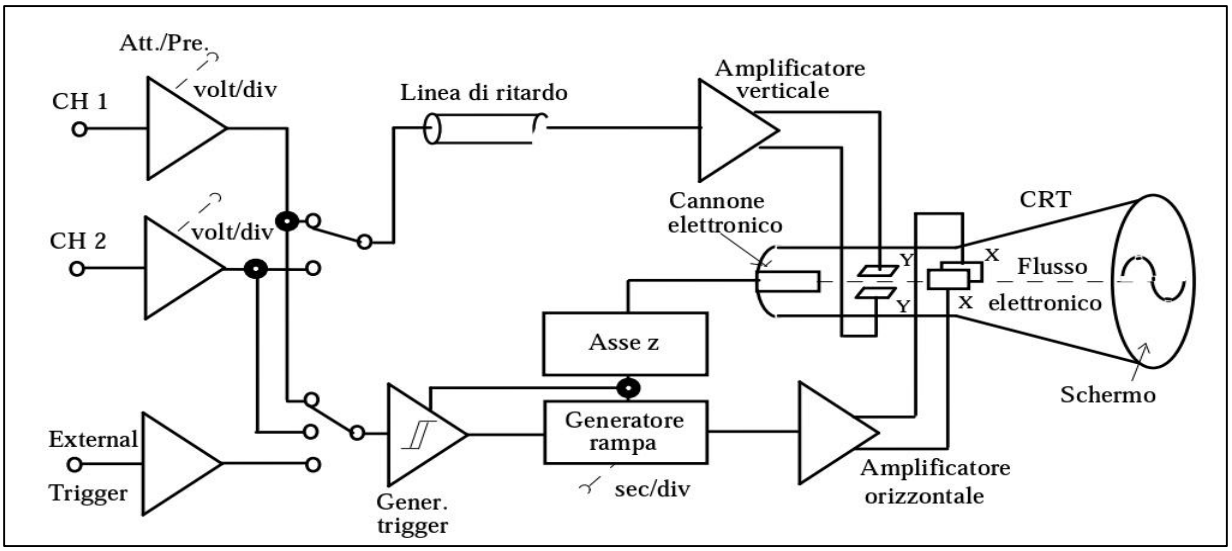
L'evoluzione tecnologica ha portato alla definizione di due tipi di oscilloscopio: di tipo Numerico o Digitale, e di tipo Analogico. Quest'ultimo è antecedente al primo per motivi tecnologici. Al giorno d'oggi il più largamente diffuso è quello digitale anche se l'analogico riveste ancora un notevole utilizzo.

1.1 L'Oscilloscopio Analogico

Di seguito è riportato lo schema a blocchi di un oscilloscopio analogico. Il principio di funzionamento è molto semplice: un cannone elettronico genera un fascio di elettroni che viene deflesso verticalmente e orizzontalmente da due coppie di placche colpendo uno schermo il quale è ricoperto da una particolare sostanza che permette di visualizzare la traccia luminosa che rappresenta il segnale. Le sezioni principali dello strumento sono tre:

1. *Sezione verticale:* controllata dal segnale preso in esame ed agisce sul movimento verticale del fascio di elettroni.
2. *Sezione orizzontale:* tramite un circuito di trigger va a comandare la base dei tempi e quindi il movimento orizzontale del fascio.

3. *CRT (tubo a raggi catodici):* converte i segnali elettrici in immagini.



In linea di principio, quando non si ha un ingresso, non è applicata alcuna tensione alle placche e sarà visualizzato un punto. Supponendo di applicare una tensione periodica a rampa alle placche orizzontali, il punto traccerà una linea orizzontale. Applicando infine una tensione alle placche verticali, sarà visualizzata la forma d'onda del segnale. Quest'ultima ha ampiezza in funzione della tensione applicata alle suddette placche che a sua volta è proporzionale all'ampiezza del segnale in ingresso. Agendo sulla frequenza del segnale a rampa si può aumentare o diminuire l'intervallo temporale visualizzato.

L'elemento fondamentale dello strumento è il tubo a raggi catodici che funge da trasduttore elettro-ottico. Le componenti principali sono:

1. *Il cannone elettronico:* Che, come detto, genera il fascio di elettroni tramite emissione termoionica (emissione di particelle cariche da parte di un materiale riscaldato ad alte temperature attraverso, ad esempio, la corrente elettrica) da un catodo chiuso in una griglia di controllo che presenta un foro dal quale fuoriescono gli elettroni. Il potenziale della griglia è sempre negativo rispetto al catodo in modo da evitare che la stessa sia attraversata da correnti. La differenza di potenziale tra griglia e catodo controlla l'intensità luminosa sullo schermo e può essere regolata dal pannello dello strumento.
2. *Sistema di focalizzazione e accelerazione del fascio:* Permette di convergere il fascio elettronico in modo da diminuirne le dimensioni sullo schermo. Sempre dal pannello dello strumento si possono regolare:
    a. *Fuoco:* che consente di produrre un punto luminoso il più piccolo possibile.

b. *Astigmatismo:* consente di arrotondare il punto luminoso sullo schermo.

3. *Sistema di deflessione orizzontale e verticale:* Costituito da due coppie di placche di deflessione che controllano il movimento orizzontale e verticale del fascio. Gli elettroni, di carica $q = -1{,}6 * 10^{-19}C$ attraversano le placche poste a distanza d tra cui esiste una differenza di potenziale E_d e sono quindi sottoposti ad una forza $F = q\frac{E_d}{d}$ che spinge verso la placca a potenziale maggiore. La deflessione prodotta dalle placche sarà: $D = \frac{E_d l_d L}{2dV_a}$ con E_d tensione di deflessione e V_a tensione tra anodo e catodo che influenza la velocità degli elettroni. Solitamente sono poste prima le placche a deflessione verticale in quanto il movimento verticale è molto più veloce di quello orizzontale.

4. *Schermo:* Lo schermo, di grandezza variabile, è interamente coperto di cristalli di fosforo che assorbono l'energia cinetica degli elettroni ed emettono energia luminosa. Questa proprietà è detta di luminescenza e si divide in *fluorescenza* (emissione luminosa durante l'eccitazione) e *fosforescenza* (emissione residua alla fine dell'eccitazione). Essa è dovuta all'energia acquistata dall'elettrone di un atomo della sostanza fluorescente a seguito dell'urto con gli elettroni emessi dal cannone la quale causa una radiazione luminosa. La durata della fase di fosforescenza è detta persistenza, e l'intensità della stessa è detta luminanza e può dipendere da diversi fattori (velocità d'urto, numero di elettroni, ecc.).

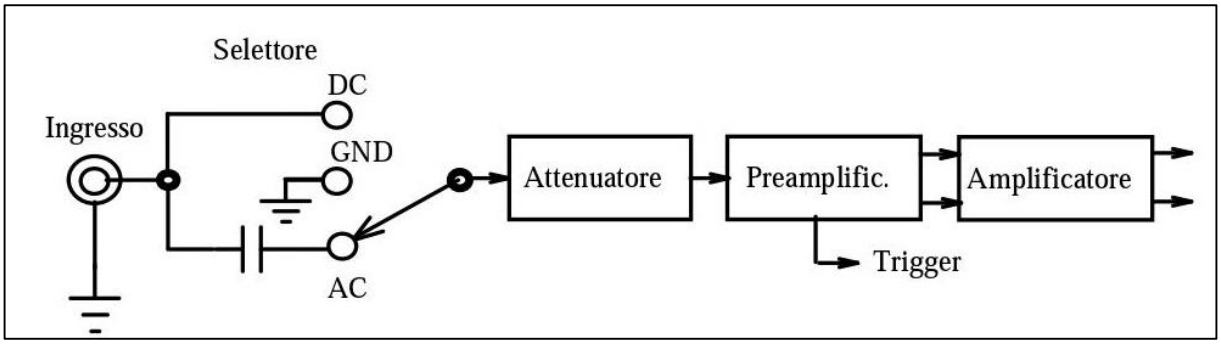
<u>La sezione verticale</u> riceve il segnale in ingresso ed è costituita da:

1. *Selettore d'ingresso:* Posto subito dopo il connettore a cui il segnale viene applicato, ha tre posizioni:

   a. *DC:* Il segnale non subisce condizionamenti.

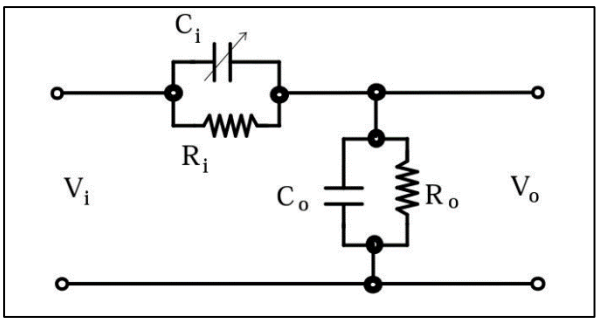
   b. *AC:* Il segnala passa attraverso una capacità che blocca la componente continua del segnale.

   c. *GND:* L'ingresso è posto a massa per centrare la traccia sul reticolo dello schermo.

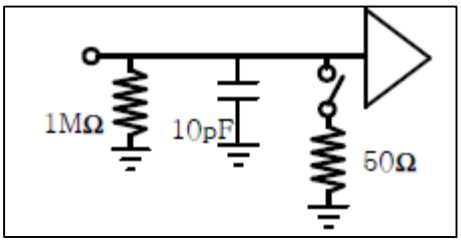


2. *Attenuatore:* Modifica l'ampiezza del segnale in ingresso a seconda di quella che si vuol visualizzare. Al variare della frequenza, deve mantenere inalterato il rapporto di attenuazione. Può, idealmente, essere realizzato con un partitore resistivo. Tenendo conto delle capacità parassite (schematizzata da C_o) possiamo porre in parallelo alla resistenza R_i una capacità C_i in modo che la variazione di impedenza Z_o (data dal parallelo tra C_o e R_o) sia compensata. Questi attenuatori si dicono RC compensati.

$$\frac{V_o}{V_i} = \frac{Z_o}{Z_o + Z_i} = \frac{\frac{R_o}{1 + j\omega R_o C_o}}{\frac{R_o}{1 + j\omega R_o C_o} + \frac{R_i}{1 + j\omega R_i C_i}} = \frac{R_o(1 + j\omega R_i C_i)}{R_o(1 + j\omega R_i C_i) + R_i(1 + j\omega R_o C_o)}$$



Regolando opportunamente C_i in modo da soddisfare la relazione seguente $R_i C_i = R_o C_o$ si ottiene $\frac{V_o}{V_i} = \frac{R_o}{R_o + R_i}$. La compensazione si esegue mediante una opportuna vite di regolazione. Dalle formule si nota che alle basse frequenze il rapporto di attenuazione dipende dalle sole resistenze. L'impedenza d'ingresso è solitamente di $1\,M\Omega$ che consente di rendere trascurabili gli assorbimenti di correnti continue ed a basse frequenze, mentre la capacità parassita è di $10 \div 30\,pF$. In alcuni casi si usano ulteriori resistenze per l'adattamento dei cavi coassiali di collegamento.



3. *Amplificatore verticale:* L'amplificatore verticale ha elevato guadagno (circa 2000) ed essendo il prodotto guadagno-larghezza di banda, fisso (10^6), il progetto dell'amplificatore condiziona le caratteristiche dell'intero dispositivo.

Il sistema di amplificazione è costituito da un preamplificatore a FET e da un amplificatore a guadagno fisso con caratteristiche passa basso. In generale i requisiti richiesti da un sistema di amplificazione sono:

a. *Linearità:* di funzionamento per non introdurre distorsioni.
b. *Banda Passante:* che deve essere la più elevata possibile.
c. *Stabilità:* del guadagno a breve e lungo termine

4. *Linea Di Ritardo:* Oltre ai ritardi introdotti dai circuiti dello strumento, la sezione orizzontale ne aggiunge uno di circa $100\ ns$ dovuto a trigger e base dei tempi. Per cui è necessario ritardare anche la sezione verticale attraverso cavi coassiali ad elevata induttanza per unità di lunghezza.

La sezione orizzontale comprende:

1. *Selettore e circuito di trigger:* Sincronizza la generazione del segnale a rampa per la deflessione orizzontale con il segnale di ingresso che comanda la deflessione verticale. Tale circuito ha un primo commutatore per la selezione della sorgente del segnale di sincronismo che può avere diverse posizioni:

   a. *CH1:* Seleziona il segnale presente sul canale 1.
   b. *CH2:* Seleziona il segnale presente sul canale 2.
   c. *EXT:* Seleziona il segnale sul connettore di trigger.
   d. *LINE:* Seleziona il segnale di tensione sulla linea di alimentazione.

   A questo commutatore ne segue un altro per la selezione del modo di accoppiamento del segnale di sincronizzazione al circuito di trigger. Anche qui abbiamo diverse posizioni:

   a. *DC:* Il segnale è applicato direttamente.
   b. *AC:* Si filtrano le componenti minori di $60Hz$.
   c. *LF REJ:* Si filtrano le componenti minori di $30kHz$..
   d. *HF REJ:* Si filtrano le componenti maggiori di $30kHz$ e minori di $60Hz$.

2. *Base dei tempi:* E' un circuito che ha il compito di fornire una tensione linearmente crescente nel tempo, che varia da un minimo negativo ad un massimo positivo. Il suo

tempo di salita deve essere modificabile in modo da poter regolare il tempo di scansione orizzontale. Il circuito è costituito da un generatore di tensione a rampa, da un circuito che genera un segnale di cancellazione (per eliminare la traccia di ritorno cioè immagini indesiderate) e da un circuito di hold-off.

a. *Generatore di rampa:* è un integratore di Miller costituito da un operazione con una capacità in controreazione. Variando la costante di tempo RC se ne può modificare la pendenza.

b. *Il circuito di hold-off:* Fa sì che si eviti che un nuovo impulso di trigger dia inizio ad una nuova rampa prima che il circuito sia tornato in condizioni di riposo.

c. *Circuito di blanking:* Garantisce l'annullamento dell'immagine di ritorno (che si ha sul fronte decrescente della rampa) generando un impulso negativo che annulla il fascio elettronico. Essenzialmente è un amplificatore-invertitore.

La base dei tempi, ha diverse modalità operative che sono:

1. *Modo trigger:* La visualizzazione è controllata dal segnale di trigger, che provvede a fornire il comando per la partenza della rampa. Quando la rampa ha raggiunto valore massimo, il circuito di hold-off si porta ad in livello alto facendo si che eventuali impulsi provenienti dal circuito di trigger non producano effetti sull'integratore. La durata dell'impulso di hold-off può essere regolata in modo da evitare problemi di sovrapposizioni di immagini.
2. *Modo auto:* Un circuito di auto-trigger è posto tra generatori di impulsi di sincronismo e base dei tempi. Controlla se per un periodo di tempo non vengono generati impulsi e in tal caso ne genera uno. Se il circuito di trigger si attiva, quello di auto-trigger si disattiva automaticamente.
3. *Modo single:* Si usa per segnali non ripetitivi o in caso si stia prendendo in esame il transitorio, dove non è possibile avere una visualizzazione stabile. Si utilizza quindi una particolare fotocamera per registrare l'immagine.

In conclusione, è possibile visualizzare due o più segnali attraverso gli oscilloscopi a doppia traccia. Essi hanno un solo sistema di deflessione verticale che viene alternativamente collegato ai segnali da visualizzare. I due canali hanno comunque attenuatori e preamplificatori separati.

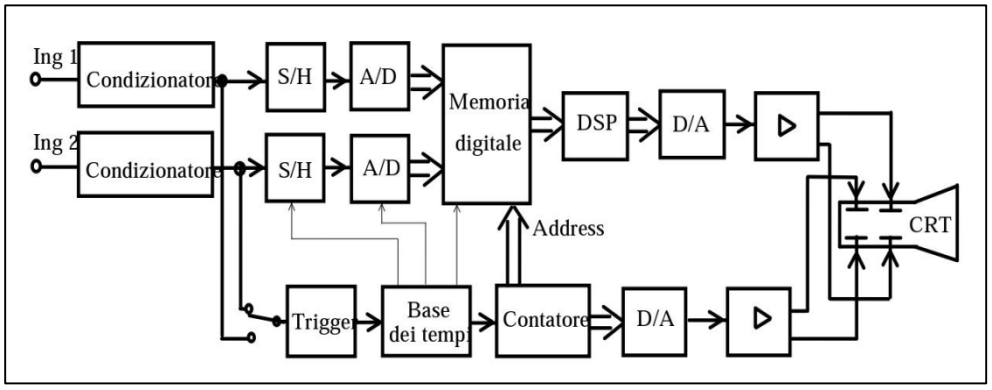
Questi oscilloscopi emulano un comportamento a doppio flusso (cioè l'ingresso simultaneo di due segnali) attraverso due tecniche:

a. *Alternate:* Il commutatore viene controllato da un segnale sincrono con l'impulso di comando della base dei tempi alternando la visualizzazione dei due canali.

b. *Chopped:* Il commutatore è comandato da un apposito segnale e vengono visualizzati tratti successivi dei due segnali. Questa tecnica non può essere utilizzata se la frequenza del segnale è maggiore di quella di commutazione.

1.2 L'Oscilloscopio Digitale

L'oscilloscopio digitale, può essere rappresentato tramite il seguente schema a blocchi:



Dallo schema a blocchi, si può vedere che il segnale, una volta condizionato, viene campionato da un circuito di S/H (Sample and Hold) e quindi convertito in forma numerica ad un apposito ADC. Nei primi modelli, la visualizzazione avveniva mediante un tubo a raggi catodici come nei modelli analogici, utilizzando degli appositivi convertitori numerico-analogici: DAC. Anche le sezioni di condizionamento e di trigger sono simili a quelle degli strumenti analogici. In particolare la prima prevede un selettore, un attenuatore e un preamplificatore. La base dei tempi genera anche segnali di controllo per S/H, ADC e memoria digitale. Un ruolo molto importante è rivestito dall' ADC, presente su ogni canale di ingresso, che deve essere molto veloce e preciso. La scelta degli ADC quindi ricade ADC ad approssimazioni successive, o di tipo flash. La memoria utilizzata quindi deve anch'essa essere veloce in fase di scrittura in modo da riuscire a memorizzare segnali ad elevata frequenza. La fase di lettura può invece essere più lenta in quanto i circuiti a valle della memoria possono operare con tempi maggiori rispetto a quelli di acquisizione. L'ausilio di un oscilloscopio digitale rispetto ad uno di tipo analogico, offre diversi vantaggi quali:

a. Possibilità di acquisire fenomeni transitori.

b. Possibilità di visualizzare la forma d'onda del segnale acquisito in istanti precedenti quello di trigger.

c. Possibilità di memorizzare il segnale acquisito.

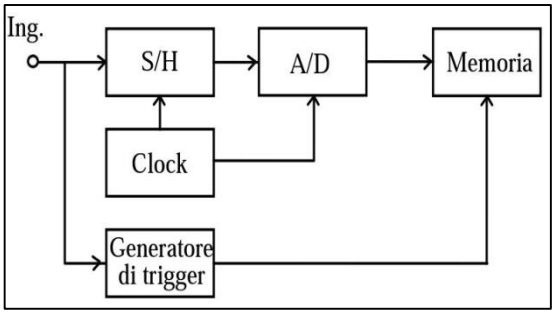
d. Possibilità di controllare lo strumento mediante un elaboratore.

Per l'acquisizione del segnale, possono essere utilizzate diverse tecniche di campionamento che sono:

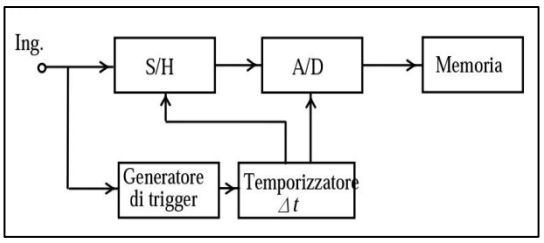
1. *In tempo reale.*
2. *In tempo equivalente di tipo:*
    a. *Sequenziale.*
    b. *Random.*

Si riconoscono quindi tre tipi di oscilloscopi:

a. *Digitalizzatori di transitori:* Il convertitore campiona continuamente il segnale in istanti di tempo scanditi da un generatore di clock ed i campioni sono registrati in locazioni di memoria successivi. Quest'ultima è gestita in maniera circolare ricominciando la scrittura dalla prima locazione di memoria. Il circuito di trigger non si occuperà di dare inizio alla conversione ma di interromperla. Dopo il verificarsi dell'evento di trigger la memoria continua a riempirsi per un numero di campioni post-trigger selezionato dall'operatore. La tecnica di campionamento a tempo reale, si applica nel caso di segnali di tipo transitorio o di segnali ripetitivi. Per il teorema di Nyquist, una rappresentazione significativa del segnale in ingresso si ottiene con frequenza di campionamento superiore al doppio della frequenza armonica di ordine superiore.

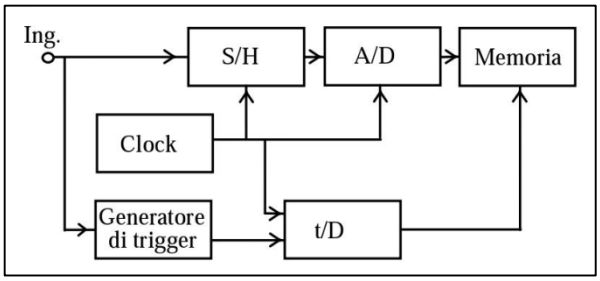


b. *Digitalizzatori a campionamento sequenziale:* Le tecniche di campionamento a tempo equivalente, superano i limiti di banda imposti dallo strumento e di campionare il segnale a frequenza molto minore rispetto quella di Nyquist. D'altro canto queste tecniche sono applicabili a soli segnali di tipo ripetitivo. La tecnica più comune è il campionamento sequenziale a tempo equivalente che consiste nell'acquisire un solo campione ad ogni colpo di trigger in un istante che viene ritardato di un intervallo di tempo costante rispetto l'evento di trigger. Essendo noto il ritardo Δt di acquisizione è possibile ricostruire il segnale. Essendo il



campionamento in tempo equivalente e non reale, il teorema di Nyquist non trova applicazione per cui non è necessario utilizzare ADC ad elevata frequenza di conversione. Questa tecnica consente una rapida ricostruzione dei segnali acquisiti, ma non permette l'acquisizione di eventi pre-trigger.

c. *Digitalizzatori a campionamento random:* Prevede l'ausilio di un blocco t/D che converta in forma numerica il tempo relativo all'acquisizione di un dato campione. Per ogni campione, viene registrato in memoria un valore che mette in relazione l'istante di trigger con quello di acquisizione. Il convertitore è controllato da un generatore di clock e i campioni sono prelevati continuamente senza aspettare l'evento di trigger. Quindi oltre a memorizzare il valore campionato, si memorizza la distanza temporale tra istante di trigger e di campionamento, in modo da poter ricostruire la forma d'onda. Il campionamento si dice random perché i punti sono acquisiti in maniera asincrona rispetto al trigger. Potendo i campioni del segnale, precedere o succedere il punto di trigger, è consentita l'osservazione di eventi precedenti ad esso. Questa tecnica consente l'acquisizione di eventi pre-trigger ed è adatta per una rapida acquisizione di segnali a bassa frequenza.



Applicando la modalità di acquisizione in tempo reale bisogna tener conto del teorema di Nyquist il quale, sinteticamente, asserisce che : *"campionando un segnale a frequenza* $2f$*, dai campioni non è possibile estrarre informazioni relative alle componenti del segnale a frequenza superiore a* f*".* Per cui bisogna scegliere con criterio la frequenza di campionamento. Frequenze superiori a quella di Nyquist possono essere eliminate da opportuni filtri anti-aliasing che evitano, appunto, l'aliasing ma causando la perdita di informazioni circa la presenza di quelle componenti. Si ottiene un'eccellente ricostruzione del segnale se lo stesso è campionato a frequenza $10f$. Per valori inferiori, è necessario sopperire al minor numero di informazioni facendo ipotesi sul segnale. La forma d'onda acquisita, può essere rappresentata secondo diverse modalità:

1. *visualizzando i campioni come punti isolati:* Richiede almeno $20 \div 25$ campioni per ciclo. I campioni per ciclo sono definiti come $Campioni\ per\ ciclo = \frac{f_{campionamento}}{f_{segnale}}$. Se la frequenza del segnale in ingresso aumenta, il numero di campioni per ciclo diminuisce dando

luogo ad un aliasing percettivo (l'occhio umano associa ad ogni punto quello spazialmente più vicino e non quello temporalmente più vicino) che può essere eliminato con l'interpolazione tra punti.

2. *eseguendo un interpolazione fra i punti:* Quest'ultima può essere di tipo:

   a. *Lineare:* Permette di ridurre il numero di campioni per ciclo a 10 con ovvi vantaggi sulle prestazioni. La precisione è in funzione della lunghezza dei vettori, se corti la ricostruzione sarà più fedele.

   b. *Sinusoidale:* Può ridurre il numero di campioni e 2.5 se il segnale è di tipo sinusoidale. Per cui è ottimo a livello di prestazioni ma richiede ipotesi molto forti riguardanti la forma d'onda.

Le modalità di acquisizione per un DSO sono diverse e selezionabili in base a se si acquisisce una singola forma d'onda ed altre quando si acquisisce ripetutamente una stessa forma d'onda. Spesso, non si campiona alla massima frequenza possibile o per motivi di memoria o per motivi di tempo. E' però possibile campionare alla massima frequenza ma acquisendo i campioni a frequenza inferiore. Per l'acquisizione di una singola forma d'onda riconosciamo:

a. *Acquisizione del campione:* Prevede la memorizzazione del campione acquisito in ogni intervallo di acquisizione.

b. *Rilievo dei picchi:* Consiste nel far funzionare l'oscilloscopio alla massima frequenza di campionamento ma con velocità di scansione ridotta.

c. *Smoothing:* Consente di ridurre il rumore sulle forme d'onda visualizzate elaborando indipendentemente ogni forma d'onda attraverso il seguente algoritmo:

   $Y_n = \frac{1}{N}[\sum_{i=1}^{M} Y_{n-i} + \sum_{i=0}^{M} X_{n+i}]$ dove N è il numero di punti elaborati, M è il numero di punti usati da ogni banda, n è la posizione orizzontale, Y_n è il valore calcolato, X_n è il valore di ingresso. L'algoritmo considera due punti (A, B) precedenti un punto C e due punti successivi (D, E). Si calcola la media dei cinque punti e la si inserisce in C, ripetendo l'operazione per i punti successivi.

Per segnali ripetitivi invece riconosciamo le seguenti tecniche:

a. *Inviluppo:* Lo strumento ha tre locazioni di memoria per ogni campione dove registra il valore massimo (a quella iniziale), il valore corrente e il valore minimo (a quella finale).

Ad ogni acquisizione si confronta il valore minimo o massimo corrente con quello memorizzato alle precedenti acquisizioni, ed eventualmente lo aggiorna.

b. *Media:* Consiste nel visualizzare un segnale ottenuto mediando i campioni relativi a più forme d'onda. Lo strumento ha per ogni campione, tante locazioni di memoria quante sono le forme d'onda acquisite. Ci sono tre tipi di media:

   1. *Media Semplice:* Si ha l'acquisizione di M forme d'onda, se ne fa la media e la si visualizza, tramite il seguente algoritmo: $V_m = \frac{\sum_{i=1}^{M} V_i}{M}$. Un inconveniente è il dover attendere l'acquisizione di tutte le forme d'onda.

   2. *Media Pesata:* Ad ogni acquisizione di una nuova forma d'onda, si esegue la media e si aggiorna la visualizzazione. La media si esegue confrontando punto per punto la nuova acquisizione con la precedente e sommando un fattore correttivo: $S_n = S_{n-1} + \frac{V_n - S_{n-1}}{2^p}$ dove S_n è il nuovo segnale visualizzato, S_{n-1} è il segnale visualizzato precedentemente, V_n è il segnale acquisito, p intero tale che 2^p sia la prima potenza di 2 maggiore o uguale del numero di acquisizione corrente.

   3. *Media esponenziale:* La media è pesata esponenzialmente. L'algoritmo è uguale al precedente eccetto che 2^p è sostituito con il numero totale M di medie effettuate: $S_n = S_{n-1} + \frac{V_n - S_{n-1}}{M}$.

Una volta stampata la forma d'onda, essa viene aggiornata secondo tre modalità:

a. *Normale:* La forma viene stampata ad ogni fine acquisizione.

b. *Scan:* Consente di visualizzare in modo continuo i campioni man mano che vengono acquisiti. Il campione viene visualizzato partendo dalla sinistra.

c. *Roll:* La visualizzazione avviene ponendo il nuovo campione alle destra dello schermo.

Essendo il segnale, negli oscilloscopi digitali, campionato e quantizzato dall'A/D, qualunque sia la risoluzione dello stesso, si avrà sempre un'incertezza dovuta al fatto che esiste sempre un intervallo in cui valori analogici produrranno lo stesso risultato. Le dimensioni di questo intervallo è la risoluzione verticale del DSO ed è espressa dal numero di bit usati per rappresentare l'ingresso analogico. La risoluzione orizzontale invece, è il numero di campioni che può essere visualizzato. Sarà maggiore quanto più grande è il numero di campioni. E' inoltre possibile aumentarla, aumentando la dimensione della memoria. Una memoria estesa consente inoltre di acquisire segnali di elevata durata mantenendo una frequenza di campionamento

altrettanto elevata, e migliora la precisione nella ricostruzione. Alcune particolare operazione che i DSO, a differenza di quelli analogici, permettono di svolgere, sono il calcolo della FFT, e la misura di parametri impulsivi come tempo di salita e discesa, tensione di picco, durata di un impulso, ecc.

Altro elemento molto importante sono le sonde che permettono il collegamento tra lo strumento e il circuito in esame. Le sue funzioni sono di trasmettere il segnale correttamente, attenuare o amplificare il segnale, convertire particolari segnali, ecc. In generale una sonda è costituita da tre parti:

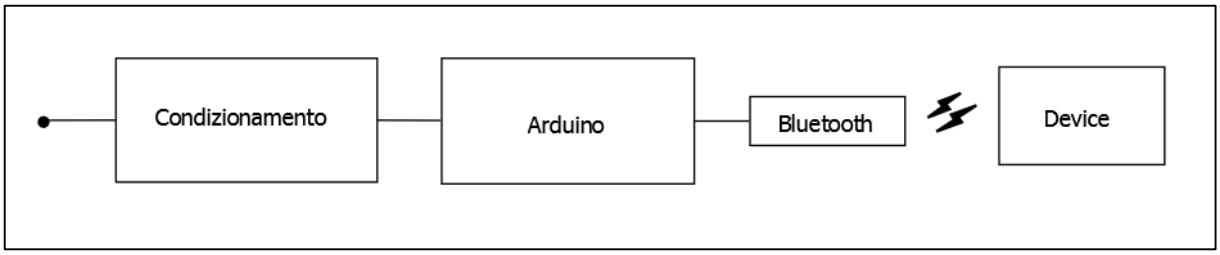
1. *Puntuale:* che preleva il segnale.
2. *Cavo di interconnessione.*
3. *Terminazione:* impedisce che il segnale venga riflesso.

Le sonde sono di tre tipi:

1. *Sonde Passive:* Possono avere diversi rapporti di attenuazione (1: 1 10: 1 50: 1 100:1). Alcuni oscilloscopi hanno particolari dispositivi che consentono l'individuazione in modo automatico del fattore di attenuazione e la regolazione del guadagno dell'amplificatore. La compensazione della sonda può essere fatta sia all'inizio che alla fine del cavo.
2. *Sonde Attive:* In queste sonde il segnale subisce condizionamenti da parte di circuiti attivi. L'alimentazione di tali circuiti può essere prelevata dall'oscilloscopio. Il circuito d'ingresso è solitamente un amplificatore a FET a guadagno unitario con lo scopo di eliminare i problemi dovuti alla capacità del cavo.
3. *Sonde Differenziali:* Permettono la visualizzazione di segnali tra due nodi nessuno dei quali connesso a terra. Queste sonde sono sia attive che passive e forniscono in uscita un segnale riferito rispetto a terra.

2. L'Oscilloscopio Portatile

L'idea alla base del lavoro svolto, è quella di riprodurre, nei limiti delle capacità personali e soprattutto delle tecnologie utilizzate, un oscilloscopio in versione tascabile. In particolare, in questa sezione, se ne considera una sua versione preliminare. L'esigenza nasce prima di tutto a livello di portabilità, in quanto lo strumento si limiterebbe all'ausilio di un telefono (ormai strumento quotidiano), e di un controllore Arduino del quale esistono versioni, diversamente da quella utilizzata in questa sede, molto piccole. Oltre alla portabilità, anche l'aspetto economico è di particolare interesse in quanto il costo di produzione, e quindi di vendita, di questa versione del DSO, resta molto inferiore al più economico tra gli attuali in commercio avendo fatto uso di tecnologie ormai di uso quotidiano e al quale chiunque può accedere. Un altro punto di forza è l'universalità dello strumento. Infatti si consideri lo scenario in cui si abbia un guasto al device e diventi malauguratamente inutilizzabile, basterà reinstallare l'applicazione su un nuovo device per poter riutilizzare lo strumento. Medesimo scenario nel caso in cui si guasti l'Arduino, od in generale lo stampato che si sta utilizzando, basterà sostituirlo affinché il tutto rifunzioni. Basterà inoltre sostituire l'hardware del nostro oscilloscopio per meglio adattare lo strumento alle nostre esigenze. Ad esempio basterà utilizzare un controllore che monti un ADC a frequenza di funzionamento maggiore, affinchè lo strumento sia più preciso. Si può quindi dire che lo strumento gode di modularità che lo rende estremamente versatile per quanto riguarda le esigenze del singolo. Gli utilizzi al quale lo strumento è rivolto sono i medesimi, seppur nei limiti di funzionamento imposti, di quello classico, ovvero la visualizzazione dell'andamento nel tempo di segnali. Definiti gli aspetti generali dello strumento, di cui si è riportato lo schema



a blocchi, si ritiene opportuno dividere la progettazione dello stesso sotto due punti di vista, quello hardware e quello software. In ognuna di queste sezioni si parlerà delle tecnologie utilizzate e delle sue applicazioni al progetto

stesso. Per quanto riguarda la versione dello strumento che fa uso di Arduino, lo schema a blocchi risulta estremamente semplice. E' presente un semplice blocco di condizionamento

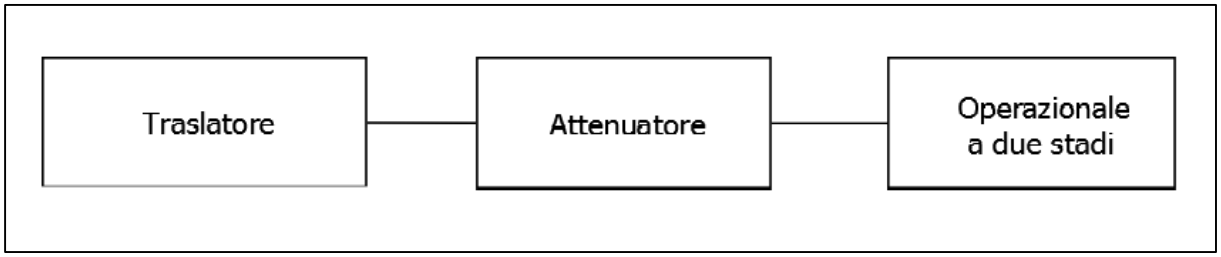
iniziale che permette di adattare il segnale in ingresso in modo da poter essere ricevuto, senza problemi, da Arduino. Arduino riceverà il segnale e dopo opportune operazioni lo invierà, tramite il modulo Bluetooth, ad un determinato device il quale si occuperà di stampare la forma d'onda.

2.1 L'Hardware

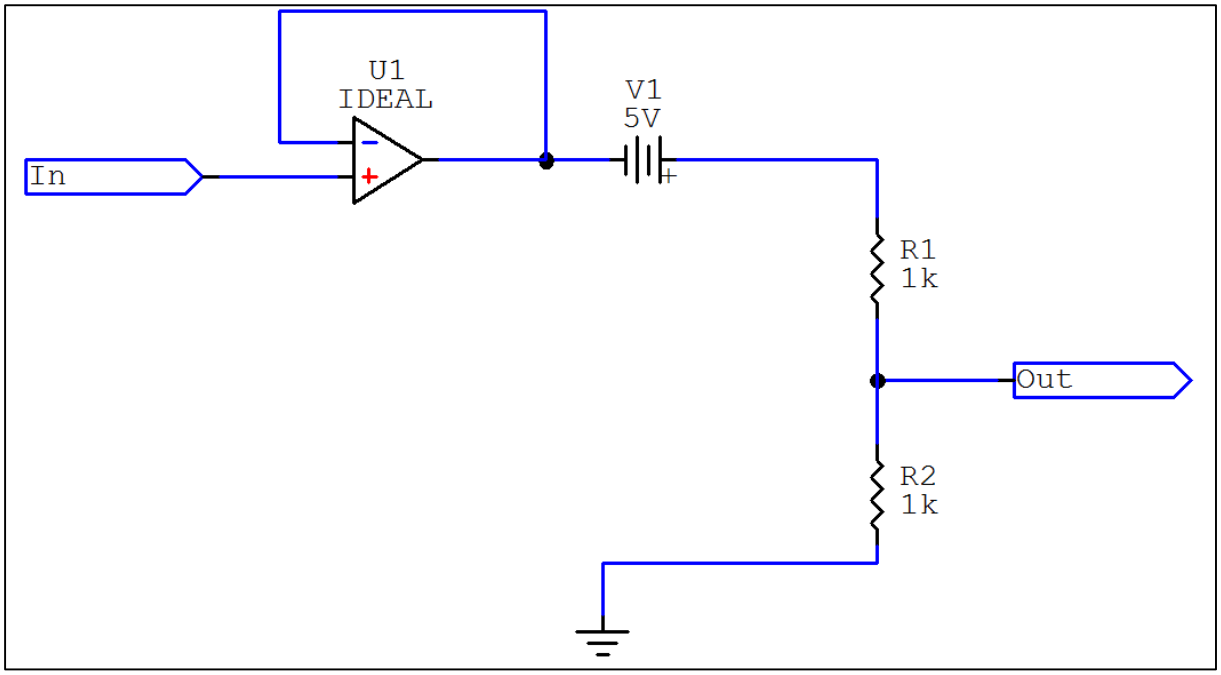
L'unica nota da fare riguardante l'hardware, in questa versione dello strumento, è che i valori dei parametri circuitali sono puramente indicativi in quanto versione preliminare non messa in pratica.

2.1.1 Condizionamento

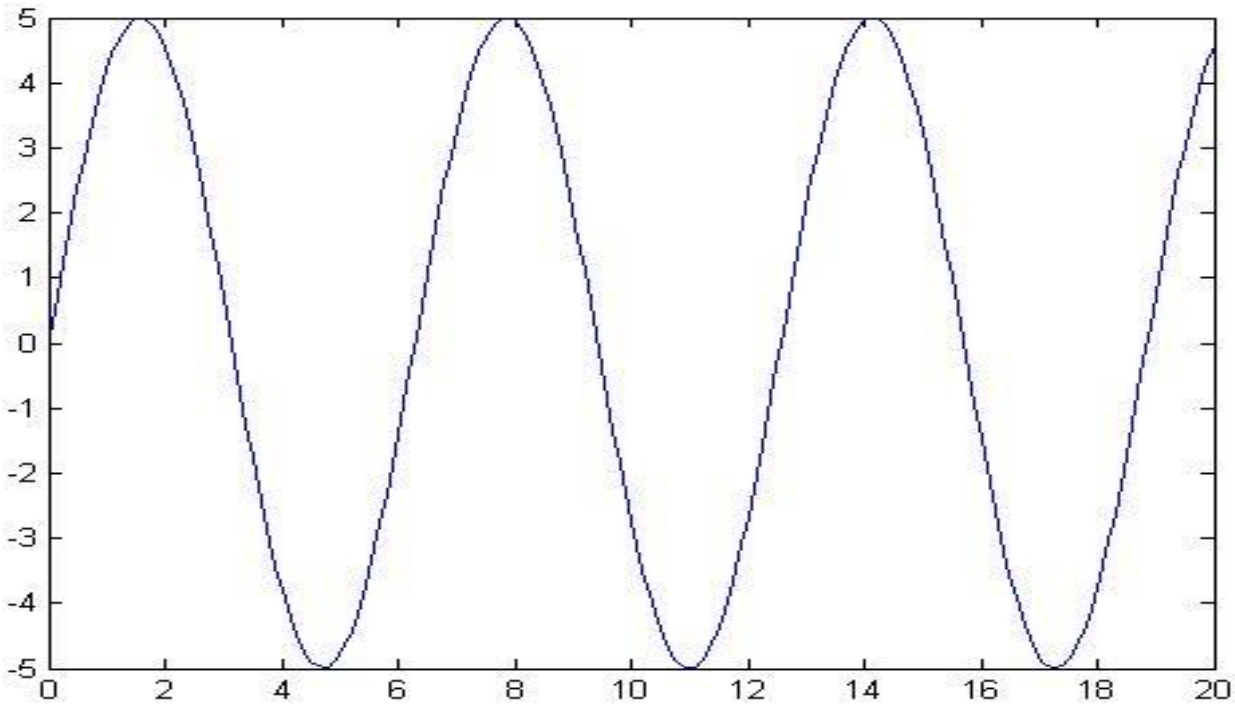
Il blocco di condizionamento, come si può vedere, è costituito da tre blocchi. Il primo è formato da una semplice batteria iniziale da $5\ V$ e da un partitore di tensione. Il senso di questo circuito è il dato in ottemperanza al fatto che, come specificato nella sezione 2.1.2, Arduino può ricevere unicamente tensioni positive nel range $0 - 5\ V$.

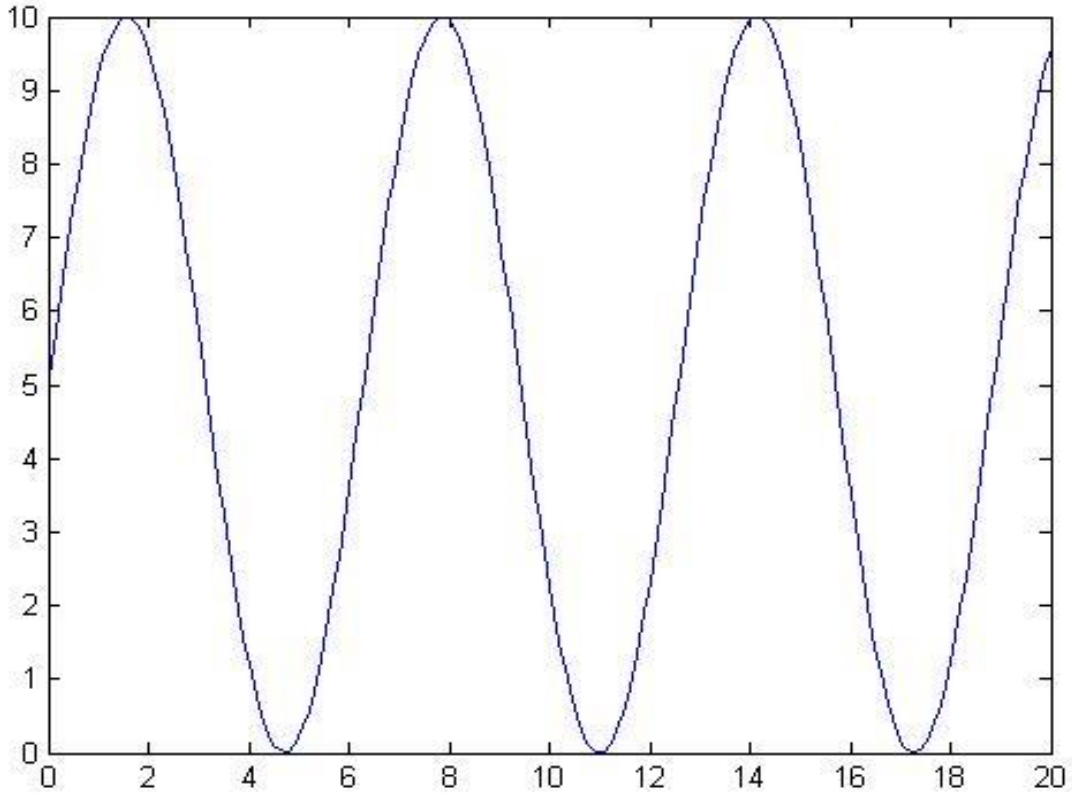


Stabilito il range di funzionamento dell'oscilloscopio che, in questa sede, si è deciso per $(-5, 5)\ V$, risulta evidente un problema riguardante le tensioni in ingresso all' Arduino il quale, senza il seguente blocco, potrebbe essere irreversibilmente danneggiato.

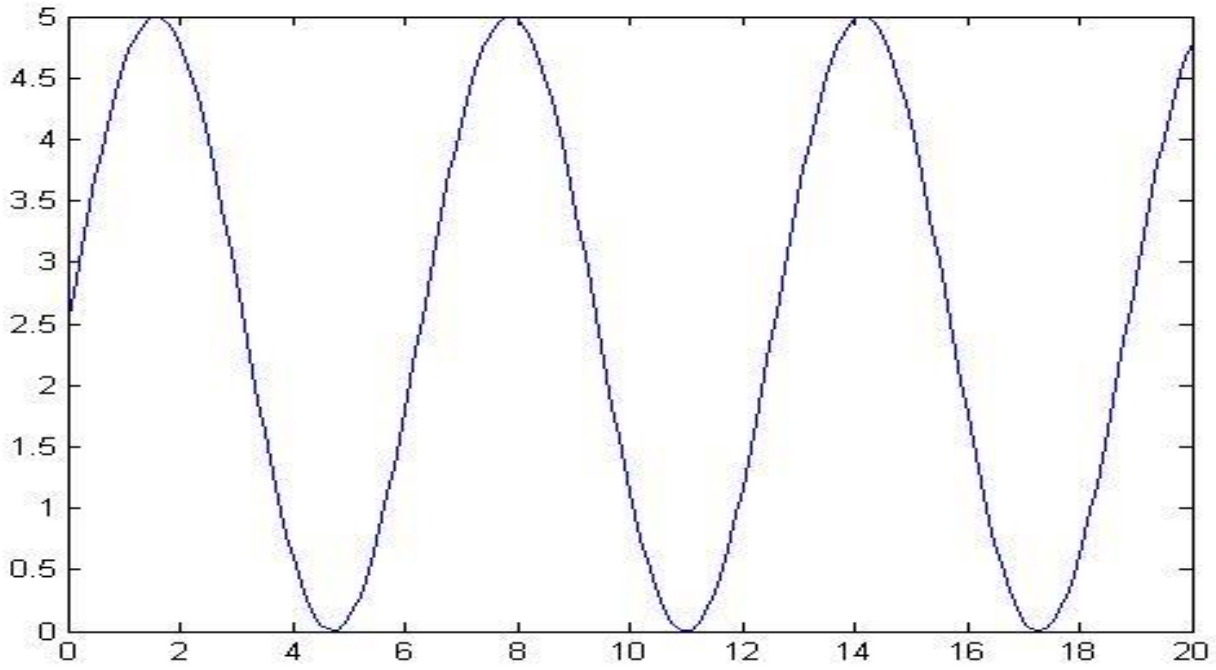


Quello che quindi si fa, è traslare qualsiasi segnale in ingresso, di $5\,V$ rispetto l'asse delle ordinate. Questo fa si che un segnale, supponiamo una sinusoide di ampiezza $(-5, 5)\,V$, sia traslata fino ad assumere ampiezza $(0, 10)\,V$ e che quindi diventi positiva come si può vedere dalle immagini.

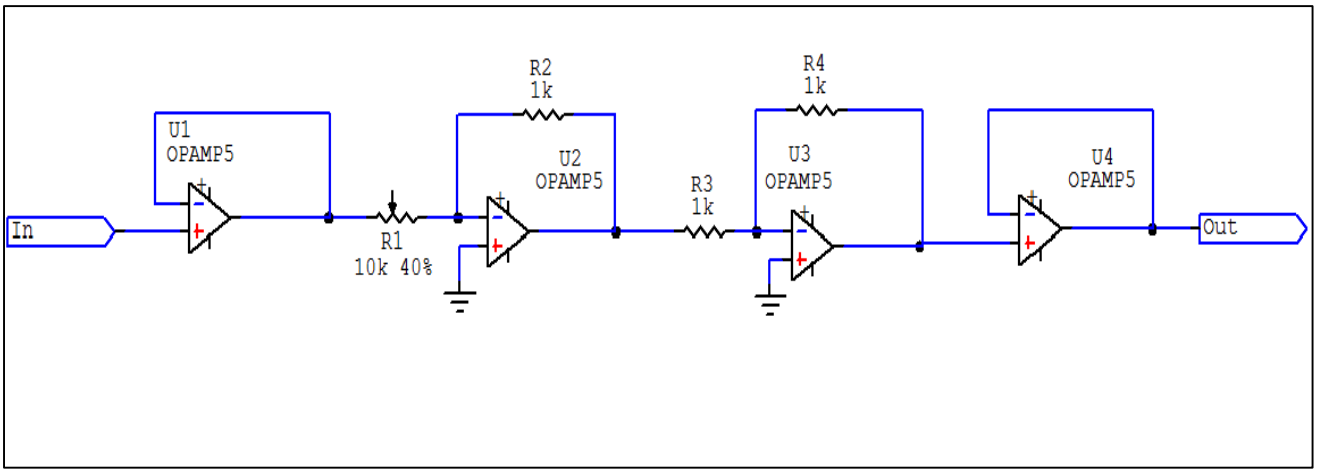




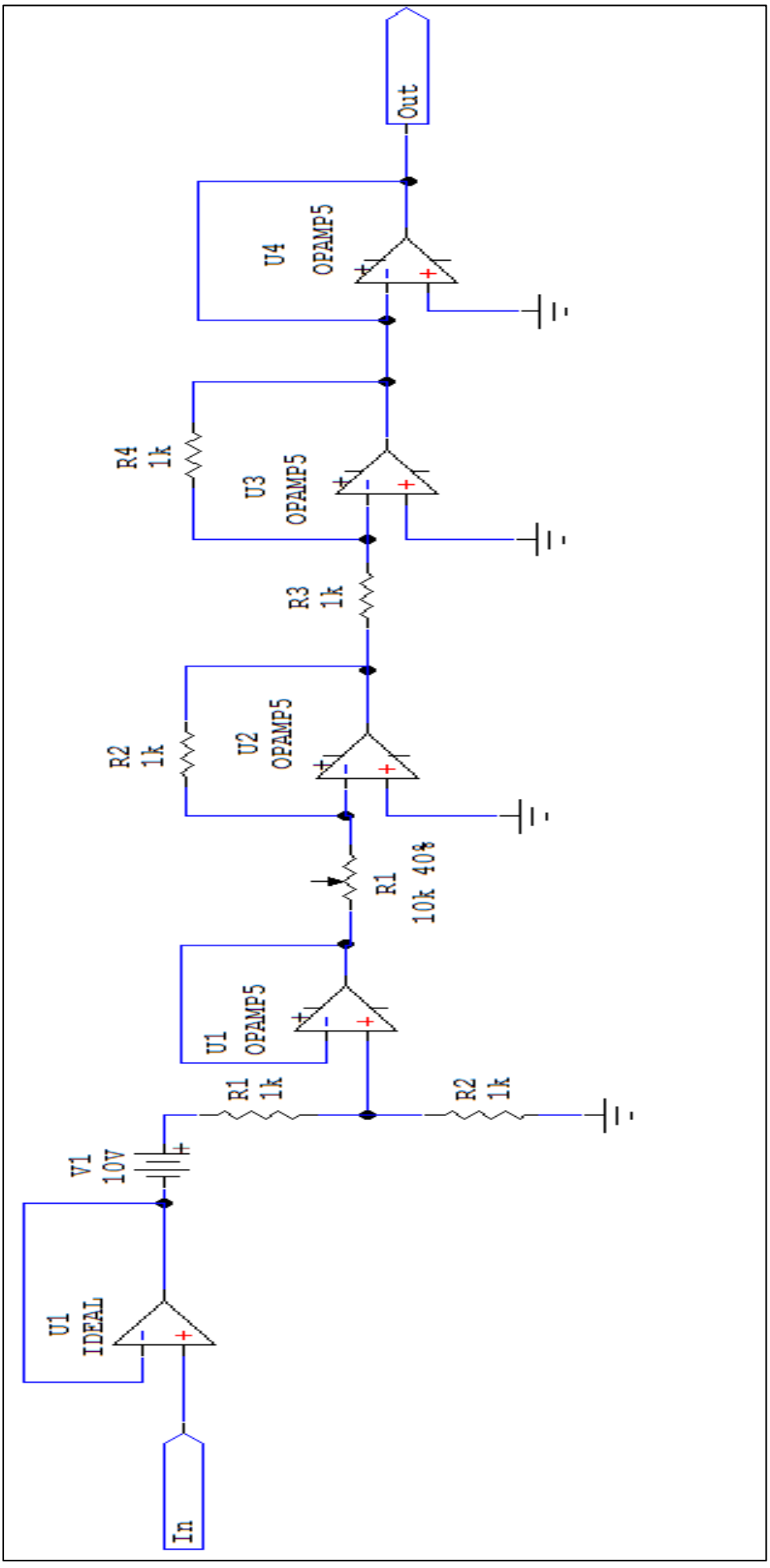
Ora sorge un altro problema riguardante l'ampiezza. Infatti si è detto che Arduino può ricevere tensioni comprese tra $0 - 5\ V$. Ma avendo effettuato una traslazione abbiamo un ampiezza di $(0, 10)\ V$. Qui interviene il partitore di tensione. Esso è composto da due semplici resistenze serie e si prende, come uscita, la caduta di tensione su R_2. Le resistenze sono uguale di grandezza $R_1 = R_2$. Questo perché dalla legge di Ohm risulta: $V_{in} = I(R_1 + R_2)$ da cui $I = \frac{V_{in}}{R_1+R_2}$. Essendo la tensione in uscita data dalla caduta di tensione su R_2 risulta che $V_{out} = IR_2 = \frac{R_2}{R_1+R_2}V_{in}$ ed essendo $R_1 = R_2$ risulta $V_{out} = \frac{1}{2}V_{in}$. Dall'esempio in esame, essendo V_{in} di ampiezza, al massimo di $10\ V$, si ha che la tensione in uscita risulta dimezzata portandoci al caso di una sinusoide di ampiezza $(0, 5)\ V$ adatta per essere ricevuta da Arduino. Ovviamente questa modifica del segnale sarà poi compensata via software in modo da avere l'ampiezza del segnale originale. I valori scelti impongo quindi, che l'ampiezza del segnale in ingresso sia al più compresa tra i $(-5, 5)\ V$.



Come si può vedere il secondo blocco è costituito da due inseguitori di tensione (voltage follower) realizzati mediante amplificatore operazionale, e un amplificatore operazionale a due stadi. Gli inseguitori di tensione sono impiegati per il trasferimento di una tensione da un primo circuito, ad elevato livello di impedenza, ad un secondo circuito ad un livello di impedenza inferiore, disaccoppiando la sorgente del segnale dal resto del circuito. Il disaccoppiamento consente di non assorbire corrente dalla sorgente del segnale riducendo così gli effetti di carico. Può essere realizzato molto semplicemente riportando il segnale in uscita, all'ingresso invertente di un amplificatore operazionale a guadagno unitario, ed applicando il segnale in ingresso al suo ingresso invertente. L'operazionale a due stadi, è costituito da un primo operazione di tipo invertente costituito da un potenziometro (una resistenza variabile) R_1 di



valore $R_1 \in [R_2, xR_2]$ con $x > 1, x \in \mathbb{R}$ e da una resistenza R_2. Essendo che per l'operazione vale la relazione $V_{out} = A_1 V_{in}$ con A_1 che vale, nel nostro caso, $A_1 = -\frac{R_2}{R_1}$, abbiamo che $A_1 \in \left[-\frac{1}{x}, -1\right]$ e quindi $V_{out} \in \left[-\frac{V_{in}}{x}, -V_{in}\right]$. Cioè regolando opportunamente il potenziometro possiamo riportare in uscita lo stesso segnale (invertito) in ingresso, oppure attenuarlo di un fattore x. Il secondo stadio invece, è un altro amplificatore invertente con $A_2 = -\frac{R_4}{R_3}$. Essendo la larghezza di banda di un operazione definita come $GBW = A_2 f_t = 10^6\ MHz$, ponendo la frequenza di taglio f_t pari alla frequenza massima che l'ADC di Arduino può ricevere, che è, come si vedrà, $f_t = 10\ kHz$ si ha che $A_2 = \left|-\frac{R_4}{R_3}\right| = \frac{10^6}{10*10^3} = 100$ da cui si ricava la relazione $R_4 = 100R_3$. Per cui questo operazionale, oltre a rinvertire il segnale compensando il primo operazionale invertente, funge da filtro passa basso. Infine il segnale passa attraverso il secondo inseguitore di tensione che inverte, compensando il primo, il segnale, per poi arrivare al modulo Arduino.


Out
U4
OPAMP5
R4
1k
U3
OPAMP5
R3
1k
R2
1k
U2
OPAMP5
R1
10k 40%
U1
OPAMP5
R1
1k
R2
1k
V1
10V
U1
IDEAL
In

2.1.2 Arduino

L'hardware di Arduino, si basa su un circuito stampato che integra un microcontrollore (che può variare nelle diverse versioni) ad 8 bit prodotti dalla ATMEL. Molte schede includono inoltre un regolatore di tensione a 5 volt, e un oscillatore a cristallo solitamente a $16\ MHz$. Al microcontrollore sono affiancati, per l'interazione con il mondo esterno, numerosi pin di I/O, grazie ai quali Arduino può ricevere segnali raccolti da sensori esterni. In base ai valori ricevuti, e dalle decisioni determinate dal programma in esecuzione sulla scheda, il microcontrollore gestisce il comportamento della stessa grazie all'ausilio di opportuni attuatori. Essendoci diverse versioni di Arduino, si prenderà in considerazione quella utilizzata al progetto ovvero Arduino Uno. Esso offre 14 connettori per l' I/O digitale le quali direzioni di funzionamento sono decise dallo "sketch" (il programma caricato sul microcontrollore). Sei di questi canali di I/O sono in grado di produrre segnali PWM (pulse-width modulation) che permette di ottenere una tensione media variabile dipendente dal rapporto tra la durata dell'impulso positivo e di quello negativo. Sono inoltre presenti altri 6 connettori dedicati a ingressi di segnali analogici che sono collegati ad un ADC a 10 bit e quindi a 1024 livelli di quantizzazione. Questi sono in grado di ricevere una tensione compresa tra $0 - 5\ V$ e grazie all'A/D di convertirla in un intero compreso tra $0 - 1023$ questo fa si che si abbia una risoluzione di $\frac{5}{1024} = 4.9\ mV$ che, entro certi limiti, può essere modificata con l'ausilio di apposite funzioni. La frequenza di campionamento dell'ADC è ricavata dalle seguenti relazioni: $\begin{cases} f_{ck} = 16\ MHz \\ f_{ADC} = \frac{f_{ck}}{p.f.} \\ f_{sample} = \frac{f_{ADC}}{13} \end{cases}$ dove $p.f.$ è il prescale factor e 13 sono i periodi di clock usati dall'ADC. Di default il $p.f.$ è impostato a 128 ma può essere cambiato fino ad portarlo ad un massimo consigliabile di 16 per aumentare la frequenza di campionamento. Considerandolo 128 $f_{ADC} = 125\ kHz$ da cui quella di campionamento è $f_{sample} \approx 10\ kHz$ da cui, dal teorema di Shannon, per una buona ricostruzione del segnale la massima frequenza del segnale in ingresso deve essere $f_{Max} = \frac{f_{sample}}{2} \approx 5\ kHz$. Vi sono inoltre alcuni pin con speciali funzioni. Oltre ai già menzionati pin PWM, si riconoscono quelli di alimentazione tra i quali abbiamo:

a. V_{in}: La tensione in ingresso alla board Arduino quanto è alimentato esternamente.

b. $5\ V$: Fornisce la tensione stabilizzata utilizzata per comandare il microcontrollore e altre componenti sulla board.

c. $3.3\ V$: Fornisce una tensione generata dal regolatore sulla board.

d. GND: Pin di terra.

Abbiamo inoltre pin con funzioni speciali, tra i quali:

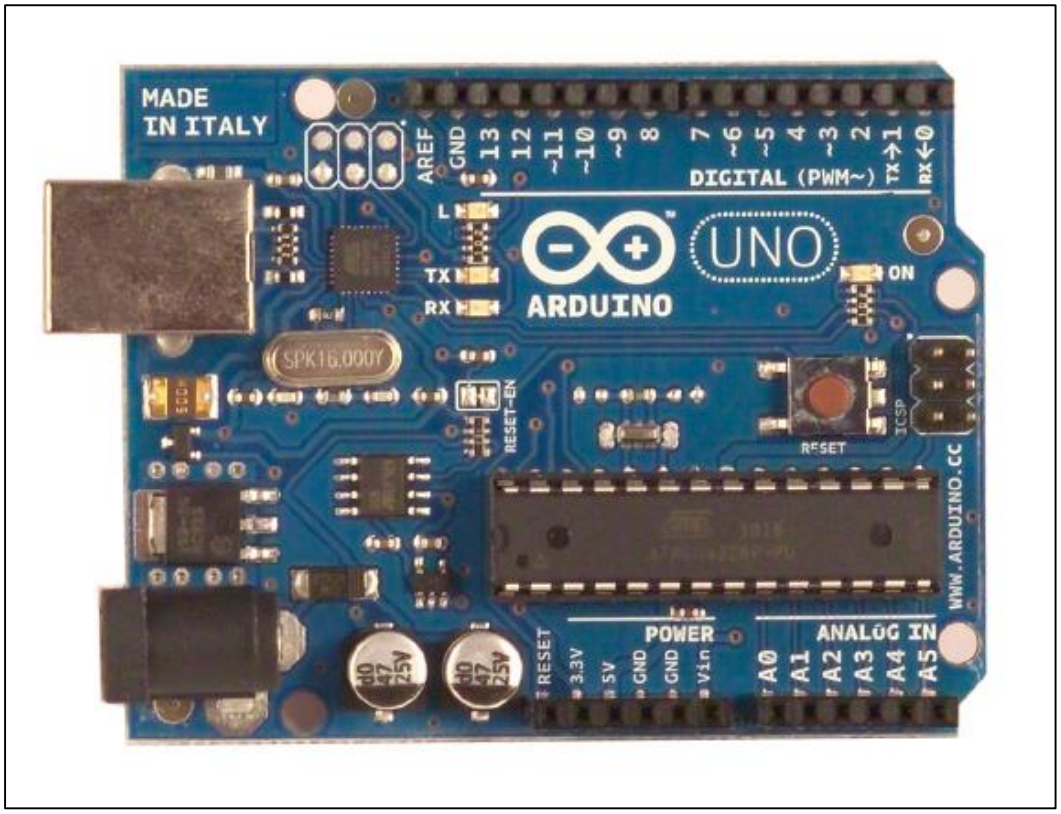
a. *Serial: 0 (RX) 1 (TX)* usati per ricevere e trasmettere dati seriali.

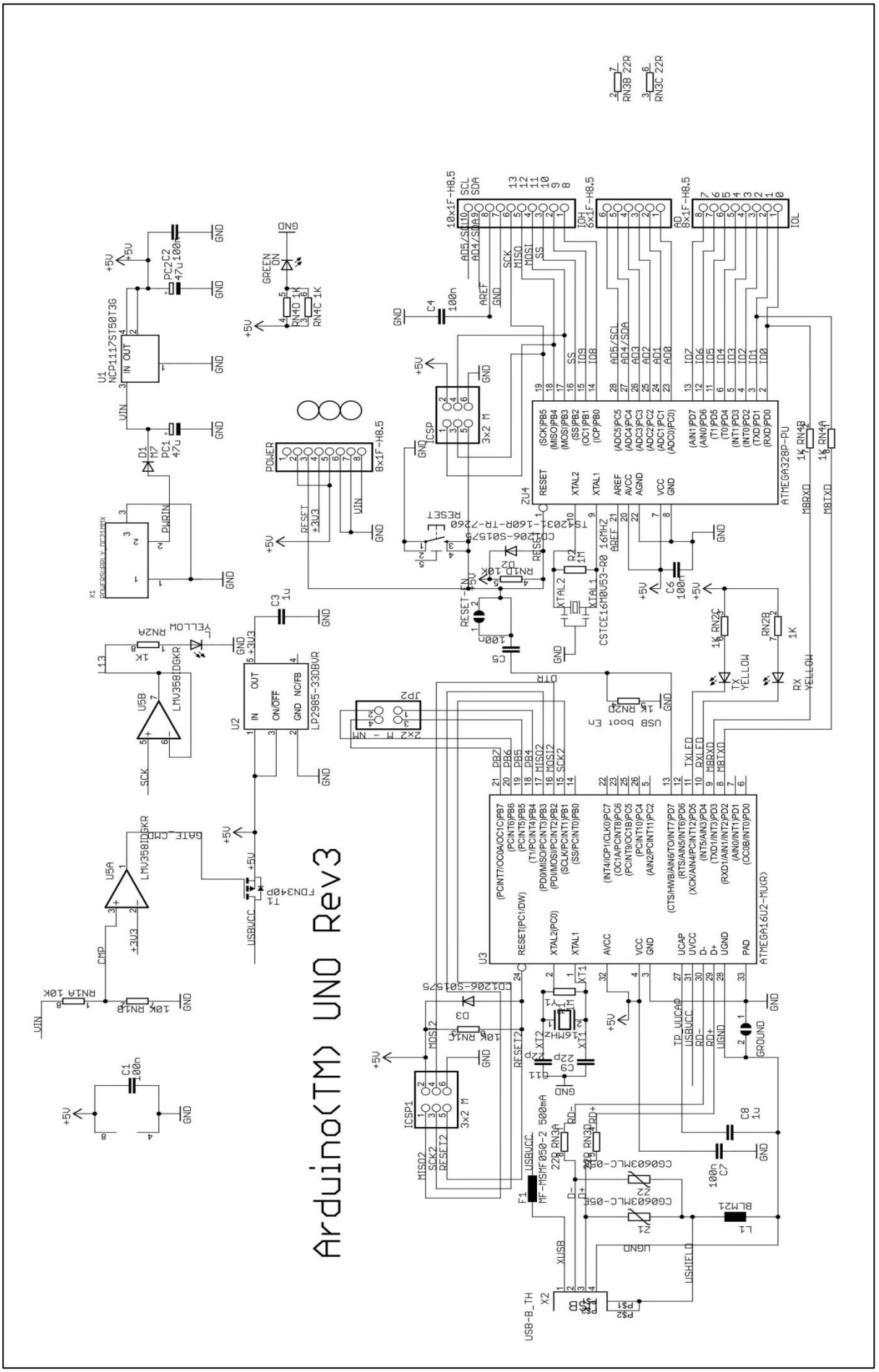
b. *SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK).* Questi pin supportano la comunicazione SPI (Serial Peripheral Interface) usata dal microcontrollore per comunicare rapidamente con dispositivi su brevi distanze. Può essere usata, ad esempio, per permettere la comunicazione tra due microcontrollori. Essa avviene quindi tra due dispositivi, uno master e uno slave. Il master emette un segnale sul pin SS per scegliere con quale dispositivo comunicare. Emette inoltre, grazie al pin SCK, un segnale di clock per la sincronizzazione. Il MISO consente di trasmettere dallo slave al master e il MOSI permette il contrario.

c. *External interrupts: 2, 3* Possono essere configurati per rispondere ad un determinato evento (come il cambiamento di stato del pin).

d. *AREF:* Voltaggio di riferimento per gli ingressi analogici.

Ed altri. In ogni caso la corrente attraversante i pin, non deve superare i $40\ mA$. L'Arduino Uno può ricevere l'alimentazione in diversi modi, o tramite il connettore USB dal quale preleva una tensione di circa $5\ V$ o tramite l'apposito connettore di alimentazione dal quale riceve una tensione compresa tra i $7 - 12\ V$ che viene portata a $5\ V$ tramite un regolatore di tensione. Sono inoltre ammesse altre modalità di alimentazione che prevedono l'ausilio dei pin analogici sopra illustrati. La scheda possiede tre diversi tipi di memorie. Una memoria flash fornita dal microcontrollore, delle dimensioni di $32\ KB$ di cui $0.5\ KB$ usati per il bootloader, una SRAM da $2\ KB$ e EEPROM da $1\ KB$ (tipo di memoria a transistor non volatile). Nella figura di può vedere una board Arduino.


MADE
IN ITALY
AREF
GND
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TX→1
RX←0
DIGITAL (PWM~)
L
TX
RX
ARDUINO
UNO
ON
SPK16.000Y
RESET-EN
RESET
ICSP
WWW.ARDUINO.CC
POWER
ANALOG IN
RESET
3.3V
5V
GND
GND
Vin
A0
A1
A2
A3
A4
A5


Arduino(TM) UNO Rev3

2.1.3 Modulo Bluetooth

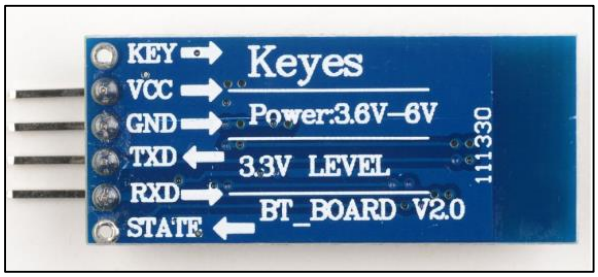
Il modulo Bluetooth utilizzato è il modulo HC-06. Come si può vedere dalle immagini presenta sei pin di cui quattro, i principali, già saldati che sono:

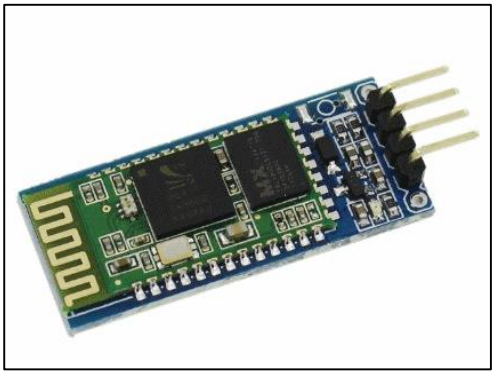
a. *RXD:* Ingresso seriale.
b. *TXD:* Uscita seriale.
c. V_{cc} *:* Tensione di alimentazione del modulo che va dai 3.6 ai 6 Volt.
d. *GND:* Pin per la connessione a massa.

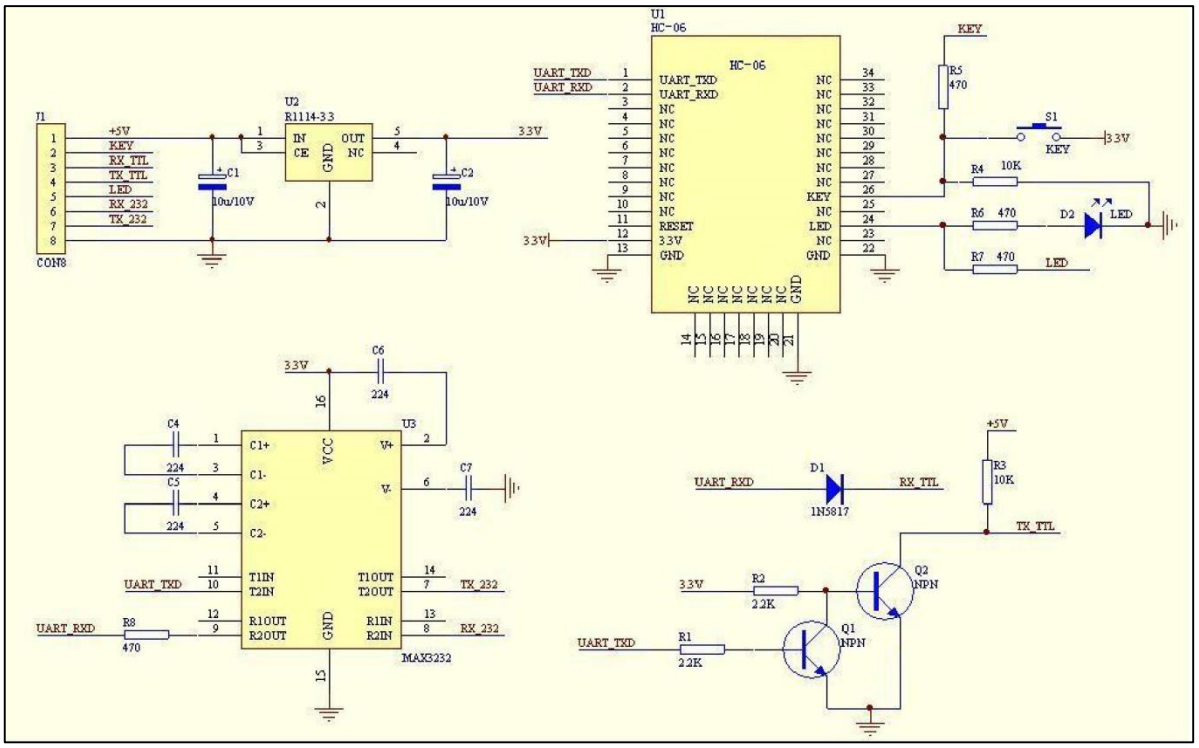
I pin non saldati invece sono:

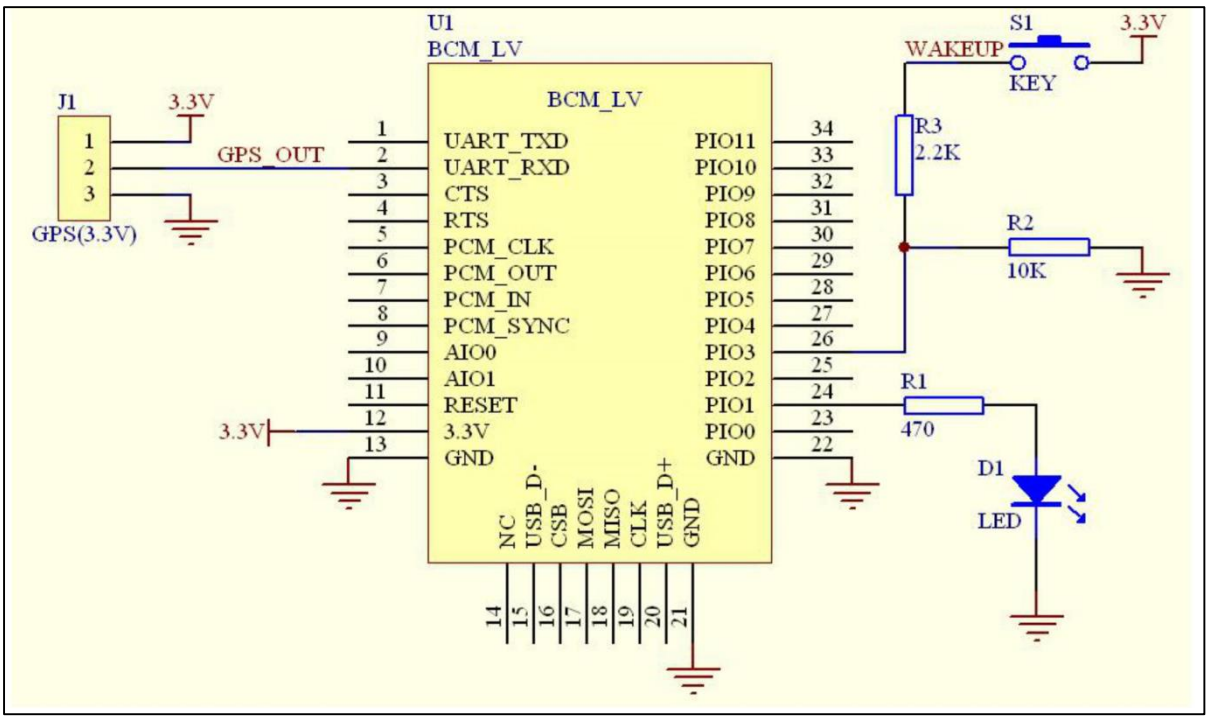
a. *Key:* Pin utilizzato per entrare in modalità AT.
b. *State:* Può facoltativamente essere collegato. Questo pin restituisce lo stato della connessione Bluetooth.

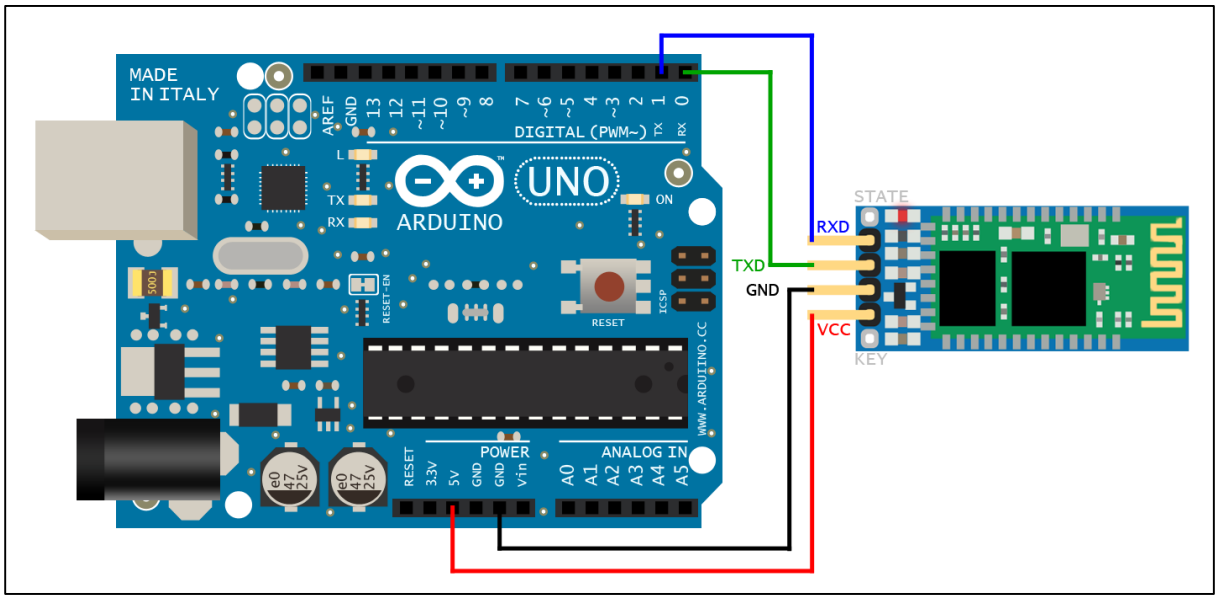
Alcune delle caratteristiche che hanno spinto a scegliere questo modulo in particolare sono il basso consumo e il basso costo. Ai fini del progetto infatti non erano richieste particolari specifiche per lo stesso. Di seguito si riporta lo schema elettrico del modulo ed un generico collegamento tra lo stesso e la board Arduino.






U1
HC-06
HC-06
UART_TXD
UART_RXD
NC
RESET
3.3V
GND
KEY
LED
U2
R1114-3.3
IN
OUT
CE
GND
NC
C1
C2
10u/10V
J1
CON8
+5V
KEY
RX_TTL
TX_TTL
LED
RX_232
TX_232
3.3V
S1
R5
470
R4
10K
R6
470
D2
R7
470
C6
224
U3
C1+
C1-
C2+
C2-
VCC
V+
V-
C4
C5
C7
T1IN
T2IN
T1OUT
T2OUT
R1OUT
R2OUT
R1IN
R2IN
MAX3232
R8
470
D1
1N5817
R3
10K
R2
2.2K
R1
2.2K
Q1
NPN
Q2
NPN


U1
BCM_LV
BCM_LV
J1
GPS(3.3V)
GPS_OUT
3.3V
UART_TXD
UART_RXD
CTS
RTS
PCM_CLK
PCM_OUT
PCM_IN
PCM_SYNC
AIO0
AIO1
RESET
GND
PIO11
PIO10
PIO9
PIO8
PIO7
PIO6
PIO5
PIO4
PIO3
PIO2
PIO1
PIO0
NC
USB_D-
CSB
MOSI
MISO
CLK
USB_D+
S1
WAKEUP
KEY
R3
2.2K
R2
10K
R1
470
D1
LED

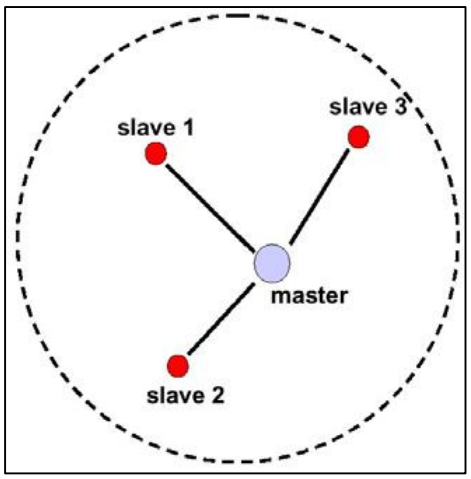

MADE
IN ITALY
UNO
ARDUINO
DIGITAL (PWM~)
POWER
ANALOG IN
RESET
ON
TX
RX
L
WWW.ARDUINO.CC
STATE
RXD
TXD
GND
VCC
KEY

2.2 Il Software

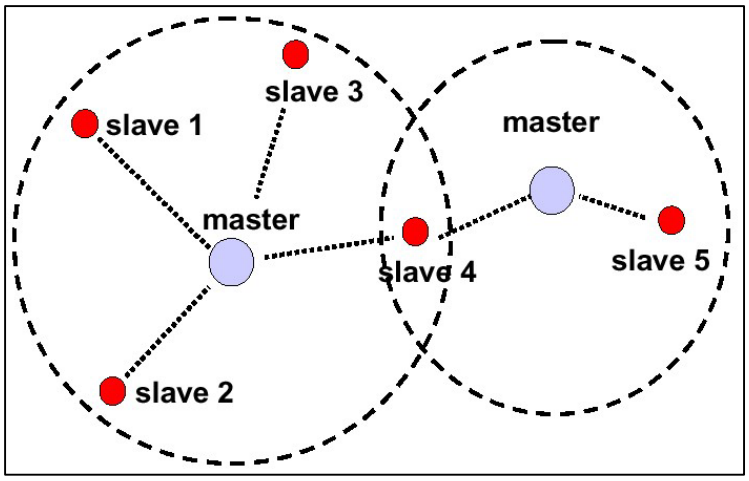
Le tecnologie software utilizzate sono diverse. In particolare si va dai protocolli di comunicazione per l'invio dei dati, ai linguaggi di programmazione usati per, appunto, programmare il modulo Arduino e l'applicazione che sarà installata sul device.

2.2.1 Generalità sul protocollo Bluetooth

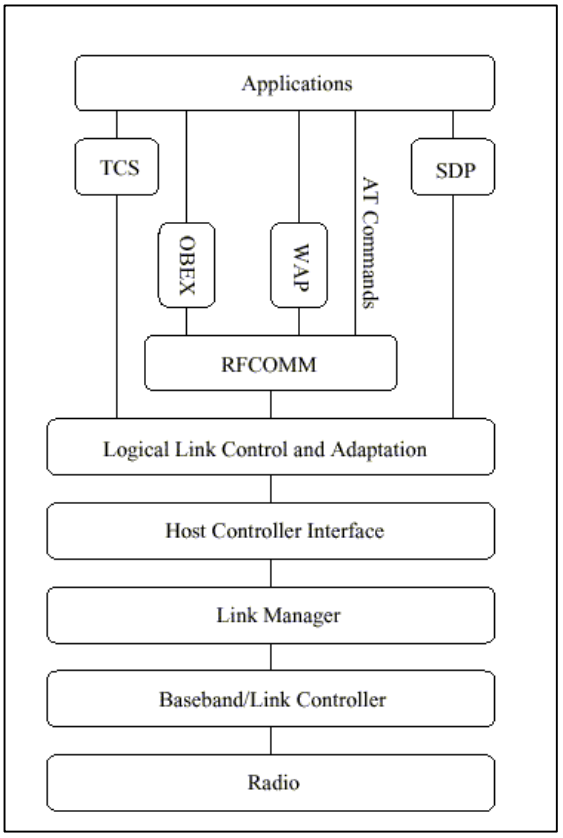
I primi studi sulla tecnologia Bluetooth partono nel 1994, quando la Ericsson Mobile Communication intraprende una serie di studi allo scopo di trovare delle alternative wireless per il collegamento tra telefoni e accessori (es. auricolari). Bluetooth è una tecnologia radio studiata appositamente per trasmissioni a corto raggio (tipicamente 10 m) ed è caratterizzata da un bassissimo consumo di potenza, da un basso costo e da una bassa complessità. Queste caratteristiche consentono a Bluetooth di proporsi come la tecnologia del futuro nel campo delle comunicazioni wireless per apparecchi di piccole dimensioni alimentati a batteria. Inoltre è studiata anche per connettere e fare interagire tra di loro dispositivi diversi (telefoni, stampanti, notebook, TV, PC, elettrodomestici, etc.). Lo standard Bluetooth consente ai dispositivi di connettersi e comunicare tra loro in una regione limitata attraverso una rete ad hoc chiamata piconet che è costituita da un massimo di otto dispositivi attivi, di cui uno è il master, ossia colui che inizia lo scambio di dati, e gli altri sette sono gli slave, che funzionano in risposta al master, e non hanno altri collegamenti oltre a quello col master. Inoltre è possibile avere fino a 200 dispositivi in modalità "park", cioè che non possono essere attivi nello scambio di dati col master, ma che restano sincronizzati con esso. Caratteristica importante dei dispositivi Bluetooth è la loro capacità di creare e riconfigurare dinamicamente queste piconet senza richiedere l'intervento umano. Infatti se un dispositivo entra o esce dalla zona coperta dal master viene riconosciuto o tolto dalla rete automaticamente. Questo grazie all'operazione di inquiry, attraverso la quale i dispositivi Bluetooth



periodicamente sono in grado di scoprire l'esistenza di altri nelle vicinanze. Tutto questo è possibile grazie al "Service Discovery Protocol" (SDP), che permette ad un dispositivo Bluetooth di determinare quali sono i servizi che gli altri apparecchi presenti nella piconet mettono a disposizione. Altra caratteristica importante dei dispositivi Bluetooth è quella di poter essere presenti in più di una piconet contemporaneamente, e di poter addirittura funzionare da master in una piconet e da slave nell'altra. Un gruppo di piconet legate tra loro viene chiamata scatternet



Lo standard Bluetooth opera nella banda centrata attorno ai 2.4 GHz. Questa banda è occupata da un elevato numero di emettitori RF, dalle applicazioni wireless proprietarie, allo standard Wireless Ethernet (IEEE 802.11b), fino ad arrivare a generatori di rumore come i forni a microonde e le lampade da strada. Ciò ha reso necessario l'utilizzo di una tecnica di modulazione robusta alle interferenze: si è scelto una 2-FSK con il frequency hopping. Questo sistema di comunicazione funziona secondo uno schema Time Division Multiplexing (TDM), dove l'unità base di operazione è uno slot di durata pari a 625 μs. Quando i dispositivi sono connessi, negli slot pari è abilitato a trasmettere il master, mentre negli slot dispari sono abilitati a rispondere gli slave, uno alla volta; in particolare può rispondere lo slave che aveva ricevuto dati dal master nello slot precedente. Il bitrate massimo raggiungibile in aria è pari a 1 Mbps. Lo stack Bluetooth è definito da una serie di livelli:



1. *Radio*: Il ricetrasmettitore Bluetooth opera nella banda ISM centrata attorno ai 2.4 GHz. Utilizza una modulazione di tipo 2-FSK, ossia una modulazione numerica che trasmette una certa frequenza per il livello logico alto, e un'altra per il livello logico basso. In particolare si utilizza la tecnica del frequency hopping (FHSS), che consiste nello sparpagliare il segnale su una banda maggiore facendo cambiare in continuazione la frequenza della portante. Le specifiche Bluetooth definiscono tre classi di potenza e le relative distanze che si possono raggiungere in trasmissione:

Power Class	Output Power (Max)	Distance (Max)
1	100 mW (20 dBm)	100 m
2	2.5 mW (4 dBm)	10 m
3	1 mW (0 dBm)	1 m

2. *Baseband:* Bisogna innanzitutto fare una distinzione tra "Link Controller" e "Baseband" in quanto le specifiche Bluetooth usano a volte questi due termini in modo ambiguo. Il Link Controller (LC) è responsabile del trasporto sul link dei vari pacchetti scambiati provenienti dal Link Manager, e deve mantenere questo collegamento attivo una volta stabilito, mentre il Baseband controlla il livello Radio, è responsabile della temporizzazione a basso livello, del controllo degli errori e della gestione dei collegamenti durante la trasmissione di un pacchetto. Come descritto in precedenza il sistema Bluetooth utilizza uno schema di trasmissione TDM con slot temporali di 625 μs, che è una combinazione tra una comunicazione a commutazione di pacchetto e una a commutazione di circuito. Alcuni time slot possono essere dedicati a pacchetti sincroni: infatti è possibile supportare fino a tre canali sincroni contemporaneamente, in coesistenza con canali dati asincroni. Il canale è rappresentato da una sequenza pseudo casuale che indica il salto di frequenza attraverso 79 canali. Questa sequenza di salto è unica per la piconet ed è determinata dall'indirizzo del dispositivo master; la fase è invece determinata dal clock del master. Il canale è diviso in slot temporali associati ad un determinato salto di frequenza. Ogni unità che partecipa alla piconet dovrà essere sincronizzata sia temporalmente che frequenzialmente al canale. Ogni dispositivo deve essere in un determinato istante o master o slave. Questi due ruoli sono così definiti:

- master: dispositivo che inizia lo scambio di dati;
- slave: dispositivo che risponde al master.

Inoltre, quando sono in comunicazione, gli slave utilizzano la temporizzazione del master e saltano in frequenza seguendo la sequenza del master. Come molti protocolli di comunicazione, Bluetooth sincronizza la maggior parte delle operazioni a un clock interno. Questo assicura, per esempio, la sincronizzazione Tx-Rx nello scambio di dati tra due dispositivi, differenziando i pacchetti persi e quelli rispediti. Il clock Bluetooth è un contatore a 28 bit che si resetta a zero all'accensione e poi conta in "free-run", incrementandosi ogni mezzo slot, cioè 312.5 μs. Ogni dispositivo ha un suo clock interno denominato CLKN. Quando un dispositivo opera come master, il suo clock interno viene usato come clock della piconet. Se un dispositivo opera come slave si deve invece sincronizzare al clock del master. Per fare questo, uno slave deve aggiungere un "offset value" al suo clock nativo, dal quale si ottiene una stima del clock del master, chiamata "piconet clock", CLK. Uno slave, per mantenere la sincronizzazione col master, si risincronizza ogni volta che riceve, grazie alla "synchronisation word", attraverso la quale si riallinea al dispositivo trasmittente. C'è un altro clock definito nel Bluetooth: CLKE, che si ottiene aggiungendo un altro offset al CLKN dello slave ed è usato nel caso specifico di in cui si stabilisce una connessione con uno slave prima di essere sincronizzati con il master.

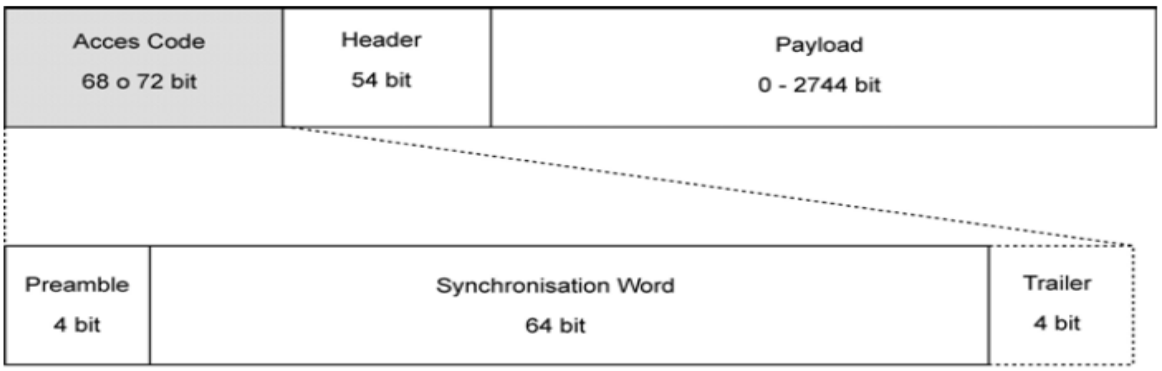
Il livello Baseband gestisce due tipi di collegamenti:

- ACL (Asynchronous Connection-Less) E' un collegamento punto-multipunto tra il master e gli slave della piconet. Negli slot non riservati ai collegamenti SCO, il master può scambiare pacchetti con ogni slave della piconet. Tra un master e uno slave può esistere solo un collegamento ACL.

- SCO (Synchronous Connection Oriented) E' un collegamento simmetrico punto-punto tra il master e uno slave della piconet. Per questo tipo di collegamenti vengono riservati degli slot e può pertanto essere considerato come una connessione a commutazione di circuito. Il collegamento SCO viene utilizzato per la trasmissione della voce; un master può supportare un massimo di tre collegamenti SCO contemporaneamente. I pacchetti SCO non vengono mai ritrasmessi.

Segue ora una descrizione dei differenti tipi di pacchetti che vengono usati nella comunicazione tra dispositivi nei link ACL e SCO. Un pacchetto standard Bluetooth è costituito da tre parti, ossia l'access code, il packet header e il payload.

Access Code: Ogni pacchetto inizia con un access code. I compiti di questo campo sono la sincronizzazione, la compensazione dell'offset in continua e l'identificazione, in quanto l'access code viene ricavato dal Bluetooth Device Address del master.

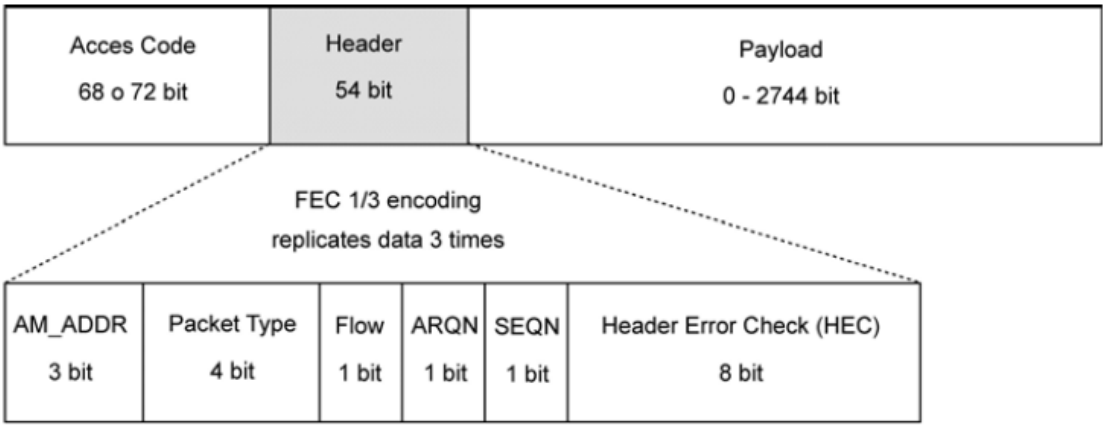


La prima parte dell'access code (preamble) è costituita da 4 bit (0101 o 1010 a seconda del primo bit della synchronisation word, per formare una parola di 5 bit, o 01010 o 10101), e serve per rilevare i fronti dei dati ricevuti e creare così un clock con il quale campionare il resto dei dati in ricezione. Sono definti tre tipi di access code:

- Channel access Code (CAC): identifica la piconet.
- Device Access Code (DAC): utilizzato per speciali procedure di segnalazione.
- Inquiry access Code (IAC): utilizzato nelle operazioni di inquiry.

Gli ultimi 4 bit dell'access code (trailer) sono presenti solo se è presente un payload, sono simili al preamble e servono per eseguire migliori compensazioni in continua e recupero del clock.

Header: Questo campo contiene delle informazioni di controllo del collegamento protetti dal codice Forward Error Correction (FEC)

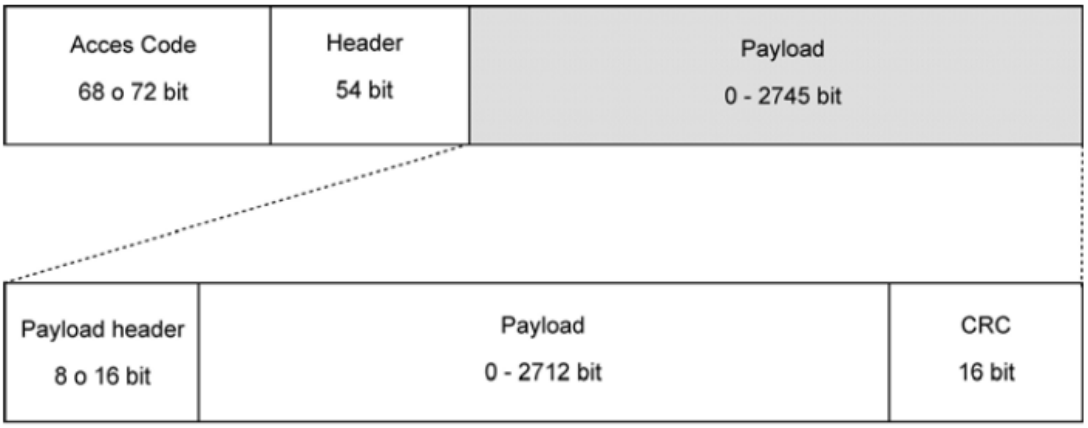


Il payload header consiste di sei campi:

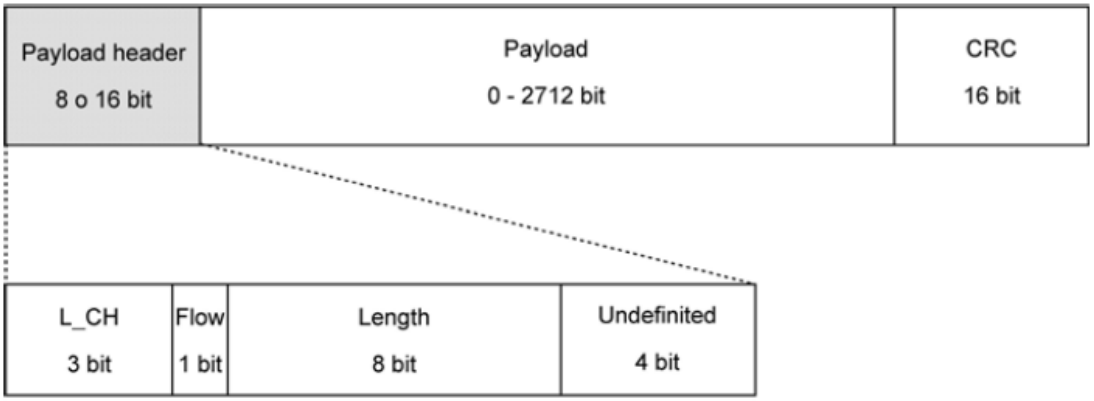
- AM_ADDR (3 bit): rappresenta l'indirizzo di un membro attivo della piconet; ad ogni slave viene assegnato dal master un indirizzo temporaneo di tre bit, e tutti i pacchetti scambiati tra master e slave devono avere l'AM_ADDR dello slave. Un AM_ADDR di tutti zeri viene usato per i messaggi di broadcast.
- TYPE (4 bit): tipo di pacchetto.
- FLOW (1 bit): viene usato per il controllo di flusso dei pacchetti sui collegamenti ACL; FLOW = 0 significa che il buffer di ricezione è pieno e la trasmissione deve essere temporaneamente fermata.
- ARQN (1 bit): indicatore di acknowledge, indica al trasmettitore se i dati sono stati ricevuti correttamente o meno.
- SEQN (1 bit): fornisce una numerazione sequenziale al flusso dei dati.
- HEC (8 bit): Header Error Check, è utilizzato per controllare l'integrità dell'header.

Payload: Il formato del payload è diverso nel caso di pacchetto SCO o ACL.

- ACL payload: E' costituito da un massimo di 2744 bit, ed è formato da tre campi: il payload header, il payload vero e proprio e il codice controllore d'errore Cyclic Redundancy Check (CRC).



Il payload header è formato a sua volta a tre campi, cioè il L_CH (Logical Channel), che indica se è l'inizio o la continuazione di un pacchetto L2CAP o LMP, il flow flag e il length field (lunghezza).



- SCO payload: Ha una lunghezza fissa di 240 bit ed è preceduto dagli stessi access code e header di un pacchetto ACL. E' assente il campo CRC.

Le specifiche Bluetooth definiscono cinque canali logici che sono trasportati sui link fisici SCO e ACL, e servono per trasportare informazioni a differenti livelli:

- Link Control (LC): è un canale di controllo che viene trasportato nel packet header, e contiene informazioni di basso livello per il controllo del collegamento.
- Link Manager (LM): è un canale di controllo e trasporta informazioni scambiate tra i livelli Link Manager di due dispositivi collegati.
- UA/UI: il canale UA (User Asynchronous) è utilizzato per trasportare i dati L2CAP mentre il canale UI (User Isochronous) serve nel caso di pacchetti che devono essere trasmessi via ACL.
- US: il canale US (User Synchronous) viene usato per il trasporto dei dati SCO.

Lo standard Bluetooth specifica di utilizzare delle code FIFO (First In First Out) sia in ricezione che in trasmissione, per i collegamenti SCO e ACL. In trasmissione il compito del Link Manager è quello di riempire le code, mentre il Link Controller si occupa dello svuotamento di esse. Sono definiti quattro tipi di indirizzi:

- BD_ADDR: è formato da 48 bit ed è unico per ogni dispositivo Bluetooth.
- AM_ADDR: è composto da tre bit, e identifica univocamente un dispositivo all'interno della piconet.
- PM_ADDR: è composto da 8 bit, ed è assegnato ai dispositivi in modalità park.
- AR_ADDR: è composto da 8 bit, ed è utilizzato dai dispositivi in modalità park per determinare lo slot in cui possono mandare il proprio messaggio d'accesso. Questo indirizzo non è univoco e può essere assegnato a più dispositivi in park.

3. Link Controller: Il Link Controller è costituito da parti hardware e software, è responsabile del mantenimento del link una volta che è stato creato, esegue protocolli di livello inferiore, come il protocollo ARQ e la codifica FEC. Le funzioni svolte dal Link Controller sono le seguenti:

- Trasferimento dati con le caratteristiche selezionate coi parametri del QoS (Qualità of Service).
- Trasferimento asincrono con successo garantito utilizzando un protocollo ARQ.
- Trasferimento sincrono.
- Codifica audio.
- Criptaggio.

Un apparecchio Bluetooth può essere in una serie di stati diversi:

- Standby: In questo stato, il dispositivo è inattivo.
- Inquiry: E' il processo tramite cui il dispositivo che intende diventare il master della piconet può provare a scoprire tutti gli altri apparecchi Bluetooth attivati nella sua area locale.
- Inquiry scan: E' l'atra metà della procedura di inquiry, e viene eseguita da tutti i dispositivi disponibili a diventare slave della piconet.
- Connection Active: Entrando nello stato connection, lo slave commuta al CLK del master e poi si sposta sulla sequenza di frequency hopping del master.
- Connection Sniff: Nel modo sniff, a uno slave è dato un predefinito slot temporale e una periodicità per ascoltare il traffico.

[…]

4. Link Manager: Un dispositivo Bluetooth viene guidato dall'host controller attraverso i comandi HCI, ma è il Link Manager che si occupa della traslazione di questi comandi in operazioni al livello Baseband, gestendo le seguenti operazioni:

- legando gli slave alla piconet e allocando il loro indirizzo attivo;
- rompendo la connessione per staccare gli slave dalla piconet;
- configurando il link;
- controllando le commutazioni master/slave;
- stabilendo i collegamenti SCO (voce) e ACL (dati);
- mettendo le connessioni in modalità basso consumo;
- controllando la modalità test.

Un LM Bluetooth comunica con un LM su un altro dispositivo Bluetooth usano l'LMP (Link Manager Protocol). In pratica, il LM controlla la gestione della piconet (crea e distrugge collegamenti), configura i collegamenti e le funzioni di sicurezza.

5. Host Controller Interface: Questo livello mette a disposizione dei livelli superiori un'interfaccia uniforme per accedere alle capacità hardware. In sistemi dove i livelli più alti vengono eseguiti sul processore di un dispositivo che opera da host, mentre i livelli più bassi sono su un device Bluetooth, è richiesta un'interfaccia tra i livelli superiori e quelli inferiori. Lo standard Bluetooth definisce a questo scopo il livello HCI. Il firmware HCI presente nel modulo Bluetooth traduce tutti i comandi provenienti dall'interfaccia HCI in comandi comprensibili dal Link Manager e dal livello Baseband. Per mandare pacchetti HCI dall'host al modulo Bluetooth è necessario avere un livello di trasporto; le specifiche Bluetooth definiscono tre tipologie di interfacce:

- USB, Universal Serial Bus;
- UART, Universal Asynchronous Receiver Transmitter;
- RS232, interfaccia seriale con correzione dell'errore.

6. Logical Link Control and Adaptation Protocol: Questo livello preleva dati dai livelli superiori dello stack Bluetooth e dalle applicazioni e li inoltra verso i livelli inferiori. L2CAP mette a disposizione dei livelli superiori servizi dati connection-less e orientati alla connessione, oltre a eseguire operazioni di multiplexaggio, segmentazione e riassemblaggio dei pacchetti, e permette ai protocolli di livello superiore di trasmettere e ricevere pacchetti fino a 64 kbyte di lunghezza. Le principali funzioni del livello L2CAP sono:

- Multiplexaggio dei diversi livelli superiori, permettendo loro di condividere i livelli inferiori;
- Segmentazione e riassemblaggio dei pacchetti;

- Gestione dei gruppi, mettendo a disposizione dei livelli superiori l’astrazione di gruppo sulla piconet;
- Gestione della Qualità of Service per i livelli superiori.

7. RFCOMM: Il protocollo RFCOMM è posizionato sopra il livello L2CAP e mette a disposizione un’emulazione della porta seriale che permette di far girare tutte le applicazioni che si basano su porta seriale senza apportare loro alcuna modifica.

8. SDP: Il Service Discovery Protocol mette a disposizione dei dispositivi Bluetooth un mezzo per scoprire le caratteristiche e i servizi offerti da altri dispositivi bluetooth presenti nelle vicinanze. Una volta che una connessione L2CAP è stata stabilita tra un client e un server SDP, il client può navigare all’interno delle informazioni presenti nel server e decidere se la comunicazione tra i due dispositivi può continuare perché è stata trovata l’applicazione che serve oppure se disconnettersi.

2.2.2 Html

L'HyperText Markup Language (HTML) è il linguaggio di markup solitamente usato per la formattazione e impaginazione di documenti ipertestuali.

La sintassi è stabilita dal World Wide Web Consortium (W3C), e che è derivato da un altro linguaggio avente scopi più generici, l'SGML. L'HTML è stato sviluppato verso la fine degli anni ottanta da Tim Berners-Lee al CERN di Ginevra assieme al protocollo HTTP con lo scopo di trasferire documenti in tale formato.

Attualmente i documenti HTML sono in grado di incorporare molte tecnologie, che offrono la possibilità di aggiungere al documento ipertestuale interazioni dinamiche con l'utente, animazioni interattive e contenuti multimediali, grazie a linguaggi come CSS, JavaScript, JSON, ecc.. L'HTML è un linguaggio di formattazione che descrive le modalità di impaginazione o visualizzazione grafica del contenuto testuale di una pagina

web attraverso tag di formattazione. Sebbene l'HTML supporti l'inserimento di script e oggetti esterni, non è un linguaggio di programmazione in quanto non prevede alcuna definizione di variabili, strutture dati, funzioni ecc. che possano realizzare programmi. Il suo codice è in grado soltanto di strutturare e decorare dati testuali.

Il linguaggio HTML, ha come scopo quello di gestire i contenuti associandone o specificandone allo stesso tempo la struttura grafica all'interno della pagina web da realizzare grazie all'utilizzo

di tag diversi. Ogni tag (ad esempio *<h1>* o *<p>)* specifica un diverso ruolo dei contenuti che esso contrassegna (ad es. il tag *<h1>* definirà un'importanza maggiore del tag *<p>*). La formattazione consiste nell'inserimento nel testo di tag, che descrivono caratteristiche come la funzione, il colore, le dimensioni, la posizione relativa all'interno della pagina, ecc.. I browser che leggono il codice mostrano all'utente formattazioni predefinite per ogni tag che incontrano. Il componente principale della sintassi di questo linguaggio è l'elemento, inteso come struttura di base a cui è delegata la funzione di formattare i dati o indicare al browser delle informazioni. Gli elementi HTML consistono generalmente di quattro parti:

1. Un tag di apertura che definisce l'inizio di un elemento ** ;
2. Uno o più attributi di tale elemento con i loro rispettivi valori;
3. Il contenuto informativo da visualizzare;
4. Un tag di chiusura **;

Alcuni tag presentano un'applicazione puntuale, come per esempio il tag *<img>* che serve per inserire un'immagine in un determinato punto della pagina, e in quanto tali non richiedono il tag di chiusura e si parla di tag a chiusura implicita.
Un documento HTML comincia con una dichiarazione del tipo di documento, una breve stringa che indica in quale sintassi e relativa versione esso sia scritto.
Tale informazione è necessaria al browser per identificare le regole di interpretazione e visualizzazione appropriate per lo specifico documento, ed è per questa ragione che la dichiarazione debba precedere il documento vero e proprio. Dopo la dichiarazione del tipo di documento, lo stesso presenta una struttura ad albero annidato, composta da sezioni delimitate da tag opportuni che al loro interno contengono a loro volta sottosezioni più piccole, sempre delimitate da tag. All'interno dei tag *<html>* lo standard prevede sempre la definizione di due sezioni ben distinte e disposte in sequenza ordinata:

1. La sezione di intestazione o header: delimitata tra i tag *<head>* e *</head>*, contiene informazioni di controllo normalmente non visualizzate dal browser.
2. La sezione del corpo o body: delimitata tra i tag *<body>* e *</body>*, contiene la parte informativa vera e propria, ossia il testo, le immagini e i collegamenti che costituiscono la parte visualizzata dal browser.

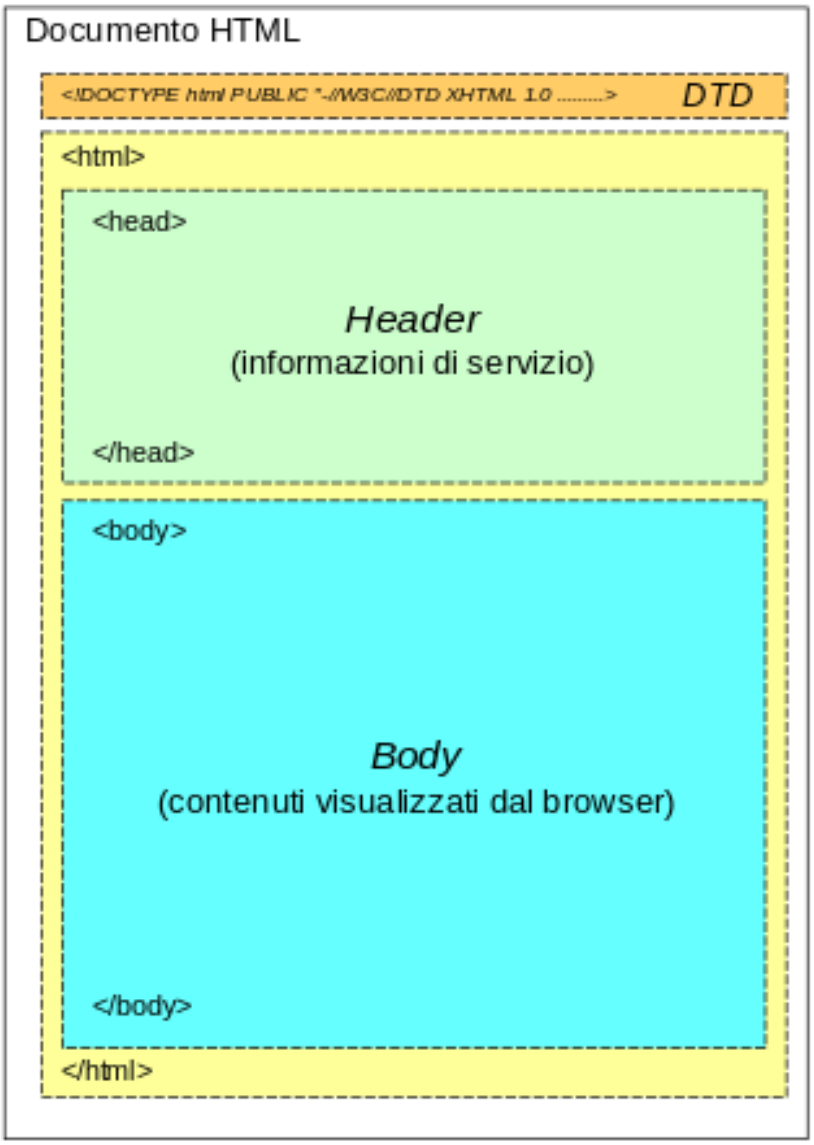
Al di sotto di questa suddivisione generale, lo standard non prevede particolari obblighi per quanto riguarda l'ordine e il posizionamento delle ulteriori sottosezioni all'interno dell'header o

del body. I tag utilizzati nella sezione head sono tipicamente di tipo diverso da quelli utilizzati nella sezione body, essendo destinati a scopi differenti.
Normalmente non vengono visualizzati dal browser ma servono come informazioni di controllo e di servizio quali:

1. Metadata per convogliare informazioni utili ad applicazioni esterne o al browser.
2. Collegamenti verso file di servizio esterni (CSS, script)
3. Inserimento di script (codice eseguibile) utilizzati dal documento
4. Informazioni di stile (CSS locali)
5. Il titolo associato alla pagina e visualizzato nella finestra principale del browser.

All'interno della sezione di body, che racchiude la parte visualizzabile del documento, si utilizzano i tag specifici previsti per la formattazione dei contenuti accessibili all'utente finale, ossia per il controllo di:

1. Intestazioni (titoli di capitoli, di paragrafi eccetera).
2. Strutture di testo (testo indentato, paragrafi, eccetera).
3. Aspetto del testo (grassetto, corsivo, eccetera).
4. Elenchi e liste (numerate, generiche, di definizione).
5. Tabelle.
6. Moduli elettronici (campi compilabili dall'utente, campi selezionabili, menu a tendina, pulsanti eccetera).
7. Collegamenti ipertestuali.
8. Layout generico del documento.



2.2.3 CSS

Il CSS (Cascading Style Sheets), è un linguaggio usato per definire la formattazione di documenti HTML, ad esempio i siti web. Le regole per comporre il CSS sono contenute in un insieme di direttive redatte dal W3C. L'introduzione del CSS si è resa necessaria per separare i contenuti delle pagine HTML dalla loro formattazione e permettere una programmazione più chiara e facile da utilizzare, sia per gli autori delle pagine stesse sia per gli utenti, garantendo contemporaneamente anche il riutilizzo di codice ed una sua più facile manutenzione.

Per permettere agli autori di poter plasmare liberamente l'aspetto delle pagine web, dal 1993 in poi Netscape Navigator ed Internet Explorer, presentarono tag proprietari, ovvero non aderenti agli standard né compatibili con i browser concorrenti. Un esempio di questi tag è <font> che va a definire il font dei caratteri. Tuttavia questi tag presentavano tre problemi:

1. Il primo problema è costituito dalla lunghezza dei tag. Se confrontata con una pagina che adotta il linguaggio CSS, una pagina che non lo adotta è in genere più pesante. Inoltre, le istruzioni CSS possono essere raccolte in un file esterno che rimane memorizzato nella cache del browser, riducendo ulteriormente la quantità di dati che i server devono trasmettere.

2. Il secondo problema risiede nella mancanza di logica del codice HTML. Un codice non aderente agli standard, ridondante e confuso comporta infatti molto lavoro aggiuntivo per i browser.

3. Il terzo problema, che comincia a diventare sempre più rilevante, è la mancanza di compatibilità con i nuovi computer palmari e gli smartphone. Queste pagine infatti sono progettate per schermi con risoluzione minima 800x600 pixel. Dispositivi come i palmari (o i più moderni tablet) che possono avere una risoluzione inferiore ed una forma dello schermo ben diversa dal rapporto 4:3 dei monitor per computer, si trovano quindi impossibilitati a visualizzare correttamente la pagina con il risultato che l'utente la visualizzerà confusa e scomoda da leggere.

Per tentare di risolvere questa situazione, nel 1996 il W3C emanò le specifiche CSS 1.

I CSS 1 erano un interessante sistema per separare contenuto da formattazione. La base di questo linguaggio, infatti, consisteva nel fatto che il contenuto sarebbe stato sempre definito dal codice HTML, mentre la formattazione si sarebbe trasferita su un codice completamente separato, il CSS appunto. I richiami tra i due codici venivano effettuati tramite due particolari attributi: class e ID. Queste specifiche:

1. Erano un'efficace soluzione al primo problema perché riducevano notevolmente le dimensioni delle pagine.

2. Risolvevano il secondo in modo parziale perché consentivano al codice HTML di ritornare in gran parte semplice ed essenziale, ma presentavano qualche mancanza che costringeva a ricorrere ai tag HTML.

3. Non prendevano però in considerazione il terzo problema, dato che nel 1996 i PDA (i palmari) erano scarsamente diffusi.

I CSS 1 sviluppavano un'idea semplice ma efficace, ma nonostante le loro grandi potenzialità non ebbero successo a causa della mancanza di browser in grado di supportarli. Per includere le nuove funzionalità e rendere i CSS un linguaggio ben supportato, nel 1998 il W3C pubblicò le specifiche CSS 2, e nel 2004 iniziarono i lavori sulle specifiche aggiornate CSS 2.1. I CSS 2 sono la naturale evoluzione dei CSS 1 ed offrono potenti soluzioni per risolvere soprattutto il terzo problema, con la possibilità di creare fogli di stile separati per i dispositivi portatili. Anche il secondo problema è ormai pienamente risolvibile, scrivendo una pagina HTML esclusivamente indirizzata alla struttura e ai contenuti e manovrandola poi esclusivamente con i CSS per impaginarla. A partire da Internet Explorer 5, Firefox e Opera 7, i CSS 2 hanno potuto avvalersi di browser in grado di interpretarli e sono quindi entrati a far parte del codice di molti siti web. A differenza delle specifiche CSS 2, che è costituita da un'unica specifica, quelle CSS3 sono costituite da sezioni separate dette "moduli".

A causa di questa modularizzazione, le specifiche CSS3 hanno differenti stati di avanzamento. Essi dovrebbero presentare soluzioni per la correzione di alcuni bug di interpretazione di Internet Explorer, migliorie nella gestione degli sfondi e una soluzione per realizzare i bordi arrotondati.

L'inserimento di codice CSS nelle pagine web può essere effettuato in tre modi diversi:

1. Inserendo nel tag <head> della pagina in codice HTML un collegamento ad un foglio di stile esterno, cioè un file contrassegnato dall'estensione .css tramite il tag link o tramite la direttiva import.

2. Inserendo, sempre all'interno dell' <head> tra gli specifici tag <style> e </style> le dichiarazioni css.

3. In linea all'interno degli elementi.

Molto importanti sono i selettori, che permettono di specificare elementi HTML. Questi sono:

1. Selettori di tipo: Applicano la regola a tutti gli elementi della pagina del tipo determinato. Esempio:

```
body {
    [...]
}
```

2. Selettori di classe: Applicano la regola a tutti gli elementi della pagina che presentano la proprietà *class="nome_classe"*. La sintassi CSS è la seguente:

. *nome_classe {*

[...]

}

3. Identificatori: Comunemente ID, applicano la regola a quell'elemento della pagina che presenta la proprietà *id="nome_identificatore"*. Gli ID contraddistinguono elementi unici. La sintassi CSS è la seguente:

#nome_identificatore {

[...]

}

4. Pseudoclassi: Identificano elementi in base alle loro proprietà.

   i. first-child individua un elemento solo se è il primo figlio dell'elemento padre.

   p:first-child {

   [...]

   }

   ii. link e visited si applicano ai collegamenti. La prima identifica i collegamenti non visitati, la seconda quelli visitati. La sintassi CSS è:

   a:link {

   [...]

   }

5. active, focus e hover identificano gli elementi solo in particolari condizioni, la prima se l'elemento è attivo, la seconda se è selezionato, la terza se il puntatore è sopra di lui.

   p:hover {

   [...]

   }

6. Selettore di discendenza, figlio e fratello: Identificano solamente gli elementi che si trovino in una particolare condizione di discendenza nella struttura HTML della pagina.

   i. Il selettore di discendenza identifica solo gli elementi contenuti in altri elementi.

   p span {

```
    [...]
}
```

Identifica solo gli elementi contenuti in elementi <*p*>.

ii. Il selettore figlio identifica invece solo gli elementi che siano contenuti direttamente nell'elemento padre.

```
div > p {
    [...]
}
```

Individua solo i <*p*> contenuti direttamente in un <*div*>.

iii. Il selettore fratello identifica il primo elemento immediatamente successivo ad un altro con cui condivida lo stesso padre.

```
h1 + p {
    [...]
}
```

Individua solo il primo <*p*> fratello di <*h1*>

Le proprietà CSS sono numerose, circa 60. Alcune di queste sono:

a. Background: Definisce lo sfondo di un elemento.

b. Border: Definisce il bordo di un elemento.

c. Color: Definisce il colore del testo di un elemento.

d. Float: Questa proprietà definisce un blocco flottante, ovvero che permette la disposizione di altri elementi ai suoi lati.

e. Font: Definisce le proprietà del carattere.

f. Margin e Padding: Definiscono lo spazio circostante gli elementi. La prima lo spazio esterno ai bordi, la seconda quello interno.

g. Text-align: Definisce l'allineamento degli elementi, tra cui il testo.

h. […].

2.2.4 Javascript

JavaScript è un linguaggio di scripting orientato agli eventi, comunemente utilizzato nella programmazione Web lato client per: la creazione di siti e applicazioni web, e di effetti dinamici interattivi tramite funzioni di script invocate da eventi innescati a loro volta in vari modi dall'utente sulla pagina web in uso (mouse, tastiera, caricamento della pagina ecc...). Fu originariamente sviluppato da Brendan Eich con il nome di Mocha e successivamente di LiveScript, ma in seguito è stato rinominato "JavaScript" ed è stato formalizzato con una sintassi più vicina a quella del linguaggio Java. E' stato standardizzato per la prima volta tra il 1997 e il 1999 dalla ECMA (European Computer Manufacturers Association associazione che si occupa di standardizzazioni nel settore informatico e delle comunicazioni) con il nome ECMAScript. E' inoltre uno standard ISO. Inizialmente usato solo dal browser netscape navigator, dato il suo successo Microsoft ne sviluppo' una sua versione nota come Jscript. Nonostante abbia in comune la parola Java, javascript e' profondamente diverso da Java e vedono un punto di incontro nella sola semantica in quanto entrambi derivano dal linguaggio C. Le caratteristiche principali di JavaScript sono:

1. L'essere un linguaggio interpretato: il codice non viene compilato, ma interpretato (in JavaScript lato client, l'interprete è incluso nel browser che si sta utilizzando).
2. Definisce le funzionalità tipiche dei linguaggi di programmazione ad alto livello (strutture di controllo, cicli, ecc.) e consente l'utilizzo del paradigma object oriented anche se debolmente. In particolare gli oggetti stessi sono degli array associativi.
3. E’ un linguaggio debolmente tipizzato.

In JavaScript lato client, il codice viene eseguito direttamente sul client e non sul server. Il vantaggio di questo approccio è che, anche con la presenza di script particolarmente complessi, il web server non viene sovraccaricato a causa delle richieste dei client. Di contro, nel caso di script che presentino un codice sorgente particolarmente grande, il tempo per lo scaricamento può diventare abbastanza lungo. Un altro svantaggio è che ogni informazione che presuppone un accesso a dati memorizzati in un database remoto deve essere rimandata ad un linguaggio che effettui la transazione, per poi restituire i risultati ad una o più variabili JavaScript. Questa problematica è stata poi superata Con l'avvento di AJAX. A differenza di altri linguaggi, quali

il C o il C++, che permettono la scrittura di programmi completamente stand-alone, JavaScript viene utilizzato soprattutto in quanto linguaggio di scripting, integrato, quindi, all'interno di un altro programma. L'idea di base è che il programma ospite (quello che ospita ed esegue lo script) fornisca allo script un'API ben definita, che consente l'accesso ad operazioni specifiche, la cui implementazione è a carico del programma ospite stesso. Lo script, quando eseguito, utilizza riferimenti a questa API per richiedere (al programma ospite) l'esecuzione di operazioni specifiche, non previste dai costrutti del linguaggio JavaScript in sé. L'esempio tipico di programma ospite per uno script JavaScript è quello del browser che incorpora un interprete. Quando viene visitata una pagina web che contiene il codice di uno script JavaScript, quest'ultimo viene portato in memoria primaria ed eseguito dall'interprete contenuto nel browser. Le interfacce che consentono a JavaScript di rapportarsi con un browser sono chiamate DOM (Document Object Model). Un uso principale del Javascript in ambito Web è la scrittura di piccole funzioni integrate nelle pagine HTML che interagiscono con il DOM del browser per compiere determinate azioni non possibili con il solo HTML statico: controllare i valori nei campi di input, nascondere o visualizzare determinati elementi, ecc..

Sfortunatamente, gli standard DOM imposti dal W3C non sempre vengono rispettati dai vari browser: browser diversi espongono diversi oggetti o metodi allo script (Internet Explorer è solito aderire agli standard con piccole modifiche), ed è quindi spesso necessario implementare controlli aggiuntivi ad una funzione JavaScript, per garantirne la compatibilità con ciascun browser. Al di fuori del Web, interpreti JavaScript sono integrati in diverse applicazioni come Adobe Acrobat e Adobe Reader supportano JavaScript nei file PDF. Le varie implementazioni di JavaScript, spesso non sono conformi agli standard, ma piuttosto sono costruite per funzionare con uno specifico browser web. L'attuale standard ECMAScript dovrebbe essere teoricamente la base di tutte le implementazioni Javascript, ma in pratica i browser Mozilla usano JavaScript, Microsoft Internet Explorer usa JScript, e altri browser come Opera e Safari usano altre implementazioni di ECMAScript, spesso con ulteriori caratteristiche non standard. Per cui ciascun browser tratta lo stesso script in modo diverso e ciò che funziona in un browser potrebbe non funzionare in un altro. Ci sono due tecniche principali per gestire le incompatibilità: browser sniffing e object detection. Inizialmente il browser sniffing era la tecnica più diffusa. Controllando un certo numero di proprietà del client, che restituivano informazioni su piattaforma, browser e versione, era possibile per il codice discernere esattamente in quale browser veniva eseguito. Più tardi, le tecniche di sniffing divennero più difficili da implementare. L'object detection si basa sul controllare l'esistenza della proprietà di

un oggetto. Se esso esiste non ci saranno problemi di esecuzione, altrimenti si procede diversamente. Nel contesto dell'applicazione, oltre all'ausilio dello stesso linguaggio, si sono utilizzate le librerie Highcharts.js, ovviamente scritte in linguaggio javascript, per la gestione dei grafici, le jquery mobile, un framework ottimizzato per i dispositivi mobili il quale fornisce anche i CSS per la formattazione.

2.2.5 Librerie Cordova

Le applicazioni mobile giocano un ruolo trainante nel panorama software attuale. Basti pensare al numero di app presenti sui diversi marketplace. Nonché alla varietà di piattaforme disponibili in continua evoluzione: iOS, Android, Windows Phone, Blackberry OS, ecc. Al momento chi vuole creare e pubblicare un'app ha di fronte almeno due possibilità:

- Imparare a sviluppare su una specifica piattaforma e rilasciare la propria app soltanto per essa
- Imparare a lavorare su più piattaforme.

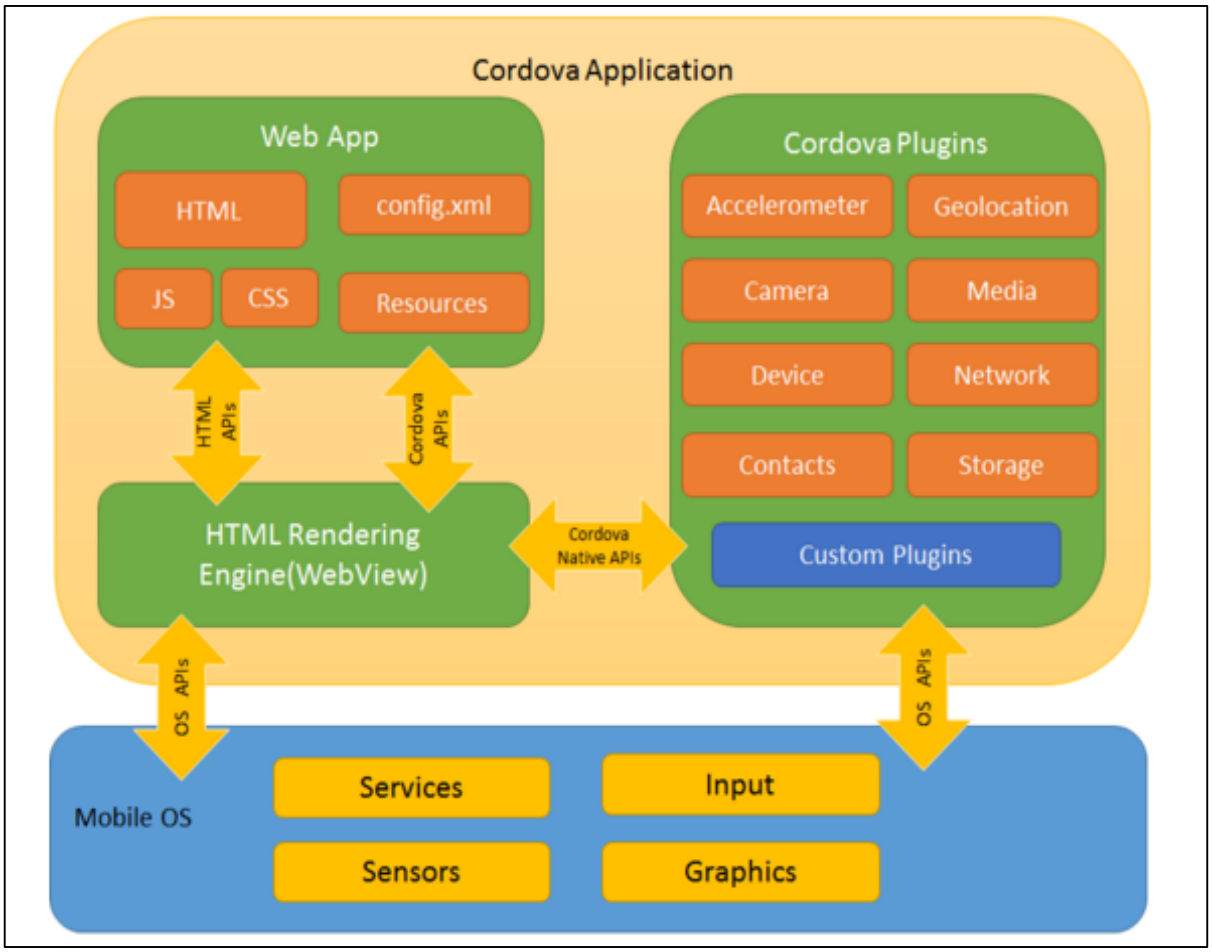
Per un singolo sviluppatore imparare a lavorare per bene su più piattaforme è impresa ardua, considerata anche la rapida evoluzione dei sistemi che costringe ad continuo aggiornarsi. Di fronte a questo dilemma può essere utile valutare una terza possibilità che consente di avere una visione unificata delle diverse piattaforme attraverso l'ausilio delle tecnologie Web. Quest'ultime, hanno avuto un forte impulso negli ultimi anni grazie anche al settore mobile. La definizione delle specifiche di HTML5 e di CSS3 includono molte opzioni per il mondo mobile ed il loro utilizzo ha delineato nuovi approcci per lo sviluppo di app. Allo stato attuale possiamo classificare le applicazioni per il mondo mobile in base alle tecnologie utilizzate in tre categorie:

1. App native: sono le app scritte e compilate per una specifica piattaforma utilizzando uno dei linguaggi di programmazione supportati dal particolare sistema.
2. Web app: sono pagine Web ottimizzate per dispositivi mobili sfruttando le tecnologie Web, in particolare HTML5, JavaScript e CSS3.
3. App ibride: sono le app che cercano di sfruttare il meglio delle due categorie precedenti: sono scritte con tecnologie Web ma vengono eseguite localmente all'interno di un'applicazione nativa.

Ogni tipologia ha aspetti positivi e negativi e la scelta deve essere fatto valutando attentamente le diverse condizioni ed il contesto in cui andiamo ad operare. Ad esempio, un'applicazione nativa ha dalla sua parte una maggiore velocità di esecuzione e una maggiore integrazione con il look della piattaforma, ma non è direttamente portabile su altre piattaforme. Una Web app invece risulta indipendente dalla piattaforma ma richiede una connessione Internet attiva e non è in grado di accedere al file system o ad altre risorse hardware del dispositivo. La tipologia di app ibrida, coniuga i vantaggi delle web app con quelle delle app native, consentendo di utilizzare le tecnologie Web per sviluppare l'applicazione senza necessità di una connessione Internet costante e avendo accesso alle risorse locali del dispositivo. Anche in questo caso ci sono degli svantaggi da considerare tra cui:

1. Una minore efficienza nel rendering grafico;
2. La potenziale lentezza di esecuzione nell'accesso alle risorse locali;
3. senza particolari accorgimenti l'aspetto dell'interfaccia grafica potrebbe non risultare abbastanza omogeneo con quello nativo della piattaforma.

Ai fini del progetto, si è usato l'approccio ibrido per lo sviluppo dell'app, attraverso l'ausilio del framework apache cordova. Esso consente di utilizzare tecnologie web standard - HTML5 , CSS3 e JavaScript per lo sviluppo di applicazioni cross-platform mettendo a disposizione delle API per accedere alle funzionalità di ciascun dispositivo, come sensori, dati, lo stato della rete, ecc. Nativamente cordova non fornisce nessun widget UI oppure MVC framework, ma provvede solo al runtime. Se si desidera usare queste funzionalità esse devono essere incluse nell'applicazione stessa. Lo schema a pagina successiva mostra l'architettura di una applicazione cordova.

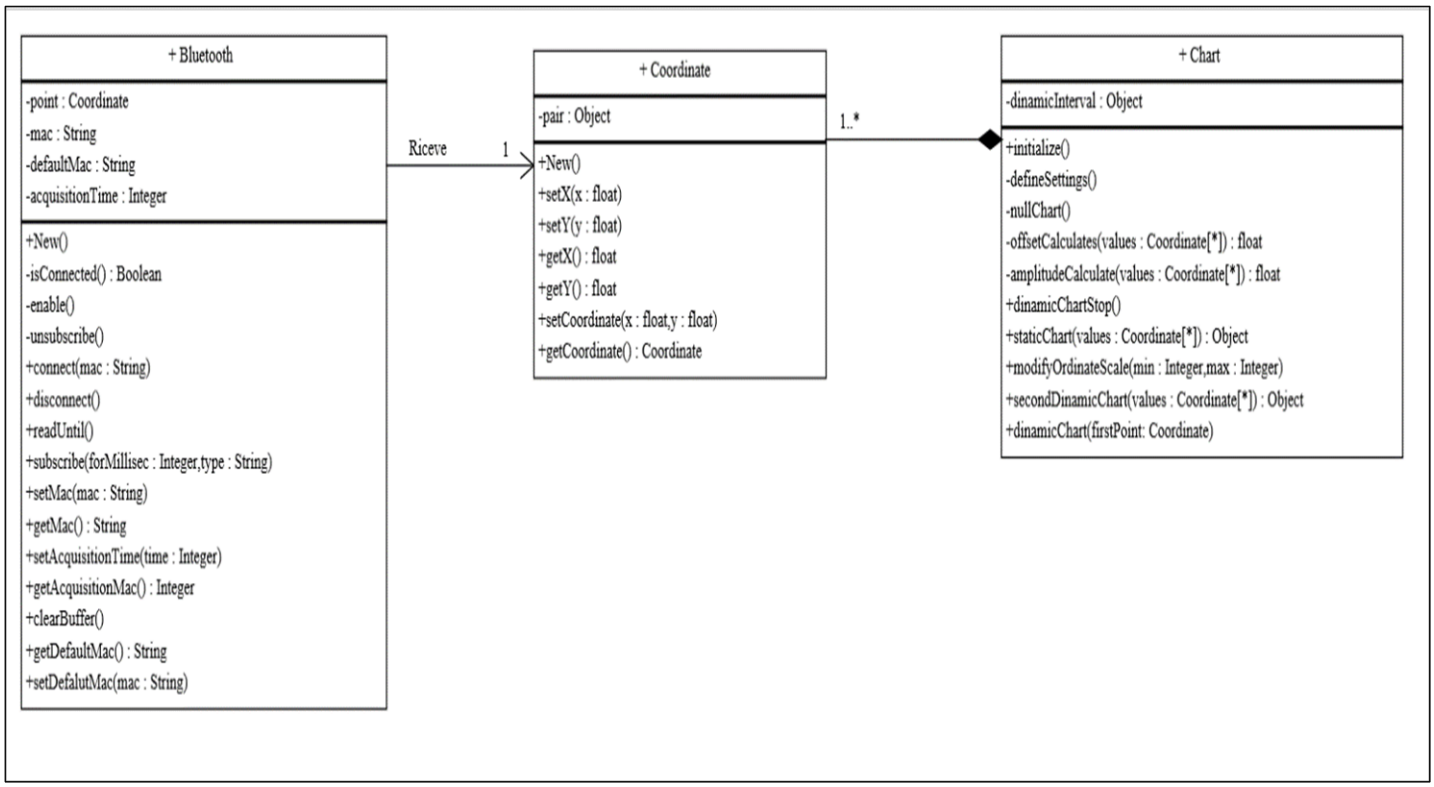

Cordova Application
Web App
HTML
config.xml
JS
CSS
Resources
Cordova Plugins
Accelerometer
Geolocation
Camera
Media
Device
Network
Contacts
Storage
Custom Plugins
HTML APIs
Cordova APIs
HTML Rendering Engine(WebView)
Cordova Native APIs
OS APIs
OS APIs
Mobile OS
Services
Input
Sensors
Graphics

3. Realizzazione

Ultimata una panoramica sulle tecnologie, sia hardware che software, utilizzate, si passa a descrivere l'effettiva realizzazione dello strumento descritto, riportando codice sorgente e schema elettrico dello stesso.

3.1 Realizzazione Software

Strutturalmente l'applicazione, programmata interamente col paradigma ad oggetti, prevede tre classi: classe Bluetooth, classe Coordinate e classe Chart. La classe Bluetooth si occupa della gestione dell'omonimo protocollo sul device, ed offre funzionalità come l'attivazione dello stesso, la connessione ad un altro dispositivo bluetooth, la ricezione dei dati, ecc. La classe Coordinate si occupa della creazione di oggetti "coordinata" che sono a loro volta usati dalla classe Chart che vede in ricezione un vettore di coordinate e si occupa della creazione di grafici e calcolo delle grandezze relative ad essi. La relazione tra classi prevede associazioni di tipo: associazione tra classe Bluetooth e classe Coordinate e composizione tra classe Coordinate e classe Chart. Di seguito sono riportati i codici delle diverse classi:



Bluetooth:

```
var Bluetooth = (function () {
    var point = Coordinate;
    var mac;
    var defaultMac = "98:D3:31:20:47:6A";
    var acquisitionTime = 500;
    var isConnected = function () {
        bluetoothSerial.isConnected(function () {
             return true;
        }, function () {
             return false;
        });
    };
    var enable = function () {
        bluetoothSerial.enable(function () {

        }, function () {

        });
    };
    var unsubscribe = function () {
        bluetoothSerial.unsubscribe(function () {

        }, function () {

        });
    }
    return {
        connect: function (mac) {
            var macArduino;
            if (mac != null) {
                this.setMac(mac);
                macArduino = this.getMac();
            } else {
                macArduino = this.getDefaultMac();
            }
            bluetoothSerial.connect(macArduino, function () {
                isConnected();
            }, function () {
                isConnected();
            });
        },
        disconnect: function () {
            bluetoothSerial.disconnect(function () {
                isConnected();
            }, function () {
                isConnected();
            });
        },
        isEnabled: function () {
            bluetoothSerial.isEnabled(function () {

            }, function () {
                enable();
            });
        },
        readUntil: function () {
            bluetoothSerial.readUntil("\n", null, null);
            bluetoothSerial.readUntil("\n", function (data) {
```

```
    if (data != "" && isNaN(parseFloat(data)) == false) {
       point.setCoordinate((newDate()).getTime(),parseFloat(data));
       setPoint(point.getCoordinate());
       } else {

        }
    }, function () {
    });
    this.clearBuffer();
},
subscribe: function (forMilliSec,type) {
    var i = 0;
    var j = 0;
    if (forMilliSec != null) {
        this.setAcquisitionTime(forMilliSec)
    } else {
    }
    bluetoothSerial.subscribe("\n", function (data) {
        if (j < 2) {
            j++;
        } else {
            if (data != "" && isNaN(parseFloat(data)) == false) {
                point.setCoordinate(i, parseFloat(data));
                setArrayOfCoordinate(point.getCoordinate())
                i++;
            } else {

            }

        }
    }, function () {

    });
    setTimeout(function () {
        i = 0;
        j = 0;
        unsubscribe();
        sendDataForChart(type, getArrayOfCoordinate())
    }, this.getAcquisitionTime());
},
setMac: function (setMac) {
    mac = setMac;
},
getMac: function () {
    return mac;
},
getDefaultMac: function () {
    return defaultMac;
},
 setDefaultMac: function (setMac) {
    defaultMac = setMac;
},

setAcquisitionTime: function (time) {
    acquisitionTime = time;
},
getAcquisitionTime: function () {
    return acquisitionTime;
},
clearBuffer: function () {
    bluetoothSerial.clear(function () {
```

```
            }, function () {

            });
        }
    };
}());
```

Coordinate:

```
    var Coordinate = (function(){
        var pair = {
            x: null,
            y: null
        };
        return {
            setX: function(coordinateX){
                pair.x = coordinateX;
            },
            setY: function(coordinateY){
                pair.y = coordinateY;
            },
            getX: function(){
                return pair.x;
            },
            getY: function(){
                return pair.y;
            },
            setCoordinate: function(coordinateX,coordinateY){
                pair.x = coordinateX;
                pair.y = coordinateY;
            },
            getCoordinate: function(){
                return pair;
            }
        }
    }());
```

Chart:

```
var Chart = (function () {
    var dinamicaInterval;
    var defineSettings = function () {
        Highcharts.setOptions({
            chart: {
                renderTo: "chartdiv",
                type: "spline"
            },
            global: {
                useUTC: false
            },
            exporting: {
                enabled: false
            },
            tooltip: {
                enabled: false
            },
            plotOptions: {
```

```
            spline: {
                marker: {
                    enabled: false
                },
                lineWidth: 2
            }
        },
        yAxis: {
            title: {
                text: "Voltage"
            },
            max: null,
            min: null
        }
    });
};

var nullChart = function () {
    var zeroArray = [];
    for (i = 0; i < 10; i++) {
        zeroArray.push({
            x: i,
            y: 0
        })
    }
    ;
    var option = {
        title: {
            text: ""
        },
        series: [{
                showInLegend: false,
                data: zeroArray
            }]
    };
    new Highcharts.Chart(option);
};
var offsetCalculates = function (values) {
    var numberOfElements = values.length;
    var sum = 0;
    var offset = 0;
    values.forEach(function (item) {
        sum = sum + Math.abs(item.y);
    });
    offset = (Math.round((sum / numberOfElements) * 100)) / 100;
    return offset;
};
var amplitudeCalculate = function (values) {
    var min = 0;
    var max = 0;
    var amplitude = 0;
    var voltageValue = [];
    values.forEach(function (item) {
        voltageValue.push(item.y)
    });
    min = Math.min.apply(null, voltageValue);
    max = Math.max.apply(null, voltageValue);
    amplitude = (Math.round((max - min) * 100)) / 100;
    return amplitude;
};
return {
```

```
initialize: function () {
    defineSettings();
    nullChart();
},
modifyOrdinateScale: function (min, max) {
    if (min == 0 && max == 0) {
        Highcharts.setOptions({
            yAxis: {
                max: null,
                min: null
            }
        });
    } else {
        Highcharts.setOptions({
            yAxis: {
                max: max,
                min: min
            }
        });
    }
    nullChart();

},
dinamicChartStop: function () {
    clearInterval(dinamicaInterval);
},
staticChart: function (values) {
    var offset = offsetCalculates(values);
    var amplitude = amplitudeCalculate(values);
    var option = {
        title: {
            text: "Static Chart"
        },
        xAxis: {
            title: {
                text: "Sample"
            },
        },
        series: [{
                showInLegend: false,
                data: values
            }]
    };
    new Highcharts.Chart(option);
    return {
        offset: offset,
        samplesNumber: values.length,
        amplitude: amplitude
    };
},
secondDinamicChart: function (values) {
    var firstValue = [];
    var offset = offsetCalculates(values);
    var amplitude = amplitudeCalculate(values);
    var j = 1;
    firstValue.push({
        x: 0,
        y: values[0].y
    })
    var option = {
        title: {
```

```
                    text: "Dinamic II Chart"
                },
                xAxis: {
                    min: 0,
                    minRange: values.length,
                    title: {
                        text: "Sample"
                    },
                },
                chart: {
                    animation: Highcharts.svg,
                    events: {
                        load: function () {
                            var series = this.series[0];
                            var interval = setInterval(function () {
                             if (j < values.length) {
                              var xCoordinate = j
                              var yCoordinate = values[j].y

series.addPoint([xCoordinate,yCoordinate],true,false);
                              j++;
                              } else {
                                clearInterval(interval)
                              }
                            }, 500);
                        }
                    }
                },
                series: [{
                        showInLegend: false,
                        data: firstValue
                    }]
            }
            new Highcharts.Chart(option);
            return {
                offset: offset,
                samplesNumber: values.length,
                amplitude: amplitude
            };
        },
        dinamicChart: function (firstPoint) {
            var option = {
                plotOptions: {
                    spline: {
                        marker: {
                            enabled: true
                        },
                        lineWidth: 2
                    }
                },
                chart: {
                    animation: Highcharts.svg,
                    events: {
                        load: function () {
                            var series = this.series[0];
                            dinamicaInterval = setInterval(function () {
                                receive();
                                var point = getPoint();
                                series.addPoint([point.x, point.y], true, true);
                            }, 500);
                        }
                    }
```

```
                },
                title: {
                    text: "Dinamic I Chart"
                },
                series: [{
                        showInLegend: false,
                        data: (function () {
                            var data = [];
                            for (i = -10; i <= 0; i++) {
                                data.push({
                                    x: firstPoint.x + i * 500,
                                    y: firstPoint.y
                                });
                            }
                            return data;
                        }())
                    }],
                xAxis: {
                    title: {
                        text: "Time"
                    },
                    type: 'datetime',
                    tickPixelInterval: 150,
                }
            }
            new Highcharts.Chart(option);
        }
    }
}());
```

E' stato utilizzato il module pattern come paradigma di programmazione, che permette la creazione di oggetti mantenendo l'importante caratteristica, propria della programmazione ad oggetti, dell'incapsulamento. In particolare, dentro il costrutto di return saranno presenti i metodi e gli attributi accessibili (o pubblici), mentre tutti gli elementi privati sono definiti esternamente lo stesso. Di seguito sono riportati i codici dei file main.js e dynamics.js che permettono l'interazione tra classi e gestiscono gli eventi:

Main.js:

```
document.addEventListener("deviceready", onDeviceReady, false);
var values = [];
var value;

function onDeviceReady()
{
    bluetooth = Bluetooth;
    chart = Chart;
    bluetooth.isEnabled();
    setInitialOptionsAndEvents(bluetooth, chart);
}

$(document).on("mobileinit", function () {
    $.mobile.allowCrossDomainPages = true;
    $.support.cors = true;
    $.mobile.buttonMarkup.hoverDelay = 0;
```

```
    $.mobile.pushStateEnabled = false;
    $.mobile.defaultPageTransition = "none";
});

function receive() {
    bluetooth.readUntil();
}
function setArrayOfCoordinate(point) {
    values.push({
        x: point.x,
        y: point.y
    });
}

function getArrayOfCoordinate() {
    var arrayOfValues = values.slice(0);
    values.length = 0;
    return arrayOfValues;
}

function setPoint(point) {
    value = {
        x: point.x,
        y: point.y
    };
}
function getPoint() {
    return value;
}
function sendDataForChart(typeChart, values) {
    var signalValues;
    $.mobile.loading("hide");
    if (typeChart == "Static") {
        signalValues = chart.staticChart(values);
        printSignalValues(signalValues, bluetooth);
    } else {
        if (typeChart == "secondDinamic") {
            signalValues = chart.secondDinamicChart(values);
            printSignalValues(signalValues, bluetooth);
        }
    }
}
```

Dynamics.js:

```
function manageAttr(idTag, attribute, value, remove) {
    if (remove) {
        return $("#" + idTag).removeAttr(attribute);
    } else {
        return $("#" + idTag).attr(attribute, value);
    }
}
function manageProp(idTag, attribute, value, check) {
    if (check) {
        return $("#" + idTag).prop(attribute);
    } else {
        return $("#" + idTag).prop(attribute, value);
    }
```

```
}
function connectionEstablished() {
    $.mobile.loading("hide");
    navigator.notification.alert("The device is connected", null, "Bluetooth State");
    manageAttr("homeToChart","href", "#chart", false);
    manageProp("connectButton", "disabled", true, false);
    manageProp("disconnectButton", "disabled", false, false);
}
function connectionFailed() {
  $.mobile.loading("hide");
      navigator.notification.alert("The device is disconnected",null,"Bluetooth
                                    State");
  manageAttr("homeToChart", "href", null, true);
  manageProp("connectButton", "disabled", false, false);
  manageProp("disconnectButton", "disabled", true, false);
}
function printSignalValues(arrayOfCharacteristics, bluetooth) {
 $("#popupInfoOffset").text("Offset: " + arrayOfCharacteristics.offset);

$("#popupInfoSamples").text("NumberOfSamples:"+arrayOfCharacteristics.samplesNumber
+ " in: " + bluetooth.getAcquisitionTime() + " milliSec.");
 $("#popupInfoAmplitude").text("Amplitude:  "  +  arrayOfCharacteristics.amplitude+"
V")
}

function setInitialOptionsAndEvents(bluetooth, chart) {

    var defaultMac = bluetooth.getDefaultMac();

    manageProp("disconnectButton", "disabled", true, false)
    manageProp("macA", "disabled", true, false)
    if (manageProp("defaultMac", "checked", null, true)) {
        manageAttr("macA", "value", defaultMac);
    } else {

    }

    $("#disconnectButton").click(function () {
        $.mobile.loading("show", {
            text: "I'm Disconnecting",
            textVisible: true
        })
        bluetooth.disconnect();
    });

    $("#connectButton").click(function () {
        $.mobile.loading("show", {
            text: "I'm Trying To Connect",
            textVisible: true
        })
        var mac = $("#macA").val();
        bluetooth.connect(mac);
    });

    $("#defaultMac").change(function () {
        if (manageProp("defaultMac", "checked", null, true)) {
            manageAttr("macA", "placeholder", "", false);
            manageAttr("macA", "value", defaultMac, false);
            manageProp("macA", "disabled", true, false);
        } else {
            manageAttr("macA", "value", null, false)
```

```
                manageAttr("macA", "placeholder", "AA:BB:CC:DD:EE:FF", false)
                manageProp("macA", "disabled", false, false);
            }
        });




        chart.initialize();

        $("#setYMinMax").click(function () {
            chart.modifyOrdinateScale($("#minY").val(), $("#maxY").val());
        });

        $("#secondDinamicChart").click(function () {
            chart.dinamicChartStop();
            $.mobile.loading("show", {
                text: "Loading",
                textVisible: true
            })
            bluetooth.clearBuffer();
            bluetooth.subscribe(null, "secondDinamic");
        })

        $("#staticChart").click(function () {
            chart.dinamicChartStop();
            $.mobile.loading("show", {
                text: "Loading",
                textVisible: true
            })
            bluetooth.clearBuffer();
            bluetooth.subscribe(null, "Static");
        });

        $("#dinamicChart").click(function () {
            bluetooth.readUntil();
            chart.dinamicChart(getPoint());
        });
    }
```

Si riporta infine il codice html che definisce la grafica dell'applicazione, omettendo i file .css in quanto si è fatto largo uso dello stylesheet fornito dal framework jQuery Mobile.

Index.hmtl

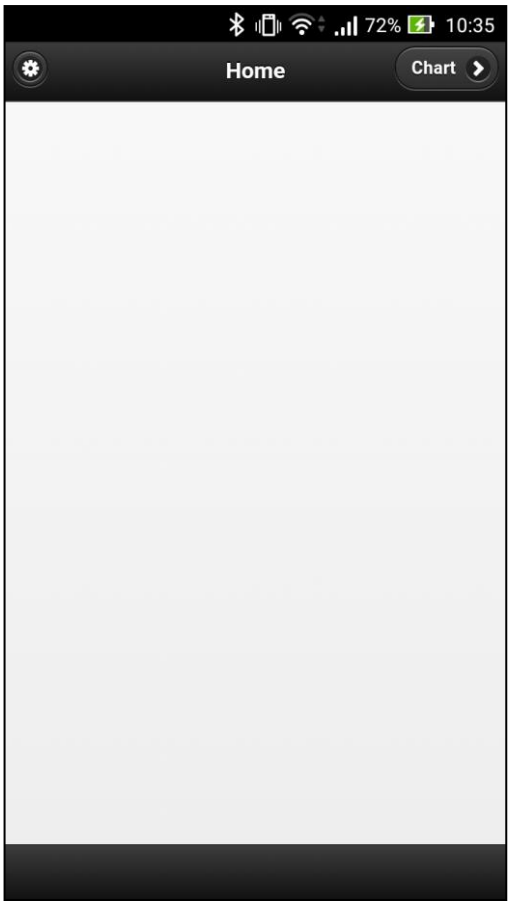
```
<!DOCTYPE html>
<html>
    <head>
      <meta charset="UTF-8" />
      <meta name="viewport" content="width=device-width, initial-scale=1">
      <title>Digital Oscilloscope</title>
      <link rel="stylesheet" href="css/jquery.mobile-1.3.1.min.css" type="text/css">
      <link rel="stylesheet" href="css/app.css" type="text/css">
      <script type="text/javascript" src="js/lib/highcharts.js"></script>
      <script type="text/javascript" src="js/lib/jquery.js"></script>
      <script type="text/javascript" src="js/Coordinate.js"></script>
```

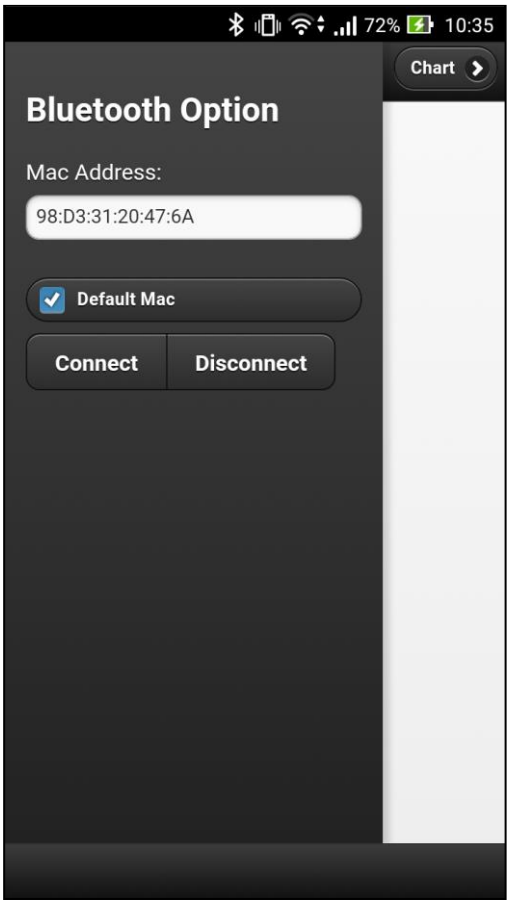
```
  <script type="text/javascript" src="js/Chart.js"></script>
  <script type="text/javascript" src="js/Bluetooth.js"></script>
  <script type="text/javascript" src="js/dynamics.js"></script>
  <script type="text/javascript" src="js/app.js"></script>
  <script type="text/javascript" src="js/lib/jquery.mobile-1.3.1.min.js"></script>
  <script type="text/javascript" src="cordova.js"></script>
</head>
<body>
   <!-- HOME PAGE -->
   <div data-role="page" id="home">
     <div data-role="header">
     <h1>Home</h1>
     <a href="#bluetoothOption" data-icon="gear" data-iconpos="notext"></a>
     <a id="homeToChart" href="" class="ui-btn-right" data-role="button" data-iconpos="right" data-
     icon="arrow-r">Chart</a>
     </div>
     <div data-role="content" id="homeContent">
         <!--panel-->
            <div  data-role="panel"  data-position="left"  data-position-fixed="true"  data-display="overlay"
            data-theme="a" id="bluetoothOption">
                <h2>Bluetooth Option</h2>
                <label for="macA">Mac Address:</label>
                <input type="text" id="macA" placeholder="" value="" data-mini="true">
                <br>
                <label for="defaultMac">Default Mac</label>
                <input id="defaultMac" data-theme="a" type="checkbox" checked="checked" data-mini="true">
                <div data-role="controlgroup" data-type="horizontal" id="connectionButton">
                <input type="button" value="Connect" id="connectButton" data-theme="a">
                <input type="button" value="Disconnect" id="disconnectButton" data-theme="a">
                </div>
            </div><!-- /panel -->
     </div>
        <div data-role="footer" data-position="fixed">
            <br>
            <br>
        </div>
    </div>



    <!-- CHART PAGE -->
    <div data-role="page" id="chart" data-add-back-btn="true">
        <div data-role="header">
            <h1>Chart</h1>
             <a href="#popupInfo" data-rel="popup" data-role="button" class="ui-btn-right" data-inline="true"
             data-transition="pop" data-icon="info" data-theme="a" data-iconpos="notext"></a>
        </div>
        <!--Popup-->
        <div data-role="popup" id="popupInfo" class="ui-content" data-theme="a">
           <p id="popupInfoOffset"></p>
           <p id="popupInfoSamples"></p>
           <p id="popupInfoAmplitude"></p>
        </div>
        <!--Popup-->
        <div data-role="content" id="chartdiv">
        </div>
        <div id="setYAxe">
            <form>
                <div data-role="rangeslider" data-mini="true">
                    <input type="range" name="range-2a" id="minY" min="-10" max="10" value="0">
                    <input type="range" name="range-2b" id="maxY" min="-10" max="10" value="0">
                </div>
            </form>
            <input type="button" value="Submit" id="setYMinMax" data-role="button" data-mini="true">
        </div>
        <div data-role="footer" data-position="fixed">
            <div data-role="navbar" data-iconpos="right">
                <ul>
```

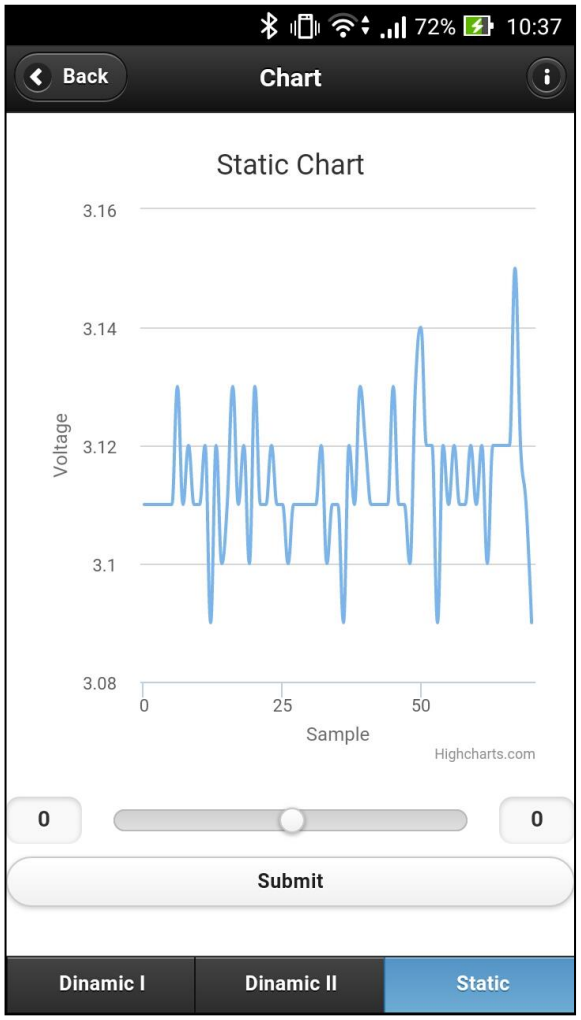
```
                        <li><a id="dinamicChart">Dinamic I</a></li>
                        <li><a id="secondDinamicChart">Dinamic II</a></li>
                        <li><a id="staticChart">Static</a></li>
                    </ul>
                </div>
            </div>
        </div>
    </body>
</html>
```

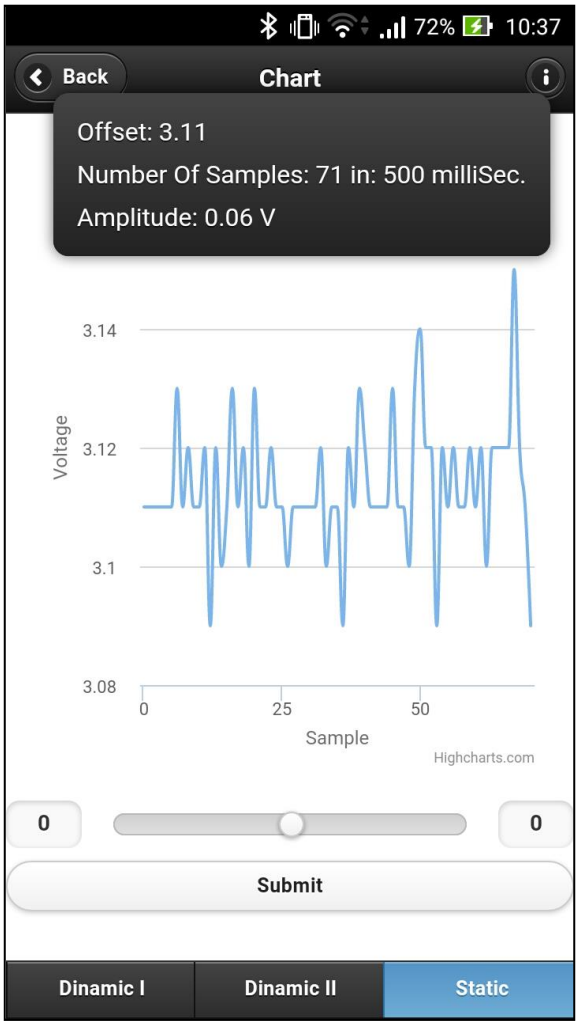
Dall'html si vede che l'applicazione è di tipo Single-Page, dove tutto il codice necessario al funzionamento è recuperato in un singolo caricamento della pagina. Questo fa in modo che non sia necessario ricaricare in nessun punto le pagine html e rende più fluido l'applicativo.

Quest'ultimo è composto da due schermate, una prima, la home, dalla quale si effettua la connessione bluetooth per la ricezione dei dati, ed è fornita di un campo input nel quale inserire il MAC address del dispositivo. La seconda schermata è la parte principale dell'applicazione e si occupa di graficare la forma d'onda, di fornire informazioni relative al segnale (come offset, ampiezza e numero di campioni acquisiti), e di modificare la scala dell'asse delle ordinate. In particolare questa ha un range massimo di ±10 V che è adattativo alle esigenze dello strumento progettato. Il valore di default ±0 V permette il resize automatico dell'asse delle ordinate, centrandola rispetto ai valori massimi e minimi del segnale.







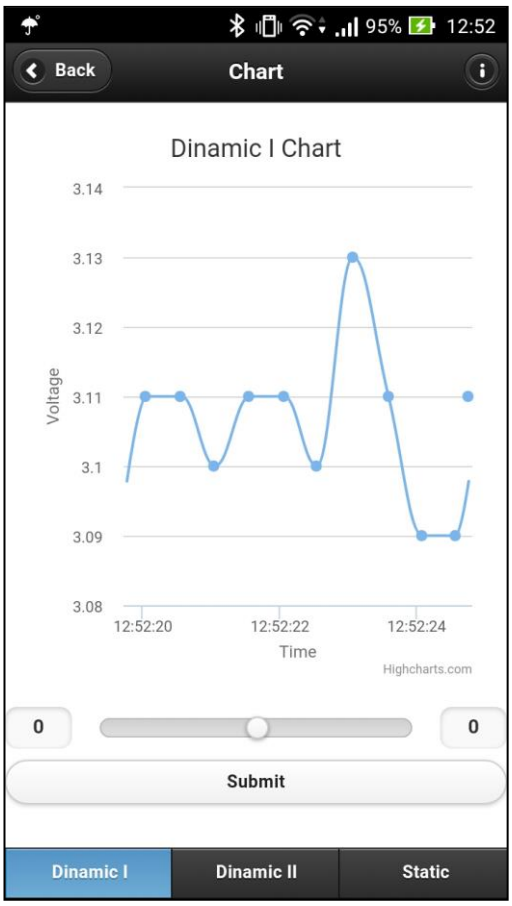


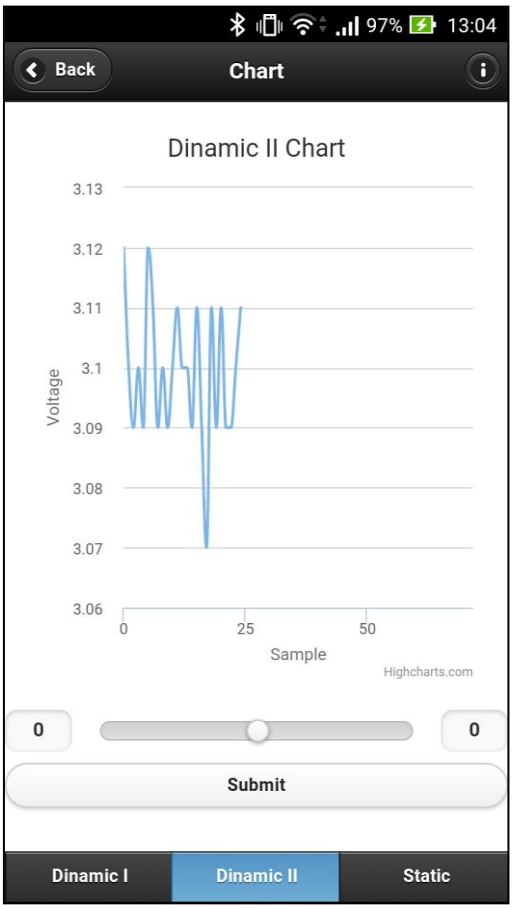
Si sono implementate tre modalità di graficamento del segnale:

a. *Static:* Consiste nell'acquisizione di un numero arbitrario di campioni. Quest'ultimo dipende dalla velocità del bluetooth in quanto viene salvato ogni campione ricevuto nell'arco di 500 ms. Questi vengono poi graficati. Un'esempio di questa modalità è dato dalle immagini soprastanti. Si vede che l'asse delle ordinate ha come unità di misura il Volt, mentre quella delle ascisse, il numero di campioni.

b. *Dinamic I:* Consiste nel visualizzare l'evoluzione del segnale, stampando un campione ogni 500 ms. Questo tempo è stato scelto come miglior compromesso tra: tempo minore possibile e tempo minore possibile che le librerie grafiche riescono a gestire senza creare problemi di stampa della forma d'onda. Si riporta una schermata di questa modalità, e da come si può vedere si ha, come unità di misura dell'asse delle ordinate, il tempo reale.

c. *Dinamic II:* L'ultima modalità progettata, è un misto tra le due già descritte. Cioè si acquisiscono un numero arbitrario di campioni in un arco di tempo predefinito (500 ms). Questi campioni non vengono stampati come nella modalità Static, ma viene stampato ogni 500 ms un campione per evidenziare l'evoluzione del segnale, fino all'esaurimento del numero di campioni che si era acquisito. Si riporta in basso lo screen relativo a quest'ultima modalità.

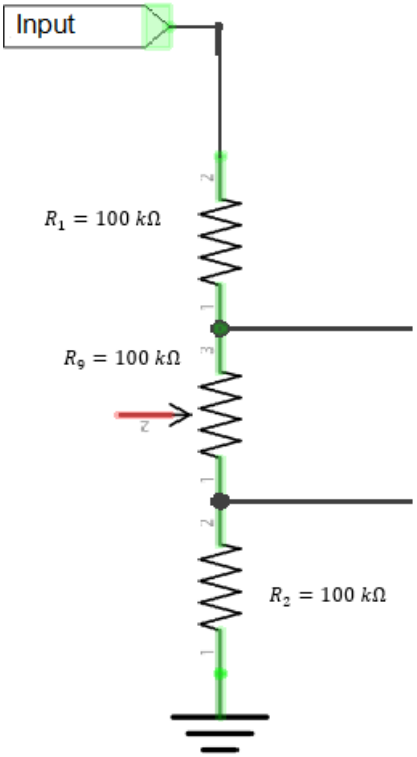


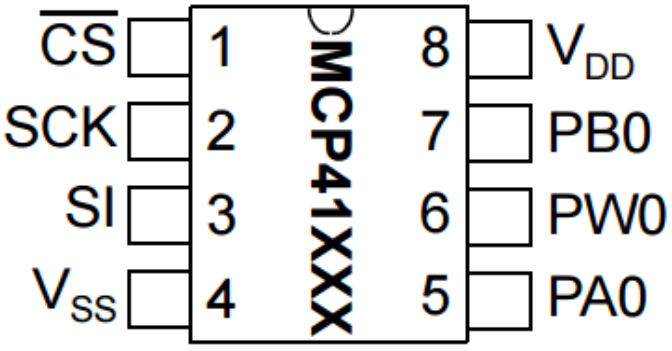


3.2 Realizzazione Hardware

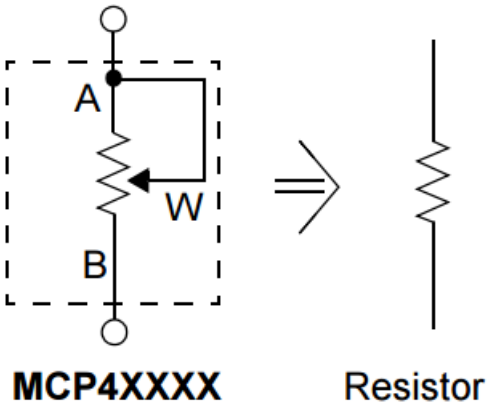
Si riporta, a fine paragrafo, lo schema elettrico della versione definitiva del circuito. Come si vede, rispetto a quelle preliminare si è notevolmente semplificato. Questa scelta è stata fatta in relazione al fatto che si è posta maggiore importanza alla parte implementativa piuttosto che elettronica. Quest'ultima può essere oggetto di approfondimento da parte di specialisti nel campo al fine di avere uno strumento migliore in qualità, potenzialità e precisione.

Il primo stadio del circuito è un semplice partitore di tensione composto da tre resistenze, di cui una variabile. Questo per permettere di avere un fattore di attenuazione variabile in base alle esigenze. In particolare se consideriamo il caso in cui il potenziometro è al suo valore massimo ossia: $R_9 = 100\ k\Omega$, si ha che il fattore di attenuazione sarà pari a: $A = \frac{R_9+R_2}{R_9+R_1+R_2} = \frac{2}{3}$ e $A = \frac{R_2}{R_9+R_1+R_2} = \frac{1}{3}$, da cui $V_1 = \frac{2}{3} V_{in}$ e $V_2 = \frac{1}{3} V_{in}$ e quindi $\Delta V = \frac{1}{3} V_{in}$. Per poter regolare il fattore di attenuazione, basterà modificare il valore del potenziometro. Il potenziometro scelto è in particolare l' MCP41100 da $100\ k\Omega$, del quale si riporta lo schema:

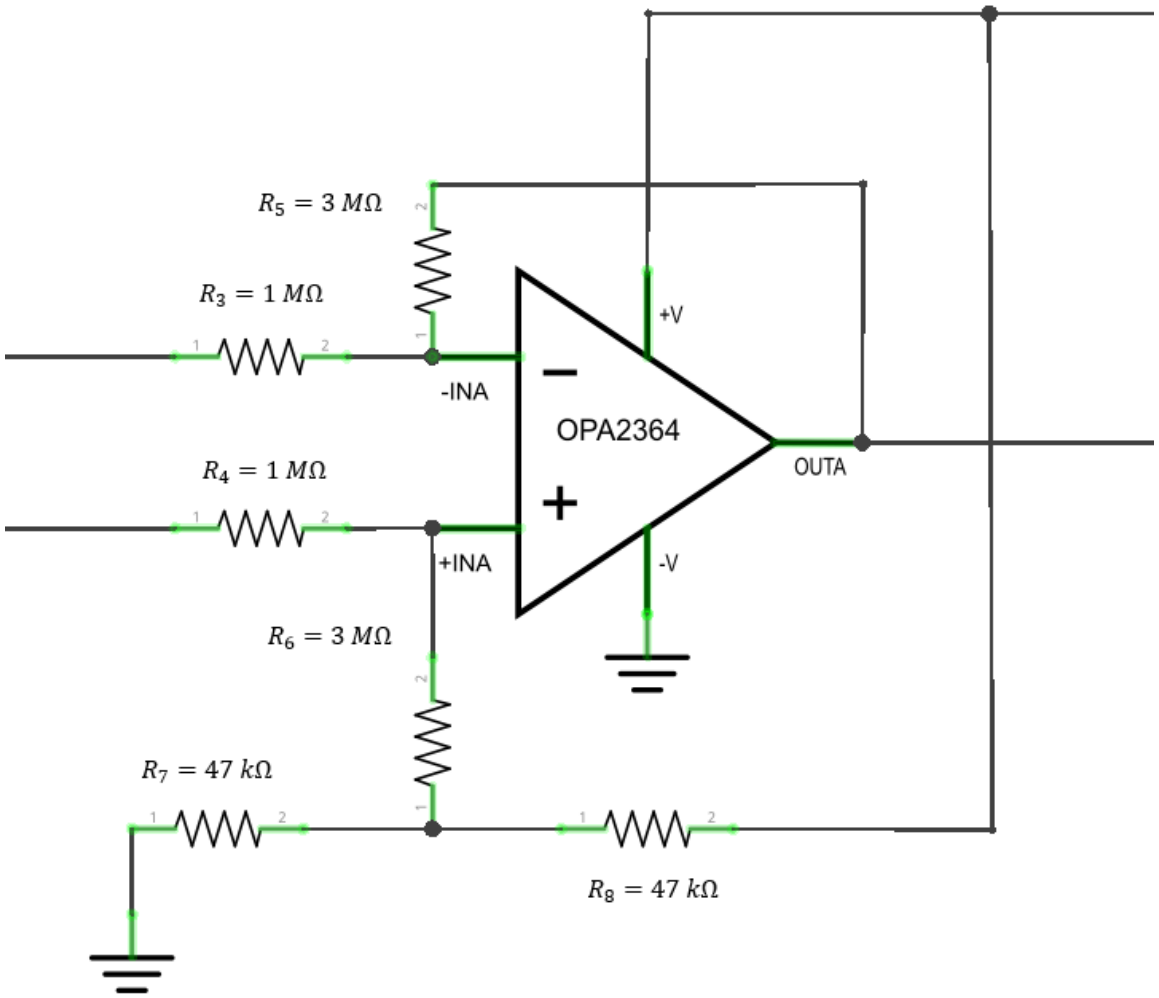




Dove il Pin $\overline{CS}$ è collegato al Pin digitale 10 di Arduino, SCK è collegato al Pin digitale 13 di Arduino, SI è collegato al Pin digitale 11 di Arduino, V_{ss} a massa, V_{dd} ai $5\ V$ di Arduino, P_{A0} a massa, e P_{B0} e P_{W0} sono rispettivamente la prima e la seconda uscita del partitore. Il segnale di ingresso è collegato al Pin P_{B0} del componente. Quest'ultimo comunica con Arduino attraverso il protocollo SPI che facilita la sincronizzazione. Questo componente ha un range di resistenza che va da un minimo di $125\ \Omega$ dovuto alla resisteste di Wiper, ad un massimo di $100\ k\Omega$



secondo l'equazione $R = \frac{R_{AB} * D_n}{256} + R_w$ con $R_w = 125\Omega$ $D_n \in [0,255]$ e $D_n \in \mathbb{N}$. Il componente, che permette due modalità di utilizzo (reostato e potenziometro), è usato in modalità reostato, che permette di vedere il componente come un resistore. Il secondo stadio invece è dato da un operazionale differenziale. I valori delle resistenze sono presi molto grandi in modo che sia assorbita la minore corrente possibile. Si



suppone il caso ideale in cui non l'operazionale non assorba corrente. Gli ingressi sono le uscite del partitore di tensione, ed inoltre a quello non invertente, è presente una continua di $2{,}5\ V$ data da un partitore di tensione con due resistenze uguali, e dai $5\ V$ di Arduino. L'uscita di un amplificatore differenziale è data da:

$V_{out} = A_d \Delta V + A_c(\frac{V^+ + V^-}{2})$. Con $A_d = \frac{R_5}{R_3} = 3$ guadagno di modo differenziale, ed A_c guadagno di modo comune che si suppone idealmente nullo. Questo impone che la frequenza di taglio dell'operazionale sia: $f_t = \frac{GBW}{3} = 333\ kHz$. L'uscita dell'operazionale coincide con l'ingresso analogico di Arduino. Infine, l'alimentazione è di $5\ V$ per quanto riguarda il Pin positivo (prelevata da Arduino), e massa per quanto riguarda quello negativo.

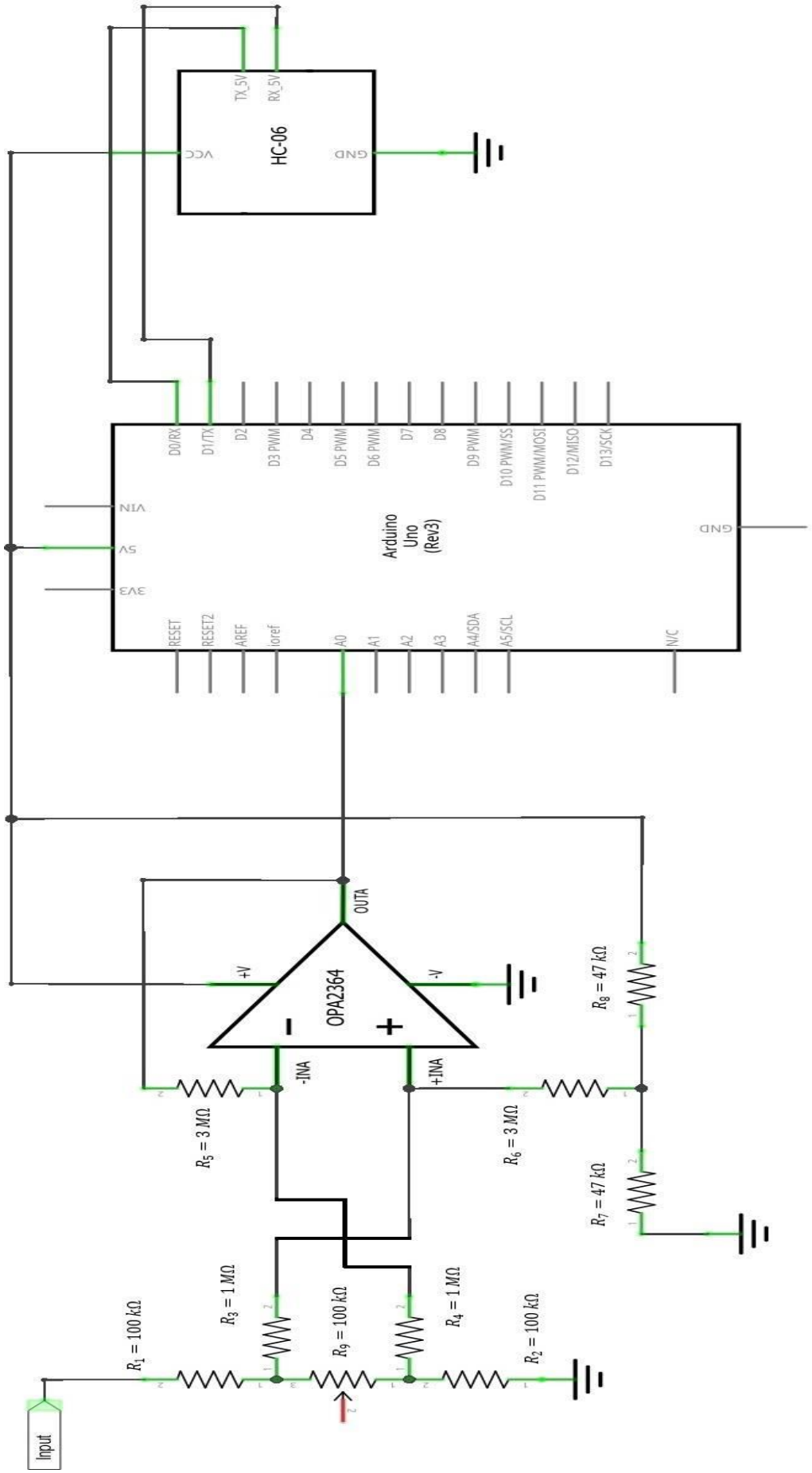
Dai calcoli eseguiti risulta quindi che, essendo $\Delta V = \frac{1}{3} V_{in}$ e $V_{out} = 3\Delta V + 2{,}5$ (sempre supponendo che $A_c = 0$, $R_9 = 100\ k\Omega$, la continua sull'ingresso non invertente e che V_{in} sia un segnale a valore medio nullo centrato in zero) da cui $V_{out} = V_{in}$, per cui si ha in uscita $V_{out} = V_{in} + 2{,}5V$, cioè lo stesso segnale che si ha in ingresso, ma con dinamica centrata a $2{,}5\ V$. Questo impone che, supponendo di avere in ingresso un segnale alternato, esso abbia al più ampiezza di $2{,}5\ V$ in modo da rispettare le imposizioni sugli ingressi analogici di Arduino ed in modo da evitare di essere tagliato dal''operazionale. I $2{,}5\ V$ vengono poi sottratti a livello software da Arduino in modo da stampare il segnale centrato in 0. Come detto, modificando il valore del potenziometro, cambierà il fattore di attenuazione. In particolare minore sarà il valore della resistenza R_9, maggiore sarà il fattore di attenuazione, ad esempio con $R_9 = \frac{R_1}{2} = \frac{R_2}{2}$, risulterà $V_{out} = \frac{3}{5} V_{in} + 2{,}5$ e quindi si ha in uscita il segnale che si ha in ingresso, con dinamica centrata in $2{,}5V$ e attenuato di un fattore $\frac{3}{5}$. In generale, posto $R_9 = \frac{R_1}{n} = \frac{R_2}{n}$ con $n \to \infty$, si ha che $V_2 = \frac{\frac{R}{n}+R}{\frac{R}{n}+2R} V_{in} = \frac{n+1}{2n+1} V_{in} \approx \frac{1}{2} V_{in}$ e $V_1 = \frac{R}{\frac{R}{n}+2R} V_{in} = \frac{n}{2n+1} V_{in} \approx \frac{1}{2} V_{in}$ da cui $\Delta V = 0$ e quindi $V_{out} = 0$ cioè si ha attenuazione massima tale da annullare il segnale. L'idea alla base del circuito è quella di sfruttare il fatto di conoscere a priori la resistenza R_9 (essendo programmata da Arduino stesso) e poter quindi ricostruire il segnale a livello software. In particolare si partirà da un valore di R_9 minimo (in modo da avere un fattore di attenuazione massimo), se Arduino non rileva alcun segnale vuol dire che si è attenuato troppo per cui si aumenta la resistenza e quindi si diminuisce l'attenuazione. Quando si riceve il segnale, si conoscerà a priori il fattore di attenuazione applicato che si indica con A_t, si conosce il guadagno dell'operazionale $A_d = 3$ e si conosce il valore della tensione continua sull'ingresso non invertente $2{,}5\ V$. Per cui eseguendo specularmente, a livello software, le operazioni suddette cioè:

a. Si moltiplica per $1/A_t$.

b. Si divide per $A_d = 3$.

c. Si sottrae la tensione aggiunta $2{,}5$.

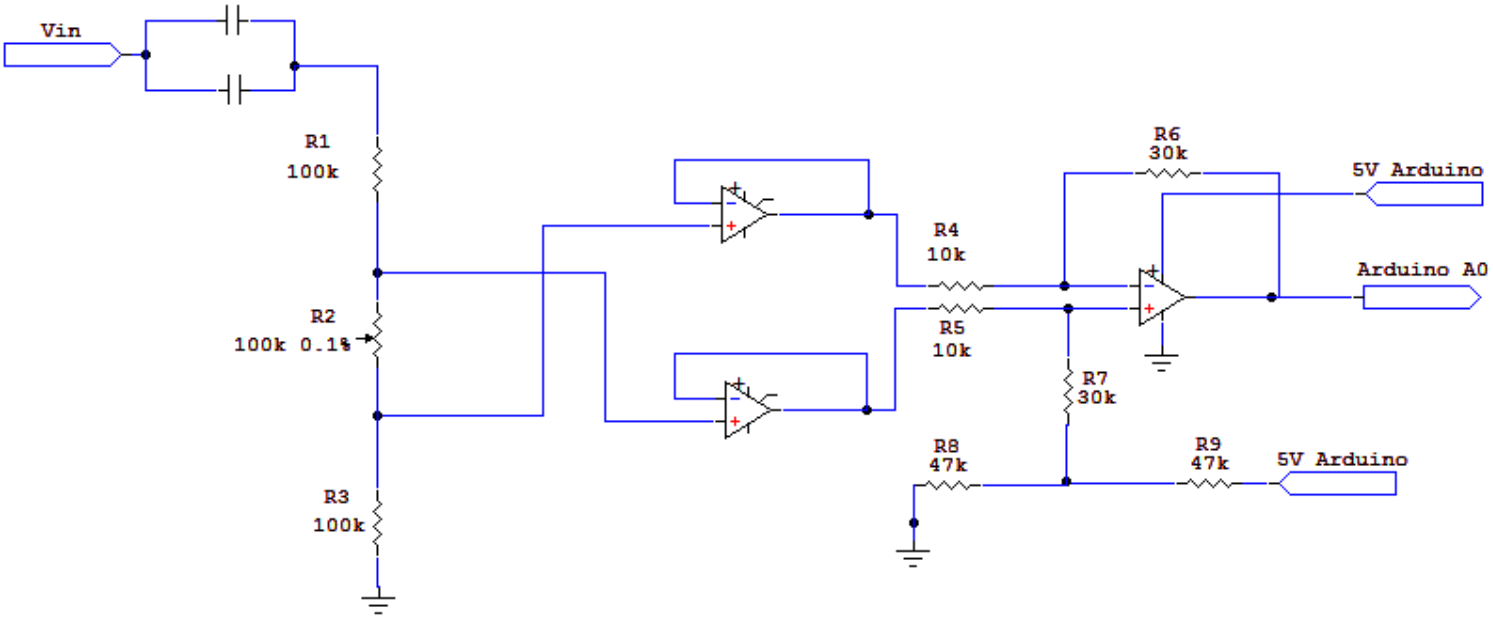
Abbiamo ricostruito il segnale in ingresso. Il segnale, una volta ricostruito, sarà inviato da Arduino tramite protocollo bluetooth ad un terminale, dove sarà stampato. Il vantaggio di questo circuito è che, non necessita di alimentazione esterna in quanto alimentato da Arduino stesso.


HC-06
TX_5V
RX_5V
VCC
GND
Arduino
Uno
(Rev3)
D0/RX
D1/TX
D2
D3 PWM
D4
D5 PWM
D6 PWM
D7
D8
D9 PWM
D10 PWM/SS
D11 PWM/MOSI
D12/MISO
D13/SCK
VIN
5V
3V3
RESET
RESET2
AREF
Ioref
A0
A1
A2
A3
A4/SDA
A5/SCL
N/C
OUTA
+V
-V
OPA2364
-INA
+INA
$R_1 = 100\,k\Omega$
$R_2 = 100\,k\Omega$
$R_3 = 1\,M\Omega$
$R_4 = 1\,M\Omega$
$R_5 = 3\,M\Omega$
$R_6 = 3\,M\Omega$
$R_7 = 47\,k\Omega$
$R_8 = 47\,k\Omega$
$R_9 = 100\,k\Omega$
Input

4. Conclusioni

Si è visto, sperimentalmente, che per quanto riguarda le frequenze di lavoro dello strumento, contrariamente ai valori di targa forniti da Arduino, il segnale si ricostruisce correttamente se la frequenza dello stesso non supera i $400\ Hz$. Questa limitazione può essere eliminata usando un controllore diverso da Arduino Uno, come ad esempio il Due il quale ha prestazioni più elevate, o sviluppando un circuito che campioni e converta il segnale in digitale, quindi non usando il campionatore di Arduino, e inviare direttamente il segnale digitalizzato. A quel punto il limite superiore è dato dalla velocità di trasmissione del protocollo Bluetooth che a sua volta dipende dalla versione dello stesso, ad esempio la versione 2.0 gestisce velocità fino a $3\ {Mbit}/{s}$ che aumenta nella versione 4.0. Tramite simulazione si è visto che, pur essendo le resistenze di ingresso dell'operazionale notevolmente grandi, lo stesso assorbe corrente rendendo inutile la regola del partitore, ed inoltre introducono delle distorsioni del segnale dovute proprio alla loro grandezza. Inoltre la limitazione che il segnale debba essere centrato in 0 è dovuta al fatto che, essendo ulteriormente traslato di $2{,}5\ V$, la dinamica del segnale sarebbe drasticamente ridotta per far fronte alle esigenze di Arduino. In ogni caso, per ovviare a questo problema basterebbe inserire un parallelo di condensatori in ingresso, i quali eliminano la componente continua del segnale. Quindi un leggero perfezionamento dello stadio di ingresso è dato dal circuito a fine paragrafo, dove sono stati aggiunti dei condensatori per eliminare la componente continua, e una coppia di buffer per evitare assorbimenti di corrente da parte dell'operazionale.

Ovviamente non avendo modo di conoscere a priori a quanto questa corrisponde, non si può ricostruire fedelmente il segnale, ma sempre centrato ad una dinamica di $2{,}5\ V$. In conclusione, lo strumento si è rilevato molto utile a livello didattico per esercitazioni di laboratorio dove si lavora a basse frequenze, e, con opportuni perfezionamenti fisici, potrebbe essere usato per la diagnostica e commercializzato.



5. Riferimenti e sitografia

1. Dispense di Misure Elettroniche del Prof. Bucci.
2. The principles of Object-Oriented Javascript – Nicholas Zakas
3. HC-06 Datasheet.
4. https://cordova.apache.org/
5. http://www.w3schools.com/
6. https://jquerymobile.com/
7. https://www.arduino.cc/
8. http://www.microchip.com/wwwproducts/en/MCP41100
9. http://www.ing.unibs.it/~wsnlab/download/bluetooth/bosio/WSNlab_tutorial04_bluetooth.pdf
10. https://www.princeton.edu/ssp/exp.-methods/mae511.pdf
11. https://it.wikipedia.org/wiki/HTML
12. https://it.wikipedia.org/wiki/CSS
13. https://it.wikipedia.org/wiki/JavaScript