ITIS GALILEO GALILEI ROMA

Via Conte Verde 51 00185 Roma (RM)
E-mail istituto: rmtf090003@istruzione.it

GENERATORE DI FUNZIONI CON CONVERTITORE DIGITALE-ANALOGICO E MICROCONTROLLORE

31 / 05 / 2021 Emanuele Coletta

Indice

Ragioni e motivazioni dietro lo sviluppo di questo progetto

Ragioni, motivazioni e scopi

La decisione di costruire un generatore di segnali digitale con microcontrollore fa parte del più grande progetto di costruire un laboratorio di elettronica in casa, dotato di tutti gli strumenti per la progettazione, l'assemblaggio e la prototipazioni di schede e circuiti elettronici. La necessità di avere un luogo del genere in casa si è fatta sempre più pressante nel corso dell'ultimo anno e mezzo, in cui, a causa della pandemia, l'accesso ai laboratori della scuola non è sempre stato facile e per alcuni periodi addirittura impossibile. Alla lista di strumenti già presenti nel mio piccolo laboratorio casalingo (saldatori, oscilloscopio, alimentatore stabilizzato) mancava solamente un generatore di segnali. La decisione migliore sembrava essere l'uso di un generatore di segnali digitale, poiché questi ultimi offrono un minore ingombro rispetto a quelli analogici, caratteristica ottima per i limitati spazi a mia disposizione, e sono estremamente precisi in frequenza e non ne soffrono la deriva. Essi sono inoltre composti di una serie di elementi che sintetizzano l'esperienza didattica di tutto il triennio del nostro corso di studi: i convertitore digitali-analogici, i filtri attivi e passivi del primo e di ordini superiori, l'uso e la programmazione dei microcontrollori, il condizionamento dei segnali e l'uso di componenti a semiconduttori come amplificatori operazionali, transistor e diodi. A questo si aggiungono le tecniche digitali per la generazione di segnali, che in parte sfuggono alla programmazione del corso di automazione ma espandono l'ambito di quanto imparato in questi tre anni.
Il progetto è comunque da considerarsi solamente uno studio di fattibilità: i margini per l'espansione e il miglioramento sono ancora enormi, e offrono l'osccasione per imparare molte tecnologie interessanti.

Perché un generatore di segnali digitale?

Un classico generatore di segnali analogico è costituito da un oscillatore principale, che generalmente offre un segnale sinusoidale, da cui vengono poi ricavate, con singoli circuiti appositi, tutte le altre forme d'onda. Anche l'aggiunta di funzioni come lo *sweep* richiede l'aggiunta di un altro circuito.
L'avvento negli ultimi anni dei microcontrollori ha indubbiamente aperto le frontiere per una rivoluzione nel mondo dell'elettronica. La possibilità di riprogrammarne le azioni li rende virtualmente adattabili a ogni situazione, facendo sì che una stessa configurazione *hardware* possa eseguire azioni diverse al variare del *software.*
Un microcontrollore e un DAC (Digital-Analog Converter / Convertitore Digitale Analogico) possono sostituire facilmente anche maggior parte del circuiti presenti un un generatore di segnali analogico.
La Sintesi Diretta Digitale (Direct Digital Synthesis/DDS in inglese) è una tecnica digitale per la generazione di segnali. Inizialmente implementata con componenti a logica sequenziale, oggi viene implementata soprattutto con microcontrollori o FPGA. I generatori di segnali digitali sono quelli che fanno uso di questa tecnica. Essi offrono enormi vantaggi nell'ambito della frequenza: sono estremamente precisi, anche a frequenze minori di 1 Hertz, e non soffrono la deriva di frequenza tipica dei generatori analogici, fenomeno che aumenterebbe con il surriscaldamento, l'usura e l'invecchiamento dei componenti passivi all'interno di un generatore analogico. I generatori di segnali digitali impiegano inoltre un tempo infinitesimo a passare da un segnale di una determinata frequenza ad uno di una frequenza totalmente diversa rispetto a quelli analogici. Questa precisione elimina anche la necessità di un frequenzimetro per mostrare all'utente la frequenza del segnale generato.
La presenza dei microcontrollori permette anche la riprogrammazione remota dei generati di funzioni digitali, caratteristica sempre più richiesta nell'ambito dell'automazione industriale.

Perché due microcontrollori?

La scelta della coppia di microcontrollori usata deriva dalla mia esperienza con entrambi nel laboratorio di robotica della scuola. Inoltre, all'interno del laboratorio di TPSEE, ogni studente della classe ha sviluppato una scheda microcontrollore "general purpose" basata su Atmega 32u4.
Il Teensy 3.5 è un potente microcontrollore Arduino-compatibile con processore ARM Cortex M4, con frequenza di clock a 120 MHz, "overclockabile" fino a 168 MHz per . Ogni ciclo di clock (che dura solo nove nanosecondi) di questo microcontrollore è dedicato esclusivamente alla generazione dei segnali. Interrompere il ciclo della generazione di segnali con altre azioni (come aggiornare il display LCD, oppure leggere i dati in input da altre fonti come l'encoder) porterebbe inevitabilmente all'instabilità della frequenza della forma d'onda e talvolta alla deformazione della stessa.
Per evitare questo problema Teensy 3.5 viene affiancato da un Atmega 32u4, che si occupa esclusivamente dell'interfacciamento con l'utente (attraverso l'encoder), della lettura delle informazioni (dopo il condizionamento) sul segnale e mostrare il tutto su un display LCD. Questo toglie carico computazionale al Teensy 3.5, permettendo l'effettiva stabilità in frequenza del segnale generato, evitando deformazioni e deriva.
Il Teensy 3.5 non è tuttavia il microcontrollore più adatto per questo lavoro, per motivazioni discusse in seguito, ma ad un modico prezzo offre ben due DAC e un'elevata frequenza di clock. Il circuito stampato è stato comunque progettato ed assemblato per poter sostituire facilmente il Teensy 3.5 con un altro microcontrollore, con DAC integrato o esterno, previa progettazione di una scheda di adattamento che si colleghi ai pin già presenti ed esposti sulla scheda, a cui normalmente è collegato il Teensy.

Costi

I componenti scelti per questo progetto sono componenti di tipo consumer, che rendono il progetto facilmente accessibile e replicabile da chiunque. Di seguito segue una tabella dei principali componenti utilizzati.
L'effettivo importo speso per la realizzazione di questo progetto è leggermente più alto, in quanto sono stati scelti rivenditori che potessero offrire una spedizione molto rapida (il Teensy 3.5 per esempio viene spedito dagli Stati Uniti, e scegliendo la spedizione più economica i tempi di spedizione sarebbero arrivati fino a due mesi). Al totale dei componenti e della board si devono aggiungere 9€, costo di spedizione fisso dal sito tme.eu dove sono stati reperiti i componenti. Inoltre, su questo store online, è possibile l'acquisto di una quantità minima di 100 componenti, quando si tratta di resistenze e condensatori.
Le PCB sono state fatte stampare a Shenzen, in Cina, dall'azienda JLCPCB, che offre la stampa (in cinque copie) di PCB con entrambi i lati più corti di 10cm a soli 2€. A questi si deve aggiungere un costo minimo di 5€ di spedizione col corriere ePacket.

Componente	Costo/unità [€]	Quantità	Costo totale [€]
Teensy 3.5	20	1	20
OpAmp AD8034	3.50	6	21
Encoder incrementale	3	1	3
Connettore bnc	1.5	1	1.5
Relè di segnale	1.1	2	2.2
Atmega 32u4	3	1	3
Componentistica varia*	\	\	8
Realizzazione PCB	0.40	5	2
Totale	\	\	63.70
Totale con spedizioni (costi minimi possibili)	\	\	79.70

*(resistenze, condensatori, strip, oscillatori al quarzo, potenziometri e trimmer, transistor, diodi)

I limiti minimi di acquisto di TME, l'acquisto di qualche componente di scorta per effettuare dei test e la scelta dei servizi di spedizione più veloce ha fatto aumentare leggermente il prezzo finale. Tuttavia, questi problemi sarebbero facilmente risolti in un contesto di produzione di massa.
Io personalmente non considero l'acquisto di componenti in eccesso una spesa, ma un investimento che allevierà il costo dello sviluppo di progetti futuri.

Il generatore

Direct Digital Synthesis: tecniche digitali per la generazione di segnali

La sintesi diretta digitale (Direct Digital Synthesis in inglese, abbreviato in DDS) è una tecnica per la generazione di segnali che si basa su componenti digitali invece che analogiche. Il segnale viene campionato in un numero *N* di punti, il cui valore di tensione corrispondente viene immagazzinato in una *tavola di lookup*, che sostanzialmente è una memoria ROM o flash. In base alla frequenza che si vuole generare, si utilizzeranno più o meno campionamenti della forma d'onda in un singolo periodo. La frequenza del segnale e il numero di campioni usati in un singolo periodo sono inversamente proporzionali. Un normale generatore DDS si compone principalmente di tre blocchi funzionali:
Il primo è un clock di sample. Dà la frequenza di clock a tutto il circuito per decidere "il ritmo" delle operazioni. La frequenza del segnale in uscita non è però vincolato a quella di clock.
Il secondo è *l'accumulatore di fase*. È il blocco che si occupa di scegliere quali e quanti campioni utilizzare. A sua volta è costituito da due componenti: un *addizionatore* e un *numero a virgola mobile*, il cui valore dipende dalla lunghezza della *tavola di lookup*, la frequenza di clock del circuito e quella desiderata per il segnale in uscita. Ad ogni ciclo di clock *l'addizionatore* aggiunge il *numero* ad un registro. Nel caso il *numero* diventi maggiore della grandezza della *tavola di lookup,* il valore di quest'ultima gli viene sottratto per evitare overflow.
La tavola di lookup, ultimo stadio prima del DAC, viene usato come convertitore fase-ampiezza (spesso definito anche come convertitore fase-tensione). Ad ogni ciclo di clock il punto all'interno della *tavola di lookup* corrispondente al valore dell'accumulatore di fase viene convertito in valore analogico dal DAC. Proseguendo questo procedimento più volte nel tempo, si modula l'uscita del DAC in ampiezza in modo che formi un segnale con andamento sinusoidale nel tempo. Ovviamente la *tavola di lookup* può essere impostata per immagazzinare un segnale di qualsiasi forma d'onda, non necessariamente una sinusoide.
Questi differenti blocchi funzionali possono essere implementati sia con componenti hardware che software. La *tavola di lookup* diventa quindi un *array* calcolato in *runtime*. L'accumulatore di fase diventa una variabile a cui viene addizionato un valore, con controllo *if* per limitare il valore. Per la natura intrinseca dei DAC tuttavia è necessario l'uso di un filtro passa-basso che smussi il segnale in uscita per evitare la caratteristica "a gradini" tipicamente derivamente dall'uso di un DAC.

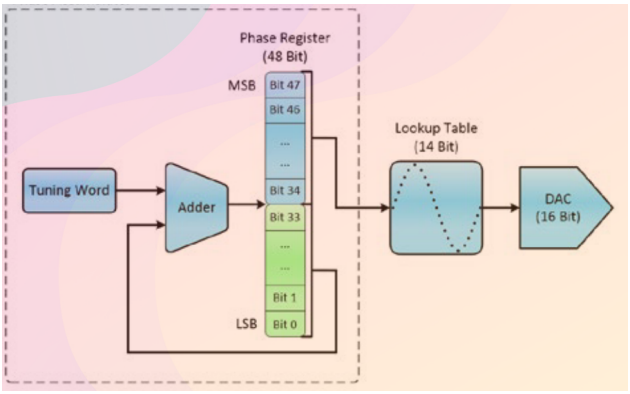


Diagramma dei blocchi funzionali di un generatore DDS. Immagine di National Instruments

Direct Digital Synthesis su Teensy 3.5

È stato deciso di programmare il Teensy 3.5 in linguaggio C++, con l'utilizzo del framework Arduino, mediante l'uso dell'IDE Visual Studio Code con plugin PlatformIO. Al contrario di un normale sketch Arduino, che usa una versione abbastanza semplificata del linguaggio C++, l'uso di VSCode e PlatformIO permette l'uso di un C++ standard a cui vengono aggiunte le funzionalità del framework Arduino.

Generazione di forme d'onda arbitrarie

L'implementazione della tecnica DDS è avvenuta traducendo i blocchi funzionali in codice, applicando alcune ottimizzazioni per ridurre l'uso di memoria e i tempi di calcolo.
All'avvio il microcontrollore calcola la *tavola di lookup* per diverse forme d'onda utilizzandone la formula analitica. Quando possibile, ne viene calcolata solo una parte (il primo quarto per i segnali sinusoidali) e il resto viene derivato per simmetria, riducendo anche i tempi di calcolo.
L'accumulatore di fase è implementata come una variabile di tipo float con precisione a 6 cifre decimali. Entrando in un ciclo infinito, viene preso l'indice della *tavola di lookup* corrispondente alla parte intera dell'*accumulatore di fase* e messo in uscita dal DAC. All'inizio del ciclo, l'*accumulatore di fase* viene incrementato di una quantità $\Delta\varphi$ pari a:

$$\Delta\phi = \frac{f}{f_S} G$$

Dove f è la frequenza che si vuole avere in uscita, f_S è la frequeza di campionamento del DAC che si sta utilizzando e G è la risoluzione (grandezza) della *tavola di lookup*. Nel caso il risultato superasse la grandezza della tavola di lookup, essa gli viene sottratta per evitare overflow.
La stessa tecnica può essere implementata per calcolare altre forme d'onda, come la triangolare, dente di sega o la quadra a duty cycle variabile, semplicemente cambiando la formula analitica usata. Il resto rimane invariato.
L'uso delle funzioni già presenti nel framework Arduino per comandare i DAC risultano avere un'esecuzione abbastanza lenta, richiedendo fino a più di 50 cicli di clock (540 nanosecondi), per questo sono state necessarie ottimizzazioni che velocizzassero l'esecuzione del codice.

Generazione di onde quadre a duty cycle variabile

Le onde quadre possono essere generate anche utilizzando un pin digitale del microcontrollore e cambiandolo di livello logico nei tempi giusti. A tale fine, è stato necessario abilitare il registro del Teensy che conta i cicli di clock passati dall'accensione del microcontrollore. Sapendo che ogni ciclo di clock dura approssivamente nove nanosecondi, è possibile calcolare ogni quanti cicli di clock è necessario cambiare lo stato del pin digitale. Per falo, è necessario prima sapere il periodo dell'onda in nanosecondi, calcolare la durata dei due livelli logici (sempre in nanosecondi) e successivamente convertire questa durata in cicli di clock.

È necessario eseguire questo calcoli una sola volta, prima della generazione del segnale, dopodiché è necessario impostare il pin digitale a livello alto e basso per le durate calcolate.

Svantaggi nell'uso dei DAC e filtri

Esistono vari tipi di DAC, ma, a prescindere da questo, è una loro caratteristica intrinseca fornire un'uscita "a gradini".
Derivando da un segnale digitale, un DAC può fornire solamente una quantità discreta e finita di valori in uscita. Ogni gradino è chiamato un "quanto", l'ampiezza di ogni quanto dipende dalla precisione in bit del DAC e dalla tensione di fondo scala utilizzata. Per mascherare la caratteristica visibile dei quanti, ai DAC vengono spesso fatti seguire dei filtri passa-basso, che hanno anche il compito di eliminare rumore e spurie dal segnale in uscita.

Per i generatori DDS si usano spesso filtri attivi di ordini superiori al secondo. Seguendo le fonti riportate in fondo al testo, è stato deciso di utilizzare un filtro Butterworth del sesto ordine per i segnali sinusoidali e le altre forme d'onda arbitrarie. I filtri Butterworth offrono contemporaneamente una caratteristica lineare per quasi tutta la banda passante ed buoni tempi di risposta.
Per le onde quadre era stato inizialmente deciso di utilizzare, per gli elevati tempi di risposta forniti, un filtro attivo Bessel del sesto ordine per eliminare le oscillazioni presenti sul transitorio del segnale. Tuttavia, dopo simulazione in ambiente MultiSim, risultava come esso ritardasse troppo la salita/discesa del transitorio, facendo diventare l'onda quadra simile alla caratteristica di carica e scarica di un condensatore. Il filtro Bessel è stato dunque sostituito da un semplice filtro RC passa-basso del primo ordine, che esegue perfettamente il lavoro. È inoltre predisposto nello schematico e sulla PCB un amplificatore operazionale usato come Trigger di Schmitt per filtrare ulteriormente il transitorio, ma al momento non è utilizzato.

Interfaccia utente con Atmega 32u4

L'intefacciamento del generatore con l'utente è affidata ad un microcontrollore secondario, un Atmega 32u4. Collegato al 32u4, un encoder incrementale con pulsante. L'interfaccia viene messa a video su un display LCD 16x2, che comuncia con il 32u4 attraverso il protocollo I2C.
La scelta dell'Atmega 32u4 deriva dalla mia precedente esperienza nel laboratorio di robotica con questo microcontrollore, e dall'esperienza in orario curriculare nel laboratorio di TPSEE.
Esistono svariate schede Arduino basate su Atmega 32u4, come il Micro o il Leonardo, il che permette di utilizzare agilmente il framework Arduino con questo microcontrollore.
Il display ha integrato una scheda di adattamento con un integrato che comunica in I2C con il 32u4 e gestisce il pannello. Questo facilita enormemente lo sbroglio del circuito a fronte di una modica spesa, dovendo collegare solamente quattro piste (due dati e due alimentazione) invece che le sedici normalmente richieste da un pannello LCD di questo tipo.
L'interfaccia grafica si fa un di un menù principale, nella cui riga superiore vengono alternativamente mostrati il tipo e il duty cycle scelto, oppure la frequenza, con un intervallo di due secondi.
Nella riga inferiore vengono mostrate la tensione di picco e la tensione di offset del segnale in uscita, ottenute tramite un circuito di condizionamento che riporta i segnali nel range [0V,5V] leggibili dagli ADC integrati del 32u4.
Premendo una volta il pulsante dell'encoder si accede al menu di *edit*. Viene prima chiesto di decidere il tipo di segnale (sinusoidale, quadro, triangolare…) e successivamente, se si è scelto un segnale quadro, è possibile deciderne il duty cycle.
Si passa successivamente all'impostazione della frequenza del segnale. Con una frequenza massima di 1MHz, è possibile modificare ognuna delle sette cifre del numero. Non è dunque necessario un frequenzimetro come nei generatori di segnali analogici, poiché la tecnica DDS permette di generare segnali che rispettino accuratamente la frequenza decisa a priori. Una volta decisa la frequenza del segnale si torna alla schermata principale. Le informazioni vengono mandate al Teensy, che si occuperà della generazione del segnale con le caratteristiche richieste.

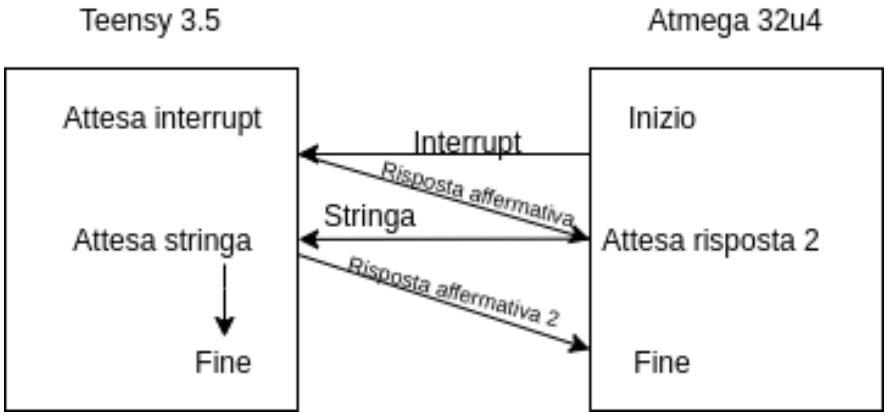
Protocollo di comunicazione tra i due microcontrollori

Ogni singolo ciclo di clock del Teensy è dedicato alla generazione del segnale. Questo significa che il Teensy non può permettersi di interrompere periodicamente questo compito per controllare se sono state ricevute informazioni dal 32u4.
Il protocollo di comunicazione è basato sull'invio di una stringa dal 32u4 al Teensy, che contiene tutte le informazioni impostate nel menu di *edit.*
La stringa ha un carattere alfabetico 'w' che segna l'inizio del messaggio e uno 'W' che segna la fine. Tra questi, ogni informazione numerica è inserita tra due caratteri alfabetici identificativi, secondo questa legenda:

Carattere inizio	Carattere fine	Contenuto	Possibili valori
w	W	Stringa con informazioni	Combinazione degli altri valori
t	T	Forma d'onda	0: sinusoide 1: quadra
f	F	Frequenza	[1, 1000000] [Hz]
d	D	Duty Cycle	[10, 90]

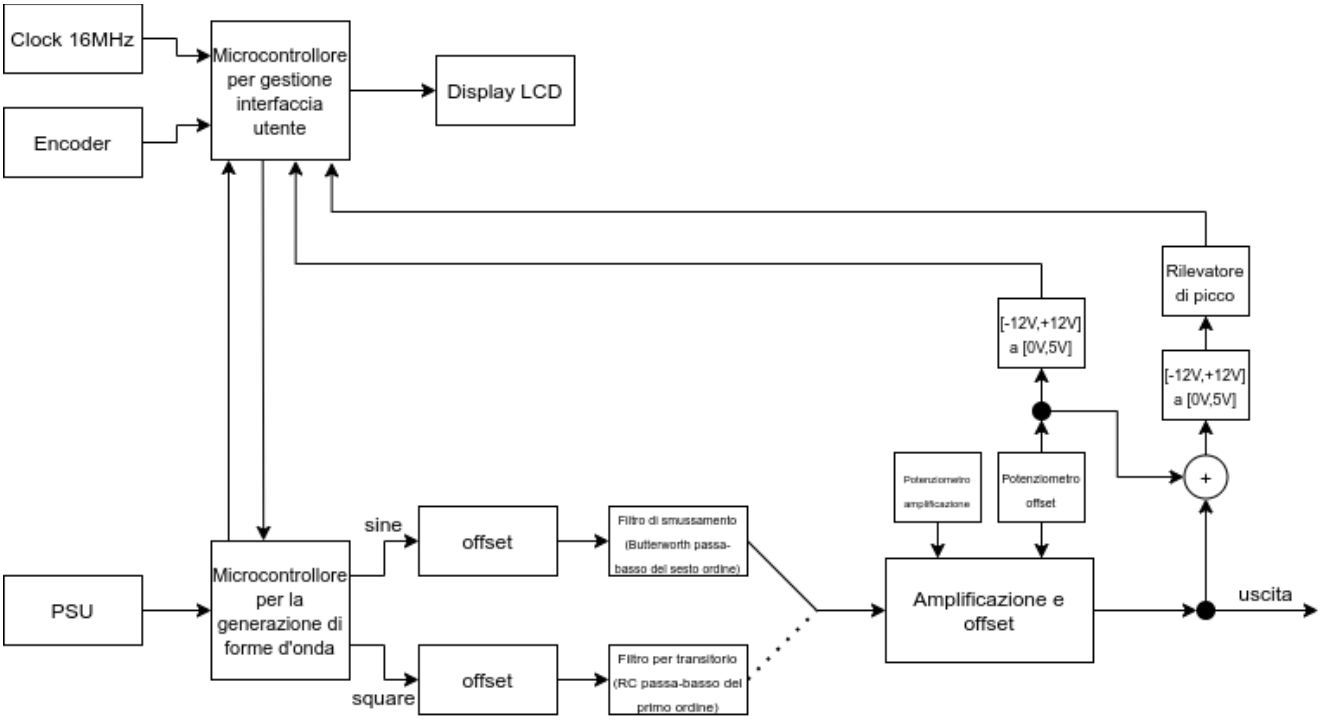
Ad esempio, la stringa *wt0Tf10000FW* indica un segnale sinusoidale di frequenza 10kHz. La stringa *wt1Td75Df14050FW* invece indica un segnale quadro di frequenza 14050Hz(14.05kHz) e duty cycle 75%.

Alla ricezione della stringa, il Teensy ottiene le informazioni necessarie tramite i metodi dell'oggetto String presente nel framework Arduino, ricavando le sottostringhe contenute tra due caratteri e convertendole in valori interi.
La comunicazione tra i due microcontrollori inizia dunque con l'invio di un interrupt al Teensy da parte del 32u4. Il 32u4 continua a generare questo interrupt fino alla ricezione di una risposta da parte del Teensy. Quest'ultimo, appena ricevuto l'interrupt, interrompe la generazione del segnale per mettersi in attesa dell'arrivo della stringa. Di nuovo, la stringa viene inviata ripetutamente a intervalli di tempo regolari fino all'arrivo di una risposta positiva da parte del Teensy. A questo punto la comunicazione viene interrotta e i buffer delle linee seriali di entrambi i microcontrollori vengono svuotati per evitare che il loro contenuto possa interferire con le comunicazioni successive.



Schemi

Schema a blocchi

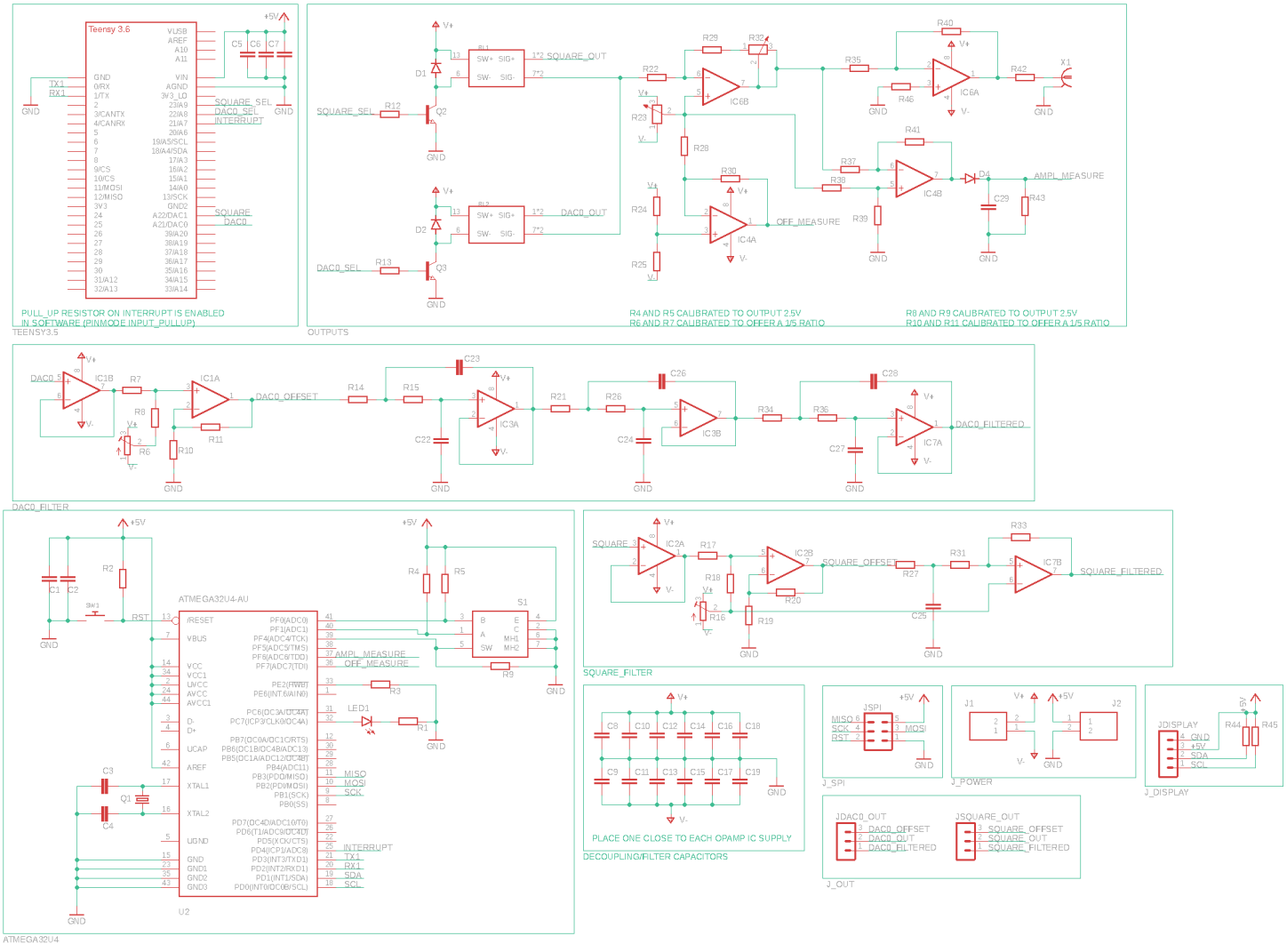


Legenda:

- PSU (Power Supply Unity: Alimentatore): circuito di alimentazioni con trasformatore lineare e regolati di tensioni lineari delle famiglia 78xx per le tensioni positive e 79xx per le tensioni negative (7805, 7812, 7912).
- Microcontrollore per la generazione di forme d'onda: Microcontrollore Teensy 3.5, ha due DAC a 12bit integrati con cui genere le forme d'onda
- Offset (2x): I DAC del Teensy 3.5 hanno un range di valori [0V,3.3V]. Usando un amplificatore operazionale in configurazione sommatore non invertente, viene azzerato l'offset del segnale
- Filtro di smussamento: Filtro Butterworth passa-basso del sesto ordine, scelto per la linearità nella banda passante. Viene usato per smussare il segnale in uscita dal DAC, che altrimenti apparrebbe a "gradini"
- Filtro del transitorio: Filtro RC passa-basso del primo ordine, utilizzato per eliminare picchi in salita o in discesa sul transitorio delle onde quadre
- Amplificazione e offset: Amplificatore operazionale non invertente, con offset
- Potenziometro amplificazione: Potenziometro sul ramo di reazione per modificare l'ampiezza del segnale in uscita
- Potenziometro offset: Potenziometro usato come partitore di tensione per modificare l'offset del segnale in uscita
- [-12V,+12V] a [0V, 5V] {1}: Amplificatore operazionale invertente con offset per riportare il valore dell'offset nel range [0V, 5V] perché possa essere letto dagli ADC integrati nell'Atmega 32u4

- Rilevatore di picco: Circuito diodo-condensatore, usato per trovare l'ampiezza del segnale in uscita. È preceduto da un amplificatore operazionale in configurazione amplificatore di differenze che somma il valore dell'offset al segnale in uscita, e scala il valore nel range [0V, 5V] (Blocco [-12V,+12V] a [0V, 5V] {2})
- Encoder: Encoder incrementale con pulsante con cui navigare l'interfaccia utente
- Clock 16MHz: Clock 16MHz per Atmega 32u4
- Microcontrollore per gestione interfaccia utente: Microcontrollore AtMega 32u4 usato per gestire l'interfaccia utente (decisione forma d'onda, duty cycle e frequenza) e comunicare le informazioni al Teensy 3.5
- Display LCD: Display LCD 16x2 che comunica con l'AtMega 32u4 con il protocolllo I2C

Schema elettrico



Condizionamento del segnale con amplificatori operazionali

Offset e amplificazione

Dopo lo stadio di generazione e filtro, l'utente può amplificare e applicare un offset un offset al segnale tramite due potenziometri. Per questa funzione è stato deciso di utilizzare un amplificatore operazionale in configurazione invertente, sul cui ramo di retroazione sono presenti una resistenza da 100Ω e un potenziometro da 10kΩ. La presenza della prima resistenza sul ramo di retroazione fa sì che il segnale non venga mai annullato, ma che possa essere anche attenuato e non solo amplificato. La presenza del potenziometro invece permette un amplificazione variabile.
L'offset a sua volta è implementato mediante l'uso di un potenziometro utilizzato come partitore di tensione tra [-12V,+12V], collegato all'ingresso non invertente dello stesso operazionale che si occupa dell'amplificazione. Alcuni integrati espongono un pin dedicato propriamente alla gestione dell'offset, ma gli AD8034 scelti per questo progetto non li posseggono. Dopo l'amplificazione, un amplificatore operazionale invertente a guadagno unitario annulla l'effetto di inversione introdotto durante l'amplificazione. Se quest'ultimo stadio non ci fosse, i segnali avrebbero duty cycle invertito. L'uso di amplificatori operazionali permette inoltre il disaccoppiamento delle impedenze, di fatto impedendo ai circuiti di misura o a qualsiasi carico attaccato al generatore di interferire con lo stadio di uscita.

Condizionamento del range di valori

Non essendo presente un controllo digitale dell'amplificazione dell'offset, ma essendo questi ultimi controllati tramite due potenziometri, deve essere inserito un modo per l'utente di sapere queste due caratteristiche del segnale. Queste caratteristiche vengono ottenute dal segnale stesso, sono lette tramite gli ADC integrati nel 32u4 e poi mostrate sul display LCD.
Il potenziometro che regola l'offset può assumere un valore che rientra in [-12V, +12V]. Questi valori sono fuori dalle tolleranze degli ADC del 32u4 e necessitano di essere condizionate in un range [0V, +5V] Per farlo è stato usato un amplificatore operazionale in configurazione invertente, che attenuasse il segnale per portarlo nel range [-2.5V, +2.5V], scegliendo dunque resistori che offrissero un rapporto di circa 1/5. Al terminale non invertente di questo operazionale è collegato un partitore di tensione, che offre una tensione di 2.5V in uscita, riportando di fatto il valore di offset nel range [0V, +5V].
Discorso diverso è invece ottenere l'ampiezza dell'onda. Dato che al segnale in uscita può essere applicato un offset, l'uso di un semplice rilevatore di picco non è sufficiente, poiché risentirebbe dell'effetto dell'offset. Per annullare quest'effetto, al segnale di uscita viene sottratto il valore dell'offset tramite un amplificatore operazionale in configurazione amplificatore di differenze, e al risultato viene applicato un rilevatore di picco. L'equazione di uscita di questa configurazione permette inoltre di applicare direttamente un fattore di scala al risultato della sottrazione. Scegliendo un fattore di scala di 5/12, il valore viene riportato nel range [0V, +5V].

Problemi attuali e possibilità di espansione

Allo stadio attuale, il generatore non possiede grandi problemi che ne prescindono il corretto funzionamento o utilizzo. Quello che potrebbe essere considerato un impedimento è la frequenza massima raggiungibile dal segnale generato con i DAC integrati del Teensy. Al momento, dopo una serie di ottimizzazioni lato software e overclock del microcontrollore, è di 1 MHz. Le quadre generate con i pin digitali del microcontrollore tuttavia riescono a toccare anche i 2 MHz. Il Teensy risulta dunque non essere esattamente lo strumento più adatto al lavoro richiesto, ed inoltre ha un costo abbastanza consistevole se si tiene conto delle "scarse" performance e

dell'ingrombro di spazio sulla PCB. Al fine di fare esperimenti e prove con altri microcontrollori, la PCB è stata predisposta, in fase di montaggio, con dei connettori strip che permettono il facile smontaggio del Teensy dalla scheda. Questo apre la possibilità alla progettazione di altre schede, che utilizzando i pin predisposti per il Teensy 3.5, adattino la PCB già esistente all'uso con altri microcontrollori con DAC interni o esterni. Sarebbe così possibile esplorare un abbassamento dei costi utilizzando il nuovissimo Raspberry Pi Pico con un DAC esterno a basso costo. Oppure, al fronte di un leggero incremento dei costi, sarebbe interessante provare l'implementazione di un DAC esterno come l'AD7542, che, stando al datasheet, promette tempi di risposta incredibilmente bassi e quindi una frequenza massima del segnale di uscita molto più elevata. Oppure cambiando totalmente famiglia di microcontrollori, sarebbe possibile sperimentare l'uso di microcontrollori della famiglia STM, basati su architettura ARM Cortex, che offrirebbe inoltre l'occasione didattica per l'utilizzo di microcontrollori diversi da quelli della famiglia Atmega, dato che richiedono un framework diverso da quello Arduino (molti STM usano il framework mbed, anche se supportano quello Arduino. Altri non supportano nessuno dei due e richiedono l'utilizzo diretto dei registri del microcontrollore).
Inoltre, per avvicinare il progetto più ad un prodotto finito che ad uno palesemente hobbystico ed autocostruito, si potrebbero utilizzare dei resistori comandati in tensione o dei resistori programmabili per avere un controllo totalmente digitale di amplificazione offset, senza bisogno di circuiti di feedback che ricavino le caratteristiche del segnale dal segnale stesso. Implementare l'uso di un display OLED invece che un LCD 16x2 inoltre permetterebbbe un'interfaccia grafica più complessa.
Dopo questi sviluppi, arrivati a una soluzione soddisfacente e che offra un buon rapporto qualità/prezzo sarebbe possibile iniziare a pensare ad una soluzione modulare, che implementi una singola base (con un microcontrollore che abbia lo stesso compito dell'Atmega 32u4 nel generatore attuale) a cui collegare altre schede che si occupano esclusivamente della generazione di segnali, aprendo la possibilità all'uso di diversi canali liberamente aggiungibili e removibili.
Nonostante io sia un grande ammiratore della filosofia open-source, in un futuro vorrei esplorare la possibilità di mettere in commercio questo progetto. Il progetto è attualmente disponibile open-source sul mio profilo GitHub sotto licenza GPL3, che permette l'uso di questi file a scopi di lucro, e sarebbe interessante mettere in commercio il progetto sotto forma di kit già montato o da montare, con o senza chassis o circuito di alimentazione. Questo lascia la possibilità ai capaci di usare liberamente i file, ai meno capaci di prendere comunque vantaggio del progetto ad un modico costo.

Sitografia

- Teoria della tecnica DDS:
  - https://www.analog.com/en/analog-dialogue/articles/all-about-direct-digital-synthesis.html
  - https://https://www.analog.com/media/en/training-seminars/design-handbooks/Technical-Tutorial-DDS/Section8.pdf
  - https://www.ni.com/it-it/innovations/white-papers/06/understanding-direct-digital-synthesis--dds-.html
  - https://en.wikipedia.org/wiki/Direct_digital_synthesis
  - http://lib.tkk.fi/Diss/2000/isbn9512253186/isbn9512253186.pdf

- Pratica della DDS:
  - https://zipcpu.com/dsp/2017/08/26/quarterwave.html
  - https://zipcpu.com/dsp/2017/07/11/simplest-sinewave-generator.html
  - https://stackoverflow.com/questions/13466623/how-to-look-up-sine-of-different-frequencies-from-a-fixed-sized-lookup-table
  - https://stackoverflow.com/questions/16889426/fm-synthesis-using-phase-accumulator

Tutto il progetto (i file CAD sia per il case che per la PCB, il codice e la documentazione, incluso questo elaborato) è disponibile open source sotto licenza GPLv3 sul mio profilo GitHub: https://github.com/EmaMaker/SignalGenerator-Teensy

Fotografie