POLITECNICO DI TORINO


Dipartimento di Elettronica e Telecomunicazioni

# Soluzione integrata intelligente per la gestione di un motore DC

Tesi di Laurea Magistrale in
Ingegneria Elettronica

Relatore:
Prof. M.R. Casu

Candidato:
Matteo Baldina

Marzo 2020

**Sommario**

La seguente tesi di laurea magistrale esamina un progetto realizzato presso l'azienda Bylogix s.r.l. di Gurgliasco (TO). Lo scopo è stato quello di progettare una scheda elettronica che si occupasse della gestione di un motore DC brushed tramite ponte ad H. La scheda è stata pensata per un ambito automotive e quindi utilizza componenti hardware non convenzionali. Inizialmente, è stato dedicato del tempo allo studio di una soluzione già presente all'interno dell'azienda e alla ricerca e comparazione di componenti presenti sul mercato. Inoltre, sono state fatte delle considerazioni sulla tipologia di soluzione da adottare come ad esempio l'utilizzo di mosfet discreti esterni oppure di chip che integrino il tutto, al fine di elaborare varie soluzioni valide su cui effettuare dei test pratici. Una volta selezionati i componenti, questi sono stati installati su alcune schede PCB, su cui sono stati effettuati i test. I risultati dei test hanno evidenziato un miglior comportamento di un componente che è stato quindi selezionato per ulteriori test. I test sono stati eseguiti con l'aggiunta di una scheda di valutazione dell'azienda NXP in modo da renderli automatizzati. Per garantire il corretto funzionamento della scheda di valutazione è stato richiesto lo sviluppo di driver a basso livello per alcuni moduli e un'applicazione per il controllo del motore. Il metodo di programmazione adottato permette di poter riutilizzare il codice scritto anche per la versione finale della scheda. In questo modo, dapprima si è studiato come implementare al meglio le funzionalità dei diversi moduli, come la sincronizzazione delle letture del modulo ADC con la forma d'onda PWM. Successivamente, tale approccio ha portato ad ottenere una struttura gerarchica che astrae il livello dell'applicazione da quello dei driver, al fine di aumentare la portabilità. In conclusione, è stata progettata la scheda seguendo le specifiche ottenute in seguito alla valutazione dei test. Sono state fatte delle analisi sui circuiti da utilizzare, come quelli adibiti alla protezione e alcuni studi hanno riguardato anche i componenti ausiliari ai chip principali. Inoltre, sono stati dimensionati i filtri per la riduzione dei disturbi elettromagnetici. In conclusione, una volta completati gli schemi elettrici è stato realizzato il layout della scheda, che presenta un stack a 4 layer.

**Abstract**

The following master's degree thesis deals with a project carried out at the Bylogix s.r.l. of Gurgliasco (TO). The aim was to design an electronic board that manages a DC brushed motor via an H-Bridge. The board was designed for an automotive business, therefore it makes use of unconventional hardware components. At the beginning, some time was dedicated to the study of a solution already present within the company and to the research and comparison between the components on the market. Then, considerations were made on the type of solution to be adopted such as the use of external discrete mosfets or chips that can integrate the whole, in order to have a range of solutions to be evaluated through practical tests. Once the components were chosen, they were installed on some PCB boards that were designed and that allow to carry out the tests. The results of these tests showed a better behavior of one particular component that was so chosen for further tests. The latter were performed with the addition of a NXP company evaluation board, in order to automate them. In order to guarantee the correct functioning of the evaluation board, the development of low-level drivers for some modules and an application for motor control were required. The programming method adopted permits to reuse the written code also for the final version of the board. This approach first led to the study of the best implementation of the functionality of the different modules, for example synchronizing the readings of the ADC module with the PWM waveform. Then, this method has brought a hierarchical structure that abstracts the application level from that of the drivers, in order to increase portability. Once the tests were completed, the board was designed following the specifications. Analyzes were made on the circuits to be used, such as those used for protection. Moreover, some studies involved auxiliary components to the main chips. In addition, filters have been sized to reduce electromagnetic disturbances. In conclusion, the schematic sheets were completed, the board layout was created, which features a 4-layer stack.

# Indice

# Elenco delle figure

# Listings

# Capitolo 1

# Introduzione

## 1.1 Scopo

Il lavoro di tesi è stato svolto presso l'azienda Bylogix s.r.l. di Gurgliasco (TO). Lo scopo di questa tesi è studiare una soluzione efficace per la movimentazione di un motore a corrente continua (*DC Motor*) di tipo *Brushed* e realizzare una scheda che possa essere integrata con questo tipo di motori, all'interno di un attuatore[1]. L'idea iniziale era quella di utilizzare un *FPGA* per la logica di controllo, assieme a dei componenti discreti per la parte di potenza, usati per fornire energia al motore. Inoltre, la componente *FPGA* avrebbe dovuto implementare interfacce di comunicazione, come ad esempio protocollo *CAN*, *SPI*. Dopo un accurato studio delle soluzioni presenti sul mercato, si è deciso di attuare una modifica all'idea iniziale, sostituendo l'*FPGA* con un microcontrollore per la logica di controllo.

## 1.2 Struttura della tesi

La prima fase ha riguardato lo studio di un precedente lavoro di tesi, effettuato da un altro candidato. Successivamente sono state definite alcune specifiche di progetto per la scheda *PCB* prototipale elencate di seguito:

- Ingombro massimo della scheda, 80 $mm$ di larghezza, 100 $mm$ di lunghezza;
- Supporto di una corrente di Picco/Stallo $I_{peak}$= 10 $A$ e funzionamento a regime con una corrente di $I_{nom} = 5$ $A$;

[1] Con attuatore si intende un sistema comprendente motore, ingranaggi e pistone o vite senza fine chiusi assieme.

- Funzionamento con un range di tensione da un massimo di $V_{MAX} = 18$ $V$ ed un minimo di $V_{min} = 9$ $V$. Inoltre sopportare una tensione di 24 $V$ senza essere soggetta a guasti;
- Compatibilità con la norma *ISO 7637-2*;
- Protezione per il collegamento inverso dell'alimentazione;
- Gestione di sensori di movimento.

Dopo aver deciso le specifiche, in accordo con l'azienda ospitante, si è passati ad uno studio delle varie soluzioni presenti sul mercato attuale, in quanto il precedente lavoro di tesi presentava dei componenti utilizzati per il controllo del *DC Motor* ormai in obsolescenza e quindi irreperibili sul mercato. Inoltre, si è lavorato con lo scopo di minimizzare gli ingombri dei componenti atti al controllo ed attuazione del *DC Motor*.
Infatti, nella ricerca dei nuovi componenti è stato tenuto conto sia delle esigenze dell'azienda, come reperibilità e costo dei componenti, ma anche delle richieste di progetto come l'ingombro in termini di area ed altezza dei componenti. La fase di ricerca ha scaturito una serie di candidati su cui effettuare dei test.

La seconda parte della tesi ha riguardato il test dei componenti selezionati assieme all'utilizzo di materiale di laboratorio, in particolare una scheda *PCB* di ridotte dimensioni ed una scheda di valutazione della *NXP* usata dall'azienda. Dapprima le schede *PCB* per i test sono state disegnate con un tool *CAD*, poi sono state stampate da una ditta esterna e successivamente si è provveduto alla fase di saldatura. Le schede sono servite per lo più ad alloggiare i componenti sotto test, quindi risultavano di dimensioni ridotte e con un *layout* molto semplice. Per quanto riguarda gli altri componenti passivi richiesti dai *datasheet* dei componenti sotto test, si è preferito usare una scheda millefori. Tale scelta è stata adottata per la semplicità di utilizzo e la velocità con cui è possibile apportare modifiche. Le due schede alla fine sono state unite attraverso degli *header* maschio-femmina.
Con l'ausilio di strumenti da laboratorio, si è provveduto a verificate le forme d'onda dei componenti. Dopo la fase di test, è stato scelto il componente da utilizzare e quindi da andare a testare in modo più approfondito.

La terza fase della tesi ha riguardato il test specifico del componente scelto tra quelli provati. Per rendere i test più automatici e testare le funzionalità, una parte del tempo di tesi è stato impiegato per la scrittura di *driver* a basso livello ed applicazioni *firmware* su una scheda di valutazione

della *NXP*. Questa è stata scelta in quanto utilizza lo stesso microcontrollore utilizzato nella scheda finale da realizzare. In questo modo, il codice scritto per la fase di test potrà essere riutilizzato nell'applicazione finale, andando a diminuire i tempi di progettazione.

Nella quarta fase della tesi, ovvero dopo aver deciso il componente che si occupa dell'attuazione del motore ed avere fatto i test, si è passati a dimensionare i componenti da utilizzare nella scheda *PCB* da realizzare. Alcuni di questi ultimi sono stati scelti in accordo con il magazzino presente nell'azienda. In questo modo i costi di produzione possono essere ridotti tramite l'acquisto di molte unità e la condivisione tra diversi progetti. Sono state dimensionate le protezioni richieste dal progetto, i filtri per le misure di tensione e per i disturbi introdotti dal *DC Motor*. I valori dei componenti passivi sono stati scelti in base a quelli di tabelle standard più vicino possibile ai valori calcolati.

Dopo aver dimensionato e scelto tutti i componenti da utilizzare si è provveduto attraverso un tool *CAD* a realizzare gli schematici, divisi per fogli, ed alla fine a realizzare il *layout* della scheda *PCB*. La scheda, oltre a rispettare le dimensioni massime, sarà realizzata a *4 layer* per due motivi principali: una miglior disposizione delle piste sulla superficie e una migliore conduzione del calore. La scheda sarà poi realizzata ed i componenti saldati all'esterno dell'azienda ospitante.

## 1.3 Studi preliminari

Come descritto precedentemente è stato necessario uno studio preliminare della soluzione già esistente per ricercare i punti critici e poterli ottimizzare. Il precedente lavoro di tesi utilizzava una scheda *PCB* accoppiata ad una scheda di valutazione della *Microchip* dal nome *$\mu$CAN*. Per quanto riguarda la scheda di valutazione usata, non si è riusciti a recuperare il software nè la documentazione necessaria, rendendo il lavoro svolto antecedentemente difficilmente replicabile.
Invece, per quanto riguarda la scheda progettata dal candidato, si sono ricavati i *part number* per la maggior parte dei componenti utilizzati andando ad ispezionare gli schematici lasciati. Dal numero di serie dei componenti, attraverso i *datasheet* trovati *on-line*, si è potuto risalire alla logica della scheda.
La maggior criticità riscontrata analizzando il progetto precedente è stata

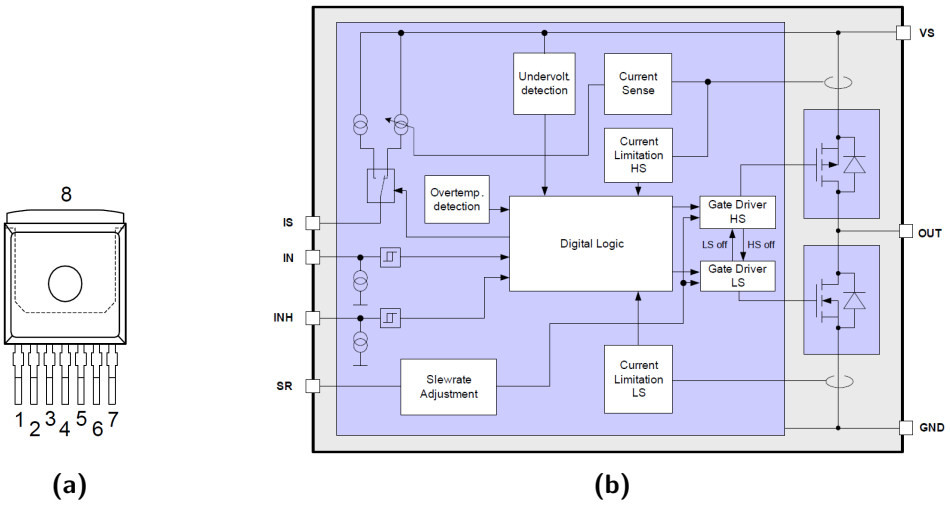



**Figura 1.1:** Package e diagramma a blocchi chip *BTN8982*.

quella di non poter replicare ed aggiornare la soluzione adottata in modo rapido ed efficiente. Questo ha portato all'azienda il compito di ripartire dalla fase di progettazione iniziale. Un'ulteriore criticità è stata quella relativa alla non facile reperibilità del materiale tecnico utile alla comprensione della soluzione adottata.
Come punto di partenza, in accordo con l'azienda, è stato quindi deciso di andare a studiare il componente usato per l'attuazione del motore. Il chip in questione contiene un *Half H-BRIDGE* e della logica per alcune funzionalità che saranno descritte in seguito. Per garantire il controllo su entrambi i sensi di rotazione del *DC Motor* sono stati usati 2 chip in parallelo. I componenti sotto esame sono prodotti dall'azienda *Infineon* ed il loro *part number* è *BTN8982TA*, in Fig. 1.1a è riportato il *packaging*, mentre in Fig. 1.1b è presente lo schema interno a macro blocchi.

Il componente si presenta con un *package* standard *TO-63-7-1* da 7 pin ed 1 pad (fig. 1.1a). Le funzionalità dei pin sono riportate in Tabella 1.1 e spiegate di seguito.
Per quanto riguarda i pin di alimentazione ed output non saranno fatti approfondimenti, mentre saranno trattati altri pin che si riferiscono ad funzionalità più interessanti. Si può notare un pin (INH) deputato all'enable del componente, il quale è collegato direttamente alla logica interna del chip. Si ha poi un pin (IN) usato come ingresso per un segnale, il quale segnale viene portato in ingresso al blocco logico andando a decidere quale *mosfet* del ramo si dovrà accendere. Troviamo poi un pin (SR) usato per limitare la pendenza del

**Tabella 1.1:** Descrizione pin componente *BTN8982*.

| PIN NUMBER | SYMBOL | FUCTION |
|---|---|---|
| 1 | GND | Ground |
| 2 | IN | Input |
| 3 | INH | Input Enable |
| 4,8 | OUT | Power output |
| 5 | SR | Slew rate |
| 6 | IS | Current sense |
| 7 | VS | Voltage supply |

fronte di accensione e spegnimento della coppia di *mosfet*. Infine, troviamo un pin (IS) usato fornire in uscita una corrente proporzionale alla corrente usata nel carico quando è acceso il *mosfet* superiore. Inoltre, questo pin può essere usato con funzionalità di diagnostica quali *Overtemperature detection*, *Undervoltage detection*, *Current limitation*.

Per quanto riguarda lo schema a blocchi interno (fig 1.1b), si può dedurre che questo chip sia molto avanzato, in quanto al suo interno non presenta solo i transistor usati per movimentare il *DC Motor* ma anche la logica che gestisce l'accensione, lo spegnimento ed anche sistemi di protezione e misura della corrente. In particolare, osservando il blocco logico (*Digital Logic*) si può notare che esso riceve in ingresso vari segnali tra cui l'enable (INH), un segnale (IN) di tipo *PWM* che determina quale dei due *mosfet* accendere e quale spegnere. Inoltre, gestisce in ingresso i segnali derivanti da due sonde che indicano se la corrente limite è stata raggiunta o dal *mosfet* superiore o da quello inferiore. Sempre in ingresso, troviamo sia le protezioni per la temperatura che per la tensione. Per quanto riguarda le uscite da questo blocchetto, gestisce i *Gate Driver* dei due *mosfet* in accordo con il segnale applicato sul pin IN. Nella gestione dei *driver* sono compresi anche dei ritardi tra l'accensione e lo spegnimento dei *mosfet*, così facendo si evitano i cortocircuiti tra l'alimentazione e la massa della scheda che potrebbero essere distruttivi. Inoltre, in uscita da questo blocco troviamo un segnale che è utilizzato internamente per far scambiare un selettore sul piedino IS.
Infatti, tramite questo pin, in condizioni normali è possibile avere in uscita una corrente proporzionale a quella del *drain* del *mosfet* superiore, mentre quando si ha una qualsiasi situazione di errore questo pin eroga una corrente nota di valore maggiore rispetto a quella massima misurabile. Il valore di

tale corrente è descritto nei dati del *datasheet* e permette di riconoscere un errore dal normale funzionamento.

Dopo aver studiato e assimilato il funzionamento della logica del componente, sono state analizzate le caratteristiche elettriche quali tensione di alimentazione, corrente nominale massima, frequenza massima del segnale d'ingresso ed altri parametri. Questo passaggio è risultato importante per capire quale sia stata la logica alla base del progetto precedente ed inoltre questi dati sono stati adottati come base di riferimento per la comparazione con altri componenti dello stesso genere.
L'azienda ospitante ha deciso di comprare alcuni di questi componenti e testarli. Il primo esperimento non è risultato molto efficace in quanto i chip in questione sono stati testati saldando dei fili sui pin e applicando i segnali con un generatore di segnale. È stato possibile provare la logica interna ma solo per delle correnti molto basse in quanto il chip, essendo in aria e non saldato su una scheda *PCB*, non aveva la giusta conduzione termica per dissipare il calore. Il problema derivante dall'usare basse correnti è stato quello di non riuscire a valutare il funzionamento del pin IS deputato alla misura della corrente. Alla fine dei test sul componente è stato deciso di valutare altre soluzioni presenti sul mercato.

## 1.4 Cenni teorici

Prima di iniziare la trattazione delle comparazioni tra varie soluzioni è utile fornire qualche accenno teorico relativo all'attuazione di un motore a corrente continua di tipo *brushed*. La soluzione più semplice è quella di usare un solo ramo con un solo *switch*. Il *DC Motor* viene posto prima o dopo allo *switch* creando rispettivamente le cosi dette configurazioni *low side* e *high side* riportate in Figura 1.2. Lo *switch* può essere di tipo meccanico usando ad esempio un relè, oppure di tipo elettronico usando un transistor (*BJT*, *mosfet*). Questa soluzione, sebbene sia la più economica e la più semplice dal punto di vista del controllo, non può andare bene nel caso sotto esame, perché il *DC Motor* con queste due configurazioni potrà ruotare in un solo verso.
In contraddizione alle richieste avanzate dall'azienda ad inizio lavoro di tesi, infatti si deve avere la possibilità di controllare entrambi i versi di rotazione del *DC Motor*. In aggiunta al controllo del verso di rotazione, è stato richiesto anche il controllo della velocità di rotazione. La soluzione più efficace è quella di utilizzare 2 rami contenenti ciascuno 2 *switch* uguali a quelli menzionati per la configurazione descritta precedentemente. Questa tipo di

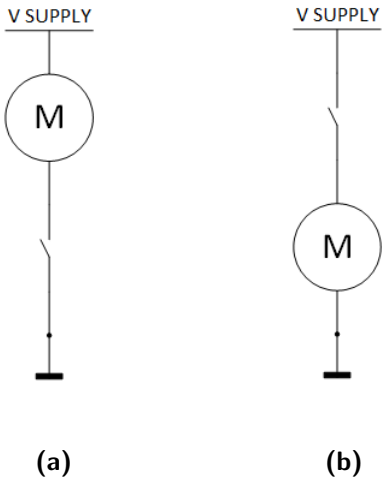



**Figura 1.2:** Configurazione *low side* e *high side* degli *switch*.

struttura viene definita *ponte ad H* e la si può osservare in Figura 1.3. Come si può notare, il *DC Motor* viene collegato nel mezzo dei due *switch* di ogni ramo e tramite la chiusura/accensione degli *switch* è possibile avere diverse combinazioni di cui le più significative sono riportate in Tabella 1.2.
Come si può osservare, alcune combinazioni sono proibite in quanto possono risultare dannose per i componenti, vedesi il caso di cortocircuito. Per ridurre o evitare questo tipo problema si dovrà lavorare dal punto di vista del controllo sul singolo *switch*.

Scelto il tipo di configurazione *ponte ad H* degli *switch*, è utile capire come questi vengano comandati. La soluzione più semplice è quella di applicare un segnale costante a livello logico alto o basso, dipendentemente dal tipo di *switch*, così da chiudere/aprire gli stessi. Si deve prestare attenzione ad attivare gli *switch* su rami diversi in quanto se i due fossero sullo stesso ramo si andrebbe incontro ad un cortocircuito. Questa tecnica non permette un reale controllo della velocità del motore quindi non soddisfa il requisito imposto dal progetto.
Una seconda possibile soluzione proposta è quella di usare dei segnali *Pulse With Module (PWM)*, in Figura 1.4, ovvero segnali periodici ad onda quadra dove è possibile cambiare l'ampiezza della parte positiva da un minimo dello 0% (segnale logico 0) ad un massimo del 100% (segnale logico 1) fra un periodo all'altro dell'onda. Anche in questo caso, abbiamo più di un segnale applicato e si dovrà sempre considerare il problema dei cortocircuiti.

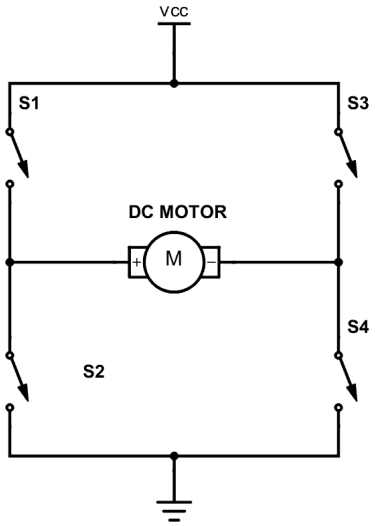



**Figura 1.3:** Configurazione *H-Bridge* degli *switch*.

**Tabella 1.2:** Configurazioni degli *switch* del *H-Bridge*.

| CONFIGURAZIONE SWITCH | | | | FUNZIONAMENTO |
|---|---|---|---|---|
| S1 | S2 | S3 | S4 | |
| Aperto | Aperto | Aperto | Aperto | Non definito |
| Chiuso | Chiuso | Chiuso | Chiuso | Cortocircuito |
| Chiuso | Aperto | Aperto | Chiuso | Rotazione senso orario |
| Aperto | Chiuso | Chiuso | Aperto | Rotazione senso antiorario |
| Chiuso | Aperto | Chiuso | Aperto | Freno verso *Vcc* |
| Aperto | Chiuso | Aperto | Chiuso | Freno verso *GND* |

Quindi, i contatti di uno *switch* a cui è applicato il PWM vengono chiusi/aperti (dipendentemente dal tipo di *switch*) per una frazione di tempo con una certa frequenza. Combinando due *switch* di rami diversi si possono creare dei percorsi per il passaggio della corrente e questo mette in rotazione il *DC Motor*. Agendo sul tempo di chiusura/apertura, cioè l'ampiezza della parte positiva all'interno del periodo, è possibile controllare la velocità di rotazione se il carico è costante, in quanto esiste una relazione tra la tensione[2] ai capi del *DC Motor* e la velocità. Quindi, sfruttando questo metodo è possibile soddisfare la seconda condizione imposta dai requisiti del progetto.
In aggiunta, esiste una tecnica mista dove viene applicato un segnale a livello logico costante per attivare uno *switch* di un ramo ed un segnale *PWM* per attivare con una certa frequenza uno *switch* di un altro ramo. Anche in questo caso, si riesce a soddisfare le richieste di progetto in quanto cambiando il tempo di attivazione del secondo *switch* si varia la tensione hai capi del *DC Motor*, di conseguenza si modula anche la velocità.
Come descritto precedentemente, usando le due tecniche appena proposte si deve evitare di far entrare in conduzione nello stesso periodo due *switch* sullo stesso ramo, per evitare che si crei un cortocircuito indesiderato fra la tensione di alimentazione e la massa. Per risolvere il problema si può inserire un piccolo tempo tra i segnali che comandano gli *switch* dello stesso ramo, che evita che gli *switch* siano attivati contemporaneamente. Tuttavia, questa tecnica non è facile da realizzare senza avvalersi di un qualche chip esterno che gestisca in ingresso i segnali *PWM* ed in uscita li riproponga con un piccolo tempo "morto".

## 1.5 Comparazione tra varie soluzioni

In questo paragrafo si intende giustificare la scelta del componente finale, andando ad evidenziare aspetti più pratici come l'ingombro in termini di area su *PCB* ed altezza, l'affidabilità ed altri parametri descritti in seguito.
Si possono distinguere due principali tipi di *switch*:

- Meccanici: relè;
- Elettronici: transistor *BJT*, *mosfet*.

Considerando gli *switch* meccanici, si è preso in esame l'uso di relè, ma tale soluzione è stata successivamente scartata. Infatti, sebbene i relè siano più

[2]Si fà riferimento a tensione media in quanto si ha una frequenza di attivazione e il segnale non è costante nel tempo.

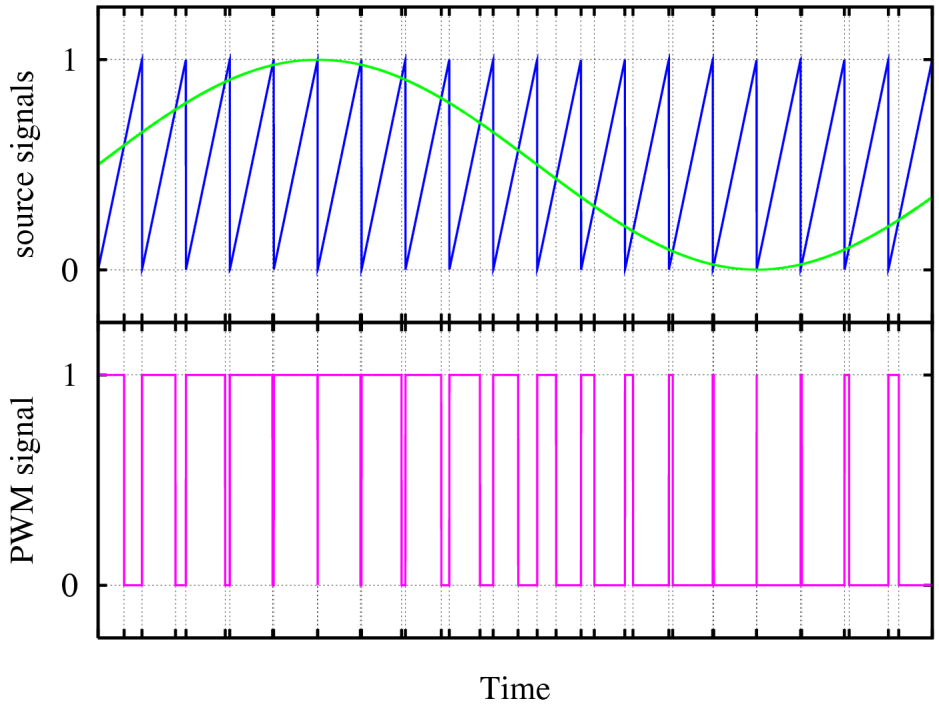



**Figura 1.4:** Esempio di segnale $PWM$

robusti rispetto al tipo elettronico e più immuni ai disturbi, presentano delle caratteristiche che non coincidono con il progetto in corso. Inoltre, questi presentano il problema di avere un limitato numero di commutazioni, quindi hanno un tempo di vita ridotto rispetto ad un componente elettronico. Un altro aspetto negativo è rappresentato dall'ingombro ed dal costo, infatti se trovare dei relè a basso costo per un montaggio su quadro elettrico è molto semplice, questo non è altrettanto vero per relè con montaggio su *PCB*. Questa categoria presenta un costo notevole e un ingombro verticale considerevole, quindi trovare dei relè compatti comporta molti compromessi. Considerando le caratteristiche del componente, questo ha un azionamento che fa muovere una parte meccanica, ma questa parte produce un rumore che non soddisfa gli standard dell'azienda. La scheda che si vuole progettare può avere diversi utilizzi, uno dei quali all'interno di un autoveicolo e tale rumore potrebbe comportare una riduzione del confort. Inoltre, un'altra considerazione è che si potrebbero creare degli archi elettrici tra i contatti interni ed la parte meccanica interno quando si usano delle correnti elevate. Questi portano ad un ulteriore riduzione del tempo di vita del componente. Introducendo gli *switch* di tipo elettronico, è stato dapprima fatto un confronto tra transistor di tecnologia $BJT$ e *mosfet*. Sono stati preferiti questi

ultimi perché si è visto che per basse produzioni, la tecnologia *mosfet* ha un costo inferiore ed inoltre risulta di più facile utilizzo.
Una volta scelta la tecnologia, si è passati a studiare l'applicazione, per la quale sono state proposte diverse soluzioni:

1) Quattro transistor *mosfet* distinti ed ognuno con il proprio segnale di controllo. In questo caso sarà necessario usare un componente che aiuti la gestione dei tempi morti per evitare dei cortocircuiti.

2) Un integrato contenente 4 *mosfet* con l'ausilio di un secondo chip per la gestione dei tempi morti.

3) Due integrati contenenti ciascuno due *mosfet* in modo da creare ognuno un mezzo *ponte ad H*. Gli integrati gestiscono i tempi morti autonomamente ed inoltre possono presentare più o meno funzionalità come misure di corrente, tensione.

4) Un integrato singolo con all'interno i quattro *mosfet* per creare il *ponte ad H*. Anche in questo caso la gestione dei tempi morti è lasciata all'integrato che può anche contenere delle funzionalità aggiuntive.

Le prima soluzione proposta utilizza dei componenti discreti ed è sicuramente la più adatta nel caso in cui si cerchino delle performance particolari. Potendo scegliere i *mosfet* si possono preferire delle caratteristiche particolari come una bassa capacità di *gate*, in modo che la frequenza del segnale $PWM$ possa essere aumentata, oppure una resistenza $R_{on}$ bassa in modo da ridurre la potenza dissipata o ancora *mosfet* che riescano a resistere al passaggio di alte correnti.
Nonostante ciò, questa configurazione tende ad occupare un'area su $PCB$ maggiore se il tipo di montaggio del *mosfet* è *Surface Mounting Device (SMD)*, perché si deve tenere conto sia dell'area del pad di ciascuno dei 4 componenti, sia dell'area per dissipare il calore prodotto da ogni chip quando in funzionamento. Nel caso invece si usino dei mosfet di tipo *through hole* questi occuperebbero teoricamente meno area su $PCB$ ma aumenterebbe l'altezza verticale. Inoltre si dovrebbe tenere conto anche di uno spazio per un eventuale dissipatore esterno, generalmente incollato sul lato posteriore del componente, per poter smaltire il calore in eccesso.
Per ridurre l'area su $PCB$ di quest'ultima tipologia descritta, avendo a disposizione dello spazio verticale da occupare, potrebbe essere utile condividere il dispositivo di dissipazione aggiuntivo o sfruttare il *case* dell'applicazione nel caso in cui esso dovesse essere metallico.
Il problema della dissipazione appena descritto è comune a tutte le soluzioni

proposte e deve essere tenuto in considerazione nelle valutazioni fatte. Per le soluzioni che usano componenti *SMD*, l'unico modo per dissipare il calore è quello di prevedere un'area maggiore al proprio pad su entrambe le facce della scheda ed utilizzare dei cosiddetti *thermal via* per facilitare la diffusione del calore da una faccia all'altra. Il rapporto area di dissipazione/prestazioni viene fornito come dato all'interno del *datasheet* sotto forma di °C/W per tutti i componenti.

Per quanto riguarda la seconda soluzione proposta, questa è una ottimizzazione della prima per quanto riguarda il fattore di area, un singolo chip contenente i 4 *mosfet* necessari occupa meno spazio di 4 *mosfet* distinti. Tuttavia vi sono alcuni svantaggi in termini di corrente massima, dissipazione termica, ma anche parametri generali come la carica di *gate* o la $R_{on}$ maggiori.
L'utilità di usare questa tipologia di componenti sta nel fatto che se la scheda *PCB* presenta una forma non standard o con delle dimensioni ridotte è possibile aver un miglior e più facile posizionamento fisico. Inoltre, se uniamo questo vantaggio ad usi in bassa corrente dove si ha poca dissipazione si presenta un ottimo trade-off tra area occupata e prestazioni.
La prima e seconda soluzione proposte presentano entrambe un fattore comune, infatti si deve prevedere un componente aggiuntivo chiamato *pre-driver*. Questi si occupano di gestire i segnali di comando da inviare ai *mosfet*, possono essere più o meno evoluti con la tecnologia. Nel caso più semplice, introducono un tempo morto fisso dipendente dalla frequenza di ingresso dei segnali *PWM*, parametro da considerare nel caso si voglia lavorare con dei periodi brevi. In casi più evoluti, si può avere la gestione di questi tempi andandolo a regolare con una resistenza esterna, che però comporta lo svantaggio di usare un componente aggiuntivo e, se di precisione, più costoso del normale. Un altro aspetto importante da tenere in considerazione è l'interfaccia per i segnali d'ingresso, infatti si possono trovare dei componenti che richiedono all'ingresso un solo segnale *PWM* oppure ne vengono richiesti più di uno o ancora alcuni *pre-driver* riescono a gestire dei segnagli di *enable*. Da tenere in considerazione poi il fattore occupazione area, poichè molte volte si trovano dei *pre-driver* con le dimensioni di un piccolo microcontrollore quindi non facili da posizionare. Anche il fattore costo incide, infatti si deve acquistare un componente aggiuntivo. Le considerazioni relative a costo, area ed unite al fatto di lavorare con delle correnti contenute hanno portato al scartare queste prime due soluzioni per il progetto di tesi.

Le restanti due soluzioni proposte presentano una filosofia quasi totalmente diversa da quelle presentate in precedenza. Vengono utilizzati dei *mosfet*

integrati all'interno di un chip come per la seconda soluzione proposta, ma non sono accessibili direttamente, cioè la gestione di questi viene fatta dal chip stesso. Quindi all'interno dell'integrato non solo troviamo presenti solo i quattro *mosfet* (solitamente tutti di tipo n) utilizzati per il *ponte ad H* ma anche tutta la logica per evitare cortocircuiti. Inoltre, questi chip si possono considerare intelligenti in quanto presentano altre funzioni aggiuntive come ad esempio quelle accennate in precedenza nel paragrafo 1.3 per il componente *BTN8982*. La terza e la quarta soluzione presentano molti punti in comune che possono essere descritti in un'unica trattazione. Si vuole quindi procedere in tal senso ed andare a distinguere le differenze solo alla conclusione.
Per quanta riguarda l'area occupata dai chip, sicuramente in questa tipologia è la minore rispetto alle precedenti, tuttavia si deve tenere in conto della dissipazione termica che per via di *package* degli integrati risulta complicata. Questo si può presentare in un formato non standard con la presenza di pin laterali usati sia per i segnali sia per la parte di potenza, in aggiunta a dei pad utilizzati sempre per la parte di potenza per dissipare il calore. Altri casi più semplici presentano dei *package* simili ad un semplice *case* come il *D2PAK* restando comunque non standard. Il vantaggio di occupare meno area a livello chip non è un vero e proprio vantaggio in quanto si deve tenere conto dello specifico *layout* da utilizzare, a volte molto complicato e questo incide anche sul grado di dissipazione del chip durante il funzionamento.
Questo tipo di soluzioni cosiddette "intelligenti" presentano tuttavia delle limitazioni sulla frequenza massima del segnale *PWM* in ingresso che può variare dai kilo Hertz alle decine di kilo Hertz. Queste limitazioni sono dovute alla logica interna che viene usata per gestire i *mosfet*. Tuttavia, per la maggior parte dei progetti queste frequenze di *switch* sono sufficienti e non presentano alcuna limitazione come nel caso di questo progetto.
Le correnti massime che si possono utilizzare per queste soluzioni sono inferiori rispetto a quelle che si potrebbero utilizzare se usassimo la prima soluzione ma si possono considerare comunque elevate in quanto con alcuni chip si riesce a raggiungere anche correnti di decine d'Ampere. Di conseguenza, queste due soluzioni si possono considerare per un impiego in progetti che utilizzano correnti medio-basse. Inoltre, dato l'utilizzo dei chip, questi prevedono al loro interno un blocchetto logico che gestisce la misura di corrente del *mosfet high side* (la maggior parte delle volte) e la rendono disponibile su un pin. Questo pin di *sensing* può fornire una corrente proporzionale a quella del carico, oppure può fornire una tensione sempre proporzionale. Tuttavia, la misura non risulta precisa come con l'utilizzo di un sensore apposito o una resistenza di *shunt* quindi potrebbe essere necessario aggiungere dell'hardware esterno come per le altre soluzioni. Inoltre, l'integrato è in grado di riconoscere il

superamento della soglia di corrente massima e andare a disabilitare l'uscita per protezione.
Utilizzando dei chip smart si riescono ad integrare delle funzioni aggiuntive che non si potrebbero avere con le prime due soluzioni a meno che non si aggiunga dell'hardware esterno. Una di queste è la protezione dalla tensione minima, infatti se la tensione scende al di sotto di una soglia il chip si spegne e spegne l'uscita andando a proteggere il carico non sovraccaricandolo in corrente. Oppure si può trovare la protezione opposta cioè quella per il superamento della tensione massima e anche in questo caso abbiamo un distaccamento dell'uscita volto a proteggere il carico ed il chip. Un'altra funzione aggiuntiva che potrebbe essere presente è quella per il monitoraggio della temperatura del chip. Questa è volta a proteggere l'integrato stesso, infatti quando si attiva il chip viene portato in una condizione di *idle* e viene segnalato l'errore attraverso un pin apposito. La particolarità di poter effettuare della diagnostica sugli errori e sul funzionamento senza l'aggiunta di hardware esterno rende questi chip molto pratici da utilizzare.
Le differenze tra la terza e la quarta soluzione sono minime. Nella terza soluzione si ha un integrato che comprende della logica e due *mosfet* che formano un *mezzo ponte ad H*, quindi si ha bisogno di utilizzare due integrati per questo progetto. Per la quarta soluzione si ha un integrato che presenta al suo interno l'intero *ponte ad H*. La differenza è presente anche a livello di *package*, infatti la terza soluzione ne presenta uno standard *TO263-7-1* mentre i chip scelti sotto la quarta soluzione presentano *package* non standard, proprietari dell'azienda che li produce.

In conclusione, in accordo con l'azienda, si è deciso di adottare come soluzione al problema posto usando una tra le ultime due soluzioni. Questo per diversi motivi, tra cui compattezza della soluzione e il fatto di riuscire a testare delle soluzioni innovative da poter portare sul mercato. Inoltre, avere delle funzionalità di diagnostica integrate può essere utile quando la scheda non è di facile accesso. I chip proposti per essere testi vengono riportati nell'elenco sottostante:

- Soluzione 3
  - Infineon BTN8982
- Soluzione 4
  - STMicroelectronics VNH5019
  - STMicroelectronics VNH5050
  - STMicroelectronics VNH7040

Gli integrati saranno descritti in modo più dettagliato nel capitolo successivo. Comunque, si può notare che nella quarta soluzione si ha una scelta maggiore rispetto alla terza soluzione.

## 1.6 Comparazione con FPGA

L'ultima parte del capitolo è utilizzata per spiegare il motivo dell'abbandono della soluzione con *FPGA*. Analizzando la prima e la seconda soluzione, si può notare che l'utilizzo di un *FPGA* può portare un grande vantaggio. Infatti con il suo utilizzo si possono realizzare tutte le componenti atte al controllo dei *mosfet*, quali gestione dei tempi morti, generazione dei segnali di controllo dei *gate* ed anche *enable* vari. Inoltre, si possono avere delle prestazioni a livello *timing* maggiori rispetto a dei chip *pre-driver* e si hanno maggiori possibilità di implementazioni di controlli particolari, di modifica ed adeguamento alle esigenze di produzione senza modificare la parte hardware.
Invece, con la scelta di adottare una tra la terza e la quarta soluzione si perdono i vantaggi descritti precedentemente, infatti nelle soluzioni proposte la logica di controllo è interna al chip e la funzione dell'*FPGA* sarebbe ridotta ad una sola generazione di forme d'onda. L'apporto portato da una logica complessa quindi non giustificherebbe il tempo di sviluppo delle componenti necessarie al progetto, nè il costo di implementazione rispetto all'utilizzo di un microcontrollore. I chip scelti adottano, come descritto in precedenza, logiche di protezione non disponibili se non con dell'hardware esterno in una soluzione con logica programmabile.

# Capitolo 2

# Test degli integrati

## 2.1 Specifiche dei chip

Prima di presentare i test svolti, si vogliono esporre le specifiche principali degli integrati utilizzati per l'attuazione del *DC Motor*.
Per quanto riguarda il componente *BTN8982*, scelto per la terza soluzione, si trovano le seguenti specifiche di lavoro:

- Tensione di alimentazione da 8 V a 18 V;
- Range della temperatura di lavoro della giunzione da -40 °C a +150 °C;
- Corrente massima 55 A con un opportuno layout;
- Frequenza massima *PWM* 17 kHz.

Mentre sempre lo stesso componente presenta queste protezioni e funzionalità:

- Protezione tensione minima;
- Protezione temperatura massima;
- Protezione corrente massima;
- Protezione contro cortocircuito;
- Regolazione del *rise time*;
- Misurazione di corrente e diagnostica.

**Tabella 2.1:** Comparazione tra integrati proposti nella soluzione numero 4

| Part Number | Tensione di lavoro V | Corrente di lavoro A | Temperatura di lavoro °C | Frequenza massima PWM kHz | Package |
|---|---|---|---|---|---|
| VNH5019 | 5,5 ÷ 24 | 30 | -40 ÷ +150 | 20 | MultiPowerSO-30 |
| VNH5050 | 5,5 ÷ 18 | 30 | -40 ÷ +150 | 20 | PowerSSO-36 TP |
| VNH7040 | 4 ÷ 28 | 35 | -40 ÷ +150 | 20 | PowerSSO-36 TP |

Per la quarta soluzione sono stati presentati più chip, quindi risulta utile andare a porre le specifiche nella tabella 2.1 confrontando i diversi integrati. Come si può notare non ci sono evidenti differenze tra i diversi chip, il *package* per il chip *VNH5019* cambia rispetto le altre due proposte, mentre per quanto riguarda le tensioni di alimentazione e le correnti massime sopportate, queste rientrano all'interno dei range richiesti dall'applicazione.
Per quanto riguarda le funzionalità "smart" di questi integrati, queste sono pressoché uguali per tutti e tre e risultano molto simili alle funzionalità descritte in precedenza per l'integrato *BTN8982*. Le differenze si notano nella modalità di accesso a queste funzionalità, infatti per i *chip VNH5019* e *VNH5050* si ha un solo piedino di *sensing* ed in base alla corrente erogata ed alla condizione di altri pin, che svolgono funzione anche di enable, si possono ricavare le informazioni desiderate come ad esempio capire quale protezione è intervenuta o misurare la corrente sta assorbendo il carico in condizioni di funzionamento. Questo principio non vale per il *chip VNH7040*, infatti presenta due *pin* chiamati *SEL0* e *SEL1* che indirizzano un *multiplexer* ed in base alla combinazione di essi e di altri *pin* è possibile ricavare informazioni riguardo alle diagnosi che si vogliono eseguire o che si stanno eseguendo. Oppure, si possono ricavare informazioni relative a quale grandezza si stia misurando. Tutte queste informazioni sono date come un *feedback* in corrente su un piedino specifico come negli altri casi. Questo *chip* si occupa anche di diagnosi e misure, non si limita quindi alla sola misura della corrente erogata, ma può effettuarne anche altre come la temperatura interna del integrato o la tensione a cui è alimentato. Si vogliono riportare in Figura 2.1 le combinazione dei *pin* quando il *chip* è nello stato attivo per avere le diverse misure o diagnosi, mentre in Figura 2.2 si vogliono riportate le combinazioni quando il *chip* è in stato inattivo. Si possono notare le varie operazioni eseguibili in questa fase se viene predisposto una certa configurazione del hardware descritta. In conclusione, questo integrato ha una maggior quantità di funzioni che, se integrate con la parte software, possono portare ad avere un sistema completo in tutto.

| INA | INB | PWM | SEL0 | SEL1 | MS |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 0 | High-Z |
| | | 1 | 1 | 0 | High-Z |
| 0 | 1 | 0 | 0 | 0 | Current Monitoring HSB |
| | | 1 | 0 | 0 | Current Monitoring HSB |
| 0 | 1 | 0 | 1 | 0 | High-Z |
| | | 1 | 1 | 0 | High-Z |
| 1 | 0 | 0 | 0 | 0 | High-Z |
| | | 1 | 0 | 0 | High-Z |
| 1 | 0 | 0 | 1 | 0 | Current Monitoring HSA |
| | | 1 | 1 | 0 | Current Monitoring HSA |
| 1 | 1 | X | 0 | 0 | Current Monitoring HSB |
| | | | 1 | 0 | Current Monitoring HSA |
| 0 | 0 | 0 | 1 | 0 | Off state diagnostic OUTA |
| 0 | 0 | 0 | 0 | 0 | Off state diagnostic OUTB |
| X(1) | X | X | 0 | 1 | Tchip Monitoring |
| X | X | X | 1 | 1 | Vcc Monitoring |
| X | X | X | X | X | High-Z(2) |

**Figura 2.1:** Tabella di verità *VNH7040* in stato operativo

| INA | INB | SEL0 | SEL1 | PWM | OUTA | OUTB | MultiSense | Description |
|---|---|---|---|---|---|---|---|---|
| **Off-state diagnostic** | | | | | | | | |
| 0 | 0 | 1 | 0 | 0 | $V_{OUTA} > V_{OL}$ | X | $V_{SENSEH}$ | **Case 1: $OUT_A$ shorted to $V_{CC}$** if no pull-up is applied.<br>**Case 2: NO open-load** in full bridge configuration with an external pull-up on OUTB<br>**Case 3: open-load** in half bridge configuration with an external pull-up on $OUT_A$ (motor connected between Out and Ground) |
| | | | | | $V_{OUTA} < V_{OL}$ | X | Hi-Z | **Case 1: open-load** in full Bridge configuration with an external pull-up on $OUT_B$<br>**Case 2: NO open-load** in half Bridge configuration with external pull-up on $OUT_A$ (motor connected between Out and Ground) |
| | | 0(1)(2) | 0(1)(2) | | X | $V_{OUTB} > V_{OL}$ | $V_{SENSEH}$ | **Case 1: $OUT_B$ shorted to $V_{CC}$** if no pull-up is applied<br>**Case 2: NO open-load** in full bridge configuration with external pull-up on $OUT_A$<br>**Case 3: open-load** in half bridge configuration with external pull-up on $OUT_B$ (motor connected between Out and Ground) |
| | | | | | X | $V_{OUTB} < V_{OL}$ | Hi-Z | **Case1: open-load** in full Bridge configuration with an external pull-up on $OUT_A$<br>**Case 2. NO open-load** in half Bridge configuration with external pull-up on $OUT_B$ (motor connected between Out and Ground) |

**Figura 2.2:** Tabella di verità *VNH7040* in stato inattivo

## 2.2 Impiego componenti scelti

Dopo aver descritto i componenti nel paragrafo precedente, si vuole trattare il metodo impiegato per poter utilizzare i componenti. Come scritto in precedenza gli integrati adottano la tecnologia *SMD*, quindi non possono essere utilizzati su una scheda di sviluppo "millefori" oppure su una scheda "breadboard". Inoltre, i chip presentano un *package* non standard quindi è stato necessario sviluppare quattro diverse schede *PCB*, di dimensioni adeguate, con i *footprint* esatti per ogni integrato e per poterli testare. Queste schede presentano ai lati degli *header*, cioè dei pin saldabili con passo 2,54 mm, che riportano i segnali presenti nei pin dell'integrato. Sono stati poi aggiunti 2 morsetti a vite con passo 2,54 mm. Il primo a due posizioni per poter collegare i cavi di alimentazione dell'integrato, mentre il secondo a 4 posizioni è usato per collegare i pad delle uscite dell'integrato con i cavi del carico elettronico o del motore e le GND corrispondenti dei due rami del ponte ad H. Sulle schede sono stati aggiunti anche due condensatori, uno elettrolitico da 1000 $\mu$F ed uno a poliestere da 100 nF, usati per stabilizzare la tensione di alimentazione in ingresso al *chip*. Si riporta nella Figura 2.3 le quattro schede realizzate, come visto nella Tabella 2.1 i *chip VNH5050* e *VNH7040* hanno lo stesso *packcage* ma presentano un differente *pinout* è stato necessario quindi realizzare due schede *PCB* differenti. Guardando la Figura 2.3 partendo dall'alto a sinistra si vede la scheda per il chip *VNH5050*, mentre alla sua destra è presente quella per i chip *BTN8982*. Nella seconda riga troviamo a sinistra la scheda per il chip *VNH7040* mentre nella destra quella per il chip *VNH5019*. Sotto alle schede PCB realizzare si può notare la scheda millefori realizzata per collegare i componenti aggiuntivi richiesti da datasheet ai pin del chip. Questi componenti comprendono anche una resistenza collegata tra il piedino di sensing e *GND* per trasformare il riferimento uscente dal chip in corrente in un riferimento in tensione più facile da misurare.

Per realizzare le schede PCB è stato utilizzato un tool apposito di progettazione. Per i 3 chip della quarta soluzione si è dovuto prima creare la libreria che ospita il componente, comprendente il simbolo da inserire nello schematico associato poi con il *footprint* da usare nel momento del layout, poi si è creata la scheda vera e propria, mentre la libreria contenente il chip *BTN8982* è stata reperita dal sito ufficiale del produttore. Per queste prove, la scheda PCB è composta da due layer *TOP* e *BOTTOM* di spessore 35$\mu$m separati da uno strato di materiale *FR4*.
Dopo la fase di progettazione, andando a seguire delle linee guida fornite all'interno dei *datasheet* dei componenti, le schede progettate sono state mandate in stampa da un'azienda esterna. Per semplicità e minor costo di stampa le 4 schede sviluppate sono state raggruppate e duplicate in un unico

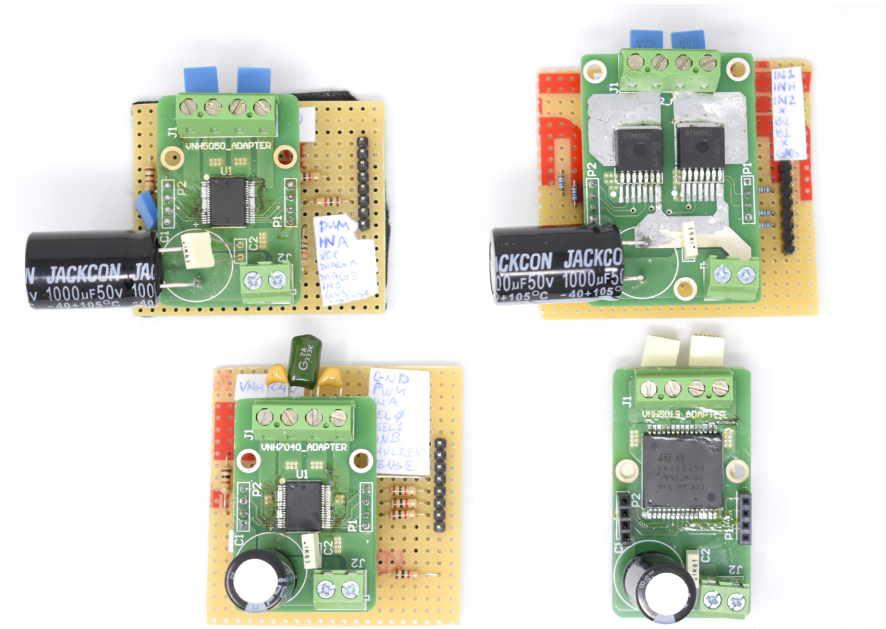



**Figura 2.3:** Schede PCB realizzate per adattare i *footprint* non standard dei chip

pannello di stampa delle dimensioni massime fornite dall'azienda produttrice di PCB. Durante il tempo di attesa è stata fatta una ricerca dei componenti necessari a completare l'attività su vari siti di distribuzione di componenti elettronici. Una volta arrivate le schede ed i componenti si è passati alla fase più pratica del progetto, ovvero la saldatura dei componenti ed i successivi test.

### 2.2.1 Schede aggiuntive

Dalla figura 2.3 si notano delle schede di prototipazione sottostanti alle schede PCB. Queste sono realizzate con un foglio detto "millefori". Servono per collegare dei componenti discreti agli header presenti su PCB, inoltre tramite delle saldature eseguite nella parte sottostante si sono riportati i segnali su un *header* maschio da una fila e 8 posizione per facilitare:

- l'uso delle schede PCB;
- l'invio di comandi;
- la lettura di feedback.

I valori delle resistenze e dei condensatori usati invece sono standard della serie E12 per facilitarne la reperibilità e sono di tipo through-hole. Nella

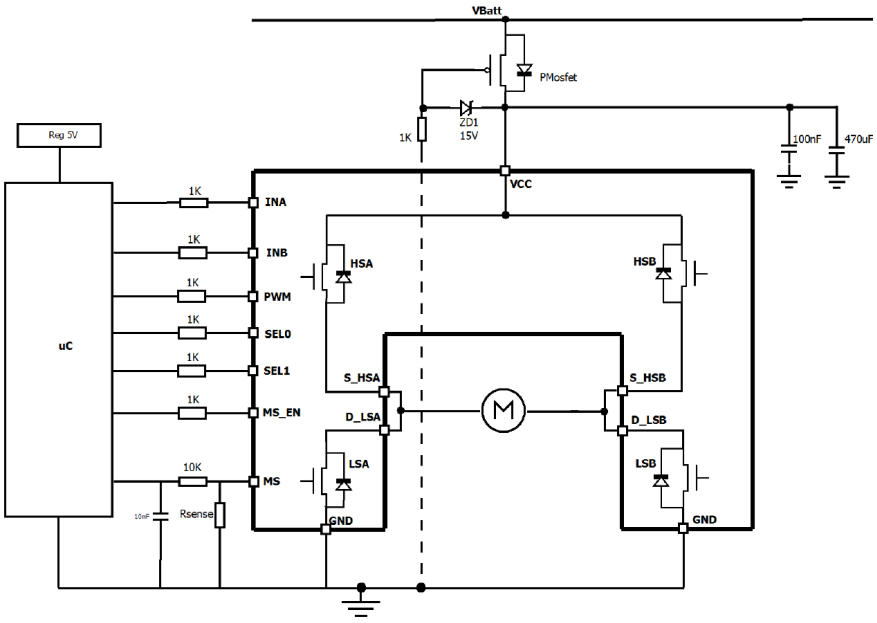



**Figura 2.4:** Schema estratto dal datasheet del componente VNH7040

figura 2.4 è riportato uno schematico di esempio estratto dal datasheet del componente *VNH7040* dove vengono indicati i valori delle resistenze e condensatori da usare. Per la precisione per quanto riguarda i segnali digitali di comando sono state usate delle resistenze da 1kΩ invece per la resistenza di sensing è stato usato il valore di 1,5 kΩ. La tensione di sensing viene poi filtrata con un filtro passa basso:

- frequenza di taglio $f_t = 100$kHz;
- valore resistenza filtro $R_f = 150\ \Omega$;
- valore condensatore filtro $C_f = 10$ nF.

Il filtro passa basso è presente per tutte le applicazioni da testare ed il valore della resistenza è stato ricavato a partire dal valore del condensatore imposto. Si è agito in questo modo in quanto nel laboratorio dell'azienda non erano disponibili tutti i valori della serie E12 ma solo alcuni. Per gli altri chip, gli schemi tipici sono simili per quanto riguarda il valore delle resistenze, sono però presenti delle resistenze di *pull-up* su alcuni segnali di enable dal valore di 3,3 kΩ.

La scelta di usare questo tipo di soluzione è derivata dalla facilità di poter cambiare valori alle resistenze e al condensatore di filtro, quindi per aver

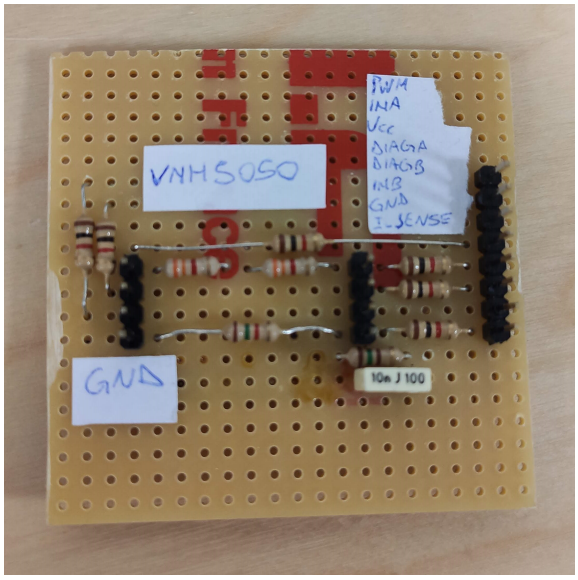



**Figura 2.5:** Scheda millefori per il chip VNH5050

maggiori possibilità di progettazione senza andare a mettere mano alla scheda PCB. Inoltre, in questo caso sono state usate solo 3 diverse schede realizzate con "millefori" al posto di 4, perché il chip *VNH5019* ed il chip *VNH5050* condividono gli stessi segnali con la stessa nomenclatura e funzionalità. In questo modo si riutilizza la stessa scheda per 2 integrati risparmiando il tempo impiegato per le saldature ed il costo dei componenti. In figura 2.5 è stata riportata in dettaglio la "millefori" usata per il chip *VNH5050*.

## 2.3 Prova dei componenti

Le prove effettuate sui componenti sono avvenute in due modalità differenti: la prima è descritta in questo paragrafo, la secondo modalità è descritta nel capitolo 4. Per le prime prove sui componenti in modo immediato sono stati utilizzati gli strumenti da laboratorio quali:

- Generatore di funzioni;
- Generatore di tensione;
- Oscilloscopio;
- Breadoard;
- Resistenza da 1 Ω e 100 W per simulare un carico.

Si è inizialmente provveduto ad eseguire i collegamenti tramite l'utilizzo di fili di colore rosso e nero con sezione 1 $mm^2$ per la parte di potenza. In questa fase le due uscite del ponte ad H presenti sul morsetto a quattro posizioni sono state collegate ognuna con una delle estremità della resistenza, mentre i due riferimenti di GND del ponte ad H, sempre dal morsetto a vite a 4 posizioni, sono stati collegati attraverso un filo nero di eguale sezione con il riferimento di GND dell'alimentazione utilizzando il morsetto a due posizioni presente nella scheda PCB. Con altri due fili con gli stessi colori usati in precedenza è stato collegato il generatore di tensione al morsetto a due posizioni, in questo modo è stata fornita l'alimentazione. Gli altri collegamenti, cioè quelli dall'header da una fila ed 8 posizioni sono state eseguite tramite l'uso di fili sottili che presentavano da una parte un header femmina connesso sulla scheda di sviluppo e dall'altra un pin maschio. Nella tabella 2.2 vengono riportati i collegamenti dei fili dall'header ad 8 posizioni della scheda di sviluppo dei chip *VNH5019* e *VNH5050*. I collegamenti per queste due schede sono uguali in quanto utilizzano lo stesso nome per i pin degli integrati ed inoltre condividono le stesse funzionalità, in questo modo è stata saldata solo una scheda di sviluppo. Nella tabella 2.3 e 2.4 troviamo rispettivamente i collegamenti per la scheda di sviluppo del chip *VNH7040* e *BTN8982*.

**Tabella 2.2:** Collegamenti pin header scheda di sviluppo VNH5019/VNH5050

| Integrato | Pin Number | Pin Name | Collegamento |
|---|---|---|---|
| | 1 | PWM IN | Gen. Funzioni |
| | 2 | INA | 1/0 Logico |
| | 3 | $V_{cc}$ | $V_{cc}$ |
| VNH5019 | 4 | Diag A | Sonda Osclilloscopio |
| VNH5050 | 5 | Diag B | Sonda Osclilloscopio |
| | 6 | INB | 1/0 Logico |
| | 7 | GND | GND |
| | 8 | ISense | Sonda Osclilloscopio |

In tutte le tabelle presentate sono presenti dei segnali logici che possono essere posti ad 1 logico collegandoli alla tensione logica, oppure a 0 logico collegandoli a GND. Questi sono indicati nelle celle come 1/0 Logico e servono per i chip *VNH5019* e *VNH5050* per selezionare l'abilitazione o il verso della corrente delle uscite. Per quanto riguarda i pin *DIAGA* e *DIAGB*, questi svolgono la funzione segnalazione di un problema rispettivamente su i due

**Tabella 2.3:** Collegamenti pin header scheda di sviluppo VNH7040

| Integrato | Pin Number | Pin Name | Collegamento |
|---|---|---|---|
| | 1 | GND | GND |
| | 2 | PWM IN | Gen. Funzioni |
| | 3 | INA | 1/0 Logico |
| VNH7040 | 4 | SEL0 | 1/0 Logico |
| | 5 | SEL1 | 1/0 Logico |
| | 6 | INB | 1/0 Logico |
| | 7 | MultiSense Enable | 1 Logico |
| | 8 | ISense | Sonda Osclilloscopio |

**Tabella 2.4:** Collegamenti pin header scheda di sviluppo BTN8982

| Integrato | Pin Number | Pin Name | Collegamento |
|---|---|---|---|
| | 1 | IN1 | 1/0 Logico |
| | 2 | INH | 1/0 Logico |
| | 3 | IN2 | 1/0 Logico |
| BTN8982 | 4 | NC | Non connesso |
| | 5 | IS1 | Sonda Osclilloscopio |
| | 6 | IS2 | Sonda Osclilloscopio |
| | 7 | NC | Non connesso |
| | 8 | GND | GND |

rami del ponte ad H, tramite una resistenza da 10 kΩ sono collegati a $V_{cc}$ per formare un pull-up. Questi pin vengo forzati a zero logico da parte del chip quando si attiva una protezione. Il chip *VNH7040* presenta più pin logici in quanto le funzionalità sono maggiori rispetto ai precedenti descritti. In questo caso, oltre a scegliere la direzione e l'abilitazione delle uscite con i segnali *INA* e *INB* possiamo scegliere anche che diagnostica eseguire sul chip, questo grazie ai segnali *SEL0* e *SEL1*. Inoltre, dobbiamo abilitare anche l'enable per l'usicta *ISense* collegando il pin *MultiSense Enable* ad 1 logico. Anche i chip *BTN8982* per funzionare necessitano di un enable, che è lo stesso per tutti e due ed indicato come *INH*. Presenta poi i due pin che indicano la direzione *IN1*, *IN2* e due pin di sensing uno per ogni chip chiamati *IS1* e *IS2*.

Una volta terminati i collegamenti, si è passati a configurare gli strumenti da laboratorio, per il generatore di funzione si è partiti con un'onda quadra con dulty cycle 50% e frequenza 1 kHz. Lo stesso segnale è stato collegato ad un canale dell'oscilloscopio attraverso un cavo coassiale per fornire una forma d'onda priva di disturbi su cui eseguire l'operazione di trigger. Sui tre canali rimanenti dell'oscilloscopio sono stati collegati la tensione di alimentazione del chip, una delle due uscite scelta in base alla configurazione dei segnali di direzione e sull'ultimo canale disponibile è stata collegato il segnale di sensing. Queste prove sono servite per controllare l'effettivo funzionamento del chip, cioè verificare se il layout realizzato fosse corretto ed anche se il chip fosse saldato nel modo corretto. Per provare le varie combinazioni di funzionamento si è provveduto a spostare manualmente i segnali *INA* ed *INB* collegandoli a 5V oppure GND. Verificato che il funzionamento dei chip era corretto, si è passati a sostituire la resistenza di in uscita con un motore Brushed DC fornito dell'azienda. Non è stato possibile ricavare tutti i dati del motore ma solo la corrente massima di 10 A di picco e tensione di alimentazione $V_{alim}$ di 12 V nominale. In figura 2.6 si possono vedere immagini acquisite con l'oscilloscopio durante l'utilizzo del chip *BTN8982*. I canali dell'oscilloscopio sono utilizzati nel seguente modo:

- canale 1 (blu): segnale di PWM;
- canale 2 (rosso): segnale dell'usita;
- canale 3 (verde): segnale di sensing del piedino *IS*;
- canale 4 (arancione): segnale di diagnostica relativo all'uscita misurata.

Come si nota dalla figura, durante l'utilizzo di questo chip con il motore sono sorte delle complicazioni. La prima dovuta all'utilizzo del motore senza un adeguato carico meccanico. Questo ha comportato ad avere una corrente utilizzata dal motore bassa che ha reso difficile la lettura da parte del sensore interno presente nel chip, con conseguenza una difficoltà nella lettura su oscilloscopio del segnale di sensing. Una seconda difficoltà è derivata dalle capacità ed induttanze parassite introdotte con i collegamenti, infatti il segnale di sensing risultava molto disturbato con presenza di picchi ed oscillazioni. Per risolvere il problema dei disturbi si è pensato di modificare il filtro passa basso sul segnale di sensing ed inoltre aggiungere dei condensatori sulle uscite collegati tra uscita $V_{cc}$, uscita GND ed uscita-uscita. Invece per risolvere il problema della bassa corrente si è cercato di aggiungere un carico meccanico al motore in modo da poterlo testare in una condizione più simile alla realtà. Come si può vedere in figura 2.7 con queste accortezze

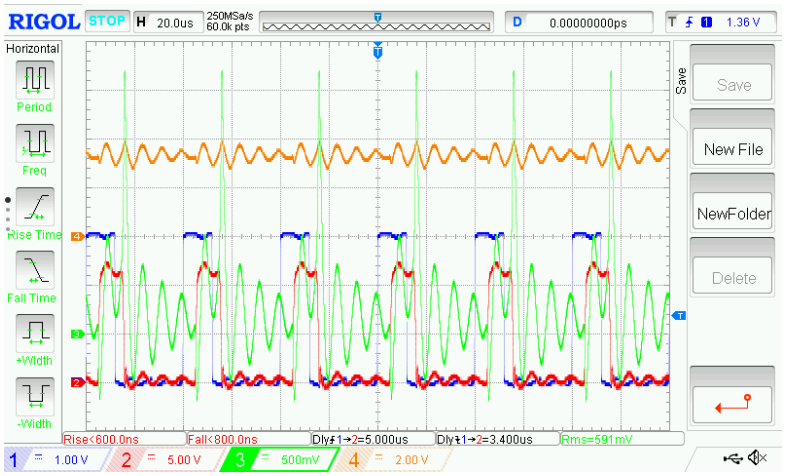



**Figura 2.6:** Prova chip BT8982 motore senza carico meccanico

il segnale risulta molto più leggibile anche se presenta ancora dei picchi al momento dello spegnimento dell'uscita.

Sono stati provati anche altri motori con lo stesso chip ed i risultati sono stati i medesimi quindi si è pensato di andare a provare gli altri integrati per capirne il funzionamento. Si può anticipare che i chip *VNH5019* e *VNH5050* hanno avuto lo stesso comportamento in quanto molto simili tra loro e la misura interna della corrente e la segnalazione di errori attraverso la diagnostica è la medesima. In queste prove la frequenza è stata abbassata a 1 kHz anzichè 20 kHz e il risultato si può vedere in figura 2.8. Si nota che è presente un problema simile alle precedenti prove cioè quello di avere una corrente non elevata,ma si può affermare che questi chip presentano meno sensibilità hai disturbi rispetto al chip *BTN8982* presentato in precedenza. La configurazione dell'oscilloscopio in queste prove è la seguente:

- canale 1 (arancione): segnale di sensing del piedino *ISense*;
- canale 2 (verde): segnale dell'usita;
- canale 3 (blu): tensione di alimentazione $V_{cc}$;
- canale 4 (viola): segnale sonda di corrente posta su un'uscita del ponte.

Per quanto riguarda invece il chip *VNH7040* la prova è riportata in figura 2.9 ed è effettuata ad una frequenza di PWM di 2 kHz e dulty cycle del 50%.

Per quanto riguarda i canali dell'oscilloscopio si ha la seguente configurazione:

- canale 1 (blu): Segnale del PWM in ingresso al chip;

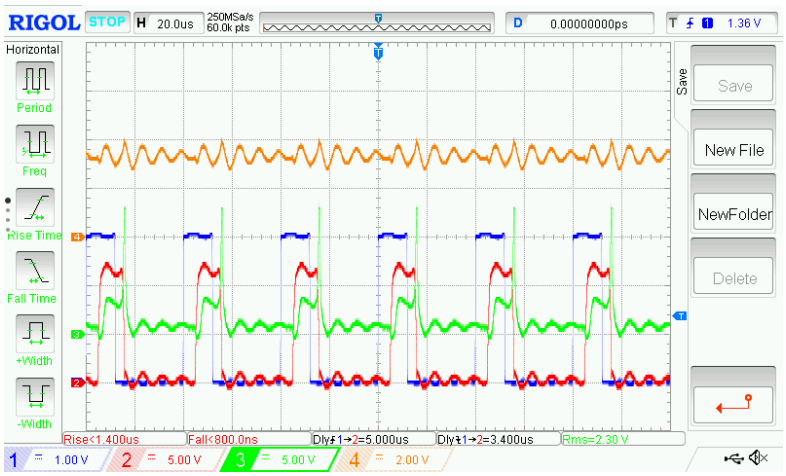



**Figura 2.7:** Prova chip BT8982 con ottimizzazioni

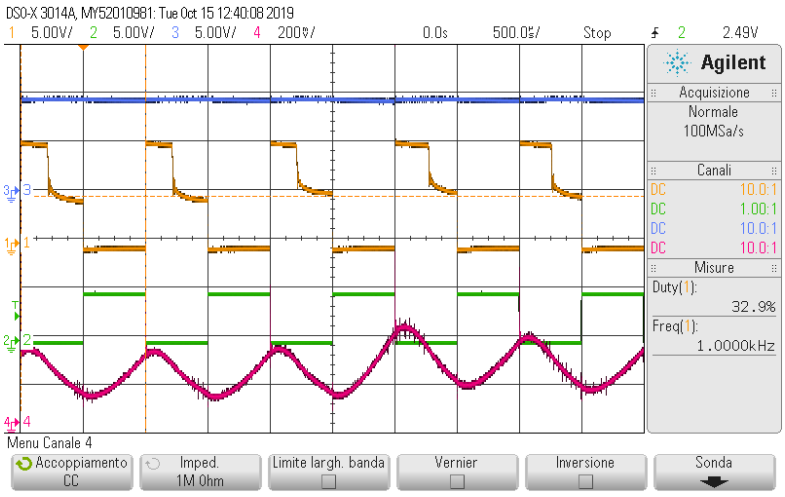



**Figura 2.8:** Prova chip VNH5019 con ottimizzazioni

- canale 2 (rosso): segnale del piedino multisense;
- canale 3 (verde): tensione presente su un uscita del ponte ad H;
- canale 4 (arancio): tensione di alimentazione $V_{cc}$.

Dalla figura 2.9 si nota come questo chip sotto esame abbia la miglior risposta per quanto riguarda il segnale di sensing. Infatti, presenta un picco quando si ha lo switch dei mosfet interni sui quali viene misurata la corrente, per poi scendere fino ad un valore stabile e costante. Si può notare inoltre che quando i mosfet dove viene misurata la corrente, cioè nella parte positiva del periodo dell'onda presente sul canale 3 dell'oscilloscopio la corrente decresca fino allo zero.

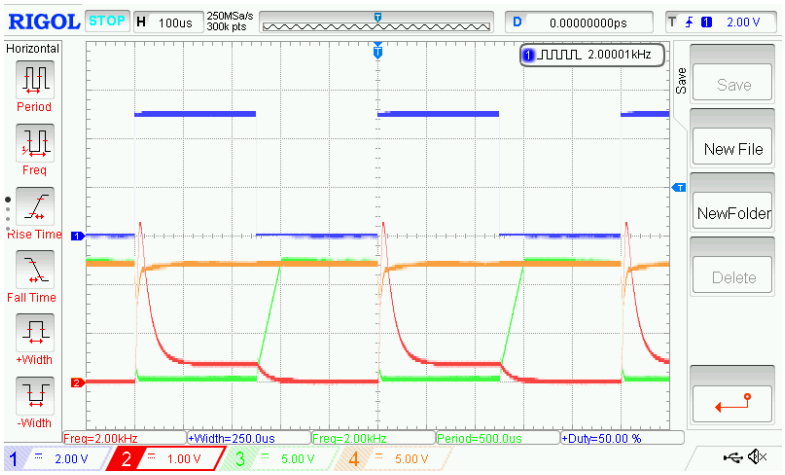



**Figura 2.9:** Prova chip VNH7040 con ottimizzazioni

Dopo queste prove effettuate manualmente si è deciso di concentrare gli studi sul funzionamento del solo chip *VNH7040* visto i risultati ottenuti. Per facilitare queste prove ed iniziare a creare un'applicazione vera e propria si è pensato di passare ad un sistema che usasse una scheda dotata di microprocessore ed in grado di comandare il componente attraverso la decodifica di messaggi inviati tramite rete *CAN*[1]. Nel capitolo successivo verrà descritto nel dettaglio il modo in cui è stato implementato il sistema.

[1]La rete CAN è un protocollo di comunicazione largamente utilizzato nell'ambito automotive.

# Capitolo 3

# S32K116EVB e driver

## 3.1 Evaluation board

Per lo svolgimento di questa parte di test, oltre a fare uso del componente *VNH7040* riportato in figura 3.1 assieme alla sua basetta millefori, è stata utilizzata una scheda di prototipazione sviluppata dall'azienda *NXP* la cui sigla commerciale è *S32K116EVB* riportata in figura 3.2. Questa scheda consente di testare ed eseguire il debug del programma scritto collegandosi all'ambiente di sviluppo *S32 Desing Studio*, concesso in licenza gratuita dalla *NXP*. Di seguito si riportano in sintesi alcune delle specifiche rilevanti della scheda:

- microcontrollore 32-bit ARM®-M0 S32K116;
- System Basis Chip (*SBC*);
- potenziometro;
- LED RGB;
- 2 pulsanti;
- interfacce CAN, LIN, UART/SCI;
- possibilità di fornire alimentazione tramite USB o 12 V esterni;
- OpenSDA per il debug;

La scelta di questa scheda è stata giustificata dal fatto che fosse già presente all'interno del panorama dell'azienda per lo sviluppo di altri progetti. Infatti, alcuni driver di basso livello erano già stati sviluppati per lo stesso microcontrollore come ad esempio quello per la comunicazione *CAN*, il settaggio dei

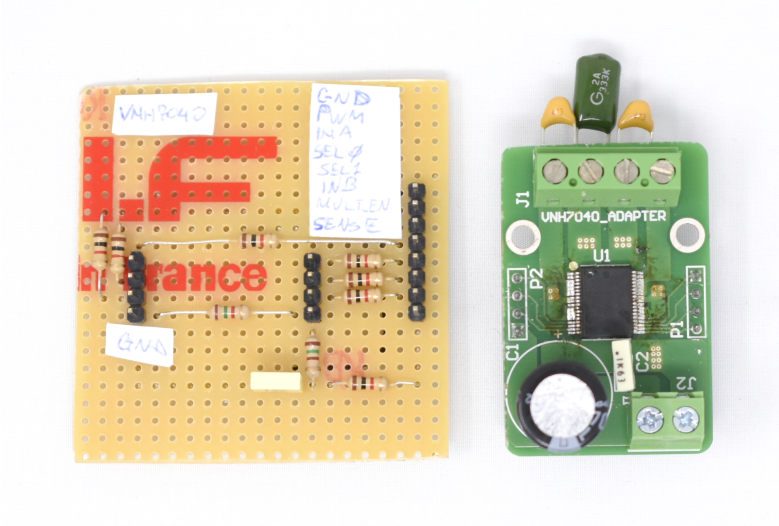



**Figura 3.1:** Scheda VNH7040 usata con ottimizzazioni

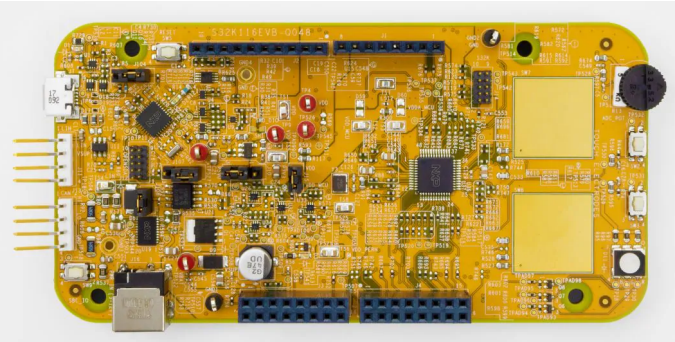


**Figura 3.2:** NXP S32K116EVB

registri interni per il corretto funzionamento del clock ed infine erano stati definiti parte delle funzionalità per ogni pin del microcontrollore. Ulteriori motivazioni sono il fatto di aver installato un microcontrollore certificato automotive, il basso costo, come accennato in precedenza, l'ambiente di sviluppo fornito con licenza gratuita ed infine il fatto di poter riutilizzare alcuni componenti anche nella scheda da progettare come ad esempio il componente *SBC*. Quest' ultimo componente integra delle funzioni utili come la funzione di 'wake-up' utile per i consumi, la parte per la gestione della rete *CAN* ed altre discusse nel successivo capitolo. Inoltre, la scheda *S32k116EVB* presenta dei connettori *header* femmina che permettono di collegare delle schede di espansione nel caso di bisogno oppure, come in questo caso, attraverso dei fili sottili si possono collegare le due schede. Per le prime fasi di sviluppo è

stato fatto uso di componenti installati sulla board, come ad esempio il LED RGB per quanto riguarda il corretto funzionamento dei timer programmati, il potenziometro per modificare con velocità il valore di tensione all'ingresso del canale dell'ADC ed i pulsanti per eseguire delle diverse azioni quando premuti.

## S322k116

In questo sotto paragrafo si vuole introdurre il microcontrollore *S32k116* usato, il quale presenta le seguenti caratteristiche principali:

- frequenza high Speed 48 Mhz;
- numero di pin 48 package LQFP;
- 2 moduli Timer configurabili da 8 canali ciascuno;
- 1 Programable Delay Block (PDB);
- 1 Trigger muc (TRGMUX);
- 1 modulo ADC ad approssimazioni successive (ADC SAR) da 12 bit con 13 canali d'ingresso;
- 1 modulo Low Power SPI (LPSPI);
- 1 modulo FlexCAN compatibile con CAN-FD[1];
- JTAG per DEBUG.

In figura 3.3 viene riportato lo schema dell'architettura ad alto livello del microcontrollore. In particolare, i blocchi in arancione indicano le periferiche presenti su ogni microcontrollore della famiglia *S32K*, quelli in grigio scuro invece sono i blocchi dell'architettura del microcontrollore ed infine quelli in grigio sono i blocchi opzionali che possono differire da modello all'altro. Un'altra precisazione riguarda il clock, infatti all'interno sono presenti 3 differenti regimi riportati in elenco:

- Fast Internal Clock Reference (FIRC) per lavorare a frequenza di core 48 MHz;
- Slow Internal Clock Reference (SIRC) per lavorare a frequenza di core 8 MHz;

[1]Nuovo standard per la comunicazione CAN in grado di aumentare il Baud Rate sul BUS dati fino a 12 MBit/s da norma ISO1898-2.

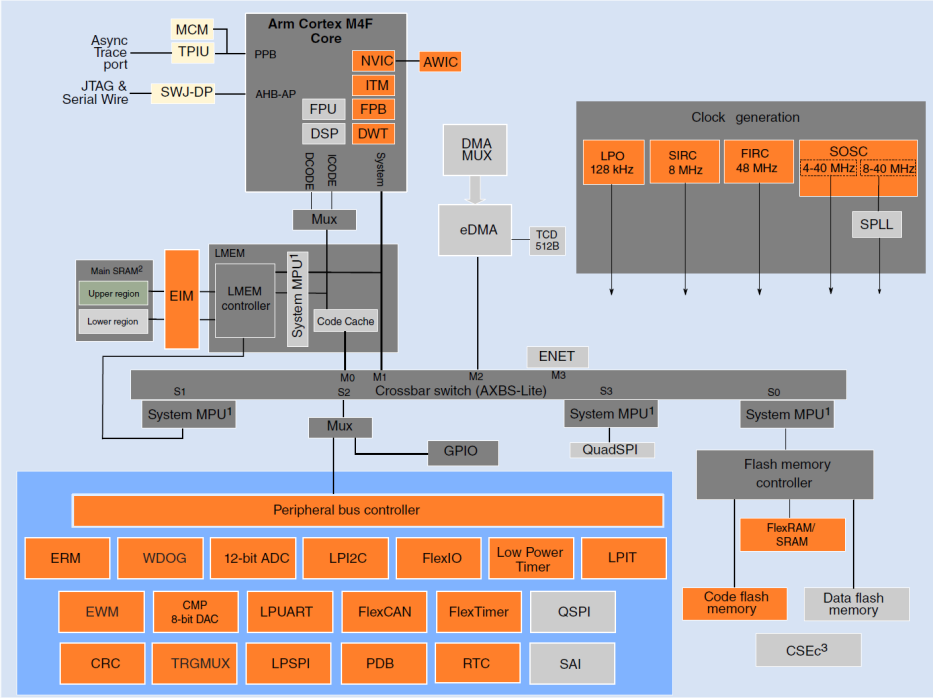



**Figura 3.3:** Architettura ad alto livello S32K116

- External crystal OSCillator (OSC) per lavorare con cristalli di frequenza massima 40 MHz.

L'utilizzo del FIRC è indicato quando non si hanno precise richieste relative alla precisione, infatti presenta una deviazione del 1%, come anche il regime SIRC, utile nel caso si voglia avere un risparmio energetico. Questi due regimi non hanno bisogno di un cristallo esterno e ciò favorisce un risparmio sulla componentistica. Per quanto riguarda l'applicazione studiata, si deve fare uso del regime con cristallo esterno per aumentare la precisione e rimanere all'interno delle norme che regolamentano il settore automotive. Il cristallo esterno installato è stato scelto con frequenza di 40 Mhz per poter usare la rete CAN in modalità High Speed cioè fino ad una velocità di 1 MBit/s. Si riporta in figura 3.4 lo schema dell'architettura del clock e si noti che nella versione di microcontrollore S32k116 non è presente il blocco *Phase Lock Loop* (PLL).

## 3.2 Driver a basso livello

Come accennato precedentemente, alcuni driver erano già stati sviluppati internamente all'azienda, tuttavia è stato richiesto un ulteriore lavoro per

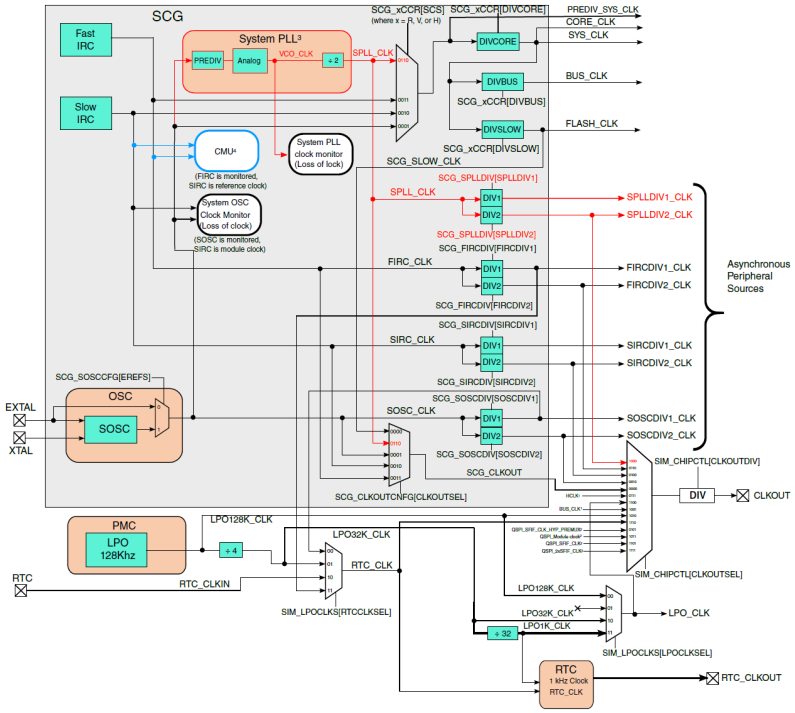



**Figura 3.4:** Architettura clock S32K116

aggiungere quelli mancanti e correggerne altri. I driver sono stati pensati e scritti in modo da poter funzionare su un qualsiasi progetto, sono stati quindi resi il più generale possibile andando ad usare definizioni, strutture e metodi automatici dove possibile. Anche le funzioni all'interno dei driver sono pensate con nomi generali e non specifici. Utilizzando questa filosofia di programmazione è possibile sfruttare il codice scritto su diversi progetti andando a minimizzare il tempo di sviluppo. Inoltre, il codice risulta molto più robusto dal punto di vista dei *BUG* in quanto essendo riutilizzato anche da altri collaboratori è possibile scoprirne e correggerli in tempi più rapidi.

### 3.2.1 GPIO

Il primo driver scritto per questo controllore è stato quello riguardante le *GPIO*. Queste possono essere configurate ed in base alla configurazione data si attivano le diverse funzioni. Si pensi per esempio ai pin dedicati al JTAG : possono assumere altre funzioni dopo la fase di bootloader se riconfigurati e ogni pin ha una funzione di default da riconfigurare in caso di necessità. Il lavoro è stato impostato in modo da andare a definire prima una struttura di array, dove ciascuno contiene vari elementi per il setup del pin, ottenendo

quindi 43 array da 7 elementi ciascuno. Il listato della struttura è riportato di seguito:

```
static const e_PinConfiguration_t pinConfiguration[PIN_NUMBER] = {
    /* PTD1  PIN1 */  { PORT_D, PORTN_1,  ALT_1, PULL_ENABLED,  PULL_DOWN,
    OUTPUT, GPIO_LOW},
    /* PTD0  PIN2 */  { PORT_D, PORTN_0,  ALT_1, PULL_ENABLED,  PULL_DOWN,
    OUTPUT, GPIO_LOW},
    /* PTE5  PIN3 */  { PORT_E, PORTN_5,  ALT_5, PULL_DISABLED, PULL_DOWN,
    OUTPUT, GPIO_LOW},
    /* PTE4  PIN4 */  { PORT_E, PORTN_4,  ALT_5, PULL_DISABLED, PULL_DOWN,
    OUTPUT, GPIO_LOW},
    /* PTB7  PIN8 */  { PORT_B, PORTN_7,  ALT_1, PULL_DISABLED, PULL_DOWN,
    INPUT,  GPIO_LOW},
```

**Listing 3.1:** GPIO driver - definizione pin

L'array di configurazione è definito nel seguente modo. Il primo elemento rappresenta la porta a cui appartiene il pin nell'esempio di codice 3.1, in particolare il pin numero 1 è associato alla porta D del microcontrollore. Il secondo elemento rappresenta il numero del pin associata alla porta, il terzo elemento invece rappresenta la funzione che il pin deve assumere dopo la configurazione che può andare da 0 a 7, rispettivamente *ALT_0* - *ALT_7*. Successivamente si trova il comando di attivazione o disattivazione delle resistenze interne di *pull-up* o *pull-down*. Come quinto elemento si ha la definizione dell'eventuale *pull-up* o *pull-down*. Tale campo viene definito anche se l'opzione non è abilitata, così che in questo modo la struttura subisce cambiamenti nel caso di un altro progetto. Si trovano infine gli ultimi due elementi che sono la definizione di pin input oppure output e lo stato logico che deve assumere dopo la configurazione. Tutti gli elementi fanno uso di definizioni di *enum* in modo da essere esplicative e quindi chiare nella lettura e tali per cui possono essere cambiate in modo più semplice ed in un solo punto del codice. Di seguito se ne riportano alcune a titolo di esempio.

```
typedef enum {
  PORT_A = (uint08t)0,
  PORT_B = (uint08t)1,
  PORT_C = (uint08t)2,
  PORT_D = (uint08t)3,
  PORT_E = (uint08t)4
}e_Port_t;

typedef enum {
  PORTN_0 = (uint08t)0,
  PORTN_1 = (uint08t)1,
  PORTN_2 = (uint08t)2,
  PORTN_3 = (uint08t)3,
  PORTN_4 = (uint08t)4,
  PORTN_5 = (uint08t)5,
  PORTN_6 = (uint08t)6,
  PORTN_7 = (uint08t)7,
  PORTN_8 = (uint08t)8,
  PORTN_9 = (uint08t)9,
  PORTN_10 = (uint08t)10,
```

```
  PORTN_11 = (uint08t)11,
  PORTN_12 = (uint08t)12,
  PORTN_13 = (uint08t)13,
  PORTN_14 = (uint08t)14,
  PORTN_15 = (uint08t)15,
  PORTN_16 = (uint08t)16,
  PORTN_17 = (uint08t)17
}e_PortNumber_t;

typedef enum {
  ALT_0 = (uint08t)0,
  ALT_1 = (uint08t)1,
  ALT_2 = (uint08t)2,
  ALT_3 = (uint08t)3,
  ALT_4 = (uint08t)4,
  ALT_5 = (uint08t)5,
  ALT_6 = (uint08t)6,
  ALT_7 = (uint08t)7
}e_Alternative_t;
```

**Listing 3.2:** GPIO driver - typedef

Su tutte le definizioni è stato eseguito un *cast* ad 8 bit. In questo modo il numero intero di facile lettura, al posto di essere scritto forma di parola ad 8 bit, è convertito e poi usato per la configurazione dei registri non andando ad eseguire shift futuri. Al contrario se si fosse tenuto il numero intero, sarebbe stato rappresentato con 32 bit, andando a complicare la funzione che configura i registri del *GPIO*. Questa funzione è scritta nel codice 3.3, il cui nome dato è generico ma esplicativo e non contiene argomenti. Questa scelta è dettata dal fatto che questa funzione viene richiamata da un altro file ma soprattutto nel caso in cui il microcontrollore venisse cambiato la funzione potrebbe essere riutilizzata andando a cambiare solo l'array che definisce i pin.

```
void  GPIOInitApiFct (void)
{
  uint08t locCounter;
  e_Port_t locPort;
  e_PortNumber_t locPortN;
  e_Alternative_t locAlternative;
  e_PullConfig_t locPullConfig;
  e_PullUpDown_t locPullUpdown;
  e_InputOutput_t locInputOutput;
  e_GpioStatus_t locOutputInit;

  for(locCounter = (uint08t)0; locCounter < PIN_NUMBER; locCounter++)
  {
    locPort = pinConfiguration[locCounter].port;
    locPortN = pinConfiguration[locCounter].portNumber;
    locAlternative = pinConfiguration[locCounter].alternative;
    locPullConfig = pinConfiguration[locCounter].pullConfig;
    locPullUpdown = pinConfiguration[locCounter].pullUpdown;
    locInputOutput = pinConfiguration[locCounter].inputOutput;
    locOutputInit = pinConfiguration[locCounter].outputInit;

    portBase[locPort]->PCR[locPortN] |= PORT_PCR_MUX(locAlternative)|
                        PORT_PCR_PS(locPullUpdown)|
```

```
228                         PORT_PCR_PE(locPullConfig);

230     portBase[locPort]->PCR[locPortN] |= PORT_PCR_LK(1U); /* lock the port */

232     /* if pin is input or output */
        if(locAlternative == ALT_1)
234     {
          if(locInputOutput == INPUT)
236       {
            gpioBase[locPort]->PDDR &= ~(GPIO_PDDR_PDD(((uint32t)1 << locPortN))
        );
238       }
          else
240       {
            if(locOutputInit == GPIO_LOW)
242         {
              gpioBase[locPort]->PCOR |= (GPIO_PCOR_PTCO(((uint32t)1 << locPortN
        ))); /* initialize output status */
244           gpioBase[locPort]->PDDR |= (GPIO_PDDR_PDD(((uint32t)1 << locPortN)
        ));
              gpioBase[locPort]->PCOR |= (GPIO_PCOR_PTCO(((uint32t)1 << locPortN
        ))); /* confirm output status */
246         }
            else
248         {
              gpioBase[locPort]->PSOR |= (GPIO_PSOR_PTSO(((uint32t)1 << locPortN
        ))); /* initialize output status */
250           gpioBase[locPort]->PDDR |= (GPIO_PDDR_PDD(((uint32t)1 << locPortN)
        ));
              gpioBase[locPort]->PSOR |= (GPIO_PSOR_PTSO(((uint32t)1 << locPortN
        ))); /* confirm output status */
252         }
          }
254     }
        else
256     {
          ;
258     }
      }
260 }
```

**Listing 3.3:** GPIO driver - Funzione inizializzazione pin

Questo driver contiene inoltre altre funzioni, che non saranno riportate, per leggere i *GPIO* configurati come input, comandare i *GPIO* configurati come output oppure eseguire il *toggle* di un output.

### 3.2.2 Timmer

All'interno di questo microcontrollore, la periferica che si occupa della gestione dei timer è chiamata *FlexTimer Module* (FTM). Sono presenti 2 moduli da 16 canali ciascuno e in questo progetto vengono usati dei canali di entrambi i moduli che differiscono sulla frequenza di funzionamento. Ogni canale del FTM può essere abilitato singolarmente mentre le configurazioni più generali, come frequenza di funzionamento o modo di contare, vengono definite a

coppie di canali. Altre configurazioni più specifiche come ad esempio la polarità del segnale in uscita, quando si accende o si spegne il canale, è decisa per singolo canale. Si è voluto utilizzare due FTM al posto di uno perché si ha un range di frequenze di lavoro di 10 a 1 e perchè si voleva avere più flessibilità nell'applicazione. Un canale del primo FTM0 viene usato per generare e fornire in uscita la PWM per il chip del controllo motore. Questo modulo opera ad una frequenza di 10 kHz. Il secondo modulo FTM1 invece è stato utilizzato in un secondo momento e non genera un segnale di uscita ma è utilizzato come trigger per iniziare la conversione dell'ADC. Questo lavora ad una frequenza di 1 kHz, si vuole quindi andare a fare una misura ogni 10 periodi della PWM di comando motore. Questo valore potrebbe essere modificato in futuro e quindi se si fosse utilizzato lo stesso modulo che genera la PWM del controllo per frequenze di misura basse oppure se la frequenza per il controllo motore aumentava o entrambi i casi, si rischiava l'overflow del contatore generando un BUG. Queste sono state le considerazioni alla base della scelta unite poi anche dal fatto che il popolamento dei pin del microcontrollore lasciasse spazio di manovra.

Anche in questo caso si è cercato di creare delle strutture che possano essere riutilizzate per più progetti, quindi andando a far uso di definizioni *enum* come nel caso di GPIO e di array di strutture per definire le funzioni dei due moduli e dei singoli canali. In questo caso non vengono riportati gli esempi di codice in quanto molto simili a quelli visti per il GPIO. Si vuole far presente che anche in questo caso si ha una funzione di inizializzazione ma questa, in aggiunta alla configurazione dei canali, va a modificare dei registri che riguardano la parte che gestisce *Peripheral Clock Controller* (PCC), cioè dei registri che abilitano il segnale di clock sul modulo FTM designato e dove viene deciso anche da quale sorgente prendere il segnale. Senza questa modifica, il modulo risulterebbe spento e quindi impossibile da configurare. Lo schema è riportato in figura 3.5 dove si vede il alto a sinistra il blocco che si occupa dell'abilitazione del clock, mentre sottostante è presente un multiplexer dove sono collegate le varie sorgenti di clock tra cui segliere. Anche nel modulo FTM è presente un secondo multiplexer che permette un ulteriore scelta sulla sorgente.

I due moduli sono stati configurati in modo simile, entrambi usano il canale 0, la polarità di uscita è la stessa come anche la polarità del segnale quando il modulo è spento mentre le differenze risiedono nei valori di prescaler. Il modulo FTM0 presenta una configurazione con prescaler più basso in modo da avere una frequenza di aggiornamento del counter superiore così che si possano ottenere delle frequenze per la PWM dai 10 kHz a salire. Il modulo FTM1 ha un valore di prescaler più alto, così da poter lavorare a frequenze più basse senza generare un overflow del counter. Un aspetto da

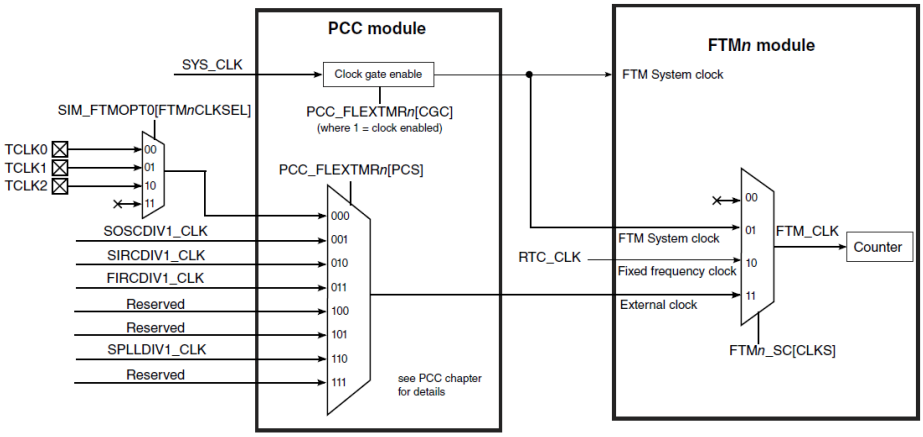



**Figura 3.5:** PCC riguardante il modulo FTM

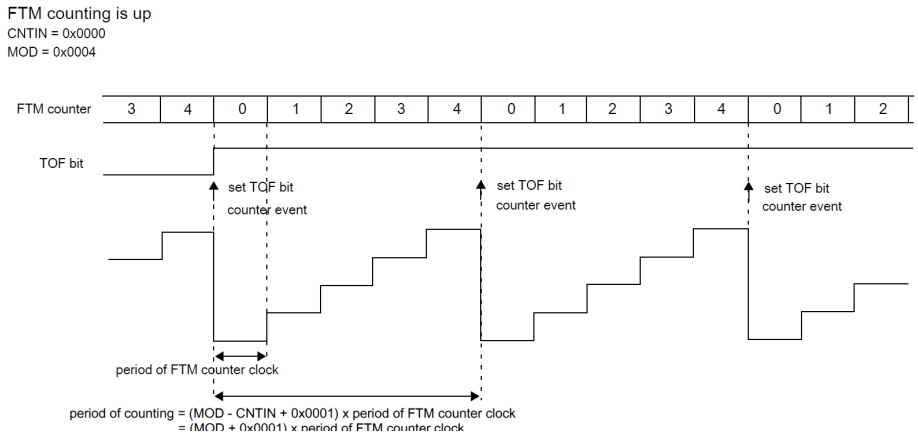



**Figura 3.6:** Modalità di funzionamento a rampa del modulo FTM

approfondire è la modalità di conteggio e ne esistono di due tipi. La prima detta a rampa up o down riporta in figura 3.6 prevede che il conteggio parta dal valore minimo impostato e quando arriva al valore massimo impostato genera un segnale che commuta l'uscita del canale del modulo FTM, contemporaneamente riporta il valore del counter al valore minimo, questo se la modalità è up. Al contrario se la modalità selezionata è down il conteggio parte dal valore massimo impostato e quando arriva a quello minimo l'uscita del canale del modulo FTM viene commutata ed in contemporanea viene ricaricato il valore massimo all'interno del counter. Si ha anche la possibilità di generare dei segnali di trigger impostando una soglia oppure quando si ha il ripristino del valore da cui iniziare a contare all'interno del counter. Questo primo metodo però non è utilizzato per il progetto sviluppato per un motivo

ben preciso e riguarda la generazione dei trigger. Nel modulo FTM1 questi trigger vengono utilizzati per far partire le conversione dei canali in ingresso all'ADC ma se una soglia può essere decisa, e viene impostata ad esempio a metà del periodo del segnale di PWM di uscita per il controllo motore, l'altra invece no perchè sarà sempre quando avverrà la reinizializzazione del valore all'interno del registro counter. Questo comporta il cambiamento di polarità del segnale del PWM che significa uno switch dei mosfet del ponte con relativo tempo morto. Ricordando dai capitoli precedenti che il feedback dell'integrato è analogico otterremo una lettura errata o senza alcun senso.

Il secondo metodo, in figura 3.7, utilizza un altro tipo di conteggio definito come up-down cioè, a differenza del primo, quando si parte da un valore inizializzato all'interno del counter, si inizia a contare sommando le unità. Una volta arrivati al valore massimo al posto di reinizializzare il registro si comincia a decrementare il valore all'interno del registro. Come nel metodo precedente, anche in questo si cambia la polarità al segnale di uscita e nel caso si può generare un segnale di trigger. Arrivati di nuovo al valore minimo, si rincomincia a sommare il registro di counter e anche in questo caso cambia la polarità del segnale di uscita e viene generato un segnale di trigger. Come si vede, si hanno già due segnali di trigger ma sono in corrispondenza del cambio di polarità del segnale di uscita, quindi sono inutilizzabili per fare delle letture del feedback che abbiano un senso. Il vantaggio di questo metodo però sta nel fatto di poter impostare un'altra soglia che generi un segnale di trigger. Anche nel precedente metodo era possibile farl,o ma veniva attraversata una sola volta, invece qui si ha che questa soglia viene attraversata due volte, una volta quando si ha il conteggio sommando, l'altra quando si sottrae. Quindi vengono generati due segnali di trigger che possono essere sfruttati per far partire due conversioni all'interno di un periodo di PWM e se la soglia corrisponde a metà di mezzo periodo, si avranno tutte le letture dell'ADC effettuate teoricamente all'altezza del valor medio dell'onda PWM. Questo vantaggio porta ad avere delle letture più precise e con un senso rispetto a primo metodo.

Nel progetto si è deciso di impostare il valore minimo a 0 e lasciare variare solo il valore massimo del registro counter per poter agire sul duty cycle dell'onda PWM quando è in funzione. Questo è stato fatto per una futura implementazione di un controllore che agirà su questo parametro. Viene riportato il codice 3.4 della funzione scritta, che riceve come primo argomento il canale su cui eseguire il cambio e come secondo il valore di duty cycle in percentuale. Come si può osservare, viene effettuata una proporzione da percentuale a valore intero e poi attraverso le formule fornite dal produttore si calcola il rispettivo valore da scrivere all'interno del registro counter.

```
uint08t Dulty_Cycle_Change_1(uint08t channel, uint16t dultycycle)
```

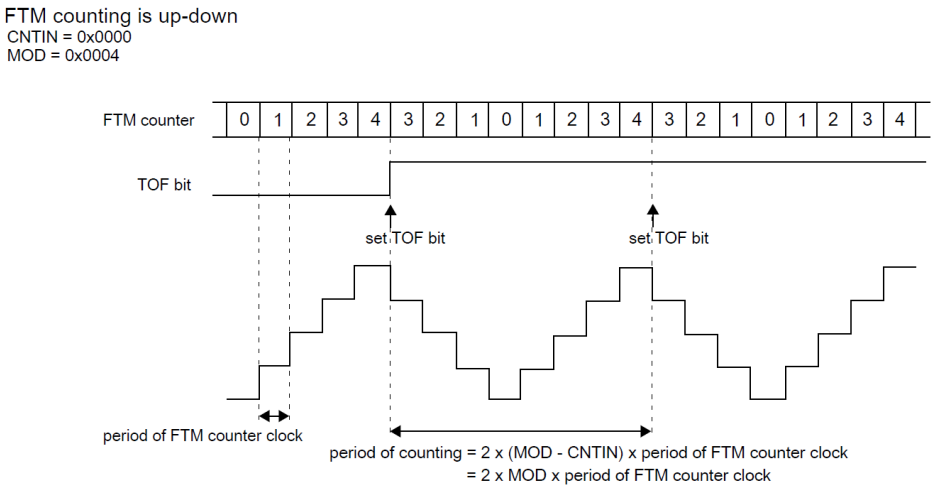



**Figura 3.7:** Modalità di funzionamento up-down del modulo FTM

```
718 {
    uint32t temp = 0;
720
    if (channel > 7)
722 {
      return 1;
724 }
    else
726 {
      /* Dulty cycle: dt=2*(CnV-CNTIN) so                    */
728     /*               dt = 100% -> CnV = MOD                */
      /*              dt = 1% - 99% -> CnV = (MOD*dt)/2000 */
730   /*              dt = 0% -> CnV = 0x0000               */
      if (dultycycle < 0)
732   {
        return 1;
734   }
      else if (dultycycle == 0)
736   {
          FTM1->CONTROLS[channel].CnV = FTM_CnV_VAL((FTM1->MOD+10));
738   }
      else if ((dultycycle == 1000)||(dultycycle > 1000))
740   {
        FTM1->CONTROLS[channel].CnV = FTM1->CNTIN;
742   }
      else
744   {
        temp = FTM1->MOD - ((((((uint32t)dultycycle)/10)*FTM1->MOD)>>2)/25) +
    FTM1->CNTIN;
746     FTM1->CONTROLS[channel].CnV = FTM_CnV_VAL(temp);
      }
748 }
    return 0;
750 }
```

**Listing 3.4:** FTM driver - modifica duty cycle

In questa funzione vengono gestiti anche i casi limite:

- duty cycle $< 0$, uscita dalla funzione e segnalazione dell'errore;
- duty cycle $= 0\%$;
- duty cycle $= 100\%$.

Questo perché negli estremi i registri devono essere impostati in una maniera determinata descritta nel *Reference Manual* del microcontrollore.
Altre funzioni all'interno del driver sono quelle per l'accensione o spegnimento del modulo FTM oppure quelle di accensione e spegnimento del singolo canale di un FTM. Per testare il funzionamento di questo driver si è provveduto a porre due sonde sui corrispettivi *header* che riportavano i pin.

### 3.2.3 ADC

Un altro driver scritto è stato quello per poter configurare ed usare l'ADC presente all'interno del microcontrollore. Questo presenta 13 canali e ne sono stati usati in questa prima applicazione 4, mentre nella scheda finale ne saranno utilizzati 8. Anche per questa periferica è necessario come prima cosa abilitare il clock e andare a scegliere anche la sorgente, che da datascheet consigliano essere FIRCDIV2_CLK, cioè il regime FIRC diviso 2 ottenendo una frequenza di lavoro intorno ai 24 MHz. Il modulo ADC come già definito nelle specifiche effettua conversioni con una precisione di 12 bit ed il tipo di ADC è ad approssimazioni successive. Questo potrebbe risultare poco preciso se vogliamo misurare canali diversi uno in successione all'altro e che presentano tra di loro un cambio della dinamica del segnale ampio. Nel caso di questo progetto non è richiesto una precisione elevata quindi il problema non è stato trattato. La dinamica di ingresso va da 0 V ad un massimo di 5 V ed i segnali misurati sono stati condizionati con un partitore di tensione nel caso in cui questi superassero il valore massimo consentito. La prima versione di questo driver è stata scritta per misurare solamente il valore del potenziometro installato sulla scheda, infatti si è cercato di far funzionare la configurazione base prima di procedere con una configurazione via via sempre più complessa. Dal manuale del microcontrollore sono stati ricavati i passi per poter configurare al meglio questa periferica e si vogliono riportare i principali per questa prova. Nel caso sotto studio si è pensato di avere un trigger software per far partire la conversione dell'unico canale inizializzato e questo è stato inserito in un task del sistema. Sempre all'interno del driver è stata creata una funzione che vada a leggere il risultato della conversione memorizzato in un registro apposito. Per vedere il funzionamento dell'ADC si è pensato di far cambiare colore al LED RGB installato in modo proporzionale al valore di tensione letto. Si è voluto procedere in questo modo perché

andando a controllare i registri del microcontrollore attraverso il *debugger* si incorreva in un problema che portava la funzione di lettura del valore convertito a fallire generando un errore. Per spiegare il problema è utile capire prima come funziona la lettura del risultato dove il codice 3.5 mostra la gestione della stessa per più canali ed effettuando una media su un numero di campioni scelto dall'utente.

```
void ADCConversionMgmFct(void)
{
  e_AdcConfigIndex_t locCounter;
  uint16t TempBuffer[ADC_SIGNALS_NUMBER] = {0};

  for(locCounter = (e_AdcConfigIndex_t)0; locCounter < ADC_SIGNALS_NUMBER;
    locCounter++)
  {
    /* check if last conversion is finished */
    if((adcConfigBuffer[locCounter].adc->SC1[adcConfigBuffer[locCounter].
    adcRegister] & ADC_SC1_COCO_MASK) != 0U)
    {
      TempBuffer[locCounter]  = (uint16t)(adcConfigBuffer[locCounter].adc->R
    [adcConfigBuffer[locCounter].adcRegister]);

      if(avg_sample[locCounter] < avg_value)
      {
        AccBuffer[locCounter] += (uint32t)(TempBuffer[locCounter]);
        avg_sample[locCounter]++;
      }
      else
      {
        adcReadBuffer[locCounter] = (uint16t)(AccBuffer[locCounter]/
    avg_value);
        avg_sample[locCounter] = 0;
        AccBuffer[locCounter] = 0;
      }

      /* check if COCO flag is cleared */
      if((adcConfigBuffer[locCounter].adc->SC1[adcConfigBuffer[locCounter].
    adcRegister] & ADC_SC1_COCO_MASK) != 0U)
      {
        adcConversionStatus = ADC_FAILURE;
      }
      else
      {
        ;
      }
    }
    else
    {
      adcConversionStatus = ADC_FAILURE;
    }
  }
}
```

**Listing 3.5:** ADC driver - gestione lettura valore convertito

Questa funzione non presenta nessun argomento ed è scritta in maniera tale possa essere usata con qualsiasi numero di canali e nel caso anche da diversi ADC, qualora si volesse esportarla su altri microcontrollori della stessa

famiglia. Questa funzione attraverso il primo ciclo *for* a riga 226 controlla ogni canale e lo fa attraverso la prima istruzione di *if* a riga 229. Questa interrogazione va a controllare se il bit di flag di fine conversione è stato settato, questo bit è chiamato *COCO* ed è di sola lettura, viene poi resettato una volta letto. Se il bit è a valore logico 1 allora si entra all'interno dell'istruzione *if* altrimenti viene generata una segnalazione di errore, proprio per questo funzionamento durante la fase con il debugger sono stati riscontrati dei problemi. Il bit *COCO* veniva settato correttamente ma mentre si guardavano i registri corrispondenti e si procedeva step-by-step con il *debugger* questo veniva resettato portando al risultato di far generare sempre errore alla funzione questo è il motivo dell'utilizzo de LED.
Una volta scoperto il problema e risolto si è passati al successivo passo, cioè quello di avere un trigger hardware al posto di uno software per poter collegare il trigger generato dal modulo FTM1 ed avere delle misure sincrone eseguite in un preciso istante, di contrario se il trigger fosse rimasto software si sarebbe incappati nel problema di non sapere quando la conversione iniziasse perché dipendente dal tempo di ciclo delle istruzioni. Il codice 3.6 riporta la funzione di inizializzazione dove si vede la configurazione del registro *PCC* come prima cosa e di seguito la risoluzione dell'ADC. Anche in questo caso si utilizzano molte *define* in modo da rendere il codice più leggibile.

```
324 static void ADCInit(void)
    {
326   e_AdcConfigIndex_t locCounter;

328   PCC->PCCn[PCC_ADC0_INDEX] &= ~PCC_PCCn_CGC_MASK;  /* Disable clock to
        change PCS */

330   PCC->PCCn[PCC_ADC0_INDEX] |= PCC_PCCn_PCS(ADC_FIRCDIV2_CLK); /* PCS=3 */

332   PCC->PCCn[PCC_ADC0_INDEX] |= PCC_PCCn_CGC_MASK;  /* Enable bus clock in
        ADC */

334   ADC0->CFG1 |= ADC_CFG1_MODE(RES12BIT)|  /* 0b00: 8-bit, 0b01: 12-bit, 0b10
        : 10-bit */
              ADC_CFG1_ADICLK(ALTCLK1); /* Only ALTCLK1 is available */
336
      //ADCCalibration();
338
      ADC0 ->CFG2 |= ADC_CFG2_SMPLTS(100); /* CLK Cycle for Sample*/
340
      ADC0->SC2 &= ~ADC_SC2_ADTRG_MASK;
342
      ADC0->SC2 |= ADC_SC2_ADTRG(HW_TRG);
344
      ADC0->CFG1 &= ~ADC_CFG1_ADIV_MASK;
346
      ADC0->CFG1 |= ADC_CFG1_ADIV(DIV1);
348
      for(locCounter = (e_AdcConfigIndex_t)0; locCounter < ADC_SIGNALS_NUMBER;
        locCounter++)
350   {
```

```
        adcConfigBuffer[locCounter].adc->SC1[adcConfigBuffer[locCounter].
        adcRegister] = ADC_SC1_ADCH(adcConfigBuffer[locCounter].adcInputChannel)
        ;
352   }

354   ADC0->SC1[ADC_SC1D] |= ADC_SC1_AIEN(ENABLE);
}
```

**Listing 3.6:** ADC driver - inizializzazione periferica

Dopo aver definito la risoluzione è possibile anche effettuare una calibrazione dei canali dell'ADC per aumentare la precisione della lettura, in questo caso la funzione è stata scritta per completezza ma non utilizzata per questo progetto. Alla riga 346 viene impostato il tempo di *sampling* del segnale sotto forma di periodi di clock e l'istruzione successiva va a definire il tipo di trigger che l'ADC dovrà utilizzare per cominciare la conversione. Alla riga 356 invece viene scritto un ciclo *for* per andare ad associare l'input dell'ADC con il registro del risultato questo per un numero arbitrario di canali scelto dall'utente. L'ultima istruzione abilita il modulo ADC al funzionamento che risulta pronto per l'utilizzo con un trigger hardware.
Il terzo passo è stato quello di collegare gli interupt generati dal canale del FTM1 con l'inizio conversione del primo input dell'ADC ed inoltre far eseguire le successive conversioni degli altri canali di input in sequenza al termine della precedente, mentre alla fine dell'ultima conversione viene generato un interupt per segnalare la fine della sequenza ed andare a leggere i valori contenuti nei registri. Per fare questi collegamenti vengono usati due ulteriori moduli presenti all'interno del microcontrollore e spiegati in seguito. La filosofia dietro questa scelta è stata quella di non andare ad appesantire il *core* con la gestione degli interrupt d'inizio conversione rendendo più funzionale il sistema.

### 3.2.4 TRGMUX e PDB

Come accennato in precedenza questi due moduli sono stati usati per alleggerire il carico di lavoro sul core. In figura 3.8 si può vedere lo schema generale di come verrà collegato il sistema con il modulo FTM, *Trigger Multiplexer* (TRGMUX), *Programable Delay Block* (PDB) ed ADC. Per i due moduli sono stati scritti due file diversi per avere più flessibilità nel caso si voglia aggiungere altri trigger o nel caso si voglia implementare altro codice per altre applicazioni.
Il TRGMUX è un meccanismo fornito all'interno del microcontrollore che permette di collegare in maniera flessibile le varie sorgenti di trigger con le periferiche di destinazione. Ognuna di esse ha a disposizione un determinato numero di ingressi di trigger su cui scegliere. Il codice 3.7 presenta nello

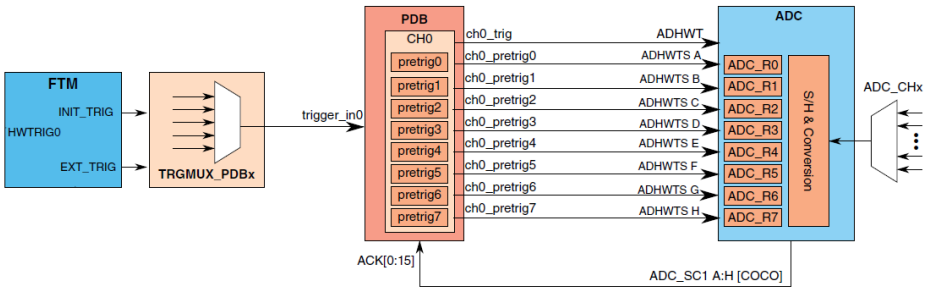



**Figura 3.8:** Schema catena di trigger

specifico il registro adibito per collegare l'ingresso del PDB con l'uscita del FTM attraverso il TRGMUX.

```
126 void TRGMUXInit(void)
{
128   TRGMUX -> TRGMUXn[TRGMUX_PDB0_INDEX] |= TRGMUX_TRGMUXn_SEL0(FTM0_INIT_TRIG
        );
}
```

**Listing 3.7:** TRGMUX driver - collegamento FTM1 con PDB

Come si nota, in questo caso è necessaria una solo istruzione per settare nel modo corretto questa associazione. Anche in questo caso leggendo il manuale del microcontrollore è stata creata una *define* con all'interno tutte le possibili sorgenti di trigger per una maggiore completezza.
Per quanto riguarda il modulo PDB, questo è più complesso e non si occupa solo di rendere possibile il collegamento con l'hardware trigger dell'ADC. Questo modulo si occupa anche di collegare il segnale di fine conversione dell'ingresso precedente con quello successivo, come si può vedere in figura 3.9 ed inoltre è il modulo che segnala l'interupt di fine della sequenza di conversioni. Questa modalità è chiamata *back-to-back* all'interno del manuale. Il PDB si occupa anche di gestire ad esempio un delay da quando si riceve il segnale di trigger a quando si dà il segnale al modulo ADC di iniziare la conversione. Inoltre può fornire dei pre-trigger all'ADC. Lo schema interno di un PDB è riportato in figura 3.10 dove si pUò notare la presenza di comparatori e counter. Questo risulta utile quando si vuole una misura ad un tempo preciso senza far uso di un canale del modulo FTM. Nel caso sotto esame non è stato possibile effettuare questa scelta perché si è cercato la sincronia con il segnale PWM che comanda il chip *VNH7040*. Il modulo presenta anche la possibilità di gestire eventuali errori che si possono creare nella generazione dei trigger. Come per gli altri blocchi, anche il PDB ha bisogno dell'abilitazione del clock nel modulo PCC. Nel codice 3.8 viene riportata la funzione di inizializzazione di questa periferica. La funzione non risulta essere molto generica in quanto il S32K116 presenta un solo PDB quindi le configurazioni

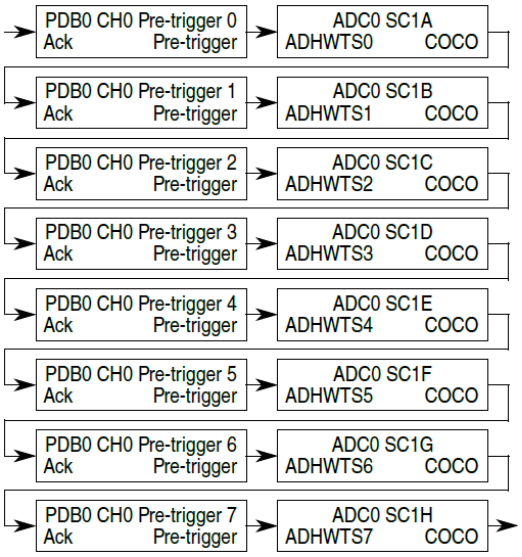



**Figura 3.9:** Schema PDB back-to-back

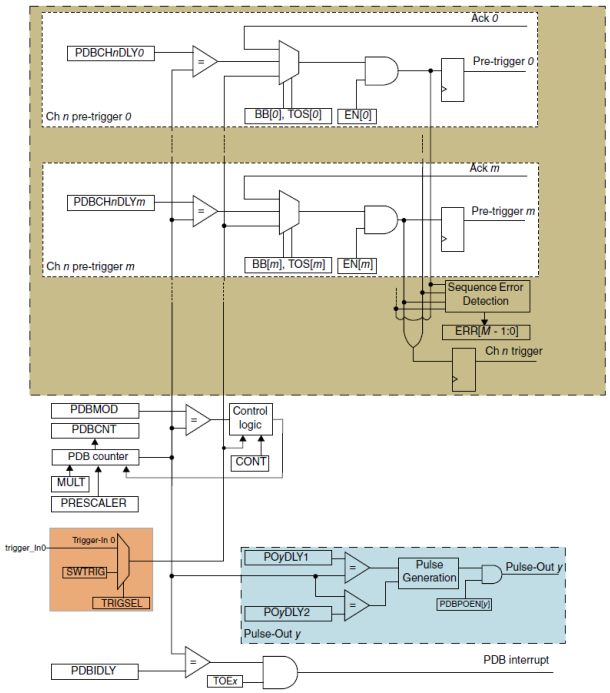



**Figura 3.10:** Schema interno PDB

sono state pensate per il singolo modulo. Si può notare che la frequenza di funzionamento è quella del core, non sono presenti prescaler nè si introducono delay tra l'arrivo del trigger e l'inizio della conversione. Si abilita la funzione di *back-to-back* per gli 8 canali necessari ed alla fine si abilita la periferica.

```
void PDB0Init(void)
{
  PDB0DisableClock();
  PDB0EnableClock();  /* Enable bus clock in PDB0 */

  PDB0->SC |= PDB_SC_PRESCALER(DIV1)|    /* Prescaler:  1 => sysclck/(MULT) =
     48MHz */
          PDB_SC_TRGSEL(TRIGGER_IN0)| /* Connect the PDB with the external
    trigger */
              PDB_SC_MULT(MULT1)|        /* Multiplication factor is 1. */
              PDB_SC_PDBEIE(ENABLE);

  PDB0->CH[CHANNEL0].C1 |= PDB_C1_BB((PDB0_CH0_MASK - 1U))| /* 23-16 -> PDB
    Chan Pre-TRG Back-to-Back  */
                       PDB_C1_TOS(0x00u)|                /* Operation enable
       */
                           PDB_C1_EN(PDB0_CH0_MASK);         /* Pretrigger
    Output Select: 0=bypassed */
                                                          /* 1=enabled */
                                                          /* 7-0 -> PDB
    Chan Pre-TRG ENABLE */
  PDB0->SC |= PDB_SC_LDOK_MASK;
}
```

**Listing 3.8:** PDB driver - funzione inizializzazione

Il codice 3.9 invece si occupa di rilevare se ci sono errori generati dal PDB e nel caso risultassero provvede ad reinizializzare il modulo.

```
void PDBErrorMgmFct(void)
{
    uint32t pdb0ch0s;
    pdb0ch0s = (uint32t)(PDB0->CH[0].S);

    /* check if PBD has no error */
    if( PDB0_CH0_S_ERR != pdb0ch0s)
    {
      /* clear error */
      PDB0->CH[0].S = ~PDB_S_ERR_MASK;

      PDB0->SC &= ~PDB_SC_PDBEN_MASK; /* dsable PDB 0 */

      PDB0Init(); /* restart PDB */
    }
    else
    {
      /* do nothing */
    }
}
```

**Listing 3.9:** PDB driver - funzione gestione errori

## 3.3 Altri driver

In questa trattazione non sono inseriti altri driver presenti nel progetto in quanto sviluppati all'interno dell'azienda da parte dei dipendenti, tuttavia si vuole comunque ricordare quali sono e darne una piccola descrizione funzionale. Il **clock driver** si occupa di tutta la configurazione dei registri che riguardano il funzionamento, l'impostazione del clock. Il **LPIT driver** si occupa di configurare un low power interrupt timer, il quale è usato per generare un interrupt ogni microsecondo, creando una specie di sistema operativo *realtime*. Andando a sommare, all'interno di una funzione che gestisca questo interrupt, una variabile possono essere create delle soglie di millisecondi che corrispondono a delle specifiche azioni come ad esempio la velocità di un task del sistema operativo. Un altro driver molto importate è quello per la gestione della rete CAN chiamato **CAN driver**. In questo caso sono state create delle code per memorizzare i messaggi in arrivo e poterli servire quando il core è libero, ma anche code per la trasmissione del messaggio in questo modo se il bus è occupato si ha il tempo di aspettare che si liberi senza generare errore o bloccare il core su questa unica operazione. Un ultimo driver presente è quello per l'inizializzazione di tutti gli interrupt ed è chiamato **interrupt driver**, il quale è modificato per installare l'interrupt relativo alla fine sequenza di conversioni.

# Capitolo 4

# Test Automatizzati

## 4.1 Applicazione per i test

Dopo aver concluso la parte di driver a basso livello per il microcontrollore, lo step successivo è stato quello di scrivere una piccola applicazione per controllare un motore provvisto di un sensore di posizione analogico. In questo caso si è lavorato in modo gerarchico a due livelli. Il primo nella gerarchia più basso sfrutta le funzioni definite dai driver per creare una funzione più complessa che può ricevere anche degli argomenti. Il livello più alto invece raggruppa sotto un'unica funzione, inserita poi nel task designato, tutte le funzioni necessarie al funzionamento dell'applicazione. L'uso di una struttura gerarchica e non orizzontale favorisce la portabilità e la comprensione del codice. Al fine di automatizzare i test, si è pensato di sfruttare la comunicazione tramite rete CAN, in modo tale che si possano inviare comandi di direzione, valore del duti cycle, comando di enable/disable per il chip *VNH7040*, valore di posizione da raggiungere. Inoltre, attraverso la rete CAN è stato possibile ricevere dati riguardanti il valore letto dall'ADC su diversi ingressi. I principali sono stati quello di feedback dato dal chip e quello del sensore di posizione del motore. Per inviare e ricevere dati è stato usato un CAN transciver esterno collegato tramite USB con il computer, insieme a un programma gratuito dove sono stati definiti alcuni comandi e la decodifica dei messaggi ricevuti. A tal proposito, sono state definite due modalità di funzionamento per l'applicazione. Una più manuale in cui si dichiara direzione, duty cycle e viene lanciato il comando di start e successivamente di stop. Questa funzione è utile per portare l'attuatore in una posizione preferita. La seconda modalità, più intelligente prevede invece solo l'invio di una posizione in mm e del duty cycle. In questo caso è il microcontrollore ad occuparsi di dare direzione e di fermare il motore quando questa viene raggiunta. Come si potrà osservare in

seguito, non è presente un vero e proprio controllo del motore, ma solo una lettura del feedback del sensore di posizione opportunamente linearizzato. Per l'automatizzazione dei test potevano essere sfruttate anche altre comunicazioni, ad esempio tramite SPI oppure USB, ma non sarebbero risultate poi utili ai fini del progetto essendo questo pensato per un'applicazione *automotive*. Si ricorda infatti che l'unica comunicazione presente tra diverse schede all'interno di un autoveicolo è la rete CAN ed altre comunicazioni sono usate solo all'interno della stessa scheda.

Partendo dal file contenente i *task*, andando a verificare poi le funzioni a più alto livello, si andranno a spiegare più nel dettaglio le funzioni descritte al livello sottostante che fanno uso delle funzioni scritte all'interno dei driver. Il codice 4.1 riporta il contenuto del *task* eseguito più velocemente. Al suo interno viene eseguita la funzione di gestione della ricezione dei messaggi CAN e qualora riceva un messaggio valido aggiornerà il bit (**CAN_update_state**). Quando il bit risulta valido, il codice contenuto all'interno del *if* viene eseguito, cosi facendo tramite la funzione descritta si va a scegliere la modalità più manuale oppure quella più automatica. Invece, la funzione sottostante viene eseguita sempre in quanto è quella che controlla il funzionamento automatico dell'attuatore.

```
134 void TaskFastFct(void)
{
136   CanAppliRxFct();
    if(CAN_update_state)
138   {
        MotorManagment();
140     CAN_update_state = 0;
      }
142    PositionbyCAN(CAN_dulty_cycle, CAN_position);
}
```

**Listing 4.1:** Task file - Funzione task veloce

## Scelta della modalità

Per la scelta della modalità di funzionamento si fa uso della funzione ad alto livello **MotorManagment()** riportata nel codice 4.2.

```
44 void MotorManagment()
{
46   if(CAN_update_cmd & CAN_update_dir & CAN_dulty_update)
    {
48     StopGoToPosition();
       StartStopMotorCAN(CAN_command);
50     DultyCyclePWMbyCAN(CAN_dulty_cycle);
       DirectionbyCAN(CAN_direction);
52   }
    else if(CAN_update_cmd || CAN_update_dir || CAN_dulty_update)
```

```
54    {
        /*do nothing*/
56    }
      else
58    {
        StartGoToPosition();
60      DultyCyclePWMbyCAN(CAN_dulty_cycle);
      }
62  }
```

**Listing 4.2:** Application file - Applicazionescelta modalità

Al fine di discriminare vari casi, viene usato un controllo su dei bit di update, estratti tramite una maschera di bit all'interno della funzione di ricezione messaggio da rete CAN. Si possono verificare 3 differenti situazioni. La prima condizione per mettere in movimento il motore è che sia scritta la sequenza giusta di bit all'interno del messaggio CAN e quindi tutti e 3 i bit di update siano validi. La seconda condizione gestita è che almeno uno dei bit di update sia valido e in questo caso si resta insensibile al comando ricevuto in quanto potrebbe essere un errore. La terza condizione invece riguarda tutti gli altri casi non trattati, dove si pensa che si voglia utilizzare la modalità automatica di gestione del movimento.

## 4.2 Modalità manuale

Con questo paragrafo si vogliono spiegare nel dettaglio alcune delle funzioni usate all'interno del codice 4.2. Come prima cosa si va a disabilitare la funzione automatica tramite tramite l'utilizzo della funzione **StopGoToPosition()** che disabilita un enable usato all'interno della funzione **PositionbyCAN()**, vista nel codice 4.1. Dopo aver disabilitato questa modalità, con le successive istruzioni si vanno a definire gli enable o disable per la PWM, il duty cycle e la direzione. I codici di queste funzioni del livello più basso rispetto al precedente sono riportati in seguito e come si può osservare questi fanno uso delle funzioni descritte all'interno dei driver presentati.

```
56  void StartStopMotorCAN(uint08t data)
    {
58    if(data == 0x0F)
      {
60      PWM_Enable(FTM_0_BRIDGE_PWM);
      }
62    else
      {
64      PWM_Disable(FTM_0_BRIDGE_PWM);
      }
66  }
```

**Listing 4.3:** Power stage file - Funzione enable o disable PWM chip

```
76 void DultyCyclePWMbyCAN(uint16t data)
   {
78   (void)Dulty_Cycle_Change(FTM0_BRIDGE_PWM, data);
   }
```

**Listing 4.4:** Power stage file - Funzione cambio duty cycle

```
88 void DirectionbyCAN(e_Direction_t data)
   {
90   static uint08t unlock = 1;

92   if(data == BRAKE_TO_VCC)
     {
94     GPIOSetOutputFtc(DO_BRIDGE_INA, GPIO_HIGH);
       GPIOSetOutputFtc(DO_BRIDGE_INB, GPIO_HIGH);
96     unlock = 1;
     }
98   else if (data == BACKWARD_DIR)
     {
100    if (unlock)
       {
102      GPIOSetOutputFtc(DO_BRIDGE_INA, GPIO_HIGH);
         GPIOSetOutputFtc(DO_BRIDGE_INB, GPIO_LOW);
104      unlock = 0;
       }
106    else
       {
108      /* do nothing*/
       }
110  }
     else if(data == FORWARD_DIR)
112  {
       if (unlock)
114    {
         GPIOSetOutputFtc(DO_BRIDGE_INA, GPIO_LOW);
116      GPIOSetOutputFtc(DO_BRIDGE_INB, GPIO_HIGH);
         unlock = 0;
118    }
       else
120    {
         /*do nothing*/
122    }
       }
124    else if(data == BRAKE_TO_GND)
       {
126      GPIOSetOutputFtc(DO_BRIDGE_INA, GPIO_LOW);
         GPIOSetOutputFtc(DO_BRIDGE_INB, GPIO_LOW);
128      unlock = 1;
       }
130    else
       {
132      GPIOSetOutputFtc(DO_BRIDGE_INA, GPIO_LOW);
         GPIOSetOutputFtc(DO_BRIDGE_INB, GPIO_LOW);
134      unlock = 1;
       }
136 }
```

**Listing 4.5:** Power stage file - Funzione cambio direzione

Le più semplici da come si può notare sono quella di start/stop del motore e quella di cambio duty cycle. Per quanto riguarda invece la funzione per il

set della direzione vista nel codice 4.5 presenta degli accorgimenti maggiori che verranno spiegati di seguito attraverso lo studio di alcuni casi tipici. Le possibilità di funzionamento del chip sono 4:

- freno a $V_{cc}$ - BRAKE_TO_VCC;
- direzione con senso di rotazione orario - FORWARD_DIR;
- direzione con senso di rotazione antiorario - BACKWARD_DIR;
- freno a GND - BRAKE_TO_GND.

Il problema scaturisce dal fatto che alcuni passaggi da un caso all'altro sono proibiti se si vogliono evitare rotture di tipo meccanico. Si pensi infatti che il motore elettrico muove un carico meccanico attraverso degli ingranaggi e che questo può avere una certa inerzia e di conseguenza un tempo di risposta ai comandi più elevato rispetto all'invio del comando di cambio direzione da parte del microcontrollore al chip. I casi proibiti sono quelli di cambio verso di rotazione mentre il motore è attivo, cioè mentre si ha una rotazione oraria e si vuole cambiare verso andando su una rotazione antioraria e la stessa condizione è valida viceversa. Per gestire queste situazioni si è pensato di passare sempre per una *safe zone*, ovvero prima eseguire un'azione di freno e poi, se si vuole cambiare direzione, ridare il comando attraverso il messaggio CAN. Questa doppia sicurezza evita problemi di rotture meccaniche in questa fase di test. Come si può vedere dal codice, il metodo è stato quello di usare una variabile globale denominata **unlock**, che viene asserita nel caso di passaggio per un freno e viene bloccata a valore logico basso quando ci si trova ad operare in uno stato proibito.
Questo è quanto riguarda la funzione manuale attraverso un messaggio CAN, che non risulta l'unico modo per movimentare il motore. Infatti, per utilità sono stati sfruttati anche i pulsanti presenti sulla evaluation board per movimentare il motore in senso orario o antiorario; il codice 4.6 chiamato da un task lasciato in background si occupa di questo aspetto. Anche in questo caso per il cambio direzione sono state prese le stesse misure del caso precedente.

```
void Direction(void)
{
  uint08t unlock;

  if(GPIOReadInput(DI_SW2))
  {
    if(GPIOReadInput(DI_SW3))
    {
      GPIOSetOutputFtc(DO_BRIDGE_INA, GPIO_HIGH);
      GPIOSetOutputFtc(DO_BRIDGE_INB, GPIO_HIGH);
      PWM_Disable(FTM_0_BRIDGE_PWM);
      unlock = 1;
```

```
    }
    else
    {
      if (unlock)
      {
        GPIOSetOutputFtc(DO_BRIDGE_INA, GPIO_HIGH);
        GPIOSetOutputFtc(DO_BRIDGE_INB, GPIO_LOW);
        PWM_Enable(FTM_0_BRIDGE_PWM);
        unlock = 0;
      }
    }
  }
    else
  {
      if(GPIOReadInput(DI_SW3))
      {
        if (unlock)
        {
          GPIOSetOutputFtc(DO_BRIDGE_INA, GPIO_LOW);
          GPIOSetOutputFtc(DO_BRIDGE_INB, GPIO_HIGH);
          PWM_Enable(FTM_0_BRIDGE_PWM);
          unlock = 0;
        }

      }
      else
      {
        GPIOSetOutputFtc(DO_BRIDGE_INA, GPIO_LOW);
        GPIOSetOutputFtc(DO_BRIDGE_INB, GPIO_LOW);
        PWM_Disable(FTM_0_BRIDGE_PWM);
        unlock = 1;
      }
  }
}
```

**Listing 4.6:** Power stage file - Funzione attuazione motore attraverso pulsanti

## 4.3 Modalità automatica

Con questa modalità si intende che il microcontrollore si occupa di gestire la direzione della rotazione del motore ed anche il tempo di attivazione. Questo avviene fino al raggiungimento del target di posizione inviato attraverso il messaggio CAN. Non è presente un vero e proprio tipo di controllo per il motore e la posizione dell'attuatore viene rilevata attraverso la lettura di un sensore di posizione che fornisce un feedback analogico. Il controllo della posizione viene fatto ad ogni esecuzione del task veloce dalla funzione vista nel codice 4.1 ed è abilitata da un'altra funzione vista nell'ultimo caso del codice 4.2. Infatti **StartGoToPosition()** asserisce un enable, che è una variabile globale che abilita la modalità automatica.
Il sensore è di tipo magnetico e dal datasheet si ricava che il feedback è una retta con un'escursione da un valore minimo di 0,5 V ad un massimo di 4,5 V. Non esistono indicazioni sul verso di rotazione. Dal datasheet dell'attuatore

si ricava che lo spostamento di 1 mm è uguale a 71,55 mV letti dal sensore. Grazie a questo dato è possibile mettere in relazione lo spostamento di un millimetro dell'attuatore con la lettura letta dall'ADC. Infatti un LSB è 1,22 mV dividendo 71,55 mV si trova un valore letto quantizzato uguale a 58,61 che arrotondato è 59. Questo valore è stato usato nella funzione per il controllo della posizione che verrà spiegata di seguito ma prima di procedere è utile dare un ulteriore spiegazione del funzionamento del sensore e dei problemi riscontrati. Infatti, sebbene il feedback è una retta, al momento del montaggio della coppia sensore-magnete è impossibile conoscere l'accoppiamento del campo magnetico e il risultato è che ci si può trovare in qualsiasi punto della retta. Un altro problema è il cambio di polarità del campo magnetico che causa un salto della retta dal valore minimo a quello massimo o viceversa. Si deve quindi prevedere questa discontinuità e come gestirla nel modo ottimale. Per fare questo si è deciso di attuare diverse tecniche che hanno portato al risultato voluto. La prima riportata nel codice 4.7 riguarda l'effettuare una lettura all'accensione del sistema ed usare il valore letto come offset e punto di zero dell'attuatore.

```
void OffsetValueMelexis()
{
  old_position_value = (sint16t)GetMelexisValue();
  actual_position_mm = 0;
}
```

**Listing 4.7:** Sensor management file - Funzione offset

In questo modo è stato risolto il problema dell'accoppiamento della coppia sensore-magnete tramite una delle soluzioni più semplici e veloci da attuare. Altre soluzioni più complesse potrebbero integrare anche la lettura di corrente del chip ed andare a trovare i due estremi dell'attuatore, totalmente aperto e totalmente chiuso. Una volta trovati gli estremi, si conosce la corsa dell'attuatore e si può quindi impostare una posizione di zero conoscendo anche il valore di feedback del sensore.
Il problema della discontinuità è stato risolto attraverso l'uso di una funzione che effettua il modulo e delle informazioni sul verso di rotazione del motore noto, in quanto imposto dallo stesso microcontrollore. Il codice 4.8 riporta la funzione.

```
void UpdateActualPosition(e_Direction_t direction)
{
  sint16t sdeltaV;
  uint16t dV;

  new_position_value = (sint16t)GetMelexisValue();

    if ((new_position_value < old_position_value) && (direction ==
    FORWARD_DIR))
  {
    sdeltaV = VMAX_MELEXIS - old_position_value + new_position_value;
```

```
226   }
      else if((new_position_value > old_position_value) && (direction ==
        BACKWARD_DIR))
228   {
        sdeltaV = old_position_value + VMAX_MELEXIS - new_position_value;
230   }
      else
232   {
        sdeltaV = old_position_value - new_position_value;
234   }

236   dV += myABS(sdeltaV);

238   if(dV >= SCREW_PITCH)
      {
240     if(direction == FORWARD_DIR)
        {
242       actual_position_mm++;
        }
244     else if (direction == BACKWARD_DIR)
        {
246       actual_position_mm--;
        }
248     else
        {
250       /* do nothing we are in the case of BRAKE_TO_GND*/
        }
252     dV -= SCREW_PITCH;
        old_position_value = new_position_value;
254   }
}
```

**Listing 4.8:** Sensor management file - Funzione acquisizione posizione attuatore

Come prima cosa si acquisisce la nuova posizione, poi attraverso l'uso di istruzioni *if* si discrimina il caso particolare in cui ci si trova andando ad ovviare al problema della discontinuità. In base a queste informazioni, si calcola un differenza tra il vecchio valore di posizione e quello nuovo. Il secondo step è quello di andare effettuare l'operazione di modulo attraverso una funzione scritta all'interno del file. Successivamente, sempre con l'uso di *if* si controlla in un primo momento se la differenza è maggiore di un millimetro. In caso affermativo in base alla direzione, ricevuta come argomento della funzione, si aumenta o diminuisce un contatore che rappresenta la posizione attuale in millimetri. Prima di uscire dal corpo dell'istruzione si sottrae il valore quantizzato di un millimetro dalla differenza calcolata prima. Questo è stato fatto per non aggiungere un ulteriore arrotondamento al valore calcolato. Si pensi infatti che il nostro attuatore, durante il ciclo precedente si sia mosso di meno più di un millimetro: in questo caso la condizione per entrare nell'istruzione *if* non è soddisfatta. Il ciclo successivo l'attuatore si trova ad avere uno spostamento di più di un millimetro e nel caso peggiore quasi 2 mm, e poichè la condizione è più che rispettata, posso fare l'update della posizione andando a controllare la direzione. In questo caso, è necessario evitare

di perdere l'informazione relativa al movimento maggiore di un millimetro. Questo viene fatto grazie all'istruzione a riga 254 che elimina la parte già contata e lascia quella ancora da contare. In conclusione, si imposta il valore letto deall'ADC per cui la condizione è risultata vera.
La funzione appena descritta viene richiamata da un'altra funzione sullo stesso livello gerarchico che è eseguita ciclicamente all'interno del task veloce, ed è **PositionbyCAN()** con il codice 4.9.

```
112 void PositionbyCAN(uint16t dultycycle, uint16t given_position)
{
114   uint16t position_value;

116   if(gotoposition)
  {
118     position_value = GetActualPosition();

120     if(position_value < (given_position - RANGE))
    {
122       if(old_direction == BACKWARD_DIR)
      {
124         new_direction = BRAKE_TO_GND;
      }
126       else
      {
128         new_direction = FORWARD_DIR;
      }
130   }
    else if(position_value > (given_position + RANGE))
132   {
      if(old_direction == FORWARD_DIR)
134     {
        new_direction = BRAKE_TO_GND;
136     }
      else
138     {
        new_direction = BACKWARD_DIR;
140     }
    }
142   else
    {
144     new_direction = BRAKE_TO_GND;
      gotoposition = 0;
146   }

148   if(old_direction != new_direction)
    {
150     if(new_direction == BRAKE_TO_GND)
      {
152       DirectionbyCAN(BRAKE_TO_GND);
        PWM_Disable(FTM_0_BRIDGE_PWM);
154     }
      else
156     {
        DirectionbyCAN(new_direction);
158       PWM_Enable(FTM_0_BRIDGE_PWM);
      }
160     old_direction = new_direction;
    }
162
```

```
    UpdateActualPosition(new_direction);
164   }
}
```

**Listing 4.9:** Sensor management file - Funzione motore automatica

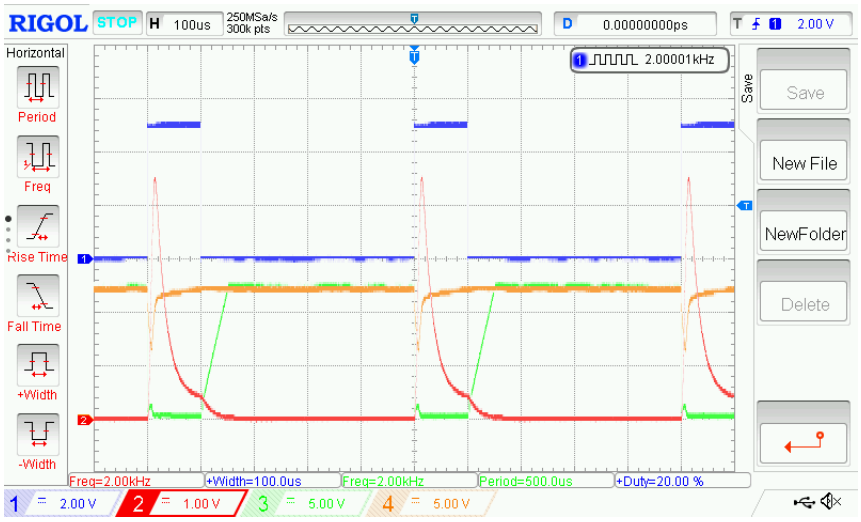
Come argomenti riceve il duty cycle e la posizione da raggiungere che sono dati attraverso messaggio CAN. Se l'enable è asserito, allora in base alla posizione calcolata dalla funzione presentata in precedenza, viene deciso tramite istruzioni *if* in quale caso entrare. In questa modalità automatica si è voluto aggiungere un piccolo range corrispondente a 2 mm per semplicità del controllo. Come osservato per la modalità manuale, anche qui, prima di invertire il senso di rotazione si passa per uno stato sicuro in modo da evitare blocchi meccanici. Una volta raggiunta la posizione prestabilita viene eseguito un'operazione di freno del motore e viene disabilitato l'enable alla funzione stessa.

## 4.4 Risultati test automatici

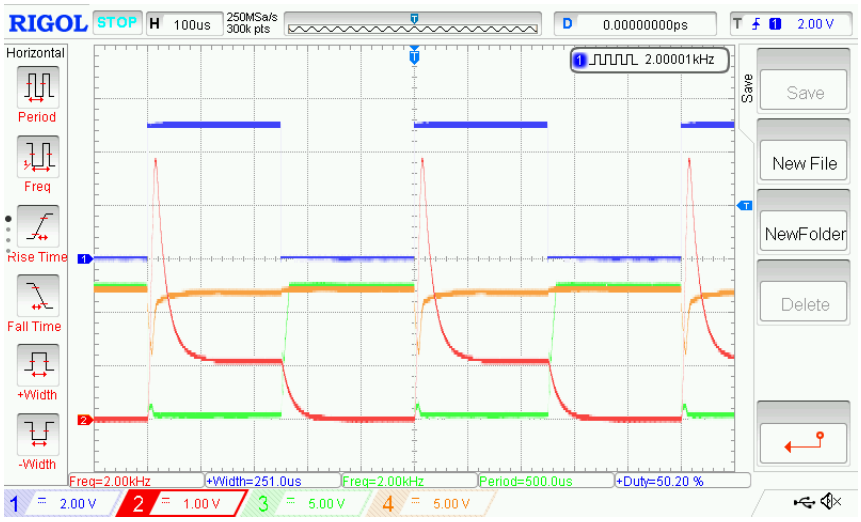
Per eseguire questo tipo test è stato allestito un banco di prova con un attuatore messo in movimento da un motore, dove è stato posto in serie anche il carico elettronico. In questo modo è stato possibile variare la corrente erogata dal chip impostando le soglie. Sono stati eseguiti test a frequenze diverse e duty cycle diversi ottenendo i risultati voluti riportati di seguito. La figura 4.1 riporta l'acquisizione della schermata dell'oscilloscopio dove è stata mantenuta la stessa frequenza di PWM di 2 kHz, andando a cambiare però il duty cycle dal 20% al 50%. La legenda dei canali dell'oscilloscopio è la stessa per tutti i risultati presentati:

- canale 1 (blu): Segnale del PWM in ingresso al chip;
- canale 2 (rosso): segnale del piedino multisense;
- canale 3 (verde): tensione presente su un uscita del ponte ad H;
- canale 4 (arancio): tensione di alimentazione $V_{cc}$.

Come si può osservare dalla figura 4.1,è evidente un picco all'accensione del mosfet all'interno del chip. Le differenze che si possono evidenziare tra le due immagini sono il comportamento della corrente, infatti con un duty cycle maggiore si nota una fase dove questa resta costante ed il suo valore è di 3 A. In aggiunta, la tensione sulla fase del motore quando si ha la parte spenta di PWM presenta una crescita lineare. La corrente, durante questa fase, decresce fino a portarsi ad un valore di 0 A. In figura 4.2 invece si vuol




**(a)**




**(b)**

**Figura 4.1:** Scheda VNH7040 frequenza 2 kHz duty cycle differenti

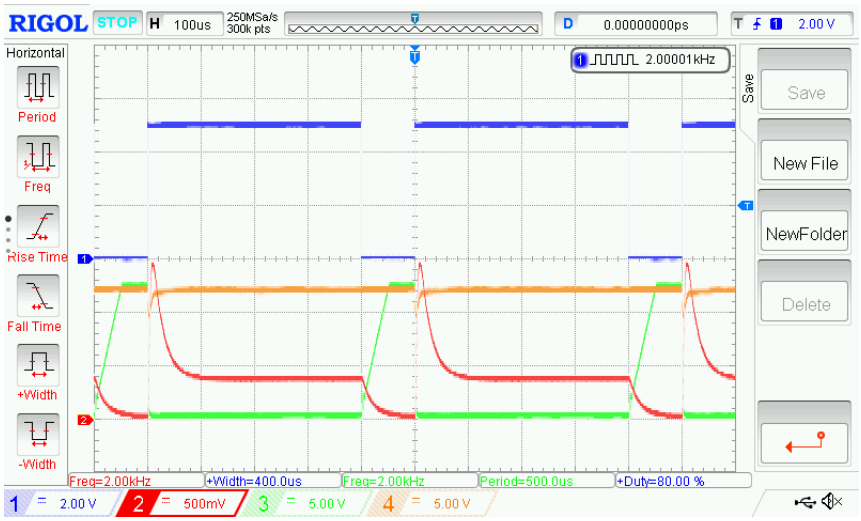



**Figura 4.2:** Scheda VNH7040 frequenza 2 kHz duty cycle 80%

riportare il comportamento del chip per valori di duty cycle superiori al 50 %, per l'esattezza 80 % di duty cycle e sempre 3 A di corrente. In questo caso, il picco di corrente è ridotto rispetto le precedenti configurazioni e la parte in cui la corrente rimane stabile è più lunga. Diversamente, si ha una fase decrescente della corrente quando il segnale PWM è allo 0 logico più corta.
Durante queste prove è stato scoperto il limite fisico della scheda prodotta per i test e questo riguarda la capacità di dissipazione del calore prodotto. Infatti, dopo avere ripetuto i test precedenti ad una frequenza non standard per il controllo, si è deciso di aumentare la frequenza del segnale PWM a 10 kHz, mentre il duty cycle è stato portato al 50% come si può vedere in figura 4.3. Il limite di corrente raggiunto in questa occasione è stato di 6 A ma dopo pochi minuti il chip ha attivato la sua protezione interna. Sono state eseguite ulteriori prove con l'ausilio di sonde di tempertatura collegate a dei multimetri digitali portatili. Le misure fatte con la sonda di temperatura sono state di 75°C per quanto riguarda il case esterno, mentre per la parte sottostante del chip è stata rilevata una misura di 65°C. La prova è stata ripetuta solo alcune volte, in quanto non si è ritenuto necessario indagare più approfonditamente rischiando la distruzione del componente. Si può affermare che di questo risultato è stata tenuta in considerazione la progettazione del prototipo da sviluppare, infatti si è optato per riservare una maggiore area di dissipazione ed anche un maggior numero di *thermal via* sulla scheda. Tale aspetto verrà approfondito nel capitolo successivo, relativo alla progettazione della scheda vera e propria.

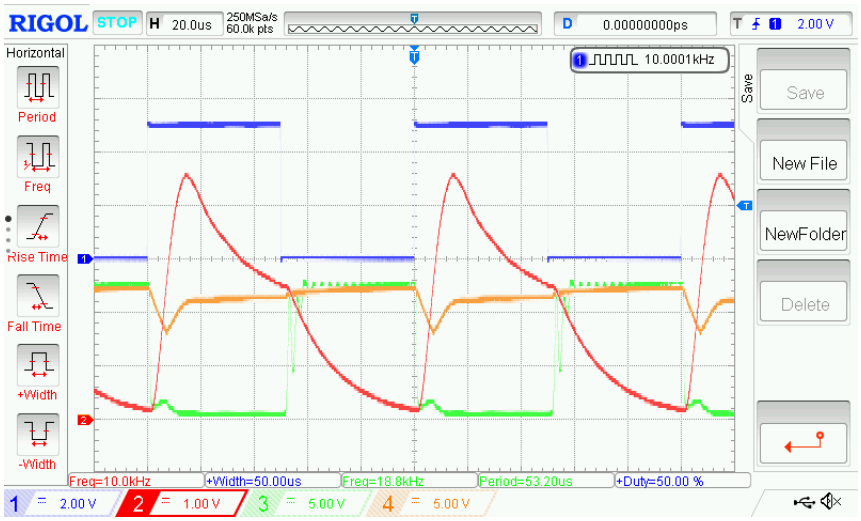



**Figura 4.3:** Scheda VNH7040 frequenza 10 kHz duty cycle 50%

In conclusione, osservando la figura 4.3, si può notare come il picco di corrente del motore risulti con una larghezza simile alla parte positiva dell'onda quadra. Inoltre, la corrente, a causa dell'aumento della frequenza, non si porta ad un valore stabile come visto nelle situazioni precedenti.

# Capitolo 5

# Progettazione scheda

In questo capitolo si vogliono giustificare le scelte progettuali adottate per la realizzazione hardware della scheda, il quale schema a macro blocchi è riportato in figura 5.1. Ciascun blocco sarà approfondito nel dettaglio durante la trattazione di questo capitolo. Lo schema ha costituito la prima fase di lavoro al fine di definire alcune specifiche in modo più chiaro e comprensibile. Ad esempio, i segnali sullo schema oltre che riportare un nome, usato poi anche nello schema elettrico vero e proprio, presentano colori diversi:

- nero: segnali digitali, PWM;
- giallo: tensione di alimentazione a 5 V;
- rosso: tensione di alimentazione a 12 V;
- verde: segnali di tipo analogico;
- blu: segnale di GND.

La stesura di questo schema, eseguito con una codifica di colori, è stata richiesta dall'azienda per ottenere una documentazione più chiara anche per altri colleghi. Grazie a questa tecnica, qualora in futuro si volesse decidere se usare o no il progetto risulterebbero subito chiare le funzionalità generali della scheda.
Alcuni componenti hardware che verranno presentati sono stati selezionati in quanto già presenti nel magazzino dell'azienda. Questa scelta è stata fatta assieme al tutor aziendale per non andare a creare troppa diversificazione nei prodotti da acquistare. Il capito in questione sarà suddiviso in modo eguale ai macro blocchi. Partendo dal basso verso l'alto dello schema.

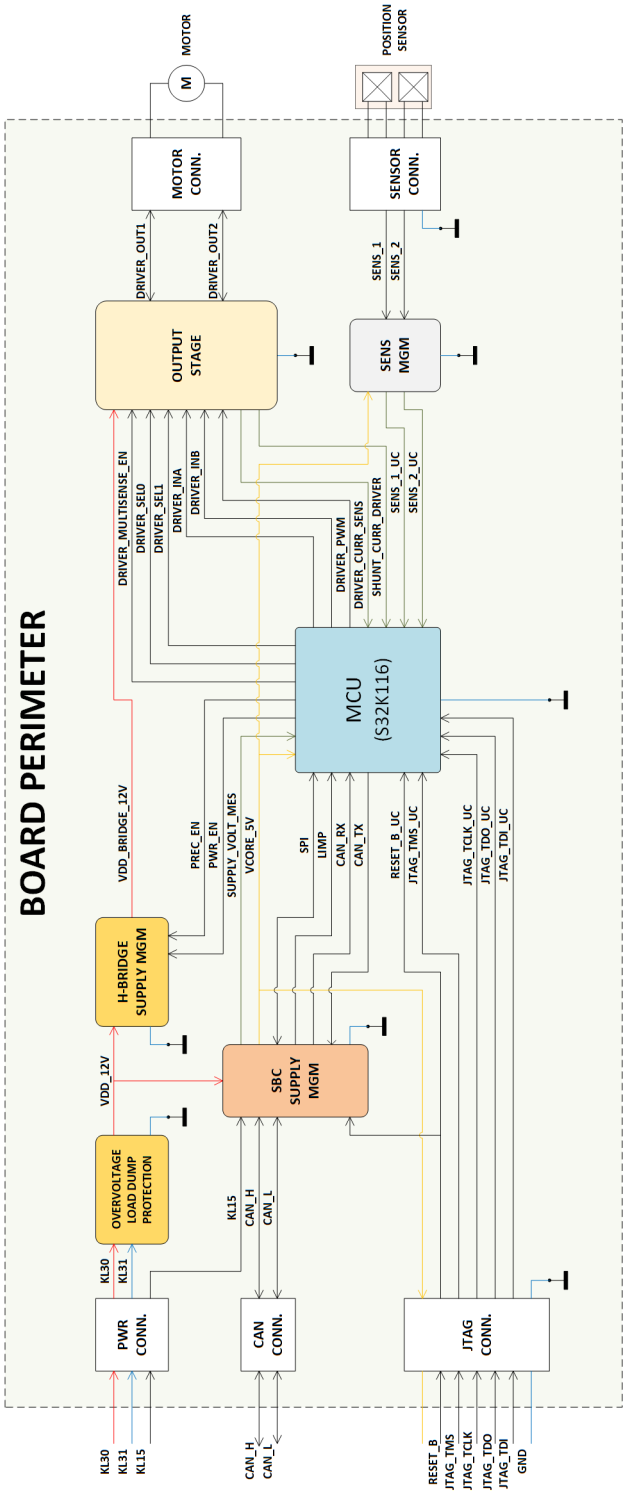



**Figura 5.1:** Schema a blocchi scheda progettata

## 5.1 Gestione tensioni d'alimentazioni

Con questo paragrafo si vuole racchiudere la parte di gestione delle alimentazioni della scheda, partendo dalla gestione della sovratensione e prova di *load dump*. Successivamente si passerà al blocco che si occupa della gestione dell'alimentazione del ponte ed infine al blocco che si occupa della gestione dell'alimentazione del microcontrollore.

### 5.1.1 Blocco overvoltage and load dump

Il blocco overvoltage and load dump idealmente rappresenta la parte dello schema che si occupa di fornire le protezioni necessarie alla scheda qualora ci siano problemi sull'alimentazione esterna. Si vogliono ricordare i parametri dei test effettuati nell'ambito *automotive* per certificare l'idoneità della stessa scheda.
**Protezione sovratensione**, la scheda deve essere in grado di sopportare una tensione di 24 V per un'ora di tempo senza subire danni. Inoltre, deve poter lavorare in modo corretto fino ad un massimo di 18 V.
**Protezione da load dumb**, questo test ipotizza che la batteria sia staccata mentre un generatore la sta caricando, quindi provocando un impulso che si ripercuote sulla scheda. Ci sono diversi tipi di impulso divisi in categorie, le cui differenze riguardano la durata e l'ampiezza dell'impulso.
**Protezione per l'anti inversione** della tensione di alimentazione, questo guasto accorre quando il connettore della batteria viene collegato in modo errato ed i poli vengono invertiti.
Per le prime due protezioni è stato usato un diodo di tipo *Transient Voltage Suppression* (TVS) bidirezionale subito dopo il connettore usato per l'alimentazione. In figura 5.2 viene riportata la sua transcaratteristica. Si vogliono evidenziare in particolare due valori che incidono sulla scelta di questo componente:

- $V_{BR}$ tensione di breakdown;
- $V_C$ tensione di clamping.

La prima tensione ($V_{BR}$) è la massima tensione che si ha ai capi del diodo per una corrente di valore uguale a $I_T$. Inoltre, è la soglia che indica l'inizio della regione di breakdown. La seconda tensione ($V_C$) è la tensione misurata quando un corrente di picco $I_{pp}$ scorre all'interno del diodo. Per il diodo scelto, questi due valori sono:

- $V_{BR}$ = 33 - 36 V @ $I_T$ = 1 mA;

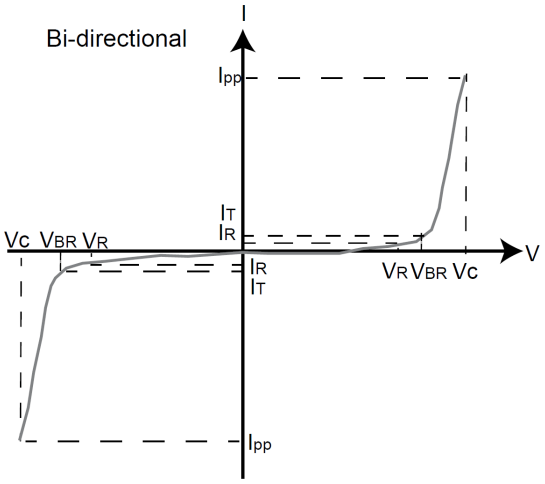



**Figura 5.2:** Transcaratteristica diodo TVS bidirezionale

- $V_C = 48$ V @ $I_{pp} = 12{,}5$ A.

Le caratteristiche del componente scelto sono state dettate dal fatto che viene utilizzato da altri progetti, di conseguenza è già disponibile in magazzino.
Si è scelto di utilizzare questa tipologia di diodi (TVS) rispetto a quelli di tipologia *Zener* dopo aver studiato delle comparazioni tra i due tipi di diodo. Si può affermare che i TVS sono costruiti con una tecnologia che permette la gestione, in un tempo breve, di una grossa potenza. Infatti, questi si attivano e vanno a limitare il picco di corrente e tensione come si può osservare in figura 5.3. Contrariamente, gli *Zener* sono ottimi componenti se usati per stabilizzare la tensione ai loro capi, tuttavia presentano delle dinamiche di innesco molto più lente. In aggiunta, i diodi *Zener* presentano una gestione della potenza dissipata minore rispetto i diodi TVS e quindi non sono adatti a dissipare la potenza dei picchi usati come riferimento nei test automotive.

Per la protezione da anti inversione della tensione di alimentazione sono stati studiati diversi metodi che verranno presentati di seguito, tra questi ne sono stati selezionati due. Prima di introdurli, è doveroso fare una premessa riguardante il motivo per cui ne sono stati usati due differenti anzichè di uno unico per l'intera scheda. La ragione di questa scelta è dovuta al fatto che la scheda presenta un unico connettore per quanto riguarda l'alimentazione, ma si ha la necessità di sdoppiare la traccia di $V_{cc}$ in due tracce distinte. Una prima traccia è più larga e con capacità di conduzione della corrente maggiore per alimentare il ponte ad H. Una seconda traccia è più stretta ed è usata per alimentare il chip *System Basic Chip* (SBC) che fornisce poi l'ali-

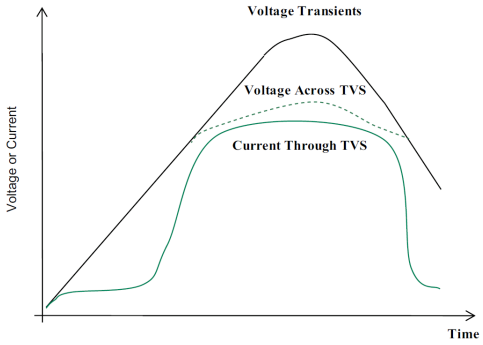



**Figura 5.3:** Gestione picco tensione diodo TVS

**Tabella 5.1:** Protezioni anti inversione alimentazione

| NOME PROTEZIONE | TIPOLOGIA | COLLEGAMENTO |
|---|---|---|
| Schottky Diode | Passiva | $V_{cc}$ |
| MOSFET a canale P | Attiva | $V_{cc}$ |
| Reverse FET | Attiva | $V_{cc}$ |
| MOSFET a canale N | Attiva | GND |

mentazione a 5 V. La distinzione delle due tracce è utile anche per diminuire i disturbi generati dal ponte ad H sulle altre alimentazioni. Questo modo di operare sulle tracce viene detto anche alimentazione a stella in quanto ogni traccia parte dal connettore. Data proprio la natura differente delle due tracce, adottare lo stesso metodo di protezione avrebbe comportato un impiego di area maggiore unito anche ad un costo più elevato. In tabella 5.1 sono riportati i nomi e la tipologia della protezione, unito anche alla connessione.

### Schottky Diode

Questa protezione fa uso di un diodo *Schottky* messo in serie sulla linea di alimentazione. Di conseguenza, tutta la corrente di alimentazione scorre attraverso il diodo che deve essere scelto in modo che la temperatura di giunzione non superi quella massima ricava come dato dal datasheet. Grazie alle formule scritte di seguito, la temperatura che viene raggiunta dalla giunzione può essere stimata per il valore di corrente che scorre all'interno del diodo.

$$P = V_{TO} \cdot I_{F(AV)} + rd \cdot {I^2}_{F(RMS)}$$

I valori della tensione $V_{TO}$ e della resistenza al piccolo segnale del diodo rd vendono estratti dal datasheet del componente. Invece, i valori della corrente media $I_{F(AV)}$ e quelli del valore efficace della corrente $I_{F(RMS)}$ sono dati da specifiche del progetto. Una volta calcolata la potenza dissipata, si può calcolare anche la temperatura che raggiunge la giunzione.

$$T_{j(D)max} = T_{amb,max} + P \cdot R_{thj-amb}$$

Il valore della temperatura ambiente massima $T_{amb,max}$ viene definito dalle specifiche di progetto mentre il valore della resistenza termica $R_{thj\text{-}amb}$ è stimato e ricavato dal datasheet del componente.
La scelta di questa tipologia di diodo è dettata proprio dal fatto di dover mantenere bassa la potenza. Infatti, se si osserva l'espressione della potenza, la tensione di soglia incide linearmente ed occorre avere un valore basso. I diodi *Schottky* hanno proprio la caratteristica di avere questo valore di soglia più basso rispetto ad altre tipologie di diodo costruiti con tecnologia differente.
Il comportamento durante l'inversione della tensione è tale per cui il diodo rimane spento e l'unica corrente che scorre è quella di leakage. Tuttavia, bisogna fare attenzione alla tensione di breackdown che deve essere maggiore del valore inverso di tensione della batteria tipicamente -16 V.
In conclusione, questo tipo di protezione è molto economico e occupa poca area, ma non può essere adottato per alte correnti per problemi di dissipazione.

### MOSFET a canale P

In questa tipologia viene usato un mosfet a canale p assieme ad altri componenti per creare la protezione e il relativo schema è riportato in figura 5.4. Durante il funzionamento normale, dopo la fase di *power-up* dove il *body diode* è polarizzato correttamente, una tensione negativa sul *gate-source* accende il mosfet. Invece, durante l'inversione della tensione, il mosfet si spegne in quanto la tensione *gate-source* risulta positiva grazie alla caduta di tensione sul diodo Zener. In questo stato, la corrente non può scorrere e così facendo si vanno a proteggere tutti i componenti a valle. Dato che questa soluzione prevede l'uso di più componenti, questi debbono esser dimensionati seguendo alcuni parametri.
In particolare, il diodo Zener deve essere dimensionato seguendo 2 principi base:

- La tensione di Zener $V_Z$ deve essere maggiore della tensione nominale della batteria in modo che si eviti così un passaggio elevato di corrente durante la fase di standby;

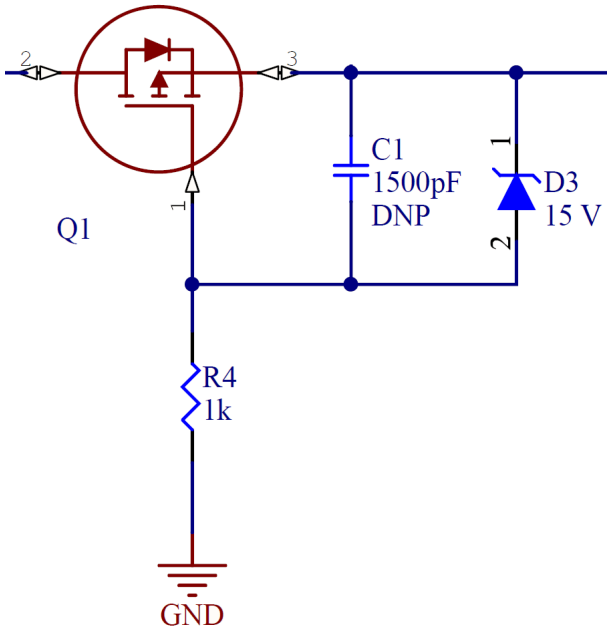



**Figura 5.4:** Schema elettrico protezione con mosfet a canale p

- La tensione di Zener $V_Z$ deve essere minore della tensione massima di *gate-source* del mosfet di tipo p. Così facendo si viene protetti da possibili sovratensioni.

Di solito questa tensione per progetti automotive a 12 V di tensione di alimentazione è di $V_Z = 16$ V.
Per quanto riguarda la resistenza posta tra gate e GND, questa è usata per limitare la corrente che scorre attraverso il diodo Zener quando la tensione di alimentazione supera la tensione di Zener. Inoltre, limita la corrente di carica/scarica del gate. Di conseguenza, la resistenza insieme alla capacità del gate del mosfet determina il tempo di *turn-off* quando si hanno dei picchi negativi di tensione o l'inversione della stessa. Un valore tipico di questa resistenza è 1 kΩ e risulta un buon compromesso tra la minimizzazione della corrente di gate ed un veloce tempo di spegnimento del mosfet.
Per ciò che concerne il mosfet di tipo P, questo deve essere scelto in modo tale da riuscire a sopportare la corrente richiesta da specifiche, senza incorrere in un guasto. Alcuni parametri principali da considerare sono $R_{ds(on)}$, $V_{(BR)DSS}$ e $V_{gs\_th}$. In particolare, la tensione di soglia è data in base alla tensione minima della batteria. Invece, la resistenza di on dovrebbe essere il più bassa possibile per avere una minore dissipazione di potenza, che di conseguenza porta ad avere una temperatura sulla giunzione più bassa della temperatura massima consentita. La tensione di breakdown $V_{(BR)DSS}$ del mosfet deve es-

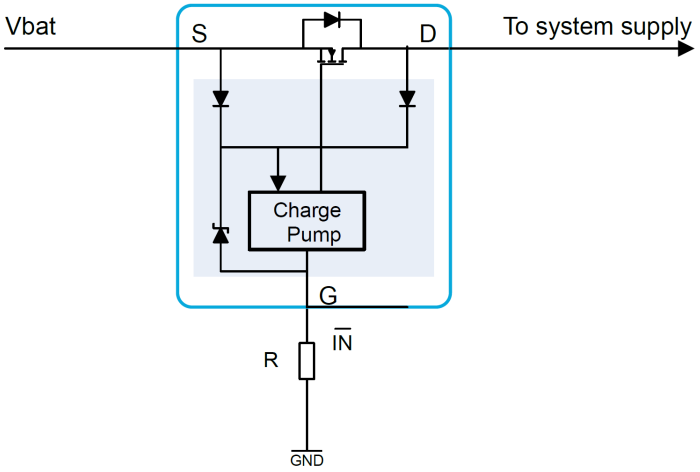



**Figura 5.5:** Schema interno reverse FET

sere maggiore della tensione inversa di batteria.
Per migliorare il comportamento di questa protezione verso degli impulsi negativi veloci, può essere posto un condensatore tra il gate ed il source del mosfet. Così facendo, si va a creare un filtro RC assieme alla resistenza descritta in precedenza, che se dimensionati propriamente rendono il dispositivo trasparente agli impulsi veloci. Questo permette di non andare a spegnere il mosfet, aprendolo e quindi di continuare a fornire corrente al carico che sarebbe altrimenti alimentato da eventuali condensatori usati come filtro sulla tensione di alimentazione.

### Reverse FET

Questa soluzione è formata da un chip che integra al suo interno un mosfet di tipo n e un circuito di driver per il mosfet. In figura 5.5 è riportato lo schema interno. Il chip presenta 2 pin per la potenza (drain, source) e un pin per il controllo ($\overline{IN}$). Quando quest'ultimo pin è portato a valore logico 0 allora la pompa di carica interna permette l'accensione del mosfet andando ad alzare la tensione del gate sopra alla tensione di batteria. Se il pin invece è lasciato flottante, il mosfet resta spento e si comporta solo come un diodo tra i pin di drain e source. In condizioni di normalità, questo chip deve essere in grado di dissipare una potenza data dalla moltiplicazione della resistenza di on per la corrente che gli scorre all'interno.
Durante l'evento di inversione della tensione, la pompa di carica smette di operare e questo causa uno spegnimento del mosfet.

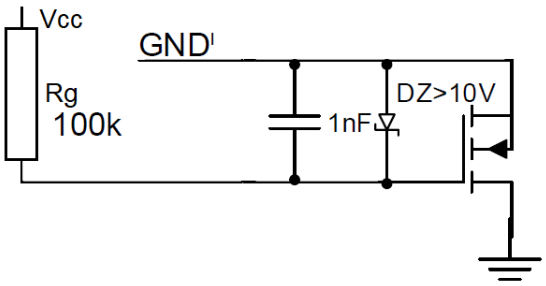



**Figura 5.6:** Schema elettrico protezione con mosfet a canale n

### MOSFET a canale N

Per creare la protezione da anti inversione si usa un mosfet n unito ad altri componenti esterni.In figura 5.6 si può osservare lo schema elettrico. Durante il funzionamento normale il *body diode* del mosfet n provvede il riferimento di GND al chip da proteggere. Il diodo Zener, assieme alla resistenza genera abbastanza tensione di *gate-source* per accendere il mosfet.
Quando accade la condizione di inversione della polarità della batteria, il mosfet si spegne in quanto la tensione sul *gate* è bassa. La resistenza $R_g$ limita il passaggio di corrente attraverso il diodo Zener quando la tensione di alimentazione supera la tensione di breakdown del diodo. Inoltre, la resistenza limita anche carica e scarica del *gate* ed insieme alla capacità del *gate* determina il tempo di switch-off del mosfet. La resistenza deve essere dimensionata in modo corretto, tenendo conto dell'aspetto descritto in precedenza, ma anche tenendo conto che una prolungata fase di inversione della tensione porta ad avere una elevata dissipazione. Alcuni valori tipici sono sulle decine di kΩ. Per quanto riguarda il condensatore, tra *gate-source* serve per essere trasparente agli impulsi negativi veloci come quanto visto per la protezione con mosfet di tipo p.
Anche in questa tipologia di protezione sono presenti componenti esterni che dovranno essere dimensionati. Partendo dal diodo Zener si possono fare le seguenti considerazioni:

- La tensione di Zener $V_Z$ deve essere maggiore della tensione nominale della batteria così che si eviti un passaggio elevato di corrente durante la fase di standby;

- La tensione di Zener $V_Z$ deve essere minore della tensione massima di *gate-source* del mosfet di tipo n. Così facendo si viene protetti da possibili sovratensioni.

Per quanto riguarda il mosfet, i parametri principali sono $R_{ds(on)}$, $V_{(BR)DSS}$ e $V_{th}$. Anche per il mosfet di tipo n valgono le linee guida date per il mosfet di tipo p, la resistenza di on molto piccola per contenere la dissipazione di potenza. Di conseguenza, limitare la temperatura di giunzione. La tensione di breakdown $V_{(BR)DSS}$ dovrebbe essere maggiore della tensione massima inversa che si applica alla scheda. Inoltre il mosfet deve essere in grado di gestire la potenza dell'impulso negativo.
Lo svantaggio di questo tipo di protezione è che non può essere condiviso con altri dispositivi ma solo con uno e in aggiunta si ha anche uno spostamento del riferimento di GND dovuto hai componenti posti tra la GND del componente e quella della scheda.

#### Conclusioni sulle protezioni

Alla fine della spiegazione delle varie protezioni si vuole riassumere quelle adottate, giustificandone il motivo. Per quanto riguarda la linea dell'SBC, la scelta migliore risulta quella di utilizzare un diodo *Schottky* per via della corrente bassa che scorre su quella determinata traccia. I vantaggi riguardano sicuramente l' area occupata e il costo molto contenuto del componente. Per quanto riguarda invece la linea che alimenta il ponte ad H, è stato deciso di adottare la protezione con mosfet di tipo p, perché la corrente che scorre al suo interno è troppo alta per un reverse FET. Mentre la protezione con mosfet di tipo n, oltre ad aggiungere un ulteriore offset al riferimento di GND data la presenza di uno shunt opzionale, avrebbe portato anche un ulteriore consumo nella fase di standby.

### 5.1.2 Blocco H-Bridge supply management

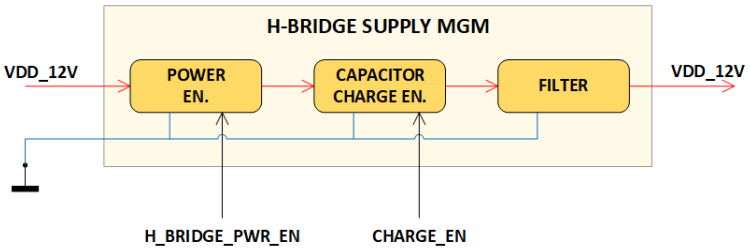



**Figura 5.7:** Schema a blocchi interno del macro blocco H-Bridge supply management

Questo macro blocco si occupa di gestire l'abilitazione e disabilitazione della tensione sul ponte ad H, della pre carica dei condensatori del filtro sull'alimentazione del ponte ad H ed include anche il filtro per i disturbi. La

scelta di una gestione di questo tipo è stata adottata dopo aver studiato i vari aspetti della scheda. Per quanto riguarda l'enable dell'alimentazione, al ponte serve per ragioni di sicurezza e per ragioni di consumi. La sicurezza è che quando si ha una sovratensione sull'alimentazione che viene letta dal microcontrollore allora si disattiva la tensione sul ponte ad H e lo si mette in sicurezza. Invece per quanto riguarda i consumi, sono state prese in considerazione alcuni aspetti e si è supposto che data la natura del progetto il motore non sia sempre azionato e quindi non debba essere sempre alimentato. Questo aspetto porta a valutare che, se si perdono pochi millisecondi per caricare i condensatori di filtro, all'inizio del comando del motore non risulta un problema. Al contrario, se non si adotta questo sistema, si avranno dei condensatori sempre collegati all'alimentazione con una loro corrente di leakage di certo più elevata rispetto a quella di un interruttore formato da un mosfet. In aggiunta, si avrebbe anche la corrente di perdita dell'integrato che resterebbe sempre alimentato e anche la dissipazione di potenza da parte della resistenza di pre carica.
L'abilitazione e la disabilitazione della pre carica dei condensatori serve per non avere picchi di corrente quando viene abilitata l'alimentazione del ponte ad H. Per realizzare ciò si usa una resistenza, che limita la corrente, in parallelo ad un mosfet sfruttato come interruttore. Quando il mosfet è spento, la corrente scorre all'interno della resistenza limitandola, ma questo non potrebbe essere la condizione di normalità in quanto la resistenza presenta una dissipazione. Di conseguenza, durante il funzionamento normale del ponte ad H il mosfet viene acceso e dato che la corrente predilige dei percorsi con una bassa resistenza si avrà che questa scorre attraverso il mosfet riducendo la dissipazione, il che comporta la possibilità di utilizzare correnti più elevate. Il filtro è posto per poter essere conformi alle norme ISO riguardo ai disturbi elettromagnetici oltre che per il corretto funzionamento del chip stesso. Data la sua complessità, questo verrà trattato in un paragrafo apposito.

### Power enable e Capacitor charge enable

Data la somiglianza dei due stadi, è utile fare un' unica trattazione per quel che concerne il funzionamento e la scelta dei componenti. La differenza tra i due stadi è di una resistenza messa in parallelo al mosfet di tipo p nello stage della pre carica. Lo schema elettrico è riportato in figura 5.8 e riporta i due stadi collegati in serie tra di loro. Uno stadio è composto da due mosfet, uno di tipo p di potenza ed un altro di tipo n di segnale. A questi due componenti si aggiungono anche delle resistenze, un condensatore opzionale ed un diodo Zener. La logica di funzionamento è basilare: quando si vuole abilitare il

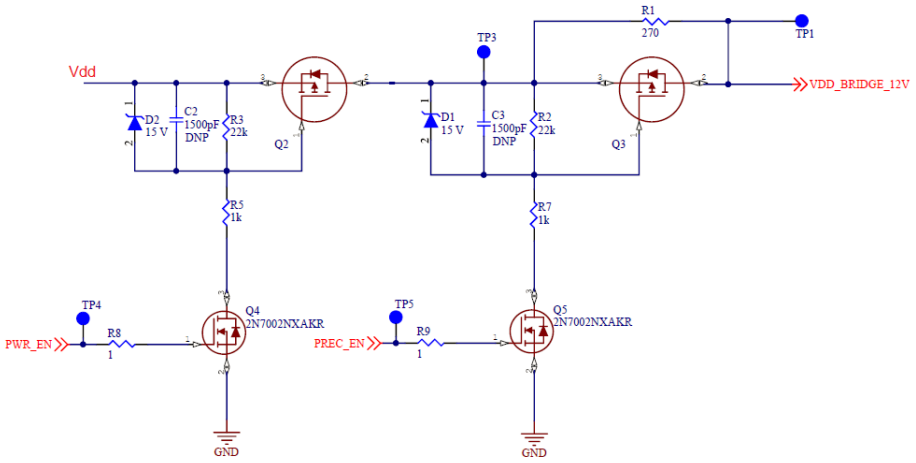



**Figura 5.8:** Schema elettrico abilitazione alimentazione e pre carica condensatori

mosfet di tipo p e quindi andare a far passare corrente attraverso si deve accende il mosfet di tipo n. Questo mosfet di tipo n è un formato piccolo e poco ingombrante in quanto al suo interno non scorre una corrente elevata. Una resistenza opzionale è stata posta in serie al gate per limitare la corrente che scorre nel ramo. Sul drain del mosfet di tipo n è collegata una serie di resistenze che fungono da partitore di tensione per la tensione di *gate-source* del mosfet di tipo p. Quando il mosfet n è acceso, sapendo che la corrente che scorre al suo interno è nell'ordine dei $\mu$A, la caduta di tensione tra drain e source risulta molto bassa e trascurabile. Quindi è necessario ricavare dalla tensione di alimentazione la tensione di soglia per accendere il mosfet di tipo p e questo è realizzato attraverso il partitore di tensione creato con le due resistenze. Per proteggere il gate del mosfet di tipo p dalle sovra tensioni, è stato posto in parallelo tra *source-gate* un diodo Zener. Inoltre, questo diodo ha un ulteriore scopo quello di mantenere attivo il mosfet sempre nel caso di sovra tensioni. Da notare che la resistenza presente tra nodo del gate del mosfet di tipo p - anodo diodo Zener e drain del mosfet di tipo n è usata per limitare la corrente quando il diodo Zener lavora in regione di breakdown. Il condensatore è stato predisposto per poter intervenire nel caso nascessero problemi di disturbo elettromagnetico in fase di test.
L'unica differenza presente nello stage di pre carica dei condensatori è una resistenza posta in parallelo tra il drain ed il source del mosfet di tipo p. Questa è usata quando il mosfet di tipo p è spento, e limita la corrente che scorre all'interno dei condensatori quando questi sono scarichi. Operando in questo modo si evitano picchi di assorbimento della corrente nella fase di accensione del mosfet.

### 5.1.3 Filtro alimentazione chip H-Bridge

Risulta utile introdurre alcune considerazioni sull'applicazione del ponte ad H. Con questo chip si vuole azionare un motore DC *brushed* e il controllo deve far fronte agli effetti delle induttanze parassite dovute alle alte correnti usate ed alle alte frequenze di commutazione. Una gestione inappropriata di queste induttanze può portare ad avere problemi di compatibilità elettromagnetica, dei comportamenti non desiderati e rottura del chip. Queste induttanze parassite sono introdotte dal cablaggio esterno della scheda, ma anche all'interno della scheda dalle tracce del PCB e dai *vias* usati. La commutazione di una corrente induttiva causa una sovra tensione sui terminali del dispositivo, questa tensione si può calcolare attraverso la formula 5.1.

$$V_L = -L \cdot dI_L/dt \tag{5.1}$$

Dove $V_L$ è la tensione ai capi dell'induttanza, L è il valore dell'induttanza e $I_L$ è la corrente che scorre al suo interno. Dalla formula si può constatare che ogni variazione alta della corrente porta ad avere una tensione sull'induttore. Inoltre, questo immagazzina una certa quantità di energia che viene rilasciata quando la corrente non scorre più. L'energia si dissipa su tutti i moduli collegati all'induttanza, per questo va gestita in modo da non avere rotture dei componenti. La formula 5.2 mostra l'espressione per il calcolo dell'energia di un'induttanza.

$$E_L = 0,5 \cdot L \cdot I_L^2 \tag{5.2}$$

Il primo componente di questo filtro sono due condensatori collegati in parallelo tra di loro e connessi tra l'alimentazione e GND. Il primo condensatore **C1($V_{CC}$)** è di tipo elettrolitico e serve per mantenere l'oscillazione (ripple) della tensione di alimentazione sui pin di $V_{CC}$ dell'integrato ad un valore prestabilito durante l'uso del PWM per azionare il motore. L'ampiezza picco picco del ripple per il progetto è stata considerata di 3 V. Il valore di capacità di questo condensatore dipende dalla tensione di alimentazione, dalla temperatura di esercizio. Un altro aspetto da considerare è la resistenza serie parassita (ESR) del condensatore che può degradare le prestazione del filtro. Il secondo condensatore **C2($V_{CC}$)** è di tipo ceramico e serve per assorbire le variazioni veloci tra $V_{CC}$ e GND. Il valore di questo condensatore è determinato dalle prove *CISPR25* sulle emissioni condotte sulla linea della batteria. Per entrambi i condensatori vale la regola di essere installati il più vicino possibile ai pin di $V_{CC}$ e GND in modo da evitare il più possibile induttanze parassite. Un modo per ovviare a questo problema è quello di usare dei piani di alimentazione e di massa quando si effettua il layout della scheda.
In figura 5.9 si può notare una raffigurazione del ripple sulla tensione di alimentazione dovuto al PWM e alla corrente del carico pilotato. Questo ripple

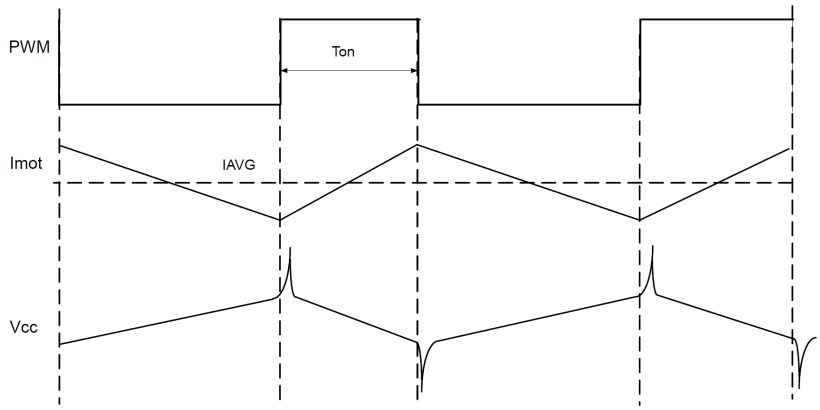



**Figura 5.9:** Rappresentazione del ripple della tensione in relazione al PWM

si può generare da diverse cause spiegate di seguito, dunque si devono studiare i vari casi per trovare quello che richiede un condensatore con la capacità maggiore ed essere sicuri dell'efficacia di questo condensatore.

**Ripple causato dall'induttanza del motore**, durante lo stato di funzionamento la corrente che scorre è la somma di due componenti:

- una corrente media $I_{AVG}$ fornita dalla batteria;
- una corrente di ripple $I_{Mripple}$ fornita dai condensatori posizionati vicino all'integrato.

Quindi la batteria fornisce l'energia quando si ha la fase on del periodo di PWM. Una supposizione fatta implicitamente durante questa trattazione è che la corrente media sia fornita dalla batteria. Nel caso peggiore, cioè quando i condensatori sono disaccopiati dalla linea di alimentazione a seguito di un induttanza parassita dei fili elevata, la potenza che i condensatori devono fornire è uguale alla potenza che il motore sta usando.
Per dimensionare il condensatore si è partiti dalle equazioni caratteristiche del motore DC:

$$V_{batt} = R_{motor} \cdot I_{avg} + L_{motor} \cdot \frac{dI_{motor}}{dt} + BEMF = V_{motor,avg} + V_{motor,ripple} \quad (5.3)$$

Dove:

$$V_{motor,avg} = R_{motor} \cdot I_{avg} + BEMF \quad (5.4)$$

Invece se si trascura la resistenza del motore $R_{motor}$ nell'equazione di partenza si ottiene che:

$$V_{motor,ripple} \cong L_{motor} \cdot \frac{dI_{motor}}{dt} \quad (5.5)$$

La corrente durante il ripple può essere semplificata secondo la seguente espressione:

$$I_{motor,ripple} = \frac{(V_{batt} - V_{motor,avg})}{L_{motor}} \cdot T_{on} \tag{5.6}$$

Dove:

$$V_{motor,avg} = \frac{T_{on}}{T} \cdot V_{batt} \tag{5.7}$$

A questo punto considerando un incremento lineare della corrente del motore, la carica richiesta dai condensatori per alimentare il motore è la seguente:

$$DQ = 0,5 \cdot T_{on} \cdot I_{motor,ripple} \tag{5.8}$$

Quindi la capacità del condensatore potrà essere ricavata dalla formula seguente:

$$C1_{(V_{CC})} \geq \frac{DQ}{DV_{CC}} \tag{5.9}$$

Definito il ripple che si vuole ottenere ed applicando le formule precedenti si ottiene la formula finale per il condensatore:

$$C1_{(V_{CC})} \geq \frac{0,5 \cdot T_{on}^2 \cdot V_{batt} \cdot (1 - \frac{T_{on}}{T})}{L_{motor} \cdot DV_{CC}} \tag{5.10}$$

**Ripple causato da variazioni di corrente sull'induttanza dei cavi**, durante le operazioni del motore con PWM la batteria è come se venisse attacca e staccata continuamente. In questo caso il ripple sulla tensione di alimentazione dipende:

- dall'induttanza tra i pin di alimentazione del chip e la batteria;
- dalla massima velocità con cui sono accesi e spenti i low side mosfet all'interno del chip;
- dalla massima corrente richiesta dal motore nel peggior caso possibile.

La variazione di corrente sull'induttanza parassita è:

$$DV_{CC} = L_{stray} \cdot \frac{dI_{motor}}{dt} \tag{5.11}$$

In questo caso $L_{stray}$ è l'induttanza tra $V_{batt}$ e i pin di $V_{CC}$ del dispositivo. Per quanto riguarda $dt$ si deve tenere in conto sia il tempo $t_{fall}$ che $t_{rise}$ del low side mosfet che commuta. In questo caso per dimensionare il condensatore di filtro si è imposto che l'energia contenuta al suo interno debba eguagliare l'energia immagazzinata dall'induttanza parassita. Se supponiamo che l'energia

dell'induttanza è trasferita istantaneamente al condensatore, trascurando il tempo di switching del mosfet low side, otteniamo la seguente equazione:

$$0,5 \cdot L_{stray} \cdot I_{motor}^2 = 0,5 \cdot C1_{(V_{CC})} \cdot DV_{CC}^2 \tag{5.12}$$

Assumendo che:

$$DV_{CC,max} = V_{CC\_AMR} - V_{batt\_max} \tag{5.13}$$

Dove con $V_{CC\_AMR}$ si intende il valore di massimo che può sopportare il dispositivo (*Absolute Maximum Rating*) ricavato dal datasheet. Questo porta ad avere che il condensatore di filtro debba avere un valore maggiore di quello calcolato con la seguente espressione:

$$C1_{(V_{CC})} \geq \frac{L_{stray} \cdot I_{motor}^2}{DV_{CC,max}^2} \tag{5.14}$$

**Ripple dovuto allo spegnimento della corrente a causa di un corto-circuito verso batteria o GND**. Il chip *VNH7040* presenta una corrente definita, all'interno del datasheet, di *shut-down* del mosfet loww side. Di conseguenza quando capita un cortocircuito il mosfet low side si spegne non appena raggiunge questa soglia di corrente. A causa del cortocircuito l'energia immagazzinata nell'induttanza parassita viene dissipata attraverso i dispositivi collegati ai capi. Per come è strutturato il ponte ad H, quando si presenta un impulso positivo sul drain del mosfet low side questo viene trasferito sui pin di $V_{CC}$ del componente. Quindi il condensatore di filtro aiuta ad assorbire questa energia. Partendo dal bilanciamento delle energie si ottiene l'espressione seguente:

$$0,5 \cdot L_{stray\_sc} \cdot I_{SD\_LS,max}^2 = 0,5 \cdot C1_{(V_{CC})} \cdot DV_{CC,max}^2 \tag{5.15}$$

Dove si definisce:

- $L_{stray\_sc}$ l'induttanza parassita durante il cortocircuito;
- $I_{SD\_LS,max}$ la corrente di *shut-down* del mosfet low side;
- $DV_{CC,max} = V_{CC,max} - V_{CC,initial}$ che rappresenta il valore massimo sopportabile dal chip.

Oltretutto, se si impone che $DV_{CC,max} = V_{AMR} - V_{batt,max}$, anche in questo caso $V_{AMR}$ è dato dal valore trovato su datasheet. Si ottiene che:

$$C1_{(V_{CC})} \geq \frac{L_{stray_sc} \cdot I_{SD\_LS,max}^2}{DV_{CC,max}^2} \tag{5.16}$$

Se si tengono in conto le stesse considerazioni per il caso di cortocircuito verso GND, si trova che è il mosfet high side ad avere una corrente limite, unita alla protezione termica. Entrambe le protezioni spengono il componente, ma nel caso peggiore la scarica dell'energia immagazzinata dall'induttanza parassita avviene alla massima corrente limite ($I_{lim,hs}$). La formula da considerare risulta:

$$0,5 \cdot L_{stray\_sc} \cdot I_{lim,hs}^2 = 0,5 \cdot C1_{(V_{CC})} \cdot DV_{CC,max}^2 \tag{5.17}$$

Disponendo $DV_{CC,max}$ come nel caso precedente troviamo:

$$C1_{(V_{CC})} \geq \frac{L_{stray\_sc} \cdot I_{lim,hs}^2}{DV_{CC,max}^2} \tag{5.18}$$

**Ripple dovuto ad una disconnessione della batteria quando il motore lavora al massimo regime**. Sotto queste condizioni l'energia immagazzinata all'interno del motore dovrà essere assorbita dal condensatore di filtro per evitare che il chip del ponte ad H raggiunga la tensione di breakdown con una possibile rottura dello stesso. Per calcolare l'energia del motore può essere usata l'espressione:

$$E_{motor} = 0,5 \cdot C1_{(V_{CC})} \cdot DV_{CC,max}^2 \tag{5.19}$$

Imponendo $DV_{CC,max}$ come nel caso precedente, la capacità risulta:

$$C1_{(V_{CC})} \geq \frac{E_{motor}}{0,5 \cdot DV_{CC,max}^2} \tag{5.20}$$

**Calo della $V_{CC}$ sotto la soglia di *undervoltage* a causa della corrente di *in-rush* del motore**. L'alimentazione del dispositivo è induttivamente disaccoppiata dal positivo della batteria della macchina normalmente. Questo è dovuto al fatto che la scheda è alimentata dall'induttanza dei cavi che la collegano alla batteri, inoltre come si introdurrà in seguito è presente un filtro di tipo *PI* per attenuare i disturbi. Allo start, la corrente del motore risulta molto più alta rispetto alla corrente nominale, questo succede a causa dell'inerzia iniziale del meccanismo meccanico. Dopo le premesse fatte, il condensatore dovrà avere una capacità tale da poter fornire almeno una parte dell'energia richiesta. Questo per evitare di avere una tensione di alimentazione troppo bassa che di conseguenza farebbe scattare la protezione interna di *under voltage* del chip. Partendo dalla formula:

$$dI_{motor,inrush} = C1_{(V_{CC})} \cdot \frac{DV_{CC}}{dt} \tag{5.21}$$

Imponendo $DV_{CC} = V_{batt,min} - V_{USD,max}$, dove $V_{USD,max}$ è la tensione massima per cui non scatta la protezione da sotto tensione, trovata tramite datasheet. Si ottiene:

$$C1_{(V_{CC})} \geq \frac{dI_{motor,inrush} \cdot dt}{V_{batt,min} - V_{USD,max}} \tag{5.22}$$

In conclusione quindi, per dimensionare questo componente la capacità deve soddisfare tutte le condizioni trovate dai vari casi descritti. Dopo aver fatto dei calcoli con i dati ricavati dal datasheet si è trovato che il valore della capacità del condensatore:

$$C1_{(V_{CC})} \geq 520\mu F \rightarrow 560\mu F \tag{5.23}$$

Il valore trovato è stato arrotondato a quello più vicino della serie *E12*.

### Filtro PI

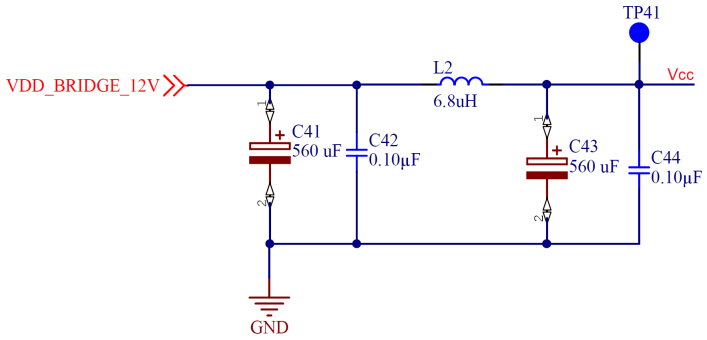



**Figura 5.10:** Schema elettrico filtro *PI*

Il solo condensatore $C1_{V_{CC}}$ potrebbe non essere sufficiente al fine di superare i test di compatibilità elettromagnetica a cui la scheda dovrà essere posta prima di entrare in commercio. A tale scopo, sfruttando il condensatore appena trovato ed aggiungendo un'induttanza $L_{(V_{CC})}$ ed un altro condensatore $C_{(V_{batt})}$, si crea così un filtro *PI*, dove in figura 5.10 è riportato lo schema elettrico. La frequenza di risonanza dei componenti $L_{(V_{CC})}$ e $C_{(V_{batt})}$ del filtro viene calcolata come:

$$fr = \frac{1}{2 \cdot \pi \cdot \sqrt{L_{(V_{CC})} \cdot C_{(V_{batt})}}}$$

La funzione di trasferimento invece risulta:

$$\frac{V_{out}}{V_{in}} = \frac{\frac{1}{j\omega C_{(V_{batt})}}}{j\omega L_{(V_{CC})} + \frac{1}{j\omega C_{(V_{batt})}}}$$

Dove $V_{in}$ è la tensione $V_{CC}$ ed invece $V_{out}$ è la tensione $V_{batt}$. Il filtro in questione è pensato per attenuare il disturbo generato dal ponte ad H mentre è acceso ed funziona con un segnale PWM. Dalla funzione di trasferimento si può ricavare l'attenuazione in $dB$:

$$Att_{dB} = 20 \cdot log \left| \frac{V_{out}}{V_{in}} \right| = 20 \cdot log \left| \frac{\frac{1}{j\omega C_{(V_{batt})}}}{j\omega L_{(V_{CC})} + \frac{1}{j\omega C_{(V_{batt})}}} \right|$$

Dal quale si ottiene:

$$Att_{dB} = 20 \cdot log \left| \frac{1}{1 - 4 \cdot \pi^2 \cdot f^2 \cdot L_{(V_{CC})} \cdot C_{(V_{batt})}} \right| \cong 20 \cdot log \left| \frac{1}{4 \cdot \pi^2 \cdot f^2 \cdot L_{(V_{CC})} \cdot C_{(V_{batt})}} \right|$$

Se vengono considerate delle frequenze per cui l'attenuazione è maggiore o uguale a 40dB in valore assoluto, può essere fatta la seguente semplificazione:

$$1 \ll 4 \cdot \pi^2 \cdot f^2 \cdot L_{(V_{CC})} \cdot C_{(V_{batt})} \tag{5.24}$$

Alcuni criteri da tenere in considerazione per scegliere l'induttanza sono il suo valore, la corrente che è in grado di sopportare, la resistenza serie parassita, la dissipazione di potenza che può sopportare ed infine lo spazio che occupa su scheda. La potenza dissipata da un'induttanza si può calcolare partendo dalla sua resistenza serie ($R_{DCR}$), da cui dipende linearmente, moltiplicata per la corrente in input. La formula è:

$$P_{L(V_{CC})} = R_{DCR} \cdot I^2_{V_{CC}}$$

Dei valori tipici di induttanza usati vanno da 1 $\mu$H fino a 30 $\mu$H. Partendo dalla scelta del valore dell'induttanza $L_{(V_{CC})}$, si può ricavare il valore del condensatore $C_{(V_{batt})}$ per il valore di attenuazione del rumore che si vuole ottenere. Grazie alla seguente formula si ottiene il valore di capacità:

$$C_{(V_{batt})} = \frac{1}{L_{(V_{CC})}} \cdot \left( \frac{10^{-\frac{Att_{dB}}{40}}}{2 \cdot \pi \cdot f} \right)^2 = \frac{1}{L_{(V_{CC})}} \cdot \left( \frac{10^{\frac{|Att_{dB}|}{40}}}{2 \cdot \pi \cdot f} \right)^2 \tag{5.25}$$

Tuttavia, la reale attenuazione del filtro è minore di quella voluta per via delle caratteristiche parassite dei componenti. La principale ragione di questa minor attenuazione è da attribuire alla resistenza serie del condensatore *ESR*. Considerando la formula 5.24 si può ricavare un'altra formula che consente di calcolare il valore effettivo di attenuazione.

$$Att_{dB} = 20 \cdot log \sqrt{\frac{1}{1 + \omega^2 \cdot L \cdot C \cdot \frac{\omega^2 \cdot L \cdot C - 2}{1 + \omega^2 \cdot ESR^2 \cdot C^2}}}$$

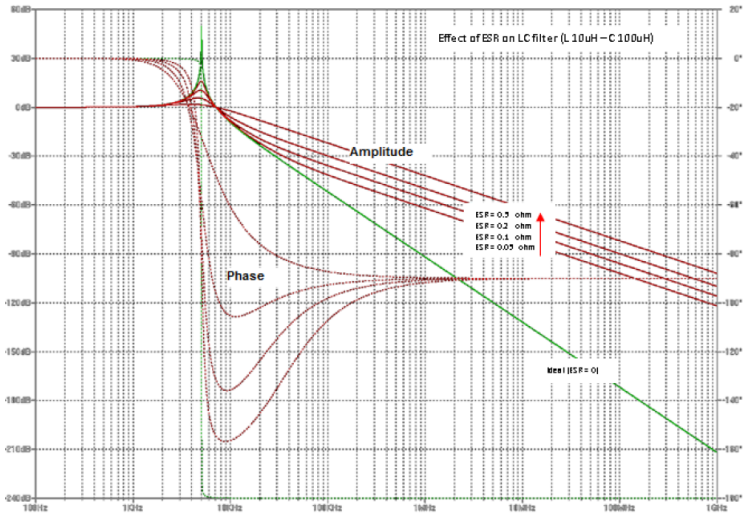
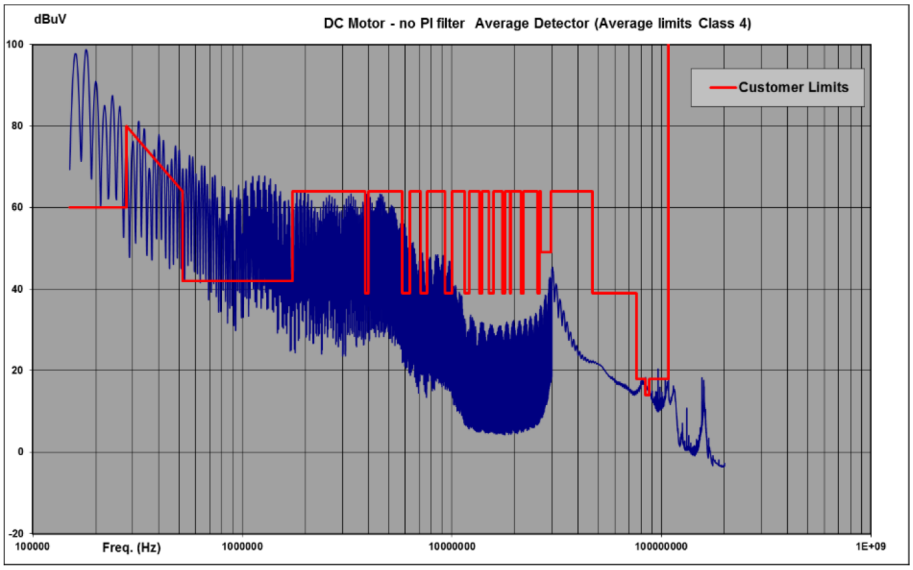



**Figura 5.11:** Effetto del cambio di ESR sull'attenuazione del filtro

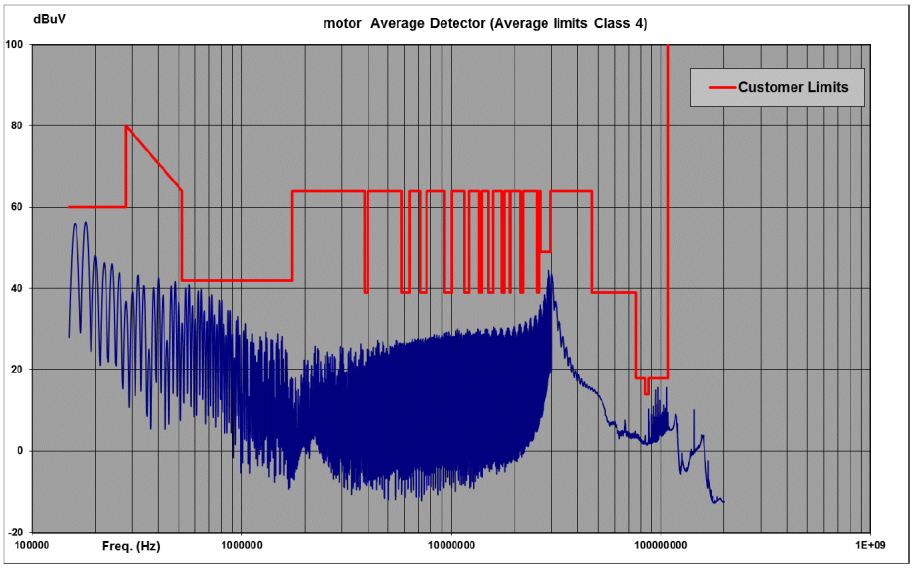
In figura 5.11 vengono riportate, a titolo di esempio, delle curve sperimentali[1] dove viene messo in relazione l'impatto del cambio di ESR sul valore di attenuazione del filtro. Il valore usato per l'induttanza è $L_{(V_{CC})} = 10\ \mu H$, mentre per il condensatore è $C_{(V_{batt})} = 100\ \mu F$ per queste curve. L'utilità del filtro si può riscontrare osservando le due figure che saranno presentate. Nella prima figura 5.12a, si può osservare lo spettro misurato[1] durante il funzionamento senza l'applicazione del filtro. Questa presenta delle frequenze, traccia in blu, che superano il limite consentito dalle norme per i disturbi elettromagnetici rappresentate con una traccia rossa. La figura 5.12b riporta lo spettro misurato[1] con l'applicazione del filtro *PI*. Si nota che le frequenze dello spettro sono al di sotto del limite del limite imposto dalle norme.

In conclusione si vogliono riportare i valori scelti per i due componenti presentati. Per l'induttanza è stato scelto il valore commerciale $L_{(V_{CC})} = 6{,}8\ \mu H$. Partendo da questo valore tramite la formula 5.25 è stato ricavato il valore del condensatore $C_{(V_{batt})} = 538\ \mu F$, approssimato a $560\ \mu F$. Il valore è uguale al condensatore calcolato in precedenza,questa scelta permette l'acquisto di un solo modello.

[1]Le curve sono state ricavate da un application note fornito dall'azienda STMicroelectronics.




**(a)** senza l'installazione del filtro *PI*




**(b)** con l'installazione del filtro *PI*

**Figura 5.12:** Spettro misurato

### 5.1.4 Blocco SBC suppply management

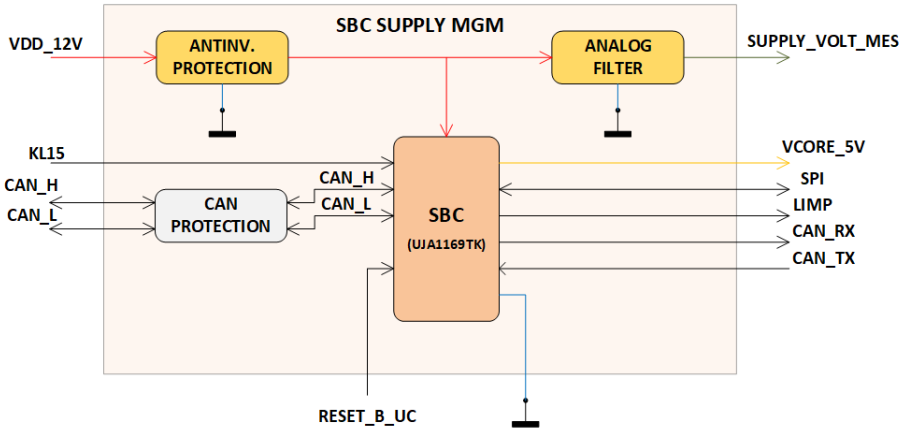



**Figura 5.13:** Schema a blocchi interno del macro blocco SBC suppply management

L'interno di questo macro blocco è riportato in figura 5.13 e presenta diverse funzionalità. La prima è l'anti inversione della tensione già trattata nei paragrafi precedenti. Si trova poi un blocco chiamato *analog filter*, con la funzione di abbassare la tensione di alimentazione in primo luogo, come seconda funzione ridurre i disturbi sulla misura da effettuare. Il motivo dell'abbassare la tensione di alimentazione tramite un partitore di tensione è dato dal fatto che si vuole monitorare il valore tramite il microcontrollore, per disattivare nel caso di sovra tensione il chip del ponte ad H. In figura 5.14 si riporta lo schema elettrico e si può riscontrare anche un diodo Zenner posto come protezione nel caso la tensione di batteria risultasse troppo elevata. La funzione del diodo è quella di saturare la tensione hai capi dell'ingresso dell'ADC ad un valore massimo di 5,6 V nel caso peggiore. Il filtro invece è ottenuto sfruttano la resistenza superiore del partitore di tensione ed il condensatore in parallelo alla seconda resistenza ed al diodo Zener. Per quanto riguarda il

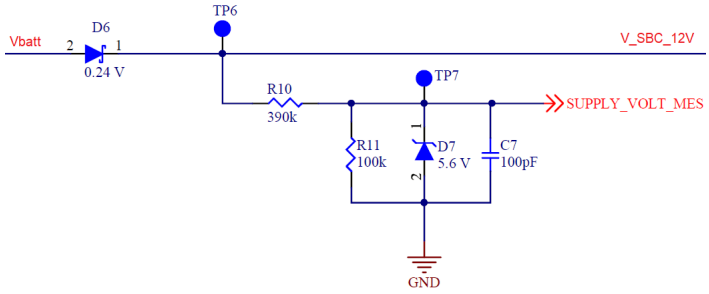



**Figura 5.14:** Schema elettrico *analog filter*

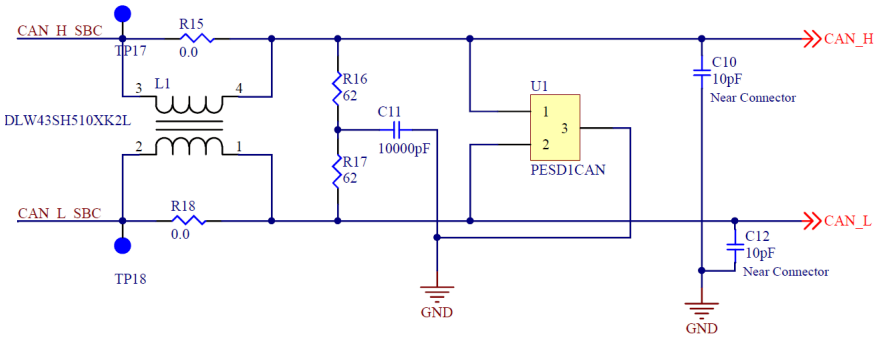



**Figura 5.15:** Schema elettrico protezioni rete CAN

blocco delle protezioni della rete CAN si trovano due componenti. Il primo è un diodo *ESD* per la protezione da scariche elettrostatiche e da altri impulsi. Il secondo componente è un filtro *choke* per attenuare i disturbi presenti sul bus CAN. Sulla scheda è presente anche lo schema elettrico per terminare la rete CAN qualora nel caso si presenti la necessità. Viene riportato lo schema elettrico in figura 5.15.
Per il componenti aggiunti al corretto funzionamento del SBC sono usati quelli indicati nel suo datasheet. Anche i collegamenti di questi componenti sono stati copiati dall'applicazione tipica ricavata dal datasheet.

## 5.2 Blocco MCU S32K116

Questo blocco comprende l'aspetto di alcuni componenti collegati al microcontrollore, come ad esempio il cristallo esterno per l'oscillatore, oppure i collegamenti di alcune linee come *JTAG*. Partendo proprio dai collegamenti, seguendo le linee guida fornite dal datasheet del *S32K116* viene consigliato di adottare alcune soluzioni. Per le linee di JTAG prevedere delle resistenze di pull-up o pull-down di valore 10 kΩsecondo l'elenco sottostante:

- pull-up per le linee *JTAG_TMS*, *JTAG_TDO*, *JTAG_TDI*;
- pull-down per la linea *JTAG_TCLK*.

Una considerazione simile vale anche per le linee di *SPI* dove è consigliato di mettere un pull-up con resistenza 10 k Ω sulla linea di chip select. Invece, collegare una resistenza da 10 kΩ a pull-down sul segnale del clock seriale.
Il datasheet fornisce una linee guida su come calcolare i condensatori da collegare ai capi del cristallo in modo da soddisfare le condizioni di *Barkhausen* per l'oscillazione. Si anticipa che la frequenza del cristallo scelto è uguale a 40 Mhz, per poter usufruire in un futuro della rete CAN FD senza violare

il timing. Per ottenere delle oscillazioni stabili si deve soddisfare il seguente criterio:

$$g_{mXOSC} > 5 \cdot g_{mcrit} \tag{5.26}$$

Dove $g_{mXOSC}$ è dato nel datashet e varia da un minimo di 16 mA/V ad un massimo di 47 mA/V. Mentre $g_{mcrit}$ viene calcolato nel seguente modo:

$$g_{mcrit} = 4 \cdot (ESR + R_s) \cdot (2 \cdot \pi \cdot f)^2 \cdot (C_0 + C_L)^2 \tag{5.27}$$

Dove si ha:

- ESR è uguale alla resistenza serie equivalente del cristallo e per quello scelto è massimo 70 Ω;
- $R_s$ è la resistenza connessa tra il pin *XTAL* del microcontrollore ed un pin del cristallo, in questo caso non è connessa quindi il dato è 0 Ω;
- f è la frequenza di oscillazione del cristallo uguale a 40 MHz;
- $C_0$ è la capacità di shunt del cristallo, che in questo caso vale 6 pF;
- $C_L$ è la capacita di carico totale del cristallo.

La capacita di carico totale $C_L$ può essere calcolata con la seguente formula:

$$C_L = C_S + \left[\frac{C1 \cdot C2}{C1 + C2}\right]$$

Dove per $C_S$ si intende la capacità parassita dei pin ed è dovuta anche alle tracce sul pcb, può essere stimata attraverso dei calcoli. Per i seguenti calcoli è stata considerata 5 pF, mentre C1 e C2 sono i condensatori collegati ai due nodi, uno tra il pin *XTAL* del microcontrollore e un terminale del cristallo, mentre il secondo tra il pin *EXTAL* del microcontrollore e l'altro terminale del cristallo. Hanno valore incognito e sono da ricavare, dunque per farlo si parte dalla espressione 5.26 ottenendo:

$$g_{mcrit} < \frac{g_{mXOSC}}{5}$$

Sostituendo a $g_{mcrit}$ l'equazione 5.27 trovata in precedenza:

$$4 \cdot (ESR + R_s) \cdot (2 \cdot \pi \cdot f)^2 \cdot (C_0 + C_L)^2 < \frac{g_{mXOSC}}{5}$$

Dopo alcune trasformazioni algebriche, espandendo $C_L$ si ottiene la formula finale:

$$\frac{C1 \cdot C2}{C1 + C2} < \sqrt{\frac{g_{mOSC}}{20 \cdot (ESR + R_s) \cdot (2 \cdot \pi \cdot f)^2}} - (C_0 + C_S)$$

Se si lavora sotto l'ipotesi che C1 = C2 = C allora si ottiene che:

$$C < 2 \cdot \left[ \sqrt{\frac{g_{mOSC}}{20 \cdot (ESR + R_s) \cdot (2 \cdot \pi \cdot f)^2}} - (C_0 + C_S) \right]$$

In conclusione per i condensatori C1, C2 è stata calcolata una capacità di 10 pF. Un ulteriore prova per controllare se i valori potessero essere corretti è stata quella di controllare se la condizione imposta con l'espressione 5.26 fosse soddisfatta per il valore minimo di $g_{mXOSC}$. Il risultato è stato affermativo.

## 5.3 Blocco output stage

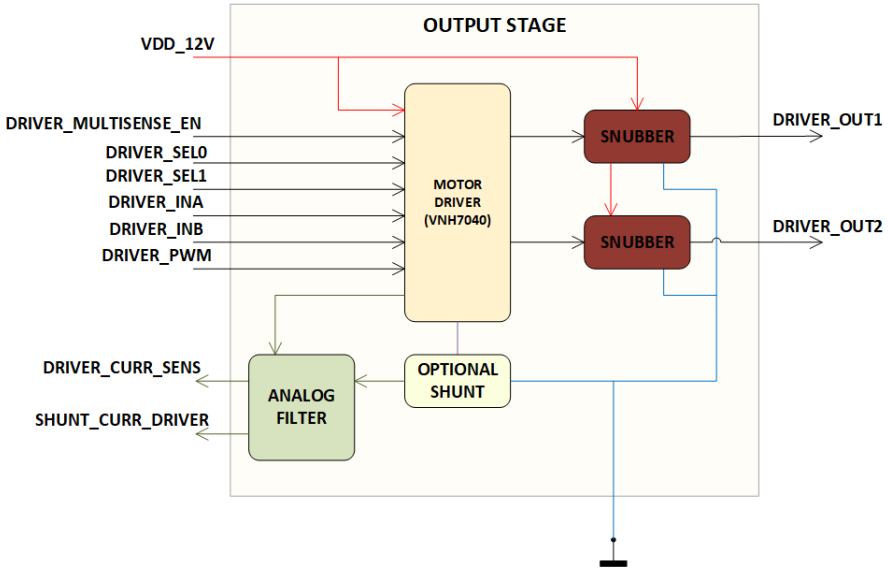



**Figura 5.16:** Schema a blocchi interno del macro blocco output stage

Questo blocco tratta nello specifico gli aspetti che riguardano chip *VNH7040*. Tra questi troviamo il dimensionamento della resistenza di sensing, il circuito per lo shunt, alcuni approfondimenti sulla questione di condensatori di filtro e dei circuiti di snuber opzionali.

### 5.3.1 Resistenza di sensing

Per dimensionare questa resistenza, $R_{SENSE}$, si devono tenere conto dei diversi casi operativi che possono subentrare durante il funzionamento del chip ed inoltre, anche la misura che si vuole effettuare. Si ricorda che il chip *VNH7040*

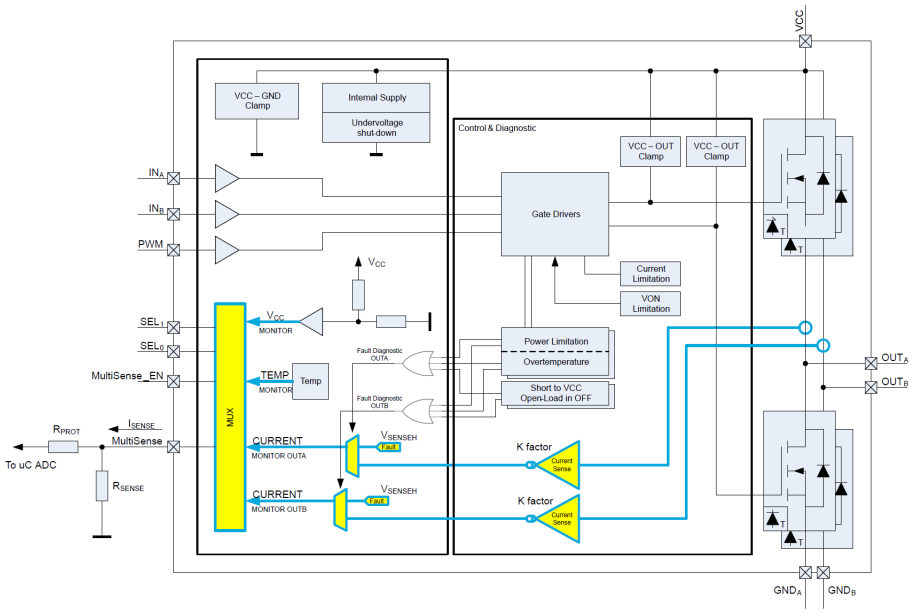



**Figura 5.17:** Schema interno multisense

ha quattro tipi di misura selezionabili tramite i segnali *SEL0* e *SEL1*, queste sono:

- misura della corrente nello stato ON dei mosfet high side;
- misura della corrente nello stato OFF per scoprire eventuali rotture;
- misura della tensione di alimentazione;
- misura della temperatura interna del chip.

In figura 5.17 è riportato uno schema interno riassuntivo di come vengono effettuate le varie misure.

### Misura della corrente

La misura della corrente in questo chip si divide in due sotto categorie. La prima riguarda la misura della corrente nello stato ON, la seconda la misura nello stato OFF. Inoltre, per lo stato ON abbiamo due sotto casi: uno è di funzionamento normale ed uno quando si presenta un malfunzionamento.
**Stato ON funzionamento normale**, la corrente ($I_{SENSE}$) fornita dal pin *multisense* è:

$$I_{SENSE} = \frac{I_{OUT}}{K}$$

Come si osserva, questa corrente è proporzionale rispetto alla corrente ($I_{OUT}$) che scorre nel mosfet high side attivo. Quindi, la tensione ($V_{SENSE}$) sulla resistenza di sense. ($R_{SENSE}$) risulta:

$$V_{SENSE} = R_{SENSE} \cdot I_{SENSE} = R_{SENSE} \cdot \frac{I_{OUT}}{K} \tag{5.28}$$

$K$ è il fattore di proporzionalità presente tra il Power mosfet high side ed il Sense MOS. Questo dato viene fornito dal datasheet ed include fattori geometrici, parametri di processo.
Si deve porre particolare attenzione nella scelta della resistenza di sense in quanto questa deve garantire una certa linearità alla misura effettuata, ma si deve essere in grado di distinguere un caso di corrente sovracorrente o fault imponendo una tensione di saturazione ($V_{SENSE,sat}$). Dato che la tensione di sense è letta dall'ADC, che presenta un tensione massima di ingresso pari a 5 V è stato deciso di utilizzare lo stesso valore anche per la tensione $V_{SENSE,sat}$. Partendo dalla formula 5.28 si possono calcolare gli estremi per il valore della resistenza, tenendo in conto di utilizzare la massima corrente che si può avere sul carico ($I_{OUT,max}$) ed il fattore tipico $K$. In condizioni normali per non raggiungere la tensione di saturazione la seguente equazione deve essere rispettata:

$$R_{SENSE} \leq K_{min} \cdot \frac{V_{SENSE,sat\_min}}{I_{OUT,max}} \tag{5.29}$$

La tensione di saturazione massima deve essere letta dall'ADC, che potrebbe presentare una qualche soglia. Di conseguenza la resistenza di sensing deve rispettare anche la seguente condizione:

$$R_{SENSE} \geq \frac{V_{SENSE,max}}{I_{SAT,min}} \tag{5.30}$$

Con $V_{SENSE,max}$ si intende la massima tensione che si vuole leggere quando si ha la massima corrente di carico voluta.
**Stato ON segnalazione malfunzionamento**, questa situazione di fail si ha quando una protezione viene attivata, ad esempio una sovracorrente sul mosfet high side attivo. Il dimensionamento della $R_{SENSE}$ deve tenere in conto anche di differenziare il normale funzionamento da quello di fail. Per includere questa condizione sono stati usati i valori di tensione in uscita al pin multisense in condizioni di fault ($V_{SENSE\_H,min}$) e la corrente in condizione di fault ($I_{SENSE\_H,min}$). Si trova quindi l'espressione:

$$R_{SENSE} \geq \frac{V_{SENSE\_H,min}}{I_{SENSE\_H,min}} \tag{5.31}$$

**Protezione da anti inversione della tensione**, $R_{SENSE}$ deve essere in grado di fornire una protezione al pin *multisense* durante questo evento. La causa di ciò è dovuta alla costruzione del chip che presenta un diodo parassita tra il pin di $V_{CC}$ e il pin *multisense*, durante la fase di inversione della tensione questo viene polarizzato direttamente, quindi se ne deve limitare la corrente. Per fare ciò si ricava dal datasheet il valore massimo di $I_{SENSE\_REV,max}$ = 20 mA, supponendo una tensione inversa di $V_{CC}$ = -16 V e una tensione del diodo $V_f$ = 0,7 V si ottiene:

$$R_{SENSE} \geq \frac{-V_{CC} - V_f}{I_{SENSE\_REV,max}} = \frac{16V - 0,7V}{20mA} = 765\Omega \tag{5.32}$$

**Stato OFF segnalazione malfunzionamento**, la corrente è limitata a $I_{SENSE\_H}$. In questo caso si possono sfruttare le equazioni già trovate per i casi precedenti impostando $V_{SENSE\_H}$ = 5 V. con questo caso si va ad impostare un valoe minimo della $R_{SENSE}$.

### Misura della tensione di alimentazione e della temperatura

Il canale di misura della tensione di alimentazione provvede la seguente espressione:

$$V_{SENSE} = \frac{V_{CC}}{4}$$

Mentre per la misura della temperatura si ricava la seguente formula:

$$V_{SENSE\_TC}(T) = V_{SENSE\_TC}(T_0) + \frac{dV_{SENSE\_TC}}{dT}(T - T_0)$$

Dove si definisce il coefficiente di temperatura uguale a:

$$temperature\ coefficient = \frac{dV_{SENSE\_TC}}{dT} \sim -5,5\frac{mV}{^\circ C}$$

Entrambe le sorgenti di misura sono soggette ad un'incertezza qupiù alta quando il chip è attivo.

### Calcolo $R_{SENSE}$

In conclusione tramite le espressioni descritte in precedenza si trovano i valori limite minimo e massimo per la resistenza. Il valore scelto deve ricadere nel suddetto range.

$$R_{SENSE,min} \geq \frac{V_{SENSE,max}}{I_{SENSE\_SAT,min}} = \frac{4,5}{4 \cdot 10^{-3}} = 1,125k\Omega$$

$$R_{SENSE,max} \leq K_3 \cdot \frac{V_{SENSE,sat\_min}}{I_{OUT,max}} = 4436 \cdot \frac{5}{10} = 2,218k\Omega$$

Il valore scelto è ricaduto su uno commerciale della serie E12:

$$R_{SENSE} = 1,5k\Omega$$

### 5.3.2 Resistenza di protezione

Da datasheet è consigliato inserire un resistenza ($R_{prot}$) lungo i collegamenti tra microcontrollore e chip. Questo per limitare la corrente durante degli impulsi negativi che si potrebbero presentare. Partendo dallo schema di figura 5.18 si osserva che quando avviene un impulso negativo la tensione sui pin di controllo viene portata ad un valore negativ odi circa $-2 \cdot V_f \sim -1,5V$. Si ha questo valore di tensione in quanto è la caduta di tensione dei diodi intrinseci dei mosfet di potenza. Inoltre, il transistor npn parassita presente sui pin logici entra in zona di saturazione. La corrente di saturazione di questo

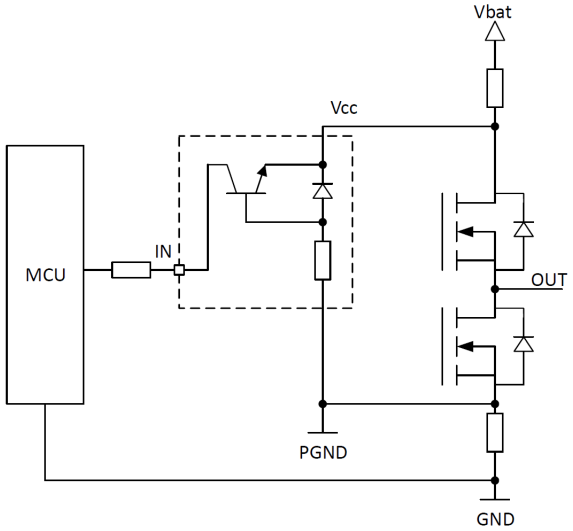



**Figura 5.18:** Schema interno semplificato durante un impulso negativo

npn potrebbe raggiungere dei valori elevati e questi potrebbero non essere compatibili con le correnti di input del microcontrollore. Di conseguenza si potrebbe verificare una rottura dell'ingresso o uscita logica del microcontrollore.
Il valore consigliato dal datasheet di questa resistenza va dalle centinaia di Ω alle unità di kΩ. Dato il range il valore scelto è uguale a quello usato per la resistenza di sense. Questo porta ad una minore differenza dei componenti da acquistare e installare abbassando così i costi.

### 5.3.3 Condensatori sulle uscite

Nella sezione del datasheet sull'applicazione tipica di questo chip viene consigliato di collegare dei condensatori sulle uscite. Per verificare la loro efficacia si vuole mostrare in figura 5.19, come fatto nel paragrafo del filtro *PI*, la comparazione tra i due spettri dell'emissione con e senza i condensatori. Gli spettri in figura 5.19 sono ricavati dal datasheet del componente. I con-

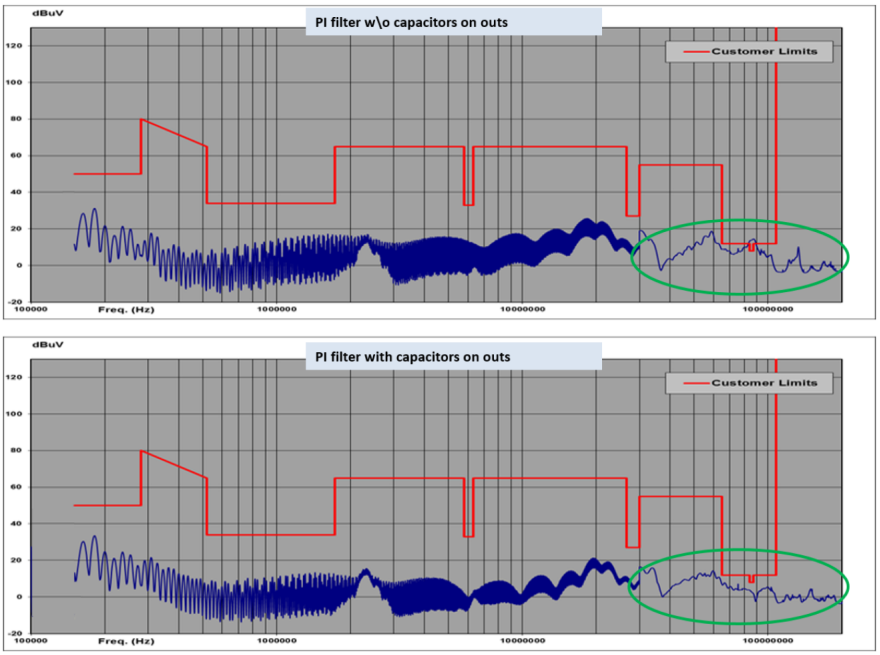



**Figura 5.19:** Spettro dell'emissione condotta

densatori sono di tipo ceramico ed hanno valori compresi tra 47 nF e 100 nF. Questi servono per migliorare le prestazioni dal punto di vista EMI ed ESD. La ragione del loro utilizzo risiede nel fatto che quando il chip è pilotato con una PWM, i disturbi elettromagnetici sono iniettati nelle uscite del *VNH7040*. La causa di questa iniezione di disturbi è dovuta all'antenna che si crea a causa del cablaggio, inoltre anche a causa delle spazzole del motore che possono generare scintille. Grazie a questi condensatori si crea un percorso corto verso GND evitando quindi che ritornino sulla linea di alimentazione. Un secondo beneficio è quello di smussare i fronti di salita e discesa della forma d'onda di uscita. Di conseguenza si va a trarre beneficio sulle emissioni ad alte frequenze.

### 5.3.4 Filtro analogico

La struttura di questi filtri, applicati al segnale proveniente dal pin multisense, al segnale dello shunt è la stessa vista per il filtro sulla lettura della tensione di alimentazione. L'unica differenza sta nel fatto che in questo caso non c'è bisogno del partitore di tensione. Un filtro analogico uguale a quello presentato per misurare la tensione dell'alimentazione è stato posto su ognuna delle due uscite per monitorare il corretto funzionamento del chip.

### 5.3.5 Shunt

Sebbene la corrente possa essere misurata internamente al chip è stato predisposta una resistenza di shunt tra GND' del chip e GND della scheda. La scelta è ricaduta su questo tipo di configurazione per aver una precisione migliore della corrente assorbita dal carico. Inoltre, così facendo si ottiene un doppio feedback sulla misura, il che aumenta la sicurezza. In figura 5.20 viene riportato lo schema elettrico. Questo stadio è formato da due parti.

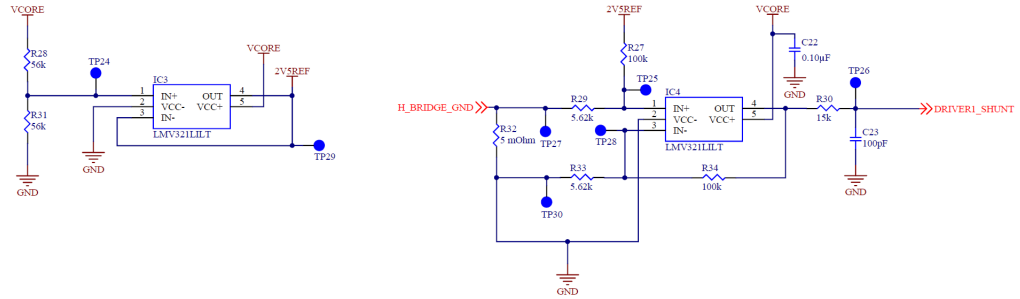



**Figura 5.20:** Schema elettrico circuito di shunt

La prima parte è formata da un partitore di tensione unito ad un *voltage follower*, questo genera un riferimento fisso a 2,5 V. La seconda parte è un comparatore formato da un operazionale, dove viene collegata la resistenza di shunt, a monte, in parallelo ai due terminali, positivo e negativo. Si collega poi al morsetto positivo anche la tensione di riferimento creata in precedenza. Il motivo di questo *offset* è dovuto al fatto che la corrente sulla resistenza scorre in entrambi i versi, dipende dal senso di rotazione del motore. Questo comporterebbe ad avere tensioni negative che però a causa dell'alimentazione dell'operazionale verrebbero saturate a 0 V. Ecco spiegato la causa di questo *offset* introdotto, quindi quando il motore è a velocità nulla l'uscita dell'operazionale è 2,5 V. Quando si ha la massima corrente di carico in un verso o nell'altro si avrà una tensione di 5 V oppure 0 V. Anche in questo caso prima

di arrivare all'ingresso dell'ADC del microcontrollore è stato posto un filtro passa basso.

## 5.4 Blocco sensor management

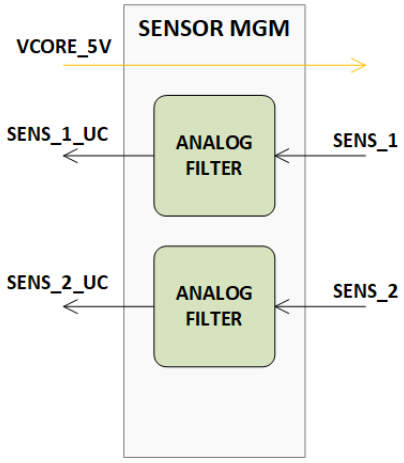



**Figura 5.21:** Schema a blocchi interno del macro blocco sensor management

L'ultimo macro blocco di questa scheda presentato è quello relativo alla gestione hardware delle misure dei sensori. Come descritto nel capitolo precedente questi sono di tipo analogico, dove il valore massimo raggiunto dalla tensione è inferiore alla tensione massima di input del canale dell'ADC. Questo blocco presenta quindi solo una parte di filtraggio del segnale che arriva dall'esterno scheda. Lo schema del filtro usato è quello già presentato al quale sono stati aggiunti dei condensatori ceramici nelle vicinanze del connettore. Il motivo di questi condensatori è quello di proteggere l'ingresso del microcontrollore da eventuali scariche elettromagnetiche.

## 5.5 Layout della scheda

Per la scheda progettata si è scelto di utilizzare uno stack da 4 layer con uno spessore del rame di 35 $\mu$m. I layer più esterni, *top* e *bottom*, sono stati usati per il routing delle piste di segnale. Mentre i layer interni sono stati usati come piano di massa e piano di alimentazione, rispettivamente quello sotto il top per GND, mentre quello sopra il bottom per $V_{CC}$.
L'uso di un piano di massa è importante per andare a minimizzare il l'area di loop della corrente di ritorno. Inoltre, se si analizza lo stack si nota che il secondo e il terzo layer con questa configurazione vanno a formare un condensatore che va ad attenuare i disturbi. I vantaggi portati con questo metodo sono molti rispetto all'usare delle singole tracce, anche se larghe.
L'usare 4 layer anziché 2 porta anche ad avere una migliore dissipazione termica, il calore viene trasferito più velocemente tra il *top* ed il *bottom*.
Nella scheda realizzata i componenti sono di tipo *SMD* è sono installati tutti sul layer *top*. Questo comporta che anche i componenti così detti di potenza siano saldati su questo layer. Per aiutare il trasporto del calore da questo layer al *bottom* sono stati usati *thermal vias*. In aggiunta per i componenti con la dissipazione maggiore sono state predisposte delle piazzole a rame scoperto nel layer *bottom*. In questo modo si possono stendere sopra delle paste termiche, appoggiare delle spugne a base di siliconi che poi vadino a contatto con un case metallico, il tutto per riuscire a dissipare una potenza maggiore. Per l'area di rame da utilizzare per la dissipazione del singolo componente il dato è stato ricavato dal datasheet, che fornisce una linea guida.
Per agevolare la fase di test di questa scheda sono stati predisposti *test point* per quasi tutti i segnali secondari e per tutti i segnali principali. In questo modo risulta facilitato l'accesso da parte di sonde, come puntali di un multimetro, ma anche nel caso entri in produzione può essere creato un così detto letto ad aghi per testarle con una velocità maggiore.

Il flusso di progettazione è stato quello di definire prima la geometria del perimetro esterno ricavando così l'area massima della scheda a disposizione. In secondo luogo sono stati posizionati i componenti che richiedevano un'area di dissipazione notevole, quindi i componenti di potenza come ad esempio il chip del ponte ad H, la protezione da inversione della tensione di batteria. Trovato questo primo posizionamento sono stati predisposti gli altri componenti con la filosofia di piazzare prima quelli con molti collegamenti. Ad esempio il chip del microcontrollore, oppure quello dell'SBC. Finito il posizionamento è stato eseguita manualmente la fase di routing delle tracce su PCB. Per quanto riguarda i componenti di potenza sono stati utilizzati dei poligoni di rame al posto di tracce, questo per non avere problemi di corrente

e aiutare lo smaltimento di calore avendo più area dissipativa a disposizione. Come ultima operazione è stato creato un piano di massa per tutta la scheda. Questo è presente su 3 layer di 4 disponibili, i due esterni e quello interno sotto al layer *top*. In questo modo tutte le zone dove sarebbe rimasto del rame scollegato da qualsiasi riferimento sono collegate a GND.
Nelle figure sottostanti vengono riportate le viste 3D del layer *top* e del layer *bottom*, rispettivamente figura 5.22 e 5.23. Si è voluto lavorare con librerie di componenti che sfruttino anche la vista 3D per poter poi passare il file ad un ufficio meccanico che ne realizzi un case.

## 5.6 Conclusioni

Il progetto realizzato ha presentato diverse sfide sotto il punto di vista delle scelte effettuate per potersi adattare a più casi possibili. Questo aspetto ha toccato tutte le parti sviluppate, a partire dal codice pensato già inizialmente come fosse un'applicazione finale. Ha portato a pensare delle ottimizzazioni utili ad avere un risparmio di tempo nei successivi sviluppi. Mentre per la parte di schematico sono state adottate delle scelte che da una parte hanno aumentato i componenti presenti sulla scheda, ma dall'altra parte rendono possibile una buona flessibilità dell'hardware durante alla fase di test. Si pensi a quasi tutti i componenti passivi quali resistenze, condensatori, diodi predisposti in caso di necessità. Un ulteriore sfida è derivata dal layout della scheda essendo uno stack a 4 layer e dovendo gestire dei valori di potenza media (100 - 200 W) su un'area ridotta. Questo ha portato a riprovare più volte la fase di placement per garantire una giusta area per poter dissipare al meglio il calore, anche per il routing si è proceduto a provare più volte prima di definire quello finale. Viene poi presentata la versione della scheda in modello 3D in quanto non è stato possibile realizzarne una versione fisica. Questo è successo per tempi dell'azienda, infatti non si sarebbe riusciti a stamparla e poi completarla con la saldatura dei componenti. In aggiunta questo è stato un progetto interno e non per la commercializzazione però di fatto questo prototipo è pronto per una possibile fase di pre-produzione e test.

Possibili sviluppi futuri di questa tesi possono racchiudere diversi ambiti dell'ingegneria. Ponendo come base la scheda progettata questa può essere testata ed introdurre quei miglioramenti necessari. Inoltre, potrebbe portare dei vantaggi introdurre dei meccanismi di controllo dello stato del carico, come disconnessione o cortocircuito ad alimentazione o GND. Si potrebbe sviluppare un controllo per il motore, anche avanzato facendo uso di un doppio anello di retroazione. In aggiunta, questa tesi potrebbe portare ad

un lavoro congiunto con altri dipartimenti per studiare un materiale ed un involucro che aiuti lo smaltimento del calore ed al tempo stesso possa essere resistente agli agenti atmosferici oppure essere un involucro di tipo stagno.

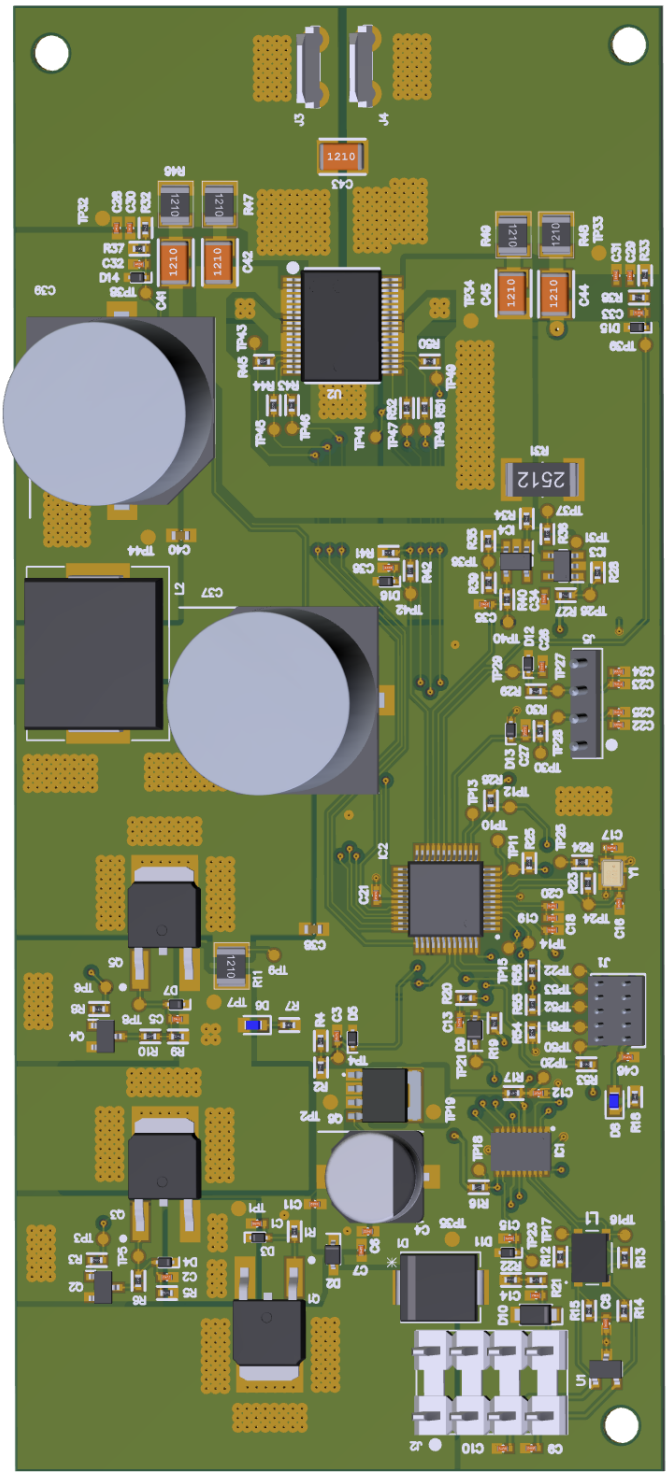


**Figura 5.22:** Top layer scheda realizzata

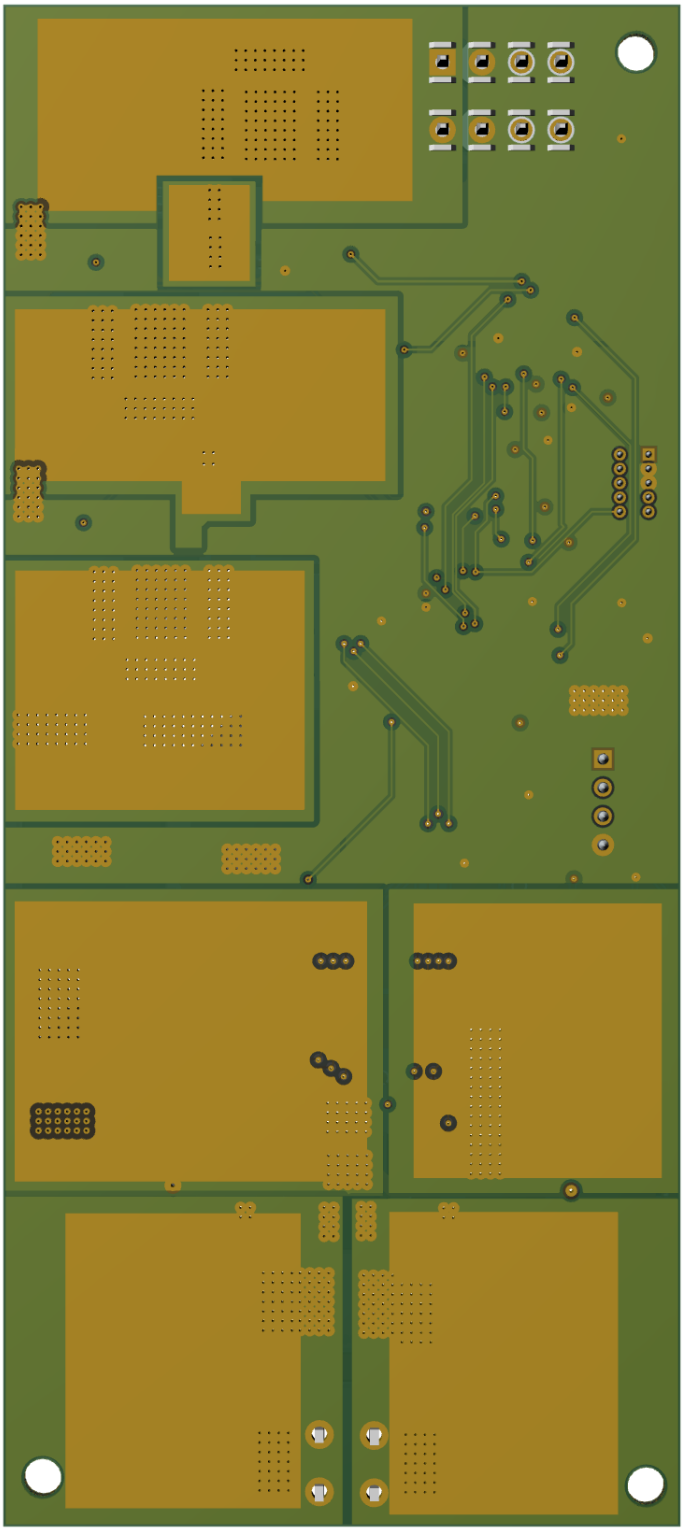
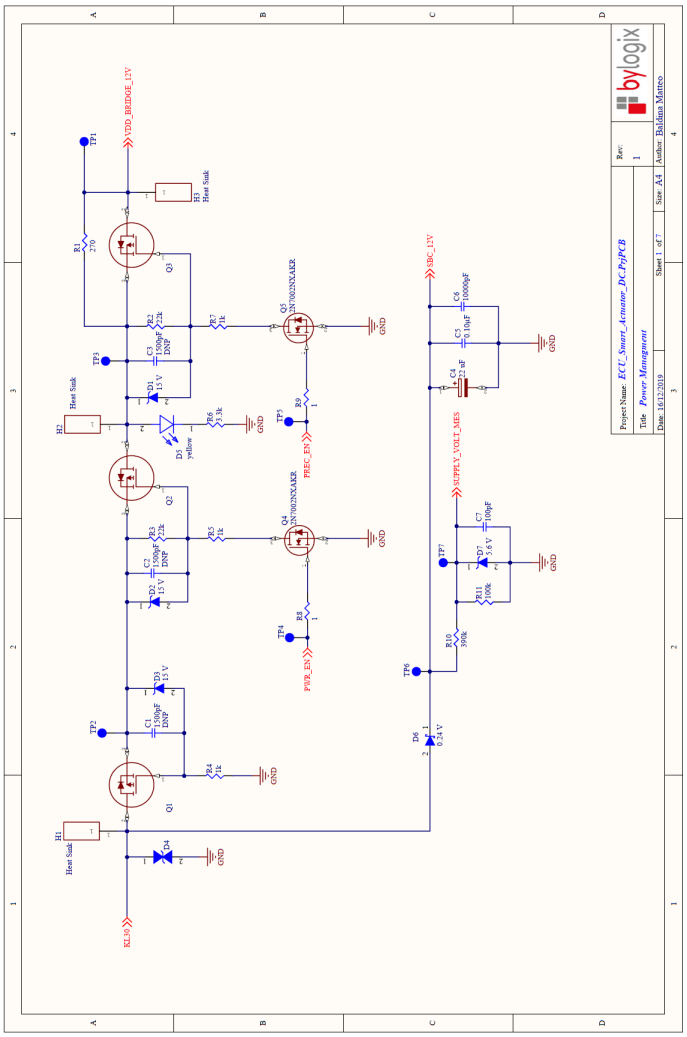
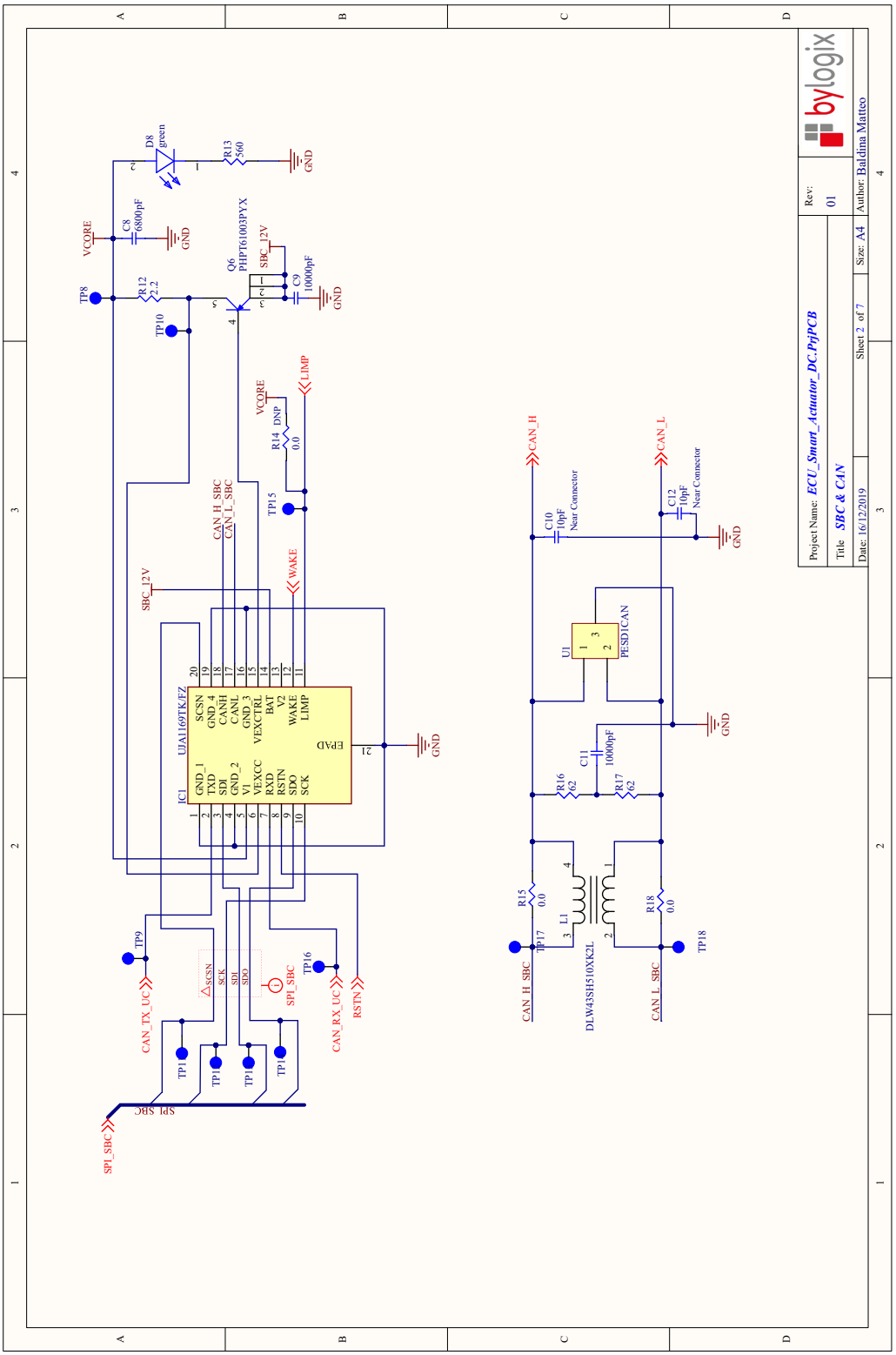


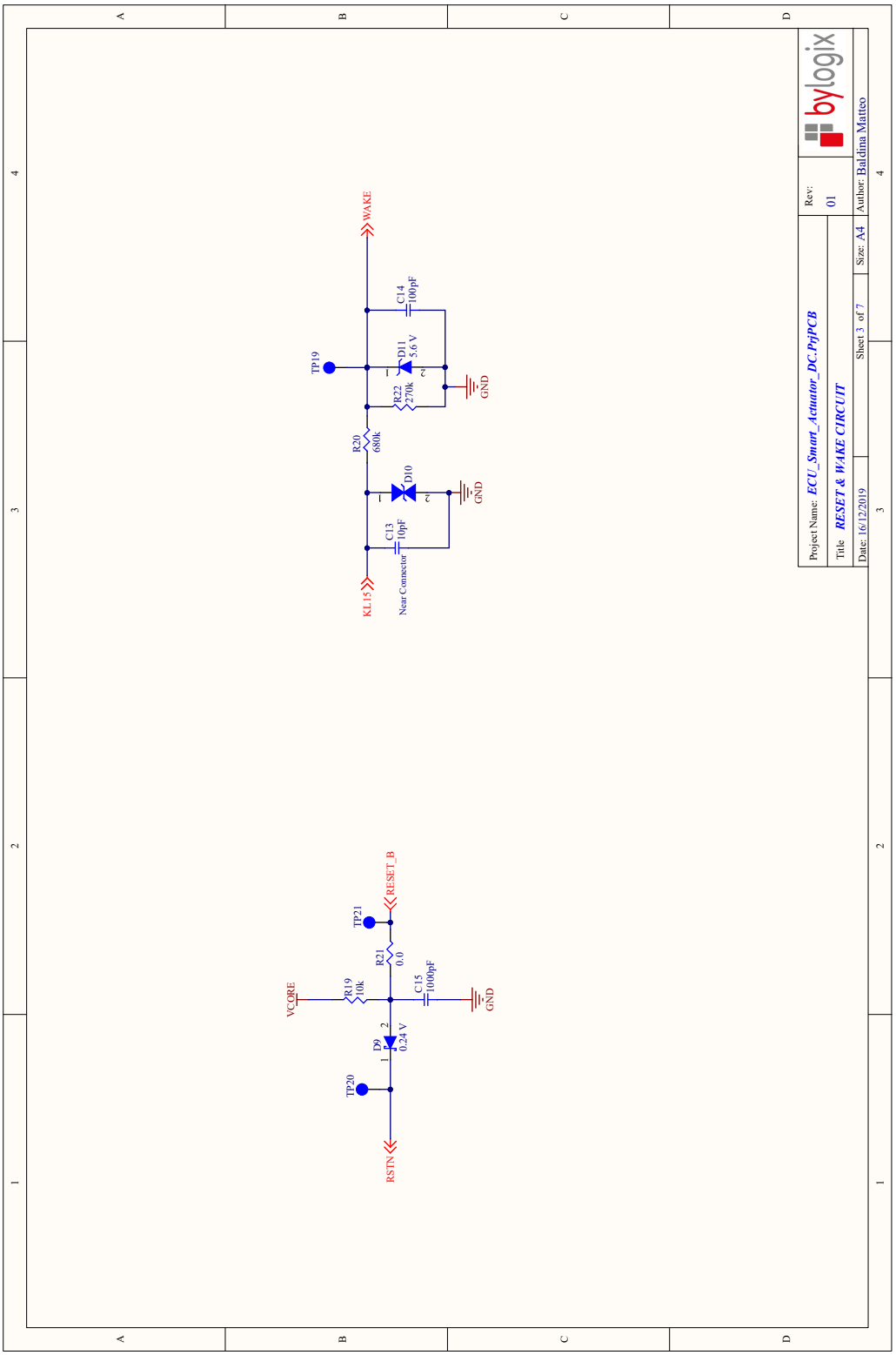
**Figura 5.23:** Bottom layer scheda realizzata

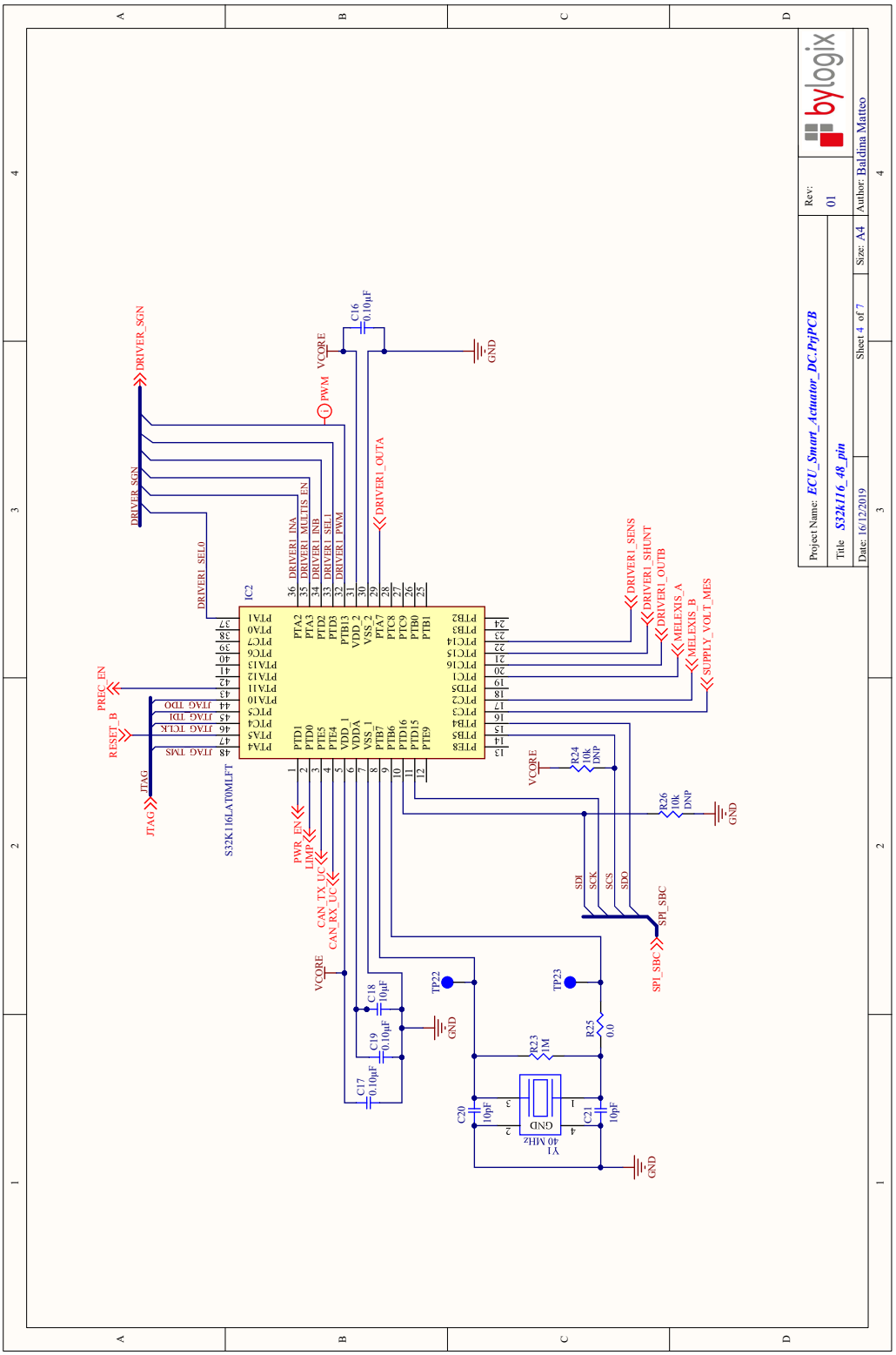
# Appendice A

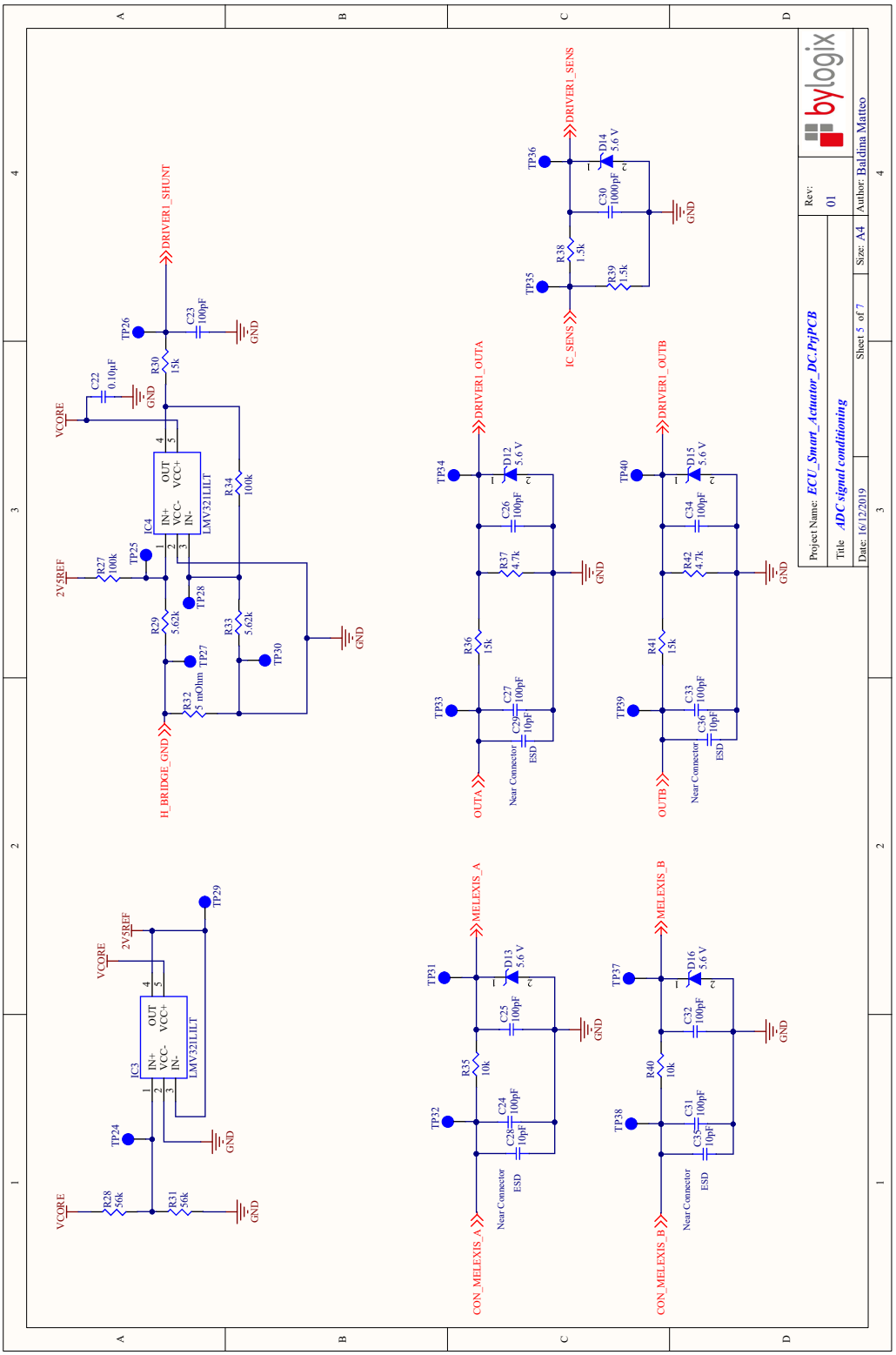
# Schemi elettrici

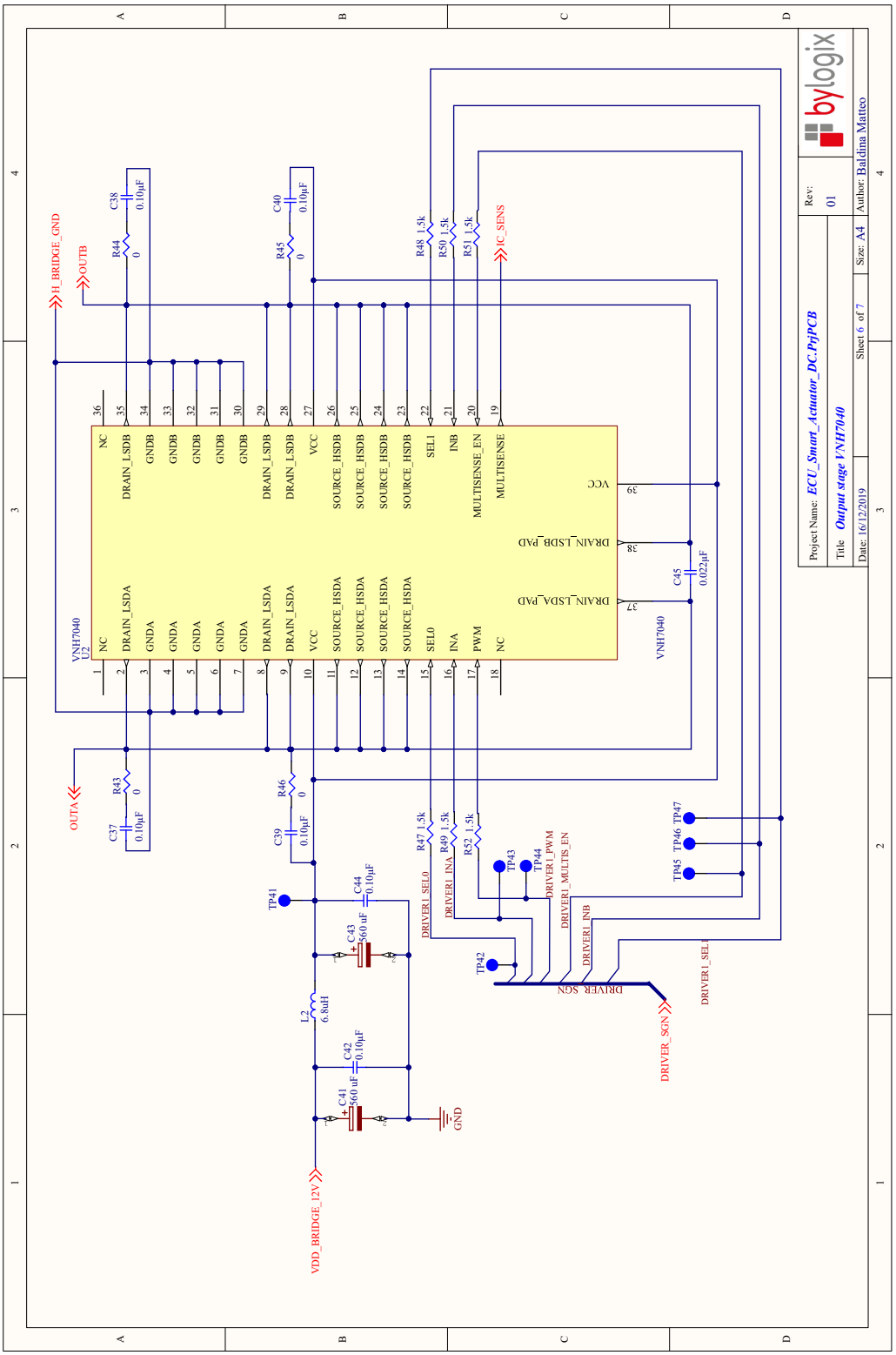


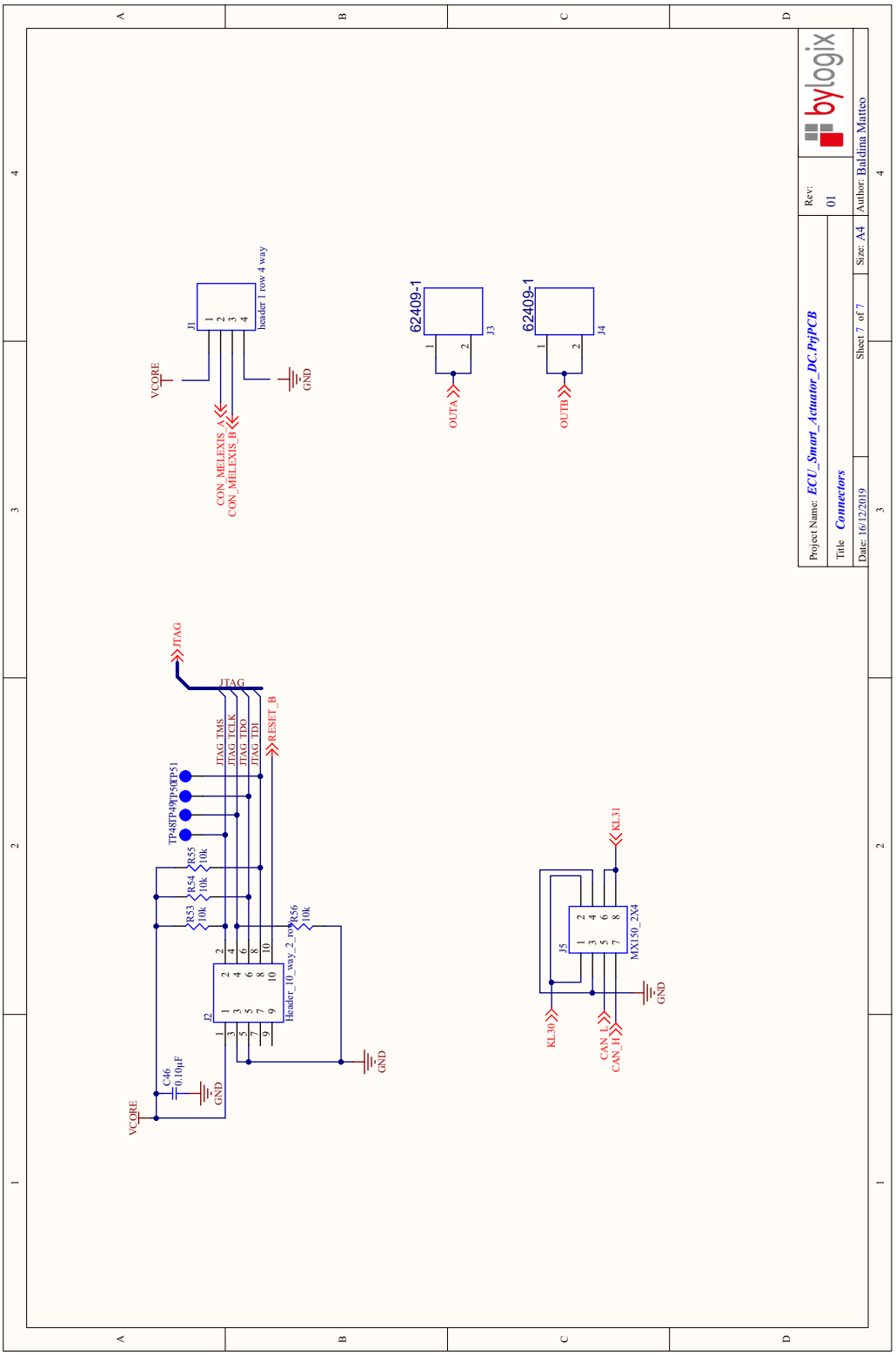

Project Name: ECU_Smart_Actuator_DC.PrjPCB
Title SBC & CAN
Date: 16/12/2019
Sheet 2 of 7
Size: A4
Rev: 01
Author: Baldina Matteo
bylogix
IC1
UJA1169TK/F/Z
GND_1
TXD
SDI
GND_2
V1
VEXCC
RXD
RSTN
SDO
SCK
SCSN
GND_4
CANH
CANL
GND_3
VEXCTRL
BAT
V2
WAKE
LIMP
EPAD
D8
green
R13
560
C8
6800pF
R12
2.2
Q6
PHPT61003PYX
C9
10000pF
R14_DNP
0.0
C10
10pF
Near Connector
C12
10pF
Near Connector
U1
PESD1CAN
C11
10000pF
R16
62
R17
62
R15
0.0
R18
0.0
L1
DLW43SH510XK2L
VCORE
SBC_12V
LIMP
WAKE
CAN_H_SBC
CAN_L_SBC
CAN_H
CAN_L
GND
TP8
TP9
TP10
TP15
TP16
TP17
TP18
CAN_TX_UC
CAN_RX_UC
RSTN
SPI_SBC
SCSN
SCK
SDI
SDO


WAKE
C14
100pF
TP19
D11
5.6 V
R22
270k
GND
R20
680k
D10
C13
10pF
KL15
Near Connector
GND
RESET_B
TP21
R21
0.0
C15
1000pF
VCORE
R19
10k
GND
D9
0.24 V
TP20
RSTN
Project Name: ECU_Smart_Actuator_DC.PrjPCB
Title RESET & WAKE CIRCUIT
Date: 16/12/2019
Sheet 3 of 7
Size: A4
Rev: 01
Author: Baldina Matteo
bylogix


Project Name: ECU_Smart_Actuator_DC.PrjPCB
Title S32k116_48_pin
Date: 16/12/2019
Sheet 4 of 7
Rev: 01
Size: A4
Author: Baldina Matteo
bylogix
S32K116LAT0MLFT
IC2
DRIVER_SGN
DRIVER1_SEL0
DRIVER1_INA
DRIVER1_MULTIS_EN
DRIVER1_INB
DRIVER1_SEL1
DRIVER1_PWM
DRIVER1_OUTA
PWM
VCORE
GND
C16 0.10µF
PREC_EN
RESET_B
JTAG
PWR_EN
LIMP
CAN_TX_UC
CAN_RX_UC
DRIVER1_SENS
DRIVER1_SHUNT
DRIVER1_OUTB
MELEXIS_A
MELEXIS_B
SUPPLY_VOLT_MES
R24 10k DNP
R26 10k DNP
SDI
SCK
SCS
SDO
SPI_SBC
TP22
TP23
C18 10µF
C19 0.10µF
C17 0.10µF
R23 1M
R25 0.0
C20 10pF
C21 10pF
Y1 40 MHz


Project Name: ECU_Smart_Actuator_DC.PrjPCB
Title ADC signal conditioning
Date: 16/12/2019
Sheet 5 of 7
Size: A4
Rev: 01
Author: Baldina Matteo
bylogix
DRIVER1_SHUNT
H_BRIDGE_GND
DRIVER1_OUTA
DRIVER1_OUTB
DRIVER1_SENS
IC_SENS
MELEXIS_A
MELEXIS_B
CON_MELEXIS_A
CON_MELEXIS_B
OUTA
OUTB
VCORE
2V5REF
GND
LMV321LILT
Near Connector
ESD


Project Name: ECU_Smart_Actuator_DC.PrjPCB
Title Output stage VNH7040
Date: 16/12/2019
Sheet 6 of 7
Size: A4
Rev: 01
Author: Baldina Matteo
bylogix
VNH7040
H_BRIDGE_GND
OUTB
OUTA
IC_SENS
VDD_BRIDGE_12V
GND
DRIVER_SGN
DRIVER1_SEL0
DRIVER1_INA
DRIVER1_PWM
DRIVER1_MULTIS_EN
DRIVER1_INB
DRIVER1_SEL1


Project Name: ECU_Smart_Actuator_DC.PrjPCB
Title: Connectors
Date: 16/12/2019
Sheet 7 of 7
Size: A4
Rev: 01
Author: Baldina Matteo
bylogix
header 1 row 4 way
J1
VCORE
GND
CON_MELEXIS_A
CON_MELEXIS_B
62409-1
J3
OUTA
62409-1
J4
OUTB
JTAG
JTAG_TMS
JTAG_TCLK
JTAG_TDO
JTAG_TDI
RESET_B
TP48TP49TP50TP51
R55 10k
R54 10k
R53 10k
R56 10k
J2
Header_10_way_2_row
C46 0.10µF
KL31
J5
MX150_2X4
KL30
CAN_L
CAN_H

# Bibliografia


[1] STMicroelectronics, *VIPower® M0-7 standard high-side drivers hardware design guide*, User manual UM1922, 2018.

[2] STMicroelectronics, *VIPower VNHD7xxAY H-Bridges drivers*, Application note AN5265, 2019.

[3] STMicroelectronics, *VIPower M0-7 H-Bridges for Automotive DC Motor Control*, Application note AN5026, 2019.

[4] STMicroelectronics, *Thermal dissipation and how to clear diagnostic registers in case of under-voltage*, Application note AN4695, 2015.

[5] STMicroelectronics, *Thermal dissipation and how to clear diagnostic registers in case of under-voltage*, Application note AN4695, 2015.

[6] ONSemiconductor®, *Thermal dissipation and how to clear diagnostic registers in case of under-voltage*, Application note AND9596/D, 2017.

[7] STMicroelectronics, *Hardware design guideline power supply and voltage measurement*, Application note AN4218, 2015.

[8] STMicroelectronics, *EMC design guides for motor control applications*, Application note AN4694, 2015.

[9] STMicroelectronics, *How to improve EMI behavior in switching applications*, Application note AN5084, 2019.

[10] NXP Semiconductors, *Hardware Design Guidelines for S32K1xx Microcontrollers*, Application note AN5426, 2019.

[11] NXP Semiconductors, *S32K1xx ADC guidelines, spec and configuration*, Application note AN12217, 2018.

[12] NXP Semiconductors, *Some Characteristics and Design Notes for Crystal Feedback Oscillators*, Application note AN2049, 2004.