UNIVERSITÀ DI PISA

Facoltà di Ingegneria

Corso di Laurea Specialistica in Ingegneria Elettronica

Nuovo Ordinamento

Tesi di Laurea

Progetto e realizzazione di una piattaforma hardware a struttura gerarchica per gestione e monitoraggio di batterie al litio

Relatori
Prof. Roberto Saletti
Prof. Roberto Roncella

Candidato
Federico Magni

Anno Accademico 2009 - 2010

Indice

Capitolo 1

Introduzione

I veicoli tradizionali a combustibile fossile sono molto comodi ed efficienti ma presentano il notevole problema dell'inquinamento atmosferico, inoltre gli esperti stimano che tra circa 40 anni le scorte di petrolio saranno esaurite. Il grande consumo di combustibile fossile ha portato al riscaldamento globale, dovuto alle emissioni di CO_2 immesse in grandi quantità durante l'anno. Per i problemi citati sopra i veicoli elettrici hanno suscitato attenzione da parte degli scienziati e negli ultimi anni alcuni modelli sono usciti sul mercato.
Il primo veicolo elettrico fu fatto nel 1883 e il primo a combustione interna fu fatto nel 1886. Alla fine del XIX secolo il veicolo elettrico ebbe il suo primo picco di sviluppo ma nei primi anni del XX secolo furono trovati molti pozzi petroliferi ed il veicolo a combustione interna prese il sopravvento. Oltre la metà del XX secolo, una crisi petrolifera spinse gli scienziati a tornare a sviluppare i veicoli elettrici, ma presto il combustibile fossile piombò di nuovo sul mercato mondiale con prezzi vantaggiosi ed i veicoli a combustione interna vinsero la competizione con i veicoli elettrici, grazie ad una maggiore competitività economica. All'epoca il fallimento dei veicoli elettrici può essere riassunto in questi punti [1]:

- prezzo basso degli idrocarburi;
- elevato prezzo della batteria utilizzata nei veicoli elettrici;
- tempo di ricarica della batteria molto lungo;
- bassa autonomia delle batterie.

Negli ultimi anni i veicoli elettrici sono tornati alla ribalta per i seguenti motivi:

- il combustibile fossile sta per esaurire;
- crescente aumento del costo del carburante;
- necessità di ridurre le emissioni di CO_2.

Per rendere i veicoli elettrici più competitivi sul mercato, è necessario migliorare le prestazioni delle batterie:

- migliorare la capacità della batteria, che dipende dal materiale con cui è fatta;
- implementare un sistema di gestione della batteria (Battery Management System: *BMS*).

In generale le prestazioni dei veicoli elettrici, a livello di consumo e di potenza massima, sono influenzate profondamente dalle caratteristiche del pacco batteria. Esistono tuttavia diverse tipologie di batteria per tali applicazioni:

- batterie al piombo;
- batterie NiMH;
- batteria al Litio.

Quest'ultima tipologia è l'ideale per i trasporti elettrici grazie alla capacità di fornire elevata potenza ed energia.
In Tabella 1.1 vengono mostrati i parametri delle tre tipologie di batterie:

Batteria	**Densità di energia [Wh/Kg]**	**Densità di potenza [W/Kg]**	**Cicli di vita [tempo]**	**Costo [$]**
piombo-acido	$30 \sim 50$	$200 \sim 400$	$400 \sim 600$	$120 \sim 150$
NiMH	$50 \sim 70$	$150 \sim 300$	> 800	$150 \sim 200$
Litio	$120 \sim 140$	$250 \sim 450$	1200	$150 \sim 180$

Tabella 1.1: Caratteristiche delle batterie

1.1 Battery Management System

In generale un pacco batteria, in grado di alimentare un veicolo elettrico, è composto da più batterie in serie, le quali a loro volta, consistono in più celle in serie in modo da raggiungere tensioni adeguate come *130-160V*. Con la struttura appena descritta le celle, durante il funzionamento, possono trovarsi in condizioni di carica diverse, creando un sbilanciamento che riduce la capacità totale della batteria e può portare ad un danneggiamento permanente di una o più celle. Questo sbilanciamento è dovuto al fatto che le celle non sono tutte identiche, i loro parametri caratteristici (capacità, resistenza interna, etc...) sono diversi tra loro. Risulta necessario periodicamente monitorare la tensione di ciascuna cella ed effettuare un adeguato riequilibrio, questo è uno tra i molti compiti che deve adempiere un sistema di gestione delle batterie (*BMS*).

In generale un *BMS* è un sistema di controllo elettronico che deve monitorare, proteggere e calcolare lo stato di più celle ricaricabili collegate in serie tra loro, inoltre deve essere in grado di interfacciarsi con altri sistemi di controllo per lo scambio di dati.

Obiettivi:

- proteggere le celle e la batteria dal danneggiamento;
- prolungare la vita del pacco batteria;
- mantenere il pacco batteria in uno stato in cui può soddisfare i requisiti funzionali relativi all'applicazione per la quale è destinato.

Per ottenere gli obiettivi sopra citati è necessario che un *BMS* includa alcune di queste funzionalità:

- **protezione celle:** proteggere la batteria da condizioni di funzionamento fuori dalle specifiche è fondamentale per tutte le applicazioni, altrimenti è inevitabile che si danneggino una o più celle ed il costo di sostituzione delle stesse può essere ingente;
- **determinazione dello stato di carica:** è importante conoscere lo stato di carica (*SOC*) di una batteria o di una cella sia per fornire all'utente l'autonomia residua e sia per controllare in modo efficiente la fase di carica e il bilanciamento;

- **determinazione dello stato di salute:** lo stato di salute (*SOH*) è un parametro che quantifica la capacità della cella di garantire il suo buon funzionamento, la conoscenza di tale parametro è importante per indicare, se è necessario, un intervento di manutenzione;
- **bilanciamento celle:** all'aumentare dei cicli di carica-scarica aumenta la differenza delle tensioni tra le celle che compongono il pacco, in sintesi questo porta ad una diminuzione della capacità totale della batteria. E' necessario che il *BMS* implementi un sistema di bilanciamento per aumentare l'autonomia del pacco batteria;
- **invecchiamento:** monitorare la storia della batteria è un'altra possibile funzione del *BMS*, questo è necessario per stimare lo stato di salute ma anche per determinare se essa è stata soggetta ad abusi. Parametri come, numero di cicli di carica-scarica, temperature massime e minime di carica e scarica, correnti massime e minime di carica, possono essere registrati per una successiva analisi e per valutare le richieste di garanzia;
- **identificazione:** può essere utile provvedere a memorizzare informazioni riguardo al fabbricante, alla chimica e al numero di serie delle celle per permettere la tracciabilità delle stesse in caso di guasti;
- **comunicazione:** è fondamentale avere un canale di comunicazione tra il *BMS* e l'utente esterno, in modo tale che quest'ultimo possa modificare i parametri e controllare lo stato del pacco batteria.

In ambito automobilistico, il *BMS* deve comunicare con altri sistemi di bordo e deve lavorare con le batterie in rapida evoluzione di carica e scarica quando il veicolo frena e accelera rispettivamente. In questo ambito, oltre alle funzionalità precedentemente elencate, se ne aggiungono altre riportate di seguito:

- implementare un meccanismo di emergenza in caso di malfunzionamenti, perdita della comunicazione e abusi incontrollati del sistema;
- possibilità di isolare le celle in caso di malfunzionamento;
- fornire informazioni al display di bordo;
- prevedere l'autonomia residua del veicolo.

Quello automobilistico è un ambiente di lavoro duro sia per il *BMS*, sia per il pacco batteria, quindi è fondamentale realizzare una procedura di controllo di ciascuna cella, in modo tale che il sistema intervenga se essa lavori fuori dalle specifiche. Per esempio se una batteria lavora ad una temperatura troppo elevata, il sistema di controllo deve essere in grado di attivare un sistema di raffreddamento.

Una funzione importante del *BMS* è quella del calcolo dello stato di carica (*SOC*) di ciascuna cella. E' importante conoscere il *SOC* sia per il calcolo dell'autonomia residua del veicolo, sia per effettuare un bilanciamento delle celle ed evitare quindi che vadano in sovra-carica o in sovra-scarica.
Nei veicoli ibridi le batterie devono essere in uno stato di carica opportuno per accettare potenza in caso di frenata e fornirla in caso di accelerazione, per questo motivo ciascuna cella deve essere mantenuta ad un *SOC* adeguato per evitare che si scarichino o si carichino troppo (Figura 1.1).

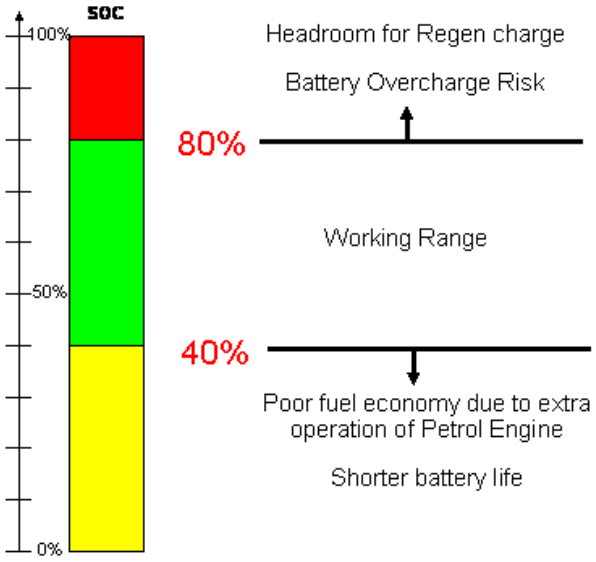

Figura 1.1: range del *SOC* nei veicoli ibridi ([2])

L'architettura di un *BMS* può essere suddivisa in unità (o blocchi), le quali dovranno essere progettate in modo da implementare le funzioni sopra riportate. Il nucleo del sistema è composto da un circuito dotato di intelligenza (un microcontrollore, un *DSP*, un processore) in grado di acquisire i dati esterni e tramite il modello della batteria, valutarne lo stato in

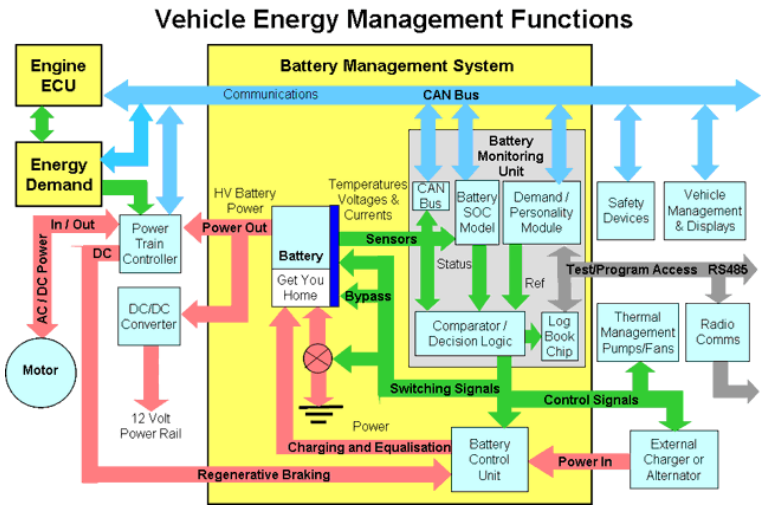

Figura 1.2: esempio architetturale di un *BMS* ([2])

qualunque istante e condizione. Il modello può essere utilizzato per registrare la storia passata del pacco batteria per scopi di manutenzione, valutare l'autonomia residua, calcolare la distanza percorsa dall'ultima ricarica ed il consumo energetico medio. Quest'ultimi due punti sono maggiormente importanti in veicoli completamente elettrici, dato che l'unica fonte di alimentazione è proprio il pacco batteria. Per evitare che la rottura di una cella comprometta il funzionamento del veicolo è possibile inserire degli interruttori per isolare la parte compromessa, di contro la potenza che viene fornita dal pacco al motore sarà ridotta. L'intelligenza del sistema inoltre riceve gli ingressi per stimare lo stato di salute (*SOH*) della batteria. I segnali relativi al *SOH* possono essere acquisiti con frequenza minore rispetto a quelli relativi al *SOC*, che hanno una dinamica più veloce. Per esempio per calcolare il *SOH* è possibile acquisire una volta al giorno i segnali relativi quando il veicolo è fermo.

I risultati del modello della batteria devono essere monitorati per evitare che escano fuori dalle dinamiche previste, quindi un'unità (hardware o software) confronta tali risultati con dei parametri di riferimento ed in caso di errore prende delle contromisure informando l'utente (o il sistema gerarchicamente superiore).

I segnali che vengono utilizzati dal microcontrollore (o simili) per stimare il *SOC* della batteria ed effettuare i controllori necessari, provengono da un'unità di misura che, con opportuni sensori, acquisisce la tensione e la

temperatura di ciascuna cella. Per garantire le procedure di misura e di bilanciamento delle celle il sistema è dotato di opportuni circuiti di potenza che consentono, per esempio, di selezionare le celle ed isolare segmenti di batteria danneggiati.

Un'altra funzione essenziale del *BMS* è il bilanciamento, procedura che consente di portare tutte le celle del pacco batteria al solito stato di carica, per evitare danneggiamenti e diminuzione della capacità complessiva.

Per evitare che una cella danneggiata interrompa il flusso di corrente nel pacco, è necessario inserire degli interruttori, capaci di sopportare centinaia di ampere, per isolarla. In questa condizione la potenza massima trasferibile dal pacco al motore è ridotta, è necessario comunicare alla centralina principale la riduzione delle prestazioni del veicolo.

1.1.1 Esempi architetturali

Esistono diverse architetture per realizzare un *BMS*, di seguito ne vengono presentate due: *topologia a stella* e *topologia ad anello*.

***Topologia a stella*:**
Le celle che compongono il pacco batteria sono suddivise in gruppi e ciascuno di essi viene gestito da un modulo intermedio (*slave*). Ogni modulo intermedio comunica con un altro modulo (*master*) gerarchicamente superiore. Ciascuno *slave* misura, progressivamente, la temperatura e la tensione di ciascuna cella appartenente al solito gruppo, ed invia i risultati al *master* che li elabora. Il *master* calcola il modello della batteria, gestisce i meccanismi di protezione e comunica con i sistemi esterni gerarchicamente superiori (Figura 1.3).
Questa configurazione consente di non utilizzare un circuito stampato per ciascuna cella, inoltre è possibile dividere il carico di elaborazione tra gli *slave* ed il *master*. Gli *slave* sono collegati a tensioni progressivamente maggiori, rispetto al terminale negativo della batteria, quindi, per garantire la comunicazione con il *master*, è necessario utilizzare degli accoppiatori ottici.

***Topologia a catena*:**
In questa topologia viene collegata a ciascuna cella una scheda elettronica. Ciascuna scheda è alimentata dalla cella ed è dotata di un convertitore A/D per digitalizzare la tensione ed il segnale proveniente dal sensore di tempera-

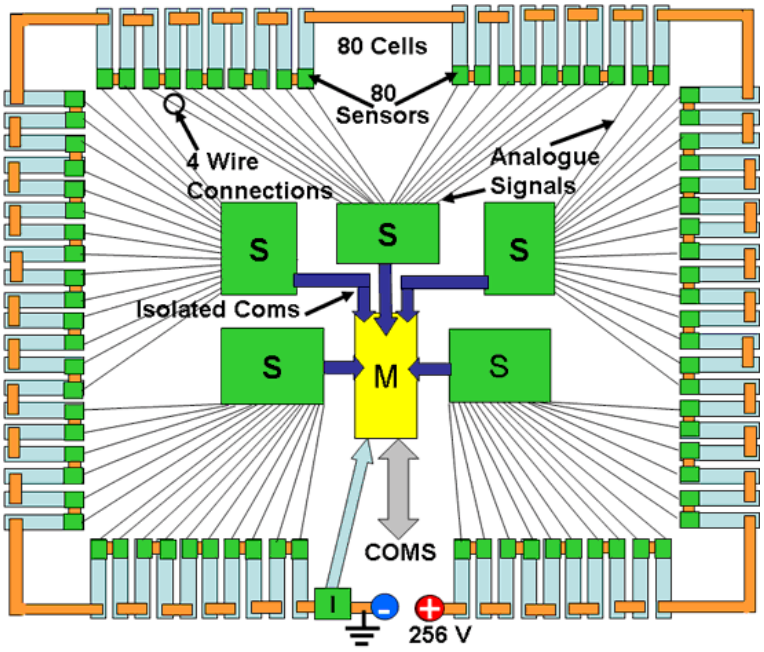

Figura 1.3: struttura a stella ([2])

tura, posto su ogni cella. I vari *slave* condividono un unico bus utilizzato per inviare al *master* le informazioni. L'elaborazione e il condizionamento dei dati digitali è effettuato esclusivamente dal *master*, dato che lo *slave*, essendo alimentato dalla singola cella, deve prelevare la minor potenza possibile. Quindi il compito dello *slave* è esclusivamente quello di prelevare i dati dalle celle, convertirli in digitale ed inviarli al *master*, il quale li utilizzerà, per esempio, per calcolare lo stato di carica di ciascuna cella tramite il modello (Figura 1.4).

I principali vantaggi di questa topologia sono la flessibilità e la riduzione dei collegamenti. Il tutto porta ad un notevole miglioramento dell'affidabilità, cosa importante soprattutto in ambito automobilistico.

Lo svantaggio principale è dovuto al fatto che ciascuna cella deve ospitare una scheda elettronica, quindi aumentano i costi per la realizzazione.

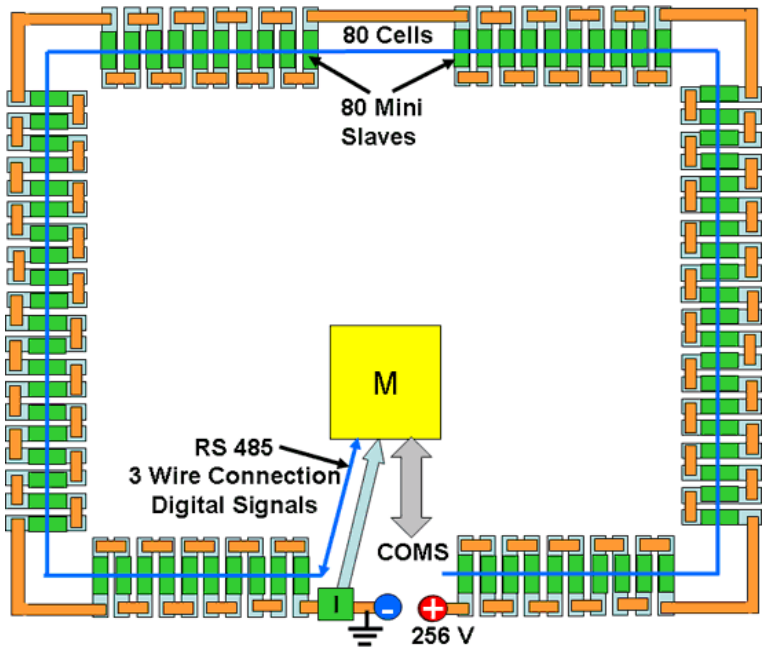

Figura 1.4: struttura a catena ([2])

Capitolo 2

Stato dell'arte

2.1 Battery Management System

Per soddisfare i requisiti tipici di un *BMS* e per risolvere le varie problematiche, sono stati realizzati, negli ultimi anni, molti sistemi che differiscono per scelte architetturali e circuitali.

Il sistema descritto in seguito ([3]), presenta un'architettura a due livelli (vedere Figura 2.1), in cui il gruppo di celle che compone la batteria è gestito da un microcontrollore, che, una volta elaborati i dati, invia i risultati ad un *master module*. Le celle sono connesse tramite degli *switch relays* la cui struttura è mostrata in Figura 2.2.

Tale *BMS* ha due tipi di sensori, uno con uscita analogica, che dopo la multiplazione, giunge ad un convertitore *A/D*, l'altro con uscita digitale, capace di comunicare con il microcontrollore tramite le usuali interfacce: *one-wire, two-wire, SPI,* I^2C. I sensori con uscita analogica misurano le correnti mentre quelli con uscita digitale, principalmente, misurano la temperatura.

Per misurare la corrente che attraversa tutto il pacco batteria è stato utilizzato un sensore ad effetto *Hall*. Le tensioni, relative alle celle e all'uscita dei sensori di corrente, pilotano una rete resistiva di condizionamento. Un mux analogico, controllato dal microcontrollore, seleziona ciascun segnale da misurare, che, dopo un ulteriore condizionamento ad opera di un amplificatore, giunge in ingresso ad un *ADC*.

Per misurare la corrente che attraversa la cella viene utilizzata una resistenza di *shunt*, la quale non viene connessa come in Figura 2.3(a) ma come in Figura 2.3(b), questo per evitare di collegarla tra la connessione di due

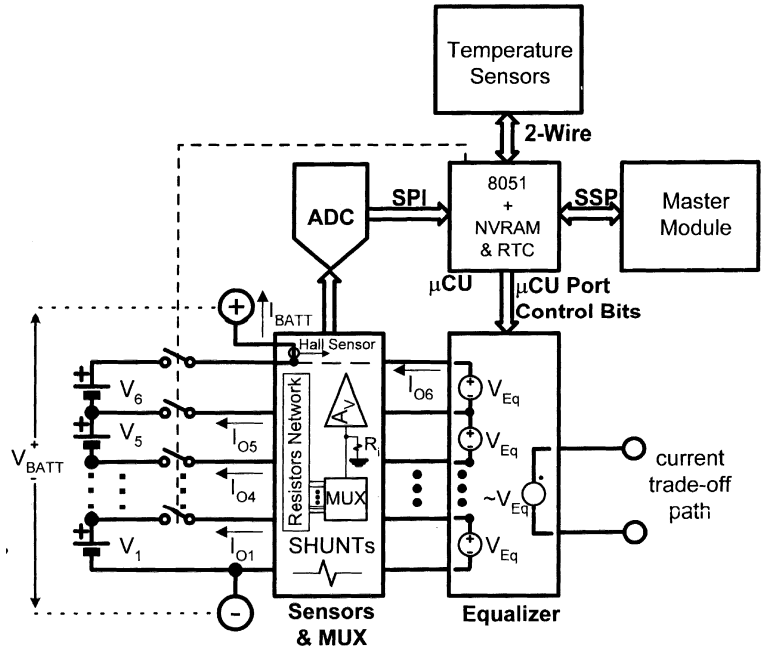

Figura 2.1: struttura ([3])

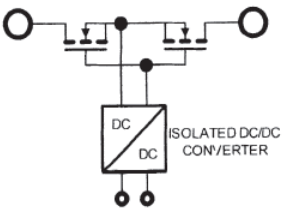

Figura 2.2: switch relays ([3])

celle e la connessione di due convertitori dell'equalizzatore.
Lo svantaggio della connessione in Figura 2.3(b) è quello di avere la resistenza di *shunt* in serie al generatore di equalizzazione, questo rende elevate le perdite di energia. Tale perdita risulta più importante quando sono chiusi solo gli interruttori agli estremi del pacco, dato che la corrente di equilibrio attraversa tutte le resistenze di *shunt*. Comunque la dissipazione risulta piccola perché le resistenze di *shunt* tipicamente sono molto minori della resistenza interna del generatore dell'equalizzatore.
Considerando la Figura 2.3(a):
$I_{C6} = I_{BATT} - I_{O6}$
$I_{C5} = I_{C6} - I_{O5} = I_{BATT} - I_{O6} - I_{O5}$
$I_{C4} = I_{C5} - I_{O4} = I_{BATT} - I_{O6} - I_{O5} - I_{O4}$
In generale:

$$I_{C_n} = I_{BATT} - \sum_{n_{max}}^{n} I_{On}$$

Un problema di questa configurazione, è quello che la corrente non può essere calcolata indipendentemente da quella delle altre celle, quindi l'errore sulla singola misura è legato all'errore delle misure sulle altre correnti. In definitiva l'errore dipende dalla posizione della cella, ed è maggiore per le celle posizionate in alto, perché si sommano gli errori sulle misure delle correnti sottostanti.
Considerando la Figura 2.3(b):

$$I_{C_n} = I_{BATT} - I_{E_n}$$

La misura di corrente, in questa configurazione, ha il solito errore per ogni cella e risulta basso rispetto al collegamento precedente.

Un altro aspetto importante è l'affidabilità delle comunicazioni tra i vari sistemi dotati di intelligenza. A tal proposito, tipicamente, viene utilizzato il *CAN-bus* che garantisce sicurezza e velocità dell'informazione negli ambienti più inquinati elettromagneticamente (Figura 2.4). Nel sistema in analisi il trasferimento dei dati, prelevati da ciascuna cella, avviene tramite un *CAN-bus* denominato *inner-CAN*, mentre lo scambio delle informazioni tra il *master module* e la centralina del veicolo avviene con un secondo *CAN-bus* denominato *outer-CAN* (Figura 2.4).
I moduli *samples module* misurano la tensione di cella e la relativa tempera-

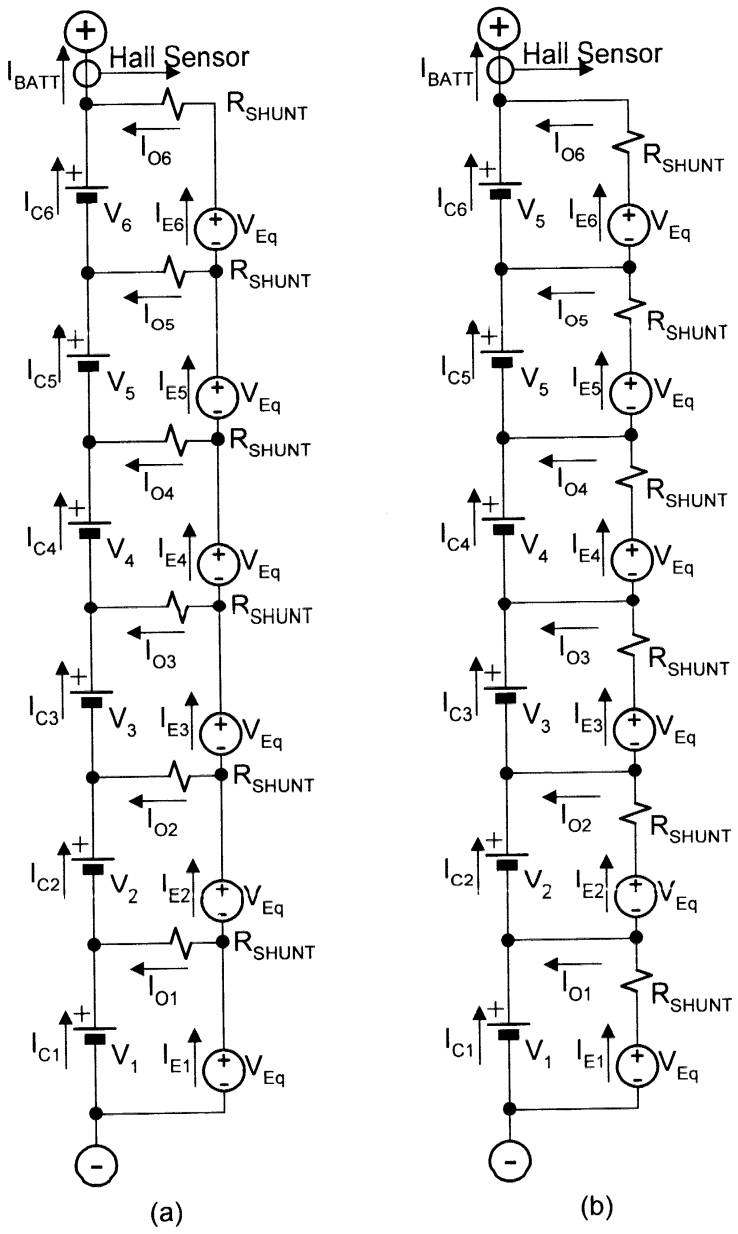

Figura 2.3: posizionamento della resistenza di *shunt* per la misura di corrente ([3])

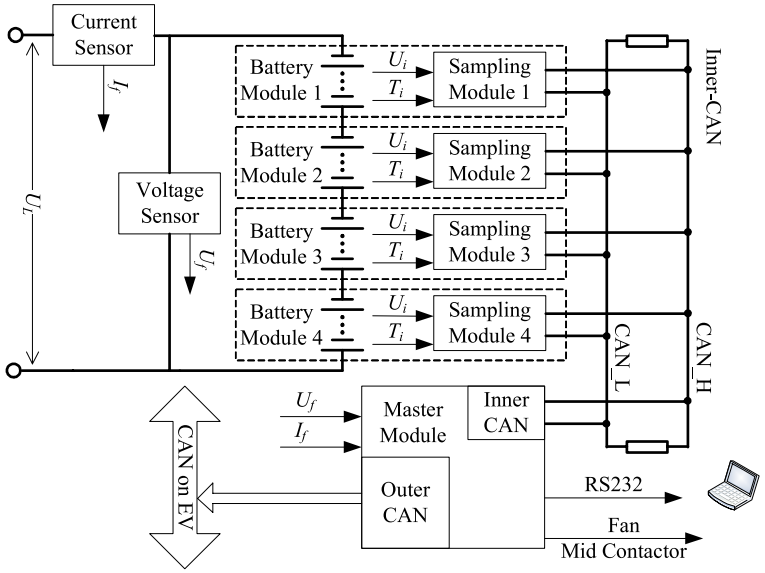

Figura 2.4: architettura ([4])

tura e tramite *inner-CAN*, inviano i dati al *master module*. A quest'ultimo giungono inoltre le misure fatte dal sensore di corrente e quelle del sensore di tensione per la misura della tensione complessiva del pacco.
Il *master module* calcola il *SOC*, anche di ciascuna cella, gestisce le procedure di sicurezza e comunica con la centralina principale tramite *outer-CAN*.

Per selezionare la cella è necessario realizzare un opportuno circuito come quello in Figura 2.5.

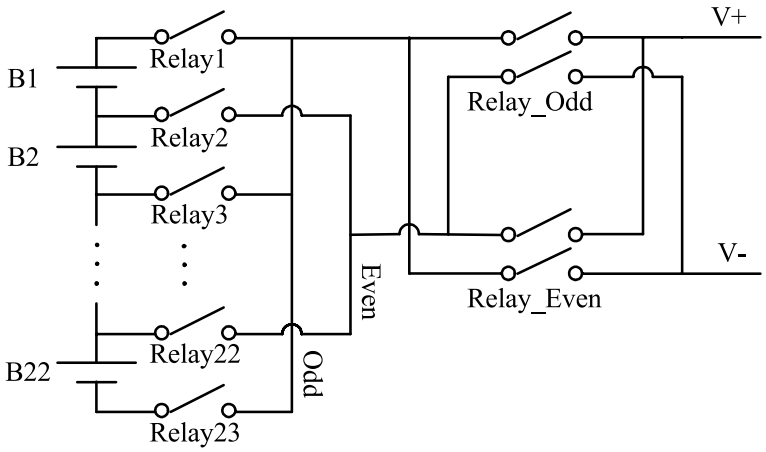

Figura 2.5: circuito per la selezione delle celle ([4])

Solo una cella alla volta può essere collegata al convertitore *A/D* per la misura della tensione. Per effettuare la misura, gli interruttori relativi alla cella (per esempio *Relay1 e Relay2*) vengono chiusi con opportuni segnali, successivamente, se la numerazione della cella è pari, viene chiuso *Relay_Odd* altrimenti *Relay_Even*, in modo tale che la polarità della tensione di cella sia sempre positiva verso l'alto e negativa verso il basso (in altre parole *V+* deve essere sempre il terminale positivo della cella selezionata).

Quando il veicolo è in movimento, l'inquinamento elettromagnetico all'interno è importante, quindi è necessario proteggere dai disturbi il *BMS*. Tra il filo di misura della cella ed il corrispondente relay è inserita una resistenza per limitare l'eventuale corrente di corto circuito, che potrebbe scorrere se i disturbi compromettessero il pilotaggio degli interruttori. Prima del convertitore *A/D*, utilizzato per la misura della tensione di cella, è inserito un amplificatore isolato per ridurre l'interferenza sul filo di misura. I circuiti di comunicazione sono isolati da dei foto-accoppiatori per limitare i disturbi provenienti dalle linee vicine e per proteggere il nucleo del *BMS* da sovra tensioni([4])).

Un altro esempio architetturale di *BMS* è quello mostrato in Figura 2.6. in cui è evidenziata una struttura modulare. Nel sistema analizzato sono

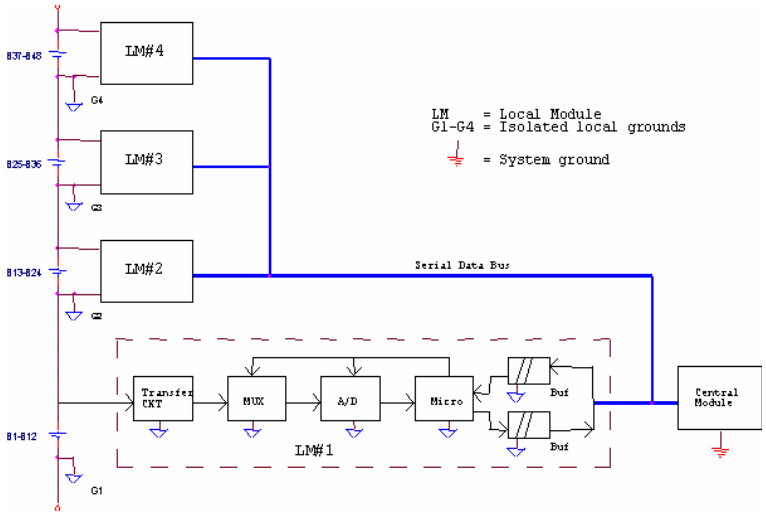

Figura 2.6: architettura modulare del *BMS* ([6])

presenti quattro moduli locali (*LM*), che gestiscono un segmento da dodici celle, e per ciascuna di esse viene misurata la tensione. I moduli locali devono avere il potenziale di massa al solito livello del terminale negativo del

segmento, altrimenti non sarebbe possibile effettuare la misura di tensione della cella. Ciascun modulo locale seleziona la cella tramite un *MUX*, la tensione selezionata viene convertita in digitale tramite un convertitore *A/D* ed il risultato arriva ad un microcontrollore, che, tramite un trasmettitore *CAN-bus* isolato, comunica l'informazione al modulo centrale.

Una caratteristica comune alla maggioranza dei *BMS* presi in considerazione è la modularizzazione. L'architettura viene suddivisa in moduli, per esempio moduli locali, che gestiscono un gruppo di celle, moduli centrali che gestiscono un gruppo di moduli locali, etc.... Questo approccio consente di ottenere una struttura flessibile e di ridurre il numero di connessioni, un vantaggio importante soprattutto in un ambiente elettromagneticamente inquinato come quello di un veicolo. Di contro un sistema così strutturato risulta più costoso, a causa del maggior numero di schede e di componenti. In generale i compiti di un modulo locale sono:

- misura della tensione di cella;
- misura della temperatura di cella;
- comunicazione con il modulo centrale;
- controllo dell'equalizzatore.

I compiti di un modulo centrale sono:

- accensione e spegnimento dei moduli locali;
- memorizzazione ed elaborazione dati;
- misura della corrente del pacco batteria;
- determinazione dello stato di carica;
- gestione della sicurezza del pacco batteria;
- monitoraggio durante la modalità *sleep*;
- bilanciamento della carica tra i segmenti del pacco batteria.
- attivazione di sistemi di raffreddamento (ventole).

Quando il veicolo non è in funzione, il *BMS* dovrà entrare in modalità a basso consumo (modalità *sleep*), e si riattiverà periodicamente per controllare lo stato del pacco batteria.
Una funzionalità importante che deve implementare un *BMS* è quella del bilanciamento delle celle. Dall'analisi delle ultime ricerche, è emerso che l'architettura di maggiore successo è quella di tipo modulare distribuita, la quale presenta i vantaggi e gli svantaggi sopra riportati. Per i circuiti di bilanciamento sembra che non ci sia ancora una strada comune che consenta di ottenere i giusti compromessi tra efficienza energetica, semplicità circuitale e velocità di bilanciamento.

2.2 Bilanciamento attivo

In ambito veicolare, tipicamente, le celle che compongono la batteria vengono connesse in serie. Lo sbilanciamento di tensione tra le varie celle (in teoria identiche) usualmente è causato da differenze di capacità, di resistenza interna e di gradiente di temperatura durante la carica e la scarica. Una cella sbilanciata in una stringa, che compone la batteria, può andare in sotto-scarica o in sovra-carica rischiando il danneggiamento permanente. Una carica eccessiva favorisce la creazione di litio metallico, che è un materiale altamente infiammabile, una carica troppo bassa favorisce la formazione di rame metallico che genera un corto circuito interno con conseguente surriscaldamento ed eventuale esplosione della cella.

La tensione di ciascuna cella deve essere monitorata e deve essere impiegato un circuito di bilanciamento, che consente di mantenere le celle al solito stato di carica, in modo da migliorare il ciclo di vita della batteria. I circuiti di bilanciamento possono essere suddivisi in due categorie (Figu-

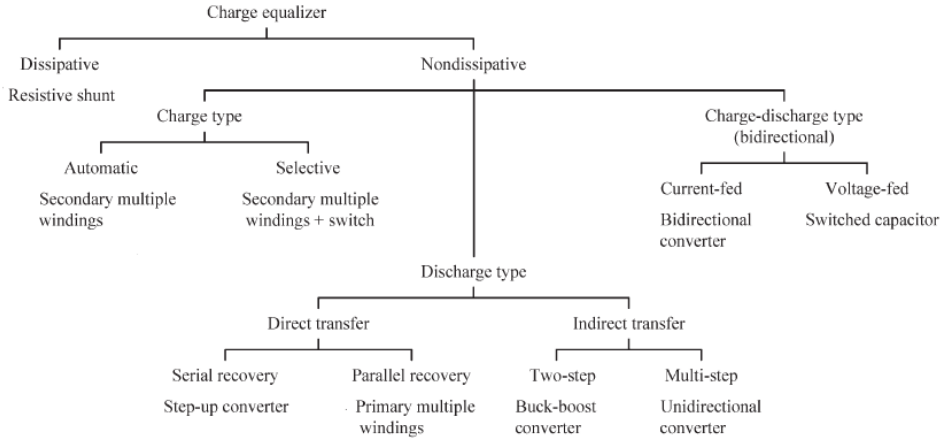

Figura 2.7: tipologie di bilanciamento ([7])

ra 2.7): dissipativi (bilanciamento passivo) e non dissipativi (bilanciamento attivo). In generale un circuito di bilanciamento attivo deve consentire di prelevare energia da una cella più carica per fornirla ad una cella meno carica, limitando il più possibile la dissipazione di energia. Un circuito di bilanciamento passivo preleva l'energia in eccesso da una cella per dissiparla su una resistenza, in tal modo si può ottenere con facilità una stringa di celle

equilibrate tra loro ma con un rendimento nullo. I circuiti di bilanciamento attivo possono essere a loro volto suddivisi in altre categorie:

- ***charge type***:
  - ***automatic***: Figura 2.22(a);
  - ***selective***: Figura 2.19.
- ***discharge type***:
  - ***direct transfer***: per esempio viene utilizzato un convertitore *switching step-up* alimentato da una cella o da una serie di celle per fornire energia alla cella più scarica;
  - ***indirect transfer***: Figura 2.9, Figura 2.8, Figura 2.16.
- ***charge-discharge type***:
  - ***current fed***: l'energia viene trasferita da una cella all'altra alimentando un trasformatore a commutazione (*switching transformer*);
  - ***voltage fed***: Figura 2.23(a).

Di seguito verranno descritti ed analizzati alcuni circuiti che sono classificati nell'elenco sopra.

Un circuito per il trasferimento di energia da una cella all'altra può essere quello riportato in Figura 2.8 (buck-boost equalizer), dove supponendo $Cell_j$ più carica di $Cell_{j+1}$, il bilanciamento avviene accendendo il *MOS* Q_j, l'induttore L_j accumula energia, e quando Q_j viene spento, L_j fornisce corrente a $Cell_{j+1}$ tramite il diodo di substrato di Q_{j+1}. Per il trasferimento di energia nel verso opposto il funzionamento è duale a quello riportato sopra. Uno svantaggio di questo circuito è quello di avere elevate perdite di commutazione, principalmente perché i *MOS* vengono interdetti quando ancora conducono corrente.

Un circuito che tenta di limitare le perdite di commutazione è chiamato *QRZCS* (Quasi Resonant Zero Current Switching) mostrato in Figura 2.9. Fissata la direzione del flusso di energia, il *MOS* viene interdetto quando la corrente che lo attraversa non è al valore massimo (come nel buck-boost in Figura 2.8) ma è ad un valore inferiore, I_0 (Figura 2.11, I_0 è il valore della corrente nel *MOS* quando questo viene interdetto ed è legato all'energia

trasferita alla cella da bilanciare). Quindi a parità di valore massimo di corrente nel *MOS*, il *QRZCS* ha perdite di commutazione inferiori rispetto al *buck-boost* in Figura 2.9 ma è anche più lento perché l'energia trasferita è inferiore.

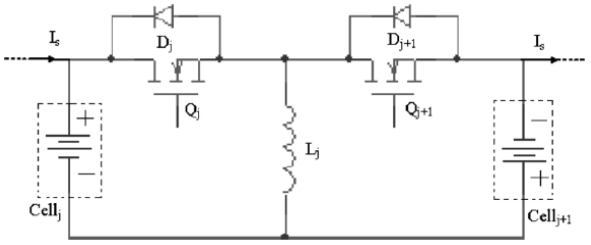

Figura 2.8: circuito *buck-boost* ([8])

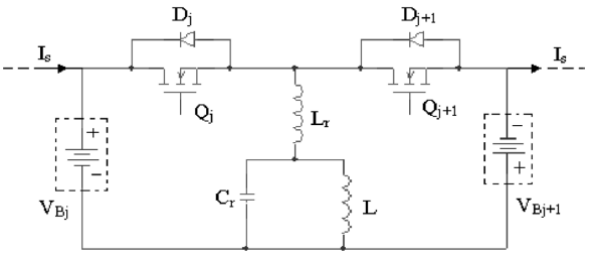

Figura 2.9: circuito *QRZCS* ([8])

Di seguito viene analizzato il comportamento del circuito *QRZCS* in Figura 2.9. Viene assunto che l'induttanza L sia adeguatamente grande in modo tale che la corrente che l'attraversa sia praticamente costante (I_L). Per analizzare il circuito è possibile sostituire tale elemento con un generatore di corrente costante, la cui direzione è determinata dalla modalità di pilotaggio dei due *MOS* (Figura 2.10).

Si considera che $V_{bj} > V_{Bj+1}$, quindi è necessario trasferire energia da V_{bj} a V_{bj+1}.

Fase 1 ($t_0 < t < t_1$)**:**

Per $t = t_0$ viene attivato Q_j e la corrente scorre negli elementi reattivi, i quali immagazzinano energia (Figura 2.12).

Fase 2 ($t_1 < t < t_2$)**:**

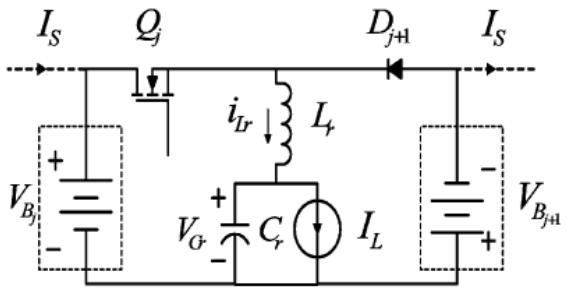

Figura 2.10: schema analizzato per il circuito $QRZCS$([8])

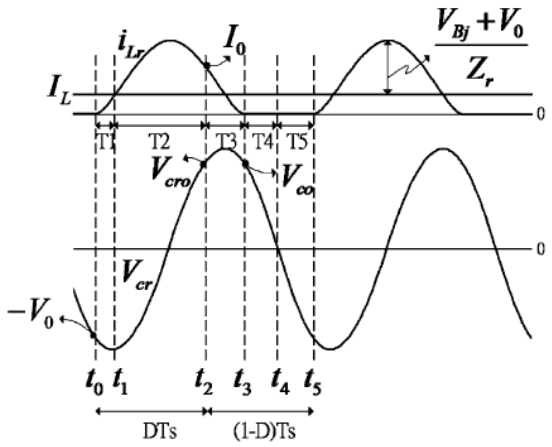

Figura 2.11: segnali del circuito $QRZCS$([8])

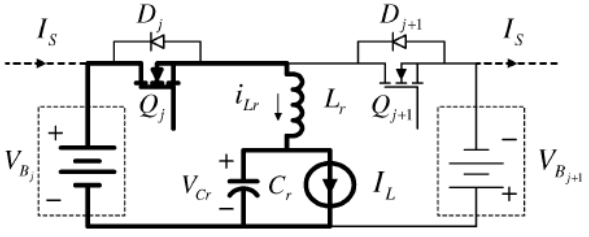

Figura 2.12: Fase 1 ([8])

Fintantochè Q_j è ancora acceso, come mostrato in Figura 2.11, la corrente in L_r cresce fino ad un valore massimo, poi decresce fino all'istante t_2 in cui Q_j si spenge.

Fase 3 ($t_2 < t < t_3$)**:**
Per $t = t_2$, Q_j viene spento ed istantaneamente la corrente scorre attraverso il diodo di substrato D_{j+1} del *MOS*, andando a caricare la cella V_{Bj+1} (Figura 2.13).

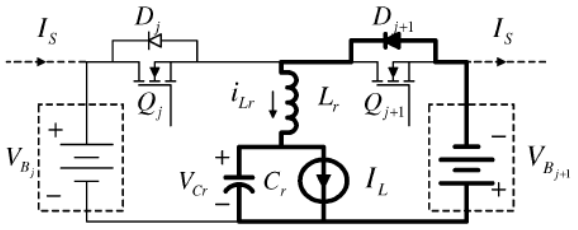

Figura 2.13: fase 3 ([8])

Fase 4 ($t_3 < t < t_4$)**:**
In questa fase i *MOS* e D_{j+1} sono interdetti, la corrente $i_{Lr}(t)$ è nulla e la capacità C_r si scarica a corrente costante I_L fornita dall'induttore L (Figura 2.14):

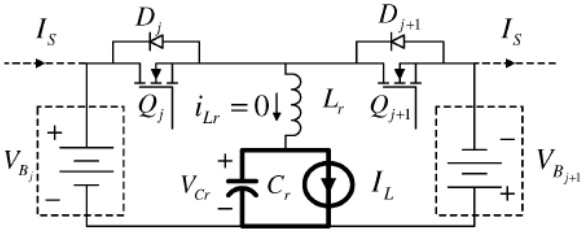

Figura 2.14: fase 4 ([8])

La tensione ai capi del condensatore decresce linearmente fino a raggiungere il valore di V_{Bj+1}.

Fase 5 ($t_4 < t < t_5$)**:**
Quando la tensione ai capi del condensatore C_r raggiunge il valore V_{Bj+1}, il diodo D_{j+1} entra in conduzione. Il circuito equivalente in questa fase è mostrato in Figura 2.15. La tensione ai capi del condensatore V_{Cr} ha un'an-

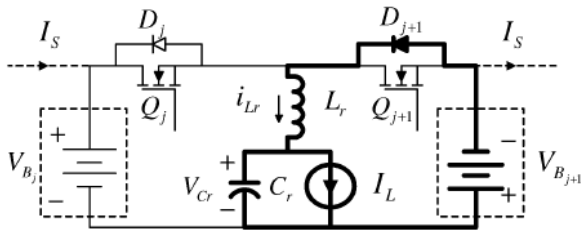

Figura 2.15: fase 5 ([8])

damento sinusoidale che molto velocemente giunge al valore negativo $-V_0$, a tal punto viene acceso nuovamente Q_j per un altro ciclo di scambio. In questa fase la corrente i_{Lr} è molto piccola, in prima approssimazione può essere trascurata.

Pilotando il gate di Q_j con un segnale rettangolare, il circuito è in grado di trasferire energia da V_{Bj} a V_{Bj+1}, analogamente pilotando il gate di Q_{j+1} il circuito preleva energia da V_{Bj+1} e la fornisce a V_{Bj}.

Uno svantaggio del circuito *QRZCS* è quello di avere un tempo di bilanciamento elevato rispetto a quello che si avrebbe con un circuito di tipo *buck-boost* (come quello in Figura 2.8), perché il periodo di commutazione del *MOS* non può essere inferiore a $t_0 - t_5$, tale intervallo può risultare lungo, rispetto al periodo di pilotaggio nel *buck-boost*, a causa dell'elevato valore dei parametri dei componenti reattivi. Inoltre è necessario implementare un *gate driver* per pilotare i *MOS* ed è necessario invertire il verso di una cella per consentire lo scambio di energia. In definitiva il *QRZCS*, per funzionare correttamente, ha bisogno di una complessità hardware superiore ad un circuito *buck-boost* per esempio (come quello in Figura 2.8).

Un altro circuito di bilanciamento, interessante per la facilità con cui è possibile pilotare i due *MOS* (non necessita di un *gate driver*), è quello in Figura 2.16, composto da due induttori L_1 e L_2, una capacità di trasferimento di energia C_1 e due *MOS* di potenza ([9]).

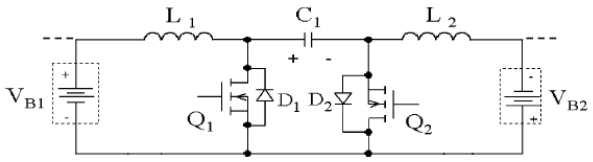

Figura 2.16: circuito analizzato ([9])

In base alla differenza di tensione delle due celle, vengono pilotati i gate dei due *MOS* in modo opportuno per raggiungere un bilanciamento. In particolare se $V_{B1} > V_{B2}$ viene pilotato Q_1 mentre se $V_{B2} > V_{B1}$ viene pilotato Q_2.

Inizialmente: $V_{C1} = V_{B1} + V_{B2}$.
Periodo DT_S $(t_0 < t < t_1)$**:**
quando Q_1 viene acceso l'energia immagazzinata in C_1 viene trasferita a V_{B2} e L_1 immagazzina energia da V_{B1}.
Periodo $(1 - D)T_S$ $(t_1 < t < t_2)$**:**
Q_1 viene spento e D_2 entra in conduzione, il condensatore viene caricato dalla cella V_{B1} e la cella V_{B2} continua a ricevere energia da L_2, che nel tempo si scarica.

Alcuni circuiti di bilanciamento, per equalizzare una cella selezionata con un opportuno circuito, prelevano la corrente da tutta la batteria tramite un convertitore *DC-DC*, come mostrato in Figura 2.17. Gli interruttori di selezione consistono in relay bidirezionali che permettono di ottenere un percorso per la corrente tra il convertitore e la cella selezionata. La selezione deve essere tale da distinguere due percorsi, uno per la cella pari e uno per la cella dispari. La Figura 2.18 evidenzia i due diversi percorsi della corrente tra la cella selezionata ed il convertitore *DC-DC*.
Un esempio circuitale è quello riportato in Figura 2.19, in cui vengono descritte anche le varie fasi di funzionamento. Nella descrizione successiva viene ipotizzato che la cella da bilanciare sia B_3.

- **Fase 1:** il sistema di controllo chiude gli interruttori bidirezionali S_3 e S_4, viene imposto il percorso della corrente di bilanciamento (Figura 2.19(a));
- **Fase 2:** il controllo chiude l'interruttore Q_a, ancora non scorre corrente perché non è stato acceso il convertitore *DC-DC* (Figura 2.19(b));
- **Fase 3:** viene acceso Q_m ed il convertitore viene attivato. La cella viene attraversata dalla corrente di equalizzazione fornita da tutta la batteria (Figura 2.19(c)).

In generale, appena viene acceso Q_m, scorre corrente nel primario del trasformatore che accumula energia. Appena Q_m si interdice, l'energia im-

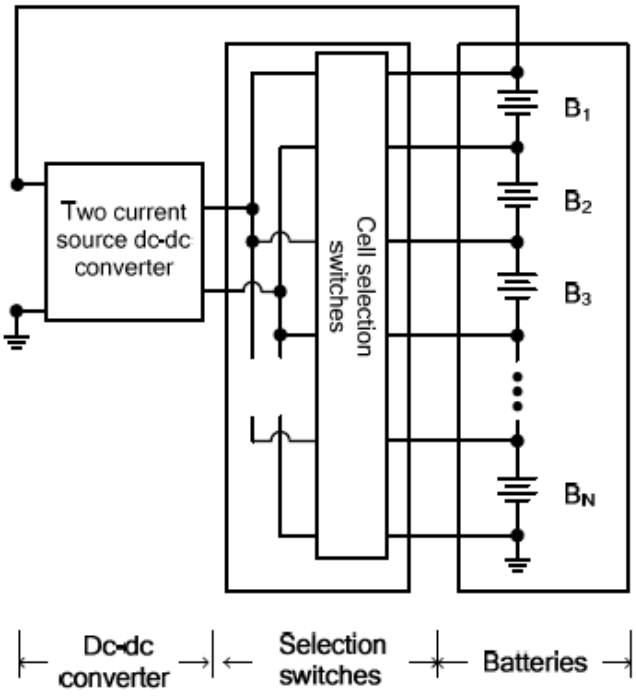

Figura 2.17: schema a blocchi dell'equalizzatore ([10]).

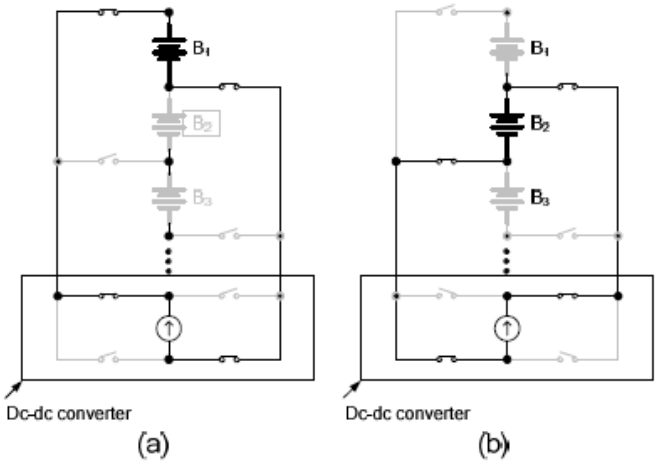

Figura 2.18: principio di funzionamento della selezione a due vie. (a)batteria pari (b)batteria dispari ([10]).

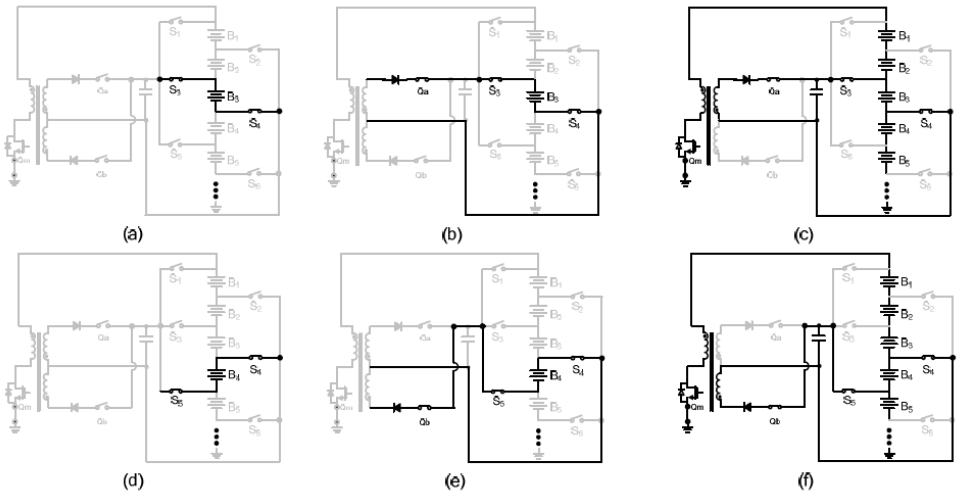

Figura 2.19: fasi per il bilanciamento ([10]).

magazzinata viene liberata dal secondario che fa fluire corrente nella cella attraverso il diodo.

La Figura 2.19(a), (b) e (c) mostrano le fasi per il bilanciamento della cella dispari (per esempio B_3), sostanzialmente la procedura è la solita menzionata sopra, cambia solo la modalità di attivazione degli interruttori.

Finora sono stati analizzati solo circuiti per bilanciare una singola cella, può essere necessario bilanciare tra loro anche i segmenti di celle che compongono la batteria. E' necessario un circuito che consenta di caricare la cella più scarica nel segmento più carico, il quale deve cedere energia al segmento meno carico. Il circuito in questione viene chiamato *equalizzatore modulare*.

La stringa di celle che compone la batteria viene divisa in moduli, vengono utilizzati sia degli equalizzatori per gli stessi sia equalizzatori intramodulo (Figura 2.20 e Figura 2.21). Con tale tecnica è possible sia bilanciare i vari moduli, sia bilanciare la cella più scarica con l'energia fornita dal modulo più carico.

Di seguito vengono descritte, in generale, alcune topologie di equalizzatori modulari effettuando un confronto con l'analogo circuito non modulare (approccio convenzionale).

In Figura 2.22(a) è mostrata una struttura di bilanciamento con un trasformatore formato da un primario e otto secondari. Dopo aver selezionato

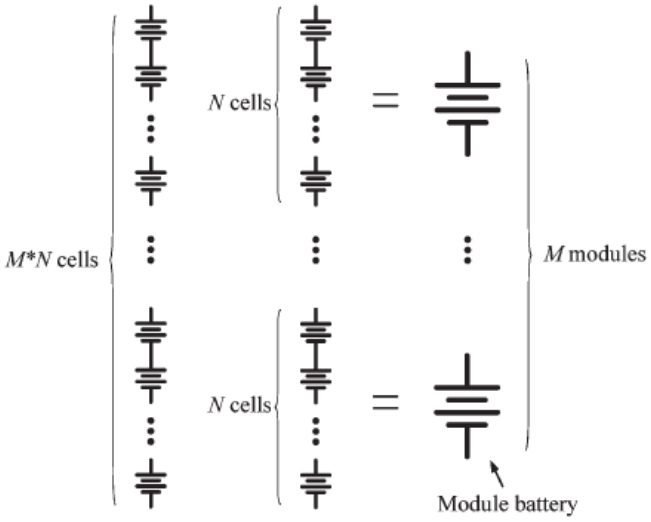

Figura 2.20: Struttura batteria ([11])

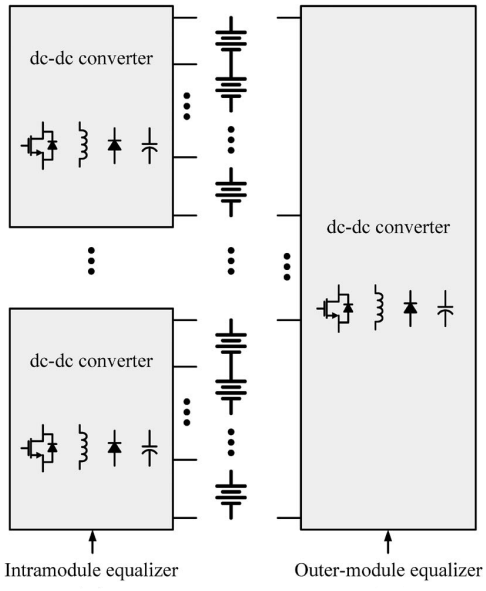

Figura 2.21: Struttura del bilanciamento ([11])

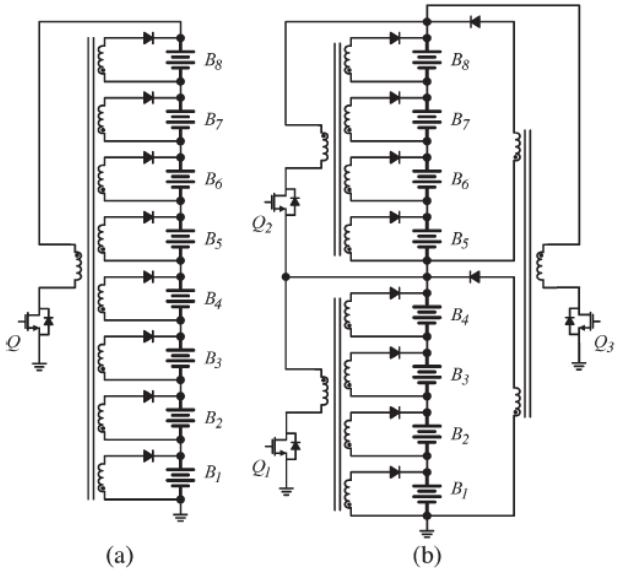

Figura 2.22: confronto sistema con trasformatori (a) approccio convenzionale (b) approccio modulare ([11])

la cella da bilanciare è possibile fornirle energia, la quale viene prelevata da tutta la batteria pilotando opportunamente il *MOS*. In particolare viene selezionata la cella da bilanciare con due interruttori (per esempio opto-isolati), il *MOS* Q viene acceso e l'induttore del primario del trasformatore accumula energia, nel secondario selezionato non scorre corrente perché il diodo è interdetto. Quando Q viene spento, nell'avvolgimento primario non scorre più corrente, ai suoi capi la tensione si inverte di segno ed il secondario fornisce corrente alla cella tramite il diodo.

L'approccio sopra considerato è poco costoso ma presenta problemi di sbilanciamento energetico dovuto alla differenza degli induttori parassiti degli avvolgimenti secondari del trasformatore, questo incide sulla capacità di equalizzazione del sistema. Con un approccio modulare questo problema influenza meno il funzionamento dell'equalizzatore. In Figura 2.22(b) viene mostrato un sistema modulare formato da otto celle e diviso in due moduli. Considerando un caso più realistico, per esempio 80 celle raggruppate in otto moduli, si hanno otto trasformatori con dieci secondari e un trasformatore con otto secondari. Con l'approccio convenzionale sarebbe necessario un trasformatore con 80 secondari, con i relativi problemi di sbilanciamento energetico dovuti agli avvolgimenti parassiti dei secondari.

In Figura 2.23(b) è mostrato un altro esempio di circuito di bilanciamento modularizzato che fa uso di capacità commutanti. La Figura 2.23(a) mostra un equalizzatore a capacità commutanti con un approccio convenzionale. Un possibile funzionamento è il seguente: una capacità viene collegata sui contatti della cella superiore, si carica, successivamente viene collegata sui contatti della cella sottostante per fornirle energia. L'approccio convenzionale può richiedere molto tempo per bilanciare un numero considerevole di celle, dato che ciascun condensatore (con elevata capacità) deve commutare da una cella all'altra per trasferire energia. Il sistema in Figura 2.23(b) implementa un equalizzatore intra-modulo, per scambiare energia tra le celle, e un equalizzatore per lo scambio di energia tra i moduli. Per bilanciare i moduli (riferimento al sistema in Figura 2.23(b)), la capacità viene collegata ai terminali inferiori degli interruttori (a destra), si carica, e successivamente viene collegata ai terminali superiori per caricare il modulo adiacente. Il sistema in Figura 2.23 ha un gran numero di interruttori isolati, men-

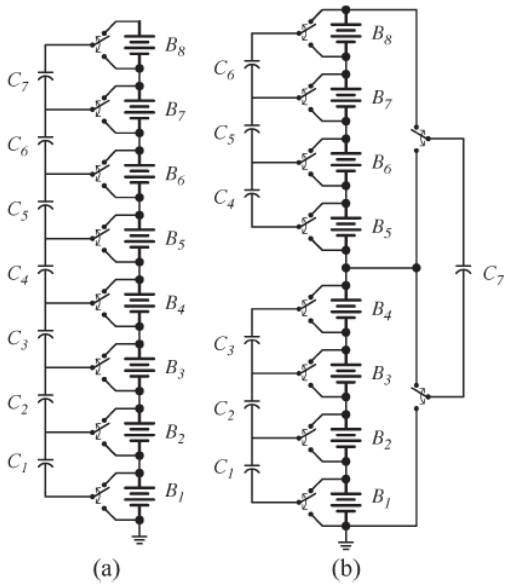

Figura 2.23: confronto sistema con capacità commutanti (a) approccio convenzionale (b) approccio modulare ([11])

tre nel sistema in Figura 2.22 il numero di interruttori isolati è ridotto ma i *MOS* sono sottoposti ad elevati impulsi di tensione (soprattutto Q_3 in Figura 2.22(b)).

Una soluzione di compromesso tra, minimizzare il numero di interruttori

e ridurre i picchi di tensione dei *MOS*, può essere quello mostrato in Figura 2.24. L'equalizzatore intra-modulo è basato sull'utilizzo di trasformatori

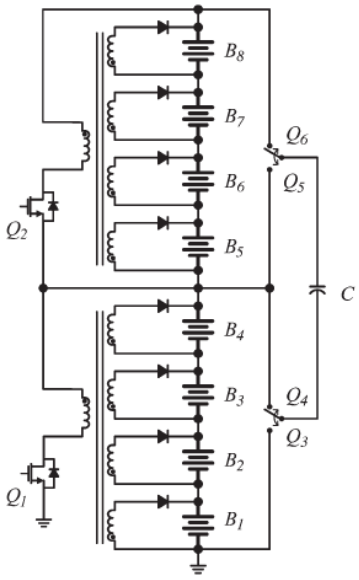

Figura 2.24: Equalizzatore con approccio misto ([11])

mentre quello extra-modulo è basato su capacità commutanti. I possibili vantaggi di tale struttura possono essere i seguenti:

- il bilanciamento intra-modulo è ottenuto a basso costo utilizzando dei trasformatori;
- il problema della differenza tra le induttanza parassite dei secondari dei trasformatori è ridotto, perché un unico trasformatore ha un numero di secondari minore rispetto a quelli che avrebbe in un approccio convenzionale;
- l'utilizzo di capacità commutanti, per il bilanciamento tra i moduli, consente di ridurre le sovratensioni che si avrebbero sui *MOS* in un approccio come quello in Figura 2.22.

2.2.1 Tabelle riassuntive

In Tabella2.1 ed in Tabella2.2 vengono riassunti i vantaggi e gli svantaggi dei sistemi descritti in precedenza.

Circuito	**Vantaggi**	**Svantaggi**
buck-boost	tempo di bilanciamento basso	elevate perdite di commutazione
QRZCS	basse perdite di commutazione	tempo di bilanciamento alto
Circuito L-C	pilotaggio semplice	elevate perdite di commutazione

Tabella 2.1: caratteristiche circuiti di bilanciamento *cella-cella*

Circuito	**Vantaggi**	**Svantaggi**
a trasformatori	tempo di bilanciamento basso basso numero di deviatori	impulsi di tensione
a condensatori	assenza di impulsi di tensione	tempo di bilanciamento elevato elevato numero di deviatori

Tabella 2.2: caratteristiche circuiti di bilanciamento modulare

Capitolo 3

Piattaforma di sviluppo per un *BMS*

3.1 Descrizione generale

Lo sviluppo e la realizzazione di un *BMS* è un tema di ricerca caldo con problematiche ancora aperte. Attualmente non sono ancora state trovate soluzioni soddisfacenti in termini di consumo di energia e di strategie di bilanciamento delle celle.
Le problematiche aperte sono relative agli scopi per cui viene sviluppato un *BMS*, di seguito ne vengono elencate alcune riferite all'ambito *automotive*:

- studiare un'architettura che consenta di realizzare un *BMS* efficiente, dal punto di vista delle funzionalità, minimizzando il costo complessivo;
- ridurre al minimo l'assorbimento di potenza del *BMS* dal pacco batteria;
- studiare un circuito ed una strategia per bilanciare le celle con la massima efficienza energetica nel minor tempo possibile.

Per caratterizzare al meglio un *BMS* è necessario realizzare un sistema che consenta di studiare diverse soluzioni architetturali e di confrontare le varie strategie di equalizzazione, con un occhio di riguardo all'efficenza energetica. Una piattaforma per lo sviluppo di un *BMS* deve consentire di provare varie soluzioni architetturali, vari circuiti di bilanciamento e deve comprendere tutti i dispositivi dedicati alla misura delle grandezze elettriche e alla

temperatura. In questo ambito è importante che il sistema sia flessibile in modo da includere il maggior numero possibile di soluzioni implementabili.

Per realizzare la piattaforma sono state scelte delle celle di tipo *Litio-polimeri* con capacità e dimensioni ridotte rispetto a quelle utilizzate nei veicoli ibridi e puramente elettrici. La scelta delle celle è stata tale da poter utilizzare i consueti strumenti di misura presenti in laboratorio, altrimenti sarebbe stato necessario utilizzare della strumentazione per elevate potenze. Il pacco è composto da tre batterie, ciascuna delle quali è composta da 14 celle *Li-poly* collegate in serie. La scelta del numero di celle, che compongono la batteria, è stata fatta in base all'andamento della funzione di costo dei foto-accoppiatori utilizzati per selezionare le celle dal pacco batteria. In particolare il costo aumenta al crescere della tensione di isolamento e per tensioni superiori a $60V$ la pendenza della funzione aumenta drasticamente, quindi è stato scelto questo punto per fissare la tensione massima della batteria. La totalità delle celle consente di avere una tensione ai capi del pacco adeguata per applicazioni automobilistiche, in particolare è compresa tra $113V$ e $176V$.
L'architettura della piattaforma in fase di sviluppo e realizzazione, mostrata in Figura 3.1, è di tipo modulare con tre livelli di gerarchia. Una struttura gerarchica permette di avere una maggiore flessibilità, in quanto è possibile far lavorare il sistema sia con tre livelli di gerarchia sia con due, è possibile distribuire il carico di elaborazione secondo le funzionalità dei livelli ed inoltre si ha un importante riduzione del numero di collegamenti. Per avere un ulteriore grado di flessibilità ciascun livello della gerarchia contiene un circuito dotato di intelligenza, questo consente di implementare la maggior parte delle funzioni tipiche di un *BMS*. Il livello inferiore, a stretto contatto con la cella, consiste in una scheda contenente un microcontrollore *ATtiny32* ed è denominato *Level0*. Ad ogni cella che compone la batteria viene collegato un *Level0* (in totale ci sono tre gruppi da 14), l'insieme di essi è gestito da un'altra scheda, a livello gerarchico superiore, denominata *Level1*. Ciascun *Level1* riceve le informazioni inviate dal *Level0*, le elabora e dirige le varie operazioni di controllo e gestione della singola batteria. Per motivi che verranno chiariti in seguito il *Level1* è otticamente isolato dai *Level0*, quindi quest'ultimo per inviare i dati utilizza dei foto-accoppiatori.

I tre *Level1* fanno capo ad una scheda denominata *Level2* su cui è implementato il modello della cella. Il *Level2* gestisce tutte le operazioni legate

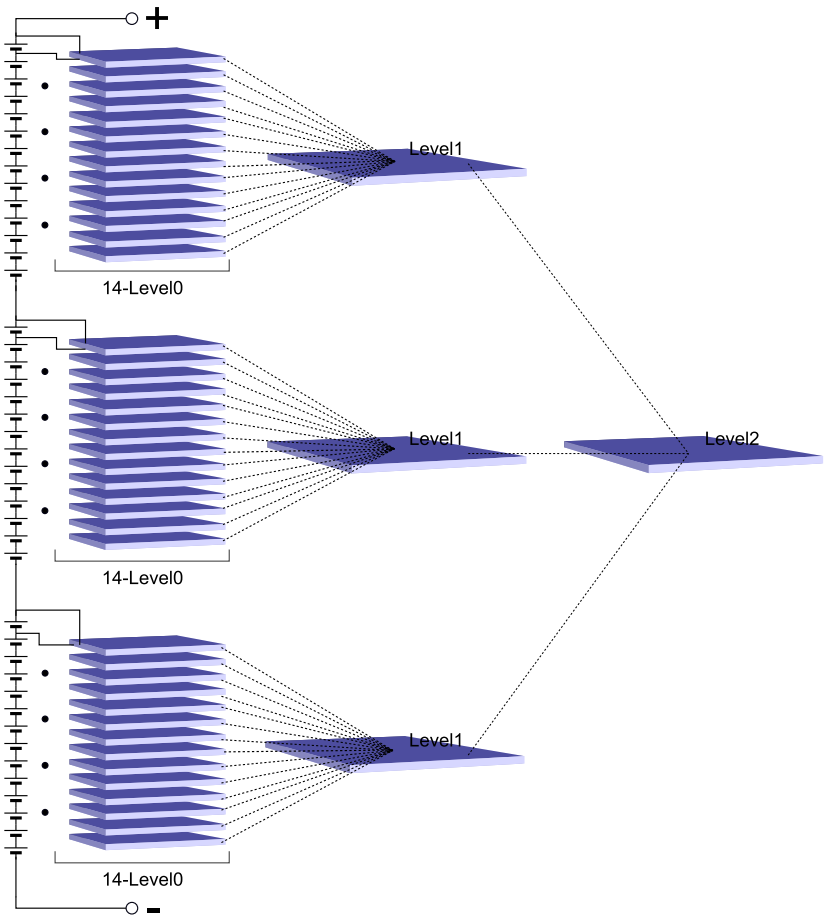

Figura 3.1: struttura *BMS*

al pacco batteria, ad esempio il bilanciamento tra i segmenti, scollega una parte di batteria in caso di danneggiamento. I tre *Level1* comunicano con il *Level2* con il protocollo *CAN-bus*, che tipicamente viene utilizzato nelle applicazioni in cui l'informazione deve viaggiare in un ambiente inquinato elettromagneticamente.

Funzioni del *Level0*:

- misura di tensione della cella a cui è collegato;
- misura di temperatura della cella;
- quando richiesto dal *Level1*, il *Level0* gli invia i risultati delle misure dopo averli convertiti in digitale;

Funzioni del *Level1*:

- misura sequenziale della tensione di ciascuna cella;
- misura della corrente che scorre nella batteria;
- misura della temperatura di batteria;
- bilanciamento tra celle;
- interfacciamento per lo scambio di dati con il *Level2* ed il *PC*.

Funzioni del *Level2*:

- calcola lo stato della cella attraverso il modello;
- gestisce il bilanciamento tra i segmenti di batteria;
- tenta di risolvere gli eventuali problemi senza disturbare il livello gerarchico superiore (per esempio l'utente);
- etc....

Il *Level0* è alimentato dalla cella a cui è collegato mentre il *Level1* è alimentato da una cella esterna (detta anche ausiliaria). Durante il funzionamento il *Level1* interroga un *Level0* il quale misura la tensione e la temperatura di cella ed invia le informazioni. Il *Level1* elabora le informazioni ricevute e le memorizza in una zona di memoria per inviarle successivamente al *Level2*.

Quest'ultimo utilizza i dati della cella per calcolare lo stato di carica tramite il modello e decide di conseguenza la strategia di riequilibrio. Una volta stabilita la strategia di riequilibrio il *Level1* la mette in atto, seleziona dalla batteria la cella da bilanciare e tramite un circuito di equalizzazione controlla lo scambio di energia tra la cella ausiliaria e quella selezionata. Il *Level1*, oltre ai vari circuiti di controllo e di selezione delle celle, è dotato di circuiti che misurano, con elevata precisione, la tensione della cella selezionata dalla batteria, la tensione della cella ausiliaria, la corrente di bilanciamento e la corrente che scorre nel pacco batteria.

La struttura gerarchica della piattaforma consente di distribuire il carico di elaborazione sui tre livelli senza appesantirne uno rispetto agli altri. Ciascun livello di gerarchia è isolato ed indipendente, nel senso che non conosce i dettagli implementativi del livello sottostante.

La progettazione della piattaforma è stata tale da garantire i requisiti di sicurezza e affidabilità (descritti nel Capitolo 1) tipici di un *BMS*. In caso di malfunzionamenti dei dispositivi programmabili, essi possono essere resettati dal livello di gerarchica superiore, quindi ciascun livello può essere ripristinato da quello subito superiore. Questo è uno dei vantaggi di un'architettura modulare a più livelli di gerarchia, infatti una struttura così fatta non solo garantisce un'elevata flessibilità ma consente di avere un sistema più affidabile e sicuro. Come già menzionato sopra sicurezza e affidabilità sono due caratteristiche importantissime per un *BMS* che spesso si scontrano con i requisiti di costo.

3.1.1 Realizzazione meccanica

Ciascun *Level0* deve essere a stretto contatto con la propria cella per facilitare sia la misura di tensione sia la misura di temperatura, a tal proposito è stato realizzato un apposito contenitore mostrato in Figura 3.2 e Figura 3.3).
Ciascun contenitore viene collegato in verticale alla scheda del *Level1* tramite un connettore come mostrato in Figura 3.4 e Figura 3.5. Questo tipo di collegamento consente di avere una struttura solida, compatta e pratica in fase di test e debug.

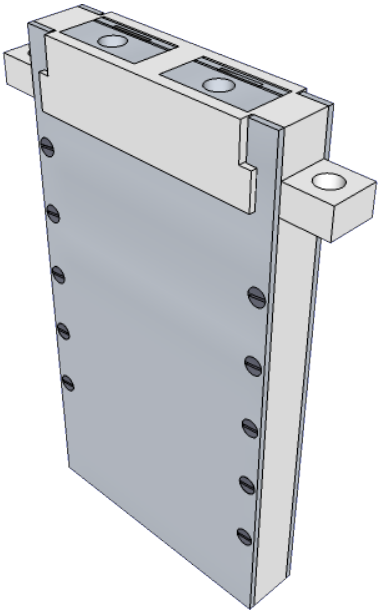

Figura 3.2: contenitore cella e *Level0*

.

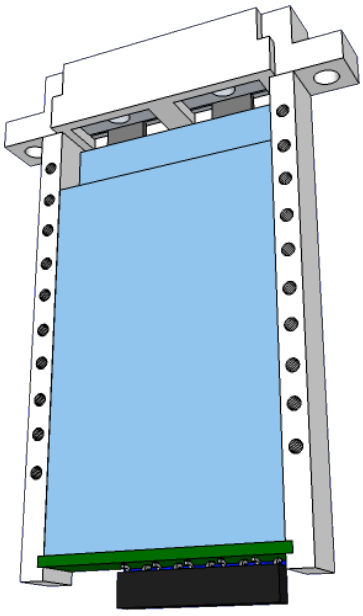

Figura 3.3: contenitore cella e *Level0*

.

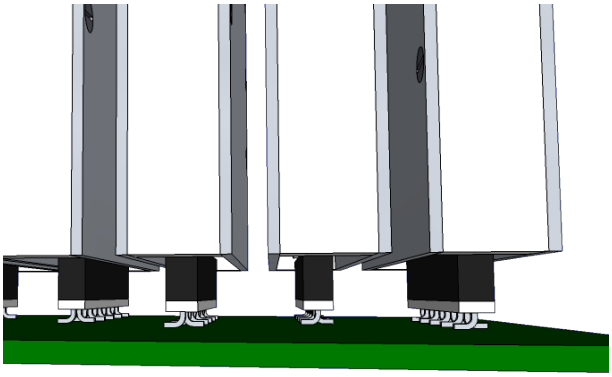

Figura 3.4: connessione *Level0* a *Level1*

.

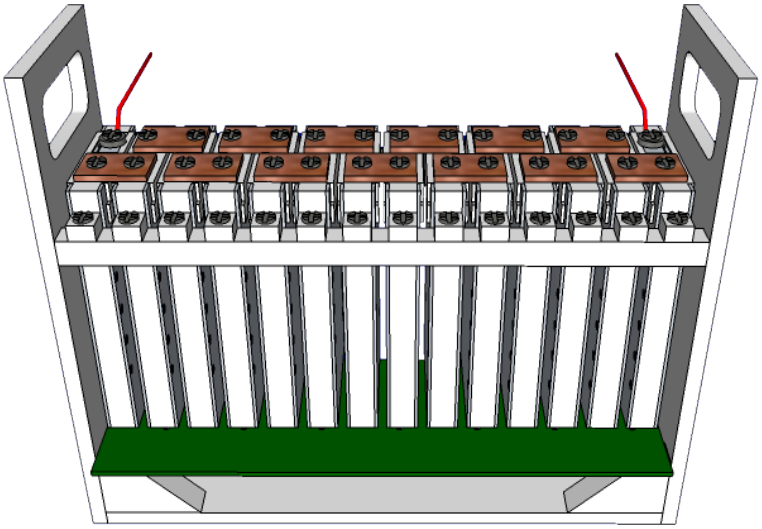

Figura 3.5: struttura completa

.

3.2 Level0

Le funzioni principali del *Level0* sono quelle di misurare tensione e temperatura della cella e di memorizzare informazioni utili, ad esempio età della cella, numero di serie, parametri di correzione per le misure. Per implementare queste funzionalità è necessario un dispositivo flessibile come il microcontrollore *ATtiny13A*. Quest'ultimo è un dispositivo compatto facilmente piazzabile in una scheda di dimensioni ridotte. La comunicazione tra *Level0* e *Level1* avviene tramite tre foto-accoppiatori, questo garantisce un isolamento ottico tra i due sistemi. Per misurare la temperatura della cella è stato scelto un sensore compatto ed a basso costo come un *NTC*.
Il *Level0* invia le informazioni al *Level1* utilizzando il protocollo di trasmissione *SPI*, in cui il primo è impostato come *master* mentre il secondo è impostato come *slave*. La comunicazione tra i due è bidirezionale, il *Level0* invia i risultati delle misure di tensione e di temperatura della cella mentre il *Level1* può scrivere delle informazioni utili (come numero di serie della cella, età della cella,...) nella *EEPROM* del microcontrollore *ATtiny32*. Il *Level0*, oltre ad inviare i dati sulla linea *MOSI*, invia al *Level1* il segnale *SCK* per sincronizzare i dati. Tipicamente è la periferica esterna ad inviare al master il segnale di sincronizzazione, in questo caso non è possibile perché il microcontrollore nel *Level0* non contiene la periferica *SPI* (una sincronizzazione software creerebbe dei problemi).

3.2.1 Circuito

Ciascun *Level0* è alimentato dalla propria cella che compone la batteria, è importante quindi limitare il più possibile l'assorbimento di corrente. Inoltre, la realizzazione della scheda, ha seguito i vincoli di basso costo e di minima occupazione di area per poterla inserire in un opportuno contenitore (porta-contatti).

In Figura 3.6 è mostrato lo schema elettrico del *Level0*:
Il circuito viene collegato alla scheda del *Level1* tramite il connettore *J3*, da cui arriva la richiesta di invio delle informazioni e partono i dati relativi alla tensione e temperatura della cella. Per consentire lo scambio dei dati sono stati utilizzati dei transistor foto-accoppiati (*FOD617*), perché il riferimento di tensione del *Level1* è diverso da quello del *Level0* (a meno che il *Level1* selezioni proprio la cella a cui sono riferite le informazioni scambiate).

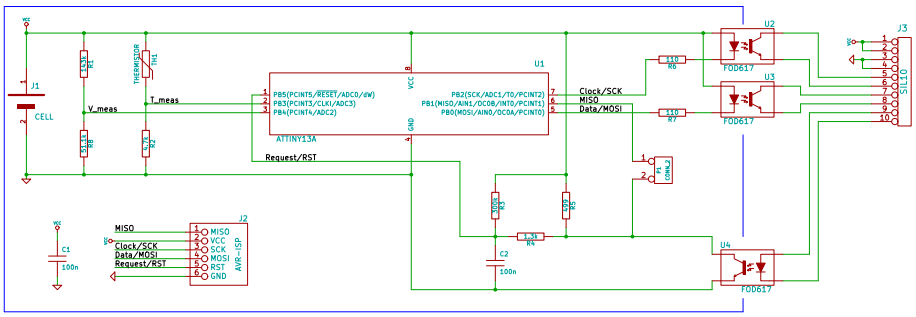

Figura 3.6: Circuito *Level0*

In base al numero di impulsi, che il *Level1* invia pilotando il diodo del foto-accoppiatore, il *Level0* risponde con tre tipologie di dati diverse:

- **1 impulso:** temperatura e tensione della cella;
- **2 impulsi:** numero di serie della cella;
- **3 impulsi:** età della cella;
- grazie alla flessibilità introdotta dai componenti programmabili possono essere implementati altri comandi.

Gli impulsi di richiesta di invio dati, controllano direttamente il pin *PB1* del microcontrollore ed indirettamente il pin di *reset* attraverso una rete passiva formata da *R3, R4, R5* e *C2*. Tale rete consente di resettare il dispositivo programmabile con un impulso basso di durata maggiore rispetto a quelli per la richiesta dati. Ai fini dell'affidabilità del sistema è importante avere la possibilità di resettare il microcontrollore con un segnale proveniente dal *Level1*. In Figura 3.7 è mostrato il circuito per la simulazione *spice* della rete passiva con segnale, attivo basso, di richiesta dati e di reset.
Pilotando il diodo del foto-accoppiatore U_4 con un impulso di durata di 10μS si ottengono i risultati mostrati in Figura 3.8 per $V_{batt} = 3V$ e $V_{batt} = 4.2V$.

Sia per $V_{batt} = 3V$, sia per $V_{batt} = 4.2V$ il segnale di richiesta dati ($V_{request}$) oltrepassa in basso la soglia di riconoscimento del microcontrollore ($0.2 \cdot V_{cc}$) mentre il segnale di reset (V_{reset}) rimane, in entrambi i casi, sufficientemente alto da non andare nella zona indeterminata compresa tra $0.1 \cdot V_{cc}$ e $0.9 \cdot V_{cc}$. In Figura 3.9 sono mostrati l'andamenti di V_{reset} e $V_{request}$ per $V_{batt} = 3V$ e $V_{batt} = 4.2V$, con un impulso di pilotaggio di U_3 di durata di $1mS$.

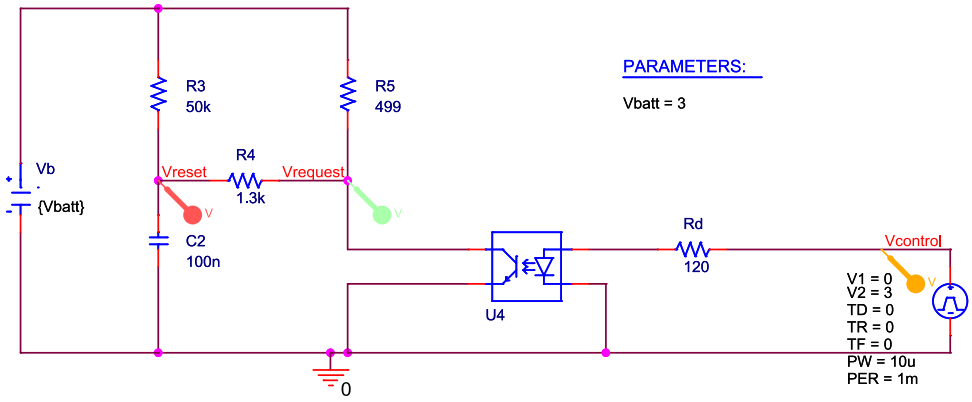

Figura 3.7: circuito per la simulazione della rete passiva con segnale di richiesta e di reset.

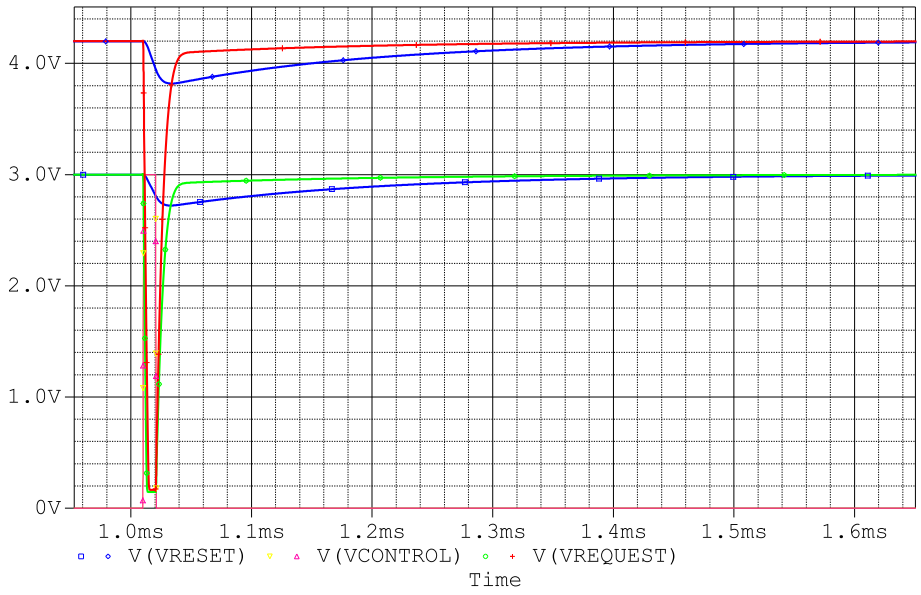

Figura 3.8: risultato della simulazione *spice* parametrica della rete passiva con impulso di richiesta, con $V_{batt} = 3V$ e $V_{batt} = 4.2V$

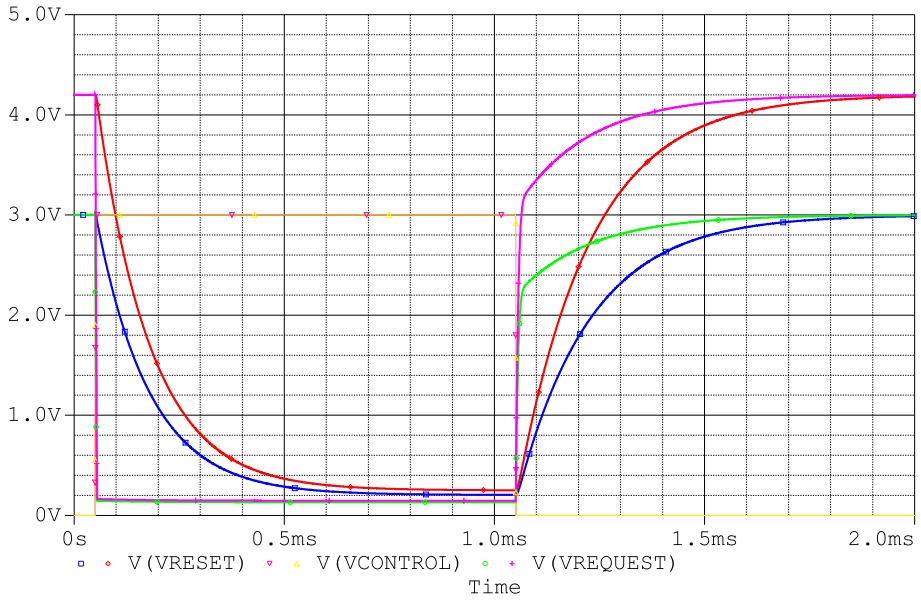

Figura 3.9: risultato della simulazione *spice* parametrica della rete passiva con impulso di reset, con $V_{batt} = 3V$ e $V_{batt} = 4.2V$

.

Per entrambi i valori di V_{batt}, si nota come i due andamenti di V_{reset} oltrepassano la soglia bassa di attivazione del reset del microcontrollore ($0.1 \cdot V_{cc}$). In definitiva, se il *Level1*, tra i pin 9 e 10 di J_3, invia un impulso di tensione di durata pari a $1mS$, il microcontrollore viene resettato, se invia uno o più impulsi, a seconda dell'informazione richiesta al *Level0*, di durata pari a 10μS, il microcontrollore non viene resettato e risponde alla richiesta di invio dati.

Per misurare la tensione della cella viene utilizzato l'ADC interno al microcontrollore ($ATTINY13A$) con riferimento interno pari a $1.1V$. La tensione di cella viene adattata alla dinamica dell'ADC tramite un partitore di tensione formato da $R1$ e $R8$, dimensionato in modo che, per $0 < V_{batt} \leq 4.2$ si ha $0 < V_{meas} \leq 1.1V$. Per misurare la temperatura viene utilizzata una resistenza variabile a coefficiente di temperatura negativo (NTC), collegata in un partitore con $R2$. Per ottenere i dati relativi alla tensione di cella ed alla temperatura, l'ADC converte in digitale le tensioni ai nodi V_{meas} e T_{meas} su 8 bit ed invia il dato al *Level1*.
In Figura 3.10 viene mostrato l'andamento della resistenza dell'NTC al variare della temperatura, ricavato dal modello analitico i cui parametri sono

riportati nel datasheet:

$$R_{NTC} = 47000 \cdot e^{(-15.5+\frac{5230}{T}-\frac{160451}{T^2}-\frac{5400000}{T^3})}\Omega$$

Dove T è in gradi *Kelvin*.

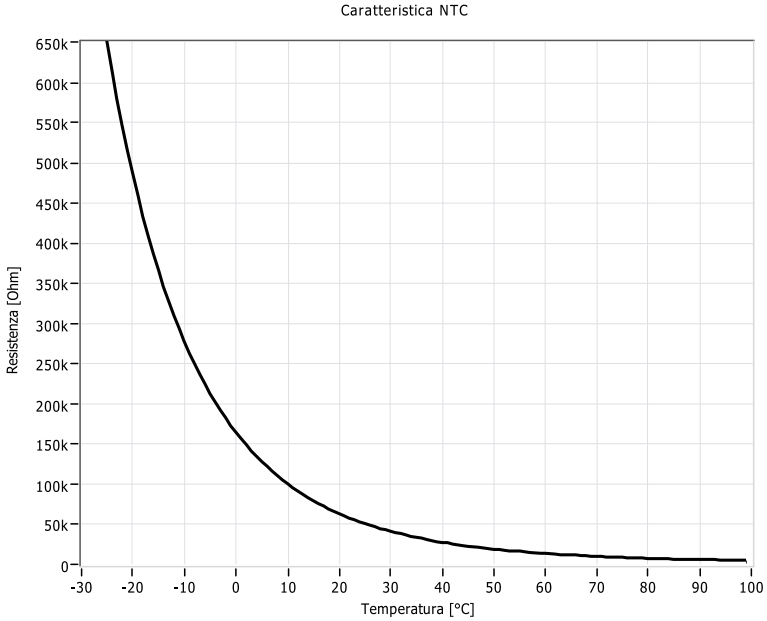

Figura 3.10: andamento della resistenza dell'*NTC* al variare della temperatura

.

In Figura 3.11 è mostrata la tensione sul nodo T_{meas} al variare della temperatura, per $V_{batt} = 3V$ e $V_{batt} = 4.2V$. L'andamento è stato ottenuto utilizzando, nella formula del partitore di tensione, il modello dell'*NTC*:

$$V_{T_{meas}} = V_{batt} \cdot \frac{R_2}{R_2 + 47000 \cdot e^{(-15.5+\frac{5230}{T}-\frac{160451}{T^2}-\frac{5400000}{T^3})}} \tag{3.1}$$

La tensione $V_{T_{meas}}$ è proporzionale a V_{batt} ed ha un andamento non lineare, quindi il *Level1* dovrà tener conto di queste caratteristiche per ricavare con precisione la temperatura della cella.

Il *Level0* invia i dati al *Level1* pilotando due foto-accoppiatori, U_2 e U_3, tramite i pin *PB2* e *PB0* (i rispettivi nodi sono denominati *Clock/-SCK* e *Data/MOSI*, rispettivamente) del microcontrollore. La trasmissione

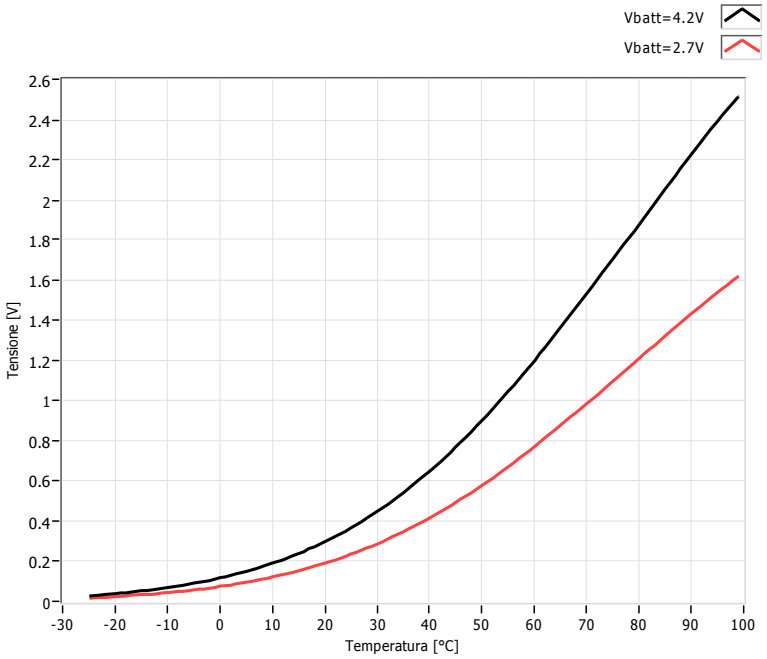

Figura 3.11: andamento della tensione T_{meas} al variare della temperatura

.

utilizzata è di tipo seriale *SPI*.

3.2.2 Realizzazione

Le dimensioni del circuito stampato sono tali da poterlo inserire in un porta contatti, come quello in Figura 3.2 e Figura 3.3, contenente anche la cella che lo alimenta.

In Figura 3.13 e in Figura 3.12, sono mostrati i layout del *PCB*, rispettivamente del lato *back* e del lato *front*.

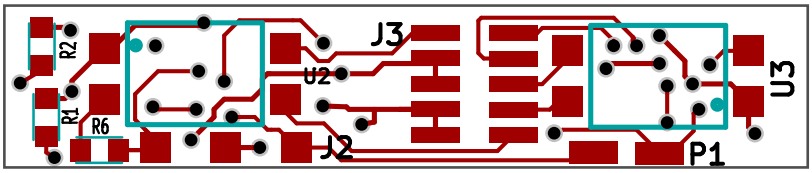

Figura 3.12: *PCB* del *Level0* lato *front*

.

La difficoltà principale, nel realizzare il circuito stampato, è stata quella relativa al piazzamento ed al routing a causa del poco spazio disponibile.

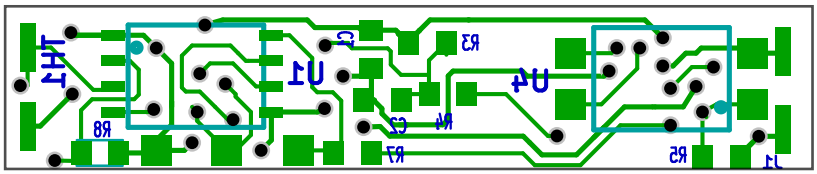

Figura 3.13: *PCB* del *Level0* lato *back*

.

Per far fronte a questa problematica sono stati utilizzati tutti componenti a montaggio superficiale. Gli unici vincoli di piazzamento sono stati quelli relativi al connettore *J3* ed alle piazzole per la programmazione (*J2*) del microcontrollore. Per quanto riguarda *J3* la posizione è vincolata da questioni di montaggio dei vari porta-conttati sulla scheda del *Level1*, per quanto riguarda *J2* è fondamentale che sia piazzato ai bordi del *PCB* per poter programmare il microcontrollore facendo toccare i pin di un connettore direttamente sulle piazzole.

Il dimensionamento delle piste, con uno spessore del rame di 35μm, è stato realizzato in base alla corrente massima e alle regole di layout fornite dal costruttore, in particolare, piste di segnale e di potenza di spessore: $W = 0.2mm$.

Le dimensioni delle vie, uguali per le piste di segnale e di potenza, sono state ricavate seguendo la formula 3.2, ottenuta eguagliando la superficie laterale della pista con la superficie superiore del cilindro di rame che realizza la metallizzazione nel foro.

$$D2 = \frac{W1}{\pi} \cdot \frac{T1}{T2} - T2[13] \tag{3.2}$$

I parametri *D2*, *W1*, *T1* e *T2*, sono specificati in Figura 3.14.

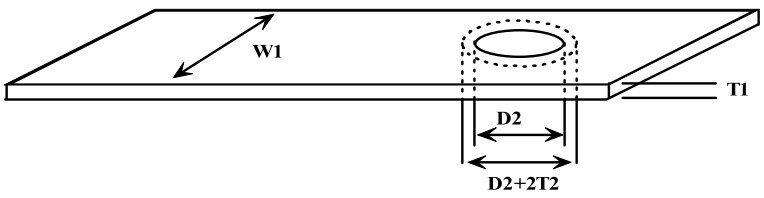

Figura 3.14: Immagine con i parametri dell'equazione 3.2 [13]

.

3.2.3 Firmware

Il microcontrollore, dopo avere effettuato le operazioni di inizializzazione (attivazione interrupt esterni, direzione delle porte di uscita, impostazione dell'*ADC*... Pag.86 da riga 32 a 51), entra in modalità *sleep* di tipo *power down* (Pag.86 riga 61), in cui toglie il clock alle periferiche e disattiva la maggior parte dei moduli interni, tranne quelli che consentono il risveglio tramite interrupt esterni.
Il microcontrollore viene attivato quando riceve, dal *Level1*, un impulso attivo basso sul pin *PB1* (*PCINT1*), va ad eseguire la routine di interrupt esterno (Pag.86 da riga 21 a 25), in cui azzera il timer *TCNT* ed incrementa un registro che memorizza il numero di impulsi ricevuti. Successivamente rimane in attesa attiva (Pag.86 da riga 65), contando tramite l'interruzione il numero di impulsi ricevuti (Pag.86 da riga 21 a 25), fintantochè il timer *TCNTO* va in overflow. A tal punto, dall'ultimo impulso ricevuto, sono trascorsi circa $3.5mS$, un tempo sufficientemente lungo per considerare la trasmissione conclusa, dato che gli impulsi sono intervallati da $1mS$. L'overflow del timer genera un'interruzione, la cui routine disabilita gli interrupt esterni e pone una variabile ad 1 per concludere il ciclo di attesa attiva (Pag.86 da riga 27 a 30). L'*ADC* converte in digitale prima la tensione al nodo V_{meas} poi la tensione al nodo T_{meas} e pone i risultati su due registri di appoggio (Pag.88 da riga 8 a 34). In base a quanti impulsi sono stati inviati dal *Level1* al microcontrollore, esso invia i dati richiesti (tensione e temperatura, numero di serie della cella, età della cella, Pag.86 da riga 75 a 98), infine torna in modalità *sleep* (Figura 3.15).

Il firmware è stato realizzato con l'obiettivo di far assorbire al microcontrollore la minor potenza possibile, a questo proposito l'*ADC*, che è il dispositivo interno che assorbe più corrente, viene acceso esclusivamente quando è necessario effettuare la conversione.

3.2.4 Misure

Dopo aver messo a punto una piattaforma di test per verificare la comunicazione del *Level0*, in cui *Level1* è simulato da un *ATmega32* posto in una scheda di sviluppo (*STK500*), sono state effettuate delle misure per constatare la bontà del funzionamento. Il *Level0* comunica i risultati delle misure al *Level1*, il quale, tramite un'interfaccia seriale, le invia al computer dove è

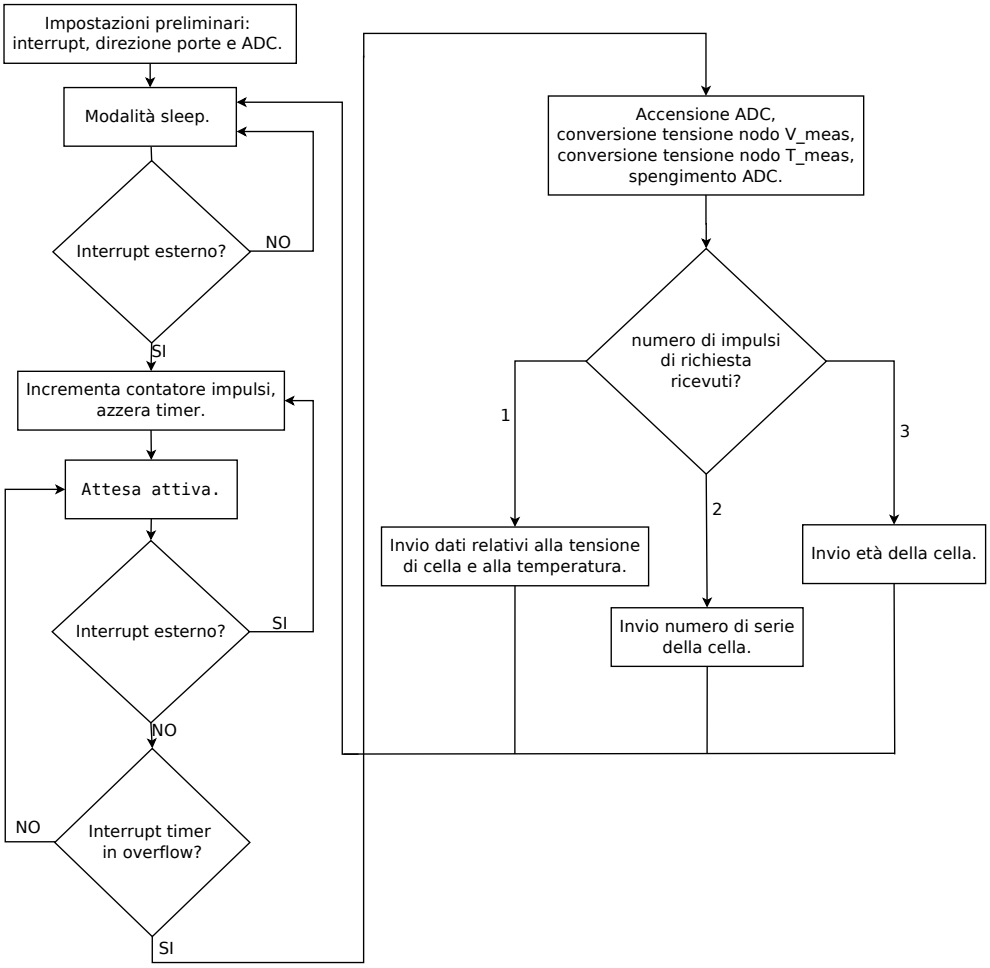

Figura 3.15: diagramma di flusso del firmware del microcontrollore

.

implementato un *VI* di *LabVIEW*, che ha il compito di elaborare i dati ed applicare i fattori di correzione alle misure.

Per risalire alla tensione e alla temperatura di cella, l'ADC acquisisce la tensione ai nodi T_{meas} e V_{meas}. Il datasheet dell'*ATtiny32A* mostra che la tensione di riferimento per la conversione dell'ADC ha un range di variabilità del 10%rispetto al valore nominale, quindi se $V_{R_{nom}} = 1.1V$ al variare del dispositivo V_R è compresa tra $0.99V$ e $1.21V$. La conoscenza della V_R è importante per convertire i campioni quantizzati che genera l'ADC nei valori di tensione ai nodi V_{meas} e T_{meas}, non essendo nota a priori è necessario estrapolare dai campioni dei fattori correttivi per diminuire l'errore introdotto. Per quanto riguarda la misura della tensione di cella, viene fatta variare l'alimentazione in modo lineare da $3V$ a $4.2V$, facendo campionare l'ADC del microcontrollore sul nodo V_{meas}. I campioni ottenuti vengono sottoposti ad un fittaggio lineare rispetto ai valori dell'alimentazione (Figura 3.16). Ricavando la pendenza e l'intercetta con l'asse delle ascisse della retta di

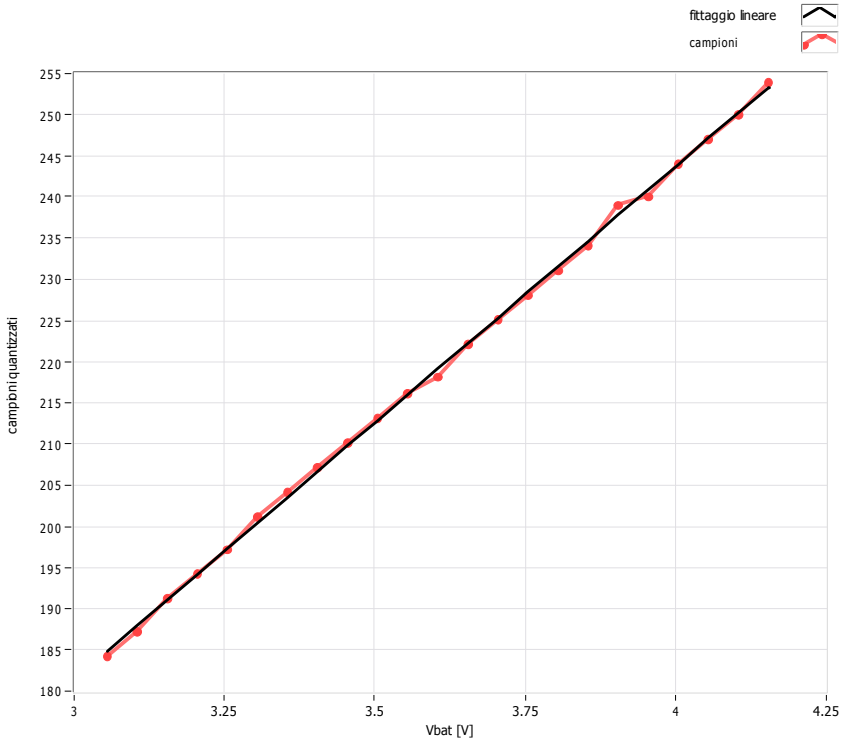

Figura 3.16: fitting lineare dei campioni dell'ADC

.

fitting si ottengono i parametri di correzione degli errori di off-set e di gua-

dagno (tra quest'ultimi c'è il contributo della variabilità della V_R), questi dovranno essere utilizzati in un'opportuna formula in fase di decodifica per risalire alla tensione misurata:

$$V_{meas_{cor}} = \frac{D - I}{S}$$

dove, D: campioni quantizzati forniti dall'ADC, I: intercetta della retta di fitting con l'asse delle ascisse, S: pendenza della retta di fitting.
Con la procedura sopra descritta è possibile correggere gli errori di off-set e di guadagno relativi sia alla variazione di V_{ref} sia alle non linearità del convertitore.

La procedura descritta sopra è stata effettuata per tutte le misure di tensione di cella che verranno descritte successivamente. Per quanto riguarda la temperatura non sono stati applicati alcuni parametri correttivi, dai campioni forniti dall'ADC è stata ricavata la tensione T_{meas} con la seguente formula:

$$V_{T_{meas}} = \frac{V_{R_{nom}} \cdot D}{255}$$

dove, $V_{R_{nom}} = 1.1V$, D: campioni quantizzati forniti dall'ADC.
La scelta di non applicare alcuna correzione alla misura di temperatura è motivata dal fatto che la variabilità della V_R introduce un errore che è stato ritenuto accettabile, inoltre, per ricavare i fattori correttivi, è necessario mettere a punto una procedura di test complessa che è stata rimandata a futuri sviluppi.

Le misure sono state effettuate alimentando il *Level0* tramite un misuratore di parametri (che simula la cella) e tenendo l'NTC a temperatura costante tramite un termostato ([16]).
Di seguito vengono elencate le tipologie di misure fatte:

- **Misura 1:** facendo variare linearmente l'alimentazione da $3V$ a $4.2V$ o $4.6V$ e termostatando l'NTC a 0°C, 20°C e 40°C, è stata acquisita la tensione al nodo V_{meas};
- **Misura 2:** tenendo fissata a $4.2V$ l'alimentazione, con un termostato, è stata fatta variare la temperatura dell'NTC da -5°C a 40°C;
- **Misura 3:** è stata misurata la corrente media, assorbita dal *Level0*, durante una fase di comunicazione con il *Level1*.

Misura 1:
Gli obiettivi della misura sono quelli di evidenziare, a temperatura dell'*NTC* fissata, l'andamento dei dati rilevati dal microcontrollore, quindi ricavare la precisione con cui misura la tensione di cella e la temperatura.

L'andamento della tensione T_{meas}, al variare della temperatura, è non lineare ed è direttamente proporzionale alla tensione di cella (equazione 3.1), quindi facendo variare l'alimentazione linearmente viene influenzata direttamente anche la misura della temperatura. I dati acquisiti dal microcontrollore, relativi alla temperatura, dovranno essere sottoposti ad un'elaborazione successiva da parte del *Level1*. Questa elaborazione in fase di test del *Level0* è stata effettuata da un *VI* di *LabVIEW*.

Il microcontrollore converte in digitale la tensione T_{meas}, per ricavare la temperatura è necessario invertire l'equazione 3.1, in cui, in prima analisi, si fissa nell'equazione $V_{bat} = 3.6V$.
La Figura 3.17 mostra l'andamento della temperatura, ottenuta invertendo l'equazione 3.1 (con $V_{bat} = 3.6V$) utilizzando i campioni di misura del microcontrollore, facendo variare l'alimentazione da $3V$ a $4.2V$.

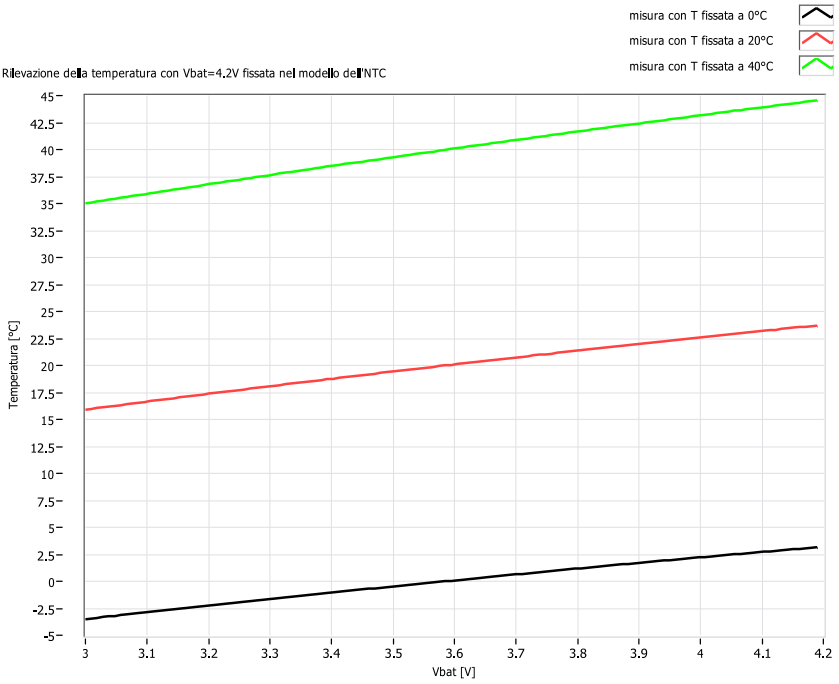

Figura 3.17: temperatura in funzione della tensione di alimentazione in variazione lineare, senza correzione di V_{bat}

.

L'errore sulla misura è nullo quando l'alimentazione è a $3.6V$ perché nel modello V_{bat} è fissato proprio a quel valore.
La variazione dell'errore aumenta all'aumentare della temperatura con cui viene termostata l'NTC, questo comportamento è dovuto all'andamento di T_{meas} (equazione 3.1), che ha una pendenza crescente all'aumentare della temperatura.

Per ridurre l'errore sulla rilevazione della temperatura è necessario invertire l'equazione 3.1 fissando il parametro V_{bat} al valore che assume l'alimentazione al momento della misura.

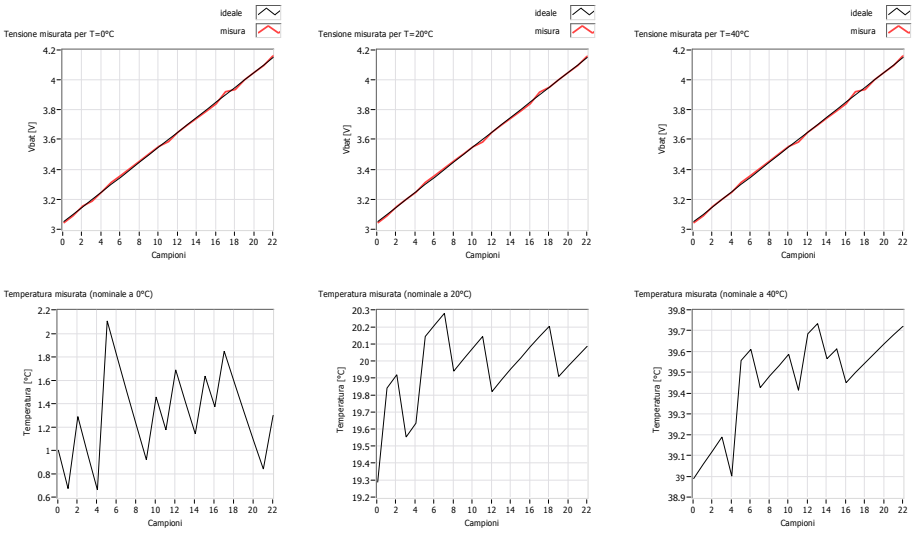

Figura 3.18: misure di tensione e temperatura con correzione di V_{bat} con l'NTC posta a *0°C*, *20°C* e *40°C*

.

In Figura 3.18 sono mostrati i risultati della misura di tensione e di temperatura, dove il parametro V_{bat} nell'equazione 3.1, assume il valore dell'alimentazione che varia linearmente da $3V$ a $4.2V$. In particolare, le misure sono state effettuate ponendo l'NTC a *0°C* (Figura 3.18(a)), *20°C* (Figura 3.18(b)) e *40°C* (Figura 3.18(c)). Nel grafico in alto, in rosso, è mostrato l'andamento della misura di tensione (effettuata dal microcontrollore) ed in nero è mostrato l'andamento della tensione di alimentazione imposta dall'esterno. Nel grafico in basso, è mostrato il risultato della misura di temperatura.

I risultati sperimentali evidenziano che la misura della tensione di cella, per le tre temperature, è affetta da un errore massimo di circa $1.5LSB$, un

risultato che può essere considerato soddisfacente. Per quanto riguarda la misura di temperatura, è possibile notare come i valori rilevati aumentino all'aumentare dell'alimentazione, questo effetto può essere imputato all'autoriscaldamento dell'NTC che è attraversata da una corrente crescente. L'errore sulla misura di temperatura decresce all'aumentare della temperatura con cui viene termostata l'NTC. Si ha questo effetto perché a temperature basse la tensione al nodo T_{meas} è piccola rispetto alla V_{ref} dell'ADC, quindi le non linearità del convertitore, la tolleranza sulla resistenza nel partitore e l'accuratezza con cui è nota la caratteristica dell'NTC vanno ad influenzare il risultato della misura.

Misura 2:
Tenendo fissata l'alimentazione del *Level0* a $4.2V$, è stata fatta variare la temperatura dell'NTC da *-5°C* a da *40°C*, inserendola in un termostato ([16]).

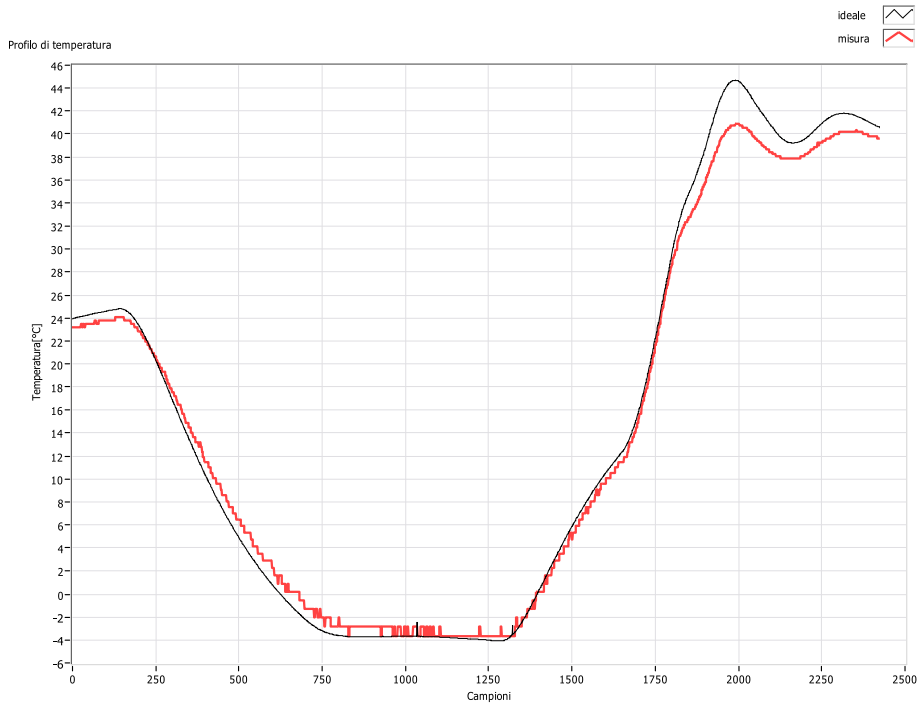

Figura 3.19: misura di temperatura con l'NTC posta in un termostato, ad alimentazione costante a $4.2V$

.

In Figura 3.19 è mostrato il risultato della misura, in nero è mostrato l'anda-

mento della temperatura impostata dal termostato ed in rosso l'andamento della rilevazione. L'errore massimo è di circa *1°C*.

Misura 3:
L'obiettivo di questa misura è quello di ricavare l'assorbimento medio di corrente del *Level0* durante una fase di comunicazione con il *Level1*.
Il *Level0*, alimentato a tensione costante di $4.2V$ e con l'NTC termostata a *20°C*, invia al *Level1* ogni 5 secondi la tensione di cella e la temperatura. Misurando la corrente media fornita dall'alimentazione è possibile ricavare l'assorbimento, che è di 110μA. Tale valore era facilmente prevedibile ricavando la corrente che viene dissipata sui due partitori: $R1$, $R8$ e $R2$, R_{NTC}.

$$I_{media_{min}} = V_{batt}(\frac{1}{R1 + R8} + \frac{1}{R2 + R_{NTC}}) = 101\mu A$$

con $R_{NTC}(20°C) = 4.7k\Omega$.
Un assorbimento medio di corrente di 110μA, è una quantità relativamente elevata considerando che la corrente di auto-scarica della cella è di circa 5μA.

3.2.5 *Level0* modificato

Come evidenziato in precedenza, più del 90% della corrente media assorbita dal *Level0* dalla cella, viene dissipata sulle resistenze che compongono i due partitori.

Per ridurre il consumo del *Level0* è stata apportata una modifica allo schema circuitale.
In Figura 3.19 è mostrato lo schema del *Level0* modificato, in cui è stata eliminata $R8$, $R2$ è stata collegata tra i nodi T_{meas} e V_{meas} e $R1$ ha assunto il valore di $15k\Omega$. Questa modifica si ripercuote sul firmware che deve subire una leggera modifica.
Quando non è necessario effettuare delle misure, i pin *PB3* e *PB4* vengono messi in alta impedenza e la rete composta da $R1$, $R2$ e l'NTC non assorbe corrente. Per misurare la tensione di cella, il pin *PB3* viene posto, via firmware, a livello logico basso (Pag.91 da riga 12 a 14) e viene misurata la tensione su V_{meas} (Pag.91 da riga 21 a 23). Per misurare la temperatura, viene posto allo zero logico il pin *PB4* (Pag.91 da riga 27 a 29) e viene misurata la tensione sul nodo T_{meas} (Pag.91 da riga 31 a 38).

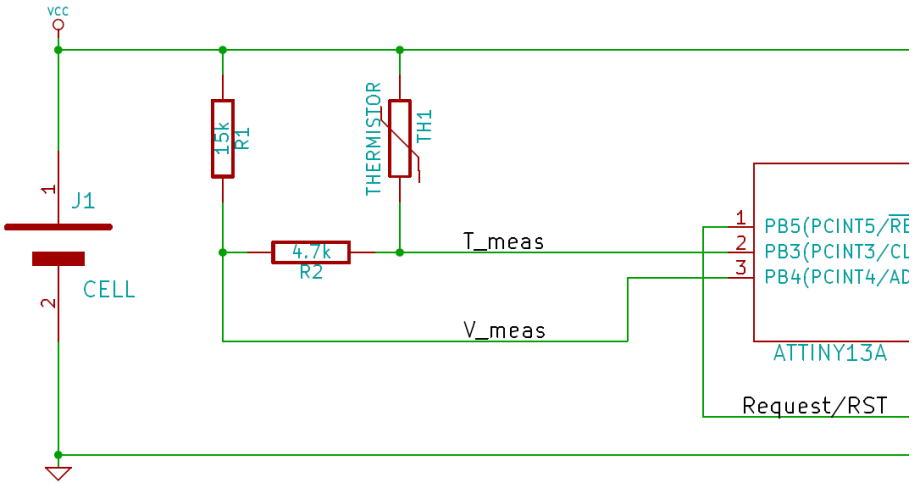

Figura 3.20: schema *Level0* modificato

.

Effettuando la solita misura descritta in precedenza per il *Level0* senza modifica, la corrente media assorbita dall'alimentazione risulta essere di 7μA.
Con questa modifica circuitale si ha una notevole riduzione di assorbimento di corrente, di contro le misure perdono di accuratezza.

Le misure eseguite sul *Level0* originale vengono ripetute anche sul *Level0* modificato, la Figura 3.21 mostra i risultati del primo tipo di misura (**Misura 1**).
La Figura 3.21 evidenzia una riduzione di accuratezza sia per la misura di tensione sia per la misura di temperatura, in particolare è possibile notare come l'errore sulla temperatura misurata aumenti al diminuire della temperatura nominale. Si ha una diminuzione dell'acuratezza perché il terminale negativo del partitore di misura non è più collegato stabilmente alla massa del circuito, ma è collegato ad un pin del microcontrollore, quindi è ad un potenziale V_{OL}, il cui valore varia in base alla corrente che l'attraversa ed alla tensione di alimentazione (Figura 3.22).
Le relazioni per determinare V_{meas} e T_{meas} sono le seguenti:

$$V_{meas} = V_{bat}\frac{R2}{R2+R1} + V_{OL}\frac{R1}{R1+R2}$$

$$T_{meas} = V_{bat}\frac{R2}{R2+R_{NTC}} + V_{OL}\frac{R_{NTC}}{R_{NTC}+R2}$$

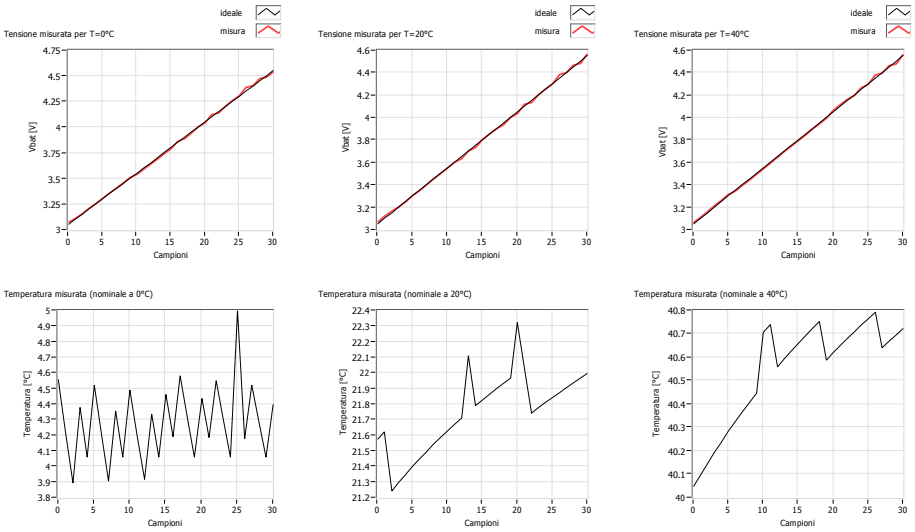

Figura 3.21: misure di tensione e temperatura con correzione di V_{bat} con l'*NTC* posta a *0°C*, *20°C* e *40°C*, *Level0* modificato

.

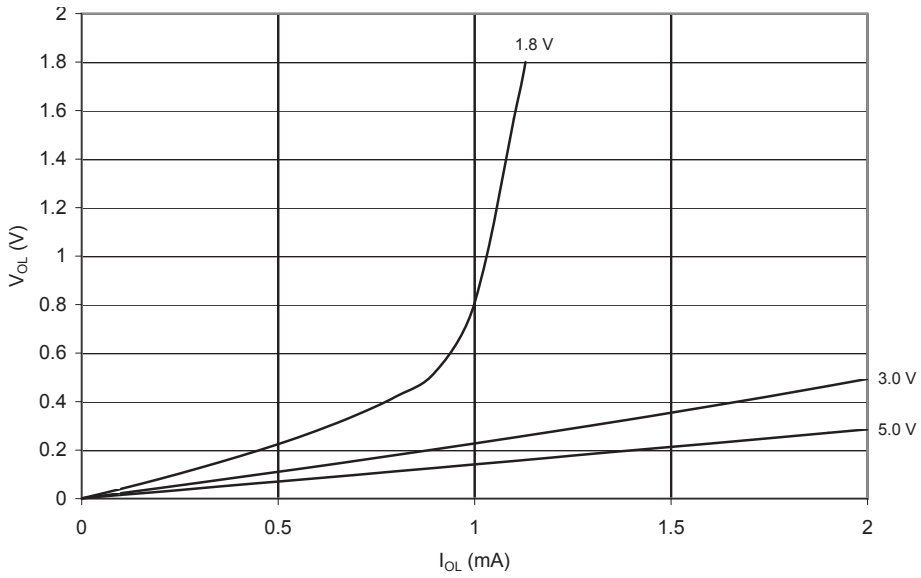

Figura 3.22: andamento della V_{OL} del microcontrollore al variare della I_{OL} e dell'alimentazione [14]

.

I termini che introducono l'errore sulla misura sono: $V_{OL}\frac{R1}{R1+R2}$ per la tensione, $V_{OL}\frac{R_{NTC}}{R_{NTC}+R2}$ per la temperatura.
Per basse temperature la tensione sul nodo T_{meas} è bassa (a *-5°C* è circa $0.1V$), quindi il termine di errore influenza maggiormente la misura rispetto alle alte temperature.
In fase di misura della tensione, all'aumentare dell'alimentazione, aumenta la corrente entrante nel pin del microcontrollore, quindi aumenta la V_{OL}. Questo effetto, come evidenziato in Figura 3.21, causa un errore sulla misura di tensione che cresce all'aumentare dell'alimentazione (Per V_{bat}=4V, $V_{OL} = 20mV$, $V_{OL}\frac{R1}{R1+R2} = 15mV$).
La Figura 3.23 mette in evidenza che per bassi valori di temperatura, la misura è affetta da un errore maggiore rispetto a quella effettuata con il *Level0* senza modifica (errore massimo di circa *4°C* con l'*NTC* termostata a *-4°C*).

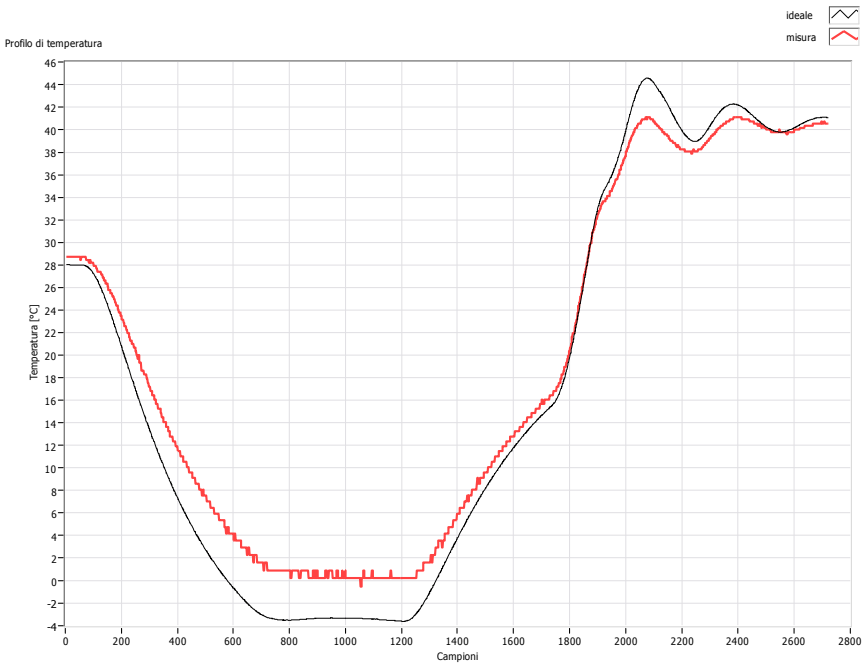

Figura 3.23: misura di temperatura con l'*NTC* posta in un termostato, ad alimentazione costante a $4.2V$, *Level0* modificato

.

In definitiva il *Level0* modificato assorbe dalla cella una corrente media molto più bassa rispetto al *Level0* originale (7μA contro 110μA), di contro

l'accuratezza delle misure è più bassa.

3.3 Level1

Il *Level1* è il sistema che gestisce la batteria, in particolare mette in atto le procedure di bilanciamento dettate dal *Level2*, effettua ulteriori misure rispetto al *Level0* sulla tensione e corrente di cella, seleziona la cella dalla batteria. Il *Level1* è un sistema autonomo, in grado di funzionare in assenza degli altri *Level1* ed in assenza di uno o più *Level0*. All'interno è posto un microcontrollore (*AT90CAN*) che ricevendo i comandi dal *Level2*, tramite l'interfaccia *CAN*, comanda la circuiteria ausiliaria per mettere in atto le funzionalità sopracitate.

La progettazione di ciascun blocco è stata fatta con l'obiettivo di far dissipare la minor potenza possibile ai circuiti, in modo tale da non scaricare troppo velocemente la cella ausiliaria. Quest'ultima, una volta scarica, può ricevere energia, tramite il processo di bilanciamento, sia dalla cella più carica nella batteria sia dal pacco batteria.

Lo schema elettrico del *Level1* è stato suddiviso in sette blocchi in base alle loro funzionalità (Figura 3.24):

- **Connettori Level0**;
- **Selezione celle**;
- **Microcontrollore**;
- **Alimentazione**;
- **Bilanciamento**;
- **Misura**;
- **Interfacce**.

Un vantaggio di alimentare il *Level1* da una cella esterna alla batteria è quello di avere una maggiore affidabilità, infatti se per qualche motivo la batteria si interrompe, il sistema di controllo è in grado comunque di prendere delle contromisure (per esempio, avverte il *Level2* che isolerà la parte della batteria danneggiata).

Prima di descrivere il funzionamento generale del *Level1*, verranno analizzati separatamente i blocchi che lo compongono.

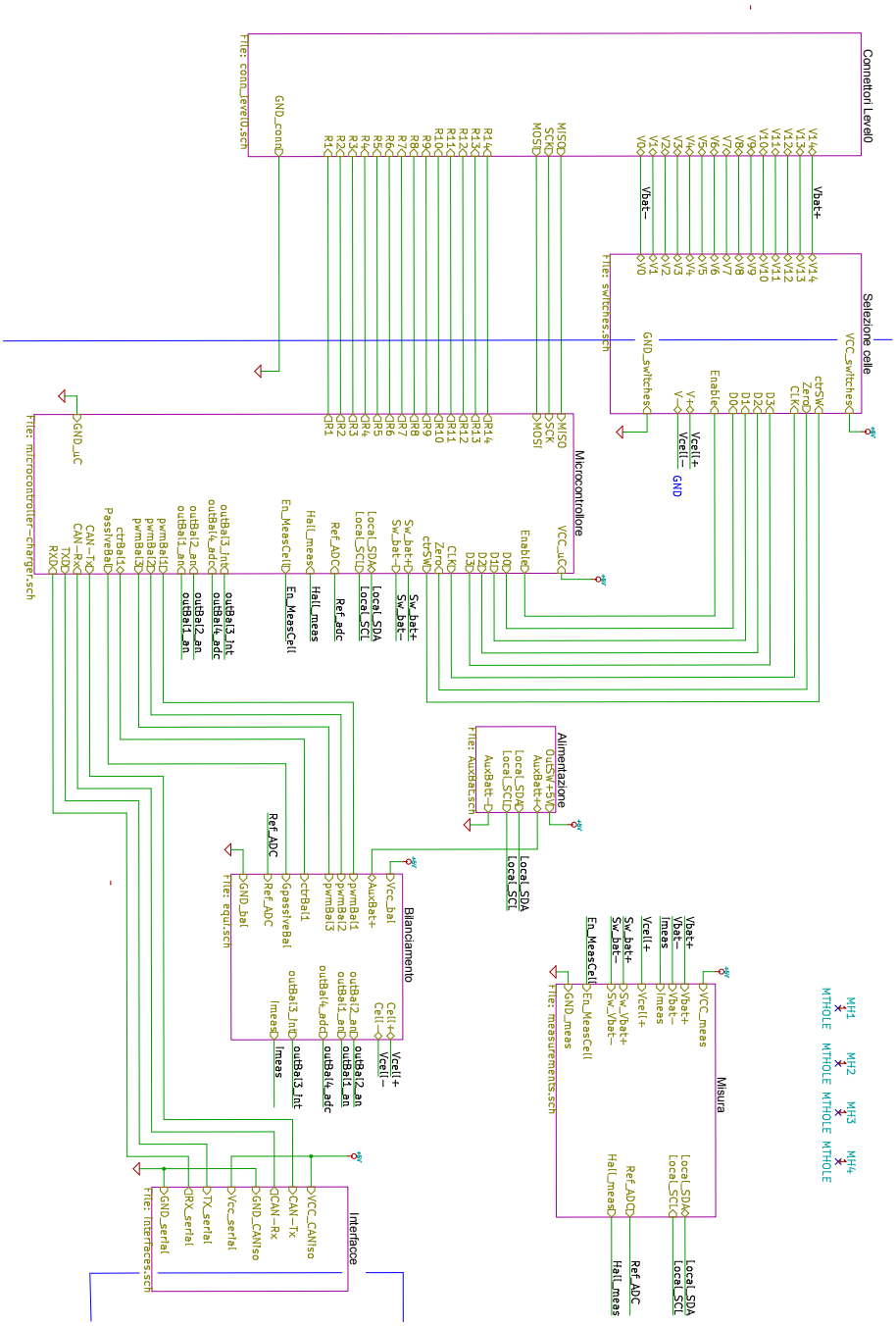

Figura 3.24: circuito del *Level1* suddiviso in blocchi

.

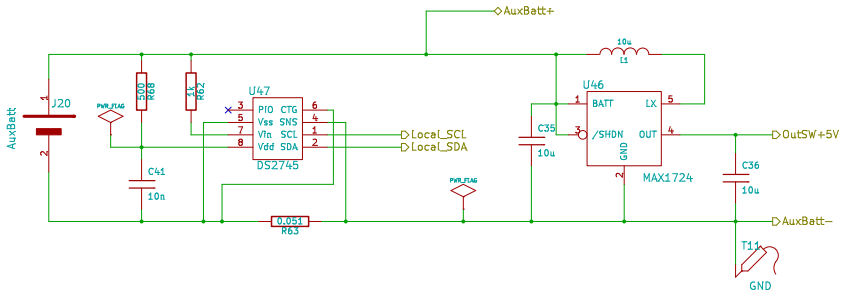

Figura 3.25: blocco *Alimentazione*

.

3.3.1 Blocco *Alimentazione*

L'alimentazione a tutti i circuiti del *Level1* è fornita da una cella esterna, del solito tipo di quelle che compongono la batteria, tramite un convertitore *DC-DC switching step-up*. Tale convertitore, partendo dalla tensione fornita dalla cella, deve garantire i $5V$ con la minor dissipazione di potenza possibile (rendimento energetico almeno del 85%) per avere un'elevata efficienza energetica del sistema. Il *DC-DC* utilizzato è il *MAX1724*, che fornisce al massimo $150mA$ ed ha un rendimento superiore all 85% (considerando l'assorbimento di corrente del *Level1* compreso tra $10mA$ e $100mA$) (Figura 3.25).

Per controllare lo stato della cella ausiliaria è stato utilizzato un *gauge* (DS2745) con interfaccia I^2C che trasmette i dati direttamente al microcontrollore. Un gauge è un tipo di circuito in grado di misurare la corrente, la tensione e la temperatura della cella a cui è collegato, per poi trasmettere i dati all'esterno tramite le usuali interfacce.

Il terminale positivo della cella è portato fuori dal blocco per effettuare il bilanciamento (di tipo *cella-cella*), mentre il terminale negativo, dopo la resistenza di *sense* del *gauge*, va alla massa del *Level1*.

3.3.2 Blocco *Connettori Level0*

Contiene i 14 connettori per collegare i porta-contatti, contenenti la cella ed il *Level0*, alla scheda del *Level1* (Figura 3.26).
Di seguito viene descritta la funzionalità di ciascun pin del connettore:

- Pin 1 e 2: vengono collegati al terminale positivo della cella e portati al blocco *Connessione celle*;

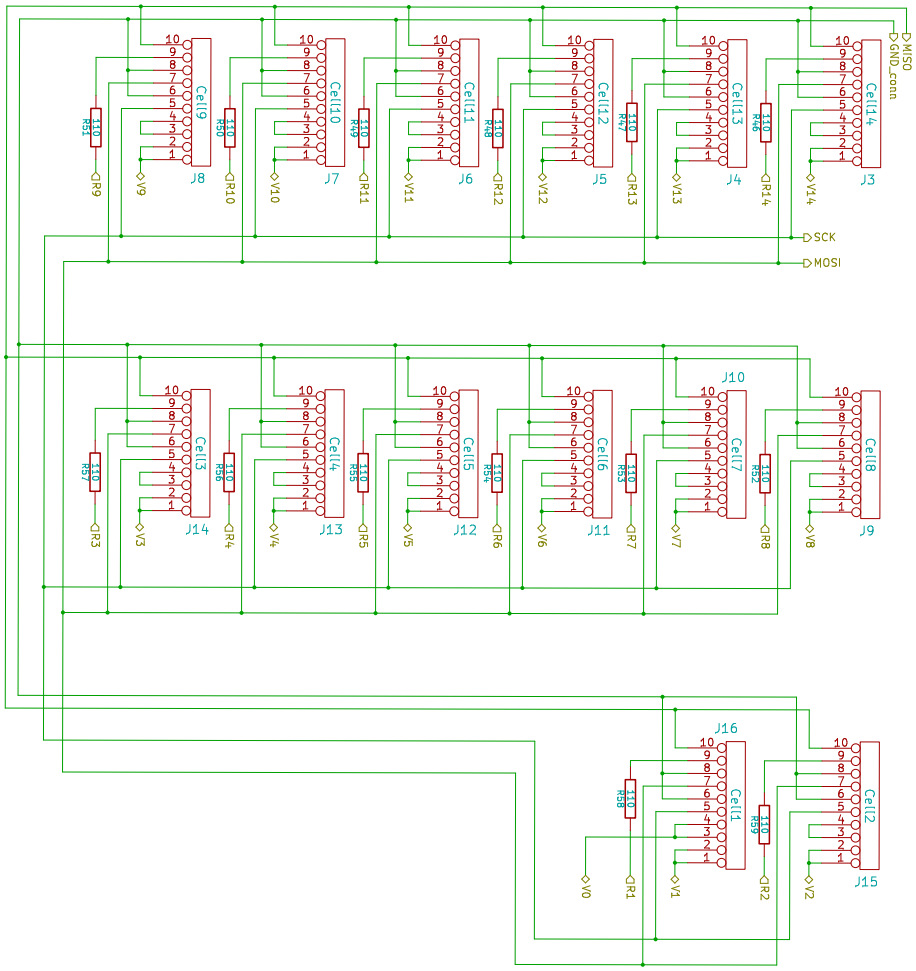

Figura 3.26: blocco connettori *Level0*

.

- Pin 3 e 4: vengono collegati al terminale negativo della cella. Solo per *Cell1* questi due pin sono portati fuori e collegati al blocco *Connessione celle*;
- Pin 5: trasmette il segnale di clock per la comunicazione *SPI* tra il *Level0* ed il *Level1*, è collegato al blocco *Microcontrollore*;
- Pin 6 e 8: sono gli emettitori dei foto-accoppiatori per la trasmissione del *Level0*, vengono portati fuori dal blocco e collegati alla massa del *Level1*;
- Pin 7: trasmette i dati inviati dal *Level0* al *Level1*, è collegato al blocco *Microcontrollore*;
- Pin 9: controlla l'anodo del foto-accoppiatore per inviare i dati (per esempio la richiesta di invio temperatura e tensione di cella) al *Level0*, è collegato al blocco *Microcontrollore*;
- Pin 10: è collegato al catodo del foto-accoppiatore e portato fuori verso il blocco *Microcontrollore*. Consente la selezione dei *Level0* in fase di invio dati dal *Level1*.

3.3.3 Blocco *Selezione celle*

All'interno sono contenuti i circuiti per la selezione delle celle e per l'inversione della polarità delle stesse. Tutti i relays allo stato solido (*PVG612A*) sono pilotati dal *PLD* (*ATF150AS*), in particolare quelli denominati da *U1* a *U15* selezionano la cella, quelli da *U18* a *U20* fungono da invertitore di tensione controllato (invertono o meno la tensione di cella). Il *PLD* deve pilotare i relays in modo tale che in nessun caso e in nessuna condizione vengano messe in corto circuito due o più celle (Figura 3.27).
Il funzionamento del circuito logico implementato nel *PLD*, mostrato in Figura 3.28, è il seguente (Figura 3.29 per i segnali):

- il microcontrollore (nel blocco omonimo) indica al *PLD* la cella da selezionare tramite i segnali *D0*,*D1*,*D2* e *D3*, inoltre tramite il segnale *ctrSW* indica se invertire o meno la polarità della tensione. Questi cinque segnali vanno in ingresso a *LATCH-D 1* e *LATCH-D 2*;
- il microcontrollore pone alto il segnale *Enable*, i dati sugli ingressi *IN* e *ctrSW* vengono trasferiti sulle linee *X1* e *X2*. Il segnale *edge* va alto;

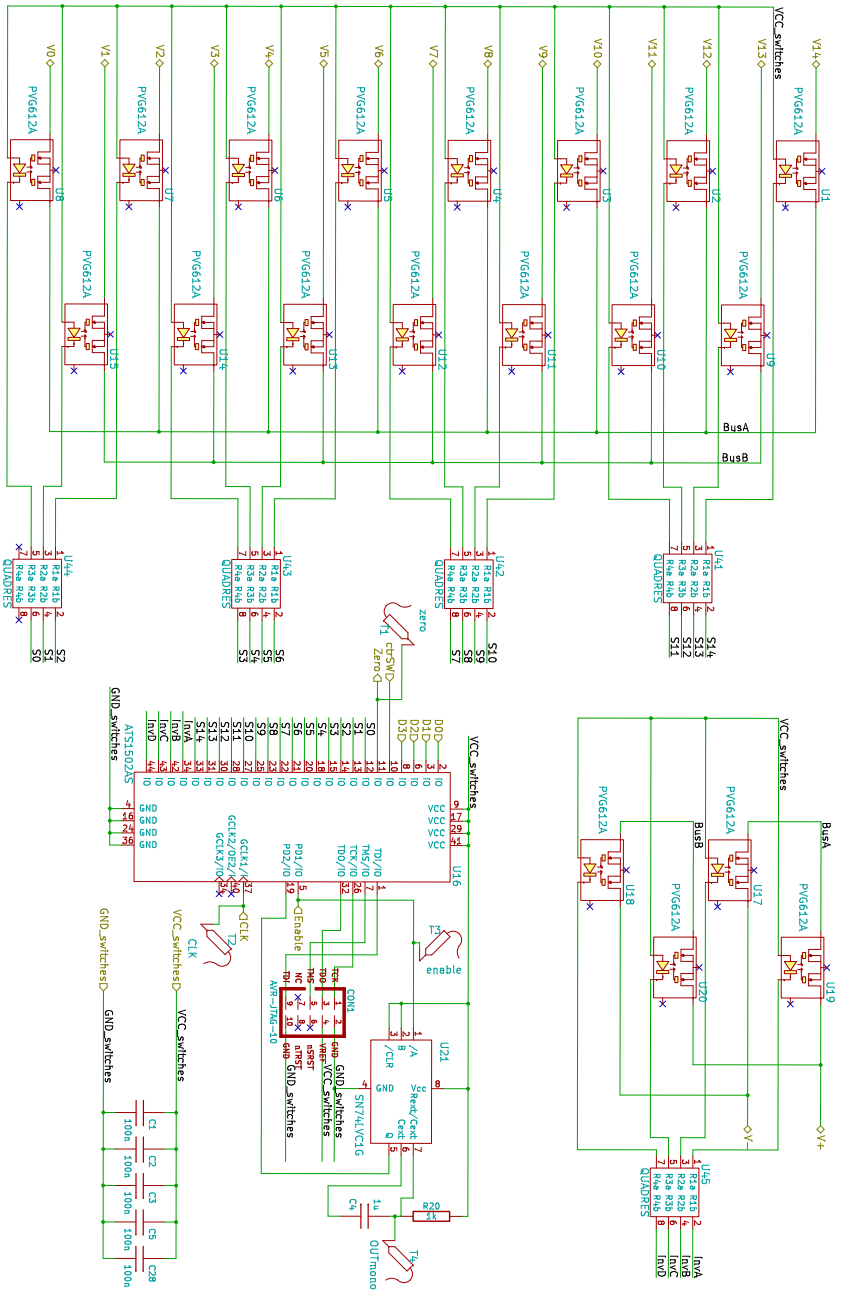

Figura 3.27: blocco *Selezione celle*

.

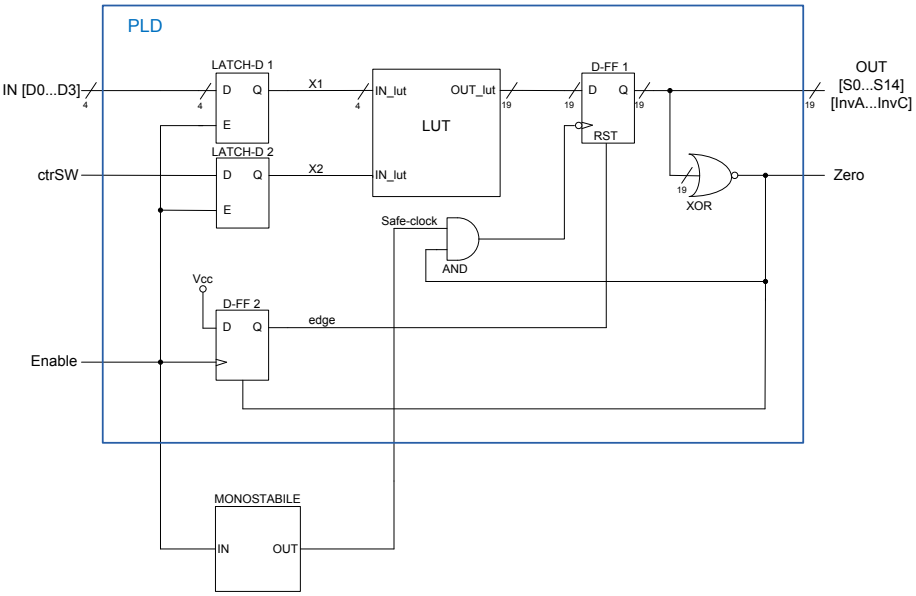

Figura 3.28: circuito logico implementato nel *PLD*

.

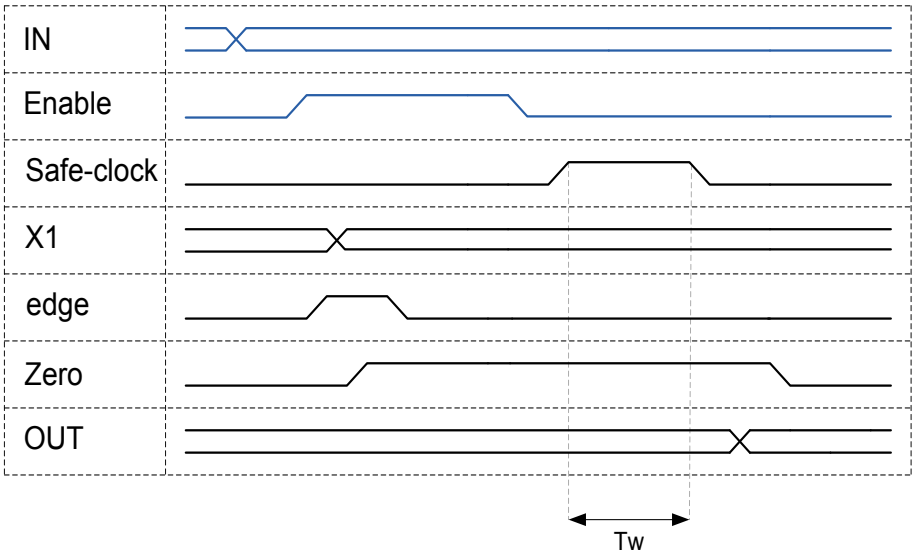

Figura 3.29: segnali *PLD*

.

- il segnale alto *edge* resetta il flip-flop *D-FF 1*, il segnale *zero* va alto e va a resettare il flip-flop *D-FF 2*. A tal punto *D-FF 1* è in grado di far evolvere l'uscita appena riceve un fronte alto-basso del clock;
- il microcontrollore pone basso *Enable*, questo fa partire il monostabile che manda a livello alto la sua uscita (*Safe-clock*) per un tempo Tw;
- appena *Safe-clock* va basso, *D-FF 1* pone in uscita i 19 segnali per pilotare i relays allo stato solido, infine *Zero* va a livello basso;
- i 19 segnali in uscita vengono mantenuti fintantochè il microcontrollore pone nuovamente alto *enable*, a tal punto il circuito pone a zero tutte le uscite (relays aperti) ed evolve come descritto nei punti precedenti.

Il *PLD* spegne tutti i relays prima di accenderli nuovamente con una nuova configurazione.

Il monostabile consente di avere un intervallo temporale di guardia (Tw), dimensionato in base al tempo di dissaturazione dei *PVG612A*, il quale garantisce che non venga attivato un relay mentre un altro si sta spegnendo.
Dimensionamento di Tw:
Dal datasheet del *PVG612A* si ricava che per una corrente di pilotaggio del led, $I_{LED} = 10mA$, si ha un tempo di spegnimento $t_{OFF} = 0.2mS$ valutato per una corrente di attraversamento di $0.5A$. Considerando l'importanza di Tw a livello di affidabilità della batteria e che i relays, nel nostro caso, sono attraversati da circa $1A$ di corrente massima, l'intervallo di guardia viene scelto di $1mS$. I componenti esterni del monostabile (*SN74LVC1G123*) vengono dimensionati per avere: $Tw = 1mS$.

Per quanto riguarda l'accensione dei relays, il microcontrollore deve garantire, tra il fronte negativo ed il successivo fronte positivo di *enable*, un tempo adeguato per completarne la saturazione.
Il segnale *Zero* indica al microcontrollore quando i relay sono tutti scollegati (uscite del *PLD* a zero), questa informazione può essere utile per un controllo del funzionamento del *PLD*.

Tramite i connettori, nel blocco *ConnettoriLevel0*, giungono al blocco *Selezione celle* i potenziali delle celle (etichettati con *V0,...,V14*), tra cui soltanto due vengono selezionati, ed eventualmente invertiti, dal *PLD* comandato dal microcontrollore. I due potenziali in questione (etichettati con *V+* e *V-*) escono fuori dal blocco ed uno di essi, denominato *V-*, viene col-

legato alla massa del *Level1*. Il potenziale di massa del *Level1*, rispetto al terminale negativo della batteria, varia in base alla cella selezionata.

3.3.4 Blocco *Microcontrollore*

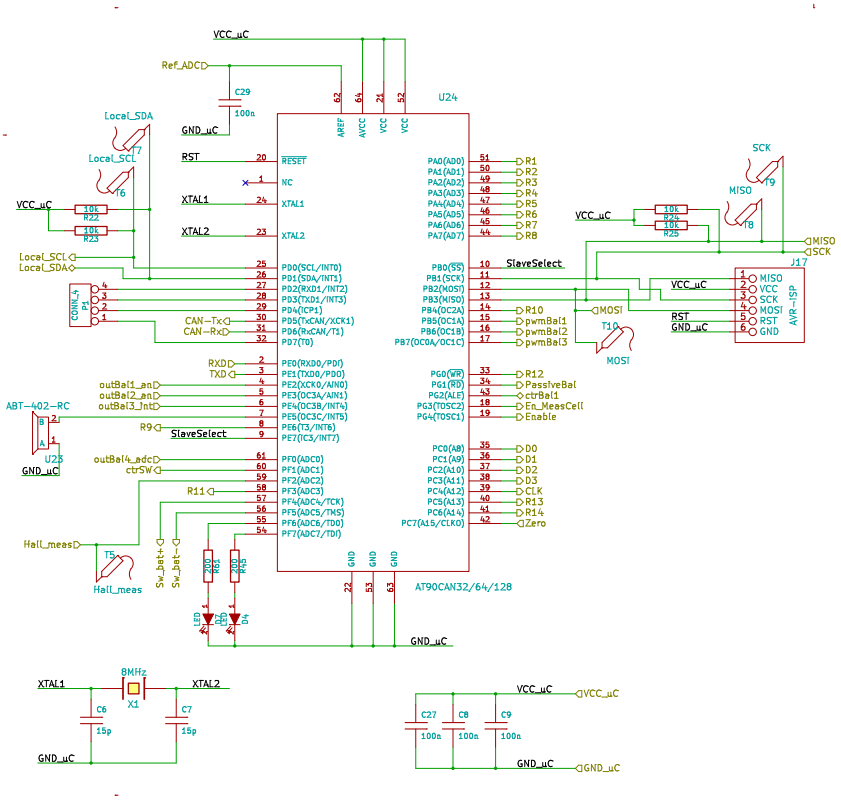

Figura 3.30: blocco selezione celle

.

Questo blocco contiene il microcontrollore (*AT90CAN*), che controlla e gestisce le varie operazioni del *Level1* (Figura 3.30).
Di seguito vengono elencate le funzionalità del microcontrollore che, rispetto alla comunicazione *SPI* con i *Level0*, è impostato come *slave*:

- interroga i *Level0* tramite i 14 segnali denominati *R1*,...,*R14* e riceve le risposte sulla linea *MOSI* sincronizzate dal segnale *SCK*;
- con il segnale *MISO* ed i vari *R1*,...,*R14*, il microcontrollore è in grado di multiplexare l'informazione da inviare al *Level0*. Per esempio, se

il microcontrollore intende comunicare al *Level0 1* un informazione, pone alta la linea *R1* e invia il dato tramite *MISO*;

- invia al *PLD* il clock ed i segnali per la selezione della cella (*Enable*, *D0*,..., *D3*, *ctrSW*);
- dalle linee I^2C *Local_SDA* e *Local_SCL* riceve i risultati delle misure effettuate dai componenti nel blocco *Misura* (temperatura di batteria, corrente di batteria, tensione della cella e tensione della batteria). Tramite i segnali *SW_bat+ SW_bat-* pilota il circuito di misura della tensione di batteria nel blocco *Misura*;
- riceve i segnali che devono essere convertiti in digitale con l'*ADC* interno (per esempio *Hall_meas* e *outBal4_adc*), il cui riferimento a $4.5V$, giunge dal blocco *Misura*;
- con i segnali che entrano nel blocco *Bilanciamento* gestisce una scheda, collegata al *Level1* con dei connettori, contenente un circuito di bilanciamento. Tramite il segnale *PassiveBal* controlla il *gate* di un *MOS* per il bilanciamento passivo;
- tramite le linee *TX-CAN* e *RX-CAN* gestisce la comunicazione *CAN-bus* con un dispositivo esterno, mentre con le linee seriali *TXD* e *RXD* comunica con una periferica via *USB*.

Al microcontrollore sono stati collegati dei dispositivi per il debug del firmware, due led e un buzer.

3.3.5 Blocco *Bilanciamento*

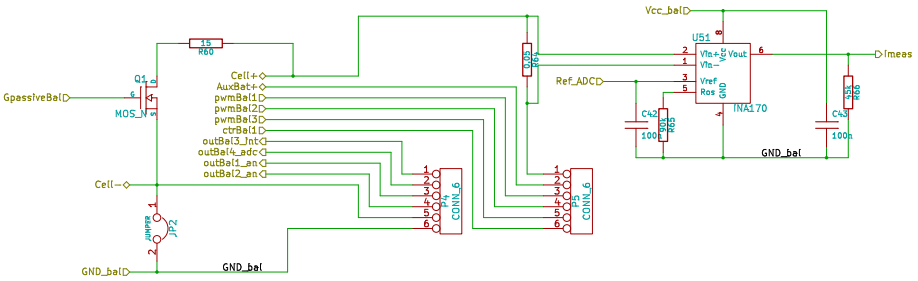

Figura 3.31: blocco *Bilanciamento*

.

Nel blocco in questione sono contenuti i circuiti e le strutture per effettuare il bilanciamento tra la cella selezionata dalla batteria e la cella ausiliaria (Figura 3.31). La scheda esterna al *Level1*, contenente il circuito di bilanciamento, viene connessa tramite i connettori *P4* e *P5*, a cui giungono i terminali positivi delle due celle, la massa del *Level1* e i segnali di controllo provenienti dal microcontrollore.

Per misurare la corrente fornita dalla cella selezionata dalla batteria durante il bilanciamento, è stato utilizzato un misuratore di corrente bidirezionale, l'*INA170*. L'uscita in tensione del misuratore, che è collegata ad un *ADC* nel blocco *Misura*, è espressa dalla seguente formula:

$$V_{I_{meas}} = V_{Ref\_ADC}\frac{R66}{R65} \pm I_{cell}\frac{R64 \cdot R66}{1k\Omega}$$

Il circuito è stato dimensionato per avere $V_{I_{meas}} = V_{Ref\_ADC} = 4.5V$ per $I_{cell} = 1A$ e $V_{I_{meas}} = 0$ per $I_{cell} = -1A$. La resistenza di *sense* (*R64*) è stata dimensionata per avere ai suoi capi al massimo $50mV$ di caduta di tensione, valore che rientra nel range consigliato dal produttore per avere un buon compromesso tra precisione della misura e caduta di tensione.

Per effettuare il bilanciamento passivo sono stati utilizzati un *MOS*(*PH3418*, con $I_{Dmax} = 1.4A$, $P_{Dmax} = 0.9W$), con il *gate* pilotato dal microcontrollore, ed una resistenza (*R60*, 15Ω da $2W$). Dall'intercetta tra la retta di carico ($V_{cell} = I_D \cdot R60 + V_{DS}$) e la caratteristica del *MOS* per $V_{GS} = 5V$ (nel datasheet) si ottiene una $V_{DS} = 0.7$ circa. Al variare della tensione di cella varia la pendenza della retta di carico, comunque considerando una $V_{DS} = 0.6V$ con $2.7 < V_{cell} < 4.2$, l'errore commesso per ricavare la corrente di bilanciamento è relativamente basso.

$$I_D = \frac{V_{cell} - 0.6V}{R60}$$

Per $V_{cell} = 4.2V$ si ha $I_D = 233mA$.
Per far funzionare correttamente il circuito di bilanciamento passivo è necessario che il *source* del *MOS* sia collegato a massa, quindi il terminale negativo della cella selezionata deve essere collegato alla massa del *Level1* tramite il *jumper JP2*.

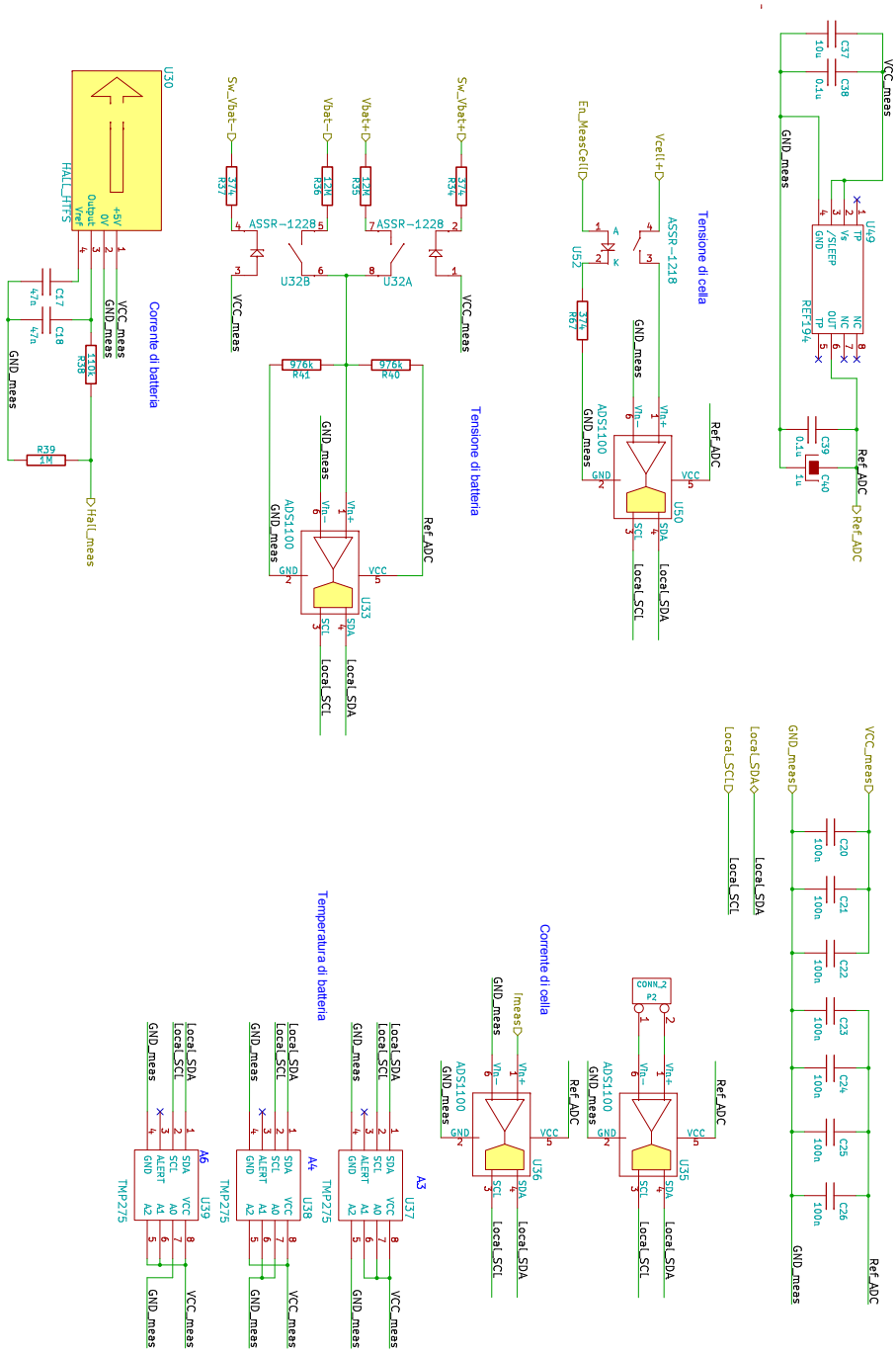

Figura 3.32: blocco *Misura*

.

3.3.6 Blocco *Misura*

Sono contenuti i circuiti per la conversione analogico-digitale, misura della tensione di cella e di batteria, misura della corrente e della temperatura di batteria. Tutti i dispositivi, tranne il sensore *Hall*, inviano i risultati delle misure al microcontrollore, in digitale, tramite l'interfaccia I^2C (Figura 3.32).

Il *REF194* genera un riferimento di tensione per gli *ADC* (*ADS1100*), che sono di tipo pseudo-differenziali (il modo comune della tensione d'ingresso $V_{in} = V_{in+} - V_{in-}$ deve essere compreso tra V_{cc} e $GND - 0.3V$).
La tensione tra V_{cell+} e massa può essere negativa (la polarità della tensione di cella può essere invertita dai foto-accoppiatori nel blocco di *Selezione cella*), quindi è necessario inserire un interruttore (*U52*) che scolleghi l'ingressi dell'*ADC* (*U33*), perché quest'ultimo non può accettare su V_{in+} o su V_{in-} una tensione inferiore a $GND - 0.3V$.

La differenza di potenziale tra la massa del *Level1* (nel blocco in questione è *GND_meas*) ed il polo positivo o negativo della batteria, varia in base a quale cella viene selezionata (il terminale negativo della cella selezionata viene connesso alla massa del *Level1*). Il circuito composto da *U32*, *U33* e le resistenze annesse, consente di ottenere la tensione di batteria per qualunque cella selezionata. Di seguito viene descritto il funzionamento ipotizzando che sia stata selezionata una qualunque cella nella batteria:

- il microcontrollore, tramite il segnale *Sw_bat+*, accende il foto-accoppiatore *U32A*. La tensione del terminale positivo della batteria (*Vbat+*), viene condizionata dalla rete resistiva formata da *R35*, *R40* e *41*, infine giunge all'ingresso V_{in+} dell'*ADC*. La tensione tra *Vbat+* e la massa dell'*ADC* varia tra $60V$ e $0V$ in base a quale cella viene selezionata. La rete resistiva non fa altro che portare tale dinamica tra $4.5V$ e $2.25V$. In definitiva in questa fase viene misurata la tensione tra il polo positivo della batteria e la massa del *Level1* (la quale è ad un potenziale intermedio).

- il microcontrollore, tramite il segnale *Sw_bat-*, accende il foto-accoppiatore *U32B*. Alla rete passiva di condizionamento giunge la tensione *Vbat-*, che rispetto alla massa dell'*ADC*, è compresa tra $-60V$ e $0V$, quindi all'ingresso V_{in+} giunge una tensione compresa tra $0V$ e $2.25V$. In que-

sta fase viene misurata la tensione tra il polo negativo della batteria e la massa del *Level1*.

I risultati ricavati dalle misure effettuate nelle due fasi appena descritte, vengono inviate al microcontrollore che risalirà alla tensione di batteria.

Per misurare la corrente nella batteria è stato utilizzato un sensore ad effetto *Hall* con un range di misura di $\pm 300A$ e con uscita $V_{out} = V_{ref} \pm 1.24\frac{I_{batt}}{I_{PN}}$ con $V_{ref} = \frac{V_{cc}}{2} = 2.5V$ e con $I_{PN} = 200A$. La tensione V_{out}, dopo un condizionamento con un partitore resistivo per portarla in range compreso tra $0V$ e $4.5V$, giunge al microcontrollore per la conversione in digitale.

L'*ADC U36* converte in digitale la tensione che giunge dall'*INA170*, proporzionale alla corrente della cella in fase di bilanciamento.

Per misurare la temperatura della batteria vengono utilizzati tre sensori digitali, *TMP275*, che tramite l'interfaccia I^2C comunicano con il microcontrollore.

3.3.7 Blocco *Interfaccia*

In questo blocco ci sono i dispositivi per far comunicare il *Level1* con l'esterno. Un transceiver *CAN* isolato (*ISO1050*) per comunicare con il *Level2* ed un interfaccia *seriale-USB* (*FTDI232*), isolata otticamente dall'*ADUM1201*, per il trasferimento dati con un *PC* (Figura 3.33).

Nel caso in cui la batteria venga collegata ad un caricatore esterno, il polo negativo della batteria è ad un potenziale imposto dalla rete elettrica. In base a quale cella viene selezionata, la differenza di potenziale tra la massa del *Level1* ed il polo negativo della batteria può essere di circa $60V$. Il *PC*, utilizzato per trasferire i dati dal *Level1* via *USB*, avrà la massa imposta dalla rete elettrica, quindi la differenza di potenziale tra la massa della trasmissione *USB* e la massa del *Level1* è prossima a $60V$. Per evitare che si danneggino i due dispositivi, collegando direttamente la porta *USB* al *Level1*, è necessario isolare otticamente la trasmissione.

3.3.8 Principio di funzionamento

Di seguito viene descritto il funzionamento generale del *Level1*:

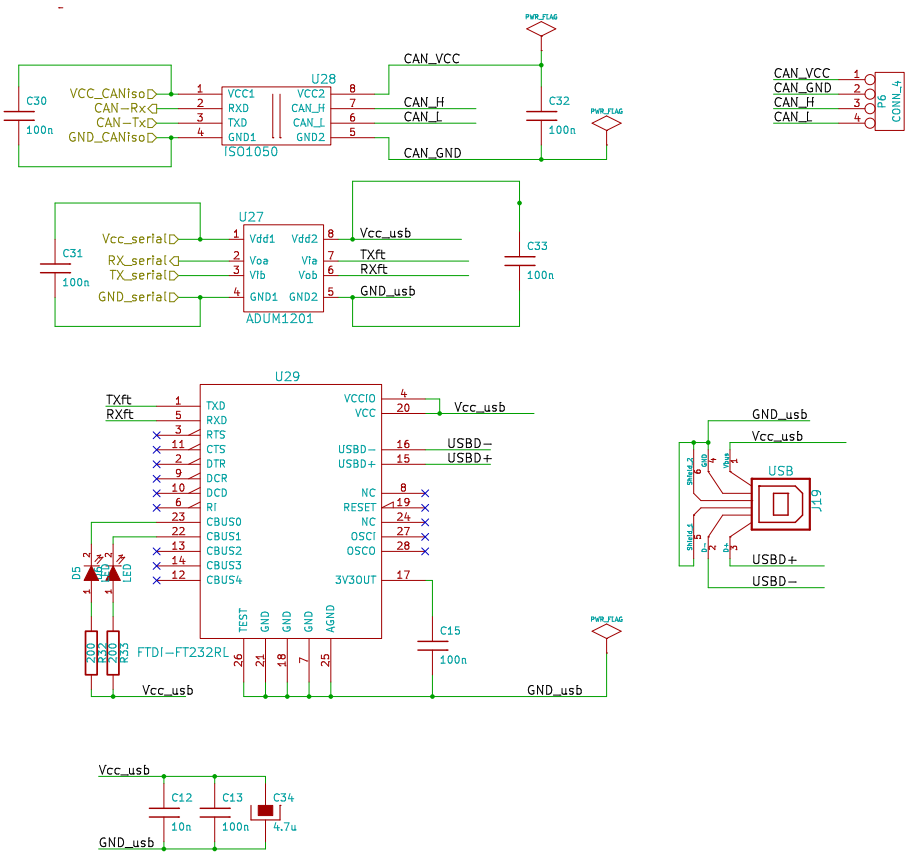

Figura 3.33: blocco *Interfaccia*

.

- il microcontrollore, tramite le linee di richiesta (*R1,...,R14*), interroga i vari *Level0* per conoscere la tensione e la temperatura delle celle e riceve i dati sulla linea *MOSI* sincronizzati dal segnale *SCK*;

- il microcontrollore monitora lo stato di ciascuna cella e della batteria misurando la corrente che l'attraversa e la tensione complessiva, ed invia i dati, tramite *CAN-bus*, al *Level2*. Inoltre i dati raccolti possono essere inviati ad un *PC* via *USB* per un'ulteriore elaborazione;

- in fase di bilanciamento, il microcontrollore comanda al *PLD*, tramite le linee *D0,...,D3*, quale cella selezionare e se invertirne o meno la polarità della tensione. I terminali della cella selezionata e quelli della cella ausiliaria giungono al *plug-in* per il bilanciamento insieme ai segnali di controllo provenienti dal microcontrollore. A tal punto può partire la fase di equalizzazione tra le due celle, per quella nella batteria viene misurata la corrente (*INA170*) e la tensione (*ADS110 U50*), per quella ausiliaria le misure vengono effettuate dal gauge (*DS2745*). Il microcontrollore gestisce la fase di bilanciamento in base ai risultati che gli giungono dai dispositivi di misura collegati alle due celle.

3.3.9 Realizzazione

Per realizzare il *PCB* del *Level1* sono stati utilizzati 4 layer, *front* (Figura 3.34), *back* (Figura 3.37), *Vcc_Req* (Figura 3.35) e *GND_Sw* (Figura 3.36).

- **lato *front*:** vengono saldati i connettori per collegare il porta-contatti contenete il *Level0* e la cella. A sinistra, al centro ed a destra della scheda sono posizionati tre sensori di temperatura (*TMP275*), infine sono posizionati alcuni dei *PVG612A* utilizzati. Su questo lato sono stati posizionati pochi componenti perché vengono inseriti i porta-contatti, la cui struttura copre la maggior parte della superficie rendendo scomodo contattare i terminali dei dispositivi in fase di debug;

- **lato *back*:** su questo lato sono posizionati la maggior parte dei componenti ed i vari connettori (per collegare il *plug-in*, per la programmazione del microcontrollore e del *PLD*,...);

- **layer *Vcc_Req*:** sono presenti principalmente le piste (*R1*,...,*R14*) per la richiesta dati al *Level0* ed un piano di alimentazione a forma di "C";

- **layer *GND_Sw*:** sono presenti le piste di controllo dei foto-accoppiatori (*PVG612A*) per la selezione delle celle e l'inversione della tensione delle stesse, inoltre c'è un piano di massa della solita forma del piano di alimentazione.

Per il *Level1* è stata scelta una scheda a quattro *Layer* per facilitare il routing dato l'elevato numero di componenti, tra cui un *PLD* ed un microcontrollore che introducono un'elevata complessità per disegnare le piste.

La procedura per il dimensionamento delle piste e delle via è la solita utilizzata per realizzare la scheda del *Level0*, in particolare per la larghezza delle piste sono stati utilizzati i seguenti valori:

- Piste di segnale: $W = 0.3mm$;

- Piste di potenza: $W = 0.55mm$.

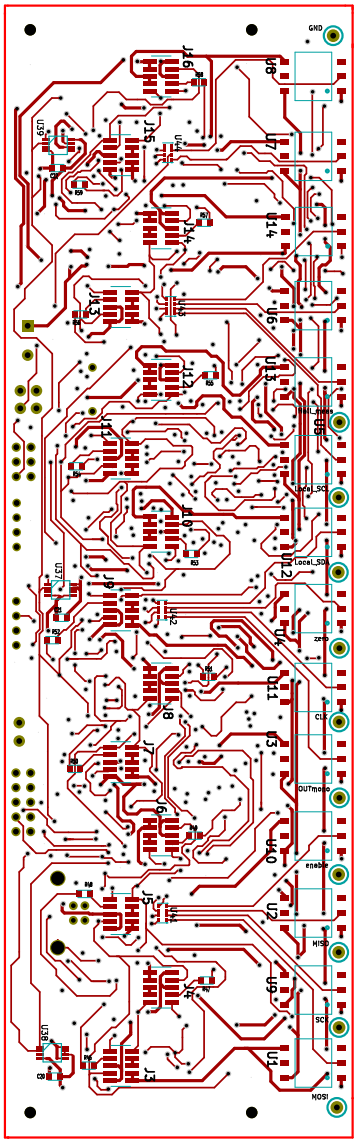

Figura 3.34: Layout *Level1* lato *front*

.

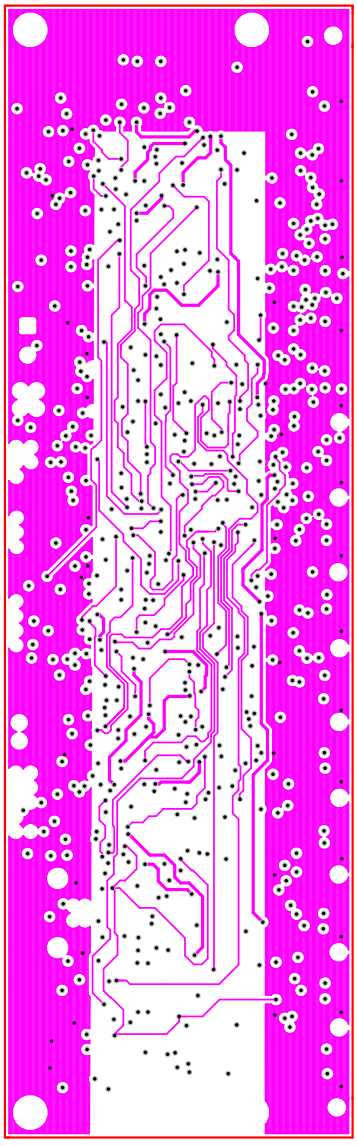

Figura 3.35: Layout *Level1*, layer interno di alimentazione

.

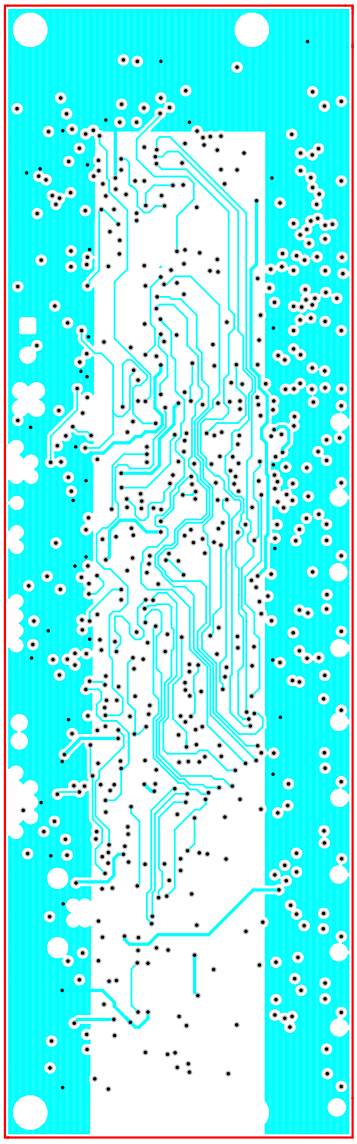

Figura 3.36: Layout *Level1*, layer interno di massa

.

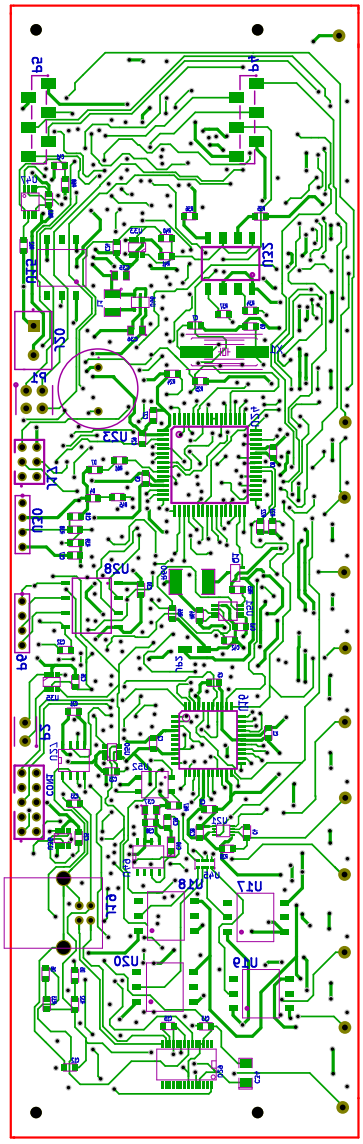

Figura 3.37: Layout *Level1* lato *back*

.

Capitolo 4

Sviluppi futuri

Dopo aver montato il *Level1* ed i porta-contatti in un intelaiatura, è necessario sviluppare un firmware per testare il funzionamento di tutti i *Level0* e di tutti i dispositivi montati. Tale firmware dovrà verificare le seguenti funzionalità:

- trasferimento dati tra i vari *Level0* ed il *Level1*;
- selezione delle celle senza il rischio di porle in corto circuito;
- comunicazione seriale tra il *Level1* ed il *PC*;
- bilanciamento tra la cella della batteria e quella ausiliaria.

Per ottenere con precisione la temperatura della cella, via firmware, dovrà essere implementata una LUT contenente i campioni della caratteristica dell'*NTC*, inoltre, per diminuire gli errori di off-set e di guadagno sulla conversione analogico-digitale sarà necessario mettere a punto una procedura che consenta di estrarre i parametri correttivi da un fitting lineare dei campioni. I parametri correttivi diversi per ciascun *Level0* e relativi sia alla misura di tensione sia alla misura di temperatura, saranno memorizzati nella *EEPROM* del microcontrollore *ATtiny32A*, in questo modo il *Level0* invierà al *Level1* direttamente i dati corretti.

Attualmente non è stata sviluppata una procedura software che consenta al *Level1* di scrivere direttamente nella *EEPROM* del microcontrollore del *Level0*. Questa funzionalità, una volta realizzata, consentirà di memorizzare tutti i parametri caratteristici della cella e del *Level0* e le varie informazioni di tracciabilità come l'età della cella.

Conclusa la fase di test sulla struttura completa della batteria (le 14 celle con i rispettivi *Level0* ed il *Level1*), dovrà essere sviluppato un sistema di gestione delle tre batterie, il *Level2* che gestirà tutto il pacco. Nel *Level2* verrà implementato il modello della batteria e ricevendo i dati dai tre *Level1* eseguirà tutte le operazioni atte a migliorare la prestazioni del pacco batteria (bilanciamento tra i segmenti del pacco, esclusione di un segmento in caso di malfunzionamento,...).

Capitolo 5

Conclusioni

Un *BMS* è un sistema di controllo fondamentale per un veicolo elettrico, migliora le prestazioni del pacco batteria e ne garantisce la sicurezza, quindi è importante studiare nel miglior modo possibile le varie problematiche cercando le soluzioni che limitano al massimo la dissipazione di potenza. In letteratura sono descritti diversi sistemi, ma pochi sono stati realizzati e quindi non sono stati caratterizzati a livello di consumo ed efficenza energetica, inoltre le varie strutture proposte appaiono poco flessibili, nel senso che non danno la possibilità di studiare diverse scelte circuitali ed architetturali. Nella maggior parte dei casi la circuiteria dedita alla misura della tensione di cella (raramente si trova un sistema che acquisisce la temperatura della singola cella), è composta da misuratori di tensione e multiplexer, quindi non è possibile memorizzare informazioni sulla vita della singola cella, non è possibile avere la tracciabilità necessaria per un sistema tanto importante quanto critico, a livello di sicurezza ed affidabilità, quale un *BMS* per veicoli elettrici.

Una piattaforma per lo sviluppo di un *BMS* è importante per studiare le scelte architetturali e circuitali dal punto di vista dell'efficienza energetica e della sicurezza del pacco batteria. A questo proposito, le scelte progettuali per realizzare la piattaforma, sono state tali da avere una struttura il più flessibile possibile, che consenta di misurare grandezze ed elaborare dati per valutare le varie scelte circuitali (per esempio un particolare circuito di bilanciamento).

La piattaforma finora realizzata consente lo studio di varie architetture di *BMS*:

- bilanciamento di tipo *cella batteria-cella ausiliaria*;
- bilanciamento di tipo *cella batteria-cella batteria*;
- bilanciamento di tipo *intra-modulo* e di tipo *modulare* (modulo inteso come segmento di batteria);

Analizzando le varie architetture è possibile fare delle scelte di compromesso per realizzare un *BMS* ottimizzato in tutti i suoi aspetti di affidabilità, rendimento energetico e costo.

Per quanto riguarda il livello di gerarchia più basso della piattaforma sono state progettate due tipi di schede, una che consente di effettuare delle misure accurate ma con elevato assorbimento di corrente, un'altra che effettua misure meno accurate ma con un assorbimento di corrente più basso. Sicuramente un obiettivo è quello di limitare il più possibile il consumo di potenza del *BMS*, allo stesso tempo è importante effettuare delle misure accurate per gestire al meglio il processo di bilanciamento dettato dal livello gerarchico più elevato, su cui è implementato il modello della cella. Il livello di accuratezza delle misure dovrà essere scelto in base a quanto richiesto dal processo di bilanciamento.

Per il livello di gerarchia intermedio è stata progettata e realizzata una scheda in grado di implementare la maggior parte delle funzionalità relative alla gestione della batteria. La progettazione e la scelta dei componenti è stata tale da limitare al massimo l'assorbimento di potenza dall'alimentazione.

Durante la fase di collaudo delle schede realizzate sono stati risolti alcuni errori, per quanto emerso dai risultati dei primi test sembrano funzionare tutte correttamente.

Appendice A

Appendice

A.1 Firmware *Level0*

A.1.1 *Level0-main.c*

```
#include <avr\io.h>
#include <avr\interrupt.h>
#include <avr\sleep.h>
#include <avr/eeprom.h>
#include <stdlib.h>
#include <stdint.h>
#include <util\delay.h>
#include "Level0-comm.h"
#include "Level0-ADC.h"
#include "Level0-pinout.h"
#include "Level0-EEPROM.h"

uint8_t voltage,temperature,command;
uint8_t i;
uint8_t EEMEM eeprom_space[EE_SIZE] = { 1, 2, 3, 4, 5, 6 };
uint8_t serial_number[SN_SIZE];
uint8_t age[AGE_SIZE];
volatile uint8_t pulses_count;
volatile uint8_t sending;

ISR(INT0_vect) {
  pulses_count++;
  TCNT0=0; //azzera timer0 interno
  while (!(PINB & _BV(REQ))); //attende ritorno a uno
  }
```

```
ISR(TIM0_OVF_vect) {
  GIMSK &= ~_BV(INT0); //finito tempo lettura impulsi disab
      interrupt ext
  sending=1; //semaforo verde
}

int main() {

DDRB |= (1<<CLK) | (1<<SDA);

PORTB |= (1<<REQ) | (1<<CLK) | (1<<SDA);

while(!eeprom_is_ready());
eeprom_read_block((void *)serial_number, (const void*)(
    eeprom_space+EE_ADR_SN), SN_SIZE);

TCCR0B |= (3<<CS00); //set prescaler
TIMSK0 |= _BV(TOIE0); //genera interrupt quando timer0 va in
    overflow

GIMSK |= _BV(INT0); //abilita interrupt esterno

set_sleep_mode(SLEEP_MODE_PWR_DOWN);

ADMUX |= (1<<ADLAR) | (1<<REFS0); //rif. interno e
    giustificazione a sinistra
ADCSRA |=  (7 << ADPS0); //max. prescaler

while (1) {

  //sleeping
  DDRB=0; //ogni uscita disabilitata

  GIMSK |= _BV(INT0); //abilita interrupt esterno
    sei();
  pulses_count=0;
  sending=0;

  sleep_mode();

  //wake up !!!

  while (!sending); //semaforo
```

```
67    DDRB |= (1<<CLK) | (1<<SDA); //enable outputs

69

      Send_byte(pulses_count);
71
      ADC_conversion(&voltage,&temperature);
73

75    switch (pulses_count){

77      //richiesta tensione e temperatura
        case TEMP_VOLT:
79                  Send_byte(voltage);
                    Send_byte(temperature);
81                  break;

83      //richiesta numero di serie
        case SN:          for (uint8_t j=0;j<SN_SIZE;j++) Send_byte(
            serial_number[j]);
85                  break;

87      //richiesta eta'
        case READ_AGE:        while(!eeprom_is_ready());
89                  eeprom_read_block(age,eeprom_space+EE_ADR_AGE,
                        AGE_SIZE);
                    for (uint8_t j=0;j<AGE_SIZE;j++) Send_byte(age[j
                        ]);
91                  break;

93      default:          break;
        }
95

      };
97

    }
```

A.1.2 *Level0-ADC.c*

```
1   #include <avr\io.h>
    #include <stdlib.h>
3   #include <stdint.h>
    #include <util\delay.h>
```

```
#include "Level0-comm.h"
#include "Level0-pinout.h"

void ADC_conversion (uint8_t* volt, uint8_t* temp ) {

  ADCSRA |= (1<<ADEN); //abilita ADC

  ADMUX &= ~3;
  ADMUX |= AI_VOLT; // selezione canale

  _delay_ms(1);

  ADCSRA |= (1<<ADSC); //Start ADC

  while (ADCSRA & (1<<ADSC)); //attende fine conversione

  *volt = ADCH;

  ADMUX &= ~3;
  ADMUX |= AI_TEMP;

  _delay_ms(1); //attesa

  ADCSRA |= (1<<ADSC); //Start ADC

  while (ADCSRA & (1<<ADSC)); //attende fine conversione

  *temp = ADCH;

  ADCSRA &= ~(1<<ADEN); //disabilita adc
}
```

A.1.3 *Level0-comm.h*

```
#ifndef COMM
#define COMM

#define CLK_PERIOD_US 50 //semiperiodo
#define LO_RATIO   (1)
#define HI_RATIO   (1)

#include "Level0-pinout.h"
```

```
9
#define SET0(bit) PORTB &= ~_BV(bit)
11 #define SET1(bit) PORTB |=  _BV(bit)

13
#define SEND_BIT_1 {SET0(CLK);\
15            SET1(SDA);\
              _delay_us(CLK_PERIOD_US*LO_RATIO);\
17            SET1(CLK);\
              _delay_us(CLK_PERIOD_US*HI_RATIO);\
19            SET1(CLK);\
              SET1(SDA);\
21            }

23
#define SEND_BIT_0 {SET0(CLK);\
25            SET0(SDA);\
              _delay_us(CLK_PERIOD_US*LO_RATIO);\
27            SET1(CLK);\
              _delay_us(CLK_PERIOD_US*HI_RATIO);\
29            SET1(CLK);\
              SET1(SDA);\
31            }

33 //commands
#define TEMP_VOLT        1
35 #define SN            2
#define READ_AGE        3
37 #define WRITE_AGE       4
#define READ_GP_EEPROM     5
39 #define WRITE_GP_EEPROM    6

41 uint8_t Read_byte();
void Send_byte(uint8_t byte);
43
#endif /*COMM*/
```

A.1.4 *Level0-EEPROM.h*

```
1 #ifndef LEVEL0EEPROM
#define LEVEL0EEPROM
3
```

```
#define EE_SIZE    0x40
#define SN_SIZE    6
#define AGE_SIZE   4

//Definizioni indirizzi eemprom

#define EE_ADR_SN         0x00
#define EE_ADR_AGE        0x10
#define EE_ADR_GP      0x20 //General Purpose

#endif
```

A.1.5 *Level0-pinout.h*

```
#ifndef PINOUT
#define PINOUT

#define CLK 2
#define SDA 0
#define REQ 1

#define PBTemp 3
#define PBVolt 4

#define AI_VOLT 2
#define AI_TEMP 3

#endif
```

A.2 Firmware *Level0* modificato

Viene riportata l'unica parte di codice modificata rispetto al firmware del *Level0* originale.

A.2.1 *Level0-ADC.c*

```
#include <avr\io.h>
#include <stdlib.h>
```

```
#include <stdint.h>
#include <util\delay.h>
#include "Level0-comm.h"
#include "Level0-pinout.h"

void ADC_conversion (uint8_t* volt, uint8_t* temp ) {

  ADCSRA |= (1<<ADEN); //abilita ADC

    DDRB |= (1<<PBTemp); // portB3 in uscita
    DDRB &= ~(1<<PBVolt); //portB4 in ingresso
    PORTB &= ~(1<<PBTemp);  //pone a 0 portb3 (pin2)

  ADMUX &= ~3;
  ADMUX |= AI_VOLT; //seleziona canale

  _delay_ms(1);

  ADCSRA |= (1<<ADSC); //Start ADC

  while (ADCSRA & (1<<ADSC)); //aspetta fine conversione

  *volt = ADCH;

  DDRB |= (1<<PBVolt); //portB4 in uscita
    DDRB &= ~(1<<PBTemp); //portB3 in ingresso
    PORTB &= ~(1<<PBVolt); //pone a 0 portb4 (pin3)

  ADMUX &= ~3;
  ADMUX |= AI_TEMP;

  _delay_ms(1); //attesa

  ADCSRA |= (1<<ADSC); //Start ADC

  while (ADCSRA & (1<<ADSC)); //aspetta fine conversione

  *temp = ADCH;

  ADCSRA &= ~(1<<ADEN); //disabilita adc

  DDRB &= ~(1<<PBTemp); //portB4 in ingresso
  DDRB &= ~(1<<PBVolt); //portB3 in ingresso
}
```

A.3 Firmware *Level1* su scheda di sviluppo *STK500*

A.3.1 *Level1-main.c*

Dopo la fase di inizializzazione (da riga 0 a 54), il programma entra in un ciclo infinito (da riga 54 a 85) in cui, per prima cosa si sincronizza con l'alimentatore controllato da un *VI* di *LabVIEW* con l'invio e la ricezione tramite la seriale di due caratteri (da riga 64 a 69), successivamente invia ad un *Level0* la richiesta di invio di temperatura e tensione di cella (riga 71). Rimane in attesa attiva fintantochè il numero di byte ricevuti è pari a al numero atteso (riga 75). Ogni volta che il *Level1* riceve un byte sulla periferica *SPI*, viene eseguita una routine di interrupt che pone il dato nel buffer di ricezione in un vettore ed infine aggiorna la variabile che conteggia il numero di byte ricevuti (da 19 riga a 22). Quando ha ricevuto tutti i dati il programma esce dal ciclo di attesa ed invia, tramite la porta seriale, i dati al *VI LabVIEW* (da 77 riga a 78).

```
#include <avr\io.h>
#include <stdlib.h>
#include <stdint.h>
#include <util\delay.h>
#include <avr\interrupt.h>
#include "Level1-pinout.h"
#include "Level1-comm.h"
#include "Level1-serial.h"

uint8_t data,i,command;
volatile uint8_t n_bytes_rec;
volatile uint8_t buffer[7];

char four_bits_to_ascii (uint8_t bin) {
  if (bin<0x0a) return '0'+bin;
  else return 'A'+(bin-0x0a);
}

ISR (SPI_STC_vect) {            //viene eseguita quando la perif.
    SPI riceve un dato
```

```
20 buffer[n_bytes_rec] = SPDR;
   n_bytes_rec++;
22 }

24 int main() {

26 unsigned char waitAlim = 'b';

28 for(int i=0; i<8; i++)
     buffer[i] = i;
30
   DDR_REQ  |=  _BV(REQ);
32 DDR_CLK &= ~(1<<CLK);
   DDR_SDA &= ~(1<<SDA);
34
   DDR_L0_CAT  |=  _BV(L0_CAT);
36 PORT_L0_CAT &= ~_BV(L0_CAT); //sempre a 0 V

38 PORT_SDA |=  _BV(SDA); //pull up attivi
   PORT_CLK |=  _BV(CLK);
40 PORT_REQ &= ~_BV(REQ);

42 DDR_LED  |= (1<<1);

44 DDRB &= ~(1<<SS);  //slave select in ingresso
   DDRA |= (1<<ctrSS);   //utilizzato A0 come SS control
46 PORTA &= ~(1<<ctrSS);

48 SPCR = _BV(CPHA) | _BV(CPOL) | _BV(SPIE) | _BV(SPE);

50 sei(); //abilita le interruzioni

52 USART_Init(); //inizializzazione periferica seriale

54 while(1) {

56 PORT_LED |= (1<<1); //heartbeat
   _delay_ms(100);
58 PORT_LED &= ~(1<<1);

60 n_bytes_rec = 0;

62 command = TEMP_VOLT; //richiede al Level0 temp. e tens. cella
```

```
64 waitAlim = 'b';

66 //aspettare che alimentatore imposti tensione
   while (!(waitAlim == 'a')){
68   waitAlim = USART_Receive();
     }
70
   for (uint8_t k=0;k<command;k++) SEND_REQ;  //invio richiesta
72
   switch (command) {
74
       case TEMP_VOLT: while(n_bytes_rec<(2+DEBUG_MODE));
76       for(int i=0; i<3; i++) {
           USART_Transmit(four_bits_to_ascii((buffer[i]>>4)&0x0F));
78         USART_Transmit(four_bits_to_ascii(buffer[i]&0x0F));
           }
80       break;

82     case SN: while(n_bytes_rec<(6+DEBUG_MODE)); break;

84     case READ_AGE: while(n_bytes_rec<(4+DEBUG_MODE)); break;
       }
86
       _delay_ms(5000);
88
                }
90
}
```

A.3.2 *Level1-serial.c*

```
1 #include <avr/io.h>
  #include <avr/interrupt.h>
3
  void USART_Init() {
5   UBRRH = 0;     //Set baud rate a 115200 senza prescaler interno
        e fosc=3.6864Mhz
    UBRRL = 3; //error = 0%
7   UCSRB = (1<<RXEN)|(1<<TXEN); //abilita TX & RX
    UCSRA = (1<<U2X); //abilita Double Speed
9 }
```

```
11
   void USART_Transmit(uint8_t data ) {
13   while ( !( UCSRA & (1<<UDRE)) ); //aspetta buffer TX vuoto
     UDR = data; //dato nel buffer TX e invio
15 }

17 unsigned char USART_Receive() {
     while ( !(UCSRA & (1<<RXC)) ); //prende il dato dal buffer RX
19 return UDR;
   }
```

A.3.3 *Level1-comm.h*

```
   #ifndef COMM
2  #define COMM

4  #include "Level1-pinout.h"

6  #define SET0(bit) PORTB &= ~_BV(bit)
   #define SET1(bit) PORTB |=  _BV(bit)
8

10 #define SEND_BIT_1 {SET1(CLK);\
              SET1(SDA);\
12            _delay_us(CLK_PERIOD_US/2);\
              SET0(CLK);\
14            _delay_us(CLK_PERIOD_US/2);\
              SET1(CLK);\
16            SET1(SDA);\
              }
18

20 #define SEND_BIT_0 {SET1(CLK);\
              SET0(SDA);\
22            _delay_us(CLK_PERIOD_US/2);\
              SET0(CLK);\
24            _delay_us(CLK_PERIOD_US/2);\
              SET1(CLK);\
26            SET1(SDA);\
              }
28
```

```
30 #define SEND_REQ  {PORT_REQ |=  _BV(REQ);\
              _delay_us(10);\
32            PORT_REQ &= ~_BV(REQ);\
              _delay_us(1000);\
34            }


36
   //commands
38 #define TEMP_VOLT        1
   #define SN            2
40 #define READ_AGE       3
   #define WRITE_AGE      4
42 #define READ_GP_EEPROM     5
   #define WRITE_GP_EEPROM    6
44
   #define DEBUG_MODE       1
46


48 uint8_t Read_byte();
   uint8_t Read_cmd(uint8_t* cmd);
50 void Send_byte(uint8_t byte);

52 #endif /*COMM*/
```

A.3.4 *Level1-serial.h*

```
1 #ifndef SERIAL
  #define SERIAL
3
  void USART_Init();
5 void USART_Transmit(uint8_t);
  unsigned char USART_Receive(void);
7
  #endif
```

A.3.5 *Level1-pinout.h*

```
  #ifndef PINOUT
2 #define PINOUT
```

```
4 #define PORT_CLK   PORTB
  #define PORT_SDA   PORTB
6 #define PORT_REQ   PORTB
  #define PORT_LED   PORTC
8 #define PORT_L0_CAT PORTB
  #define PORT_CS    PORTB
10
  #define DDR_CLK    DDRB
12 #define DDR_SDA    DDRB
  #define DDR_REQ    DDRB
14 #define DDR_LED    DDRC
  #define DDR_L0_CAT   DDRB
16 #define DDR_CS     DDRB

18 #define PIN_CLK    PINB
  #define PIN_SDA    PINB
20


22 #define CLK       7
  #define SDA       5
24 #define REQ       2
  #define L0_CAT      0
26 #define CS        2

28 #define SS        4
  #define ctrSS     0
30
  #endif
```

A.4 Batteria al Litio

A.4.1 Batterie Litio-ioni

Le batteria *Litio-ioni* sono formate da celle i cui elettrodi, formati da composti di *Litio*, scambiano ioni di tipo $Li+$. I flussi di ioni $Li+$ variano la loro direzione, da un elettrodo all'altro, a seconda che la cella sia in fase di carica o in fase di scarica.
Il catodo della cella solitamente è costituito da un ossido metallico con una struttura a strati, per esempio l'ossido di litio cobalto ($LiCoO2$), connesso

ad una linguetta in alluminio, con cui contattare l'elettrodo positivo. L'anodo della cella è solitamente realizzato con materiali a base di carbonio (capaci di accettare e donare significative quantità di litio; $Li : C = 1 : 6$) come la graffite, sempre con struttura a strati, collegata ad una linguetta questa volta in rame. Nei processi di carica e scarica, gli ioni di litio vengono inseriti o estratti nello spazio tra i livelli atomici dei materiali attivi ([15]).

Vantaggi	**Svantaggi**
Alta efficienza energetica	Degrado alle alte temperature
Basso invecchiamento dovuto ai cicli di utilizzo	Perdita di capacità e rischio di fuga termica con sovraccarica
Assenza di effetto memoria	Necessità di un circuito di gestione
Alti valori di energia specifica e densità di energia	Possibilità di rotture o fuga termica se batteria sottoposta a pressioni eccessive
Bassa auto-scarica	Densità di potenza inferiore rispetto *NiMH* e *NiCd*

Tabella A.1: Vantaggi e svantaggi delle batterie al Litio-ioni ([15])

I maggiori vantaggi di questo tipo di batterie risiedono negli elevati valori di energia specifica e di densità di energia, per questo sono usate molto in ambiti in cui il peso e l'ingombro è un problema. Inoltre le batterie *Li-ion* presentano valori di auto-scarica molto bassi (dal 2% al 8% per mese), range di temperature di esercizio abbastanza largo (si parla di una range da *-20˚C* a *60˚C* in fase di carica e da *-40˚C* a *65˚C* in scarica), tensioni della singola cella che vanno da 2.5 a 4.2 V (circa il triplo di batterie *NiCd* o *NiMH*).
Di contro si ha un degrado irreversibile se la tensione della cella scende sotto i $2V$ e nel caso in cui si sovracarichi. Sono necessari dei sistemi di controllo per proteggere la batteria in fase di carica-scarica e da condizioni di temperatura anomali (si ha infatti perdita permanente di capacità se si superano i *65˚C*, possibilità di generare gas infiammabili ed addirittura deflagrazioni) ([15]).

A.4.2 Batterie litio-polimeri

Le batterie *polymer Li-ion* (*Li-poly*) garantiscono le solite prestazioni delle batterie *Li-ion*, a cui va aggiunto il vantaggio di avere una dispositivo dalle dimensioni più compatte (spessore ridotto) che le rendono adatte in quelle applicazioni in cui l'ingombro è un fattore determinante. Tale vantaggio è

dovuto al fatto che l'elettrolita è un materiale plastico e non liquido come nelle celle *Li-ion* che necessitano quindi di un contenitore robusto per assicurarne l'impermeabilità. Il contenitore delle celle *Li-poly*, che può essere realizzato con un foglio di alluminio, è necessario che sia di tipo stagno per i seguenti motivi ([15]):

- il polimero viene impregnato con una soluzione acquosa o con elettroliti in gel per per aumentare la conduttività dell'elettrolita solido (che a temperatura ambiente è più bassa di quello in soluzione acquosa);
- la cella può essere danneggiata dall'ingresso di umidità all'interno della struttura.

Bibliografia

[1] Li Siguang , Zhang Chengning, “Study on Battery Management System and Lithium-ion Battery”.

[2] Elecctropaedia: “http://www.mpoweruk.com/bms.htm”.

[3] John Chatzakis, Kostas Kalaitzakis, Nicholas C. Voulgaris, and Stefanos N. Manias, “Designing a New Generalized Battery Management System”.

[4] Minxin Zheng, Bojin Qi, Hongjie Wu School of Mechanical Engineering and Automation, Beijing University of Aeronautics and Astronautics No.37 Xueyuan Road, Beijing 100083, P.R.China, “A Li-ion Battery Management System Based on CAN-bus for Electric Vehicle”.

[5] Li Siguang, Zhang Chengning School of Mechanical and Vehicular Engineering Beijing Institute of Technology Beijing,China, “Study on Battery Management System and Lithium-ion Battery”.

[6] Thomas Stuart and Fang Fang The University of Toledo, Xiaopeng Wang Visteon Co., Cyrus Ashtiani, DaimlerChrysler, Ahmad Pesaran National Renewable Energy Laboratory, “A Modular Battery Management System for HEVs”.

[7] Hong-Sun Park, Student Member, IEEE, Chong-Eun Kim, Member, IEEE, Chol-Ho Kim, Student Member, IEEE, Gun-Woo Moon, Member, IEEE, and Joong-Hui Lee, Member, IEEE, “A Modularized Charge Equalizer for an HEV Lithium-Ion Battery String”.

[8] Yuang-Shung Lee, Member, IEEE, and Guo-Tian Cheng, “Quasi-Resonant Zero-Current-Switching Bidirectional Converter for Battery Equalization Applications”.

[9] Yuang-Shung Lee, Member, IEEE, and Ming-Wang Cheng, "Intelligent Control Battery Equalization for Series Connected Lithium-Ion Battery Strings".

[10] Chol-Ho Kim, Young-Do Kim, Gun-Woo Moon1 and Hong-sun Park, "Individual Cell Voltage Equalizer Using Selective Two Current Paths for Series Connected Li-ion Battery Strings".

[11] Hong-Sun Park, Associate Member, IEEE, Chol-Ho Kim, Student Member, IEEE, Ki-Bum Park, Student Member, IEEE, Gun-Woo Moon, Member, IEEE, and Joong-Hui Lee, Member, IEEE, "Design of a Charge Equalizer Based on Battery Modularization".

[12] Hong-Sun Park, Student Member, IEEE, Chong-Eun Kim, Member, IEEE, Chol-Ho Kim, Student Member, IEEE, Gun-Woo Moon, Member, IEEE, and Joong-Hui Lee, Member, IEEE, "A Modularized Charge Equalizer for an HEV Lithium-Ion Battery String".

[13] Doug Brooks and Dave Graves, "Current Carrying Capacity of Vias".

[14] datasheet Atmel, ATTiny13A.

[15] E. Leonardi, "Studio e caratterizzazione di celle litio polimero per applicazioni su veicoli ibridi e d elettrici" (Tesi di laurea).

[16] F. Baronti, G. Fantechi, E. Leonardi, R. Roncella, R. SAletti, "Enhanced Model for Lithium-Polymer Cells including Temperature Effects," IEEE Industrial Electronics, IECON 2010 - 36nd Annual Conference on