Università degli Studi di Padova

Dipartimento di Ingegneria dell'Informazione

Corso di Laurea Triennale in
Ingegneria dell'Informazione

Controllo della temperatura in un pacco batteria, caso d'uso per LTC6811

Relatore:
Prof. Matteo Meneghini

Laureando:
Luca Bombonati
Matr.2000247

Anno Accademico 2022/2023

Dedicato a mamma e papà

Abstract

In questa tesi di laurea si espone il lavoro di progettazione e realizzazione di un sistema capace di misurare la temperatura in un pacco batteria mediante l'utilizzo di LTC6811.

Sviluppato dalla Analog Devices, LTC6811 è un dispositivo capace di misurare la tensione e la temperatura su uno stack di batterie composto da 16 segmenti.

Il caso in analisi è l'implementazione del sistema di gestione della batteria in un prototipo di monoposto conforme al regolamento della Formula SAE e pronto a partecipare agli eventi di categoria. Dopo una breve presentazione della competizione, verrà trattato il BMS, di quali schede si compone e di come si possa configurare il dispositivo della Analog Devices. Verrà in seguito discussa, in particolare, la conversione analogico-digitale, la lettura dei registri e la rappresentazione in scala Celsius della temperatura. Infine seguirà una piccola riflessione sulla realizzazione del Battery Management System mediante l'utilizzo del dispositivo AURIX, uno dei prodotti della nota società Infineon.

Indice

Capitolo 1

LTC6811: caso d'uso

L'avvento delle vetture elettriche è un fenomeno indiscutibile in grado di rivoluzionare l'intero mercato dell'automotive e dei molti panorami ad esso collegati. Solo in Italia la vendita di auto ibride ed elettriche ha segnato, nel periodo 2020-2021, un incremento del 92.9%, crescita marcata principalmente delle immatricolazioni delle vetture elettriche, che da circa 58 mila sono arrivate a oltre 118 mila [1], toccando le 200 mila unità al termine del 2022.

È possibile riscontrare questo trend non solo tra i veicoli prodotti in serie, ma anche le maggiori categorie del motosport seguono molto attentamente questa rivoluzione. Mentre la Formula E acquista sempre più rilevanza, la Formula 1 ha già annunciato che, a partire dal 2026, aumenterà la capacità delle power unit, ossia la componente ibrida delle monoposto, dagli attuali 120kW a 350kW, garantendo circa 476 cavalli.

Tuttavia, le batterie agli ioni di litio, le più utilizzate al momento, possono essere molto pericolose se utilizzate in alcune condizioni e si rende, pertanto, molto importante e necessario l'utilizzo di tecnologie sempre più avanzate che possano monitorare i parametri vitali dei pacchi batteria per valutarne, poi, lo State of Health (SOH) complessivo ed evitare spiacevoli incidenti.

I parametri da monitorare sono molteplici, ma di particolare importanza è la temperatura delle celle. Gli accumulati Li-ion, infatti, sono molto sensibili a questo valore, non solo per la perdita di prestazioni al di fuori del range operativo ottimale, ma anche per il rischio che l'esposizione alle alte o alle basse temperature comporta.

Il caso d'uso che si vuole analizzare è l'applicazione di queste tecnologie di monitoraggio, attraverso l'implementazione di un Battery Management System, con particolare attenzione al rilevamento della temperatura, in una monoposto

elettrica che rispetti il regolamento previsto dalla Formula Student Germany e che possa, infine, gareggiare nelle competizione di categoria.

Le sfide affrontate Il lavoro che si descrive in questa tesi, come parte del contributo dato dal sottoscritto alla realizzazione della vettura per la stagione 2022-2023, è irrisorio a confronto con l'immenso sistema di gestione della batteria implementato nella *SG-06*. La trattazione si concentrerà, infatti, dopo un breve studio sugli effetti che le temperature possono causare sui pacchi batterie, sul comportamento delle funzioni che permettono la lettura e l'elaborazione dei dati provenienti dai sensori posti a contatto con le celle.

Le sfide più insidiose affrontate durante l'anno sono state: individuare la funzione di regressione che potesse rendere i dati misurati precisi, possibile grazie alle nozioni acquisite durante il corso di "Introduzione al Machine Learning", e ottimizzare le funzioni sopra citate con l'obiettivo di ottenere il codice più performante possibile conformandolo al regolamento.

1.1 La Formula SAE

La formula SAE è una competizione di design ingegneristico dove studenti provenienti dalle università di tutto il mondo progettano una monoposto stile Formula. L'obiettivo che ogni squadra si pone è progettare, verificare e, infine, assemblare un prototipo di auto da corsa completandolo entro la stagione delle gare nelle quali verrà decretato il migliore.

Dopo la sua prima edizione nel 1981, la formula SAE si è diffusa a tal punto da comprendere ben tre diverse categorie di vetture: *Combustion* (classe 1C), composte da un motore a combustione interna, *Electric* (classe 1E), spinte da un motore completamente elettrico, e, infine, *Driverless* (classe 1D), versione alternativa della vettura elettrica, ma a guida autonoma.

Essendo una competizione di design, è presente un regolamento ferreo che mira, innanzitutto, a salvaguardare la sicurezza dei piloti, anch'essi componenti del team, ma anche a rendere cosciente ogni studente partecipante delle scelte ingegneristiche implementate nella vettura. Così facendo l'automobile migliore non è necessariamente la più veloce e performante in pista, ma può essere anche quella il cui prototipo è stato studiato più approfonditamente.

Ogni evento è suddiviso in sette prove: tre dette "statiche" e le "dinamiche".

La prime tre prove sono:

- **Engineering Design**: il prototipo viene discusso con un gruppo di giudici provenienti da importanti aziende dell'automotive, giustificando le scelte operate nel progetto;
- **Cost and Manufacturing**: il team presenta il costo dell'intera vettura dimostrando di saper valutare le opzioni, l'effettività dei piani finanziarti e i rischi associati alla gestione del progetto;
- **Business Plan Presentation**: presentazione di un Business Case di focus automotive davanti a una giuria che valuterà il grado di realisticità e innovazione del progetto.

Le prove dinamiche invece sono costituite da:

- **Acceleration**: il prototipo deve portare a termine un rettilineo lungo 75 metri nel minor tempo possibile;
- **Skidpad**: il cui obiettivo è valutare la tenuta laterale della vettura, la quale è obbligata a percorrere due giri di un circuito a forma di "8";
- **Autocross**: la monoposto deve percorrere un tracciato lungo quasi un chilometro composto da rettilinei, curve veloci e chicanes nel minor tempo possibile;
- **Endurance**: è la conclusione di ogni evento con l'obiettivo di coprire una distanza di 22 chilometri all'interno di un tracciato, valutando affidabilità e performance.

I principali eventi europei sono: *Formula ATA*, in Italia, presso il circuito di Varano de' Melegari (PR), *Formula Student EAST* all'"Hungaroring", casa del Gran Premio d'Ungheria di Formula 1, *Formula Student Austria*, ospiti al "Red Bull Ring", e, infine, *Formula Student Gemrany*, l'evento più celebre ed atteso dell'anno.

1.1.1 L'Università di Padova

L'Università degli Studi di Padova partecipa a questa competizione con il nome di *RaceUp Team* dal 2006 grazie alla lungimiranza del Professor Giovanni Meneghetti, faculty advisor del progetto. Nato come sola vettura a combustione interna, nell'ottobre 2014 venne dato inizio allo sviluppo del prototipo elettrico

che verrà battezzata, 2 anni dopo, come 'Origin-E' al suo debutto all'evento in Germania.

Nel 2017, arrivarono subito i primi successi per la squadra elettrica, la quale si aggiudicò il primo posto nelle categorie di "Design" e "Business Plan". Durante gli anni successivi molti riconoscimenti ed obiettivi vengono raggiunti e il palmares delle vittorie viene arricchito sempre più. L'incoronazione più importante, tuttavia, venne raggiunta nel 2022 all'evento ungherese, dove la divisione elettrica vinse il premio come primo team italiano di categoria, classificandosi sopra alle vetture del Politecnico di Milano e dell'Università di Bologna.

Nel 2023, infine, la vettura `SG-06` ha portato a termine, per la prima volta nella storia del team elettrico, la prova dell'endurance, concludendo l'evento con un ottimo ottavo posto complessivo.

Capitolo 2

Le batterie agli ioni di litio

Le prime batterie agli ioni di litio in commercio vennero introdotte dalla Sony nel 1991 e guidarono una vera e propria rivoluzione del mercato. Questi accumulatori ricaricabili sfruttano la grande proprietà di ossidoriduzione reversibile di questo elemento per immagazzinare energia.

Gli accumulatori agli ioni di litio possiedono delle caratteristiche chiave che hanno portato, nel corso del XX secolo, all'adozione di massa di questi dispositivi.

2.1 Le proprietà

La proprietà principale delle batterie Li-ion è la tensione per cella: grazie all'utilizzo di materiali attivi che rendono più basso il potenziale dell'anodo, si riesce ad ottenere una differenza di potenziale di circa 3.2-4.2 V, molto più alto rispetto alle precedenti in piombo-acido. Nonostante l'evidente vantaggio nell'utilizzo di questo nuovo materiale, questo range di tensione impone una grossa limitazione: se le celle vengono sovraccaricate si verifica la sintesi dell'ossido di cobalto, mentre la scarica eccessiva produrrà ossido di litio rendendo irreversibili le reazioni elettrochimiche. In entrambi i casi la cella risulterà inutilizzabile.

In secondo luogo, le batterie in analisi possiedono un'alta efficienza di carica-scarica. Si introduce, per caratterizzare questo parametro, l'*efficienza di Coulomb*, definito come il rapporto tra la capacità di scarica dopo un ciclo di carica completa e la capacità di carica del medesimo ciclo. Si può quindi verificare, data la resistenza interna molto piccola di questi dispositivi, che è possibile raggiungere facilmente un'efficienza di circa il 93,08% [2].

Infine grazie alle semplici reazioni chimiche che avvengono, che non comportano modifiche morfologiche rilevanti nei materiali attivi durante i normali cicli

di carica e scarica, questi accumulatori possono essere molto longevi.

Tuttavia, non tutte le batterie agli ioni di litio sono uguali. Le principali differenze tra il caso d'uso in analisi e applicazioni in elettronica di consumo sono:

- il sistema di controllo, in grado si supportare il funzionamento ottimale e sicuro;
- il design termico che permetta di prevenire la fuga termica.

2.2 Gli effetti della temperatura

Una delle maggiori limitazioni è l'impatto che ha la temperatura sulle operazioni delle batterie Li-ion.

Si individua la regione compresa tra i -20°C e i 60°C come il range di temperature generalmente accettato per gli accumulatori in questione [3]; tuttavia, l'intervallo ottimale per raggiungere il rendimento maggiore si riduce all'intorno $15°C \div 35°C$ [4]. Al di fuori di questi valori, le batterie agli ioni di litio si degradano molto facilmente ed aumenta, conseguentemente, il rischio che si verifichino problemi di sicurezza.

L'innalzamento della temperatura è dovuta principalmente alle reazioni chimiche che avvengono in ogni cella. In particolare, la relazione 2.1, conosciuta come l'*equazione di Arrhenius*

$$k = k_0 e^{-\frac{\triangle E^{\dagger}}{RT}} \tag{2.1}$$

mette in relazione la costante di velocità k, necessaria per raggiungere l'energia d'attivazione e dare avvio alle reazioni elettrochimiche, con la variazione di temperatura.

Si possono, quindi, distinguere gli effetti delle alte e delle basse temperature, ma ciò che accomuna entrambe le situazioni è la reazione che si può verificare se la sollecitazione termica persiste nel tempo: nel peggiore dei casi, i pacchi batteria possono prendere fuoco o esplodere, ferendo chi è nei dintorni, rilasciando gas nocivi nell'ambiente.

È quindi di facile comprensione l'importanza dell'implementazione di un sistema che possa costantemente monitorare le temperature delle celle che costituiscono l'accumulatore per evitare spiacevoli inconvenienti.

2.2.1 Le basse temperature

Le performance delle batterie agli ioni di litio diminuiscono per l'effetto delle temperature inferiori ai 0°C. A basse temperature, infatti, la viscosità degli elettroliti aumenta e, pertanto, si riduce la conduttività ionica.

Si verifica, inoltre, un aumento della resistenza interna causato dall'aumento dell'impedenza della migrazione direzionale degli ioni [5].

È stato dimostrato, infine, che, se esposte a una temperatura di -40°C, le celle possono perdere in termini di densità di potenza ed energia, rispettivamente, fino al 98.75% e fino al 95% [6].

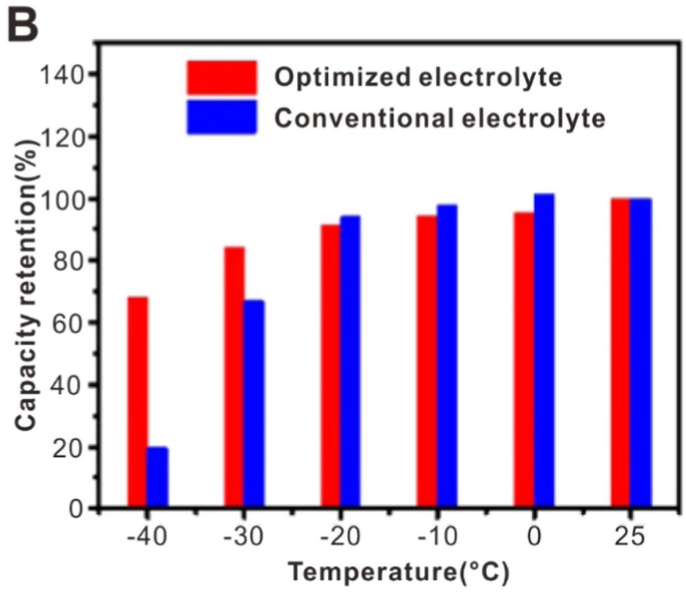


Figura 2.1: Effetto della temperatura sugli accumulatori Li-ion

Un altro effetto tipico delle batterie lithium-ion che avviene alle basse temperature è il "lithium plating". Con questo termine si intende l'accumulo di litio sulla superficie dell'anodo che, non trasferendosi nei siti di intercalazione del carbonio [7], comporta la riduzione della capacità delle celle. È inoltre possibile che questo fenomeno, il quale si presenta comunemente sotto forma di cristalli, causi corti circuiti interni.

2.2.2 Le alte temperature

Gli effetti delle alte temperature sono molto più complessi e comportano un rischio per la salute di maggior grado.

Durante i normali cicli di carica e scarica di una batteria agli ioni di litio si possono individuare processi di natura reversibile ed irreversibile che generano calore.

Processi reversibili Sono quei processi che generano calore entropico [8], originato dalla variazione reversibile di entropia durante le reazioni elettrochimiche.

Processi irreversibili Questi processi, invece, comprendono la polarizzazione attiva, dovuta alla differenza tra il potenziale operativo e il potenziale di circuito aperto che porta all'aumento della resistenza di carica, il riscaldamento ohmico e il cambiamento di entalpia, dovuto al cambiamento di fase nei catodi [9]. Il calore ohmico, invece, è dovuto alla resistenza degli elettrodi e degli elettroliti che impediscono il trasporto di cariche.

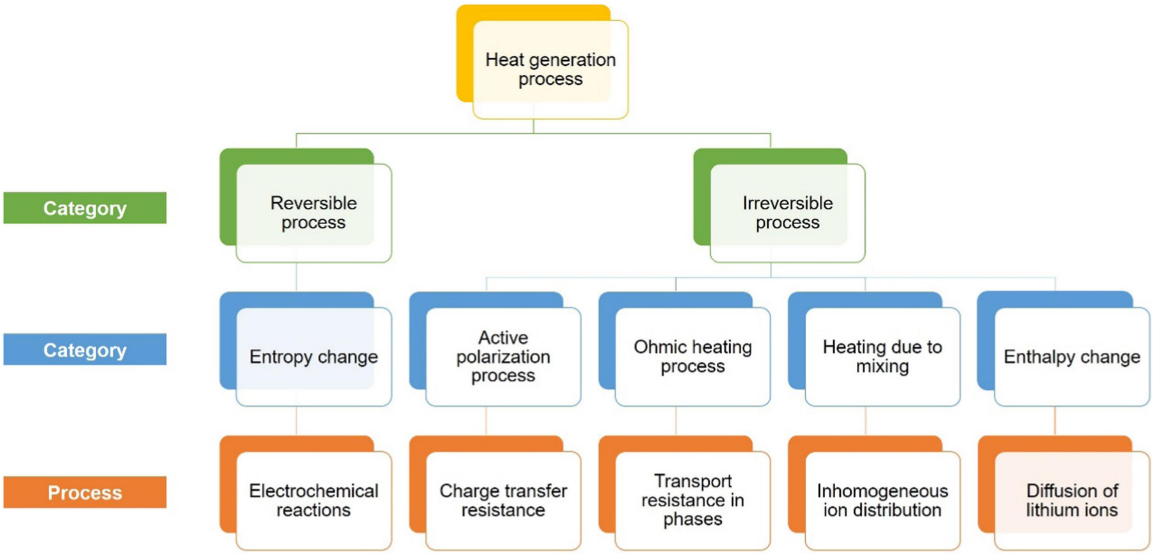


Figura 2.2: Categorie e processi che generano calore nelle batterie Li-ion

Infine, si possono anche verificare delle fughe termiche. Questi eventi sono molto pericolosi in quanto le alte temperature scatenano reazioni esotermiche che, rilasciando ulteriore calore, innalzano in modo incontrollato le temperature dell'intero pacco batteria

Capitolo 3

Battery Management System

Un cardine dello sviluppo del prototipo SG-06 2023 è stato, senza dubbio, il nuovo Battery Management System (BMS) ovvero il sistema grazie al quale è possibile misurare e controllare i valori della temperatura e della tensione di ogni cella della batteria.

Conosciuto anche con il nome di Accumulator Management System, il BMS è di così cruciale importanza che il regolamento della Formula Student [10] gli dedica una sezione completa. È proprio da tale documento che si apprende l'obbligatorietà di questo sistema per ogni accumulatore connesso al TS (Traction System):

> EV 5.8.1, *"Each TS accumulator must be monitored by an AMS whenever the LVS is active or the accumulator is connected to a charger"*

Alla successiva regola EV 5.8.3 vengono, inoltre, definite quali misure minime devono essere effettuate in modo continuativo:

- tensione di ogni cella;
- corrente in uscita dal pacco batteria;
- temperatura di almeno 30% delle celle.

Di particolare importanza sono le condizioni previste dalla regola EV 5.8.7 del regolamento sotto le quali il BMS ha il compito di disattivare immediatamente il sistema di trazione. Queste si verificano quando tensione, corrente o temperatura raggiungono valori definiti "critici", valutati utilizzando i datasheet dei produttori delle celle, persistenti per:

- 500 millisecondi nel caso di tensione e corrente;

- 1 secondo per temperature.

3.1 BMS 2023

Per il prototipo 2023, il sistema in analisi è stato implementato grazie all'utilizzo della board "Arduino Due" [11] alla quale viene connessa successivamente una shield che, mediante una connessione in isoSPI, permetterà di comunicare con i sedici slave montati sulle 288 celle che compongono il pacco batteria.

Figura 3.1: Board "Arduino Due"

L'implementazione del sistema sopra descritto è stata resa possibile, già dalla stagione 2021-2022, dall'utilizzo di due specifici dispositivi: `LTC6820` e `LTC6811` entrambi sviluppati dalla Analog Devices, importante multinazionale statunitense attiva nella produzione di semiconduttori.

Nello specifico, `LTC6820` [12] fornisce una semplice comunicazione SPI tra due dispositivi isolati mediante l'utilizzo di una singola connessione a doppino intrecciato; in altre parole, questo componente è un "transceiver" capace di convertire la trasmissione a quattro linee in una che ne utilizza due.

L'utilizzo della comunicazione tramite isoSPI risulta molto conveniente nel sistema che si vuole costruire in quanto si utilizzano segnali differenziali su una coppia bilanciata di cavi, nessuno dei quali a massa, capaci di rigettare il rumore di modo comune generato da interferenze esterne.

Il secondo dispositivo, `LTC6811`, sarà oggetto di una più approfondita analisi nel successivo capitolo.

Per la comprensione dei successivi codici, viene più sotto riportata la definizione della variabile globale `g_bms`:

```
1 struct BMS {
2   Slave slaves[SLAVE_NUM];
```

```
  uint16_t max_volt;
  uint16_t min_volt;
  uint32_t tot_volt;
  uint16_t max_temp;
  uint16_t min_temp;
  uint16_t tot_temp;
  uint8_t max_temp_slave;
  LEM lem;
  bool sdc_closed;
  uint32_t fault_volt_tmstp;
  uint32_t fault_temp_tmstp;
  Mode mode;
  Precharge precharge;
};

BMS g_bms = {};
```

3.1.1 Le schede utilizzate

Il BMS 2023 si compone, complessivamente, di tre schede progettate dai colleghi del reparto 'Electronics':

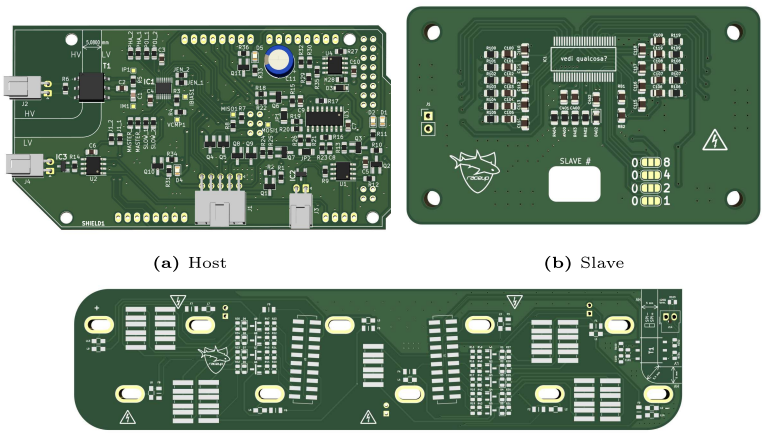
- **Host** (figura 3.2a), una shield connessa alla board Arduino;
- **Slave** (3.2b), le schede (sedici) che monteranno `LTC6811` e con le quali l'Host comunicherà;
- **Interfaccia** (3.2c), la scheda che verrà connessa direttamente con le celle della batteria

Il risultato finale, dopo essere stato montato sulle celle, sarà come in figura 3.3a e 3.3b.

3.2 LTC6811

`LTC6811` [13] è, come da definizione del datasheet, un monitor di pile di batterie capace di misurare fino a dodici celle collegate in serie con un errore di misura totale inferiore a $1.2mV$.

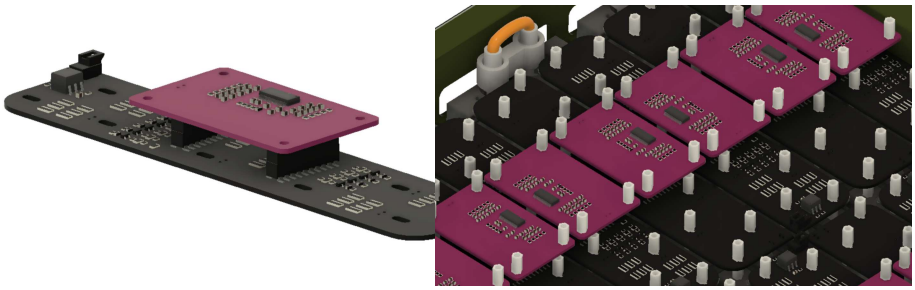
Per l'obiettivo che il team si pone, questo dispositivo, date le sue proprietà, è uno dei candidati migliori per la progettazione del sistema BMS.



(a) Host (b) Slave

(c) Interfaccia

Figura 3.2: Modelli 3D delle schede usate



(a) BMS complessivo (b) Sul pacco batteria

Figura 3.3: Modello 3D del BMS

Questo prodotto, infatti, assicura:

- un'ottima risoluzione dei dati acquisiti;
- rapidità durante la fase di acquisizione;
- la possibilità di essere connesso in serie o in parallelo con altri dispositivi del suo genere per misurare contemporaneamente più celle.

Ne esistono due versioni:

- **LTC6811-1**: per l'implementazione di una "daisy chain";
- **LTC6811-2**: utilizzabile attraverso gli indirizzi.

Utilizzare la "daisy chain" implica, per la sua costituzione, la connessione in serie di tutti i dispositivi; tale approccio comporta aspetti sia positivi sia negativi. Il motivo principale che può spingere ad utilizzare questa tecnica è, indubbiamente, la minore quantità di cavi rispetto all' "addressing" con la conseguenza

di ottenere un cablaggio molto semplificato. A questo beneficio si contrappone la criticità dell'utilizzo di un unico comando che attraversa ogni dispositivo della catena fino alla periferica desiderata; infatti, oltre a introdurre numerosi ritardi nella comunicazione, comporta il rischio di essere interrotto qualora un dispositivo posto in posizione intermedia dovesse guastarsi.

Ovviamente, l'utilizzo della catena rimane vantaggioso in applicazioni in cui il peso e la complessità del cablaggio risultino prioritari rispetto ai potenziali ritardi nella comunicazione. Tuttavia, assicurare un maggiore controllo e flessibilità nella trasmissione dei dati è, nel caso specifico, la preoccupazione principale, e implementando la soluzione dell'*addressing* si consentirà, inoltre, una gestione più efficiente del sistema. Gli slave verranno, pertanto, connessi in parallelo e verrà definito un indirizzo di quattro bit per ogni dispositivo.

Dal datasheet del componente si evince, inoltre, la facilità di utilizzo del `LTC6811` utilizzato in un sistema basato sull'addressing. È possibile, infatti, comunicare mediante la conversione da segnale SPI a isoSPI, controllata dal `LTC6820`, attraverso pochi e semplici comandi che devono essere correttamente strutturati.

3.2.1 Inizializzazione di un comando

Per inizializzare un comando è necessario, innanzitutto, conoscerne la destinazione: se l'obiettivo è eseguire l'operazione su ogni periferica connessa si utilizzerà un comando 'broadcast', altrimenti sarà obbligatorio specificare l'indirizzo del dispositivo desiderato.

Composto da due byte, il comando sarà così formato:

CMD0	[1:0]	A[3]	A[2]	A[1]	A[0]	CC[10]	CC[9]	CC[8]
CMD1	CC[7]	CC[6]	CC[5]	CC[4]	CC[3]	CC[2]	CC[1]	CC[0]

I parametri `A[X]` corrispondono ai quattro bit identificativi dell'indirizzo del dispositivo e, pertanto, vanno impostati, insieme al primo bit di `CMD0`, al valore binario `0` se si vuole comunicare con tutti i dispositivi.

`CC[X]`, invece, rappresenta il Command Code, di cui seguirà, nei capitoli successivi, una trattazione specifica.

Dopo aver generato il comando, è necessario calcolare un *Packet Error Code* e aggiungerlo in coda alla trasmissione. Il PEC è un controllo di ridondanza ciclico (CRC), composto da due byte, utilizzato per controllare la correttezza dei dati: ogni comando o dato inviato e/o ricevuto viene ritenuto valido se e solo se i due codici di controllo coincidono.

Si ottiene, pertanto, una trasmissione formata da:

CMD0	CMD1	PEC0	PEC1

A questo punto segue:

- il campionamento dei dati in ingresso nel caso sia stato inviato un comando `Read`;
- l'invio dei dati dal master allo o agli slave in un comando `Write`.

In entrambi i casi, le informazioni ricevute o inviate sono seguiti dal loro codice PEC.

La trasmissione finale, pertanto, può assumere le seguenti forme:

- `Read`:

CMD0	CMD1	PEC0	PEC1	DBL	...	DBH	PEC0	PEC1

- `Write`:

CMD0	CMD1	PEC0	PEC1	DBL	...	DBH	PEC0	PEC1

La generazione del comando viene effettuata, a livello di codice, dalla seguente funzione:

```
1  void init_cmd(uint8_t* packet, CommandCode cc, CommandMode cm,
       uint8_t addr) {
2    uint16_t cmd = (uint16_t)cc;
3    if (cm == CommandMode::BROADCAST) {
4      packet[0] = cmd >> 8;
5      packet[1] = cmd & 0xFF;
6    }
7    else {
8      packet[0] = 0b10000000 | (addr << 3) | (cmd >> 8);
9      packet[1] = cmd & 0xFF;
10   }
11   uint16_t cmd_pec = pec15_calc(CMD_LEN, packet);
12   packet[2] = cmd_pec >> 8;
13   packet[3] = cmd_pec & 0xFF;
14 }
```

All'interno della variabile `packet`, argomento della funzione, verrà salvato il comando finale.

Il primo byte del pacchetto si costruisce, come si può notare, a partire dalla destinazione: se il comando dev'essere inviato in broadcast, il codice di comando viene traslato verso destra di otto posizioni, eliminando, di conseguenza, l'ultimo byte di questa variabile e mantenendo solo i primi tre bit preceduti da cinque zeri. Se il comando è inviato ad uno slave specifico, invece, si maschera il valore binario `1000 0000` con l'indirizzo desiderato traslato verso sinistra di tre posizioni e mantenendo i primi tre bit del `CC`.

Il secondo byte del pacchetto è generato, quindi, mascherando il codice di comando, con il valore esadecimale `0xFF`; l'operazione di 'bitwise and' tra il comando e il valore binario `1111 1111`, risulta in un troncamento delle prime tre posizioni binarie, restituendo, pertanto, una variabile uguale all'ultimo byte del command code scartato precedentemente.

Per il calcolo del codice PEC, si fa uso, invece, della funzione:

```
uint16_t pec15_calc(uint8_t len, uint8_t *data) {
    uint16_t remainder, addr;
    remainder = 16;

    for (uint8_t i = 0; i<len; i++)
    {
        addr = ((remainder>>7)^data[i])&0xff;
        remainder = (remainder<<8)^pgm_read_word_near(pec_table+
    addr);
    }

    return (remainder * 2);
}
```

riportata dal datasheet.

Durante l'esecuzione della funzione `pec15_calc`, si fa uso della variabile `pec_table`, che è, semplicemente, una tabella pre-calcolata di valori necessari per il calcolo del codice di controllo. Il valore restituito da questa funzione viene aggiunto al comando ottenendo, pertanto, un comando Read o la base di un comando Write.

Per aggiungere i dati da scrivere, invece, si utilizza la funzione:

```
void init_data(uint8_t *packet, uint8_t *data, int len) {
  for (int i = 0; i < len; i++) {
    packet[i] = data[i];
  }
```

```
  uint16_t data_pec = pec15_calc(len, data);
  packet[len] = data_pec >> 8;
  packet[len + 1] = data_pec & 0xFF;
}
```

la quale computa il pacchetto da inviare e il relativo codice PEC salvando i valori nel puntatore 'packet'.

3.2.2 Configurazione

Per poter iniziare ad utilizzare ogni slave è, innanzitutto, fondamentale configurare i dispositivi in questione. Per effettuare questa operazione è necessario inviare, ad ogni periferica, il comando `WRCFGA`, abbreviazione di 'Write Configuration Register Group A', e i dati della configurazione.

Vediamo ora, nel dettaglio, come generare un comando da inviare utilizzando, a scopo d'esempio, il problema della configurazione iniziale.

Dal datasheet si trova il `CC` in binario per ogni comando che si vuole inviare; nel nostro caso, questo è, semplicemente, `000 0000 0001`. L'operazione che vogliamo svolgere è da effettuare trasversalmente in tutti gli slave e, pertanto, utilizzeremo un comando inviato in broadcast.

Otteniamo, quindi, i seguenti due byte:

CMD0	0	0	0	0	0	0	0	0
CMD1	0	0	0	0	0	0	0	1

Come sopra descritto, va, ora, calcolato il codice PEC del comando appena trovato. Il codice di controllo, calcolato mediante l'utilizzo della funzione precedente, risulta:

PEC0	0	0	1	1	1	1	0	1
PEC1	0	1	1	0	1	1	1	0

A questo punto è necessario definire la configurazione delle periferiche.

Utilizzando nuovamente il datasheet si può individuare la seguente tabella:

CFGR0	GPIO5	GPIO4	GPIO3	GPIO2	GPIO1	REFON	DTEN	ADCOPT
CFGR1	Vuv[7]	Vuv[6]	Vuv[5]	Vuv[4]	Vuv[3]	Vuv[2]	Vuv[1]	Vuv[0]
CFGR2	Vov[3]	Vov[2]	Vov[1]	Vov[0]	Vuv[11]	Vuv[10]	Vuv[9]	Vuv[8]
CFGR3	Vov[11]	Vov[10]	Vov[9]	Vov[8]	Vov[7]	Vov[6]	Vov[5]	Vov[4]
CFGR4	DCC8	DCC7	DCC6	DCC5	DCC4	DCC3	DCC2	DCC1
CFGR5	DCTO[3]	DCTO[2]	DCTO[1]	DCTO[0]	DCC12	DCC11	DCC10	DCC9

dove:

- **GPIO[x]** viene mantenuto al valore di default `1` in modo da avere il pin di Pull-Down del LTC spento;
- **REFON** impostato al valore di default `0`;
- **DTEN** per abilitare o disabilitare il timer per la scarica delle celle;
- **ADCOPT** per selezionare il rate di campionamento;
- **Vuv[X]** rappresenta la soglia di undervoltage calcolata mediante la funzione

$$Vuv_{bin}[V] = \frac{(Tensionediundervoltage) \times 10000}{16} - 1 \tag{3.1}$$

- **Vov[X]**, duale a `Vuv`, esprime la soglia di over-voltage calcolata mediante la funzione

$$Vov_{bin}[V] = \frac{(Tensionediundervoltage) \times 10000}{16} \tag{3.2}$$

- **DCC[x]**, acronimo di 'Discharge Cell X', per attivare, `1`, o disattivare, `0`, la scarica controllata della cella `X`;
- **DCTO**, infine, per impostare il valore, in minuti, del tempo di scarica delle celle.

La configurazione utilizzata dal `Race Up Team`, è, ad esempio, in esadecimale, `0xFA 0x0D 0x18 0xA4 0x00 0x00`, di cui si calcola il codice di controllo `0x50BA`.

In conclusione, si ottiene la seguente trasmissione da inviare mediante SPI agli slave:

CMD0	CMD1	PEC0	PEC1	D0	D1	D2	D3	D4	D5	PEC0	PEC1
0x00	0x01	0x3D	0x6E	0xFA	0x0D	0x18	0xA4	0x00	0x00	0x50	0xBA

Questa configurazione, come detto in precedenza, una volta ricevuta e convalidata da ogni slave, verrà salvata all'interno del registro A di ogni `LTC6811` connesso al sistema.

Capitolo 4

ADC readout

La trattazione di questo dispositivo si sofferma, in particolare, sulla misura della temperatura, parametro estremamente importante per valutare lo stato di salute complessivo del pacco batteria.

La lettura di questo valore è resa possibile mediante l'utilizzo di un termistore, in particolare del `103JT` (figura 4.1), sviluppato dalla Semitec, nota società nipponica specializzata nello sviluppo di sensori. Il dispositivo in questione, e tutti gli altri della sua specie, è in grado, modificando la propria resistenza interna, di produrre in uscita una caduta di potenziale proporzionale alla temperature dell'ambiente in cui si trova.

Il semiconduttore usato, il cui spessore è inferiore di $500\mu m$, permette di assicurare l'elevata precisione nella misurazione di cui il team ha la necessità.

Verranno utilizzati tre termistori per ogni interfaccia, pertanto, ognuno di questi, misurerà la temperatura media di tre celle. Questi dispositivi verranno connessi mediante l'utilizzo di tre pin a scopo generico (GPIO) presenti sull'interfaccia 3.2c.

Figura 4.1: 103JT

4.1 La conversione ADC

All'interno di ogni `LTC6811` sono presenti due ADC che operano contemporaneamente durante la misurazione dei parametri delle celle connesse allo slave.

Utilizzando il bit `ADCOPT`, specificato nella configurazione iniziale del registro 'A', e i bit di selezione della modalità `MD`, si possono definire ben otto modalità di conversione che differiscono tra loro per il rapporto di campionamento.

Si può quindi scegliere tra:

- **27 kHz**, modalità *veloce*. La frequenza a cui opera impone una conversione analogico-digitale con 10 bit di precisione e, viene, dunque, introdotto più errore nella misura media;
- **14 kHz**;
- **7 kHz**. A questa velocità si ottiene un'alta risoluzione, 14 bit, per un tempo di misurazione totale accettabile ed è, pertanto, considerata la modalità *normale*;
- **3 kHz**;
- **2 kHz**;
- **1 kHz**;
- **422 Hz**;
- **26 Hz**.

La scelta della velocità è molto importante ed è influenzata principalmente dall'applicazione che si vuole realizzare del dispositivo. Ad esempio, se la priorità assoluta risulta essere la precisione, sarà necessario selezionare un "rate" molto lento, in modo da ottenere, a 26Hz, una precisione di 16 bit con errore totale minore di $\pm 50 \mu V_{PP}$.

L'opzione migliore, tuttavia, per lo scopo del team, è utilizzare la modalità veloce in quanto il regolamento stabilisce dei limiti temporali che non possono essere trascurati, nonostante si introduca un errore medio relativamente notevole sulle misure. I messaggi provenienti dall'AMS, considerati segnali critici (`EV 5.8.10`), devono rispettare, come già anticipato in precedenza, la regola `EV 5.8.10` che impone l'intervallo massimo consentito per questa tipologia di pacchetti.

L'accuratezza dei dati che si ottiene risulta essere, tuttavia, molto soddisfacente: il rumore massimo che comporta l'utilizzo del rate a 27 kHz è $\pm 4mV_{PP}$.

Per leggere la temperatura è necessario misurare la caduta di potenziale che si verifica sui pin "general purpose" a cui sono connessi i termistori sopra citati. Tale operazione è possibile mediante il comando `ADAX`, il cui codice di comando `CC` è:

1	0	MD[1]	MD[0]	1	1	0	0	CHG[2]	CHG[1]	CHG[0]

Impostando il bit `ADCOPT` a zero e i due valori binari di `MD` a `10`, si ottiene, per quanto precedentemente detto, una conversione analogico-digitale a 27 kHz.

I tre bit `CHG[2:0]` devono essere impostati in base a quali GPIO si vogliono utilizzare durante la conversione. Nel sistema progettato dal `Race Up Team`, questa terna di valori viene impostata a `000` in modo da ottenere la conversione di tutti i cinque GPIO. È stato sperimentalmente verificato che l'invio di tre comandi consecutivi per ciascuno dei termistori necessita, complessivamente, più tempo rispetto a richiedere la conversione tramite unica esecuzione. L'attesa necessaria per il completamento di questa operazione, al rate scelto, è, dunque, di $1.1ms$.

L'invio del comando al LTC6811 è compito della funzione:

```
void adax() {
  constexpr uint8_t packet_len = CMD_LEN + PEC_LEN;
  uint8_t packet[packet_len] = {};
  init_cmd(packet, CommandCode::ADAX, CommandMode::BROADCAST);
  wakeup_idle();
  tx(packet, packet_len);
}
```

Questa funzione inizializza il comando da inviare e, analogamente al problema della configurazione, utilizzando `tx`, lo invia mediante l'interfaccia SPI.

Il comando `ADAX`, tuttavia, non restituisce alcun valore misurato dal dispositivo; questi, infatti, vengono memorizzati negli `Auxiliary Register Group` che andranno, pertanto, letti per poter consultare i dati raccolti.

4.2 La lettura dei registri

Come si evince dalla tabella 45 del datasheet, il registro che contiene i valori misurati per i primi tre pin, l'`Auxiliary Register Group A`, ha la seguente forma:

AVAR0	G1V[7]	G1V[6]	G1V[5]	G1V[4]	G1V[3]	G1V[2]	G1V[1]	G1V[0]
AVAR1	G1V[15]	G1V[14]	G1V[13]	G1V[12]	G1V[11]	G1V[10]	G1V[9]	G1V[8]
AVAR2	G2V[7]	G2V[6]	G2V[5]	G2V[4]	G2V[3]	G2V[2]	G2V[1]	G2V[0]
AVAR3	G2V[15]	G2V[14]	G2V[13]	G2V[12]	G2V[11]	G2V[10]	G2V[9]	G2V[8]
AVAR4	G3V[7]	G3V[6]	G3V[5]	G3V[4]	G3V[3]	G3V[2]	G3V[1]	G3V[0]
AVAR5	G3V[15]	G3V[14]	G3V[13]	G3V[12]	G3V[11]	G3V[10]	G3V[9]	G3V[8]

Come precedentemente evidenziato, e come conferma la tabella riportata, la temperatura viene immagazzinata come differenza di potenziale tra i GPIO presenti sulla scheda: `G[X]V`, infatti, rappresenta il valore della tensione misurata sul pin `G[X]`.

I pin general purpose dell'interfaccia a cui sono connessi i termistori sono GPIO[1], GPIO[2] e GPIO[3].

Nel codice viene definita la seguente funzione che gestisce l'operazione di lettura dei registri della temperatura:

```
void read_temps() {
  for (int i = 0; i < SLAVE_NUM; i++) {
    for (char reg = 'A'; reg <= 'B'; reg++) {
      uint8_t raw_temps[GREG_LEN] = {};
      if (rdaux(g_bms.slaves[i].addr, reg, raw_temps) == 0) {
        delay(MEAS_DELAY);
        save_temps(i, reg, raw_temps);
        g_bms.slaves[i].err = false;
      }
      else {
        g_bms.slaves[i].err = true;
      }
    }
  }
}
```

Questa funzione, dopo aver pre-allocato lo spazio necessario all'array dove salvare i valori grezzi `raw_temps[GREG_LEN]`, esegue, consecutivamente, per ogni slave presente nel Battery Management System e per ogni registro contenente le temperature, due semplici funzioni:

- `rdaux(...)`, per leggere i valori convertiti dall'ADC;
- `save_temps(...)`, per immagazzinare i dati letti.

Inoltre, il metodo `read_temps` ha l'onere di impostare il flag `err` qualora venisse rilevato un errore durante la comunicazione con lo slave i-esimo.

4.2.1 Campionamento dei dati

La prima funzione `rdaux` è definita, nel file di libreria `.h`, come:

```
int rdaux(uint8_t addr, char reg, uint8_t *gpio_buf)
```

La sua implementazione è:

```
int rdaux(uint8_t addr, char reg, uint8_t *gpio_buf) {
  uint8_t packet[CMD_LEN + PEC_LEN + GREG_LEN + PEC_LEN] = {};
  CommandCode cc = CommandCode::RDAUXA;
  switch (reg) {
  case 'A': cc = CommandCode::RDAUXA;
    break;
  case 'B': cc = CommandCode::RDAUXB;
    break;
  }
  init_cmd(packet, cc, CommandMode::ADDRESSED, addr);
  wakeup_idle();
  txrx(packet, CMD_LEN + PEC_LEN, &(packet[CMD_LEN + PEC_LEN]),
   GREG_LEN + PEC_LEN);
  uint16_t rec_pec = (packet[CMD_LEN + PEC_LEN + GREG_LEN] << 8)
   | (packet[CMD_LEN + PEC_LEN + GREG_LEN + 1] & 0xFF);
  if (rec_pec == pec15_calc(GREG_LEN, &(packet[CMD_LEN + PEC_LEN
   ]))) {
    for (int i = 0; i < GREG_LEN; i++)
      gpio_buf[i] = packet[CMD_LEN + PEC_LEN + i];
    return 0;
  }
  return 1;
}
```

In questa funzione, il pacchetto da inviare è generato in modo differente dal problema della configurazione iniziale.

Il comando in analisi, infatti, non può essere inviato in broadcast altrimenti, nella fase successiva, si verificherebbero delle collisioni nel bus dell'isoSpi con la dannosa conseguenza della perdita dei pacchetti. Viene, quindi, come si può notare dall'intestazione della funzione, specificato l'indirizzo dello slave `addr` a cui si vuole inviare il messaggio.

Dopo aver "svegliato" il dispositivo con `wakeup_idle`, si utilizza la funzione `txrx` che, dopo aver inviato il comando, comincia il campionamento della trasmissione in entrata memorizzandola nella variabile `packet`.

Il metodo `txrx` è, quindi, così definita:

```
void txrx(uint8_t* tx_data, int tx_bytes, uint8_t* rx_data, int
    rx_bytes) {
  digitalWrite(SPI_CS_PIN, LOW);
  for (int i = 0; i < tx_bytes; i++) {
    SPI.transfer(tx_data[i]);
  }
  for (int i = 0; i < rx_bytes; i++) {
    rx_data[i] = SPI.transfer(0xFF);
  }
  digitalWrite(SPI_CS_PIN, HIGH);
}
```

Infine, `rdaux`:

- verifica la correttezza del pacchetto ispezionando il codice di controllo PEC del pacchetto ricevuto;
- memorizza i valori ricevuti all'interno del puntatore `gpio_buf`;
- ritorna il valore `0` se il codice di controllo è corretto, altrimenti `1`.

4.2.2 Memorizzazione delle misure

Per memorizzare i dati viene, infine, chiamata la funzione `save_temps`, definita come:

```
void save_temps(int slave_idx, char reg, uint8_t* raw_temps) {
  for (int i = 0; i < GREG_LEN - 2 ; i += 2) {
    uint16_t volt = (raw_temps[i + 1] << 8) | (raw_temps[i] & 0
   xFF);
    uint16_t temp = parse_temp(volt);
    g_bms.slaves[slave_idx].temps[i / 2] = temp;
    if (temp > g_bms.max_temp) {
      g_bms.max_temp = temp;
      g_bms.max_temp_slave = slave_idx;
    }
    if (temp < g_bms.min_temp) g_bms.min_temp = temp;
    g_bms.tot_temp += temp;
  }
}
```

Quest'ultimo metodo, dati in input i valori "grezzi" delle temperature, espressi come cadute di potenziale presenti ai GPIO, converte e salva, all'interno della variabile `slaves` contenuta nella struttura `g_bms` definita precedentemente, i valori corretti.

Durante le operazioni effettuate viene, inoltre, individuato il valore massimo e minimo della temperatura e calcolando la temperatura media del pacco batteria.

4.3 La regressione

Il datasheet del termistore 103JT [14] non presenta alcuna formula che possa effettuare la conversione tensione-temperatura di cui necessita il team, bensì viene riportata esclusivamente la seguente tabella:

°C	R [$k\Omega$]
-50	367.7
-40	204.7
-30	118.5
-20	71.02
-10	43.67
0	27.7
10	18.07
20	12.11
25	10
30	8.301
40	5.811

°C	R [$k\Omega$]
50	4.147
60	3.011
70	2.224
80	1.668
85	1.451
90	1.267
100	0.9753
110	0.7597
120	0.5981
125	0.5331

La formula che converte i valori letti dai registri deve, quindi, essere estratta a partire unicamente dai valori sopra riportati. Bisogna, dunque, risolvere un problema di regressione polinomiale scegliendo il grado massimo della funzione e, in seguito, stimando i coefficienti.

4.3.1 Estrazione della funzione

Per estrarre la funzione, il primo passo è comprendere la disposizione dei punti.

Il modello di regressione da utilizzare, osservando il grafico riportato in figura 4.2, deve essere della forma:

$$y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \beta_m x_i^m + \varepsilon_i \tag{4.1}$$

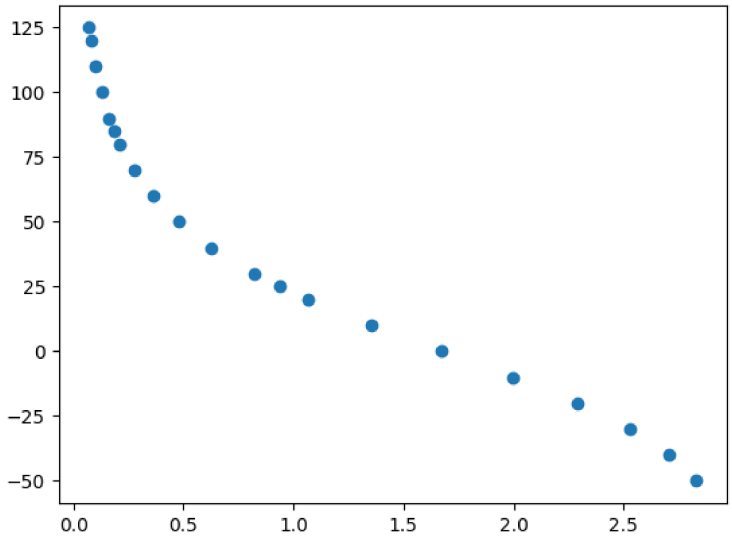


Figura 4.2: Disposizione dei punti

che, in forma matriciale, si può riscrivere come

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^m \\ 1 & x_2 & x_2^2 & \cdots & x_2^m \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^m \end{bmatrix} + \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_m \end{bmatrix} + \begin{bmatrix} \varepsilon_0 \\ \varepsilon_1 \\ \vdots \\ \varepsilon_m \end{bmatrix} \tag{4.2}$$

o, più semplicemente:

$$\overrightarrow{y} = \mathbf{X}\overrightarrow{\beta} + \overrightarrow{\varepsilon} \tag{4.3}$$

dove $\overrightarrow{y}$ rappresenta il vettore dei punti che si vogliono regredire a funzione, $\overrightarrow{\beta}$ il vettore dei coefficienti e, infine, ε costituisce l'errore che si introduce.

Una volta scelto il grado m da utilizzare, si ottengono i coefficienti $\overrightarrow{\beta}$ risolvendo

$$\widehat{\overrightarrow{\beta}} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\overrightarrow{y} \tag{4.4}$$

4.3.2 Python

Per semplificare i calcoli ed ottenere una precisione molto elevata, si è fatto uso di `Jupyter Notebook` [15] e delle librerie:

- **NumPy** [16], utile per semplificare le operazioni tra array;
- **Pandas** [17], per manipolare i dati;
- **matplotlib** [18], necessaria per poter visualizzare i grafici

I dati del datasheet, convertiti in un file `CSV`, vengono importati nell'ambiente di lavoro grazie al seguente codice:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv('termistori.csv', sep=";")
df = df.astype({'T': 'float'})
df = df.astype({'R': 'float'})
```

Si ottiene, dunque, una semplice tabella che contiene tutti i dati necessari allo scopo. È utile, nonostante la perdita di precisione, troncare i dati importati ai valori di temperature esclusivamente positive, in quanto si suppone che il BMS spenga la vettura in caso di temperature troppo basse.

Per semplificare i calcoli, è vantaggioso generare una nuova colonna contenente le cadute di potenziale che si possono ottenere. Questi valori sono frutto del partitore di tensione che si genera tra il termistore e una resistenza del valore di $22k\Omega$:

$$V_{GPIO} = 3V \frac{R}{22000\Omega \times R} \tag{4.5}$$

dove `R` è il valore della resistenza riportato dalla tabella.

Si esegue, pertanto, il codice:

```
df["voltage"] = 3 * df["R"] / (22000 + df["R"])
df = df[df['T'] >= 0]
df
```

Dalle manipolazioni dei dati effettuate precedentemente, si è ottenuto il grafico tensione-temperatura a figura 4.3.

Si procede, pertanto, alla ricerca della funzione che meglio approssima questi punti.

Osservando come si dispongono i punti forniti dal datasheet, si nota che questi si dispongono lungo una curva che ricorda una funzione polinomiale di grado dispari.

Si trova, inoltre, empiricamente, che utilizzando una regressione di quinto grado la complessità totale della funzione di conversione rende i tempi d'esecuzione totale del codice di gran lunga superiori rispetto ai `500 ms` previsti dal regolamento. Escludendo, quindi, il primo grado a priori, la scelta della curva ricade su una funzione cubica. Questa, anche qualora non fosse una scelta "obbligata", sarebbe comunque l'opzione migliore: nonostante si azzeri l'errore aumentando il

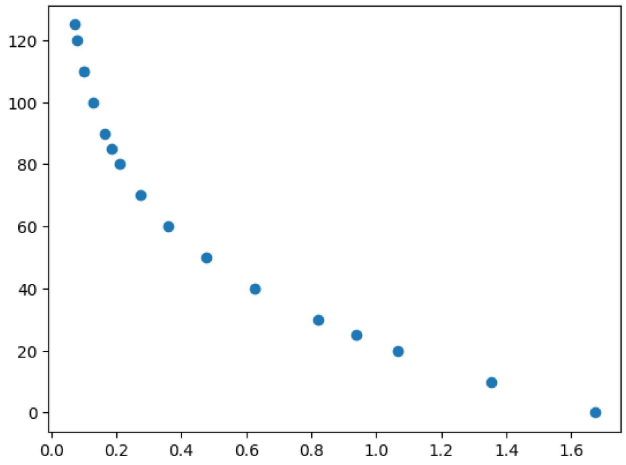


Figura 4.3: Tensione - temperatura

grado totale, si amplifica la sensibilità ai dati in ingresso rischiando, pertanto, di generare delle informazioni completamente errate.

Si vuole, tuttavia, aumentare la precisione su alcuni punti rispetto ad altri: la temperatura è ottimale se rimane confinata nell'intervallo $[20, 60]°C$, e, pertanto, deve essere definito un vettore dei pesi, normalizzato in $[0, 1]$, che sia in grado di soddisfare questa richiesta.

Specificando, quindi, il vettore dei pesi `w` come:

```
w = np.where(
    (df["T"] >= 20) & (df["T"] <= 60),
    1,
    np.where(
        df["T"] <= 20,
        (1 - abs((df["T"] - 20)/20)),
        (1 - abs((df["T"] - 60)/60))
    )
)
```

si ottiene un vettore della stessa dimensione della colonna `T` dove, se il punto appartiene all'intervallo desiderato il suo peso è `1`, viceversa questo diminuisce con l'aumentare della distanza.

La funzione

```
fit = np.polyfit(df["voltage"], df["T"], 3, w = w)
```

restituirà, quindi, i coefficienti della funzione che avrà forma:

$$V(T) = -67.16T^3 + 199.73T^2 - 230.70T + 121.38 \tag{4.6}$$

La figura 4.4, mostra come la funzione di regressione, ottenuta con le considerazioni fin qui adottate, venga tracciata molto vicino ai punti dell'intervallo prestabilito, evidenziando che la precisione in un intorno di quei punti sarà molto elevata.

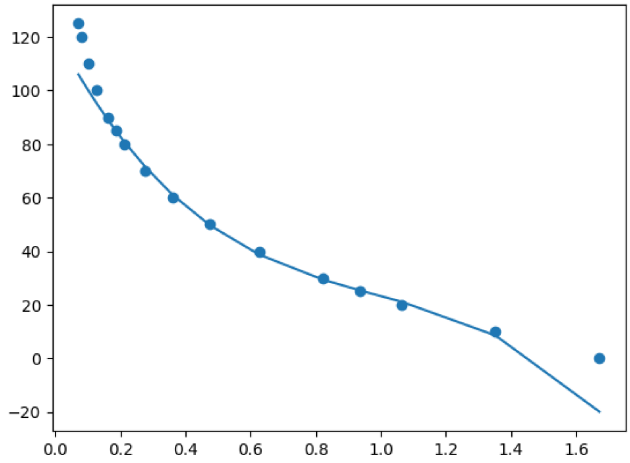


Figura 4.4: Funzione 4.6

4.3.3 Conversione

Per quanto sopra discusso, la funzione `parse_temp`, citata nei codici precedenti, avrà la seguente specifica:

```
constexpr float TEMP_FIT_COEFF[4] = {
  -67.16444614,     // x^3
  199.72749311,     // x^2
  -230.70073783,    // x^1
  121.37693535      // x^0
};

uint16_t parse_temp(uint16_t volt) {
  float temp = volt / 10000.0;
  return (uint16_t) (
    TEMP_FIT_COEFF[0] * pow(temp, 3) +
    TEMP_FIT_COEFF[1] * pow(temp, 2) +
    TEMP_FIT_COEFF[2] * temp +
    TEMP_FIT_COEFF[3]
  );
}
```

4.4 La ventola

Nel BMS host viene mantenuto inalterato il funzionamento di un pin analogico di Arduino a cui è connessa una ventola. Questa specifica non è necessaria, tuttavia implementando questo piccolo sistema di raffreddamento si permette alla vettura di aumentare l'efficienza energetica riducendo il rischio di raggiungere temperature al di fuori del range ottimale.

Il pin analogico `A4` della board Arduino Due viene, pertanto, utilizzato per generare un'onda quadra che possa gestire il duty cycle della ventola connessa. In base ai valori massimo e minimo della temperatura misurata si genera un Pulse Width Modulation (PWM) adeguato.

```
1  void set_fan_dutycycle() {
2    float m = (MAX_FAN_SPEED - MIN_FAN_SPEED) / (MAX_TEMP_FAN -
       MIN_TEMP_FAN);
3    float q = MAX_FAN_SPEED - (MAX_TEMP_FAN * m);
4    float fan_speed = 0.0;
5    if (g_bms.max_temp > MAX_TEMP_FAN) {
6      analogWrite(FAN_EN_PIN, 255 * MAX_FAN_SPEED);
7      g_bms.prev_max_temp = g_bms.max_temp;
8      return;
9    }
10   if (g_bms.max_temp < MIN_TEMP_FAN) {
11     analogWrite(FAN_EN_PIN, 255 * MIN_FAN_SPEED);
12     g_bms.prev_max_temp = g_bms.max_temp;
13     return;
14   }
15   if ((g_bms.max_temp - g_bms.prev_max_temp) > VAR_MIN || (g_bms.
       max_temp - g_bms.prev_max_temp) < -VAR_MIN) {
16     fan_speed = m * g_bms.max_temp + q;
17     analogWrite(FAN_EN_PIN, 255 * fan_speed);
18     g_bms.prev_max_temp = g_bms.max_temp;
19   }
20 }
```

Questa funzione, dopo aver recuperato il massimo valore della temperatura dalla variabile `g_bms`, ha l'onere di calcolare il nuovo duty cycle da impostare.

Il funzionamento è molto semplice e si possono verificare tre situazioni. La temperatura infatti può essere:

- maggiore della soglia `MAX_TEMP_FAN`. In questo caso, il duty cycle verrà impostato al 100%;

- minore di `MIN_TEMP_FAN`. La velocità della ventola sarà molto bassa, ma non zero.
- compresa tra i due livelli indicati. In questa situazione, il tempo di lavoro della ventola sarà aggiornato solo se si osserva una variazione minima tra la temperatura massima precedente e l'attuale di almeno `VAR_MIN`.

Così facendo, la velocità non viene costantemente aggiornata e si riduce il rischio di danneggiare il dispositivo.

Utilizzando, infine, la funzione `analogWrite` di Arduino [19], si genera, mediante il pin scelto, un'onda rettangolare costante con duty cycle specificato fino alla successiva chiamata di `set_fan_dutycycle`.

Capitolo 5

Conclusioni

LTC6811 è un monitor di pile di batterie multi celle che, connesso in daisy chain o mediante l'utilizzo di indirizzi, può essere utilizzato per controllare interi pacchi batterie.

L'implementazione di un Battery Management System è uno dei casi d'uso possibili più importanti per LTC6811; infatti, come discusso nei capitoli precedenti, il nuovo AMS del team di corse dell'Università degli Studi di Padova è uno dei cardini per il prototipo della stagione 2022-2023 ed utilizza proprio questo prodotto della Analog Devices.

Questo dispositivo non si limita a misurare la caduta di potenziale delle celle o la sola temperatura in un pacco batteria, ma porta con se numerose altre funzioni che non sono state trattate in questo lavoro.

La facilità d'utilizzo, le numerose celle collegabili, la possibilità di essere utilizzato in serie o in parallelo, il relativo costo contenuto per singola unità, la precisione e la velocità nelle misurazioni sono solo alcuni punti di forza del `LTC6811` che vede, nell'avvento della mobilità elettrica e nel suo dinamico e continuo sviluppo del mercato, la possibilità di diventare uno dei controllori più utilizzati nel settore dell'automotive.

(a) Tesla Model 3 **(b)** BMW i3

Figura 5.1: Vetture elettriche che utilizzano LTC6811

Non è un caso, infatti, che la nota casa automobilistica statunitense Tesla [20] abbia utilizzato, nella sua versione più economica e più comune, la "Model 3" [21], figura5.1a, una versione custom del "fratello" del `LTC6811`, `LTC6812` [22], per la realizzazione del sistema di controllo.

Anche la tedesca BMW [23], figura5.1b, nella sua versione elettrica "i3" [24], ha optato per l'utilizzo del dispositivo trattato per monitorare la batteria implementando, peraltro, in collaborazione con LION Smart [25], un sistema wireless SmartMesh, rendendo il BMS un dispositivo dell'*Internet of Things*.

La casa bavarese è stata in grado, inoltre, di ottimizzare la durata totale di viaggio utilizzando la funzione di bilanciamento delle celle del `LTC6811`.

5.1 AURIX

Infineon Technologies è una società fondata nel aprile 1999 a Neubiberg, in Germania. Questa compagnia, per capitalizzazione, fa parte delle prime 20 produttrici al mondo di semiconduttori ed è uno degli sponsor principali di *Race Up*.

Molto presente anche nel mercato automobilistico ha recentemente proposto un nuovo prodotto mirato per questo settore: l'AURIX, figura 5.2, acronimo per "Automotive Realtime Integrated Next Generation Architecture", una famiglia di microcontrollori basata su un'architettura multiprocessore di tre CPU a 32 bit.

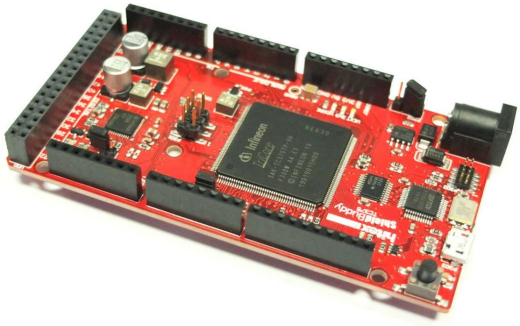


Figura 5.2: AURIX

Questa scheda, molto simile ad un Arduino Due, è straordinariamente potente. Le sue principali funzionalità sono:

- tre processori fino a 300MHz ciascuno;
- 1.8 miliardi di operazioni in virgola mobile al secondo;

- acceleratore per trasformate di Fourier.

Questo dispositivo, in collaborazione con LTC6811 o LTC6812 è, attualmente, il connubio perfetto per sviluppare un sistema di gestione delle batteria altamente avanzato ed affidabile. Non è escluso, infatti, che AURIX venga utilizzato nel prototipo SG-07 2023-2024 del *Race Up Team* per portare il BMS a un livello superiore. L'implementazione di questo microcontrollore, infatti, garantirebbe al team tempi di elaborazione dei dati ridotti di circa il 72%.

Bibliografia

[1] *Rapporto SDGs 2022*, pag. 89, https://www.istat.it/storage/rapporti-tematici/sdgs/2022/Rapporto-SDGs-2022.pdf

[2] Wang, S., Fan, Y., Stroe, D.-I., Fernandez, C., Yu, C., Cao, W., & Chen, Z. (2021). *Chapter 1 - Lithium-ion battery characteristics and applications*. In S. Wang, Y. Fan, D.-I. Stroe, C. Fernandez, C. Yu, W. Cao, & Z. Chen (Eds.), Battery System Modeling (pp. 1–46). Elsevier. https://doi.org/https://doi.org/10.1016/B978-0-323-90472-8.00003-2

[3] Y. Ji, Y. Zhang, C.Y. Wang, J. *Electrochem.* Soc. 160 (2013) A636–A649. https://pubs.acs.org/doi/10.1021/cm901452z

[4] A. Pesaran, S. Santhanagopalan, G.H. Kim, *Addressing the Impact of Temperature Extremes on Large Format Li-Ion Batteries for Vehicle Applications*, Presented at: Proceedings of the 30th International Battery Seminar, Ft. Lauderdale, Florida, 2013. https://www.nrel.gov/docs/fy13osti/58145.pdf

[5] Ma, S., Jiang, M., Tao, P., Song, C., Wu, J., Wang, J., Deng, T., & Shang, W. (2018). *Temperature effect and thermal impact in lithium-ion batteries: A review*. Progress in Natural Science: Materials International, 28(6), 653–666. https://doi.org/https://doi.org/10.1016/j.pnsc.2018.11.002

[6] Nagasubramanian, G. *Electrical characteristics of 18650 Li-ion cells at low temperatures*. Journal of Applied Electrochemistry 31, 99–104 (2001). https://doi.org/10.1023/A:1004113825283

[7] Lin, X., Khosravinia, K., Hu, X., Li, J., & Lu, W. (2021). *Lithium Plating Mechanism, Detection, and Mitigation in Lithium-Ion Batteries*. Progress in Energy and Combustion Science, 87, 100953. https://doi.org/https://doi.org/10.1016/j.pecs.2021.100953

[8] Ma, S., Jiang, M., Tao, P., Song, C., Wu, J., Wang, J., Deng, T., & Shang, W. (2018). *Temperature effect and thermal impact in lithium-ion batteries: A review*. Progress in Natural Science: Materials International, 28(6), 653–666. https://doi.org/https://doi.org/10.1016/j.pnsc.2018.11.002

[9] Nazari, A., & Farhad, S. (2017). *Heat generation in lithium-ion batteries with different nominal capacities and chemistries*. Applied Thermal Engineering, 125, 1501–1517. https://doi.org/https://doi.org/10.1016/j.applthermaleng.2017.07.126

[10] *Formula Student Rules 2023*, https://www.formulastudent.de/fileadmin/user_upload/all/2023/rules/FS-Rules_2023_v1.0.pdf.

[11] Arduino Due, https://store.arduino.cc/products/arduino-due

[12] Analog Devices, *LTC6820 datasheet REV. C*, https://www.analog.com/media/en/technical-documentation/data-sheets/LTC6820.pdf.

[13] Analog Devices, *LTC6811-2 datasheet REV. C*, https://www.analog.com/media/en/technical-documentation/data-sheets/LTC6811-1-6811-2.pdf.

[14] Semitec, *JT Thermistor datasheet*, https://www.mouser.it/datasheet/2/362/P9-JT-Thermistor-1621687.pdf.

[15] Jupyter Notebook, https://jupyter.org/

[16] NumPy, https://numpy.org/

[17] Pandas, https://pandas.pydata.org/

[18] matplotlib, https://matplotlib.org/

[19] analogWrite(), https://www.arduino.cc/reference/en/language/functions/analog-io/analogwrite/

[20] Tesla, https://www.tesla.com/

[21] Tesla Model 3, https://www.tesla.com/it_it/model3

[22] Analog Devices, *LTC6812-1 datasheet*, https://www.analog.com/media/en/technical-documentation/data-sheets/ltc6812-1.pdf

[23] BMW, https://www.bmw.it

[24] BMW i3, https://www.bmw.it/it/gamma/bmw-i/i3/2021/panoramica-bmw-i3.html

[25] LION Smart, https://lionsmart.com/en/en-lion-smart-gmbh/