UNIVERSITA' POLITECNICA DELLE MARCHE

FACOLTA' DI INGEGNERIA

Corso di Laurea triennale in Ingegneria Elettronica

Sviluppo di firmware per il controllo dell'alimentazione di sensori wireless indossabili

Firmware development for power control of wearable wireless sensors

Relatore:

Giorgio Biagetti

Tesi di Laurea di:

Leonardo Falcioni

A.A. 2019/2020

Indice

1. Introduzione

La finalità di questa tesi è fornire un sistema di alimentazione più efficiente e con più funzionalità al seguente dispositivo: https://www.mdpi.com/2079-9292/9/6/934. Quest'ultimo è un sensore inerziale dotato di EMG, ECG wireless basato sul Bluetooth-low-energy (BLE) che può essere utilizzato per monitorare l'attività di ogni giorno del paziente in differenti possibili situazioni. Il progetto del sensore wireless prevede che l'alimentazione sia fornita da un caricabatterie associato ad alcuni interruttori e regolatori lineari per fornire potenza ai differenti sottosistemi solo quando ci sia effettivamente necessità, in modo da migliorare l'efficienza energetica.[1] L'adozione di sensori wireless indossabili in ambito medico e ospedaliero comporta numerosi ed evidenti vantaggi; attraverso sensori wireless è possibile offrire un monitoraggio continuo della mobilità e delle funzioni vitali del paziente, senza ostacolarne le attività quotidiane. Attualmente infatti, il metodo di monitoraggio più diffuso necessita l'impiego di cavi e connessioni tra paziente e macchinario. In secondo luogo, l'utilizzo di suddetti dispositivi permette di avere a disposizione istantaneamente dati riguardo la condizione di salute del paziente; questi ultimi possono essere elaborati per giungere ad una diagnosi e ad una pronta risposta da parte del personale sanitario. Inoltre, una ulteriore conseguenza positiva riguarderebbe la minimizzazione del tempo e della pressione generale sul sistema sanitario; infatti, il monitoraggio costante e continuo del paziente non presuppone l'ospedalizzazione dell'individuo nel caso di utilizzo di sensori wireless indossabili.[2] Nonostante i potenziali vantaggi e benefici enunciati precedentemente, esistono importanti problematiche da affrontare inerenti all'utilizzo di questi dispositivi. Di fatti, sono identificabili diverse caratteristiche che condividono pressoché la totalità delle reti di sensori medicali: l'indossabilità delle piattaforme (inficiata dalle dimensioni delle batterie e delle antenne), comunicazione affidabile, trasmissione dati a molteplici ricevitori (costituiti da più individui del personale medico), mobilità del dispositivo e sicurezza delle trasmissioni wireless. In particolare, assume rilevanza la gestione del consumo energetico e dell'alimentazione per rendere il sistema il più efficiente possibile sotto questo punto di vista. [3]

1.1 Obbiettivo

Come accennato precedentemente, l'obbiettivo che si pone questa tesi consiste nello sviluppo di un firmware per garantire la gestione e la configurazione del dispositivo di alimentazione del sensore wireless indossabile. In particolare, si vogliono apportare migliorie al sistema, configurando opportunamente il caricabatterie programmabile via IIC in modo da aumentarne la versatilità, garantendo il funzionamento del sensore anche se alimentato via USB.

1.2 Hardware impiegato

Per la gestione del caricabatterie BQ25155 della Texas Instruments (https://www.ti.com/product/BQ25155), configurabile tramite la periferica IIC, si è optato per il microcontrollore nRF52840 Dongle della Nordic Semiconductor (https://www.nordicsemi.com/Software-and-tools/Development-Kits/nRF52840-Dongle); quest'ultimo offre la connettività wireless necessaria insieme con una connessione USB utile in fase di programmazione e per comunicare con il dispositivo.

1.3 Software impiegato

Sistema operativo: *Windows 10*;
IDE: *Eclipse*;
Emulatore per porte seriali: *PuTTY*;
Applicazione scanner Bluetooth: *BLE scanner*;
Cross-platform development software for Bluetooth Low Energy: *nRF Connect Programmer*.

2. Le Batterie agli ioni di litio

2.1 Panoramica

L'alimentazione del progetto viene fornita da un caricatore di batterie al litio (nello specifico, si tratta di batterie al litio che utilizzano un elettrolita polimerico e che verranno indicate con Li-Poly); di seguito viene descritto brevemente il funzionamento delle batterie agli ioni di litio e sono riportate le fasi che caratterizzano il processo di carica delle batterie.

2.2 Struttura fisica

Le batterie agli ioni di litio (Li-ion) sono costituite da celle in cui gli elettrodi positivo e negativo sono rappresentati da composti del litio che costituiscono materiali elettrochimicamente attivi. Durante il ciclo di carica/scarica della batteria, gli ioni di litio (Li^+) si scambiano tra suddetti elettrodi. Il materiale che va a costituire l'elettrodo positivo (catodo) è rappresentato da un ossido di metallo, caratterizzato da una struttura a strati; ad esempio, si tratta di materiali basati tipicamente sull'ossido di cobalto ($LiCoO_2$) o sull'ossido di maganese ($LiMn_2O_4$) sotto i quali si trova un foglio di alluminio (che rappresenta il collettore di corrente). L'elettrodo negativo (anodo) è generalmente costituito da grafite (anch'esso un materiale con struttura stratificata), posta sopra ad un foglio di rame. Durante il processo di carica/scarica, gli ioni di litio sono estratti o iniettati attraverso lo spazio tra uno strato e l'altro dei materiali. Gli elettrodi sono elettricamente isolati da un film di separazione di polipropilene o polietilene (in prodotti che utilizzano un elettrolita liquido), da uno strato di gel elettrolita polimerico (nel caso di batterie Li-Poly) o da uno strato di elettrolita solido (per batterie allo stato solido). In particolare, le batterie Li-Poly si contraddistinguono dalle batterie convenzionali per una struttura fisica leggermente differente: oltre alla pellicola di separazione interposta tra gli elettrodi che è sostituita da uno strato di PVDF-HFP (fluoride–hexafluroropropylene) o di un altro polimero impregnato con un elettrolita liquido, il foglio metallico collocato al di sotto dei materiali stratificati è tipicamente sostituito da una griglia metallica. Nelle celle gel-polimeriche, gli strati positivo, negativo e di separazione sono quindi legati dal polimero (generalmente il PVDF-HFP) andando a costituire un singolo dispositivo monolitico. [4]

2.3 Vantaggi delle batterie al litio

Il litio è il più leggero tra i metalli, ha un elevato potenziale elettrochimico e fornisce la più alta energia specifica per unità di peso. Conseguentemente, l'elevata energia specifica (150 Wh/kg) e la densità di energia (400 Wh/l) dei prodotti commerciali li rende adatti, sia per peso che per volume, a diverse applicazioni. La maggior parte di tali batterie offrono un basso tasso di auto-scarica (dal 2% all'8% al mese), un numero abbastanza elevato di cicli di carica/scarica (superiore a 1000 cicli) e un ampio range di temperatura di funzionamento (per la carica da -20°C a 60°C, per la scarica da -40°C a 65°C). Tuttavia, le batterie agli ioni di litio presentano lo svantaggio di necessitare di circuiti di gestione o dispositivi di disconnessione meccanica per fornire protezione da condizioni di over-discharge, sovraccarico o condizioni di temperatura non corrispondenti al normale range di operatività. Grandezze di merito che vengono introdotte per paragonare le caratteristiche delle batterie sono la capacità teorica (Theoretical o Coulombic capacity), la tensione teorica (Theoretical Voltage) e l'energia teorica. La capacità teorica è essenzialmente determinata dalla quantità di materiale elettrochimicamente attivo presente nella cella della batteria. È espressa come la quantità totale di elettricità che viene coinvolta nella reazione elettrochimica ed è definita in Coulomb (C) o Ampere-ore (Ah). Il valore della capacità teorica relativa alla singola cella è calcolato dal peso equivalente dei reagenti. In generale, una batteria al litio/ossido di cobalto e una batteria al litio/ossido di manganese presentano rispettivamente i seguenti valori di capacità su unità di massa: circa 286 mAh/g e 100 mAh/g. Per quanto concerne la tensione teorica, quest'ultima è determinata dal tipo di materiale contenuto nella singola cella. Può essere calcolata (a partire dai valori del potenziale di ossidazione o riduzione degli elementi) o ottenuta sperimentalmente:

anodo (potenziale di ossidazione) + catodo (potenziale di riduzione) = potenziale singola cella;

Infine, l'energia teorica è pari alla massima energia che la batteria è in grado di fornire:

$$Wh = V \cdot Ah$$

Per confrontare la longevità delle batterie viene definita la CE (Coulombic Efficiency) della singola cella della batteria agli ioni di litio:

CE = [carica fornita durante la scarica] / [carica accumulata durante la carica precedente];

fondamentalmente, se sono assenti fenomeni indesiderati quali reazioni parassite degli elettroliti o degradazioni meccaniche, il valore della CE sarà esattamente pari all'unità; conseguentemente, la longevità della cella (in specifiche condizioni di test) sarà maggiore più ci si avvicina all'unità.[7] L'elevato valore della CE delle celle di batterie agli ioni di litio (in genere pari a 99,9%) è riconducibile all'assenza di effetti parassiti quali la formazione di gas (comune in sistemi acquosi). I principali vantaggi e svantaggi delle batterie agli ioni di litio, rispetto ad altri tipi di batterie, sono riassunti di seguito; [4][5] per quanto concerne i vantaggi:

- Le celle non richiedono manutenzione;
- Lunghi cicli di vita;
- Operatività per un ampio range di temperatura;
- Lunga durata di conservazione;
- Basso tasso di self-discharge (meno della metà di quello di batterie basate sul nichel);
- Capacità di scarica ad elevata potenza;
- Elevata CE (Coulombic Efficiency);
- Alta energia specifica e densità di energia;
- Assenza di un meccanismo di memoria.

Mentre gli svantaggi:

- Prezzo;
- Le prestazioni degradano ad elevate temperature;
- Fuga termica e diminuzione della capacità in caso di sovraccarica;
- Necessità di circuiti di controllo.

2.4 Processo elettrochimico

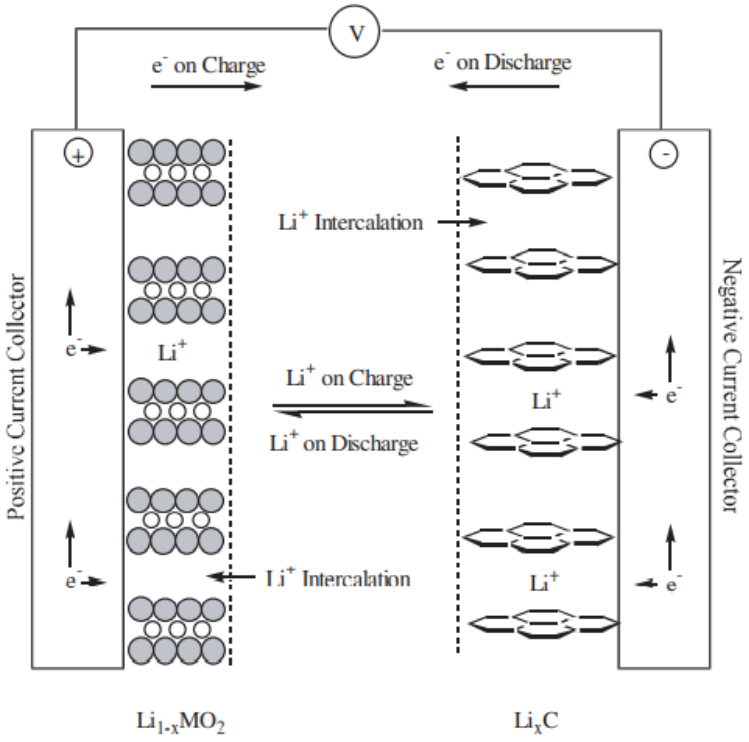
I materiali elettrochimicamente attivi in ogni singola cella della batteria operano incorporando in maniera reversibile gli ioni di litio attraverso un processo di intercalazione; ovvero si ha l'inserzione di una molecola o di un gruppo in una struttura ospite che va ad occupare porzioni specifiche di quest'ultimo e senza causarne rilevanti modifiche strutturali. [6] Nel caso in esame, la grafite e l'ossido di metallo rappresentano la struttura ospitante che ingloba gli ioni di litio. Quando una singola cella è carica, il materiale positivo risulta essere ossidato (cede elettroni) mentre quello negativo si riduce (acquista elettroni). In questa reazione di ossidoriduzione, gli ioni di litio vanno incontro ad un processo di intercalazione nel materiale negativo e ad un processo inverso nel materiale positivo; il fenomeno inverso avviene durante la scarica. Poiché il litio metallico è del tutto assente, le batterie agli ioni di litio sono meno reattive chimicamente, più sicure, e offrono cicli di vita più duraturi delle batterie che utilizzano il litio metallico come elettrodo negativo. Dunque, nell'ossido di metallo (M), che rappresenta l'elettrodo positivo, si ha l'inserzione e la rimozione di ioni di litio:

$$LiMO_2 \rightleftharpoons Li_{1-x}MO_2 + xLi^+ + xLi^+ + xe^-$$

Mentre nell'elettrodo negativo:

$$C + xLi^+ + xe^- \rightleftharpoons Li_xC$$

Dove il valore di x è tale che $0<x<1$. [4]



2.5 Processo di carica

Il processo avviene secondo il regime di carica imposto dal circuito di controllo; in particolare, il ciclo di carica può essere a corrente costante (CC) o a corrente e tensioni costanti con taper charge (CCCV). Generalmente, le celle delle batterie al litio sono caricate fino a 4.1 V o 4.2 V. Consideriamo il regime di carica CCCV: inizialmente, si ha una corrente costante (CC), la capacità cresce linearmente e la tensione si avvicina asintoticamente a 4.2 V. Nel momento in cui la carica avviene a tensione costante (CV), la corrente decade mentre la batteria raggiunge la carica finale.

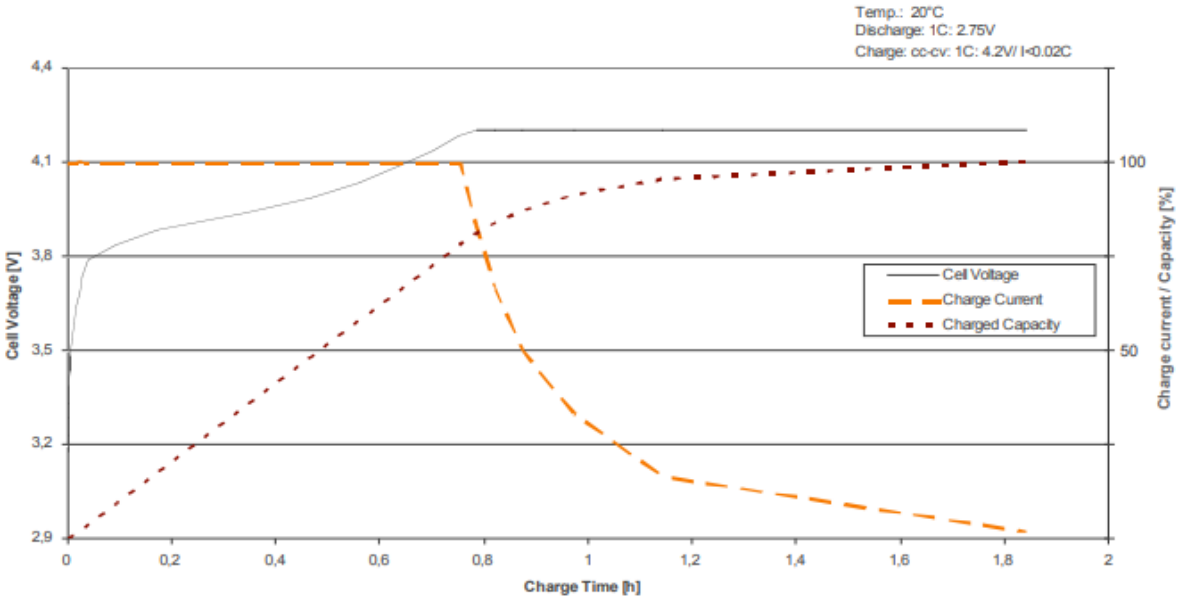
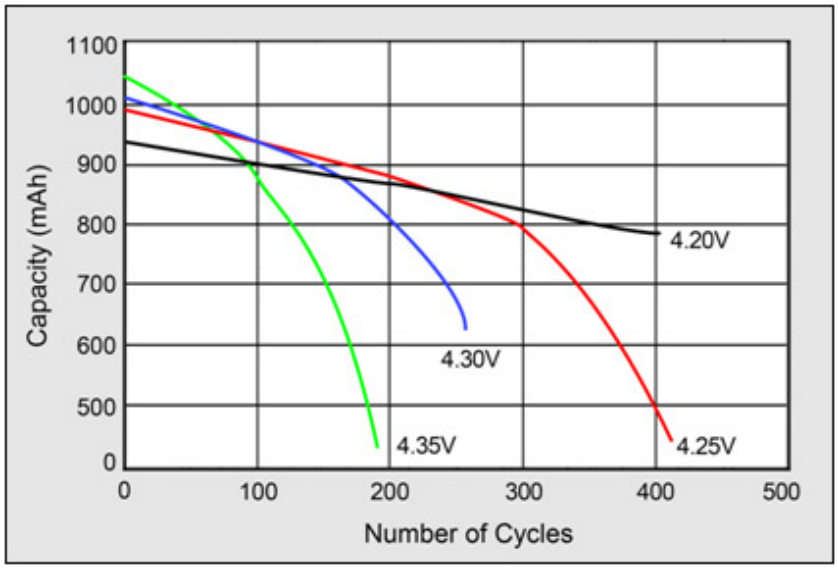


FIG. 9
Cell charging characteristics

Da notare che se la batteria viene caricata fino a 4.2 V invece che a 4.1 V, si ottiene una maggiore capacità dovuta al materiale dell'elettrodo positivo; tuttavia allo stesso tempo, si ha una riduzione della longevità e della

stabilità della batteria. Per ottenere elevati valori di capacità è possibile imporre tensioni superiori a 4.3 V; occorre tener presente che aumentare la tensione comporta comunque gli svantaggi descritti precedentemente. [4]

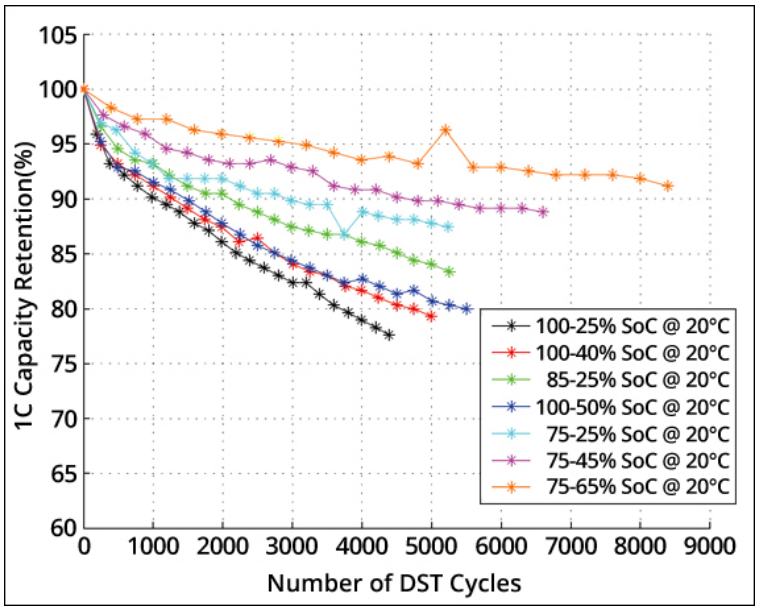


Dunque, la carica viene completata quando la batteria raggiunge la tensione di soglia desiderata e la corrente decade sotto il 3% della corrente massima. Incrementare la corrente non accelera il completamento della carica in maniera rilevante, ma consente il raggiungimento del valore di tensione massimo più velocemente (e permette di caricare la batteria a circa il 70%), riducendo la fase di carica a corrente costante. Tuttavia, ciò aumenterebbe la durata della fase successiva a tensione costante. Una volta terminata la carica, la tensione inizia a decrescere fino a stabilizzarsi tra i 3.7 V e i 3.9 V. Nel caso in cui le batterie dovessero essere lasciate in carica per garantire la pronta operatività, alcuni caricabatterie forniscono una breve corrente per compensare l'auto-scarica della batteria ed il consumo del circuito di protezione. Ad esempio, la carica può essere abilitata quando la tensione scende sotto i 4.00 V ed essere interrotta ancora al raggiungimento dei soli 4.05 V; in questo modo, si ha una riduzione dello stress dovuto all'alto voltaggio e un aumento della longevità della batteria. Le batterie agli ioni di litio operano in maniera sicura all'interno di valori di tensione che soddisfano le specifiche. Infatti, le batterie divengono instabili se inavvertitamente sottoposte a tensioni maggiori di quelle specificate; la prolungata carica sopra i 4.3 V di batterie al litio progettate per 4.2 V determina la formazione di placche di litio metallico in corrispondenza dell'anodo. Il materiale al catodo diviene un agente ossidante, perde stabilità e produce diossido di carbonio. Il verificarsi di esplosioni è connesso ovviamente all'elevata temperatura. Una batteria completamente carica presenta una minore fuga termica e potrebbe andare incontro a fenomeni di esplosione prima di una batteria parzialmente carica. [8]

2.6 Longevità della batteria

Fenomeni quali usura, elevate temperature e l'invecchiamento determinano un peggioramento delle performance nel tempo. L'indicatore principale in grado di quantificare lo stato di efficienza della batteria riconducibile al suo stato di salute risulta essere la capacità della batteria. Le condizioni che impattano sulla durata di utilizzo della batteria sono descritte di seguito:

- Un fattore rilevante che influenza la longevità della batteria è sicuramente la Depth of Discharge (DoD); quest'ultima determina il numero di cicli che la batteria sarà in grado di compiere. La diminuzione della DoD provocherà un aumento della vita della batteria; per questo è consigliabile evitare una scarica completa e caricare la batteria frequentemente (la batteria non presenta un meccanismo di memoria). Come si evince dal grafico seguente, al crescere dell'entità della DoD si ha un decremento più accentuato della capacità. Contemporaneamente, si verifica un innalzamento della resistenza interna delle celle.

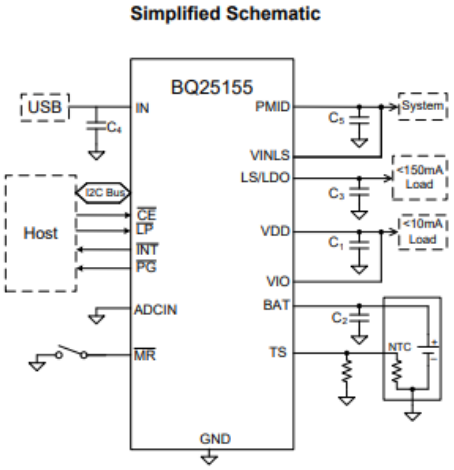


- Le batterie agli ioni di litio risentono negativamente dell'esposizione a calore e ad elevate temperature; in maniera analoga, deteriorano la propria performance se soggette ad una tensione di carica elevata per un periodo di tempo prolungato. A 30°C, una batteria è considerata soggetta ad elevata temperatura e per la maggior parte delle batterie agli ioni di litio una tensione superiore ai 4.1 V è ritenuta elevata. In questi casi la capacità della batteria diminuisce progressivamente. [9]

3. Caricabatterie

3.1 Panoramica del dispositivo

Il caricabatterie BQ25155 della Texas Instruments presenta un'ampia gamma di funzionalità utili nella realizzazione del progetto in esame; in particolare, il dispositivo è programmabile attraverso l'interfaccia IIC e presenta numerose funzionalità sia di configurazione che di monitoraggio. Inoltre, si tratta di un circuito altamente integrato destinato al controllo della carica che soddisfa le prerogative più diffuse tra i dispositivi portatili e/o indossabili.



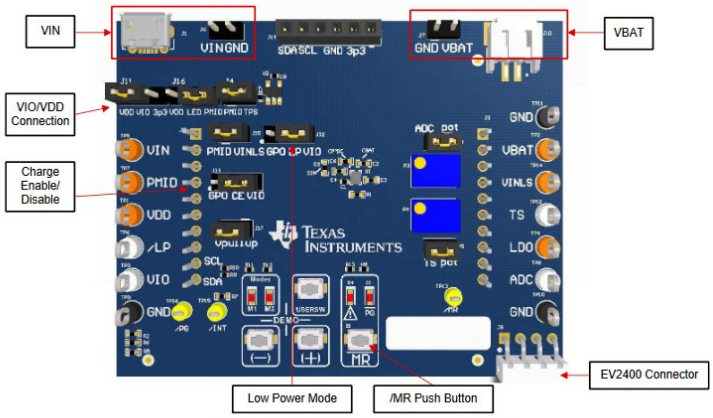


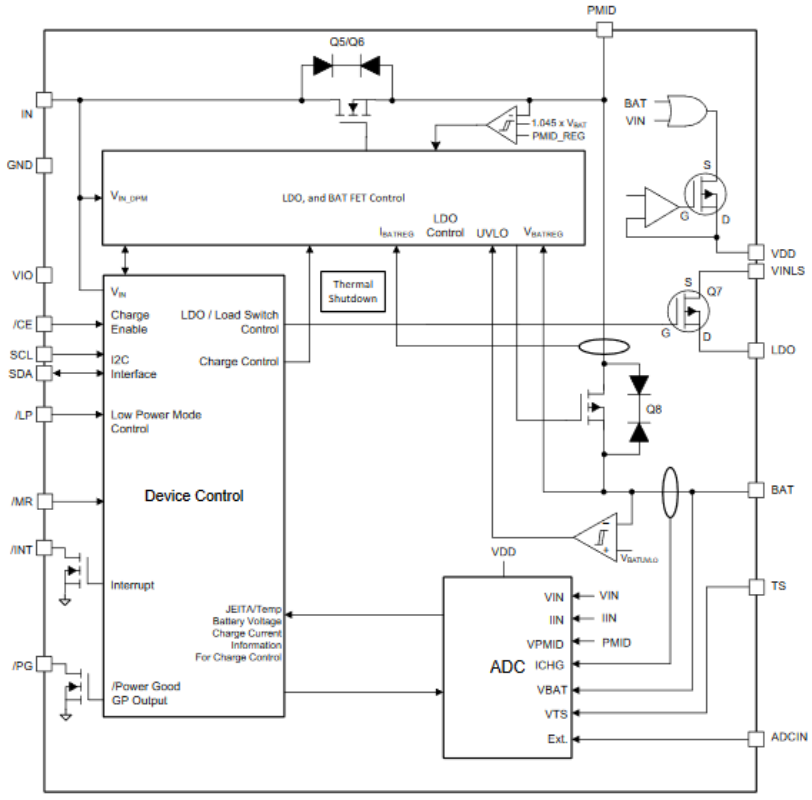
Figure 1. BQ2515xEVM Connections

In linea generale, il dispositivo presenta le seguenti caratteristiche chiave:

- Linear battery charger con un range di corrente di carica che va da 1.25 mA a 500 mA;
- Regolatore di tensione della batteria, programmabile tramite IIC con un range da 3.6 V fino a 4.6 V, accuratezza dello 0.5% e intervalli configurabili di 10 mV;

- Termination current configurabile con valore minimo di 0.5mA;
- Regime di carica termico configurabile dotato di valori di soglia per Hot, Warm, Cool, Cold temperature;
- Power Path Management per l'alimentazione del sistema e la carica della batteria;
- Quiescent current Iddq estremamente bassa per garantire una maggiore longevità della batteria. In particolare, in Ship Mode il valore della Iq viene mantenuto sui 10 nA, mentre quando viene alimentato il sistema (PMID e VDD on) si aggira sui 400 nA;
- Uscita LDO o Load Switch entrambi configurabili via IIC con una corrente fino a 150 mA;
- Pulsante /MR per il wake-up e l'HW reset del dispositivo dotato di timers configurabili;
- ADC a 16 bit, anch'esso configurabile tramite IIC e finalizzato al monitoraggio della corrente di carica, del termistore della batteria (TS), della tensione dell'input e del sistema (PMID); inoltre, è presente anche un general purpose ADC input;
- Carica ottimizzata per USB a 5 V, con un valore massimo di tolleranza di 20 V.

L'host (nel nostro caso l'nrf52480 Dongle) gestisce il caricabatterie andando a monitorare e regolando diversi aspetti del suo funzionamento mediante l'interfaccia IIC: in primo luogo, è possibile controllare il Power Path in modo da disconnettere l'alimentazione fornita dall'adattatore in ingresso e/o la batteria senza la rimozione fisica di questi ultimi. Inoltre, nel progetto viene sfruttata la trasmissione IIC per impostare i valori della corrente massima in ingresso, della corrente di carica, della tensione sull'uscita LDO e di altri parametri rilevanti per il corretto funzionamento del dispositivo. In particolare, sono personalizzabili anche le soglie di temperatura, fondamentali per garantire che la batteria operi in uno stato di salute ottimale. A tal proposito, da notare che la temperatura della batteria viene monitorata da un voltage-based JEITA compatibile (Japan Electronics and Information Technology Industries Association, protocollo industriale per migliorare le condizioni di sicurezza per la carica delle batterie [12]) battery pack thermistor monitoring input (TS); quest'ultimo modifica automaticamente i parametri della carica per evitare che avvenga a temperature che non rientrano in quelle definite dal range di sicurezza. [10][11]



Da notare che l'ingresso del caricabatterie presenta back to back FETs di blocco per evitare che la corrente scorra dal pin PMID a quello di ingresso. Per la corretta gestione del dispositivo è necessario, in via preliminare,

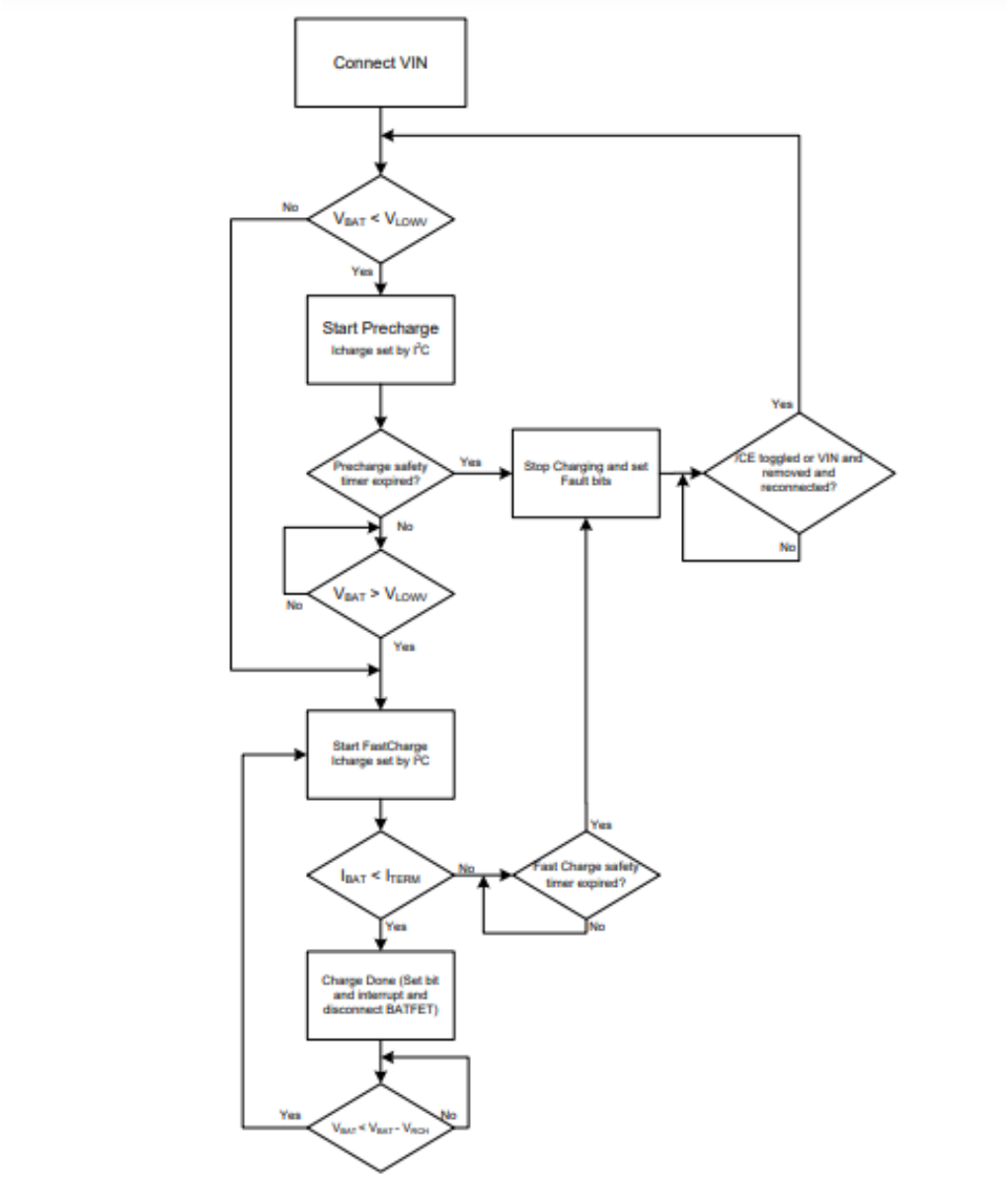
descriverne il funzionamento e le eventuali variazioni del comportamento del circuito a stimoli specifici. Di seguito verranno riassunte le funzionalità del dispositivo utili alla realizzazione del progetto.

3.2 PowerPath system

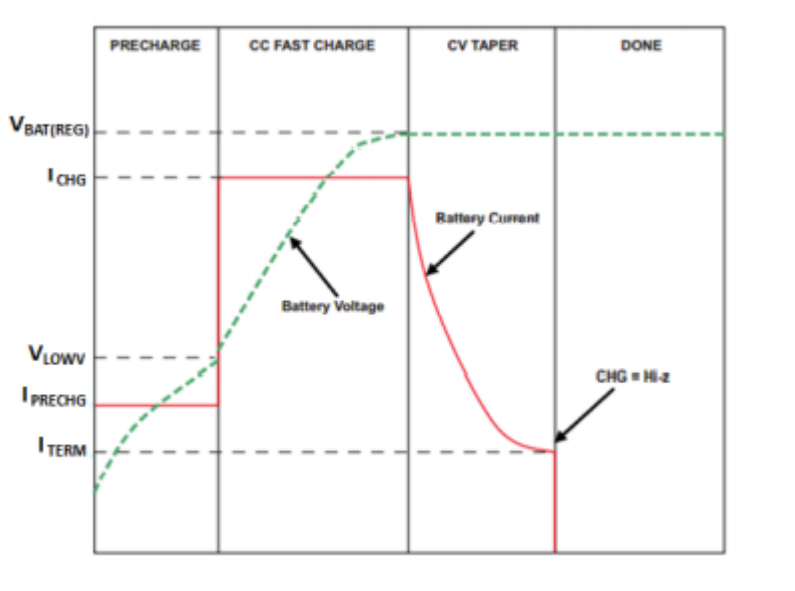
I sistemi di carica USB di seconda generazione o come vengono definiti comunemente PowerPath systems consentono al sistema di essere alimentato dal pin PMID (regulated system output), anche nel caso in cui la batteria sia assente o in carica, fornendo potenza direttamente dal pin di ingresso IN. Inoltre, nel caso in cui fosse necessario, è possibile rendere prioritaria l'alimentazione del carico rappresentato dal sistema a discapito della corrente di carica; in questo modo è possibile conservare la funzionalità del sistema anche quando la potenza in ingresso fosse limitata. Nel caso in cui l'alimentazione dal pin di input venga rimossa e la tensione della batteria sia superiore alla soglia indicata dal parametro $V_{BATUVLO}$, l'alimentazione del sistema verrà fornita automaticamente da parte della batteria.[10][13]

3.3 Processo di carica della batteria

Il processo di carica previsto dal caricabatterie è schematizzato nel seguente diagramma a blocchi:



Nel momento in cui viene rilevata la connessione con una idonea sorgente di alimentazione ($V_{IN} > V_{UVLO}$ e $V_{BAT} + V_{SLP} < V_{IN} < V_{OVP}$, dove V_{UVLO} è pari a 3.4 V per V_{IN} rising e 3.25 V per V_{IN} falling), l'inizio del ciclo di carica è stabilito dallo stato del pin /CE (Charge Enable). Infatti, quando il pin /CE risulta essere ad un livello logico alto, la carica è disabilitata (spegnendo il FET che permette la carica della batteria), mentre se il pin /CE assume un valore basso e contemporaneamente il bit CHARGE_DISABLE è pari a 0, si verifica l'inizio del ciclo di carica. In sintesi, la carica è abilitata esclusivamente se sia il pin /CE che il bit CHARGE_DISABLE si trovano al livello logico basso.



Il processo di carica è suddivisibile in 4 fasi, come si osserva anche nel grafico precedente: pre-charege, fast charge (CC), costant voltage (CV) e termination. Inizialmente, per prevenire che la batteria venga danneggiata, il dispositivo eseguirà la carica imponendo un livello di corrente di gran lunga minore rispetto al valore di regime ($I_{PRECHG} < I_{CHG}$) finché la tensione della batteria non avrà superato il valore di soglia V_{LOWV}. Completata la fase di pre-charge, il caricabatterie presenta due loop principali di gestione della carica: Costant Current (CC) e Constant Voltage (CV). L'esecuzione del primo avviene generalmente se $V_{BAT} < V_{BATREG}$ -50 mV a partire da $V_{BAT} = V_{UVLO}$ (con $V_{UVLO} = 3$ V di default); in tal caso la carica della batteria avviene al massimo livello di corrente (I_{CHARGE}), a meno che non sia soddisfatta una delle seguenti condizioni: fault condition sul pin TS, thermal charge current foldback attiva, VINDPM attiva o DPPM attiva. Una volta che la batteria raggiunge la tensione definita da V_{BATREG}, si innesca il secondo loop di gestione e la corrente di carica tende a diminuire (ogni variazione della I_{CHARGE} presuppone un'interruzione temporanea di circa 1 ms della carica). Infine, il dispositivo terminerà automaticamente la carica nel momento in cui la corrente di carica diviene minore o uguale alla I_{TERM}; il caricabatterie funzionerà quindi in high impedence mode, disabilitando l'accesso alla batteria e scollegandola. L'alimentazione del sistema (sul pin PMID) sarà comunque garantita dall'alimentazione in ingresso finché $V_{IN} > V_{UVLO}$ e $V_{BAT} + V_{SLP} < V_{IN} < V_{OVP}$. Da notare che la fase di termination sarà disabilitata durante l'esecuzione dei loop VINDPM e DPPM o quando il dispositivo sta operando nella regione di temperatura WARM. Precedentemente, si è accennato a due loop di gestione che il dispositivo mette a disposizione per fornire modalità differenti di controllo della carica:

- *Input Voltage Based Dynamic Power Management (VINDPM)*; si tratta di un loop volto a prevenire la condizione in cui la carica debba essere interrotta riducendo la corrente fornita dal caricatore per evitare che la tensione in ingresso V_{IN} scenda sotto la soglia stabilita da V_{IN_DPM}. Infatti, se la sorgente in ingresso non fosse più in grado di sostenere la corrente di carica desiderata e di alimentare il sistema, la tensione in ingresso sul pin IN decrementerebbe ulteriormente. Il loop di gestione descritto da VINDPM permette di ridurre la corrente in ingresso bloccando i FETs e prevenendo ulteriori abbassamenti di tensione da parte dell'alimentazione.
- *Dynamic Power Path Management Mode (DPPM)*; il DPPM loop permette di deviare la corrente in ingresso esclusivamente sul sistema interrompendo la carica nel caso in cui la corrente in ingresso non

sia sufficiente a garantire sia la carica della batteria che l'alimentazione del sistema. Infatti, se la V_{IN} è idonea la corrente viene ripartita dal PowerPath management tra il sistema e la batteria. Nel caso in cui la corrente richiesta dal processo di carica e dall'alimentazione del sistema ecceda quella massima effettivamente fornita dall'ingresso e configurata dal valore di I_{LIM}, si ha un progressivo decadimento della tensione su PMID. A questo punto, se PMID decresce sufficientemente da divenire minore della soglia imposta dal loop descritto dalla DPPM, quest'ultima riduce la corrente di carica (tramite il BATFET). In seguito, se la tensione su PMID continua a diminuire, la corrente di carica viene totalmente azzerata e il dispositivo entra in supplement mode. In questo caso, si ha che la corrente richiesta dal sistema sul pin di PMID eccede quella imposta in ingresso e conseguentemente la tensione su PMID si riduce ulteriormente. Quando tale tensione scende sotto la soglia imposta da V_{BSUP1} la batteria inizia ad alimentare il sistema e termina tale processo quando la tensione su PMID raggiunge un secondo valore di soglia V_{BSUP2} [11]

Per le finalità del progetto in esame, entrambe le funzionalità descritte precedentemente sono disabilitate attraverso l'interfaccia IIC.

3.4 Modalità di funzionamento

Il BQ25155 presenta 4 modalità di funzionamento principali: Active Battery Mode, Low Power Mode, Charger Mode e Adapter Mode; le prime due riguardano l'alimentazione attraverso la sola batteria, le ultime due richiedono la connessione all'alimentazione in ingresso. Di seguito sono riportate le funzionalità del caricabatterie che risultano essere attive per ogni modalità:

FUNCTION	CHARGE/ ADAPTER MODE	SHIP MODE	LOW POWER MODE	ACTIVE BATTERY MODE
VOVP	Yes	No	Yes	Yes
VUVLO	Yes	Yes	Yes	Yes
BATOCP	Yes	No	No	Yes
BATUVLO	Yes	No	Yes	Yes
VINDPM	If enabled	No	No	No
DPPM	If enabled	No	No	No
VDD	Yes	No	Yes	Yes
LS/LDO	Yes	No	If enabled	If enabled
BATFET	Yes	No	Yes	Yes
TS Measurement	Yes	No	No	If enabled
Battery Changing	If enabled	No	No	No
ILIM	Yes (Register Value)	No	No	No
$\overline{MR}$ input	Yes	Yes	Yes	Yes
$\overline{LP}$ input	No	No	Yes	Yes
$\overline{INT}$ output	Yes	No	No	Yes
I^2C	Yes	No	No	Yes
$\overline{CE}$ input	Yes	No	No	No
ADC	Yes	No	No	Yes

- **Ship Mode**
  Permette l'operatività del dispositivo con la minima quiescent current. Questo modo di funzionamento impone il blocco del dispositivo e conseguentemente della carica della batteria finché non viene premuto il pulsante di reset (/MR = 0) per un periodo di tempo fissato e poi rilasciato, o il valore della tensione V_{IN} superi quello della V_{UVLO}. Tale modalità entra in funzione anche quando non è presente un'alimentazione idonea in ingresso. Se il bit corrispondente a EN_SHIPMODE è pari a 1, il dispositivo attenderà per entrare in ship mode finché l'alimentazione in ingresso non sarà rimossa.
- **Low Power**
  Tale modalità è operativa nel momento in cui il pin /LP è al livello logico basso, $V_{IN} < V_{UVLO}$, il pin /MR è alto e tutte le transazioni IIC e gli interrupts che erano stati avviati durante le altre modalità

(Active Battery mode e Charging mode) sono stati completati ed inviati. Da notare che sia l'ADC che l'interfaccia IIC risultano essere disabilitate. Tutte le altre componenti circuitali saranno attive in low power state. Il dispositivo esce dalla Low Power Mode se il pin /LP assume il valore logico alto o $V_{IN} > V_{UVLO}$. Inoltre, se la connessione all'alimentazione risulta essere tale da imporre una tensione $V_{IN} > V_{OVP}$ (condizione di overvoltage) mentre il pin /LP è al livello logico basso, il dispositivo sarà alimentato dalla batteria ed opererà in Active Battery Mode. Infine, è possibile interrompere temporaneamente tale modalità e passare alla Active Battery Mode tenendo premuto il pulsante /MR per un certo intervallo di tempo programmabile e poi rilasciandolo.

- **Active Battery**
  Se il dispositivo non si trova in Ship Mode e la tensione sulla batteria è superiore alla soglia minima V_{UVLO} benché non sia presente una tensione di ingresso valida, la connessione tra il pin PMID e la batteria viene abilitata. La corrente quindi scorrerà dalla batteria al sistema su PMID e verrà monitorata dall'over-current protection circuitry. Nel momento in cui la tensione della batteria scende sotto la soglia, la scarica della batteria sul sistema verrà interrotta.
- **Charger/Adapter Mode**
  Questa modalità entra in funzione quando $V_{IN} > V_{UVLO}$; l'ADC è abilitato e svolge la sua attività continuamente su tutti i canali. Se l'alimentazione è idonea e superiore a quella fissata dalla soglia VIN_DPM, il sistema sul pin PMID sarà alimentato dall'ingresso; infine, la carica della batteria avverrà come stabilito dalla configurazione. [11]

3.5 Convertitore analogico-digitale

Il caricabatterie dispone di un ADC a 16 bit che permette di avere a disposizione informazioni inerenti alla tensione d'ingresso, la corrente in ingresso, la tensione su PMID, la tensione della batteria, la corrente di carica della batteria e la tensione sul pin TS. Può anche essere utilizzato per effettuare misurazioni su una sorgente esterna attraverso il pin ADCIN. La tabella seguente riporta i parametri misurabili dall'ADC e le relative formule per ricavare gli effettivi valori a partire dalla sequenza di bit letta nei registri dell'ADC

MEASUREMENT	FULL SCALE RANGE (ABSOLUTE MAX CODE)	FULL LINEAR RANGE (RECOMMENDED OPERATING RANGE)	FORMULA
VIN	6 V	2 V - 5 V	$V_{IN} = \frac{ADCDATA\_VIN^{16bit}}{2^{16}} \times 6V$ (6)
PMID	6 V	2 V - 5 V	$V_{PMID} = \frac{ADCDATA\_PMID^{16bit}}{2^{16}} \times 6V$ (7)
IIN	750 mA	0 - 600 mA	For ILIM ≤ 150mA: $I_{IN} = \frac{ADCDATA\_IIN^{16bit}}{2^{16}} \times 375mA$ (8) For ILIM >150mA: $I_{IN} = \frac{ADCDATA\_IIN^{16bit}}{2^{16}} \times 750mA$ (9) Note: IIN reading only valid when $V_{IN} > V_{UVLO}$ and $V_{IN} < V_{OVP}$
VBAT	6 V	2 V - 5 V	$VBAT = \frac{ADCDATA\_VBAT^{16bit}}{2^{16}} \times 6V$ (10)
TS	1.2 V	0 - 1 V	$V_{TS} = \frac{ADCDATA\_TS^{16bit}}{2^{16}} \times 1.2V$ (11)
ADCIN	1.2 V	0 - 1 V	$V_{ADCIN} = \frac{ADCDATA\_ADCIN^{16bit}}{2^{16}} \times 1.2V$ (12)
% ICHARGE	-	-	$\%I_{CHARGE} = \frac{ADCDATA\_ICHG^{16bit}}{0.8 \times 2^{16}} \times 100$ (13) where I_{CHARGE} is the charge current setting. Note that if the device is in pre-charge or in the TS COLD region, I_{CHARGE} will be the current set by the IPRECHRG and TS_ICHRG bits respectively.

Quando il BQ25155 è alimentato dalla batteria, è assolutamente necessario minimizzare i consumi per prolungare la durata della carica della batteria. A questo scopo, il numero delle conversioni analogico-digitale viene sensibilmente ridotto in modo che l'ADC, in Active Battery Mode, operi ad una frequenza imposta dai bit ADCREAD_RATE. Al termine di ogni conversione, il flag ADC_READY verrà settato a 1 e sarà inviato un interrupt all'host. Inoltre, come già accennato precedentemente, in Low Power Mode, l'ADC è completamente disattivato. Infine, occorre notare che se il dispositivo risulta essere alimentato dalla sorgente VIN, l'ADC sarà attivo costantemente; ovvero le operazioni di conversione si susseguiranno ininterrottamente. Per evitare quindi che l'host venga travolto dagli interrupt dell'ADC_READY inviati continuamente dal caricabatterie, questi ultimi non verranno abilitati e l'host potrà leggere i valori della conversione analogico-digitale in qualsiasi momento. Un'altra rilevante funzionalità messa a disposizione dall'hardware del caricabatterie riguarda i comparatori dell'ADC programmabili; questi ultimi possono essere utilizzati per monitorare qualsiasi canale del convertitore analogico-digitale accedendo tramite IIC ai registri di controllo; in questo modo il dispositivo invierà un interrupt all'host nel momento in cui la misurazione compiuto dall'ADC sia superiore o inferiore alla soglia impostata all'interno dei registri di configurazione del suddetto (ADC_ALARM_COMPx). Da notare che il valore che indica la soglia del comparatore è espresso in 12 bit, dunque solo i primi 12 bit più significativi sui 16 bit della misurazione dell'ADC sono presi in considerazione per il confronto. [11]

3.6 Meccanismi di protezione

Il BQ25155 mette a disposizione diversi sistemi per garantire la sicurezza in differenti condizioni di operatività:

- **Input Over-Voltage Protection**
  Permette, come si può dedurre dal nome, di evitare danni al dispositivo e ad altre componenti connesse ai pin PMID e BAT a seguito di tensioni eccessivamente elevate dell'alimentazione in ingresso. In particolare, nel momento in cui viene riscontrata una condizione tale per cui $V_{IN} > V_{OVP}$, si determinerà un OVP fault all'uscita. Durante quest'ultimo, il dispositivo interrompe la carica della batteria, nello specifico l'input FET viene interdetto ed è inviato un singolo impulso di interrupt. Infine, il flag VIN_OVP_FAULT e lo STAT bit dei registri accessibili via IIC sono aggiornati in modo da rendere a conoscenza l'host dell'evento di fault. Una volta che la condizione di overvoltage non è più presente il dispositivo ritorna nelle condizioni di operatività regolari. Il bit di status viene ripristinato ma non quello di flag che deve essere necessariamente letto tramite IIC (R/C). La soglia di overvoltage viene fissata di default a 5.5 V in conformità con lo standard dell'alimentazione USB.

- **Safety Timer e IIC Watchdog Timer**
  All'inizio di un ciclo di carica, il dispositivo innesca il safety timer. Se la carica non è terminata prima che sia trascorso un intervallo di tempo fissato dal valore programmabile di t_{MAXCHG}, il safety timer scade e la carica viene disabilitata. In maniera analoga, il pre-charge safety timer presenta un intervallo di tempo oltre il quale il timer scade che è fissato da t_{PRECHG} ed è pari a circa il 25% di t_{MAXCHG}. Quando il fault relativo al safety timer viene rilevato, si ha l'invio di un impulso di interrupt sul pin /INT e il corrispettivo flag nei registri IIC viene impostato ad 1. A questo punto, per riabilitare la carica e uscire dalla condizione di fault resettando il timer è necessario il toggle del pin /CE o dell'alimentazione in ingresso. Il flag verrà resettato esclusivamente dopo la lettura del registro. Oltre al safety timer, il dispositivo presenta un IIC watchdog timer capace di tener conto di un intervallo di tempo della durata massima di 50 secondi. Quest'ultimo permette di monitorare l'host attraverso l'interfaccia IIC. È abilitato di default ed è attivato immediatamente a seguito dell'abilitazione; viene successivamente resettato ogni volta che avviene una transizione IIC da parte dell'host. Se l'intervallo di tempo fissato dal watchdog timer scade senza che avvenga nessuna trasmissione IIC da parte dell'host, tutti i parametri del caricatore vengono resettati (I_{CHARGE}, $I_{PRECHARGE}$, I_{TERM}, V_{LOW} …).

- **Thermal Protection e Thermal Charge Current Foldback**

In condizioni di operatività, per proteggere il dispositivo da eventuali danni a causa di surriscaldamenti, viene monitorata la temperatura di giunzione T_J; quando quest'ultima raggiunge la soglia fissata da $T_{SHUTDOWN}$, il dispositivo interrompe le operazioni ed è spento. Il dispositivo riprende le funzionalità quando T_J scende sotto $T_{SHUTDOWN}$ di un certo valore T_{HYS}.

- **Battery Short e Over Current Protection**
  Per proteggere il dispositivo da sovracorrenti e prevenire una eccessiva corrente di scarica della batteria, il BQ25155 rileva se la corrente sul FET della batteria ecceda la soglia I_{BAT_OCP}. Se il limite di corto circuito viene raggiunto entro il tempo fissato da T_{DGL_OCP}, il FET della batteria è disabilitato e riabilitato ogni circa 250 ms, operando quindi a intermittenza. Ciò può avvenire da 3 fino a 7 volte, dopo di che, se viene riscontrata ancora una condizione di over-current, il BATFET viene disabilitato definitivamente, finché V_{IN} non sarà connessa ad una sorgente valida. [11]

3.7 LDO output

Il dispositivo presenta un pin LDO che consiste in un'uscita a bassa corrente sempre attiva; ciò funge da alimentazione di I/O digitale al dispositivo; la potenza viene fornita dall'ingresso VIN o direttamente dalla batteria. L'utilizzatore finale potrà imporre che vengano erogati fino a 10 mA di corrente attraverso il pin VDD per alimentare un led di stato o un alimentatore I/O. La VDD LDO rimarrà attiva in tutte le modalità di funzionamento del dispositivo, fatta eccezione per la Ship Mode. Il BQ25155 integra inoltre un load switch con bassa Iq che può essere usato come un'uscita a tensione regolata. L'uscita LDO/LS presenta un pin di input VINLS e può supportare fino a 150 mA di corrente di carico. Inoltre, l'host può controllare direttamente tale uscita attraverso il bit di enable (EN_LS_LDO) del registro LDOCTRL configurabile via IIC. Per le esigenze imposte dal progetto in esame, la tensione sul pin LS/LDO verrà imposta a 2.8 V.

3.8 PMID power control

Il caricabatterie mette a disposizione l'opzione di controllare l'operatività del pin PMID attraverso i due bits PMID_MOD del registro ICCTRL1 configurabile via IIC. Questi ultimi possono imporre che l'alimentazione di PMID venga fornita dalla batteria anziché dall'ingresso, anche nella condizione in cui VIN > VBAT + VSLP. In maniera analoga, è possibile disconnettere il sistema sul pin PMID. Sono quindi riportate le combinazioni di funzionamento per il pin PMID:

- *PMID_MODE = 00*
  Si tratta della condizione di operatività di default del caricabatterie. Il sistema sarà alimentato da VIN se quest'ultima risulta essere una tensione valida o sarà alimentato dalla batteria in caso contrario. PMID sarà disconnesso esclusivamente se si verifica un reset hardware o il dispositivo entra in Ship Mode. Questa configurazione è anche quella che sarà adottata dal progetto in questione.
- *PMID_MODE = 01*
  In tal caso, il sistema su PMID sarà alimentato dalla batteria se la tensione su quest'ultima risulterà essere maggiore della soglia minima imposta da $V_{BATUVLO}$, indipendentemente dalla tensione in ingresso. In questo modo sarà possibili minimizzare la corrente assorbita a partire dall'adattatore. Nel caso in cui la tensione della batteria dovesse scendere sotto la $V_{BATUVLO}$, si passerà automaticamente alla configurazione precedente.
- *PMID_MODE = 10*
  Permette la disconnessione dall'alimentazione (sia dalla batteria che dall'ingresso VIN) del sistema su PMID in modo che il pin rimanga floattante. Inoltre, il pin LDO sarà disabilitato e l'uscita da tale modalità potrà avvenire esclusivamente attraverso l'interfaccia IIC o la pressione del pulsante /MR per un reset hardware.
- *PMID_MODE = 11*

Il sistema su PMID sarà disconnesso dall'alimentazione (sia fornita dalla batteria che dall'ingresso) e il pin PMID sarà connesso al riferimento di ground. Inoltre, la VDD rimarrà attiva mentre l'uscita LDO sarà disabilitata. Come la configurazione precedente, tale modalità può essere disabilitata esclusivamente tramite IIC o con reset hardware. [11]

Da notare che tutte le combinazioni diverse dalla prima dovranno essere evitate per garantire la compatibilità del dispositivo con il progetto.

3.9 Interfaccia IIC

Come già descritto in precedenza, il BQ25155 dispone di un'interfaccia IIC per permettere all'host di programmare e leggere i parametri di controllo, i bits di stato, i bits di flag e così via. L'IIC è un protocollo di comunicazione basato su due linee di interfaccia seriale che consistono in una data line (SDA) e una clock line (SCL), entrambe dotate di resistori di pull-up. In questo modo, quando il bus è in idle, entrambe le linee risultano essere ad un livello logico alto. Il dispositivo master controlla il bus e nel nostro caso è rappresentato dall'nrf52840; quest'ultimo è responsabile della generazione del segnale di clock della SCL e dell'indirizzo dello slave da trasmettere sulla SDA per stabilire il dispositivo con cui comunicare. Inoltre, il master si occupa dell'esecuzione delle START e STOP conditions per il trasferimento dati. Da notare che le modalità di trasferimento del BQ25155 che opera come slave sono compatibili con quelle dell'nrf52840. Infatti, entrambi i dispositivi operano seguendo la specifica per il bus IIC di 400 kbps (fast mode). Le specifiche dettate dell'interfaccia IIC adoperata possono essere reperite dettagliatamente dal seguente documento: *I 2C-Bus Specification, Version 2.1, January 2000* (https://www.csd.uoc.gr/~hy428/reading/i2c_spec.pdf). [11]

3.10 External NTC Monitoring (TS)

Una importante applicazione dell'interfaccia IIC messa a disposizione dal dispositivo consiste nel permettere all'utilizzatore di implementare gli standard JEITA per la sicurezza del sistema attraverso il monitoraggio del termistore della batteria da parte dell'utente. In particolare, l'input pin TS permette di monitorare la tensione sul termistore della batteria e conseguentemente di garantire che la batteria sia ad una temperatura tale da poterne eseguire la ricarica. Per soddisfare quelli che sono i requisiti dello standard JEITA, sono monitorate quattro temperature di soglia: cold battery threshold, cool battery threshold, warm battery threshold e hot battery threshold; queste ultime corrispondono rispettivamente alle tensioni di soglia V_{COLD}, V_{COOL}; V_{WARM}; V_{HOT}. Da notare che la ricarica e il safety timer vengono interrotti se $V_{TS} < V_{HOT}$ o $V_{TS} > V_{COLD}$, mentre quando $V_{COOL} < V_{TS} < V_{COLD}$, la corrente di ricarica viene ridotta a quella imposta dall'host. Le tensioni di soglia enunciate precedentemente corrispondono a differenti temperature a seconda della resistenza del termistore; ad esempio, per un termistore da 10 kΩ vale la seguente corrispondenza:

THRESHOLD	TEMPERATURE (°C)	VTS (V)
Open	--	>0.9
Cold	0	0.585
Cool	10	0.514
Warm	45	0.265
Hot	60	0.185

Infine, si ha che quando si verifica un TS fault o nel momento in cui la tensione su TS attraversa una delle soglie definite precedentemente verrà inviato un interrupt all'host; inoltre, nel primo caso viene anche disabilitata la ricarica. [11]

3.11 Interrupts del caricabatterie

Indispensabile per il controllo del dispositivo risulta essere sicuramente il meccanismo di interrupt del caricabatterie. Il BQ25155 presenta un output open-drain per la segnalazione degli interrupt che risulta operativo esclusivamente dopo che il dispositivo ha completato l'avvio in uno stato valido. Il pin /INT è in condizioni normali in alta impedenza; se viene riscontrata una condizione di interrupt, /INT viene imposto al livello logico basso per un intervallo di tempo di 128 µs per poi ritornare alla condizione iniziale come rappresentato di seguito:

Dunque, nel momento in cui si verifica un qualsiasi evento in grado di innescare un interrupt (status change o fault condition), viene inviato all'host un impulso della durata di 128 µs attraverso il pin /INT. Ovviamente, gli interrupts possono essere mascherati dall'host attraverso la configurazione dei registri via IIC in modo che quest'ultimo venga notificato esclusivamente degli eventi per cui si è implementata una corretta routine di gestione. Di seguito è riportato il comportamento del caricabatterie in risposta al verificarsi di fault conditions:

FAULT / FLAG	DESCRIPTION	INTERRUPT TRIGGER BASED ON STATUS BIT CHANGE	CHARGER BEHAVIOR	CHARGER SAFETY TIMER	LS/LDO BEHAVIOR	PMID BEHAVIOR	NOTES
CHRG_CV_FLAG	Set when charger enters Constant Voltage operation	Rising Edge	Enabled	No effect	N/A	IN powered if V_{IN} is valid	
CHARGE_DONE_FLAG	Set when charger reaches termination	Rising Edge	Paused-Charging resumes with VIN or CE toggle or when V_{RCH} is reached	Reset	N/A	IN powered if V_{IN} is valid	
IINLIM_ACTIVE_FLAG	Set when Input Current Limit loop is active	Rising Edge	Enabled. Reduced charge current.	Doubled if option is enabled	N/A	IN powered VIN powered unless supplement mode condition is met.	
VDPPM_ACTIVE_FLAG	Set when DPPM loop is active	Rising Edge	Enabled. Reduced charge current.	Doubled if option is enabled	N/A	VIN powered unless supplement mode condition is met.	
VINDPM_ACTIVE_FLAG	Set when VINDPM loop is active	Rising Edge	Enabled. Reduced charge current.	Doubled if option is enabled	N/A	VIN powered unless supplement mode condition is met.	
THERMREG_ACTIVE	Set when Thermal Charge Current Foldback (Thermal Regulation) loop is active	Rising Edge	Enabled. Reduced charge current.	Doubled if option is enabled	N/A	VIN powered unless supplement mode condition is met.	
VIN_PGOOD_FLAG	Set when VIN changes PGOOD status	Rising and Falling Edge	If VIN_PGOOD_STAT is low, charging is disabled.	Reset	N/A	VIN powered (if VIN_PGOOD_STAT=1) unless PMID_MODE is not 00.	Interrupt will not be sent if device powers up with VIN_PGOOD condition and $V_{BAT} < V_{BATUVLO}$
VIN_OVP_FAULT_FLAG	Set when $V_{IN} > V_{OVP}$	Rising Edge	Charging is paused until condition disappears	Reset	N/A	BAT powered	
BAT_OCP_FAULT_FLAG	Set when $I_{BAT} > I_{BATOCP}$	Rising Edge	Disabled (BAT only condition)	N/A	N/A	Disconnect BAT	
BAT_UVLO_FAULT_FLAG	Set when $V_{BAT} < V_{BATUVLO}$	Rising Edge	Enabled	No effect	N/A	IN powered of V_{IN} is valid	
TS_COLD_FLAG	Set when $V_{TS} > V_{TS\_COLD}$	Rising Edge	Charging paused until condition is cleared	Paused	N/A	IN powered of V_{IN} is valid	
TS_COOL_FLAG	Set when $V_{TS\_COLD} > V_{TS} > V_{TS\_COOL}$	Rising Edge	Enabled. Reduced charge current.	Doubled if option is enabled	N/A	IN powered of V_{IN} is valid	

FAULT / FLAG	DESCRIPTION	INTERRUPT TRIGGER BASED ON STATUS BIT CHANGE	CHARGER BEHAVIOR	CHARGER SAFETY TIMER	LS/LDO BEHAVIOR	PMID BEHAVIOR	NOTES
TS_WARM_FLAG	Set when $V_{TS\_HOT} < V_{TS} < V_{TS\_WARM}$	Rising Edge	Enabled. Reduce battery regulation voltage.	No effect	N/A	IN powered of V_{IN} is valid	
TS_HOT_FLAG	Set when $V_{TS} < V_{HOT}$	Rising Edge	Charging paused until condition is cleared	Paused	N/A	IN powered of V_{IN} is valid	
ADC_READY_FLAG	Set when ADC conversion is completed	Rising Edge	N/A	N/A	N/A	N/A	
COMP1_ALARM_FLAG	Set when ADC measurement meets programmed condition	Rising Edge	N/A	N/A	N/A	N/A	
COMP2_ALARM_FLAG	Set when ADC measurement meets programmed condition	Rising Edge	N/A	N/A	N/A	N/A	
COMP3_ALARM_FLAG	Set when ADC measurement meets programmed condition	Rising Edge	N/A	N/A	N/A	N/A	
TS_OPEN_FLAG	Set when $V_{TS} > V_{TS\_OPEN}$	Rising Edge	Charging is paused until condition disappears	Paused	N/A	N/A	
WD_FAULT_FLAG	Set when I^2C watchdog timer expires	Rising Edge	Enabled	N/A	N/A	N/A	
SAFETY_TMR_FAULT_FLAG	Set when safety Timer expires. Cleared after VIN or $\overline{CE}$ toggle	Rising Edge	Disabled until VIN or $\overline{CE}$ toggle	Reset after flag is cleared	N/A	IN powered of V_{IN} is valid	
LS_LDO_OCP_FAULT_FLAG	Set when LDO output current exceeds OCP condition	Rising Edge	N/A	N/A	Enabled (host must take action to disable the LDO if desired)	N/A	
MRWAKE1_TIMEOUT_FLAG	Set when $\overline{MR}$ is low for at least t_{WAKE1}	Rising Edge	N/A	N/A	N/A	N/A	
MRWAKE2_TIMEOUT_FLAG	Set when $\overline{MR}$ is low for at least t_{WAKE2}	Rising Edge	N/A	N/A	N/A	N/A	
MRRESET_WARN_FLAG	Set when $\overline{MR}$ is low for at least $t_{RESETWARN}$	Rising Edge	N/A	N/A	N/A	N/A	
TSHUT	No flag. Die temperature exceeds thermal shutdown threshold is reached	N/A	Disabled	Disabled	Disabled	Disabled	

[11]

3.12 Parametri configurabili

Di seguito, viene fatto riferimento ai valori di interesse che sono stati configurati mediante l'interfaccia IIC attraverso la scrittura negli appositi registri. Inoltre, sono riportate le tabelle che descrivono alcune caratteristiche chiave di tali parametri e la notazione associata a ciascun valore. Nello specifico, i parametri che riguardano la carica della batteria sono riportati nella tabella seguente con i relativi range di operatività; in particolare, per il progetto in esame, i parametri principali che verranno impostati tramite l'interfaccia IIC sono V_{BATREG}, I_{CHARGE}, $I_{PRECHARGE}$, V_{LOW}.

BATTERY CHARGER						
V_{DPPM}	PMID voltage threshold when charge current is reduced	V_{PMID} - V_{BAT}		200		mV
$R_{ON(BAT-PMID)}$	Battery Discharge FET On Resistance	V_{BAT} = 4.35V, I_{BAT} = 100mA		100	175	mΩ
V_{BATREG}	Charge Voltage	Programmable charge voltage range	3.6		4.6	V
	Voltage Regulation Accuracy		0.5		0.5	%
I_{CHARGE}	Fast Charge Programmable Current Range	V_{LOWV} < V_{BAT} < V_{BATREG}	1.25		500	mA
	Fast Charge Current Accuracy	T_J = 25°C, I_{CHARGE} > 5mA	–5		5	%
$I_{PRECHARGE}$	Precharge current	Precharge current programmable range	1.25		77.5	mA
	Precharge Current Accuracy	-40°C < T_J < 85°C	–10		10	%
I_{TERM}	Termination Charge Current	Termination Current Programmable Range	1		31	%
	Accuracy	T_J = 25°C, I_{TERM} = 10% I_{CHARGE}, I_{CHARGE} = 100mA	–5		5	%
		-10°C < T_J < 85°C, I_{TERM} = 10% I_{CHARGE}, I_{CHARGE} = 100mA	–10		10	%
V_{LOWV}	Programmable voltage threshold for pre-charge to fast charge transitions	VBAT rising. Programmable Range	2.8		3	V
V_{SHORT}	Battery voltage threshold for short detection	VBAT falling, VIN = 5V	2.41	2.54	2.67	V
I_{SHORT}	Charge Current in Battery Short Condition	V_{BAT} < V_{SHORT}		$I_{PRECHARGE}$		mA
V_{RCH}	Recharge Threshold voltage	V_{BAT} falling, V_{BATREG} = 4.2V, V_{RCH} = 140mV setting		140		mV
		V_{BAT} falling, V_{BATREG} = 4.2V, V_{RCH} = 200mV setting		200		mV
R_{PMID_PD}	PMID pull-down resistance	V_{PMID} = 3.6V		25		Ω

Di rilevanza, risultano essere sicuramente i parametri relativi alla temperatura; per ciascuno di essi viene riportata la corrispondenza tra range di valori espressi in volt e range di temperatura. In questo modo è possibile valutare la temperatura effettiva a partire dal valore della misurazione attuata dal convertitore analogico-digitale sul pin TS. Inoltre, è utile anche aver presente le diverse soglie di temperatura che corrispondono a differenti condizioni di fault (nello specifico open, cold, cool, warm, hot).

BATTERY PACK NTC MONITOR						
V_{HOT}	High temperature threshold	V_{TS} falling, -10°C < T_J < 85°C	0.182	0.185	0.189	V
V_{WARM}	Warm temperature threshold	V_{TS} falling, -10°C < T_J < 85°C	0.262	0.265	0.268	V
V_{COOL}	Cool temperature threshold	V_{TS} rising, -10°C < T_J < 85°C	0.510	0.514	0.518	V
V_{COLD}	Cold temperature threshold	V_{TS} rising, -10°C < T_J < 85°C	0.581	0.585	0.589	V
V_{OPEN}	TS Open threshold	V_{TS} rising, -10°C < T_J < 85°C		0.9		V
V_{HYS}	Threshold hysteresis			4.7		mV

Un altro parametro da considerare risulta essere sicuramente la soglia di tensione che distingue la condizione di undervoltage della batteria da quella di normale operatività.

$V_{BATUVLO}$	Battery undervoltage Lockout Threshold Voltage	Programmable range, 150 mV Hysteresis	2.4	3	V
	Accuracy		–3	3	%
	Battery undervoltage Lockout Threshold Voltage at Power Up	V_{BAT} rising, V_{IN} = 0V, T_J = 25°C	3.15		V

Inoltre, sono configurati anche i parametri relativi alla tensione sul pin di uscita LDO, e alla corrente di ingresso; occorre quindi tener conto dei rispettivi range di operatività e dei valori che possono essere assegnati attraverso la programmazione dei registri IIC del caricabatterie:

V_{LDO}	LDO programmable output voltage range		0.6	3.7	V
	LDO output accuracy	T_J = 25°C	–2	2	%
		V_{LDO} = 1.8V, V_{INLS} =3.6V. I_{LOAD} = 1mA	–3	3	%

I_{ILIM}	Input Current Limit	Programmable Range	50	600	mA
		I_{ILIM} = 50mA	45	50	mA
		I_{ILIM} = 100mA	90	100	mA
		I_{ILIM} = 150mA	135	150	mA
		I_{ILIM} = 500mA	450	500	mA

4. Microcontrollore

4.1 Panoramica del dispositivo

Il dispositivo utilizzato come host per il caricabatterie risulta essere il SoC *nrf52840 Dongle PC10059*; di seguito vengono riassunte le caratteristiche chiave utili per la realizzazione del progetto:

- CPU: ARM ® Cortex ® -M4 32-bit processor with FPU, 64 MHz;
- Memoria Flash: 1Mb
- Memoria RAM: 256 kB;
- Comunicazione Bluetooth: Bluetooth® 5, Low Energy, ANT™, 802.15.4;
- Sistema di sicurezza: ARM ® TrustZone ® Cryptocell 310 security subsystem;
- USB: 2.0 full speed (12 Mbps) controller;

L'aspetto sicuramente più rilevante del dispositivo risiede nel transceiver module da 2.4 GHz estremamente flessibile che lo rende adatto a soddisfare i requisiti per il Bluetooth Low Energy (BLE) e per altri protocolli da 2.4 GHz.

The nRF52840 Dongle hardware consists of the board (PCA10059).

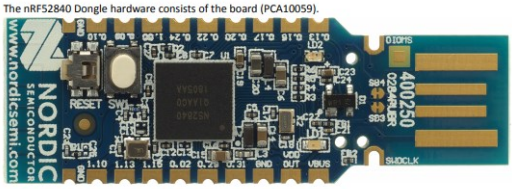


Figure 1: nRF52840 Dongle hardware (PCA10059) front

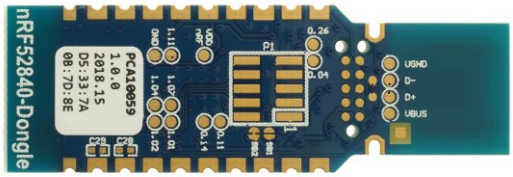


Figure 2: nRF52840 Dongle hardware (PCA10059) back

[14][16][17][18]

4.2 Periferiche della Nordic

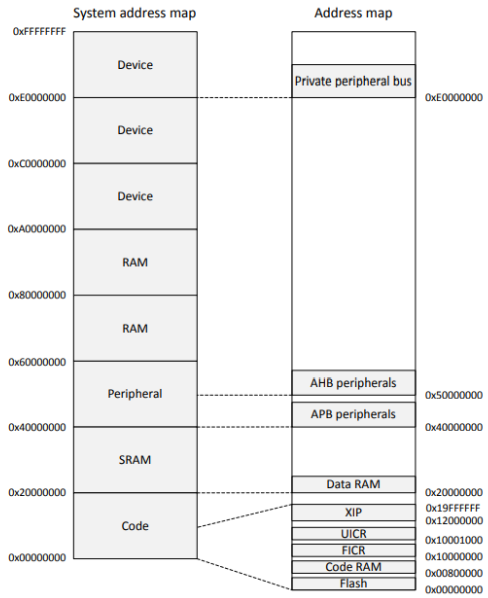
4.2.1 Easy-DMA

Si tratta di un modulo finalizzato a semplificare l'accesso diretto alla memoria; di fatti, è utilizzato da diverse periferiche per godere di un accesso diretto alla RAM, ma non alla memoria flash. Una qualsiasi periferica può implementare multiple richieste Easy-DMA; ad esempio, è possibile dedicare un canale specifico per la lettura di dati dalla RAM alla periferica e contemporaneamente disporre di un secondo canale per la scrittura di dati dalla periferica alla RAM. [15][16]

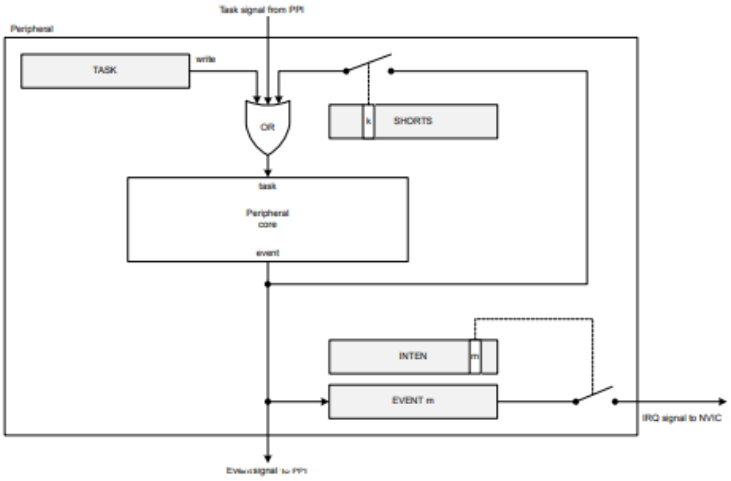
4.2.2 Periferiche

Le periferiche sono controllate dalla CPU mediante la scrittura nei registri di configurazione e di task. Gli eventi relativi alle periferiche vengono segnalati alla CPU negli appositi event registers o interrupt registers (se, in quest'ultimo caso, l'interrupt viene configurato per un dato evento). A ogni periferica è assegnato un indirizzo

con a disposizione 0x1000 bytes di memoria, che corrisponde ad un registro di 1024 x 32 bit. Da notare che sussiste una relazione specifica tra ID della periferica e l'indirizzo di base, ad esempio la periferica TWIM presenta un ID pari a 3 e come indirizzo 0x400030000. Inoltre, alcune periferiche possono condividere lo stesso ID (come TWIM e SPI), il che impone alcune limitazioni quali l'impossibilità delle periferiche di essere utilizzate contemporaneamente o un protocollo da seguire per il passaggio nell'utilizzo di una periferica ad un'altra che condividono lo stesso ID. La maggior parte delle periferiche è dotata di un registro di ENABLE e i registri delle periferiche devono essere configurati prima di abilitare la periferica corrispondente, salvo alcune eccezioni specificate nei datasheet del dispositivo. In particolare, le periferiche devono essere abilitate prima di ricorrere all'utilizzo di tasks ed events. [15][16]



Si considerino ora le connessioni tra periferica e CPU e come esse sono strutturate nel dispositivo della nordic. Ogni periferica è connessa ad un task register, shortcut register, interrupt enable register ed event register, come si evince dalla struttura seguente. Inoltre, ogni periferica può essere connessa con il modulo PPI sia in ingresso che in uscita.



I tasks sono utilizzati per innescare delle azioni sulla periferica, ad esempio per assumere uno specifico comportamento. Una singola periferica può implementare diversi tasks dove ogni task presenta un registro

separato all'interno dell'insieme dei task registers della periferica in questione. Un task viene abilitato nel momento in cui il firmware dispone la scrittura di un 1 sul task register corrispondente, o quando la periferica stessa o qualsiasi altra di esse eseguono il toggle del corrispondente segnale di task. Gli events sono impiegati per comunicare alle periferiche e alla CPU il verificarsi di una determinata condizione, per l'appunto un evento; nello specifico, un esempio di evento può essere fornito da un cambiamento dello stato di una periferica (transizione high to low o low to high del pin GPIO). In particolare, quest'ultima può generare eventi multipli, ognuno di questi aventi un registro separato nell'insieme degli events registers della periferica. La generazione di un evento è associata al toggle da parte della periferica del corrispondente segnale di evento e all'aggiornamento dell'event register; in questo modo viene riportata nel registro la generazione di un evento. Per "pulire" l'event register occorre necessariamente scrivervi uno '0'. Un'altra funzionalità messa a disposizione dall'hardware della nordic consiste nelle shortcut; si tratta di una connessione diretta tra un evento ed un task all'interno della stessa periferica. Se la shortcut viene abilitata, il task associato sarà automaticamente innescato quando l'evento collegato ad esso attraverso la shotcut si verificherà. Utilizzare una shortcut equivale essenzialmente a eseguire la medesima connessione tra evento e task fuori dalla periferica e attraverso il PPI. Tuttavia, il ritardo implicato dall'utilizzo della shortcut risulta essere sicuramente inferiore di quello dovuto all'uso della PPI. Da notare che le shortcut sono predefinite, il che significa che le connessioni non possono essere configurate dal firmaware. Ogni shortcut deve essere abilitata o disabilitata individualmente attraverso lo shortcut register, un bit per ciascuna shortcut, con un massimo di 32 shortcuts per ogni periferica. Infine, tutte le periferiche supportano gli interrupts generati da eventi. Ad ogni periferica è associato esclusivamente un interrupt e il numero del'interrupt segue dall'ID della periferica. Ad esempio, la periferica il cui ID è pari a 3 è connessa all'interrupt numero 3 nel Nested Vectored Interrupt Controller (NVIC). Utilizzando i registri INTEN, INTENSET e INTENCLR, ogni evento generato da una periferica può essere configurato per abilitare un interrupt su quest'ultima. In particolare, possono essere abilitati eventi multipli per generare interrupts simultaneamente; dunque, per determinare la fonte dell'interrupt è necessario accedere all'event register. Per quanto riguarda il reset dell'interrupt occorre tener presente la seguente considerazione: il reset di un interrupt, ottenuto scrivendo uno 0 sull'event register, o disabilitare un interrupt ricorrendo al registro INTENCLR, può richiedere fino a 4 cicli di clock della CPU affinché abbia effetto. Ciò significa che un interrupt può avvenire immediatamente, anche se non si è verificato nessun nuovo evento, nel caso in cui il programma esca dall'interrupt handler dopo che l'interrupt sia resettato o disabilitato, ma prima che siano trascorsi 4 cicli di clock. Dunque, per evitare che si verifichi un interrupt prima che un nuovo evento avvenga, il programma deve eseguire una lettura da uno dei registri della periferica, nello specifico, l'event register che è stato resettato o il registro INTENCLR utilizzato per disabilitare l'interrupt. In sintesi, il verificarsi di un evento su una determinata periferica determina la generazione di un interrupt (se abilitato) e la shortcut (se anche questa è stata opportunamente abilitata) collega tale evento al task corrispondente che determina l'azione di risposta. [15][16]

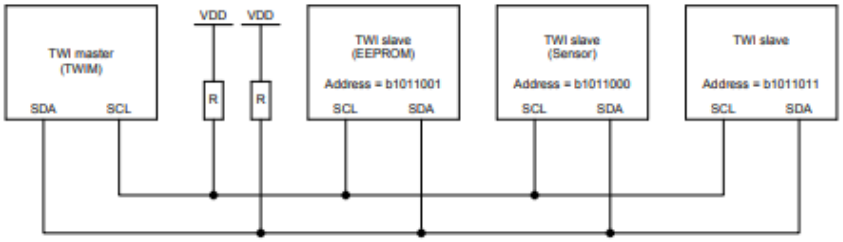
4.2.3 GPIO task and event

Il modulo GPIO tasks ed events (GPIOTE) fornisce le funzionalità per l'accesso ai pin GPIO; in particolare, ogni canale GPIOTE può essere assegnato al singolo pin. Se viene imposto un blocco al modulo GPIOTE, si abilitano i GPIO a generare eventi sul cambiamento di stato del pin che possono essere utilizzati per eseguire tasks attraverso il modulo PPI. Inoltre, un GPIO può anche essere guidato per modificare stato sugli eventi di sistema mediante PPI. In condizioni di System ON o System OFF, è possibile effettuare l'identificazione del cambiamento di stato del sistema in low power. Il numero di canali GPIOTE risulta essere in totale di 8, mentre in ciascun canale GPIOTE possono essere utilizzate fino a tre tasks per realizzare le operazioni di scrittura su un pin. Di questi tre, due sono stabiliti di default (SET, CLR), mentre uno (OUT) è configurabile per eseguire una delle seguenti operazioni: set, clear, toggle. Inoltre, un evento può essere generato in ogni canale GPIOTE da una delle seguenti condizioni di input: rising edge, falling edge o entrambe. Il modulo GPIOTE presenta diversi tasks ed eventi che possono essere configurati per operare sui singoli pin GPIO; i tasks enunciati precedentemente (SET, CLR, OUT) possono essere utilizzati per scrivere sui singoli pin, mentre gli eventi (IN) possono essere generati da cambiamenti rilevati su questi pins. In particolare, si ha che il task SET assegnerà un livello logico alto al pin selezionato in CONFIG[ ].PSEL, mentre il task CLR gli imporrà un valore logico basso.

L'effetto del task OUT sul pin è configurabile in CONFIG[n].POLARITY e, come detto precedentemente, può settare il pin alto, basso o eseguirne il toggle. I tasks e gli eventi sono invece configurabili attraverso i registri CONFIG[n]. Ogni volta che vengono configurati i tasks SET, CLR, OUT e gli eventi IN viene associata a tale configurazione un CONFIG register. Dopo che la configurazione dei registri CONFIG[n] è stata effettuata, il valore di uscita del pin sarà aggiornato solo dal modulo GPIOTE; il che significa che il valore dell'uscita del pin indicato nel GPIO verrà ignorato finché il pin rimarrà sotto il controllo del GPIOTE. Dunque, scrivere su un pin come se fosse un normale pin GPIO non avrà alcun effetto. Solo nel momento in cui il GPIOTE viene disconnesso dal pin quest'ultimo assumerà i valori di uscita e di configurazione specificati dal GPIO. Da notare, infine, che se si verificasse l'innesco di multipli tasks in un singolo canale simultaneamente, questi rispettano il seguente ordine di precedenza: OUT, CLR, SET. [15][16]

4.2.4 Two-Wire Interface Master (TWIM)

Si tratta di un'interfaccia two-wire e half-duplex (trasmissione bidirezionale alternata) in cui il master può comunicare con diversi dispositivi slave connessi allo stesso bus. Le caratteristiche chiave del TWI master riguardano: la compatibilità con l'interfaccia IIC, baud rate supportati pari a 100, 250, 400 kbps, clock stretching supportato e EasyDMA. Nello specifico, l'nrf52840 supporta modalità a 100KHz e 400 KHz e nel caso del progetto in esame si ricorrerà a quest'ultima. In breve, l'interfaccia two-wire permette di comunicare attraverso due linee (SCA e SCL) in maniera bidirezionale interconnettendo fino a 127 dispositivi. I GPIO utilizzati per l'interfaccia possono essere scelti tra qualsiasi GPIO del dispositivo e possono essere configurati indipendentemente; nel nostro caso si è scelto di collegare sul pin P0.29 la data line e sul pin P0.31 la linea di clock entrambi della porta 0. Da notare inoltre, che si tratta di una comunicazione open-drain, ovvero sono necessari dei resistori di pull-up che saranno connessi ad entrambe le linee di connessione per garantire il livello logico alto di entrambe. [15][16]

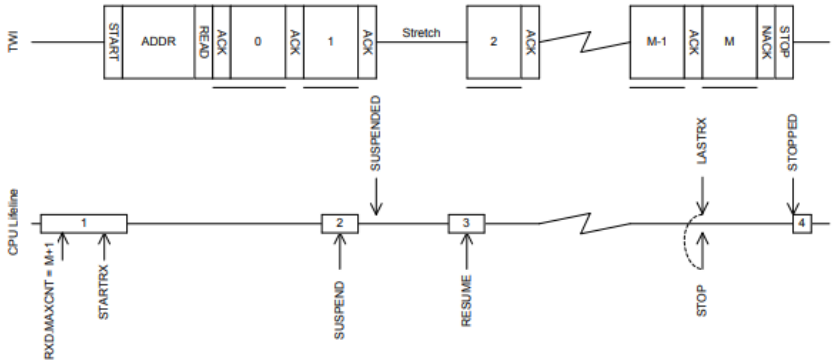


L'inizio della trasmissione viene definito da parte del master innescando uno dei seguenti task: STARTTX o STARTRX; a questo punto i tasks precedenti non dovrebbero essere triggerati di nuovo, prima che la trasmissione non sia terminata. Quest'ultima condizione si verifica quando lo STOP task viene innescato; in tal caso, il master genererà uno STOPPED event nel momento in cui la trasmissione viene interrotta a seguito del task STOP. Infine, se viene generato un NACK da parte dello slave, il master genererà un ERROR event. Il master implementa l'EasyDMA in modo da accedere alla RAM (leggendo e scrivendo dati) senza coinvolgere la CPU; abbiamo quindi che la periferica TWIM implementa due canali EasyDMA: Per quanto concerne i registri relativi a tale periferica, di rilevanza sono sicuramente TXD:PTR e RXD:PTR che corrispondono ai buffer ovvero ai data pointer rispettivamente trasmessi e ricevuti; affinché non si verifichi una possibile condizione di errore devono puntare ad una regione della RAM. I registri .PTR enunciati precedentemente e quelli .MAXCNT che contengono il numero di bytes ricevuti o inviati sono a doppio buffer. Questi possono essere aggiornati e preparati per la prossima trasmissione RX/TX immediatamente dopo aver ricevuto l'evento RXSTARTED/TXSTARTED innescato dal task STARTRX /STARTTX. L'evento STOPPED indica che l'EasyDMA ha concluso i suoi accessi al buffer in RAM. Si consideri ora una sequenza di lettura del master:

1. Viene innescato il task STARTRX;

2. Il master genera una start condition sul bus TWI;
3. Segue l'indirizzo dello slave e il bit di lettura (1) o scrittura (0);
4. L'indirizzo inviato dal master corrisponde a quello dello slave che risponde con un bit di ACK (0) o NACK (1);
5. Lo slave invia i dati al master usando il clock generato dal master;
6. I dati inviati vengono memorizzati nella RAM nell'indirizzo specificato da RDX.PTR e al termine di ogni byte ricevuto il master risponderà con un ACK;
7. La lettura termina nel momento in cui il master risponderà con un NACK al byte di dati ricevuto.

Da notare che il master non interrompe la trasmissione quando il buffer della RAM risulta essere pieno o nel momento in cui viene riscontrato un errore; il task STOP deve essere innescato attraverso l'uso di shortcut o della PPI o ancora da parte del software per gestire le condizioni di errore.

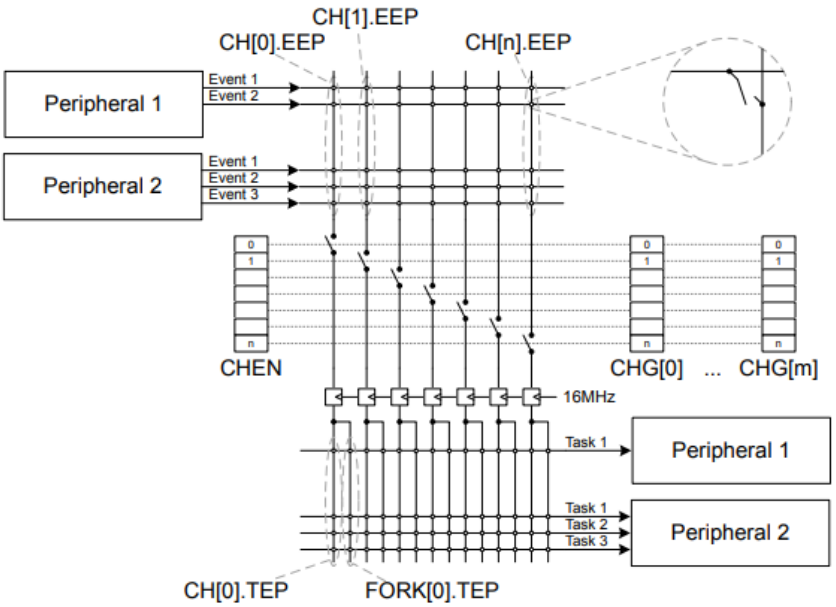


Per quanto concerne invece la sequenza di scrittura del master:

1. Il master genera il STARTTX task;
2. Il master genera una start condition sul bus TWI;
3. Segue l'indirizzo dello slave e il bit di lettura (1) o scrittura (0);
4. L'indirizzo inviato dal master corrisponde a quello dello slave che risponde con un bit di ACK (0) o NACK (1);
5. Il master invia i bytes di dati collocati nel buffer della RAM il cui indirizzo è indicato da TXD.PTR; ogni byte sarà seguito da un bit di ACK o NACK da parte dello slave.
6. La scrittura termina quando viene innescato il task STOP durante la trasmissione dell'ultimo byte. [15][16]

4.2.5 Programmable Peripheral Interconnect (PPI)

Si tratta di un modulo che permette alle periferiche di interagire in maniera autonoma tra di loro ricorrendo ai tasks e agli eventi in maniera del tutto indipendente dalla CPU. La PPI fornisce una sincronizzazione precisa tra le periferiche in modo da soddisfare i vincoli applicativi che sussistono in un'applicazione in tempo reale oltre ad eliminare la necessità di ricorrere alla CPU per implementare attività o comportamenti che possono essere semplicemente programmati e predefiniti.



La PPI fornisce quindi un meccanismo per innescare in maniera automatica un task in una periferica come diretto risultato di un evento che tuttavia si è verificato in un'altra periferica; in questo si nota sicuramente la diversità rispetto alla funzionalità offerta dalle shortcuts. Dunque, il canale della PPI permette il collegamento tra un task ed un evento. Il canale PPI è costituito di tre registri di end point, un registro EEP (event end point) e due registri TEP (task end point); il singolo task relativo ad una periferica è connesso ad un registro TEP ricorrendo all'indirizzo del task register associato al task. In maniera del tutto analoga, un evento relativo ad una periferica è connesso ad EEP register attraverso l'indirizzo dell'event register riferito a quell'evento. In sintesi, l'utilità del modulo PPI risiede nella funzionalità di connettere semplicemente un task ad un evento in modo autonomo e sincrono; ad esempio il verificarsi di un evento sulla periferica GIOTE (pressione del pulsante) viene collegato ad un task sulla periferica GPIOTE (toggle del led) senza che venga chiamata in causa la CPU. La PPI presenta un totale di 32 canali, 20 dei quali possono essere assegnati a qualsiasi EEP e TEP registers.[15][16]

5. Fasi di sviluppo

5.1 Materiale utilizzato

Per la realizzazione del codice si è ricorso al Software Development Kit (SDK) messo a disposizione dalla Nordic Semiconductor. Quest'ultimo contiene librerie, progetti di test ed esempi utili per lo sviluppo di firmawares adatti a diverse applicazioni e dispositivi. Si procede quindi con il download dell'ultima versione aggiornata (nel nostro caso la 17.0.2) dell'SDK al seguente link: https://www.nordicsemi.com/Software-and-tools/Software/nRF5-SDK/Download. In particolare, per lo sviluppo del codice in esame sarà necessario il SoftDevice S140, contenuto all'interno della cartella dell'SDK; si tratta di un Bluetooth® Low Energy protocol stack per Soc quali nRF52811, nRF52820, nRF52833 e nRF52840 che fornisce librerie e firmaware per lo sviluppo di applicazioni che sfruttano il Bluetooth e dunque notevolmente indicato per la realizzazione del progetto in questione. Come ambiente di sviluppo, per compilare e modificare agevolmente il codice si è utilizzata la IDE Eclipse, scaricabile al seguente link: https://www.eclipse.org/downloads/. Si consideri ora il contenuto dell'SDK nello specifico; quest'ultimo, in particolare quello della serie nRF5x, è fornito di Makefile, da utilizzare con la toolchain GNU Tools for ARM Embedded processors, oltre che di librerie, pacchetti integrati e convalidati con il compilatore GCC e altri strumenti necessari per lo sviluppo di software. Un Makefile è un file di testo contenente istruzioni su come creare e collegare (o compilare) un set di file di codice sorgente [19]; conseguentemente, un Makefile project è, come suggerisce il nome, un progetto gestito da Makefile e che può quindi essere compilato anche al di fuori dell'ambiente di sviluppo Eclipse utilizzando strumenti esterni. Ad esempio, un programma make legge il Makefile e richiama un compilatore, un linker e possibilmente altri programmi per creare un file eseguibile. [19] Di fatti, le istruzioni di compilazione per un Makefile project non

sono influenzate dalle impostazioni del progetto configurate nell'IDE, nel nostro caso in Eclipse. Segue necessariamente che se si vuole gestire la build, occorre modificare il Makefile direttamente. Si procede dunque nell'eseguire i passaggi necessari per integrare efficacemente i Makefile project forniti dall'SDK con Eclipse; in questo modo, sarà possibile avere a disposizione sia i benefici offerti da un IDE (che comprende un editor, un compilatore, un debugger ed un code browser in un singolo ambiente grafico) sia la grande flessibilità dei Makefile. [20][21]

5.2 Preparazione dell'ambiente di sviluppo

Per raggiungere il risultato descritto precedentemente si è seguito il procedimento che verrà esposto di seguito e che è reperibile, in maniera dettagliata, al seguente link: https://devzone.nordicsemi.com/nordic/nordic-blog/b/blog/posts/development-with-gcc-and-eclipse. In via preliminare, occorre scaricare i seguenti file:

- La toolchain GNU Tools ARM Embedded processors (che include il compilatore) al seguente link: https://developer.arm.com/tools-and-software/open-source-software/developer-tools/gnu-toolchain/gnu-rm/downloads;
- il plug-in GNU MCU build tools, necessario per compilare progetti basati su Makefile, al seguente link: https://github.com/xpack-dev-tools/windows-build-tools-xpack/releases/tag/v2.10-20180103;

Ora, è necessario assicurarsi di aggiungere il seguente percorso nella sezione path delle variabili d'ambiente di Windows: C:\...\GNU Arm Embedded Toolchain\9 2020-q2-update\bin; ciò rende possibile eseguire gli eseguibili della toolchain da qualsiasi directory utilizzando il terminale. Infine, è possibile avviare Eclipse in cui occorre effettuare le seguenti operazioni preliminari; nel menù di Eclipse:

- *Help → Eclipse Marketplace* scaricare Eclipse Embedded 5.2.1;
- *Window → Preferences → MCU → Global Build Tools Path* aggiunger il percorso C:\...\GNU MCU Eclipse\xpack-windows-build-tools-2.12.2\bin;
- *Project → Properties → MCU → ARM Toolchain Path* aggiungere il percorso C:\...\GNU Arm Embedded Toolchain\9 2020-q2-update\bin.

A questo punto, per compilare correttamente il codice è necessario impostare i percorsi nei file contenuti all'interno della cartella dell'SDK scaricato inizialmente: makefile.windows e makefile.posix in C:\...\nRF5_SDK_17.0.0_9d13099\components\toolchain\gcc; in particolare, Il makefile.windows deve contenere il percorso degli GNU Tools:

```
GNU_INSTALL_ROOT := C:/Program Files (x86)/GNU Arm Embedded Toolchain/9 2020-q2-update/bin/
GNU_VERSION := 9.2.1
GNU_PREFIX := arm-none-eabi
```

Mentre nel makefile.posix assicurarsi di avere:

```
GNU_INSTALL_ROOT ?= /usr/local/gcc-arm-none-eabi-7-2018-q2-update/bin/
GNU_VERSION ?= 7.3.1
GNU_PREFIX ?= arm-none-eabi
```

5. 3 Fase di building

A questo punto, sarà possibile provare ad eseguire la build di uno degli esempi contenuti all'interno dell'SDK; da terminale occorrerà quindi spostarsi nella directory dov'è contenuto il progetto (ad esempio, nel caso del progetto denominato blinky: <SDK>/examples/peripheral/<board name>/blank/armgcc/) e digitare il comando make. GNU make dovrà iniziare la build ricorrendo al Makefile e riportare il risultato nell'apposita _build directory. Ora è possibile creare un nuovo progetto effettuando le opportune modifiche a partire da uno di quelli di esempio e ricordandosi di inserire le librerie utilizzate che devono essere opportunamente compilate. Si tratta quindi di aggiungere sia i file.c che gli header.h necessari nelle apposite sezioni del Makefile. Alcune parti del codice inoltre potrebbero non essere abilitate, ma necessarie alla compilazione, perciò se mancano, bisogna andare nella cartella dove è contenuto il progetto da terminale e digitare il comando: make sdk_config. Si aprirà una finestra in cui sarà possibile abilitare le parti di codice mancanti inserendo una spunta sui file che ci

interessano. Salvare tutto e provare a ricompilare il file in Eclipse. Questo è un procedimento che occorrerà effettuare ripetutamente nella modifica e nella realizzazione del firmaware; ogni volta che, aggiungendo o modificando porzioni di codice, si ricorrerà a funzioni o strutture le cui definizioni sono collocate in file non indicati nel Makefile. In ogni caso, questa mancanza sarà segnalata in fase di compilazione. Infine, occorre notare che, affinché l'esecuzione del codice venga eseguita correttamente in Eclipse, è necessario assicurarsi che in *Project →Properties → C/C++ Build* non sia presente la spunta relativa al comando "generate Makefiles automatically".

6. Codice

6.1 Panoramica

L'obbiettivo del firmaware consiste essenzialmente nel rendere il più accessibile ed efficiente possibile il monitoraggio dello stato del caricabatterie e della batteria stessa. Ciò è stato ottenuto segnalando le varie condizioni di carica o eventuali faults del caricabatterie attraverso l'accensione dei leds del microcontrollore e trasmettendo via Bluetooth le misurazioni del convertitore analogico-digitale relative alla tensione sulla batteria, al valore della temperatura e alla corrente di carica. Nello specifico, si è cercato di ottenere una maggiore efficienza possibile, sia energetica che di calcolo, da parte del microcontrollore. A tale scopo l'nrf52840 opererà in polling esclusivamente se il caricabatterie è connesso direttamente all'alimentazione mentre eseguirà una gestione tramite interrupt in caso contrario. Inoltre, il microcontrollore è stato programmato affinché vada in idle dopo un breve intervallo di tempo a seguito dell'interruzione della trasmissione Bluetooth con un dispositivo; in tal caso, il wake-up da suddetta condizione avviene se viene rilevato un interrupt. Da notare che il microcontrollore risulta essere sensibile all'evento di interrupt dovuto al cambiamento di stato sul pin P1.00 collegato al caricabatterie in modo da ricevere l'impulso riconducibile ad un interrupt di quest'ultimo. Infine, l'interfaccia IIC permette all'host di:

- Configurare i registri del caricabatterie che ne regolano il comportamento;
- Effettuare la lettura dei registri di stato e di flag, indispensabile per il corretto monitoraggio del funzionamento del caricabatterie;
- Effettuare la lettura dei registri che contengono i valori delle misurazioni eseguite dal convertitore analogico-digitale.

6.2 Inizializzazione hardware e periferica IIC

Il codice inizia ovviamente con l'esecuzione del main che prevede l'inizializzazione dell'Hardware. Sono quindi inizializzati i GPIOs che includono il led verde (LD1 P0.06 Green) e il led RGB (LD2 P0.08 Red, LD2 P1.09 Green e LD2 P0.12 Blue); i led sono attivi al livello logico basso, ovvero si illuminano se viene scritto uno zero logico ' 0 ' sul pin corrispondente. Successivamente sono configurate le connessioni IIC in modo che i pin 29 e 30 di port 0 corrispondano rispettivamente ai canali SDA e SDL della periferica TWIM. Inoltre, il segnale dell'interrupt generato dal caricabatterie sul canale di output open-drain è associato al pin P1.00. dunque, quando si verifica un cambiamento dello stato del caricatore tale da generare un interrupt, un impulso di 128 µs è inviato all'host sul pin /INT. A questo punto, dopo aver verificato la corretta configurazione delle linee I2C e inizializzato l'ADC dell'nrf52840 si procede a chiamare la funzione charger_probe del file charger.c;

```
if (i2c_available) {
      if (!charger_probe())
            ch_avaiable = true;
}
```

quest'ultima si occupa in generale di gestire le prime trasmissioni e ricezioni IIC attraverso la periferica TWIM con il caricabatterie. Nello stesso file sono anche definite le funzioni i2c_event e i2c_wait che gestiscono la corretta esecuzione delle trasmissioni IIC, in particolare di quella prevista proprio dalla charger_probe; infatti, la prima viene chiamata al termine di una qualsiasi transizione sul bus IIC e assegna alla variabile i2c_finished

il valore logico booleano true una volta terminata la trasmissione o ricezione IIC. In questo modo è possibile uscire dal loop di attesa imposto dalla i2c_wait che viene chiamata subito dopo aver avviato ogni trasmissione.

```
static void i2c_event (nrfx_twim_evt_t const *p_event, void *p_context)
{
	i2c_result = p_event->type;
	i2c_finished = true;
}

static nrfx_twim_evt_type_t i2c_wait (void)
{
	while (!i2c_finished) ; // TODO: sd_app_evt_wait()
	i2c_finished = false;
	return i2c_result;
}
```

Per quanto riguarda la funzione charger_probe, questa inizialmente si occupa della configurazione dei pin SCL e SDA, della frequenza di clock del master e di altri parametri che vengono opportunamente settati attraverso la struttura nrfx_twim_config_t. In seguito, si ha l'inizializzazione del driver TWI grazie alla funzione nrfx_twim_init dove charger è stato definito precedentemente come variabile globale:
`const nrfx_twim_t charger = NRFX_TWIM_INSTANCE(0);` infine, si verifica se la procedura di inizializzazione sia andata a buon fine e si abilita la trasmissione IIC attraverso la periferica TWIM.

```
int charger_probe (void)
{
	// Structure for the TWI master driver instance configuration.
	nrfx_twim_config_t conf = {
		.scl = CHARGER_SCL,
		.sda = CHARGER_SDA,
		.frequency = NRF_TWIM_FREQ_400K,
		.interrupt_priority = APP_IRQ_PRIORITY_HIGH,
		.hold_bus_uninit = true
	};

	nrfx_err_t err = nrfx_twim_init(&charger, &conf, i2c_event, NULL);

    if(err != NRFX_SUCCESS)
    {
      nrf_gpio_pin_write(LED_R_PIN, 0);
      return 1;
    };

	nrfx_twim_enable(&charger);
```

Nel secondo blocco della funzione, la struttura transfer prepara una trasmissione di prova per comunicare l'indirizzo del registro dello slave seguita da una ricezione subito dopo una start condition (TXRX). Nella struttura sono stati quindi inseriti l'indirizzo dello slave (CHARGER_ADDRESS) e i buffer in ricezione e in trasmissione:

```
uint8_t id_reg = 0x6F; //DEVICE_ID Reg, reset 0X35
uint8_t id_val = 0;


//prima trasmissione I2C per verificare l'indirizzo del caricabatterie
nrfx_twim_xfer_desc_t transfer = {
		.type = NRFX_TWIM_XFER_TXRX,
		.address = CHARGER_ADDRESS,
		.primary_length = 1,
		.secondary_length = 1,
```

```
            .p_primary_buf = &id_reg,
            .p_secondary_buf = &id_val
};
```

la funzione I2c_wait, in combinazione della funzione i2c_event, garantisce che non avvengano ulteriori trasmissioni sul bus IIC quando quest'ultimo risulta essere già impiegato nella lettura/scrittura di dati. In sintesi, viene avviata la trasmissione (i2c_finished = false) e la i2c_wait mantiene il valore di i2c_finished a false; al termine di una transizione sul bus IIC, l'esecuzione del codice passa alla i2c_event che assegna alla stessa variabile booleana il valore true causando l'interruzione del loop di attesa della i2c_wait. Infine, l'avvio delle transizioni definite precedentemente avviene tramite la seguente funzione:

```
err = nrfx_twim_xfer(&charger, &transfer, 0);

i2c_wait();
```

Successivamente, viene effettuato un controllo sulla avvenuta lettura del registro corrispondente all'indirizzo del caricabatterie che presenta come sequenza di bit di reset 0x35 (in notazione esadecimale).

```
if (id_val != 0x35) {

      nrf_gpio_pin_write(LED_R_PIN, 0);
}
```

Il blocco successivo della funzione si occupa della configurazione dei registri utili per la gestione del caricabatterie iterando un'operazione di scrittura su diversi registri IIC del caricabatterie operata dalla funzione `nrfx_twim_tx`. In via preliminare, viene inizializzata la matrice config[][2] costituita da N array di lunghezza 2. Per ciascun array, il primo termine rappresenta l'indirizzo del registro, il secondo è la sequenza di bit del registro che deve essere trasmessa per configurare correttamente il caricabatterie secondo le esigenze del progetto. Entrambi i valori dell'array sono espressi in notazione esadecimale.

```
      //CONFIGURAZIONE REGISTRI
      uint8_t config[][2] = {
            {0x35, 0x11}, //ICCTRL0, 00010001 SW reset
            {0x14, 0x02},  //PCHRGCTRL, 00000010

            {0x12, 0x3C}, //VBAT_CTRL, 0111100
            {0x13, 0x64}, //ICHG_CTRL, 1100100
            {0x16, 0x00}, //BUVLO, 00000000
            {0x17, 0x96}, //CHARGERCTRL0, 10010110 (reset = 0x82)
            {0X18, 0xC8}, //CHARGERCTRL1, 11001000
            {0X1D, 0xDA}, //LDOCTRL, 11011010
            {0x35, 0x10}, //ICCTRL0, 00010000
            {0x36, 0x00}, //ICCTRL1, 00000000
            {0X37, 0x40}, //ICCTRL2, 01000000
            {0x19, 0x04}, //ILIMCTRL 00000100

            {0x58, 0x2E}, //ADC_READ_EN, 00101110
            {0x40, 0x83}, //ADCCTRL0, 10000011 (ogni minuto: 0xC3)
            {0x41, 0x00}, //ADCCTRL1, 00000000

            //ADCcomp2 non attivo
            //ADCcomp3 non attivo
            {0x52, Vbat_min_M}, //ADCALARM_COMP1_M, 10000000
            {0x53, Vbat_min_L}, //ADCALARM_COMP1_L, 00000000 (73 mV)
```

```
            //MASK Registers
            {0x07, 0x1E}, //00011110
            {0x08, 0x20}, //00100000
            {0x09, 0x30}, //00110000
            {0x0A, 0x77}  //01110111
      };


nrfx_twim_enable(&charger);

for (unsigned i = 0; i < sizeof config / sizeof *config; ++i) {

                err = nrfx_twim_tx(&charger, CHARGER_ADDRESS, &config[i][0], 2, false);

                if(err != NRFX_SUCCESS)
                {
                     nrf_gpio_pin_write(LED_R_PIN, 0);
                     return 1;
                };

                   i2c_wait();
                   nrf_delay_ms(1);

};
```

Inizialmente, la prima trasmissione avviene sul registro ICCTRL0 ed è finalizzata ad operare il reset del software (SW_RESET) Seguono poi in ordine le seguenti operazioni:

- Il settaggio del range della corrente di carica e il valore della corrente di precarica in modo che siano pari rispettivamente a 1.25 mA e 2.5 mA (PCHRGCTRL Register).
- Il settaggio della tensione della batteria in modo che sia pari a VBATREG = 4.2 V (VBAT_CTRL register).
- Il settaggio della corrente di carica (fast charge current) sia pari a ICHG = 125 mA (ICHG_CNTRL);
- Il settaggio della tensione di under-voltage (V_{UVLO} = 3 V), della transizione da Pre-charge a Fast Charge e della soglia di protezione della batteria da sovra-correnti pari a 1200 mA (BUVLO register).
- Vengono disabilitati sia il safety timer che il watchdog timer; di fatti, si vuole evitare che il watchdog timer scada e determini un reset del caricabatterie quando il microcontrollore entra in idle e non effettua più transizioni IIC sul relativo bus (CHARGERCTRL0 register).
- Il settaggio della Thermal Charge Current Foldback Threshold in modo che sia pari a 80°C, la disabilitazione delle funzioni DPPM e VINDPM e il settaggio della modalità del pin PMID in modo che il sistema sia alimentato dalla batteria o dalla tensione in ingresso, se presente (CHARGERCNTRL1 register).
- Il settaggio della tensione di uscita, in modo che sia pari a VLDO = 2.8 V; viene inoltre settato il bit di enable EN_LS_LDO per abilitare l'uscita LS/LDO (LDOCNTRL register).
- La Ship mode è disabilitata, il global interrupt mask è disabilitato, la carica è abilitata e la tensione sul pin PMID è pari a 4.5 V; (ICCTRL0, ICCTRL1, ICCTRL2 registers).
- Il settaggio della corrente di ingresso limite sia pari a ILIM = 300mA (ILIMCTRL register).

Inoltre, si configura il convertitore analogico-digitale in modo tale che:

- Sia abilitata la misurazione della corrente di carica, della tensione della batteria e del canale TS;
- La conversione avvenga ogni minuto (in fase di sviluppo ogni secondo) e sia impostato il canale ADC per il comparatore 1 relativamente alla tensione della batteria VBAT.

Questo passaggio infatti, permetterà al caricabatterie di generare un interrupt quando la tensione della batteria attraverserà il valore soglia settato dal registro ADCALARM_COMP1; quest'ultimo viene configurato in modo tale che la tensione di soglia sia pari a circa 70 mV e l'allarme (che determina l'interrupt) venga generato nel

momento in cui la VBAT risulta essere minore di quest'ultimo valore. In questo modo sarà possibile monitorare un eventuale scollegamento della batteria dal sistema di alimentazione. Infine, sono settati i registri per comunicare all'host solo gli interrupt del caricabatterie di interesse e mascherare i restanti. In particolare, gli interrupt per cui si innesca una gestione consona risultano essere:

- CHRG_CV interrupt;
- CHARGE_DONE interrupt;
- VIN_PGOOD interrupt;
- VIN_OVP_FAULT interrupt;
- BAT_UVLO_FAULT interrupt;
- TS_COLD, TS_COOL, TS_WARM, TS_HOT interrupts;
- ADC_READY interrupt;
- COMP1_ALARM interrupt;
- TS_OPEN interrupt.

Il ciclo for si occupa di iterare le trasmissioni non effettuando una ricezione immediatamente successiva ad una start condition tramite la funzione nrfx_twim_tx. Nell'ultimo blocco della funzione è inizializzato il GPIOTE a cui è associato l'interrupt; la struttura nrf_drv_gpiote_in_config_t è utilizzata per configurare l'input pin (la specifica false permette di utilizzare il low power clock e quindi di risvegliare l'nrf52840 dallo sleep); infine viene abilitato l'interrupt sul pin CHARGER_IRQ che corrisponde al P1.00 ed è collegato direttamente all'output open-drain del caricabatterie su cui vengono trasmessi gli impulsi di interrupt. Al verificarsi dell'interrupt la gestione viene presa in carico dalla funzione INT_handler.

```
//CONFIGURAZIONE DEGLI INTERRUPTS

ret_code_t err_code; //hold error code

err_code = nrf_drv_gpiote_init(); //initialize the GPIOTE
APP_ERROR_CHECK(err_code); //check for the errors

//create configuration function for input pin interrupt
nrf_drv_gpiote_in_config_t in_config = GPIOTE_CONFIG_IN_SENSE_HITOLO(false);
in_config.pull = NRF_GPIO_PIN_PULLUP;

//initialize the interrupt pin
err_code = nrf_drv_gpiote_in_init(CHARGER_IRQ, &in_config, INT_handler);
APP_ERROR_CHECK(err_code);

//enable interrupt pin
nrf_drv_gpiote_in_event_enable(CHARGER_IRQ, true);

sd_nvic_ClearPendingIRQ(SPIM1_SPIS1_TWIM1_TWIS1_SPI1_TWI1_IRQn);
sd_nvic_SetPriority(SPIM1_SPIS1_TWIM1_TWIS1_SPI1_TWI1_IRQn,2);
sd_nvic_EnableIRQ(SPIM1_SPIS1_TWIM1_TWIS1_SPI1_TWI1_IRQn);

return 0;


}
```

A questo punto, una volta verificata la corretta configurazione del bus IIC, l'esecuzione del codice ritorna nel main in cui vengono inizializzati i timers e le applicazioni restanti. Infine, viene chiamata la funzione btmain definita nel file bluetooth.c; quest'ultima si occupa della inizializzazione e della configurazione del BLE e che termina richiamando il main loop ed eventualmente mandando il dispositivo in idle:

```
    // Enter main loop.
    for (;;) {

        idle_state_handle();
```

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

6.3 Gestione degli interrupts del caricabatterie

Come accennato in precedenza, è la funzione INT_handler che si occupa della gestione degli interrupt in modo da operare differentemente a seconda della causa che ha innescato tale evento; in particolare, si ha che gli eventi in grado di innescare un interrupt del caricabatterie, che si traduce seguentemente in un interrupt del microcontrollore, sono stati elencati precedentemente e corrispondono agli eventi non mascherati nei registri MASKx dalla charger_probe. la funzione INT_handler chiama sequenzialmente:

- prepare_status_request, che effettua una trasmissione seguita da una ricezione per leggere i registri di stato e di flag del caricabatterie;
- read_status, che si occupa dell'accensione dei led in funzione dell'evento che ha generato l'interrupt. Questa a sua volta può chiamare la charge_disable o la charge_enable che effettuano una trasmissione rispettivamente per disabilitare o abilitare la ricarica;
- charger_adc, che effettua una trasmissione seguita da una ricezione per leggere i registri dell'ADC;
- process_adc, che permette la trasmissione via Bluetooth delle misurazioni della conversione analogico-digitale.

```
static void INT_handler (nrf_drv_gpiote_pin_t pin , nrf_gpiote_polarity_t action)
{
	//al verificarsi di un interrupt si ha il toggle del led verde
	nrf_gpio_pin_toggle(LED_GREEN_PIN);

	prepare_status_request ();
	read_status();
	charger_adc();

	process_adc();
	extern int counter;
	counter++;
	if(counter>2 || counter<0)
		counter=0;
}
```

Nello specifico, la prepare_status_request effettua una trasmissione per comunicare l'indirizzo del registro da leggere seguita da una ricezione, (eseguita subito dopo una start condition), per poter leggere i seguenti registri del caricabatterie:

```
//status and flags registers
uint8_t status_val[] = {0x00,0x00,0x00,0x00,0x00,0x00,0x00};
uint8_t status_reg[] = {0x00,0x01,0x02,0x03,0x04,0x05,0x06};

int prepare_status_request (void) {

	nrfx_twim_xfer_desc_t transfer = {
			.type = NRFX_TWIM_XFER_TXRX,
			.address = CHARGER_ADDRESS,
			.primary_length = 1,
			.secondary_length = 7,
			.p_primary_buf = &status_reg[0],
			.p_secondary_buf = &status_val[0]
	};

	nrfx_err_t err = nrfx_twim_xfer(&charger, &transfer, 0);
```

```
	i2c_wait();
	nrf_delay_ms(1);

	if(err != NRFX_SUCCESS)
		{
		   nrf_gpio_pin_write(LED_R_PIN, 0);
		    return 1;
		}

	return 0;
}
```

Segue la funzione read_status che si occupa di elaborare le informazioni provenienti dalla lettura dei registri. In particolare, segnala i diversi stati del caricabatterie attraverso l'accensione dei led secondo la corrispondenza riportata nel commento nonché impone il termine della carica se viene riscontrato un evento di CHRG_DONE o se la batteria risulta essere disconnessa, avvalendosi della funzione charge_disable. Infine, stabilisce il valore delle variabili esterne VIN_PGOOD, BAT_UVLO; TS_OPEN e TS in funzione di ciò che viene dedotto sul comportamento e le condizioni del caricabatterie a partire dalla lettura dei registri. Il valore delle ultime tre variabili in particolare (1 se la condizione relativa è stata rilevata o 0 in caso contrario) permetterà di gestire in polling il lampeggio dei leds.

```
int read_status (void){

	extern int VIN_PGOOD;
	VIN_PGOOD = 1;
	BAT_UVLO = 0;

	// OVERVOLTAGE -> led RED
	// TS OPEN -> led WHITE toggle
	// TEMPERATURE -> led RED + led GREEN toggle
	// VIN NOT GOOD -> led off
	// VIN GOOD -> led WHITE for 1 second
	// CHARGE DONE -> led GREEN
	// Battery disconnected -> led BLUE when VIN GOOD
	// CHARGING -> led RED + led GREEN
	// Battery undervoltage -> led BLUE toggle when VIN GOOD

	// OVERVOLTAGE
	if((status_val[1] & 0x80) != 0x00){
		nrf_gpio_pin_write(LED_R_PIN, 0);

		return 0;
	}

	// TS OPEN (1b0 = VTS < VOPEN)
	if((status_val[2] & 0x01) != 0x00){
			TS_OPEN = 1;

			return 1;
	}else
			TS_OPEN = 0;

	//TEMPERATURE
	if(((status_val[4] & 0x01) != 0x0) | ((status_val[4] & 0x08) != 0x0)){
		//TS Hot RegionEntrydetected or TS ColdRegionEntrynot detected
		if(((status_val[1] & 0x01) != 0x0) | ((status_val[1] & 0x08) != 0x0)){
	//TS Hot Status- VTS < VHOT(charging suspended) or S ColdStatus- VTS>
	//VCOLD(charging suspended)
```

```
                TS = 1;

                return 0;
        }else
                TS = 0;
}

//VIN NOT GOOD
if((status_val[0] & 0x01) != 0x01){
                nrf_gpio_pin_write(LED_B_PIN, 1);
                nrf_gpio_pin_write(LED_G_PIN, 1);
                nrf_gpio_pin_write(LED_R_PIN, 1);

                VIN_PGOOD = 0;
                return 1;
}

//VIN GOOD
if(((status_val[3] & 0x01) == 0x01) && ((status_val[0] & 0x01) == 0x01)){
                nrf_gpio_pin_write(LED_B_PIN, 0);
                nrf_gpio_pin_write(LED_G_PIN, 0);
                nrf_gpio_pin_write(LED_R_PIN, 0);

                VIN_PGOOD = 1;
                nrf_delay_ms(1000);

                nrf_gpio_pin_write(LED_B_PIN, 1);
                nrf_gpio_pin_write(LED_G_PIN, 1);
                nrf_gpio_pin_write(LED_R_PIN, 1);
}


//CHARGE DONE
if((status_val[0] & 0x20) != 0x0){
        nrf_gpio_pin_write(LED_G_PIN, 0);

        charge_disable();
        return 0;
}

//Battery disconnected (ADC COMP1 Threshold)
if((status_val[2] & 0x40) != 0x0){
        nrf_gpio_pin_write(LED_G_PIN, 1);
        nrf_gpio_pin_write(LED_R_PIN, 1);
        nrf_gpio_pin_write(LED_B_PIN, 0);

        BAT_UVLO = 0;
//      charge_disable();
        return 1;

}else{

//Battery undervoltage
if((status_val[1] & 0x10) != 0x0){
        BAT_UVLO = 1;

        charge_enable();
        return 0;

        }else{
                //CHARGING FAST CHARGE (sopra 3 V)
```

```
                    nrf_gpio_pin_write(LED_B_PIN, 1);
                    nrf_gpio_pin_write(LED_G_PIN, 1);
                    nrf_gpio_pin_write(LED_R_PIN, 1);
                    nrf_gpio_pin_write(LED_G_PIN, 0);
                    nrf_gpio_pin_write(LED_R_PIN, 0);

                    BAT_UVLO = 0;
                    charge_enable();
                    return 0;

            }
      }
}

int    charge_disable (void){

      //DISABILITO la ricarica
      nrfx_twim_xfer_desc_t transfer1 = {
                  .type = NRFX_TWIM_XFER_TX,
                  .address = CHARGER_ADDRESS,
                  .primary_length = sizeof id_reg1,
                  .secondary_length = 0,
                  .p_primary_buf = id_reg1,
                  .p_secondary_buf = NULL
      };

      nrfx_err_t err = nrfx_twim_xfer(&charger, &transfer1, 0);

      i2c_wait();
      nrf_delay_ms(1);

      if(err != NRFX_SUCCESS)
                {
                   nrf_gpio_pin_write(LED_R_PIN, 0);
                    return 1;
                }

      return 0;
}
```

Si osservi che la charger_enable rislulta essere del tutto identica alla charger_disable, ad essere differente è esclusivamente i bits che vengono scritti nel registro da configurare per abilitare o disabilitare la ricarica. Inoltre, occorre notare che la read_status stabilisce il valore delle variabili esterne utilizzate per garantire il lampeggio dei leds ed un comportamento del microcontrollore differente quando il dispositivo risulta essere connesso all'alimentazione. Infatti, in quest'ultimo caso, il convertitore analogico-digitale è sempre attivo; dunque per evitare di costringere il device ad inviare interrupt di avvenuto termine di conversione (ADC_READY) che sommergerebbero l'host, il dispositivo non invierà interrupt al termine della conversione. Dunque, la gestione di tale evento deve avvenire, come esposto precedentemente, in maniera differente a seconda che il caricabatterie sia connesso o meno all'alimentazione; nel primo caso il microcontrollore opera le trasmissioni IIC relative alle misurazioni dell'ADC essenzialmente in polling, nel secondo caso si avvale degli interrupt. Viene quindi dichiarata una variabile esterna (VIN_PGOOD) per tenere conto di variazioni da parte dell'alimentazione in entrambi i file charger.c e main.c insieme alle variabili BAT_UVLO, TS_OPEN, TS per il toggle dei leds. Nel caso in cui il caricabatterie riceve potenza direttamente dall'adattatore collegato all'alimentazione (VIN_PGOOD=1), le funzioni charger_adc e process_adc saranno chiamate periodicamente dalla funzione power_timer_handler nel file main.c che gestisce l'ADC dell'nfr52840:

```
static void power_timer_handler (void *p_context)
{
    UNUSED_PARAMETER(p_context);
```

```
    if(VIN_PGOOD){
            counter++;
            if(counter>2 || counter<0)
                  counter=0;

      charger_adc();
      process_adc();

      if(BAT_UVLO){
            nrf_gpio_pin_write(LED_G_PIN, 1);
            nrf_gpio_pin_write(LED_R_PIN, 1);
            nrf_gpio_pin_toggle(LED_B_PIN);
      }
      if(TS_OPEN){
            nrf_gpio_pin_toggle(LED_B_PIN);
            nrf_gpio_pin_toggle(LED_R_PIN);
            nrf_gpio_pin_toggle(LED_G_PIN);
      }
      if(TS){
            nrf_gpio_pin_write(LED_G_PIN, 1);
            nrf_gpio_pin_write(LED_R_PIN, 1);
            nrf_gpio_pin_toggle(LED_R_PIN);
            nrf_gpio_pin_toggle(LED_G_PIN);
      }

    }

      if (calibrated) {
            nrfx_saadc_buffer_convert(adc_buf, sizeof adc_buf / sizeof *adc_buf);
            nrfx_saadc_sample();
      }
}
```

Come si evince dal codice, la power_timer_handler permette anche il lampeggio dei leds volto a segnalare una determinata condizione del caricabatterie. Parallelamente, nel caso opposto (VIN_PGOOD=0), la gestione dell'ADC avviene sempre attraverso l'interrupt di ADC_READY e rimane esclusivamente il lampeggio di LED_GREEN_PIN che segnala il verificarsi di suddetto interrupt ad intervalli programmabili di un secondo o un minuto.

6.4 Gestione dell'ADC

Per il monitoraggio del convertitore analogico-digitale risultano essere fondamentali due funzioni distinte, entrambe definite in charger.c: charger_adc e process_adc; la prima effettua semplicemente una lettura dei registri che contengono le misurazioni dell'ADC che vengono eseguite in tempo reale se VIN_PGOOD = 1 o ad ogni intervallo di tempo fissato (1secondo o 1 minuto) in caso contrario. La seconda permette di inviare tramite Bluetooth tali misurazioni opportunamente convertite in valori reali ricorrendo alla funzione ble_sys_measurement_send definita in ble_system.c.

```
int charger_adc (void){

      nrfx_twim_xfer_desc_t transfer = {
                  .type = NRFX_TWIM_XFER_TXRX,
                  .address = CHARGER_ADDRESS,
                  .primary_length = 1,
```

```
                .secondary_length = 7,
                .p_primary_buf = &adc_reg[0],
                .p_secondary_buf = &adc_val[0]
        };

        nrfx_err_t err = nrfx_twim_xfer(&charger, &transfer, 0);

        i2c_wait();
        nrf_delay_ms(1);

        if(err != NRFX_SUCCESS)
                {
                   nrf_gpio_pin_write(LED_R_PIN, 0);
                     return 1;
                }

        return 0;
}
```

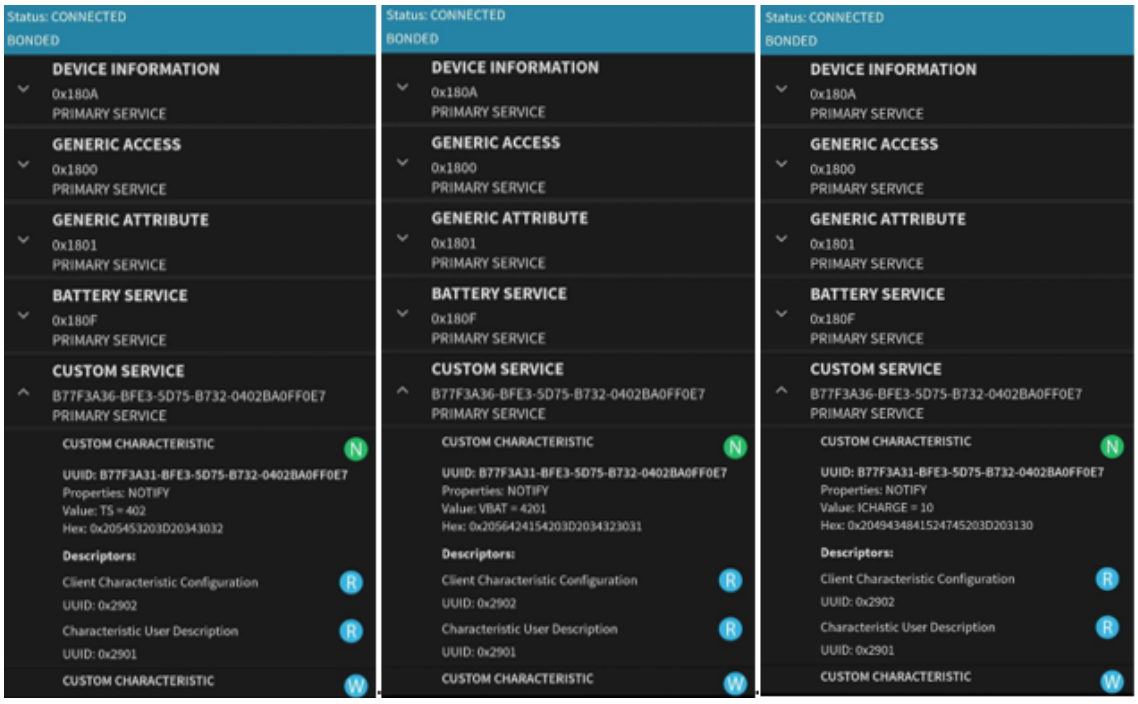
Da notare che la lunghezza del buffer che può essere trasmesso via bluetooth dalla funzione ble_sys_measurement_send è limitata. La variabile counter con l'utilizzo dello switch case permette di visualizzare, al termine di ogni conversione ADC, uno dei tre valori a rotazione tra VBAT, ICHRG e TS. Di fatti suddetta variabile verrà incrementata ad ogni ciclo all'interno della funzione power_timer_handler e resettata una volta raggiunto un valore pari a 2.

```
int process_adc (void)
{
        //CONVERSION & TRANSMISSION
        extern int counter;

        extern ble_sys_t * p_sys;
        uint16_t adcdata_Vbat;
        uint16_t adcdata_Ts;
        uint16_t adcdata_Ichg;
        unsigned char buffer[10];
        int length;

                switch (counter){
                case 0 :
                        //Vbat
                        adcdata_Vbat = ((adc_val[0]*0x100) + adc_val[1]);
                        long var = adcdata_Vbat;
                        int Vbat = ((var*6000)/(pow(2,16)));

                        length = sprintf ((char*)buffer, " VBAT = %d mV", Vbat);
                        ble_sys_measurement_send(p_sys, buffer, length);
                        break;

                case 1 :
                        //Ts
                        adcdata_Ts = ((adc_val[2]*0x100) + adc_val[3]);
                        int Ts = (adcdata_Ts*1200)/((pow(2,16)));

                        length = sprintf ((char*)buffer, " TS = %d", Ts);
                        ble_sys_measurement_send(p_sys, buffer, length);
                        break;

                case 2 :
                        //Ichg
```

```
                    adcdata_Ichg = ((adc_val[4]*0x100) + adc_val[5]);
                    int Ichg = (adcdata_Ichg*100)/(0.8*(pow(2,16)));
                    Ichg_cntrl = Ichg;

                    length = sprintf ((char*)buffer, " ICHARGE = %d", Ichg);
                    ble_sys_measurement_send(p_sys, buffer, length);
                    break;

            default:
            break;
             }

return 1;

}
```

7. Risultati

Collegando il caricabatterie ad una alimentazione idonea, il led RGB rimarrà fisso sul bianco per circa 1 secondo. Dopo di che, lo stesso led segnalerà la condizione di carica attuale o eventuali condizioni di fault. Dopo aver avviato l'applicazione BLE scanner e cliccato su Scan, è possibile selezionare il dispositivo con il quale effettuare il pairing, nel nostro caso l'nrf52480 verrà identificato con "BTSMU" e, attivando la spunta "N", sarà possibile visualizzare le notifiche che corrispondono alle misurazioni dell'ADC:



Intervallati da circa un secondo, vengono visualizzati i valori relativi alla temperatura, alla corrente di carica e alla tensione sulla batteria. Il colore del led RGB del microcontrollore permetterà inoltre di stabilire immediatamente lo stato di carica della batteria o di segnalare eventuali condizioni di fault; riassumendo:

- led rosso fisso → Overvoltage;
- led bianco lampeggiante → TS open;
- led rosso + led verde lampeggiante → temperatura (caldo o freddo);
- led spento → alimentazione scollegata;

- led bianco fisso per circa un secondo → alimentazione idonea;
- led verde → carica completata;
- led blu fisso → batteria disconnessa;
- led rosso + led verde fissi → carica in corso (CV o CC);
- led blu lampeggiante → condizione di undervoltage (sotto i 3 V) della batteria.

Per tutto il tempo in cui il caricabatterie rimarrà collegato direttamente all'alimentazione, la gestione avverrà tramite polling; il che consente di relegare il processo che presuppone il maggior dispendio di energia e di calcolo ad una condizione in cui tale lato negativo sia sostenibile. Da notare infatti che nel progetto definitivo è previsto che il microcontrollore venga ovviamente alimentato dal caricabatterie stesso. Nel momento in cui il dispositivo viene scollegato dall'alimentazione, il led RGB non sarà più attivo, in modo da segnalare l'assenza di un'alimentazione idonea e garantendo inoltre la gestione tramite interrupts. Infatti, durante questa modalità di funzionamento, rimarrà esclusivamente il lampeggio del led verde a segnalare il verificarsi periodico dell'interrupt relativo alla conversione analogico-digitale. Tuttavia, se il caricabatterie dovesse riscontrare una condizione di undervoltage della batteria, l'ADC sarà disattivato (non si verificherà il lampeggio del led verde) e collegando il device all'alimentazione sarà possibile verificare il lampeggio del led blu. Comunque, sia che si consideri la prima (VIN_PGOOD = 1) o la seconda (VIN_PGOOD = 0) modalità, trascorso circa un minuto dal momento in cui l'nrf52480 rileva il verificarsi di un interrupt, il microcontrollore entrerà in idle per garantire un maggior risparmio energetico. Il che significa che verrà interrotta l'accensione del led RGB; per effettuare il wake-up dell'nrf52840 sarà sufficiente forzare un interrupt del caricabatterie, ad esempio scollegando o ricollegando quest'ultimo all'alimentazione.

8. Conclusioni e sviluppi futuri

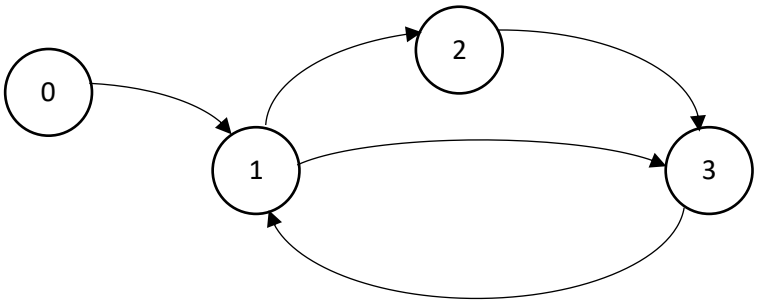
8.1 Implementazione con modulo PPI

Una implementazione che potrebbe garantire un migliore risparmio energetico, soprattutto da parte del microcontrollore che sarà poi alimentato dal caricabatterie stesso, presupporrebbe l'utilizzo della PPI per non coinvolgere direttamente la CPU. In effetti, in una prima versione del codice si era pensato di ricorrere al modulo PPI per gestire gli interrupt del caricabatterie; in particolare, l'idea consiste nell'associare al cambiamento di stato sul pin CHARGER_IRQ un evento in modo tale che sia rilevato quando si verifichi un interrupt sul caricabatterie; tale evento viene poi collegato tramite la PPI ad un task. Quest'ultimo consiste nell'avvio di una trasmissione seguita da una ricezione sul bus IIC, innescata senza coinvolgere la CPU e preparata precedentemente per la lettura dei registri di stato e di flag del BQ25155. L'inizializzazione del modulo PPI viene eseguita dalla charger_probe, immediatamente dopo aver inizializzato Il funzionamento del codice può essere rappresentato da una macchina a stati in cui le uscite sono determinate in funzione dei soli stati correnti (e non anche dagli stati d'ingresso); nel momento in cui termina una transizione sul bus IIC, la funzione i2c_event si occupa di chiamare le funzioni in relazione dello stato corrispondente associato al valore della variabile status_variable:

```
//channel automatically assigned by the SDK
static nrf_ppi_channel_t ppi_channel;

extern int state_variable;

static void i2c_event (nrfx_twim_evt_t const *p_event, void *p_context)
{
	if (state_variable == 0) {
		i2c_finished = true;
		return;
	}

	if (p_event->type == NRFX_TWIM_EVT_DONE) {
```

```
            nrf_gpio_pin_toggle(LED_GREEN_PIN);

            if(state_variable == 1){
                  nrf_drv_ppi_channel_disable(ppi_channel);
                  read_status();
                //state_variable -> 2
            }

            if(state_variable == 2)
                charger_adc();
                //state_variable -> 3

            if(state_variable == 3) {
                  process_adc(); // does not cause a state change - no I²C transactions
involved!
                extern int counter;
                  counter++;
                  if(counter>2 || counter<0)
                        counter=0;
                  prepare_status_request();
                  //state_variable -> 1
            }

      }
}
```

Successivamente, queste funzioni effettuano nuove trasmissioni, questa volta istantanee, subito dopo aver modificato il valore della variabile status_variable e quindi imponendo un passaggio ad uno stato successivo. In sintesi, ciascun passaggio di stato presuppone una transizione IIC. In particolare, vengono definiti i seguenti stati:



- *Stato 0*
  viene chiamata la charger_prob che configura i registri del caricabatterie e iterando una serie di scritture, configura il GPIOTE ed inizializza la PPI; da notare che le precedenti trasmissioni non effettuano un cambio di stato. Infine viene chiamata la prepare_status_request.
- *Stato 1*
  la prepare_status_request prepara appunto la lettura dei registri di stato e di flag del caricabatterie e imposta la variabile di stato a 1; in questo modo il microcontrollore attende che si verifichi un interrupt che innescherà la trasmissione preparata precedentemente. Al termine di quest'ultima la gestione passerà nuovamente alla i2c_event che chiamerà la read_status. Tale funzione determina l'evento che ha innescato l'interrupt e chiamerà opportunamente una delle seguenti funzioni: charge_disable , se si è verificato un evento di CHARGE_DONE e occore disabilitare la carica, o charger_ADC per effettuare la lettura dei registri dell'ADC.
- *Stato 2*
  Viene imposto dalla charge_disable, terminata la trasmissione IIC effettuata da quest'ultima, la funzione i2c_event determinerà il passaggio allo stato 3.
- *Stato 3*

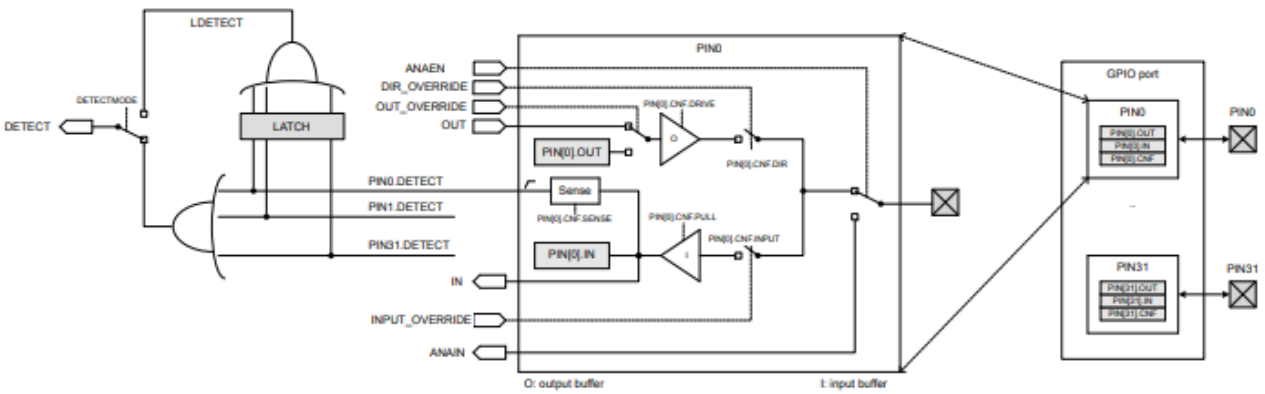
Corrisponde alla funzione charger_ADC; una volta letti i registri del convertitore analogico-digitale la i2c_event chiama la process_adc che si occupa di inviare tramite Bluetooth le misurazioni dell'ADC.

8.2 Problematiche riscontrate

L'implementazione descritta precedentemente ha rilevato diverse problematiche nella sua pratica realizzazione. In particolare, l'ostacolo principale è rappresentato nella difficoltà del conciliare il modulo PPI con un interrupt del caricabatterie basato su impulsi di durata limitata. Per comprendere i punti di debolezza del codice adottato è stato utile semplificare notevolmente il firmware ed aggiungere il supporto USB per poter comunicare in maniera immediata con il computer. In questo modo è stato possibile avere a disposizione un sistema di debugguing attraverso la stampa a video. Il microcontrollore va quindi a creare una periferica USB di tipo CDC ACM che emula una porta seriale e a cui è possibile accedere con un emulatore di terminale da Windows. Nel caso in esame si è ricorso al programma puTTY, scaricabile al seguente link: https://www.putty.org/. Conseguentemente, è stato possibile stampare a video il codice di errore restituito dalla funzione nrfx_twim_xfer che avrebbe dovuto preparare la trasmissione:

```
nrfx_err_t err = nrfx_twim_xfer(&charger, &transfer, NRFX_TWIM_FLAG_HOLD_XFER |
NRFX_TWIM_FLAG_REPEATED_XFER);
```

Tuttavia si è riscontrato che quest'ultima trasmissione restituiva l'errore BUSY, non andando a buon fine e quindi compromettendo inevitabilmente l'intera implementazione del codice. Infatti, per garantirne il corretto funzionamento è indispensabile che non si verifichino interrupt multipli sul pin CHARGER_IRQ; di fatti, al verificarsi di un interrupt inviato dal caricabatterie, la trasmissione preparata precedentemente verrebbe innescata e il codice dovrebbe seguire l'iter descritto dalla macchina a stati. Tuttavia, un secondo evento di interrupt riscontrato prima che la trasmissione preparata abbia termine inficerebbe sul corretto completamento di quest'ultima. Sarebbe quindi necessario mantenere bassa la linea di interrupt del caricabatterie fino al compimento di suddetta transizione IIC realizzando così un interrupt basato su livelli. Per far questo si è pensato di ricorre al registro di LATCH in cui può essere immagazzinato il cambiamento di stato del segnale in ingresso al pin. Come si evince dallo schema della porta GPIO, quando il livello configurato (high o low) viene riconosciuto su uno dei pin e il segnale PINx.DETECT va alto, verrà settato ad 1 un flag del LATCH register. Ad esempio, quando il PIN0:DETECT va alto, il bit 0 del LATCH verrà settato ad 1. Il segnale LDETECT andrà alto se uno o più bit del LATCH saranno pari a 1, mentre andrà basso se tutti i bits del LATCH sono pari a 0; da notare che, come detto precedentemente, il registro LATCH sarà resettato (tutti i bit a 0) quando la CPU scrive esplicitamente un 1 sul o sui bits che devono essere resettati. L'idea è quindi quella di abilitare la connessione tra la porta di DETECT del GPIO e il segnale LDETECT agendo sul registro DETECTMODE in modo da avere un segnale DETECT mantenuto alto finché la CPU non modifichi direttamente il registro di LATCH.[16]



Tuttavia, questa procedura comporta l'inconveniente che, finché la CPU non effettua il reset del latch, il microcontollore non risulta essere sensibile a nessuno degli eventuali interrupts inviati dal caricabatterie in questo lasso di tempo. In tal modo, si rischia di compromettere comunque la gestione efficace del caricabatterie.

Date quindi le problematiche riscontrate e descritte precedentemente che probabilmente richiedono come risoluzione un'implementazione hardware o una gestione più accurata dei registri LATCH, si è deciso di escludere l'utilizzo della PPI e ricorrere al firmware descritto nei capitoli precedenti.

Bibliografia

[1] G. Biagetti - P. Crippa – L. Falaschetti – C. Turchetti, “A multi-channel Electromyography, Electrocardiography and Inertial Wireless Sensor Module Using Bluetooth Low-Energy” in Electronics, 2020, Volume 9, Issue 6.

[2] H. Wang - M. S. Mahmud - H. Fang - C. Wang, “Wireless Health”, Springer, 2016.

[3] V. Shnayder, B. Chen, K. Lorincz, T. Jones, and M. Welsh, “Sensor networks for medical care,” in SenSys ’05: Proceedings of the 3rd International Conference on Embedded Networked Sensor Systems, 2005.

[4] D. Linden – T. B. Reddy, “Handbook of batteries”, New York NY, McGraw-Hill, 2001.

[5] How do Lithium Batteries Work?, Battery University, https://batteryuniversity.com/learn/article/lithium_based_batteries.

[6] D. McNaught - A. Wilkinson. “IUPAC. Compendium of Chemical Terminology”, 2nd ed. (the "Gold Book"), Oxford, ABlackwell Scientific Publications, 1997.

[7] J. R. Dahn - J. C. Burns - D. A. Stevens, “Importance of Coulombic Efficiency Measurements in R&D Efforts to Obtain Long-Lived Li-Ion Batteries” in The Electrochemical Society Interface, 2016, volume 25, number 3.

[8] Charging Lithium-ion, Battery University, https://batteryuniversity.com/index.php/learn/article/charging_lithium_ion_batterie.

[9] How to Prolong Lithium-based Batteries, Battery University, https://batteryuniversity.com/index.php/learn/article/how_to_prolong_lithium_based_batteries.

[10] BQ2515xEVM Evaluation Module, Texas Instruments, https://www.ti.com/lit/ug/sluubv0a/sluubv0a.pdf?ts=1611412088676&ref_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252FBQ25155.

[11] BQ25155I2C Controlled1-Cell 500-mALinearBatteryChargerWith 10-nAShip Mode, PowerPath With Regulated System (PMID) Voltage, ADC and LDO, Texas Instruments, https://www.ti.com/lit/ds/symlink/bq25155.pdf?ts=1611402603783&ref_url=https%253A%252F%252Fwww.google.sk%252F.

[12] Li-ion battery-charger solutions for JEITA compliance, Texas Instruments Incorporated, https://www.tij.co.jp/jp/lit/an/slyt365/slyt365.pdf.

[13] PowerPath control and battery-fed system, https://www.analog.com/media/en/technicaldocumentation/product-information/battery_management_soulutions_powerpath.pdf.

[14] nrf52840 Dongle, Nordic Semiconductor, https://www.nordicsemi.com/-/media/Software-andother-downloads/Product-Briefs/nRF52840-Dongle-productbrief.pdf?la=en&hash=8DDD17CCF2E574021A06A05C71B46C505D492361.

[15] Infocenter, Nordic Semiconductor, https://infocenter.nordicsemi.com/index.jsp.

[16] nrf52840 Product Specification, Nordic Semiconductor, https://infocenter.nordicsemi.com/pdf/nRF52840_PS_v1.1.pdf.

[17] nRF52840 Dongle User Guide, Nordic Semiconductor, https://infocenter.nordicsemi.com/pdf/nRF52840_Dongle_User_Guide_v1.0.pdf.

[18] nrf52840, Nordic Semiconductor, https://www.nordicsemi.com/-/media/Software-and-other-downloads/Product-Briefs/old-versions/nRF52840PBv20.pdf?la=en&hash=516968A8D06D7F41C8B14B6639DD9E14099E2853;

https://docs.rs-online.com/a8d7/0900766b80b23eef.pdf

[19] Microsoft, Creare un progetto makefile C++, https://docs.microsoft.com/it-it/cpp/build/reference/creating-a-makefile-project?view=msvc-160.

[20] R. Mecklenburg, "Managing Projects with GNU Make", Gravenstein Highway North , O'Reilly media, 2004.

[21] Nordic Semiconductor DevZone, Development with GCC and Eclipse, https://devzone.nordicsemi.com/nordic/nordic-blog/b/blog/posts/development-with-gcc-and-eclipse.