ALMA MATER STUDIORUM - UNIVERSITA' DI BOLOGNA
CAMPUS DI CESENA
DIPARTIMENTO DI
INGEGNERIA DELL'ENERGIA ELETTRICA E
DELL'INFORMAZIONE
"GUGLIELMO MARCONI"

CORSO DI LAUREA IN
INGEGNERIA ELETTRONICA PER L'ENERGIA
E L'INFORMAZIONE

PROGETTAZIONE DI UN NODO SENSORE WIRELESS PER IL MONITORAGGIO DELLA BATTERIA DEI DRONI

Elaborato di
Misure Elettroniche

Relatore:
Prof. Marco Crescentini

Presentata da:
Ion Lucian Petrisor

Correlatore:
Dott.ssa Roberta Ramilli

Sessione I
Anno Accademico 2020/2021

Indice

Abstract

Le batterie sono sempre più appetibili negli ultimi anni, essendo impiegate non più solo per applicazioni a bassa potenza, ma risultando una soluzione attuabile anche per applicazioni di potenza come ad esempio il settore automotive e l'accumulo energetico. Fra le più utilizzate vi sono quelle agli ioni di litio per via della loro alta densità energetica e dimensione contenuta. Per aumentarne l'efficienza, nonchè garantire una maggior sicurezza durante l'operabilità, è di primaria importanza ricavare State of Charge (SoC) e State of Health (SoH) in vari scenari, consentendo una caratterizzazione che permetta di sviluppare algoritmi per la gestione delle stesse. Mediante la spettroscopia d'impedenza si ottengono i parametri del modello della batteria, consentendo di ricavare in modo indiretto SoC e SoH.
Al giorno d'oggi come punto di riferimento per la misurazione d'impedenza delle batterie si hanno degli strumenti di laboratorio piuttosto ingombranti che consentono di eseguire un'analisi con la batteria a riposo. Il traguardo da raggiungere per settori in pieno sviluppo, come ad esempio il biomedicale ed il monitoraggio ambientale, che hanno bisogno di sistemi sempre più performanti in termini di consumi e prestazioni, prevede la capacità di effettuare la lettura dell'impedenza durante il normale esercizio.
Il gruppo di ricerca del DEI ha sviluppato un Vector Impedance Analyzer (VIA) miniaturizzato a bassi consumi, che necessita di un collegamento via cavo ad un PC che ne limita l'applicabilità in scenari reali. Per risolvere questa problematica, si effettueranno delle modifiche: il dispositivo verrà alimentato a batterie, e comunicherà con una stazione base mediante un collegamento wireless. Quest'ultimo aspetto sarà oggetto di studio per portare alla realizzazione di un sistema di comunicazione wireless formato da un trasmettitore ed un ricevitore realizzati appositamente per l'applicazione d'interesse. La comunicazione wireless è realizzata mediante il modulo NRF24L01 che si alline con le esigenze di basso consumo energetico ed opera nella banda ISM alla frequenza di 2.4GHz, non necessitando di particolari permessi per l'utilizzo della banda. Il modulo dispone di un gestione automatica dei pacchetti ed è in grado di supportare misure ripetute, potendo operare ad una velocità di trasmissione dati fino a 2Mbps. Dapprima si farà uso dell'ambiente di sviluppo Arduino, per testare il corretto funzionamento della

tratta, per poi passare ad MPLAB, che fornisce una maggior libertà in termini di configurazione del modulo trasmettitore. La realizzazione del progetto è avvenuta con successo, testando il corretto funzionamento della tratta per poi ottenere il nodo sensore, funzionante che trasmette e riceve i dati di prova, portabile a bordo del VIA.

Introduzione

Il continuo incremento delle esigenze economiche e logistiche in termini di area di operabilità di un drone rende insufficiente la sola capacità della batteria per coprire tempi di volo sempre crescenti. Le opportunità connesse all'estensione dell'esercizio spaziano, non solo alle più comuni operazioni, quali il trasporto di carichi e la raccolta dati, ma includono anche nuove possibilità come: le operazioni di salvataggio, l'aggiornamento in tempo reale di incidenti per polizia e ambulanza, nonchè la raccolta dati a scopo agricolo e tanto altro. Un recente progetto finanziato dalla comunità europea e a cui partecipa anche l'Università di Bologna [3], ha identificato tre possibili soluzioni a questo problema:

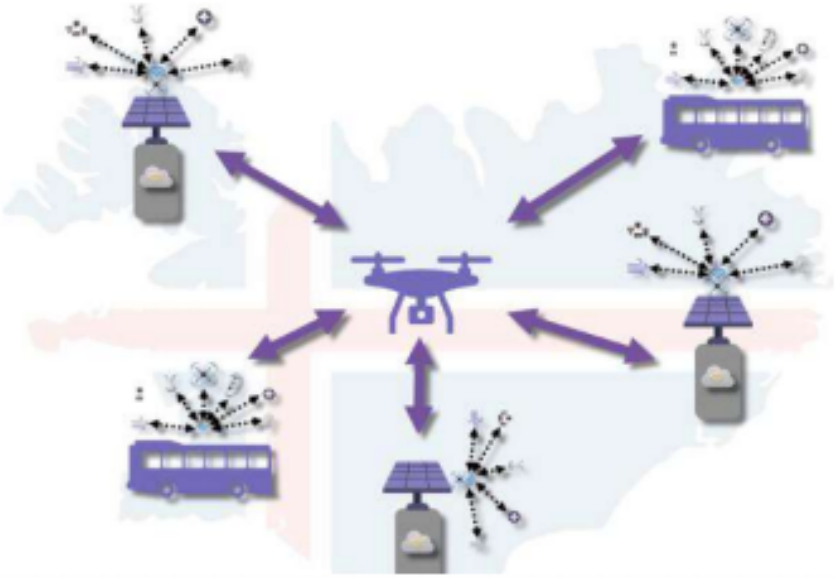

Figura 1: Illustrazione delle soluzioni in fase di studio, figura presa da [3]

- Stazione di ricarica posto sul tettuccio degli autobus
- Stazioni di ricarica statiche o mobili che facciano uso del fotovoltaico

- Efficientamento dei consumi di potenza tramite gestione di tutte le manovre di avvicinamento, atterraggio e stazionamento del drone e miglioramento dell'utilizzo delle batterie

In questo progetto di tesi ci si è concentrati sull'ultimo punto e si è principalmente trattato il monitoraggio dello stato di salute (SoH) e carica (SoC) della batteria tramite tecniche di impedenziometria per poterne allungare la vita utile. La conoscenza di tali dati consente un efficiente utilizzo tramite algoritmi che possano sfruttare la tecnologia al suo limite. L'applicazione, per sua natura, dovrà soddisfare le seguenti caratteristiche:

- accuratezza
- basso consumo
- dimensioni ridotte
- possibilità di effettuare misure ripetute

Il VIA, impiegato per effettuare le misure d'impedenza e riportato nell'articolo [1] e nell'articolo [2] rispetta tutte le caratteristiche al di fuori della dimensione ridotta, che sarà realizzata inserendolo direttamente a bordo del pacco batterie del drone inglobandolo in un unico Application specific integrated circuit (ASIC). Di seguito è stato identificato il modulo NRF24L01 per gestire la comunicazione dei dati acquisiti dal VIA con la stazione base [5], non facendo più uso della comunicazione USB, che prevede un collegamento fisico tramite cavo già presente, ma sostituendola con una wireless. Allo scopo si è realizzato in un primo momento un programma di test, facendo uso dell'ambiente di sviluppo Arduino, nonchè di due schede Arduino Uno per emulare trasmettitore e ricevitore e verificare in questo modo il corretto funzionamento della tratta e delle inizializzazioni da effettuare. Questo ha permesso di ragionare ad un livello logico senza curarsi di come i registri interni del modulo vengano configurati, scaricando il computo sui metodi presenti nelle librerie fornite dall'IDE di Arduino. Successivamente il trasmettitore è stato realizzato utilizzando un dsPIC30F3013 programmato con MPLAB, andando a configurare direttamente i registri in accordo con quanto riportato nel datasheet [5] e mantenedo la stessa configurazione logica del modulo.

Capitolo 1

Vector Impedance Analyzer

1.1 Caratteristiche Principali

Il dispositivo utilizza la tecnica della spettroscopia di impedenza elettrica (EIS) per determinare il valore d'impedenza della batteria. Essa permette di determinare molte informazioni sullo stato della batteria, interpretando correttamente l'impedenza nel piano di Nyquist, oppure mediante i diagrammi di Bode di fase e ampiezza. Tra le principali si ritrovano: la temperatura interna, state of function (SoF), C-rate, SoC e SoH. I sistemi elettrochimici sono notoriamente non lineari, pertanto per non innescare la non linearità, si dovranno adoperare come perturbazioni dei segnali di piccola entità. Le principali soluzioni sono rappresentate dalle perturbazioni in corrente (GEIS) e le perturbazioni in tensione (PEIS). Per via della natura dell'impedenza delle batterie con valori contenuti che si aggirano nell'ordine dei mΩ l'applicazione di segnali in tensione di 5-10 mV porterebbe a delle correnti in uscita elevate. Questo comprometterebbe il rapporto segnale rumore, siccome sotto queste specifiche la maggior parte dei potenziostati in commercio riscontrerebbero difficoltà a mantenere stabile la tensione. D'altro canto l'impiego di una corrente nell'intervallo 0.1-1 A che porti ad un segnale in uscita dell'ordine dei mV risulta alla portata della maggior parte degli strumenti, come riportato dall'articolo [4]. Una possibilità per realizzare lo stimolo è il metodo single-sine, nel quale si effettua una misurazione delle singole frequenze. Questo risulta particolarmente dispendioso in termini di tempo, nelle applicazione in cui si desidera uno spettro fitto, un'ampia banda oppure frequenze particolarmente basse <1 mHz. Nei casi più critici la stazionarietà delle caratteristiche della batteria risulta compromessa per via dei tempi di acquisizione troppo lunghi. La batteria risulta essere infatti, un sistema tempo variante che altera le proprie caratteristiche al variare del tempo. D'altro canto il metodo single-sine risulta molto preciso se paragonato al metodo multi-tone, seppure più lento. In quest'ultimo si utilizza

una perturbazione composta dalla somma di più sinusoidi, di conseguenza la perturbazione è distribuita su più frequenze avendo un rapporto segnale rumore più basso, ma consentendo di effettuare allo stesso tempo diverse frequenze.

1.2 Architettura

Per ridurre i consumi di potenza si è scelto di realizzare lo stimolo sinusoidale impiegando la tecnica di conversione D/A a 1-bit $\Delta\Sigma$ avendo in precedenza campionato la sinusoide con una precisione di 64 bit e salvato il dato nella SRAM. Il segnale in ingresso al sensore viene poi filtrato tramite un filtro passivo passa-banda per evitare che che vi siano presenti disturbi in alta frequenza tipici delle modulazioni $\Delta\Sigma$ e per disaccoppiare la DC del convertitore A/D da quella della batteria. L'accoppiamento capacitivo a lato del A/D separa l'alimentazione del VIA da quella della batteria evitando correnti spurie dovute alle scariche elettrostatiche dei cicuiti a monte del filtro passa banda. L'utilizzo di un filtro passivo crea dei problemi di non-linearità in quanto la frequenza di taglio del filtro dipende dall'impedenza della batteria che si vuole misurare. Inoltre, la risposta d'ampiezza di un filtro passivo non è perfettamente piatta introducendo una dispersione di guadagno in frequenza. Queste non-idealità sono attenuate tramite una specifica procedura di taratura descritta in [2]. La tensione in uscita avrà un'ampiezza molto contenuta per tale motivo si inserisce un amplificatore a basso rumore (LNA). Di seguito è presente un convertitore A/D $\Delta\Sigma$ passa banda che campiona l'uscita del LNA digitalizzandola in due vie parallele per la parte reale ed immaginaria. Utilizzando poi due filtri decimatori di tipo $sinc^3$ si recupera il corretto valore dell'impedenza diviso in parte reale ed immaginaria ciascuna a 16-bit. L'architettura consente di variare la frequenza del segnale di test durante il funzionamento, essendoci una equazione tra la f_s del ADC e del DAC e la frequenza della sinusoide generata. Il legame è definito dalla seguente forumula:

$$f_T = \frac{f_{s,DAC}}{2OSR_{DAC}} \tag{1.1}$$

dove OSR_{DAC} rappresenta il rapporto di sovracampionamento del convertitore D/A $\Delta\Sigma$, f_T è la frequenza dello stimolo sinusoidale ed $f_{s,DAC}$ è la frequenza di campionamento del segnale. A sua volta il rapporto di sovracampionamento è definito come:

$$OSR_{DAC} = \frac{1}{2}\frac{N}{M} \tag{1.2}$$

M rappresenta il numero di periodi della sinusoide presenti nella sequenza B riportata nella figura 1.1. Pertanto la frequenza della sinusoide si può esprimere

mediante la formula:

$$f_T = \frac{M}{N} f_{s,DAC} \tag{1.3}$$

risultando quindi possibile variare la frequenza dello stimolo intervenendo sulla f_s del DAC.

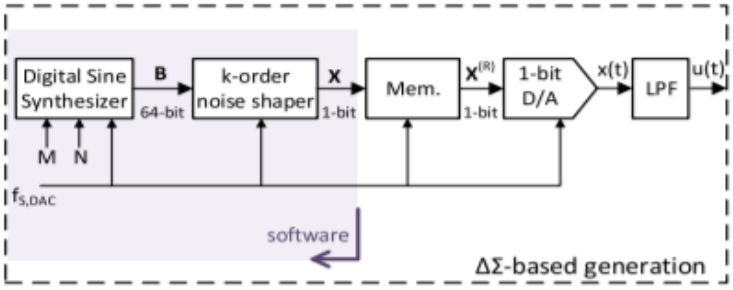

Figura 1.1: Schema a blocchi del VIA, figura presa da [2]

1.3 Modifiche Apportate

Il VIA preso in considerazione negli articoli [1] e [2] necessita alcune modifiche per potere effettuare misurazioni d'impedenza direttament on-line, come in questo caso ove si ha la necessità di impiegarlo per effettuare delle misurazioni durante il volo del drone. In precedenza i dati acquisiti contententi le misurazioni d'impedenza effettuate erano trasmessi tramite un collegamento fisico via cavo USB ad un sorgente MATLAB che gli acquisiva, al contempo l'alimentazione era fornita, mediante stabilizzazione, sempre dalla porta USB. Il campo di applicazione richiesto necessita invece di una trasmissione dei dati wireless e di una alimentazione a batteria. Lo scopo è proprio quello di riuscire a creare il nodo di comunicazione wireless e comunicare i dati dello stato della batteria ad un ricevitore posizionato a terra. Questo viene realizzato utilizzando il microcontrollore dsPIC30F3013 che in fase di test verrà impiegato per comunicare con il modulo NRF24L01 allo scopo di trasmettere i dati acquisiti, ad una ricevente posizionata a terra realizzata con un Arduino Uno. Successivamente la scheda del VIA dovrà essere rivisitata ed il programma sorgente del trasmettitore sviluppato in fase di test inserito all'interno del microcontrollore già ivi presente. Di seguito sarà neccessario effettuare uno studio sulla batteria da utilizzare per alimentare il VIA, altresì si può considerare se esso possa essere alimentato direttamente dalla batteria di cui si desidera rilevare l'impedenza.

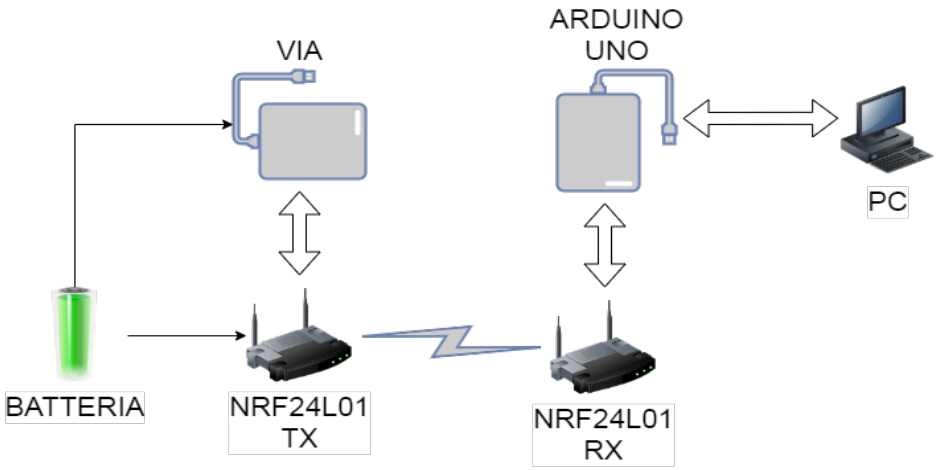

Figura 1.2: Schema modifiche apportate

Capitolo 2

Modulo NRF24L01

2.1 Caratteristiche Principali

Il modulo NRF24L01 è un trasmettitore-ricevitore operante alla frequenza di 2.4 GHz nella banda ISM, costituito da un singolo chip e adatto alle applicazioni di bassa potenza. Il chip è un QFN20 4x4 difficilmente impiegabile in fase di test, è possibile tuttavia trovare in commercio dei moduli già preassemblati che dispongono di un pinout come quello mostrato nella figura 3.1, aventi dimensioni di 29x15 mm. La tensione di alimentazione del modulo è di 3.3 V, essendo però tollerante ai 5 V per quanto riguarda i pin di comunicazione della SPI. I consumi diferiscono a seconda della modalità in cui si trova il modulo , in *power down mode* si ha un consumo di 900 nA, ma si ha un ritardo di 1.5 ms prima di entrare in *standby mode*, dal quale è possibile impostare il modulo come TX/RX. Lasciando invece il modulo in *standby mode* si hanno dei consumi di 22 uA, potendolo farlo operare in TX/RX mode in un tempo di 130 uS. Durante l'invio di pacchetti il caso peggiore per quanto riguarda i consumi è di 11.3 mA, mentre per la ricezione si ha un consumo di 12.3 mA. Per il funzionamento necessita di un microcontrollore e di alcuni componenti passivi dimensionati dal costruttore, come mostrato nella figura 2.1. I moduli preassemblati difatti realizzano su scheda il circuito direttamente fornito dal costruttore, rendendolo più facilmente fruibile.

Il dispositivo è programmabile mediante l'interfaccia seriale SPI, con la quale è possibile configurare tutti i registri necessari per la trasmissione. Ad esempio è possibile configurare uno dei 126 canali in cui operare, oppure la velocità di trasmissione dei dati tra 250 kbps 1 Mbps 2 Mbps. Dispone inoltre di un protocollo per la trasmissione dei dati proprietario, denominato *Enhanched Shockburst* che

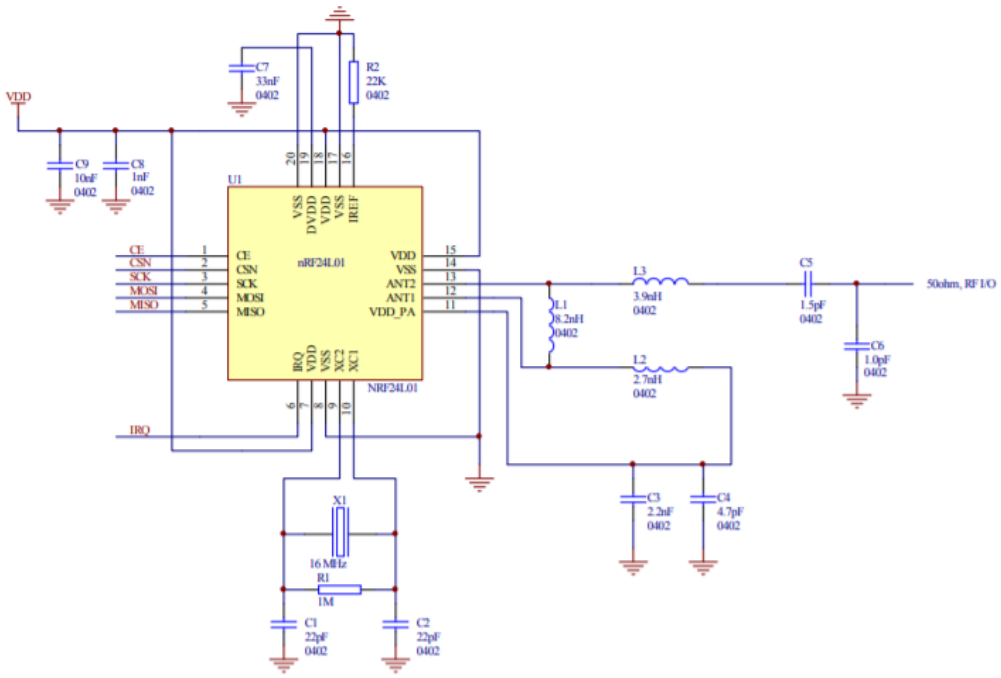

Figura 2.1: Schema elettrico NRF24L01, figura presa da [5]

opera a livello di collegamento dati ed è un protocollo basato su pacchetti, con un formato riportato nelle figure 2.2 e 2.3. Il protocollo include l'assemblamento automatico dei pacchetti, l'acknowledge automatico e la ritrasmissione dei pacchetti. Questo permette di realizzare una comunicazione a basso costo computazionale a lato del microcontrollore, consentendo un significativo miglioramento dei consumi per comunicazioni bidirezionali e unidirezionali, essendo perciò adatto ad essere usato in questo progetto.

Preamble 1 byte	Address 3-5 byte	Packet Control Field 9 bit	Payload 0 - 32 byte	CRC 1-2 byte

Figura 2.2: Esempio di un pacchetto della modalita Enhanched Shockburst, figura presa da [5]

- 1 byte di preambolo
- 3-5 byte di indirizzo
- 9 bit di controllo del pacchetto

- 0-32 byte di dati
- 1-2 byte di CRC

Il byte di preambolo è una sequenza di 0 e 1 che serve al lato ricevitore per sincronizzarsi. I 9 bit di controllo sono a loro volta ripartiti secondo il seguente schema.

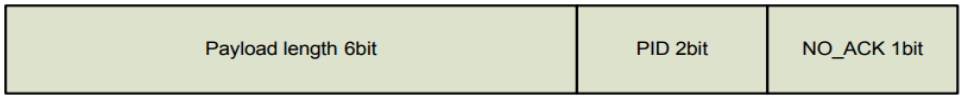

Figura 2.3: Controllo di pacchetto della modalita Enhanched Shockburst, figura presa da [5]

- 6 bit Lunghezza del pacchetto dati
- 2 bit Identificativo pacchetto
- 1 bit Disabilitazione Acknowledge (ACK)

È possibile configurare per una maggior robustezza dei dati ACK ed CRC, per poter attivare il primo è condizione necessaria avere attivato anche il secondo. La disabilitazione del ACK cambia il protocollo di comunicazione che diventa il *ShockBurst* che dispone dello stesso formato di pacchetto, avendo la possibilità di abilitare il CRC fornendo una informazione sulla bontà del dato ricevuto, senza una eventuale ritrasmissione. Sono predisposti inoltre il protocollo per il rinvio automatico dei pacchetti di cui non si è ricevuto l'acknowledge entro il tempo limite oppure non hanno superato il controllo di errore a lato ricevitore. L'impiego del CRC è di primaria importanza per garantiere la bontà dei dati ricevuti, siccome nell'applicazione d'interesse il drone potrebbe operare lontano dalla ricevente, nonchè operare in uno spazio sottoposto alle interferenze degli altri dispositivi presenti. Per evitare che i pacchetti siano persi occorre disporre una conferma della loro ricezione realizzata mediante l'ACK. La ritrasmissione dei pacchetti è altrettanto importante per riuscire a ricevere completamente tutte le acquisizioni d'impedenza effettuate, evitando di avere solo parte dell'intera spazzolata impedenziometrica. Raggiunto il numero massimo di ritrasmissioni infatti, il modulo passerà ad inviare il dato successivo, pur non avendo correttamente inviato il corrente. Occorre perciò effettuare delle prove per definire il tempo massimo neccessario al drone per trasmettere in campo aperto il pacchetto, allo stesso tempo fare uno studio su quante ritrasmissioni siano da effettuare in caso di tempo massimo raggiunto oppure CRC non superato.

2.2 Comandi SPI

I comandi impartiti mediante SPI hanno una lunghezza di 8 bit e prevedono una configurazione dell'interfaccia seriale secondo il *Mode 0*. Tale modalità prevede il clock nello stato idle basso (CPOL=0), i dati campionati sul fronte di salita ed aggiornati sul fronte di discesa (CPHA=0). Ogni comando ha come bit più significativo il primo (MSB), viceversa per i dati dove il primo bit è il meno significativo. Si deve prevedere una transizione da alto a basso del chip select (CSN) per consentire al modulo di comprendere l'invio dell'istruzione. La configurazione della seriale deve essere realizzata sia per la TX che per la RX, su Arduino utilizzando il metodo di inizializzazione e sul dsPIC30F3013 mediante i registri di configurazione della SPI.

2.2.1 Protocollo SPI

Il protocollo Serial Peripherial Intefrace (SPI) è un protocollo di tipo seriale, sincrono di tipo master slave, che consente la comunicazione tra tra un master e uno o più slave. Esso risulta essere full-duplex consentendo in contemporanea la trasmissione e la ricezione di dati. La comunicazione è realizzata comunemente a 4 fili :

- **SCK** Clock Seriale
- **SDI** Serial Data Input
- **SDO** Serial Data Outout
- **SS/CSN** Slave Select, Chip Select in logica negata

Il clock seriale è generato dal master e scandisce l'inizio e la fine della comunicazione, nonchè le tempistiche da rispettare per l'invio e la ricezione di dati. Lungo la linea SDI il master riceve i dati provenienti dallo slave che in quel momento è abilitato alla comunicazione, SDO invece è dedicata all'invio dei dati dal master verso gli slave abilitati. L'abilitazione degli slave si effettua mediante la linea di slave select, tenendo in considerazione che il numero di slave controllabili mediante essa è rapprentato da 2^N, dove N sono il numero di linee neccessarie. Per evitare ambiguità spesso si utilizzano dei nomi diversi per SDI e SDO che diventano rispettivamente master input slave output (MISO) e master output slave input (MOSI). La comunicazione si basa sulla presenza di registri a scorrimento che ad ogni clock scambiano un bit di informazione con l'interlocutore, ricevendo od inviando un dato. A tal proposito si hanno a disposizione due parametri che permettono di regolare la sincronizzazione: Clock Polarity (CPOL) e Clock Phase (CPHA). Se CPOL è impostato a zero il clock nello stato di riposo si trova a livello

logico basso, viceversa a livello logico alto. Mediante CPHA si ha la possibilità di scegliere se il dato debba essere campionato sul fronte di salita o di discesa nello shift register ed aggiornato sull'altro fronte. Ad esempio con CPOL=0, il dato sarà campionato sul fronte di salita se CPHA=0, viceversa sul fronte di discesa se CPHA=1. Normalmente questi dati sono configurabili dal registro del master che configura l'interfaccia SPI, consentendo di comunicare con tutti i dispositivi slave. La linea di slave select non è sempre impiegata, potendo non essere affatto utilizzata nell'ipotesi di una comunicazione con un solo slave.

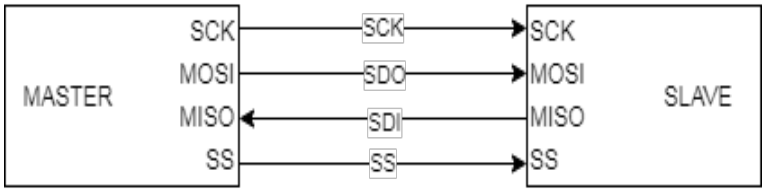

Figura 2.4: Collegamento di un master ed uno slave del protocollo SPI

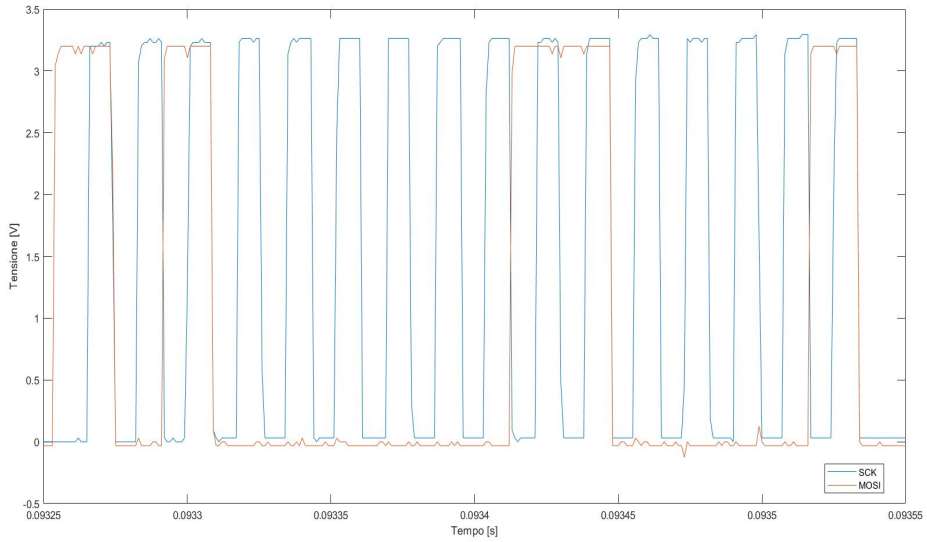

Figura 2.5: Esempio di comunicazione master to slave

2.2.2 W-REGISTER

L'istruzione consente di indirizzare il registro che si desidera scrivere secondo il comando a 8-bit 001X XXXX , ove i cinque bit meno significativi rappresentano

l'indirizzo di memoria in cui il registro è mappato. Successivamente tramite una seconda istruzione di scrittura sulla seriale si devono fornire i valori che si desiderano impostare. Fatta eccezione per il registro contenente l'indirizzo della linea di comunicazione wireless in cui si ha la necessità di fornire 3-5 byte in seguito alla W-REGISTER, tutti gli altri necesitano di una scrittura di soli 8-bit, per la corretta configurazione del registro.

2.2.3 W-TX-PAYLOAD

Il comando binario 1010 0000 viene riconosciuto dal modulo come l'intenzione di trasmettere un pacchetto dati, avente una lunghezza definita nella fase di inizializzazione previa scrittura del registro RX-PW-X. Successivamente vengono forniti i dati con una istruzione di scrittura seriale in modo analogo a quanto fatto per W-REGISTER, tenendo conto che è possibile scrivere un solo byte per volta.

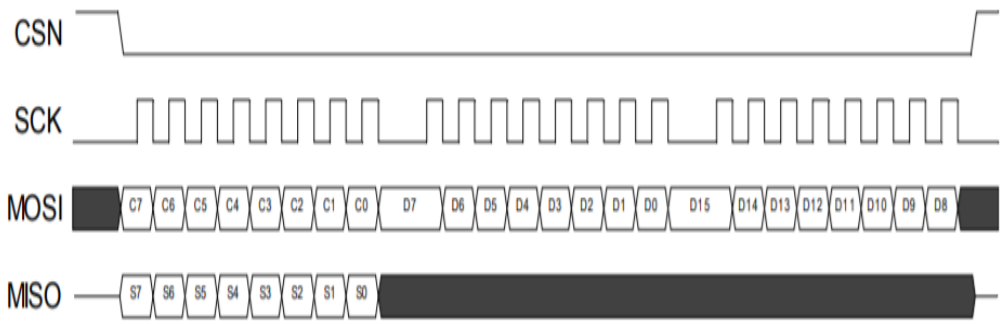

Figura 2.6: Operazione di scrittura SPI, figura presa da [5]

Nella figura 2.6 è possibile osservare un esempio di trasmissione dati, ove Cn rappresentano i bit del comando SPI, Sn i bit del registro STATUS e Dn i bit del dato. La trasmissione comincia con l'invio sulla linea MOSI da parte del master del comando SPI, seguito dal dato che si desidera inviare, suddiviso in blocchi di 8 bit. Nella figura 2.5 che riporta i segnali SCK e MOSI acquisiti a lato trasmettitore con l'oscilloscopio si può osservare come la struttura sia rispettata. I primi 8 bit rappresentano il comando di scrittura 1010 0000, mentre gli altri 8 bit codificano il carattere 'a' che in binario risulta essere 0110 0001.

2.3 Registri di configurazione

Il registro CONFIG permette di impostare il modulo in TX mode o RX mode, oppure impostare in modalità standby-I, standby-II o power down mode. Le ultime tre diferiscono per i consumi che sono rispettivamente decrescenti nell'ordine sopra riportato, ma al contempo necessitano di tempi maggiori per poter passare alla modalità di trasmissione/ricezione di un dato. Si può selezionare la lunghezza del codice di controllo di errore, essendo possibile anche disabilitarlo completamente per avere un overhead minore e trasmettere più dati a discapito della loro affidabilità. I sei bit meno significativi del registro EN-AA pemettono l'attivazione del ACK sulle sei linee dati disponibili, mentre quelli di EN-RXADDR consentono l'attivazione della sei linee dati a disposizione. Le linee dati, contradistinte da un indirizzo univoco, consentono di ricevere dati sullo stesso canale radio e di conseguenza sulla stessa frequenza di fino a 6 dispositivi differenti che li trasmettono, permettendo di realizzare un multiricevitore. La comunicazione avviene una sola linea dati per volta e presume la stessa configurazione di CRC, lunghezza indirizzo, canale radio, velocità dati e guadagno del LNA per tutte le linee. Il registro SETUP-AW è impiegato per selezionare la lunghezza dell'indirizzo di TX e RX da 3-5 byte con la configurazione dei 2 LSB. Il registro SETUP-RETR permette di impostare il tempo massimo da attendere senza aver ricevuto ACK entro cui ritrasmettere, inoltre consente di configurare il numero massimo di ritrasmissioni senza ACK. Il registro RF-CH consente di selezionare uno dei 126 canali per operare tra quelli disponibili per il modulo. RF-SETUP è il registro utilizzato per selezionare la velocità di trasmissione dei dati tra 250 kbps, 1Mbps, 2Mbps. Al contempo ha la funzione di regolare la potenza di uscita in TX mode tra -18dBm e 0dBm. RX-ADDR-P-X è un registro che permette di memorizzare l'indirizzo della X-esima linea, disponendo per la linea 0 di 40 bit nel caso in cui si utilizzino indirizzi da 5 byte. La linea 1 dispone di 40 bit al massimo per la configurazione, condividendo i 4 byte più significativi con le linee 2-5 che differiscono per il solo byte meno significativo. Il registro RX-PW-P-X configura il campo dati del pacchetto impostandolo nel range 1-32 byte lungo la X-esima linea disponibile del modulo.

Registro	*Indirizzo*	*Configurazione*
CONFIG	0x00	TX/RX Mode, CRC Length, Stand-by Mode
EN_AA	0x01	Enable ACK on a Pipe
EN_RXADDR	0x02	Enable Data Pipe
SETUP_AW	0x03	Address Length
SETUP_RETR	0x04	Re-transmission Timeout
RF_CH	0x05	RF Channel
RF_SETUP	0x06	Data Rate
RX_ADDR_P_X	0x0A	Data Pipe 0-5 Address
RX_PW_P_X	0x12	Payload in Data Pipe 0-5

2.4 Diagramma di flusso

Nella figura 2.7 si può osservare lo schema di principio secondo cui il modulo NRF24L01 dovrà essere configurato per il corretto funzionamento. Le operazioni di configurazione del modulo non devono seguire per forza l'ordine riportato, potendo essere eseguite con un ordine arbitrario. Risulta neccessaria tuttavia la configurazione della seriale prima di tutte le altre istruzioni, siccome essa è impiegata per comunicare la configurazione da impartire ai registri. Si noti poi come ad esempio l'abilitazione dell'ACK sia vincolante per quanto riguarda la scelta del CRC, dato che per il suo funzionamento è neccessario.

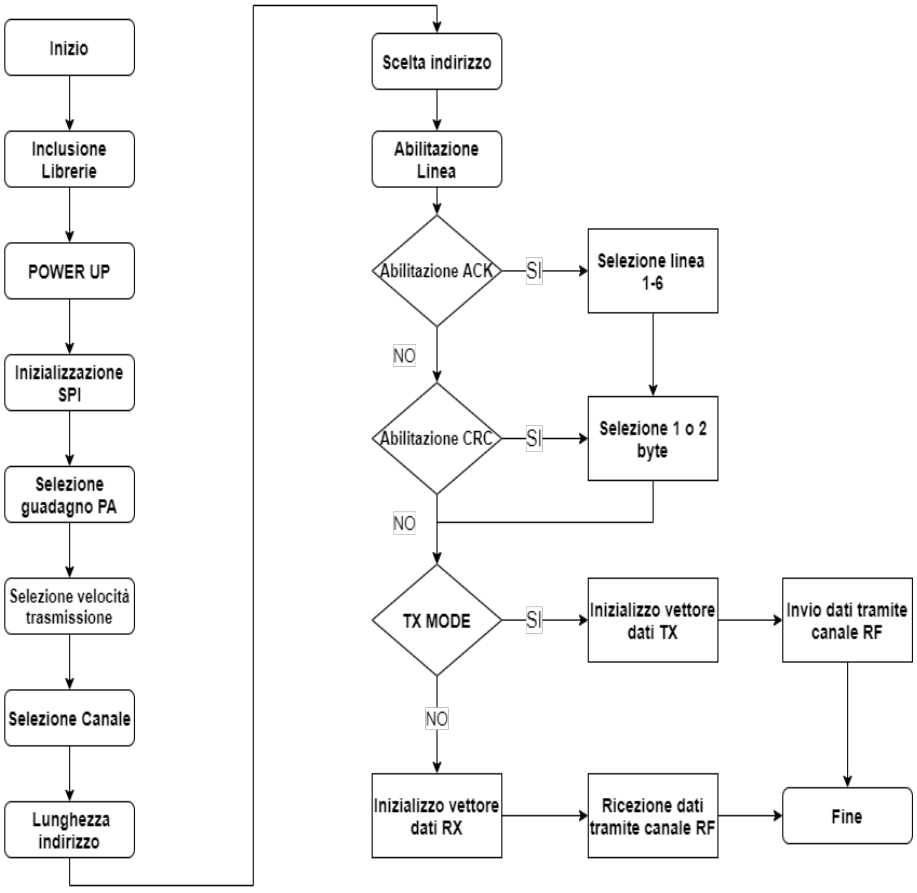

Figura 2.7: Diagramma di flusso

Capitolo 3

Codice Arduino

La prima fase del progetto si è focalizzata sulla realizzazione del nodo sensore impiegando sia in ricezione che in trasmissione la scheda Arduino Uno, allo scopo di verificare il corretto funzionamento del modulo. L'IDE contiene delle funzioni di libreria che consentono di testare la trasmissione di dati e la corretta ricezione senza il bisogno di considerare e configurare direttamente i registri del modulo, nonché il protocollo di comunicazione con il microcontrollore.

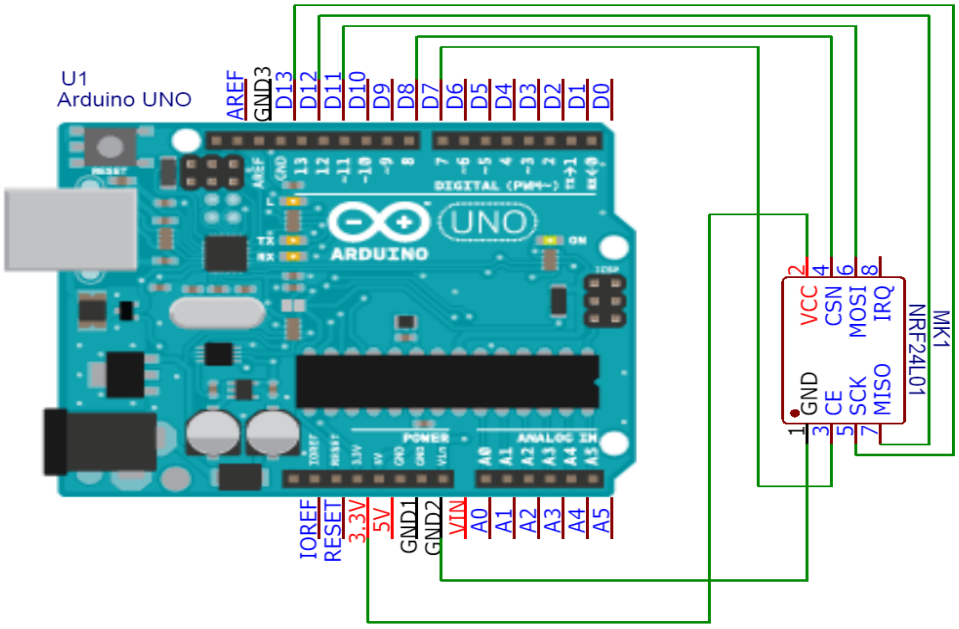

Figura 3.1: Collegamenti modulo NRF24L01 con Arduino Uno

Nella Figura 3.1 è possibile osservare come i collegamenti siano stati effettuati

sia per la TX che per la RX. Per facilitare la fase di test è possibile trovare in commercio degli adattatori per il modulo, che rendano i collegamenti più semplici, siccome il pinout non è compatibile con la breadboard, come è possibile osservare nella figura 3.2.

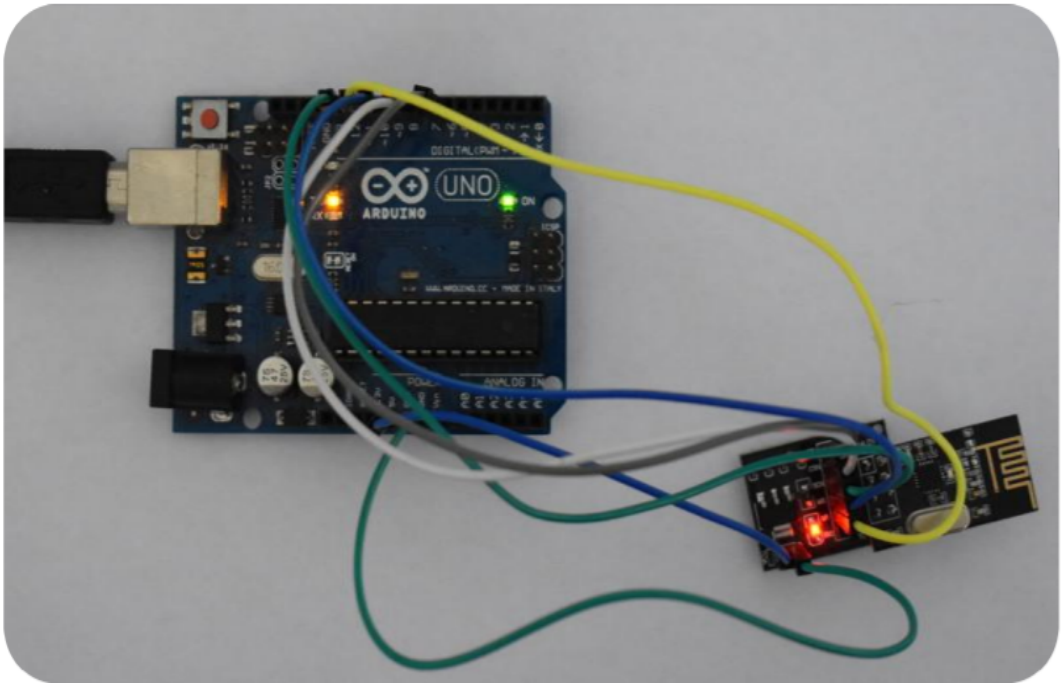

Figura 3.2: Collegamenti fisici modulo NRF24L01 con la scheda Arduino Uno

3.1 Trasmettitore

La prima operazione da compiere è l'inclusione delle librerie contenenti le funzioni per la configurazione e comunicazione. Si dichiara poi un oggetto radio specificando i pin sui quali si sono connessi chip enable (CE), rispettivamente chip select (CSN).
Utilizzando il metodo *begin* si inizializza la seriale a 115200 bps e l'oggetto radio precedentemente dichiarato. Il livello di amplificazione è impostato al massimo della potenza per mezzo del metodo *setPALevel*, *setDataRate* definisce la velocità di comunicazione a 2Mbps e *setChannel* seleziona il canale 124.
Per una prima analisi si decide di disabilitare il CRC, impostare il campo dati a 4 byte e disabilitare l'ACK per mezzo delle funzioni di libreria *disableCRC*,

setPayloadSize e *setAutoAck*. L'indirizzo di comunicazione viene definito mediante *openWritingPipe* a 5 byte e sarà lo stesso della ricevente. I dati da inviare saranno quindi quattro caratteri precedentemente dichiarati in un vettore, fornite alla proprietà *write* che invierà lungo la linea l'informazione.
Per avere un maggiore dettaglio della configurazione dei registri si può ricorrere a *printDetails*, che è contenuta nella libreria *printf*, dopo avere eseguito la sua inizializzazione mediante *begin*. I dati verranno visualizzati sul dispositivo di output di default che in questo caso è rappresentato dal monitor seriale.

```
STATUS           = 0x0e RX_DR=0 TX_DS=0 MAX_RT=0 RX_P_NO=7 TX_FULL=0
RX_ADDR_P0-1     = 0x4b4b4b4b4b 0xc2c2c2c2c2
RX_ADDR_P2-5     = 0xc3 0xc4 0xc5 0xc6
TX_ADDR          = 0x4b4b4b4b4b
RX_PW_P0-6       = 0x20 0x00 0x00 0x00 0x00 0x00
EN_AA            = 0x00
EN_RXADDR        = 0x03
RF_CH            = 0x7c
RF_SETUP         = 0x0f
CONFIG           = 0x06
DYNPD/FEATURE    = 0x00 0x00
Data Rate        = 2MBPS
Model            = nRF24L01+
CRC Length       = Disabled
PA Power         = PA_MAX
```

Figura 3.3: Dati stampati sul monitor seriale

La configurazione dei registri viene riportata in esadecimale per una maggiore leggibilità. Come si può osservare andando a confrontare la configurazione dei registri riportata nel datasheet, con quella imposta mediante il linguaggio Arduino esse sono consistenti.
Osservando ad esempio la configurazione del registro RF-CH, si esegue una conversione da esadecimale a decimale del numero 0x7C ottenendo il numero 124 che coincide con il canale configurato.

3.1.1 Codice sorgente Trasmettitore

```
//Autore:Ion Lucian Petrisor

#include <nRF24L01.h>
#include <SPI.h>
#include <RF24.h>
#include <printf.h>

int i;
char dati[4] = {'a', 'b', 'c', 'd'};
char data;

RF24 radio(7, 8); // CE e CSN

void setup() {

  Serial.begin(115200);

  printf_begin();

  Serial.println("modulo trasmettitore");

  radio.begin();

  radio.setPALevel(RF24_PA_MAX);

  radio.setDataRate(RF24_2MBPS);

  radio.setChannel(124);

  radio.openWritingPipe(0x4b4b4b4b4b);

  radio.disableCRC();

  radio.setPayloadSize(4);

  radio.setAutoAck(0);

  radio.stopListening();
}
```

```
40
41 void loop() {
42
43   for (i = 0; i < 4; i++) {
44     data = dati[i];
45     radio.write(&data, sizeof(char));
46     radio.printDetails();
47     delay(1000);
48
49   }
50
51 }
```

3.2 Ricevitore

In modo analogo a quanto fatto per il trasmettitore si eseguono le configurazioni della seriale e del modulo per quanto concerne il canale, la potenza dell'amplificazione, la velocità dei dati, l'ACK ed il CRC. Per mezzo di *startListening* il dispositivo viene configurato in ricezione e di seguito mediante *read* si esegue la lettura del buffer di una quantità di byte pari a quella che codificano una unsigned long. Per mezzo di shift a destra di 8 bit si ha la possibilità di visualizzare sul monitor seriale i caratteri trasmessi. Volendo trasmettere dei numeri complessi, che rappresentano l'impedenza caratteristica della batteria con parte reale ed immaginaria di 16 bit, si può sempre sfruttare l'invio di un carattere per volta. Mediante uno shift a destra di 16 bit si possono leggere i 16 bit più significativi rappresentanti la parte reale, mentre per visualizzare i 16 bit meno significativi è sufficiente un cast di tipo nella fase di lettura, interpretando il dato unsigned long come un intero. Visualizzando la configurazione dei registri si nota una differenza nel registro CONFIG ove il LSB è a 1 che sta ad indicare che il modulo è impostato in modalità di ricezione. Per quanto riguarda l'indirizzo, il canale e la velocità di trasmissione si nota la congruenza con quanto configurato dall'IDE di Arduino.

```
STATUS           = 0x0e RX_DR=0 TX_DS=0 MAX_RT=0 RX_P_NO=7 TX_FULL=0
RX_ADDR_P0-1     = 0x4b4b4b4b4b 0xc2c2c2c2c2
RX_ADDR_P2-5     = 0xc3 0xc4 0xc5 0xc6
TX_ADDR          = 0xe7e7e7e7e7
RX_PW_P0-6       = 0x20 0x00 0x00 0x00 0x00 0x00
EN_AA            = 0x00
EN_RXADDR        = 0x03
RF_CH            = 0x7c
RF_SETUP         = 0x0f
CONFIG           = 0x07
DYNPD/FEATURE    = 0x00 0x00
Data Rate        = 2MBPS
Model            = nRF24L01+
CRC Length       = Disabled
PA Power         = PA_MAX
```

Figura 3.4: Dati stampati sul monitor seriale

3.2.1 Codice Sorgente Ricevitore

```
1  //Autore:Ion Lucian Petrisor
2
3  #include <nRF24L01.h>
4  #include <SPI.h>
5  #include <RF24.h>
6  #include <printf.h>
7
8  unsigned long data;
9  char* stringa;
10
11 RF24 radio(7, 8); //CE e CSN
12
13
14 void setup() {
15
16   Serial.begin(115200);
17
18   printf_begin();
19
20   Serial.println("modulo ricevitore");
21
22   radio.begin();
23
24   radio.setPALevel(RF24_PA_MAX);
25
26   radio.setDataRate(RF24_2MBPS);
27
28   radio.setChannel(124);
29
30   radio.openReadingPipe(0, 0x4b4b4b4b4b);
31
32   radio.disableCRC();
33
34   radio.setAutoAck(0);
35
36   radio.startListening();
37
38 }
39
```

```
40 void loop() {
41
42   radio.read(&data, sizeof(unsigned long));
43   //radio.printDetails();
44   Serial.println((char)data);
45   Serial.println((char)(data >> 8));
46   Serial.println((char)(data >> 16));
47   Serial.println((char)(data >> 24));
48   Serial.println((int)(data >> 16));
49   Serial.print((int)(data));
50   Serial.println(" I");
51   Serial.println((unsigned long)data);
52   delay(1000);
53
54 }
```

Capitolo 4

Codice dsPIC30F3013

4.1 Introduzione

La seconda parte del progetto si è focalizzata sulla realizzazione del trasmettitore, non facendo più uso di un Arduino Uno, ma impiegando un microcontrollore PIC. La scelta è ricaduta sul dsPIC30F3013 per via della maggior semplicità di utilizzo in fase di test, essendo compatibile con la breadboard e mantenedo una architettura simile a quella del controllore presente a bordo del VIA impiegato, ovvero il dsPIC33EP512GP806-I/PT. Tutte le informazioni necessarie per la configurazione dei registri del controllore sono state ottenute consultando i documenti [6] e [7].
Come ambiente di sviluppo si è fatto uso di MPLAB X IDE v5.45 , come programmatore PICkit 3 e come compilatore XC 16 v1.70 . I collegamenti del programmatore sono stati effettuati seguendo le indicazioni fornite dal costruttore nel seguente documento [8] . Per i bit di configurazione si è fatto uso del configuratore presente nell'IDE di MPLAB. L'alimentazione del microcontrollore è fornita dal programmatore PICkit 3, mentre per quanto riguarda il modulo si è ricorso ad un adattatore disponibile in commercio che abbassa la tensione fino a 3.3V, infatti il modulo non può operare alla tensione di 5V. Tale modulo ricorre ad uno stabilizzatore lineare per compiere l'operazione, si tratta del AMS1117 prodotto dalla Maxim. Come è possibile osservare dalla figura 4.1 come CE si è utilizzato il pin RB3, mentre come CSN è stato adoperato RB2.

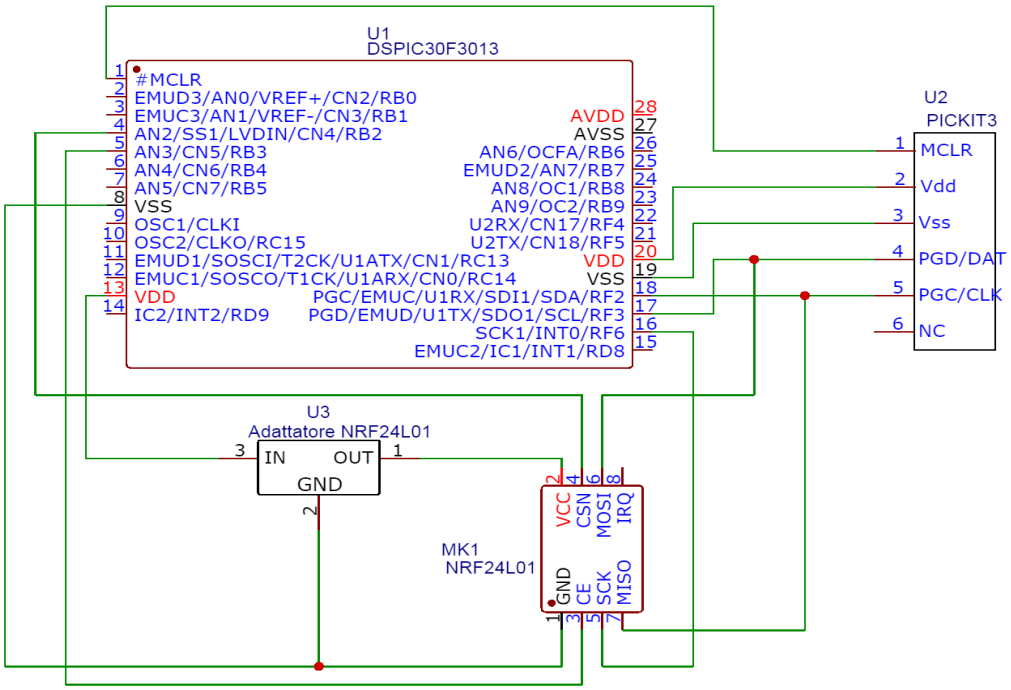

Figura 4.1: Schema elettrico TX con dsPIC30F3013

4.2 Main

La prima operazione prevede il richiamo delle funzioni di configurazione della SPI e successivamente si inizializza il modulo con la procedura dedicata. Si dichiara poi un vettore contenete le informazioni da inviare, in questo caso quattro caratteri a 8 bit, che mediante il comando di invio dati sono forniti al modulo per la trasmissione. Il processo si ripete trasmettendo con cadenza di un secondo i dati che devono essere consegnati al modulo.

4.3 Funzioni

Per una maggiore portabilità del programma si è scelto di strutturare il codice facendo uso di apposite funzioni. Tale scelta è giustificata da una futura integrazione della parte di comunicazione direttamente a bordo del VIA. Le funzioni implementate nel codice sono le seguenti:

- CONFIG-NRF24
- WRITESPI1-REGISTER
- SPI-CONFIG

CONFIG-NRF24 è la funzione che inizializza i registri del modulo NRF24L01 facendo uso della funzione WriteSPI-1-register, per impartire il comando SPI e successivamente per fornire il valore di configurazione del registro. Nel caso specifico il modulo è impostato come TX, CRC e ACK sono disabilitati e le linee dati attive sono la 0 e la 1. L'indirizzo impostato è identico a quello presente nel trasmettitore realizzato con Arduino ovvero, 0x4B4B4B4B4B ed è a 5 byte. Il campo dati del pacchetto ha una lunghezza di 4 byte, consentendo di inviare al massimo quattro caratteri per pacchetto. La velocità di trasmissione dati è di 2Mbps, la potenza di uscita è di 0dBm, ed il canale su cui si opera è il numero 124.
WRITESPI1-REGISTER è una funzione che consente di inviare un carattere ad 8 bit sulla SPI scrivendolo nel registro SPI1BUF, imposta poi un'attesa fintanto che il flag SPITBF del registro SPI1STAT non segnala che il dato è stato inviato e che il buffer è nuovamente libero.
SPI-CONFIG è una funzione che opera sul registro SPI1STAT sul bit SPIEN per disabilitare la seriale durante la configurazione, ripristinando poi lo stato iniziale. La configurazione fa uso del registro SPI1CON per impostare una comunicazione a 8 bit, il PIC è in modalità master ed esegue l'inizializzazione alla Mode 0.

4.4 Dati ricevuti

Come possibile visualizzare nella figura 4.2 la comunicazione avviene con successo, infatti al lato ricezione vengono correttamente ricevuti in sequenza i quattro caratteri presenti nel buffer di trasmissione, ovvero *'a' 'm' 'm' 'a'*. Impiegando la trasmissione di un carattere per volta si ha comunque modo di trasmettere dati aventi una lunghezza superiore, trasmettendo tali dati in più blocchi da 8 bit ed aspettando di riempire il buffer di ricezione con tutte le parti componenti il dato. Risulta perciò possibile trasmettere dei numeri complessi sfruttando tale tecnica trasmettendo prima due caratteri che codificano la parte reale e successivamente due caratteri che rappresentano la parte immaginaria. In questo modo si riesce a soddisfare la richiesta di trasmettere dei dati complessi a parte reale ed immaginaria ciascuna di 16 bit. Facendo attenzione al primo numero *1634561377* che è un unsigned long a 32 bit, rappresentante il valore contenuto nel buffer di ricezione, si può convertire in esadecimale nel numero *0x616D6D61*. Prendendo a blocchi di due cifre esadecimali, cioè a blocchi di 8 bit, il numero esadecimale, possiamo notare che tali numeri convertiti in decimale, corrispondono alla codifica ASCII associata a ciascun carattere. Ad esempio il numero *0x61* corrisponde al numero decimale *97* che è il numero associato al carattere *'a'* nella codifica ASCII. Considerando invece blocchi di quattro cifre esadecimali, ovvero 16 bit, è possibile ottenere la codifica di un numero immaginario a parte reale ed immaginaria di 16 bit ciascuna.
Il numero esadecimale *0x616D* corrisponde in decimale a *24941* e rappresenta la parte reale di un numero immaginario, al contempo *0x6D61* in decimale corrisponde a *28001* e rappresenta la parte immaginaria.

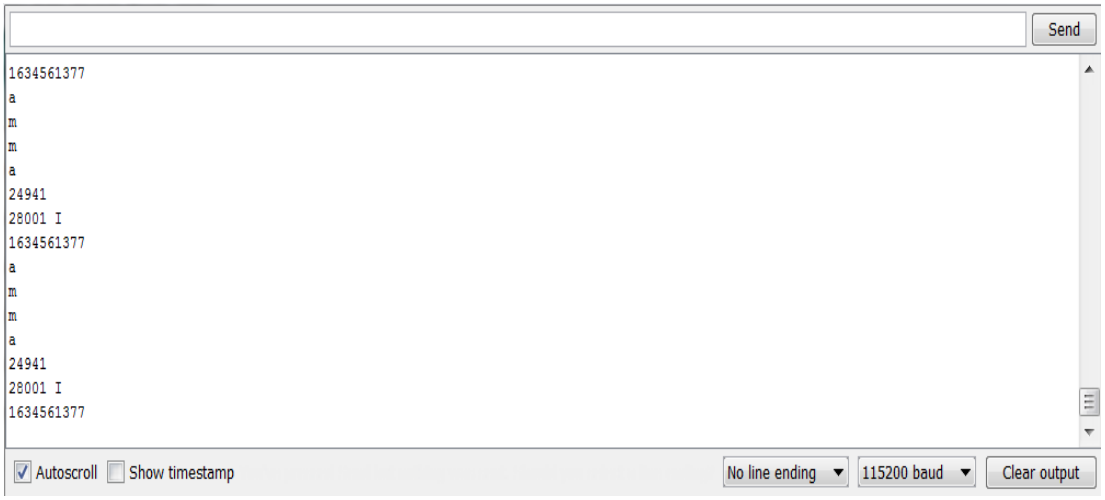

Figura 4.2: Dati stampati sul monitor seriale RX

Conclusioni

Il progetto che si desiderava realizzare è stato completato e la comunicazione dei dati avviene correttamente con il trasmettitore realizzato mediante il dsPIC30F3013. Rimane spazio per un ulteriore miglioramento del codice sviluppato che preveda l'implementazione del CRC e del ACK per consentire una maggior robustezza dei dati. Per consentire una riduzione delle dimensioni si dovrà portare il codice sviluppato a bordo del microcontrollore del VIA, bisognerà perciò sviluppare un altro printed circuit board (PCB) che includa anche il modulo NRF24L01. A differenza di quanto fatto nella fase di test, ove si è impiegato il modulo direttamente assemblato da un costruttore esterno, contenete tutti i componenti neccessari al corretto funzionamento, si dovrà integrare il chip NRF24L01 sullo stesso circuito stampato del VIA. Questo risulta possibile seguendo le indicazioni riportate nel datasheet [5] e consente risparmiare ulteriore spazio. Servirà poi considerare come si ritiene opportuno alimentare il nuovo VIA, avendo fra le scelte possibili sia l'impiego diretto della batteria che si intende monitorare, che l'utilizzo di un'altra esterna da dimensionare. Bisogna inoltre considerare un problema in fase di accensione della trasmittente che in qualche occasione tarda il proprio funzionamento. Tale evento è da intendersi come un problema di temporizzazione dovuto molto probabilmente a dei contatti imperfetti sulla scheda di test, pertanto con la realizzazione su circuito stampato si dovrebbe risolvere.

Appendice

Codice sorgente dsPIC30F3013

```
/*
 * File:   newmainXC16.c
 * Author: Petrisor
 *
 * Created on 9 luglio 2021, 13.22
 *
 */

// DSPIC30F3013 Configuration Bit Settings

//Frequenza di lavoro: 7.37 MHz


// 'C' source line config statements

// FOSC

// Oscillator (Internal Fast RC )
#pragma config FOSFPR = FRC
// Clock Switching and Monitor (Sw Disabled, Mon Disabled)
#pragma config FCKSMEN = CSW_FSCM_OFF

// FWDT

// WDT Prescaler B (1:16)
#pragma config FWPSB = WDTPSB_16
// WDT Prescaler A (1:512)
#pragma config FWPSA = WDTPSA_512
// Watchdog Timer (Disabled)
#pragma config WDT = WDT_OFF

// FBORPOR

// POR Timer Value (64ms)
#pragma config FPWRT = PWRT_64
```

```
// Brown Out Voltage (2.0V)
#pragma config BODENV = BORV20
// PBOR Enable (Enabled)
#pragma config BOREN = PBOR_ON
// Master Clear Enable (Enabled)
#pragma config MCLRE = MCLR_EN

// FGS

// General Code Segment Write Protect (Disabled)
#pragma config GWRP = GWRP_OFF
// General Segment Code Protection (Disabled)
#pragma config GCP = CODE_PROT_OFF

// FICD

// Comm Channel Select (Use PGC/EMUC and PGD/EMUD)
#pragma config ICS = ICS_PGD

#include <xc.h>

#define FCY 7370000
#include <libpic30.h>


#define DELAY_105uS asm volatile ("REPEAT, #4201"); Nop();
#define CANALE 124
#define BUFFER 4 //Bytes
#define ADDRESS_TX 75 //repeated 5 times
#define ENDRX 75

#define CSN_TX PORTBbits.RB2
#define CE_TX PORTBbits.RB3


void config_nrf24(void);
void WriteSPI_1(char data);
void WriteSPI_1_register(char data);
void spi_config(void);

int main(void) {

    char send[BUFFER] = {'a', 'm', 'a', 'a'};
    int i = 0;

    TRISB = 0x00; //Imposto la PORT B come uscita

    ADPCFG = 0xFF; //I/O in digitale (pag 398)
    ADCON1bits.ADON = 0; //Spengo il convertitore A/D (pag 393)
```

```
85      spi_config();
86      config_nrf24();
87
88      //Enter in TX mode
89
90
91      while (1) {
92
93
94          CE_TX = 0;
95          //PORTBbits.RB3=0;
96
97
98          //Flush_TX
99          /*
100         CSN_TX=0;
101         WriteSPI_1_register((char)0xE1);
102         __delay_us(10);
103         CSN_TX=1;
104          * */
105
106         //Status register Reset
107
108         /* CSN_TX=0;
109
110          //PORTBbits.RB2=0;
111          WriteSPI_1_register((char)0x07);
112          WriteSPI_1_register((char)0x70);
113          __delay_us(10);
114          CSN_TX=1;*/
115
116         //PORTBbits.RB2=1;
117
118
119         //Serial data send
120
121
122         CSN_TX = 0;
123         //PORTBbits.RB2=0;
124
125         WriteSPI_1_register((char) 0xA0); //command send data
126         for (i = 0; i < BUFFER; i++) {
127             WriteSPI_1_register((char) (send[i])); //data to
    send
128             __delay_us(10);
129         }
130
131         //PORTBbits.RB2=1;
132         CSN_TX = 1;
```

```
133
134         //__delay_us(10);
135
136
137
138         //Pulse to send data
139
140
141         //PORTBbits.RB2=0;
142         CSN_TX = 0;
143         __delay_us(15);
144         //PORTBbits.RB2=1;
145         CSN_TX = 1;
146
147         //PORTBbits.RB3=1;
148         CE_TX = 1;
149
150
151         //Delay of 1s before transmitting other data
152         for (i = 0; i < 9524; i++) {
153             Nop();
154         }
155
156     }
157     return 0;
158 }//chiudo main
159
160
161
162 //All the addresses are shifted by 0x20 not in according to
    datasheet
163
164 void WriteSPI_1(char data) {
165
166     unsigned int i;
167
168
169     for (i = 0; i < 3; i++) {
170         Nop();
171     }
172
173
174     SPI1BUF = data;
175     while (SPI1STATbits.SPITBF);
176
177     //Necessario per interpretare le transizioni del SS
178     for (i = 0; i < 50; i++) {
179         Nop();
180     }
```

```
181
182
183
184 }
185
186 void WriteSPI_1_register(char data) {
187
188
189     SPI1BUF = data;
190     while (SPI1STATbits.SPITBF);
191
192
193 }
194
195 void config_nrf24(void) {
196
197
198     CE_TX = 0;
199
200     //CONFIG: Configure in TX Mode,Power up, disable CRC, 2
    bytes CRC
201
202
203     CSN_TX = 0;
204     WriteSPI_1_register((char) (0x00 | 0x20));
205     WriteSPI_1_register((char) 0b00000110);
206     __delay_us(10);
207     CSN_TX = 1; //DONE
208
209     //EN_AA:disable auto-acknowledge on all pipes
210     CSN_TX = 0;
211     WriteSPI_1_register((char) (0x01 | 0x20));
212     WriteSPI_1_register((char) 0b00000000); //
213     __delay_us(10);
214     CSN_TX = 1; //DONE
215
216     //EN_RXADDR: Enable PIPE0 & PIPE 1
217     CSN_TX = 0;
218     WriteSPI_1_register((char) (0x02 | 0x20));
219     WriteSPI_1_register((char) 0x03); //
220     __delay_us(10);
221     CSN_TX = 1; //DONE
222
223     //SETUP_AW: Configure TX ADDRESS length
224     CSN_TX = 0;
225     WriteSPI_1_register((char) (0x03 | 0x20));
226     WriteSPI_1_register((char) 0x03); //5 bytes
227     __delay_us(10);
228     CSN_TX = 1; //DONE
```

```
229
230     //SETUP_RETR:Retransmission delay 1250 us
231     CSN_TX = 0;
232     WriteSPI_1_register((char) (0x04 | 0x20));
233     WriteSPI_1_register((char) 0xb01011111);
234     __delay_us(10);
235     CSN_TX = 1; //DONE
236
237     //RF_CH: Configure Channel using CANALE definition
238     CSN_TX = 0;
239     WriteSPI_1_register((char) (0x05 | 0x20));
240     WriteSPI_1_register((char) CANALE);
241     __delay_us(10);
242     CSN_TX = 1; //DONE
243
244     //RF_SETUP:Configure data rate LNA gain and RF output power
245     CSN_TX = 0;
246     WriteSPI_1_register((char) (0x06 | 0x20));
247     //2Mbps,0dBm,activate LNA
248     WriteSPI_1_register((char) 0b00001111);
249     __delay_us(10);
250     CSN_TX = 1; //DONE
251
252     //STATUS: Reset status register
253     CSN_TX = 0;
254     WriteSPI_1_register((char) (0x07 | 0x20));
255     WriteSPI_1_register((char) 0b01110000);
256     __delay_us(10);
257     CSN_TX = 1; //DONE
258
259
260     // TX_ADDR:Configure TX ADDRESS
261     CSN_TX = 0;
262     WriteSPI_1_register((char) (0x10 | 0x20));
263     WriteSPI_1_register((char) ADDRESS_TX);
264     WriteSPI_1_register((char) ADDRESS_TX);
265     WriteSPI_1_register((char) ADDRESS_TX);
266     WriteSPI_1_register((char) ADDRESS_TX);
267     WriteSPI_1_register((char) ADDRESS_TX);
268     __delay_us(10);
269     CSN_TX = 1; //DONE
270
271
272     //RX_PW_P0:Configure buffer length
273     CSN_TX = 0;
274     WriteSPI_1_register((char) (0x11 | 0x20));
275     WriteSPI_1_register((char) BUFFER); //BUFFER modified 0x20
276     __delay_us(10);
277     CSN_TX = 1; //DONE
```

```
278
279
280     //DYNPD:Disable Dynamic palyload length on all pipes
281
282     CSN_TX = 0;
283     WriteSPI_1_register((char) (0x1C | 0x20));
284     WriteSPI_1_register((char) 0x00);
285     __delay_us(10);
286     CSN_TX = 1; //DONE
287
288     //FEATURE:Disable DPL,ACK,W_TX_PAYLOAD_NOACK command
289
290     CSN_TX = 0;
291     WriteSPI_1_register((char) (0x1D | 0x20));
292     WriteSPI_1_register((char) 0x00);
293     __delay_us(10);
294     CSN_TX = 1; //DONE
295
296     //Rise time from stand-by to TX
297     __delay_ms(2);
298     CE_TX = 1;
299     __delay_us(150);
300 }
301
302 void spi_config(void) {
303
304     //For mode 0 configure: CKP=0,CKE=1,CPHA=0
305
306     SPI1STATbits.SPIEN = 0; //Disable SPI
307     SPI1CONbits.DISSDO = 0; // RX mode only off
308     SPI1CONbits.MODE16 = 0; // 8-bit mode
309     SPI1CONbits.SMP = 1; //Data sampled at end
310     SPI1CONbits.CKE = 1; //data transition from active to idle
311     SPI1CONbits.CKP = 0; //Idle clock low level
312     SPI1CONbits.MSTEN = 1; //Master Mode
313     SPI1CONbits.SSEN = 0; // SS driven by port not module
314
315     //bit 0-1 PRE_1 00:64, 01:16,10:4,11:1
316     //bit 4-2 PRE_2 000:8, 110:2, 111:1
317     SPI1CON = SPI1CON | 0b00011110; //PRE_2=1, PRE_1=4
318
319     SPI1CONbits.FRMEN = 0; //Framed SPI disabled
320     //SPI1CONbits.SPIFSD=0;//Framed sync pulse output (master)
321
322     SPI1STATbits.SPIEN = 1; //Enable SPI;
323
324 }
```

Ringraziamenti

Mi ritrovo dopo tre anni trascorsi della mia vita, dedicati allo studio con dedizione ed interesse, a dover ricordare chi mi allegerì questo percorso cominciato in tenera età già alle scuole superiori. I primi tempi mi permisero di mettere le basi, cominciando dalle prime nozioni di fisica e matematica per poi trovare le materie d'indirizzo. Lungo questo percorso ho avuto la fortuna di incontrare persone capaci, modeste e comprensive che mi hanno in primis insegnato a vivere e poi introdotto la loro materia. Tra questi due in particolare mi hanno fatto appassionare all'elettronica, credendo sempre in me e spronandomi a fare sempre il meglio possibile, sto parlando di *Ing. Casadei Lelli Daniele* e *PhD Roberto Versari* . Vorrei pertanto ringraziarli di avermi trasmesso l'amore per l'elettronica e per avermi indirizzato in maniera saggia verso questo percorso universitario che si è ormai concluso. L'università mi ha permesso di sviluppare un metodo di studio, di avere un pensiero critico e non dare per scontati argomenti già appresi. Ho constatato infatti che senza delle buone conoscenze degli argomenti di base risulti assai difficile riuscire a domare problemi più complessi. In questo cammino contraddistinto da alti e bassi ho avuto il piacere di incontrare un gruppo docenti molto aperto al dialogo e al confronto, che mi ha permesso di apprendere argomenti nuovi ed affinare quelli che a grandi linee già avevo affrontato. Nonostante alcuni disguidi, per la maggior parte fortuiti, considero che il corso di studi sia ottimo e ricco di contenuti, nonché particolarmente responsivo alle segnalazioni degli studenti e sono lieto di avere avuto l'occasione di seguirlo. Avrei poi il piacere di ringraziare il *Prof. Marco Crescentini* che mi ha fornito tutte le indicazioni necessarie per svolgere la tesi, nonché per l'ottimo corso che ci ha tenuto; ringrazio poi la *Dott.ssa. Roberta Ramilli* per l'infinita disponibilità e per il supporto fornito. Per ultimo avrei piacere di ringraziare la mia famiglia che mi ha sempre supportato in tutto quello che faccio, permettendomi di concentrarmi ed ottenere i migliori risultati possibili.

Bibliografia

[1] Crescentini, M. et al. Online EIS and diagnostics on lithium-ion batteries by means of low-power integrated sensing and parametric modeling. IEEE Transactions on Instrumentation and Measurement, 70, 1-11.

[2] Luciani, G., Ramilli, R., Romani, A., Tartagni, M., Traverso, P. A., Crescentini, M. (2019). A miniaturized low-power vector impedance analyser for accurate multi-parameter measurement. Measurement, 144, 388-401.

[3] (2020) Energy-ECS: Smart and secure energy solutions for future mobility, ECSEL H2020 Proposal, pp. 16-20, Private document.

[4] Meddings, Nina, et al. "Application of electrochemical impedance spectroscopy to commercial Li-ion cells: A review." Journal of Power Sources 480 (2020): 228742.

[5] nRF24L01 Product Specification v2 0-9199

[6] dsPIC30F2011/2012/3012/3013 Data Sheet

[7] dsPIC30F Family Reference Manual

[8] PICkit™ 3 Programmer/Debugger User's Guide