ALMA MATER STUDIORUM - UNIVERSITÀ DI BOLOGNA
CAMPUS DI CESENA
SCUOLA DI INGEGNERIA E ARCHITETTURA

CORSO DI LAUREA IN INGEGNERIA ELETTRONICA E TELECOMUNICAZIONI

PROGETTO DI UN NODO SENSORE WIRELESS ULTRA LOW-POWER PER APPLICAZIONI DI ENERGY HARVESTING

Elaborato in
Elettronica dei sistemi digitali

Relatore
Prof. Aldo Romani

Presentata da
Davide Casadei

Correlatore
Matteo Filippi

Sessione III
Anno Accademico 2012/2013

Indice

Introduzione

Negli ultimi anni si è assistito ad una notevole evoluzione dei sistemi elettronici, che ha portato ad aumentarne l'efficienza energetica e diminuirne le dimensioni.
In questo modo è stato possibile creare dispositivi mobili, alimentati a batterie, con una durata media sempre maggiore. Tuttavia l'uso di batterie è limitante sia per l' autonomia, che per i costi. Proprio per questa ragione ci si sta sensibilizzando sempre di più sul tema dell'Energy Harvesting, processo attraverso il quale, piccole quantità di energia presenti nell'ambiente possono essere catturate, trasformate in energia elettrica, ed accumulate per essere poi utilizzate da un sistema elettronico, consentendo l'esecuzione di alcuni semplici operazioni, senza la necessità di utilizzare una fonte di energia convenzionale.[1]

L'obbiettivo della tesi è proprio quello di svincolarsi dalle fonte energetiche tradizionali e realizzare un sistema in grado di autoalimentarsi, prendendo energia dall'ambiente che lo circonda. Più in particolare, questo progetto si propone di realizzare un nodo wireless, alimentato attraverso l'Energy Harvesting, in grado di misurare la temperatura ambiente ed inviarla ad un sistema ricevente che la visualizzerà su uno schermo LCD.

La difficoltà del progetto starà proprio nel risolvere tutte le problematiche che derivano dall'utilizzo di questa tecnologia, infatti l'energia fornita dalla sorgente, a causa della sua natura, sarà molto bassa, perciò si dovrà cercare di ridurre i consumi del sistema trasmittente il più possibile e gestire al meglio la poca energia a disposizione. Il sistema ricevente invece, nell'ottica di un'eventuale applicazione finale, sarà situato in una zona dove le informazioni della temperatura saranno gestite e processate, quindi sarà progettato supponendo di avere la disponibilità di un'alimentazione cablata esterna.

Per riuscire nell'obbiettivo, come prima cosa ci si è occupati della ricerca dei componenti, focalizzandosi su dispositivi ultra low-power, in grado di avere bassissimi consumi, e scegliendo, in base allo studio dei Data Sheet, quelli più adatti al progetto. Una volta scelti i componenti si è proceduto all'implementazione fisica, sia del trasmettitore che del ricevitore, verificandone il corretto funzionamento. In seguito, attraverso delle misure, si è trovata la soluzione

ottimale per il progetto, cioè quella che permette di avere il minor consumo possibile ma allo stesso tempo di riuscire a svolgere l'obbiettivo prefissato in maniera soddisfacente.

Infine dopo aver implementato la soluzione definitiva se ne è verificata la funzionalità, osservando se questa applicazione soddisfa l'obbiettivo proposto e se risulta realmente implementabile ed utilizzabile.

1 L'Energy Harvesting

L'energia è ovunque nell'ambiente che ci circonda, come ad esempio l'energia termica, eolica, elettromagnetica, meccanica, solare, etc. Inoltre, intorno a noi viene continuamente dispersa energia da processi naturali e artificiali, che non viene utilizzata e quindi risulta sprecata. Tuttavia, l'energia ricavabile da queste fonti, spesso è così piccola da non essere sufficiente a svolgere qualsiasi compito pratico. Per tale motivo questo tipo di energia non è stata presa in considerazione, finché, lo sviluppo tecnologico non lo ha reso possibile, aumentando sia l'efficienza energetica dei componenti elettronici, tra cui i microcontrollori, riducendone il loro consumo, sia l'efficienza di conversione dei trasduttori, i quali sono i grado di catturare piccole quantità di energia dall'ambiente e trasformarle in energia elettrica.

1.1 Cos'è

L'Energy Harvesting (dall'inglese "raccolta di energia") è il processo di acquisizione di piccole quantità di energia inutilizzata, da una o più fonti, naturali o artificiali, presenti nell'ambiente circostante, la sua trasformazione in energia elettrica ed il suo successivo accumulo in condensatori o piccole batterie ricaricabili per un suo uso successivo.

I sistemi di Energy Harvesting quindi si occupano di catturare, accumulare, immagazzinare e gestire in maniera efficace ed efficiente l'energia presente nell'ambiente in cui si trovano ad operare, per fornirla poi in una forma che può essere utilizzata per eseguire un compito utile, e per fare tutto ciò è necessaria la presenza dei seguenti elementi chiave:

- fonte di energia nei pressi del dispositivo, come ad esempio:
    - vibrazioni meccaniche, deformazioni, movimento o rumore acustico (energia meccanica);
    - luce solare o di una stanza (energia data dalla luce);
    - radiazioni date dalle trasmittenti radio o da trasformatori, bobine e induttori (energia elettromagnetica);
    - calore dato da fonti d'attrito, forni, stufe o dal sole (energia termica);
    - vento, correnti oceaniche, radiazioni solari (energia naturale);
    - movimento o calore dato dal corpo umano.

È importante notare che tutte queste fonti di energia sono praticamente illimitate ed essenzialmente gratuite;

- un trasduttore in grado di trasformare l'energia presente nell'ambiente in forma elettrica. Alcuni esempi sono i dispositivi piezoelettrici che trasformano l'energia meccanica in energia elettrica o i pannelli fotovoltaici o solari che trasformano la radiazione elettromagnetica luminosa in energia elettrica;
- un circuito elettrico in grado di estrarre intelligentemente tale energia e immagazzinarla temporaneamente in un condensatore o una piccola batteria ricaricabile, che consumi durante ogni sua attivazione meno energia di quella estratta;
- ed infine un'applicazione finale come una rete di sensori wireless o dispositivi di controllo e monitoraggio.

Questo tipo di tecnologia sta diventando un'alternativa sempre più attraente soprattutto nel caso in cui il punto da raggiungere con la tradizionale alimentazione sia scomodo o inconveniente o nel caso di batterie costose. Infatti attraverso l' Energy Harvesting, è possibile alimentare un dispositivo senza il bisogno di allacciarlo ad una rete di distribuzione elettrica o utilizzare una batteria, garantendo così un risparmio economico dato dall'alimentazione gratuita e dall'assenza di manutenzione. Inoltre, può essere utilizzata anche come fonte d'energia alternativa, per integrare una fonte di energia primaria e migliorare così l'affidabilità del sistema complessivo e prevenire interruzioni dell'alimentazione.

1.2 Applicazioni

Grazie ai grandi vantaggi citati poco fa, sempre più applicazioni beneficiano di alimentazioni derivanti da sistemi di Energy Harvesting, come ad esempio:

- Domotica: una rete di sensori wireless all'interno di un edificio può essere in grado di gestire in maniera efficiente l'illuminazione, il riscaldamento e aria condizionata per ridurre i costi che derivano dal loro utilizzo;
- monitoraggio strutturale, come ad esempio quello di un ponte, dove si può prendere energia dal sole, dal vento o dalle vibrazioni;
- monitoraggio ambientale: viene usata soprattutto quando i luoghi siano di difficile accesso o pericolosi per l'operatore, come a ridosso di un vulcano,

in un territorio rischio frane o valanghe, oppure in mezzo al mare, dove viene trasformata l'energia delle onde in energia elettrica, usata poi dai sensori per il monitoraggio oceanografico;

- gestione di sistemi agricoli: attraverso l'utilizzo di sensori e interruttori comandati via wireless, e alimentati attraverso l'energia solare o eolica, è possibile monitorare i parametri ambientali, come intensità luminosa, umidità del terreno, temperatura e prendere decisioni di conseguenza, in modo da gestire in modo efficiente gli impianti necessari alla coltivazione, come per esempio decidere quando accendere l'impianto idraulico o di ventilazione in una serra;
- monitoraggio di macchinari e apparecchiature, utilizzando come fonte di energia il calore o le vibrazioni emesse proprio da quest'ultime;
- monitoraggio remoto dei pazienti, cioè misurare i parametri vitali di un paziente attraverso dei sensori impiantabili, che non necessitano di batterie, per controllarne continuamente lo stato di salute, consentendo così un notevole miglioramento dello stile di vita di un paziente affetto da patologie croniche. L'energia con cui può essere alimentato il dispositivo, in questo caso, può essere presa dal calore naturalmente emesso dal nostro corpo o attraverso il movimento, come camminare, pedalare o anche il movimento delle braccia.

Questi sono solo pochi esempi, e alcuni di essi sono possibili solamente in prospettiva, ma già da questi si può capire come l' Energy Harvesting sia applicabile ad una molteplicità di ambiti e applicazioni, che saranno destinati inevitabilmente a crescere grazie al costante sviluppo tecnologico.[2]

2 Sistema implementato

2.1 Descrizione del sistema

La tesi si propone l'obbiettivo di progettare un nodo sensore wireless alimentato attraverso l' Energy Harvesting in grado di misurare la temperatura ed inviarla ad un ricevitore. Quindi Il sistema da implementare è formato da un trasmettitore ed un ricevitore.

Il sistema "trasmettitore", rappresentato in Figura 2.1, può essere suddiviso in due macro-blocchi: il primo, formato da trasduttore piezoelettrico, capacità ed il voltage monitor, che si occuperà della gestione dell'energia, ed il secondo, formato da sensore, modulo wireless e microcontrollore, che costituisce l'applicazione finale vera e propria, che avrà il compito di misurare la temperatura ed inviarla al ricevitore utilizzando l'energia fornita dal primo macro-blocco. Andiamo ora a dare una descrizione generale dei blocchi:

- trasduttore piezoelettrico: si occupa di trasformare l'energia fornita dall' ambiente, nel nostro caso sotto forma di vibrazioni, in energia elettrica;
- condensatore: cattura ed immagazzina l'energia fornita dal trasduttore, caricandosi con la corrente fornita da quest' ultimo;
- voltage monitor: adatta la tensione ai capi del condensatore alle esigenze dell'applicazione finale ed eventualmente decidere quando renderla disponibile;
- sensore di temperatura: misura la temperatura ambiente e la fornisce in formato digitale al microcontrollore;
- microcontrollore: gestirà la comunicazione e le tempistiche fra i componenti;
- modulo wireless: trasmette il dato di temperatura, fornito dal microcontrollore, al sistema ricevente.

Quando l'energia immagazzinata sul condensatore è tale da permettere il corretto funzionamento di tutti i componenti presenti nell'applicazione finale, viene fatta partire la fase di acquisizione ed invio del dato di temperatura. L'energia necessaria per svolgere quest'ultima fase viene prelevata dal condensatore che quindi si scarica, non permettendo più altre trasmissioni. Per questo motivo, dopo l'invio segue un periodo di inattività per permettere la ricarica del condensatore. Durante questo periodo bisognerà decidere, attraverso delle misure, cos'è più

conveniente a livello energetico fra: scollegare l'alimentazione o mandare i dispositivi nella modalità a più basso consumo energetico possibile (sleep e shutdown).

Il sistema "ricevitore" è formato da: modulo wireless, microcontrollore e schermo LCD. Il modulo wireless riceve il dato inviato dal trasmettitore e lo fornisce al microcontrollore, che lo invierà a sua volta allo schermo LCD, con cui dovrà essere opportunamente interfacciato per poter permettere all'utente la visualizzazione del valore di temperatura in formato decimale. L'implementazione del ricevitore è meno complessa, in quanto viene utilizzata la tradizionale alimentazione cablata. Infatti, dato che non si sa a priori quando il dato di temperatura verrà inviato, è necessario che il modulo wireless stia sempre in ricezione, inoltre il dato fornito dalla temperatura, in una futura applicazione, dovrà essere elaborato per eseguire uno scopo pratico, quindi, per queste ragioni, dal lato ricevitore sarà plausibile la disponibilità di un'alimentazione di tipo cablato.

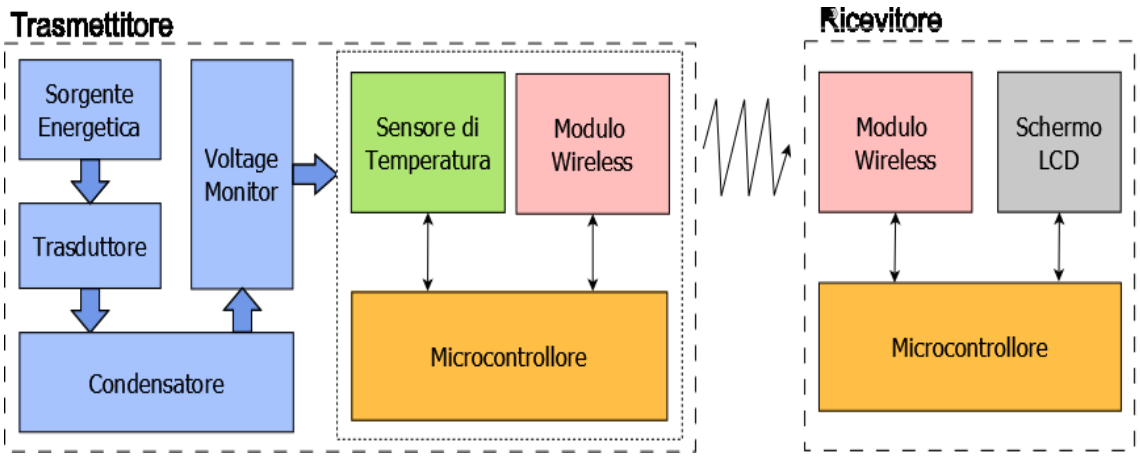

Figura 2.1: schema a blocchi del progetto da realizzare

2.2 Scelta dei componenti

La scelta dei componenti è stata svolta durante l'attività di tirocinio ed è stata portata avanti ed approfondita durante questa tesi. Di seguito verranno esposte le principali caratteristiche su cui si è basata la scelta dei componenti.

Come già detto a causa della natura della sorgente, questa potrà fornire solo piccole quantità di energia al sistema, quindi la scelta dei componenti è stata fatta principalmente in base a:

- consumo in potenza: l'energia immagazzinata nel condensatore deve essere sufficiente per completare la fase di acquisizione e invio del dato di temperatura, perciò i componenti durante il loro normale funzionamento in questa fase, devono consumare il meno possibile. Inoltre il consumo deve essere limitato anche durante il periodo di inattività, cioè quando impostati nella modalità a più basso consumo possibile in modo da permettere la ricarica del condensatore per la trasmissione successiva;
- presenza del protocollo SPI: si è preferito il protocollo SPI rispetto all'I^2C in quanto si adatta meglio al progetto, dato che si ha la presenza di un solo master, identificato dal microcontrollore, che dovrà coordinare le due periferiche: sensore di temperatura o schermo LCD e modulo wireless, cioè gli slave. Inoltre il protocollo SPI permette una comunicazione full-duplex durante la quale sono inviati solo i bit corrispondenti al dato, senza bit aggiuntivi per il controllo, diminuendone così la complessità. Infine la comunicazione SPI, a differenza dell'I^2C non richiede la presenza di resistenze di pull-up sulle linee di comunicazione. Queste resistenze impattano negativamente sulle prestazioni energetiche poiché creano un percorso conduttivo verso massa e quindi un consumo statico di corrente. Esistono tecniche per evitare questo tipo di consumo, ma si è scelto comunque di utilizzare la comunicazione SPI per non aumentare la complessità del sistema.
- tempo di conversione del sensore di temperatura: il tempo necessario al sensore per ottenere il valore del dato di temperatura in formato digitale, deve essere il più basso possibile, in modo da limitare l'energia dissipata durante la fase di acquisizione della temperatura. Infatti è inutile che il

sistema abbia bassi consumi se poi impiega tempi molto lunghi per raggiungere il suo scopo.

2.3 Trasduttore

Il trasduttore è la parte incaricata alla trasformazione dell'energia disponibile nell'ambiente, in energia elettrica e rappresenta un elemento chiave per un sistema Energy Harvesting che va progettato in base al tipo di fonte disponibile. In questo progetto l'applicazione è stata studiata in maniera tale da non essere dipendente da un preciso tipo di trasduttore, quindi possono essere indifferentemente utilizzati trasduttori piezoelettrici, elettromagnetici o fotovoltaici. Per questo lavoro si è scelto di utilizzare come trasduttore dei sensori di tipo piezoelettrico, in quanto già presenti nel laboratorio in cui si è svolta la tesi.

Il trasduttore piezoelettrico, come già accennato, è un dispositivo in grado di trasformare l'energia meccanica, data dalle vibrazioni, in energia elettrica, grazie alla presenza di elementi dotati di proprietà piezoelettriche. Questa proprietà permette ai cristalli che ne sono forniti di polarizzarsi elettricamente in conseguenza ad una deformazione meccanica, o viceversa, visto che il fenomeno e reversibile, a deformarsi meccanicamente in conseguenza ad una polarizzazione elettrica. Perciò grazie a questa proprietà, se collegassimo il cristallo ad un circuito esterno, potremo osservare la generazione di una corrente elettrica durante la deformazione del cristallo.

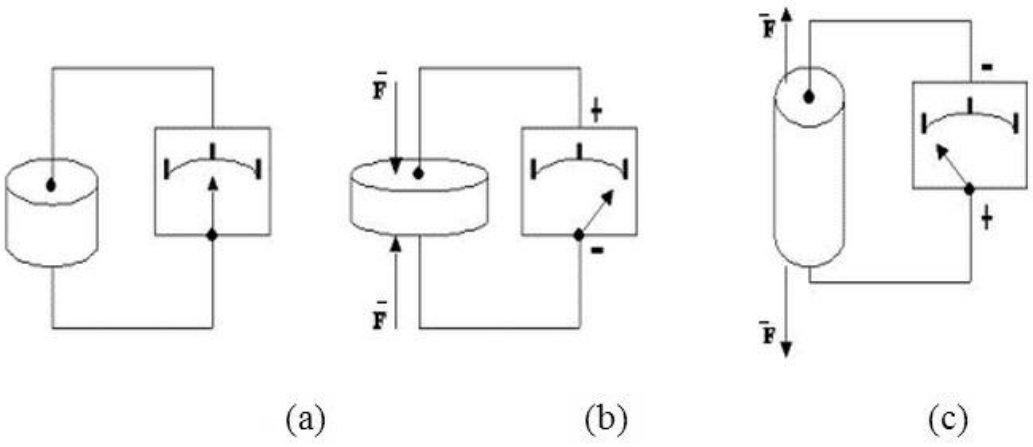

Figura 2.2: cristallo piezoelettrico (a) a riposo; (b) sottoposto ad una compressione; (c) sottoposto ad una trazione.[3]

2.4 Microcontrollore

Per il microcontrollore è stato scelto il PIC16LF1508 prodotto dalla Microchip.
La scelta di questo microcontrollore è stata fatta principalmente in base a:

- possibilità di utilizzare un oscillatore interno a bassa frequenza (31KHZ) per ridurre ulteriormente i consumi;
- basso consumo durante la normale modalità di funzionamento e sleep;
- intervallo della tensione di alimentazione, che va dai 1,8V ai 3,6V;
- presenza del protocollo SPI.

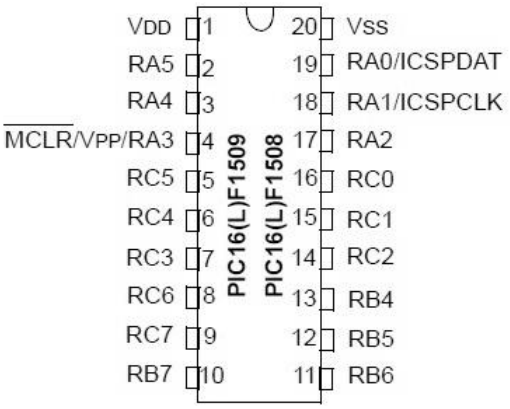

Figura 2.3: configurazione dei pin

2.4.1 MSSP

Il modulo MSSP (master synchronous serial port) è un'interfaccia seriale utilizzata per comunicare con le altre periferiche o con altri microcontrollori.

Il modulo MSSP può operare in due modalità:

- Serial Peripheral Interface (SPI);
- Inter-Integrated Circuit (I^2C™).

Per questo progetto, si userà la modalità SPI.

Il bus dell'interfaccia periferica seriale (Serial Peripheral Interface-SPI), è un bus sincrono, seriale che opera in modalità full-duplex per la comunicazione di dati e specifica quattro connessioni di segnale:

- Serial Clock (SCK);
- Serial Data Out (SDO);
- Serial Data In (SDI);
- Slave Select (/SS).

I dispositivi comunicano in un ambiente master/slave in cui l'unico dispositivo master, in questo caso il MCU, gestisce la comunicazione con i dispositivi slave: sensore, modulo wireless e schermo LCD, attraverso il Chip Select, noto come Slave Select, grazie al quale si seleziona l'unico slave con cui si vuole comunicare.

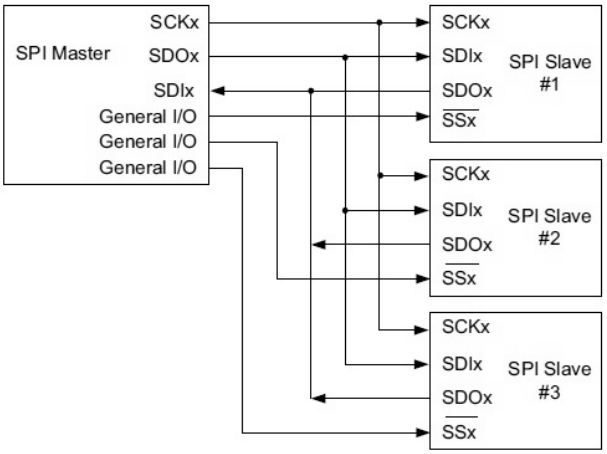

Figura 2.4: connessione fra un master e più slave

Le trasmissioni coinvolgono due registri a scorrimento (shift register) di dimensione 8 bit, uno nel master e l'altro nello slave. In entrambi i dispositivi, master e slave, il dato viene sempre fatto scorrere (shiftato) fuori un bit alla volta a partire dal bit più significativo (MSB). Inoltre si ha anche la presenza di un buffer che fornisce l'accesso indiretto al registro a scorrimento per andare a leggerlo e scriverlo.

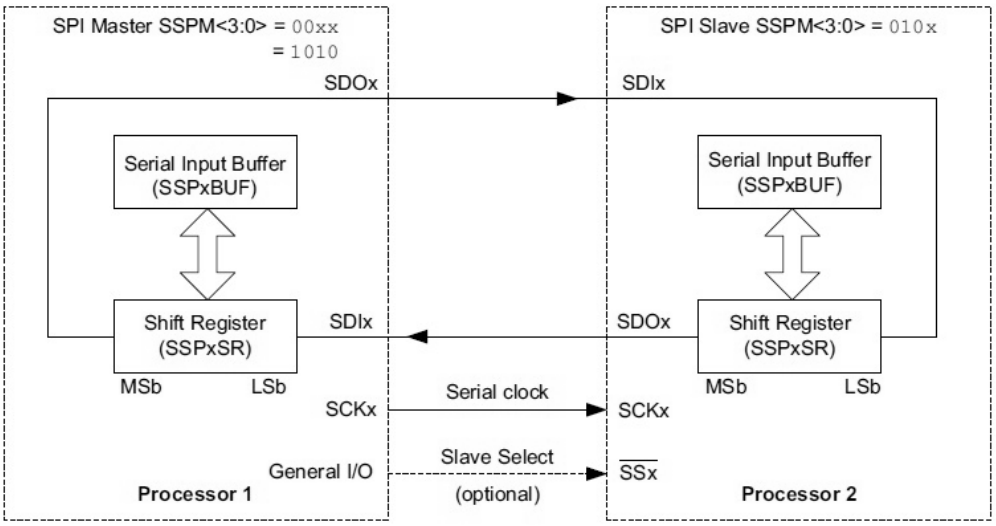

Figura 2.5: connessione SPI fra master e slave

Durante ogni ciclo di clock, si verifica una trasmissione di dati full-duplex. Ciò significa che mentre il dispositivo master invia il MSB dal suo registro a scorrimento (sul pin SDO) e il dispositivo slave legge (sul pin SDI) e salva questo bit come LSB nel suo registro, succede anche la cosa inversa, cioè lo slave invia il MSB, mentre il master lo legge e lo salva come LSB. Quindi, dopo che otto bit sono stati shiftati fuori, il master e lo slave si saranno scambiati i valori nei registri, che vengono poi trasferiti nel buffer.

Per questa comunicazione, è necessario che, sia il master che lo slave siano configurati con la stessa polarità del clock, cioè tutti i dispositivi devono far scorrere il dato su un fronte del clock e salvarlo sul fronte opposto. Per questo progetto la scelta della polarità del clock è stata dettata dalle esigenze degli slave, i quali salvano sul fronte di salita e scrivono sul fronte di discesa, come rappresentato in Figura 2.6.

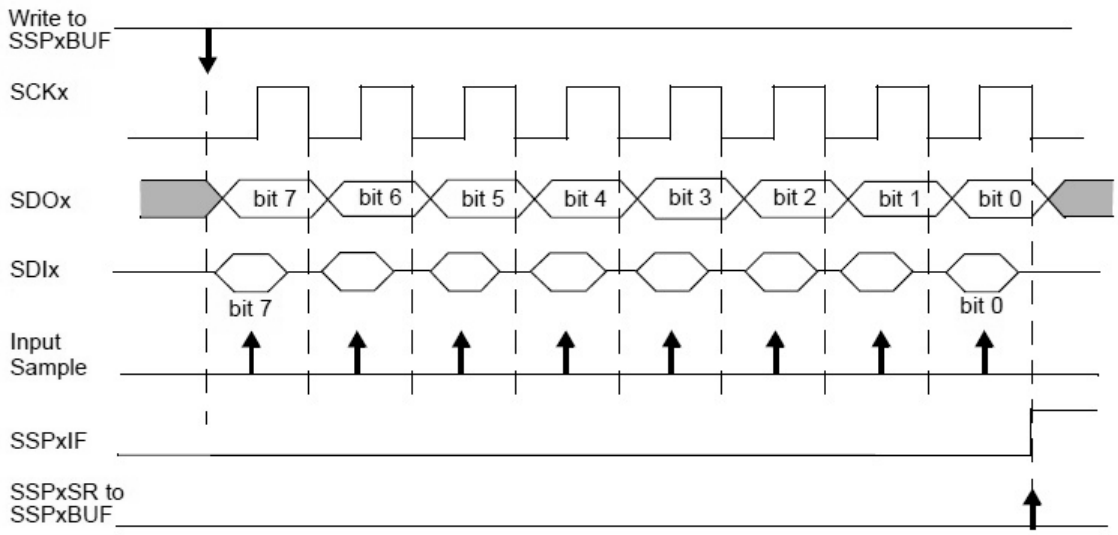

Figura 2.6: polarità del clock

Se ci sono più dati da scambiare, i registri a scorrimento saranno caricati con nuovi dati, attraverso il buffer, e il processo si ripeterà. Quando non ci sono più dati da trasmettere, il master smette di inviare il segnale di clock e deseleziona lo slave.

2.4.2 Oscillatore

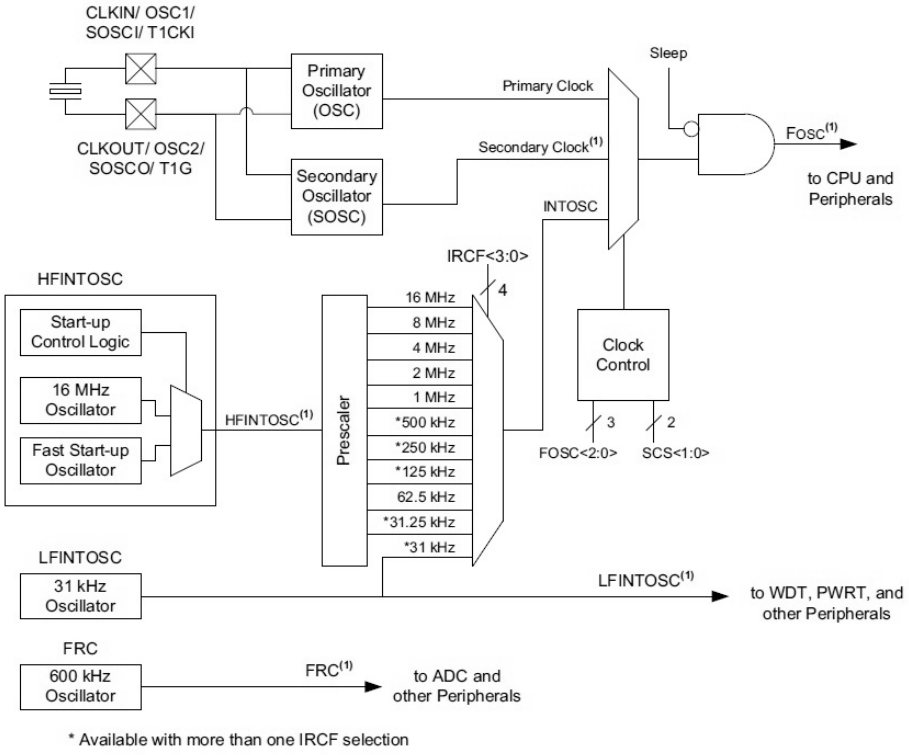

Figura 2.7: schema a blocchi semplificato per la generazione del clock

Le sorgenti del clock possono essere classificate come: esterne o interne:

- esterne: per la generazione del clock si affidano a circuiti esterni. Esempi sono: moduli di oscillatori, risonatori in cristallo di quarzo o ceramica e Resistore-condensatore;
- interne: la sorgente del clock è generata dal microcontrollore. Il blocco oscillatore interno contiene due oscillatori che vengono utilizzati per generare le sorgenti di clock interne del sistema: un oscillatore interno a 31 kHz a bassa frequenza ed uno a 16 MHz ad alta frequenza, da cui, attraverso un divisore di frequenza, è possibile ottenere frequenze minori.

Per il progetto si utilizza la sorgente interna in quanto permette di avere più libertà e facilità nell'andare a variare la frequenza da utilizzare. Infatti, in questa fase, ancora non si conosce la frequenza tale per cui si ha il minor consumo possibile, quindi con l'oscillatore interno risulterà più facile andare a variarla, dato che si dovrà solo programmare un registro, senza dover smontare e rimontare un oscillatore esterno.

2.4.3 Timer0

Il modulo Timer0 è stato utilizzato per introdurre nel programma dei ritardi usandolo nella modalità cronometro a 8 bit.

Nella modalità cronometro, il registro a 8 bit, associato al modulo Timer0, verrà incrementato ad ogni ciclo d'istruzione, se usato senza prescaler. Essendo un contatore ad 8 bit potrà assumere valori compresi fra 0 e 255. Quando il contatore raggiunge il valore 255 e arriva il momento dell'incremento, il registro si azzera, ricominciando il conteggio dal valore 0; questo evento è segnalato attraverso la configurazione di un flag. È possibile anche applicare un prescaler al Timer0, cioè dividere la frequenza dell'oscillatore interno per un valore programmato, per aumentare così il tempo fra un incremento ed un altro, e quindi avere un ritardo più lungo. Inoltre per avere più gradi di libertà è anche possibile impostare un valore iniziale al registro, in modo da far partire il conteggio da un valore prestabilito.

Calcolo per ottenere il ritardo voluto:

$$t = \frac{4}{f_{osc}} \times PS \times (256 - TMR0)$$

Dove:

- f_{osc}: frequenza dell'oscillatore interno;
- PS: prescale;
- TMR0: valore iniziale del registro.

2.4.4 Registri utilizzati

Di seguito verrà data una breve descrizione dei registri impostati e le decisioni prese per raggiungere l'obbiettivo della tesi:

- CONFIG1: configuration word 1
  programmato per selezionare l'uso dell'oscillatore interno;
- ANSELA: port A analog select register
  programmato per configurare i pin della porta A come I/O;
- ANSELB: port B analog select register
  programmato per configurare i pin della porta B come I/O;

- ANSELC: Port C Analog Select Register
  programmato per configurare i pin della porta C come I/O;
- OSCCON: Oscillator Control Register
  programmato per:
  - selezionare l'oscillatore interno come sorgente per la generazione del segnale di clock;
  - selezionare la frequenza dell'oscillatore interno.
- SSP1CON1: ssp control register 1
  programmato per:
  - selezionare la modalità master SPI e impostare la frequenza del clock come un quarto della frequenza dell'oscillatore interno($f_{ck}=f_{osc}/4$);
  - abilitare la porta seriale e configurare SCK, SDO, SDI e SS come sorgente dei pin corrispondenti;
  - Associare il livello basso allo stato di inattività del clock;
- SSP1STAT: ssp status register
  programmato per:
  - campionare il dato in ingresso a metà del tempo d'uscita dei dati (fronte di salita);
  - trasmissione del dato durante la transizione del clock dallo stato attivo a quello inattivo (fronte di discesa).
- TRISA: port A tri-state register
  programmato per selezionare i pin della porta A come input o output;
- TRISB: port B tri-state register
  programmato per selezionare i pin della porta B come input o output;
- TRISC: port C tri-state register
  programmato per selezionare i pin della porta C come input o output;
- OPTION_REG: Option Register
  programmato per:
  - selezionare il ciclo d'istruzione ($f_{osc}/4$) come temporizzazione per il Timer0;
  - assegnare il valore di prescaler al modulo Timer0;
  - Selezionare il valore di prescaler: 2, 4, 8, 16, 32, 64, 128, 256.

- TMR0:
  registro a 8 bit che contiene il valore associato al conteggio del modulo Timer0. Questo registro permette la scrittura per far partire così il conteggio da un valore prestabilito.
- INTCON: interrupt control register
  Questo registro contiene un flag che viene impostato per segnalare che il valore del conteggio è arrivato a 255, cioè il valore finale, e ricomincerà da zero. Il flag dovrà poi essere reimpostato manualmente.[4]

2.5 Sensore di temperatura

Come sensore di temperatura è stato scelto il TMP125 prodotto dalla Texas Instruments.

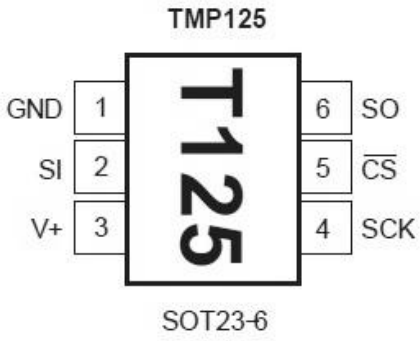

Figura 2.8: configurazione dei pin

Il TMP125 è un sensore di temperatura compatibile con il protocollo SPI, disponibile nel package SOT23-6 e non richiede componenti esterni, anche se si consiglia un condensatore di bypass di 0.1µF. Questo dispositivo è in grado di misurare la temperatura entro i 2°C di precisione nell'intervallo che va dai -25°C ai +85°C ed entro i 2.5°C di precisione nell'intervallo che va dai -40°C ai +125°C. La temperatura è convertita in una parola dati a 10 bit con risoluzione di 0,25 ° C e con un periodo di conversione di 120ms.

Il sensore di temperatura in questione è stato scelto principalmente per le seguenti caratteristiche:

- intervallo della tensione di alimentazione che va dai 2.7V ai 5.5V;
- bassa corrente di alimentazione, che nella modalità shutdown si riduce ad 1µA;
- presenza del protocollo SPI;
- ridotto numero di componenti esterni;
- tempo di conversione ridotto.

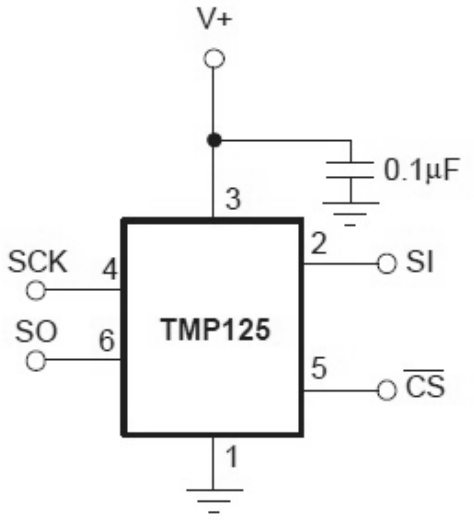

Figura 2.9: schema elettrico

2.5.1 Funzionamento

Questo sensore permette due modalità di funzionamento: la conversione continua della temperatura in dati digitali ogni 120ms o la modalità one-shot (un colpo) cioè il microcontrollore decide quando il sensore deve misurare e convertire la temperatura in formato digitale.

La scelta è ricaduta sulla modalità one-shot in quanto a causa della bassa energia a disposizione non si può pensare di misurare e inviare la temperatura molto frequentemente, in quanto, come detto in precedenza, dopo l'invio del dato si deve rispettare un tempo di inattività per permettere al condensatore di ricaricarsi, quindi far lavorare continuamente il sensore, sarebbe uno spreco inutile di energia. Inoltre c'è da dire che per la maggior parte delle applicazioni la temperatura varia molto lentamente, basti pensare alla temperatura in un ambiente esterno o all'interno di un'abitazione, quindi misurare la temperatura ogni 120ms darebbe solo informazioni ridondanti, perciò la scelta di questa modalità di funzionamento non andrà ad influire sulle prestazioni del progetto.

2.5.2 Comunicazione

Per utilizzare la modalità one-shot è necessario inviare il comando corrispondente, che consiste nell'abbassare il CS ed inviare nel pin SI una parola a 16 bit tutti a zero ad eccezione del quarto bit che deve essere a livello logico alto, come descritto in Figura 2.10. In questo modo si forza il sensore ad eseguire una singola conversione che durerà 60ms. Una volta che il sensore ha convertito la

temperatura in un dato digitale, questo viene salvato nel registro a scorrimento (shift register). Successivamente se si vuole mandare il sensore in modalità shutdown, e non si desidera avviare una nuova conversione (come in questo caso), il dato digitale verrà letto impostando il bit power-down a livello logico alto, vedi Figura 2.11.

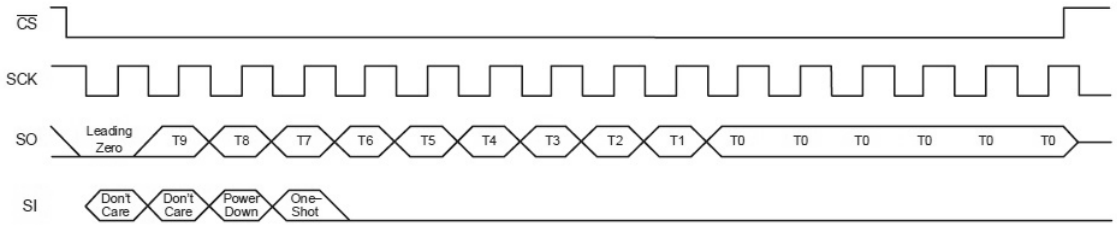

Figura 2.10: comando one-shot

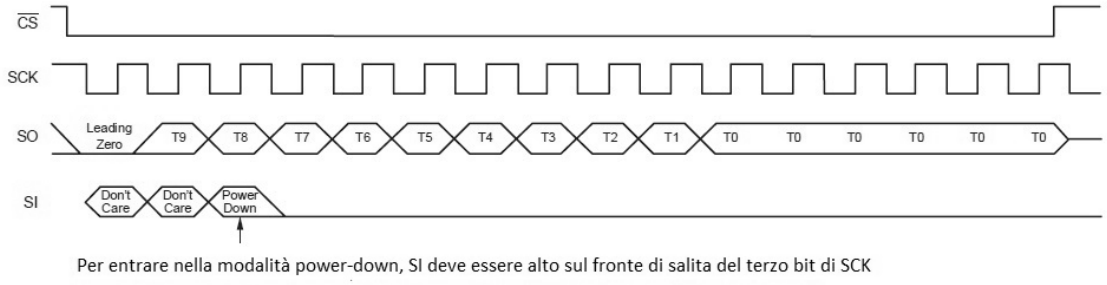

Figura 2.11: lettura dei dati.

2.5.3 Registro

Il registro di temperatura del TMP125 è di sola lettura e a 16-bit, in questo registro viene memorizzata la temperatura convertita più recente. È da notare che comunque la temperatura è rappresentata da solo 10 bit nel formato complemento a due. Il primo bit del registro, D15, è il leading zero (è impostato a livello logico basso) e i bit D4 fino a D0 sono gli stessi del bit D5 (vedi Figura 2.12). Il formato dei dati è sintetizzato nella Tabella 2.1. Quindi quando si dovrà andare a calcolare il valore di temperatura con segno in complemento a due, bisogna assicurarsi di usare solo i 10 bit di dato necessari.

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
0	T9	T8	T7	T6	T5	T4	T3	T2	T1	T0	T0	T0	T0	T0	T0

Figura 2.12: registro della temperatura

TEMPERATURE (°C)	DIGITAL OUTPUT D14…D5
+127	01 1111 1100
+125	01 1111 0100
+100	01 1001 0000
+75	01 0010 1100
+50	00 1100 1000
+25	00 0110 0100
+10	00 0010 1000
+0.25	00 0000 0001
0	00 0000 0000
−0.25	11 1111 1111
−25	11 1001 1100
−50	11 0011 1000
−55	11 0010 0100

Tabella 2.1: formato dei dati della temperatura

2.5.4 Caratteristiche elettriche

Con T_A = −40°C a +125°C e V_S = +2.7V a 5.5V, se non diversamente specificato

PARAMETRO	CONDIZIONI	MIN.	TYP.	MAX.	UNITÀ
INGRESSO IN TEMPERATURA					
Intervallo		-40		+125	°C
Accuratezza (errore di temperatura)	−25°C a +85°C −40°C a +125°C		±0.5 ±1.0	±2.0 ±2.5	°C °C
Risoluzione			10 0.25		Bits °C
INGRESSO/USCITA DIGITALE					
Tempo di Conversione	10-Bit		60		ms
Frequenza di Aggiornamento			120		ms
ALIMENTAZIONE					
Intervallo di Lavoro		2.7		5.5	V
Corrente a Riposo, con T_A=25°C	Bus seriale inattivo		36	50	µA
Sovratemperatura	−40°C a +125°C			60	µA
Corrente di Arresto			0.1	1	µA
Sovratemperatura				1	µA

PARAMETRO	CONDIZIONI	MIN.	TYP.	MAX.	UNITÀ
INTERVALLO DI TEMPERATURA					
Intervallo Specificato		-40		+125	°C
Intervallo di Lavoro		-55		+125	°C
Intervallo di Conservazione		-60		+150	°C

Tabella 2.2: caratteristiche elettriche

Il TMP125 è compatibile al protocollo SPI ed il periodo minimo di SCK risulta 100ns, quindi la massima frequenza da utilizzare con il sensore risulta 10 MHz.[5]

2.6 Modulo wireless

Per il modulo wireless si è scelto l'MRF89XAM9A prodotto dalla Microchip.

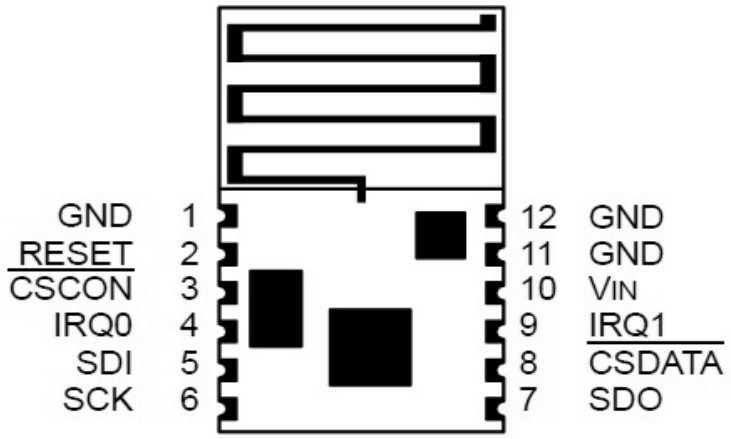

Figura 2.13: configurazione dei pin

La scelta di questo modulo è stata fatta principalmente in base a:

- basso consumo durante la modalità trasmissione e sleep;
- Intervallo della tensione di alimentazione, che va dai 2.1V ai 3.6V;
- presenza del protocollo SPI;
- ridotto numero di componenti esterni.

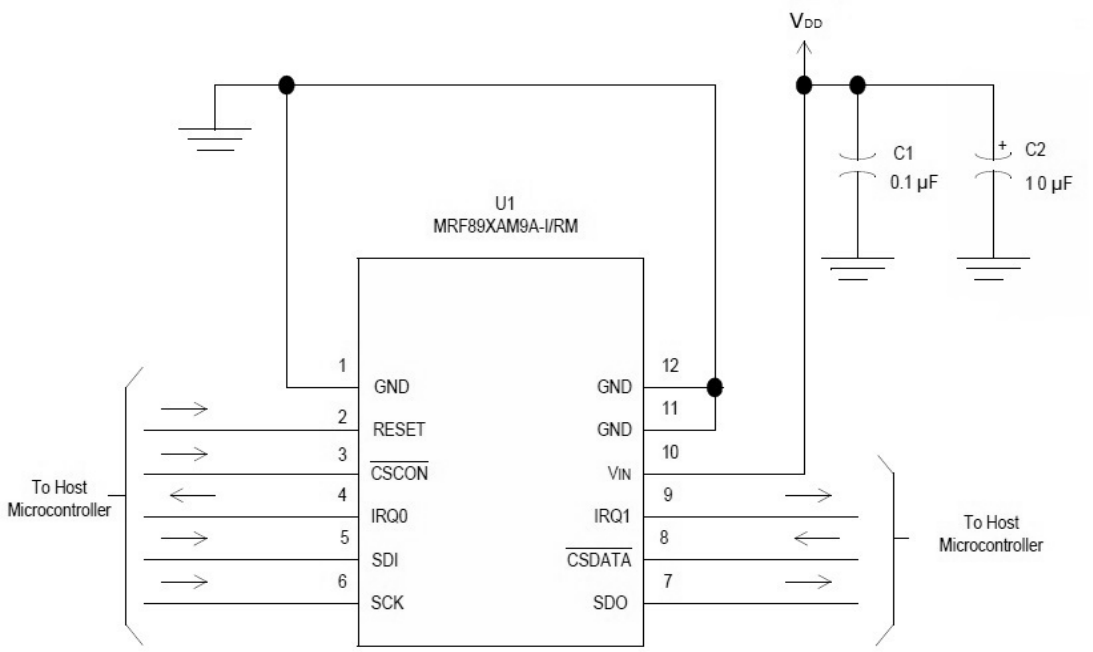

Figura 2.14:schema elettrico

2.6.1 Serial peripheral interface - SPI

Il modulo wireless comunica con il microcontrollore tramite una porta SPI a 4 fili come dispositivo slave. Il modulo SPI è costituito, come illustrato nella Figura 2.15, dai due seguenti sotto-blocchi:

1. SPI CONFIG: Questo sotto-blocco è utilizzato in tutte le modalità di funzionamento per leggere e scrivere i registri di configurazione che controllano tutti i parametri del dispositivo (modalità operativa, frequenza, bit rate, ecc…);
2. SPI DATA: Questo sotto-blocco è utilizzato per leggere e scrivere i byte di dati da e verso il buffer.

Entrambe le SPI sono configurate in modalità slave. Hanno i pin di selezione separati (CSCON e CSDAT), ma condividono quelli restanti:

- SCK (SPI Clock): segnale di clock fornito dal microcontrollore.
- SDI (ingresso SPI): Segnale di ingresso per i dati forniti dal microcontrollore.
- SDO (uscita SPI): Segnale di uscita per i dati forniti dal MRF89XAM9A.

Come indicato nella Tabella 2.3, può essere selezionata una sola interfaccia alla volta con CSCON che detiene la priorità:

$\overline{\text{CSDAT}}$	$\overline{\text{CSCON}}$	SPI
0	0	CONFIG
0	1	DATA
1	0	CONFIG
1	1	None

Tabella 2.3: Pin di Selezione e Configurazione nell'SPI

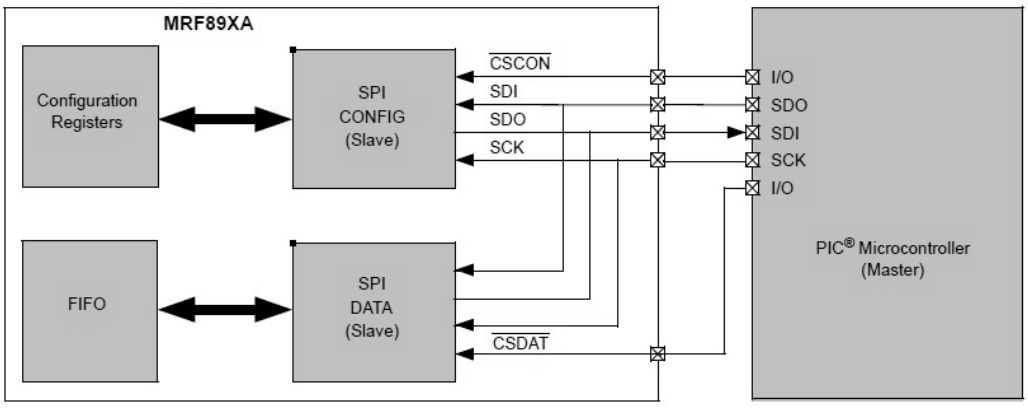

Figura 2.15: panoramica sull'SPI e connessione con il microcontrollore

Scrittura del registro: il diagramma dei tempi nella Figura 2.16 illustra la procedura che il microcontrollore deve seguire per scrivere un valore nel registro di configurazione. Il nuovo valore nel registro è veramente effettivo dal fronte di salita del CSCON.

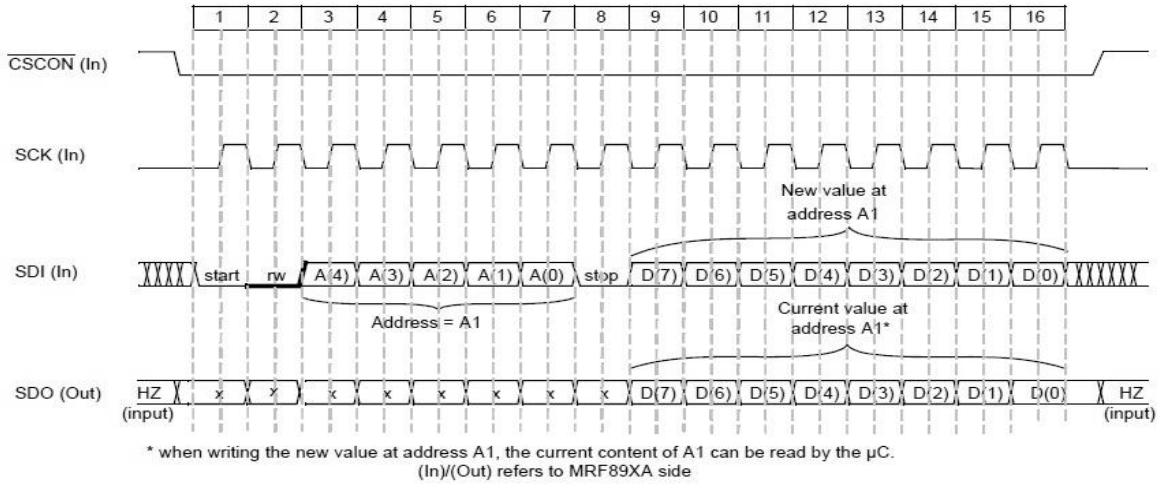

Figura 2.16: sequenza di scrittura del registro

Lettura del registro: il diagramma dei tempi nella Figura 2.17 illustra la procedura che il microcontrollore deve seguire per leggere un valore nel registro di configurazione.

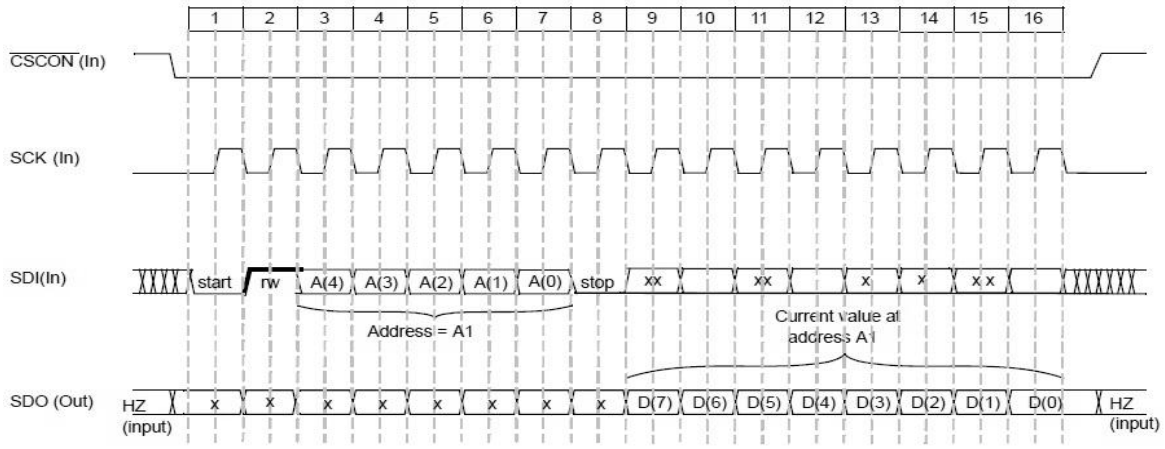

Figura 2.17: sequenza di lettura del registro

Note: Durante la lettura o la scrittura in successione di più registri, tra due sequenze di scrittura o lettura, non è obbligatorio riportare CSCON alto.

I byte vengono alternativamente considerati indirizzo e valore.

Dopo l'accensione, i registri sono impostati su valori predefiniti.

I valori programmati vengono mantenuti durante la modalità Sleep.

Scrittura dei Byte (prima o durante la trasmissione): il diagramma dei tempi nella Figura 2.18 illustra la procedura che il microcontrollore deve seguire per scrivere dei byte nel buffer.

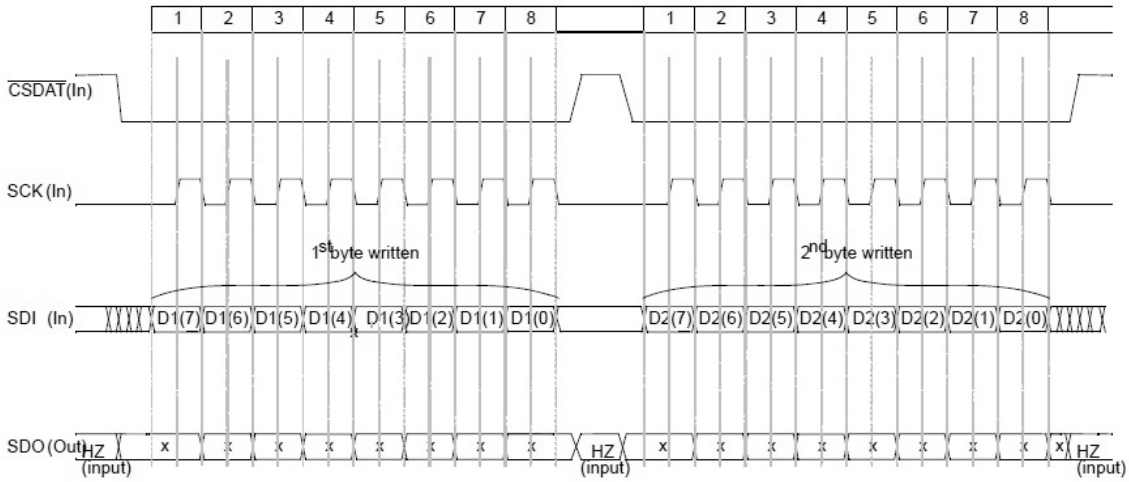

Figura 2.18: sequenza di scrittura dei byte (esempio per 2 byte)

Lettura dei Byte (dopo o durante la trasmissione): il diagramma dei tempi nella Figura 2.19 illustra la procedura che il microcontrollore deve seguire per leggere i byte dal buffer.

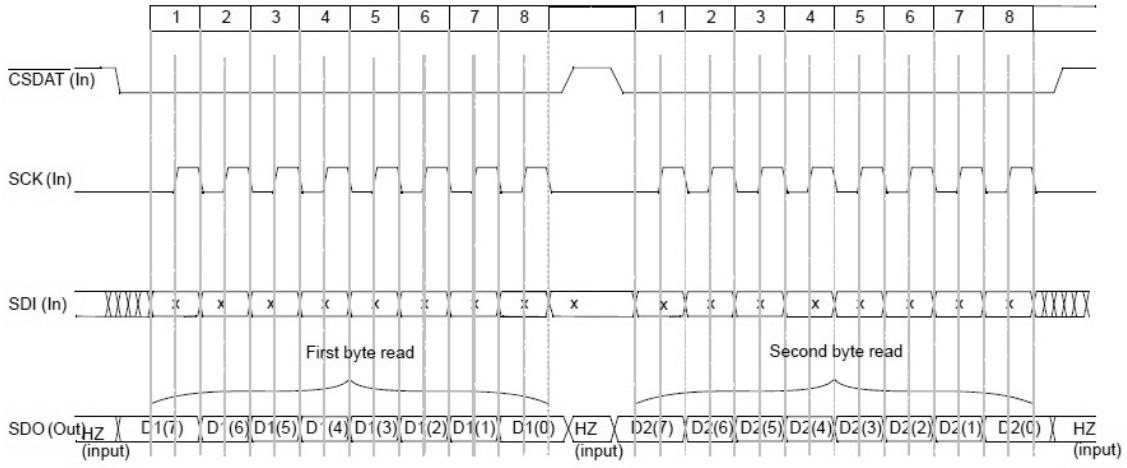

Figura 2.19: sequenza di lettura dei byte (esempio per 2 byte)

Nota: Tra ogni byte letto o scritto è obbligatorio riportare CSCON alto.

2.6.2 Fifo ed il shift register

Durante il funzionamento del modulo wireless, i dati da trasmettere e quelli che sono stati ricevuti vengono memorizzati nel buffer First In First Out (FIFO). Il buffer è accessibile tramite l'interfaccia dati SPI e fornisce vari interrupt per la gestione del trasferimento.

Ogni cella del FIFO ha dimensione di 1 byte (8 bit); pertanto esegue solo operazioni parallele sui byte, mentre le funzioni del demodulatore sono eseguite serialmente. Viene quindi impiegato un registro a scorrimento (SR) per interfacciare il demodulatore ed il buffer. Nella modalità trasmissione il registro a scorrimento prende i byte dal buffer e li invia serialmente al modulatore (MSB per primo), alla bit rate programmata; allo stesso modo, in ricezione, prende bit per bit dal demodulatore e li scrive byte per byte nel FIFO. Queste procedure sono mostrate nella Figura 2.20.

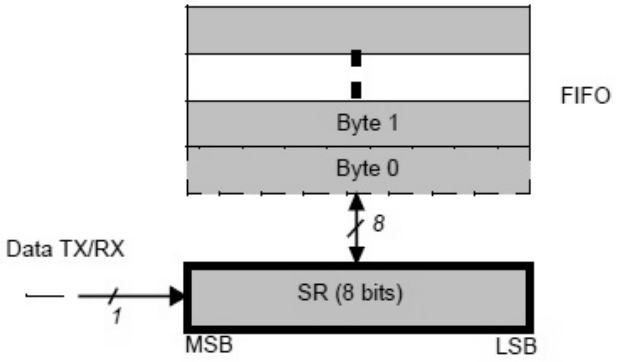

Figura 2.20:ill FIFO ed il Shift Register

2.6.3 Reset d'accensione:

Se l'applicazione richiede lo scollegamento dell'alimentazione dal modulo wireless, l'utente deve attendere 10 ms la fine del ciclo POR (power-on reset) prima di iniziare la comunicazione utilizzando la SPI. La *Figura 2.21* illustra la tempistica per il ciclo POR.

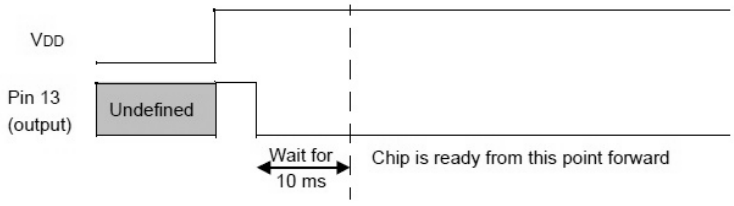

Figura 2.21:diagramma dei tempi del ciclo POR

2.6.4 Funzionamento

Il modulo wireless ha due differenti modi di operare: Buffer e Packet. Queste modalità sono molto simili fra loro, entrambe infatti permettono la trasmissione e la ricezione interfacciandosi con il buffer FIFO, con la differenza però che nella modalità packet, il dato da inviare può essere caricato nel buffer mentre il modulo wireless è in stand-by, ottenendo così un evidente risparmio energetico, ed è proprio per questa ragione che si è deciso di utilizzare la modalità packet, che verrà descritta qui di seguito.

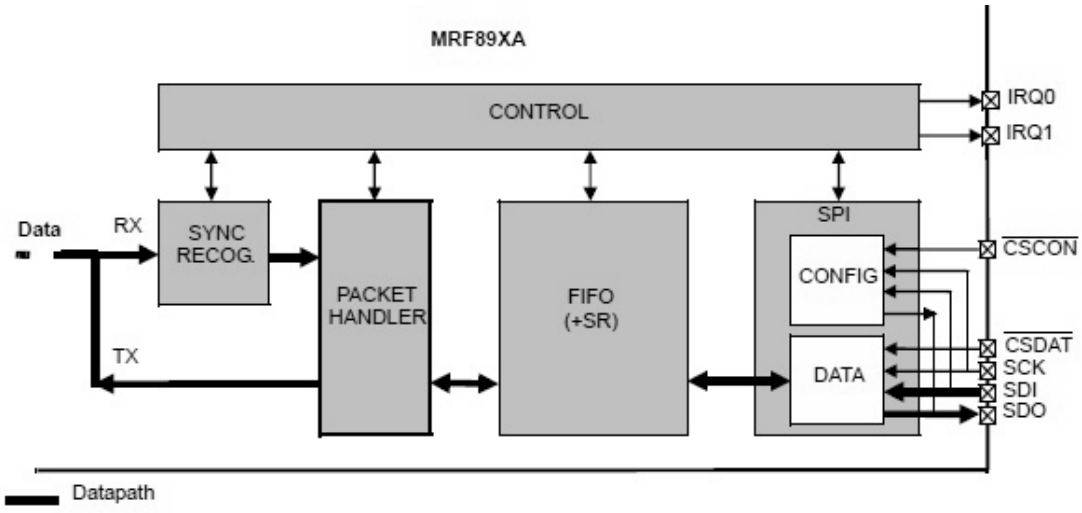

Figura 2.22: schema a blocchi per la modalità packet

Formato del pacchetto

Sono supportati due formati per I pacchetti: lunghezza fissa e variabile. La dimensione massima del payload è limitata dalla dimensione del buffer selezionata (16, 32, 48 o 64 byte). Per il progetto si è deciso di utilizzare pacchetti a lunghezza fissa, dato che si conosce a priori la lunghezza del messaggio da inviare, infatti l'informazione della temperatura è contenuta in 2 byte.

formato del pacchetto a lunghezza fissa: la lunghezza del pacchetto è fissata in anticipo ed è limitata dalla dimensione del buffer selezionata. Sia che, solo il TX, o solo il RX o entrambi siano impostati su questa modalità, tutti I nodi saranno programmati con la stessa lunghezza del pacchetto.

Il formato del pacchetto a lunghezza fissa è illustrata nella Figura 2.23 e contiene i seguenti campi:

• preambolo (1010 ...);
• parola di sincronizzazione (Network ID);
• byte di indirizzamento opzionale (Node ID);
• dati del messaggio;
• 2-byte opzionali di controllo CRC

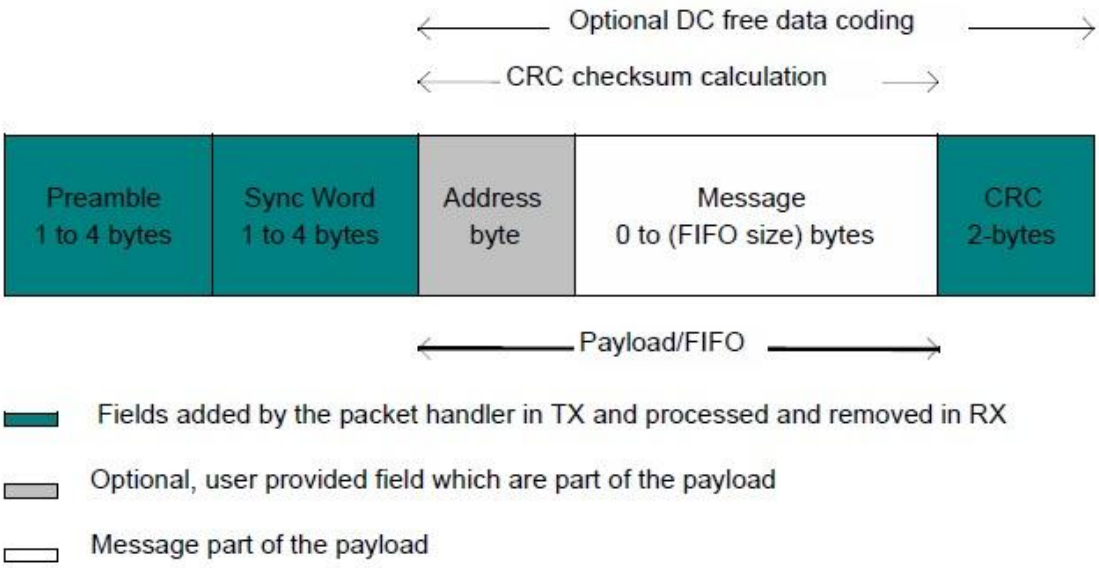

Figura 2.23:formato del pacchetto a lunghezza fissa

Elaborazione in trasmissione

Per eseguire la trasmissione si deve inizialmente scrivere il dato di temperatura nel buffer del modulo mentre questo è in stand-by, successivamente si entra nella modalità trasmissione vera e propria, in cui il modulo andrà a verificare la condizione programmata per iniziare l'invio del dato, cioè il buffer non è vuoto, e se questa viene soddisfatta, il pacchetto viene inviato.

Per gestire la trasmissione, vengono forniti due interrupt programmabili, IRQ0 e IRQ1. Per il progetto se ne è utilizzato uno per segnalare quando il buffer è pieno o vuoto (/FIFOEMPTY) e l'altro per segnalare la fine della trasmissione (TXDONE). In particolare, l'interrupt TXDONE è molto importante dato che permette di vedere se l'energia a disposizione è sufficiente per compiere correttamente la trasmissione.

In trasmissione, i pacchetti vengono costruiti dinamicamente dal modulo stesso, grazie al blocco packet handler, eseguendo le seguenti operazioni sul payload disponibile nel buffer:

1. vengono aggiunti un numero programmabile di byte di preambolo;
2. viene aggiunta una parola programmabile di sincronizzazione;
3. Opzionalmente viene calcolato il CRC sul payload;
4. Codifica opzionale per i messaggi (Manchester or Whitening).

In questa modalità, l'utente dovrà fornire solo il payload nel buffer.

Elaborazione in ricezione

In modalità ricezione, il modulo estrae il payload dal messaggio ricevuto grazie al blocco packet handler, eseguendo le seguenti operazioni:

1. riceve e toglie il preambolo;
2. rileva la parola di sincronizzazione e la toglie;
3. opzionalmente decodifica i dati;
4. opzionalmente controlla il byte di indirizzo;
5. opzionalmente controlla i byte CRC e riflette il risultato su un apposito flag.

Così facendo solo il payload è reso disponibile nel buffer.

Come in trasmissione, per gestire la ricezione, vengono forniti due interrupt programmabili, IRQ0 e IRQ1. Anche in questo caso, se ne è utilizzato uno per segnalare quando il FIFO è pieno o vuoto (/FIFOEMPTY), l'altro invece è stato programmato per segnalare quando il numero di byte scritti nel buffer raggiunge la soglia programmata, cioè due byte (FIFO_THRESHOLD). Quest'ultimo interrupt quindi, ha il compito fondamentale di segnalare l'arrivo del dato di temperatura, espresso in due byte, e quindi il momento dopo al quale è possibile accedere al FIFO per andare a leggerlo.

Filtraggio dei pacchetti

Il blocco packet handler del modulo wireless, offre diversi meccanismi per il filtraggio dei pacchetti garantendo così al microcontrollore la ricezione dei soli pacchetti utili e riducendo di conseguenza in modo significativo il consumo di energia del sistema e la complessità del software.

Di seguito verrà data una descrizione generale di questi meccanismi per il filtraggio dei pacchetti:

1. Parola di sincronizzazione:
   la parola di sincronizzazione per il filtraggio è utilizzata per identificare l'inizio del payload e anche per l'identificazione di rete. È possibile configurare la parola di sincronizzazione andando a deciderne il valore, la dimensione e la tolleranza d'errore. Ogni pacchetto ricevuto che non inizia con la parola di sincronizzazione programmata viene automaticamente scartato e non viene generato alcun interrupt. Al contrario, quando viene rilevata, si avvia automaticamente la ricezione del payload e vengono generati i corrispondenti interrupt. Inizialmente si è preferito non metterla per ridurre il tempo impiegato per trasmettere il pacchetto e quindi per ridurre il consumo energetico del trasmettitore, ma una volta implementato il sistema, ci si è visti costretti ad inserirla in quanto altrimenti il ricevitore, a causa delle interferenze, riceveva dati indesiderati;
2. Lunghezza:
   questo campo non è utilizzato nella modalità a lunghezza fissa, ma solo in quella a lunghezza variabile dove non si conosce a priori le dimensioni del payload;
3. Indirizzo:
   questo campo non è utilizzato, dato che si ha la presenza di solo due nodi nella rete, quindi non c'è bisogno di segnalare con quale nodo si vuole comunicare.
4. CRC:
   Questi byte sono utilizzati per controllare l'integrità del messaggio. Anche in questo caso si è preferito non utilizzare questa funzione in quanto si è supposto inizialmente, e poi verificato sperimentalmente, che non ci sarebbe stata abbastanza energia per effettuare questo tipo di controllo. Infatti dopo la trasmissione il modulo wireless dovrebbe passare nella modalità ricezione e aspettare di sapere se il messaggio arrivato al ricevitore sia integro, e in caso contrario dovrebbe rimandarlo, ma tutto questo non è possibile a causa della bassa energia a disposizione e dell'alto consumo del modulo nella modalità trasmissione/ricezione.

2.6.5 Registri del modulo wireless

Attraverso la configurazione dei registri del modulo wireless viene deciso sostanzialmente come quest'ultimo dovrà operare, cioè vengono scelti i parametri relativi alla trasmissione, ricezione, parola di sincronizzazione, interrupt, gestione del pacchetto e altri parametri generali.

Di seguito verrà data una breve descrizione dei registri configurati:

- GCCONREG: general configuration register
  per selezionare:
  - La modalità di funzionamento del modulo: trasmissione, ricezione, sleep, stand-by frequency synthesizer;
  - La banda di frequenza nel range sub-GHz: 915-928 MHz.
- DMODREG: data and modulation configuration register
  per selezionare:
  - Il tipo di modulazione da utilizzare: FSK o OOK;
  - La modalità di ricezione/trasmissione dei dati con il microcontrollore: packet.
- FIFOCREG: fifo configuration register
  per selezionare:
  - La dimensione del buffer: 16 byte;
  - La soglia nel buffer per la generazione dell'interrupt in ricezione: 2 byte.
- FTXRXIREG: fifo transmit and receive interrupt request
  per selezionare:
  - la sorgente dell'interrupt IRQ0 nella modalità ricezione e stand-by: /FIFOEMPTY, in questo modo nel pin corrispondente si ha "0" logico se il buffer è vuoto o "1" logico in caso contrario;
  - la sorgente dell'interrupt IRQ1 nella modalità ricezione e stand-by: FIFO_THRESHOLD, in questo modo nel pin corrispondente si ha la generazione di un interrupt quando i dati scritti nel buffer sono pari alla soglia programmata nel registro FIFOCREG;
  - La sorgente dell'interrupt IRQ1 nella modalità trasmissione: TXDONE, in questo modo nel pin corrispondente si ha la generazione di un interrupt al termine della trasmissione.

- FTPRIREG: Fifo Transmit Pll And Rssi Interrupt Request
  per selezionare la condizione per l'inizio della trasmissione e la sorgente dell'interrupt IRQ1 nella modalità trasmissione: si è deciso di mappare IRQ0 come /FIFOEMPTY in modo che la trasmissione inizi quando il buffer non è vuoto;
- SYNCREG: Sync Control Register
  per selezionare:
  - se abilitare o meno la parola di sincronizzazione: si è deciso di abilitarla per le ragioni descritte in precedenza;
  - la lunghezza della parola di sincronizzazione: 32 bit;
  - il numero di errori tollerati nel riconoscimento delle parole di sincronizzazione in ricezione: 0 errori.
- SYNCV31REG: Sync Value First Byte Configuration Register
  per selezionare il valore della prima parola di sincronizzazione: 'S';
- SYNCV23REG: Sync Value Second Byte Configuration Register
  per selezionare il valore della seconda parola di sincronizzazione: 'Y';
- SYNCV15REG: Sync Value Third Byte Configuration Register
  per selezionare il valore della terza parola di sincronizzazione: 'N';
- SYNCV07REG: Sync Value Fourth Byte Configuration Register
  per selezionare il valore della quarta parola di sincronizzazione: 'C';
- TXCONREG: Transmit Parameter Configuration Register
  per selezionare la potenza di trasmissione: -8, -5, -1, +2, +1, +4, +7, +10, +13 dBm;
- PLOADREG: Payload Configuration Register
  per selezionare:
  - se abilitare o no la codifica/decodifica Manchester: disabilitata;
  - la lunghezza del payload del pacchetto: 2 byte.
- PKTCREG: Packet Configuration Registe
  per selezionare:
  - il formato della lunghezza del pacchetto: lunghezza fissa;
  - la dimensione della parola di preambolo: 1 byte.

- FCRCREG: Fifo Crc Configuration Register
  per selezionare il tipo di accesso al buffer durante la modalità stand-by: lettura (in ricezione) o scrittura (in trasmissione).

2.6.6 Caratteristiche elettriche:

PARAMETRO	MINIMO	MASSIMO	UNITÀ
Temperatura di Funzionamento	-40	+85	°C
Alimentazione	2.1	3.6	V

Tabella 2.4: condizioni di funzionamento consigliate

MODALITÀ DI FUNZIONAMENTO	TIPICO	MASSIMO	UNITÀ	CONDIZIONE
Sleep	0.1	2	µA	Clock e tutti I blocchi disabilitati
Idle	65	80	µA	Oscillatore e baseband abilitati
Frequency Synthesizer	1.3	1.7	mA	Frequency Synthesizer in esecuzione
TX	25 16	30 21	mA mA	potenza TX= +10 dBm potenza TX= +1 dBm
RX	3.0	3.5	mA	-

Tabella 2.5: consumo di corrente

PARAMETRO	MINIMO	TIPICO	MASSIMO	UNITÀ	CONDIZIONE
Intervallo di Frequenza	863	-	870	MHz	Programmabile
	902	-	928	MHz	
	950	-	960	MHz	
Bit Rate (FSK)	1.56	-	200	Kbps	NRZ
Bit Rate (OOK)	1.56	-	32	kbps	NRZ
Tempo di Risveglio dell'oscillatore	-	1.5	5	ms	Dalla modalità Sleep
Tempo di risveglio del Frequency Synthesizer		500	800	µs	Dalla modalità Stand-by

Tabella 2.6: parametri caratteristici AC del PLL

PARAMETRO	MINIMO	TIPICO	MASSIMO	UNITÀ	CONDIZIONE
Potenza d'uscita dell'antenna, programmabile con 8 passi tipicamente di 3 dB	-	+12.5	-	dBm	Massima potenza impostabile
	-	-8.5	-	dBm	Minima potenza impostabile
Tempo di riattivazione del trasmettitore	-	120	500	µs	Dalla modalità Frequency Synthesizer alla trasmissione
Tempo di riattivazione del trasmettitore	-	600	900	µs	Dalla modalità stand-by alla trasmissione
Tempo di riattivazione del ricevitore	-	280	500	µs	Dalla modalità Frequency Synthesizer alla ricezione
Tempo di riattivazione del ricevitore	-	600	900	µs	Dalla modalità stand-by alla ricezione

Tabella 2.7: caratteristiche AC in trasmissione e ricezione

PARAMETRO	MINIMO	TIPICO	MASSIMO	UNITÀ
Frequenza di clock per la configurazione della SPI	-	-	6	MHz
Frequenza di clock per i dati della SPI	-	-	1	MHz

Tabella 2.8: specifiche sulla tempistica della comunicazione SPI

2.6.7 Tempi di commutazione e procedure

Come dispositivo ultra low-power, l'MRF89XAM9A può essere configurato per abbassare il consumo medio. Verranno mostrate le transizioni fra i modi, ottimizzate per ridurre al minimo il consumo.

Ottimizzazione del ciclo in ricezione:

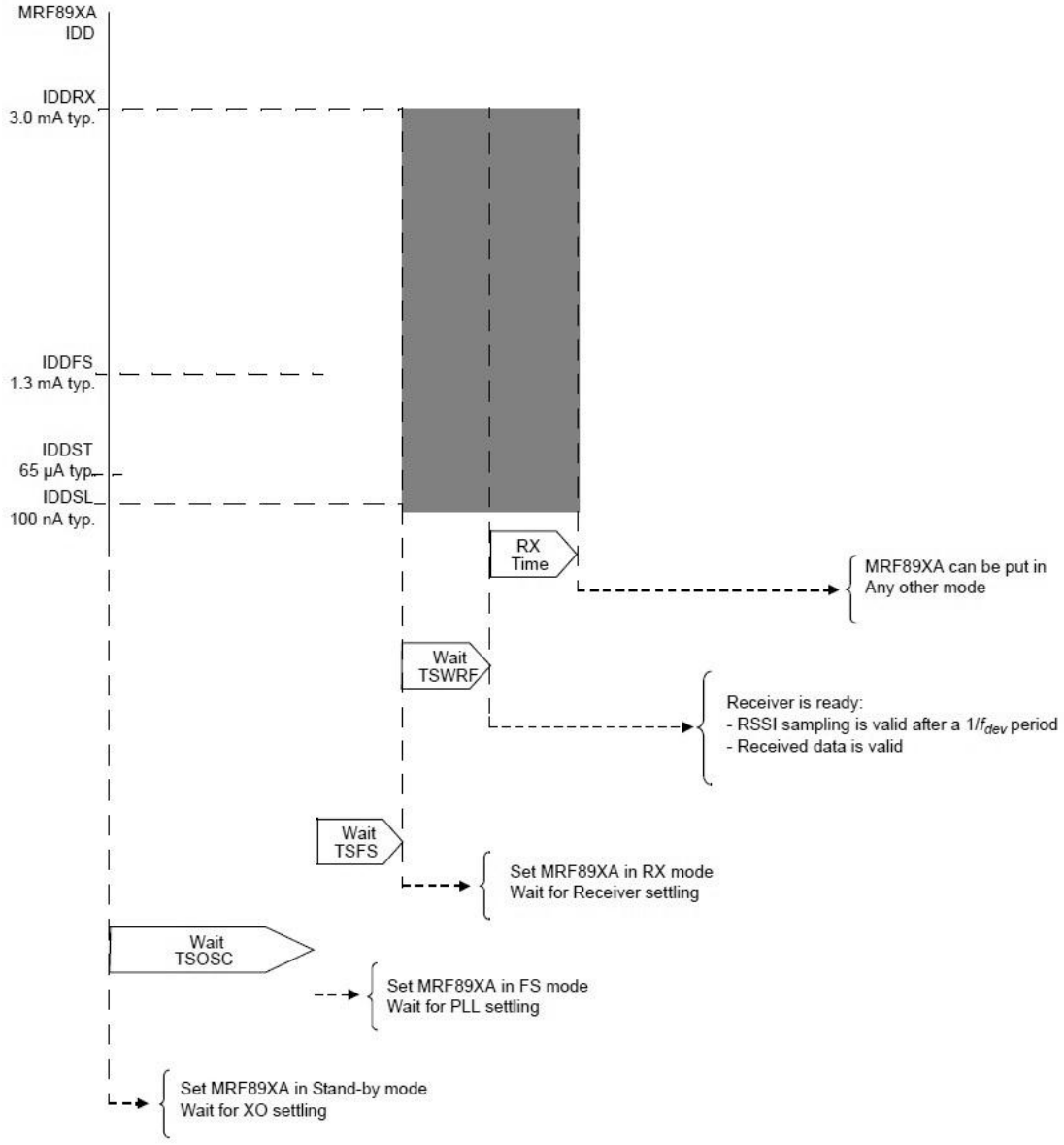

Figura 2.24: ottimizzazione del ciclo in ricezione

Ottimizzazione del ciclo in trasmissione:

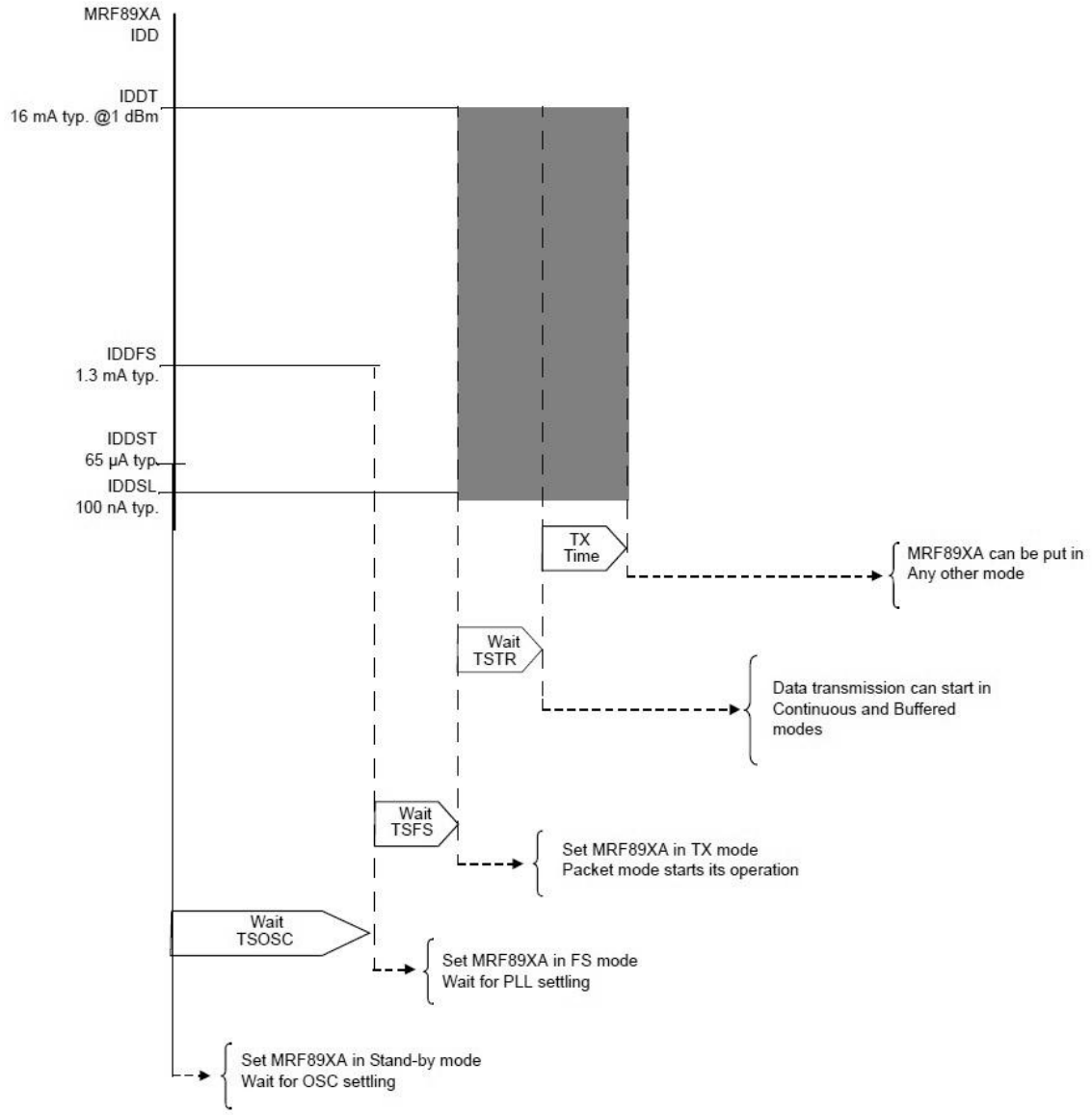

Figura 2.25: ottimizzazione del ciclo in trasmissione

[6] [7]

2.7 Regolatore di tensione

Per il regolatore di tensione si è scelto il TPS780270200 prodotto dalla Texas Instruments. La scelta di questo regolatore è stata presa principalmente in base al suo basso consumo e la tensione fornita in uscita.

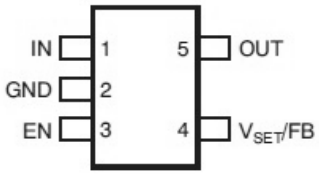

Figura 2.26: configurazione dei pin

La tensione d'uscita dei regolatori, della famiglia TPS780, può essere selezionata attraverso il pin V_{SET} fra due valori fissi oppure può essere regolata attraverso il collegamento di due resistenze esterne. Per il progetto si è preferito utilizzare un componente in grado di fornire una tensione d'uscita fissa di 2.7V, senza l'aggiunta di componenti esterni, per ridurre al minimo il consumo di corrente, scegliendo così il TPS780270200. La decisione di utilizzare come tensione d'uscita 2.7V è stata dettata dalla tensione minima di funzionamento del sensore di temperatura, corrispondente a 2.7V, che comunque è compatibile con il range di funzionamento del microcontrollore e del sensore.

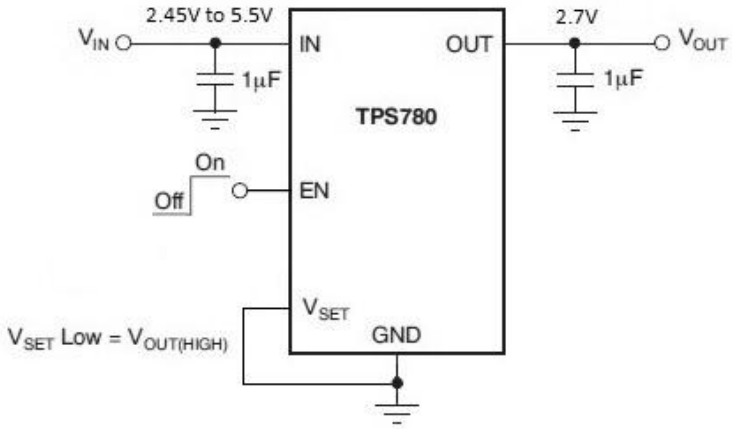

Figura 2.27: Schema elettrico

Come si può vedere dallo schema elettrico in Figura 2.27, per il corretto funzionamento del regolatore, cioè per avere in uscita una tensione regolata di 2.7V, si è collegato il pin V_{SET} a massa, inoltre, in ingresso è necessario avere un intervallo di tensione che va dai 2.45V ai 5.5V.[8]

2.8 Schema elettrico globale

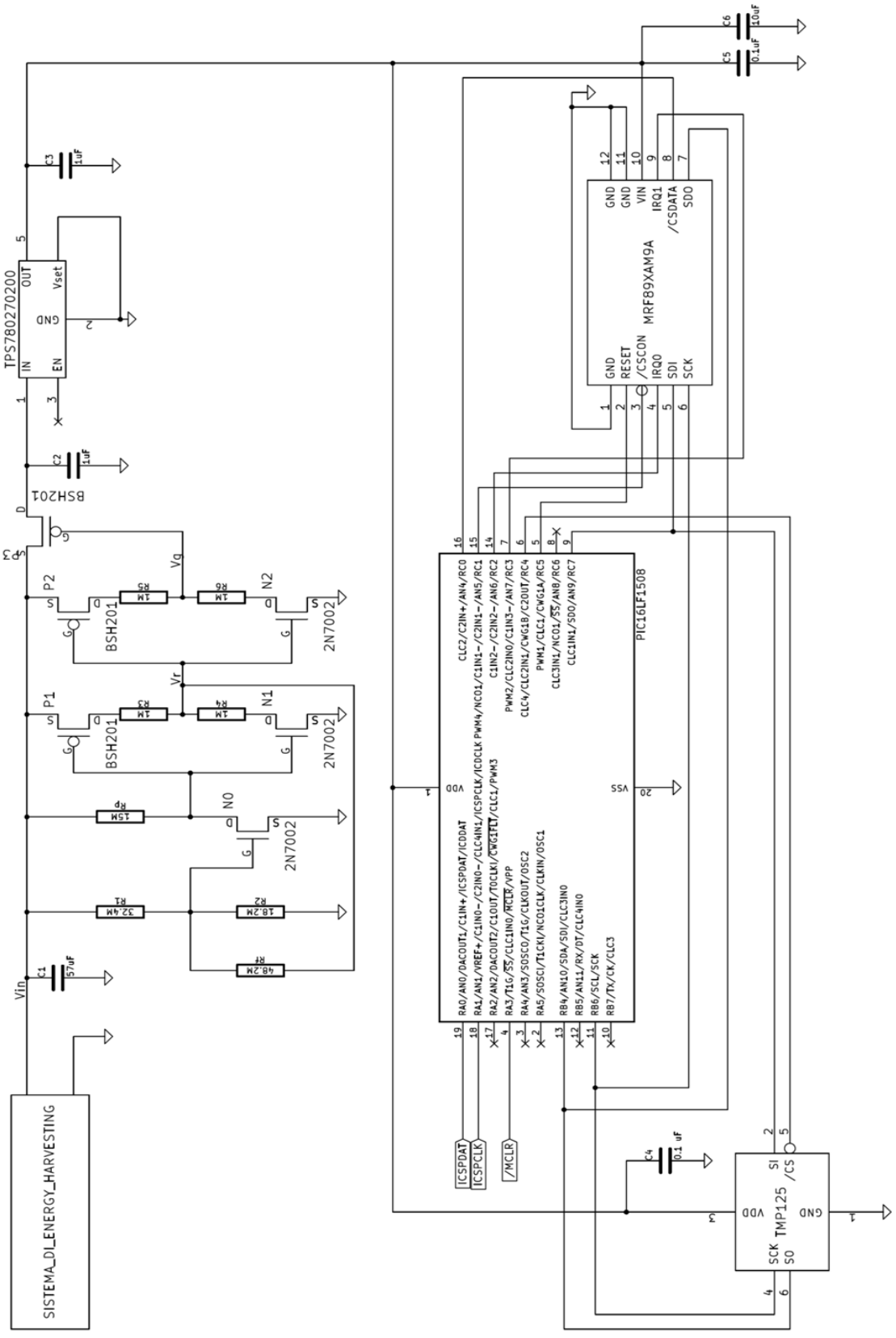

Figura 2.28: schema elettrico trasmettitore

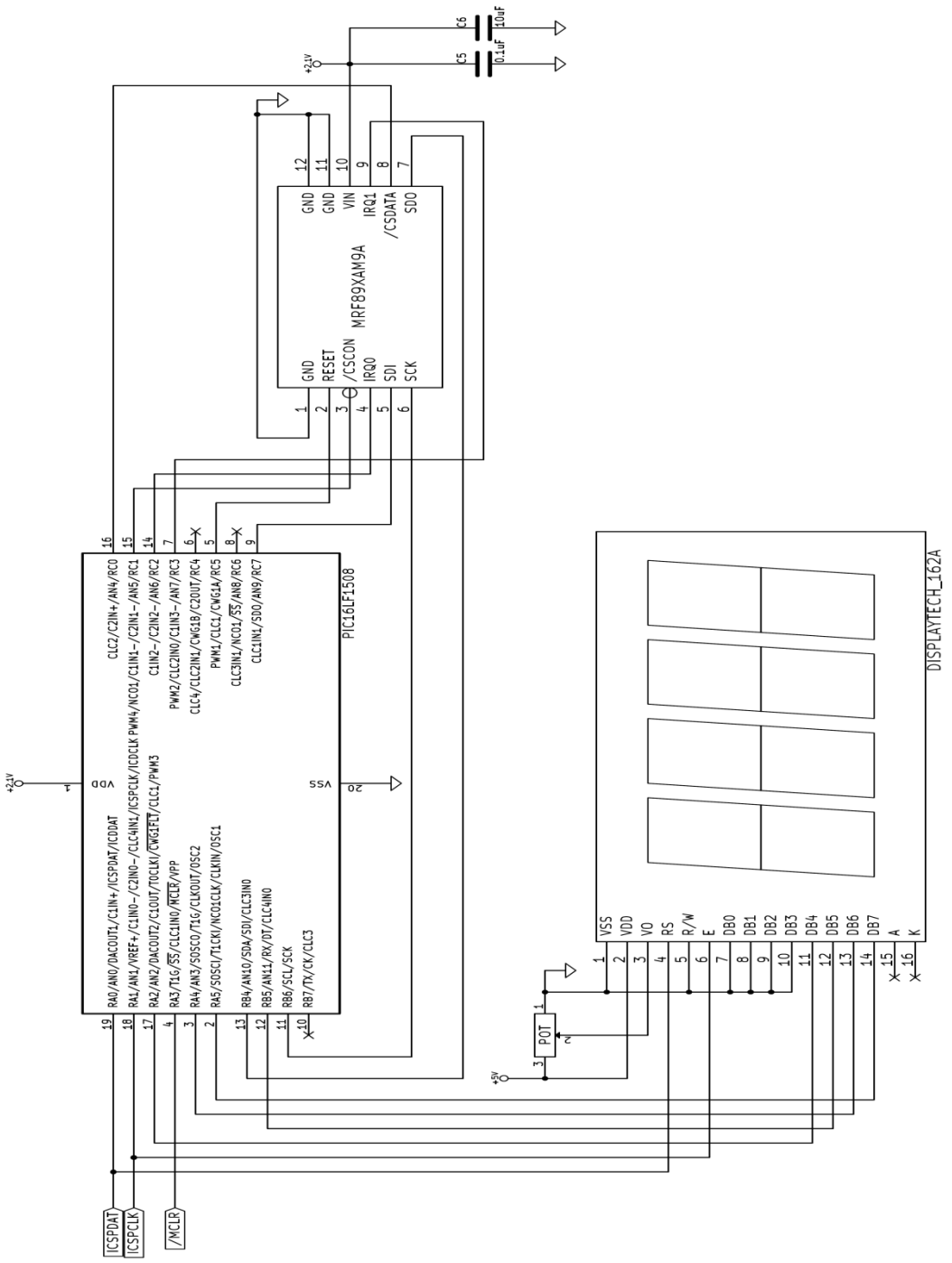

Figura 2.29: schema elettrico ricevitore

Nel ricevitore, il modulo wireless è programmato per rimanere continuamente nella modalità ricezione. Quando il modulo genera l'interrupt FIFO_THRESHOLD significa che il dato inviato dal trasmettitore, è stato ricevuto e salvato nel buffer. Il dato viene poi prelevato dal microcontrollore, che dopo averlo convertito nel formato richiesto dallo schermo LCD, lo invia a quest'ultimo, attraverso un opportuno interfacciamento. In questo modo lo schermo visualizzerà il dato di temperatura in formato decimale.

3 Misure

In questo capitolo si andranno a svolgere delle misure per trovare la modalità di funzionamento ottimale per avere il minor consumo possibile e compiere l'obbiettivo proposto dalla tesi.

Per le misure oltre ai componenti descritti precedentemente si sono utilizzati un multimetro Agilent 34401A ed un alimentatore Agilent E3631A, collegati come mostrato in Figura 3.1.

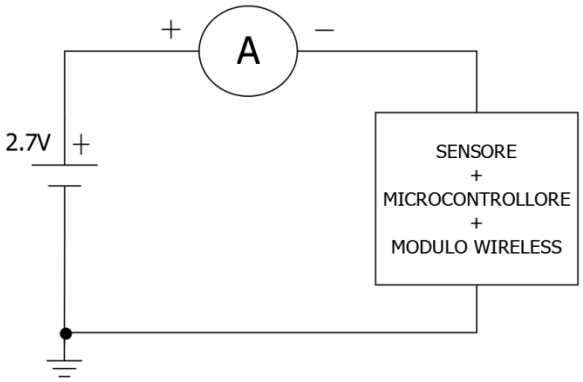

Figura 3.1: collegamento per le misure

3.1 Scelta della modulazione e della potenza di trasmissione

L'obbiettivo, consiste nel misurare il consumo di corrente durante la trasmissione, in funzione della modulazione (FSK o OOK) e della potenza di trasmissione (+13, +10, +7, +4, +1, -2, -5, -8 dBm) per scegliere poi la giusta combinazione per avere il miglior trade-off fra basso consumo ed efficienza.

Inizialmente è stato implementato il circuito come descritto nella Figura 3.1. Come si può vedere, in questa fase non è stato utilizzato il sistema di Energy Harvesting per alimentare il nodo wireless, dato che ancora non si sa se fornirà abbastanza energia per la trasmissione, ma è stato utilizzato un alimentatore impostato per dare una tensione d'uscita pari a 2.7V. Per queste misure, dato che si ha a disposizione un multimetro in grado di misurare la corrente media assorbita dal circuito, il nodo wireless è stato programmato in modo che invii continuamente una serie infinita di pacchetti di dati, formati da 2 byte noti, corrispondenti alla temperatura, più la parola di preambolo e sincronizzazione, in questo modo il

multimetro potrà acquisire molti più campioni, e quindi rendere la misura molto più precisa. In questa fase, il sensore è tenuto in modalità shutdown, mentre il microcontrollore usa l'oscillatore interno impostato a 4MHz.

Nella prima fase è stata misurata la corrente media assorbita dal sistema, impostando come potenza di trasmissione del modulo wireless +13dBm, ed utilizzando prima la modulazione FSK e poi la OOK.

Modulazione	**I_{DD} [mA]**
FSK	21,6
OOK	15,35

Tabella 3.1: consumo di corrente in funzione della modulazione

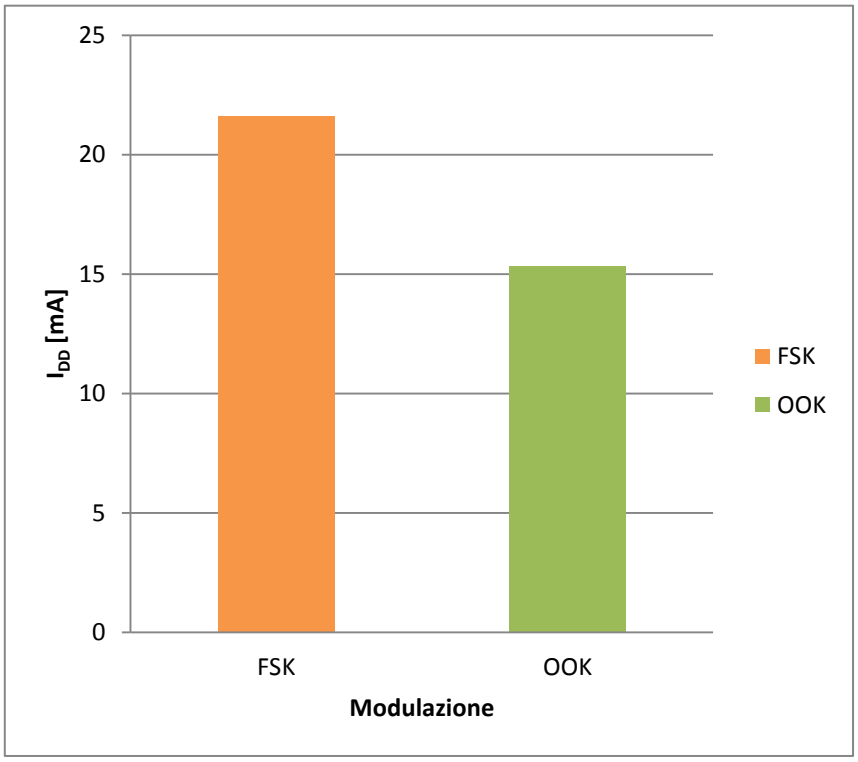

Figura 3.2: confronto dei consumi in funzione della modulazione

Come si può vedere dalla Figura 3.2 utilizzare la modulazione OOK rispetto alla FSK permette un rilevante risparmio energetico. Questo risultato non ci sorprende, anzi, era previsto in quanto già predetto dal Data Sheet del modulo wireless.

Nella seconda fase, seguendo le stesse modalità, si è andati a misurare la corrente media assorbita dal sistema in funzione della potenza di trasmissione impostabile dal modulo wireless, utilizzando la modulazione OOK.

P_{OUT} [dBm]	I_{DD} [mA]
+13	15,35
+10	15,30
+7	15,20
+4	14,68
+1	13,55
-2	13,00
-5	12,35
-8	12,03

Tabella 3.2: consumo di corrente in funzione della potenza di trasmissione

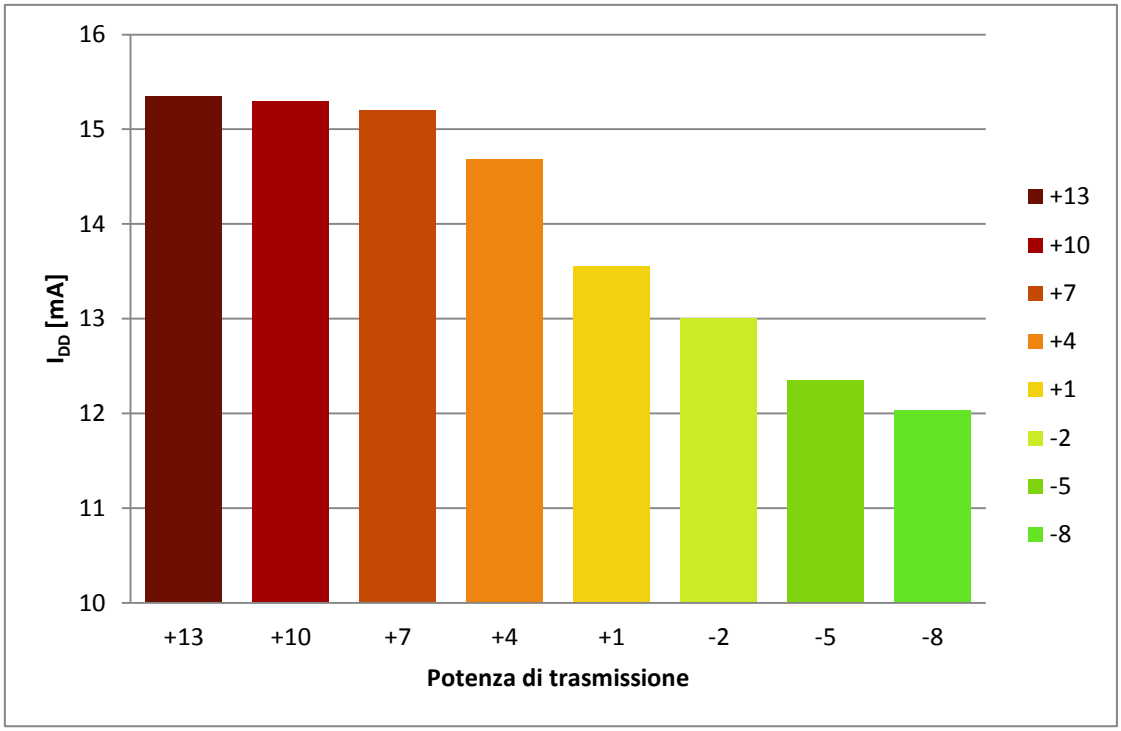

Figura 3.3: confronto dei consumi in funzione della potenza di trasmissione

Anche in questo caso i risultati sono prevedibili, infatti aumentando la potenza di trasmissione aumenta il consumo di corrente.

A seguito di questa misura si è deciso di utilizzare d'ora in avanti la modulazione OOK ed una potenza di trasmissione pari a +1dBm. Non si è scelto di usare la potenza minima, anche se più conveniente a livello energetico, per permettere una maggior distanza fra trasmettitore e ricevitore.

3.2 Funzionamento durante il periodo d'inattività

Come è stato già accennato, dato che l'alimentazione può fornire solo piccole quantità di energia, non si può pensare di tenere i componenti in modalità attiva per tutto il tempo inviando continuamente la temperatura, ma potranno essere attivati solo quando l'energia immagazzinata nella capacità sarà sufficiente per permettere al nodo wireless di adempiere correttamente al proprio compito.

L'obbiettivo di questa misurazione sarà decidere la modalità di funzionamento del trasmettitore durante il periodo di inattività, cioè se durante la carica della capacità, è più conveniente scollegare dall'alimentazione i tre componenti (sensore, microcontrollore, modulo wireless) o lasciarli collegati impostandoli nella modalità sleep e shutdown, cioè quella a più basso consumo possibile.

Dato che per la misura si ha a disposizione un multimetro, il quale permette di misurare la corrente media assorbita dal circuito, è stato deciso di dividere l'intera fase di acquisizione e invio del dato di temperatura in sotto-fasi, misurando la corrente media assorbita dal sistema mentre la sotto-fase viene ripetuta in un loop infinito, in modo da diminuire il range di variazione della corrente e aumentare così la precisione della misura.

Per questa misura, l'oscillatore interno del microcontrollore è stato impostato alla frequenza di 31KHz, dato che, se utilizzato a frequenze più alte avrebbe un consumo di corrente troppo elevato durante la modalità sleep e quindi renderebbe molto difficoltosa se non impossibile la carica del condensatore.

Di seguito verrà data una descrizione della modalità di funzionamento nel caso dello scollegamento dell'alimentazione durante il periodo di inattività del nodo wireless:

1. Risveglio:
   tutti i componenti necessitano di un certo tempo dopo il collegamento della tensione di alimentazione per stabilizzarsi e funzionare così correttamente: il microcontrollore necessita di 5,44ms, il modulo wireless di 10ms mentre il sensore di temperatura, si è verificato sperimentalmente che, prima di essere utilizzato necessita di un tempo di startup di 120ms;

2. Temperatura e configurazione:

   durante tutta questa fase il modulo wireless è mantenuto in modalità stand-by. Il microcontrollore invia il comando one-shot al sensore, per farli iniziare la conversione della temperatura in formato digitale, che dura 60ms, tempo durante il quale vengono impostati i registri del modulo wireless per avere la modalità di funzionamento desiderata. Infine, come ultima operazione, viene letta dal sensore la temperatura in formato digitale, mandandolo contemporaneamente nella modalità shutdown, attraverso il comando corrispondente;

3. Invio:

   mentre il modulo wireless è in modalità stand-by gli vengono caricati in memoria due byte noti, corrispondenti alla temperatura. Successivamente il modulo wireless viene, prima impostato nella modalità Frequency Synthesizer e successivamente, dopo un'attesa di 1 ms per la sua attivazione, nella modalità trasmissione e si aspetta fino alla generazione dell'interrupt TXDONE, segnale che indica la fine della trasmissione. A questo punto, dopo aver aspettato otto cicli di clock, vengono reimpostati i flag ed il FIFO del modulo wireless. Durante tutta questa fase, il sensore è mantenuto nella modalità shutdown.

4. Inattività:

   dopo l'invio viene scollegata l'alimentazione, che rimarrà tale durante tutto il periodo di inattività, dove il consumo quindi sarà nullo.

Fase	**I_{DD} [mA]**	**V_{DD} [V]**	**t [ms]**	**Energia [µJ]**
Risveglio	0,13	2,7	120	42,12
Temperatura e configurazione	0,13	2,7	69,20	24,29
Invio	4,20	2,7	27	306,18
Inattività	0	2,7	60000	0

Tabella 3.3: caratteristiche di ogni fase nella modalità in cui si scollega l'alimentazione durante il periodo d'inattività

Nota: per il risveglio non è stato possibile creare un loop infinito per la misurazione della corrente media assorbita dal circuito durante questa fase, quindi è stato supposto che, il consumo sia lo stesso della fase successiva in cui viene acquisita la temperatura e configurato il modulo wireless. La correttezza di questa ipotesi è

stata poi verificata sperimentalmente osservando la scarica della capacità di storage, la quale risulta avere lo stesso andamento in entrambe le fasi.

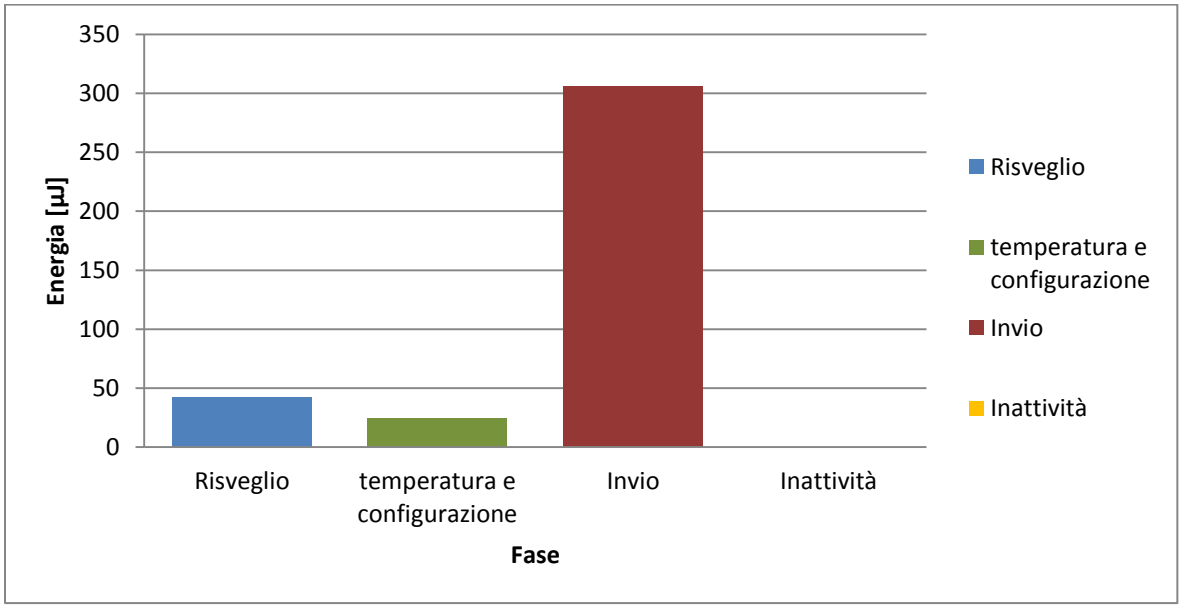

Figura 3.4: consumo energetico durante le fasi nella modalità in cui prevede lo scollegamento dell'alimentazione durante il periodo d'inattività

Nel caso in cui, invece, si volessero mandare i componenti nella modalità low-power durante il periodo di inattività, la modalità di funzionamento sarà la seguente:

1) Risveglio:

   attraverso un'interrupt esterno il microcontrollore si "sveglia" dalla modalità sleep. Anche in questo caso si deve attendere un certo tempo prima del corretto funzionamento dei dispositivi.

2) Acquisizione della temperatura:

   questa fase è molto simile a quella della modalità precedente, cioè Il microcontrollore invia il comando one-shot al sensore, per fargli iniziare la conversione della temperatura in formato digitale, che dura 60ms, dopo la quale si va a leggere il dato ottenuto, mentre contemporaneamente il sensore viene impostato nella modalità power down, attraverso il comando corrispondente. L'unica differenza al caso precedente è che la configurazione dei registri del modulo wireless si effettua solo la prima volta dopo l'accensione, dato che i valori impostati nei registri vengono tenuti in

memoria durante la modalità sleep. Durante questa fase il modulo è impostato nella modalità stand-by.

3) Invio:

   Questa fase è identica alla fase d'invio descritta nella modalità precedente.

4) Modalità sleep:

   Finita la trasmissione, si configurano i componenti in modalità low power: il modulo wireless ed il microcontrollore nella modalità sleep e il sensore nella modalità shutdown.

5) Inattività:

   Questa fase coincide con il periodo di inattività, dove i componenti sono impostati, dalla fase precedente, nella modalità a basso consumo.

Fase	**I_{DD} [mA]**	**V_{DD} [V]**	**t [ms]**	**Energia [µJ]**
Risveglio	0,12	2,7	1,47	0,79
Temperatura e configurazione	0,12	2,7	69,60	22,55
Invio	4,20	2,7	27,00	306,18
Modalità sleep	0,088	2,7	4,26	1,01
Inattività	0,0012	2,7	60000	194,40

Tabella 3.4: caratteristiche di ogni fase nella modalità in cui si mandano i componenti in modalità sleep durante il periodo d'inattività

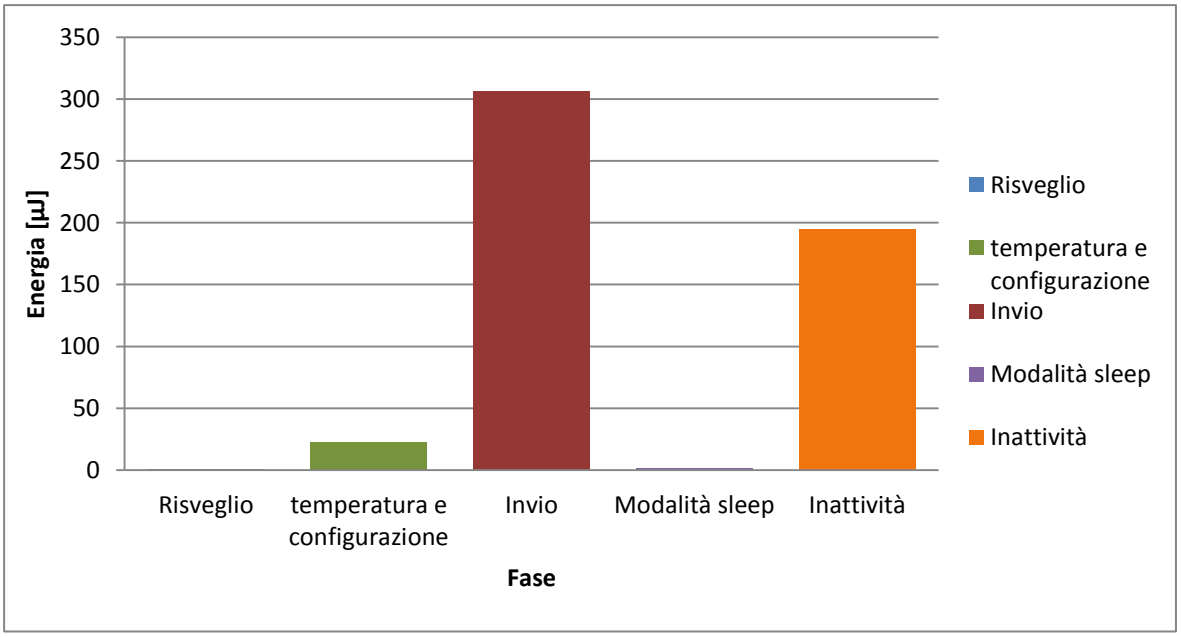

Figura 3.5: Consumo energetico durante le fasi nella modalità in cui prevede di mandare i componenti nella modalità sleep durante il periodo d'inattività

Nota: per la realizzazione del grafico in Figura 3.5 è stato utilizzato come tempo d'inattività 60 secondi. Questo dato è stato preso in modo arbitrario con il solo scopo di visualizzare meglio l'effetto che ha la modalità sleep sui consumi durante il periodo d'inattività.

Queste due modalità appena descritte, presentano vantaggi e svantaggi l'una rispetto all'altra. Infatti, nel caso in cui andassimo a scollegare la tensione di alimentazione, durante il periodo di inattività il consumo sarebbe nullo, mentre nell'altro caso, ci sarebbe il consumo dovuto dal configurare e mantenere i componenti nella modalità low-power. Quest'ultimo consumo è molto piccolo ma andrà comunque ad influire sulla velocità della carica del condensatore e potrebbe diventare molto rilevante con l'aumentare della durata del periodo d'inattività. Tuttavia mandare i componenti nella modalità sleep permette di non dover andare a impostare tutte le volte i registri del modulo wireless e di risvegliarsi molto più velocemente, quindi dopo il periodo di inattività si dovrà aspettare molto meno tempo per il corretto funzionamento dei componenti.

Per scoprire qual è la modalità migliore, si è andati a calcolare l'energia assorbita dal sistema in funzione del periodo di inattività.

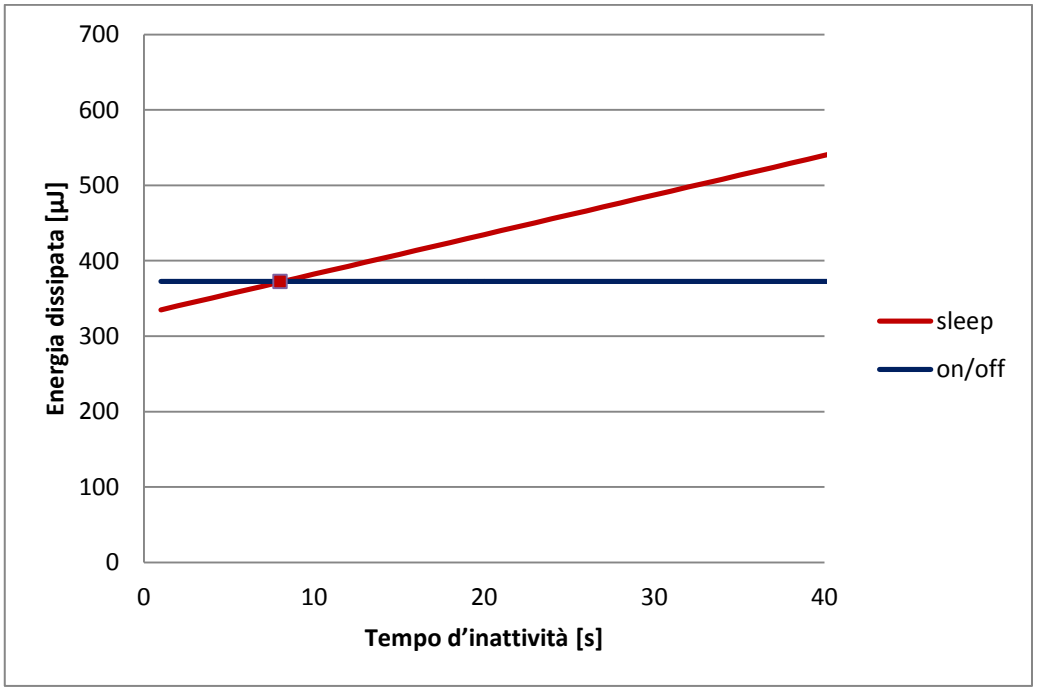

Figura 3.6: confronto del consumo energetico delle due modalità in funzione del tempo d'inattività

La Figura 3.6 riassume il risultato della misura e mostra che non è possibile determinare a priori la configurazione migliore. Questa infatti dipende dall'energia disponibile in ingresso; se la sorgente è in grado di fornire molta energia, tale da permettere l'invio della temperatura ogni 8 secondi o meno, conviene, durante il periodo di inattività, mandare i componenti in modalità low power, mentre se la sorgente fornisce poca energia permettendo così l'invio meno frequentemente, cioè se passano più di 8 secondi fra un invio e l'altro, allora conviene scollegare l'alimentazione.

In pratica più è lungo il periodo di inattività e più il consumo dei componenti nella modalità sleep diventa rilevante, rendendo questa modalità sconveniente, mentre, al contrario, più il periodo di inattività è corto e più il consumo durante questo periodo diventa irrilevante rispetto al consumo dato dai componenti nella fase iniziale per stabilizzarsi dopo il collegamento dell'alimentazione, rendendo la modalità sleep più conveniente.

Viceversa, lo scollegamento della tensione d'alimentazione diventa sempre più conveniente con l'aumentare del periodo di inattività, rendendo così il consumo dei componenti nella fase iniziale irrilevante rispetto al consumo dovuto alla modalità sleep.

Per il progetto si è scelto di scollegare l'alimentazione durante il periodo di inattività, mettendosi così nel caso peggiore, cioè nel caso in cui la sorgente possa fornire piccole quantità d'energia, restando così il più generale possibile. Inoltre, come già detto, leggere la temperatura molto frequentemente (ogni 8 secondi o meno) darebbe informazioni ridondanti e quindi sarebbe uno spreco inutile d'energia.

3.3 Scelta della frequenza

Dato che si è scelto di scollegare l'alimentazione durante il periodo d'inattività, ora si può misurare il consumo energetico, durante l'intero funzionamento del nodo wireless, in funzione della frequenza dell'oscillatore interno al microcontrollore, usata sia per la temporizzazione di quest'ultimo sia per la comunicazione SPI[1]. Infatti, più la frequenza è bassa e più lo è anche il consumo di corrente, ma contemporaneamente la durata delle comunicazioni si allungano, quindi bisogna valutare la frequenza migliore per avere un giusto trade-off fra consumo di corrente e tempo impiegato, per avere così il più basso consumo energetico possibile.

Per eseguire le misure si è seguita la stessa procedura utilizzata nel caso precedente, andando a variare solo la frequenza dell'oscillatore interno.

La frequenza massima utilizzabile nella comunicazione SPI è dettata dal modulo wireless, che da il vincolo più stringente, infatti per la configurazione dei registri la frequenza del clock non può essere superiore ad 1 MHz.

Quindi, visto che la frequenza del clock dell'SPI è quella data dall'oscillatore interno divisa per 4, la massima frequenza imponibile all'oscillatore non dovrà superare i 4 MHz.

Fase	**I_{DD} [mA]**	**V_{DD} [V]**	**t [ms]**	**W [µJ]**
Risveglio	0,13	2,7	120	42,12
Temperatura e configurazione	0,13	2,7	69,20	24,29
Invio	4,20	2,7	27	306,18
W_{TOT} [J]				**372,59**

Tabella 3.5: oscillatore interno a 31 KHz

[1] Si ricorda che, usando l'altro funzionamento, in cui si settano i componenti nella modalità sleep durante il periodo di inattività, è possibile utilizzare solo la frequenza di 31KHz altrimenti il consumo del microcontrollore durante la modalità sleep è troppo elevato per permettere la carica del condensatore.

Fase	I_{DD} [mA]	V_{DD} [V]	t [ms]	W [µJ]
Risveglio	0,35	2,7	120	113,40
Temperatura e configurazione	0,35	2,7	60	56,70
Invio	5,70	2,7	10	153,90
W_{TOT} [µJ]				**324,00**

Tabella 3.6: oscillatore interno a 125 KHz

Fase	I_{DD} [mA]	V_{DD} [V]	t [ms]	W [J]
Risveglio	0,36	2,7	120	116,64
Temperatura e configurazione	0,36	2,7	60	58,32
Invio	6,88	2,7	6,70	124,46
W_{TOT} [µJ]				**299,42**

Tabella 3.7: oscillatore interno a 250 KHz

Fase	I_{DD} [mA]	V_{DD} [V]	t [ms]	W [µJ]
Risveglio	0,37	2,7	120	119,88
Temperatura e configurazione	0,37	2,7	60	59,94
Invio	5,28	2,7	7,65	109,06
W_{TOT} [µJ]				**288,88**

Tabella 3.8: oscillatore interno a 500 KHz

Fase	I_{DD} [mA]	V_{DD} [V]	t [ms]	W [µJ]
Risveglio	0,42	2,7	120	134,46
Temperatura e configurazione	0,42	2,7	60	67,23
Invio	4,48	2,7	8,70	105,24
W_{TOT} [µJ]				**306,93**

Tabella 3.9: oscillatore interno a 1 MHz

Fase	I_{DD} [mA]	V_{DD} [V]	t [ms]	W [J]
Risveglio	0,44	2,7	120	142,56
Temperatura e configurazione	0,44	2,7	60	71,28
Invio	4,08	2,7	9,05	99,69
W_{TOT} [µJ]				**313,53**

Tabella 3.10: oscillatore interno a 2 MHz

Fase	I_{DD} [A]	V_{DD} [V]	t [ms]	W [J]
Risveglio	0,56	2,7	120	181,44
Temperatura e configurazione	0,56	2,7	60	90,72
Invio	9,56	2,7	3,82	98,60
W_{TOT} [µJ]				**370,76**

Tabella 3.11: oscillatore interno a 4 MHz

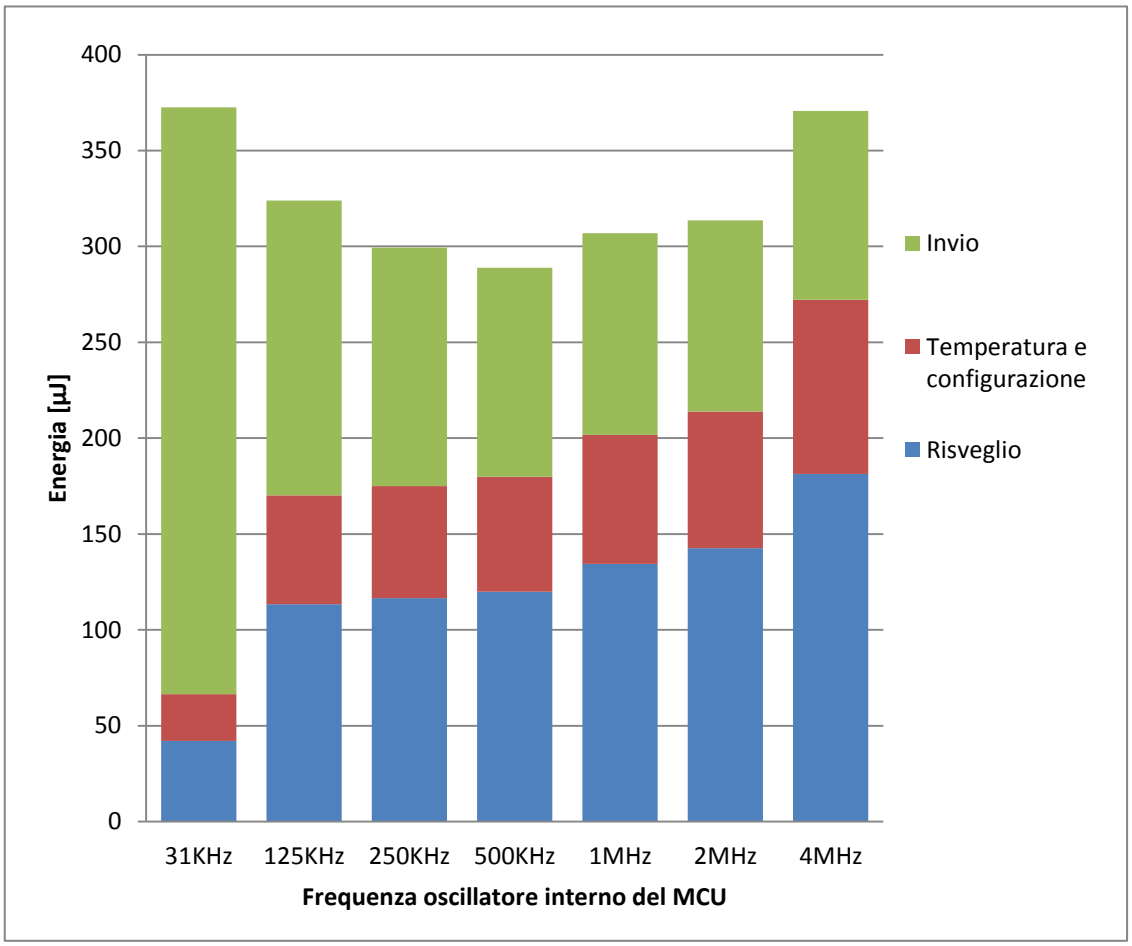

Figura 3.7: consumo energetico in funzione della frequenza dell'oscillatore interno del microcontrollore

Come ci si aspettava, più aumenta la frequenza e più aumenta il consumo di corrente e diminuisce la durata delle comunicazioni, anche se comunque ci saranno delle tempistiche da rispettare, come i 120ms iniziali ed i 60ms per la conversione del dato di temperatura.

Il giusto trade-off, come illustrato in Figura 3.7, si verifica alla frequenza di 500kHz, alla quale si avrà il minor consumo energetico possibile, e quindi questa sarà la frequenza utilizzata da qui in avanti.

3.4 Progettazione del voltage monitor

Ora che si è trovato il funzionamento ottimale per i tre componenti: sensore, microcontrollore e modulo wireless, di seguito ci si occuperà dello studio del sistema di Energy Harvesting vero e proprio, andando ad implementare nel circuito il trasduttore, la capacità per lo stoccaggio dell'energia ed il voltage monitor.

Come abbiamo già detto, durante il periodo di inattività l'alimentazione dovrà essere scollegata e ricollegata quando la capacità avrà immagazzinato abbastanza energia per il corretto invio del dato di temperatura, questo compito sarà svolto dal Voltage monitor illustrato nella Figura 3.8.

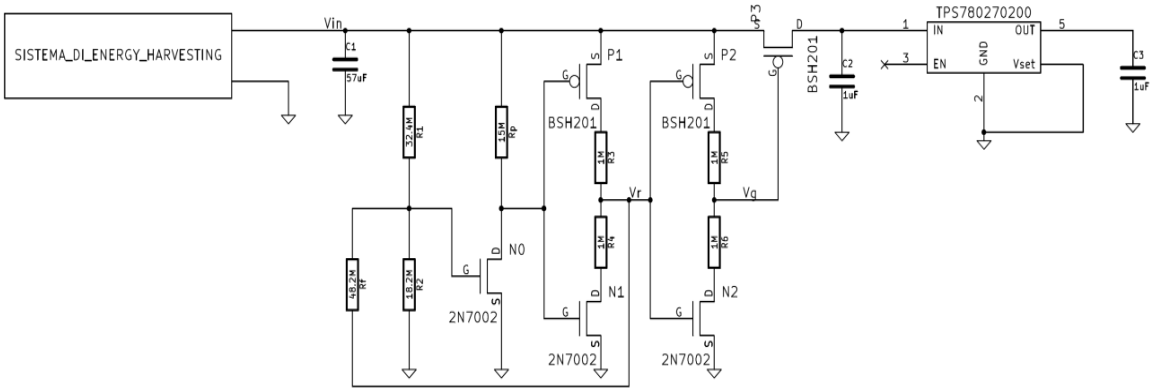

Figura 3.8: voltage monitor

Il voltage monitor è stato progettato ed implementato in modo che la capacità di stoccaggio venga collegata al regolatore di tensione, e quindi all'intero circuito, al raggiungimento di 5V e scollegata a 3V. Questi limiti di tensione, superiore ed inferiore sono stati scelti a seguito delle caratteristiche elettriche del regolatore, cioè all'interno di questo range si ha la sicurezza che il regolatore riesca a fornire in uscita una tensione stabile di 2,7V.

Di seguito verrà data una breve descrizione del funzionamento:
Inizialmente si ha la capacità scarica e quindi la tensione V_{in}=0V. Quando il sensore piezoelettrico inizia a fornire energia la capacità di storage si carica, facendo crescere così la tensione ai suoi capi. Quindi nella fase iniziale, l'n-mos N0 è spento, ed il secondo invertitore ha in ingresso un livello logico alto grazie alla resistenza Rp, dando in uscita un livello logico basso, cioè V_r=0V. In questa

fase quindi la resistenza Rf è collegata in parallelo alla resistenza R2, di conseguenza all'uscita del terzo invertitore si ha un livello logico alto che tiene il p-mos P3 spento e quindi non permette il passaggio della corrente dal source al drain, tenendo la capacità scollegata dal regolatore.
Quando la tensione ai capi della capacità raggiungerà i 5V, le resistenze R1, R2 ed Rf devono essere progettate in modo tale che il partitore fra R1 ed il parallelo fra R2 ed Rf diano una tensione nel gate dell'n-mos N0 pari ad 1V, cioè la tensione alla quale si accende, in questo modo i due invertitori commutano dando in ingresso al p-mos P3 un livello logico basso, il quale, così facendo si accende facendo scorrere corrente dal source al drain, e collegando la capacità al regolatore.
Quando la capacità viene collegata al regolatore, questa inizierà a scaricarsi, in quanto il trasmettitore consumerà energia per l'acquisizione e l'invio del dato di temperatura.
durante questa fase la resistenza Rf non sarà più in parallelo ad R2 in quanto ora Vr=Vin, ma si troverà in parallelo a R1. In questo modo si può creare un'isteresi e progettare il partitore fra R2 ed il parallelo fra R1 ed R2 in modo che nel gate dell'n-mos N0 ci sia una tensione superiore ad 1V fino a quando Vin rimane sopra i 3V. Quando quest'ultima condizione non è più soddisfatta, l'n-mos N0 si spegne, e la capacità viene scollegata dal regolatore, e comincia a caricarsi facendo ripartire il ciclo di isteresi

Di seguito verranno illustrati i calcoli eseguiti per il progetto delle resistenze per avere l'isteresi voluta.

V_r="0"

$$V_{nmos} = \frac{R_2//R_f}{R_1 + R_2//R_f} \cdot V_{in} \xrightarrow[V_{nmos}=1V]{} V_{in} = \frac{R_1 + R_2//R_f}{R_2//R_f} \cdot 1 = 5 \xrightarrow[a=R_2//R_f]{} \frac{R_1 + a}{a} = 5$$

$$R_1 = 4a$$

V_r="1" (V_r=V_{in})

$$V_{nmos} = \frac{R_2}{R_2 + R_1//R_f} \cdot V_{in} \xrightarrow[V_{nmos}=1V]{} V_{in} = \frac{R_2 + R_1//R_f}{R_2} \cdot 1 = 3 \xrightarrow[b=R_1//R_f]{} \frac{R_2 + b}{R_2} = 3$$

$$R_2 = b/2$$

$$\begin{cases} R_1 = 4a \\ R_2 = b/2 \end{cases} \begin{cases} \dfrac{4}{R_1} = \dfrac{1}{a} = \dfrac{1}{R_2} + \dfrac{1}{R_f} \\ \dfrac{1}{2 \cdot R_2} = \dfrac{1}{b} = \dfrac{1}{R_1} + \dfrac{1}{R_f} \end{cases} \begin{cases} \dfrac{1}{R_f} = \dfrac{4}{R_1} - \dfrac{1}{R_2} = \dfrac{4 \cdot R_2 - R_1}{R_1 \cdot R_2} \\ \dfrac{1}{2 \cdot R_2} = \dfrac{1}{R_1} + \dfrac{4 \cdot R_2 - R_1}{R_1 \cdot R_2} \end{cases}$$

$$\begin{cases} -\dfrac{1}{R_f} = \dfrac{4 \cdot R_2 - R_1}{R_1 \cdot R_2} \\ \dfrac{-R_1 + 2 \cdot R_2 + 8 \cdot R_2 - 2 \cdot R_1}{2R_1 \cdot R_2} = 0 \end{cases} \begin{cases} -\dfrac{1}{R_f} = \dfrac{4 \cdot R_2 - R_1}{R_1 \cdot R_2} \\ 10 \cdot R_2 = 3 \cdot R_1 \end{cases}$$

$$\begin{cases} R_f = \dfrac{R_1 \cdot R_2}{4 \cdot R_2 - R_1} = \dfrac{R_1 \cdot \frac{3}{10} \cdot R_1}{4 \cdot \frac{3}{10} \cdot R_1 - R_1} \\ R_2 = \dfrac{3}{10} \cdot R_1 \end{cases} \begin{cases} R_f = \dfrac{3}{10} \cdot R_1^2 \cdot \dfrac{10}{12 \cdot R_1 - 10R_1} = \dfrac{3}{2} \cdot R_1 \\ R_2 = \dfrac{3}{10} \cdot R_1 \end{cases}$$

$$\begin{cases} R_f = \frac{3}{2} \cdot R_1 \\ R_2 = \frac{3}{10} \cdot R_1 \end{cases} \quad \text{Imponendo } R_2\text{=10MΩ:} \quad \begin{cases} R_1 = 33.33\ M\Omega \\ R_f = 50\ M\Omega \end{cases}$$

Infine per la resistenza R_p si è scelto il valore di 15MΩ. Quest'ultima resistenza non influisce sull'isteresi, perciò non si hanno vincoli, quindi può essere preso un valore molto alto di resistenza per ridurre la potenza dissipata, senza tuttavia esagerare, dato che rappresenta il carico dell'invertitore a rapporto, perciò il valore di resistenza non deve essere più grande o equiparabile alla resistenza equivalente che mostra l'n-mos N1 quando è spento, altrimenti l'uscita dell'invertitore non sarebbe prevedibile, dato che si creerebbe un conflitto, che porterebbe l'uscita ad un valore intermedio fra V_{in} e massa.

Le formule utilizzate sono semplificate e non tengono conto delle correnti di leakage dei transistor, ma comunque sono utili per dare un'idea di massima. Perciò dopo aver svolto i calcoli è stato necessario andare ad aggiustare sperimentalmente i valori delle resistenze per avere il giusto ciclo di isteresi, ottenendo i seguenti valori:

R_1=32,4 MΩ R_2=18,9 MΩ R_f=48,2 MΩ R_p=15 MΩ

3.5 Progettazione della capacità di stoccaggio

L'obbiettivo di questa misura è calcolare il giusto valore di capacità per il condensatore di stoccaggio, in modo che riesca a contenere l'energia necessaria per permettere il corretto funzionamento del nodo wireless e verificarne poi il risultato sperimentalmente.

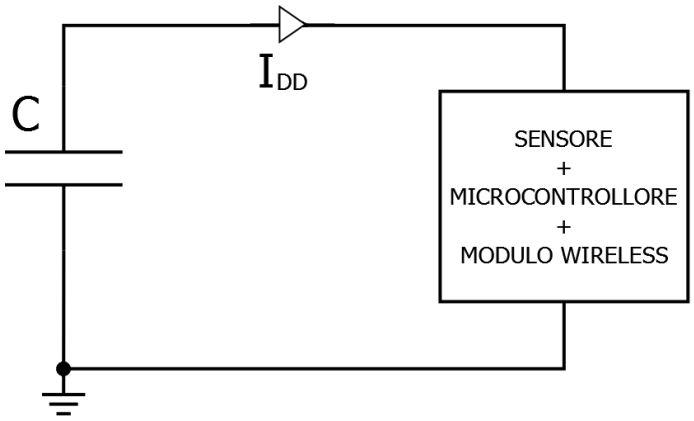

Figura 3.9: schema semplificato del collegamento fra capacità di storage e applicazione finale

$$I_{DD} = C \cdot \frac{dV(t)}{dt} \rightarrow C = I_{DD} \cdot \frac{dt}{dV(t)}$$

Andiamo a ricordiamo il consumo e le tempistiche delle varie fasi che compongono il funzionamento del nodo wireless:

Fase	**I_{DD} [mA]**	**t [ms]**
Risveglio	0,37	120
Temperatura e configurazione	0,37	61,20
Invio	5,28	7,65

Ora che si conoscono il consumo di corrente, le tempistiche di ogni fase ed il range di tensione disponibile, impostato a 2V dal voltage monitor, è possibile calcolare il valore di capacità necessario:

$$C = I_{DD} \cdot \frac{\Delta t}{\Delta V} = \frac{I_1.\Delta t_1 + I_2.\Delta t_2 + I_3.\Delta t_3}{\Delta t_{TOT}} \cdot \frac{\Delta t}{\Delta V} =$$

$$= \frac{370\mu \cdot 120m + 60m \cdot 370\mu + 5.28m \cdot 7.65m}{2} = 53.50\mu F$$

Nell'implementazione si è preferito sovrastimare il valore ottenuto, in quanto nel calcolo non viene tenuto conto dell'energia dissipata dal voltage monitor. Per questo si è utilizzato due condensatori, uno da 47uF e l'altro da 10uF, collegati in parallelo ottenendo così una capacità complessiva di 57uF. Infine, dopo aver implementato l'intero sistema, se ne è verificato il corretto funzionamento.

3.6 Energia per bit

In questa ultima fase, si è misurato il consumo energetico associato ad un singolo bit del payload, durante l'invio del pacchetto, in funzione delle sue dimensioni.

Per la misura si è utilizzato come alimentazione il sistema di Energy Harvesting e si è impostato il microcontrollore ed il modulo wireless con le caratteristiche decise fino a qui per avere il minor consumo possibile, mentre il sensore è tenuto nella modalità shutdown.

Per misurare la corrente assorbita dall'intero circuito, è stato collegato l'oscilloscopio ai capi della capacità di storage, misurandone così l'escursione di tensione dovuta alla sua scarica ed il tempo impiegato durante l'invio del dato. Infine grazie ai dati raccolti si è calcolato il consumo energetico dovuto alla trasmissione dell'intero pacchetto e quello necessario per l'invio di un singolo bit.

Dalla misura effettuata si sono ottenuti i seguenti dati:

V_{DD}=2.7V tensione con cui viene alimentato il sistema trasmittente

bit=48 bit trasmessi durante la trasmissione: 8bit di preambolo, 32bit di sincronizzazione e 8bit corrispondenti al payload

ΔV=800mV escursione di tensione ai capi del condensatore dovuta all'l'invio del pachetto

Δt=2.84ms tempo impiegato dal trasmettitore per l'invio del dato

Dai dati ottenuti ci si è calcolati la corrente assorbita dal trasmettitore durante l'invio del pacchetto:

$$\begin{matrix} \Delta V = 800\ mV \\ \Delta t = 2.84\ ms \end{matrix} \Longrightarrow I_{DD} = C\frac{\Delta V}{\Delta t} = 57 \times 10^{-6} \cdot \frac{800 \times 10^{-3}}{2.84 \times 10^{-3}} = 16.06\ mA$$

$$E_{TOT} = V_{DD} \cdot I_{DD} \cdot \Delta t = 2.7 \cdot 16.06 \times 10^{-3} \cdot 2.84 \times 10^{-3} = 123.15\ \mu J$$

$$E_{bit} = \frac{E_{TOT}}{bit} = \frac{123.15 \times 10^{-6}}{48} = 2.56\ \mu J$$

Quindi l'energia totale assorbita dal trasmettitore durante l'invio del pacchetto corrisponde a 123.15μJ. Dividendo questa quantità per i bit che compongono il pacchetto (48 bit) si ottiene 2.56μJ che corrisponde all'energia necessaria al trasmettitore per inviare un singolo bit.

In seguito, andando a variare le dimensioni del payload, si è andato a calcolare l'energia totale necessaria all'invio dell'intero pacchetto e successivamente a dividerla per i bit che compongono il payload, che corrispondono ai bit associati all'informazione. In questo modo si è trovato l'energia effettiva per bit, cioè l'energia necessaria al trasmettitore per inviare l'unità di informazione.

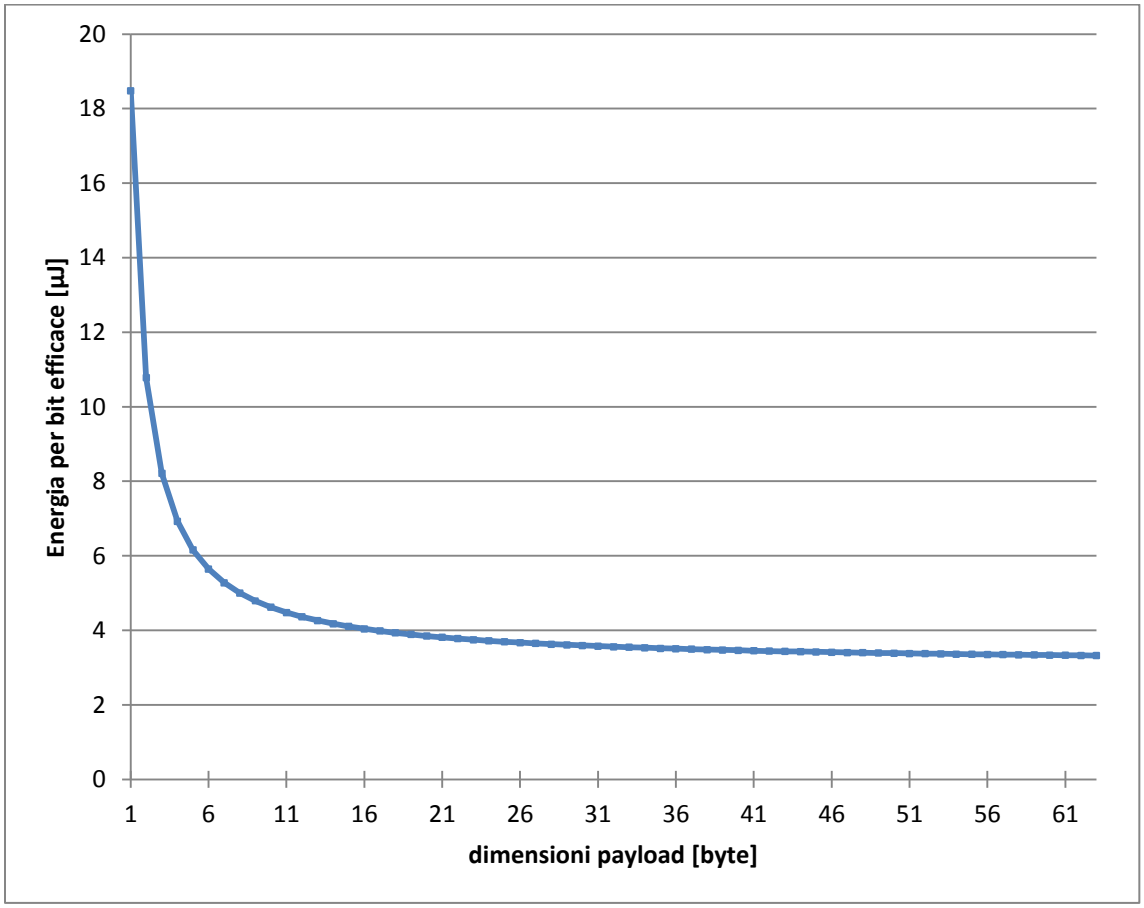

Figura 3.10: energia per bit efficace in funzione delle dimensioni del payload

Come si può vedere dalla Figura 3.10 l'energia associata all'invio di un bit di informazione, diminuisce all'aumentare della dimensione del payload. Questo

risultato è dovuto al fatto che insieme al payload inviamo ogni volta, altri bit dovuti alla parola di preambolo e di sincronizzazione (40 bit), che non trasportano informazione, ma a cui è associata un'energia fissa calcolata qui di seguito:

$$E_{ps} = E_{bit} \cdot bit = 2.56 \times 10^{-6} \cdot 40 = 102.4\mu J$$

Quindi ad ogni trasmissione, viene spesa un'energia pari a 102.4μJ, indipendentemente dal payload. Quest'ultima energia, nel calcolo dell'energia per bit effettiva, viene ripartita dal payload, quindi più aumenta la dimensione del payload e più aumenteranno i bit che si faranno carico di questa energia "indesiderata" facendo diminuire l'energia per bit.

Quest'ultima misura è stata svolta nell'ottica di applicazioni future, nel caso si volesse aggiungere altri sensori e quindi inviare altri dati oltre alla sola temperatura.

4 Implementazione finale

4.1 Codice del microcontrollore in trasmissione

Di seguito viene riportato il codice scritto, ed utilizzato dal trasmettitore.

```
#include <stdio.h>
#include <stdlib.h>

// PIC16LF1508 Configuration Bit Settings
#include <xc.h>
__CONFIG(FOSC_INTOSC & WDTE_OFF & PWRTE_OFF & MCLRE_OFF & CP_OFF &
BOREN_OFF & CLKOUTEN_OFF & IESO_OFF & FCMEN_OFF);
__CONFIG(WRT_OFF & STVREN_OFF & BORV_LO & LPBOR_OFF & LVP_OFF);

#define _XTAL_FREQ   500000                //FOSC=500KHz
#define CSDATA      LATCbits.LATC0
#define CSCON       LATCbits.LATC1
#define TXDONE      PORTCbits.RC3
#define RESETpin    PORTCbits.RC5
#define CS_ST       LATCbits.LATC4
#define PROVA       LATBbits.LATB7

void main(void){

   // inizializzazione del microcontrollore
   ANSELA=0;               //pin della porta A I/O digitali
   ANSELB=0;               //pin della porta B I/O digitali
   ANSELC=0;               //pin della porta C I/O digitali
   OSCCONbits.SCS=0b10;    //oscillatore interno genera il clock
   OSCCONbits.IRCF=0x7;    //FOSC=500KHz
   SSP1CON1bits.SSPEN=1;   //abilita la porta seriale
   SSP1CON1bits.CKP=0;     //clock inattivo nel livello basso
   SSP1CON1bits.SSPM=0;    //FCLOCK=FOSC/4
   SSP1STATbits.SMP=0;     //campionamento nel fronte di salita
   SSP1STATbits.CKE=1;     //trasmissione nel fronte di discesa
   TRISC=0x2C;      //RC7=RC6=RC4=RC1=RC=OutputRC5=RC3=RC2=Input
   TRISB=0x10;      //RB7=RB6=RB5=Output  RB4=Input
   TRISA=0x00;      //RA5=RA4=RA3=RA2=RA1=RA0=Output
   OPTION_REGbits.TMR0CS=0;    //Timer0 Clock = FOSC/4
```

```
OPTION_REGbits.PSA=0;       //Prescaler assegnato al Timer0
OPTION_REGbits.PS=0b100;    //Prescaler: 1:32

 CS_ST=1;
 CSDATA=1;
 CSCON=1;
 PROVA=1;

 //aspettare 60ms
 TMR0=21;
 INTCONbits.TMR0IF=0;
 while(!INTCONbits.TMR0IF);
 INTCONbits.TMR0IF=0;
 //aspettare 60ms
 TMR0=21;
 INTCONbits.TMR0IF=0;
 while(!INTCONbits.TMR0IF);
 INTCONbits.TMR0IF=0;

 //modalità one shot
 CS_ST=0;
 SSP1BUF=0x10;
 while(!BF);
 SSP1BUF=0x00;
 while(!BF);
 CS_ST=1;

 //configurare i registri del modulo wireless
 //1_GCCONREG
 CSCON=0;
 SSP1BUF=0x00;
 while(!BF);
 SSP1BUF=0x2A;
 while(!BF);
 //2_DMODREG
 SSP1BUF=0x02;
 while(!BF);
 SSP1BUF=0x4C;
 while(!BF);
```

```
//6_FIFOCREG
SSP1BUF=0x0A;
while(!BF);
SSP1BUF=0x02;
while(!BF);
//14_FTXRXIREG
SSP1BUF=0x1A;
while(!BF);
SSP1BUF=0xB9;
while(!BF);
//15_FTPRIREG
SSP1BUF=0x1C;
while(!BF);
SSP1BUF=0x10;
while(!BF);
//19_SYNCREG
SSP1BUF=0x24;
while(!BF);
SSP1BUF=0x38;
while(!BF);
//23_SYNCV31REG
SSP1BUF=0x2C;
while(!BF);
SSP1BUF='S';
while(!BF);
//24_SYNCV23REG
SSP1BUF=0x2E;
while(!BF);
SSP1BUF='Y';
while(!BF);
//25_SYNCV15REG
SSP1BUF=0x30;
while(!BF);
SSP1BUF='N';
while(!BF);
//26_SYNCV07REG
SSP1BUF=0x32;
```

```
while(!BF);
SSP1BUF='C';
while(!BF);
//27_TXCONREG
SSP1BUF=0x34;
while(!BF);
SSP1BUF=0x78;
while(!BF);
//28_CLKOUTREG
SSP1BUF=0x36;
while(!BF);
SSP1BUF=0x3C;
while(!BF);
//29_PLOADREG
SSP1BUF=0x38;
while(!BF);
SSP1BUF=0x02;
while(!BF);
//31_PKTCREG
SSP1BUF=0x3C;
while(!BF);
SSP1BUF=0x00;
while(!BF);
//32_FCRCREG
SSP1BUF=0x3E;
while(!BF);
SSP1BUF=0x80;
while(!BF);
CSCON=1;

//aspettare 60ms per la conversione della temperature
TMR0=21;
INTCONbits.TMR0IF=0;
while(!INTCONbits.TMR0IF);
INTCONbits.TMR0IF=0;
```

```
//sensore in modalità POWER-DOWN per leggere la temperatura
CS_ST=0;
SSP1BUF=0x20;
while(!BF);
char T1=SSP1BUF;    //primi 8 bit della temperatura
SSP1BUF=0x00;
while(!BF);
char T2=SSP1BUF;    //ultimi 8 bit della temperatura
CS_ST=1;

//scrittura del dato di temperatura nel buffer del modulo
CSDATA=0;
SSP1BUF=T1;
while(!BF);
CSDATA=1;
NOP();
CSDATA=0;
SSP1BUF=T2;
while(!BF);
CSDATA=1;

//impostare nel modulo la modalità Frequency Synthesizer
//1_GCCONREG
CSCON=0;
SSP1BUF=0x00;
while(!BF);
SSP1BUF=0x4A;
while(!BF);
CSCON=1;

//aspettare 1ms per il risveglio del Frequency Synthesizer
TMR0=252;
INTCONbits.TMR0IF=0;
while(!INTCONbits.TMR0IF);
INTCONbits.TMR0IF=0;
```

```
//trasmissione del dato di temperatura
//1_GCCONREG
CSCON=0;
  SSP1BUF=0x00;
  while(!BF);
  SSP1BUF=0x8A;
  while(!BF);
  CSCON=1;

  while(!TXDONE);

  while(1);

}
```

4.2 Implementazione fisica finale

In questa ultima fase si è implementato l'intero sistema, come mostrato in Figura 4.1 e se ne è verificato il funzionamento.

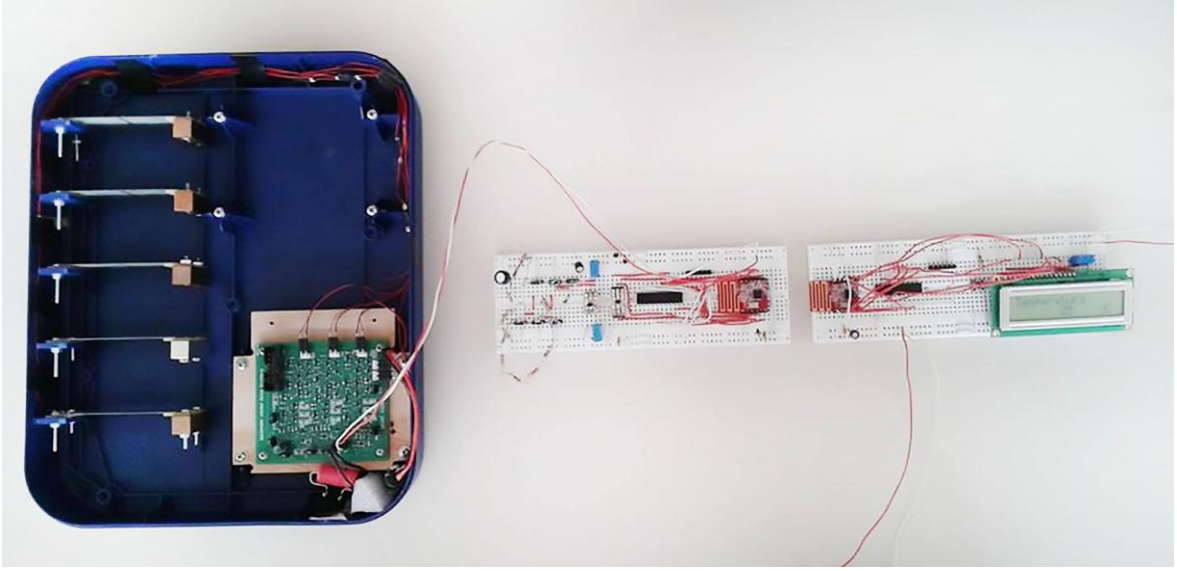

Figura 4.1: implementazione fisica

La prova del sistema ha dato esiti positivi, permettendo l'acquisizione della temperatura ed il suo invio ogni 15/20 secondi ed una distanza fra trasmettitore e ricevitore fino a 1m.

5 Conclusioni

In questo progetto si è voluto realizzare un nodo wireless, alimentato attraverso l'Energy Harvesting, in grado di misurare la temperatura ambiente ed inviarla ad un sistema ricevente, che la visualizzerà su uno schermo LCD.

Per la realizzazione del progetto si è partiti dalla ricerca dei componenti da utilizzare, che ha portato a scegliere:

- TMP125 prodotto dalla Texas Instruments, per il sensore di temperatura;
- PIC16LF1508 prodotto dalla Microchip, per il microcontrollore.
- MRF89XAM9A prodotto dalla Microchip, per il modulo wireless;

Una volta scelti i componenti si è proceduto all'implementazione fisica sia del trasmettitore che del ricevitore, verificandone il corretto funzionamento.

Successivamente si sono eseguite delle misure per trovare le giuste modalità di funzionamento dei componenti in modo da avere il più basso consumo possibile, senza andare però ad intaccare la funzionalità del sistema.
Da queste misure si è deciso di:

1) utilizzare per il modulo wireless la modulazione OOK, preferendola alla FSK, ed utilizzare un potenza di trasmissione pari a +1dBm. La scelta della potenza non è stata presa in base al minor consumo, ma si è scelto una potenza che garantisse un consumo intermedio, per non andare troppo a discapito della distanza fra trasmettitore e ricevitore.
2) Scollegare l'alimentazione ai componenti del trasmettitore durante il periodo di inattività, cioè durante tutto il tempo necessario al sistema di energy harvesting per caricare la capacità di storage per permettere l'acquisizione e l'invio della temperatura, e quindi di evitare di configurare i componenti nella modalità low-power, annullando così il consumo energetico durante il tempo di inattività.
3) Utilizzare la frequenza di 500KHz per l'oscillatore interno al microcontrollore, il quale poi andrà a dare le tempistiche per la comunicazione con tutti i componenti.
4) progettare il voltage monitor in modo da collegare la capacità di storage al sistema trasmittente al raggiungimento di una tensione di 5V e scollegarla

a 3V. questa decisione è stata presa in base alle caratteristiche elettriche dei componenti utilizzati.

5) Sciegliere il valore di 57µF per la capacità di storage, in modo che possa contenere sufficiente energia per permettere al trasmettitore l'acquisizione e l'invio della temperatura.

Infine si è implementato l'intero sistema, constatando che il progetto realizzato soddisfa l'obiettivo prefissato. Il trasmettitore riesce a misurare ed inviare il dato di temperatura ogni 15/20 secondi, il quale viene visualizzato sullo schermo LCD del ricevitore.

Dai risultati ottenuti è emerso la possibilità dell'uso del sistema in molteplici ambiti come ad esempio nel monitoraggio di macchinari e apparecchiature, utilizzando come fonte di energia vibrazioni emesse proprio da quest'ultime, quindi si potrà utilizzare su macchine, treni, lavatrici o macchine industriali. Inoltre nel progetto si potranno aggiungere altri sensori per la rilevazione dell'umidità, della pressione o dell'accellerazione, per aumentare così l'informazione associata all'ambiente in cui il sistema verrà collocato.

Ringraziamenti

Desidero ringraziare il relatore Aldo Romani ed il Co-Relatore Matteo Filippi per la grande disponibilità e per l'aiuto fornito durante tutta la durata della tesi.

Un sentito ringraziamento ai miei genitori, che con il loro supporto morale ed economico, hanno permesso il raggiungimento di questo importante traguardo.

Un ultimo ringraziamento va ai miei fratelli, amici e compagni di studio per essermi stati vicini sia nei momenti difficili che in quelli felici.

Bibliografia

[1] www.onsemi.cn/site/pdf/ONSAR2627_Selezione_di_Elettronica_0613-Copy.pdf

[2] www.energyharvesting.net/

[3] “Progettazione, realizzazione e implementazione di un trasformatore piezoelettrico per l'alta tensione” Tesi di Laurea di Giovanni Cellucci, Università degli studi diBologna.

[4] Microchip PIC16(L)F1508/9 Data Sheet www.microchip.com

[5] Texas Instruments TMP125 Data Sheet www.ti.com

[6] Microchip MRF89XA Data Sheet www.microchip.com

[7] Microchip MRF89XAM9A Data Sheet www.microchip.com

[8] Texas Instruments TPS780 Series Data Sheet www.ti.com