ALMA MATER STUDIORUM – UNIVERSITA' DI BOLOGNA
CAMPUS DI CESENA
SCUOLA DI INGEGNERIA E ARCHITETTURA

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

TITOLO DELL'ELABORATO

PROGETTO DI UN CIRCUITO DI INTERFACCIA BASATO SU TECNOLOGIA NFC PER NODI SENSORI A BASSISSIMO CONSUMO

Tesi in:
ELETTRONICA DEI SISTEMI DIGITALI

Relatore:
Prof. Aldo Romani

Presentato da:
Luca Urbinati

Correlatore:
Dott. Davide Fabbri

Sessione II
Anno accademico 2016-17

Sommario

Dopotutto, i microchip
non sono che sabbia (silicio)
assemblata in modo molto astuto.

Chris Anderson, *Gratis*

Introduzione

Piccoli sistemi elettronici, autonomi a livello energetico e connessi in Rete con la quale scambiano informazioni di vario tipo: è l'Internet of Things (IOT) che negli ultimi anni è diventato sempre più popolare. Si prevede che la sua ascesa nel mercato e la sua penetrazione nelle nostre vite sarà esponenziale [1].

Uno dei più importanti fattori che hanno contribuito alla crescita dell'IOT è l'efficienza energetica raggiunta dall'elettronica che ha permesso di creare circuiti integrati ultra-low power, cioè con dispendio di corrente dell'ordine dei nano ampere in condizioni di stand-by, senza compromettere le prestazioni. A tal riguardo si pensi al nodo sensore a nanocorrenti con funzionalità di datalogger di temperatura basato su microcontrollore[1] [2]. In breve si ricorda il suo funzionamento: il μC trascorre la maggior parte del tempo in sleep mode e periodicamente viene svegliato dall'interrupt di un timer; così facendo entra nella modalità di datalogger grazie alla quale legge il dato di temperatura acquisito dal sensore e lo salva nella propria memoria EEPROM; dopodiché ritorna a dormire.
Anche se non è possibile definirlo un dispositivo IOT perché non presenta un'interfaccia radio di connessione verso la rete Internet, questo sistema embedded ha permesso di analizzare lo stato dell'arte dei consumi degli attuali chip in commercio. Difatti è stata dimostrata [2] una durata teorica di svariati anni con una semplice batteria a bottone come alimentazione. Questa caratteristica ne permette l'utilizzo in applicazioni nelle quali la sostituzione della batteria risulta difficoltosa o non conveniente.

Ecco allora che in questo contesto si colloca il seguente elaborato. Lo scopo è quello di proseguire il progetto del nodo sensore. Infatti esso non presenta una parte essenziale ai fini del suo funzionamento: un'interfaccia di comunicazione con il mondo esterno per il recupero dei dati acquisiti. L'autore aveva accennato a questa mancanza nel Paragrafo 4.4 [2].
Invece, nel progetto che verrà discusso in questa tesi, è stata sfruttata la tecnologia di Near Field Communication (NFC) per due ragioni. In primo luogo perché la NFC si presta ad essere usata come strumento di comunicazione bidirezionale per il trasferimento di dati. In secondo luogo perché permette di alimentare il circuito compiendo *energy harvesting* dalla particolare antenna aggiunta al nodo sensore. In questo modo, in fase di lettura, il circuito viene

[1] D'ora in poi [2] sarà chiamato semplicemente "nodo sensore".

alimentato interamente dal campo magnetico generato dal lettore, senza dover attingere dalla batteria.

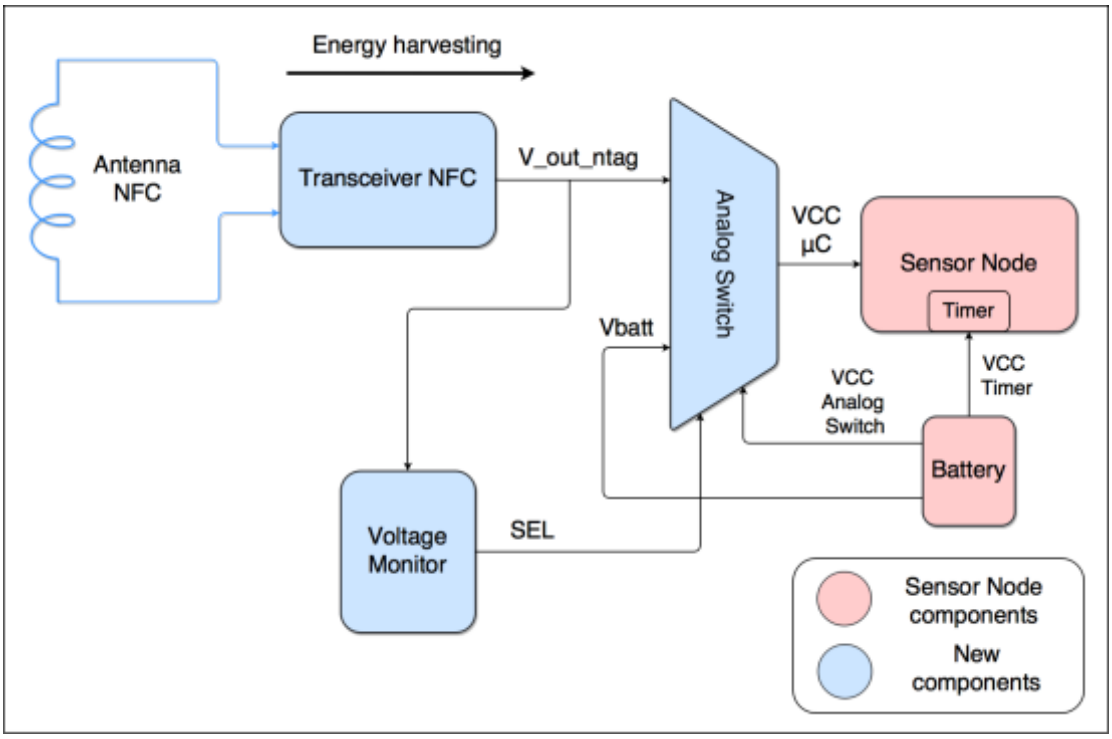


Figura 1 Schema a blocchi del progetto: flusso di energia.

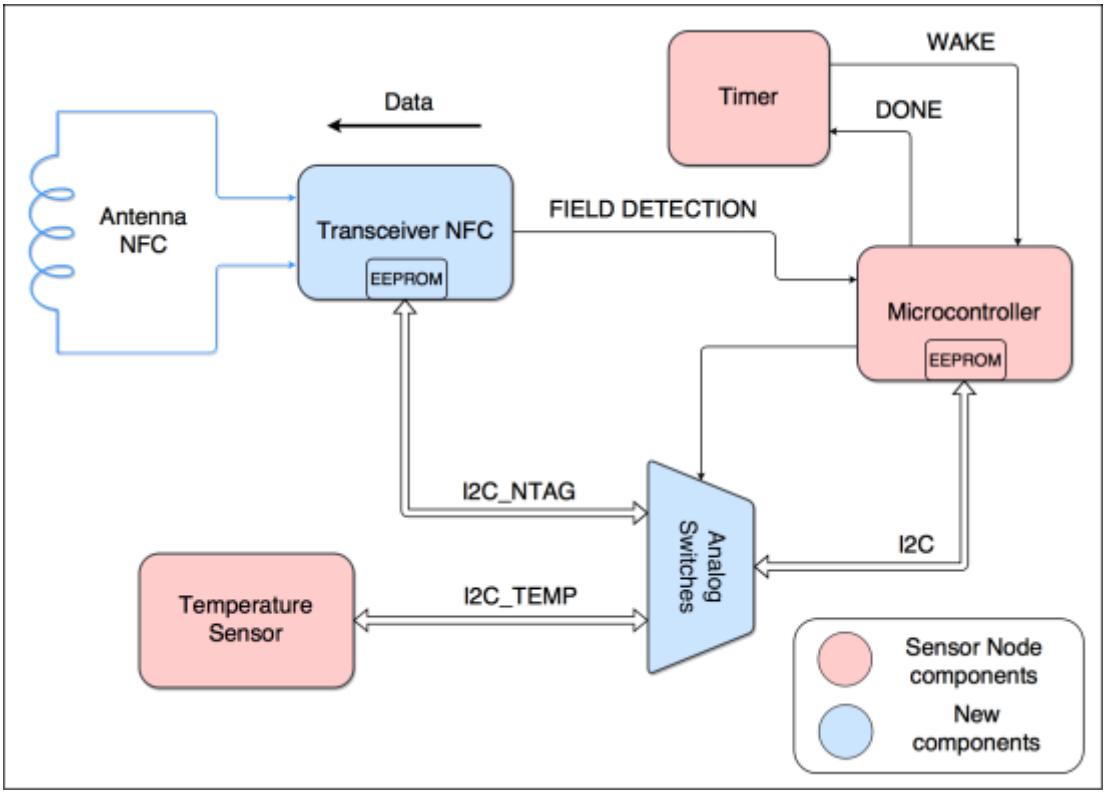


Figura 2 Schema a blocchi del progetto: flusso di dati.

Schematicamente il nuovo circuito ingloba tutte le parti del nodo sensore, ma aggiunge altri componenti come un voltage monitor e tre switch analogici, come si evince osservando Figura 1 e Figura 2.

Questo progetto può essere descritto da due flussi, uno di energia e uno di dati, i quali avvengono sempre in contemporanea.
Per quanto riguarda il flusso di energia, quando l'antenna del transceiver NFC viene immersa in un campo magnetico generato dal *reader* il transceiver fornisce una tensione di alimentazione che viene selezionata dallo switch analogico solo se il voltage monitor sancisce che essa supera i 2.1 V. In tal modo il microcontrollore viene alimentato dal *reader* e non più dalla batteria. Al contrario quando la tensione generata dal transceiver scende sotto la soglia di 1.9 V, la batteria torna ad alimentare il μC. In entrambe le situazioni la batteria alimenta costantemente il timer e gli switch analogici.
Invece nel flusso dati il pin Field Detection del transceiver segnala al microcontrollore che il *reader* è presente. Questo interrupt sveglia il μC il quale, cambiando modalità di funzionamento, va a recuperare i dati salvati nella sua memoria EEPROM e li invia tramite il bus I^2C alla EEPROM del transceiver. Infine è il "lettore" NFC che con un comando di $READ$ legge i dati contenuti nel transceiver, il quale gestisce automaticamente la ricetrasmissione.
Una spiegazione più ampia e dettagliata delle fasi del circuito verrà data nel corso della trattazione.

L'elaborato in questione parte dallo studio generale della tecnologia NFC e delle antenne NFC. Si discutono infatti le leggi fisiche dell'elettromagnetismo che stanno alla base del trasferimento di energia e di dati, fino ad arrivare alla realizzazione di un prototipo di antenna su PCB (Printed Circuit Board). Successivamente si prendono in esame le caratteristiche dei vari componenti utilizzati e i motivi della loro scelta, per poi passare alla fase di programmazione del microcontrollore. Si conclude con l'analisi delle forme d'onda catturate con oscilloscopio e *protocol analyzer* per dimostrare il corretto funzionamento del circuito e si misura l'impatto dei componenti aggiunti sui consumi del sistema.

1. Tecnologia Near Field Communication (NFC)

In questo capitolo vengono fornite le basi teoriche della tecnologia NFC spiegando cos'è e come funziona. Vengono proposti alcuni esempi applicativi di comunicazione, un rapido confronto con la tecnologia RFID e una panoramica sui tipi di NFC.

1.1. Cos'è NFC

La tecnologia Near Field Communication è una forma di comunicazione wireless nata a cavallo degli anni 2000. Essa è un sottoinsieme della più ampia famiglia della Radio Frequency Identification (RFID) che, per definizione, è una tecnologia che usa le onde radio per identificare univocamente oggetti che possiedono particolari etichette elettroniche le quali hanno la capacità di memorizzare informazioni. NFC è nata per semplificare le transazioni di denaro, lo scambio di dati, il pairing[2], le connessioni wireless tra due oggetti quando si trovano in stretta prossimità l'uno all'altro. Perciò dispone di un sistema di sicurezza di cifratura del canale di comunicazione. La distanza massima sancita dal protocollo è di 10 cm. La frequenza centrale è di 13.56 MHz e la sua larghezza di banda dipende dalla modulazione adottata. NFC fa parte della banda non licenziata ISM (Industrial, Scientific and Medical).

1.2. I dispositivi

1.2.1. Dispositivi attivi e passivi

Prima di tutto è necessario chiarire cosa si intenda per dispositivi passivi e attivi [3] [4] [5] nella tradizione RFID. I primi, chiamati *tag* o *target*, normalmente non possiedono una propria fonte di energia, ma sono alimentati a distanza da quella del campo elettromagnetico fornito dal lettore. Per questo motivo essi sono in grado solo di rispondere ad una interrogazione, essere letti e scritti, ma non riescono a iniziare un dialogo autonomamente, cioè non generano un proprio campo. I secondi, invece, denominati *reader*, sono dotati di una propria fonte di alimentazione che può essere una batteria se è richiesto che siano

[2] Pairing: accoppiamento tra dispositivi. NFC non richiede un setup della connessione come avviene per esempio con l'accoppiamento di dispositivi Bluetooth

portatili. Gli attivi iniziano il dialogo con i passivi e dunque sono anche detti *initiator*. Non solo hanno la capacità di ispezionare il contenuto dei *tag*, ma possono perfino cambiarlo e scrivere nuove informazioni a patto che ne abbiano l'autorizzazione.

Lo standard NFC presenta un parco di dispositivi categorizzabili in: smartphone NFC, *tag* passivi e *reader* attivi. Per i *tag* passivi e i *reader* attivi si applicano le stesse definizioni dell'RFID. D'altra parte per gli smartphone la distinzione tra *tag* e *reader* non è più tanto marcata. Infatti esistono tre modalità di comunicazione descritte nel Paragrafo 1.3 in cui lo smartphone è il protagonista.

1.2.2. *Tag* statici e dinamici

Soffermandoci sui *tag* NFC, essi sono detti statici se si comportano come descritto nel precedente sottoparagrafo, mentre sono dinamici se dispongono di un'interfaccia seriale aggiuntiva, oltre a quella a RF [4] [5]. Di solito è di tipo I^2C o SPI e viene usata per interagire con una MCU. Così il *target* rende disponibile la propria memoria EEPROM o, talvolta, la SRAM all'MCU. Pertanto i *tag* dinamici possono essere letti e scritti sia dall'interfaccia RF che da quella seriale. Inoltre la comunicazione attraverso il bus seriale può avvenire anche in assenza di campo magnetico esterno perché l'alimentazione delle linee può essere fornita dal microcontrollore.

1.3. Le modalità operative e di comunicazione

Le modalità operative sono due: in quella passiva solo un NFC è l'*initiator* che genera il campo RF, l'altro è passivo e gioca il ruolo di *target*; in quella attiva entrambi producono il campo elettromagnetico.

Ulteriormente NFC ha tre modalità di comunicazione [3] [4] [5]. Il tutto è riassunto in Figura 3.

- La prima prende il nome di ***Card Emulation*** ed è una modalità operativa passiva. Lo smartphone diventa un *target* NFC passivo e si comporta come una carta di credito contactless. L'*initiator* è il lettore di carte che può essere il POS (Point of Sale) Contactless che genera il campo magnetico. Questa situazione si sperimenta facilmente al momento del

pagamento di un prodotto in un negozio se si intende pagare con lo smartphone.

- La seconda viene chiamata ***Read/Write*** ed è anch'essa una modalità operativa passiva. Lo smartphone è l'*initiator* che produce il campo RF e manda comandi di lettura o scrittura al *target* che generalmente è senza batteria e fa *energy harvesting* dal campo dell'interrogatore. Si immagini un utente che sia interessato al servizio offerto da uno smart poster. Gli basta leggere con lo smartphone un'etichetta adesiva posta sul cartellone pubblicitario per essere rediretto alla pagina web del fornitore del servizio. Oppure si pensi al programmatore di quel *tag* inizialmente vuoto. Egli avrà adoperato probabilmente uno smartphone in modalità scrittura per memorizzare nel *tag* l'indirizzo URL dell'azienda promotrice del servizio.
  Grazie alla flessibilità dell'NFC il *target* può non solo essere un *tag* o una carta contacless, ma anche un altro dispositivo NFC in modalità card emulation.
- La terza è la ***Peer-to-Peer*** che è l'unica ad essere una modalità operativa attiva in cui entrambi gli NFC generano il proprio campo. Lo scopo è lo scambio di dati tra i due. L'*initiator* comincia la comunicazione e crea un canale con il *target*. Poi essi si parlano a turno, applicando la regola listen-before-talk. Quindi alternativamente uno si comporta da interrogatore e l'altro da target. Come esempio si pensi a due utenti che si vogliono scambiare contatti telefonici, canzoni o contenuti multimediali semplicemente avvicinando i loro smartphone sul retro.

La modalità di comunicazione che viene impiegata nel progetto è solamente quella di *Read* perché è l'applicazione più comune per la maggioranza dei sistemi embedded. Perciò la trattazione si focalizzerà sulla comunicazione in modalità operativa passiva.

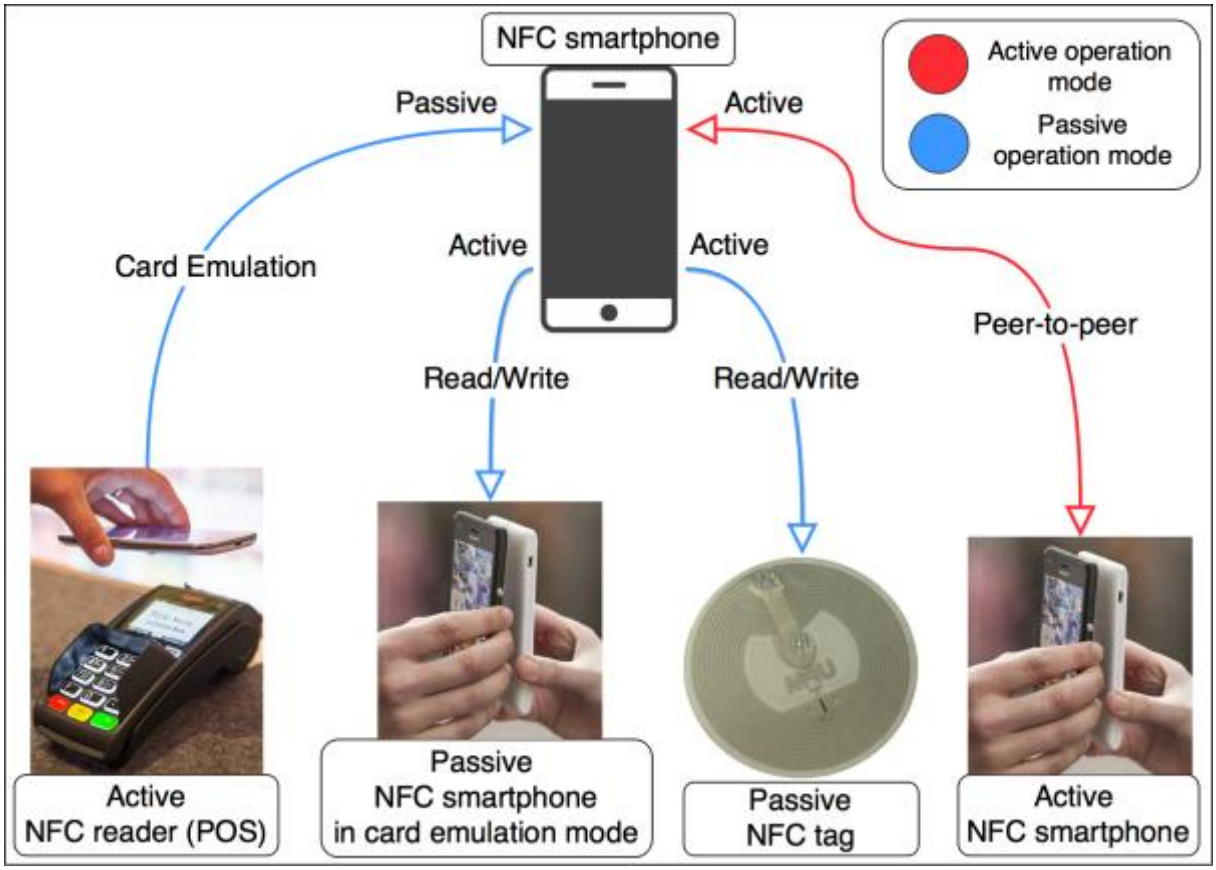


Figura 3 Riassunto delle modalità operative e di comunicazione.

1.4. NFC vs. RFID

In questa sezione viene stilato un breve confronto tra le differenze delle tecnologie NFC e RFID.

- Frequenza operativa. Come già sopracitato NFC fa parte della famiglia RFID. Infatti la sua frequenza operativa è di 13.56MHz, la stessa sfruttata nella categoria High Frequency (HF) dell'RFID, che in aggiunta comprende range in Low Frequency (LF, 125-134KHz), in Ultra High Frequency (UHF, 856-960MHz) e Microonde (2.45-5.8GHz).
- Distanza di lavoro. Per gli RFID può arrivare fino a 25 metri per i passivi e fino a 100 metri per gli attivi, mentre per gli NFC si parla di prossimità ("proximity") quando la distanza massima è di 10 cm e di vicinanza ("vicinity") quando è fino a 20 cm. La maggior parte degli NFC lavora però a meno di 5 cm.
- Distinzione *tag* e *reader*. É presente solo negli RFID. Infatti negli NFC si parla solo di "dispositivo" o più precisamente di *target* e *initiator*.

- Bidirezionalità nella comunicazione. Tra RFID la direzione di comunicazione è sempre dal *reader* al *tag*, il quale risponde agli stimoli, ma non inizia il dialogo. Tra NFC esiste pure la modalità peer-to-peer in cui entrambi i dispositivi sono attivi alternativamente.
- Compatibilità. Lo standard NFC è l'ISO 18092 NFCIP-1 e l'ISO 21841 NFCIP-2 (Near Field Communication Interface and Protocol 1 e 2) è capace di interagire con tutti i protocolli RFID HF proprio perché è stato sviluppato a partire dall'ISO 14443. In particolare è compatibile con i *tag*
- ISO 14443, ISO 15693 e NXP (ex Philips Semiconductors) MIFARE.

2. Le antenne NFC

Le antenne sono una parte chiave per il funzionamento dei dispositivi NFC e richiedono un'attenta progettazione. Infatti le antenne dei *tag* passivi devono essere in grado di assorbire energia, per ricavare potenza per l'alimentazione dell'intero sistema embedded, e di riflettere la potenza incidente per rispondere all'interrogazione.

2.1. I principi fisici

Come già detto, i *tag* sono dispositivi passivi che non hanno batteria e richiedono che sia un apparato esterno attivo a fornirgliela. In questo paragrafo vengono pertanto riassunti i fenomeni fisici dell'elettromagnetismo che determinano il trasferimento di potenza e di dati tra due dispositivi NFC in modalità operativa passiva. Per questa parte si fa riferimento a [6].

2.1.1. Il trasferimento di potenza

Partendo dalla legge della circuitazione di Ampere e dai risultati di Biot-Savart, una corrente che scorre in un conduttore genera un campo magnetico attorno ad esso che dipende dalla sua forma. Un avvolgimento con N giri della stessa area A percorso da corrente genera un campo magnetico prevalentemente perpendicolare all'area stessa perché i flussi magnetici dovuti alle singole spire si sommano tra loro. Il flusso totale ϕ può essere espresso come:

$$\phi = \sum_{N} \varphi_N = N \cdot \varphi = N \cdot \mu \cdot H \cdot A \quad [Wb] \tag{1}$$

dove μ è la permeabilità magnetica del mezzo in cui è immerso l'avvolgimento.
La relazione tra il flusso magnetico e la corrente è chiamata induttanza o autoinduttanza e si denota con L:

$$L = \frac{\phi}{I} = \frac{N \cdot \varphi}{I} = \frac{N \cdot \mu \cdot H \cdot A}{I} \quad [\mathrm{H}] \tag{2}$$

Ora si introduce un secondo avvolgimento di area A_2 nelle vicinanze del primo conduttore di area A_1 nel quale scorre una corrente. Quello che accade è che una parte del flusso magnetico totale che attraversa A_1 tende ad attraversare anche A_2. Questo fenomeno prende il nome di accoppiamento induttivo e viene descritto dalla mutua induttanza M_{21}:

$$M_{21} = \frac{\phi_{21}(I_1)}{I_1} \text{ [H]} \tag{3}$$

che sancisce che il flusso concatenato con il secondo avvolgimento ($\phi_{21}(I_1)$) è direttamente proporzionale (M_{21}) all'intensità della corrente che genera il flusso stesso (I_1). Se il campo magnetico è omogeneo, la (3) diventa:

$$M_{21} = \frac{\phi_{21}(I_1)}{I_1} = \frac{N_1 \cdot \mu \cdot H(I_1) \cdot A_1}{I_1} \text{ [H]} \tag{4}$$

Inoltre si dimostra che la relazione tra le mutue induttanze è:

$$M = M_{21} = M_{12} = \frac{\phi_{12}(I_2)}{I_2} \text{ [H]} \tag{5}$$

Nel caso più generale in cui si hanno due circuiti, ognuno dei quali collegato ad un generatore, bisogna tenere conto sia della mutua induttanza che del coefficiente di autoinduzione, per cui si ha:

$$\begin{cases} \phi_1(I_1, I_2) = L_1 \cdot I_1 + M \cdot I_2 \text{ [Wb]} & (6) \\ \phi_2(I_1, I_2) = M \cdot I_1 + L_2 \cdot I_2 \text{ [Wb]} & (7) \end{cases}$$

Il principio appena discusso è lo stesso che sta alla base del trasformatore.
A questo punto una variazione di flusso magnetico attraverso una superficie delimitata da un circuito elettrico genera una forza elettromotrice indotta pari all'opposto della variazione temporale del flusso. Il suddetto fenomeno è descritto dalla legge di Faraday-Neumann-Lenz:

$$v_i = -\frac{d\phi(t)}{dt} \text{ [V]} \tag{8}$$

Se nel primo avvolgimento precedente scorre una corrente tempo variante $i_1(t)$, essa genera un flusso magnetico tempo variante $\frac{d\phi_1(t)}{dt} = \frac{d\phi(i_1(t))}{dt}$ che, per la legge (8), porta all'induzione di una tensione ai capi di entrambi gli avvolgimenti. Per l'autoinduttanza, il cambiamento di flusso generato dalla variazione nel tempo della corrente $i_1(t)$ induce una tensione variabile $v_1(t)$ nello stesso avvolgimento; per la mutua induttanza, invece, tale cambiamento induce una tensione $v_2(t)$ tempo variante nel secondo avvolgimento. Poi se al secondo circuito è collegato un carico, si genera una ulteriore corrente $i_2(t)$ anch'essa indotta e variabile.

Si immagini ora che il primo avvolgimento sia l'antenna di un dispositivo NFC attivo e il secondo quello di un *target* passivo. In Figura 4 è rappresentato un circuito equivalente che mostra tutti i fenomeni fisici sopra descritti che avvengono quando i due dispositivi sono vicini tra loro. L_1 e L_2 denotano le antenne, M l'accoppiamento induttivo tra le due, R_2 la resistenza dell'avvolgimento dell'antenna del *target*, R_L la resistenza di ingresso del circuito a valle dell'antenna che modella il suo assorbimento di corrente, $v_1(t)$ e $v_2(t)$[3] le tensioni indotte dal flusso $\phi(t)$ del campo H generato dalla corrente $i_1(t)$ che scorre nella prima antenna, $i_2(t)$ la corrente indotta nella seconda antenna.

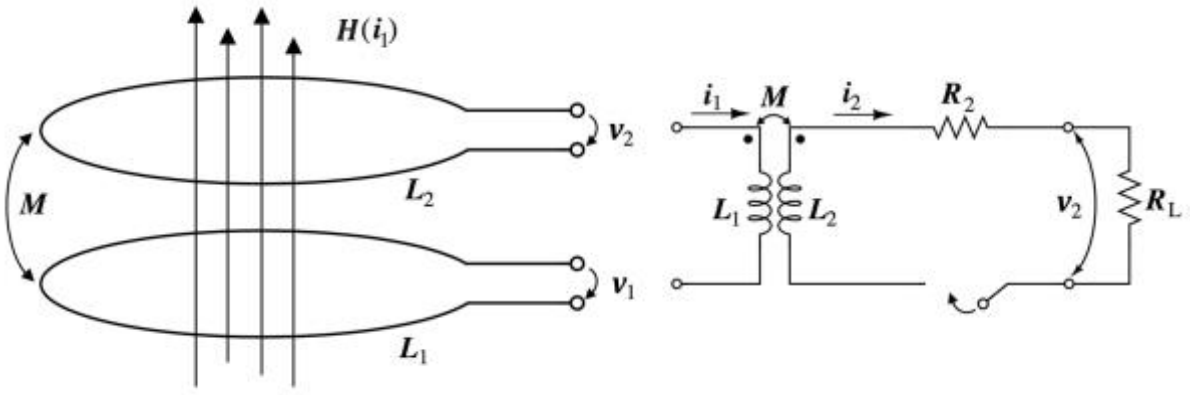


Figura 4 A sinistra, due spire accoppiate magneticamente; a destra, il circuito equivalente delle due spire accoppiate [6].

2.1.2. Il trasferimento dei dati

Per comprendere come avviene fisicamente il trasferimento di dati, illustrato in Figura 5 e Figura 6, si continua il ragionamento interrotto in Figura 4. Applicando nuovamente la legge di Faraday-Neumann-Lenz (8) alla situazione precedente, si può considerare un ulteriore flusso magnetico variabile $\phi_2(t)$, opposto a $\phi_1(t)$[4], prodotto dalla corrente $i_2(t)$ precedentemente indotta nell'antenna del *target*. Dunque derivando rispetto al tempo l'equazione (7) si ottiene:

$$v_2 = -\frac{d\phi_1(t)}{dt} = \frac{d\phi_2(t)}{dt} = M\frac{di_1}{dt} - L_2\frac{di_2}{dt} - i_2R_2 \ \text{[V]} \qquad (9)$$

[3] In realtà v_2 è la tensione indotta a cui si è sottratta la caduta di potenziale su R_2.
[4] Questo scelta del verso di riferimento fa sì che M sia < 0.

Dal momento che i_1 e i_2 sono solitamente correnti alternate sinusoidali, si può riscrivere l'equazione (9) usando i numeri complessi:

$$v_2 = j\omega M \cdot i_1 - j\omega L_2 \cdot i_2 - i_2 R_2 \ \text{[V]} \tag{10}$$

e, sostituendo i_2 con $\frac{v_2}{R_L}$, si ha l'espressione finale:

$$v_2 = \frac{j\omega M \cdot i_1}{1 + \frac{j\omega L_2 \cdot R_2}{R_L}} \begin{cases} R_L \to \infty : v_2 = j\omega M \cdot i_1 \\ R_L \to 0 : v_2 \to 0 \end{cases} \ \text{[V]} \tag{11}$$

Questa espressione permette di capire che il *target* trasmette i dati al dispositivo attivo con una tecnica chiamata back-scattering o load modulation [6] [7]. Uno switch comandato dai dati da trasmettere varia l'impedenza d'antenna del *tag* passivo vista dal dispositivo attivo, il quale riesce a distinguere lo stesso pattern di dati inviato dal *target*, rilevando la riflessione della propria onda trasmessa. Infatti modulando internamente il carico R_L si cambia la corrente i_2 che scorre nell'antenna del *target*. Così questa variazione genera un campo modulato che viene percepito dal dispositivo NFC attivo per via dell'accoppiamento induttivo e dei ragionamenti svolti in precedenza.

2.2. L'applicazione dei principi fisici

La sequenza di eventi nell'interrogazione del *target* da parte dell'*initiator* prevede i seguenti passaggi.
Dal punto di vista del dispositivo attivo, la sua logica di controllo invia i dati dell'interrogazione al blocco trasmettitore che genera il segnale per l'antenna, poi, completata questa fase, mantiene il campo magnetico non modulato per consentire al *target* di generare una risposta.
D'altra parte il *target* passivo compie *energy harvesting*, rettifica e limita la corrente raccolta con un circuito raddrizzatore unito ad un limitatore.
Difatti la corrente alternata che scorre nell'antenna dell'*initiator* induce un campo magnetico pulsante che si concatena con l'antenna del *target*. In seguito, un circuito di segnalazione abilita il funzionamento del resto della circuiteria quando la tensione di alimentazione è sufficientemente alta. Quindi l'informazione ricevuta viene demodulata e decodificata e i dati sono forniti alla logica di controllo del *target* che elabora la risposta. In questo modo i dati da reinviare pilotano uno switch che modula l'impedenza d'antenna del *target* con la sopra descritta tecnica di back-scattering.
Infine l'*initiator* percepisce le onde riflesse dell'antenna del *target* con un

circuito di rilevazione e procede alla demodulazione e decodifica della risposta. Generalmente le informazioni hanno mudulazione digitale ASK (Ampliture Shift Keying), ma si utilizza anche la PSK (Phase Shift Keying) e FSK (Frequency Shift Keying).
Per maggior chiarezza, uno schema a blocchi di un *tag* è riportato in Figura 19 al Paragrafo 3.1.

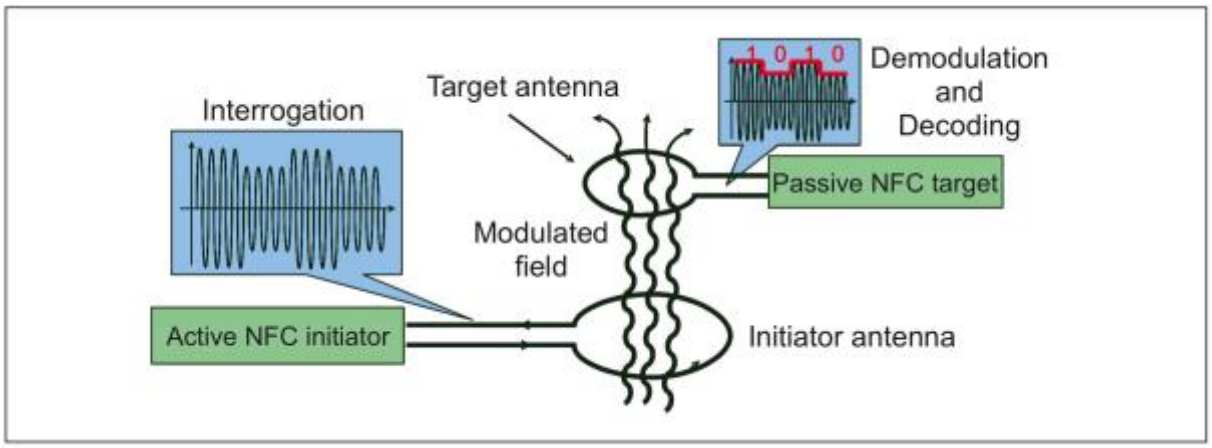


Figura 5 Comunicazione da dispositivo NFC attivo a *target* NFC passivo [7].

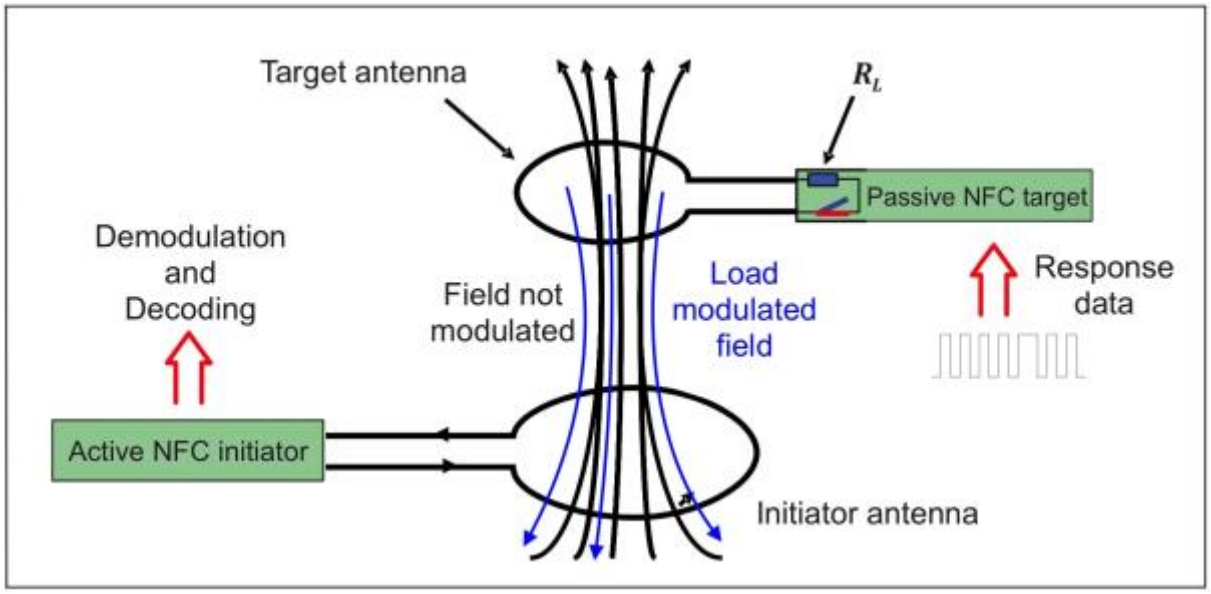


Figura 6 Comunicazione da *target* NFC passivo a dispositivo NFC attivo [7].

2.3. L'antenna di tag NFC per PCB

Questa sezione contiene una prima parte dedicata alle classi di antenne NFC e alla teoria sulla progettazione del design per Printed Circuit Board (PCB). Seguono le principali figure di merito che bisogna valutare per compiere *energy harvesting*. Nella parte finale si implementa l'algoritmo per il calcolo dei

parametri geometrici dell'antenna con Matlab e si verifica la validità dei suoi risultati con il tool online $eDesignSuite$ [8] di STMicroelectronics.
Infine si realizza un prototipo a circuito stampato di un'antenna di classe 6 con software KiCad.

2.3.1. Le classi di antenne NFC

In Figura 7 sono segnalate le misure massime e minime delle sei classi di antenne NFC definite nell'ISO/IEC 14443 [9]. La scelta del modello dipende dall'applicazione. Nel caso di smart card la classe 1 è quella più usata perché è in grado di assorbire l'energia del campo senza problemi. All'opposto la classe 6 con le sue piccole dimensioni si presta ad applicazioni in cui l'obiettivo è l'integrazione del sistema. Il suo drawback è perciò la scarsa abilità di catturare energia che obbliga l'utilizzatore a ridurre la distanza di comunicazione con il *target*.

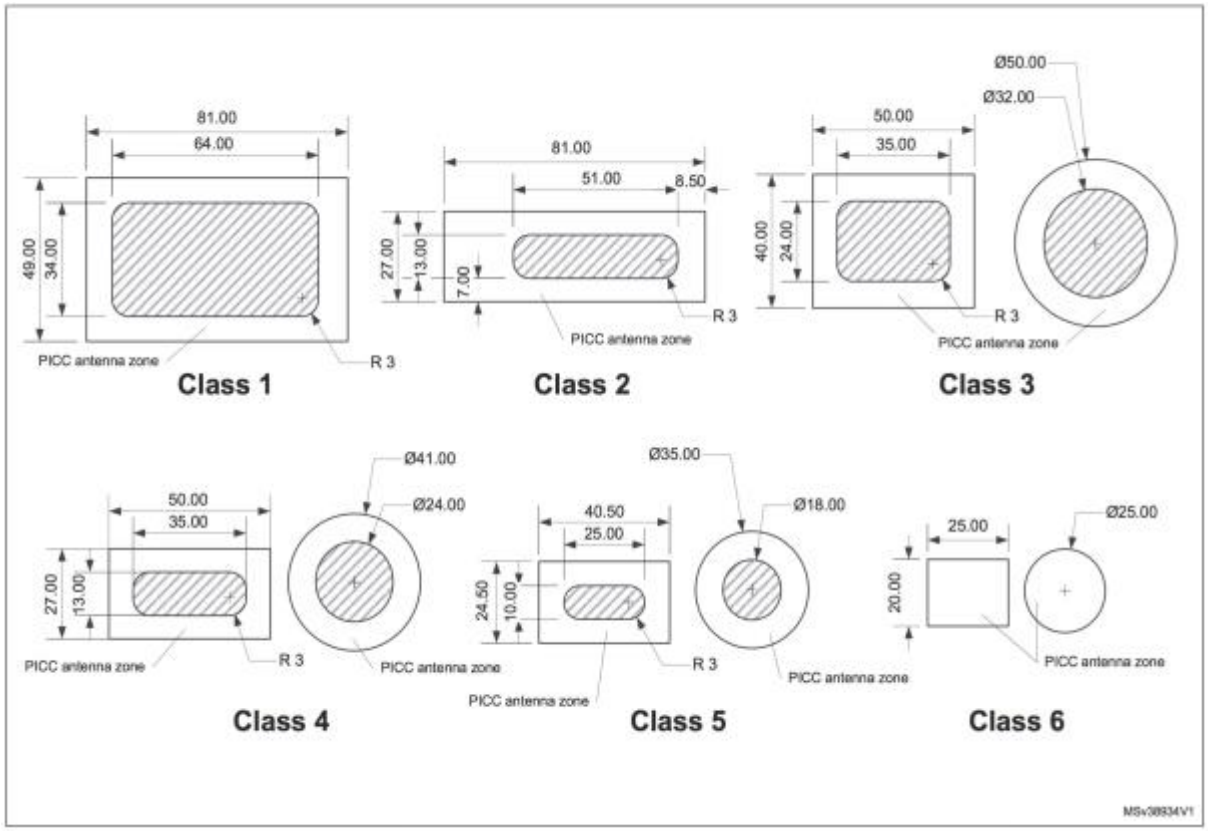


Figura 7 Le misure (in mm) delle sei classi di antenne NFC definite nell'ISO/IEC 14443 [3].

2.3.2. Come progettare un'antenna NFC su PCB

Questa prima parte teorica è basata sulla guida [10] di NXP sul design delle

antenne NFC. Per questo vengono richiamati solo i concetti chiave e per maggiori dettagli si rimanda a tale application note.

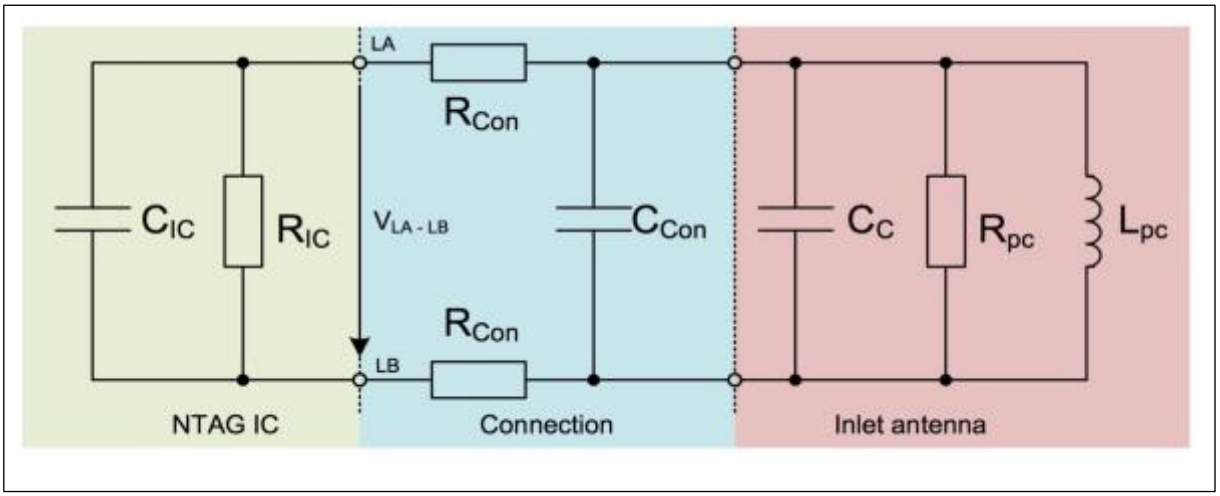


Figura 8 Circuito equivalente *tag* NFC [10].

In primo luogo, un dispositivo NFC passivo può essere rappresentato con un circuito equivalente formato da tre parti come mostrato in Figura 8.
A sinistra si ha il circuito integrato (IC) transceiver, cuore del *tag* NFC, schematizzato da una resistenza di ingresso R_{IC} e una capacità di ingresso C_{IC}[5]. É possibile recuperare questi valori dal datasheet del componente scelto.
A destra si ha l'antenna, formata da una capacità C_C, che modella principalmente gli accoppiamenti capacitivi che si instaurano tra i vari avvolgimenti dell'antenna; una resistenza parallelo R_{pc}, per le perdite del rame; un'induttanza parallelo L_{pc}, creata dall'avvolgimento.
Al centro si hanno resistenze di connessione R_{Con} e capacità parassite C_{Con} dovute al collegamento fisico tra il transceiver e l'antenna. Quasi sempre esse sono trascurabili se $R_{Con} \ll 1\ \Omega$.

Risulta vantaggioso semplificare l'attuale circuito con quello equivalente di Figura 9.

[5] Resistenza e capacità del transceiver sono di ingresso perché valutate dal punto di vista dell'antenna.

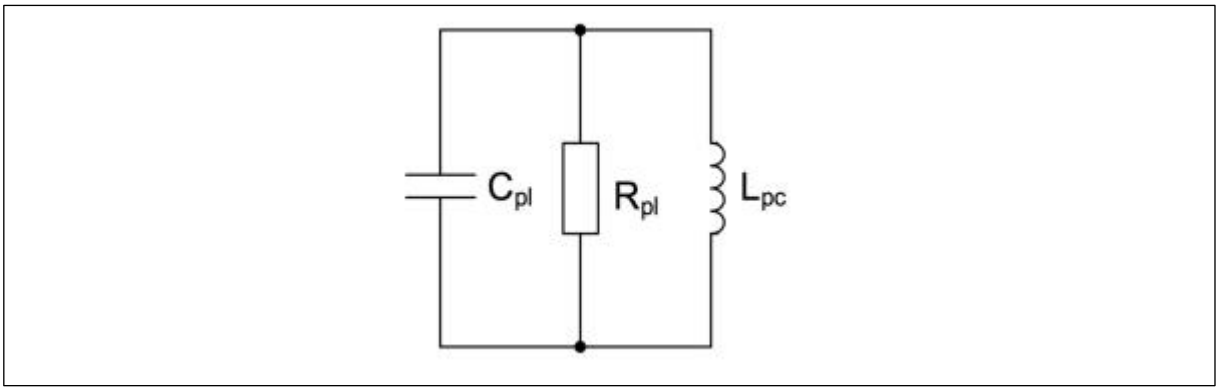


Figura 9 Circuito equivalente semplificato *tag* NFC [10].

dove la resistenza parallelo complessiva è:

$$R_{pl} = \frac{R_{IC} \cdot R_{pc}}{R_{IC} + R_{pc}} \ [\Omega] \tag{12}$$

e la capacità totale è:

$$C_{pl} = C_{IC} + C_{Con} + C_c \ [\text{F}] \tag{13}$$

Dal momento che la capacità e la resistenza di ingresso del transceiver dipendono dalla tensione di ingresso V_{LA-LB} prodotta dall'antenna, risulta necessario definire un valore di tensione costante come riferimento. É
É
stata è scelta la minima tensione operativa $V_{LA-LB\ min}$ di ingresso dell'IC. Con la lettera "T" si indicano quei parametri che vanno valutati per $V_{LA-LB} = V_{LA-LB\ min}$. Dunque le (12) e (13) diventano:

$$R_{plT} = \frac{R_{ICT} \cdot R_{pc}}{R_{ICT} + R_{pc}} \ [\Omega] \tag{14}$$

$$C_{plT} = C_{ICT} + C_{Con} + C_c \ [\text{F}] \tag{15}$$

Lo scopo del progetto di design è quello di determinare le dimensioni geometriche dell'antenna da stampare su PCB in modo che essa abbia un'induttanza L_{pc} pari all'induttanza obiettivo:

$$L_o = \frac{1}{(2 \cdot \pi \cdot f_{ideal})^2 \cdot C_{plT}} \ [\text{H}] \tag{16}$$

Infatti si vuole che l'induttanza dell'antenna L_{pc} risuoni con la capacità complessiva C_{plT} alla frequenza f_{ideal}. In letteratura [10] si consiglia di usare

una f_{ideal} di 14.50 MHz, al posto dello standard di 13.56 MHz, per avere una maggior distanza di lettura/scrittura.

Alla frequenza di risonanza l'impedenza totale del circuito equivalente è minima e diventa un valore puramente reale $Z_{tot} = R_{plT}$. Di conseguenza la corrente che scorre nell'antenna e la tensione V_{LA-LB} fornita al transceiver sono massimi, come anche l'energia raccolta. In Figura 10 si osservano tre esempi di diverso tuning di frequenza per tre *tag*. Il secondo è quello migliore perché ha il picco di risonanza in corrispondenza di quello dell'*initiator*, pertanto è in grado di raccoglie più energia rispetto agli altri.

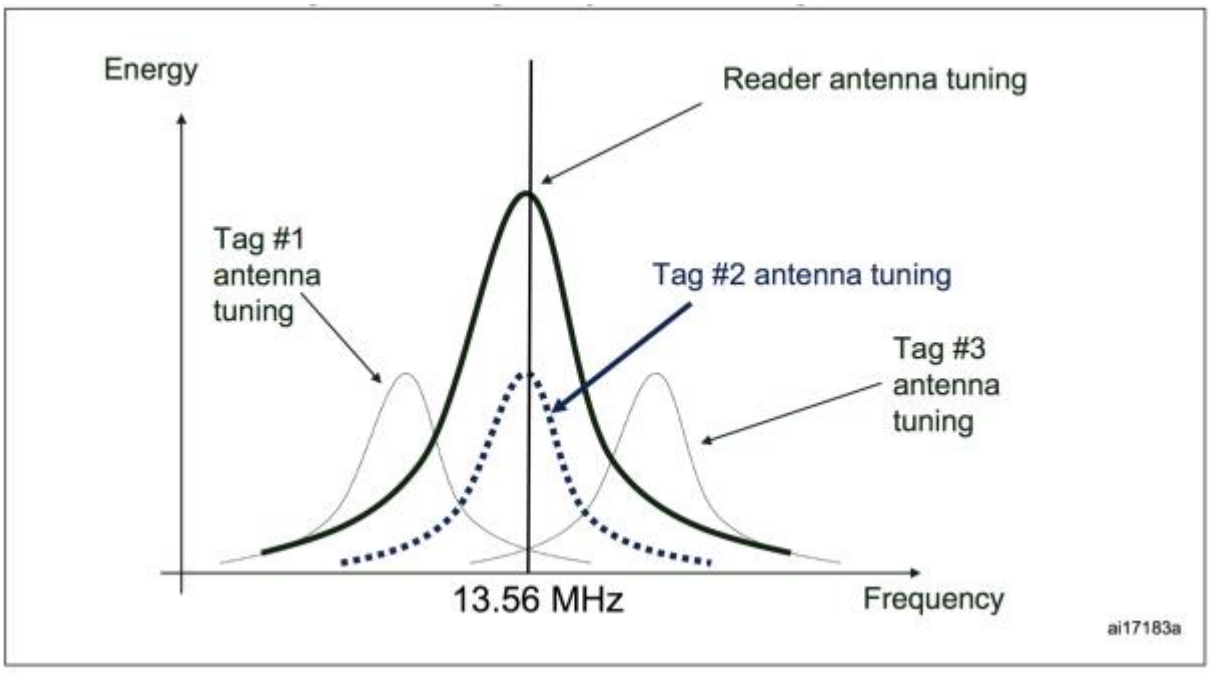


Figura 10 Tre esempi di tuning tra un *reader* e tre *tag* NFC [7].

Per determinare le dimensioni geometriche dell'antenna a partire dall'induttanza obiettivo L_o si usa la (17) valida per antenne rettangolari, uguagliandola a L_o:

$$L_{calc} = \frac{\mu_0}{\pi} \cdot [x_1 + x_2 - x_3 + x_4] \cdot N_c^p \quad [\text{H}] \tag{17}$$

con:

- $x_1 = a_{avg} \cdot ln\left[\frac{2 \cdot a_{avg} \cdot b_{avg}}{d \cdot \left(a_{avg} + \sqrt{a_{avg}^2 + b_{avg}^2}\right)}\right]$,

- $x_2 = b_{avg} \cdot ln\left[\frac{2 \cdot a_{avg} \cdot b_{avg}}{d \cdot \left(b_{avg} + \sqrt{a_{avg}^2 + b_{avg}^2}\right)}\right]$,
- $x_3 = 2 \cdot \left[a_{avg} + b_{avg} - \sqrt{a_{avg}^2 + b_{avg}^2}\right]$,
- $x_4 = \frac{a_{avg} + b_{avg}}{4}$,
- $d = \frac{2 \cdot (t + w)}{\pi}$ il diametro equivalente della pista,
- a_0 e b_0 le massime dimensioni dell'antenna,
- $a_{avg} = a_0 - N_c \cdot (g + w)$ e $b_{avg} = b_0 - N_c \cdot (g + w)$ le dimensioni medie dell'antenna,
- t lo spessore della pista, w la larghezza della pista, g il gap tra due piste, N_c il numero di giri, p l'esponente del numero di giri.

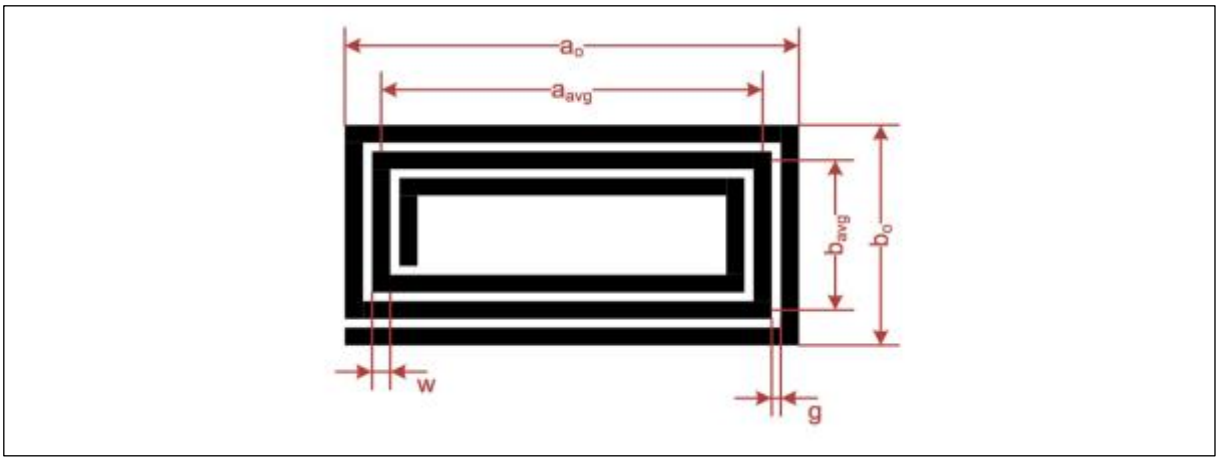


Figura 11 Design di un'antenna rettangolare [10].

2.3.3. I fattori che influenzano la raccolta di energia

Per ottenere una raccolta di energia efficiente dall'antenna occorre valutare le successive figure di merito:

- *le dimensioni dell'antenna*. Antenne più grandi riescono a fornire al transceiver più energia perché l'accoppiamento induttivo viene favorito;

- il tuning dell'antenna. Come già discusso in precedenza, si recupera più energia se *target* e dispositivo attivo hanno la stessa frequenza di risonanza (13.56 MHz circa);
- *numero di giri*. In generale un elevato numero di giri favorisce l'accoppiamento induttivo. Eppure non sempre questo permette di raccogliere più energia perché aumentano le perdite resistive. Spesso un'antenna con meno giri, quindi minor induttanza, ma con una capacità esterna aggiuntiva recupera più energia;
- *la potenza del dispositivo attivo*. Più è alta l'energia del campo che viene trasmessa dall'*initiator* (PCD: "proximity coupling device") e maggiore è quella immagazzinata e resa disponibile al circuito. In Tabella 1 sono riportati i valori massimi e minimi dell'intensità del campo magnetico ammessi dallo standard ISO/IEC 14443. Lo standard prevede che ci siano limiti diversi per ogni classe di antenna NFC (PICC: "proximity chip card");
- *la polarizzazione delle antenne*. Il rispettivo orientamento delle antenne l'una con l'altra è fondamentale per lo scambio di energia. L'orientamento migliore è quando le aree delle antenne sono affacciate l'una all'altra parallelamente. Il caso peggiore è quando sono perpendicolari perché lo scambio di energia è nullo. Questo fenomeno, descritto in Figura 12, deriva dalle proprietà fisiche del campo magnetico. Infatti, le linee di campo generate dagli avvolgimenti sono perpendicolari e più intense al centro dell'area della spira, mentre si affievoliscono ai lati fino a richiudersi su se stesse;
- *la distanza operativa*. Minor distanza tra i dispositivi, significa più energia raccolta.

Molti di questi fattori come le dimensioni dell'antenna (cioè la sua area A), il numero di giri N, la potenza del dispositivo (espressa come campo magnetico H) sono verificabili dalle (2) e (3) del Paragrafo 2.1.1.

	PCD	
	H_{min}	H_{max}
Measured with Reference PICC 1	1,5 A/m (rms)	7,5 A/m (rms)
Measured with Reference PICC 2	1,5 A/m (rms)	8,5 A/m (rms)
Measured with Reference PICC 3	1,5 A/m (rms)	8,5 A/m (rms)
Measured with Reference PICC 4 (optional)	2,0 A/m (rms)	12 A/m (rms)
Measured with Reference PICC 5 (optional)	2,5 A/m (rms)	14 A/m (rms)
Measured with Reference PICC 6 (optional)	4,5 A/m (rms)	18 A/m (rms)

Tabella 1 Valori massimi e minimi del campo magnetico generato dal dispositivo attivo permessi dallo standard ISO/IEC 14443 [9].

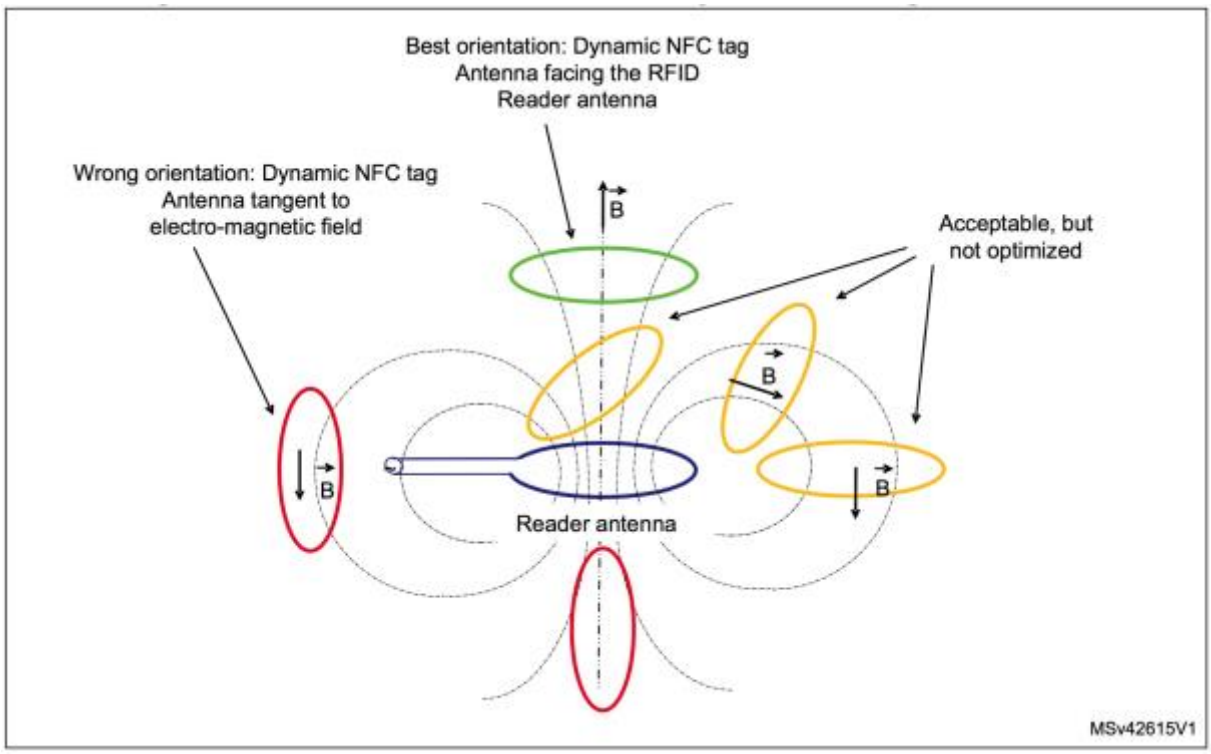


Figura 12 Il fenomeno della polarizzazione tra un *reader* e un *tag* NFC [7].

2.3.4. La realizzazione dell'antenna NFC

L'application note [[6] propone un algoritmo per trovare le caratteristiche dell'antenna NFC per PCB. Esso consiste nel determinare L_o, uguagliare L_o a L_{calc} variato entro il $\pm$ 20% ed estrarre le cinque quadruple dei valori a_0, b_0, N_c e w che soddisfano le cinque uguaglianze. Nel calcolo bisogna tenere conto

delle dimensioni massime e minime della classe di antenna che si vuole realizzare e bisogna cercare di massimizzare l'area attiva data dal prodotto $A_{active} = A_c \cdot N_c = a_{avg} \cdot b_{avg} \cdot N_c$ per un'antenna rettangolare.
Dopodiché, si devono stampare su PCB le cinque antenne per misurare la loro frequenza di risonanza e infine si sceglie quella che si avvicina di più alla f_{ideal} voluta. Nel caso in cui non si ottenga un'antenna soddisfacente, si può eseguire con qualche modifica un secondo run dell'algoritmo (vedi Paragrafo 4.8 di [6]), oppure si può aggiustare la frequenza di risonanza aggiungendo una capacità esterna in parallelo ai terminali dell'antenna di valore opportuno.

É
É
stato deciso di realizzare un'antenna di classe 6, cioè che abbia dimensioni massime di 25x20 mm. L'algoritmo implementato in Matlab con la funzione *fmincon* cerca di far tendere a zero la differenza tra due espressioni, in questo caso quella di L_o e di L_{calc}, giocando sui parametri incogniti a_0, b_0, N_c e w ai quali sono stati assegnati dei bound.
Come parametri noti sono stati scelti $t = 35$ µm e g = 0.2 mm perché rappresentano valori standard nella fabbricazione di un circuito stampato PCB, f_{ideal} = 13.56 MHz, p = 1.8 [10], $C_{plT} = 56$ pF [10] e C_{ICT} = 45 pF [11] perché classico valore della capacità di ingresso di molti transceiver, ad esempio l'AS3955 di AMS [11].
Questi sono i bound scelti per i quattro valori incogniti:

$$10 \text{ mm} \leq a_0 \leq 25 \text{ mm}$$
$$8 \text{ mm} \leq b_0 \leq 20 \text{ mm}$$
$$1 \leq N_c \leq 7$$
$$0.2 \text{ mm} \leq w \leq 0.6 \text{ mm}$$

Il codice Matlab è riportato di seguito:

```
%%
% This script calculates the geometrical parameters for a NFC PCB antenna
% of class 6.

% Author: Luca Urbinati

%% ------------------------------------------------------------
The output is x. This is how the output has to be read.
% x = x1, x2, x3, x4, x5, x6, x7, x8, x9, x10.
% x = a, b, N, w, L0, L, Err, A, Aeff, Ctot.
```

```
%%
clc
clear all
close all

u0 = 4*pi*10^(-7);

t = 35e-6;  %Thikness of the track
g = 0.2e-3;  %Gap between tracks
p = 1.8;  %Turn exponent 1.7 < p < 1.8

f_ideal = 13.56e6;  %Objective frequency
%f that takes into account the tolerances:
%f_ideal = 14.5e6;  According to NXP Antenna Design Guide

% Class 6 standard limits
a0 = 25e-3;
b0 = 20e-3;

% Bounds for the track width
wmin = 0.2e-3;
wmax = 0.6e-3;

% Starting point for w
wstart = 0.4e-3;

%IC's threshold resonance capacitance (or input capacitance)
Ctot = 45e-12;

% Connection capacitance
Ccon = 2e-12;  %0,5..2

% Antenna capacitance
Ct = 4e-12;  %2..4
Cbr = 5e-12;  %1..5
Cin = 0;
Cc = Ct + Cbr + Cin;

% Capacitance of the equivalent circuit of the tag = IC + connection + antenna,
considering the worst case (higher parasitic capacitances)
Ctot = Ctot + Ccon + Cc;

% Objective inductance (inductance we are looking for)
L0 = 1/(Ctot*(2*pi*f_ideal)^2);

%%----------------------------------------------------------
x = zeros(5,10);
```

```
A = zeros(1,5);

% Bounds
%lb = [lb(1),lb(2),lb(3),lb(4)];
lb = [10e-3, 8e-3, 1, wmin];
% ub = [ub(1), ub(2), ub(3), ub(4)];
ub = [a0, b0, 7, wmax];

% Options for fmincon algorithm
options = optimoptions( @fmincon,'OptimalityTolerance',1e-
15,'FunctionTolerance',1e-15,'StepTolerance',1e-15); %'PlotFcns', @optimplotfval);

percentage = [0.8 0.9 1.0 1.1 1.2];

% Starting point for the algorithm
x0 = [a0/2, b0/2, 4, wstart];


for k = 1:5

    %Function that has to be minimized
   fun = @(x) abs(((u0/pi)*((x(1) - x(3)*(g+x(4)))*log((2*(x(1) - x(3)*(g+x(4)))*(x(2) -
x(3)*(g+x(4))))/((2*(t+x(4))/pi)*((x(1) - x(3)*(g+x(4))) +sqrt((x(1) - x(3)*(g+x(4)))*(x(1)
- x(3)*(g+x(4))) +(x(2) - x(3)*(g+x(4)))*(x(2) - x(3)*(g+x(4))))))) +(x(2) -
x(3)*(g+x(4)))*log((2*(x(1) - x(3)*(g+x(4)))*(x(2) -
x(3)*(g+x(4))))/((2*(t+x(4))/pi)*((x(2) - x(3)*(g+x(4))) +sqrt((x(1) - x(3)*(g+x(4)))*(x(1)
- x(3)*(g+x(4))) +(x(2) - x(3)*(g+x(4)))*(x(2) - x(3)*(g+x(4))))))-(2*((x(1) -
x(3)*(g+x(4))) +(x(2) - x(3)*(g+x(4)))-sqrt((x(1) - x(3)*(g+x(4)))*(x(1) -
x(3)*(g+x(4))) +(x(2) - x(3)*(g+x(4)))*(x(2) - x(3)*(g+x(4))))) +((x(1) -
x(3)*(g+x(4))) +(x(2) - x(3)*(g+x(4))))/4)*(x(3))^p) - percentage(k)*L0);

    %fmincon algorithm
   [x(k,1:4), error, ~, log_file] = fmincon(fun,x0,[],[],[],[],lb,ub,[],options);
    a = x(k,1);
    b = x(k,2);
    N = round(x(k,3));
    x(k,3) = N;
    w = x(k,4);
    x(k,5) = percentage(k)*L0;
    x(k,6) = error+x(k,5);  %Antenna inductance L
    x(k,7) = error;  %Error

    a_average = a-N*(g+w);  %[m]
    b_average = b-N*(g+w);  %[m]
    A(k) = a_average*b_average;  %Average antenna area [m^2]
    x(k,8) = A(k);
    x(k,9) = A(k)*N;  %Active area [m^2]
```

```
    x(k,10) = Ctot;

end
```

2.3.5. I risultati

L'output di Matlab è visibile in Figura 13. Nella prima riga ci sono i risultati della differenza tra L_{calc} e $0.8 \cdot L_0$, nella seconda quelli tra L_{calc} e $0.9 \cdot L_0$ e così via fino a L_{calc} e $1.2 \cdot L_0$. L'antenna scelta è la prima perché i suoi parametri sono quelli che permettono di ottenere l'errore minore nella suddetta differenza.

x

5x10 double

	1 a	2 b	3 N	4 w	5 L0	6 L	7 Err	8 A	9 Aeff	10 Ctot
1	0.0241	0.0192	7	2.2604e-04	1.9680e-06	1.9680e-06	2.6200e-14	3.4235e-04	0.0024	5.6000e-11
2	0.0248	0.0198	7	2.0447e-04	2.2140e-06	2.2140e-06	5.9529e-14	3.7297e-04	0.0026	5.6000e-11
3	0.0250	0.0200	7	2.0001e-04	2.4600e-06	2.6483e-06	1.8830e-07	3.8182e-04	0.0027	5.6000e-11
4	0.0250	0.0200	7	2.0005e-04	2.7060e-06	3.1408e-06	4.3481e-07	3.8174e-04	0.0027	5.6000e-11
5	0.0250	0.0200	7	2.0001e-04	2.9520e-06	3.6323e-06	6.8030e-07	3.8182e-04	0.0027	5.6000e-11

Figura 13 Output dell'algoritmo implementato in Matlab.

Il layout dell'antenna migliore è stato implementato in KiCad e successivamente stampato per mezzo di una milling machine per PCB.

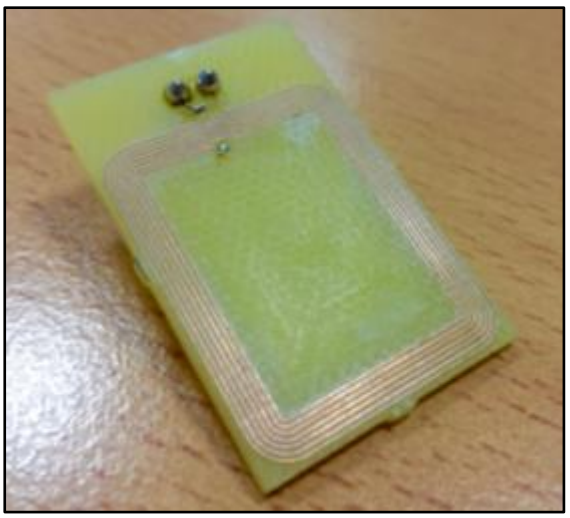


Figura 14 e 15 A sinistra impronta dell'antenna NFC di classe 6 disegnata con KiCad, a destra il prototipo realizzato su PCB.

Questa antenna richiede un tuning in frequenza mediante l'apporto di una capacità in parallelo esterna perché la frequenza di risonanza del prototipo non è esattamente 13.56 MHz. Ciò è dovuto al processo di fabbricazione, in quanto è stata usata una milling machine che asporta rame e substrato, andando a

cambiare quelle che sono le proprietà della scheda rispetto a una macchina per fotolitografia o laser.
Per capire quale valore di capacità inserire, sarebbe necessario misurare l'attuale frequenza di risonanza. Purtroppo, per mancanza di strumentazione a radiofrequenza, come analizzatore di impedenza, analizzatore di reti o LCR meter, si è scelto di continuare con un'antenna NFC già pronta prodotta dalla NXP chiamata *NTAG I2C plus Antenna Board* che fa parte dell'*Explorer Kit*.

Figura 16 *NTAG I2C plus Antenna Board* [12].

É comunque possibile verificare i risultati ottenuti usando il tool *eDesignSuite* [8] di STMicroelectronics. Questo strumento online contiene una parte dedicata al design di antenne NFC che, in seguito all'inserimento dei parametri relativi alla geometria dell'antenna, al conduttore e al substrato, stima l'induttanza dell'antenna tenendo conto delle capacità parassite. Nel tool sono stati inseriti i valori dell'antenna migliore generata da Matlab.

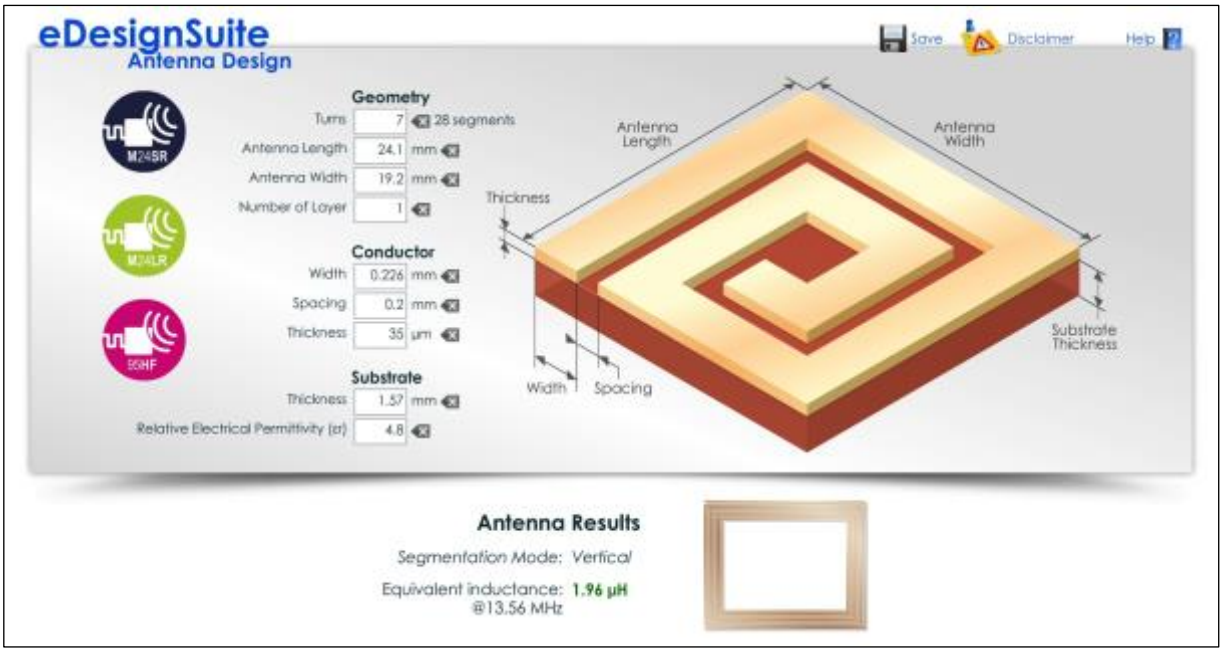


Figura 17 Output del tool online *eDesignSuite* di STMicroelectronics.

Confrontando le induttanze ottenute dai due software, si evince un match completo pari a 1.96 µH. Anche se questo non garantisce che il tuning sia ottimale, consente però di capire che non sono stati compiuti errori nell'implementazione dell'algoritmo, il quale ha restituito risultati ragionevoli.

3. I componenti e lo schema elettrico

Per questo progetto sono stati utilizzati prevalentemente gli stessi componenti che fanno parte del nodo sensore. Tra questi il microcontrollore $PIC16LF1824$ [18], il timer $TPL5010$ [19] e il sensore di temperatura $TMP112$ [20]. In aggiunta a questi sono stati impiegati un transceiver NFC, un voltage monitor e tre switch analogici. Nel presente paragrafo si approfondiscono i loro comportamenti e si motiva la loro scelta.

3.1. Transceiver NFC: NT3H2211

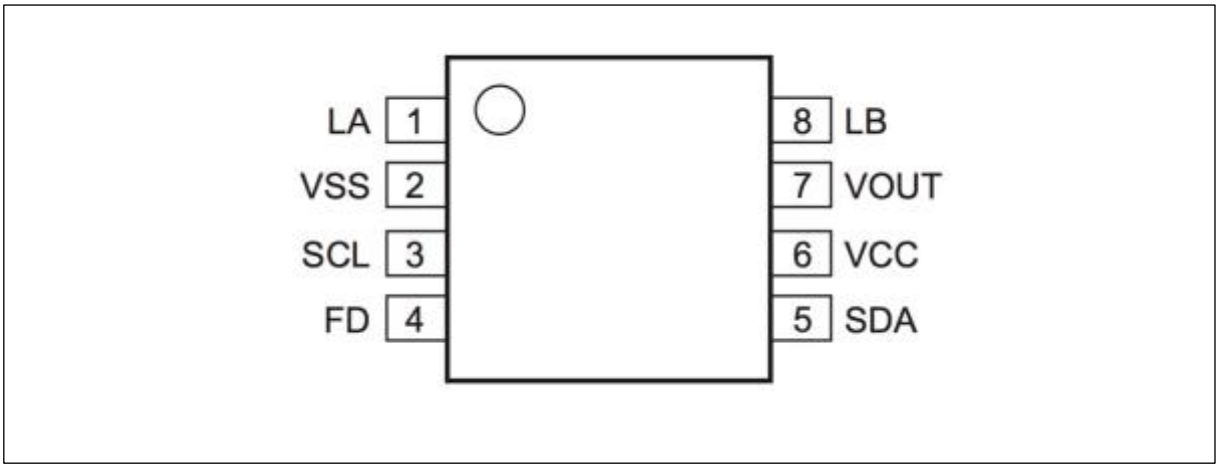


Figura 18 Pin configuration dell'$NTAG\ I2C\ plus$ [13].

L'$NT3H2211$, o $NTAG\ I2C\ plus\ 2k$, è un transceiver NFC prodotto dalla NXP [13]. Questo IC è un dispositivo NFC passivo e dinamico. Infatti è dotato di un'interfaccia RF e di una seriale I^2C con la quale è in grado di comunicare rispettivamente con un dispositivo attivo e con una MCU. Al suo interno ha una memoria EEPROM da 1912 Bytes (circa 2 KB) e un buffer SRAM da 64 Bytes per la modalità di comunicazione pass-through tra MCU e *reader* esterno. Inoltre possiede una logica di Power Management per effettuare *energy harvesting* dall'antenna che va collegata ai terminali LA e LB. Dal pin $VOUT$ è possibile generare una tensione massima di 3.3 V da sfruttare per alimentare i circuiti a valle, come il μC. Inoltre, è disponibile un pin di $Field\ Detection$ (FD) attivo basso, che permane in questo stato finché viene fornita energia all'$NTAG$ dall'interfaccia NFC. FD può essere usato come segnale per generare un interrupt e svegliare il *microcontrollore* dallo sleep quando l' $NTAG$ percepisce di essere alimentato da un campo esterno. In Figura 19 è schematizzato l'intero IC i cui blocchi fondamentali erano stati descritti nel

Paragrafo 2.2, mentre in Figura 20 è riportato un esempio applicativo del *transceiver* con uno smartphone e un microcontrollore.

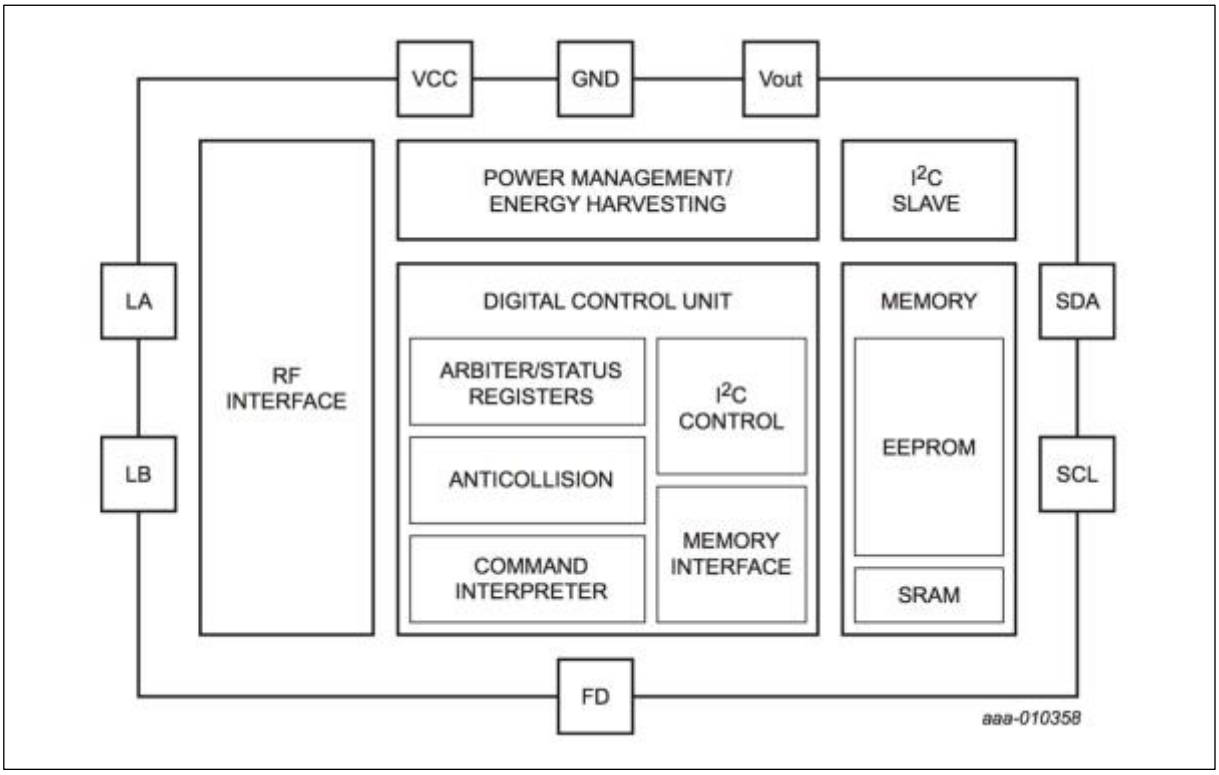


Figura 19 Blocchi interni dell'*NTAG I2C plus* [13].

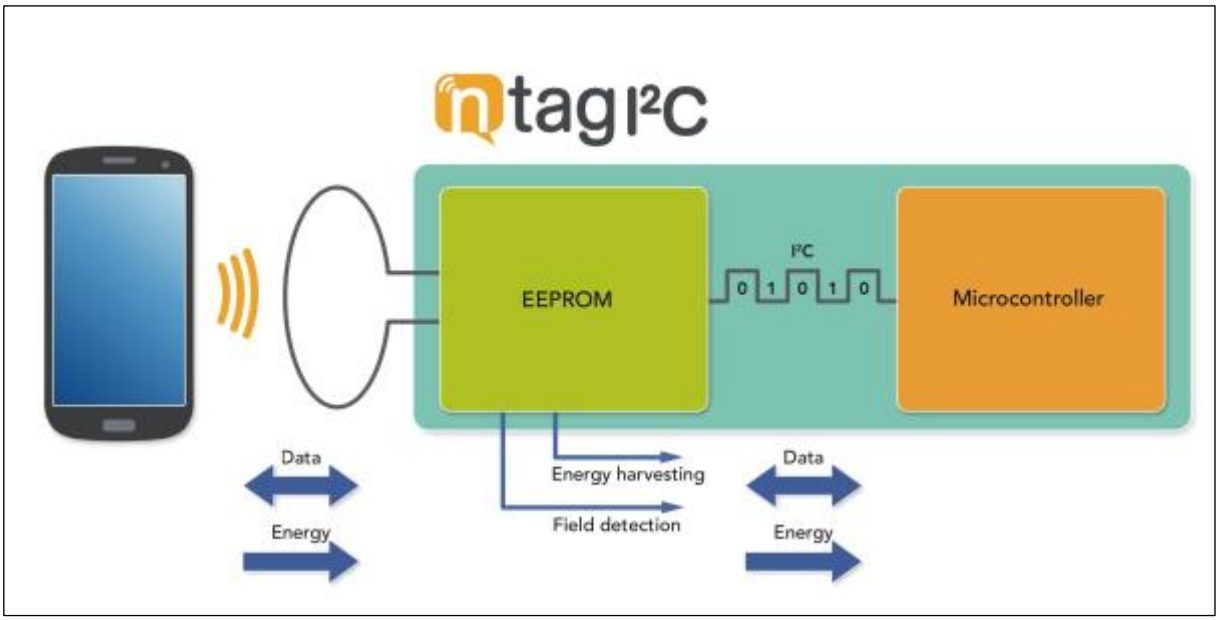


Figura 20 Esempio di utilizzo del *transceiver NTAG I2C plus* [14].

L'*NTAG* ha due modalità di trasferimento dei dati:

- passiva/statica: nata per permettere la lettura via RF dei dati salvati nella memoria EEPROM dell' $NTAG$ quando l'MCU è spento, oppure per leggere/scrivere i dati nella memoria EEPROM dell' $NTAG$ via I^2C quando nessun campo sta alimentando il *transceiver*.
- pass-through: i dati scorrono dall'interfaccia NFC attraverso la SRAM fino al bus I^2C o all'opposto, preservando i limitati cicli di scrittura della EEPROM. In sostanza la funzione svolta dall' $NTAG$ è equivalente a quella di un modem.

Per motivi di tempo è stata implementata solo la prima modalità, ma con la presenza del campo. Infatti quando un dispositivo attivo fornisce potenza all'intero sistema embedded, il μC scrive i dati di temperatura acquisiti nel tempo nella EEPROM dell'$NTAG$. In questo modo viene preservata la carica della batteria.

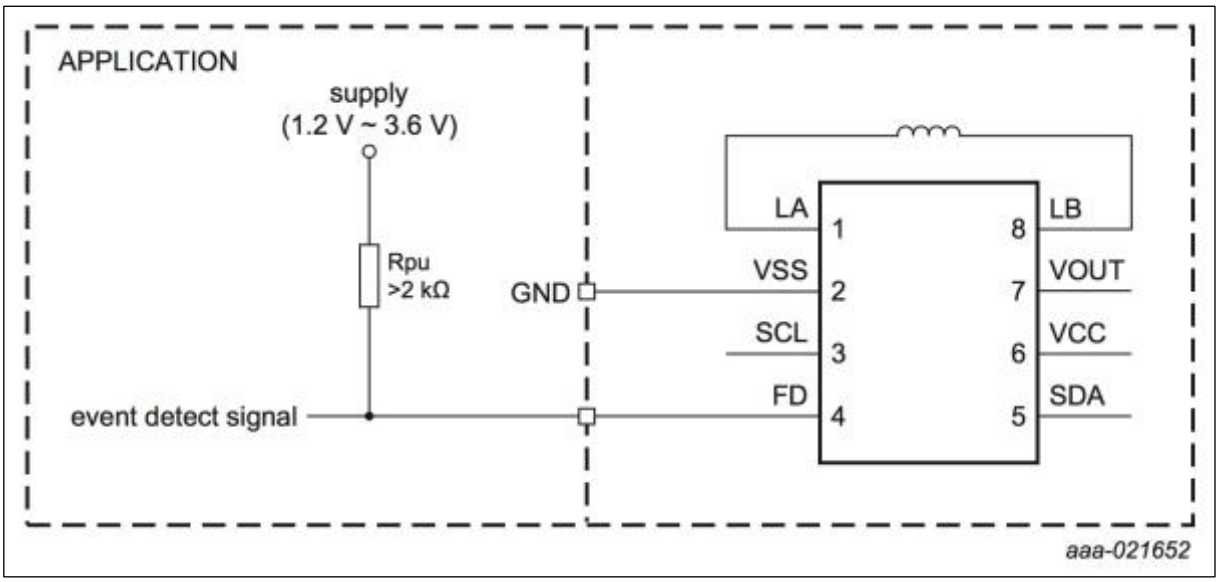


Figura 21 Collegamento di una resistenza di pull-up per il funzionamento del pin *FD* [13].

La scelta del transceiver è ricaduta sull'$NTAG\ I2C\ plus$ per diverse ragioni:

- NXP è attualmente una delle più grandi fornitrici di chip NFC, insieme a STMicroelectronics, Texas Instruments, Broadcom, Polaric, AMS;
- NXP rende disponibili numerose application notes che descrivono in modo chiaro come usare i loro prodotti;
- i transceiver NXP possiedono l'interfaccia I^2C, richiedendo minimi sforzi a livello di firmware ai fini di questo progetto;
- i transceiver NXP implementano la modalità di comunicazione pass-through, vantaggiosa per esperimenti futuri.

3.2. Voltage monitor: NCP303LSN20T1

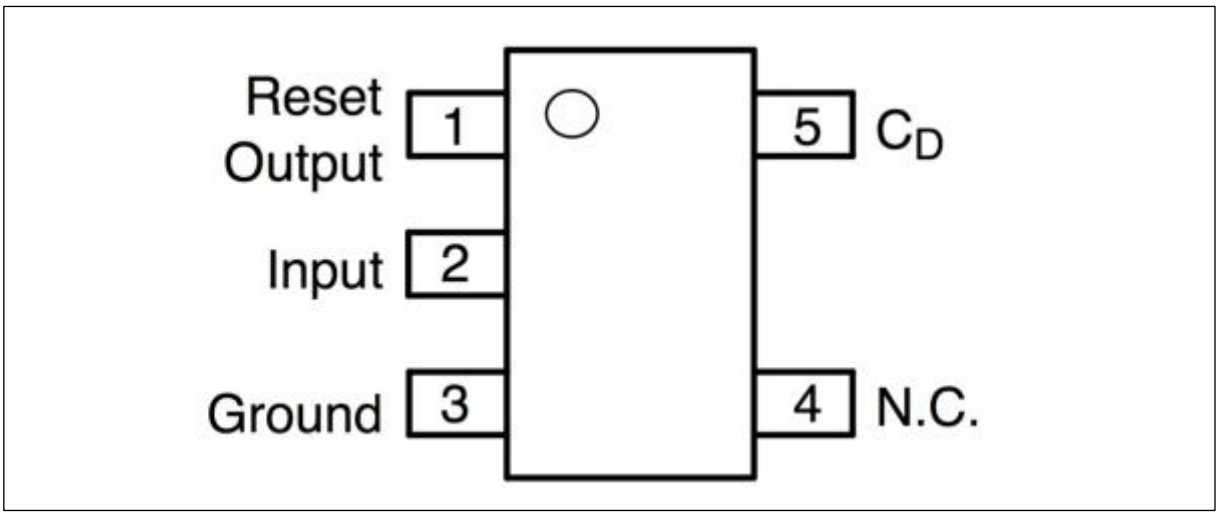


Figura 22 Pin configuration del *voltage monitor* [15].

L'$NCP303LSN20T1$ è fabbricato dalla ON Semiconductor [15]. La funzione della serie NCP303 è di avvertire, attraverso un valore logico basso/alto sul pin $Reset\ Output$, quando il segnale in $Input$ scende sotto una certa soglia. Inoltre è possibile variare l'isteresi con l'aggiunta di opportune resistenze e di regolare il ritardo dopo il quale il pin 1 debba azionarsi tramite il collegamento di un condensatore sul pin 5. L'alimentazione è ricavata direttamente dal segnale di ingresso.
Il particolare *voltage monitor* selezionato ha un reset attivo basso e una soglia a 2 V, con isteresi di 100 mV. Quando il segnale in $Input$, che è la tensione proveniente dall'$NTAG$, supera la soglia di 2.1 V, $Reset\ Output$ viene alzato dalla resistenza di pull-up (necessaria per la serie NCP303) e rimane a un valore logico alto fino a che l'ingresso non scende sotto gli 1.9 V. Altrimenti $Reset\ Output$ vale 0 V. La sua uscita è il segnale di controllo dello switch analogico $AS11P2TLR$ che gestisce la commutazione della batteria in favore dell'alimentazione proveniente dal *transceiver* NFC. Dunque il *voltage monitor* sancisce quando è opportuno che si cambi la sorgente di alimentazione al μC garantendo che si mantenga un adeguato livello di tensione per il suo funzionamento.

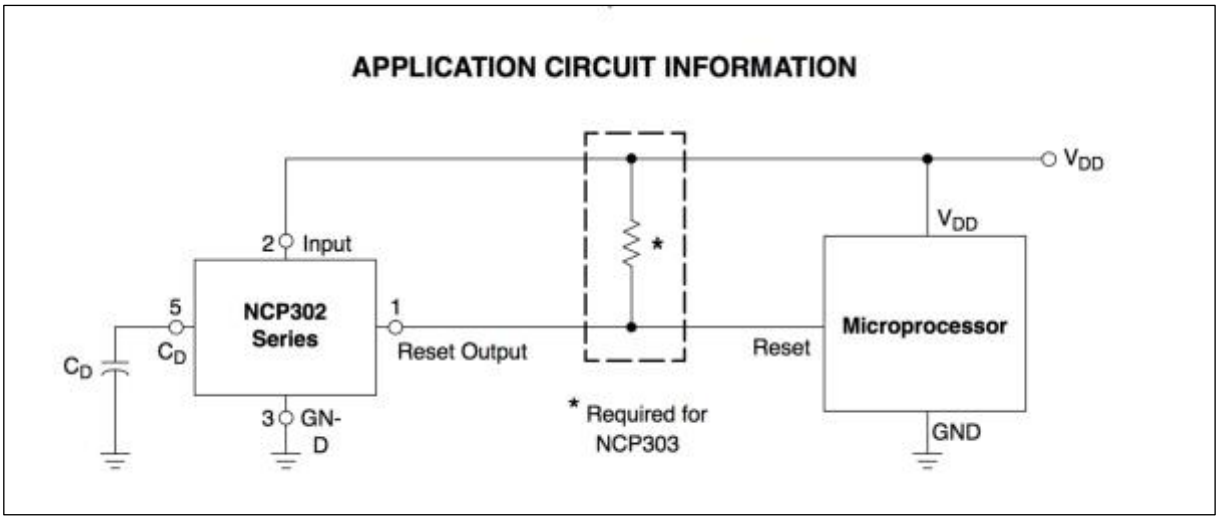


Figura 23 Esempio di applicazione del *voltage monitor* per effettuare un reset del microcontrollore [15].

Il motivo che ha indotto il suo impiego è il consumo di corrente inferiore al μA (circa 0.48 μA), mentre la soglia a 2 V è richiesta dal $PIC16LF1824$ perché ammette una tensione di alimentazione che sia almeno pari a 1.8 V.

3.3. Analog switch: AS11P2TLR

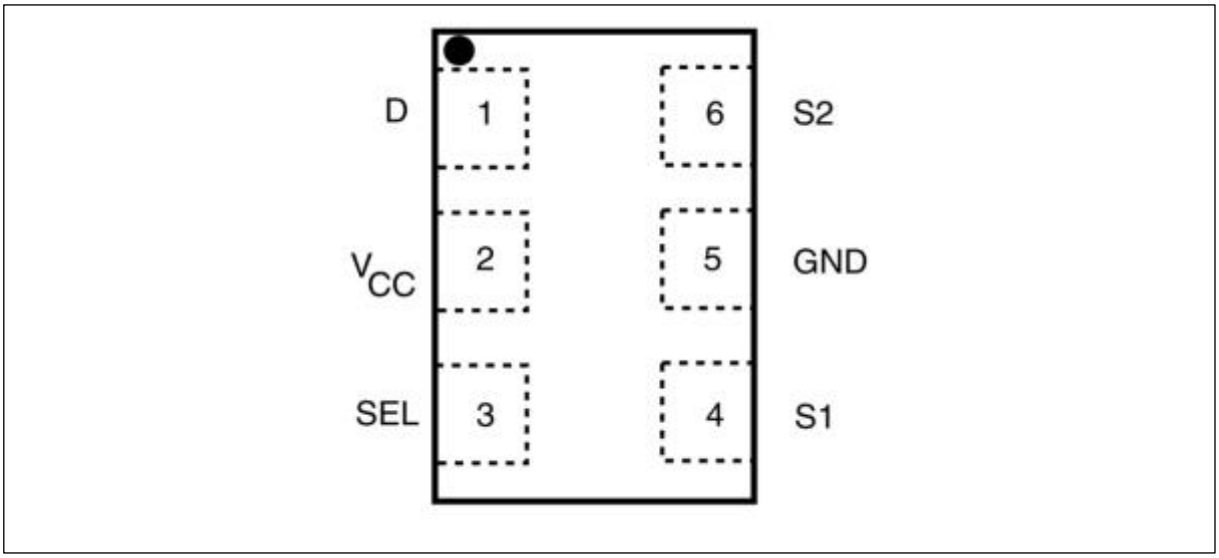


Figura 24 Pin configuration dello *switch analogico* [16].

L'$AS11P2TLR$ della STMicroelectronics [16] è uno switch analogico la cui tabella di verità è riportata in Tabella 2. Quindi il canale comune D viene connesso a $S1$ se SEL è sopra la soglia di '1' logico (V_{IH} = 1.4 V per V_{cc} = 3.3 V) e viceversa (V_{IL} = 0.6 V per V_{cc} = 3.3 V). Nel caso in cui non venga fornita

alimentazione al componente, il canale comune rimane in tristate o alta impedenza.

SEL	SWITCH S1	SWITCH S2
HIGH	ON	OFF (HIGH IMPEDENCE)
LOW	OFF (HIGH IMPEDENCE)	ON

Tabella 2 Tabella di verità dello *switch analogico*.

Nel progetto vengono adoperati tre *switch analogici* come questo. Uno viene comandato dal segnale $Reset\ Output$ del *voltage monitor* e serve ad effettuare la commutazione da batteria al pin $VOUT$ del *transceiver* e viceversa; gli altri due, invece, sono comandati dal pin SEL (o $RC3$) del $PIC16LF1824$ e assolvono la funzione di collegare le linee di trasmissione del bus seriale SDA e SCL del μC al *sensore di temperatura* e all'$NTAG$ nella modalità operativa #1 del firmware (vedi Paragrafo 3.4).

É importante sottolineare che i valori delle tensioni di ingresso devono rispettare i seguenti valori per un corretto comportamento:

Symbol	Parameter	Value	Unit
V_{cc}	Supply voltage	1.65 to 4.5	V
V_{S1}, V_{S2}	Input voltage	0 to V_{cc}	V
V_{SEL}	Control input voltage	0 to 4.5	V

Tabella 3 Valori di tensione consentiti per gli ingressi dello *switch analogico* [16].

Per questa ragione nello schematico di Paragrafo 3.4 si osserva un diodo Zener [17] da 3.3V collegato al segnale $VOUT$ del *transceiver* (chiamato V_{out_ntag} nello schematico). Infatti risulta necessario limitare la tensione prodotta dall'$NTAG$ perché si è riscontrata essere più alta del valore massimo di 3.3 V dichiarato, causando comportamenti anomali dello *switch*.

La ragione dell'utilizzo dell'$AS11P2TLR$ è in gran parte legata al suo profilo di energia ultra-low power. Infatti consuma circa 20 nA di corrente. In aggiunta esso possiede un *break-before-make time delay* di 8 ns, ossia la commutazione tra i canali avviene molto velocemente.

3.4. Lo schema elettrico e il funzionamento del circuito

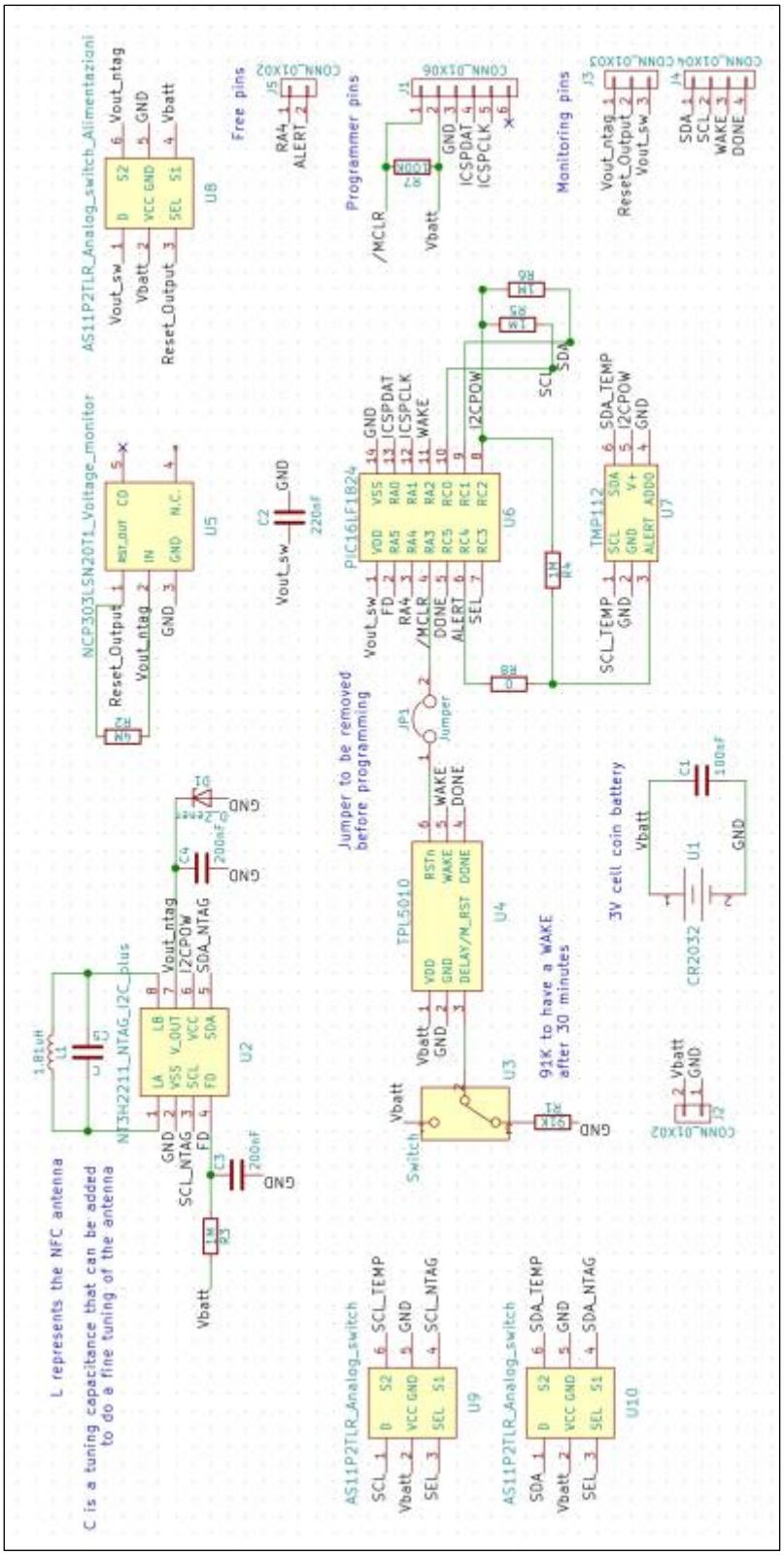


Figura 25 Schematico del circuito realizzato con il software KiCad.

Lo schematico del circuito realizzato con KiCad è proposto in Figura 25. Si è cercato di mantenere la struttura dei collegamenti simile a quella del progetto del nodo sensore [2] per permettere di comprendere istantaneamente la presenza dei nuovi componenti. Qui viene presentata una completa descrizione del funzionamento del circuito di interfaccia, motivando l'aggiunta di alcuni componenti passivi. Figura 1 e Figura 2 possono tornare utili al lettore.

- **AVVIO**. All'avvio il $PIC16LF1824$ alza e abbassa il segnale $DONE$, per consentire al timer $TPL5010$ di leggere il valore della resistenza $R2$ e programmare l'intervallo dei suoi impulsi di $WAKE$; impone un valore logico basso al segnale SEL, per comandare i due switch $AS11P2TLR$ in modo che il loro canale comune diretto verso al μC sia connesso al bus I^2C che porta al sensore di temperatura $TMP112$; tiene a '0' il segnale $I2CPOW$, per mantenere spento il *sensore*. Successivamente entra in sleep. La modalità di partenza del firmware è la #0.
- **ACQUISIZIONE TEMPERATURA**. Quando un interrupt esterno, quello di $WAKE$, arriva al pin 11 del *microcontrollore*, viene eseguita una lettura del *sensore*. Così il segnale $I2CPOW$ viene portato alto per fornire alimentazione al $TMP112$ e alle resistenze di pull-up $R6$ e $R7$ delle linee SDA e SCL. Segue la fase di acquisizione e salvataggio dei due byte restituiti dal *sensore* nella memoria EEPROM interna al PIC. Nel frattempo il *timer* legge nuovamente il valore della resistenza $R2$ per programmare i prossimi impulsi di $WAKE$, che avverranno ogni 1745 secondi circa [2]. Infine il μC invia un segnale di $DONE$ al *timer*, per indicare che l'interrupt generato è andato a buon fine, e spegne il $TMP112$ per risparmiare energia. Dopodiché entra in sleep.
- **IN PRESENZA DI CAMPO RF**. Quando un dispositivo NFC attivo, ad esempio lo smartphone, si avvicina all'antenna dell' $NTAG\ I2C\ plus$, questo invia un comando NFC di $READ$. Dopo aver ricevuto l'acknowledge dal *transceiver*, mantiene il suo campo magnetico non modulato e fornisce energia all'intero sistema. FD compie una transizione alto-basso e il *transceiver* fornisce una tensione V_{out_ntag} diversa da 0 V. A questo punto il segnale $Reset\_Output$ del *voltage monitor* $NCP303LSN20T1$ $(U5)$ si porta alto se V_{out_ntag} supera i 2.1 V. Di conseguenza lo *switch analogico* $(U8)$ esegue la commutazione per alimentare il μC con V_{out_ntag} e non più dalla batteria ($U1$). Contemporaneamente, il PIC percepisce l'interrupt negativo sul pin 2, si sveglia e cambia modalità di funzionamento (da #0 a #1 o viceversa).

Quando il campo magnetico si allontana e V_{out_ntag} scende sotto gli 1.9 V, $Reset\_Output$ torna basso e lo *switch* ricollega la batteria a V_{cc} del *microcontrollore*. FD si alza e non genera nessun interrupt.

- **MODALITA' #1: LETTURA VECCHI DATI E TRASFERIMENTO NUOVI DATI**.
  Quando il PIC entra in modalità #1, il dispositivo NFC attivo ha già letto i vecchi dati non aggiornati contenuti nella EEPROM dell' $NTAG$. Il *microcontrollore* esegue prima una lettura del *sensore di temperatura*. Poi porta alto il segnale SEL per direzionare SDA e SCL verso il *transceiver* e inizia a trasferire i nuovi dati dalla propria memoria a quella dell'$NTAG$. Solo i dati che non sono stati mai inviati vengono trasferiti. Prima di tornare in sleep, SEL torna al livello logico basso.
- **MODALITA' #0: LETTURA NUOVI DATI E NESSUN TRASFERIMENTO DATI**.
  Quando il PIC entra in modalità #0, il dispositivo NFC attivo ha già letto i nuovi dati precedentemente scritti e aggiornati contenuti nella EEPROM dell' $NTAG$. Il μC non esegue nessun trasferimento e torna immediatamente in sleep.

  La decisione di utilizzare due modalità di funzionamento non è obbligatoria. Tuttavia, poiché non è possibile evitare che lo smartphone invii il comando NFC di $READ$ ad ogni avvicinamento al *target*, questo avrebbe sempre causato la lettura di dati in ritardo, cioè si sarebbero letti i dati trasferiti nella sessione precedente. L'ideale sarebbe quello di creare un'applicazione per smartphone che dia la possibilità di generare costantemente un campo non modulato e di inviare il comando di $READ$ solo su richiesta dell'utente. Oppure si può evitare questo inconveniente configurando l'$NTAG$ in modalità pass-through.

- Lo *switch* ($U3$) opera un reset manuale per il $TPL5010$ che a sua volta provvederà a resettare il *microcontrollore* portando a livello basso il pin $RSTn$.
- I condensatori $C1$ e $C2$ sono di bypass e riducono eventuali disturbi sull'alimentazione degli integrati.
- I condensatori $C3$ e $C4$ migliorano i transitori spuri rispettivamente di FD e di V_{out_ntag}.
- Il condensatore $C5$ va scelto opportunamente per migliorare il tuning dell'antenna NFC.

- L'induttore $L1$ che rappresenta l'antenna ha valore 1.81 μH, applicando l'algoritmo del Paragrafo 2.3.4 con C_{ICT} = 50 pF, capacità di risonanza di ingresso dell' $NTAG\ I2C\ plus$. Il corrispettivo valore di induttanza prodotto dal tool di STMicroelectronics con gli stessi parametri geometrici dell'antenna restituiti da Matlab è 1.83 μH.
- Il resistore $R1$ è quello che fornisce la temporizzazione al *timer*.
- Il resistore $R2$ serve per alzare $Reset\ Output$ quando V_{out_ntag} supera i 2.1 V e poi rimane maggiore di 1.9 V.
- Il resistore $R3$ è il pull-up e garantisce le funzionalità del pin FD.
- I resistori $R4$, $R5$, $R6$ sono dei pull-up necessari per la comunicazione seriale. Sono controllati dal pin $I2CPOW$ del *microcontrollore* e non direttamente dall'alimentazione.
- Il resistore $R7$ è un pull-up per la linea $MCLR$ dedicata al reset del μC.
- Il resistore $R8$ consente di attivare la funzionalità di $ALERT$ del *timer*. É É
  stato predisposto il collegamento, ma si è lasciato un cortocircuito permettendo di utilizzarla solo se lo si ritenesse opportuno.
- Il diodo Zener $D1$ diodo limita la tensione prodotta dall'$NTAG$ perché si è riscontrata essere più alta del valore massimo di 3.3 V dichiarato, causando anomalie nel blocco del circuito dedicato alla commutazione della batteria.
- Il connettore $J1$ è il connettore di programmazione, a cui si collega il $PicKit3$ per programmare il *microcontrollore*. É importante che durante la fase di programmazione/debugging, il jumper $JP1$ sia aperto, per evitare che le alte tensioni applicate possano danneggiare il *timer*. Quando $JP1$ è aperto si perde però la funzionalità di reset manuale del PIC.
- Il connettore $J2$ consente di alimentare il circuito da una diversa fonte energetica.
- I connettori $J3$ e $J4$ servono per fornire all'esterno del circuito i principali segnali per essere controllati mediante oscilloscopio o altra strumentazione.
- Il connettore $J5$ rende disponibili i pin inutilizzati del μC, consentendo di essere sfruttati in caso di necessità.
- Il componente $U1$ è la batteria CR2032, fonte di alimentazione principale del circuito.

Concludendo, tutti i resistori sono stati scelti da 1 MΩ, o anche più ove possibile, per ridurre le correnti di leakage di tutti i percorsi conduttivi verso massa e limitare così i consumi del circuito.

4. Il firmware

Per la scrittura del firmware del PIC si è usufruito degli strumenti forniti dalla Microchip, produttrice del *microcontrollore* utilizzato. In particolare la programmazione è avvenuta all'interno dell'ambiente di sviluppo $MPLAB\ X$ che, in aggiunta al compilatore $XC8$ e a $MPLAB\ Code\ Configurator$ (MCC), ha permesso sia di scrivere codice in linguaggio C che di generare quello per la configurazione dei registri.

4.1. I registri di configurazione

Tutti i registri sono configurati come mostrato nelle tabelle che seguono.

System Module

Name		Bit 13/6	Bit 12/5	Bit 11/4	Bit 10/3	Bit 9/2	Bit 8/1	Bit 7/0
CONFIG1	—	FCMEN	IESO	CLKOUTEN	BOREN<1:0>		CPD	CP
	—	0	0	1	0	0	1	1
	—	MCLRE	PWRTE	WDTE<1:0>		FOSC<2:0>		
	—	1	1	0	0	1	0	0
CONFIG2	—	LVP	DEBUG	—	BORV	STVREN	PLLEN	—
	—	0	0	0	1	1	0	0
	—	—	—	Reserved	—	—	WRT<1:0>	
	—	0	0	0	0	0	1	1

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
OSCCON	SPLLEN	IRCF<3:0>				—	SCS<1:0>	
	0	0	0	0	0	0	1	0
OSCTUNE	—	—	TUN<5:0>					
	0	0	0	0	0	0	0	0
WDT	—	—	WDTPS<4:0>					SWDTEN
	0	0	0	1	0	1	0	0

Tabella 4 Configurazione dei bit del System Module.

I registri di configurazione $CONFIGURATION\ WORD\ 1$ $(CONFIG1)$ e $CONFIGURATION\ WORD\ 2$ $(CONFIG2)$ permettono di selezionare il clock interno, disabilitare il watchdog timer, attivare la funzionalità di reset collegata al pin $MCLR$ e selezionare la modalità di programmazione "high voltage". Tutte le altre funzioni sono rimaste disabilitate per ridurre i consumi al minimo.

Il registro $OSCILLATOR\ CONTROL$ $(OSCCON)$ contiene le informazioni relative alla frequenza dell'oscillatore nei bit $IRCF < 3:0 >$, la quale si è

impostata a 31 KHz perché è stato dimostrato [2] essere quella meno energivora. Non è stato eseguito nessun tuning di frequenza, perciò il registro $OSCILLATOR\ TUNING$ ($OSCTUNE$) è impostato a 0x00. Anche il $WATCHDOG\ TIMER$ (WDT) non è abilitato per motivi energetici [2].

Pin Module

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ANSELA	—	—	—	ANSA4	—	ANSA2	ANSA1	ANSA0
	0	0	0	0	0	0	0	0
ANSELC	—	—	—	—	ANSC3	ANSC2	ANSC1	ANSC0
	0	0	0	0	0	0	0	0
APFCON0	RXDTSEL	SDOSEL	SSSEL	—	T1GSEL	TXCKSEL	—	—
	0	0	0	0	0	0	0	0
APFCON1	—	—	—	—	P1DSEL	P1CSEL	P2BSEL	CCP2SEL
	0	0	0	0	0	0	0	0
IOCAF	—	—	IOCAF5	IOCAF4	IOCAF3	IOCAF2	IOCAF1	IOCAF0
	0	0	0	0	0	0	0	0
IOCAN	—	—	IOCAN5	IOCAN4	IOCAN3	IOCAN2	IOCAN1	IOCAN0
	0	0	1	0	0	0	0	0
IOCAP	—	—	IOCAP5	IOCAP4	IOCAP3	IOCAP2	IOCAP1	IOCAP0
	0	0	0	0	0	0	0	0
LATA	—	—	LATA5	LATA4	—	LATA2	LATA1	LATA0
	0	0	0	0	0	0	0	0
LATC	—	—	LATC5	LATC4	LATC3	LATC2	LATC1	LATC0
	0	0	1	0	0	0	0	0
TRISA	—	—	TRISA5	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0
	0	0	1	1	1	1	1	1
TRISC	—	—	TRISC5	TRISC4	TRISC3	TRISC2	TRISC1	TRISC0
	0	0	0	1	0	0	1	1
WPUA	—	—	WPUA5	WPUA4	WPUA3	WPUA2	WPUA1	WPUA0
	0	0	0	1	1	0	1	1
WPUC	WPUC7	WPUC6	WPUC5	WPUC4	WPUC3	WPUC2	WPUC1	WPUC0
	0	0	0	1	0	0	0	0

Tabella 5 Configurazione dei bit del Pin Module.

Segue l'impostazione dei pin:

- con $ANSEL(A-C)$ si fissa se essi sono digitali ('0') o analogici ('1'),
- con $APFCON0$ e $APFCON1$ si sceglie una delle varie funzionalità associate al pin,
- con $LAT(A-C)$ possono essere inizializzati con un valore logico basso ('0') o alto ('1'),
- con $TRIS(A-C)$ si stabilisce se un pin è un output ('0') o un input ('1'),
- con $WPU(A-C)$ si attivano o meno i weak pull-up interne.

É bene evidenziare che i pin non utilizzati hanno i weak pull-up abilitati per non creare percorsi conduttivi verso massa.

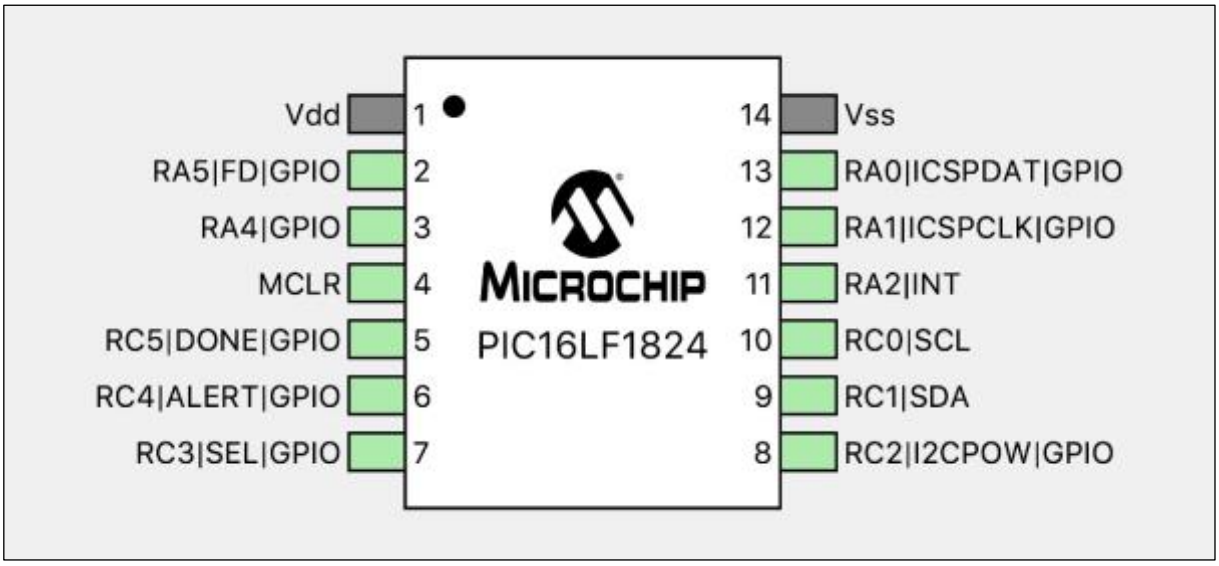


Figura 26 Package TSSOP14 del $PIC16LF1824$ in $MPLAB\ X$.

Package: TSSOP14		Pin No:	13	12	11	4	3	2	10	9	8	7	6	5
			Port A ▾						Port C ▾					
Module	**Function**	**Direction**	**0**	**1**	**2**	**3**	**4**	**5**	**0**	**1**	**2**	**3**	**4**	**5**
EXT_INT	INT	input			🔒									
MSSP ▾	SCL	in/out							🔒					
	SDA	in/out								🔒				
OSC	CLKOUT	output					🔓							
Pin Module ▾	GPIO	input	🔒	🔒	🔓	🔓	🔒	🔒	🔓	🔓	🔓	🔓	🔒	🔓
	GPIO	output	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔒	🔒	🔓	🔒
RESET	MCLR	input				🔒								

Figura 27 Pin manager di $MPLAB\ X$.

MSSP Peripheral

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
SSP1STAT	SMP	CKE	D/A	P	S	R/W	UA	BF
	1	0	0	0	0	0	0	0
SSP1CON1	WCOL	SSPOV	SSPEN	CKP	SSPM<3:0>			
	0	0	1	0	1	0	0	0
SSP1CON3	ACKTIM	PCIE	SCIE	BOEN	SDAHT	SBCDE	AHEN	DHEN
	0	0	0	0	1	0	0	0
SSP1ADD	ADD7	ADD6	ADD5	ADD4	ADD3	ADD2	ADD1	ADD0

	0	0	0	0	0	1	0	0

Tabella 6 Configurazione dei bit della periferica MSSP.

Per la comunicazione con il *sensore di temperatura* e il *transceiver* è stato utilizzato il protocollo I^2C. Il modulo $Master\ Synchronous\ Serial\ Port$ $(MSSP)$ si occupa della gestione dell'interfaccia seriale e può essere configurato per la modalità SPI o I^2C, sia in modalità master sia in quella slave.
Le impostazioni dell'interfaccia seriale si trovano nei registri di Tabella 6.
Le configurazioni più importanti sono quelle di $SSPEN$, che abilita il modulo $MSSP$, e dei bit di $SSPM < 3:0 >$ che sanciscono una modalità I^2C master con frequenza di clock di $f_{CLK} = \frac{F_{OSC}}{4 \cdot (SSP1ADD+1)} = \frac{31000}{4 \cdot (4+1)} = 1.55\ KHz$.

Interrupt Module

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTCON	GIE	PEIE	TMR0IE	INTE	IOCIE	TMR0IF	INTF	IOCIF
	1	0	0	1	1	0	0	0
OPTION_REG	WPUEN	INTEDG	TMR0CS	TMR0SE	PSA	PS<2:0>		
	0	1	0	0	1	0	0	0

Tabella 7 Configurazione dei bit dell'Interrupt Module.

Gli interrupt sono fondamentali per l'efficienza dei microcontrollori. Infatti sono usati per gestire particolari eventi solo nel momento in cui questi accadono, lasciando libera la CPU di eseguire altri task per il resto del tempo. La logica degli interrupt del $PIC16LF1824$ è riportata in Figura 28.
Per gli scopi del progetto sono necessari due interrupt esterni che sveglino il μC dalla sleep mode. Uno è generato dal segnale $WAKE$ del *timer* che è collegato al pin 11 del PIC, al quale può essere abilitato il rilevamento di interrupt con il bit $INTERRUPT\ ENABLE\ (INTE)$ del registro $INTERRUPT\ CONTROL$ $(INTCON)$ a '1'. In particolare con l' $INTERRUPT\ EDGE\ SELECT$ bit $(INTEDG)$ del registro $OPTION\_REG$ attivo, si rilevano solo i fronti positivi. L'altro interrupt proviene dal pin FD dell'$NTAG$ che, collegato al pin 2 del PIC, viene rilevato se l' $INTERRUPT-ON-CHANGE\ ENABLE$ bit $(IOCIE)$ di INTCON è attivo. L'IOC può essere attivato per tutti i pin di PORTA. Con i bit dei registri $IOC\ PORTA\ POSITIVE\ EDGE$ $(IOCAP)$ e $IOC\ PORTA\ NEGATIVE\ EDGE$ $(IOCAN)$ visibili in Tabella 5 si imposta se l'interrupt deve essere considerato solo per fronti di salita, di discesa o entrambi.

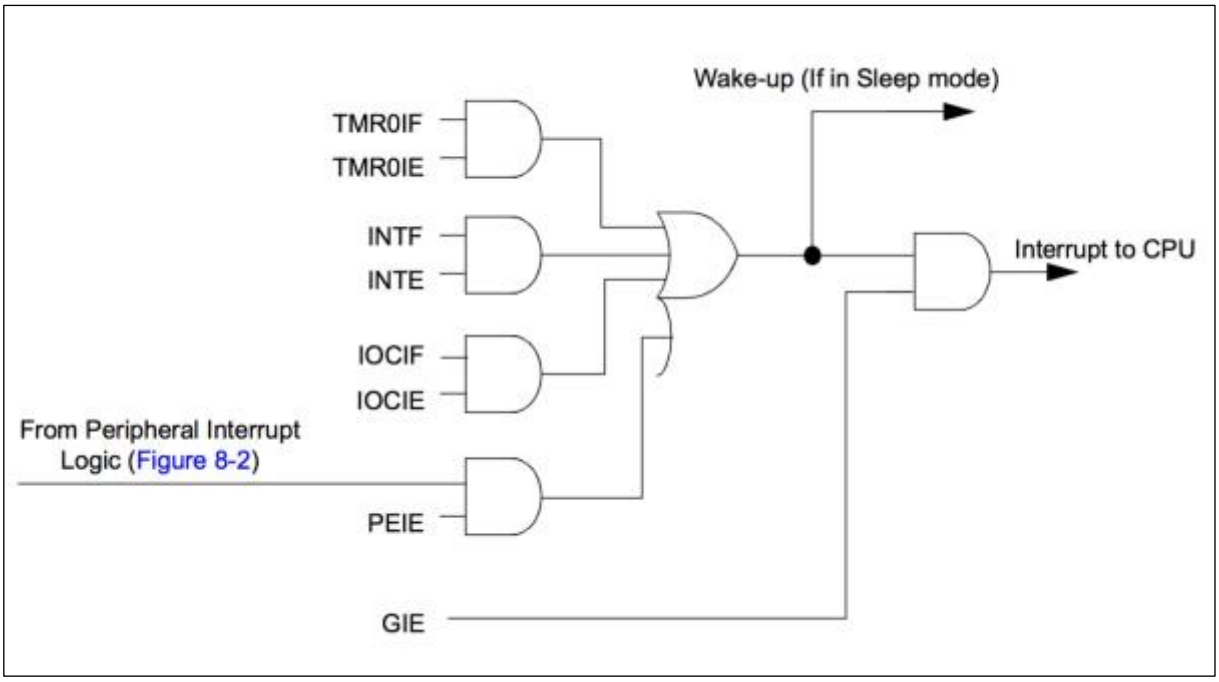


Figura 28 La logica degli interrupt [18].

4.2. Il main e le sue funzioni

Completata questa fase iniziale di configurazione, si procede con la scrittura del *main* e delle funzioni richiamate al suo interno. La spiegazione delle varie parti è illustrata nei seguenti punti:

1. Si includono le librerie di utilità generale per il linguaggio C, quella specifica del microcontrollore $PIC16LF1824$ e quelle generate da MCC. Inoltre vengono dichiarate le variabili globali, tra cui gli indici di lettura/scrittura delle memorie del *microcontrollore* e del *transceiver* e i loro valori iniziali.
2. Il μC viene inizializzato e gli interrupt vengono abilitati. Il segnale di $DONE$ è portato a un valore logico basso per rispettare le richieste del *timer* [19].
3. Questa parte di codice va utilizzata solo una volta per configurare i valori di partenza degli indici di lettura e scrittura della memoria del PIC e l'indice (di scrittura) dell'$NTAG$.
4. Quando arriva il $WAKE$ dal *timer*, il PIC disabilita gli IOC (Interrupt-on-Change) per evitare che un dispositivo NFC attivo possa interrompere questa fase. Poi esegue una lettura dal *sensore di temperatura* attraverso la funzione $readTemp$, manda il segnale di $DONE$ al *timer* e riabilita gli IOC.

5. In presenza di una transizione alto-basso di FD, solo se la modalità è la #1 e non c'è stato nessun falso allarme di FD, viene disabilitato l'interrupt esterno in modo che un eventuale $WAKE$ dal *timer* non ostruisca. Viene eseguita una lettura del *sensore* e in seguito vengono commutati i due *switch analogici* per collegare SDA e SCL all'$NTAG$ comandandoli con il segnale SEL. Successivamente si chiama la funzione $loadBuffer$ per caricare il buffer da 16 Byte poi trasferito al *transceiver*. Infatti nella sua EEPROM non possono essere scritti meno di 16B per volta. Infine, si riportano gli *switch* nel loro stato iniziale, si alza $I2CPOW$ per spegnere il *sensore* e si riattiva l'interrupt esterno.
6. Il *microcontrollore* ripulisce il flag $FD\_positive\_edge$ ed entra in sleep alla fine di entrambe le operazioni di punto 4 o punto 5.

1

```
#include <stdio.h>
#include <stdlib.h>
#include <pic16lf1824.h>
#include "mcc_generated_files/mcc.h"

#define TMP112_addr 0b1001000
uint8_t slave_address = 0b0000000;

unsigned char mode = 0;
unsigned char timer = 0;
unsigned char FD_positive_edge = 0;

uint8_t pic_eeprom_addr_w; // writing index of PIC EEPROM
uint8_t pic_eeprom_addr_r; // reading index of PIC EEPROM
uint8_t ntag_eeprom_addr; // (writing) index of NTAG EEPROM
uint8_t max_ntag_eeprom_addr = 0x37; // maximum NTAG address.

// starting value for pic_eeprom_addr_w
uint8_t start_pic_eeprom_addr_w = 0x03;
// starting value for pic_eeprom_addr_r
uint8_t start_pic_eeprom_addr_r = 0x03;
// starting value for ntag_eeprom_addr
uint8_t start_ntag_eeprom_addr = 0x01;

uint8_t readBuffer[1];
uint8_t writeBuffer[17];
```

2

```
void main(void) {

    // Initialization
    SYSTEM_Initialize();
    INTERRUPT_GlobalInterruptEnable();
    INTERRUPT_PeripheralInterruptEnable();

    // Set DONE low to start the timer
    DONE_SetLow();
```

3

```
/*
THIS CODE HAS TO BE UNCOMMENTED AND RUN ONLY ONCE
TO CONFIGURE THE CORRECT STARTING VALUES FOR THE MEMORY INDEXES

// write start_pic_eeprom_addr_w in address 0x00 of PIC EEPROM
DATAEE_WriteByte(0x00, start_pic_eeprom_addr_w);
// write start_pic_eeprom_addr_r in address 0x01 of PIC EEPROM
DATAEE_WriteByte(0x01, start_pic_eeprom_addr_r);
// write start_ntag_eeprom_addr in address 0x02 of PIC EEPROM
DATAEE_WriteByte(0x02, start_ntag_eeprom_addr);
*/
```

4

```
while(1) {

    // PIC enters here if WAKE interrupt has arrived
    if(timer == 1) {

        INTCONbits.IOCIE = 0; // Disable IOC interrupts

        I2CPOW_SetHigh(); // Turn on TMP112 and I2C bus

        readTemp();

        I2CPOW_SetLow(); // Turn off TMP112 and I2C bus

        DONE_SetHigh(); // Send DONE signal to TPL5010
        __delay_ms(20);
        DONE_SetLow();

        timer = 0; // Clear timer flag

        INTCONbits.IOCIE = 1; // Enable IOC interrupts

    }
```

5

```
    // PIC enters here if FD interrupt has arrived
    else if (mode == 1 && FD_positive_edge == 0) {

        // Disable the external interrupt
        EXT_INT_InterruptDisable();

        I2CPOW_SetHigh(); // Turn on TMP112 and I2C bus

        readTemp();

        SEL_SetHigh(); // Switch I2C bus from TMP112 to NTAG

        loadBuffer();

        I2CPOW_SetLow(); // Turn off TMP112 and I2C bus

        SEL_SetLow(); // Switch I2C bus from NTAG to TMP112

        EXT_INT_InterruptEnable(); // Enable the external interrupt
```

```
            // Clear timer flag if meanwhile a WAKE has arrived
            timer = 0;
        }
```

```
6
        FD_positive_edge = 0; // Clear FD_positive_edge flag

        SLEEP(); // Go to sleep
```

Nel main vengono chiamate due funzioni, $readTemp$ e $loadBuffer$. Il loro codice viene riportato di seguito.

In $readTemp$:

7. Si aspetta il tempo di conversione di $TMP112$, poi si effettua la richiesta dei due byte di temperatura per un massimo di dieci volte qualora non si avesse successo.
8. Dopo aver recuperato l'indirizzo di memoria nel quale scrivere, i due byte vengono salvati in EEPROM. L'indirizzo $pic\_eeprom\_addr\_w$ viene incrementato e custodito in memoria.

```
// This function read temperature from the sensor
void readTemp() {

    unsigned char timeout; // I2C initial status
    I2C_MESSAGE_STATUS status = I2C_MESSAGE_COMPLETE;
    uint8_t tempBuffer[2]; // buffer that stores the temperature

    __delay_ms(30); // conversion time

    /* Read temperature (2 bytes) from the TMP112.
       In case of errors, try try for 10 times */
    timeout = 0;
7   if (status == I2C_MESSAGE_COMPLETE) {
        while(timeout < 10) {
            // read temperature (2 bytes)
            I2C_MasterRead(tempBuffer, 2, TMP112_addr, &status);
            // wait until transmission ends
            while(status == I2C_MESSAGE_PENDING);
            // check if the message was correctly sent
            if (status == I2C_MESSAGE_COMPLETE) {
                break;
            } else {
                timeout++;
            }
        }
    }
```

```
    // retrieve pic_eeprom_addr_w from EEPROM
    pic_eeprom_addr_w = DATAEE_ReadByte(0x00);

    // write temperature (2 bytes) in PIC EEPROM
    DATAEE_WriteByte(pic_eeprom_addr_w, tempBuffer[0]); // write the 1st byte
```

```
    if (pic_eeprom_addr_w < 255) {
        pic_eeprom_addr_w++;
    } else {
        pic_eeprom_addr_w = start_pic_eeprom_addr_w;
    }
    DATAEE_WriteByte(pic_eeprom_addr_w, tempBuffer[1]); // write the 2nd byte
    if (pic_eeprom_addr_w < 255) {
        pic_eeprom_addr_w++;
    } else {
        pic_eeprom_addr_w = start_pic_eeprom_addr_w;
    }

    // save the updated index in EEPROM
    DATAEE_WriteByte(0x00, pic_eeprom_addr_w);
}
```

8

In $loadBuffer$:

9. Si recuperano gli indirizzi $pic\_eeprom\_addr\_w$ e $pic\_eeprom\_addr\_r$ dalla memoria del PIC. $pic\_eeprom\_addr\_r$ è usato per raccogliere i dati dalla EEPROM che vengono quindi salvati nel vettore $writeBuffer$ per un massimo di 16. L'indirizzo di lettura viene incrementato finché non ha raggiunto quello di scrittura. Questo potrebbe comportare il trasferimento di più pacchetti da 16 Byte se i dati sono in numero superiore. Il vantaggio di avere due indici è quello di acquisire tutti i byte di temperatura raccolti dal *microcontrollore* dal momento dell'ultima lettura da parte del *reader* NFC. I byte di $writeBuffer$ che rimangono liberi, sono posti a zero. Infine gli indirizzi di lettura e scrittura vengono aggiornati e memorizzati.
10. Si invia $writeBuffer$ all'$NTAG$ fino a un massimo di cinque tentativi con la funzione $dataTransfer$. L'indirizzo della memoria EEPROM del *transceiver* nel quale si scrive il buffer da 16 Byte è contenuto in ntag_eeprom_addr.

```
// This function load writeBuffer with data
void loadBuffer(void) {

    uint8_t i;
    uint8_t timeout;

    // retrieve pic_eeprom_addr_w from EEPROM
    pic_eeprom_addr_w = DATAEE_ReadByte(0x00);
    // retrieve pic_eeprom_addr_r from EEPROM
    pic_eeprom_addr_r = DATAEE_ReadByte(0x01);

    // when they are equal, exit
    while(pic_eeprom_addr_r != pic_eeprom_addr_w) {

        i = 1;
        while(i <= 16 && (pic_eeprom_addr_r != pic_eeprom_addr_w)) {

            // fill writeBuffer with PIC EEPROM content
```

9

```
            writeBuffer[i] = DATAEE_ReadByte(pic_eeprom_addr_r);
            if (pic_eeprom_addr_r < 255) {
                pic_eeprom_addr_r++;
            } else {
                pic_eeprom_addr_r = start_pic_eeprom_addr_r;
            }
            i++;
        }

        // fill writeBuffer with zeros if not enough data are available
        if (i < 17) {
            for (i; i <= 16; i++) {
                writeBuffer[i] = 0x00;
            }
        }

        // save the updated index in EEPROM
        DATAEE_WriteByte(0x01, pic_eeprom_addr_r);
```

10

```
        // retrieve ntag_eeprom_addr from EEPROM
        ntag_eeprom_addr = DATAEE_ReadByte(0x02);

        /* Send writeBuffer to NTAG EEPROM.
           In case of errors, try for 5 times */
        timeout = 0;
        while(timeout < 5) {

            // send writeBuffer to NTAG EEPROM
            if(dataTransfer(writeBuffer) == 1) {
                break;
            } else {
                timeout++;
            }
        }
        if (timeout == 5) {
            // warn the user data aren't correct and DON'T UPDATE
            // ntag_eeprom_addr
            //LEDR_SetHigh();
        } else {
            // update the NTAG address
            if (ntag_eeprom_addr < max_ntag_eeprom_addr) {
                ntag_eeprom_addr++;
            } else {
                ntag_eeprom_addr = start_ntag_eeprom_addr;
            }
            // save the updated index in EEPROM
            DATAEE_WriteByte(0x02, ntag_eeprom_addr);
        }
    }
}
```

La funzione *loadBuffer* al suo interno chiama altre due funzioni, *read_ns_reg* e *dataTransfer*.

read_ns_reg:

11. serve per leggere il registro di sessione *NS_REG* dell'*NTAG*. In questo registro si può controllare lo stato della comunicazione con il *transceiver*.

NS_REG si trova all'indirizzo 0xFE ($REG\ ADDR$) ed è il byte numero 0x06 ($REGA$). La funzionalità di $TRANSACTION\_REQUEST\_BLOCK$ ha lo scopo di mettere in coda i byte che si desidera inviare in modo da generare un'unica trama. Infatti, prima di eseguire l'effettiva operazione di lettura, bisogna effettuare quella di write per indicare quale registro si vuole leggere. Maggiori dettagli sulla trama I^2C sono contenuti nel Paragrafo 9.7 e 9.8 del datasheet [18].

```
// This function reads NS_REG register of the NTAG
uint8_t read_ns_reg(void) {

    I2C_MESSAGE_STATUS status = I2C_MESSAGE_COMPLETE;

    uint8_t Buffer[4];
    Buffer[0] = 0xFE; // REG ADDR
    Buffer[1] = 0x06; // REGA

11

    I2C_TRANSACTION_REQUEST_BLOCK TRB[2]; // prepare two I2C transactions
    I2C_MasterWriteTRBBuild(&TRB[0], Buffer, 2, slave_address);
    I2C_MasterReadTRBBuild(&TRB[1], readBuffer, 1, slave_address);

    I2C_MasterTRBInsert(2, TRB, &status); // insert TRB into I2C queue

    while (status == I2C_MESSAGE_PENDING); // wait end of transaction

    return *readBuffer;
}
```

In $dataTransfer$:

12. Si controlla che NS_REG non segnali errori, la cui presenza porterebbe all'azzeramento di tutto il registro. I bit di $MASK$ indicano i bit che si desiderano cambiare all'interno del registro e $REG\ DAT$ è il byte che si vuole scrivere nel registro.
13. Si verifica che la EEPROM dell'$NTAG$ sia libera controllando il bit 1 di NS_REG, fino a cinque volte se necessario, per poi restituire '0' alla funzione $loadBuffer$.
14. Il vettore $writeBuffer$, preso in ingresso dalla funzione $dataTransfer$, viene scritto all'indirizzo $ntag\_eeprom\_address$.
15. Si esamina ancora una volta NS_REG, in particolare il bit 2, per capire se il trasferimento in EEPROM ha avuto problemi. Se così fosse, bisogna provvedere a ripulire manualmente tale bit. In caso di esito positivo, si restituisce '1' alla funzione $loadBuffer$.

```
// This function transfer data from writeBuffer to NTAG EEPROM
uint8_t dataTransfer(uint8_t * writeBuffer) {
```

12

```
I2C_MESSAGE_STATUS status;
I2C_TRANSACTION_REQUEST_BLOCK TRB[2]; // prepare two I2C transactions
uint8_t ns_reg;

ns_reg = read_ns_reg(); // read NS_REG

// if there are errors, clear the whole register
if ((ns_reg & 0xFE) != 0x00) {
    uint8_t Buffer[4];
    Buffer[0] = 0xFE; // REG ADDR
    Buffer[1] = 0x06; // REGA
    Buffer[2] = 0xFF; // MASK
    Buffer[3] = 0x00; // REG DAT

    // prepare I2C transaction
    I2C_MasterWriteTRBBuild(&TRB[0], Buffer, 4, slave_address);

    I2C_MasterTRBInsert(1, TRB, &status); // insert TRB into I2C queue

    while (status == I2C_MESSAGE_PENDING); // wait end of transaction
}
```

13

```
/* Transfer writeBuffer to NTAG EEPROM.
   In case of errors, try for 5 times */
uint8_t timeout1;
uint8_t timeout2;

timeout1 = 0;
// wait until NTAG EEPROM access is possible (NS_REG bit 1)
while(timeout1 < 5) {

    timeout2 = 0;

    while(timeout2 < 5) {

        ns_reg = read_ns_reg(); // read NS_REG

        if((ns_reg & 0x02) == 0x00) { // 1b: EEPROM write cycle in progress
                                      // - access to EEPROM disabled
            break;                    // 0b: EEPROM access possible
        } else {
            timeout2++;
        }
    }
    if(timeout2 == 5) { // if NTAG EEPROM is busy for 5 times, try later
        return 0; // exit
    }
```

14

```
    writeBuffer[0] = ntag_eeprom_addr; // MEMA

    // prepare I2C transaction
    I2C_MasterWriteTRBBuild(&TRB[0], writeBuffer, 17, slave_address);
    I2C_MasterTRBInsert(1, TRB, &status); // insert TRB into I2C queue

    while (status == I2C_MESSAGE_PENDING); // wait end of transaction
```

15

```
    ns_reg = read_ns_reg(); // read NS_REG again

    // check if the writing operation had errors (NS_REG bit 2)
    if((ns_reg & 0x04) == 0x04) { // 1b: HV voltage error during EEPROM
```

```
                                        // write or erase cycle.
                                        // This bit has to be cleared by the user
            uint8_t Buffer[4];
            Buffer[0] = 0xFE; // REG ADDR
            Buffer[1] = 0x06; // REGA
            Buffer[2] = 0x02; // MASK
            Buffer[3] = 0x00; // REG DAT

            // prepare I2C transactions
            I2C_MasterWriteTRBBuild(&TRB[0], Buffer, 4, slave_address);

            I2C_MasterTRBInsert(1, TRB, &status); // insert TRB into I2C queue

            while (status == I2C_MESSAGE_PENDING); // wait end of transaction

            timeout1++;
        } else {
            return 1; // if everything went fine, return
        }
    }

    if (timeout1 == 5) { // if the transfer failed 5 times, exit
        return 0;
    }
}
```

Nella routine di interrupt per gestire il segnale $WAKE$ del $TPL5010$ si setta a '1' il flag *timer* e si azzera il flag EXT_INT per essere in grado di ricevere altri interrupt in futuro. Poi si ritorna immediatamente all'istruzione successiva in modalità $SLEEP$.
La scelta di una interrupt service routine molto corta è una prassi comune nella programmazione di microcontrollori che serve a rendere libera la CPU di gestire altre interruzioni.

```
//***User Area Begin->code
extern unsigned char timer;
//***User Area End->code

/**
  Interrupt Handler for EXT_INT - INT
*/
void INT_ISR(void)
{
    //***User Area Begin->code***
    timer = 1;
    //***User Area End->code***

    EXT_INT_InterruptFlagClear();

    // Callback function gets called everytime this ISR
executes
```

```
    INT_CallBack();
}
```

Nella routine di interrupt per gestire il segnale FD dell'$NTAG$ prima di tutto si disabilitano gli IOC. Dopo aver controllato che il pin FD sia realmente basso per due volte, si cambia il valore della variabile $mode$ e si riabilitano gli IOC. Il motivo della disabilitazione degli IOC e della necessità di questi controlli ridondanti è dovuta al fatto che FD potrebbe avere molte transizioni alto-basso se lo smartphone non è posizionato entro una distanza di qualche centimetro dall'antenna. Se invece è avvenuto un falso allarme per cui FD torna alto, si imposta il flag $FD\_positive\_edge$ a '1' per evitare di eseguire delle operazioni non volute al ritorno dalla ISR.

```
// Import two variables from main
extern unsigned char mode;
extern unsigned char FD_positive_edge;

/**
   IOCAF5 Interrupt Service Routine
*/
void IOCAF5_ISR(void) {

    // Add custom IOCAF5 code

    INTCONbits.IOCIE = 0; // don't handle other IOC interrupts

    if (FD_GetValue() == 0) { // check if FD has gone low
        __delay_ms(2000); // wait to stabilize the signal

        if (FD_GetValue() == 0) { // check again
            mode = !mode; // change mode
        } else {
            FD_positive_edge = 1; // if it was a false alarm
        }

    } else {
        FD_positive_edge = 1; // if it was a false alarm
    }

    INTCONbits.IOCIE = 1; // enable IOC interrupts

    // Call the interrupt handler for the callback registered at runtime
    if(IOCAF5_InterruptHandler)
    {
        IOCAF5_InterruptHandler();
    }
    IOCAFbits.IOCAF5 = 0;
}
```

É importante sottolineare che quando il PIC entra in sleep, tutte le porte mantengono il loro valore logico impostato e, al risveglio, esegue la prima istruzione dopo la funzione $SLEEP$, se non è arrivato nessun interrupt da gestire.

5. Collaudo

Come ultimo step è stato collaudato il circuito di interfaccia. L'interesse principale è legato alla commutazione dell'alimentazione, che avviene nella fase di lettura dell' *NTAG I2C plus* da parte di uno smartphone NFC, e al trasferimento dei valori di temperatura nella memoria EEPROM del *transceiver*. Tutte le altre funzionalità sono rimaste invariate rispetto al progetto del nodo sensore [2] e non verranno ricontrollate. Per di più sono stati esaminati i nuovi consumi in stand-by per dimostrare di aver mantenuto un basso profilo energetico. Per stand-by si intende la fase in cui il *sensore* e il *voltage monitor* sono spenti, gli *switch analogici* sono accesi, il *PIC* è in sleep e il *timer* esegue il conteggio.

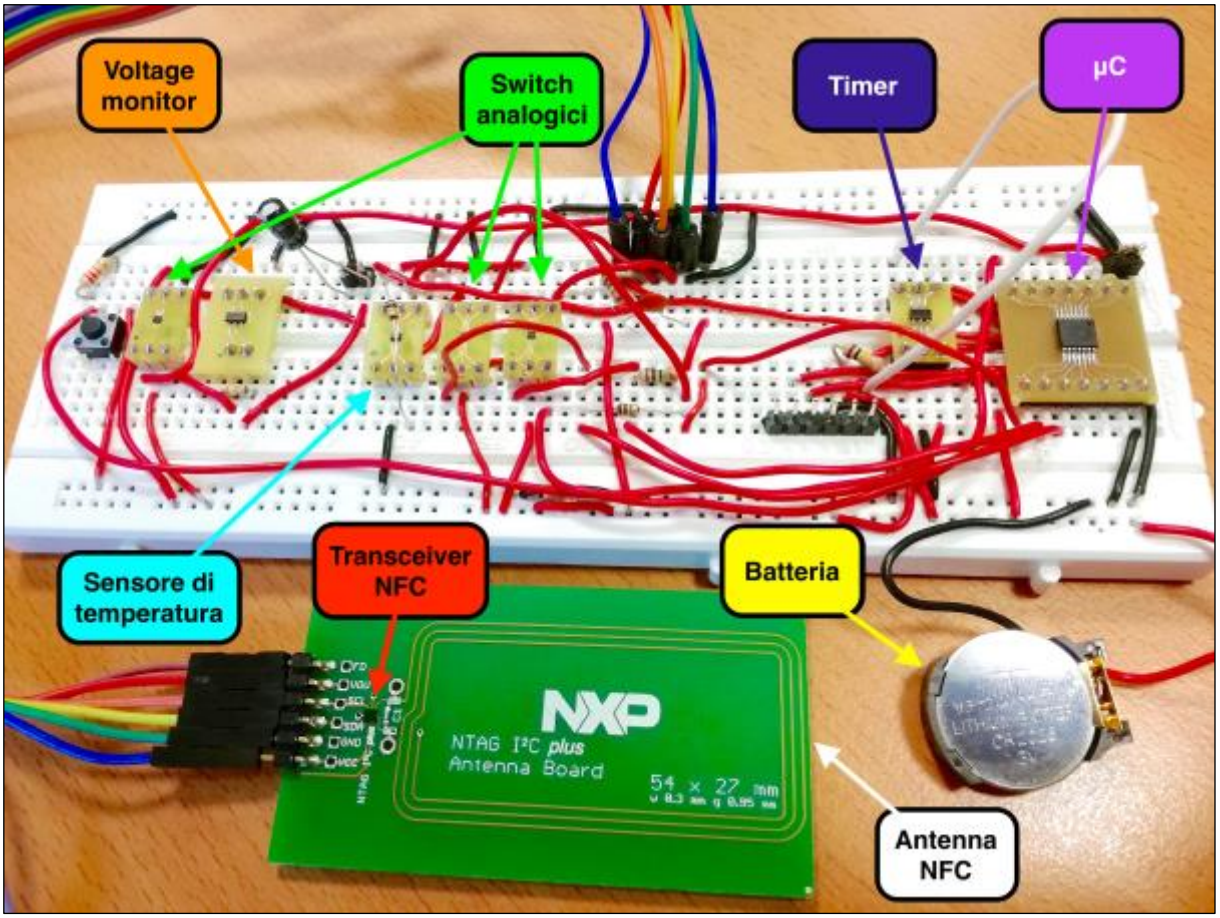


Figura 29 Il circuito di interfaccia basato su tecnologia NFC per nodi sensori a bassissimi consumi.

5.1. Test del circuito di commutazione

Al posto della batteria a bottone, il circuito è stato alimentato con una Power Supply Unit (PSU) esterna dotata di limitatore di corrente nel caso di cortocircuiti causati da un errato montaggio dei componenti.
Per l'analisi dei segnali di interesse, quali V_{out_sw} , FD , $Reset\_Output$, $I2CPOW$, è stato usato un oscilloscopio a quattro canali. I campioni raccolti sono stati esportati in fogli di calcolo formato CSV per essere ripuliti dal rumore e graficati con una maggiore risoluzione in Matlab.
Come dispositivo NFC attivo di lettura è stato sfruttato uno smartphone capace di comunicare con NFC.

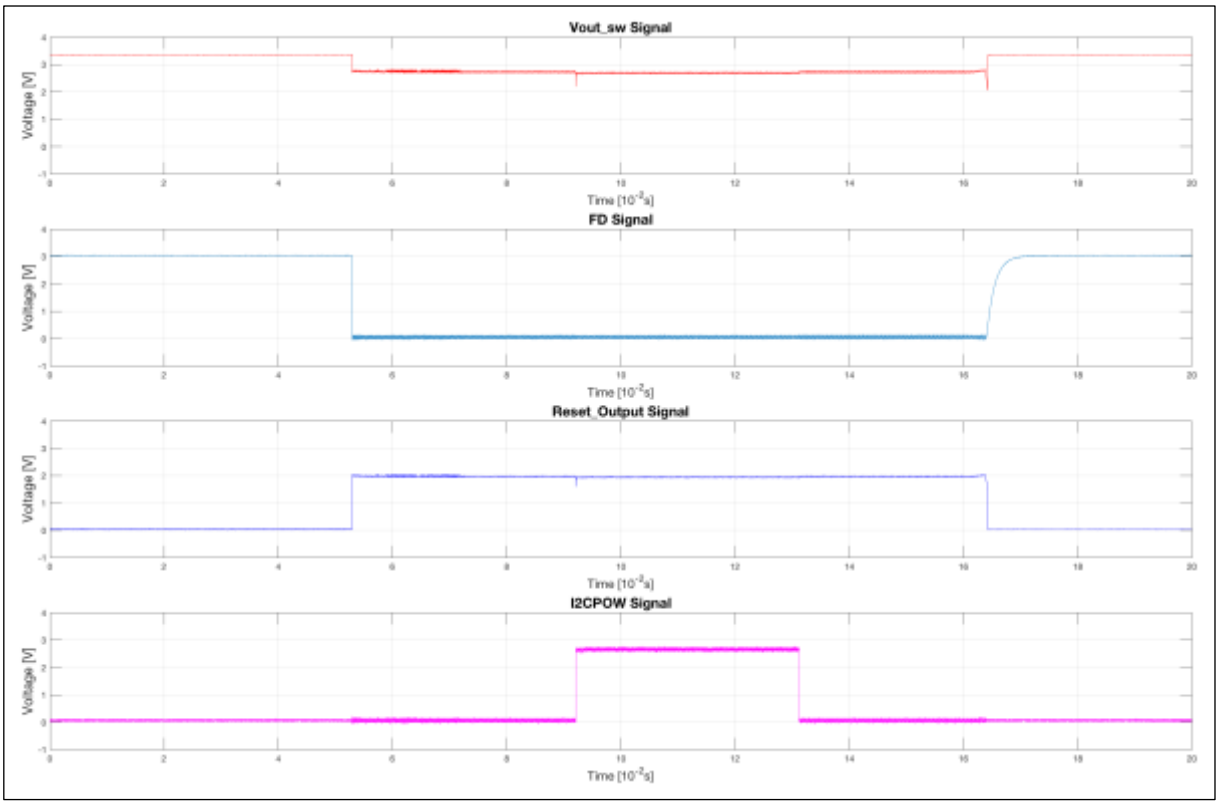


Figura 30 Modalità #1.

Nel primo test si avvicina lo smartphone al *target* passivo mantenendo una distanza operativa inferiore ai 5 cm. In questa situazione è osservabile da Figura 30 che:

- il segnale FD si abbassa per segnalare la presenza di un campo magnetico sufficiente a permettere il funzionamento dell'$NTAG$;
- il *voltage monitor* percepisce una tensione V_{out_ntag} maggiore di 2.1 V, perché alza $Reset\_Output$;

- la tensione V_{out_sw} si è modificata, perché lo *switch analogico*, comandato da $Reset\_Output$, ha commutato la sua uscita in favore di V_{out_ntag}, distaccando l'alimentazione esterna;
- il segnale $I2CPOW$ rimane alto per un certo tempo, perché il PIC è entrato in modalità #1 e ha iniziato i trasferimenti di dati descritti nel commento al main di Paragrafo 3.4 e 4.2;
- tutti i segnali tornano al loro valore iniziale al momento dell'allontanamento dello smartphone: la tensione V_{out_sw} è fornita nuovamente dalla PSU.

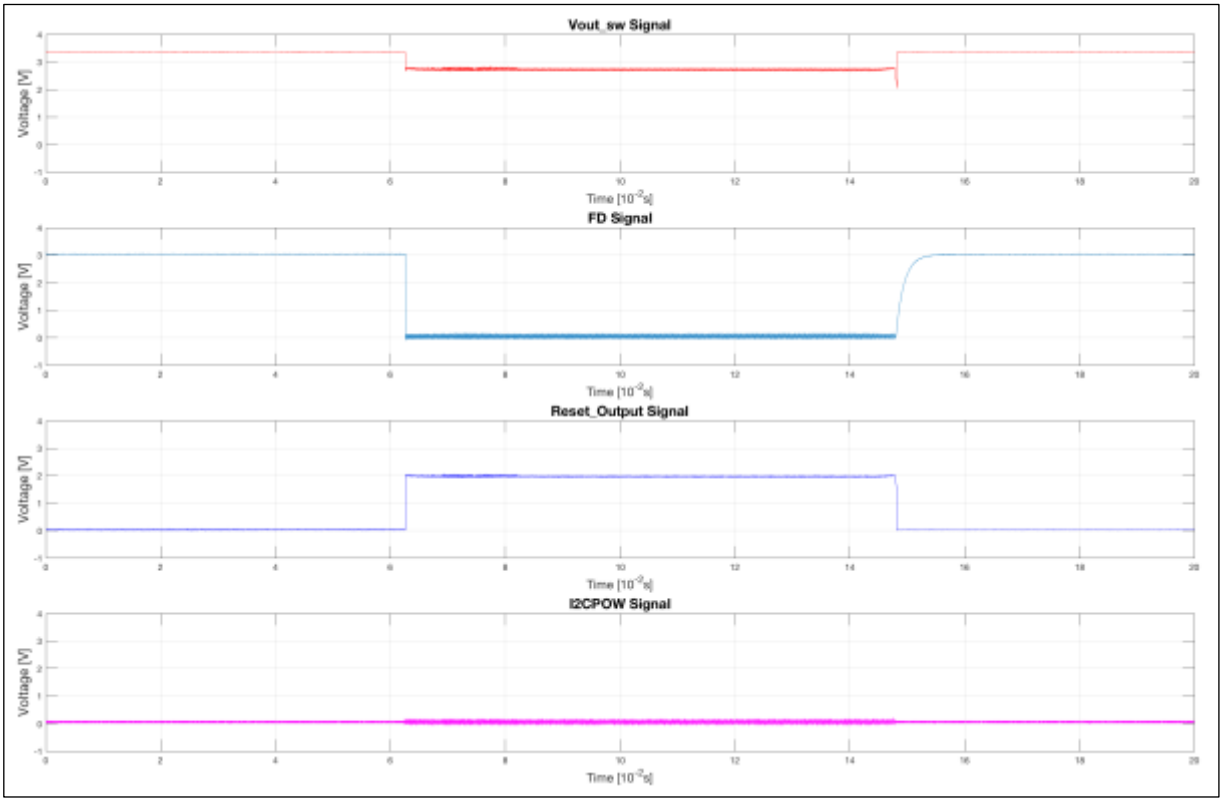


Figura 31 Modalità #0.

Nel secondo test si accosta nuovamente lo smartphone all'antenna del *transceiver*. Questa volta si nota in Figura 31 che:

- i segnali V_{out_sw}, FD e $Reset\_Output$ si comportano sempre allo stesso modo,
- $I2CPOW$ rimane basso, perché il PIC è entrato in modalità #0 nella quale non esegue trasferimenti di dati (vedi Paragrafo 3.4 e 4.2);
- tutti i segnali tornano al loro valore iniziale al momento dell'allontanamento dello smartphone la tensione V_{out_sw} è fornita nuovamente dalla PSU.

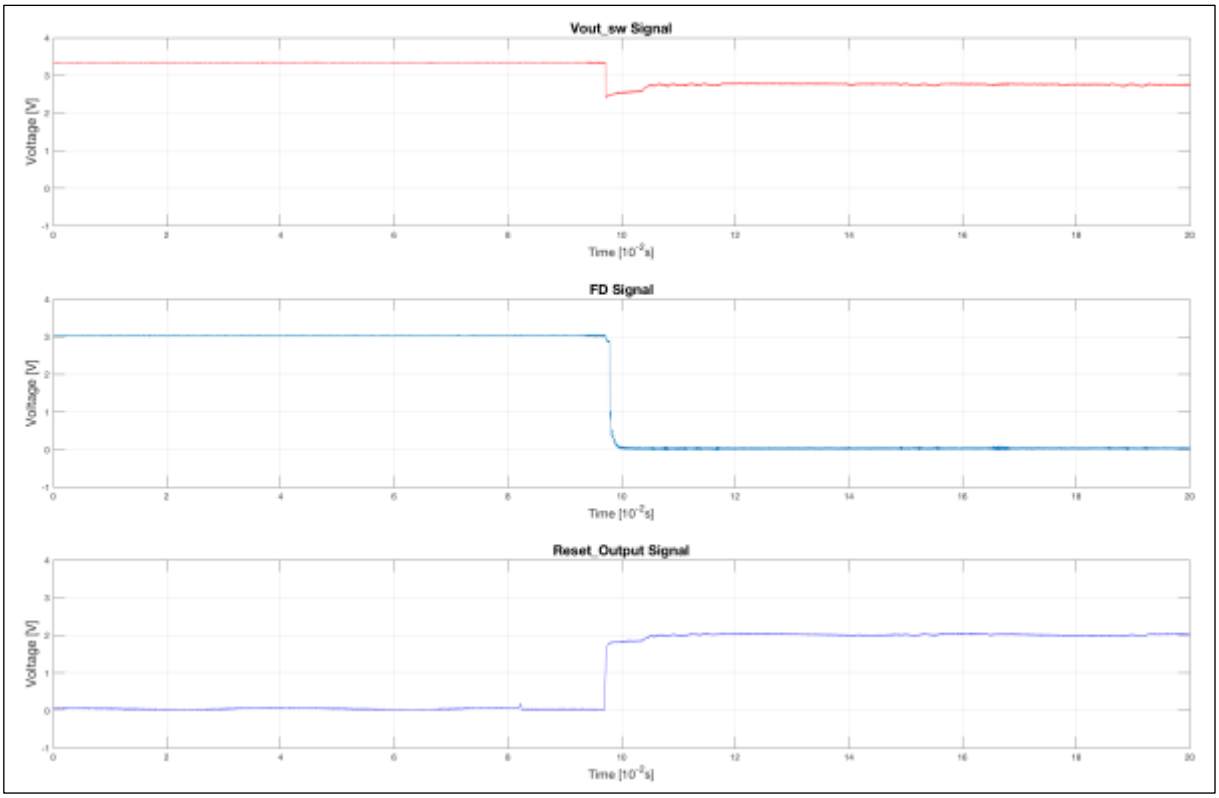


Figura 32 Zoom commutazione da PSU a V_{out_ntag}.

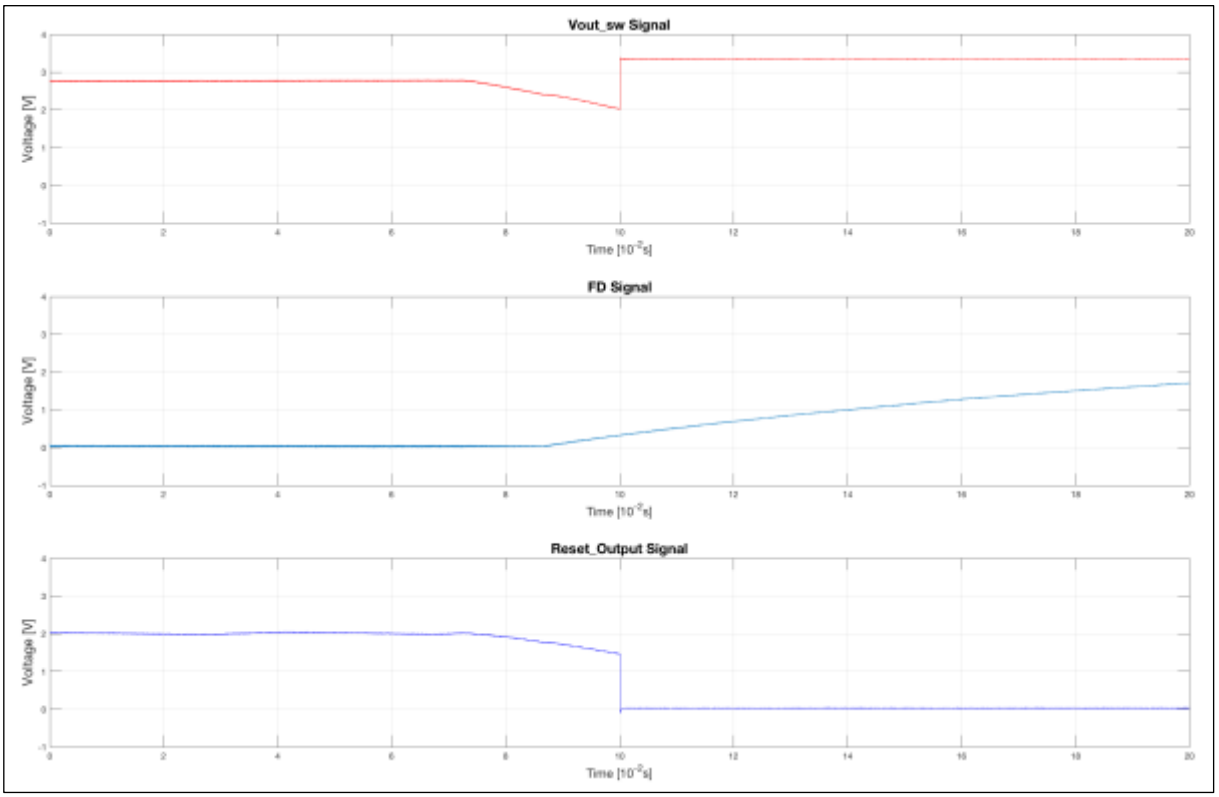


Figura 33 Zoom commutazione da V_{out_ntag} a PSU.

Per completezza si riportano anche gli zoom delle commutazioni di V_{out_sw} da PSU a V_{out_ntag} e viceversa. Nella Figura 33 è molto evidente cosa succede quando lo smartphone viene distanziato dal *target*. Il primo segnale che cambia è FD che intraprende l'inizio di un transitorio di carica RC (si ricordi la capacità $C3$ nello schema elettrico). Successivamente V_{out_sw} non torna subito ad essere la tensione fornita dalla PSU, ma impiega un tempo di circa 12 ms. Infatti, seppur il pin V_{out_ntag} non fornisca più corrente, la capacità $C4$ ad esso collegato (unita alla $C2$ in uscita dallo *switch*) aveva accumulato della carica che viene rilasciata lentamente. Questo causa una lenta diminuzione del valore di tensione del nodo in questione. Quando però V_{out_ntag} raggiunge il valore di 1.9 V, il *voltage monitor* abbassa bruscamente $Reset\_Output$ e la PSU torna a fornire energia al sistema.
La scelta dei valori delle capacità $C2$ e $C4$ tende quindi ad allungare il transitorio prima della commutazione da V_{out_ntag} a PSU. Tuttavia un valore molto elevato potrebbe causare il reset del *microcontrollore*, che avviene quando la sua alimentazione scende sotto gli 1.8 V (tensione minima di alimentazione del dispositivo [18]). Infatti al momento di questo cambio di alimentazione una capacità grande richiede un maggior apporto di corrente dalla PSU, causando una caduta di tensione su V_{out_sw} che scende pertanto sotto gli 1.8 V. Un valore complessivo inferiore alla decina di μF è tollerato.

5.2. Verifica dei dati trasmessi

Ai fini di accelerare il processo delle acquisizioni di temperatura, è stato sostituito il resistore R_1 da 91 KΩ, con uno da 22 KΩ. Così facendo, la temporizzazione del $TPL5010$ è diventata di:

$$T = a * {R_1}^2 * 10^{-6} + b * R_1 * 10^{-4} + c * 10^{-2} \cong 59.81\ s$$

con a = 0.1972, b = -19.3450, c = 692.1201 parametri validi per l'intervallo $10\ s < T < 100\ s$ [19].

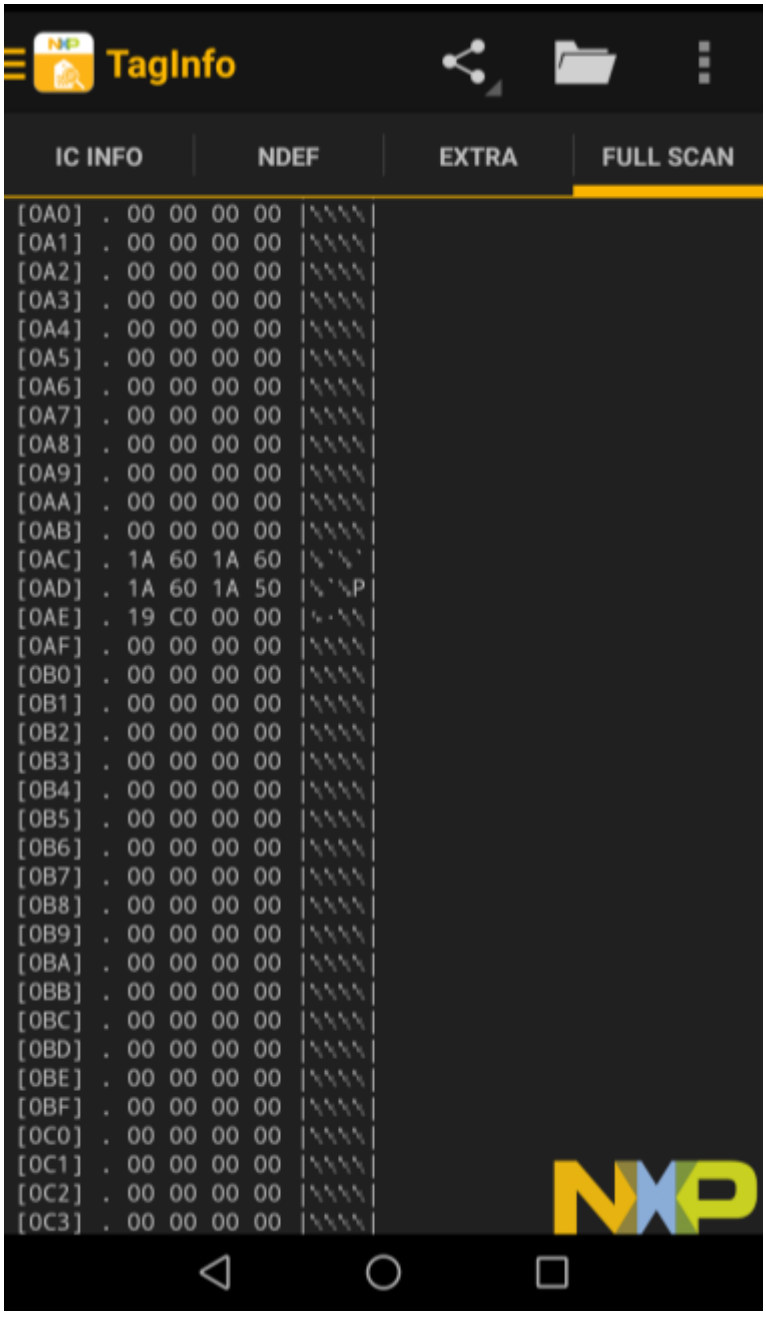
Inoltre è stata installata l'applicazione $NFC\ TagInfo\ by\ NXP$ su uno smartphone NFC per eseguire le operazioni di formattazione e lettura della EEPROM dell' $NTAG$. Infatti, prima dell'accensione del circuito, è stata ripristinata la memoria del *target* alle impostazioni di default di fabbrica in

modo da eliminare tutto il contenuto di precedenti esperimenti e capire subito quali siano i nuovi byte trasmessi.

Dopo circa 4 minuti dall'accensione del circuito, è stata eseguita una lettura con la suddetta applicazione. Il risultato ottenuto è in Tabella 8, parte sinistra. Nello stesso momento un *protocol analyzer* stava catturando il traffico delle linee SDA e SCL e ha restituito la decodifica di Tabella 8, parte destra. Confrontando i dati, si può allora concludere che il trasferimento è avvenuto correttamente.
Le varie fasi sono qui brevemente commentate:

- nel blocco 1 viene acquisito un dato di temperatura dal *sensore*,
- nel blocco 2 si controlla che il registro NS_REG non abbia errori,
- nel blocco 3 si esamina che la EEPROM sia libera,
- nel blocco 4 si trasferiscono i 16 Byte all'$NTAG$ (si tenga presente che l'indirizzo di scrittura dal punto di vista dell'I^2C, 0x44, è diverso da quello del *reader* NFC, 0x0AC, ma indicano la stessa porzione di memoria),
- nel blocco 5 si verifica che non siano avvenuti errori nella scrittura della EEPROM.

Ogni valore di temperatura del $TMP112$ è formato da 12 bit. Per ricavare la temperatura in formato analogico, si procede in due modi diversi a seconda che il dato sia positivo (MSB del byte più significativo = 0) oppure negativo (MSB del byte più significativo = 1). Nel primo caso la parola binaria dev'essere convertita in decimale e poi moltiplicata per la risoluzione dell'ADC, cioè 0.0625°C. Nel secondo la parola deve subire un'operazione di complemento a due, poi deve essere moltiplicata per la risoluzione dell'ADC e infine cambiata di segno [20].
Nella lettura di Tabella 8 si evincono i valori di 26.38 °C, 26.38 °C, 26.38 °C, 26.31 °C, 25.75 °C. Si ricorda che l'accuratezza del *sensore* è di $\pm$ 0.5 °C.



	Decoded Protocol Result
1	Setup Read to [0x91] + ACK
	0x1C + ACK
	0x70 + NAK
2	Setup Write to [0x01] + ACK
	0xFE + ACK
	0x06 + ACK
	Setup Read to [0x00] + ACK
	0x01 + NAK
3	Setup Write to [0x01] + ACK
	0xFE + ACK
	0x06 + ACK
	Setup Read to [0x00] + ACK
	0x01 + NAK
4	Setup Write to [0x01] + ACK
	0x44 + ACK
	0x1A + ACK
	0x60 + ACK
	0x1A + ACK
	0x60 + ACK
	0x1A + ACK
	0x60 + ACK
	0x1A + ACK
	0x50 + ACK
	0x19 + ACK
	0xC0 + ACK
	0x00 + ACK
	0x00 + ACK
	0x00 + ACK
	0x00 + ACK
	0x00 + ACK
	0x00 + ACK
5	Setup Write to [0xAA] + ACK
	0xFE + ACK
	0x06 + ACK
	Setup Read to [0xAB] + ACK
	0x01 + NAK

Tabella 8 A sinistra uno screenshot dell'applicazione per smartphone, a destra il risultato del *protocol analyzer*.

É lecito fare una considerazione sulla frequenza con la quale bisogna leggere i dati raccolti dal nodo sensore prima che vengano sovrascritti per mancanza di spazio in memoria. Una temporizzazione del *timer* di 30 minuti, non è realistica. Si supponga quindi di misurare la temperatura ogni 2 ore. La memoria

EEPROM del PIC verrebbe occupata in appena 10 giorni e mezzo perché ogni acquisizione di temperatura occupa 2 Byte e la disponibilità in memoria è di 256 Byte. Una soluzione potrebbe essere quella di salvare i dati direttamente nella EEPROM dell'NTAG che dispone di molto più spazio. Infatti possiede due aree in cui operare, una va dall'indirizzo I^2C 01h al 37h (in aggiunta ai primi due byte del 38h), l'altra dal 40h al 7Fh. Ogni indirizzo contiene 16 byte, perciò nel complesso si hanno 888 Byte e 1024 Byte. Dunque la sua memoria si saturerebbe dopo circa 80 giorni.
L'alternativa è utilizzare una parte dei 7 KB di memoria FLASH del PIC non occupata dal codice oppure aggiungere una memoria EEPROM esterna al circuito. In entrambi i casi una nuova analisi dei consumi risulta necessaria.

Nel progetto di interfaccia è stata usata solo la prima porzione di memoria del *transceiver* perché è sufficiente a contenere i 256 Byte della EEPROM del *microcontrollore*.

5.3. Verifica dei nuovi consumi

Per ultimo si deve modificare il valore di assorbimento di corrente della fase di stand-by rispetto a quello calcolato in [2] perché sono stati aggiunti tre *switch analogici*. Si rammenta che il *voltage monitor* è alimentato solo dal suo segnale di $Input$ che, essendo V_{out_ntag}, quando non c'è campo RF vale 0 V. Per questo motivo il suo impatto energetico non va tenuto in conto.

I consumi teorici di un singolo switch analogico $AS11P2TLR$ sono già stati citati nel Paragrafo 3.3. Questo valore andrà quindi moltiplicato per tre. Invece quelli sperimentali sono stati misurati con il multimetro da banco a 6 cifre e ½ $Agilent\ 34401A$: si è ricavato un consumo inferiore ai 100nA. Per una valutazione più accurata, sarebbe servito uno strumento ancora più preciso.
La tabella riassuntiva dei consumi di ciascuna fase, aggiornato con i quelli dei nuovi componenti, è la seguente:

		Valori teorici		**Valori sperimentali**	
Fase	**Simbolo**	**Durata**	**Assorbimento**	**Durata**	**Assorbimento**
Conversione	c	26ms	44.04 µA	26 ms	50 µA
Trasmissione	t	18ms	19.04 µA	65 ms	8 µA
Elaborazione	e	31ms	4.04 µA	100 ms	8 µA
Scrittura EEPROM	m	Non prevista		6 ms	95 µA
Stand-by	s	1745 s	70 + 20*3 nA	1745 s	100 nA

Tabella 9 Confronto tra i consumi teorici e quelli sperimentali con $R1$ = 91 KΩ [2].

Si conclude con una stima dei consumi calcolando la corrente media:

$$I_{medio} = \frac{I_c t_c + I_t t_t + I_e t_e + I_m t_m + I_s t_s}{T} = 101.82\ nA$$

Con una classica batteria $CR2032$, che ha almeno C = 210 mAh, ignorando i fenomeni di invecchiamento, il tempo di vita del sistema sarebbe:

$$t = \frac{C}{I_{medio}} = 2062532{,}332\ ore \sim 85938\ giorni \sim 235\ anni$$

Da notare che la fase di stand-by è calcolata con una resistenza $R1$ = 91 KΩ, ma aumentando la temporizzazione del *timer*, il tempo di vita risulterebbe ancora più alto. Ovviamente questa durata non è raggiungibile nella realtà perché le batterie attualmente in commercio hanno dei tempi di vita massimi intorno ai 10 anni. Tuttavia l'obiettivo di mantenere un bassissimo consumo del sistema è stato comunque raggiunto.

Conclusioni

Gli scopi che questo elaborato si era prefissato sono stati raggiunti.

Il circuito di interfaccia, grazie alla tecnologia NFC, permette di leggere i dati di temperatura raccolti e collezionati dal *microcontrollore* per renderli disponibili ad un possibile utilizzatore. Inoltre, la sua circuiteria di commutazione e gestione dell'energia preserva la batteria quando si effettua una lettura con un *reader* NFC, sfruttando la capacità di *energy harvesting* dell'antenna, senza comportare maggiori consumi di corrente nella fase di stand-by. Infatti una singola batteria a bottone è ampiamente in grado di soddisfare le richieste energetiche del circuito che, teoricamente, potrebbe durare più di 200 anni.

La tecnologia NFC consente il trasferimento bidirezionale delle informazioni e dell'energia tra i dispositivi oltre al vantaggio di disporre di uno standard largamente adottato nei terminali mobili commerciali facilitando la trasmissione delle informazioni all'utente finale. Costituisce, così, la migliore soluzione per il tipo di applicazione oggetto di tesi. É
É
stato anche realizzato un prototipo di antenna NFC per PCB le cui caratteristiche elettriche andranno testate.

Con l'impiego di una memoria sufficientemente capiente per contenere i dati desiderati, una successiva implementazione su PCB e un'applicazione per smartphone che visualizzi automaticamente le temperature raccolte, un dispositivo simile si presta ad essere impiegato in scenari di monitoraggio ambientale in aree geografiche in cui è richiesta una lunga autonomia.

Bibliografia

[1] J. Rifkin, *La società a costo marginale zero,* Mondadori, 2014.

[2] D. Fazi, *Progetto di un nodo sensore a nanocorrenti basato su microcontrollore,* Università di Bologna, Italia, 2017.

[3] STMicroelectronics, *TN1216. Technical note. ST25 NFC guide,* 2016.

[4] «NearFieldCommunication.org,» [Online]. Available: http://nearfieldcommunication.org.

[5] NFC Forum, «NFC Forum,» [Online]. Available: https://nfc-forum.org.

[6] B. Bilginer e P.-L. Ljunggren, *Master's Thesis in: Near Field Communication,* Università di Lund, Svezia, 2011.

[7] STMicroelectronics, *AN2972. Application note. How to design an antenna for dynamic NFC tags.*

[8] «eDesignSuite by STMicroelectronics,» [Online]. Available: http://www.st.com/content/st_com/en/support/resources/edesign.html.

[9] ISO/IEC, *ISO/IEC JTC 1/SC 17: Cards and personal identification,* 2010.

[10] NXP, *AN11276: NTAG Antenna Design Guide,* 2016.

[11] AMS, *AS3955: NFC Forum Compliant Dynamic Tag.*

[12] NXP, *UM10967: NTAG I2C plus Explorer Kit Peek and Poke.*

[13] NXP, *NT3H2211: NTAG I2C plus, NFC Forum Type 2 Tag compliant IC with I2C interface,* 2016.

[14] NXP, *NFC for embedded applications,* 2014.

[15] ON Semiconductor, *NCP303: Voltage Detector Series with Programmable Delay*, 2014.

[16] STMicroelectronics, *AS11P2TLR: Low voltage 1 Ω single-pole double-throw analog switch with break- before-make feature,* 2014.

[17] Diodes Incorporated, *DZ23C3V3-7-F: 300mW Dual Surface Mount Zener Diode,* 2013.

[18] Microchip, *PIC16(L)F1824/1828: 28/40/44-Pin, Low-Power, High-Performance Microcontrollers with nanoWatt XLP Technology,* 2010.

[19] Texas Instruments, *TPL5010: Nano-power System Timer with Watchdog Function,* 2015.

[20] Texas Instruments, *TMP112: High-Accuracy, Low-Power, Digital Temperature Sensor With SMBus and Two-WireSerial InterfaceinSOT563,* 2015.