EMULATORE SRIX4K CON TRF7970A

Ver 1.0 – By Ptr

Indice generale

1 Introduzione

In questo documento vi mostro come ho provato a realizzare un emulatore di un tag SRIX4K usando l'integrato TRF7970A della Texas Instruments. Dalle prove fatte ho notato che l'emulatore funziona se la lettura viene effettuata con il PN532 oppure con la demoboard M24LR-DISCOVERY di ST (basata sul chip CR95HF) ma non con il CRX14. Questo purtroppo impedisce di realizzare un emulatore che funzioni con ogni lettore disponibile (il perché non funzioni con il CRX14 rimane ancora da indagare).

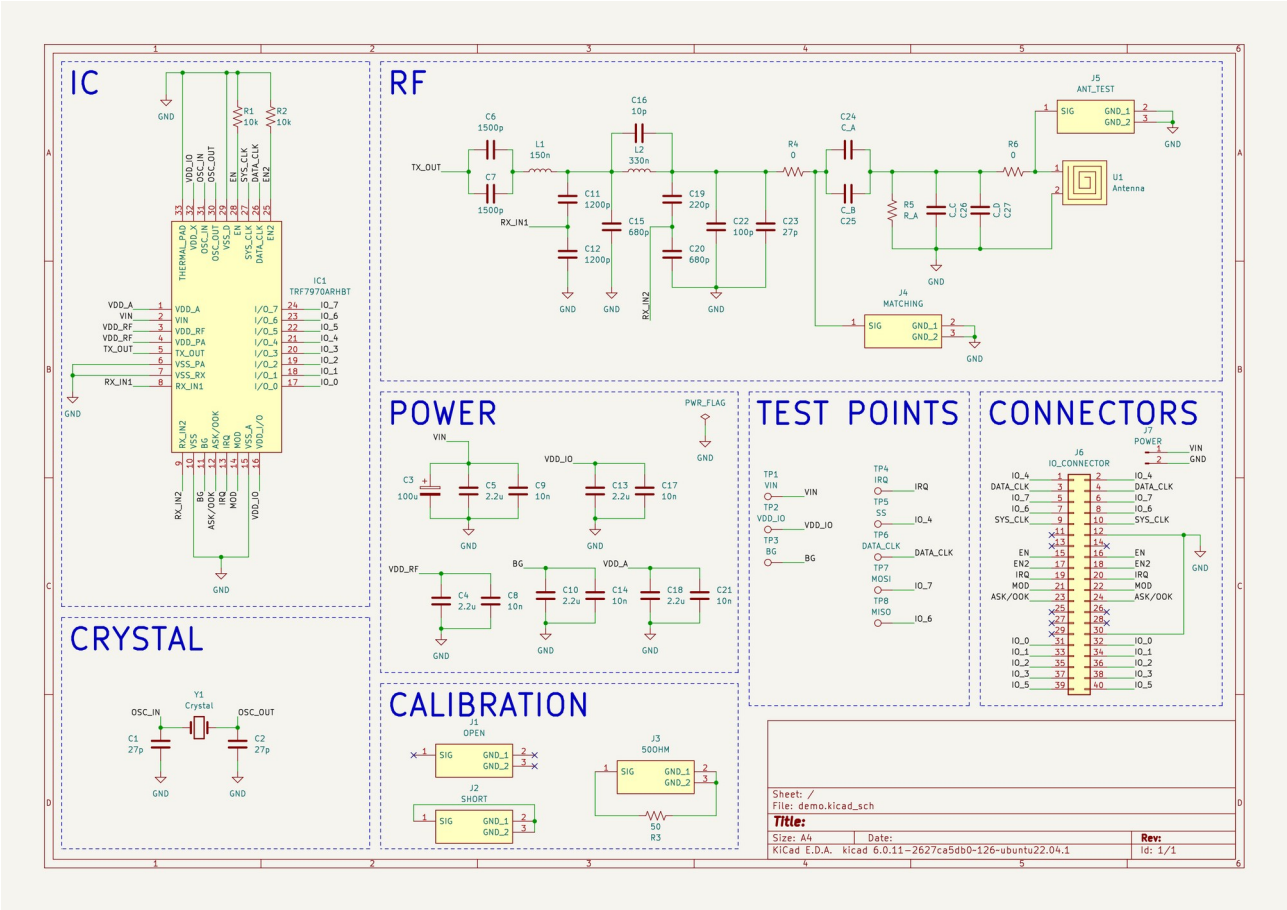
Per l'integrato TRF7970A esisteva una demoboard di Texas Instruments che purtroppo non è più in commercio (o meglio, non sono riuscito a trovarla online). Ho dovuto realizzare io una pseudo demoboard basata sullo schema di esempio che si trova nel datasheet.
Dato che ho fatto solo una versione della board, nel documento trovate anche alcuni commenti su eventuali migliorie o modifiche che si possono effettuare.

Il progetto originale che prevedeva la realizzazione di un emulatore con il TRF7970A è di lilz0C ed è stato proposto anni fa su inforge (https://www.inforge.net/forum/threads/srix4k-anti-lock-id-progetto-black-phoenix.593402/). Specifico che questa guida non si riferisce in alcun modo a quel progetto (non so se sia mai stato iniziato e non so cosa sia stato fatto in questi anni).

2 Schema

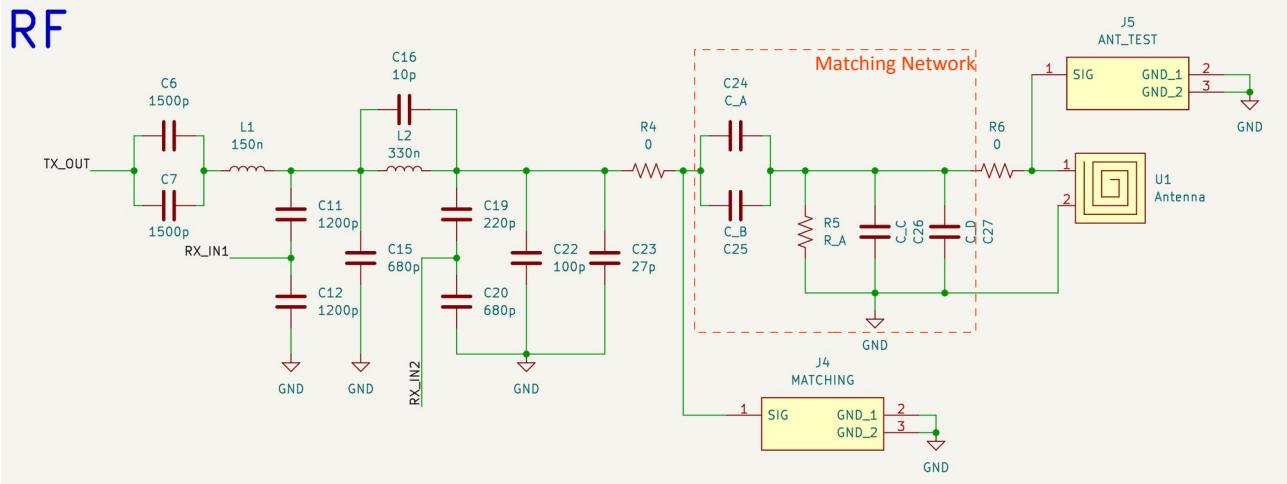
Uno schema di esempio per il TRF7970A è presente nel datasheet (https://www.ti.com/lit/ds/symlink/trf7970a.pdf) a pagina 75.

Lo schema della board che ho realizzato è mostrato nella figura seguente:



È sostanzialmente identico allo schema presente nel datasheet, a parte alcune modifiche che verranno descritte in seguito.

2.1 Stadio RF



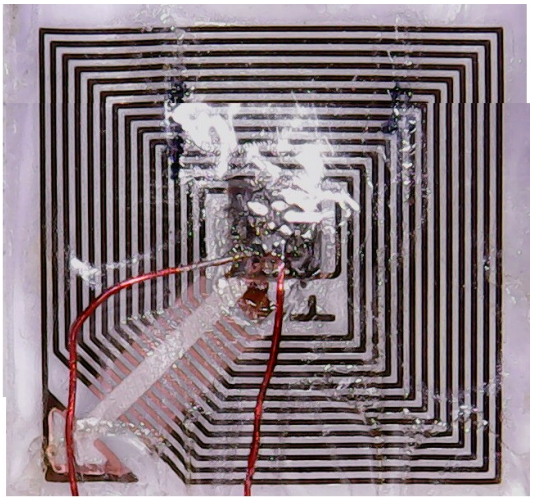
Lo stadio RF per la trasmissione/ricezione è stato copiato dal datasheet. La rete per l'adattamento di impedenza ("Matching Network", indicata con la linea rossa tratteggiata nell'immagine) è necessaria per "trasformare" l'impedenza dell'antenna nei 50Ω richiesti dal circuito. Non sono presenti dei valori fissi perché devono essere calcolati in base all'impedenza dell'antenna; non essendo nota a priori, deve essere misurata. Per fare ciò, useremo il NanoVNA, un Vector Network Analyzer molto potente e a basso costo che si può acquistare su Amazon.
I 50Ω vengono poi trasformati nei 4Ω richiesti da TX_OUT grazie ad una rete di adattamento con valori "fissi".

Per la misura dell'impedenza di antenna e della rete di adattamento sono stati inseriti dei connettori U.FL indicati nello schema come J5 e J4:

Il footprint e il simbolo li trovate qui https://componentsearchengine.com/part-view/CONUFL001-SMD-T/Linx%20Technologies
Le resistenze R4 e R6 da zero Ohm servono per isolare l'antenna (R6) e la rete di impedenza (R4) durante le misure con il NanoVNA.

2.2 Antenna

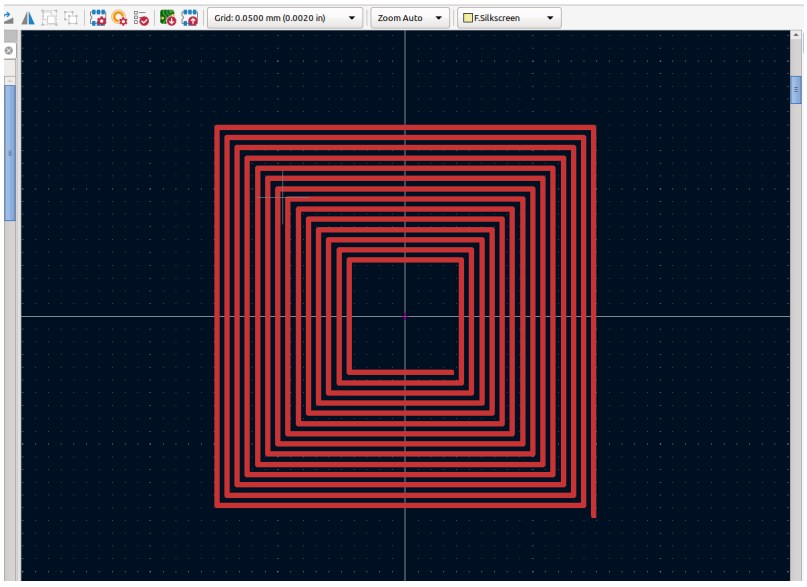
Per realizzare l'antenna su PCB ho preso spunto dall'antenna di una chiavetta (qui potete vedere un ingrandimento):



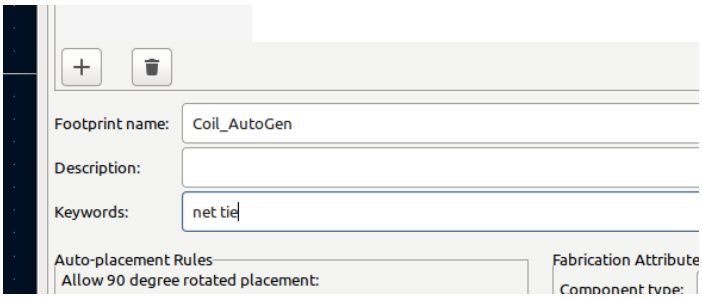
L'antenna è un quadrato di 15mm con ~14 spire di filo di ~0.2mm di diametro. Per replicare la stessa antenna con Kicad ho usato uno script trovato su Github (https://github.com/joards/Simple-Planar-Coil-Generator).
L'antenna è stata generata con questo comando:

```
python SquareCoilMaker.py -d 1 -lw 0.2 -sp 0.2 -x 15 -y 15 -N 14 > Coil_AutoGen
```

Il comando creerà un file chiamato "Coil_Autogen" che potrà essere importato in Kicad. Per fare ciò, dovete aprire il "Footprint Editor" e fare "File"→ "Import" → "Footprint" e selezionare il file "Coil_Autogen" che è stato creato.



È importante modificare le footprint properties facendo “File” → “Footprint Properties” inserendo la frase “net tie” nella casella “Keywords” (Il motivo è spiegato qui https://forum.kicad.info/t/antenna-tht-pad-clearance-error-and-difficulties-to-connect-trace-to-it/39498).

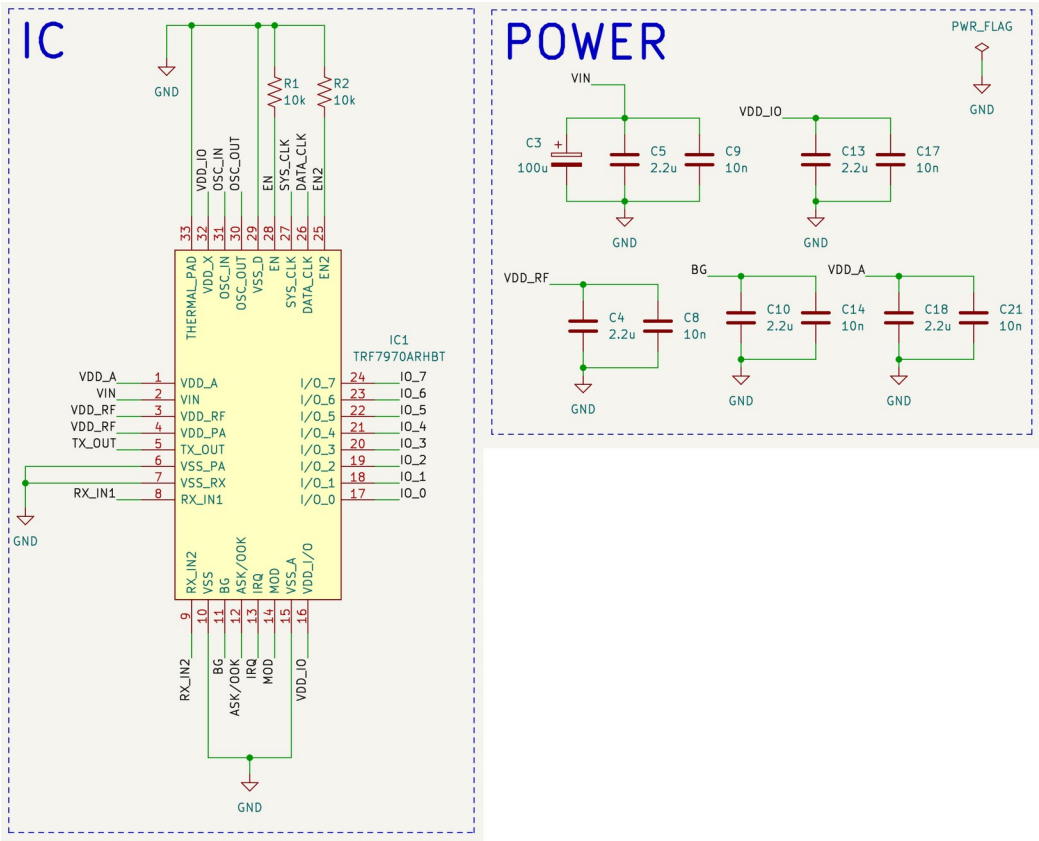


A questo punto il footprint può essere salvato all’interno di una libreria già esistente oppure è possibile creare una libreria nuova e salvare lì il footprint appena creato.

EVENTUALI MODIFICHE

- Per realizzare l’antenna è consigliato usare dei tool appositi per fare in modo di avere un valore di induttanza pari a quello consigliato dal produttore del chip (questo documento di TI https://www.ti.com/lit/an/sloa241c/sloa241c.pdf consiglia di avere un’induttanza fra 1 e 1.5 µH). Si può usare per esempio questo tool di ST https://eds.st.com/antenna/#/ per progettare l’antenna voluta.
  L’antenna che ho realizzato ha un’induttanza di 2.2 µH (vedremo poi come misurarla con il NanoVNA).

2.3 IC + Power



Il simbolo e il footprint del TRF7970 li potete scaricare da qui https://componentsearchengine.com/part-view/TRF7970ARHBT/Texas%20Instruments

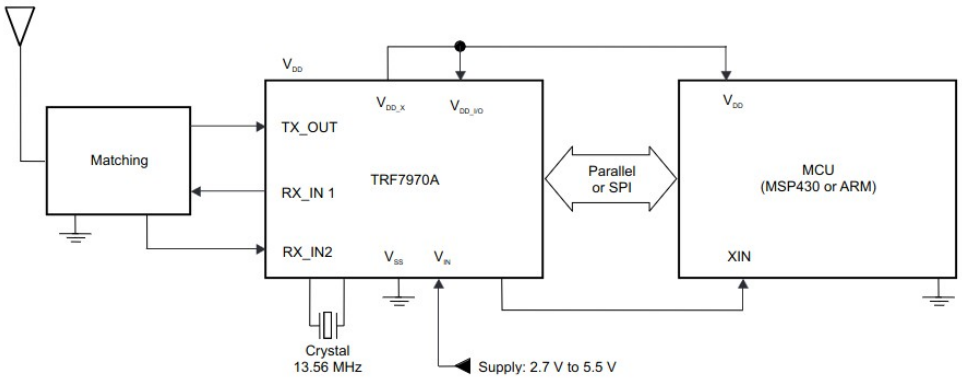
Le alimentazioni dell'IC sono le seguenti:

- VIN → Tensione di ingresso, può variare da 2.7V a 5.5V
- VDD_RF → Tensione di uscita del regolatore che alimenta lo stadio a RF dell'IC. Può essere impostato via SPI a 5V o a 3V
- VDD_A → Tensione di uscita del regolatore che alimenta la parte analogica dell'IC.
- VDD_X → Tensione di uscita del regolatore che alimenta la parte digitale dell'IC. Nello schema viene connessa a VDD_IO che è la tensione di ingresso dei level shifter presenti sui pin I/O.
- BG → Tensione di uscita del bandgap.

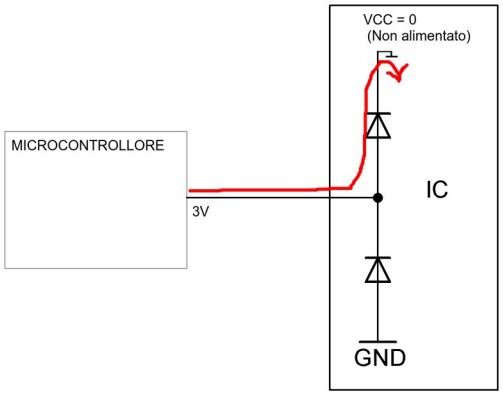
Su tutte le tensioni sono presenti dei condensatori di bypass (come specificato nello schema di esempio presente nel datasheet).

EVENTUALI MODIFICHE

- La scheda che ho realizzato è stata pensata per avere VIN come alimentazione dell'IC e come alimentazione di un microcontrollore esterno. In realtà, avendo connesso VDD_X con VDD_IO, il microcontrollore dovrebbe essere alimentato con VDD_X, come mostrato in figura:



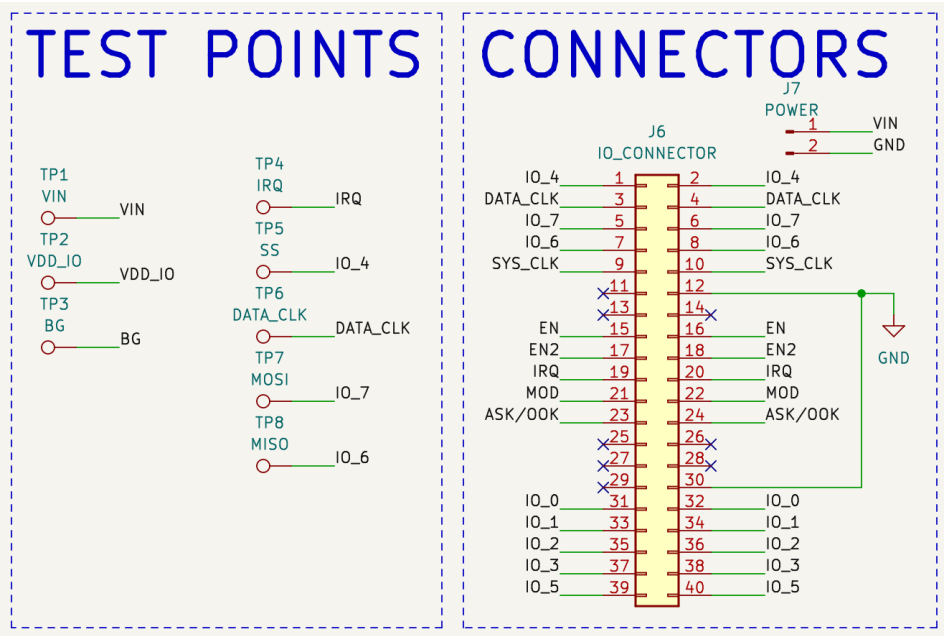
In alternativa, è possibile non collegare assieme VDD_X e VDD_IO e collegare VDD_IO all'alimentazione del microntrollore.
La soluzione di avere VDD_X e VDD_IO collegate assieme è la migliore. Si ha un'unica alimentazione e non è necessario porre attenzione al "power sequencing": se il micro viene alimentato prima del TRF7970 e ci sono dei segnali che vanno dal micro al TRF7970 che sono a valore alto, c'è il rischio che il TRF7970 venga alimentato dai diodi di protezione ESD che sono presenti sugli ingressi.



Oltre alla possibile accensione non voluta c'è anche il rischio che il diodo di protezione si danneggi e causi malfunzionamenti ai pin di ingresso.

- La massima tensione di uscita del regolatore di VDD_X è di 3.4V: avendolo connesso a VDD_IO siamo quindi forzati ad alimentare il microcontrollore che pilota il TRF7970A a 3.3V

2.4 Test points + Connectors

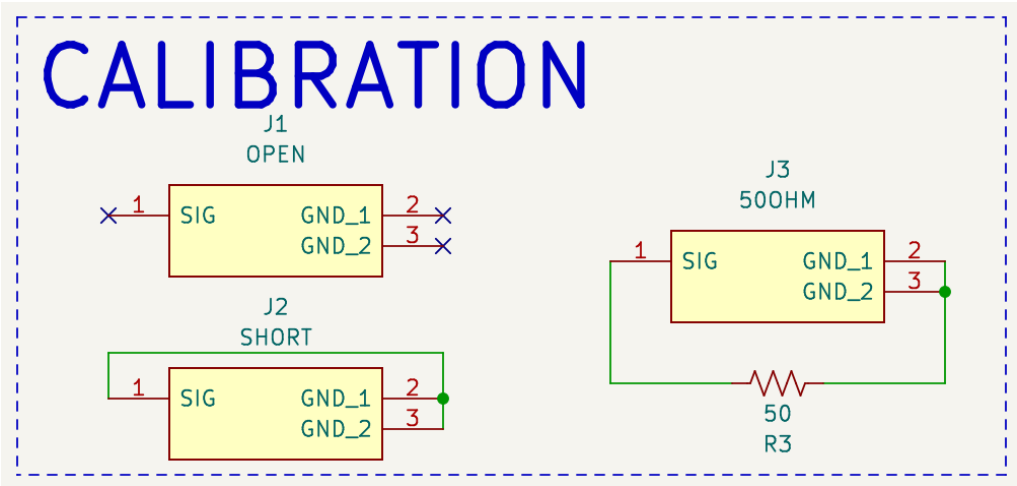


Ho inserito dei test points per poter controllare diversi punti del circuito.
Ho aggiunto un connettore a 40pin IDE per avere a disposizione tutti i pin “logici” del dispositivo. Ho usato questo connettore perché è lo stesso usato da una board FPGA che ho in casa (all’inizio pensavo che i micro che avevo fossero troppo “lenti” per le prove che volevo fare).

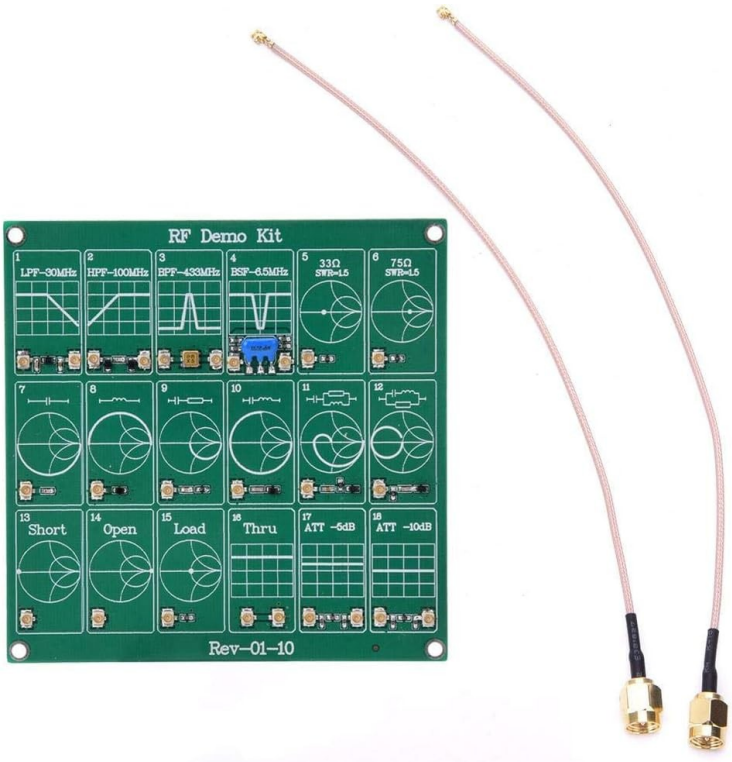
EVENTUALI MODIFICHE

- Vanno aggiunti dei test points per VDD_A e VDD_RF in maniera tale da poter controllare facilmente tutte le tensioni del circuito.
- Si può usare un connettore più piccolo solo per i pin che servono, quelli dell’SPI (IO_4, DATA_CLK, IO_7, IO_6) i pin di enable (EN, EN2) e il pin di interrupt IRQ.

2.5 Calibrazione

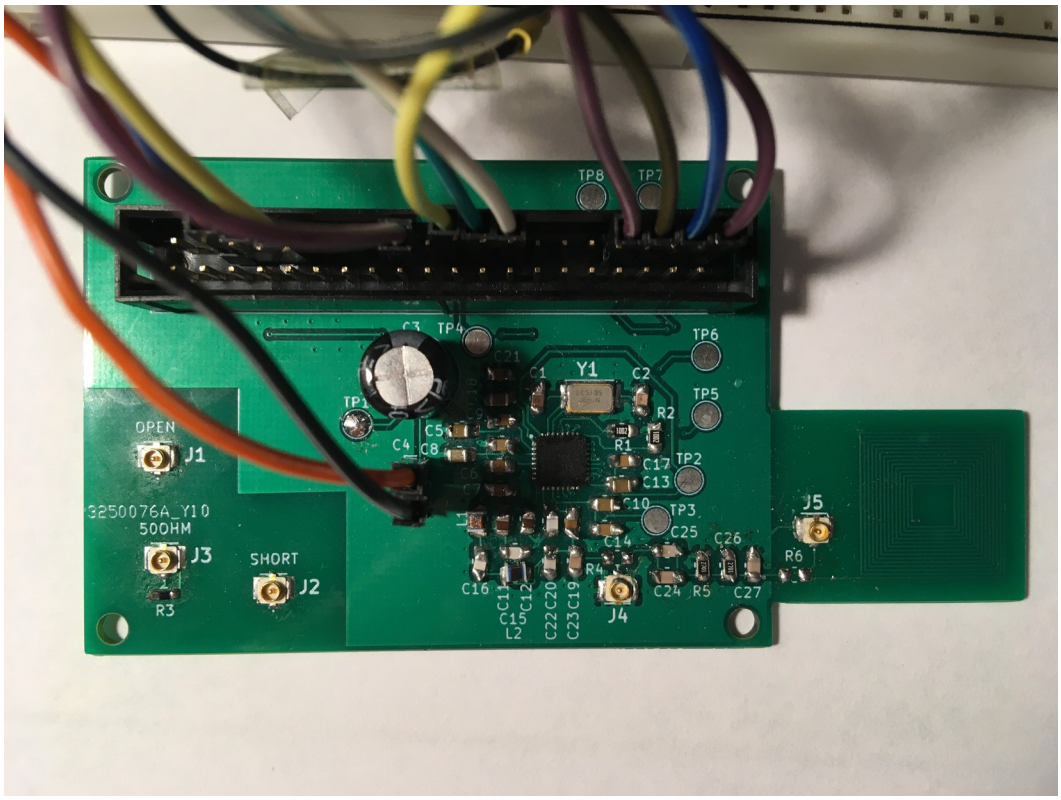


Per la calibrazione del NanoVNA ho aggiunto dei connettori U.FL (già mostrati nella sezione “Stadio RF”) in una “isola” della scheda separata dal circuito principale senza piano di massa. In alternativa si può acquistare questa scheda https://www.amazon.it/attenuatore-lapprendimento-dellanalizzatore-vettoriale-calibrazione/dp/B08DY9Y535/ per la calibrazione. Nel prodotto sono inclusi anche dei connettori SMA → U.FL che ci servono anche per la misura dell’antenna e per la sua rete di adattamento di impedenza.



2.6 Foto della board

La board finale con i componenti saldati è così:



Non pubblico il layout completo perché è stato fatto male e di fretta; consiglio di seguire le indicazioni presenti nel datasheet e in questo documento https://www.ti.com/lit/an/sloa139a/sloa139a.pdf

Ho usato dei componenti SMD 0805: sono un po' troppo grandi rispetto al passo dei pin del TRF7970A e questo complica un po' il layout. Sarebbe meglio utilizzare dei componenti più piccoli.

3 NanoVNA

Per misurare l'impedenza d'antenna usiamo il NanoVNA, un network analyzer molto potente a basso costo che si può acquistare su Amazon.

Ci sono diverse versioni del dispositivo, io ho acquistato questa https://www.amazon.it/Seesii-Analizzatore-vettoriale-Nanovna-H-Stazionaria/dp/B0B93FNW27/?th=1 ma credo che qualsiasi versione vada bene e le istruzioni per la calibrazione e la misura dell'impedenza d'antenna sono più o meno le stesse.

All'accensione il display del NanoVNA sarà così:

Dobbiamo rimuovere tutte le tracce tranne la S11. Per fare questo premiamo la rotella in alto a destra del dispositivo, selezioniamo **DISPLAY** → **TRACE** e deselezioniamo tutte le tracce (premendole due volte) tranne la **TRACE 0**.

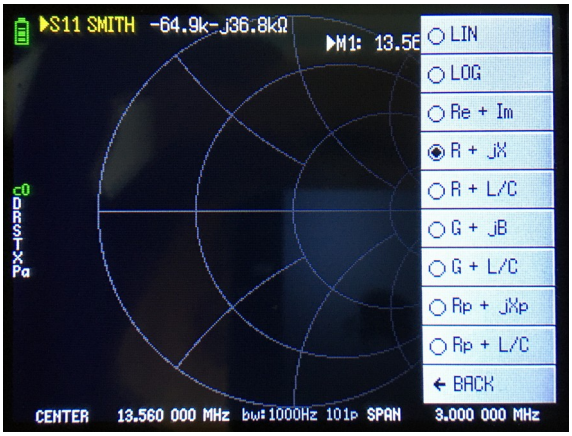
3.1 Calibrazione dello strumento

Prima di effettuare la misura dell'impedenza di antenna è necessario calibrare lo strumento. È importante che la calibrazione venga effettuata con il connettore SMA → UF.L collegato, dato che andremo a fare la misura con quello.

Come ho già accennato nella sezione 2.5 Calibrazione, possiamo usare delle schede dedicate con i componenti necessari per la calibrazione oppure possiamo usare direttamente la parte della nostra scheda che abbiamo dedicato per questo scopo.

Prima di calibrare lo strumento dobbiamo impostare il range di frequenze da visualizzare. Premiamo la rotella in alto a destra e selezioniamo **STIMULUS**. Premiamo **CENTER** e inseriamo 13.56 M, a questo punto rifacciamo l'operazione ma al posto di **CENTER** selezioniamo **START** e inseriamo 12 M, poi STOP e inseriamo 15 M.

Prima di fare la calibrazione, impostiamo il display per visualizzare la carta di Smith: premiamo la rotella in alto a destra e selezioniamo **DISPLAY** → **FORMAT S11** → **SMITH**. A questo punto dobbiamo premere un'altra volta **SMITH** e selezionare **R + JX**



Ora possiamo calibrare il nostro strumento:

- Premiamo la rotella in altro a destra e selezioniamo **CALIBRATE** → **CALIBRATE**
- Colleghiamo il cavo SMA → UF.L al connettore di calibrazione OPEN (della scheda dedicata o della nostra board)
- Premiamo **OPEN**: comparirà in alto una barra blu che sparirà una volta che il dispositivo ha concluso l'operazione.
- Colleghiamo il cavo SMA → UF.L al connettore di calibrazione SHORT.
- Premiamo **SHORT**.
- Colleghiamo il cavo SMA → UF.L al connettore di calibrazione 50OHM (LOAD per chi ha la scheda dedicata).
- Premiamo **LOAD**.
- A questo punto possiamo salvare i parametri di calibrazione in uno degli slot disponibili.

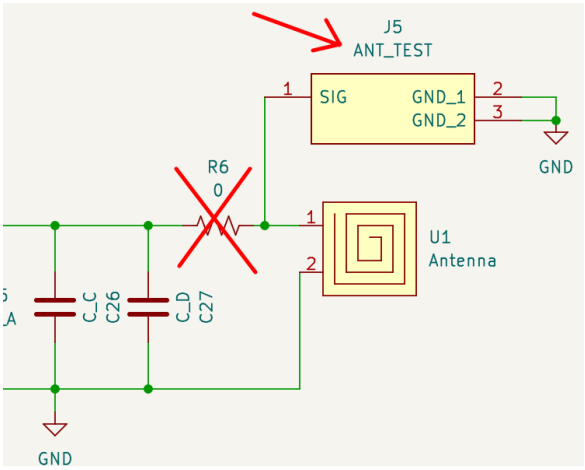
Dopo la calibrazione, le misure con 50OHM – OPEN – SHORT saranno così:

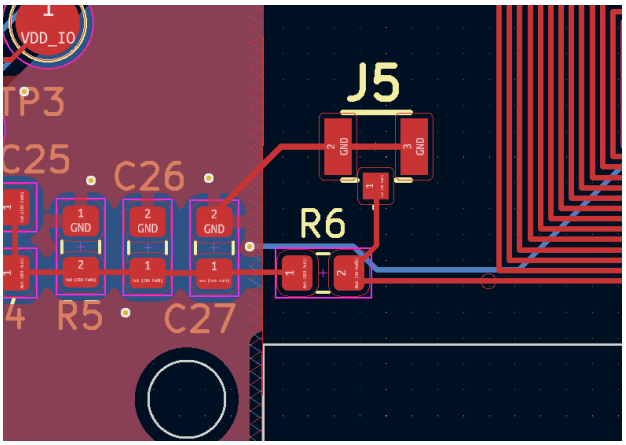
Se volete vedere una procedura dettagliata di calibrazione, ci sono diversi video su youtube che mostrano la procedura (ad esempio https://www.youtube.com/watch?v=QJYeFpiqY8c).

Ora che lo strumento è calibrato possiamo misurare l'impedenza di antenna.

3.2 Misura dell'impedenza d'antenna

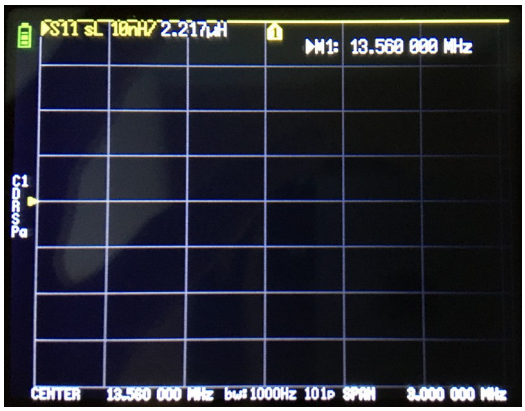
Per misurare l'impedenza d'antenna dobbiamo rimuovere la resistenza R6 dello schema e collegare il cavo di misura al connettore J5.





Questo è quello che ho misurato con la mia board:

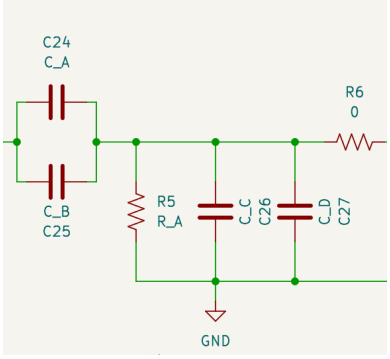
L'antenna ha un'impedenza Z = 3.6Ω + J189Ω. Possiamo anche misurare il valore di induttanza premendo la rotella in altro a destra e poi selezionare DISPLAY → FORMAT S11 → MORE → MORE → SERIES L



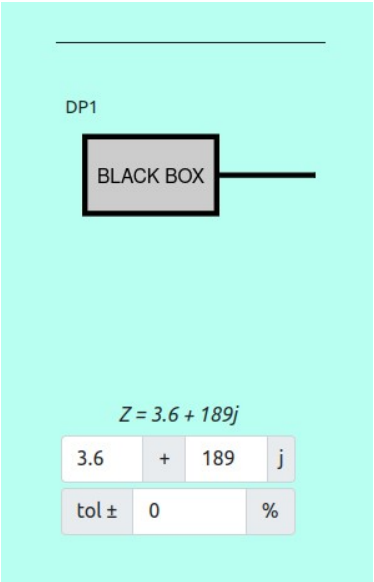
L'antenna ha un'induttanza di 2.2uH, superiore al range consigliato da TI nell'application note https://www.ti.com/lit/an/sloa241c/sloa241c.pdf. Con un'antenna progettata con un tool specifico (e non "imitando" l'antenna di una chiavetta come ho fatto io) si ottengono sicuramente dei risultati più vicini a quelli consigliati.

3.3 Dimensionamento della rete di adattamento

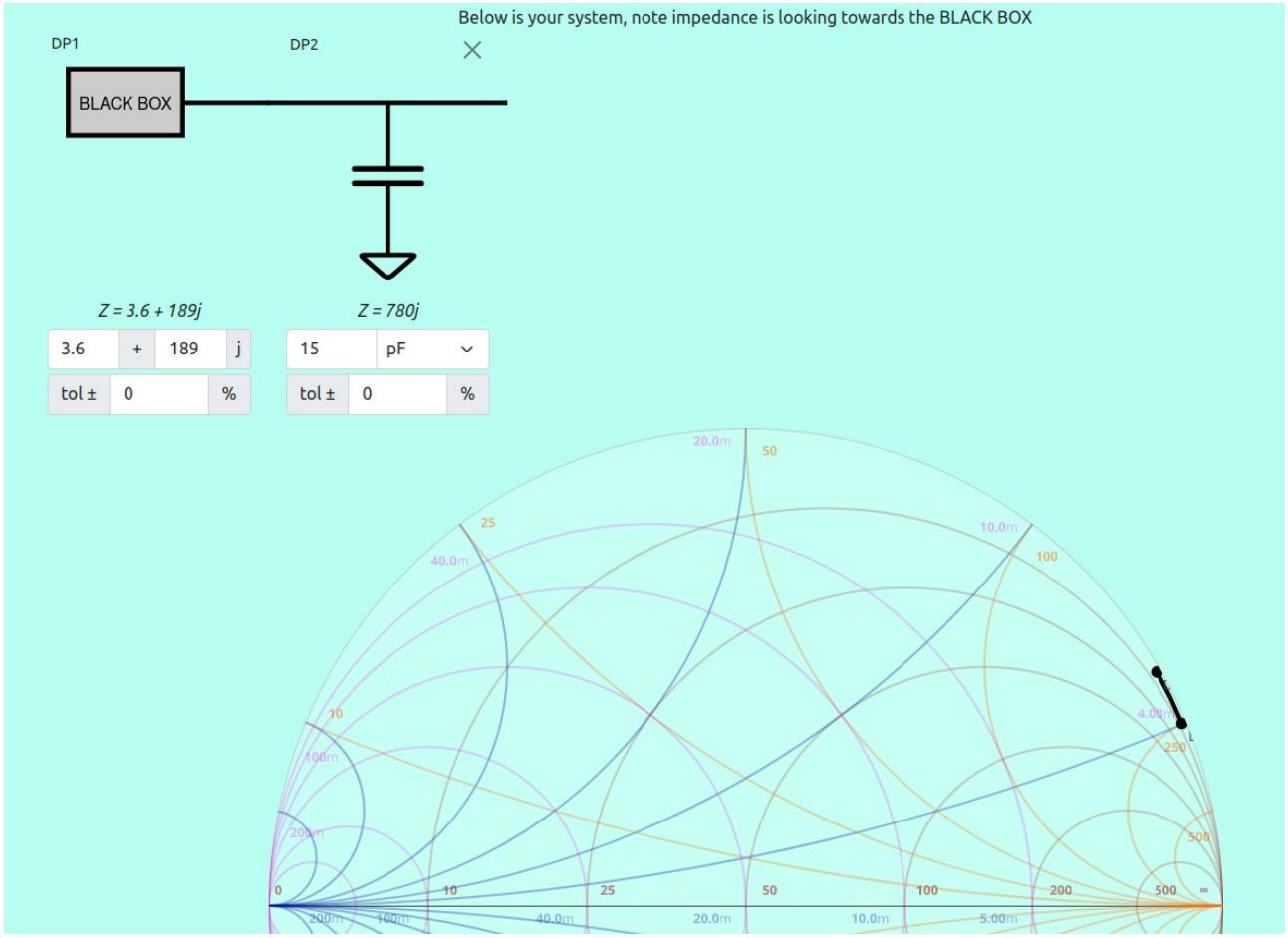
Per dimensionare la rete di adattamento possiamo seguire le istruzioni presenti nell'application note https://www.ti.com/lit/an/sloa241c/sloa241c.pdf. L'obiettivo è quello di portare al centro della carta di Smith la nostra impedenza d'antenna, usando la rete di adattamento prevista:



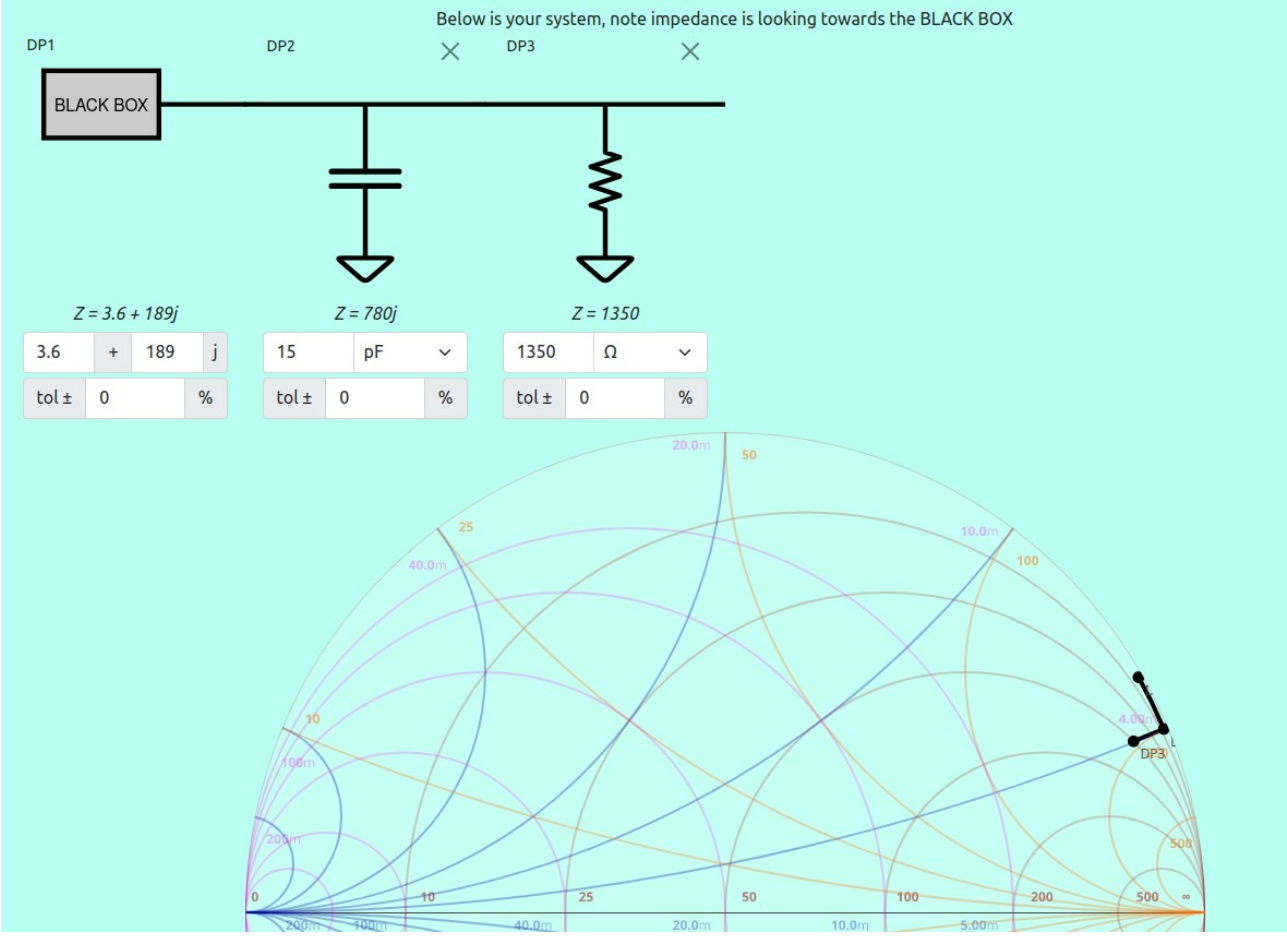
Ho usato questo sito web https://www.will-kelsey.com/smith_chart/ per calcolare il valore dei componenti da usare (ci sono anche dei software disponibili).
Per prima cosa inseriamo il valore dell'impedenza d'antenna ottenuto precedentemente e inseriamo la frequenza 13.56MHz nel campo Frequency:



Selezioniamo “Parallel Capacitor” dall’elenco dei componenti e cerchiamo quel valore che ci porta sulla curva a reattanza costante 250Ω.

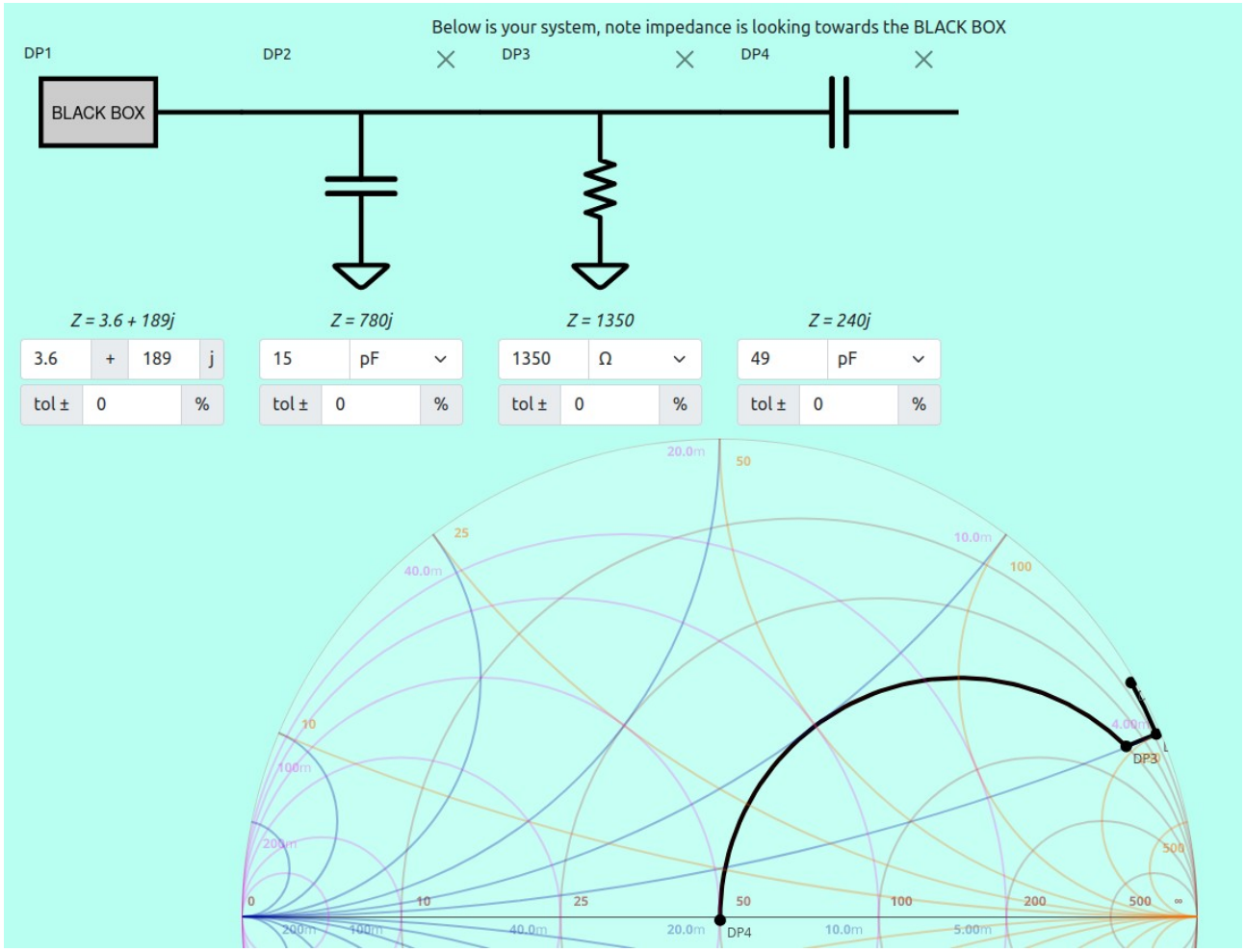


Selezioniamo “Parallel Resistor” dall’elenco dei componenti e cerchiamo quel valore che ci porta sul cerchio a resistenza costante 50Ω :



Seguendo le formule prese dall’application note ricaviamo che Q = R / JX = 1350 / 189 = 7, come consigliato da TI. Sarebbe stato possibile anche procedere a posteriori: si calcola il valore di R per avere il Q desiderato e poi si trova il valore del condensatore in parallelo che ti permette di arrivare al cerchio a resistenza costante 50Ω con la R desiderata.

Selezioniamo “Series Capacitor” dall’elenco dei componenti e cerchiamo quel valore che ci porta al centro della carta di Smith :

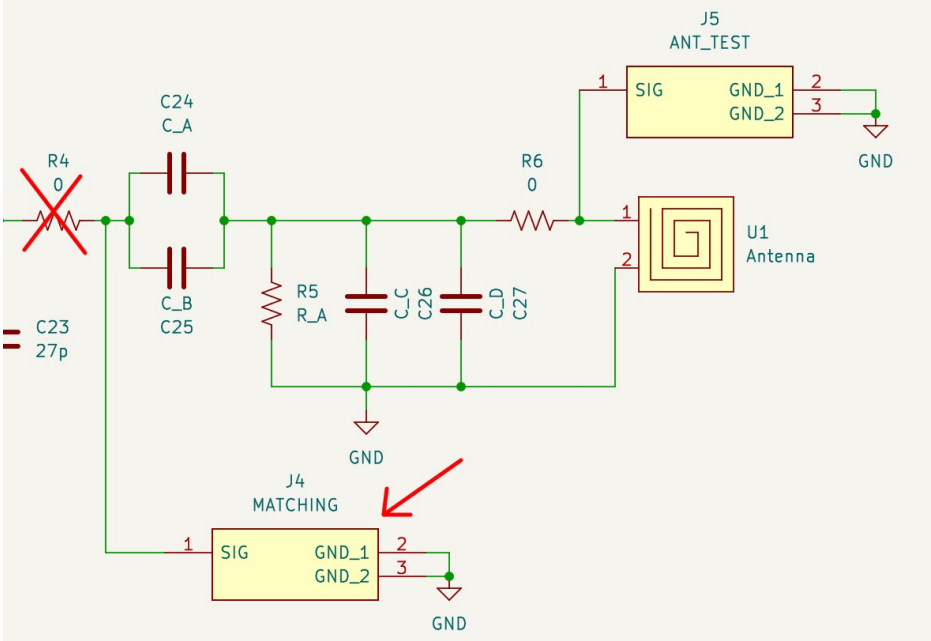


La resistenza da 1350Ω può essere realizzata come il parallelo di due resistenze da 2700Ω e il condensatore da 49pF può essere realizzato come il parallelo di 47pF e 2pF.
È consigliato usare componenti “precisi” per la rete di adattamento (TI consiglia componenti all’1%).

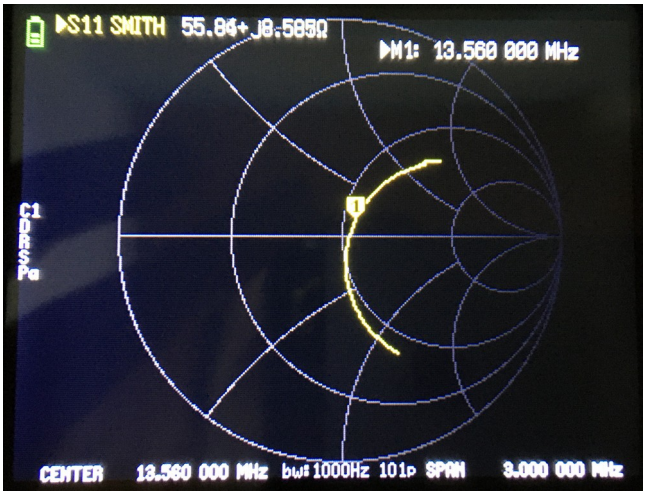
3.3.1 Misura della rete di adattamento

Per misurare la rete di adattamento dobbiamo:

- Inserire la resistenza R6 da zero Ohm.
- Rimuovere la resistenza R4.
- Collegare il cavo di misura al connettore J4.

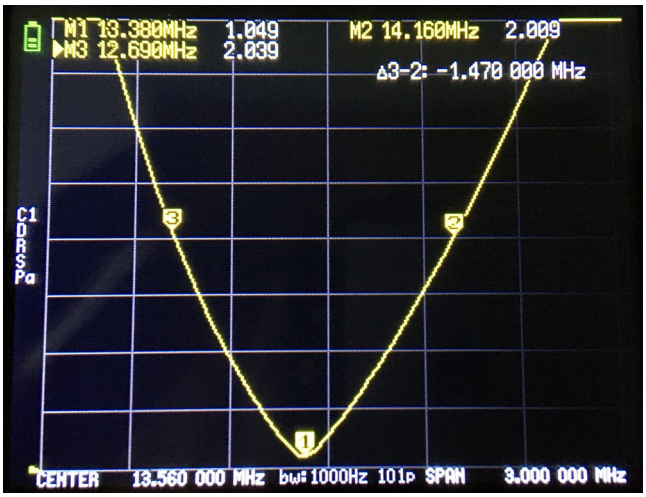


La misura ha dato questi risultati:



Il punto non è esattamente al centro: probabilmente gli errori di misura e le tolleranze dei componenti (tutti acquistati su aliexpress) non hanno aiutato. È comunque possibile modificare i componenti sulla board per cercare i centrarlo il più possibile.

È possibile misurare il rapporto di onda stazionaria (SWR): premiamo la rotella in alto a destra e selezioniamo **DISPLAY** → **FORMAT S11** → **SWR**. Impostando dei marker opportuni si può selezionare il punto in cui l'SWR è uguale a uno (adattamento perfetto) e i punti a 2dB per calcolare il Q:



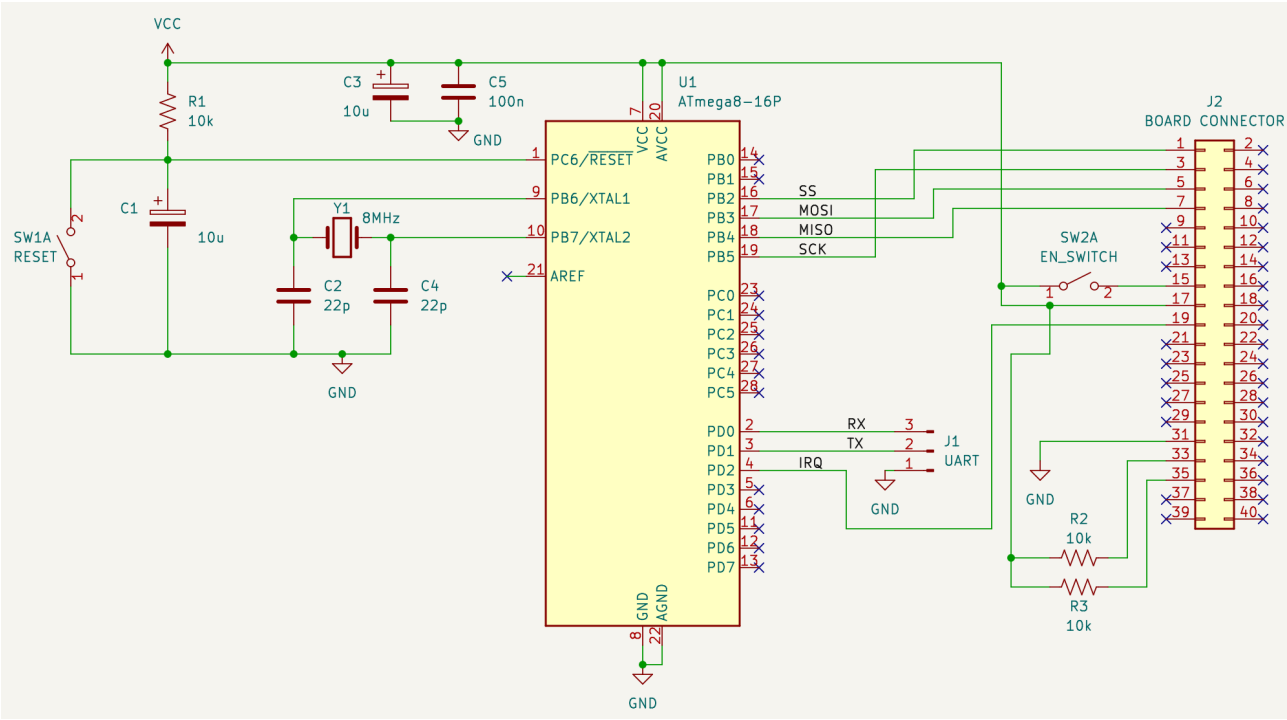
L'adattamento perfetto ce l'ho a 13.38MHz invece che i 13.56MHz desiderati, il Q è

$$Q = \frac{F_c}{BW} = \frac{13.38}{1.47} = 9$$

Il Q ottenuto è all'interno del range consigliato da TI (*For most applications, a Q factor of 7 to 10 is suitable when the application includes the use of ISO14443A (NFC Type A), ISO14443B (NFC Type B), or FeliCa™ (NFC Type F) technologies*).

4 TestBoard

Per il test della scheda con il TRF7970A ho realizzato una board di test con un ATMEGA8.



La board in se è molto semplice:

- Ho usato un ATMEGA8 ma va bene qualsiasi microcontrollore che ha una porta SPI.
- Per comunicare con il PC ho usato una board FTDI connessa al connettore UART indicato nello schema: RX del micro va collegato al pin TX della board FTDI e il pin TX del micro va collegato al pin RX della board FTDI.
- Ho usato uno switch (EN_SWITCH) per controllare il pin EN del TRF7970A ma può benissimo essere usato un pin del micro (consiglio di fare questo).
- La board va alimentata a 3.3V e l'alimentazione deve essere la stessa della board con il TRF7970A.

NOTA: non ho inserito nello schema il connettore a 10pin per la programmazione tramite USBASP (i collegamenti sono comunque quelli standard che si trovano sugli schemi in rete).

4.1 SPI Test

Per testare la comunicazione SPI e fare dei semplici test con la board ho realizzato un terminale in C per la comunicazione SPI. Il codice per l'ATMEGA8 e per il terminale li trovate qui https://github.com/Ptr-srix4k/SRIX4K-TRF7970A-Emulator

NOTA: le successive istruzioni per la compilazione dei software sono per Linux, dato che ho fatto tutto su Ubuntu. Si possono però facilmente compilare anche per Windows.

Per realizzare il terminale ho usato le librerie FTDI: la guida per l'installazione dei driver FTDI per Linux la trovate qui.
Una volta che la board FTDI è collegata al PC, vanno disabilitati i driver VCP perché sono incompatibili con i driver D2XX. Per fare ciò, usiamo questi comandi:

```
sudo rmmod ftdi_sio
sudo rmmod usbserial
```

Cloniamo la repository su disco:

```
git clone https://github.com/Ptr-srix4k/SRIX4K-TRF7970A-Emulator.git
```

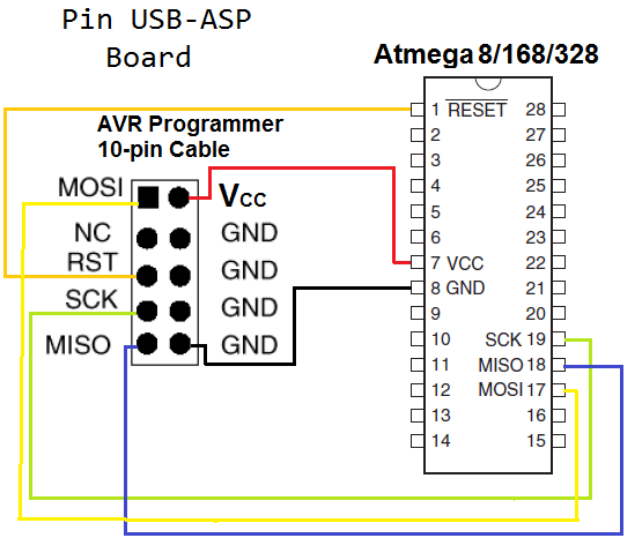
Entriamo nella cartella SPI_test:

```
cd SRIX4K-TRF7970A-Emulator/SPI_test
```

Compiliamo il terminale SPI:

```
gcc -o spi_terminal spi_terminal.c -L. -lftd2xx
```

Per la compilazione e la scrittura del firmware sull'ATMEGA vengono usati AVR-GCC (le guide per installare la toolchain sui vari sistemi operativi le trovate facilmente su internet) e il programmatore USBASP.
Il collegamento del programmatore all'ATMEGA avviene in questo modo:



Per compilare il codice, entrate nella cartella firmware (presente dentro SRIX4K-TRF7970A-Emulator/SPI_test) e utilizzate la seguente istruzione

```
avr-gcc -Wall -g -Os -mmcu=atmega8 -std=gnu99 -o main.bin main.c
```

Per generare il file .hex

```
avr-objcopy -j .text -j .data -O ihex main.bin main.hex
```

Prima di scrivere il file .hex è necessario scrivere i fuse bit (NOTA: questa operazione è da fare una volta sola, non si deve ripetere ogni volta che si scrive il .hex)

```
avrdude -p atmega8 -c usbasp -U lfuse:w:0xDF:m
avrdude -p atmega8 -c usbasp -U hfuse:w:0xD9:m
```

Per scrivere il file .hex nel microcontrollore

```
avrdude -p atmega8 -c usbasp -U flash:w:main.hex:i -F -P usb
```

Ora possiamo alimentare la TestBoard. La sequenza è la seguente:

- Dare alimentazione alla TestBoard (3.3V) tenendo il pin EN a zero e la board FTDI disconnessa dalla board.
- Chiudere lo switch SW2A per impostare EN a uno e quindi abilitare il TRF7970A.
- Collegare alla TestBoard e al PC la board FTDI.

Bisogna usare questa sequenza perché se si alimenta la TestBoard con la scheda FTDI connessa, quest'ultima viene alimentata tramite i suoi diodi ESD dal pin TX del microcontrollore (che parte con TX alto) e ho notato che se succede questo, la comunicazione seriale presenta degli errori.
Il problema si può risolvere se si alimenta la board FTDI non da USB ma direttamente con l'alimentazione della TestBoard.

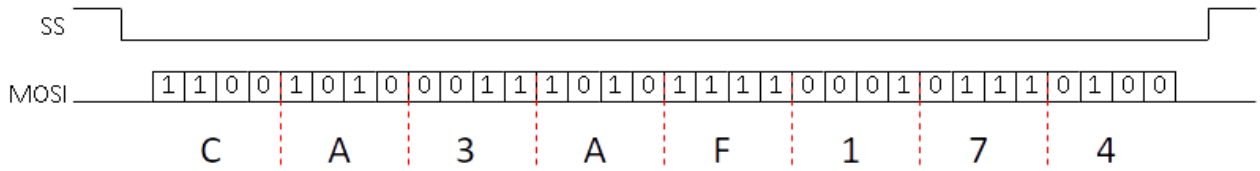
Se eseguiamo spi_terminal con la board FTDI è collegata, troveremo questo:

```
Library version = 0x10422
FT_Open succeeded.  Handle is 0x5599b0858ba0
FT_GetDeviceInfo succeeded.  Device is type 5.
>
```

A questo punto è possibile inviare direttamente dei comandi SPI alla DemoBoard.
I comandi SPI devono essere inseriti in esadecimale: ad esempio, se si inserisce il comando

```
> CA3AF174
```

Verrà trasmesso questo:



I byte ricevuti dal microcontrollore sulla linea MISO verrano viasualizzati sul terminale dopo aver inviato il comando.
I comandi SPI per il TRF7970A devono tutti iniziare con un byte speciale, che contiene il tipo di comando da inviare e l'indirizzo del registro:

Table 6-20. Address and Command Word Bit Distribution

BIT	DESCRIPTION	BIT FUNCTION	ADDRESS	COMMAND
B7	Command control bit	0 = Address 1 = Command	0	1
B6	Read/Write	0 = Write 1 = Read	R/W	0
B5	Continuous address mode	1 = Continuous mode	R/W	0
B4	Address/Command bit 4		Adr 4	Cmd 4
B3	Address/Command bit 3		Adr 3	Cmd 3
B2	Address/Command bit 2		Adr 2	Cmd 2
B1	Address/Command bit 1		Adr 1	Cmd 1
B0	Address/Command bit 0		Adr 0	Cmd 0

Inviamo un comando di “Software Initialization (0x03)” per fare un reset del TRF7970A:

```
> 83
```

In questo comando il bit B7 (Command control bit) è a uno per indicare che non si tratta di un comando che legge/scrive dei registri, ma un comando che esegue delle funzioni dedicate sul chip.

A questo punto possiamo fare una lettura in “continuous address mode” (permette di leggere più registri in sequenza con un solo comando SPI) partendo dal registro 0x00, per vedere se la comunicazione è corretta controllando i valori di default dei registri:

```
> 6000000000000000
> 0 1 21 0 0 c1 c1 0
```

Nel datasheet sono elencati i valori di reset dei registri:

Table 6-21. Register Values After Sending Software Initialization (0x03)

ADDRESS	REGISTER	VALUE
0x00	Chip status control	0x01
0x01	ISO control	0x21 (1)
0x02	ISO/IEC 14443 B TX options	0x00
0x03	ISO/IEC 14443 A high bit rate options	0x00
0x04	TX timer high byte control	0xC1 (1)
0x05	TX timer low byte control	0xC1 (1)

I valori sono quelli che vediamo nel risultato del comando, segno che la comunicazione SPI è corretta e il chip sta rispondendo correttamente ai comandi inviati.

4.1.1 Test di ricezione di un comando per SRIX4K

Prima di iniziare a ricevere i comandi da un lettore RFID, dobbiamo impostare alcuni registri:

- Il registro 0x00 (Chip Status) va scritto a 0x20 per selezionare la “3-V operation” e per impostare a uno il bit “rf_on”

```
> 0020
```

- Il registro 0x09 (Modulator and SYS_CLK Control Register) va scritto a zero per disabilitare il cristallo a 27MHz e impostare un valore di “Modulation Depth” come ASK 10%

```
> 0900
```

- Il registro 0x01 (ISO Control Register) va scritto a 0x25 per impostare a uno il bit rfid (NFC or card emulation mode), il bit iso_2 (Card Emulation Mode) e la modalità “ISO/IEC 14443 B”

```
> 0125
```

- La FIFO va resettata inviando il comando 0x0F (Reset FIFO)

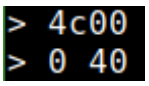
```
> 8F
```

- Il ricevitore va abilitato inviando il comando 0x17 (Enable Receiver)

```
> 97
```

A questo punto avviciniamo un lettore RFID all’antenna della board e inviamo con il lettore il comando INITIATE. La sequenza per capire se il comando è stato ricevuto correttamente è la seguente:

- Leggere il registro 0x0C (IRQ Status Register) e dobbiamo trovarci il bit Irg_srx (IRQ set due to RX start) alto (NOTA: il registro ritorna a zero una volta che viene letto):



  Il valore che leggiamo (0x40) è in linea con quanto atteso.
- Leggere il registro 0x1C (FIFO Status Register) che contiene il numero di byte che sono stati ricevuti dalla FIFO:

```
> 5c00
> 0 2
```

Leggiamo che sono stati ricevuti 2 byte, il comando INITIATE è infatti composto da due byte.

- Leggere il contenuto della FIFO dal registro 0x1F (FIFO I/O register) usando una continuous address mode:

```
> 7F000000
> 0 6 0 0
```

Il dato ricevuto è 0x06 0x00 ed è proprio il comando INITIATE inviato dal lettore RFID.

Ogni volta che si legge qualcosa dalla FIFO, il contatore che si legge nel registro 0x1C viene sempre decrementato di uno (se la FIFO ha 2 byte e ne leggo 3 come nel comando appena inviato, il registro avrà valore 0x7F, dato che la FIFO ha 127 Byte di memoria).

Dopo una ricezione, la FIFO va resettata.
Se provo ad inviare un altro comando, il registro 0x0C mi segnala che ho ricevuto qualcosa, ma la FIFO segnala 0 byte ricevuti.
Non so se sia un bug del dispositivo oppure è solo un problema legato alla mia board, ma ho notato che se voglio ricevere un altro comando, devo prima disabilitare il ricevitore e poi ri-abilitarlo. Forse il chip si aspetta che dopo ogni ricezione ci sia sempre una trasmissione (per cui il ricevitore deve essere sempre disabilitato).

Un'altra stranezza che ho notato è il comportamento del comando 0x18 *Test Internal RF (RSSI at RX Input With TX ON)*. Questo comando misura il livello della portante RF ai pin RF_IN1 e RF_IN2 con il trasmettitore accesso e lo va a scrivere nel registro 0x0F. L'esecuzione di questo comando non ha alcun effetto sul registro 0x0F che rimane sempre a zero, come se non ci fosse nessuna portante sui pin RF.
L'esecuzione del comando 0x19 *Test External RF (RSSI at RX Input with TX OFF)* che testa la ricezione del segnale con il trasmettitore spento e scrive il valore nel registro 0x0F, sembra funzionare correttamente (nel registro 0x0F misuro il massimo valore su RF_IN1 e zero su RF_IN2, se tolgo la portante misuro zero su entrambi).
Anche in questo caso non so se è un problema legato alla mia board o è il TRF7970A che si comporta in maniera strana (molti comportamenti non sono documentati correttamente).

Ora abbiamo visto la ricezione di un comando, nel capitolo successivo vedremo come realizzare un emulatore "completo".

5 Emulatore SRIX4K

Per realizzare l'emulatore completo, usiamo la TestBoard descritta nel capitolo precedente. Per scrivere il firmware dell'emulatore sull'ATMEGA, entriamo nella cartella Emulator

```
cd SRIX4K-TRF7970A-Emulator/Emulator
```

Per compilare il codice, utilizzate la seguente istruzione

```
avr-gcc -Wall -g -Os -mmcu=atmega8 -std=gnu99 -o main.bin main.c
                        usart.c spi.c
```

Per generare il file .hex

```
avr-objcopy -j .text -j .data -O ihex main.bin main.hex
```

Prima di scrivere il file .hex è necessario scrivere i fuse bit (NOTA: questa operazione è da fare una volta sola, non si deve ripetere ogni volta che si scrive il .hex)

```
avrdude -p atmega8 -c usbasp -U lfuse:w:0xDF:m
avrdude -p atmega8 -c usbasp -U hfuse:w:0xD9:m
```

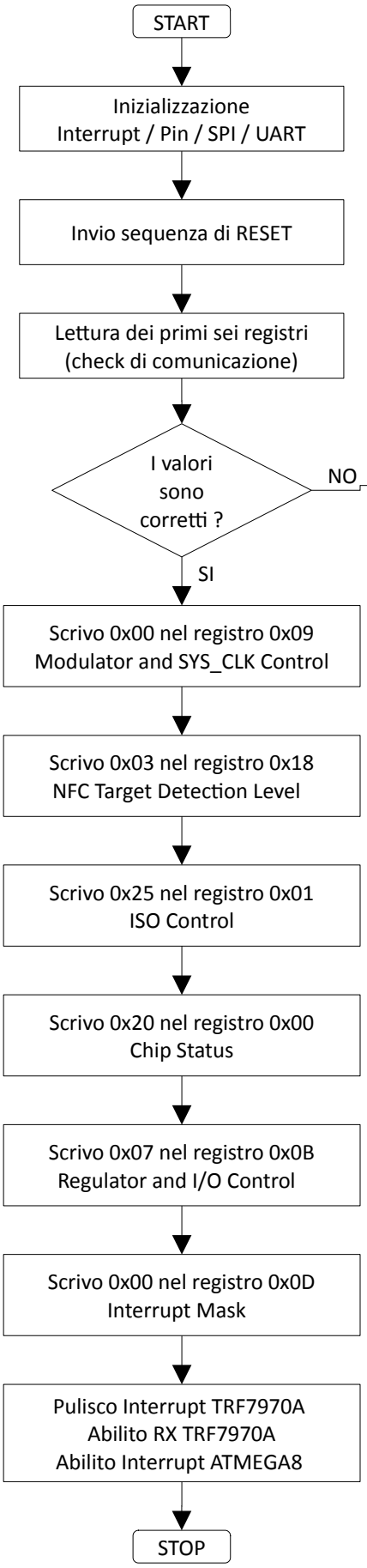
Per scrivere il file .hex nel microcontrollore

```
avrdude -p atmega8 -c usbasp -U flash:w:main.hex:i -F -P usb
```

Nella cartella sono presenti i seguenti file:

- main.c → Codice principale dell'emulatore
- spi.c → Contiene le funzioni utilizzate per la comunicazione SPI
- usart.c → Contiene le funzioni utilizzate per la comunicazione UART di debug
- TRF7970A.h → Contiene le definizioni di tutti i registri del TRF7970A
- key.h → Contiene la memoria della SRIX4K

Il codice è molto semplice ed è composto da una routine di interrupt (che gestisce comunicazione e trasmissione) e nel main() è presente l'inizializzazione dei registri del TRF7970A.
Nella pagina seguente è presente il diagramma di flusso della funzione main():



La sequenza di inizializzazione è descritta nei primi tre punti della "**TRF7970A Initialization**" presente nel datasheet a pagina 48

Se i valori non sono corretti, vengono inviati sulla seriale i valori letti e il messaggio "Error: Init failed"

Imposto il valore corretto della frequenza di clock e il modulation depth (ASK 10%)

Imposto a 250mV la soglia per il Wake Up automatico
(non è necessario dato che forniamo noi il pin EN dall'esterno)

Imposto la "card emulation mode" e la modalità ISO/IEC 14443 B

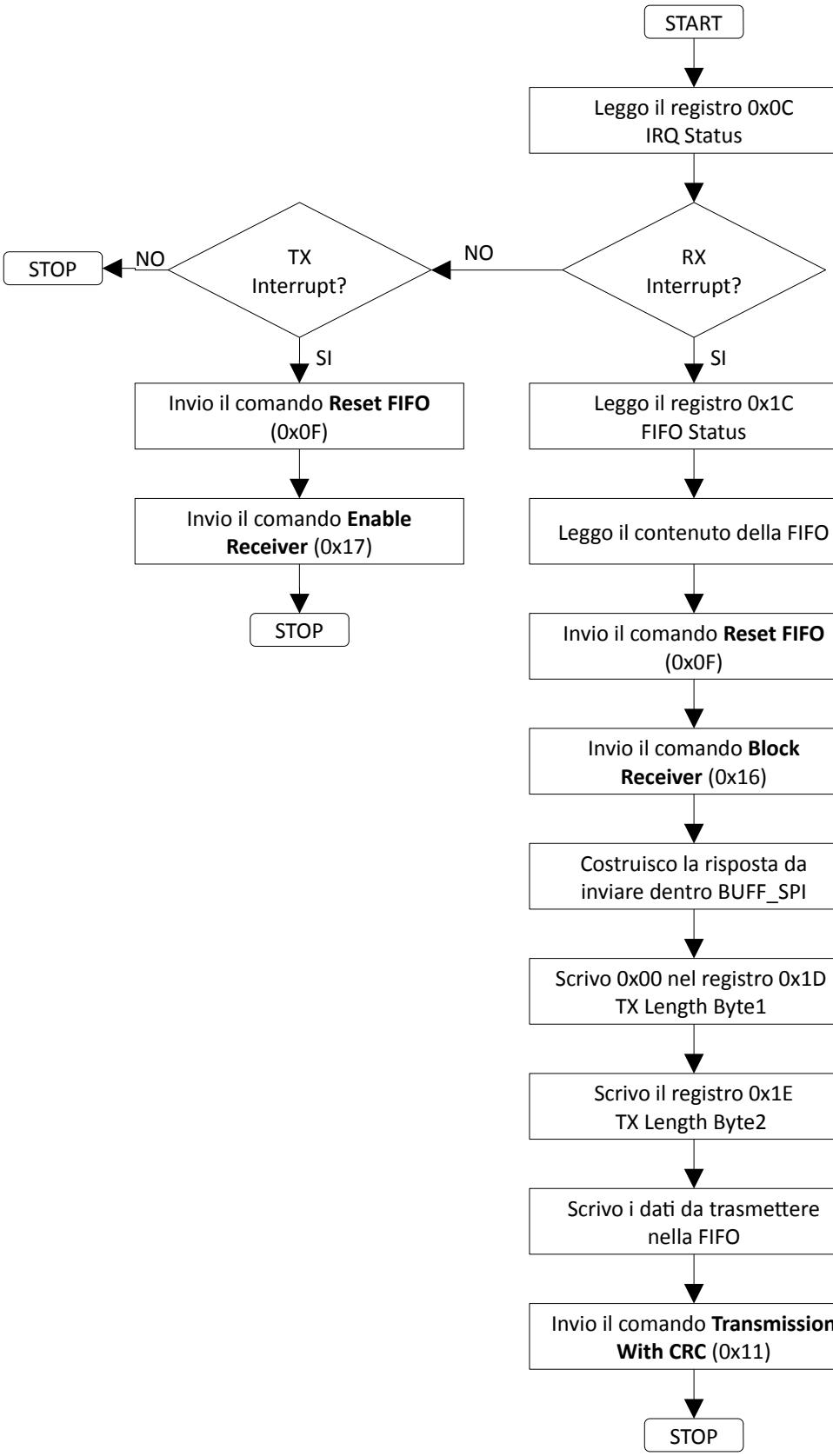
Abilito il trasmettitore e il ricevitore del TRF7970A e la modalità 3V

Imposto in maniera manuale le tensioni V_{DD_RF} / V_{DD_A} / V_{DD_X}
(non necessario)

Disabilito tutte le sorgenti di interrupt del dispositivo (a parte RX start e TX end che non si possono disabilitare).

Viene usata la UART per comunicare al PC la corretta inizializzazione del dispositivo (la comunicazione seriale è a 9600 baud, 8bit, 1 stop bit). È importante notare che in questo caso la board FDTI viene usata come una "vera" porta seriale per cui non bisogna disabilitare i driver VCP.
Le funzioni UART presenti nel codice possono essere usate per visualizzare messaggi di debug e controllare i dati ricevuti o trasmessi.

Ogni volta che il TRF7970A riceve dei dati o finisce di trasmetterli, il pin IRQ viene settato a uno. Questo pin, collegato al pin PD2 dell'ATMEGA8 viene usato per abilitare un interrupt che gestisce la ricezione e la trasmissione dei dati.
Nel chip è presente una FIFO da 127Byte che viene usata dalla logica che riceve e dalla logica che trasmette. La routine di interrupt legge la FIFO per determinare il comando ricevuto e carica i dati nella FIFO per trasmettere la risposta.
La routine in interrupt è mostrata nella pagina seguente:



B7 → Irq_tx → Lo leggo a uno alla fine di ogni trasmissione

B6 → Irq_srx → Lo leggo a uno alla fine di ogni ricezione

Controllo il numero di byte ricevuti

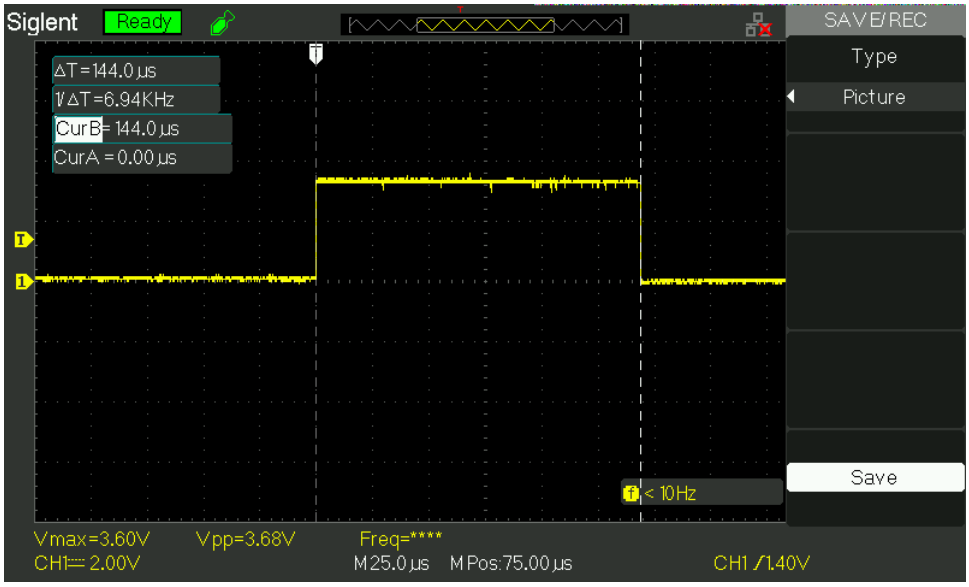
La FIFO deve essere resettata prima di ogni trasmissione

Il ricevitore deve essere disabilitato altrimenti la trasmissione fallisce

Vengono scritti in un'unica *continuous write* per velocizzare le operazioni

Il tempo che intercorre fra l'entrata nella routine di interrupt e l'invio del comando di trasmissione è un parametro importante: il protocollo di comunicazione con la SRIX4K prevede che vengano rispettati certi tempi (non posso perdere troppo tempo nella fase in cui codifico il comando da inviare).

Il tempo massimo misurato durante la ricezione del comando GET_UID (è il più lento perché dobbiamo trasmettere 8byte di dati) è di circa 144us.



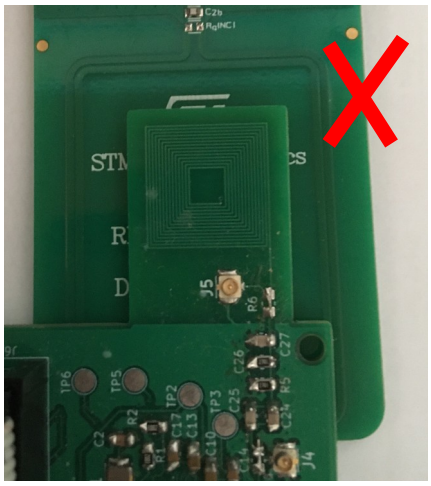
Il tempo specificato nel datasheet del SRIX4K come "Antenna reversal delay" è di 151µs, per cui non c'è nessun problema.

5.1 Risultati ottenuti

L'emulatore funziona correttamente con:

- La demoboard M24LR-DISCOVERY della ST su cui è presente il chip CR95HF
- Il PN532

La distanza di funzionamento è più corta di quella di un tag normale (bisogna stare più vicini con l'antenna al lettore) e funziona meglio solo se l'antenna "si affaccia" alla scheda del lettore e non tutta la scheda (o solo una sua parte):



Il problema della distanza di funzionamento più "corta" rispetto a quella di un tag si presenta solo per la trasmissione, per la ricezione le distanze di funzionamento sono identiche a quelle di un tag. Non so se è solo un problema della mia scheda, del fatto che l'antenna non è stata progettata nella maniera corretta o se è legato all'adattamento non ottimale fra antenna e TRF7970A.

L'emulatore NON funziona se si utilizza un lettore basato sul CRX14 !!!!
La ricezione dei dati avviene correttamente ma la trasmissione non va a buon fine. Ho provato a fare qualsiasi cosa ma non sono riuscito a farlo funzionare. Anche questo potrebbe essere un problema legato a:

- Antenna.
- Adattamento non ottimale.
- Parametri temporali del protocollo non rispettati correttamente (ad esempio, il tempo di "Antenna reversal delay" non è fisso a 151µs ma è inferiore, potrebbe causare qualche problema?).
- Funzionamento diverso del circuito di demodulazione del CRX14 rispetto agli altri chip.