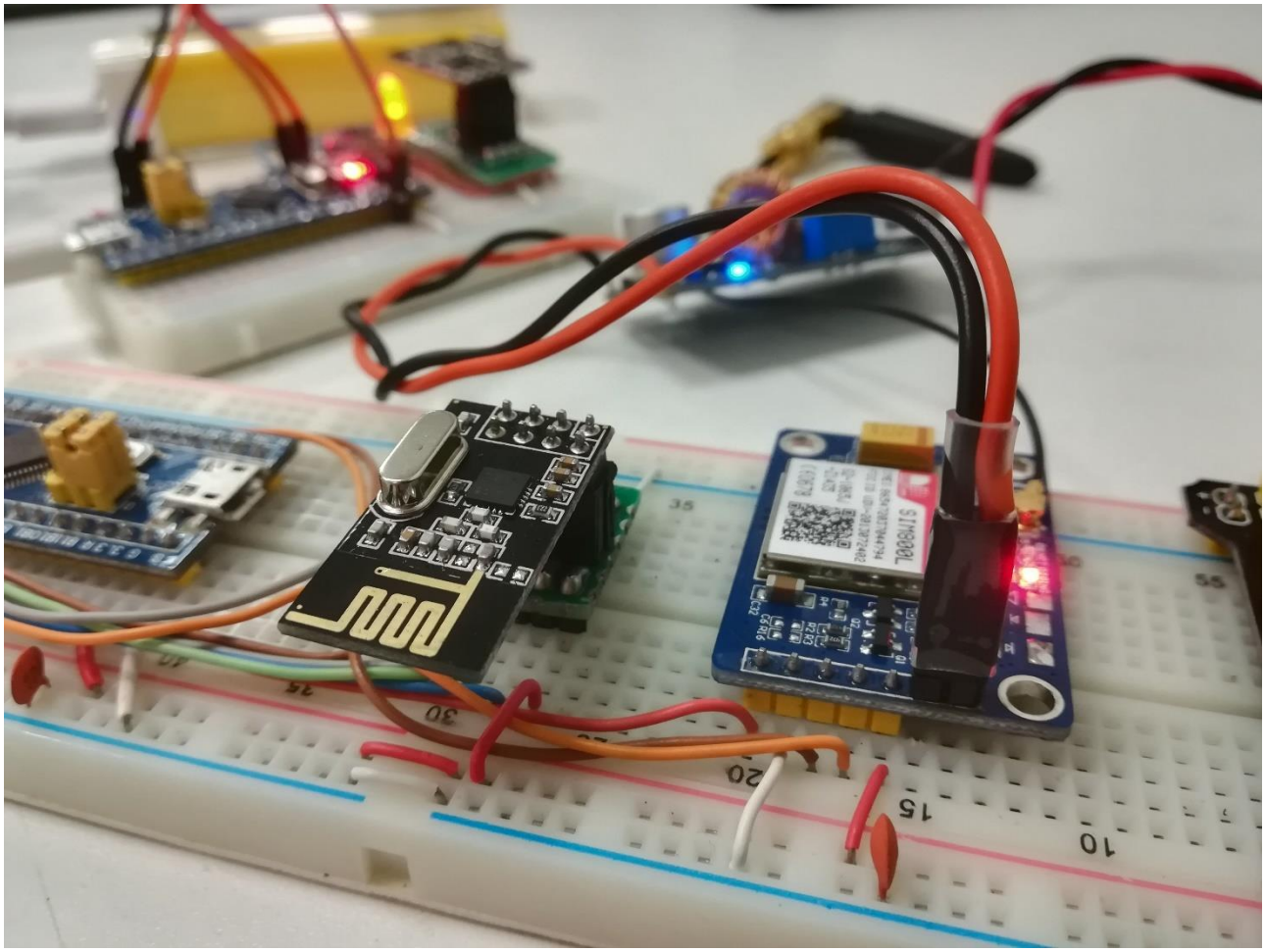
UNIVERSITA' DEGLI STUDI DI GENOVA – SCUOLA POLITECNICA – A.A. 2017/18

CORSO DI LAUREA MAGISTRALE IN INGEGNERIA ELETTRONICA

RF MICROELECTRONICS – PROF. DANIELE CAVIGLIA

Sistema di comunicazione per applicazioni domotiche



ALESSIO CALIGIURI

EMANUELE ANFUSO

Sommario

1. Introduzione

Lo scopo del progetto è sviluppare un sistema a basso costo che consenta di gestire da remoto degli apparati elettrici di un'abitazione. Un tale sistema deve essere in grado di ricevere degli SMS e, approvato il mittente sulla base di una lista di numeri autorizzati, deve poter accendere o spegnere dei dispositivi, per esempio una lampada, uno scaldabagno elettrico, ecc...

Esistono in commercio delle "prese intelligenti" che, collegandosi alla rete WiFi di casa, sono in grado di portare a termine il compito suddetto; il prezzo di questi dispositivi parte da circa 20€. Per le "prese intelligenti" dotate di una SIM card, la cifra minima richiesta sale a circa 50€.

La nostra idea di progetto si evolve nell'avere una "stazione base" dotata di connettività GSM e varie "unità remote" che vi dialogano in banda ISM a 2.4GHz. In tal modo, si possono pilotare più dispositivi in casa al prezzo di una sola SIM.

Nell'ottica di sviluppare un progetto a basso costo, si è scelto di acquistare il materiale necessario da rivenditori on-line orientali, che forniscono delle schede preassemblate (*breakout board*) su cui è montato il componente principale e l'elettronica di contorno essenziale, necessaria per il suo funzionamento. I blocchi base del nostro progetto sono:

- microcontrollore ARM Cortex-M3 della famiglia STM32 (*STMictroelectronics STM32F103C8T6*);
- modulo per connessione GSM (*SIMCom SIM800L*);
- ricetrasmettitore in banda ISM 2.4Ghz (*Nordic Semiconductor nRF24L01+*).

Per i due componenti RF sono disponibili sul web varie librerie stabili, specifiche per l'ambiente Arduino, molto diffuse e ben documentate; ciò non è vero nel mondo STM32, per il quale è stato necessario sviluppare del codice nuovo.

Più in particolare, il SIM800L si interfaccia con una seriale UART e si configura mediante comandi AT; è stato quindi necessario scrivere un *parser* per interpretare le risposte del modulo, nello specifico per quanto riguarda la ricezione di nuovi SMS.

Per il modulo nRF24L01+ è stato necessario uno studio approfondito della relativa documentazione tecnica, allo scopo di sviluppare una libreria funzionante a partire da un prototipo già esistente sul web.

2. Struttura del sistema e schemi

Il sistema di comunicazione prevede la presenza di una *stazione base* a cui si possono connettere wireless diverse *unità remote.* Il cuore di entrambi i dispositivi è il microcontrollore STM32F103C8T6 che si occupa di tutte le operazioni necessarie per la gestione della comunicazione a radiofrequenza. Questa nel caso della *stazione base* avviene tramite due modalità: il modulo SIM800 fornisce connettività GSM, sfruttata in questo progetto per lo scambio di SMS, mentre il modulo nRF24L01+ stabilisce la connessione con le *unità remote* attraverso un protocollo proprietario (*Enhanced ShockBurst™)*. Quest'ultime posseggono appunto lo stesso modulo per la comunicazione con la *stazione base*, e sono equipaggiate dell'elettronica necessaria al compimento dello specifico compito che gli viene affidato, per esempio l'azionamento di un relais o la lettura dei valori di un qualche sensore analogico o digitale.

Il prototipo del sistema di comunicazione completo comprende la *stazione base* e una sola *unità remota*, che sulla base dei comandi ricevuti può compiere diverse azioni, tra cui l'accensione o spegnimento di due LED o l'azionamento di due relais.

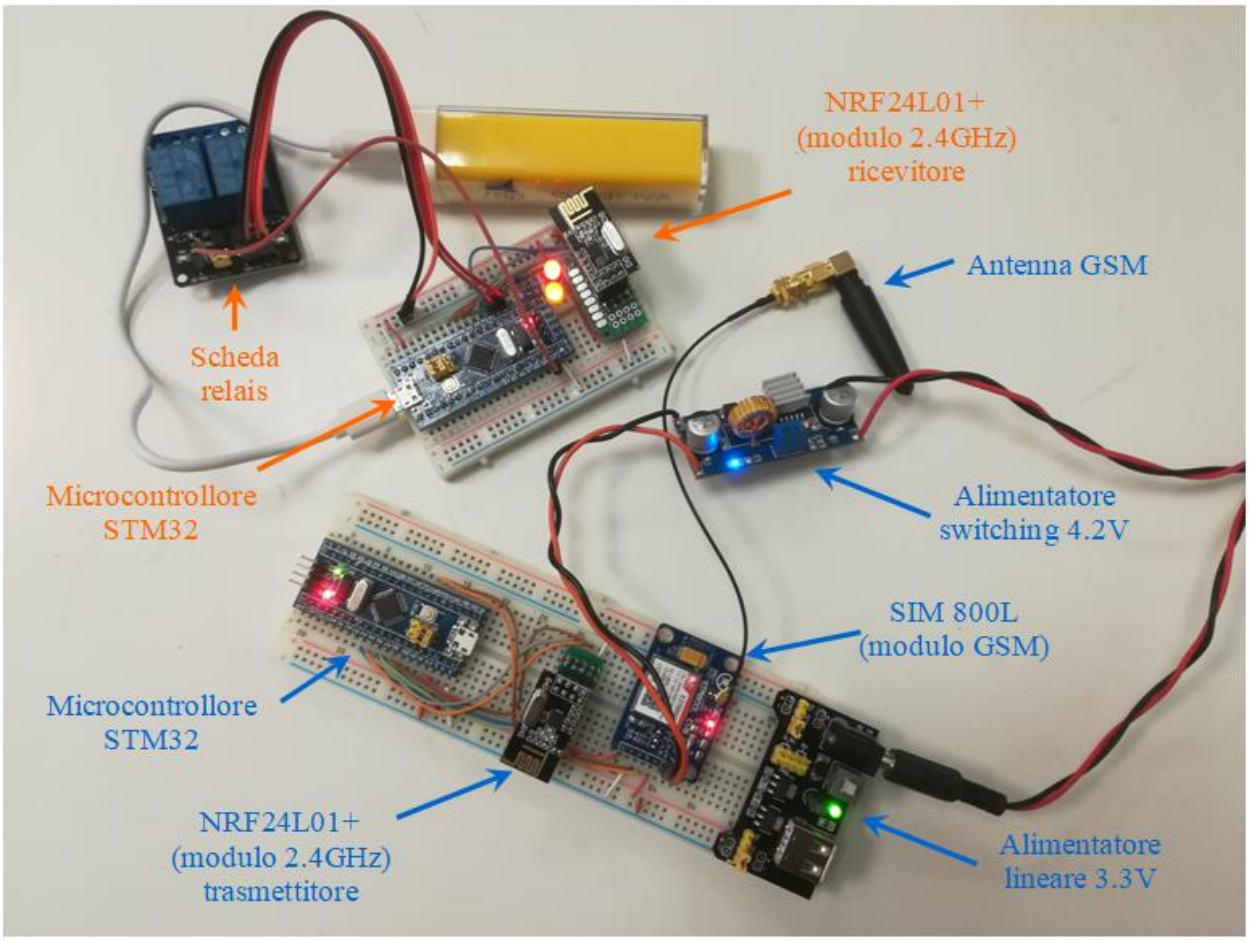


Figura 1 - Prototipo del sistema di comunicazione. In blu sono evidenziati i componenti della stazione base, mentre in arancione quelli dell'unità remota.

2.1 *Stazione base*

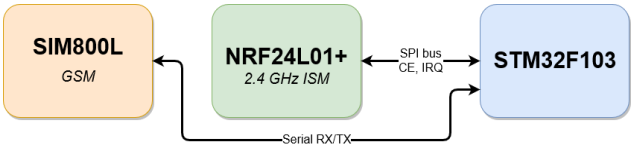


Figura 2 - Schema a blocchi della stazione base.

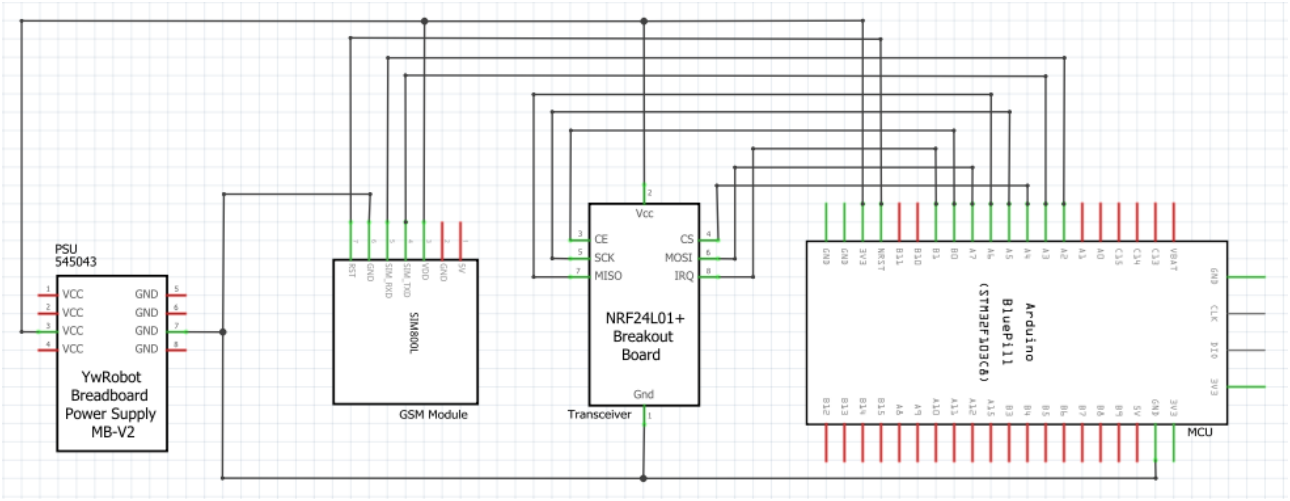


Figura 3 - Schema elettrico della stazione base.

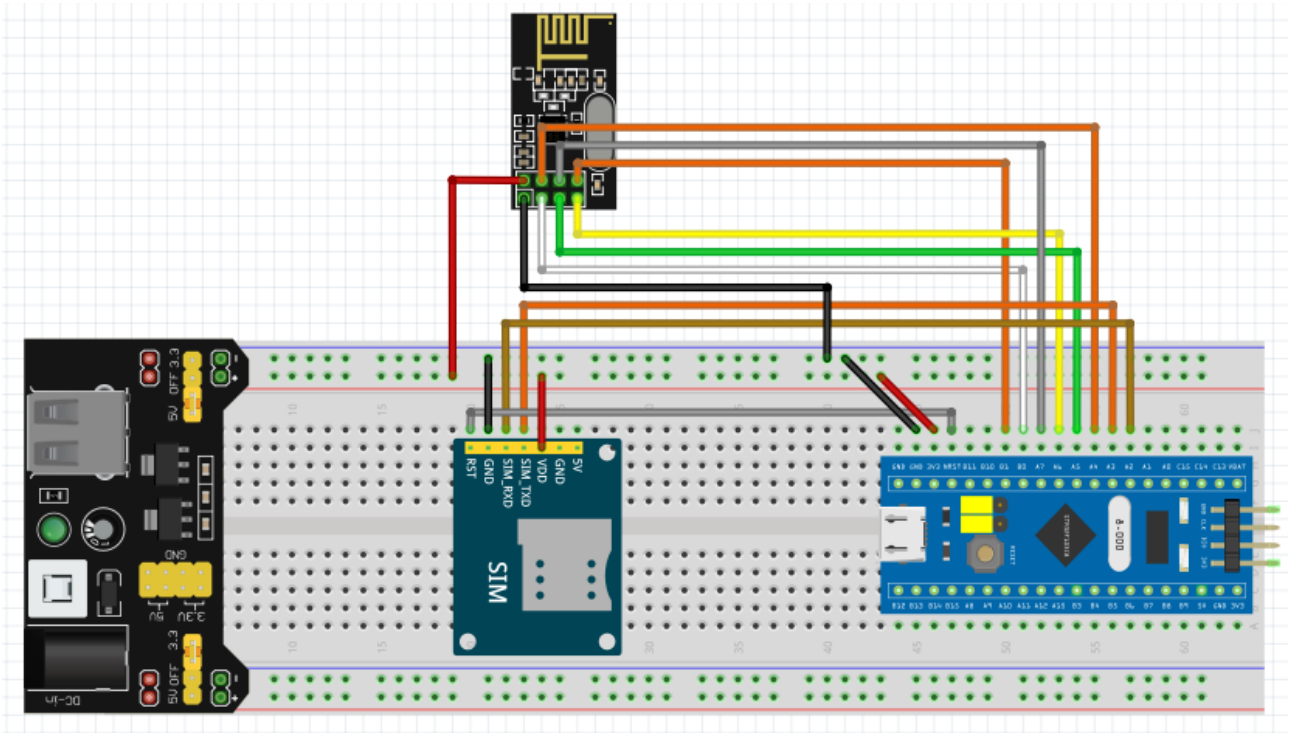


Figura 4 - Schema realizzativo del prototipo di stazione base. Nel nostro setup il modulo nRF24L01+ è montato direttamente sulla breadboard con un adattatore artigianale.

2.2 Unità remota

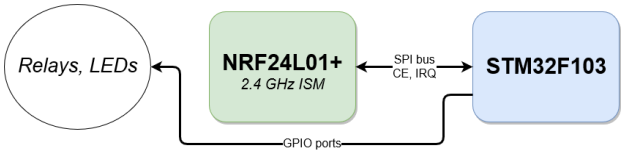


Figura 5 - Schema a blocchi dell'unità remota.

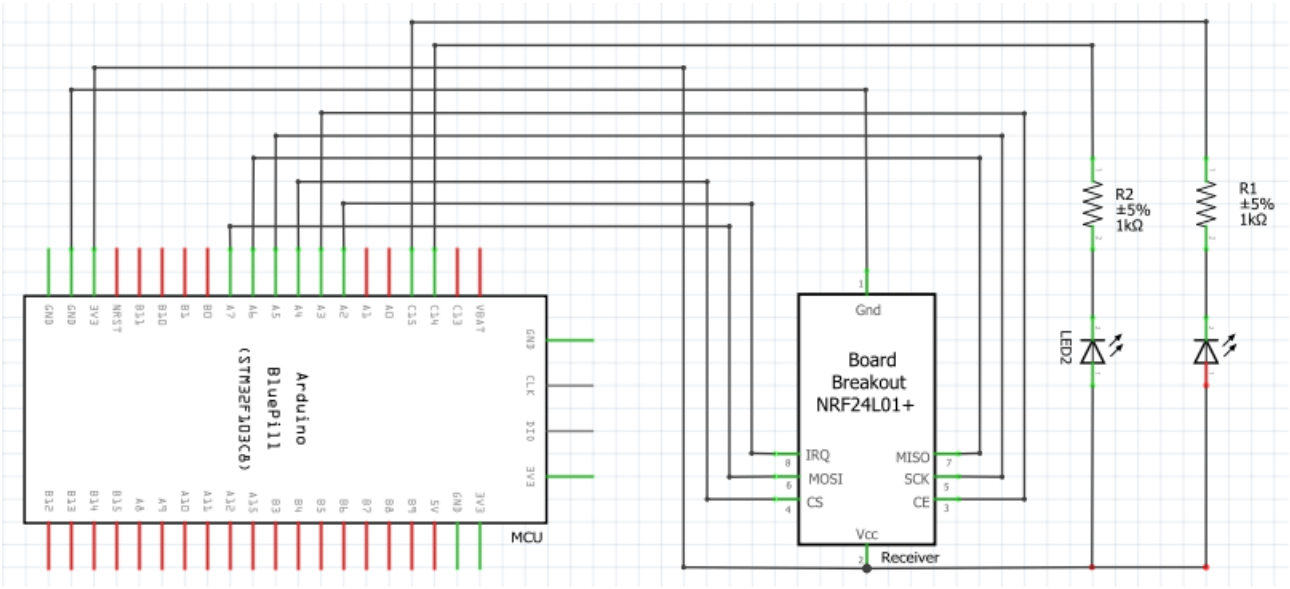


Figura 6 - Schema elettrico dell'unità remota.

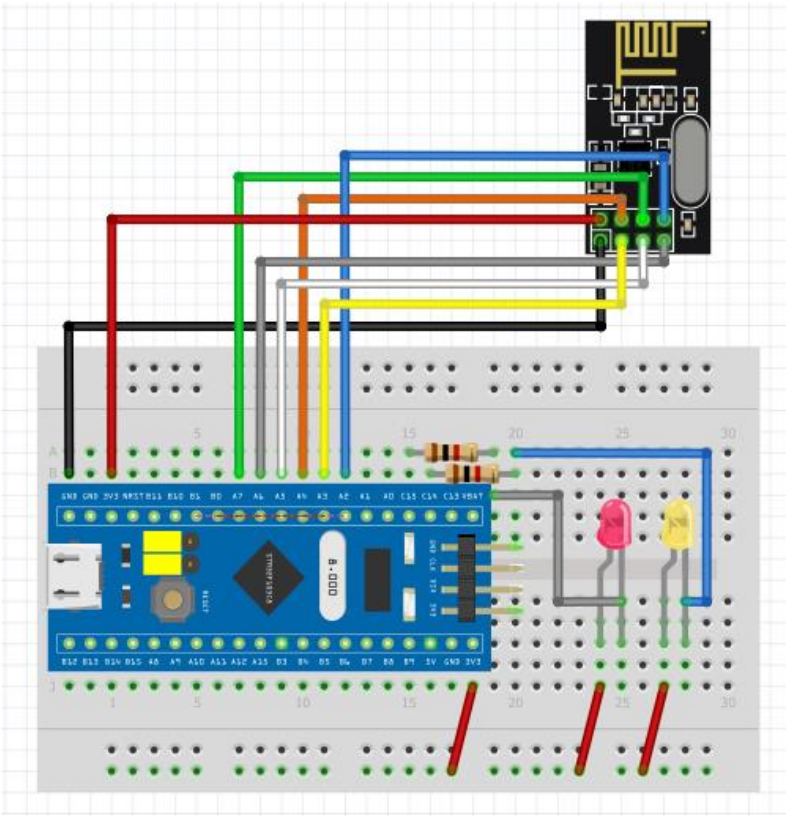


Figura 7 - Schema realizzativo del prototipo di unità remota. Nel nostro setup il modulo nRF24L01+ è montato direttamente sulla breadboard con un adattatore artigianale.

3. Modulo SIM800

Il chip SIM800, prodotto da *SIMCom*, è una soluzione completa GSM/GPRS; esso supporta lo standard *Quad-band 850/900/1800/1900MHz* e può scambiare voce, SMS e dati con un basso consumo di potenza. È molto piccolo (24*24*3mm) ed è agevole da utilizzare grazie al sistema *Embedded AT*, che ne consente la gestione via seriale asincrona scambiando comandi AT. Essi sono parte di un insieme di comandi originariamente sviluppati da Dennis Hayes per i modem fonici nel 1981 e attualmente diffusi per gestire qualsiasi tipo di modem.

Si osservi che il SIM800 non offre funzionalità 3G/HSUPA/HSDPA+/LTE né tecnologie superiori ma è molto efficace per inviare/ricevere SMS (che poi andranno interpretati dal microcontrollore) e aprire canali GPRS per inviare semplici pacchetti di dati (prevalentemente testo) con una portante massima di 85.6 kbps in download/upload.

Nel nostro sistema, esso è presente su una evaluation board di produzione cinese del costo di circa 10€ (SIM800L EVB V2.0), che viene venduta con una piccola antenna GSM esterna.

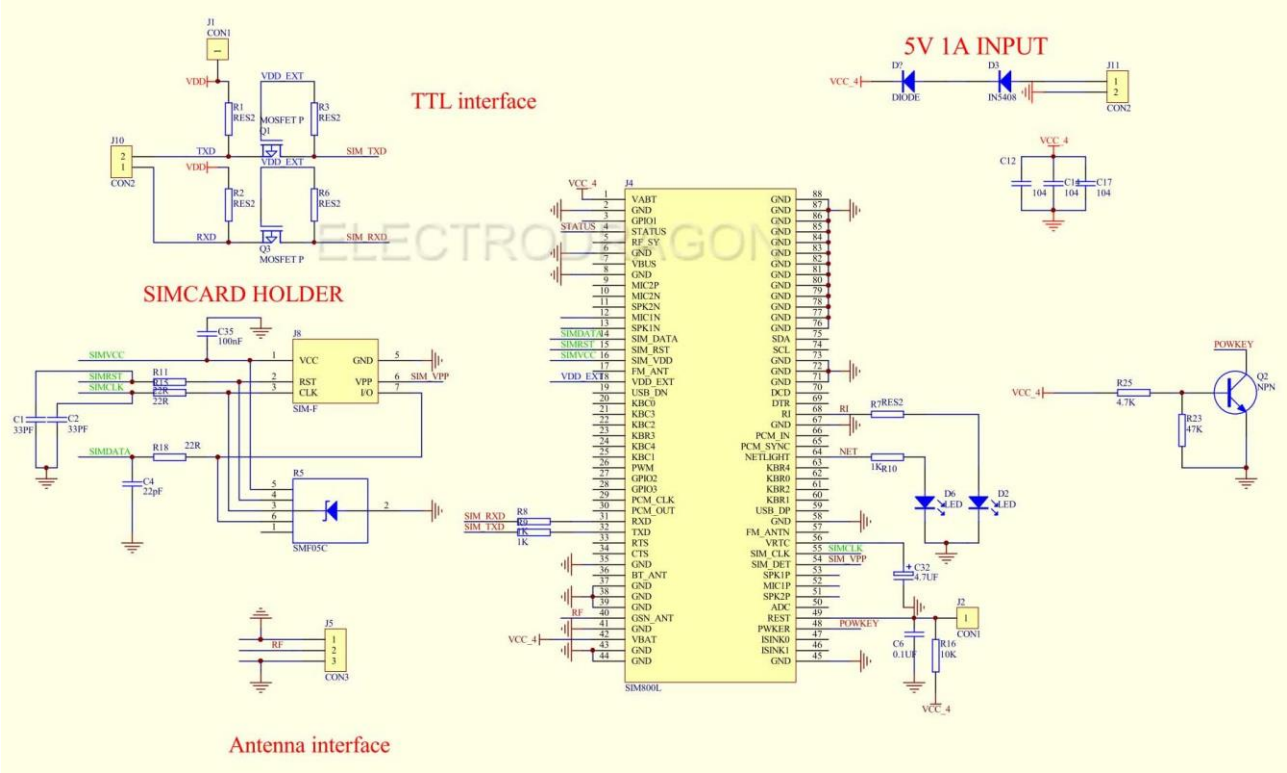


Figura 8 - Schema elettrico della scheda SIM800L EVB V2.0.

In tutta la documentazione relativa al SIM800 si utilizzano i seguenti acronimi:

- TA = Terminal Adapter: motori del GSM, ovvero il sistema SIM800L;
- TE = Terminal Equipment: chi controlla i motori GSM scambiando comandi AT, nel nostro progetto il microcontrollore STM32.

3.1 Alimentazione e livelli di tensione

Dallo schema si deduce che è presente un circuito per adattare i livelli di tensione della seriale (utilizzati sui pin RXD e TXD), traslandoli dal valore accettato dal SIM800 al potenziale applicato al pin VDD. Grazie a questo,

l'interfacciamento elettrico è immediato e rende il dispositivo utilizzabile con un computer, mediante un convertitore USB-TTL232, o direttamente con un microcontrollore.

Figura 9 - Scheda SIM800L EVB V2.0.

Il modulo assorbe al massimo 2A e dev'essere alimentato con una tensione tra 3.4V e 4.4 V, valori non molto comuni. Il classico 3.3V proveniente dai vari microcontrollori non è sufficiente per farlo operare correttamente; quando riesce ad accendersi, il modulo comunica via seriale il messaggio "UNDERVOLTAGE POWER OFF" e si spegne, mentre applicando 5V compare il messaggio "OVERVOLTAGE DANGER" e si rischia di danneggiarlo irreparabilmente. La scheda *SIM800L EVB V2.0* di produzione cinese presentava una coppia di diodi al silicio (D? e D3, 1N5408) connessi in serie alla VDD. Questi componenti sono stati aggiunti dal progettista orientale nella speranza di risolvere in modo semplice la necessità di una specifica tensione di alimentazione, sfruttando la caduta di dette giunzioni per poter alimentare il modulino direttamente a 5V.

Questa idea è tutt'altro che buona, infatti la tensione diretta del diodo 1N5408 impiegato non è costante al variare della corrente, come mostra il grafico tratto dal datasheet.

A seguito di alcuni test, il modulino funzionava in modo tutt'altro che affidabile, resettandosi imprevedibilmente; questo comportamento è stato imputato alla cattiva stabilità della tensione di alimentazione al variare delle condizioni operative.

Abbiamo quindi proceduto alla rimozione dei due diodi incriminati per poi cortocircuitare le relative piazzole, risolvendo definitivamente il problema con l'impiego di un alimentatore esterno.

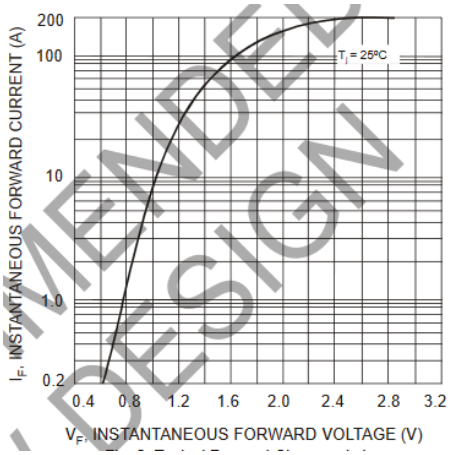


Figura 10 - Caratteristica del diodo 1N5408, non raccomandato per nuovi progetti

Nel nostro progetto è stato impiegato un convertitore DC-DC switching, del costo di circa 1€, basato sull'integrato XL4015 regolato a 4.2V, che accetta in ingresso tensioni fino a 36V.

Figura 11 - Dettaglio sulla modifica apportata alla scheda SIM800L EVB V2.0; rimossi i due diodi in serie all'alimentazione.

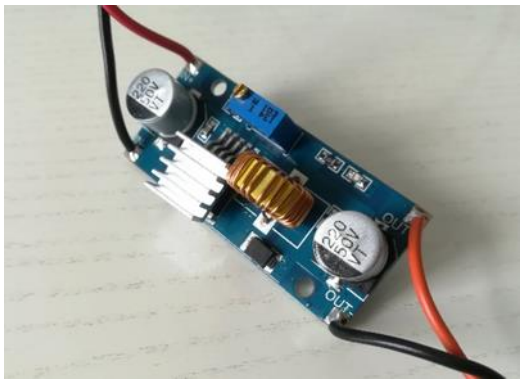


Figura 12 – Convertitore DC-DC switching con XL4015 per l'alimentazione del SIM800.

3.2 Comunicazione seriale e comandi AT

All'accensione del modulino occorre attendere qualche secondo e poi inviare il semplice comando AT; il baud rate della comunicazione seriale viene automaticamente dedotto dal SIM800, che si adegua di conseguenza rispondendo con OK.

Da questo momento si inizia ad interagire con il modulo inviando i comandi via seriale, ovvero caratteri ASCII seguiti da un *carriage return* ("\r\n"); ogni carattere ricevuto dal SIM800 viene rispedito indietro e, una volta elaborato il comando, viene aggiunta la risposta.

Di seguito si riportano alcuni comandi utilizzati durante gli esperimenti:

- AT per iniziare la comunicazione e innescare l'auto baud rate
- AT+CGMM nome del modulo
- AT+CGMR codice del modulo
- AT+CPIN? sapere se è richiesto il PIN
- AT+CNUM numero di telefono della SIM
- AT+CSQ *signal quality* (risposta sperimentale: 10 senza antenna, 26 con antenna)
- AT+CMGF=1 *text mode*, necessaria per leggere/inviare SMS
- AT+CMGS="+39....." INVIO <messaggio> CTRL-Z per inviare SMS
- AT+CMGL="ALL" legge tutti gli SMS

Tutti hanno dato esito positivo, utilizzando un convertitore USB-seriale e il software PUTTY a 115200 baud.

È anche possibile ottenere l'orario dalla rete GSM; in particolare i comandi coinvolti sono:

- AT+CLTS=1;&W abilita il servizio di orologio dalla rete e con l'opzione &W ne fa un settaggio permanente del modem (anche dopo reset HW resta)
- AT+CFUN=1,1 resetta il modulo via software
- AT+CLTS? controlla se il clock via rete è attivo
- AT+CCLK? restituisce ora corrente

I comandi effettivamente utilizzati nel nostro progetto sono invece:

- AT+CMGF=1 imposta la modalità testo per gli SMS (che altrimenti non sarebbero direttamente leggibili come stringhe)
- AT+CNMI=1,2,0,0,0 alla ricezione di un SMS il SIM800 mostrerà subito il numero di telefono e il testo del messaggio, senza salvarlo

A seguito di questa configurazione, quando un SMS viene ricevuto, il SIM800 invia sulla seriale il numero di telefono del mittente, la data e l'ora:

+CMT: "+39xxxxxxxxxx","","18/11/18,17:29:54+04"
Testo del messaggio

Il *parser* implementato su STM32 si basa proprio su questa struttura del messaggio e ne estrae le informazioni.

4. Modulo nRF24L01+

L'nRF24L01+, prodotto da *Nordic Semiconductor*, è un ricetrasmettitore integrato per la banda 2.4 GHz ISM (*industrial, scientific and medical)*. Caratterizzato da un bassissimo consumo di potenza e dotato dell'ESB (*Enhanced ShockBurst™*), periferica hardware che si occupa della gestione del protocollo di comunicazione, rappresenta la soluzione ideale quando si desidera realizzare un sistema di connettività wireless con MCUs (*microcontroller units*) di basso costo.

Per questo progetto è stata utilizzata una scheda di produzione cinese contenente tutti i componenti necessari al funzionamento del chip, incluso un quarzo per il sintetizzatore di frequenza e un'antenna direttamente disegnata sul circuito stampato.

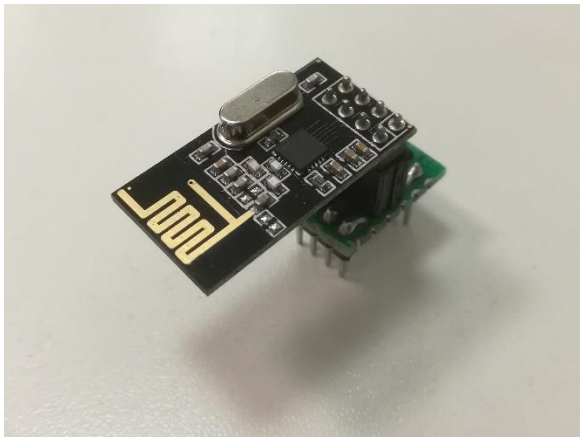


Figura 13 - Modulo nRF24L01+ con adattatore artigianale per il montaggio su breadboard.

Nell'elenco che segue sono elencate alcune tra le caratteristiche principali del chip.

- *Enhanced ShockBurst™* - motore di protocollo in banda base integrato nel chip che fornisce le seguenti funzionalità:
    - comunicazione basata sulla trasmissione di pacchetti tramite operazioni manuali o avanzate autonome;
    - gestione automatica del *data link layer*;
    - possibilità di avere la lunghezza payload dinamica (da 1 a 32 byte);
    - presenza di sei data pipe parallele con indirizzi univoci (tecnologia *MultiCeiver™*) per connessioni a stella 1:6.
- Trasmissione e ricezione nella banda ISM da 2.4GHz a 2,525 GHZ (sono disponibili 126 canali fisici).
- Comunicazione con la MCU tramite protocollo SPI, che permette, tra le altre cose, l'accesso ai registri in tutte le modalità di funzionamento del chip. *Nota:* il modulo non ha memoria non volatile e se viene rimossa l'alimentazione il contenuto dei registri va perduto.
- Presenza di FIFO interne che assicurano buon flusso dati tra radio e MCU.
- Front-end radio caratterizzato da:
    - Modulazione GFSK (*Gaussian Frequency Shift Keying*) - è una modulazione FSK nella quale gli impulsi corrispondenti al dato digitale vengono "smussati" con un filtro gaussiano prima di essere inviati al modulatore vero e proprio; in questo modo l'occupazione di banda viene ridotta (minori bande laterali attorno alla portante) ma aumenta l'interferenza inter-simbolica.
    - Parametri configurabili: frequenza del canale radio (126 canali), potenza in uscita (0 -6 -12 -18 dBm) e data-rate in aria (250kBit/s, 1MBit/s, 2MBit/s).
- Presenza di due modalità di risparmio di potenza (predisposizione ad applicazioni ultra-low power).

- Gestione ottimale dell'alimentazione:
    - varie modalità: power-down, standby-I, standby-II;
    - 1.5ms tempo tra il power-down e il power up.
- Interfaccia HOST per la comunicazione verso la MCU:
    - SPI 4 pin, max 10Mbit/s, 3FIFO TX/RX da 32byte (tot 6 FIFO);
    - Pin 5V tolerant (ma alimentazione da 1.9V a 3.6V).
- PIN:
    - SPI usa i pin MISO, MOSI, SCK e CSN;
    - il pin CE attiva la modalità RX o TX;
    - il pin IRQ rappresenta un'uscita interrupt attivo basso.

Per l'utilizzo del chip qui descritto è importante conoscere le modalità di controllo previste. Alcuni dei punti salienti tratti dal suo datasheet sono di seguito elencati.

- È presente una macchina a stati interna che prende in ingresso i valori di registro definiti dall'utente e i segnali interni al modulo.
- Il funzionamento si basa sul passaggio tra diversi modi operativi:
    - **power down**: il modulino è disabilitato; i valori dei registri sono mantenuti; la comunicazione SPI è tenuta attiva e i registri sono accessibili; in questo modo si entra con PWR_UP = 0 nel registro CONFIG;
    - standby:
        - **standby-I**: ci si entra con PWR_UP = 1 (e mantenendo il pin CE = 0); in questo stato si hanno tempi di startup brevi mantenendo un consumo non elevato; solo una parte dell'oscillatore a cristallo è attivo;
        - **standby-II**: il consumo è leggermente più elevato; ci si entra quando un PTX (*Primary TX*) ha buffer di trasmissione vuoto e CE = 1; se la TX FIFO viene riempita, si torna in TX (130us per far ripartire il PLL);
    - **RX**: ci si entra da standby-I mettendo CE = 1 e PRIM_RX = 1 (ovviamente PWR_UP = 1); il modulino funziona come ricevitore: demodula i segnali dal canale RF presentando costantemente i dati in arrivo al motore di protocollo interno, il quale è in continua attesa di un pacchetto valido, ovvero un pacchetto con:
        - CRC valido;
        - indirizzo corretto;
        - una volta ricevuto un pacchetto valido il suo payload è messo in una RX FIFO (se vi è spazio disponibile, altrimenti è perduto). A questo punto l'ESB può commutare lo stato su modalità TX per trasmettere il pacchetto di ACK, se richiesto. In RX si ha segnale RPD (*Received Power Detector*), che se alto, indica che c'è del segnale RF nel canale selezionato.
    - **TX**: ci si entra da standby-I mettendo PRIM_RX = 0 (ovviamente PWR_UP = 1), un payload nella TX FIFO e un impulso su CE di almeno 10us; si usa per trasmettere e vi resta finché il pacchetto corrente non è stato trasmesso; dopodiché se CE = 0, passa in standby-I. *Nota:* il modulo non deve restare in TX per più di 4ms (l'ESB rispetta questo vincolo).
    - In definitiva, una possibile linea guida per l'utilizzo del ricetrasmettitore è esposta di seguito. *All'accensione, quando la VDD supera 1.9V, il modulino si mette in stato "**power on reset**" che dura al massimo 100ms, per poi andare automaticamente in "power down". Per andare in standby occorre porre PWR_UP = 1 nel registro di configurazione, dopodiché attendere 1.5ms. A questo punto, per fare ricezione o trasmissione serve mettere CE = 1 e scegliere TX/RX con PRIM_RX (se =0, TX, altrimenti RX).*

- L'*air data rate* (ADR) si imposta con RF_DR nel registro RF_SETUP; il ricevitore e il trasmettitore devono avere la stessa ADR per poter comunicare insieme.
- Per quanto riguarda il canale in frequenza:
    - ogni canale occupa meno di 1MHz (per ADR 250kbps e 1Mbps) o meno di 2MHz (per ADR 2Mbps);
    - la risoluzione di scelta del canale è pari a 1MHz;
    - F0 = 2400 + RF_CH [MHz]
- È disponibile un *power amplifier control* usato per impostare potenza in TX (vi sono quattro scelte possibili).

Si riportano inoltre le funzionalità peculiari dell'ESB.

- È un gestore del *data link layer* basato su pacchetti; assembla e temporizza i questi ultimi e gestisce ACK e ritrasmissione.
- Fornisce comunicazione bidirezionale affidabile in modo semplice.
- Una transazione di pacchetto con ESB prevede la presenza di un PRX e di un PTX; inizia con trasmissione da PTX e finisce quando il PTX riceve un ACK dal PRX (il quale può restituire assieme all'ACK anche alcuni dati realizzando una comunicazione bidirezionale).
- Gestisce PTX e PRX facendoli automaticamente commutare dal modo TX al modo RX e viceversa. Il punto di partenza è la trasmissione iniziata dal PTX.
- Si possono configurare max numero ritrasmissioni e ritardo tra una e l'altra.
- Ogni pacchetto ha la seguente struttura:
    - *preambolo* 01010101 = 85d (se primo bit indirizzo è 0) oppure 10101010 = 170d (se primo bit indirizzo è 1); serve per stabilizzare il ricevitore;
    - *indirizzo* per il ricevitore; si può configurare la sua lunghezza tra 3, 4, 5 byte con il registro AW; *nota:* non utilizzare sequenze con una sola transizione tipo 000FFFFFh oppure con tutte transizioni (tipo preambolo)
    - campo *packet control* (9 bit); contiene *lunghezza del payload* (usato solo se è abilitata la *dynamic payload length* - se è 0 vuol dire vuoto e si usa per mandare solo ACK), identificativo pacchetto (*PID* - utile per far capire al PRX se il pacchetto è una ritrasmissione (stesso PID) oppure un pacchetto nuovo) e un bit di *flag* NO_ACK (usato solo se *autoACK* è attivo).
        - *Nota:* se NO_ACK = 1, vuol dire che questo pacchetto non deve essere *acknowledged* (il ricevitore non deve dire "io l'ho ricevuto"); il PTX lo può settare con il comando W_TX_PAYLOAD_NOACK (e prima si deve abilitare EN_DYN_ACK nel registro FEATURE); se NO_ACK è attivo, una volta finita la trasmissione del pacchetto corrente il PTX torna in standby-I (perché non deve attendere la risposta del PRX).
    - *CRC* da 1 o 2 byte.
- <u>Gestione automatica del pacchetto</u>:
    - lunghezza del payload statica (di default) o dinamica:
        - statica: il TX riempie adeguatamente la FIFO, mentre l'RX ha un registro (RX_PW_Px) per indicare la lunghezza; le due devono dimensioni essere coerenti;
        - dinamica (DPL - *Dynamic Payload Length*): l'RX può decodificare automaticamente perché l'informazione sulla lunghezza è contenuta nel pacchetto; la MCU può chiedere al modulino RX la lunghezza del pacchetto con il comando R_RX_PL_WID; per abilitarla EN_DPL = 1 nel registro FEATURE sia del TX che del RX; inoltre è necessaria l'impostazione del registro DYNPD (vedere pag. 29 del datasheet).
- <u>Assemblamento automatico del pacchetto</u> (preambolo, indirizzo, campo di controllo pacchetto, payload, CRC):
    - preambolo (dipende dal primo byte dell'address)

- indirizzo fetchato dal registro TX_ADDR; può avere lunghezza 3, 4, 5 byte (sulla base del registro AW);
- campo di controllo pacchetto: la lunghezza è inserita se DPL attiva sulla base dei byte nella FIFO TX; il TX aumenta il PID ogni volta che genera un nuovo pacchetto (e lo riusa in caso di ritrasmissione);
- payload è fetchato dalla FIFO TX;
- CRC calcolato automaticamente (a pag. 28 del datasheet sono indicati i polinomi generatori); n° di byte settato da CRCO nel registro CONFIG;
- validazione automatica del pacchetto: nella modalità di ricezione il chip cerca nei pacchetti in arrivo l'indirizzo valido, corrispondente al proprio (che si trova nel registro RX_ADDR); se OK, l'ESB inizia a catturare il pacchetto sulla base della lunghezza (statica o dinamica) dopodiché calcola il CRC; se il CRC è corretto, l'ESB controlla il PID, comparandolo con quello ricevuto precedentemente; se PID_nuovo == PID_vecchio allora controlla i due CRC che, se uguali, implicano stesso pacchetto ricevuto due volte (l'ultimo viene scartato); se il pacchetto invece è valutato come valido (vedi fig.6 pag.30 per flowchart);
- disassemblamento automatico del pacchetto: una volta validato, il pacchetto è caricato nella RX FIFO e si asserisce la richiesta di interrupt (IRQ) RX_DR; *nota:* l'asserzione di questa richiesta di interrupt non comporta per forza il pilotaggio del segnale IRQ; infatti, ciò dipende dal settaggio del bit MASK_RX_DR nel registro CONFIG (se 1 l'interrupt è mascherato e quindi non ha un effetto elettrico; viceversa se 0 si riflette come un interrupt attivo basso sul pin IRQ).

- Gestione automatica della transazione del pacchetto. Due funzioni in particolare:
  - *Auto Acknowledgement*: il PRX invia un pacchetto di ACK al PTX dopo aver validato un pacchetto; è abilitato settando il registro EN_AA; se il pacchetto ricevuto ha NO_ACK = 1 allora l'AA non è eseguito;
    - Auto Acknowledgement con payload: è disponibile abilitando il bit EN_ACK_PAY nel registro FEATURE del ricevitore; un pacchetto di ACK può contenere un payload opzionale (funzione disponibile solo se DPL abilitata) aggiungibile mettendo i dati nella TX FIFO del PRX e poi inviandogli il comando W_ACK_PAYLOAD; in questo modo non appena sarà ricevuto un pacchetto dal PTX, questo payload sarà inviato insieme all'ACK (se l'indirizzo del PTX è concorde con quello desiderato come destinazione per il payload); max 3 payload "pendenti";
  - *auto ritrasmissione*: ritrasmette un pacchetto se ACK non ricevuto; si usa in un sistema con Auto Acknowledgement nel PTX; il numero di ritrasmissioni massime si imposta nei bit ARC del registro SETUP_RETR; il PTX entra in modalità RX in attesa di ricevere ACK per un tempo legato ai bit ARD del registro SETUP_RETR (che impostano l'Auto Retransmission Delay); poi torna in modalità TX per ritrasmettere. Il tempo di ARD non può essere più breve della somma del tempo di startup e del tempo on-air (vedi pag. 32).
    - Quando l'ACK viene correttamente ricevuto, il PTX asserisce la richiesta di interrupt TX_DS (TX Data Sent "ho spedito e l'altro ha ricevuto correttamente"); tale IRQ viene asserita anche quando l'Auto-Retransmission non è attivo e viene trasmesso un pacchetto.
    - È possibile usare l'informazione contenuta nel registro OBSERVE_TX per avere informazioni sulla qualità del canale; esso contiene infatti il numero di ritrasmissioni per l'ultima transazione o per l'intera durata del canale corrente (vedi pag. 32)
    - In alternativa all'auto-retransmit si può impostare manualmente il modulino per ritrasmettere un certo numero di volte un pacchetto con il comando REUSE_TX_PL e con vari impulsi sul CE.
    - *nota* su TX_DS e RX_DR IRQ: sono entrambi asseriti se viene ricevuto un payload con l'ACK.

- *Nota:* se raggiunto massimo numero di ritrasmissioni, viene asserito MAX_RT IRQ

- MultiCeiver™:
  - usato nella modalità RX; contiene 6 data pipes (0 - 5) parallele con indirizzi unici; una data pipe è un canale logico nel canale fisico RF.
  - un ricevitore può ricevere da 6 diversi PTX sullo stesso canale RF, ciascuno assegnato ad una data pipe diversa (e quindi ad un indirizzo diverso). Ogni volta che un pacchetto viene ricevuto il suo indirizzo è confrontato con quello delle 6 data pipe e inoltrato a quella giusta. Attenzione: solo un pacchetto alla volta può essere ricevuto (perché fisicamente il canale radio è uno solo!).
  - Caratteristiche comuni a tutte le data pipe: CRC, larghezza indirizzo di ricezione, canale RF, air data rate, guadagno del LNA.
  - Le data pipe sono abilitate con i bit del registro EN_RXADDR; di default lo sono solo la 0 e la 1. Ciascun indirizzo è configurato nei registri RX_ADDR_Px (x = 0 ... 5); *nota:* assicurarsi che nessuna data pipe abbia lo stesso indirizzo di un'altra.
  - *nota* su indirizzi pipe: la pipe 0 ha un indirizzo totalmente arbitrario; le pipe da 1 a 5 condividono i primi 4 byte (MSB) e differiscono tra loro solo per l'LSB. Questo significa che un solo PRX fisico riceve su 6 indirizzi differenti, ovvero ha 6 propri indirizzi.

ESEMPIO – Configurazione a stella:

Consideriamo una configurazione a stella dove al centro c'è un PRX e alle punte ci sono 6 PTX (PTX0 ... PTX5). Per avere una comunicazione funzionante, ogni PTX deve trasmettere sull'indirizzo di una pipe (es. n-esima) del PRX (ovvero il suo TX_ADDR = RX_ADDR_Pn) ma lo stesso PTX deve poter ricevere gli ACK a lui destinati e quindi deve avere il proprio RX_ADDR_P0 = TX_ADDR. Si osservi che quando un TX invia un pacchetto si aspetta l'ACK di quel pacchetto sulla pipe 0. In questa configurazione, quindi, ogni PTX ha RX_ADDR_P0 = TX_ADDR e TX_ADDR uguale all'indirizzo della pipe del PRX sulla quale vuole scrivere. Se si volesse realizzare una comunicazione bidirezionale (non basata su payload attaccato all'ACK) con una delle "punte" della stella, sarebbe conveniente farlo con la pipe 0 del PRX che, quando diventa PTX, contiene già nel RX_ADDR_P0 l'indirizzo corretto del proprio ascoltatore (il nuovo PRX, ex PTX0). Notare che mentre il nodo centrale diventa un trasmettitore, esso non riceve più i dati dei PTX "punte" della stella.

*Nota: p*er evitare sovrapposizioni continue tra le trasmissioni dei 6 PTX è saggio adottare dei ritardi di ritrasmissione diversi per ogni PTX. In questo modo se per esempio PTX0 e PTX1 trasmettessero un pacchetto nello stesso istante, la successiva ritrasmissione non si sovrapporrebbe (per es. PTX0 ritrasmetterebbe prima di PTX1).

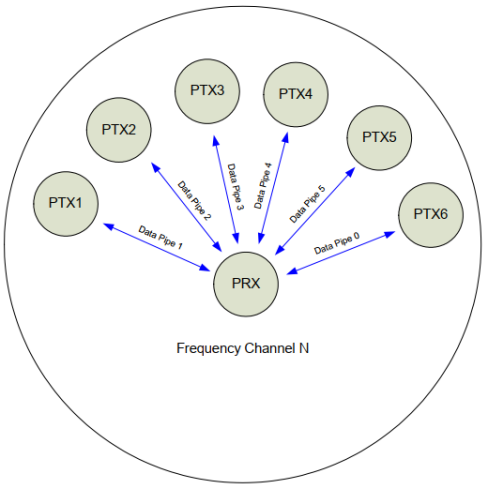


Figura 14 - Esempio di applicazione dell'nRF24L01+ in una configurazione a stella.

Si elencano ora alcune considerazioni ritenute utili per l'utilizzo del modulo descritto.

- Quando si ha trasmissione con NO_ACK = 0, l'ESB fa commutare automaticamente il PTX in RX per attendere ACK e cancella il pacchetto dalla TX FIFO solo una volta che ha ricevuto l'ACK.
- TX/RX con ACK e pacchetto perso: il PTX dopo aver trasmesso attende l'ACK, passando in modalità RX; durante l'attesa del ritardo di ritrasmissione ARD il PTX non resta sempre in ascolto (RX viene spento per risparmiare energia se non trova indirizzo valido entro 250us).
- TX/RX con ACK+payload: il PTX trasmette; il PRX riceve, asserisce RX_DR IRQ e trasmette ACK con payload; il PTX riceve ACK+payload e asserisce TX_DS (trasmissione riuscita) e RX_DR (ricevuto payload) IRQ. Il PRX asserirà TX_DS IRQ solo quando riceverà un nuovo pacchetto dal PTX.
- Se la trasmissione non va a buon fine (max ritrasmissioni finite) allora il pacchetto resta nella TX FIFO. Si può ripartire con la ritrasmissione dello stesso con un colpetto di CE; altrimenti si può svuotare FIFO con il comando FLUSH_TX.

Per quanto riguarda l'interfaccia per lo scambio di dati e controlli tra il modulo e la MCU essa è, come già detto, basata sul protocollo SPI. Alcuni aspetti salienti a tal proposito sono riportati nel seguente elenco.

- Sei pin tolleranti ai 5V:
  - IRQ (attivo basso)
  - CE (attivo alto)
  - Segnali SPI: CSN (chip select negato), SCK, MOSI, MISO (SPI da 0 a 10 Mbps)
- I comandi SPI sono grandi 1byte e devono essere sempre preceduti da una transizione del CSN da alto a basso.
- Il contenuto dello status register è sempre letto e messo sulla MISO dopo una transizione alto->basso del CSN. Il suo contenuto è inaffidabile durante la transizione alto->basso del pin IRQ, perché esso viene aggiornato in quel momento.
- I comandi sono scritti con la seguente struttura:
  - command word (1 byte) MSBit first
  - data words (x byte) LSByte to MSByte, MSBit first

Figura 15 - Analizzatore logico utilizzato durante gli esperimenti per verificare la comunicazione SPI.

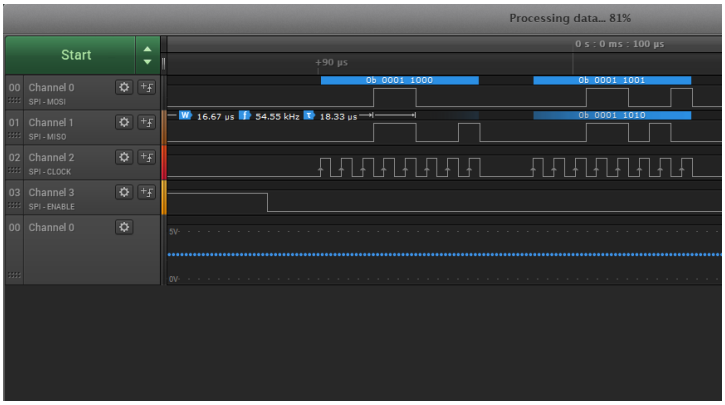


Figura 16 - Analisi della comunicazione SPI.

5. STM32

I microcontrollori a 32 bit che gestiscono la *stazione base* e le *unità remote* sono degli STM32F103C8T6, montati su delle breakout board note con il nome di *Blue Pill*; di seguito alcune caratteristiche:

- core ARM Cortex – M3
- tensione di alimentazione: da 2.0V a 3.6V
- frequenza massima operativa: 72MHz
- memoria programma: flash da 64 Kbyte
- memoria RAM: 20 Kbyte SRAM
- oscillatore interno a 8 MHz
- 2 interfacce SPI
- 3 interfacce seriali USART

Figura 17 - Scheda Blue Pill con STM32F103C8T6

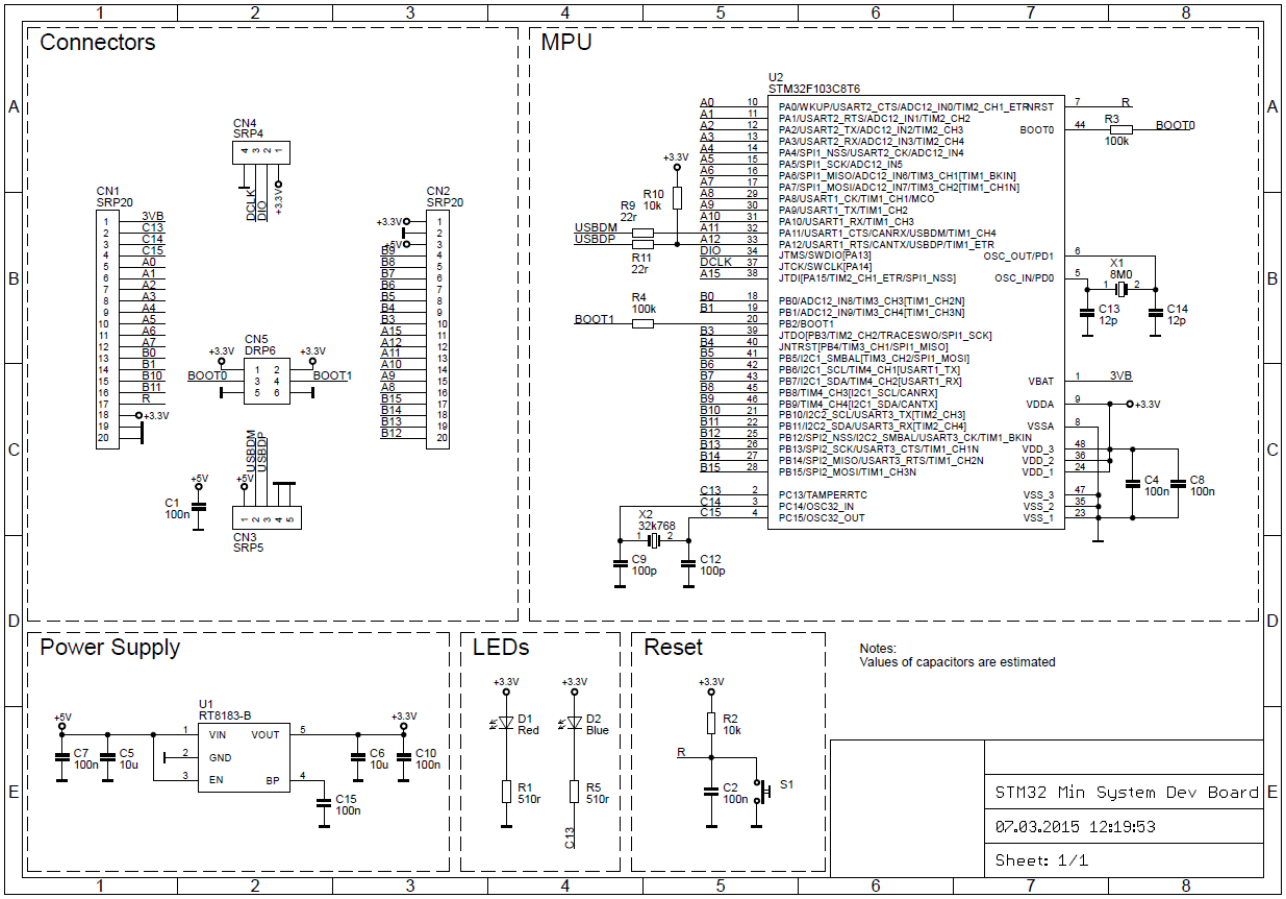


Figura 18 - Schema della scheda Blue Pill con microcontrollore STM32F103C8T6

Ciascuna di queste schedine, dal costo inferiore ai 2€, è inoltre provvista di:

- 1 LED rosso per indicare la presenza della tensione di alimentazione
- 1 LED verde collegato al pin PC13 del microcontrollore
- 1 quarzo da 8MHz per la generazione di un clock esterno
- 1 regolatore di tensione lineare da 3.3V
- 1 pulsante di reset
- 2 file di connettori passo 2.54mm da 20 pin ciascuno (pin del microcontrollore e alle alimentazioni)
- 1 connettore da 4 poli per la programmazione
- 1 connettore USB

Figura 19 - Programmatori ST-Link V2. A sinistra l'originale, a destra un clone cinese.

La schedina può essere alimentata sia fornendo direttamente una tensione appropriata, sia attraverso il connettore USB, da cui provengono 5V (ridotti a 3.3V dal regolatore).

La programmazione avviene con il programmatore *ST-Link*, connesso al computer via USB e al microcontrollore attraverso un protocollo proprietario basato su una comunicazione seriale sincrona (SWCLK, SWDIO).

Lo sviluppo del codice C è stato eseguito nell'ambiente di lavoro *Eclipse*, con il compilatore *ARM GCC*, appoggiandosi alle librerie *HAL* (*Harware Abstraction Libraries*) di ST Microelectronics per STM32.

Il codice di configurazione per il microcontrollore è stato generato automaticamente attraverso il tool *STM32 CubeMX* di ST Microelectronics.

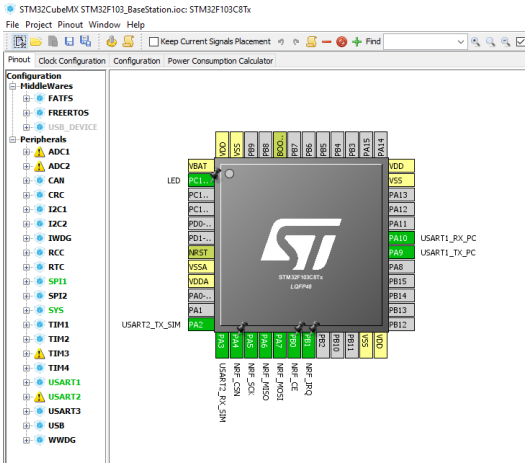


Figura 20 - Schermata del tool STM32 CubeMX durante la configurazione della stazione base.

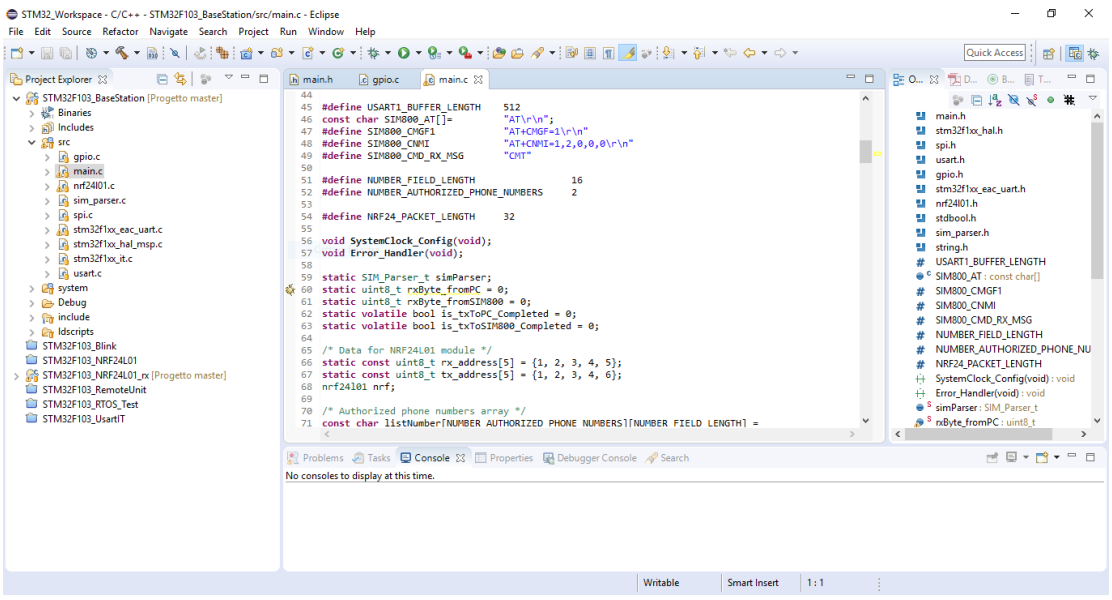


Figura 21 - L'ambiente di sviluppo Eclipse durante la scrittura del firmware per la stazione base.

Nella *stazione base*, il microcontrollore STM32 dialoga con il SIM800L attraverso la UART1 e con il modulino nRF24L01+ con la SPI1, mentre nell'*unità remota* oltre alla SPI sono usati dei pin GPIO. I programmi *main* eseguono i diagrammi di flusso seguenti:

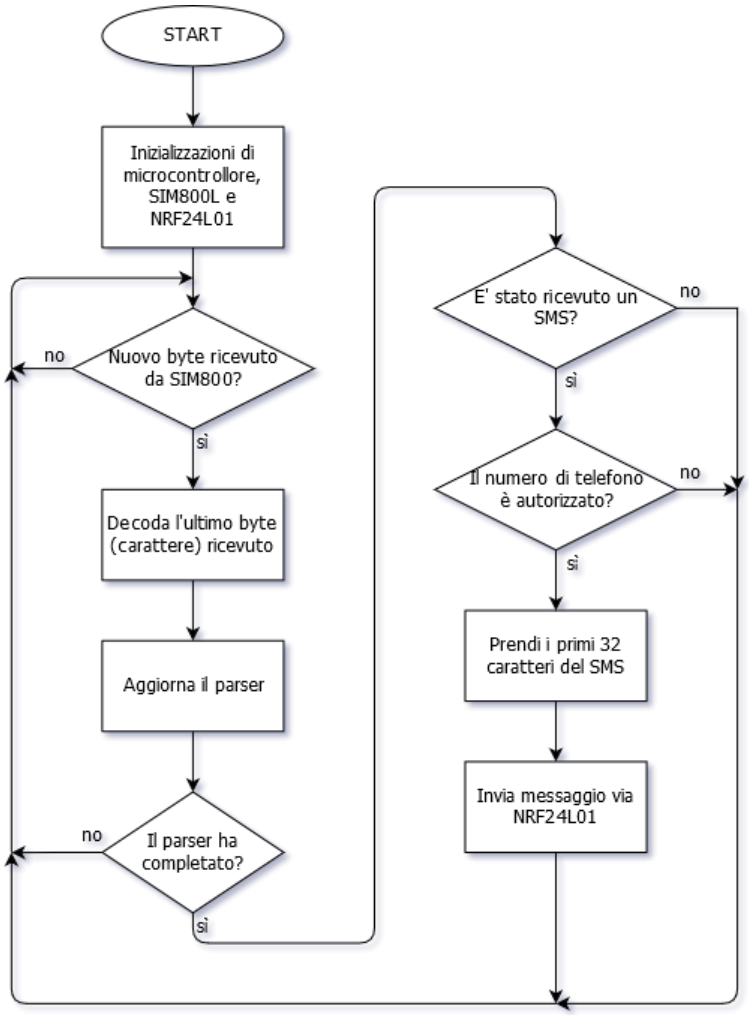


Figura 22 - Diagramma di flusso del programma della stazione base.

Figura 23 - Diagramma di flusso del programma dell'unità remota.

La gestione della seriale per il SIM800L ha richiesto di modificare le librerie HAL di ST Microelectronics, per aggiungere la possibilità di ricevere un numero di byte non noto a priori. A tal proposito, il nuovo codice, che si integra perfettamente nel restante ambiente HAL, costruisce un buffer circolare che viene scritto nella routine di interrupt della periferica seriale del microcontrollore e viene letto dal programma main.

6. Modulo relais

A titolo esemplificativo nel prototipo della *unità remota* è stato collegato un modulo contenente due relais elettricamente disaccoppiati dal resto del circuito.

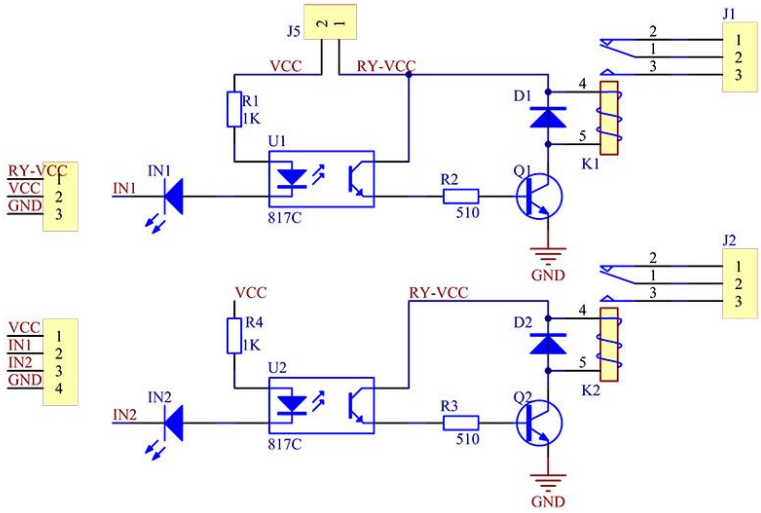


Figura 24 - Schema del modulo relais.

Il microcontrollore si collega ai morsetti IN1 e IN2 che sono attivi bassi; infatti, quando portati a massa, scorre corrente attraverso la resistenza R1 (R4) nel diodo LED del fotoaccoppiatore U1 (U2) e in quello esterno IN1 (IN2); in questa situazione il transistor Q1 (Q2) va in conduzione ed eccita la bobina del relais K1 (K2). Si noti la presenza del diodo D1 (D2) per il ricircolo della corrente in fase di smagnetizzazione dell'avvolgimento che evita il crearsi di una pericolosa sovratensione. Si osservi inoltre che la tensione di pilotaggio delle bobine è separata da quella dei fotoaccoppiatori permettendo un ulteriore disaccoppiamento che è stato sfruttato nel prototipo per separare alimentazione a 5V (bobine) e 3.3V (fotoaccoppiatori).

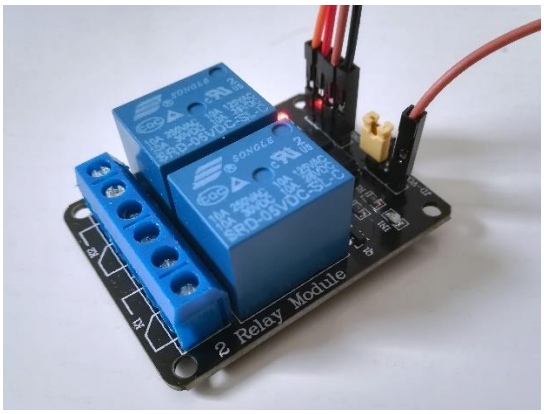


Figura 25 - Modulo relais utilizzato nel prototipo.

Sui contatti dei relais possono essere collegati carichi a 230V fino a 10A.

7. Conclusioni e sviluppi futuri

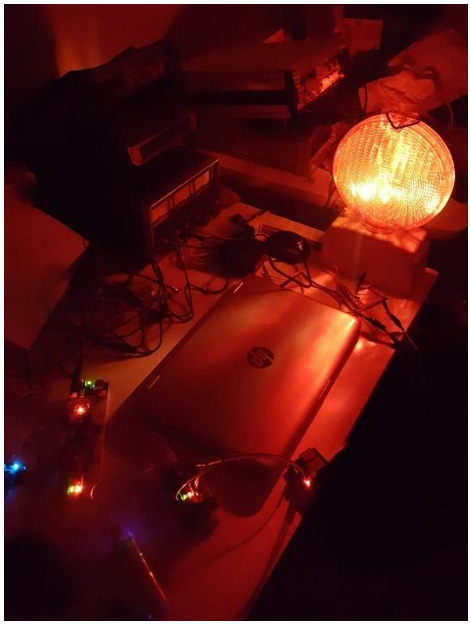


Figura 26 - Esperimenti con l'unità remota che accende una lampada da cantiere.

Questo progetto ha un elevato contenuto didattico, riguardante sia l'utilizzo di dispositivi a radiofrequenza che il relativo interfacciamento con microcontrollori. In particolare, lo studio del modulo a 2.4GHz ha consentito di approfondire le caratteristiche di una comunicazione radio (canale RF, modulazione adottata, air data rate, potenza in trasmissione) ed evidenziare le problematiche dello scambio di dati, con la necessità di protocolli per l'acknowledgment e la ritrasmissione dei pacchetti.

Il sistema ha un'applicazione pratica di notevole interesse nella vita quotidiana. Il prototipo qui presentato sarà sviluppato in futuro per essere realmente utilizzabile in ambito domestico e a tal proposito si dovrà dotarlo di tutte le parti qui non discusse ma necessarie, ovvero un contenitore ermetico e un proprio alimentatore.

Inoltre, si desidera aggiungere le seguenti funzionalità:

- comunicazione bidirezionale tra le unità remote, sparse per la casa, e l'unità base, allo scopo di inviare degli SMS di feedback ai numeri in lista; per esempio, la temperatura di una stanza o dell'acqua di uno scaldabagno o l'effettiva accensione dell'elettrodomestico controllato mediante rilevazione della corrente assorbita; questo richiederà l'impiego di componenti certificati per la parte di potenza e l'applicazione dei dovuti accorgimenti per il collegamento alla rete elettrica a norma di legge;
- interfaccia utente con display LCD sulla stazione base, per poter aggiungere o rimuovere i numeri telefonici autorizzati e configurare la comunicazione con le stazioni remote;
- lo sviluppo di un software di configurazione su computer dotato di un protocollo proprietario;
- la possibilità di attivare un solo elettrodomestico con una semplice telefonata.

Allegati

- Progetto Eclipse con codice C per microcontrollori.
- Progetto CubeMX per *stazione base*.

Bibliografia

Giuseppe Treccarichi, SIM800L il cellulare via seriale (https://www.treccarichi.net/2016/06/sim800l/)

Getting time and date from GSM modem (https://elementztechblog.wordpress.com/2016/12/28/getting-time-and-date-from-gsm-modem/)

Denis S. Sovkov, Library for NRF24L01+ using HAL libraries from STMicroelectronics (https://github.com/r2aiv/NRF24L01-1)

Nordic Semiconductor, nRF24L01+ Single Chip 2.4GHz Transceiver, Preliminary product specification v1.0 *(*https://www.sparkfun.com/datasheets/Components/SMD/nRF24L01Pluss_Preliminary_Product_Specification_v1_0.pdf*)*

SIMCom, SIM800 Series_AT Command Manual_V1.01 *(*http://simcomm2m.com/En/module/detail.aspx?id=138*)*

ST Microelectronics, Datasheet STM32F103x8 *(*https://www.st.com/resource/en/datasheet/cd00161566.pdf*)*