Scuola universitaria professionale della Svizzera italiana
Dipartimento tecnologie innovative

SUPSI

Librerie C++ per RFID via dispositivo seriale in ambiente Linux

Studente/i

Claudio Bonesana

Relatore

Dipl. Ing. Giacomo Poretti

Correlatore

-

Committente

-

Corso di laurea

-

Modulo

-

Anno

2013

Data

4 settembre 2013

Indice

Elenco delle figure

Elenco delle tabelle

Elenco dei listati

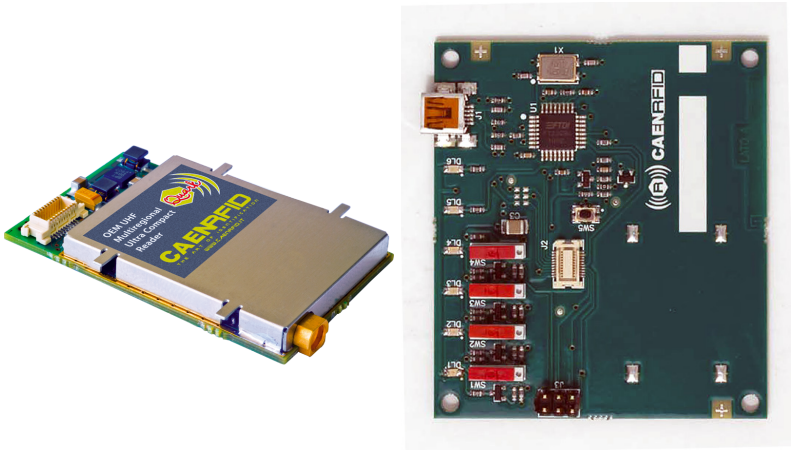
Capitolo 1

Sviluppo

1.1 Software sviluppato

Il software si compone di due parti separate ma comunicanti: un **demone** di sistema sempre attivo e un software **applicativo**.

1.1.1 Librerie RFID



(a) Lettore RFID.

(b) Modulo di sviluppo per lettore RFID.

Figura 1.1: Kit di sviluppo RFID CAEN SpA. in dotazione per il progetto.

Il kit di sviluppo della CAEN SpA. in dotazione per questo progetto è mostrato in figura 1.1 e consiste del lettore RFID Quark R1230C (a) montato su di una scheda di sviluppo (b). Il lettore RFID è praticamente autonomo e in grado di operare direttamente con i tag RFID una volta collegato ad un'antenna. La scheda di sviluppo serve unicamente come supporto per l'alimentazione e l'interfacciamento tra la seriale UART standard di cui il modulo Quark R1230C è dotato per comunicare e l'USB. Il componente che permette questo inter-

facciamento è un chip FTDI, in grado di trasferire i dati provenienti dalla UART del lettore verso l'USB e viceversa, virtualizzando una porta seriale mediante il protocollo standard VCOM. Insieme questi due elementi costituiscono il modulo RFID collegato al Raspberry Pi, essenziale al funzionamento del dispositivo.
La CAEN SpA. ha fornito una libreria con codice sorgente disponibili unicamente per Windows e scritte in C, C# e Java. In un primo momento si è pensato di eseguire il *porting* delle librerie scritte in C per Windows, l'operazione però non è riuscita a causa dell'utilizzo di librerie di sistema di Windows, non disponibili su Linux. È stato provato anche un artificio per aggirare il problema ed utilizzare le librerie di rete per comunicare con il dispositivo, senza successo. Fortunatamente la documentazione allegata al modulo descriveva in maniera esauriente il protocollo *easy2read*, utilizzato per comunicare con il dispositivo.

1.1.1.1 Protocollo easy2read

Il modulo RFID utilizza il protocollo di comunicazione proprietario di CAEN SpA **easy2read** (**??**), basato sullo schema *Attribute Value Pair* (AVP), ovvero uno scambio di pacchetti formati da piccoli coppie attributo-valore tra host e lettore RFID.
Un messaggio di questo protocollo si compone di un *header* di lunghezza fissa e di una lista di frame indicanti gli attributi con il loro relativo valore.
Gli *header*, come mostrato nel listato 1.1, sono una struttura dati di 80 bit suddivisi nei quattro campi elencati di seguito:

FIXED : (16 bit) valore fisso a $0x8001$ per i comandi e a $0x0001$ per le risposte.

Message ID : (16 bit) identificativo del messaggio. Ogni messaggio contenente un comando genera una risposta. Per identificare a quale comando corrisponde quale risposta viene utilizzato questo campo.

Vendor ID : (32 bit) per i dispositivi di CAEN SpA. il valore deve essere $0x00005358$, corrispondente al codice assegnato all'azienda dal consorzio IANA[1].

Length : (16 bit) indica la lunghezza totale del messaggio espressa in byte, *header* incluso.

```
|                   1           |        2                   3   |
|0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5|6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1|
+-------------------------------+-------------------------------+
|            FIXED              |         Message ID            |
+-------------------------------+-------------------------------+
|                           Vendor ID                           |
+-------------------------------+-------------------------------+
|            Length             |
+-------------------------------+
```

Listato 1.1: Struttura del header di un pacchetto easy2read.

[1] IANA - Internet Assigned Numbers Authority.

Dopo l'header viene accodata una lista di frame. I frame che compongono la lista di comandi sono composti secondo la struttura indicata nel listato 1.2. Le sue parti sono indicate qui di seguito:

RESERVED : (16 bit) 2 byte riservati che vanno impostati a $0x0000$.

Length : (16 bit) lunghezza complessiva dell'intero frame in byte.

Attribute Type : (16 bit) codice di 2 byte corrispondente al tipo di attributo.

Attribute Value : parte a lunghezza variabile che contiene il valore dell'attributo indicato dal frame. La lunghezza di questo campo dipende dal tipo di valore ed calcolata come $Length - 6$.

```
|                     1           |        2                    3   |
|0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5|6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1|
+-------------------------------+-------------------------------+
|            RESERVED           |            Length             |
+-------------------------------+-------------------------------+
|         Attribute Type        |       Attribute Value ...
+-------------------------------+-------------------------------+
           ... [ fino al raggiungimento di length ] ...          |
+---------------------------------------------------------------+
```

Listato 1.2: Struttura di un frame di un pacchetto easy2read.

La lista di comandi e attributi è disponibile nella descrizione del protocollo dell'**??**.

1.1.1.2 Comunicazione

La comunicazione tra lettore e *host* può avvenire su diversi canali di comunicazione: rete, seriale, Bluetooth. Le librerie sviluppate da CAEN SpA., scritte in Java, C++ e C#, sono in stretta relazione con le interfacce di rete di Windows per cui si dimostrano inutilizzabili direttamente in ambiente Linux.

Dopo alcuni tentativi infruttuosi per tentare un *porting* completo delle librerie, si è deciso di riscriverle da capo in C++, concentrandosi sulla connessione seriale come unico canale di interscambio dati tra il sistema operativo sul Raspberry Pi ed il modulo RFID. Il motivo di questa scelta è duplice: fondamentalmente è il sistema che il Raspberry Pi utilizza per comunicare fisicamente con il modulo RFID (tramite l'interfaccia USB), ma è anche il sistema più elementare per trasmettere i frame in modo controllato. Eventuali estensioni per supportare la rete o applicazioni di alto livello saranno facilmente implementabili modificando le classi che gestiscono la comunicazione seriale.

1.1.1.3 Esempio di comandi

Il protocollo easy2read contiene numerosi comandi compatibili con diversi dispositivi, moduli e lettori RFID. Anche se qui è presentato unicamente un esempio, la documentazione (**??**) offre ulteriori esempi di facile comprensione.
Come detto, la comunicazione tra lettore e *host* avviene tramite lo scambio di pacchetti. Ogni pacchetto contiene, oltre al *header*, un comando con i relativi parametri. Il nome del comando e il valore dei parametri occupano tutti un frame.
Se ad esempio si vuole impostare la potenza del segnale, occorre inviare un messaggio composto da un *header* e due frame: nel primo frame verrà indicato il comando utilizzando la coppia attributo-valore seguente:

```
Attribute Type  = CommandName          (0x0001)
Attribute Value = Set Power            (0x0064)
```

Listato 1.3: Coppia AVP per il nome del comando.

Mentre nel secondo frame verrà indicato il valore con un'altra coppia. Il valore è il massimo valore raggiungibile dal modulo RFID utilizzato.

```
Attribute Type  = Power Set            (0x0001)
Attribute Value = 199                  (0x000000C7)
```

Listato 1.4: Coppia AVP per un parametro relativo al comando.

Il messaggio completo si compone quindi della seguente struttura:

```
HEADER    Fixed           = 32769              (0x8001)
          Message ID      = 0                  (0x0042)
          Vendor ID       = 21336              (0x00005358)
          Message Length  = 28                 (0x001C)

FRAME 1   Reserved        = 0                  (0x0000)
          Length          = 8                  (0x0008)
          Attribute Type  = CommandName        (0x0001)
          Attribute Value = Set Power          (0x0064)

FRAME 2   Reserved        = 0                  (0x0000)
          Length          = 10                 (0x000A)
          Attribute Type  = Power Set          (0x0096)
          Attribute Value = 199                (0x000000C7)
```

Listato 1.5: Messaggio inviato dal host al lettore.

In risposta ai messaggi inviati dal *host*, il lettore RFID ne invierà uno simile contenente l'esito del comando ed eventualmente i parametri richiesti con il primo comando. Ovviamente, anche questa risposta sarà suddivisa in un *header* seguito da una lista di frame, contenenti questa volta una risposta. Ad esempio, la risposta al comando precedente è la seguente:

```
HEADER     Fixed           = 1                 (0x0001)
           Message ID      = 0                 (0x0042)
           Vendor ID       = 21336             (0x00005358)
           Message Length  = 26                (0x001A)

FRAME 1    Reserved        = 0                 (0x0000)
           Length          = 8                 (0x0008)
           Attribute Type  = CommandName       (0x0001)
           Attribute Value = Set Power         (0x0064)

FRAME 2    Reserved        = 0                 (0x0000)
           Length          = 8                 (0x0008)
           Attribute Type  = Result Code       (0x0001)
           Attribute Value = Success           (0x0000)
```

Listato 1.6: Messaggio in risposta proveniente dal lettore.

Analizzando il protocollo ad alto livello, i due messaggi nell'esempio del listato 1.6 non sono altro che l'equivalente di una chiamata ad un'ipotetica funzione di questo tipo:

```
bool setPower(int value)
```

1.1.1.4 Librerie implementate

Partendo dall'analisi compiuta sul protocollo, è stata scritta una libreria in C++ capace di gestire in maniera automatica i messaggi. Questa è strutturata in tre livelli distinti, gestiti da tre classi:

SerialModule classe utilizzata per comunicare con il lettore RFID attraverso un canale seriale. I parametri per stabilire la comunicazione sono indicati nella tabella 1.1.

RFIDMessage classe utilizzata per descrivere un messaggio. Questa classe viene utilizzata sia per costruire un messaggio da inviare al lettore, sia per decifrare una risposta proveniente dal lettore. La prima implementazione di questa classe fornisce unicamente i metodi per comporre il messaggio, con la possibilità di aggiungere frame utilizzando la serie di metodi `addCOmmand()`, di ricavare gli RFID letti con una richiesta di inventario con il metodo `getRFIDs()`, verificare il parametro `Result Code` della risposta con il metodo `success()`, stampare a schermo il messaggio.

RFIDModule classe utilizzata per esporre dei semplici metodi relativi ad ogni comando disponibile, nascondendo l'implementazione dello scambio dei messaggi e dell'invio attraverso la seriale. La classe offre un metodo base per l'invio e la ricezione di un messaggio qualsiasi (`sendAndRecieve()`) e una serie di comandi già implementati

nella forma vista nella sottosezione precedente. Ogni metodo ritorna il messaggio ricevuto dal lettore.

Tabella 1.1: Parametri utilizzati per la comunicazione sulla porta seriale.

Baudrate	115200 kbps
Bit dati	8
Stop bit	1
Parità	nessuna
Controllo di flusso	nessuno

L'utilizzo della libreria è pensato per semplificare la gestione e composizione dei messaggi. I passi da compiere per utilizzarla sono i seguenti:

1. Creare un oggetto di tipo `SerialModule` e configurarlo con i parametri corretti (si veda la tabella 1.1).
2. Creare un oggetto di tipo `RFIDModule` a cui verrà passato per riferimento l'oggetto di tipo `SerialModule`.
3. Utilizzare i comandi già pronti per comunicare con il lettore oppure comporre un proprio messaggio partendo da un oggetto della classe `RFIDMessage`.
4. Lavorare con l'oggetto di tipo RFIDMessage in risposta utilizzando i metodi della classe RFIDMessage.

Nella libreria sono inoltre disponibili due classi di supporto: `RFIDAttributeTypes` e `RFIDCOmmandsCode`. Queste classi non sono altro che la lista di tutti i comandi e i parametri disponibili spiegati nella documentazione del protocollo RFID. Utilizzando queste tre classi si può alleggerire il lavoro di composizione dei messaggi, passando da questa sintassi:

```
RFIDMessage *message = new RFIDMessage(0x1234);
message->addCommand(0x0001, 0x0064);
message->addCommand(0x0096, 0x000000C7);
```

a questa, più prolissa ma anche più esplicita:

```
RFIDMessage *message = new RFIDMessage(0x1234);
message->addCommand(RFIDAttributeTypes::COMMAND_NAME, RFIDCommandsCodes::SET_POWER);
message->addCommand(RFIDAttributeTypes::POWER_SET, 0x000000C7);
```

1.1.1.5 Pubblicazione

Si confida nella possibilità di fornire direttamente al produttore del modulo, CAEN SpA., queste API in modo da poter supportare appieno anche i sistemi Linux. In alternativa si spera di poter pubblicare queste API come open source.