



# **UNIVERSITÀ DEGLI STUDI DI CATANIA**

DIPARTIMENTO DI INGEGNERIA ELETTRICA, ELETTRONICA E INFORMATICA

CORSO DI LAUREA IN INGEGNERIA INFORMATICA

---

Nunzio Meli

Giuseppe Granata

**SVILUPPO DI UNA RETE PULSENET E DI UN SISTEMA PER LA GESTIONE E IL  
MONITORAGGIO DELLA RETE.**

Progetto di Reti per l'automazione industriale

---

ANNO ACCADEMICO 2013/14

# INDICE

<b>1</b>	<b>INTRODUZIONE .....</b>	<b>7</b>
1.1	SCHEMA DI RIFERIMENTO .....	8
<b>2</b>	<b>IL PROTOCOLLO PULSENET.....</b>	<b>9</b>
2.1	PULSENET .....	9
2.1.1	PROTOCOL OVERVIEW .....	9
2.1.2	SCHEMA DI RIFERIMENTO PULSENET.....	9
2.1.3	PROTOCOL .....	10
2.1.3.1	DataLink Layer .....	11
2.1.3.2	Application Layer.....	12
2.1.3.3	UART Layer.....	14
<b>3</b>	<b>IMPLEMENTAZIONE PULSENET.....</b>	<b>16</b>
3.1	PULSENET .....	16
3.1.1	DISPOSITIVI UTILIZZATI.....	16
3.1.2	PULSENET SOFTWARE ORGANIZATION.....	17
3.1.3	GENERICNODETX.....	18
3.1.4	GENERICNODERX.....	19
<b>4</b>	<b>IL GATEWAY .....</b>	<b>21</b>
<b>5</b>	<b>IMPLEMENTAZIONE GATEWAY.....</b>	<b>22</b>
5.1.1	GATEWAY SOFTWARE ORGANIZATION.....	22
5.1.2	DETTAGLI IMPLEMENTATIVI.....	25
5.1.3	INDIRECT MONITORING .....	30
5.1.4	DIRECT MONITORING .....	31
5.2	WEB APP.....	33
5.2.1	LIVE CHART.....	34
5.2.2	LIVE RECEIVED FRAME .....	36
5.2.3	DATA REPORT & SEARCH.....	37
5.2.4	SETTINGS .....	38
5.3	GESTIONE DEL DATABASE REMOTO .....	39
5.3.1	TABELLA FRAME .....	40
5.3.2	TABELLA APP_FRAME.....	40
5.3.3	TABELLA TASK .....	41
5.3.4	TABELLA APP_TYPE .....	42
5.3.5	TABELLA UART_STATE .....	42
5.3.6	TABELLA UART_TYPE .....	43
<b>6</b>	<b>SVILUPPI FUTURI.....</b>	<b>45</b>
<b>7</b>	<b>APPENDICE A: CONNESSIONE [STM32].....</b>	<b>47</b>
<b>8</b>	<b>APPENDICE B: INSTALLAZIONE E CONFIGURAZIONE GATEWAY RASPBERRY PI .....</b>	<b>49</b>
8.1.1	INTRODUZIONE .....	49
8.1.2	INSTALLAZIONE DEL SISTEMA OPERATIVO.....	51
8.1.3	ASSEGNAZIONE DI UN IP STATICO .....	53

8.1.4	RASPBERRY PI UART .....	54
8.1.5	CONFIGURAZIONE E CONNESSIONE 3G .....	56
8.1.6	SAKIS3G .....	57
<b>9</b>	<b>BIBLIOGRAFIA .....</b>	<b>59</b>

## INDICE FIGURE

Figura 1 –Schema di riferimento.....	8
Figura 2 –Scenario Applicativo .....	9
Figura 3 –Protocol Overview.....	10
Figura 4 –DataLink Layer Frame .....	11
Figura 5 –Application Layer Frame.....	12
Figura 6 –Dati per ogni nodo .....	13
Figura 7 –Stack Protocollore Nodo Ricevitore.....	14
Figura 8 –UART Frame .....	15
Figura 9 –Scenario.....	17
Figura 10 –GenericNodeTx .....	18
Figura 11 –GenericNodeRx .....	19
Figura 12 –Gateway Diagramma delle classi.....	22
Figura 13 –Main Class .....	25
Figura 14 –Thread Network Connect.....	26
Figura 15 –Thread Consumer .....	27
Figura 16 –Thread Task Manager .....	27
Figura 17 –Thread Socket Manager.....	28
Figura 18 –Thread Client Connect .....	29
Figura 19 –Indirect Monitoring.....	30
Figura 20 –Direct Monitoring (FASE A) .....	31
Figura 21 –Direct Monitoring (FASE B) .....	32
Figura 22 –Control Panel .....	33
Figura 23 –Live Chart.....	35
Figura 24 –Live Chart Particular .....	35
Figura 25 –Live Received Frame .....	36
Figura 26 –Live Received Frame Particular .....	36
Figura 27 –Data Report.....	37
Figura 28 – Numero di pagine .....	37
Figura 29 –Search in Data Report .....	37
Figura 30 –Search in Data Report .....	38
Figura 31 –phpMyAdmin.....	39
Figura 32 –phpMyAdmin dump.....	39
Figura 33 –Tabella Frame .....	40
Figura 34 –Tabella app_frame .....	40
Figura 35 –Tabella task.....	41

Figura 36 –Tabella app_type .....	42
Figura 37 –Tabella uart_state.....	42
Figura 38 –Tabella uart_type .....	43
Figura 39 –Sviluppi futuri .....	45
Figura 40 – Connessione STM32 v1 con Nrf24l01 .....	47
Figura 41 –Raspberry Pi.....	49
Figura 42 – Il desktop di Raspbian .....	52
Figura43 – Porta Seriale Pinout .....	54
Figura 44 – Listato Read2Serial.p .....	55
Figura 45 – Huawei E220.....	56

## INDICE TABELLE

Tabella 1– Connessione STM32v1 con Nrf24l01 .....	47
Tabella 2– Caratteristiche tecniche del Raspberry Pi.....	51
Tabella 3– Tabella APN.....	58

# 1 INTRODUZIONE

Una Wireless Sensor Network è un'insieme di nodi dotati di CPU, memoria, radiotrasmettitore e sensori che effettuano delle misurazioni dell'ambiente(sensing) e le trasmettono ad un nodo centrale, che le inoltra ad un sistema di elaborazione remoto. Il vantaggio principale di una WSN è che si tratta una rete a basso consumo energetico e il costo è decisamente basso. Inoltre queste reti possono arrivare a coprire aree molto estese e possono rimanere attive per molto tempo. Quindi le WSN permettono di risparmiare parecchio sui costi e permettono di monitorare aree molto estese e soggette a situazioni ambientali estreme, difficili da raggiungere. E' chiaro quindi che queste reti possono avere molteplici applicazioni: ad esempio possono essere utilizzate nella cosiddetta agricoltura di precisione. In tale ambito, la rete di sensori è impiegata per monitorare i parametri ambientali di un campo agricolo, in modo che l'agricoltore sappia quando irrigarlo o applicare determinati pesticidi. Inoltre le WSN possono essere impiegate in altri scenari: militari, di sorveglianza, domotica ecc. [2]

In questa tesina si parlerà dello sviluppo e implementazione di un protocollo PulseNet per WSN e del sistema di monitoraggio della rete.

Di seguito viene illustrato lo schema di riferimento del sistema sviluppato in questa tesina, che dovrà esser tenuto ben in mente durante la lettura di questo trattato.

Per quanto riguarda i capitoli successivi, in una prima parte verrà descritto il protocollo PulseNet e la relativa implementazione, mentre in una seconda parte si discuterà dell'integrazione tra la rete PulseNet e un Gateway, e della progettazione e implementazione di questo dispositivo, utile per connettere la PulseNet alla rete Internet.

## 1.1 Schema di riferimento

La PulseNet Wireless Sensor Network implementata è costituita da:

- N nodi di trasmissione
- 1 nodo di ricezione
- 1 Gateway

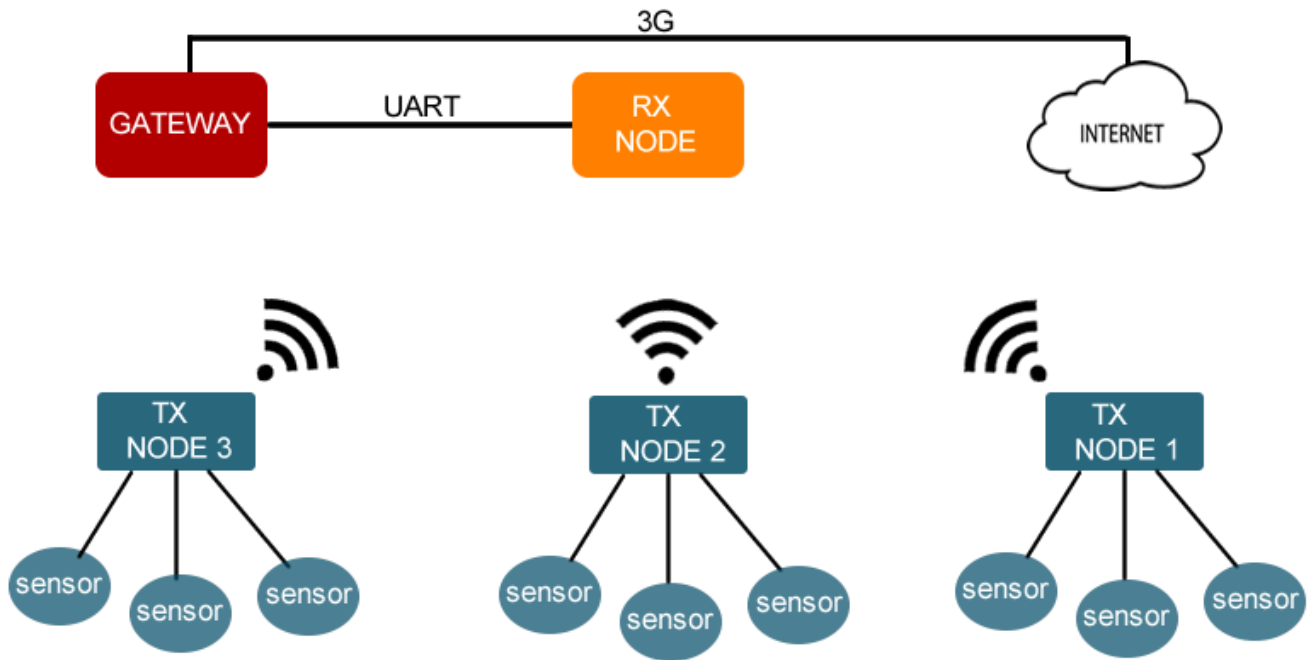


Figura 1 –Schema di riferimento

I nodi di trasmissione hanno il compito di raccogliere i dati dai sensori e di inviarli al nodo ricevitore. Il nodo ricevitore ha il compito di inviare i dati che riceve ad un gateway, che a sua volta andrà a memorizzare i dati in un database remoto.



## 2 IL PROTOCOLLO PULSENET

### 2.1 PulseNet

#### 2.1.1 Protocol Overview

Il protocollo PulseNET si pone l'obiettivo di realizzare un meccanismo di trasmissione wireless di tipo single-hop dove si hanno un numero indefinito di nodi che trasmettano ed un unico nodo che riceve. Le informazioni che si vogliono trasferire sono relative ad informazioni misurate da sensori ambientali aventi caratteristiche eterogenee sia in termini di tipologia del dato, ma soprattutto in termini di dinamica temporale. Il protocollo ha quindi come obiettivo quello di definire un meccanismo efficace per rappresentare e trasferire le informazioni provenienti dai sensori. [4]

#### 2.1.2 Schema di riferimento PulseNet

Lo scenario applicativo si compone di numerosi nodi trasmettitori, ogni nodo trasmettitore è fisicamente connesso ad uno o più dispositivi sensori in grado di raccogliere informazioni dall'ambiente. Questo scenario è presentato in Figura 2. Il nodo Sink è in grado di ricevere le informazioni rilevate dai vari nodi che sono presenti all'interno della propria copertura radio. Questo nodo dovrà essere non solo in grado di elaborare le informazioni provenienti dai dispositivi, ma anche di segnalare la non raggiungibilità di un nodo precedentemente rilevato.

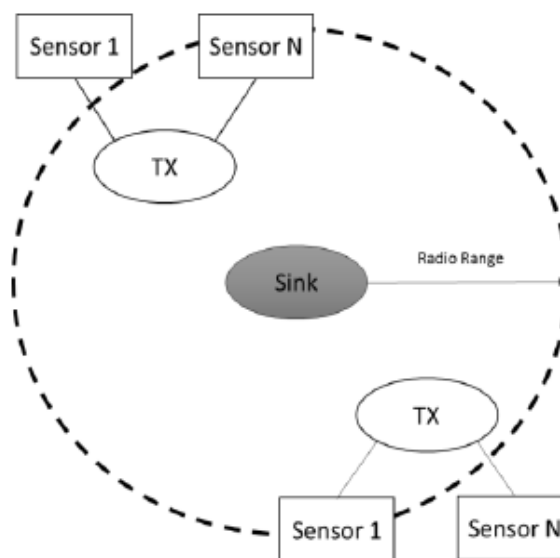


Figura 2 –Scenario Applicativo

Il protocollo implementato rispetta le seguenti caratteristiche:

- Alta flessibilità ed estensibilità del protocollo.
- Capacità da parte dei trasmettitori di trasferire informazioni relative a più sensori, in maniera efficace e in accordo alle problematiche di misura delle informazioni.
- Alta efficienza di frame e relativo utilizzo oculato della banda trasmissiva.
- Payload di lunghezza variabile per supportare le caratteristiche di flessibilità.

Per rispettare le regole sopra descritte sono stati definiti due livelli di protocollo, un livello Data Link (DLL) che si occupa di trasferire le informazioni relative al nodo fisico ed un Application Layer avente lo scopo di trasferire le informazioni rilevate dai sensori.

Il tipo di protocollo essendo unidirezionale, senza conferma e single-hop, non ha la necessità degli altri livelli del modello ISO/OSI. [4]

### 2.1.3 Protocol

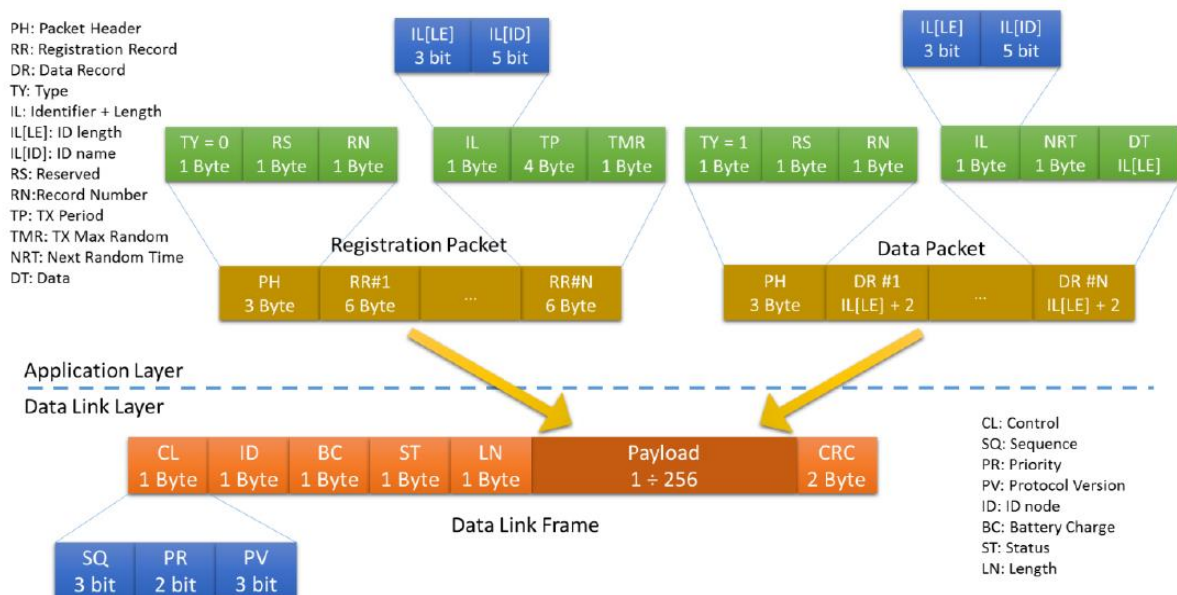


Figura 3 –Protocol Overview

### 2.1.3.1 DataLink Layer

A livello DataLink, la frame è dotata di un header che serve ad identificare il tipo di frame e il nodo trasmettitore. L'ultimo campo della frame è il CRC, che permette di verificare la correttezza della frame.

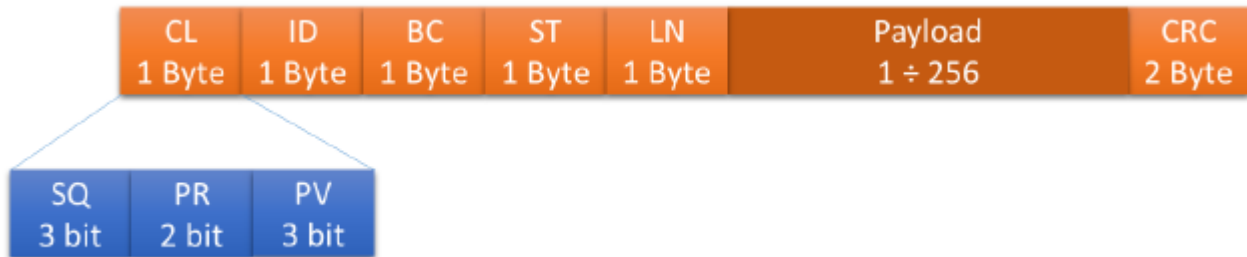


Figura 4 –DataLink Layer Frame

- **CL** Control – Campo di tipo bitmask:
- **SQ**: Sequence Number. *Bit 0,1,2*(Mask 0xE0).
- **PR**: Priority. *Bit 3,4*(Mask 0x18).
- **PV**: Protocol Version. *Bit 5,6,7* (Mask 0x07).
- **ID** ID Node – Identificatore univoco del nodo trasmettitore.
- **BC** Battery Charge – Carica della batteria del nodo sorgente
- **ST** Status – Byte di stato del dispositivo
- **LN** Length – Lunghezza del payload in range [1÷256]
- **PL** Payload – Carico informativo della frame
- **CRC** – Campo di check della frame.

La frame descritta in Figura 4 presenta diverse caratteristiche:

- Il payload è a lunghezza variabile, quindi potrà trasferire contenuto informativo in accordo alle necessità funzionali.
- Nessuna presenza dell'end e start delimiter, ma la frame è delimitata in accordo alla lunghezza.

Come descritto a livello data link avremo un'unica frame. Essa è dotata di un header che serve ad identificare il nodo trasmettitore, un payload di lunghezza variabile (1÷256 byte) in grado di trasferire contenuto informativo in accordo alle necessità funzionali, un CRC di verifica della correttezza posto alla fine di essa. [4]

### 2.1.3.2 Application Layer

L'Application Layer presenta tutte le funzionalità per registrare e trasferire le informazioni acquisite dai nodi. In particolare le fasi del trasmettitore saranno due: una prima fase di registrazione nella quale il trasmettitore dichiara le caratteristiche delle variabili che vuole trasferire in termini sia di numero e tipo di variabili che di dinamica temporale ed una seconda fase operativa nella quale trasmette il contenuto informativo rilevato dai propri sensori.

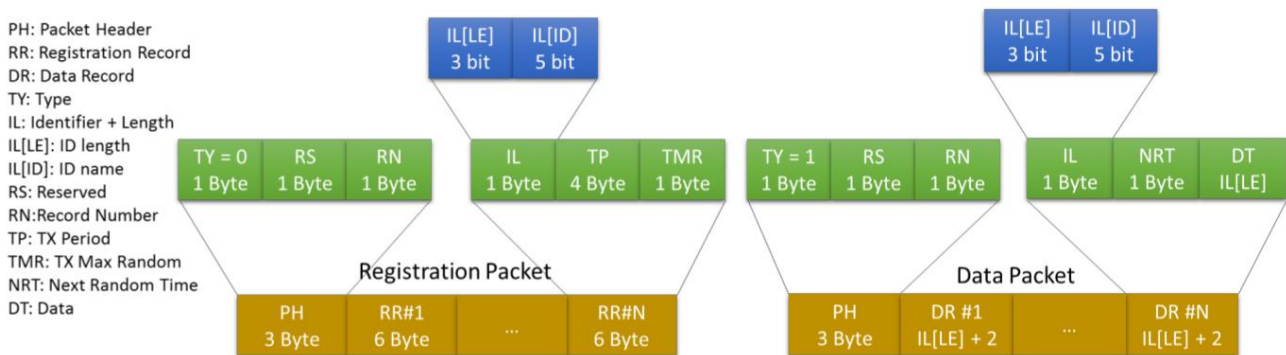


Figura 5 –Application Layer Frame

Un pacchetto di registrazione dovrà essere in grado di trasportare le informazioni relative alle variabili associate ai sensori del nodo stesso. Per caratterizzare una variabile si usa un campo compatto di un byte, IL (in Figura 5), che contiene l'identificativo della variabile nei 5 bit più significativi e la lunghezza nei 3 bit meno significativi. Definito il formato della variabile, il pacchetto di registrazione dovrà contenere anche le informazioni relative alle caratteristiche temporali di generazione della singola variabile. In particolare, il pacchetto di registrazione si compone di un header contenente la tipologia del pacchetto, un campo riservato per usi futuri e un campo RN indicante il numero di record di registrazione contenuti nel pacchetto. Un singolo record di registrazione conterrà tutte le informazioni relative alle caratteristiche della variabile, ovvero conterrà il campo IL descritto in precedenza, informazioni relative la cadenza temporale di trasmissione della variabile (TP: Trasmission Period) e un altro parametro TMR, che indica lo scostamento temporale massimo casuale di invio frame relativamente al TP. In particolare il trasmettitore trasmetterà una variabile all'interno di un intervallo che va da TP a TP + TMR con cadenza periodica pari a TP. La presenza di un intervallo variabile di trasmissione ha lo scopo di minimizzare il rischio di collisione fra trasmettitori che possiedono lo stesso Transmission period.

Il pacchetto dati avrà un formato analogo a quello di registrazione, mantenendo lo stesso header e cambiando il formato del record. I record del pacchetto dati conterranno il campo IL, il campo NRT, che indica il prossimo scostamento random rispetto al TP, in modo da far calcolare al ricevitore l'esatto istante della prossima trasmissione della variabile in esame, ed il campo DT di lunghezza variabile in accordo ad IL che trasporta l'informazione proveniente dal sensore. In particolare, grazie al campo NRT il ricevitore sarà in grado di prevedere l'istante di inizio della prossima trasmissione da parte di un nodo sensore. Ciò permetterà, in futuro, di implementare una politica di risparmio energetico nel ricevitore che non avrà bisogno di essere sempre acceso, ma che si accenderà solo in prossimità della trasmissione da parte di un nodo sensore. Inoltre, poiché il ricevitore ha la visibilità dei tempi di trasmissione di tutti i nodi sensore registrati, esso potrà prevedere possibili collisioni fra due o più nodi sensore ed in tale evenienza potrà decidere di non accendersi, avendo valutato che la trasmissione delle informazioni sarà affetta da errore. [4]

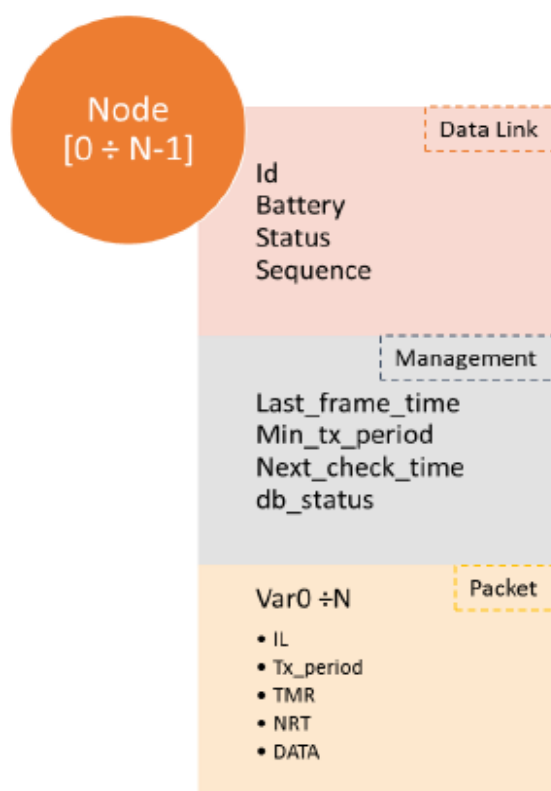


Figura 6 –Dati per ogni nodo

Come detto in precedenza, le frame vengono generate dai nodi trasmettitori ed inviate al nodo ricevitore, che effettuerà una interpretazione dei dati ricevuti.

A fine interpretazione sarà necessario memorizzare le informazioni relative i nodi. Per far ciò, il ricevitore deve implementare un'opportuna struttura dati in grado di ospitare queste informazioni.

In Figura 6 viene mostrata la definizione della struttura necessaria al ricevitore per memorizzare le informazioni inerenti le variabili del nodo. Affinché il ricevitore accetti le informazioni relative al nodo è necessario che abbia ricevuto precedentemente dei pacchetti di registrazione.

Ogni elemento della struttura dati del ricevitore contiene delle indicazioni di gestione che servono per fornire lo stato delle informazioni comunicate dal nodo. In particolare, il campo db\_status indica lo stato delle informazioni relative al nodo identificato dal campo ID. Esso può assumere valore EMPTY nel caso la struttura del singolo nodo sia preallocata e non utilizzata, VALID se la struttura contiene informazioni di un nodo attivo o DEATH se la struttura contiene le informazioni di un nodo considerato non più raggiun-

gibile. Tra le informazioni di management troviamo anche il campo `last_frame_time`, che indica l'istante nel quale è stata ricevuta l'ultima frame del nodo corrente. [4]

Il ricevitore, a sua volta, ingloba le informazioni ricevute in una frame che spedisce tramite UART al gateway.

#### 2.1.3.3 UART Layer

A differenza del nodo trasmettitore che invia i dati soltanto lungo il mezzo wireless, il nodo ricevitore riceve i dati dal mezzo wireless e li smista al gateway mediante interfaccia seriale. A tale scopo lo stack protocollare del ricevitore è leggermente diverso rispetto a quello del nodo trasmettitore.

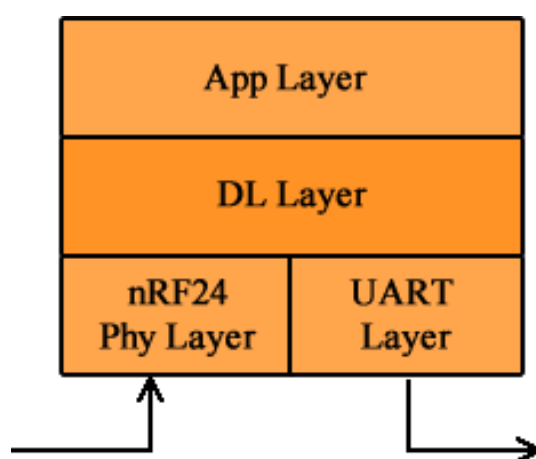
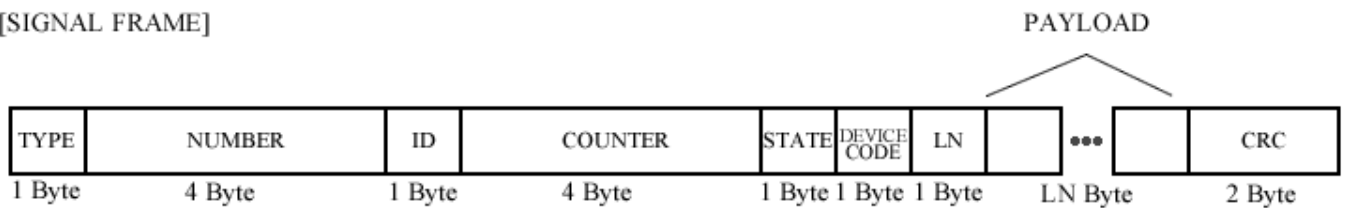


Figura 7 –Stack Protocollare Nodo Ricevitore

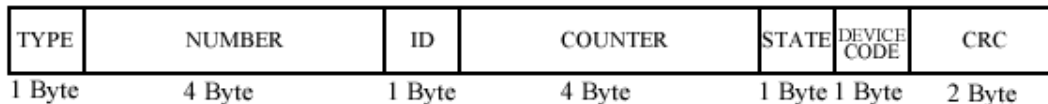
Come si nota dall'immagine, si hanno due diversi livelli fisici: un livello dove la comunicazione avviene per onde radio ed un secondo livello che è quello che vede come mezzo trasmissivo l'UART e che permette la comunicazione tra la rete di sensori ed il Gateway.

Per l'invio dei dati lungo l'interfaccia seriale, sono state definite due tipologie di frame all'interno dell'UART Layer: la `SIGNAL FRAME` e la `ALERT FRAME`.

[SIGNAL FRAME]



[ALERT FRAME]



**TYPE** is SIGNAL FRAME or ALARM FRAME.

**NUMBER** is the number of USART frame.

**ID** is the id of the node that send frame to rx device.

**COUNTER** is the time value of the rx device.

**STATE** if the TYPE is Signal Frame this can assume CRC OK if the DLFrame is correct or CRC ERROR if the DLFrame is not correct, otherwise if the TYPE is ALARM FRAME this value can be assume DEAD\_NODE, VALID\_NODE e NODE\_NOT\_REGISTERED.

**DEVICE CODE** is the code of the rx device.

**LN** is the payload length in byte.

**PAYLOAD** this value contain a DLFrame.

**CRC** is the CRC of the usart frame.

Figura 8 –UART Frame

La Signal Frame è una frame di segnalazione che ingloba una frame di livello Data Link con il CRC corretto.

Le Alert Frame, invece, sono delle frame che si differenziano dalle Signal Frame, in quanto non hanno il campo payload. Queste frame vengono generate ogni volta che un nodo muore o torna attivo oppure quando un nodo invia Data Frame, nonostante questo non sia registrato nella rete. Viene generata una Alert Frame anche quando viene ricevuta una frame con il CRC errato.

### 3 IMPLEMENTAZIONE PULSENET

#### 3.1 PulseNet

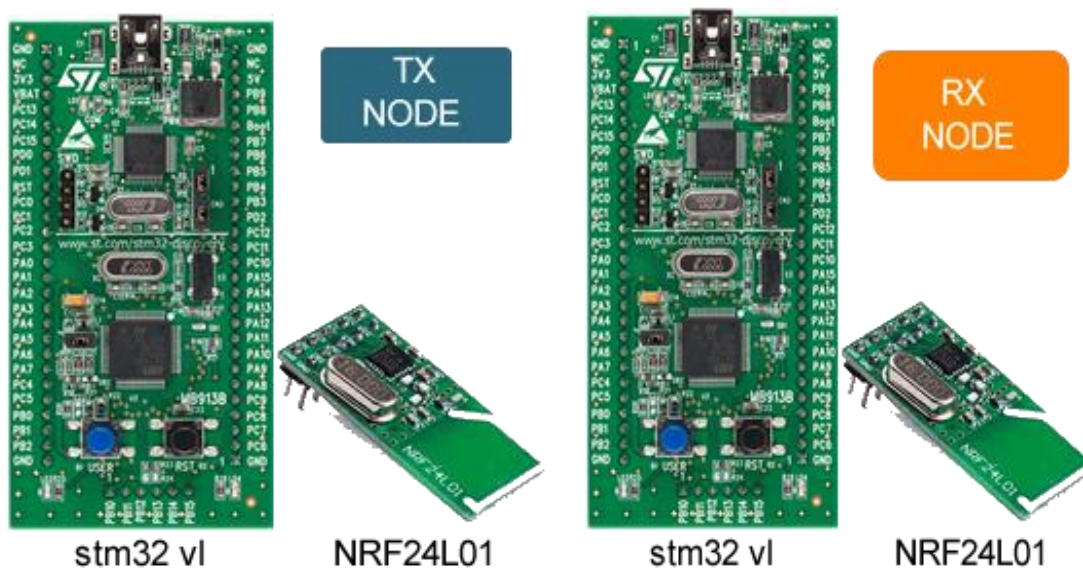
Al fine di valutare e verificare le caratteristiche trasmissive del protocollo, proporremo ora un'implementazione effettuata su dispositivi low-power.

##### 3.1.1 Dispositivi utilizzati

I nodi di trasmissione e di ricezione sono costituiti da un microcontrollore STM32F100RB con le seguenti caratteristiche:

- STM32F100RB Microcontrollore, 128 KB Flash, 8 KB RAM in 64-pin LQFP;
- On-board ST-Link;
- Alimentazione via USB;
- 2 LED;
- Un pulsante;

[3]





I nodi di trasmissione e ricezione sono collegati tramite SPI con i transceivers nRF24l01.

Il nRF24L01 è un ricetrasmittitore a bassissimo consumo energetico (Ultra Low Power) con datarate di 2 Mbps alla frequenza di 2,4 GHz. Con le correnti di picco RX / TX inferiore a 14mA, una modalità di spegnimento, e un range di alimentazione da 1,9 a 3,6 V, il nRF24L01 fornisce una soluzione Ultra Low Power, permettendo di raggiungere una durata di mesi e anni della batteria durante il funzionamento a pile a bottone o AA / AAA. Il nRF24L01 integra un completo ricetrasmittitore RF a 2,4 GHz, sintetizzatore RF, il protocollo hardware ShockBurst e supporta, inoltre, un' interfaccia SPI ad alta velocità per il controller dell'applicazione.

### 3.1.2 PulseNet Software Organization

Il protocollo si presta ad essere implementato su diverse categorie di dispositivi embended, a tale scopo è stata definita la parte protocollare come libreria da integrare all'interno del progetto. La libreria è stata implementata in ANSI C per essere fruibile su un gran numero di dispositivi. Essa si basa su funzioni e strutture dichiarate su appositi file header. Nell'implementazione si è posta particolare cura a non utilizzare librerie esterne, in modo che la fruizione delle funzionalità non fosse dipendente dal particolare ambiente.

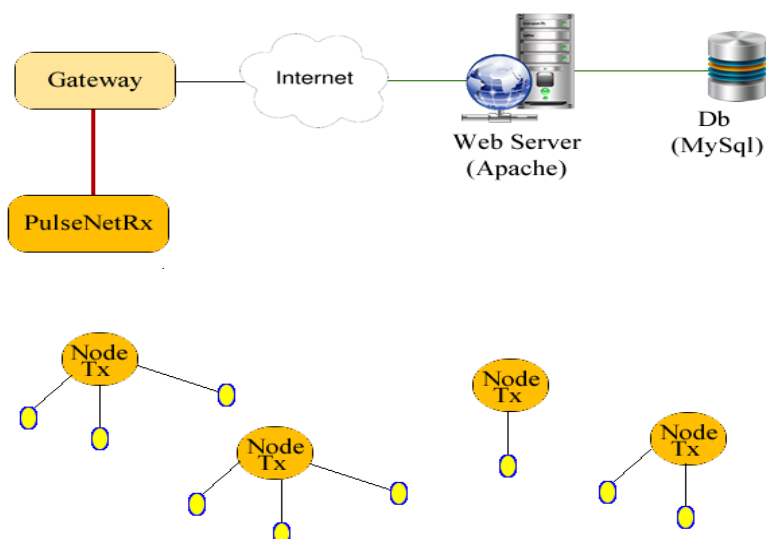


Figura 9 –Scenario

### 3.1.3 GenericNodeTx

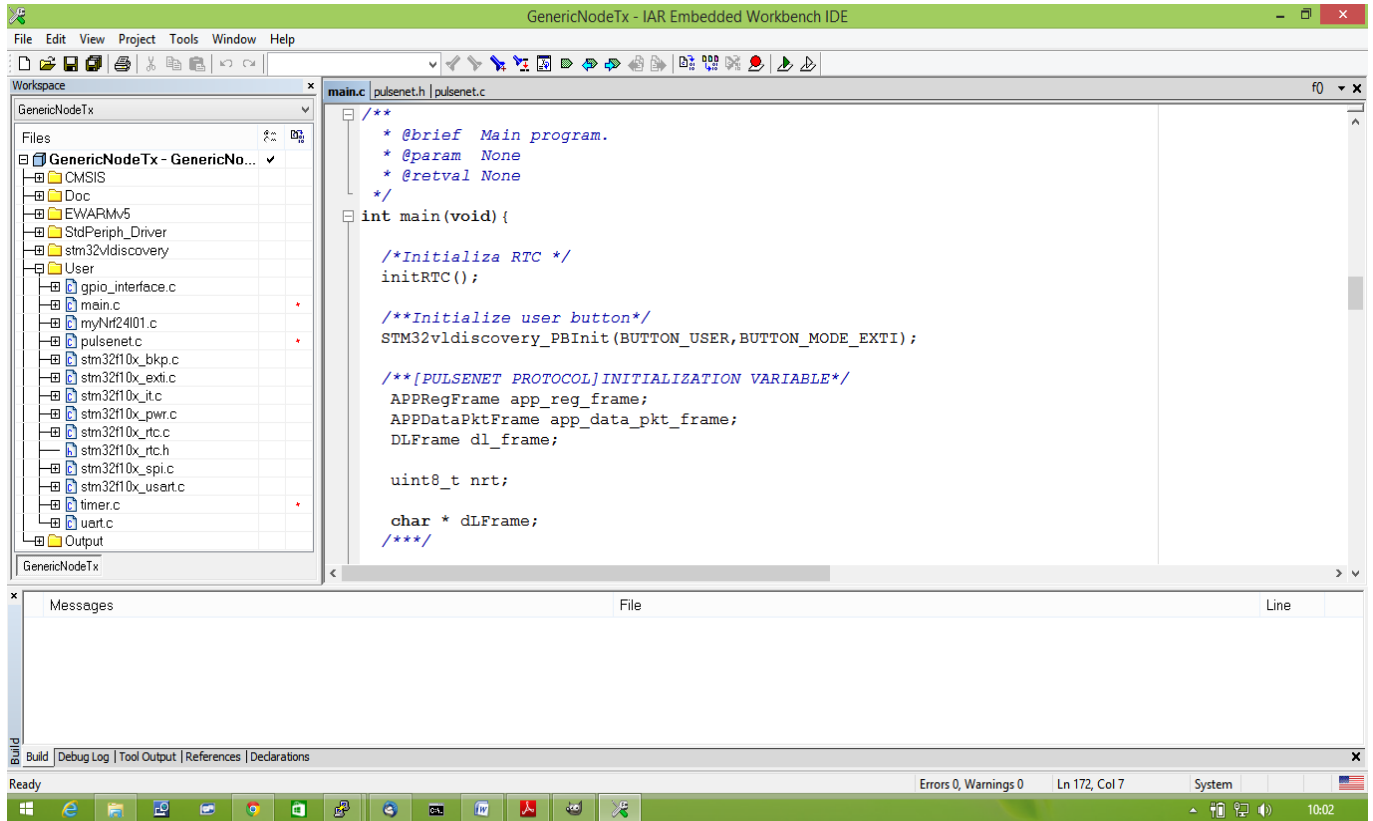


Figure 10 –GenericNodeTx

Il trasmettitore è stato progettato in accordo al protocollo, in modo che possa inviare sia pacchetti di registrazione che pacchetti dati. Inizialmente si è utilizzato il pulsante presente sulla scheda per effettuare questo cambio di modalità, ma si è visto che il pulsante generava un “rimbalzo”, cambiando più volte modalità. Per ovviare a ciò, si è deciso di impostare il pin PA2 in modalità lettura. Quando si ponticella il pin dei 5 V con il pin PA2, il micro legge una tensione di 5 V sul pin e imposta il micro in modalità registrazione. Quando il ponte viene rimosso, è nuovamente attiva la modalità di invio dati e, inoltre, sul trasmettitore è stato implementato un meccanismo di risparmio energetico, secondo il quale il trasmettitore si accende solo quando deve effettivamente trasmettere; finita la trasmissione torna in standby.

### 3.1.4 GenericNodeRx

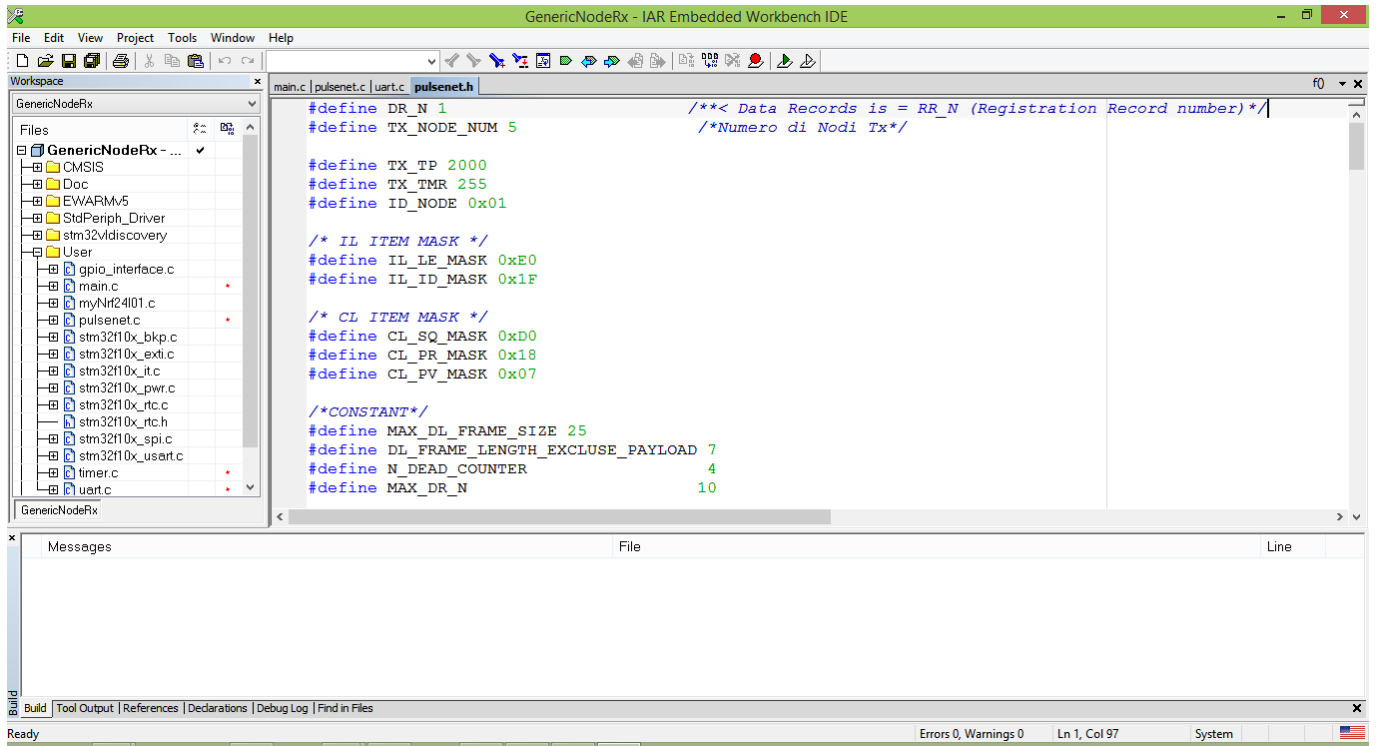


Figura 11 –GenericNodeRx

Il nodo ricevitore ha il compito di ricevere le frame da tutti gli altri nodi e salvare le informazioni ricevute in una struttura dati opportuna, che verrà poi utilizzata per creare le UART Frame che invierà al Gateway per il push sul Database. Il nodo ricevitore ha, inoltre, il compito di controllare lo stato dei nodi e contrassegnare come “DEATH NODE” i nodi di trasmissione caratterizzati da inattività.

L’algoritmo utilizzato per effettuare la verifica dei nodi morti fa uso del RealTime Clock (RTC).

Ad ogni frame ricevuta da un nodo trasmettitore, il ricevitore calcola, per quel nodo, un valore di `next_check_time`, con la formula di seguito mostrata:

$$\text{next\_check\_time} = \text{min\_tx\_period} * N$$

Dove N è un numero scelto in fase di progettazione.

Più grande è N, maggiore sarà il tempo che il nodo ricevitore attenderà prima di giudicare un nodo morto.

Alla ricezione di una frame da un determinato nodo, il parametro `next_check_time` conterrà un intervallo temporale pari a N volte il periodo della frame a più alta frequenza per un dato nodo.

Tramite un processo temporizzato, basato su interrupt, questo parametro andrà a decrementarsi ogni secondo. Non appena il `next_check_time` raggiungerà valore 0, sarà trascorso un tempo pari a N volte il `min_tx_period` senza ricevere ulteriori dati, il che significa che il nodo può essere dichiarato non più raggiungibile.

## 4 IL GATEWAY

Il Gateway ha il compito di effettuare il push sul database remoto dei dati che riceve, lungo l'interfaccia seriale, dal nodo ricevitore della rete PulseNet.

Il gateway accede ad Internet per mezzo di un modem 3G secondo due modalità:

- Modalità periodica: La connessione ad Internet avviene in modo periodico in base al periodo che viene letto da un file di configurazione, il che significa che il gateway si connette ad Internet solo per effettuare il push dei dati sul database e si disconnette subito dopo. In questo modo si riesce a garantire un basso consumo energetico;
- Modalità permanente: La connessione ad Internet viene avviata per un intervallo di tempo indefinito. In questa modalità il push dei dati sul database remoto viene effettuato quasi ad ogni frame, che viene ricevuta dal nodo ricevitore della PulseNet.

Il Gateway può essere facilmente configurato settando il valore di alcuni registri.

All'avvio, il gateway legge il file dei registri e ripristina la configurazione che si aveva prima che il gateway venisse spento. Per fare questo nel Gateway è stato creato un file contenente i valori dei registri in formato JSON.

Il file appare in questo modo:

```
{ "PeriodicTx": "10", "ListenPort": "4000" }
```

Un' interfaccia web è stata sviluppata per permette all'utente di configurare e monitorare tutta la rete.

Grazie a questa applicazione è possibile vedere in realtime i dati che sono trasmessi dai nodi sensore della Pulsenet, graficare i valori rilevati dai sensori e impostare il valore dei registri.

Nel capitolo successivo, le caratteristiche del Gateway verranno esposte con maggiore dettaglio.

## 5 IMPLEMENTAZIONE GATEWAY

### 5.1.1 Gateway Software Organization

L'attività di Gateway è svolta dal Raspberry Pi. Tutto il codice per il funzionamento del Gateway è stato scritto in Python. Presentiamo adesso il diagramma delle classi del Gateway.

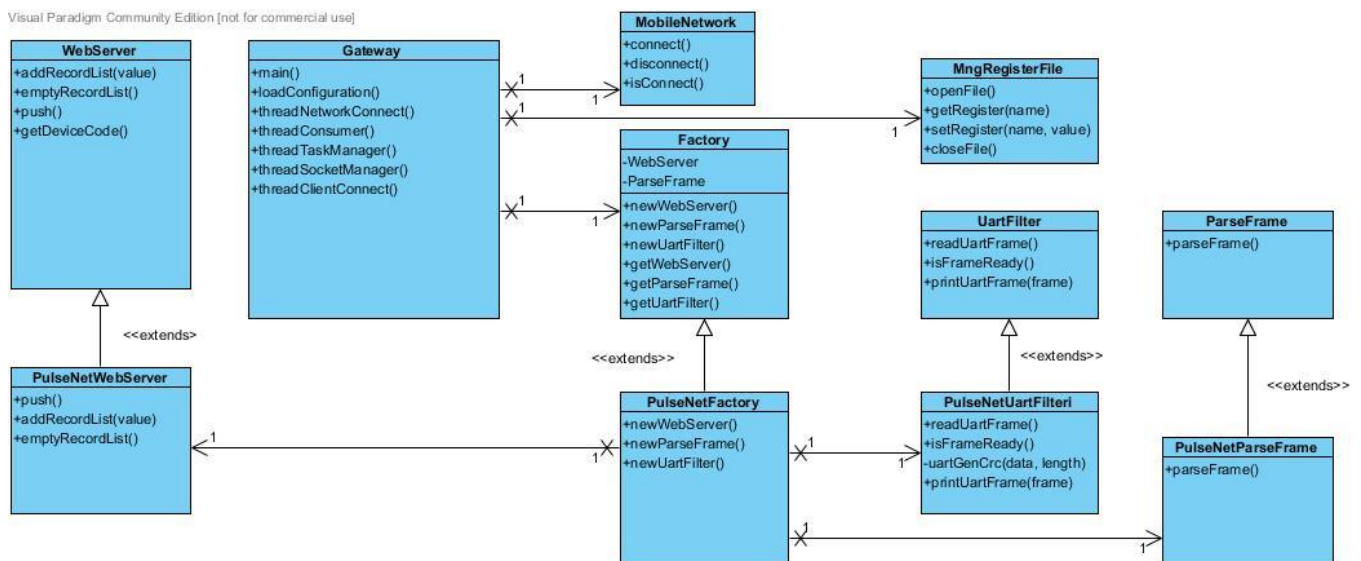


Figura 12 –Gateway Diagramma delle classi

Nell'implementazione del Gateway sono stati applicati due importanti pattern creazionali, quali il Factory Method ed il Singleton. Illustriamo adesso i metodi pubblici di ogni classe.

**WebServer:** classe astratta, che fornisce la signature dei metodi che verranno implementati nella classe figlia e che ha il compito di definire un'interfaccia per gestire l'interazione con il web server.

**PulseNetWebServer:** classe figlia di **WebServer**, caratterizzata dai metodi:

- `addRecordList(value)`: questo metodo ha il compito di inserire un nuovo elemento, in formato JSON, in una lista che contiene tutti i record che in seguito verranno inseriti nel database remoto;
- `emptyRecordList()`: questo metodo ha il compito di svuotare la lista contenente tutti i record. Esso viene invocato ogni volta che si verifica un'azione di push sul database remoto.

- `push()`: metodo che si occupa di inserire nuovi record nel database.

**MobileNetwork:** classe che si occupa di avviare la procedura di connessione ad Internet e la procedura di disconnessione da Internet. In pratica questa classe interagisce con l'applicazione “sakis3G”, la quale gestisce la comunicazione e il coordinamento del modem 3G.

Essa implementa i diversi metodi:

- `connect()`: metodo che si occupa di avviare la connessione ad Internet;
- `disconnect()`: metodo che si occupa di effettuare la disconnessione da Internet;
- `isConnect()`: metodo booleano che verifica se il dispositivo è connesso ad Internet. Torna “1” se risulta connesso, altrimenti torna “0”.

**MngRegisterFile:** classe che ha il compito di gestire le operazioni di lettura e scrittura sui dei registri salvati su un file in formato JSON. In particolare, i registri possono essere permanenti o volatili. I registri permanenti sono registri il cui valore non viene resettato quando il dispositivo viene riavviato, mentre i registri volatili sono registri il cui valore viene perso una volta che il dispositivo viene riavviato.

Ogni registro è caratterizzato da un nome, un valore, una modalità di memorizzazione (permanente o volatile) e un tipo di memorizzazione (buffer o FIFO)

Analizziamo adesso i diversi metodi della classe:

- `openFile()`: metodo che ha il compito di aprire il file “register.json” in modalità scrittura/lettura;
- `getRegister(name)`: metodo che ha il compito di recuperare il valore da un registro;
- `setRegister(name,value)`: metodo che ha il compito di impostare il valore di un registro;
- `closeFile()`: metodo che si occupa di effettuare la chiusura del file;

**UartFilter:** classe astratta che fornisce la signature dei metodi per l'interazione con l'UART.

**PulseNetUartFilter:** classe che estende UartFilter.

Implementa i seguenti metodi:

- `readUartFrame()`: metodo che ha il compito di leggere e di fare il parsing della frame ricevuta dalla seriale. Questo metodo restituisce un dizionario;
- `isFrameReady()`: metodo booleano che verifica se il parsing di una frame è stato completato. Se il parsing è completo il metodo restituisce “1”, altrimenti restituisce “0”;
- `printUartFrame(frame)`: metodo che ha il compito di stampare l'intera frame;
- `uartGenCrc(data,length)`: metodo che calcola il CRC;

**ParseFrame:** classe astratta che fornisce la signature del metodo parseFrame;

**PulseNetParseFrame:** classe che estende ParseFrame; ridefinisce il metodo parseFrame() ed effettua il parsing della frame dei livelli più alti.

**Factory:** classe astratta necessaria per realizzare il design pattern creazionale Factory Method();

**PulseNetFactory:** classe che estende Factory e restituisce le istanze di PulseNetWebServer, PulseNetUartFilter e PulseNetParseFrame.

**Gateway:** classe che contiene il main().



### 5.1.2 Dettagli implementativi

Di seguito, in Figura 13, viene illustrato il diagramma che mostra il comportamento della funzione main. Il primo metodo chiamato è il metodo `loadConfiguration()`, che ha il compito di effettuare la lettura dai registri permanenti e, quindi, ripristinare la configurazione che si aveva prima che il gateway venisse spento. Successivamente viene istanziata la classe `PulsenetFactory`, che verrà utilizzata per istanziare varie classi specifiche per la rete PulseNet.

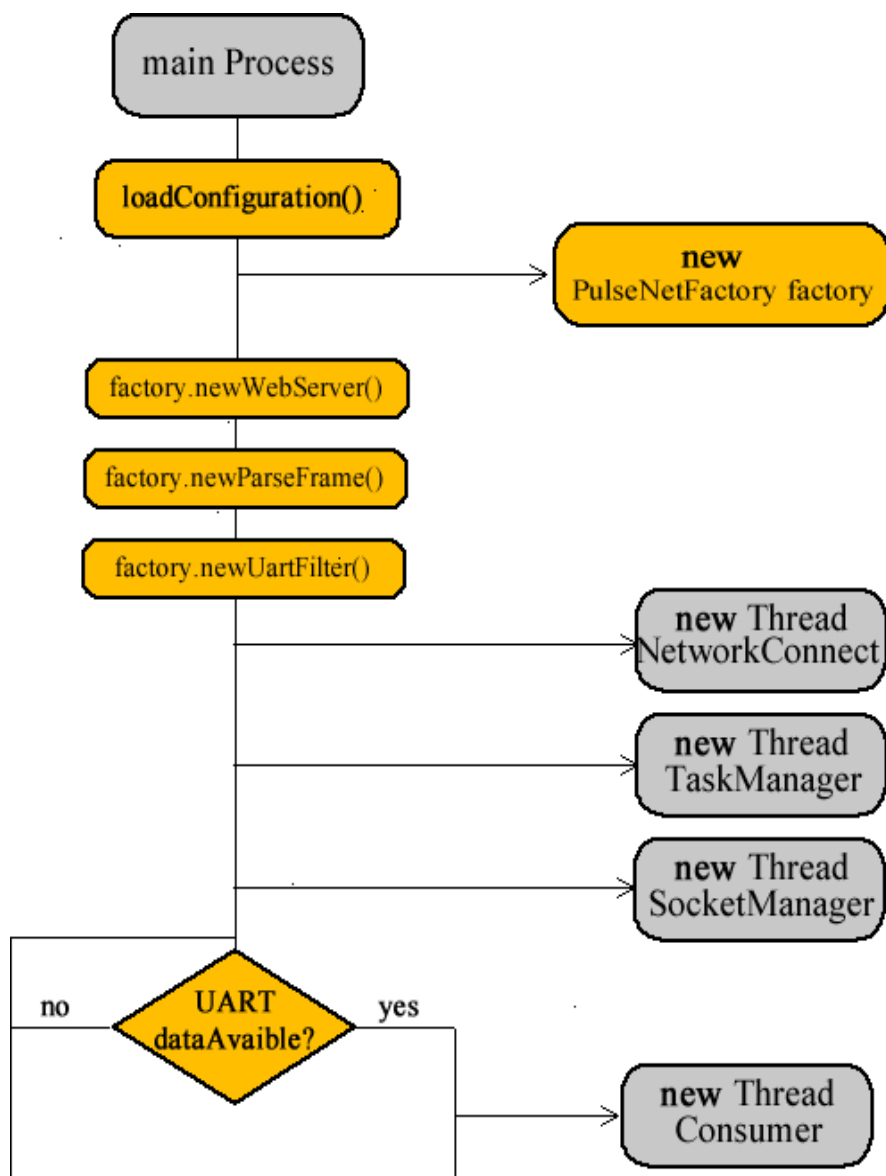


Figura 13 –Main Class

Il main() ha inoltre il compito di avviare l'esecuzione di diversi thread, che verranno di seguito illustrati:

- threadNetworkConnect(): thread che ha il compito di gestire la connessione ad Internet in due modalità:
  1. Modalità periodica: Il dispositivo si collega ad Internet solo per effettuare il push sul database remoto e dopo si disconnette;
  2. Modalità permanente: Il dispositivo si collega ad Internet per un periodo indefinito;

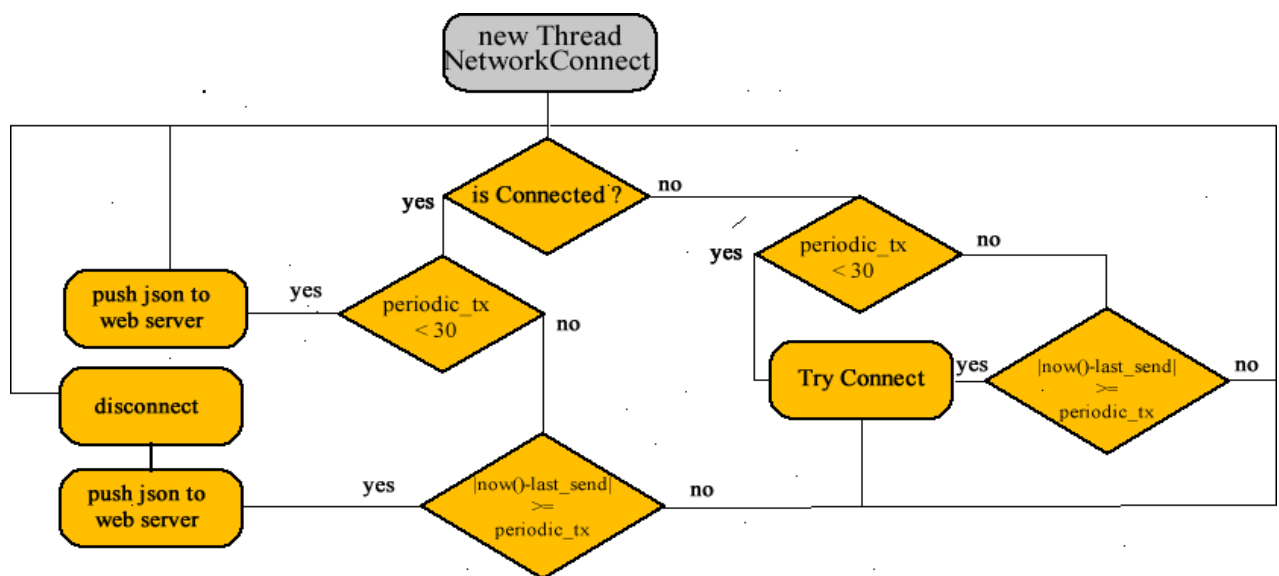


Figura 14 –Thread Network Connect

- threadConsumer(): thread che ha il compito di inserire in coda ogni record che è pronto per essere memorizzato sul database;

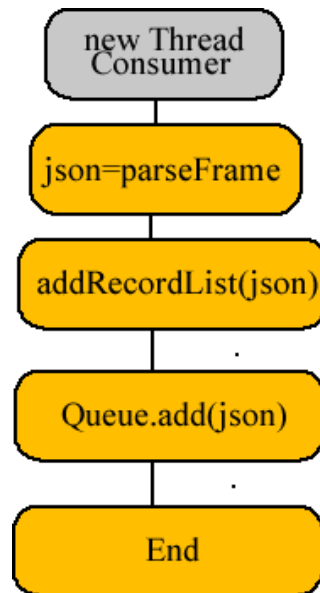


Figura 15 –Thread Consumer

- threadTaskManager(): thread che ha il compito di leggere i task, che vengono scritti dal gestore della rete nel database mediante la web application;

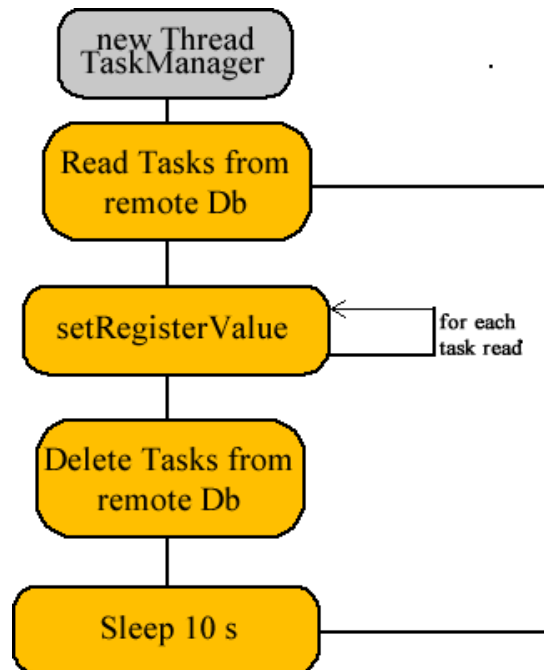


Figura 16 –Thread Task Manager

- `threadSocketManager()`: thread che ha il compito di controllare se ci sono richieste di connessione. Non appena si presenta una richiesta, avvia un thread per soddisfare la richiesta;

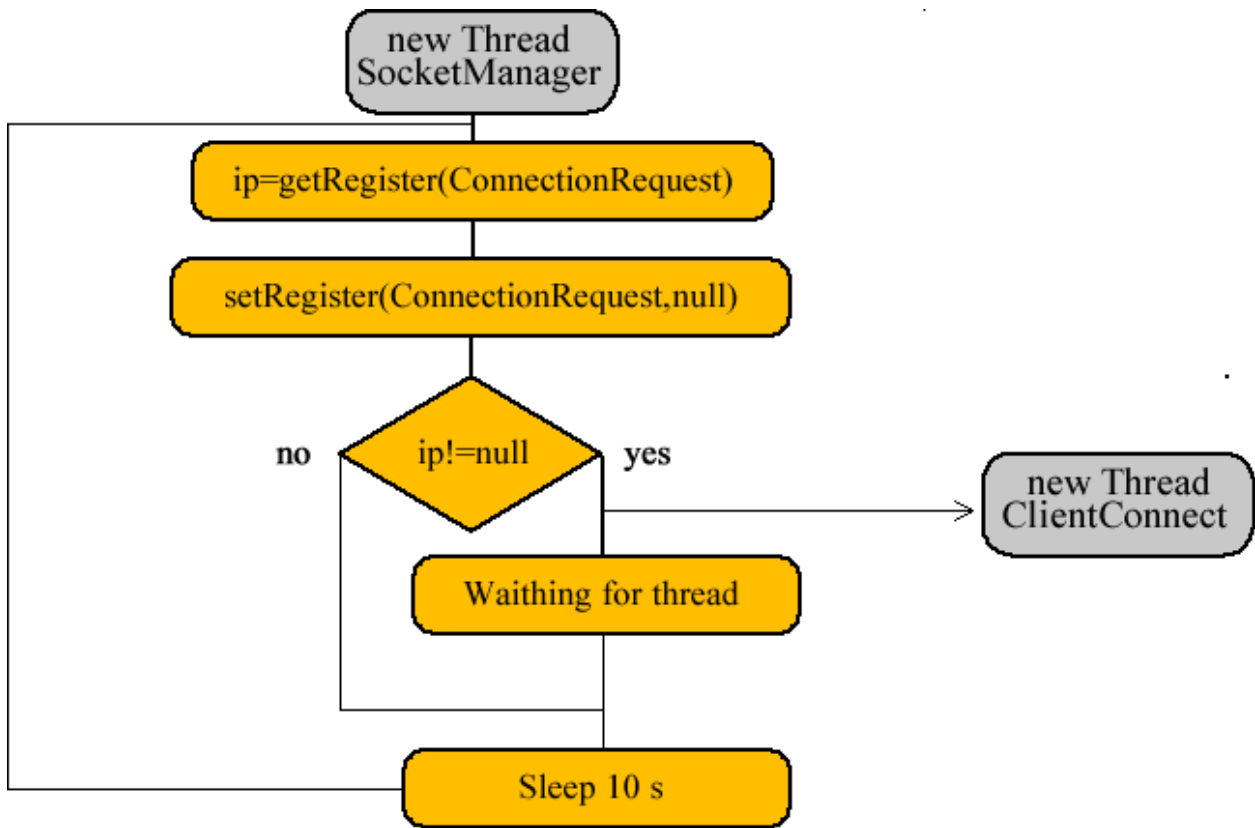


Figura 17 –Thread Socket Manager

- `threadClientConnect()`: thread che viene avviato dal `threadSocketManager()` ed ha il compito di creare una connessione TCP. Questo thread viene avviato ogni volta che si presenta una richiesta di connessione da un PC remoto: il PC remoto si comporta da Server, mentre questo thread si comporta da Client. Una volta aperta la socket TCP, il thread invierà tutte le frame che riceve dall'UART al Server TCP e quindi sul PC remoto sarà possibile visualizzare in un altro formato le frame che vengono ricevute dal Gateway;

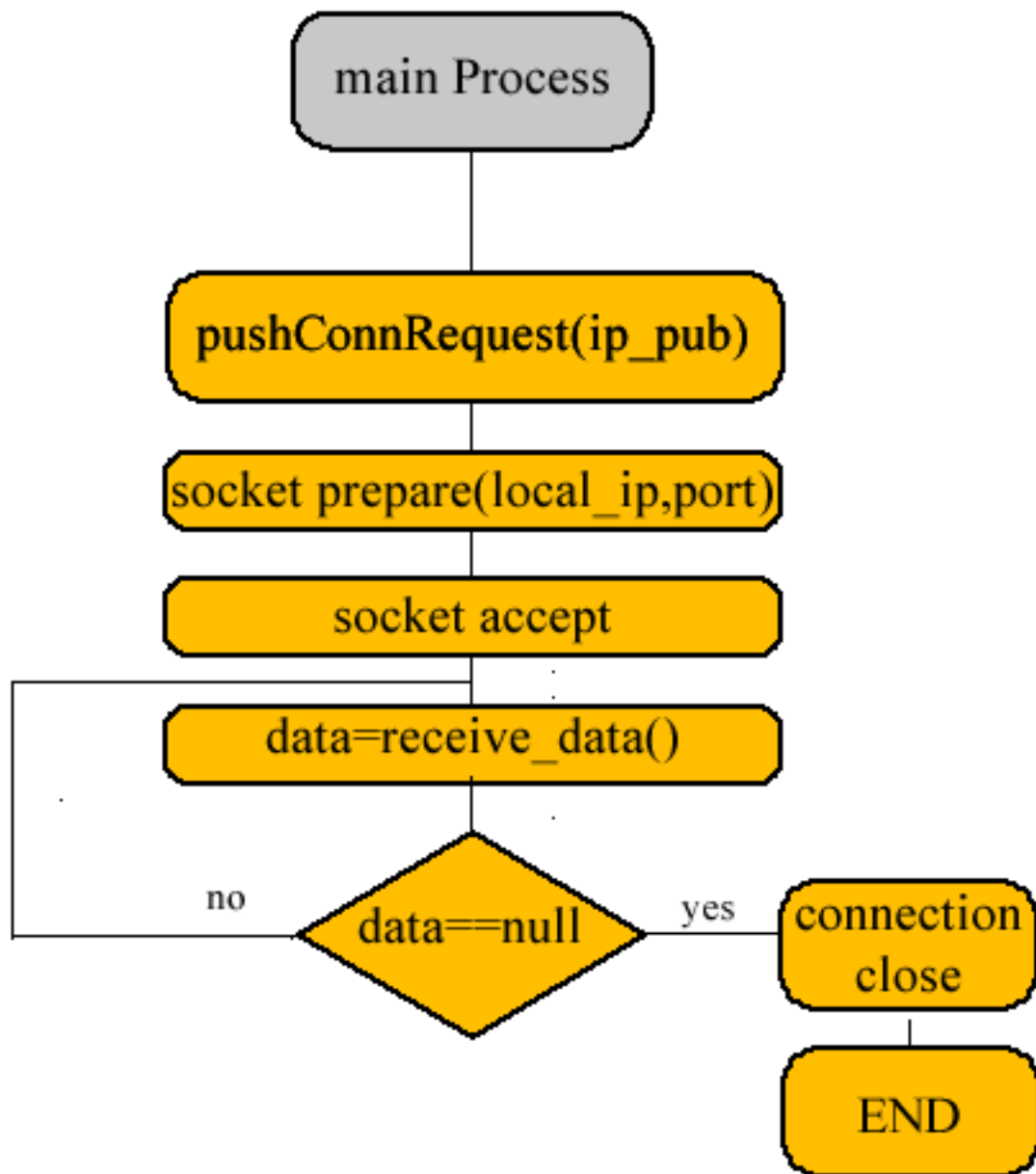


Figura 18 –Thread Client Connect

Sostanzialmente nella rete si hanno due tipi di monitoraggio: Indirect Monitoring e Direct Monitoring

### 5.1.3 Indirect Monitoring

In questa modalità non si ha alcuna interazione con la PulseNet. I dati, che sono stati caricati sul database dal Gateway, possono essere visualizzati mediante una web app in grado di leggere i dati inseriti nel database.

## Monitoring - Indirect Mode

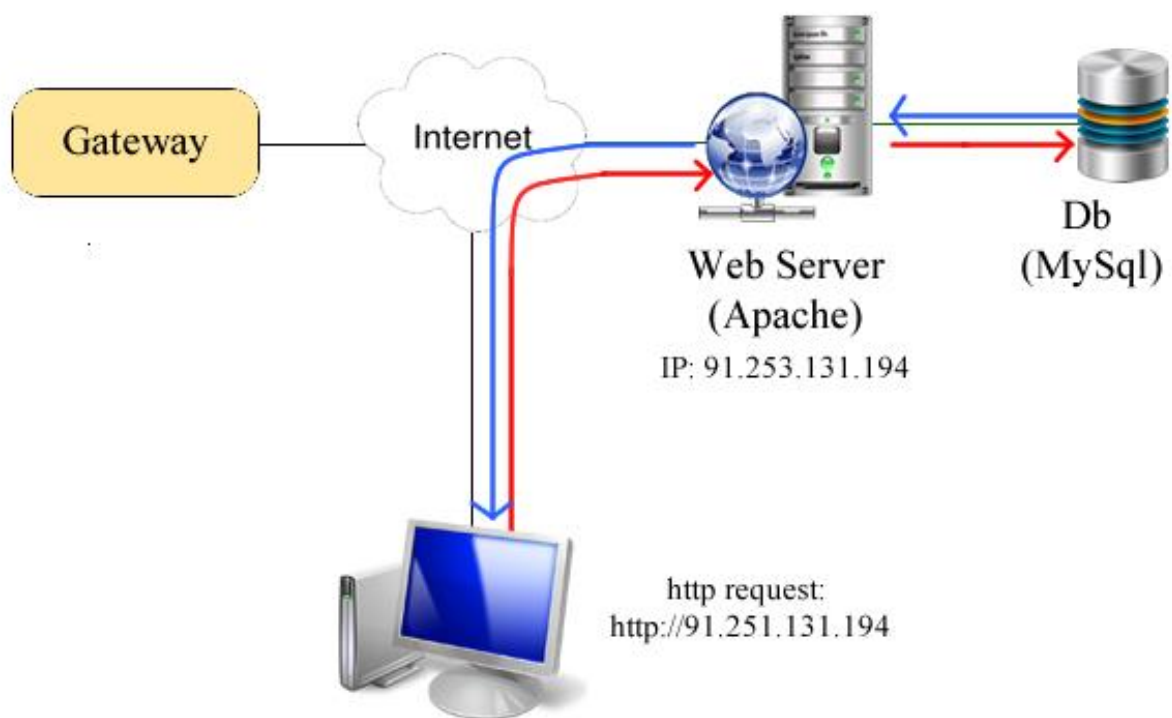


Figura 19 –Indirect Monitoring

#### 5.1.4 Direct Monitoring

In questa modalità i dati vengono inviati dal gateway al client, in modo diretto, senza usufruire di un database condiviso.

Il client avvia il software per la connessione al gateway, tale software inserisce nel database un record nella tabella dei task, in cui specifica come nome del registro: “ConnectionRequet” e come valore il suo indirizzo pubblico.

Il server, come mostrato in Figura 20, legge periodicamente ciascun record inserito nella tabella dei task e per ciascuno di questi imposta nel registro corrispondente il valore specificato.

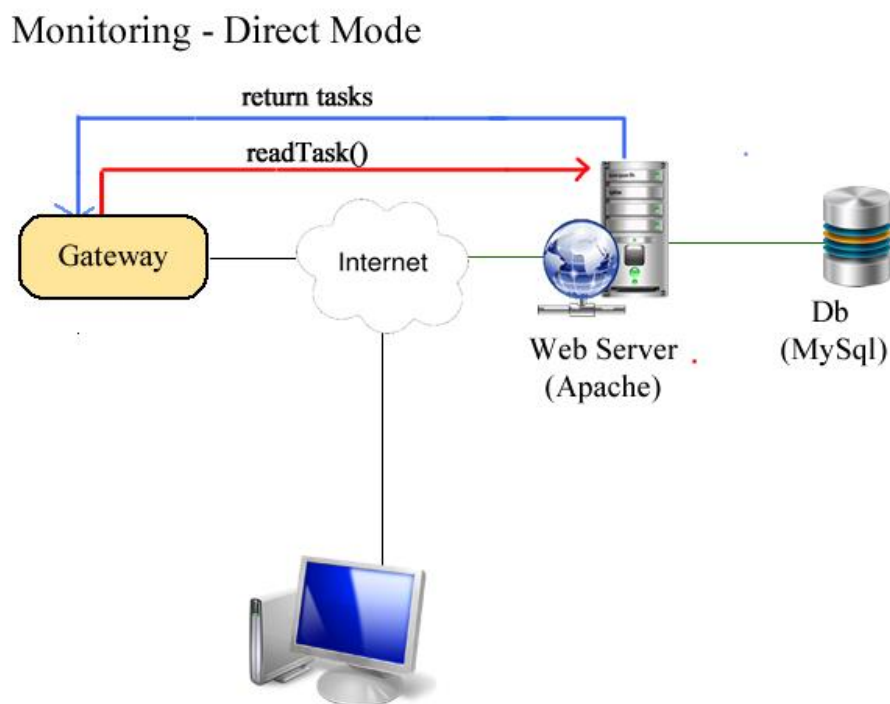


Figura 20 –Direct Monitoring (FASE A)

Il thread Socket Manager rileva una nuova richiesta di connessione poiché il registro ConnectionRequest contiene un indirizzo ip, successivamente avvia il thread Client Connect per soddisfare la richiesta e imposta il registro “ConnectionRequest” a un valore nullo.

## Monitoring - Direct Mode

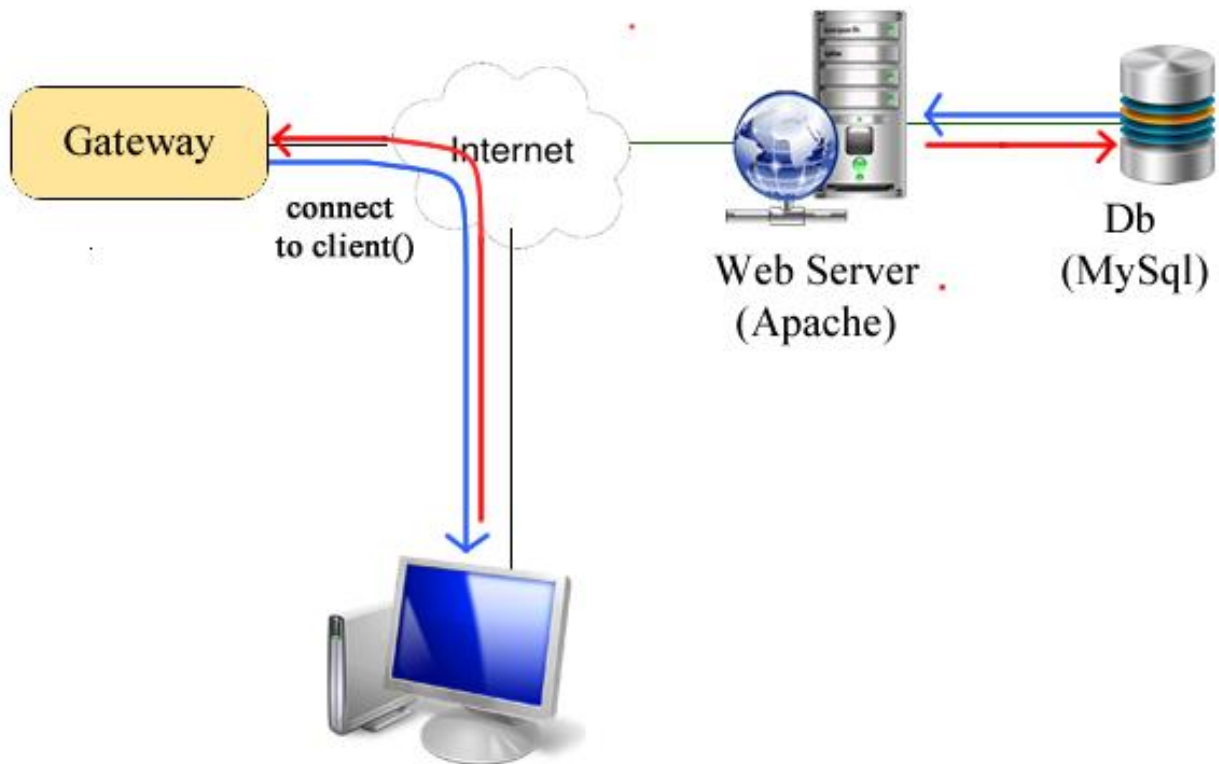


Figura 21 –Direct Monitoring (FASE B)



In questo modo viene avviata la comunicazione TCP tra la PulseNet ed un generico PC remoto. Da notare che mentre le frame vengono inviate all'host, esse vengono pure memorizzate sul database remoto.

## 5.2 Web App

La Web App permette ad un generico utente di gestire e configurare la rete. Inoltre la Web App illustra i dati acquisiti in tempo reale in modo tale da permettere un facile monitoraggio della rete: un qualsiasi utente provvisto di connessione Internet può accedere sia da smartphone che da PC al “Control Panel” della rete.

Per accedere al “Control Panel” basta aprire un browser e digitare l'indirizzo dell' host in cui è stata inserita la Web App, ad esempio:

<http://gatewayrai.altervista.org/>

Il “Control Panel” si presenterà in questo modo:

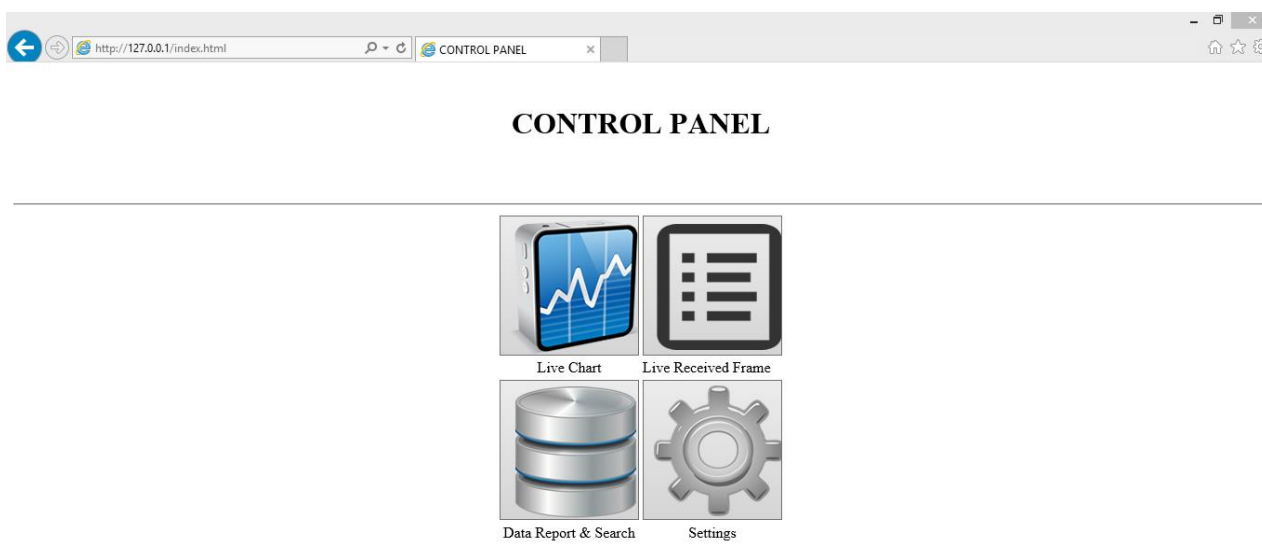


Figura 22 –Control Panel

Il “Control Panel” offre all’utente quattro possibili operazioni:



Live Chart: Illustra un grafico in tempo reale che mostra i valori dei dati ricevuti;



Live Received Frame: Illustra le ultime 100 frame ricevute in tempo reale;



Data Report & Search: Illustra tutte le frame ricevute e permette di applicare filtri di ricerca sulle frame;



Settings: Permette di configurare la rete;

### 5.2.1 Live Chart

Accedendo al Live Chart, l’utente potrà visualizzare in tempo reale il grafico dei valori rilevati da un sensore collegato a un nodo trasmettitore.

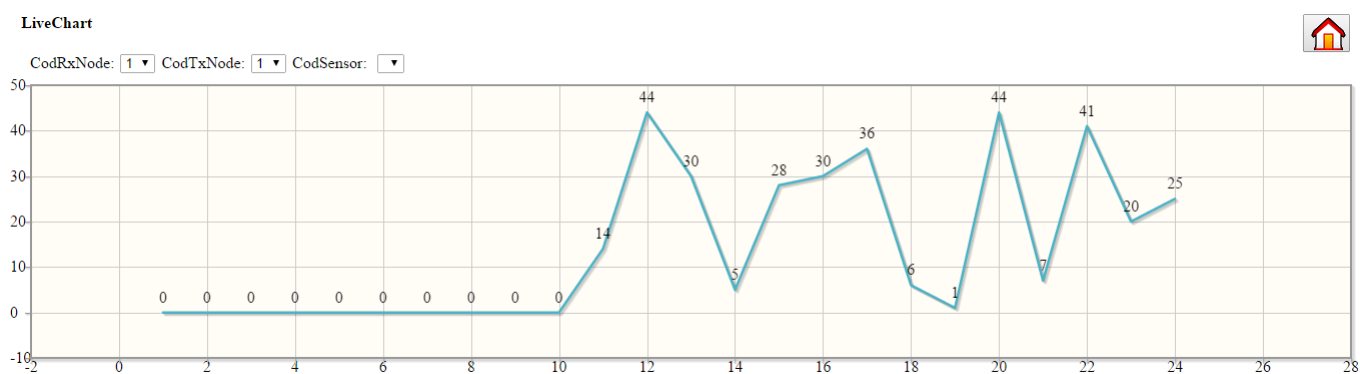


Figura 23 –Live Chart

Com'è possibile osservare in Figura 24, sono presenti tre controlli di tipo drop-down list, che permettono di selezionare il nodo ricevitore, il nodo trasmettitore e quindi il sensore di cui si vogliono graficare i valori rilevati.

CodRxNode: 1 CodTxNode: 1 CodSensor:

Figura 24 –Live Chart Particular



5.2.3 Data Report & Search

Questa pagina illustra tutti i dati ricevuti e presenti sul database remoto:

Record

Search:

TIMESTAMP

contains:

search

TIMESTAMP	TYPE	ID NODE	BC	APP FRAME TYPE	DATA	TXP	NRT	CODE_RX_DEVICE	STATE
2014-09-11 16:36:30	SIGNAL FRAME	1	255	REGISTRATION FRAME	0	4	0	1	CRC OK
2014-09-11 16:36:30	SIGNAL FRAME	1	255	REGISTRATION FRAME	0	4	0	1	CRC OK
2014-09-11 16:36:30	SIGNAL FRAME	1	255	REGISTRATION FRAME	0	4	0	1	CRC OK
2014-09-11 16:36:30	SIGNAL FRAME	1	255	REGISTRATION FRAME	0	4	0	1	CRC OK
2014-09-11 16:36:30	SIGNAL FRAME	1	255	REGISTRATION FRAME	0	4	0	1	CRC OK
2014-09-11 16:36:30	SIGNAL FRAME	1	255	REGISTRATION FRAME	0	4	0	1	CRC OK
2014-09-11 16:36:30	SIGNAL FRAME	1	255	REGISTRATION FRAME	0	4	0	1	CRC OK
2014-09-11 16:36:30	SIGNAL FRAME	1	255	REGISTRATION FRAME	0	4	0	1	CRC OK
2014-09-11 16:36:30	SIGNAL FRAME	1	255	REGISTRATION FRAME	0	4	0	1	CRC OK

Page: 1

page: 1/82

Figura 27 –Data Report

page: 1/82

Figura 28 – Numero di pagine

Inoltre, su questa pagina, è possibile utilizzare la barra di ricerca per visualizzare solo le frame che rispettano il criterio di ricerca selezionato o visualizzare, pagina per pagina, tutte le frame ricevute.

Record

Search:

TIMESTAMP

contains:

search

TIMESTAMP	TYPE	ID NODE
2014-09-11 16:36:30	SIGNAL FRAME	1
2014-09-11 16:36:30	SIGNAL FRAME	1
2014-09-11 16:36:30	SIGNAL FRAME	1
2014-09-11 16:36:30	SIGNAL FRAME	1

Figura 29 –Search in Data Report

## 5.2.4 Settings

Questa pagina permette all'operatore di configurare i registri permanenti, presenti all'interno del gateway:



The screenshot shows a web browser window with the address bar displaying `http://gatewayrai.altervista.org/edit.html`. The page title is "Dashboard". The main heading is "Configurazione dei Registri". Below this, there is a label "CodGateway:" followed by a text input field. A horizontal line separates this from a table. The table has two columns: "Register" and "Value". It contains two rows: "PeriodicTx" and "ListenPort", each with a corresponding text input field. At the bottom of the table is a "Save" button.

Register	Value
PeriodicTx	<input type="text"/>
ListenPort	<input type="text"/>

Figura 30 –Search in Data Report

Per configurare un registro all'interno del gateway bisogna inserire il codice del gateway di cui si intende modificare il valore dei registri e, in seguito, inserire i valori dei registri che si intendono modificare.

## 5.3 Gestione del database remoto

Per interagire con il database si è fatto uso del software phpMyAdmin.

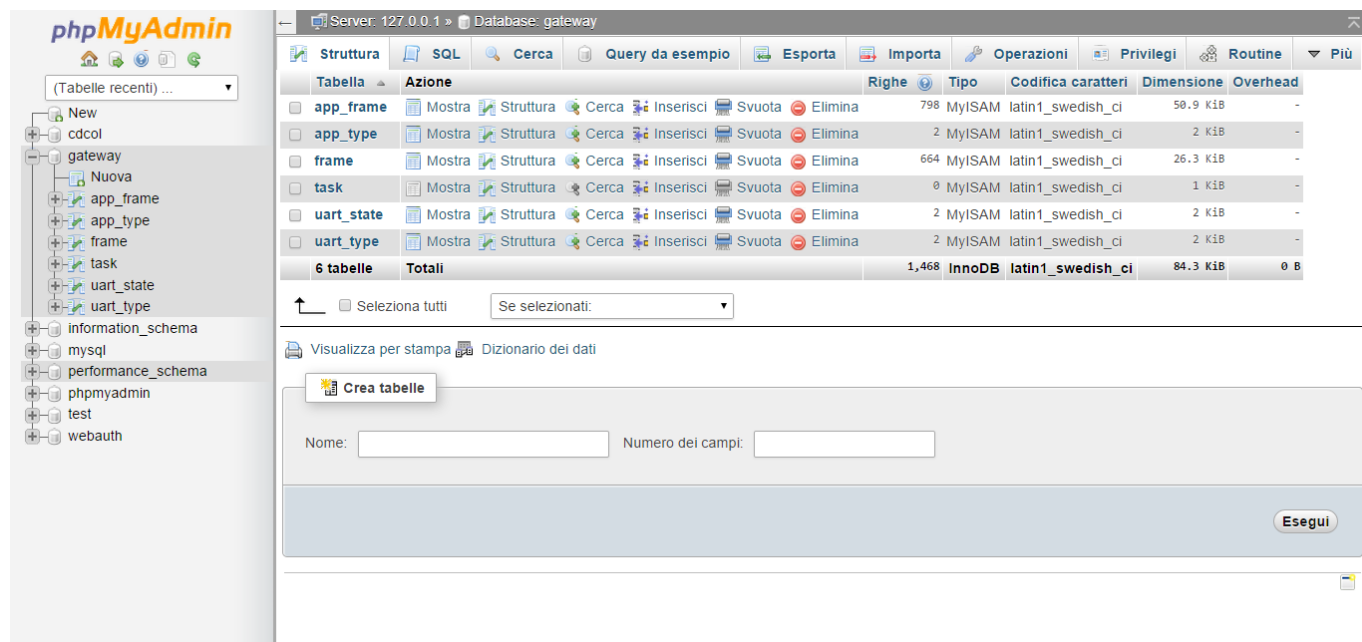


Figura 31 –phpMyAdmin

Per importare il database della PulseNet basta cliccare su “Importa”, visibile in Figura 31, e selezionare il file dump, che viene fornito in allegato a questa documentazione.

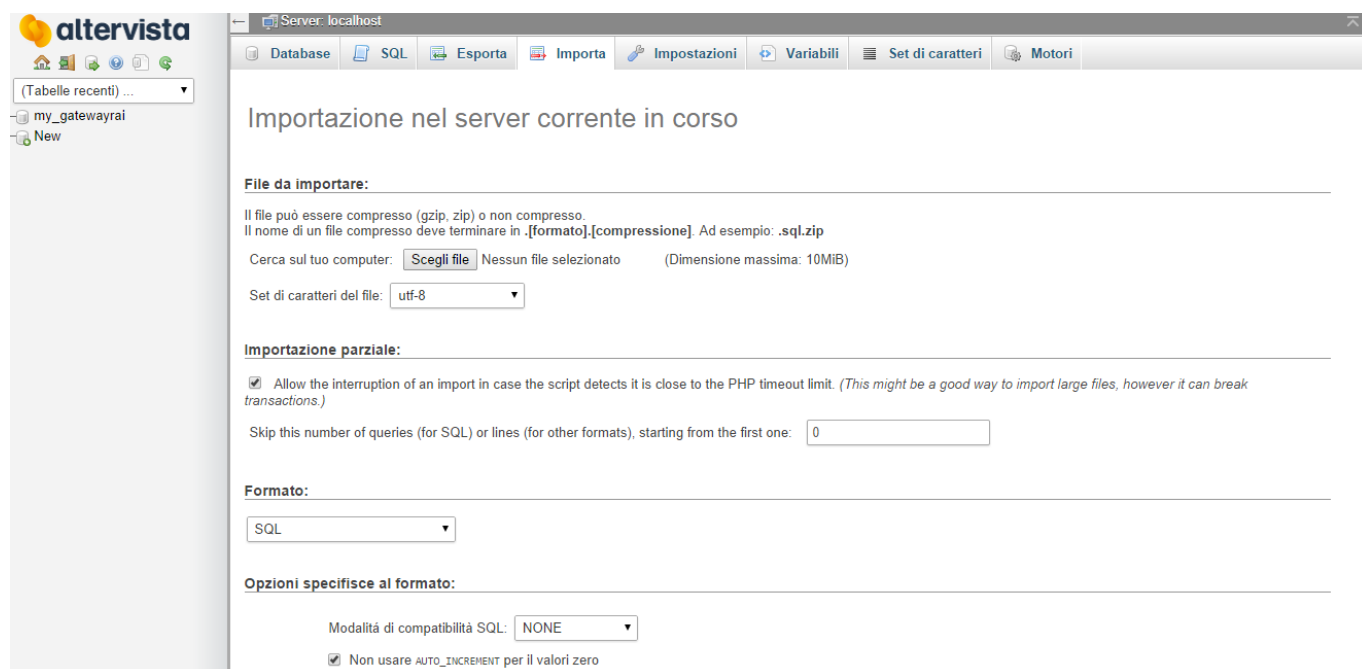


Figura 32 –phpMyAdmin dump

Analizziamo adesso le tabelle contenute nel database.

5.3.1 Tabella frame

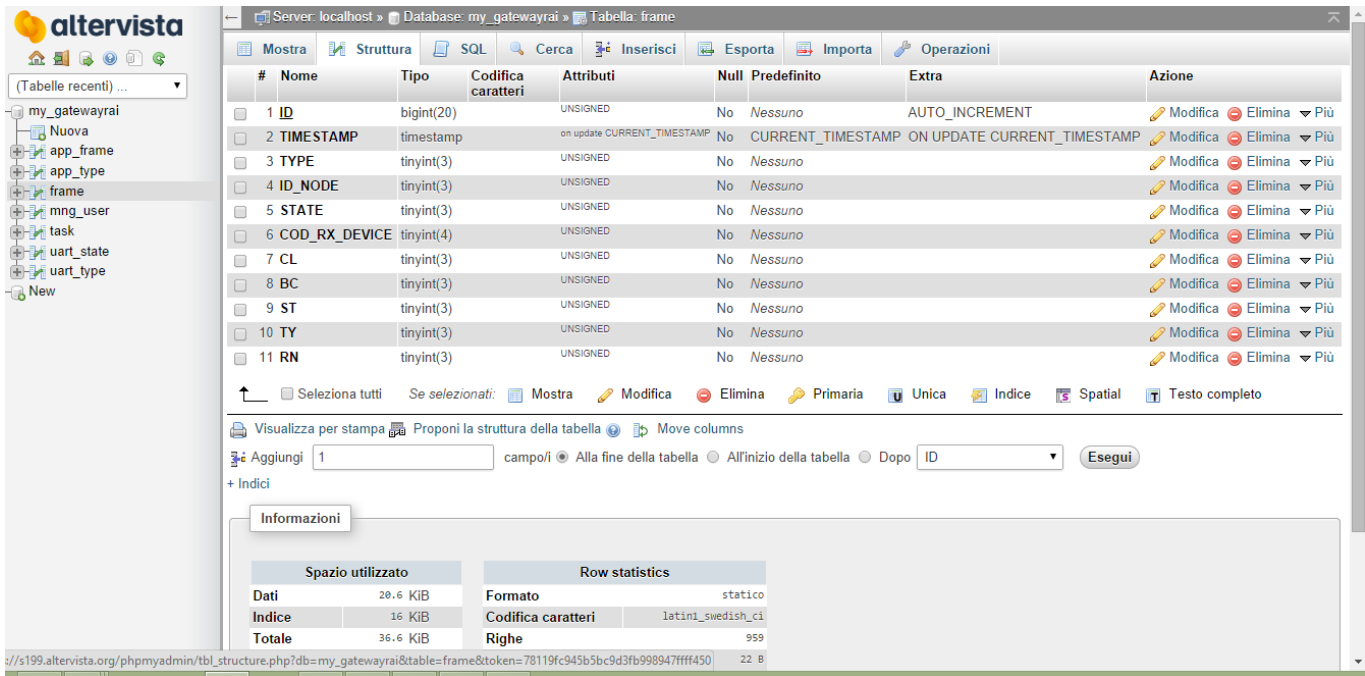


Figura 33 –Tabella Frame

Ogni record della tabella racchiude i campi della UART Frame, della DataLink Frame e dell’header del pacchetto di livello applicativo.

5.3.2 Tabella app\_frame

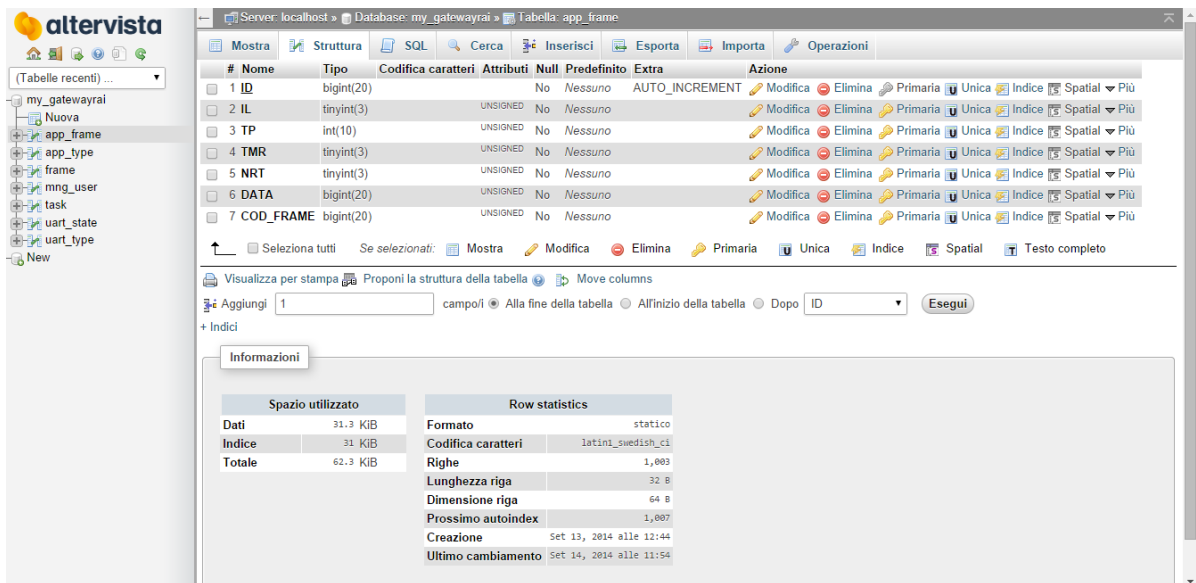


Figura 34 –Tabella app\_frame



Questa tabella racchiude i campi presenti nell’application frame, esclusi i campi dell’header.

5.3.3 Tabella task

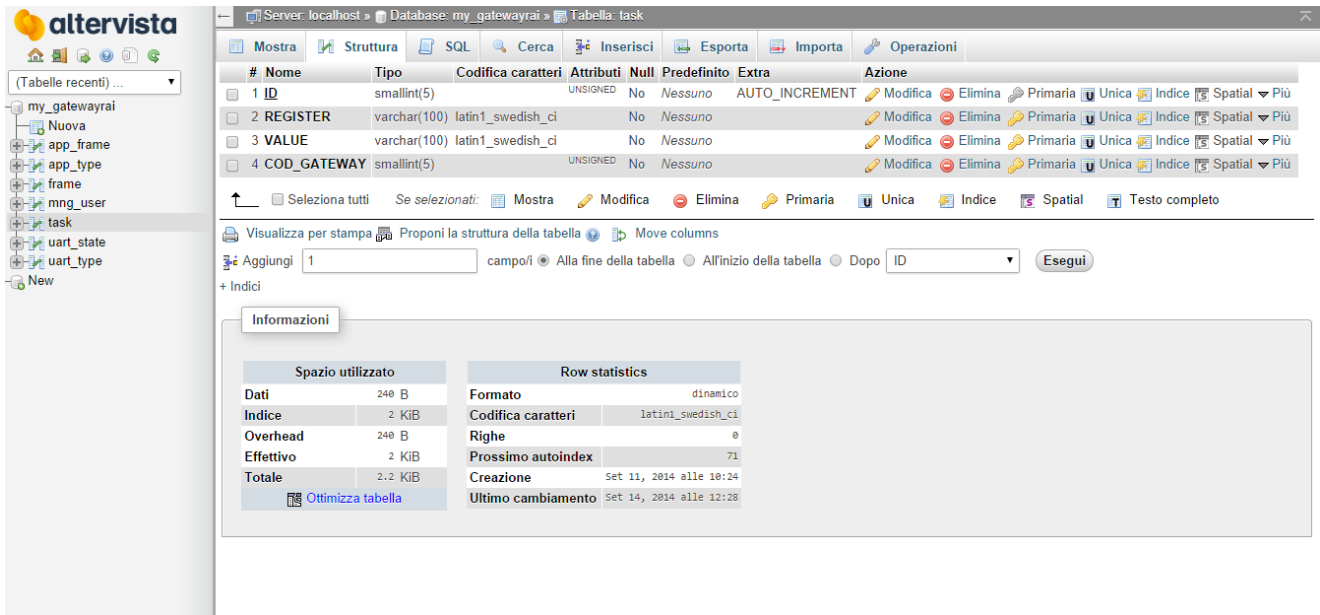


Figura 35 –Tabella task

Ogni record rappresenta un task , che può essere inserito mediante il “Control Panel”.

Un task specifica il codice del gateway a cui è indirizzato il task, il nome e il valore del registro che si intende modificare.

### 5.3.4 Tabella app\_type

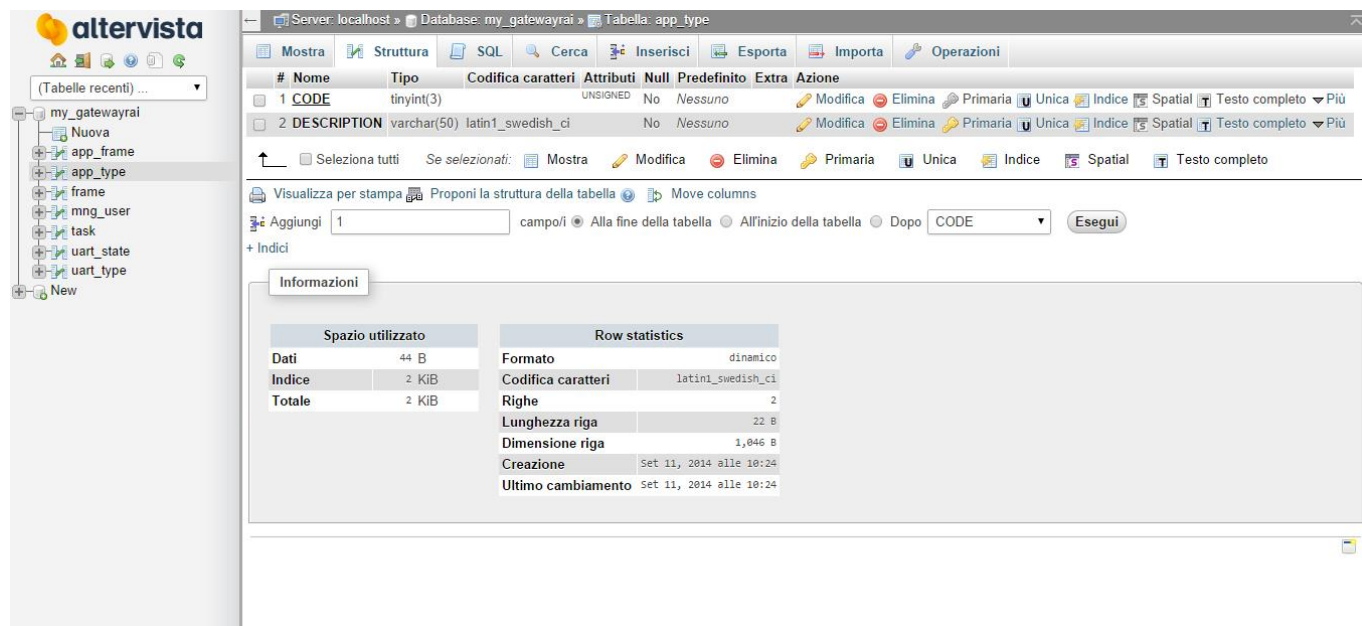


Figura 36 –Tabella app\_type

Questa tabella è stata creata per rappresentare la tipologia di “application frame”. Al momento, le “application frame” possono essere di due tipologie: “Registration Frame” e “Data Frame”. Se in futuro si vorrà inserire un’ ulteriore tipologia di frame basterà inserire un nuovo record in questa tabella. Questa scelta progettuale garantisce quindi un elevato grado di estendibilità e permette, di agevolare l’inserimento di nuove funzionalità nella PulseNet.

### 5.3.5 Tabella uart\_state

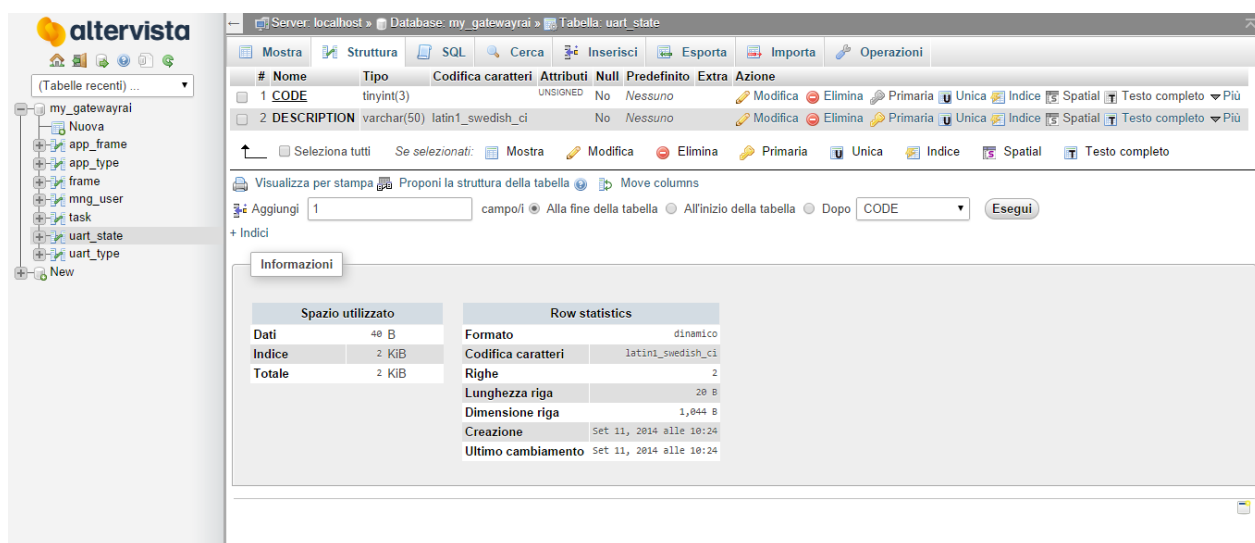


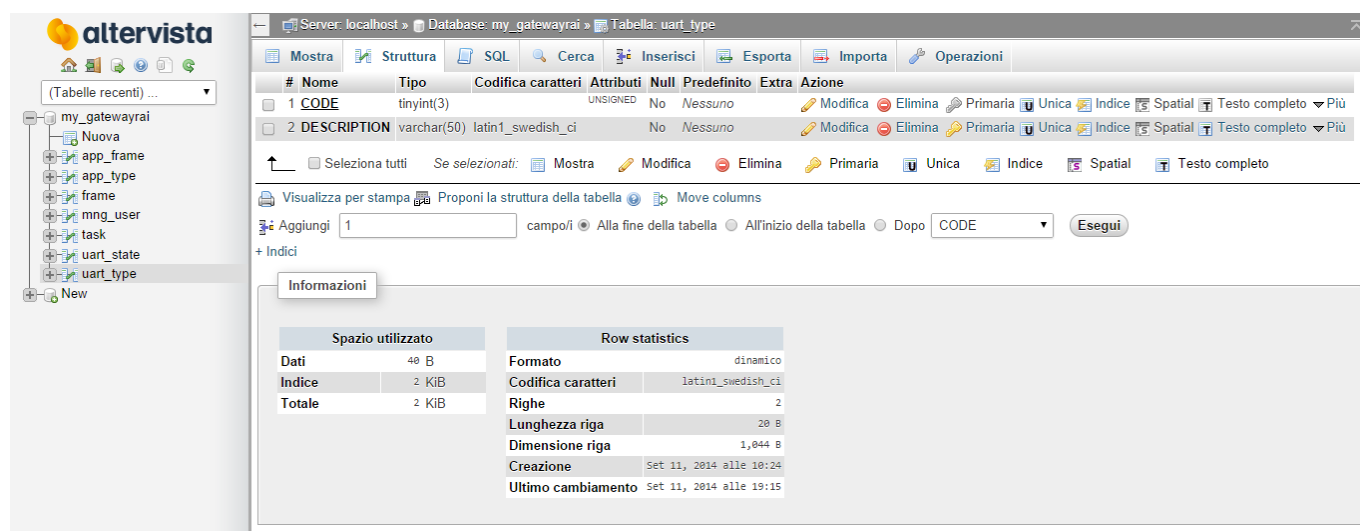
Figura 37 –Tabella uart\_state

Questa tabella contiene lo stato che le UART Frame possono assumere. Al momento si hanno cinque stati: CRC OK, CRC ERROR, DEAD NODE, VALID NODE e NODE NOT REGISTERED

CRC OK viene assegnato a tutte le frame che vengono ricevute correttamente; CRC ERROR viene assegnato alle frame che sono state ricevute in modo errato; DEAD NODE ad una frame che segnala un nodo morto; VALID NODE ad una frame che segnala che un nodo è ritornato a trasmettere e NODE NOT REGISTERED a una frame che segnala un nodo che sta trasmettendo dei pacchetti dati senza essere registrato.

Così come per la tabella precedente, la scelta progettuale di questa tabella garantisce un elevato grado di estendibilità.

### 5.3.6 Tabella uart\_type



#	Nome	Tipo	Codifica caratteri	Attributi	Null	Predefinito	Extra	Azione
1	CODE	tinyint(3)		UNSIGNED	No	Nessuno		Modifica Elimina Primaria Unica Indice Spatial Testo completo Più
2	DESCRIPTION	varchar(50)	latin1_swedish_ci		No	Nessuno		Modifica Elimina Primaria Unica Indice Spatial Testo completo Più

Spazio utilizzato		Row statistics	
Dati	48 B	Formato	dinamico
Indice	2 KiB	Codifica caratteri	latin1_swedish_ci
Totale	2 KiB	Righe	2
		Lunghezza riga	28 B
		Dimensione riga	1,044 B
		Creazione	Set 11, 2014 alle 10:24
		Ultimo cambiamento	Set 11, 2014 alle 19:15

Figura 38 –Tabella uart\_type

Questa tabella contiene la tipologia che le UART Frame possono assumere. Al momento si hanno solo due tipologie: ALARM FRAME e SIGNAL FRAME.

Le ALARM FRAME sono frame di allarme che rappresentano situazioni anomale: quando un nodo muore o ritorna attivo; un nodo che invia data frame senza che sia registrato nella rete oppure quando delle frame vengono ricevute in modo errato.

Le SIGNAL FRAME, invece, sono frame di segnalazione e contengono all'interno del loro payload le frame di livello DataLink e Application Layer.

Anche in questo caso, questa scelta progettuale garantisce un'elevata estendibilità.

## 6 SVILUPPI FUTURI

Un possibile sviluppo futuro potrebbe essere quello di integrare nel progetto una o più reti di sensori.

In Figura 39, viene illustrata la modifica che bisognerà effettuare per far sì che il gateway possa servire una o più reti. All'interno del main viene creato e avviato un processo per ogni rete che si intende servire. Le operazioni che deve effettuare lo specifico processo sono le stesse che effettua il main dell'implementazione attuale, con la differenza che istanzia la classe factory di competenza.

Così, se il processo `PulsenetManager` istanzia la classe `PulseNetFactory`, il processo `Network1Manager` istanzia la `Network1Factory`.

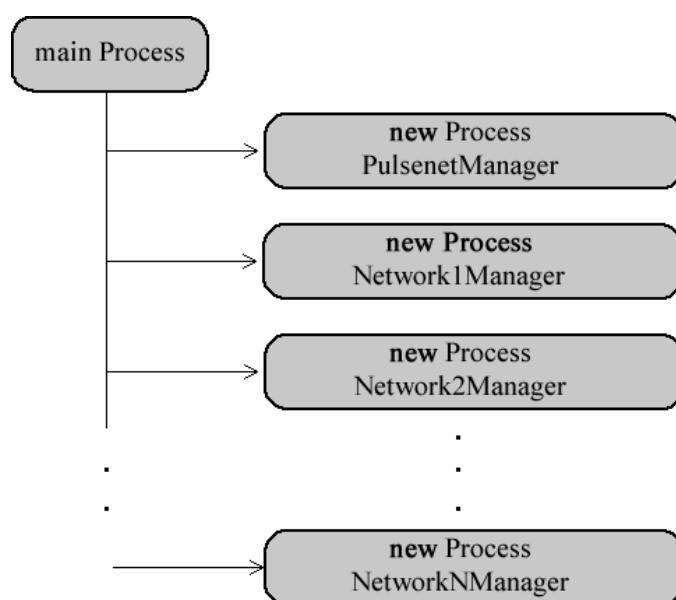


Figura 39 –Sviluppi futuri

Ulteriori sviluppi potrebbero essere:

- inserire altri grafici;
- dare la possibilità di esportare i dati in formato CSV;
- creare una pagina per il calcolo automatico delle statistiche;
- creare un meccanismo di invio degli sms, qualora un nodo della rete, particolarmente importante, si disconnettesse.



7 APPENDICE A: CONNESSIONE [STM32]

In questo paragrafo illustriamo il collegamento SPI1 tra Nrf24l01 e STM32 vl.

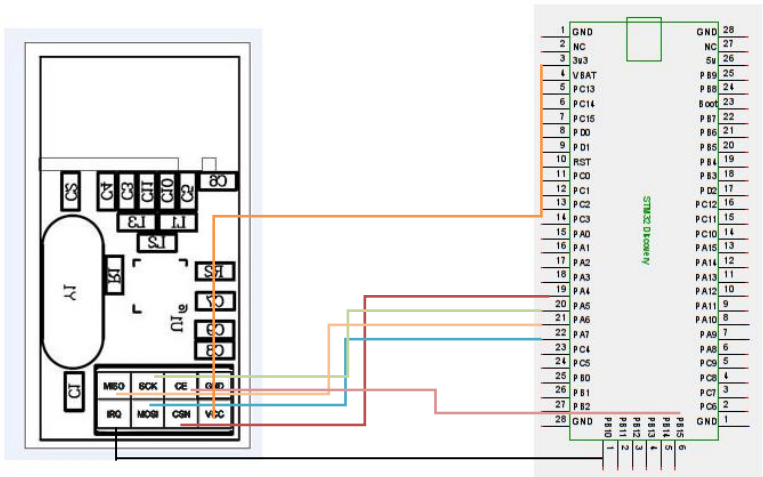


Figura 40 – Connessione STM32 vl con Nrf24l01

nRF24l01	STM32vl
MISO	PA6
SCK	PA5
CE	PB15
GND	GND
IRQ	PB10
MOSI	PA7
CSN	PA4
VCC	3V3

Tabella 1– Connessione STM32vl con Nrf24l01





## 8 APPENDICE B: INSTALLAZIONE E CONFIGURAZIONE GATEWAY RASPBERRY PI

### 8.1.1 Introduzione

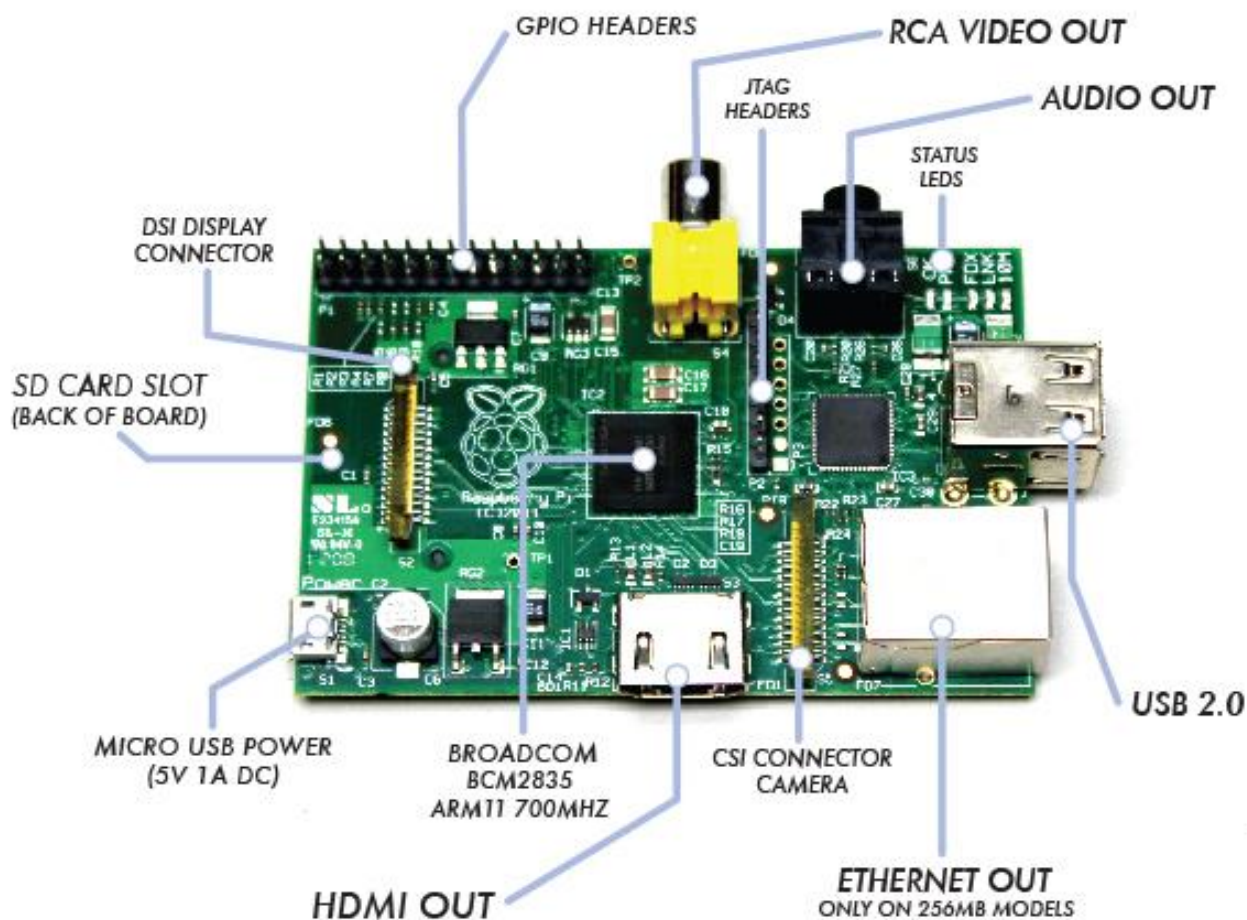


Figura 41 –Raspberry Pi

Raspberry Pi è un “single board computer” sviluppato dalla Raspberry Pi Foundation. La scheda è dotata di una scheda System On a Chip(Soc) Broadcom BCM2835, dotata di un processore ARM1176JSF-S a 700 MhZ, di una GPU VideoCore IV e di 512 MB di RAM. E’ dotata inoltre di una porta HDMI, 2 porte USB ed una porta Ethernet. La scheda è stata progettata per ospitare sistemi operativi basati su un kernel

Linux. Il sistema operativo è interamente caricato su una memoria SD, che quindi è necessaria nella fase di boot e come archivio di massa. [1]

Nel paragrafo successivo illustriamo i passi necessari per l'installazione del sistema operativo sulla scheda SD.

	Model A	Model B	Model B+
Prezzo di offerta:	USD 25 (GBP 16)	USD 35 (GBP 22)	
SoC:	Broadcom BCM2835 (CPU + GPU + DSP + SDRAM)		
CPU:	700 MHz ARM1176JZF-S core (famiglia ARM11)		
GPU:	Broadcom VideoCore IV, OpenGL ES 2.0, 1080p30 H.264 high-profile decode		
Memory (SDRAM):	256 MB (condivisa con la GPU)	256 o 512 MB (condivisa con la GPU)	
USB 2.0 ports:	1	2 (attraverso un hub USB integrato)	4
Output video:	Connettore RCA, HDMI		
Output audio:	3,5 mm jack, HDMI		
Memoria:	SD / MMC / SDIO card slot		MicroSD
Collegamenti di rete:	Nessuno	Ethernet 10/100 (RJ-45)	
Periferiche di basso livello:	2x13 header pins for GPIO, SPI, I²C, UART, +3,3 Volt, +5V		40Xgpio

<b>Real-time clock:</b>	No clock or battery		
<b>Corrente assorbita: (potenza)</b>	300 mA, (1,5 W)	700 mA, (3,5 W)	600 mA, (3,0 W)
<b>Alimentazione:</b>	5V via MicroUSB o GPIO header		
<b>Dimensioni:</b>	85,60 mm × 53,98 mm (3.370 inch × 2.125 inch)		
<b>Sistemi operativi supportati:</b>	Debian GNU/Linux, Fedora, Arch Linux, Gentoo e RISC OS (shared source)		
<b>Sistemi operativi non supportati:</b>	Windows		

Tabella 2– Caratteristiche tecniche del Raspberry Pi

#### 8.1.2 Installazione del sistema operativo

Come detto nel paragrafo precedente, è possibile installare diversi sistemi operativi sulla scheda. Per il progetto in questione si è scelto di installare Raspbian. Raspbian è una derivate del famoso sistema operative Debian. Raspbian è la soluzione suggerita dal team di sviluppo, proprio perché ottimizzata per l'hardware di Raspberry Pi.

Raspbian utilizza il desktop environment LXDE (leggero e funzionale), ed include tutta una serie di applicazioni preinstallate, tra cui Python, il browser web Midori, Scratch e molto altro.

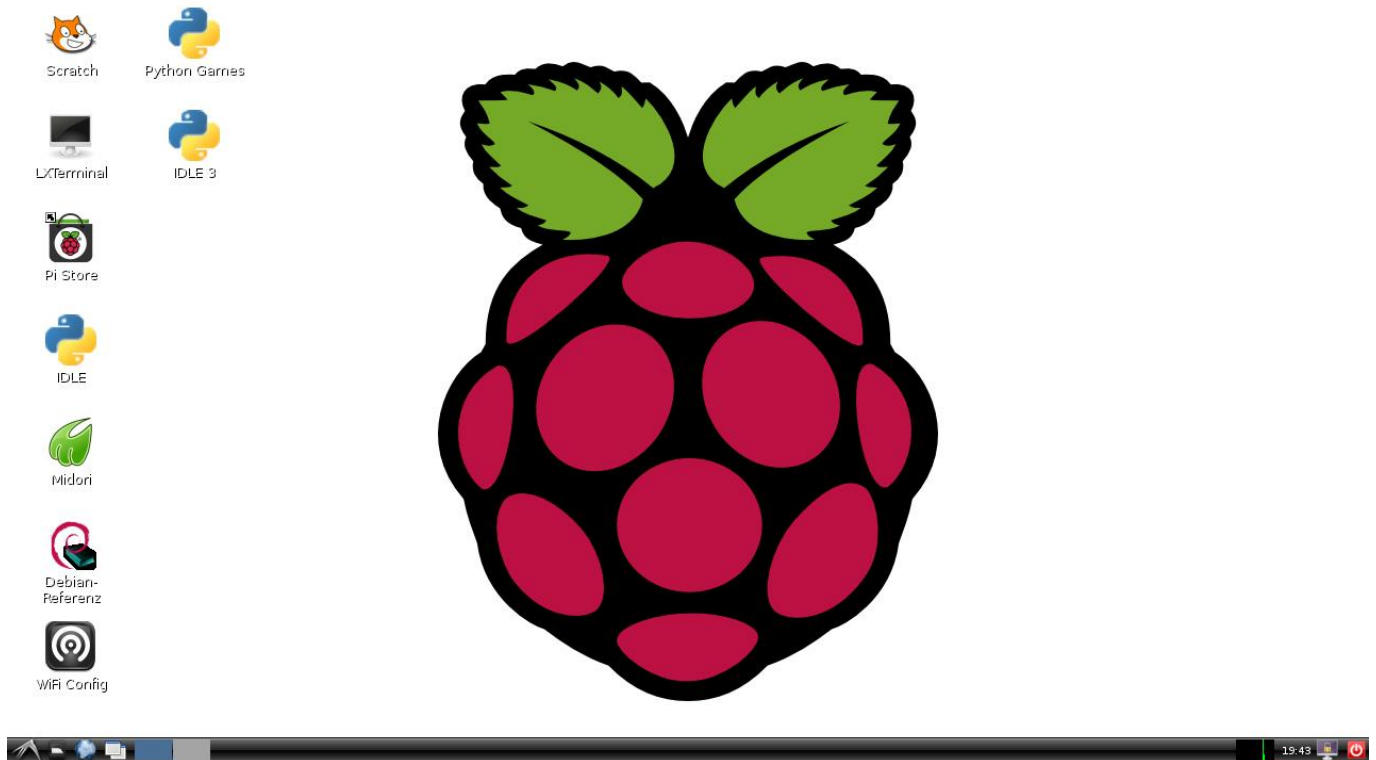


Figura 42 – Il desktop di Raspbian

Per l'installazione di Raspbian si è scelto di utilizzare una scheda SD da 16 GB classe 10.

I passi eseguiti per l'installazione del sistema operativo sono stati i seguenti:

- Scaricare la Raspbian dal sito ufficiale <http://www.raspberrypi.org/downloads>
- Inserire l'immagine scaricata sulla scheda SD. Su linux è possibile fare questo, utilizzando il comando "dd" con la seguenti sintassi:

```
dd if=/raspbian.img of=/dev/sdc
```

Nel caso in cui si utilizza Windows occorre utilizzare programmi come Win32DiskImager.

- Smontare la scheda SD dal PC, inserirla in Raspberry Pi ed attendere il completamento della installazione.

L'installazione, a questo punto, dovrebbe essere completata. Per poter utilizzare il sistema operativo appena installato si hanno due possibilità:

1. Collegare un cavo HDMI ad un monitor ed utilizzare Raspbian con interfaccia grafica;
2. Collegare un cavo ethernet ed accedere in SSH;

La connessione in SSH può essere realizzata solo se si conosce l'indirizzo IP del dispositivo. Per la prima accensione si può scoprire questo per mezzo di un "NetScanner", in questo modo è possibile sapere l'indirizzo IP che è stato assegnato dal DHCP Server.

Per rendere l'operazione di SSH più agevole si può impostare un IP statico.

### 8.1.3 Assegnazione di un IP Statico

Dopo aver eseguito l'accesso al sistema, occorre modificare il file `/etc/network/interfaces`.

La configurazione utilizzata nel nostro caso è stata la seguente:

```
auto lo

iface lo inet loopback

iface eth0 inet static
address 192.168.1.60
netmask 255.255.255.0
network 192.168.1.0
gateway 192.168.1.1

allow-hotplug wlan0
iface wlan0 inet manual
wpa-roam /etc/wpa_supplicant/wpa_supplicant.conf
iface default inet dhcp
```

In questo modo si è assegnato IP statico 192.168.1.60 e Gateway 192.168.1.1

#### 8.1.4 RaspBerry Pi UART

La porta seriale del RaspBerry Pi è del tipo UART con logic level di **3,3V**. Per collegare dispositivi con logic level diversi (5V,12V) all'UART del RaspBerry, bisogna limitare/adequare questi livelli.

L'UART si trova ai pin GPIO14(**TXD**) e GPIO15(**RXD**) del raspBerry. Per interfacciarsi con la porta, il sistema offre il file `/dev/ttyAMA0`.



Figura43 – Porta Seriale Pinout

Per utilizzare UART bisogna rimuovere la configurazione iniziale che permette di effettuare operazioni di debug.

Per fare questo, bisogna modificare il file `/boot/cmdline.txt` e `/etc/inittab`.

Prima di fare ciò effettuiamo una copia di backup:

```
cp /boot/cmdline.txt /boot/cmdline.bak
cp /etc/inittab /etc/inittab.bak
```

Rimuoviamo i parametri di configurazione `"console=ttyAMA0,115200"` e `"kgdboc=ttyAMA0,115200"` da `/boot/cmdline.txt` utilizzando nano editor.

```
nano /boot/cmdline.txt
```

```
nano /etc/inittab
```

Commentiamo l'ultima riga del file "/etc/inittab". Inserendo '#' prima di "T0:23:respawn:/sbin/getty -L ttyAMA0 115200 vt100."

Adesso RXD (GPIO15) e TXD (GPIO14) sono configurati per l'utilizzo tramite UART.

Per la comunicazione con l'UART si è scelto di utilizzare un modulo Python chiamato pyserial.

Installiamo pyserial scaricando l'ultima versione da <http://pyserial.sourceforge.net/index.html>

```
wget https://pypi.python.org/packages/source/p/pyserial/pyserial-2.7.tar.gz
tar zxvf pyserial-2.7.tar.gz
cd pyserial-2.7
python setup.py install
```

Riportiamo adesso uno script che mostra come è possibile leggere i dati ricevuti tramite UART, utilizzando il modulo pyserial.

```
#!/usr/bin/python
import serial
import string

ser = serial.Serial('/dev/ttyAMA0')

def readLine(ser):
    str = ""
    while(1):
        ch = ser.read(1)
        if (ch == "\n" or ch == "\r" or ch == ""):
            break
        str += ch
    return str

while True:
    try:
        line = readLine(ser)
        print line
    except KeyboardInterrupt:
        ser.close()
        print "Goodbye!!!"
        raise
```

Figura 44 – Listato Read2Serial.p

### 8.1.5 Configurazione e connessione 3G

L'accesso ad Internet avviene per mezzo di un modem 3G, nello specifico si tratta del modem HUAWEI E220.



Figura 45 – Huawei E220

Affinchè il modem venga riconosciuto opportunamente dal RaspBerry Pi è necessario lanciare i seguenti comandi:

```
sudo apt-get install ppp usb-modeswitch usbutils
```

Per verificare se effettivamente la chiavetta è stata riconosciuta, bisogna lanciare il comando:

```
lsusb
```

Se l'output è il seguente, allora il modem è stato riconosciuto:

```
pi@raspberrypi ~ $ lsusb
Bus 001 Device 002: ID 0424:9512 Standard Microsystems Corp.
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 001 Device 003: ID 0424:ec00 Standard Microsystems Corp.
Bus 001 Device 007: ID 12d1:1003 Huawei Technologies Co., Ltd. E220
HSDPA Modem / E230/E270/E870 HSDPA/HSUPA Modem
```



### 8.1.6 Sakis3G

Sakis3G è un'utility che gestisce la connessione internet tramite dispositivi USB UMTS.

L'installazione avviene nei seguenti passi:

1. Scaricare al seguente [indirizzo](#) la versione di **Sakis3G** desiderata e salvarla nella propria cartella **Home**.
2. Scompattare il file scaricato digitando in una finestra di terminale il seguente comando:

```
gunzip sakis3g.gz
```

3. Dare quindi i permessi di esecuzione al file digitando il seguente comando:

```
sudo chmod +x sakis3g
```

4. Avviare **Sakis3G** digitando il seguente comando:

```
./sakis3g
```

L'ultimo comando avvierà sakis3G in modalità "interactive", se si vuole utilizzare sakis3g in modalità non interactive bisogna lanciare il comando:

```
sudo /usr/bin/sakis3g connect APN="ibox.tim.it"
```

Per gli APN fare riferimento alla seguente tabella:

Gestore telefonico	APN per contratto ricaricabile	APN per contratto in abbonamento
<b>3</b>	tre.it	datacard.tre.it
<b>CoopVoce</b>	web.coopvoce.it	web.coopvoce.it
<b>Fastweb</b>	apn.fastweb.it	datacard.fastweb.it
<b>PosteMobile</b>	wap.postemobile.it	internet.postemobile.it
<b>TIM</b>	ibox.tim.it	wap.tim.it
<b>Vodafone</b>	mobile.vodafone.it	web.omnitel.it
<b>Wind</b>	internet.wind	internet.wind.biz

### Tabella 3– Tabella APN

Infine per disconnettersi da Internet:

```
sudo /usr/bin/sakis3g disconnect APN="ibox.tim.it"
```

## **9 BIBLIOGRAFIA**

[1] [http://it.wikipedia.org/wiki/Raspberry\\_Pi](http://it.wikipedia.org/wiki/Raspberry_Pi)

[2] [http://it.wikipedia.org/wiki/Wireless\\_sensor\\_network](http://it.wikipedia.org/wiki/Wireless_sensor_network)

[3] <http://www.st.com/web/en/catalog/tools/FM116/SC959/SS1532/PF250863?sc=stm32-discovery>

[4] Documentazione PulseNet – Cristina Lombardo