INTERNAL REPORT

Sistema di supervisione e prototipo per il CED di SRT

Andrea Saba, Pierluigi Ortu, Franco Buffa, Antonietta Fara

Report N. 64

released: 20/03/2017

Revisore: Francesco Gaudiomonte

OAC Osservatorio Astronomico di Cagliari

Sommario

Introduzione

Il CED (Centro Elaborazione Dati) del Sardinia Radio Telescope è allestito in una camera schermata climatizzata la cui continuità elettrica di backup è garantita da gruppi di continuità UPS. E inoltre presente nel sito un gruppo elettrogeno che entra in funzione al verificarsi di un eventuale blackout elettrico.

Per garantire il funzionamento ottimale degli apparati presenti al suo interno la camera è dotata di due split che mantengono la temperatura entro un range prestabilito. Nella camera schermata sono collocati, nei limiti dettati dalle massime distanze consentite, tutti gli apparati attivi core che gestiscono la connettività di campus, compresi quelli a sevizio delle linee telefoniche VoIP[1].

La schermatura della camera impedisce l'emissione verso l'esterno delle spurie a radiofrequenza generate dagli apparati attivi installati al suo interno attenuandoli mediamente di 100dB fino a frequenze di 18 GHz.

Per garantire questo la camera schermata rimane chiusa durante tutti i periodi di attività osservativa del SRT. È necessario quindi prevedere un sistema di monitoraggio delle condizioni ambientali all'interno della camera in modo che sia possibile, tramite un sistema di controllo e/o un operatore di prendere le decisioni necessarie a ristabilire le situazioni ambientali ottimali o a procedere ad un eventuale spegnimento selettivo degli apparati. Il sistema deve intervenire anche nel caso di interruzione dell'alimentazione elettrica e malfunzionamento del gruppo elettrogeno nel momento in cui entrano in funzione le UPS. In questo caso infatti, considerando il carico elettrico degli apparati, le UPS possono garantire un periodo limitato di autonomia.

Il presente documento intende definire l'architettura del sistema di monitoraggio del CED di SRT e lo studio di fattibilità di un prototipo dimostratore. Il sistema gestisce i segnali provenienti da una rete di sensori collocati in posizioni strategiche trasformandoli, attraverso la mediazione di un codice *supervisore*, in stati di allarme. La mediazione tra segnali e allarmi avviene attraverso una serie di regole definite dall'*amministratore* del sistema che vengono interpretate dal *supervisore*.

L'aspetto più delicato riguarda la gestione di condizioni ambigue e contraddittorie. Ad esempio, il cattivo funzionamento di un sensore potrebbe indurre ad ignorarne il segnale, ma se il guasto del sensore derivasse da un principio di incendio, tale decisione potrebbe avere gravi conseguenze.

---

[1] In telecomunicazioni e informatica con Voice over IP (Voce tramite protocollo Internet), acronimo VoIP, si intende una tecnologia che rende possibile effettuare una conversazione telefonica sfruttando una connessione Internet o una qualsiasi altra rete dedicata a commutazione di pacchetto che utilizzi il protocollo IP senza connessione per il trasporto dati.

Verrà poi illustrato lo studio di fattibilità di un prototipo dimostratore basato su schede Arduino Ethernet[1] e Raspberry Pi 2 Model B[2].

1. Sistema di supervisione

1.1 Architettura del sistema

Il sistema è formato da:

- Un supervisore;
- Una rete di sensori;
- Alcuni processi sentinella distribuiti;
- Un frontend admin.

Il supervisore è il daemon che gestisce gli eventi. È collegato ai sensori e ai processi sentinella e contiene le regole implementate per gestire gli stati e gli eventi. Il supervisore è formato da:

- Un database;
- Un server che gestire il flusso di telemetria con i sensori, i processi sentinella e l'interfaccia admin;
- Un valutatore di espressioni (rule parser).

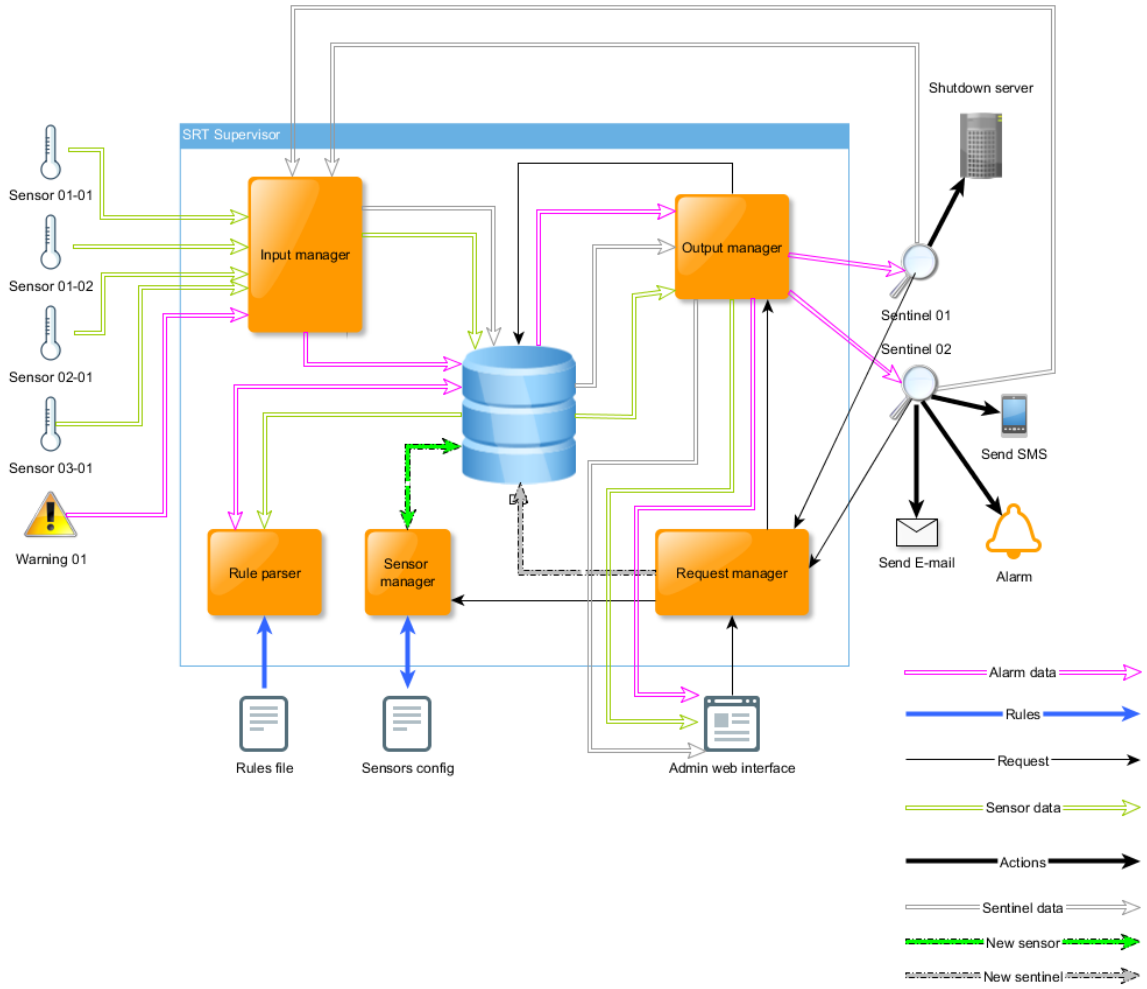

Figura 1: Architettura de sistema di supervisione

1.2 Sensori

I *sensori* sono dei componenti attivi che si occupano di generare periodicamente dei pacchetti di rete contenenti il valore o i valori istantanei letti e li inviano al supervisore. Prima dell'invio viene calcolato il checksum del dato inviato e questo viene aggiunto al pacchetto stesso.

1.3 Warnings o allarmi

Gli *warnings* possono essere di tre tipi: impliciti, espliciti o da espressioni. Quelli di tipo implicito sono generati dal supervisore stesso nel caso questo rilevi un malfunzionamento del sistema (per es. il guasto o l'inaccessibilità di un sensore). Quelli di tipo esplicito derivano da allarmi esterni che restituiscono stati (da un punto di vista logico questi possono essere considerati un tipo peculiare di sensore, che non fornisce valori misurati, ma stati). Gli ultimi derivano dalla valutazione delle espressioni e dipendono dai valori dei sensori o dagli stati di altri allarmi.
Per ogni allarme generato viene verificato a quali sentinelle deve essere inviato il messaggio relativo al nuovo stato.

1.4 Sentinelle

I *processi sentinella* sono deamon che risiedono nelle macchine che gestiscono gli spegnimenti, l'invio di messaggi, l'azionamento di allarmi acustici ecc. Le sentinelle rimangono in attesa della ricezione degli stati degli allarmi che gli sono stati attribuiti.
Le sentinelle interpretano gli allarmi generati dal supervisore e prima di provvedere all'azione comandata, dopo un opportuno timeout inviano un segnale di acknowledge receipt al supervisore. Sul database verrà salvato il timestamp.
Nel caso in cui la sentinella gestisca più allarmi questi verranno interpretati per ottenere una tabella delle azioni da compiere.
Naturalmente potrebbero coesistere azioni inconciliabili quali lo spegnimento immediato e quello con timeout, in questo caso prevarrebbe quello a più alta priorità (spegnimento immediato). L'invio delle email di alert deve precedere, se previsto, lo spegnimento delle macchine.
Ad eccezione dello spegnimento immediato, di norma gli spegnimenti sono regolati da timeout. Se una sentinella riceve più di un allarme con timeout, viene applicato il timeout più breve. Se durante il timeout che precede lo spegnimento, gli stati si modificano e gli allarmi rientrano, l'esecuzione degli spegnimenti è abortita con la sola eccezione degli spegnimenti con timeout incondizionati e quelli immediati.

Le sentinelle inviano al supervisore le richieste di autoregistrazione nel sistema. Nella richiesta è presente la lista degli allarmi per i quali rimarrà in ascolto e per i quali eseguirà delle azioni. Alla richiesta di registrazione verranno salvate nel DB le informazioni della sentinella e impostato il suo stato in “to be enabled”. Solo dopo un intervento da parte dell’amministratore lo stato verrà cambiato per permettere al supervisore di inviare i dati degli warning a quella sentinella.

1.5 Modulo sensor manager

Il modulo *sensor manager* prende in input il file di configurazione dei sensori nel quale sono definiti i loro parametri (mappatura):

- id;
- description;
- unit;
- lower_value;
- upper_value;
- out_of_range_value;
- location;
- status.

Questi parametri vengono letti all’avvio del supervisore che verifica nel database quali informazioni sono presenti.
Nel caso in cui il sensore non sia presente nel database questo viene inserito come nuovo sensore, mentre, se già presente, viene verificato se i suoi parametri, presenti nel database, debbano essere modificati. Uno dei valori normalmente modificati è quello dello stato che può essere: “enabled”, “disabled”, “untrusted”.
Questo file può essere modificato manualmente dall’amministratore o tramite l’interfaccia amministrativa. In questo secondo caso il file viene automaticamente rigenerato in base alle informazioni presenti nel DB. Tipicamente questo può avvenire quando si modifica lo stato di un sensore.

1.6 Modulo rule parser

Il modulo *rule parser* prende in input un file di configurazione nel quale sono definite le espressioni. Queste sono utilizzate per determinare lo stato degli allarmi e per farlo combinano gli stati dei sensori (e/o di altri allarmi) che vengono confrontati con valori soglia. Prima di valutare un’espressione, il sistema analizza i sensori e stabilisce se questi sono da

considerare affidabili o meno (trusted). Gli allarmi generati dalle espressioni sono salvati nel database.

1.7 Modulo input manager

Il protocollo scelto è UDP[2] che limita il traffico di rete al solo invio dei pacchetti con i valori dei sensori senza effettuare l'handshake[3]. Il modulo input manager del supervisor resta quindi in ascolto sulla porta 6555 UDP. All'arrivo di un pacchetto viene verificato il suo checksum. Nel caso in cui la verifica non vada a buon fine il pacchetto viene scartato e viene incrementato un contatore associato all'IP di provenienza. Dal blocco dati del pacchetto vengono quindi estratti:

- Il time stamp;
- Il codice del tipo di pacchetto (sensore, allarme, sentinella);
- L'ID del sensore/allarme/sentinella;
- Il numero di valori;
- Il codice del tipo di valori;
- La lista dei valori.

Nel caso in cui il numero di valori non corrisponda a quelli presenti nella lista, il pacchetto viene scartato.

Vengono inoltre effettuate ulteriori verifiche di coerenza dei dati.

Il formato del campo dati del pacchetto UDP inviati è il seguente (Tabella 1):

0	8	16	24 31
Time Stamp[4]			
Tipo pacchetto	Identificatore		Numero di valori[5]
Tipo di valori[6]	Valore 1		…
…	Valore n		Check sum[7] …
… Check sum (128 bit)			

Tabella 1: Campo dati del pacchetto UDP con i dati dei sensori

[2] In telecomunicazioni lo User Datagram Protocol (UDP) è uno dei principali protocolli di trasporto della suite di protocolli Internet. È un protocollo di livello di trasporto a pacchetto, usato di solito in combinazione con il protocollo di livello di rete IP.

[3] In informatica, è il processo attraverso il quale due calcolatori, tramite software o hardware, stabiliscono le regole comuni, ovvero la velocità, i protocolli di compressione, di criptazione, di controllo degli errori, ecc.

[4] Il campo indica 0 nel caso in cui non sia possibile avere un valore dal sistema. Nel caso di Arduino non essendoci un orologio di sistema questo campo sarà sempre a zero.

[5] Numero di valori che può restituire il sensore

[6] Il tipo di valore è un codice associato al tipo di dato che può essere (per esempio) temperatura, umidità, livello del fluido, presenza di fumi, etc.

[7] Il check sum viene calcolato come MD5 di tutti gli altri campi.

La grandezza massima del campo dati sarà quindi di 537 byte. Anche se per la maggior parte delle sue applicazioni la lunghezza sarà di 27 byte.
Se il time stamp è impostato a zero sarà compito del supervisore quello di sovrascrivere questo valore assegnandolo uguale a quello di sistema.
Il tipo di pacchetto identifica se è stato inviato da un sensore, un allarme o una sentinella. Queste informazioni, una volta acquisite e verificate vengono salvate nel DB.
Nel caso di un pacchetto inviato da un sensore in grado di restituire più tipi di dati, per esempio umidità e temperatura, verranno inviati pacchetti con ID distinti per ogni tipo di valore.
Il sensore, identificato dal suo ID, invia con una certa frequenza la sua lettura al server del supervisore.
L'identificativo associato ad ogni sensore è composto da due byte, il primo identifica l'accentratore di sensori e il secondo identifica il sensore collegato all'accentratore. Dove l'accentratore, nel nostro caso è costituito da un Arduino. Considerando l'ipotesi di un accentratore che effettua il monitoraggio di un rack con quattro sensori di temperatura, questo avrà un codice che identifica il rack mentre i sensori avranno un codice che identifica la posizione (up-top, up_middle, down_middle, down_low).
I pacchetti di tipo sentinella vengono generati dalle sentinelle quando queste stanno per compiere un'azione (shutdown di un server, invio di un'e-mail, invio di un sms, attivazione di un segnale acustico, …)

1.8 Modulo request manager

Il modulo *request manager* gestisce le richieste di informazioni presenti nel database e i comandi di impostazione dello stato di sensori, allarmi e sentinelle. Queste possono essere inviate da diverse entità, devono essere in formato UDP sulla porta 6558 del supervisore. La richiesta di dati deve avere la seguente forma (Tabella 2):

<table>
<tr><th>0</th><th>8</th><th>16</th><th>24 31</th></tr>
<tr><td colspan="4">Time Stamp</td></tr>
<tr><td>Tipo pacchetto</td><td>Tipo richiesta</td><td colspan="2">Richiedente</td></tr>
<tr><td colspan="2">Identificatore entità</td><td>Numero di valori</td><td>Tipo di valori</td></tr>
<tr><td colspan="4">Check sum (128 bit)</td></tr>
</table>

Tabella 2: Campo dati del pacchetto UDP di richiesta di dati

Nel caso di invio di comandi (Tabella 3):

0	8	16	24 31
Time Stamp			
Tipo pacchetto	Tipo richiesta	Richiedente	
Identificatore entità		Comando	Valore del comando
Check sum (128 bit)			

Tabella 3: Campo dati del pacchetto UDP dei comandi

Nel caso di invio di autoregistrazione da parte di una sentinella il campo dati del pacchetto avrà la seguente forma (Tabella 4):

0	8	16	24 31
Time Stamp			
Tipo pacchetto	Tipo richiesta	Richiedente	
Numero di warnings	Parametro del comando	Warning ID 1	
Warning ID 2		Warning ID 3	
…		Warning ID n	
Check sum (128 bit)			

Tabella 4: Campo dati del pacchetto UDP di autoregistrazione delle sentinelle

Le richieste possono provenire dalle sentinelle che si devono registrare nel supervisore e dall'interfaccia amministrativa che richiede le informazioni per permettere all'amministratore di monitorare lo stato del sistema e impostare lo stato di sensori, allarmi e sentinelle.
Il campo Tipo richiesta (byte 6) può essere di richiesta di dati o di comando. Nel caso di richieste di dati il byte 11 e 12 vengono utilizzati per definire rispettivamente il numero di dati richiesti e il tipo di dati. Nel caso di invio di un comando gli stessi byte vengono utilizzati per indicare il codice del comando da eseguire e il parametro del comando.

1.9 Struttura del DB

Il DB gestisce lo storico dei messaggi ricevuti dai sensori, dagli allarmi e dalle sentinelle. Sono definite quindi queste tre entità corredate di tutte le tabelle necessarie ad archiviare i loro stati e i valori ricevuti. Nel caso delle sentinelle vengono inoltre archiviati la lista degli warning necessari ad ogni sentinella e il log delle azioni che vengono intraprese.

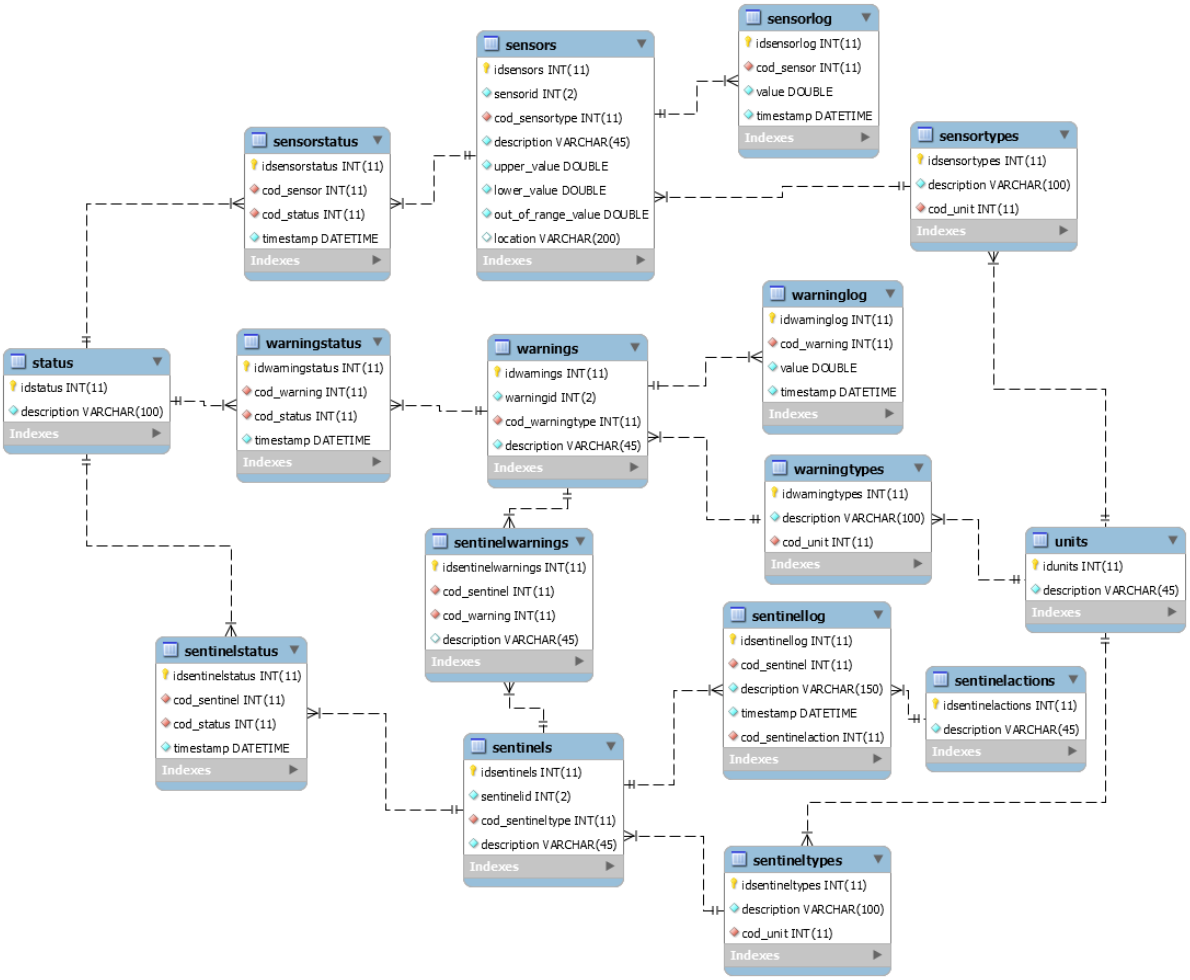

Figura 2: Struttura del Database

1.10 Stato dei sensori

Le espressioni non utilizzano il dato istantaneo dei sensori, bensì la mediana della distribuzione delle ultime letture. Il numero di dati da considerare dipende dal parametro mean_lapse, utilizzato nelle espressioni, che indica l'intervallo temporale da considerare per calcolare la mediana. La distribuzione è ottenuta interrogando il database richiedendo la serie di valori nell'intervallo mean_lapse.

Un sensore è trusted da parte del supervisore se sono tutte vere le seguenti condizioni:

- È abilitato da admin;
- È online cioè la sua epoca è "recente";
- È working cioè il suo valore è all'interno del range definito.

Se il sensore non è trusted al suo valore viene sostituito quello di default nelle espressioni dove è utilizzato. Lo status untrusted per un solo sensore presuppone ragionevolmente un possibile malfunzionamento dello stesso. Un sensore che diventa untrusted genera un allarme

che viene trattato da un'apposita sentinella che allerterà i gestori del sistema inviandogli una email e accendendo una spia sul pannello di controllo; invierà inoltre una email ai manutentori dei sensori. L'allarme (quale che sia) può essere spento dall'amministratore a seguito di una verifica del suo stato.

1.11 Stato degli warning

Agli allarmi conseguono azioni predefinite da parte dei processi sentinella (invio email, spegnimento, azionamento sirene ecc).
Ogni allarme, se abilitato, può essere preso in considerazione da una o più sentinelle, ogni sentinella gestisce uno o più allarmi.
Le tipologie di allarmi sono:

- Preallarme (non ha effetto);
- Invia una email;
- Invia una email ogni rep_t secondi;
- Azionamento sirene;
- Spegnimento condizionato con timeout;
- Spegnimento incondizionato con timeout;
- Spegnimento immediato.

1.12 Espressioni

Una delle funzioni del supervisore è l'interpretazione delle regole implementate dall'amministratore. Le regole mediano gli stati degli allarmi attraverso lo stato dei sensori e di altri allarmi. Sono codificate attraverso espressioni valutate ciclicamente.
Le espressioni combinano gli stati dei sensori e degli allarmi. Il risultato viene confrontato con le soglie definite da admin alle quali corrispondono diverse configurazioni di allarme. Le espressioni sono codificate in un file di configurazione che viene letto dal supervisore all'avvio del daemon. Il risultato della valutazione dell'espressione è un valore (float) che è confrontato con dei valori di soglia. Gli operatori di confronto ammessi sono:

Operatori di confronto:

```
== uguale a
!= diverso da
> maggiore
< minore
>= maggiore o uguale
<= minore o uguale
<< compreso tra
```

```
<=< compreso o uguale (left)
<<= compreso o uguale (right)
<=<= compreso o uguale
```

Di seguito un esempio di codifica di un allarme. L'allarme riguarda lo stato di due sensori (S01 e S02), la loro combinazione (di fatto una media pesata) è confrontata con tre soglie (trs) a cui corrispondono tre differenti stati dell'allarme Al01. I due valori passati come argomento della funzione VAL corrispondono all'identificativo del sensore e al mean_lapse (numero di campioni).

Codifica di un allarme:

```
id Al01
expr VAL('S01',120)*0.6+VAL('S02',60)*0.4
cfr <<, <=<=, >
trs [2.1, 3.6], [3.6,7.8], [7.8]
alm 00000001, 00000011, 00001111
timeout 120
```

1.13 Valutatore di espressioni

Il valutatore può essere basato sulla valutazione dell'espressione *expr* e sul confronto del valore float restituito con i valori soglia *trs.* I valori *trs* vengono confrontati utilizzando i corrispondenti operatori *cfr.* Questo confronto restituisce un valore booleano che permette la generazione dell'allarme corrispondente.

Per far funzionare il valutatore di *expr* si può, per esempio, usare la funzione *eval* di Python[3] che permette di valutare ed eseguire un'espressione nell'ambiente locale. È necessario, in questo caso, definire la funzione locale VAL.

La funzione prende due argomenti che sono una stringa *entity_id* e un intero *samples* e ha il seguente comportamento:

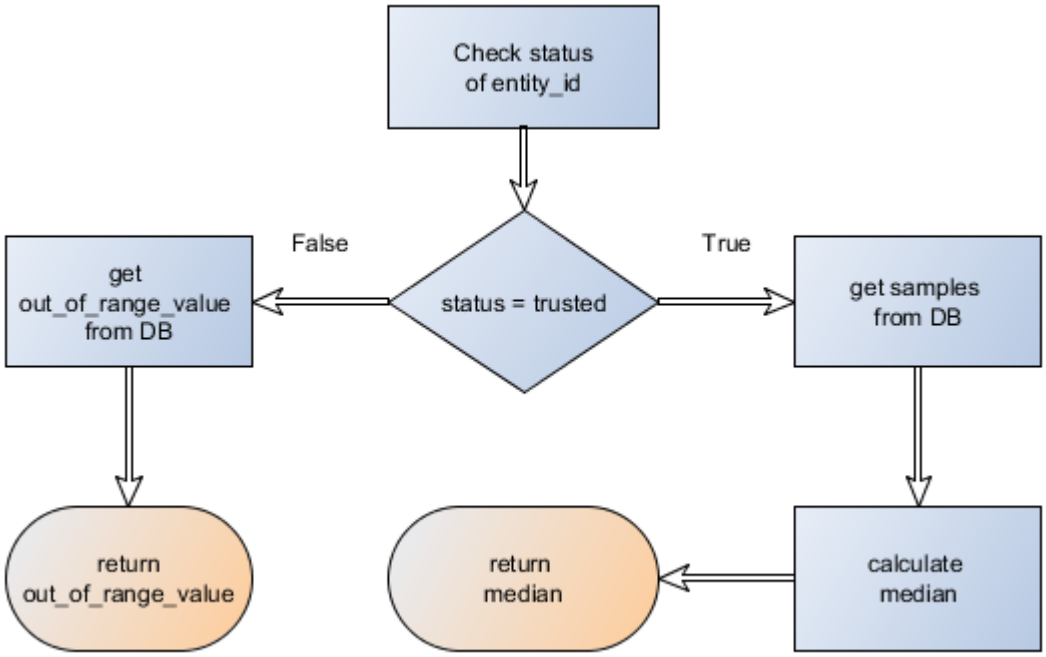

Figura 3: Schema a blocchi della funziona VAL

Le espressioni risultano quindi essere delle semplici combinazioni lineari. Valutata l'espressione questa viene confrontata tramite gli operatori con i valori di soglia *trs*.
Nel caso in cui la valutazione restituisca un valore *True* viene creato lo warning corrispondente come coppia *[id_warning, timeout]*.
Questo verrà gestito da una classe di pooling che manterrà in memoria tutti gli allarmi sino al timeout. Tramite questa classe sarà possibile annullare un allarme eliminandolo dalla lista di polling prima che venga eseguito.

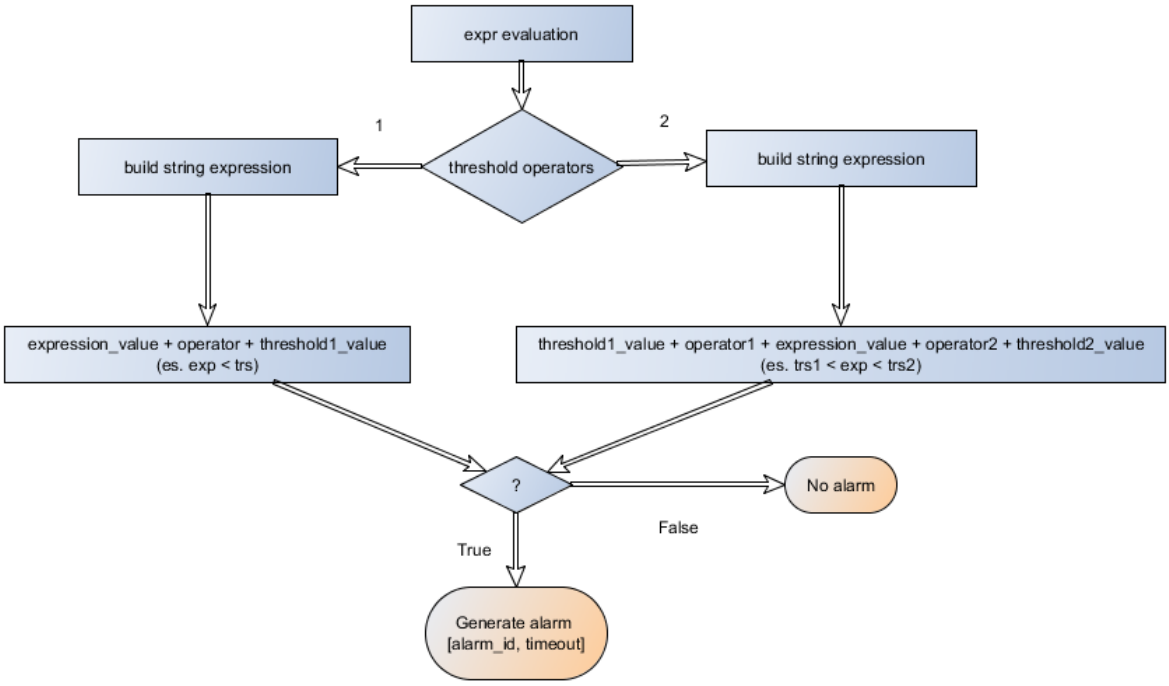

Figura 4: Schema a blocchi del valutatore di espressioni

Un esempio di pseudo-codice può essere il seguente:

```
def VAL(entity_id, samples):
    # ...
    #check status of entity_id

    if check_status(entity_id) == "trusted":
        last_samples = get_samples(entity_id, samples)
        return numpy.median(numpy.array(last_samples))
    else:
        return get_out_of_range_value(entity_id)


def eval_expression(alarm_id, expr, cfr, trs, alarm, timeout):
    expr_value = eval(expr)
    result = []
    for index in range(len(cfr)):
        if cfr[index] in ["==", "!=", "<", ">", "<=", ">="]:
```

```
            evaluation   =   eval(str(expr_value)   +   cfr[index]   +
str(trs[index][0]))
        elif cfr[index] in ["<<", "<=<", "<<=", "<=<="]:
            op1, op2 = op.split("<")
            op2 = "<" + op2
            evaluation     =     eval(str(trs[index][0])     +     op1,
str(expr_value), op2, str(trs[index][1]))
        if evaluation:
            result.append([alarm[index], timeout])
    return result
```

1.14 Frontend

Admin

Il *front end* amministrativo ha la funzione principale di visualizzare lo stato dei sensori e degli allarmi, visualizzare i log dei valori ricevuti dai sensori e dagli allarmi e le azioni intraprese dalle sentinelle.

Inoltre permette di modificare lo stato dei sensori, abilitare o disabilitare le sentinelle da parte dell'amministratore.

User

Il front end user permette la visualizzazione dello stato di tutti gli allarmi e delle sentinelle.

1.15 File di configurazione

Esempio di file di configurazione (JSON)

```
{
"info" : "rev 0.0",
"author ": "X. YYYY",
"date" : "dd/dd/dd",
"db update" : "Y",
"db clean" : "N",
"db cancel" : "N",
"expressions":
[
{
"info" : "allarme sensori rack #4",
"id" : "Al01",
"expr" : "S01(120,35)*0.8+S02(60,40)*1.2",
```

```
"cfr" : ["<<"," =<<="," >"],
"trs" : [["2.1,3.6"],["3.6,7.8"],"7.8"],
"alm" : ["00000001","00000010","00001000"],
"timeout" : "120"
},
{
"info" : "allarme sensori rack #3",
"id" : "Al02",
"expr" : "S03(220,45)+S05(10,11)*1.6"
"cfr" : " >",
"trs" : "148",
"alm" : "00000100",
"timeout" : "120"
},
{
"info" : "allarme generale",
"id" : "Al03",
"expr" : "Al01(7)*Al02(2:4)+Al02(5:8)",
"cfr" : " >",
"trs" : "0",
"alm" : "00100010",
"timeout" : "50"
},
]
}
```

2. Prototipo dimostratore

2.1 Architettura

Il sistema è costituito, nella sua architettura essenziale, da un microcontrollore per ogni rack con uno o più sensori di temperatura per un massimo di 8. Ogni microcontrollore comunica tramite interfaccia Ethernet e può inviare i suoi dati ad un management server che li analizza ed esegue delle azioni in base ad un algoritmo decisionale oppure ad un single board computer che fa da accentratore e le invia al supervisore. Si utilizzeranno dei microcontrollori con interfaccia Ethernet PoE (Power over Ethernet) per avere la possibilità di alimentarli sia tramite rete LAN sia tramite alimentazione esterna che potrà essere fornita da un alimentatore dedicato o da una porta USB.
Nell'architettura è previsto un PC dedicato all'archiviazione dei dati e al logging che nel dimostratore sarà eseguito da una Raspberry Pi 2 Model B.

2.2 Scelte implementative

Per la scelta del microcontrollore e del sensore di temperatura, ci si è indirizzati verso soluzioni testate e utilizzate in passato nei Laboratori dell'Osservatorio Astronomico di Cagliari e in particolare:

- Per il microcontrollore si è scelto l'Arduino Ethernet con alimentazione PoE;
- Per il sensore di temperatura si è scelto DHT22[4];
- Per il single-board computer si è scelto il Raspberry Pi 2 model b.

L'Arduino e la Raspberry potranno essere alloggiati in box metalliche per schermare le auto-interferenze RF (eventualmente generate) nel caso in cui vengano installate all'esterno della camera schermata e per agevolare il fissaggio nell'armadio rack. Le box potranno essere commerciali, adattate alle nostre specifiche esigenze o autocostruite. Qualora il prototipo sia collocato all'interno della camera schermata, sarà possibile utilizzare delle semplici box in ABS oppure delle Box in formato 1U.
Va evidenziato il fatto che le box per la Raspberry potranno essere acquistate come prodotti commerciali mentre quelle per l'Arduino, richiedendo una scheda aggiuntiva atta ad ospitare i connettori, dovranno essere realizzate ad-hoc.

2.3 Assorbimento elettrico

Per quanto riguarda l'assorbimento elettrico dei microcontrollori, nel caso di soluzione PoE l'assorbimento cautelativo è di 5W.
I sensori sono predisposti inoltre per essere alimentati tramite alimentatore in corrente continua da 9V e 500mA. Questo permette un riutilizzo della sensoristica anche in altre architetture dove non è disponibile il PoE.
Sarà necessario predisporre un UPS dedicata allo switch di rete, al single board computer, al PC dedicato all'archiviazione dei dati che fornirà inoltre, l'interfaccia web per il monitoraggio da remoto. Tale UPS dovrà essere alimentato da una linea elettrica dedicata.

2.4 Il Raspberry Pi

Sul single-board computer scelto può essere installato il sistema operativo voluto che risiederà, come tutti i dati, su una scheda MicroSD. Questa caratteristica permette di agevolare le operazioni di aggiornamento e manutenzione del sistema operativo e del programma di supporto alle decisioni. Inoltre agevola la sostituzione di tutto il computer procedendo solo alla replica del cablaggio e allo spostamento della MicroSD.

2.5 Manutenzione

La manutenzione prevista è stata pensata per evitare l'intervento di personale specializzato e consiste nella sostituzione di parti che richiedono solo la ricollocazione e la connessione di pochi cavi dotati di connettori standard con un unico verso di inserimento.

- Sostituzione del sensore:
    1. Il sensore sarà provvisto di un connettore che permetterà l'eventuale sostituzione senza rimuovere il cavo di connessione al microcontrollore.
    2. La sostituzione non prevedrà operazioni dal punto di vista software.
- Sostituzione del microcontrollore:
    1. Il microcontrollore può essere sostituito ricollegando il connettore di rete, il cavo del sensore ed eventualmente il connettore dell'alimentazione in caso di soluzione non PoE.
    2. È previsto che siano disponibili dei microcontrollori spare già programmati e che quelli sostituiti siano riportati in laboratorio per l'eventuale riparazione e diagnostica di malfunzionamenti.
    3. La sostituzione non prevedrà operazioni dal punto di vista software.

- Sostituzione del single-board computer:
    1. Nel caso in cui sia necessaria la sostituzione della scheda MicroSD, è prevista la disponibilità di schede MicroSD dove sia già installato il sistema operativo. In questo caso sarà necessario riavviare il computer interrompendo l'alimentazione.
    2. Nel caso in cui sia necessaria la sostituzione del single-board computer si potrà procedere all'utilizzo di uno spare al quale andranno ricollegati il cavo di rete e quello di alimentazione, oltre all'inserimento della MicroSD prelevata dal single-board computer che dovrà andare in laboratorio.
    3. La sostituzione dell'alimentatore del single-board computer è prevista tramite un suo spare.
- Manutenzione della parte firmware e software:
    1. Questa parte va pianificata come tutti i rilasci di nuove release di software con il test di versione alpha e beta prima del rilascio finale.
    2. In generale la maggior parte delle operazioni di aggiornamento del firmware e del software potranno essere effettuate da remoto.

Le segnalazioni di guasto e di malfunzionamento delle varie parti del sistema saranno comunicate a cascata dal microcontrollore e dal single-board computer. Quest'ultimo sarà monitorato da un ulteriore computer di controllo che ne verificherà lo stato on-line.
Il single-board computer scelto è provvisto di una scheda audio integrata che permetta di avere un sistema di allerta acustico nel caso di superamento delle soglie dei parametri ambientali qualora servisse.

2.6 Protocollo di comunicazione

Gli Arduino Ethernet comunicano i valori dei sensori ad essi collegati tramite Ethernet.
I sensori restituiscono un valore di 16 bit per la temperatura e 16 bit per l'umidità più un check sum di 8 bit per un totale di 40 bit.
Per la programmazione del firmware dell'Arduino è necessario definire:

- MAC address da assegnare alla scheda di rete;
- IP da assegnare all'interfaccia di rete;
- IP del server di destinazione dei pacchetti;
- Tempo di attesa per ogni lettura dei sensori (t);

- La porta di destinazione utilizzata per la comunicazione con il server (6555 non utilizzata da applicazioni note);
- La porta di origine (6557 non utilizzata da applicazioni note) [2];
- Un identificativo da associare ad ogni sensore;
- Un identificativo da associare all'Arduino.

Il compito del firmware è effettuare una lettura di tutti i sensori collegati per ogni intervallo di tempo *t* ed inviare un pacchetto UDP per ogni sensore.
Il ruolo della Raspberry è quello di gestire gruppi di Arduino che a loro volta gestiscono tra gli 1 e gli 8 sensori. In questo modo è possibile organizzare la gestione di sensori in gruppi. I pacchetti ricevuti dagli Arduino verranno rispediti al server di monitoraggio dove avverrà la sovrascrittura del timestamp con quella del sistema. La Raspberry a differenza degli Arduino, può essere facilmente configurabile da remoto modificando per esempio l'IP del server di monitoraggio o facendo delle analisi sui pacchetti UDP in arrivo dagli Arduino prima che questi vengano reinviati al server di monitoraggio.

2.7 Dimensionamento Switch PoE.

Per quanto riguarda lo Switch PoE, questo è stato dimensionato per un architettura in cui sia presente uno switch per ogni rack con 4 PDU. È necessario quindi uno switch PoE da almeno 8 porte dedicandone 4 per le PDU con controllo di rete remoto, una per l'Arduino e una, dove previsto, per il supervisore.

2.8 Piano dei costi e dei tempi

Il prototipo dimostratore che è stato pensato per questo sistema prevede l'utilizzo di 16 sensori di temperatura, 4 Arduino Ethernet con PoE, 2 Raspberry Pi, uno switch di rete con funzionalità PoE, cavi e connettori necessari. Per il prototipo non è necessario realizzare le box in metallo ma sarà necessario acquistare almeno quelle in plastica.
Non essendo ancora stato realizzato il server di monitoraggio, per il prototipo dimostratore, il compito della Raspberry PI è anche quello di gestire la comunicazione degli Arduino.
Le fasi della realizzazione del prototipo, con indicate le ore/giorni uomo sono:

- Progettazione delle schede che ospiteranno i singoli sensori (2h);
- Realizzazione delle schede (3h);
- Montaggio componenti e collaudo delle schede (8h);
- Progettazione e sviluppo del firmware per l'Arduino e test (8h);

- Installazione Rasperry Pi, progettazione e sviluppo del solo software di comunicazione (8h);
- Test del sistema Rasperry, 4 Arduino più sensori (2g);
- Test del sistema completo più stress test (3g);
- Test funzionale del sistema completo una volta installato nel CED dell'Osservatorio (5g);

I costi di realizzazione del prototipo si dividono in costi per il materiale utilizzato e costi per il personale impegnato.

I costi per il materiale (IVA e spese di spedizione escluse) sono visualizzati in Tabella 5.

Materiale	Prezzo unitario	Quantità	Totale
Sensori di temperatura DHT22	11,39€	16	182,24€
Arduino Leonardo Ethernet with POE	52,35€	4	209,40€
Raspberry Pi 2 Model B	31,90€	2	63,80€
Switch di rete con PoE	100,00€	1	100,00€
Femmina da pannello 4 poli DIN EN 60529	2,93€	16	46,88€
Maschio da cavo 4 poli DIN EN 60529	3,70€	16	59,20€
Femmina da cavo 4 poli DIN EN 60529	4,90€	16	78,40€
Connettori 2.54mm maschio 4 poli a saldare	0,568€	50	28,40€
Connettori 2.54mm femmina da cavo 4 poli	0,127€	50	6,35€
Confezione da 1000 connettori a crimpare 2.54mm molex	0,055€	1000	55,00€
Cavo ethernet da 3 metri preassemblato cat6	3,40€	10	34,00€
Bobina di cavo 4 poli per i sensori	58,71€	1	58,71€
Bobina di stagno	32,60€	1	32,60€
Guaina per cavi twistata HellermannTyton, in PET, Nero, Ø manicotto 15mm x 10m, Ø cavo 5mm - 21mm max	21,33€	1	21,33€
Punta Weller T0054444099 per MPR 80, WP80, WSP80, 1,6 mm, Scalpello diritto	4,70€	4	18,80€
Scatole in plastica per Raspberry	6,00€	2	12,00€
Totale			**1007.11€**

Tabella 5: Costi per la realizzazione del prototipo IVA esclusa

Per quanto riguarda le box dell'Arduino, potranno essere autoprodotte oppure realizzate da aziende esterne per un costo indicativo di 10 € massimi cad.

I tempi uomo totali sono previsti in 1 mese uomo per 2 unità di personale: una dedicata alla progettazione e realizzazione della parte elettronica; una dedicata alla progettazione e sviluppo della parte firmware/software.

Inoltre, vanno aggiunti 5 giorni uomo per la eventuale realizzazione delle box in plastica.

3. Conclusioni

In questo report è stata illustrata l'architettura di un sistema di supervisione di un CED ponendo l'accento sulle criticità di un ambiente in cui deve essere garantita la sicurezza delle persone e dei sistemi, con la particolarità che il supervisore si appoggia su una parte dello stesso sistema che deve monitorare.

La realizzazione presso l'Osservatorio Astronomico di Cagliari del sistema di supervisione permette di mettere a frutto le professionalità del personale maturate in diversi settori. Inoltre permette di contenere i costi di realizzazione e di manutenzione e di rendere disponibile alla comunità un sistema aperto ed adattabile alle diverse esigenze. Questi vantaggi verrebbero meno con una soluzione commerciale chiusa e legata a contratti di manutenzione e di aggiornamento.

Il paradigma alla base dell'architettura è quello di dividere il sistema in parti modulari. Per ognuna di queste è stato definito il protocollo di comunicazione con gli altri moduli. Nel caso di comunicazioni con sistemi proprietari saranno definiti degli opportuni moduli di traduzione del protocollo. La definizione del formato delle regole, che traducono le informazioni in azioni o eventi, lascerà all'amministratore la più ampia libertà.

Questo approccio modulare, permette di implementare i moduli in maniera indipendente e/o di sostituirli in futuro con delle versioni che ne estendono le funzionalità e le prestazioni. I moduli sono stati pensati per essere implementati con soluzioni aperte sia software (LAMP, Python, …) sia hardware (Arduino, Raspberry, …).

La realizzazione della parte di sensoristica è stata incentrata sul paradigma dell'economicità, del minimo consumo energetico, della riusabilità e della durabilità. L'Arduino e la Raspberry sono infatti affidabili e robusti, i sensori possono essere connessi facilmente e quindi sostituiti in breve tempo per eventuali manutenzioni. Il prototipo dimostratore sarà totalmente riutilizzabile in un'architettura definitiva, e per fare questo sono state scelte soluzioni che rendono i sensori e le box semplici da posizionare e da cablare.

La fase di realizzazione permetterà di lavorare in parallelo dal lato hardware e software/firmware e a questa fase seguirà quella di test dei protocolli e di tuning delle regole del sistema.

4. Riferimenti

[1] https://www.arduino.cc/en/Main/ArduinoBoardEthernet

[2] https://www.raspberrypi.org/

[3] G. van Rossum, Python tutorial, Technical Report CS-R9526, Centrum voor Wiskunde en Informatica (CWI), Amsterdam, May 1995.

[4] https://cdn-shop.adafruit.com/datasheets/Digital+humidity+and+temperature+sensor+AM2302.pdf