Relazione e Analisi consegna 4

Matteo Minardi, Lorenzo Chiana.
2016/2017
Corso di Programmazione di Sistemi Embedded

Raspberry

Architettura Hardware

Si compone di una scheda Raspberry Pi 3 Model B con piedinatura GPIO configurata come seguente:

- Pulsante tattile T1 -> pin GPIO 2
- Led verde L1 -> pin GPIO 3
- Led verde L2 -> pin GPIO 4
- Led rosso L3 -> pin GPIO 5

Architettura software

Nell'implementazione Java di Raspberry è stato implementata un'archietettura Event Loop che disaccoppia il flusso di controllo che genera gli eventi (l'arrivo di messaggi su Serial) e quello che esegue gli handler (Classe EventTracker). All'interno dell'event tracker è presente una logica a stati che indica se la connessione con Arduino è funzionante.

Le classi su cui ci siamo concentrati sono essenzialmente 4:

- MainEventTracker: è il main che viene eseguito che prende due argomenti. Il primo è la porta su cui è mappato Arduino e il secondo è la directory del web server che contiene i file json passati all'apposito controller.
- EventTracker: è la classe che contiene l'handler dei vari eventi che possono essere generati (ConnectrionMsg, PresenceMsg, TemperatureMsg) e contiene anche i componenti istanziati nel main connessi alla GPIO attraverso gli appositi oggetti astratti pi4j forniti in laboratorio.
  Questa classe contiene anche un timer che ogni periodo di 5 secondi verifica che sia stato ricevuto un ack che confermi la connessione con arduino.
- SerialReciver: questa classe è un thread che ad ogni ciclo aspetta un messaggio dalla seriale passata nel costruttore e fa processare l'evento ad un oggetto esterno.
- JSONController: è una classe creata che incapsula tutte le funzionalità json che mi erano necessarie facendo uso di JSON.simple. Come funzionalità ha quella di Parsing del contenuto, aggiunta in coda (append) e scrittura (write) di ciò che vi è stato inserito.

Librerie

Il sistema operativo, basato su Debian, esegue un'applicazione Java avente le seguenti librerie:

- RXTXcomm: gestisce la comunicazione seriale asincrona (in questo caso via USB).
- pi4j: interfaccia i pin GPIO (40) attraverso classi dedicate fornite negli esercizi di laboratorio di vario tipo (digitale I/O, analogico I/O, utility...)
- JSON.simple: parser dei file JSON.

Arduino

Architettura Hardware

Si compone di una scheda Arduino Uno con i seguenti sensori/attuatori:

- Sensore di temperatura digitale TMP36
- Sensore di movimento PIR
- Servomotore (gestito con libreria ServoTimer2)

- Modulo seriale Bluetooth HC-06

Architettura software

Il programma è composto da 3 task:

- DetectPresenceTask (T: 100ms): Si occupa di gestire l'allarme (rappresentato dal movimento del servo), segnalando il rilevamento di una persona da parte del PIR all'app Android via Bluetooth. Ogniqualvolta il sensore PIR rileva un cambiamento di stato (basso, alto) avverte il dispositivo e invierà un nuovo eventuale allarme solamente dopo che sarà tornato basso.
  Successivamente al rilevamento, viene atteso il messaggio di conferma presenza o di attivazione allarme dall'app; se non viene ricevuto alcun messaggio, l'allarme viene attivato dopo la scadenza default di 10 secondi con conseguente attuazione fisica.
- TemperatureTask (T: 500ms): Si occupa di inviare al sistema Raspberry Pi il valore della temperatura (periodicamente ogni 5 secondi). Semplicemente tiene un timer che indica se sono passati, nel caso la condizione sia vera scatta un'azione che ne invia il segnale e resetta il timer contestuale del task.
- ConnectionTask (T: 300ms): Questo task ha un ruolo simile a quello di watchdog. Per controllare che il sistema funzioni effettivamente ogni secondo manda un messaggio nel cavo seriale. Questo messaggio è best effort e viene elaborato dal Rasperry per capire se l'arduino è attivo oppure no. In questo modo può gestire il led di funzionamento in un modo leggermente più efficiente e orientato alla connessione.

La classe BluetoothMsgService gestisce la comunicazione Bluetooth con l'app Android con l'uso della libreria SoftwareSerial, mentre la classe SerialMsgService gestisce la comunicazione con il sistema Raspberry Pi tramite USB con il protocollo UART, sfruttando Serial. Entrambe vengono incapsulate all'interno della classe Environment globale che contiene i due bus principali comuni a tutti i task.

I 3 task sono gestiti dallo scheduler a interrupt, modularizzano il problema e vengono semplicemente schedulati in base all'ordine in cui vengono aggiunti in fase di setup. La scelta dei periodi di clock è stata fatta in base all'importanza della reattività in quel determinato task infatti sapere che vi è un allarme necessità di più priorità rispetto al monitoraggio della temperatura.

Sito HTTP

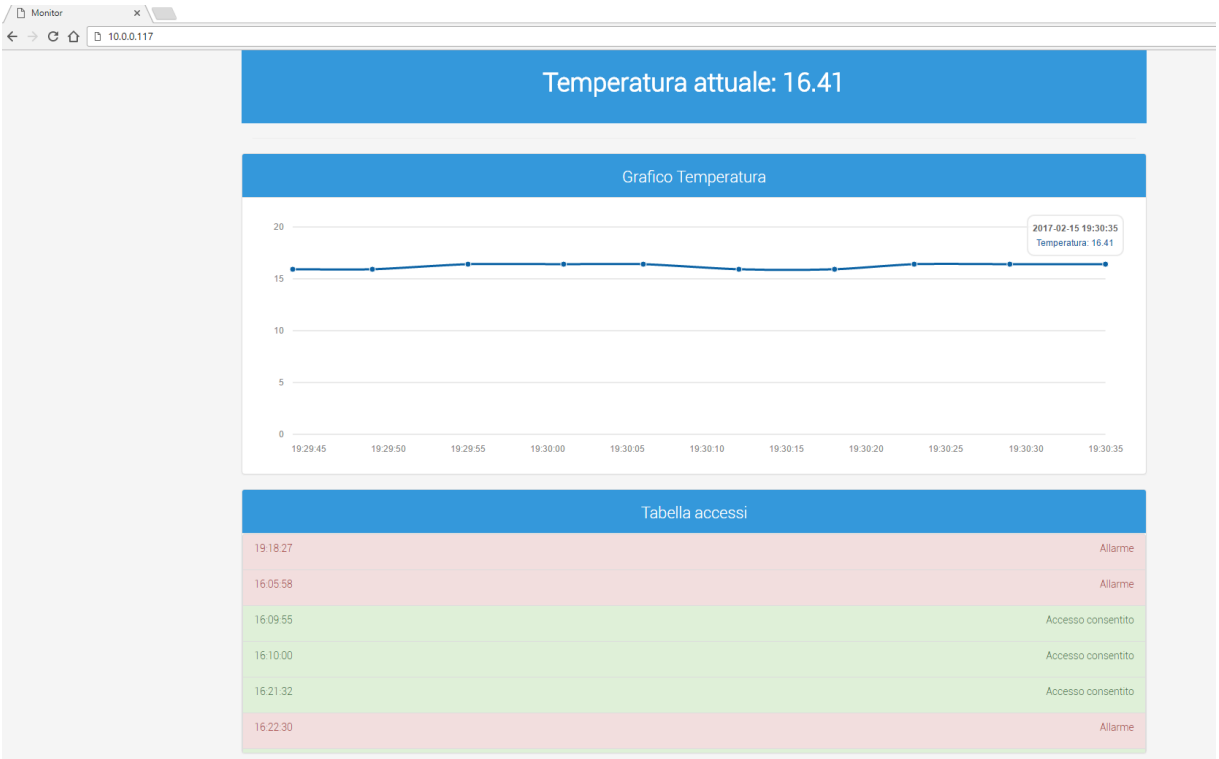
Architettura Hardware

Per rendere consultabili i dati rilevati dal sistema S_2 è stato scelto di sviluppare un sito web minimale che faccia da interfaccia verso l'utente. E' stato scelto questa implementazione per sperimentare il concetto di Web of Things e collegarsi al corso di Tecnologie Web.
Il sito è contenuto in un server apache in funzione su Raspberry Pi accessibile da rete locale digitando il suo indirizzo IP in un qualsiasi bowser mobile o pc.
Il sito utilizza librerie come:

- Bootstrap per la grafica,
- Morris.js per il grafico dei dati
- jQuery per le richieste asincrone di reperimento dati JSON.

Logica funzionamento

Il sito inizialmente non contiene temperature o informazioni su accessi. Appena entra in funzione la comunicazione, il programma Java su Raspberry Pi ottiene dati da Arduino e li salva su appositi JSON contenenti tempo di rilevazione e valore informativo. Ogni 200ms il sito si aggiorna e di conseguenza ottiene nuovamente tutti i dati JSON eventualmente aggiornati. Rappresenta graficamente il risultato nel modo opportuno con colori e label significative.



Android

Architettura software

L'applicazione Android si compone principalmente di:

- **MainActivity**: main activity che fa visualizzare lo stato della connessione Bluetooth.
  Se questa activity si trova in foreground mentre gli arriva un messaggio da Arduino che segnala la situazione di allarme, viene creata e lanciata una finestra di dialogo dove avvertirà l'utente della situazione.
- **AlarmDialogFragment**: finestra di dialogo con due bottoni nella quale verrà richiesto all'utente se segnalare la situazione di allarme. L'iterazione tra utente e applicazione avverrà attraverso i due bottoni (Sì e No).
- **MessageService**: quando la main activity viene messa in background viene avviato questo service e si occupa di creare e mostrare una notifica se viene rilevata una situazione di allarme. In questo caso se si clicca sopra alla notifica si viene rimandati alla AlarmActivity.
- **AlarmActivity**: una seconda activity creata e lanciata a seguito della pressione sulla notifica. In questa activity viene chiesto all'utente se segnalare la situazione di allarme e quest'ultimo risponderà premendo su uno dei due bottoni presenti (Sì e No).
- **MessageHandler**: handler che si occupa dell'inoltro dei messaggi. Nel caso in cui la main activity sia in foreground allora i messaggi vengono inoltrati ad essa, in caso contrario vengono inoltrati al service.

Nel package "bt" sono presenti tutte le classi per la gestione della comunicazione Bluetooth che ci sono state presentate durante le lezioni.