UNIVERSITÀ DEGLI STUDI DI MILANO

FACOLTÀ DI SCIENZE E TECNOLOGIE

Corso di Laurea in Informatica

CONVERSIONE DI STRUMENTI VINTAGE PER LA DATA PHYSICALIZATION

Relatore: Andrea Trentini

Tesi di Laurea di: Davide Busolin
Matr. 930814

Anno Accademico 2020/2021

Indice

1 Introduzione **2**

2 Attori coinvolti **3**

3 Funzionamento **4**
3.1 All'avvio 4
3.2 Modalità operative 5
3.2.1 Modalità API 5
3.2.2 Modalità MQTT 6

4 Conclusioni **7**
4.1 Reperibilità delle API 7
4.1.1 Dettagli su ATM 7

Capitolo 1

Introduzione

TCP/IP over Avian Carriers[2]

Capitolo 2

Attori coinvolti

Prima di parlare del funzionamento dell'oggetto è il caso di soffermarsi su chi interagisce con esso:

Sviluppatore

È in grado di apportare modifiche al codice sorgente. Può aggiungere, rimuovere e modificare funzionalità del programma e apportare modifiche dirette ai file di configurazione.

Utente esperto

Conosce le API ed è in grado di comprendere il formato dei dati che restituiscono. Gli è sufficiente un'interfaccia anche spartana per selezionare la sorgente dei dati e il campo specifico che vuole rappresentato.

Utente inesperto

Non conosce le API né il formato JSON: ha bisogno di un prodotto già pronto che richieda la minima configurazione possibile e questa deve essere particolarmente intuitiva, ad esempio una piccola interfaccia web per scegliere la rete WiFi e inserirne la password al primo avvio. La sorgente dei dati deve essere preconfigurata e se ne viene resa disponibile più di una la scelta deve essere molto semplice, possibilmente tramite interazione fisica con il dispositivo.

Capitolo 3

Funzionamento

3.1 All'avvio

Per prima cosa contestualmente all'accensione, mediante WiFiManager, viene effettuato un tentativo di connessione all'ultima rete WiFi utilizzata, di cui sono state salvate le credenziali. Se non è mai stata effettuata la connessione ad alcuna rete WiFi o se quella salvata non è disponibile, viene attivato un Access Point che tramite Captive Portal (lo stesso che viene usato su reti pubbliche o aziendali per richiedere l'autenticazione) chiede di selezionare una delle reti WiFi vicine e di inserirne la password.

[SCREENSHOT]

Successivamente viene tentata la connessione. Se non ha successo viene riproposto lo stesso procedimento, altrimenti viene disattivato l'Access Point temporaneo e il programma prosegue.

3.2 Modalità operative

3.2.1 Modalità API

Questa modalità di operazione si basa sull'esecuzione ad intervalli di tempo configurabili di richieste HTTP o HTTPS ad API REST e sulla conversione del risultato in una tensione di uscita. La configurazione è salvata nella memoria flash del microcontrollore in formato JSON, esempio:

```
{
  "apiUrl": "http://api.coindesk.com/v1/bpi/
     currentprice/USD.json",
  "filterJSON": "{bpi:{USD:{rate_float:true}}}",
  "path": "bpi/USD/rate_float",
  "min_value": 58700,
  "max_value": 58900,
  "min_pwm": 0,
  "max_pwm": 1023,
  "request_interval_ms": 15000
}
```

Dove:

- `apiUrl` è l'URL della risorsa (stringa)
- `post_payload` in caso di richiesta POST contiene la stringa da inviare nel body
- `filterJSON` è un documento JSON che contiene `true` come placeholder del campo che si vuole considerare dalla risposta (stringa/oggetto)
- `path` è una stringa che contiene il "percorso" del campo che si vuole rappresentare fisicamente (campo: float, path: stringa)
- `min_value` indica il valore minimo della scala per rappresentare il valore (float)
- `max_value` indica il valore massimo (float)
- `min_pwm` indica il valore minimo di uscita della PWM da mappare al valore restituito dalle API (int)
- `max_pwm` indica il valore massimo [su ESP il duty cycle massimo equivale a 1023] (int)

- `request_interval_ms` indica l'intervallo di tempo in millisecondi che intercorre tra le richieste alla risorsa (int)

La configurazione può essere alterata tramite una pagina web accessibile all'indirizzo IP nella rete locale del microcontrollore o il suo hostname.

[SCREENSHOT PAGINA WEB]

I form sono precompilati con la configurazione in esecuzione (*running-conf*) ed è presente una sezione che riporta la configurazione salvata nella flash (*saved-conf*).
È possibile inoltre inviare un JSON di configurazione tramite MQTT al topic `Phys/setFromJSON`. I campi possono essere modificati singolarmente ai topic:

- `Phys/setApiUrl`
- `Phys/setFilterJson`
- `Phys/setPath`
- `Phys/setMinValue`
- `Phys/setMaxValue`
- `Phys/setMinPwm`
- `Phys/setMaxPwm`
- `Phys/setRequestIntervalMs`

Le modifiche non sono automaticamente salvate nella memoria flash ma ciò deve essere richiesto esplicitamente. Nei form di configurazione è presente una checkbox da spuntare in caso si voglia che questo avvenga.

3.2.2 Modalità MQTT

Non ancora implementata, ma l'idea è che consenta di visualizzare un valore ricevuto direttamente tramite sottoscrizione a un topic MQTT invece che andarlo a reperire tramite richieste HTTP(S).

Capitolo 4

Conclusioni

4.1 Reperibilità delle API

Durante la ricerca di API rappresentabili mediante Data Physicalization è risultato evidente che molte di quelle più congeniali non sono pubbliche e spesso sono anche di difficile utilizzo. L'esperienza personale è stata con quelle di ATM per ottenere i minuti rimanenti all'arrivo di un mezzo pubblico ma altri si sono scontrati con quelle di Trenitalia [1]

4.1.1 Dettagli su ATM

Le API di ATM sono usate per Giromilano e l'unico modo per comprenderne il funzionamento è utilizzare gli strumenti del browser per leggere le richieste e le risposte che vengono scambiate tra client e server durante la ricerca di un percorso o di informazioni su una fermata specifica.
In particolare per ottenere i minuti rimanenti all'arrivo di un mezzo ad una specifica fermata, è necessario effettuare una richiesta POST a `https://giromilano.atm.it/proxy.ashx` e nel body inserire `url=tpPortal/geodata/pois/stops/xxxxx`, dove a `xxxxx` va sostituito il numero della fermata, leggibile (a volte) nel mondo reale sulla pensilina o sul cartello o direttamente dal sito.
Per ottenere una risposta è inoltre necessario inviare l'header `Content-Type: application/x-www-form-urlencoded`.
A questo punto sorgono alcuni problemi:

- Le risposte contengono un po' di tutto e per via degli avvisi comprensivi di tag HTML possono anche essere molto lunghe, anche decine di kb.

- In caso di richiesta non corretta, ad esempio mancante di header Content-Type, non si avranno in risposta codici HTTP specifici ma sempre 200 (OK) in cui però la risposta è vuota.

A prescindere dalle questioni tecniche il problema è anche legale: non essendo API pubbliche l'utilizzo che se ne può fare all'esterno dell'ambito aziendale è molto limitato. Può andare bene per utilizzi sperimentali personali, ma non si possono certamente incorporare in prodotti da distribuire a qualsiasi titolo.
Poter risalire al ritardo accumulato da un mezzo pubblico (specialmente per treni con cadenza oraria o semioraria) senza dover navigare siti e app ma direttamente lanciando un'occhiata a un angolo della stanza, come si farebbe con un orologio, sarebbe una comodità in più per i pendolari che al momento non è attuabile su larga scala da terzi e le aziende di trasporti non sembrano interessate a proporre prodotti simili.

Bibliografia

[1] Alberto Giunta. +++trenitalia shock+++ non crederete mai a queste api *painful*. https://medium.com/@albigiu/trenitalia-shock-non-crederete-mai-a-queste-api-painful-14433096502c, 2018.

[2] David Waitzman. Standard for the transmission of ip datagrams on avian carriers. Technical report, 1990.