Université de Toulon

IUT de Toulon

Département Génie Electrique et Informatique Industrielle (GEII)

SAE

Shelly HTTP

écrit le 14 avril 2024

par

Bruno HANNA

Encadrant universitaire : Stephane PIGNOL

Table des matières

Introduction **1**

1 Requête HTML **2**

2 Procédure de Test **3**

2.1 Test du Relai Intégré 3

2.1.1 Structure de la requête GET 3

2.1.2 Tests et vérifications 3

2.2 Test des Sondes 5

Introduction

Ce rapport explore et démontre le mode de communication du SHELLY EM, un dispositif intelligent de mesure de l'énergie électrique, utilisant le protocole HTTP pour échanger des données avec les réseaux informatiques. À travers une démarche pratique, l'utilisation d'outils tels que Wireshark, Chrome Network Developer Tools et Telnet permet de capturer et d'analyser les échanges de données entre le SHELLY EM et un ordinateur connecté au même réseau. L'analyse vise à mettre en lumière les spécificités de la communication HTTP du dispositif, fournissant une preuve concrète de son fonctionnement et illustrant la manière dont il peut être géré et interrogé à distance. Destiné aux professionnels de l'informatique et aux passionnés de technologie, ce rapport offre une compréhension approfondie des principes sous-jacents à la communication des dispositifs IoT avec leur environnement.

Chapitre 1

Requête HTML

HTTP GET

La méthode GET est utilisée pour demander la représentation d'une ressource spécifique. Les requêtes GET devraient uniquement récupérer des données et ne pas avoir d'effet sur l'état de la ressource.

HTTP POST

POST est utilisée pour envoyer des données à une ressource spécifiée, causant souvent un changement d'état ou des effets secondaires sur le serveur. Elle est fréquemment utilisée pour soumettre des formulaires.

HTTP PUT

La méthode PUT remplace toutes les représentations actuelles de la cible de ressource avec les données de la requête. C'est une méthode utilisée pour mettre à jour une ressource existante ou créer une nouvelle ressource si elle n'existe pas.

HTTP DELETE

DELETE est utilisée pour supprimer une ressource spécifiée. L'opération de suppression est idempotente, ce qui signifie que plusieurs requêtes DELETE successives ont le même effet qu'une seule requête, sans effet secondaire.

Chapitre 2

Procédure de Test

Ce chapitre détaille la procédure de test visant à vérifier le bon fonctionnement de la Shelly EM via communication HTTP.

2.1 Test du Relai Intégré

L'interface de contrôle du relai offre trois commandes principales via HTTP GET, permettant d'interagir avec le dispositif. La syntaxe ? dans l'URL est utilisée pour passer des paramètres à la requête GET, ce qui est essentiel pour spécifier les commandes et les états souhaités du relai. Les commandes disponibles sont :

- `GET /relay/0` : Cette commande récupère l'état actuel du relai et retourne les données sous forme d'un objet JSON.
- `GET /relay/0?turn=on` : Envoie une requête pour activer le relai.
- `GET /relay/0?turn=off` : Envoie une requête pour désactiver le relai.

2.1.1 Structure de la requête GET

L'utilisation du caractère ? dans une URL signale le début des paramètres de la requête. Chaque paramètre est ensuite exprimé sous la forme clé-valeur, séparé par = et chaque paire supplémentaire est séparée par &. Cette syntaxe permet de transmettre des informations spécifiques au serveur, qui peut alors modifier son comportement en fonction des instructions reçues.

2.1.2 Tests et vérifications

Les tests de communication avec la commande `GET /relay/0` sont effectués selon deux méthodes distinctes pour assurer l'intégrité et la fiabilité des échanges.

Méthode 1 : Utilisation de Chrome et Wireshark

Utilisation de Chrome configuré pour exclure les adresses IP locales du proxy, avec l'adresse `192.168.0.210/relay/0`. Le mode développeur de Chrome, via la section *Network*, permet de suivre tous les échanges réseau entre le navigateur et le dispositif.

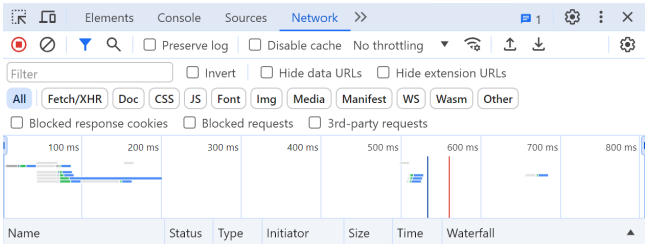

Figure 2.1 – Mode NetWork intégrer à Chrome

Méthode 2 : Utilisation de Telnet

Emploi de Telnet pour établir une connexion directe avec le dispositif via la commande suivante :

```
cmd telnet 192.168.0.210
GET /relay/0 HTTP/1.1
```

Ensuite, envoi de la requête `GET /relay/0` pour obtenir l'état du relai. La réponse du serveur est capturée et analysée pour vérifier la conformité des données retournées avec les attentes.

```
HTTP/1.1 200 OK
Server: Mongoose/6.18
Connection: close
Content-Type: application/json
Content-Length: 139
{"ison": false,"has_timer": false,"timer_started": 0,"timer_duration": 0,
    "timer_remaining": 0,"overpower": false,"is_valid": true,"source": "
    http"}
```

avec la confirmation d'échange de paquet grâce à WireShark :

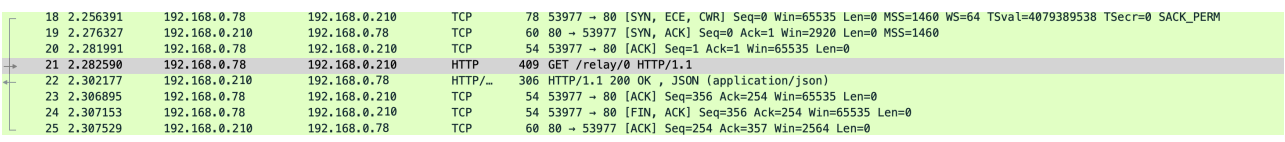

Figure 2.2 – Capture Wireshark de l'échange TCP pour le relai

Les deux méthodes fournissent des moyens complémentaires pour tester et vérifier la communication avec le relai, utilisant à la fois des outils de développement web et des applications réseau spécialisées.

2.2 Test des Sondes

Cette section est dédiée à la collecte des valeurs mesurées par les sondes énergétiques, implémentant deux commandes spécifiques :

— `GET /emeter/0` pour relever les données du premier émetteur.

— `GET /emeter/1` pour relever les données du second émetteur.

La méthode Telnet est employée pour tester ces commandes. La procédure est détaillée ci-dessous pour le premier émetteur :

```
cmd telnet 192.168.0.210
GET /emeter/0 HTTP/1.1
```

La réponse obtenue est :

```
HTTP/1.1 200 OK
Server: Mongoose/6.18
Connection: close
Content-Type: application/json
Content-Length: 103
{"power":10.08,"reactive":10.01,"voltage":244.35,"is_valid":true,"total"
    :177.1,"total_returned":204.0}
```

La vérification de cette transaction est réalisée à l'aide de Wireshark pour assurer l'intégrité des données transmises.

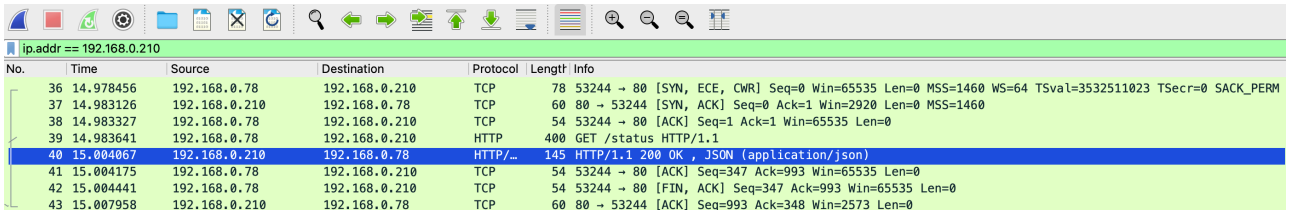

Figure 2.3 – Capture Wireshark de la transaction pour `GET /emeter/0`

Les résultats montrent une consommation de 10 W à une tension de 244 V, ce qui est conforme aux attentes basées sur les caractéristiques physiques et opérationnelles de l'installation testée.