

Contrôle d'Alarme JEEDOM par badge



Table des matières

1 Introduction	2
2 Conception du lecteur de badge	2
3 Configuration de l'ESP8266.....	3
3.1 Onglet Config	4
3.2 Onglet Controllers.....	5
3.3 Onglet Hardware.....	6
3.4 Onglet Devices.....	7
3.5 Onglet Tools/Advanced.....	8
3.6 Onglet Tools/Log.....	8
3.7 Utilisation des règles	10
4 Configuration du Plugin dans Jeedom.....	12
4.1 Configuration du Plugin ESPEasy.....	12
4.2 Scenario Jeedom.....	13
5 Schéma d'interconnexion des modules.....	14

1 Introduction :

L'objet de ce document est présenter la réalisation d'un système de contrôle de l'alarme Jeedom a partir d'un lecteur de badge type RFID. Ce système de contrôle doit permettre d'activer ou de désactiver le système d'alarme à distance au moyen de badges RFID distribué à chaque utilisateur. Pour une question de sécurité, l'activation ou désactivation de l'alarme ainsi que l'acquiescement des commandes reçues se fait par Jeedom. Avec ce système, il n'est pas directement possible de sélectionner les zones de protection. Néanmoins, on peut le faire en utilisant un badge différent par zone, à condition de ne pas avoir trop de zones à contrôler.

Pour une question pratique, le lecteur de badge est déporté de Jeedom au moyen d'une liaison wifi. Bien que sans fil, le système nécessite néanmoins une alimentation secteur.

2 Conception du lecteur de badge

La fonction de lecture du badge est confiée à un composant du commerce. Il est impératif de disposer d'un lecteur supportant le protocole Wiegand 26/34 , car le lecteur doit envoyer le tag du badge au contrôleur associé, et celui-ci doit pouvoir le lire. Je ne propose pas de modèle particulier, assurez vous cependant que le lecteur que vous avez choisi possède un voyant de contrôle, un bipper et que les badges sont compatibles avec le lecteur.

Afin de réaliser l'intégration du lecteur de badge et du contrôleur, j'ai choisi le composant ESP8266 et le code ESPEasy associé. Ce contrôleur étant livré vierge, il convient de charger le code ESPEasy dans sa mémoire flash. Pour ce faire, il existe de nombreux tutoriels qui présentent cette opération. Pour faciliter les choses, j'ai choisi une carte ESP8266 qui dispose d'un port usb, ce qui facilite le flashage de la carte, car on peut la brancher directement sur un PC sans composant additionnel.

En ce qui concerne l'interconnexion de l'ESP8266 et du lecteur de badge, compte tenu des écarts de tension entre les signaux , 5V pour le lecteur et 3,3V pour le contrôleur, il est impératif d'insérer un circuit type « Level Shifter » entre les deux, sous peine de griller les cartes. On en trouve de nombreux modèles disposant d'un nombre variable d'entrée/sortie (E/S). En ce qui me concerne, j'ai choisi un composant à 8 E/S, dont j'utilise 4 ports.

Concernant l'alimentation électrique des différents modules, il est nécessaire de fournir du 12v pour le lecteur, du 5v pour le contrôleur et le circuit « Level Shifter », et enfin du 3,3V pour le circuit « Level Shifter ». La tension de 3,3V est fournie par le régulateur interne de l'ESP8266. On a donc besoin uniquement de 5V et de 12V. Pour n'utiliser qu'un seul bloc d'alimentation secteur, on peut se procurer soit un élévateur de tension soit un abaisseur de tension. Si on choisit une alimentation de 12v on choisira un abaisseur de tension pour obtenir du 5V et inversement. Cette carte est équipée d'un potentiomètre qui permet de régler la tension en sortie, il faudra donc disposer d'un voltmètre pour cette étape de réglage.

Pour réaliser le câblage, je joins le schéma. Voir dernier chapitre.

On notera que le circuit Level Shifter que j'utilise, dispose d'une entrée OE qui permet de passer les E/S en mode tristate. Dans ce cas, il est nécessaire de connecter cette entrée soit à une sortie du contrôleur, soit directement à du 3,3V. J'ai choisi de le connecter à une sortie du contrôleur pour isoler le lecteur en cas d'échecs successifs de lecture de badge. Dans ce cas, plus aucune information n'est envoyée au contrôleur par le lecteur.

On notera qu'en plus des entrées D0 et D1 qui servent à communiquer le protocole Wiegand au contrôleur, j'ai également connecté une sortie supplémentaire qui permet d'envoyer une série de bip sonores à l'utilisateur pour confirmer une opération ou signaler un problème.

Par ailleurs, pour indiquer l'état de l'alarme, j'utilise une sortie du contrôleur qui est connectée à une

LED du lecteur RFID. Dans mon cas, quand la LED est verte, l'alarme est coupée, et quand elle est rouge, l'alarme est activée.

Une autre remarque, j'utilise comme équipement un clavier/lecteur RFID. J'avais l'idée d'utiliser le clavier du lecteur pour développer des fonctions supplémentaires, mais dans ce que je propose aujourd'hui dans ce document, la fonction clavier n'est pas utilisée du tout. Ce qui veut dire qu'un lecteur RFID sans clavier peut très bien faire l'affaire s'il possède les sorties Wiegand D0 et D1.

3 Configuration de l'ESP8266

Une fois l'opération de flashage terminée, il faut configurer l'ESP8266 au moyen de son interface web. Je mets ci-dessous une série de copies d'écran qui permettent de répliquer cette configuration.

On remarquera qu'en arrivant à la configuration de l'onglet « Devices », il faudra au préalable installer le Plugin ESPEasy dans Jeedom et activer le mode « Inclusion » pour voir apparaître automatiquement les commandes d'information. Normalement, la seule information correspondant au Tag du badge devrait être suffisantes, mais comme ça n'a pas été le cas pour moi. J'ai dû ajouter une information d'état d'une E/S pour activer automatiquement l'inclusion de l'information Tag. Je détaillerai cette étape de Jeedom dans le chapitre suivant.

Pour faciliter l'intégration du lecteur dans Jeedom, j'ai utilisé des Règles dans ESPEasy. Ces règles permettent de développer des fonctions simples qu'on appellent à partir de Jeedom. Dans ce but, j'ai créé des commandes « bip » qui permettent de générer des bips sonores sur le clavier RFID à partir de Jeedom.

Pour répliquer la configuration que j'ai utilisé, il suffit de recopier les informations figurant sur les copies d'écran ci-dessous.

- Remarque 1 : Dans l'onglet « Controllers », la valeur « Controller IP » dépend de votre installation.
- Remarque 2 : Dans l'onglet « Controllers », la valeur « Controller publish » est à copier/coller à partir de la configuration du plugin. Idem pour la valeur « Controller Port ».

3.1 Onglet Config

ESP Easy Mega: Badge

△Main ◉Config 🗨️Controllers 🚧Hardware 🛠️Devices ➔Rules 📧Notifications

Main Settings

Unit Name:

Unit Number:

Append Unit Number to hostname:

Admin Password:

Wifi Settings

SSID:

WPA Key:

Fallback SSID:

Fallback WPA Key:

WPA AP Mode Key:

Client IP filtering

Client IP block level:

Access IP lower range:

Access IP upper range:

IP Settings

ESP IP:

ESP GW:

ESP Subnetmask:

ESP DNS:

Note: Leave empty for DHCP

Sleep Mode

Sleep awake time: [sec] ?

Note: 0 = Sleep Disabled, else time awake from sleep

Sleep time: [sec (max: 13789)]


Sleep on connection failure:

3.2 Onglet Controllers

ESP Easy Mega: Badge

△Main ◉Config **Controllers** 📌Hardware 🖱️Devices ⇌Rules 📧Notifications 🛠️Tools

Controller Settings

Protocol: 

Locate Controller:

Controller IP:

Controller Port:

Controller Queue

Minimum Send Interval: [ms]

Max Queue Depth:

Max Retries:

Full Queue Action:

Check Reply:

Client Timeout: [ms]

Credentials

Use Extended Credentials:

Controller User:

Controller Password:

Controller Subscribe:

Controller Publish:

Enabled:

3.3 Onglet Hardware

ESP Easy Mega: Badge

[Main](#) [Config](#) [Controllers](#) **Hardware** [Devices](#) [Rules](#) [Notifications](#) [Tools](#)

Hardware Settings ?

Wifi Status LED

GPIO → LED:

Inversed LED:

Note: Use 'GPIO-2 (D4)' with 'Inversed' checked for onboard LED

Reset Pin

GPIO ← Switch:

Note: Press about 10s for factory reset

I2C Interface

GPIO ⇄ SDA:

GPIO → SCL:

Clock Speed: [Hz]

Note: Use 100 kHz for old I2C devices, 400 kHz is max for most.

SPI Interface

Init SPI:

Note: CLK=GPIO-14 (D5), MISO=GPIO-12 (D6), MOSI=GPIO-13 (D7)

Note: Chip Select (CS) config must be done in the plugin

GPIO boot states

Pin mode GPIO-0 (D3) ⚠:	<input type="text" value="Default"/>
Pin mode GPIO-1 (D10) TX0:	<input type="text" value="Default"/>
Pin mode GPIO-2 (D4) ⚠:	<input type="text" value="Default"/>
Pin mode GPIO-3 (D9) RX0:	<input type="text" value="Default"/>
Pin mode GPIO-4 (D2):	<input type="text" value="Output Low"/>
Pin mode GPIO-5 (D1):	<input type="text" value="Output Low"/>
Pin mode GPIO-9 (D11) ⚠:	<input type="text" value="Default"/>
Pin mode GPIO-10 (D12) ⚠:	<input type="text" value="Default"/>
Pin mode GPIO-12 (D6):	<input type="text" value="Input"/>
Pin mode GPIO-13 (D7):	<input type="text" value="Input"/>
Pin mode GPIO-14 (D5):	<input type="text" value="Output High"/>
Pin mode GPIO-15 (D8) ⚠:	<input type="text" value="Default"/>

3.4 Onglet Devices

ESP Easy Mega: Badge

◀Main ⚙️Config 🗨️Controllers 🔧Hardware 🖱️Devices ➡️Rules 📧Notifications 🛠️Tools

Task	Enabled	Device	Name	Port	Ctrl (IDX)	GPIO	
Edit	1	✓	RFID - Wiegand	RFID	1 (1)	GPIO-13 GPIO-12	Tag:
Edit	2	✓	Switch input - Switch	Dummy	1 (1)	GPIO-5	State:
Add	3						
Add	4						
Add	5						
Add	6						
Add	7						
Add	8						
Add	9						

ESP Easy Mega: Badge

◀Main ⚙️Config 🗨️Controllers 🔧Hardware 🖱️Devices ➡️Rules 📧Notifications 🛠️Tools

Task Settings

Device: RFID - Wiegand ? i

Name:

Enabled:

Sensor

GPIO ← D0 (Green, 5V): ▼

GPIO ← D1 (White, 5V): ▼

Wiegand Type: ▼

Data Acquisition

Send to Controller

1

IDX: ▲ ▼

Values

#

1

[Close](#) [Submit](#) [Delete](#)

Powered by [Let's Control It](#) community

C'est à cette étape que l'on doit intervenir au niveau de Jeedom pour obtenir l'inclusion automatique des commandes « Information ». Lire le chapitre 4.1 à ce sujet.

3.5 Onglet Tools/Advanced

The screenshot shows the 'Tools' section of the 'ESP Easy Mega: Badge' interface. At the top, there is a navigation bar with links for Main, Config, Controllers, Hardware, Devices, and Rules. Below this is a 'Tools' header. A 'Command' section contains a text input field and a 'Submit' button with a help icon. The 'System' section lists several actions:

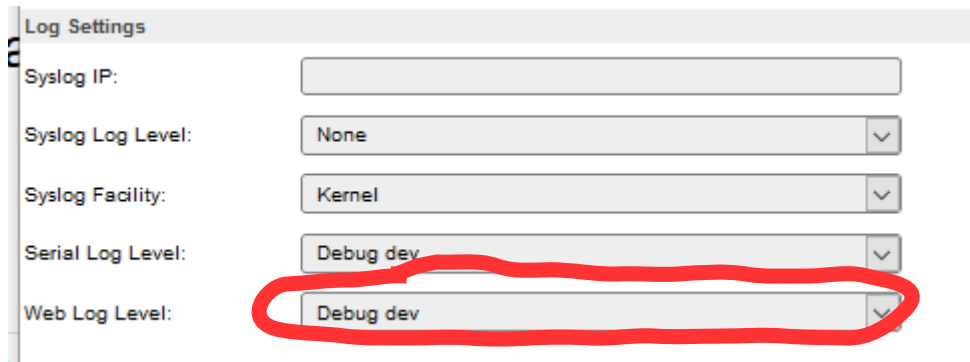
- Reboot: Reboots ESP
- Log: Open log output
- Info: Open system info page
- Advanced**: Open advanced settings (circled in red)
- Show JSON: Open JSON output
- Timing stats: Open timing statistics of system
- Pin state buffer: Show Pin state buffer
- System Variables: Show all system variables and conversions

3.6 Onglet Tools/Log

The screenshot shows the 'Tools' section of the 'ESP Easy Mega: Badge' interface. At the top, there is a navigation bar with links for Main, Config, Controllers, Hardware, Devices, and Rules. Below this is a 'Tools' header. A 'Command' section contains a text input field and a 'Submit' button with a help icon. The 'System' section lists several actions:

- Reboot: Reboots ESP
- Log**: Open log output (circled in red)
- Info: Open system info page
- Advanced: Open advanced settings
- Show JSON: Open JSON output
- Timing stats: Open timing statistics of system
- Pin state buffer: Show Pin state buffer
- System Variables: Show all system variables and conversions

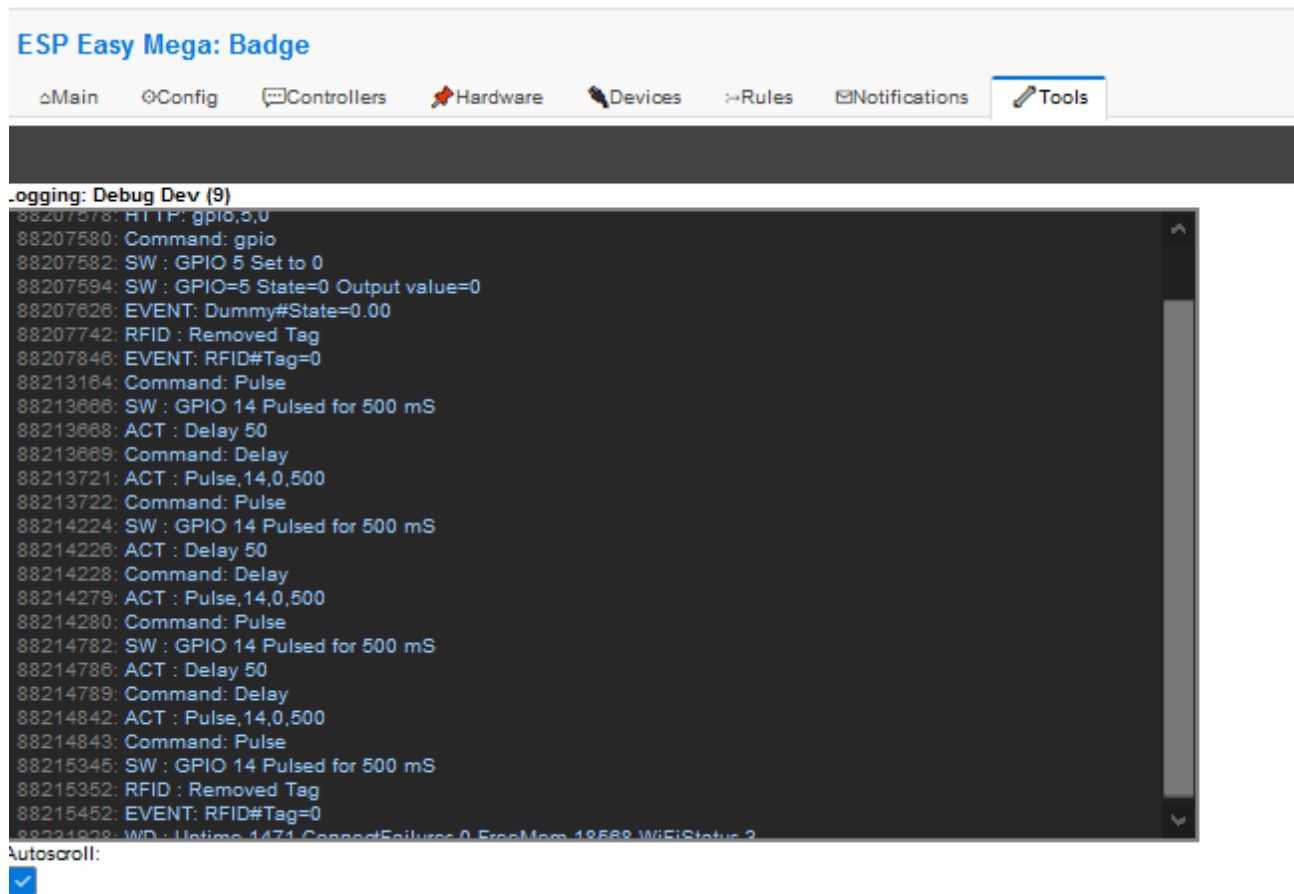
La fenêtre Log affiche des informations dont la granularité peut être définie dans l'onglet Tools/Advanced/Log settings



The screenshot shows the 'Log Settings' configuration page. It contains several dropdown menus for setting log levels and facilities. The 'Web Log Level' dropdown is highlighted with a red circle and is set to 'Debug dev'.

Syslog IP:	<input type="text"/>
Syslog Log Level:	None
Syslog Facility:	Kernel
Serial Log Level:	Debug dev
Web Log Level:	Debug dev

Fenêtre de log :



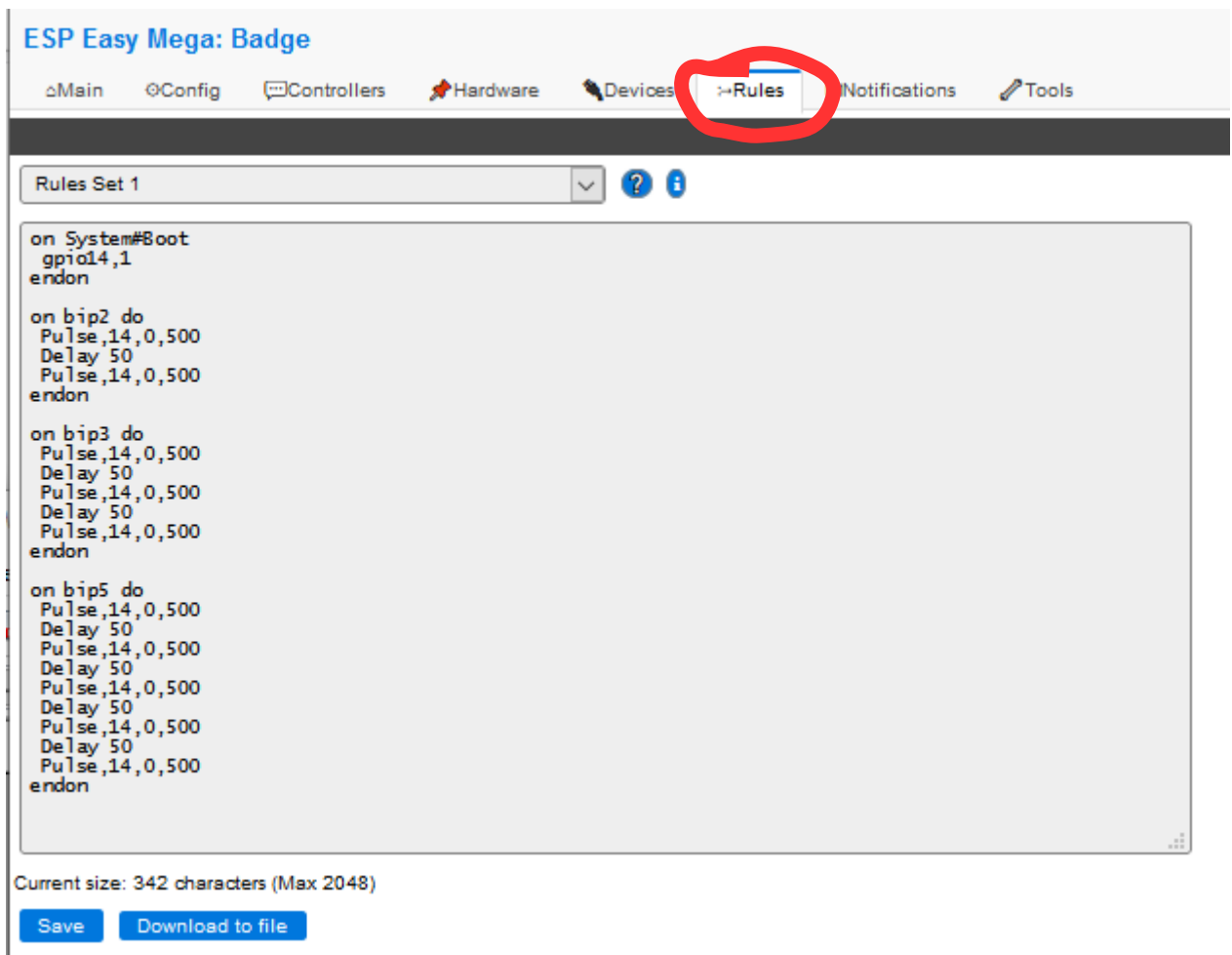
The screenshot shows the 'ESP Easy Mega: Badge' interface with the 'Tools' tab selected. The log window displays a list of log entries under the heading 'Logging: Debug Dev (9)'. The log entries include various system events and commands, such as GPIO operations, RFID tag removal, and system status reports. An 'Autoscroll' checkbox is checked at the bottom left of the log window.

```
Logging: Debug Dev (9)
88207578: HTTP: gpio,5,0
88207580: Command: gpio
88207582: SW : GPIO 5 Set to 0
88207594: SW : GPIO=5 State=0 Output value=0
88207626: EVENT: Dummy#State=0.00
88207742: RFID : Removed Tag
88207846: EVENT: RFID#Tag=0
88213164: Command: Pulse
88213666: SW : GPIO 14 Pulsed for 500 mS
88213668: ACT : Delay 50
88213669: Command: Delay
88213721: ACT : Pulse,14,0,500
88213722: Command: Pulse
88214224: SW : GPIO 14 Pulsed for 500 mS
88214226: ACT : Delay 50
88214228: Command: Delay
88214279: ACT : Pulse,14,0,500
88214280: Command: Pulse
88214782: SW : GPIO 14 Pulsed for 500 mS
88214786: ACT : Delay 50
88214789: Command: Delay
88214842: ACT : Pulse,14,0,500
88214843: Command: Pulse
88215345: SW : GPIO 14 Pulsed for 500 mS
88215352: RFID : Removed Tag
88215452: EVENT: RFID#Tag=0
88221998: WD : Uptime: 1471 ConnectedFailures:0 FreeMem: 18568 WiFiStatus: 2
```

Autoscroll:

3.7 Utilisation des règles

Pour activer l'onglet Règles/Rules il faut cocher la case Rules dans Tools/Advanced, et rafraîchir l'affichage pour voir apparaître le nouvel onglet Rules dans la fenêtre.



Je mets une copie du code ci-dessous pour exemple :

```
on System#Boot
  gpio14,1
endon
```

```
on bip2 do
  Pulse,14,0,500
  Delay 50
  Pulse,14,0,500
endon
```

```
on bip3 do
  Pulse,14,0,500
  Delay 50
  Pulse,14,0,500
  Delay 50
  Pulse,14,0,500
endon
```

```
on bip5 do
  Pulse,14,0,500
  Delay 50
  Pulse,14,0,500
  Delay 50
  Pulse,14,0,500
  Delay 50
  Pulse,14,0,500
  Delay 50
  Pulse,14,0,500
endon
```

Remarque :

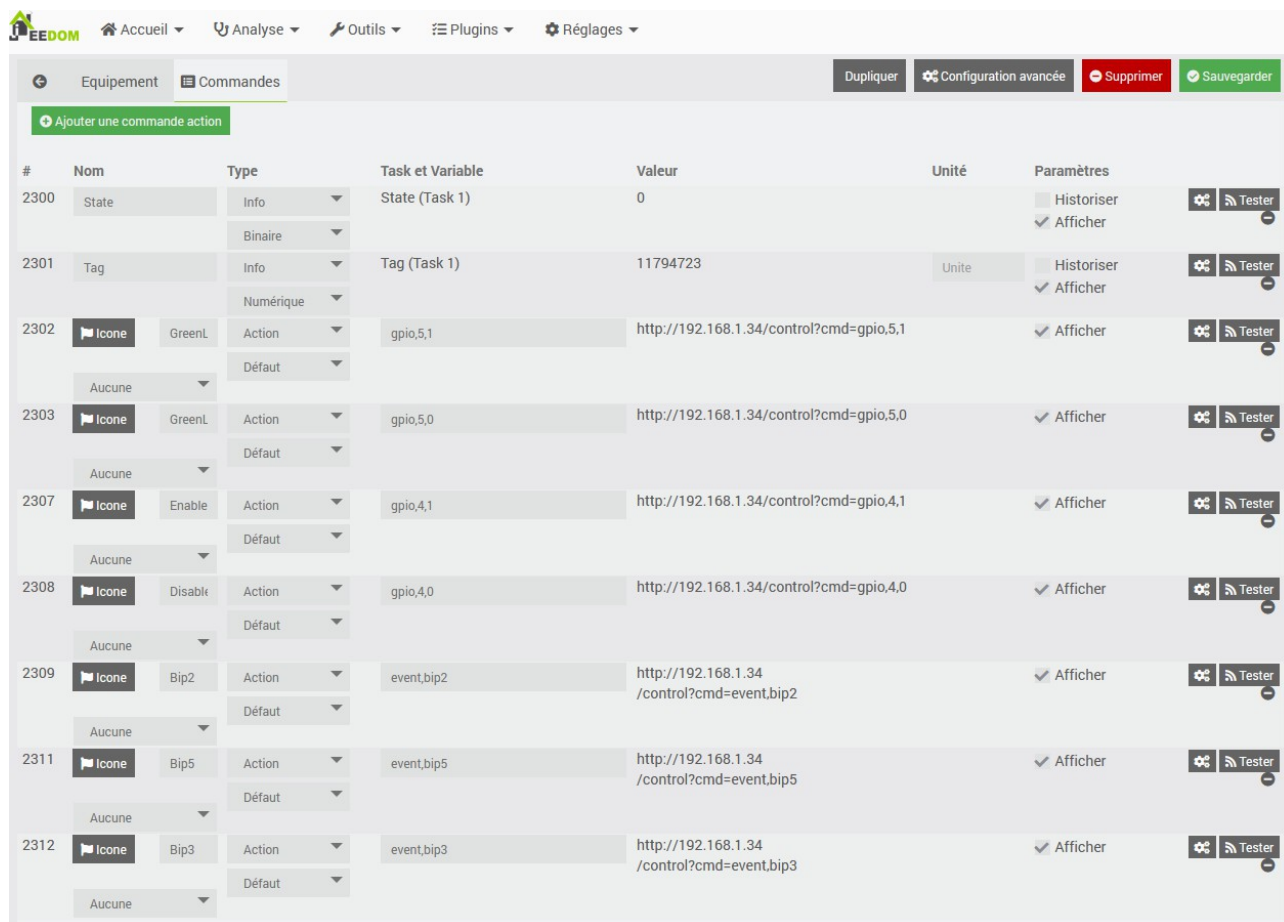
- La première fonction sert à forcer la sortie gpio14 à l'état 1, car le bipper du clavier est actif sur un front descendant.
- Les autres fonctions servent à générer une série de bips dans un scénario par exemple.

4 Configuration du Plugin dans Jeedom

Remarque : Une fois le plugin ESPEasy Installé et activé, il faut sélectionner le mode « Inclusion », et l'équipement sera créé automatiquement.

4.1 Configuration du Plugin ESPEasy

Je joins une copie d'écran pour présenter la configuration dans Jeedom.



#	Nom	Type	Task et Variable	Valeur	Unité	Paramètres	
2300	State	Info	State (Task 1)	0		<input type="checkbox"/> Historiser <input checked="" type="checkbox"/> Afficher	
2301	Tag	Info	Tag (Task 1)	11794723	Unite	<input type="checkbox"/> Historiser <input checked="" type="checkbox"/> Afficher	
2302	Icone	GreenL	Action	gpio,5,1		<input checked="" type="checkbox"/> Afficher	
2303	Icone	GreenL	Action	gpio,5,0		<input checked="" type="checkbox"/> Afficher	
2307	Icone	Enable	Action	gpio,4,1		<input checked="" type="checkbox"/> Afficher	
2308	Icone	Disabl	Action	gpio,4,0		<input checked="" type="checkbox"/> Afficher	
2309	Icone	Bip2	Action	event_bip2		<input checked="" type="checkbox"/> Afficher	
2311	Icone	Bip5	Action	event_bip5		<input checked="" type="checkbox"/> Afficher	
2312	Icone	Bip3	Action	event_bip3		<input checked="" type="checkbox"/> Afficher	

Remarque:

- La première commande « State », ne sert à rien d'autre qu'à faire fonctionner l'inclusion automatique dans Jeedom. Une fois le plugin en mode Inclusion, la sauvegarde des «Devices» coté configuration ESPEasy génère automatiquement les deux premières commandes. Les autres commandes (7) sont créées à la main par le bouton « Ajouter une commande action ».
- Les commandes Enable et Disable servent à commander l'entrée OE du circuit Level Shifter. Je l'utilise dans un scénario pour isoler le lecteur de badge du contrôleur après 3 tentatives de lecture infructueuse du badge.
- Les commandes GreenLedOn et GreenLedOff permettent de contrôler le voyant LED du clavier pour signaler à l'utilisateur l'état marche ou arrêt de l'alarme.
- Les commandes Bip génèrent des bips sonores sur le clavier.

4.2 Scenario Jeedom

Afin de profiter des nouvelles commandes du lecteur de badge, il faudra développer des scénarios dans Jeedom. L'information clé pour utiliser le badger est la valeur du Tag. Cette valeur qui remonte dans Jeedom par l'information « Tag » du plugin, permet par comparaison d'autoriser ou non l'activation/désactivation de l'alarme.

On pourra contrôler la LED du lecteur pour acquitter la commande marche/arrêt de l'alarme et également jouer sur les commandes « Bip » pour confirmer la prise en compte ou le rejet d'une action. J'utilise également les bips sonores pour signaler l'état de l'alarme, même quand j'utilise un autre moyen que le lecteur de badge pour activer/désactiver l'alarme. C'est rassurant d'entendre que l'alarme a bien pris en compte une commande.

Je ne mets pas d'exemple de scénario ici, je laisse libre cours à votre imagination.

On pourra également intégrer à la solution, le plugin « Badger » pour faciliter la gestion des badges par exemple.

5 Schéma d'interconnexion des modules

