DOCUMENTATION TECHNIQUE

INSTALLATION ET CONFIGURATION DE STREAMBOX

STREAMLOR

NETLOR

VERSION
1.0

Date de dernière mise à jour : lundi 7 juin 2021

NETLOR

Table des matières

1 - Historique du document

Date	Auteur	Version	Sujet de la modification
27/05/2021	Alexandre REMIATTE	0.1	Création de la documentation
03/06/2021	Alexandre REMIATTE	1.0	Mise à jour sur la description et l'installation
07/06/2021	Alexandre REMIATTE	1.1	Changement de configuration + indications schéma électrique

2 - Description de StreamBox

2.1 - Présentation

StreamBox est un programme s'intégrant dans la solution **Streamlor** de **NETLOR**. Grâce à des cartes électroniques ainsi que des lecteurs RFID et/ou Code-bar **StreamBox** est capable de piloter plusieurs tourniquets afin de réaliser un contrôle d'accès. Ce programme a besoin d'être intégré à un système d'informations de la solution **Streamlor** afin de pouvoir recevoir les autorisations de passages des utilisateurs. Les cartes électroniques permettent de déclencher des passages sur un tourniquet ainsi que de détecter un passage d'un utilisateur.

2.2 - Fonctionnalités

StreamBox est capable de :

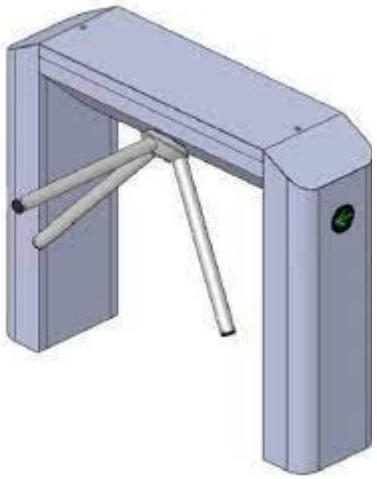
- Gérer un ou plusieurs « **Turnstiles** »
- Gérer électroniquement un tourniquet via une carte électronique
- Détecter un passage sur un tourniquet et informer le système d'information via **MQTT**
- Écouter les ordres du programme **Warden** via **MQTT** afin d'ouvrir un passage pour les utilisateurs
- Indiquer l'état des différents « **Turnstiles** » afin de permettre le monitoring du système d'informations
- Gérer plusieurs lecteurs RFID et/ou Code-bar grâce au programme **Fregate** de **Streamlor**
- Gérer plusieurs cartes électroniques de type **MiiPy** et **Yoctopuce**
- Rétablir une ou plusieurs connexions perdus au cours de son exécution
- Garder un Uptime élevé
- [à venir] Gérer des rubans de leds afin d'indiquer une information aux utilisateurs

Note importante :
La fonctionnalité qui oblige **StreamBox** de rétablir les connexions perdues avec les lecteurs RFID et Code-bar ou avec les cartes électroniques ne peut pas être totalement respecté. En fonction de la configuration (plus ou moins de lecteurs, type de connexion, docker) cela va avoir un impacte sur la capacité du programme à redémarrer une nouvelle connexion. **StreamBox** va toujours essayer de rétablir la situation, mais dans certains cas un redémarrage du conteneur docker ou de la machine hôte est obligatoire.
En revanche, **StreamBox** peut facilement détecter une perde de connexion et donc signaler les « **Turnstiles** » OFFLINE.

2.3 - Tourniquets

StreamBox prend en charge des tourniquets de marque et de modèles différents. Comme le pilotage est réalisé via des cartes électroniques, il est compatible avec beaucoup de modèles.
Il a été testé sur les modèles suivants :

- Tourniquets de la marque **Klein** (plusieurs modèles compatibles):



- Tourniquets de la marque **Perco** (plusieurs modèles compatibles) :

2.4 - Lecteurs RFID et Code-bar

La gestion des types de lecteurs RFID ou Code-bar est déporté et ne se trouve pas sur **StreamBox**. Le programme est donc compatible avec tous les lecteurs pris en charge par le programme **Fregate** de la solution **Streamlor**.

StreamBox a été testé avec les lecteurs suivant :

- Lecteur RFID PN532 NFC
- Lecteurs RFID ACS (ACR122 NFC et ACS1281)
- Lecteurs RFID IDCapt
- Lecteur de Code-bar GFS4400

À l'avenir, **StreamBox** sera capable de prendre en charge la fonctionnalité ISO-DEP présent dans **Fregate** qui permet de communiquer via NFC afin d'utiliser le Mobile Pass de **Streamlor**.

2.5 - Cartes électroniques

StreamBox doit être connecté à des cartes électroniques afin de pouvoir piloter les tourniquets. Les cartes **MiiPy** et **Yoctopuce** sont actuellement prises en charge par le programme. Le choix du type de carte est à définir en fonction du coût, de l'environnement existant, des éléments à connecter sur la carte et du type de tourniquet.

Rapidement, voici les détails de chaque carte électronique :

- **MiiPy**
  - Serveur web intégré pour du monitoring et de l'administration
  - 8 relais physiques disponibles (utilisation pour déclencher un passage sur un « **turnstile** » ou gestion de ruban de leds)
  - 8 entrées digitales tout ou rien (utilisation pour détecter un passage sur un « **turnstile** »
  - 4 entrées analogiques (utilisation pour une sonde de température par exemple)
  - Interface RJ45
  - Alimentation Jack 12v
- **Yoctopuce** (modèle Yocto-Maxi-IO-V2)
  - Connexion USB vers un périphérique hôte qui héberge **StreamBox**
  - 8 entrées/sorties digitales électriquement isolées du bus USB (Attention ceci n'est pas des relais physiques mais peuvent être utilisées pour déclencher ou détecter un passage sur un tourniquet)
  - Alimentation des entrées/sorties en 3V ou 5V par la carte ou via une alimentation externe
  - Alimentation de la carte via le port USB

3 - Installation et configuration

3.1 - Environnement d'installation

StreamBox est un projet qui utilise le framework **Dims2**, son déploiement et installation est semblable aux autres projets de **Dims2**. En revanche son lancement avec docker est un peu particulier car il a besoin de communiquer avec les lecteurs RFID et Code-bar ainsi que les cartes électroniques. Vous devez lancer l'image docker avec le mode privilège d'activé (pour le moment, ce paramètre est obligatoire pour le bon fonctionnement de **StreamBox**) ainsi que l'accès aux fichiers « devices » de Linux qui permettent d'établir la connexion avec les lecteurs et les cartes. Si vous utilisez une carte électronique de type **MiiPy** il faut également transmettre le port pour le serveur Web de détection des passages. Pour une installation stable, il est recommandé d'utiliser la solution Udev de Linux pour affecter correctement les périphériques USB.

3.2 - Configuration générale

StreamBox possède un fichier de configuration JSON issus de **Dims2**. Cette configuration permet de définir la liste des **turnstiles** gérés par le programme ainsi que la liste des cartes électroniques qui contrôlent les tourniquets. Un **turnstile** représente un sens de passage qui peut être rattaché à un lecteur RFID ou Code-bar (pour que le programme s'adapte au maximum à n'importe quel environnement il est capable de prendre en charge plusieurs lecteurs par turnstile. Cette situation n'est pas fréquente, en général, on défini un **turnstile** par lecteur). Détails de la configuration :

Un attribut « **turnstiles** » (tableau) doit contenir des objets qui doivent avoir ces attributs :

- "**streambox_name**" (string): Nom de la box (concentrateur) qui contient ce **turnstile**
- "**turnstile_id**" (string) : L'identifiant du **turnstile** (en général, une numérotation)
- "**turnstile_readers**" (tableau) : Contient des objets représentant les lecteurs :
  - "**reader_model**" (string) : Modèle de lecteurs disponible sur Fregate :
    - « ACR122 » pour un lecteur RFID ACS ACR122
    - « ACR1281 » pour un lecteur RFID ACS ACR1281
    - « PN532 » pour un lecteur RFID PN532
    - « IDCAPT » pour un lecteur RFID IDCapt
    - « GFS4400 » pour un lecteur Code-bar GFS4400
  - "**reader_port**" (string) : Le port que le programme va utiliser pour communiquer avec le lecteur. (par exemple « /dev/ttyUSB »)
  - "**reader_port_number**" (string) : Ce champ est obligatoire uniquement pour les lecteurs ACS, il permet d'indiquer le numéro du lecteur (certains lecteurs ont plusieurs lecteurs intégrés)
- "**turnstile_electronic_card_id**" (string) : L'identifiant de la carte électronique qui pilote ce **turnstile**
- "**turnstile_electronic_card_relay**" (int) : Le numéro du relai à utiliser pour déclencher un passage
- "**turnstile_electronic_card_passing**" (int) : Le numéro de la sortie sur la carte qui doit être écoutée pour détecter un passage (confirmation de passage)

La configuration doit également avoir un attribut «**electronic_cards**» (tableau représentant la liste des cartes électroniques) qui doit contenir des objets qui doivent avoir ces attributs :

- "**enable_electronic_card**" (boolean) : « true » ou « false » pour activer ou désactiver la carte
- "**electronic_card_id**" (string) : L'identifiant de la carte électronique
- "**electronic_card_type**" (string) : Le type de la carte (« **MIIPY** » ou « **YOCTOPUCE** »)
- "**electronic_card_relays**" (tableau d'entier) : La liste des numéros des relais disponibles sur la carte
- "**electronic_card_passings**" (tableau d'entier) : La liste des numéros des entrées sur la carte disponible pour détecter les passages.
- Pour une carte **MiiPy** il faut rajouter en plus les attributs :
  - "**ip_address_miipy**" (string) : L'adresse réseau de la carte
  - "**key_m2m_miipy**" (string) : Le mot de passe pour la connexion m2m
  - "**port_m2m_miipy**" (int) : Le port pour la connexion m2m

La configuration a besoin de quelques attributs supplémentaires pour le bon fonctionnement de l'application :

- "**mqtt_host**" (string): L'adresse réseau et le port du broker **MQTT** ("tcp://IP:PORT")
- "**mqtt_client_id**" (string): L'identifiant du client **MQTT**
- "**mqtt_user**" (string): L'utilisateur du client **MQTT**
- "**mqtt_pass**" (string) : Le mot de passe du client **MQTT**
- Pour une carte **MiiPy** il faut rajouter en plus les attributs :
  - "**listen_port**" (int): Port du serveur web pour la détection de passage
  - "**listen_addr**" (string): Adresse réseau du serveur web pour la détection de passage

Un exemple du fichier de configuration est disponible dans le dépôt du projet.

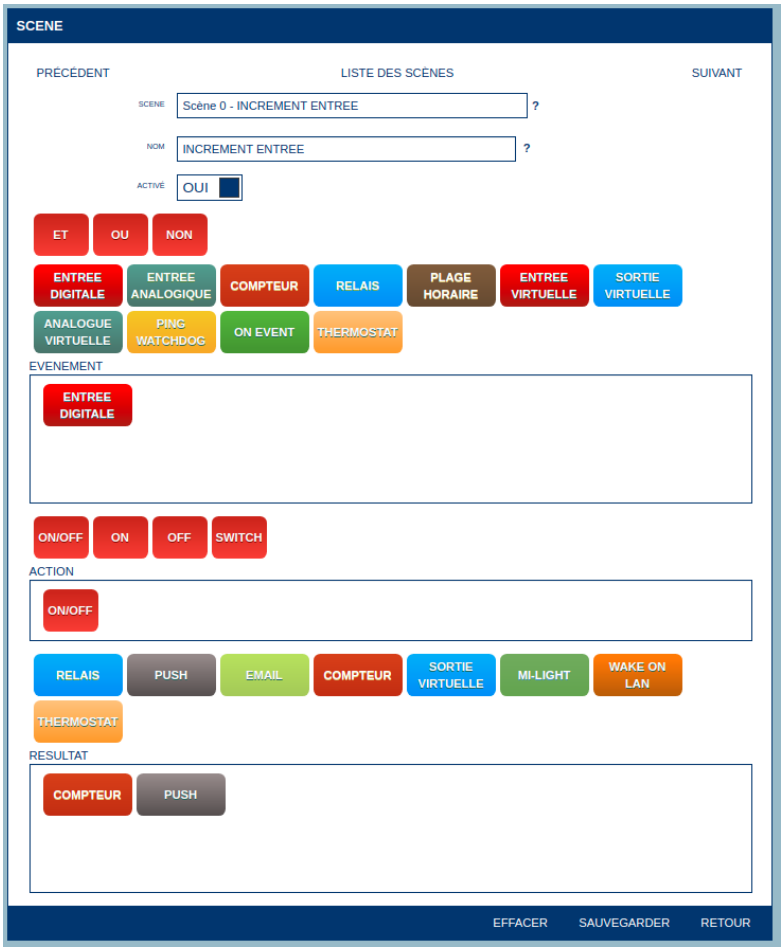
3.3 - Configuration des cartes électroniques

3.3.1 - Carte MiiPy

La configuration de la carte **MiiPy** se fait depuis son interface Web, vous devez donc lui affecter une adresse réseau et vous connecter sur son interface via un navigateur Web. Vous pouvez utiliser la documentation officielle de la carte **MiiPy** pour réaliser la configuration basique adaptée à votre environnement.

Il faut activer la connexion m2m afin que le programme **StreamBox** puisse se connecter à distance au **MiiPy**. Pour cela, allez dans les configurations administrateur dans l'onglet API puis configurer la clé API ainsi que le port.

Pour prendre en charge une détection de passage il faut configurer des scénarios (également appelés scènes). Voici un exemple de configuration :



Vous pouvez donner un nom à la scène et sélectionner une entrée digitale comme événement qui va déclencher le scénario. Il faut bien sûr que la scène soit activée.

Dans la configuration de l'entrée digitale, vous devez indiquer le numéro de l'entrée à utiliser.

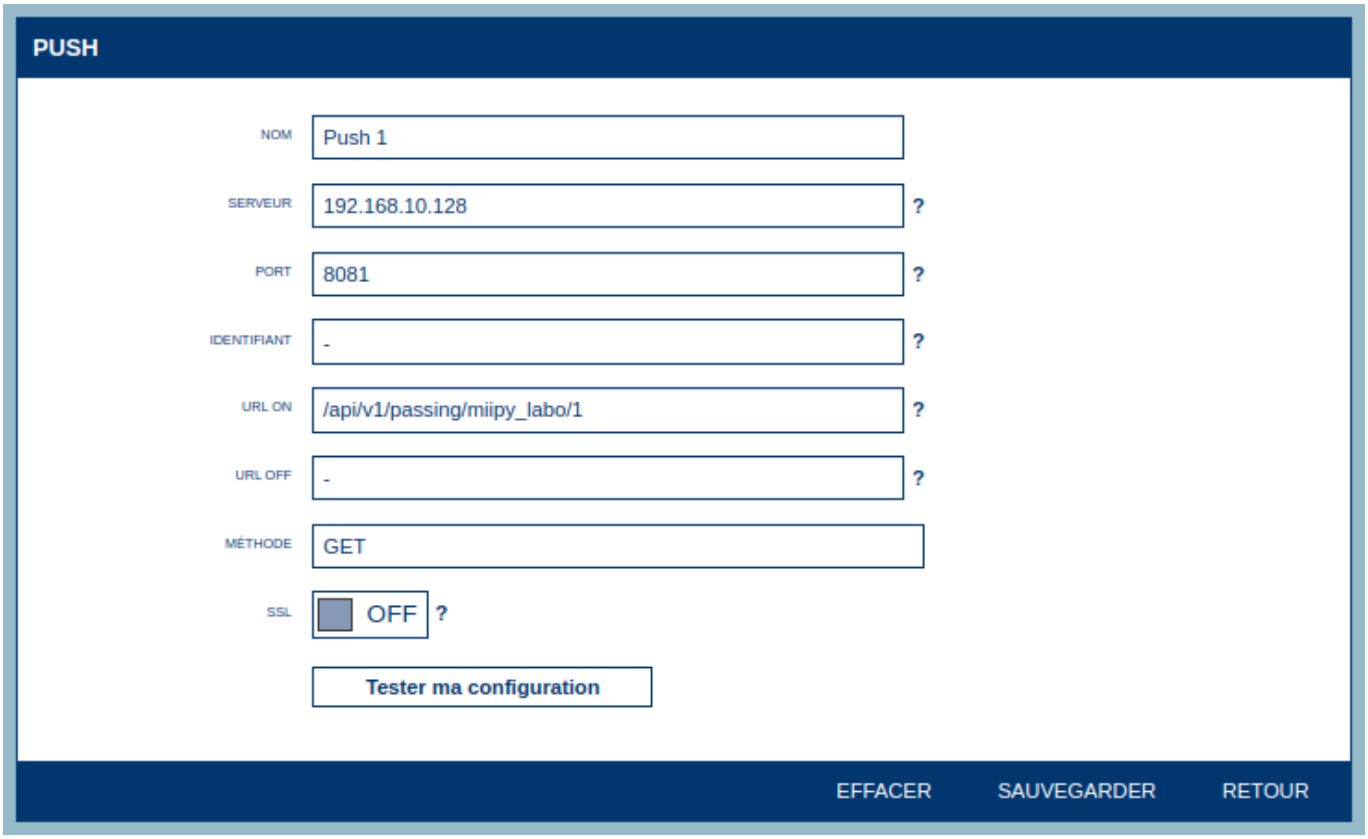
Dans cet exemple, un compteur est incrémenté à chaque détection de passage. Cette action n'est pas obligatoire car elle n'est pas utilisée par **StreamBox** mais elle permet d'afficher des compteurs d'entrées et de sortie sur le tableau de bord de l'interface Web du **MiiPy**.

Pour que le programme **StreamBox** puisse détecter un passage il faut rajouter une action « push » dans le résultat du scénario. Dans la configuration de l'action vous devez sélectionner le numéro du « push » qui va être utilisé par la scène.

Pour configurer le push vous devez aller dans les périphériques puis dans « push ».

Il faut sauvegarder la scène à la fin de la configuration.

Exemple de configuration d'un « push » :



Il faut indiquer l'IP réseau du serveur où est installé **StreamBox** ainsi que le port du serveur WEB qui va permettre à **StreamBox** de détecter un passage. Il faut également passer une URI qui respecte ce format :

« /*api*/v1/passing/**NOM_DE_LA_CARTE/ID_DU_PASSAGE** »

Le nom de la carte et l'identifiant du passage corresponds aux champs qu'il faut donner dans la configuration de **StreamBox**.

Vous devez également configurer un délai sur chaque relais utilisé par **StreamBox**. Ce délai représente le temps avant que le relai repasse à 0 une fois son activation déclenchée. Cette configuration se fait dans les réglages administrateurs, périphériques puis sorties relais. **Ce délai ne représente pas le temps où l'utilisateur peut réaliser son passage !** Il faut mettre un certain délai pour que le tourniquet déclenche un passage mais inutile de laisser un délai trop grand. Une valeur entre 400ms et 800ms semble convenir pour l'ensemble des modèles de tourniquet (la valeur que vous devez indiquer dans l'interface doit être en centième de millisecondes).

D'autres réglages sont possibles sur la carte **MiiPy**, mais ils ne sont plus obligatoires pour le bon fonctionnement de **StreamBox**. Vous pouvez configurer un tableau de bord pour afficher des informations supplémentaires et utiles pour le monitoring du tourniquet ou rajouter une sonde de température par exemple.

3.3.2 - Carte Yoctopuce

La carte **Yoctopuce** est très simple à configurer. Vous avez besoin de connecter la carte à un périphérique de développement où se trouve le dépôt du programme **StreamBox** et de lancer le programme de configuration automatique. Ce programme se trouve dans le package ***/backend/cmd/automaticConfigurationYoctopuceCards.go***

Pour lancer le programme de configuration il faut que la variable d'environnement ***LD_LIBRARY_PATH*** *soit définie et cible un dossier où se trouve la librairie libyocto.so (voir plus de détails dans le projet* ***Goctopuce*** *de* ***Streamlor****). Vous devez également lancer le programme en tant que super-utilisateur.*

Voici un exemple de lancement du programme :

```
$ sudo LD_LIBRARY_PATH=../goctopuce go run backend/cmd/automaticConfigurationYoctopuceCards.go
```

Le programme va vous guider en Anglais, la première étape va être la détection et la connexion à la carte **Yoctopuce** puis vous allez pouvoir configurer la carte pour une installation sur un tourniquet. Avec ce programme vous pourrez :

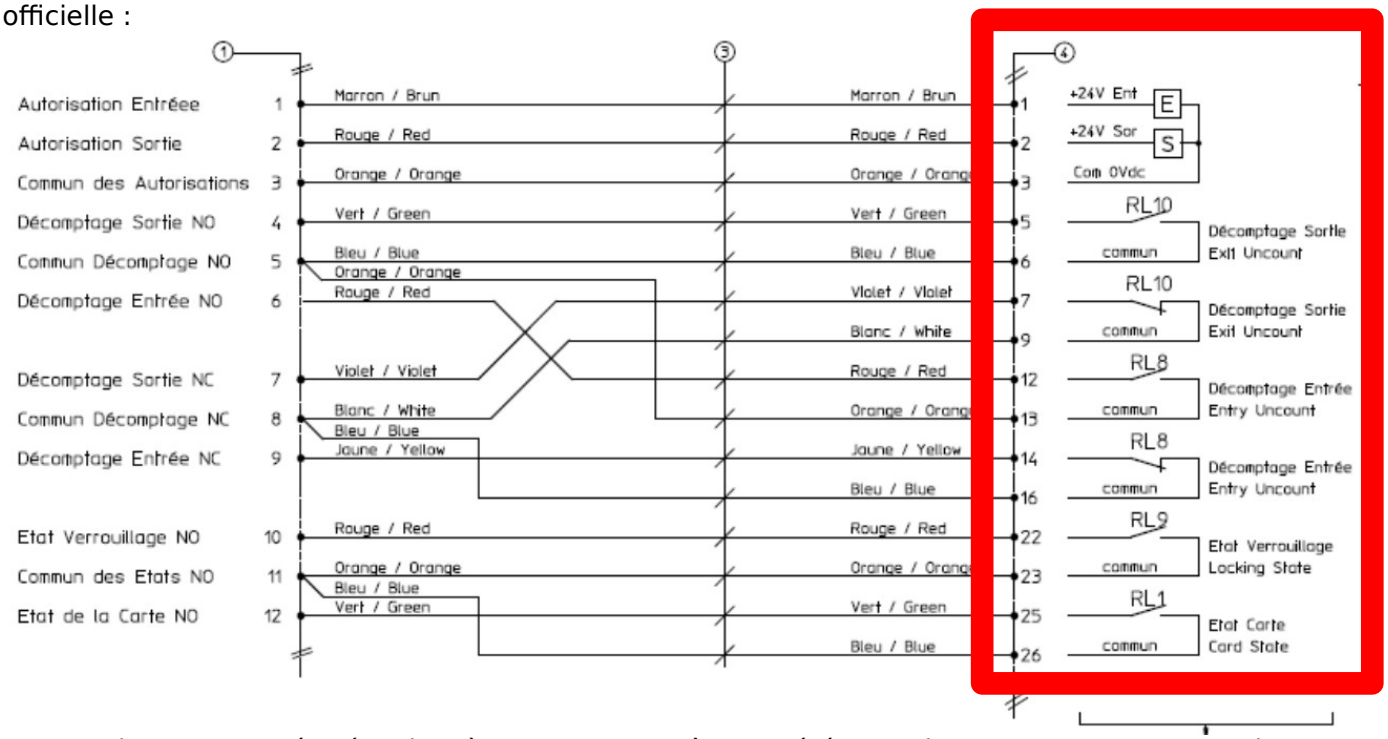
- Voir la configuration actuelle de la carte
- Changer le nom logique de la carte (obligatoire pour **StreamBox**)
- Configurer une détection de passage sur une entrée pour un tourniquet Klein ou Perco
- Configuration une activation de passage sur une sortie pour un tourniquet Klein ou Perco

Une fois la configuration appliquée vous pouvez quitter le programme en sauvegardant la configuration sur la mémoire de la carte Yoctopuce afin que cette configuration ne soit pas perdue quand la carte n'est plus alimentée. Cette étape de sauvegarde est limitée à 100 000 exécutions.

3.4 - Configuration des tourniquets

3.4.1 - Configuration d'un tourniquet Klein

Sur la carte électronique du tourniquet, vous pouvez trouver un connecteur 26 pins qui doit être utiliser pour piloter le tourniquet. Voici le schéma de ce connecteur issus de la documentation officielle :



Les pins sont numérotées de 1 à 26 et peuvent être repérées sur le connecteur comme cela :
(Le schéma représente le connecteur **mâle** présent sur la carte électronique du tourniquet)

26	25
24	23
22	21
20	19
18	17
16	15
14	13
12	11
10	9
8	7
6	5
4	3
2	1

On retrouve sur ce connecteur :

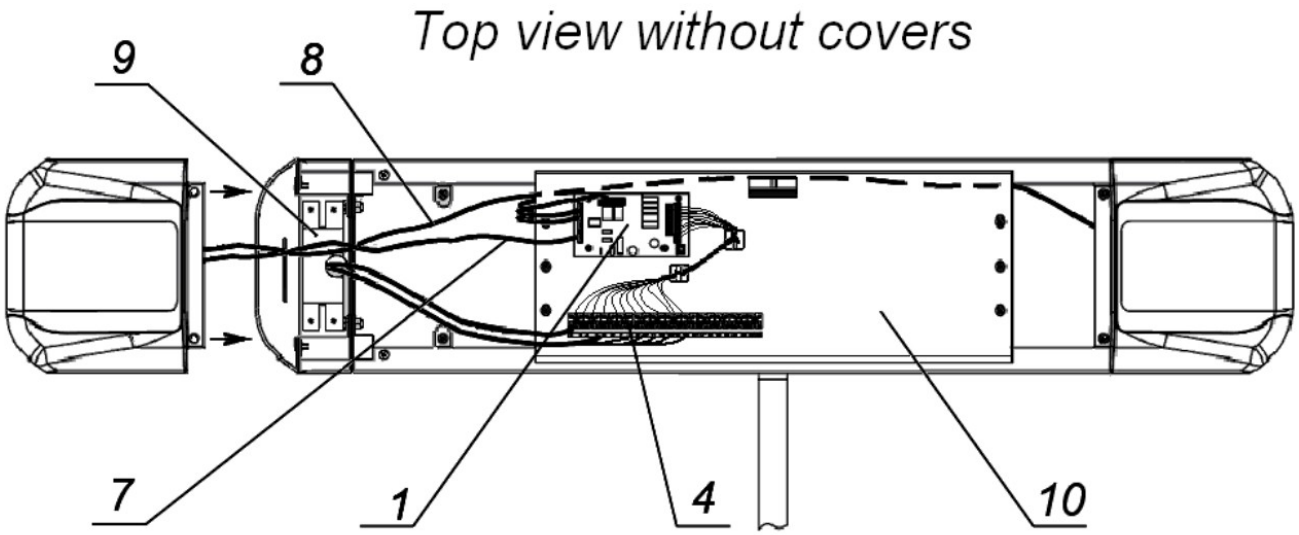
- Des contacteurs pour **détecter** un passage en **sortie**
  - *Un contacteur pour un état **normalement ouvert (5,6)***
  - *Un contacteur pour un état **normalement fermé (7,9)***
- Des contacteurs pour **détecter** un passage en **entrée**
  - Un contacteur pour un état **normalement ouvert (12,13)**
  - Un contacteur pour un état **normalement fermé (14,16)**
- Un contacteur pour **déclencher** un passage en **sortie (2,3)**
- Un contacteur pour **déclencher** un passage en **entrée (1,3)**
- Un contacteur pour **détecter** si le **bras est verrouillé (22,23)**
- Un contact pour **détecter** l'état de la **carte électronique (25,26)**

Il faut relier ces pins aux connecteurs de la carte électronique choisie (**MiiPy** ou **Yoctopuce**).

Pour plus d'informations, vous pouvez utiliser la documentation officielle du tourniquet **Klein**.

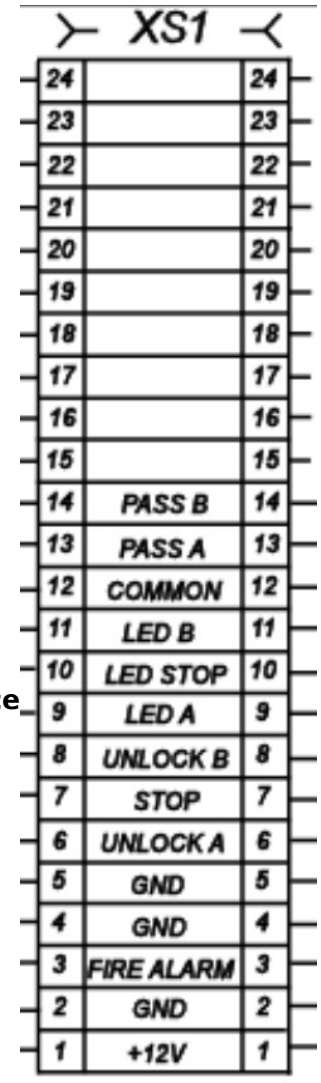
3.4.2 - Configuration d’un tourniquet Perco

En ouvrant le tourniquet, vous pouvez retrouver une organisation semblable à ce schéma :



- 1 : Carte électronique de contrôle
- 4 : Connecteur rapide XS1
- 7 et 8 : Câble pour les leds
- 9 : Emplacement pour le lecteur RFID
- 10 : Zone libre pour installation supplémentaire

Voici le schéma plus détaillé du connecteur XS1 :



- Un contacteur pour **détecter** un passage en **sortie (14,12)**
- Un contacteur pour **détecter** un passage en **entrée (13,12)**
- Un contacteur pour **déclencher** un passage en **sortie (8,GND)**
- Un contacteur pour **déclencher** un passage en **entrée (6,GND)**

Les contacteurs pour détecter un passage sont normalement fermés.

Il faut connecter les pins souhaitées à la carte électronique **MiiPy** ou **Yoctopuce**.

Pour plus d’informations, vous pouvez utiliser la documentation officielle du tourniquet **Perco**.