Rapport Pom Secure

Ambrosino, Busac, Chaput, Vergne, Yang

June 19, 2019

1 Contexte

Dans le cadre du projet de Smart Environment, nous avons concu et implémenté un détecteur de présence intelligent. Il vise à détecter les intrusions dans un appartement ou un bureau. Le produit est composé d'un module de détection de mouvement (à placer à l'entrée ou dans la pièce à protéger), ainsi que d'une interface Web.

2 Architecture logicielle

Pom secure est composé des modules suivants :

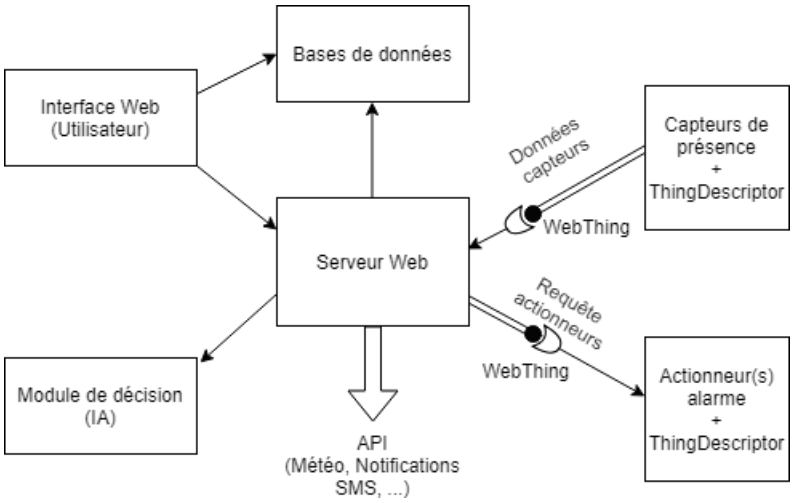

Figure 1: Architecture Globale

3 Architecture matérielle

3.1 Vue d'ensemble

L'arduino est relié à un pc portable (qui joue le rôle de serveur) via son port USB.

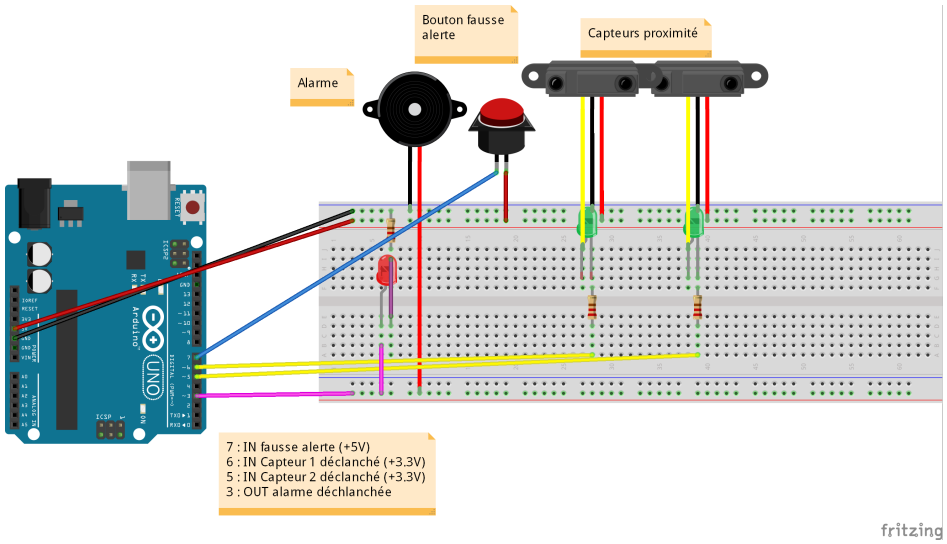

Figure 2: Architecture matérielle (Arduino)

3.2 Capteurs

L'arduino est relié à deux capteurs de présence, l'un orienté légèrement vers la gauche, l'autre vers la droite. Ceci permet de distinguer les entrées des sorties.

3.3 Actionneurs

Nous avons deux actionneurs s'activant en même temps lorsque l'alarme est déclanchée :

- Une LED rouge
- Une capsule piézoélectrique émettant un son

4 Service Web

4.1 Architecture

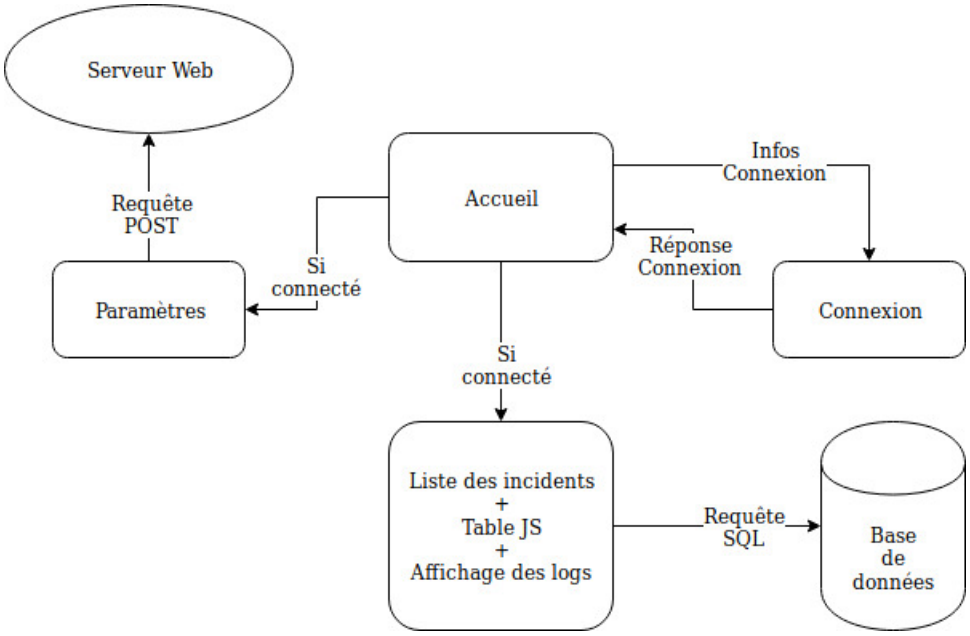

Figure 3: Interface WEB

4.2 Interface utilisateur

Nous avons mis en place une interface utilisateur (`http://piercevaldev.tk/pomSecure`). Cette dernière permet les actions suivantes :

- s'inscrire sur le site web (nom d'utilisateur + mot de passe)
- Ajouter son arduino (uniquement si connecté)
- Acceder aux paramètres de l'arduino (tester l'alarme, éteindre/allumer le arduino et passer en phase d'apprentissage)
- voir toutes les entrées prises par le arduino.

Le site est connecté a une base de donnée, qui lui permet de gerer les différents utilisateurs, et de vérifier qu'ils puissent acceder uniquement à leurs propre arduino.

La communication avec le serveur web principal se fait via des requêtes AJAX (GET pour connaître l'état du arduino, et POST pour lui donner des directives)

4.3 Standard du W3C

5 Intelligence du détecteur de présence

5.1 Objectif

Nous souhaitons permettre à notre système de, une fois installé, passer par une phase d'apprentissage où il apprend les habitudes de l'utilisateur. Une fois cette phase passée, le système doit être en mesure de discerner trois niveaux de présence :

- Présence normale
- Présence suspecte :
- Présence très suspecte : le système déclanche l'alarme

5.2 Méthode

5.2.1 Introduction

La détection d'anomalies (dite aussi détection d'outliers) est une tâche de l'apprentissage automatique qui consiste à déceler dans les données, les instances (individus) ayant un comportement différent (inhabituel) des autres instances de la base dites normales. Dans le cas de la détection d'intrusion, une présence à un moment où il n'y en a habituellement pas est considérée anormale.

L'approche de la détection des anomalies consiste simplement à apprendre à quoi ressemble l'activité normale (à l'aide d'un historique de présence supposées non-d'intrusion) et d'identifier toutes présences qui différent.

Figure 4: Clustering pour la détection d'anomalie

5.2.2 Classe de détection

Dans la logique d'un système modulable, nous avons rendu très simple le changement de méthode de détection d'anomalie. Le module prends en entrée un fichier csv indiquant "heure, durée" et renvoie le niveau d'anomalie (normal, suspect, très suspect). Nous nous sommes donc intéressés à différents algorithmes qui ont été proposés dans la littérature. On distingue les algorithmes :

- Supervisés (détection d'anomalie) : Étiquettes disponibles pour les données normales et les anomalies
- Semi-supervisés (détection de nouveauté) : seules les données normales sont disponibles, l'algorithme apprend uniquement sur des données normales
- Non supervisés (détection des valeurs aberrantes) : pas d'étiquettes, le set d'apprentissage comporte les données normales et anormales. On part du principe que les anomalies sont très rares.

5.2.3 Méthodes algorithmiques

- Statistical AD techniques. On adapte un modèle statistique pour le comportement normal. Exemple : EllipticEnvelope
- Density-based. Exemple : Facteur de valeurs aberrantes locales (LOF)
- Support estimation - OneClassSVM - MV-set estimat
- High-dimensional techniques. Exemples : Spectral Techniques, Random Forest, Isolation Forest

5.3 Solution retenue

5.3.1 Mise en place

Partie 1 :

Nos données sont composées pour chaque évènement (entrée ou sortie) d'une heure (comprise entre 0 et 168 pour les 7 jours de la semaine) et d'une durée, mais elles ne sont pas labélisées, c'est à dire qu'on ne sait pas quel évènement est suspicieux ou non. Pour ce faire nous avons recours à un algorithme de clustering non supervisé, DBSCAN, qui nous permettra d'annoter nos données en différents cluster qui seront suspicieux ou non. Nous avons choisi DBSCAN car il s'adapte bien au partionnement de nos données, celui-ce étant basé sur la densité.

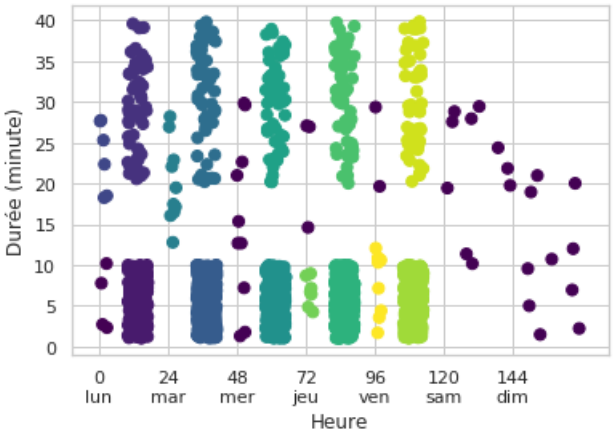

Figure 5: Données regroupées en différents clusters

Partie 2 :

Afin de prédire si une action sera suspecte ou non, nous utiliserons un algorithme de régression logistique. Celui-ci apprendra sur le dataset labélisé créé à l'étape 1 et pourra ainsi prédire si une action est sans risque, suspecte ou très suspecte.

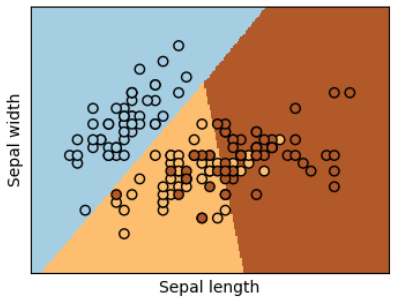

Figure 6: Exemple de régression logistique

5.3.2 Les données d'apprentissage

Dans un cas d'utilisation normale, le détecteur de présence récolte des données au cours de sa phase d'apprentissage de 30 jours pour apprendre les habitudes de l'utilisateur. Dans notre cas, nous avons générés des données artificiellement pour pouvoir tester notre système : Du lundi au vendredi, on détecte des présences entre 10 et 17h à durées variables et le weekend à toute heure.

5.4 Résultats

Nous avons testés la classification de données nouvellement insérées par le système :

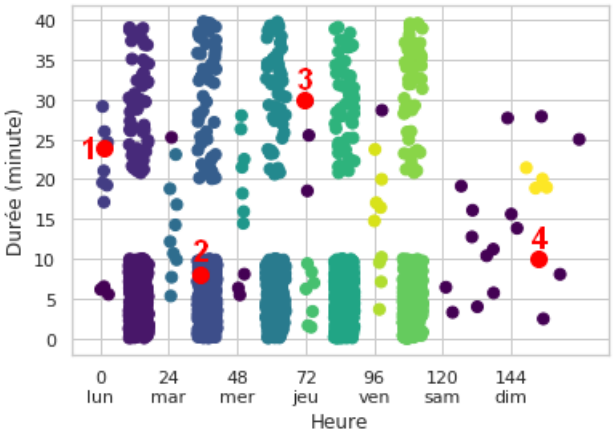

Figure 7: Classifiction des données

Les points 2 et 4 sont classés comme normaux. Le point 4 est relativement isolé, mais il fait parti d'un cluster correspondant au week end où l'on observe des présences à toute heure : il est normal que ce cette activité ne soit pas sonsidérée comme suspecte.

Le point 1 est considéré comm suspect car il se trouve proche d'un cluster composée depeu de données.

Le point 3 est considéré comme très suspect.

6 API / Interactions entre composants

- Entre l'Arduino et le programme principal :
  - Communication "Serial" (fichier "/dev/ttyACM0"), librairie pySerial.
  - Format des données : octets.
  - L'Arduino envoie un seul octet (pouvant prendre les valeurs 0, 1, 2, ou 3) à chaque changement d'état.
  - 0 signifie que les 2 capteurs ne détectent pas de mouvement.
  - 1 signifie que le capteur de gauche a détecté un mouvement.
  - 2 signifie que le capteur de droite a détecté un mouvement.
  - 3 signifie que les 2 capteurs ont détecté un mouvement.
  - Le programme envoie des octets (peu importe lesquels) à l'Arduino s'il doit activer son alarme
- Entre le programme principal et le module de décision :
  - Les deux programmes seront écrits en Python et pourront donc communiquer soit par une socket, soit par une communication inter-threads (module Queue).
  - Le programme principal envoie les caractéristiques d'un événement (timestamp + valeur des capteurs)

  - Le module de décision répond avec la classe de l'événement (non suspicieux, légèrement suspicieux, très suspicieux ; codé sous forme d'entier)

- Entre l'interface Web et le programme principal :
  - L'interface Web envoie des requêtes HTTP (notamment POST et GET) au programme principal.
  - Le programme principal exécute les actions associées, et répond notamment à travers les codes HTTP et éventuellement du JSON.