BTS Services informatiques aux organisations — SESSION 2026
ANNEXE 7-1-A : Fiche descriptive de réalisation professionnelle (recto)
Épreuve E6 - Administration des systèmes et des réseaux (option SISR) - Coefficient 4

DESCRIPTION D'UNE RÉALISATION PROFESSIONNELLE			**N° réalisation :** **1**
Nom, prénom	EMILIEN Noam		
Numéro de candidat	02428526184		
☑ Épreuve ponctuelle ☐ Contrôle en cours de formation		**Date :**	Mai 2026

Contexte de la réalisation professionnelle

Intitulé de la réalisation professionnelle

Conception et déploiement d'un tableau d'affichage de score en temps réel pour matchs de handball sur infrastructure réseau locale

Période de réalisation :	Mars 2026 – Mai 2026	**Lieu :**	Domicile (matériel prêté par UIMM Pôle formation Poitou-Charentes)
Modalité :		☑ Seul(e) ☐ En équipe	

Compétences travaillées

☑ Concevoir une solution d'infrastructure réseau
☑ Installer, tester et déployer une solution d'infrastructure réseau
☐ Exploiter, dépanner et superviser une solution d'infrastructure réseau

Conditions de réalisation (ressources fournies, résultats attendus)

Ressources fournies :
- Raspberry Pi 4 Model B (4 Go RAM) acheté personnellement.
- Commutateur réseau, borne WiFi Aruba, câbles Ethernet, adaptateur PoE prêtés par l'UIMM Pôle Formation Poitou-Charentes.

Résultats attendus :
Un tableau d'affichage opérationnel pour les matchs de handball du Grand Poitiers Handball 86, accessible depuis un ordinateur arbitre via navigateur web, affichant en temps réel les scores, le chronomètre, les pénalités de 2 minutes et les logos des équipes sur un écran TV en mode kiosque.

Description des ressources documentaires, matérielles et logicielles utilisées

Matérielles :
Raspberry Pi 4 Model B 4 Go RAM, commutateur réseau, borne WiFi Aruba Networks, ordinateur arbitre (PC Windows), écran TV HDMI.

Logicielles :
Raspberry Pi OS 64-bit (Debian Trixie), Node.js, Express.js, npm, pm2 (gestionnaire de processus), Chromium (mode kiosque), Git. Langages : HTML5, CSS3, JavaScript (Node.js + front-end).

Documentaires :

- https://github.com/KTM-EduTech/Simple-Scoreboard-with-Timer
- https://www.raspberrypi.com/documentation
- https://nodejs.org/en/docs
- https://expressjs.com/fr/4x/api.html
- https://developer.mozilla.org/fr/docs/Web/API/Fetch_API
- https://pm2.keymetrics.io/docs/usage/quick-start/

Modalités d'accès aux productions et à leur documentation

[Documentation technique complète : **PDF joint au dossier numérique**.]

- Vue spectateurs (TV kiosque) : http://192.168.10.34:3000/index.html

Authentification HTTP Basic : **identifiant** = arbitre ; **mot de passe** = handball86

- Vue arbitre (PC) : http://192.168.10.34:3000/control.html
- API REST : http://192.168.10.34:3000/api/state

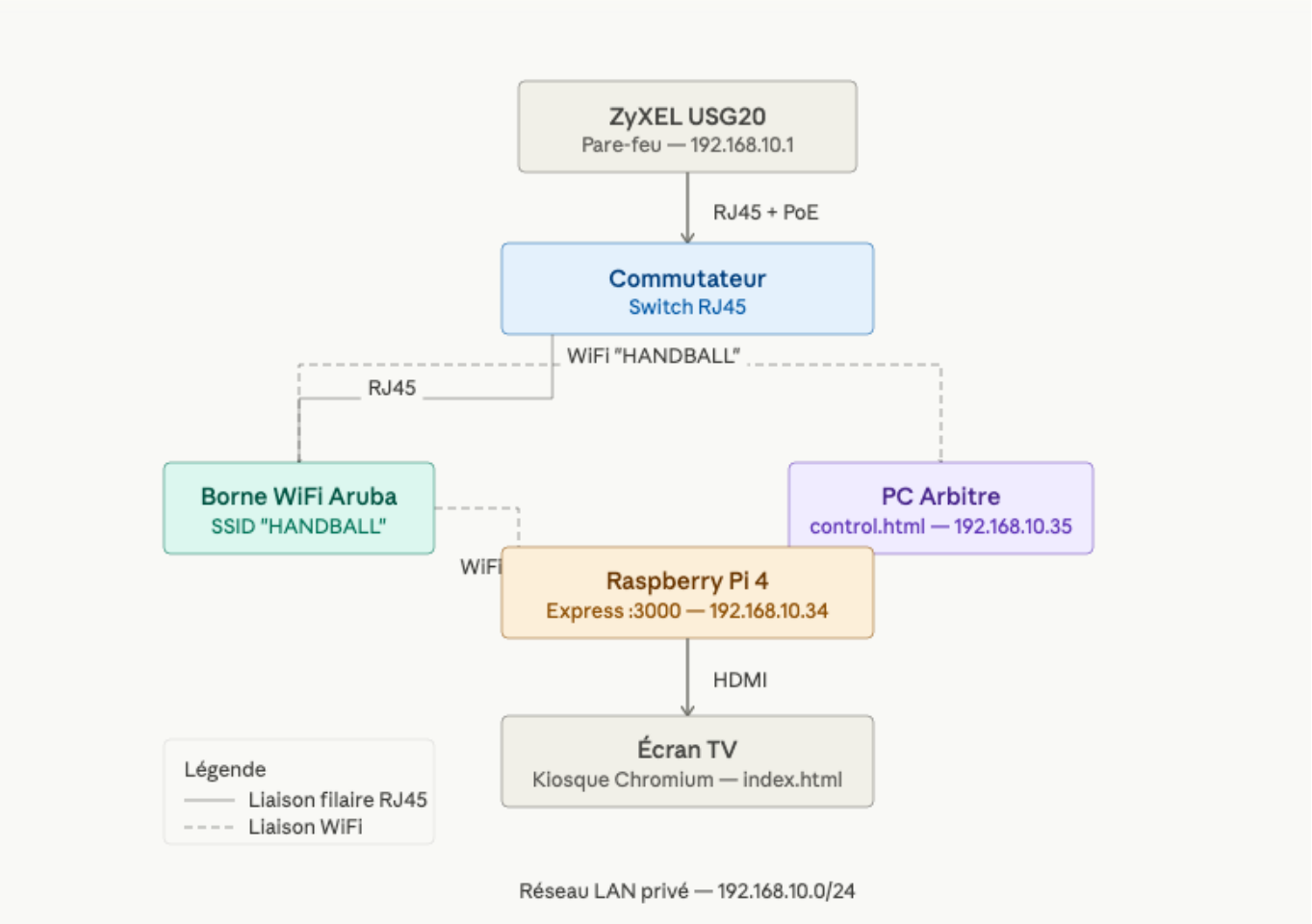
BTS Services informatiques aux organisations — SESSION 2026
ANNEXE 7-1-A : Fiche descriptive de réalisation professionnelle (verso)
Épreuve E6 - Administration des systèmes et des réseaux (option SISR) - Coefficient 4

Descriptif de la réalisation professionnelle, y compris les productions réalisées et schémas explicatifs

1. Contexte et problématique

Le Grand Poitiers Handball 86 (N3, club amateur) ne disposait d'aucun système d'affichage numérique lors des matchs à domicile. Les scores et le chronomètre étaient gérés manuellement, sans affichage en temps réel pour les spectateurs. L'objectif était de concevoir une solution économique, autonome et professionnelle, déployable dans un gymnase.

2. Architecture réseau mise en place

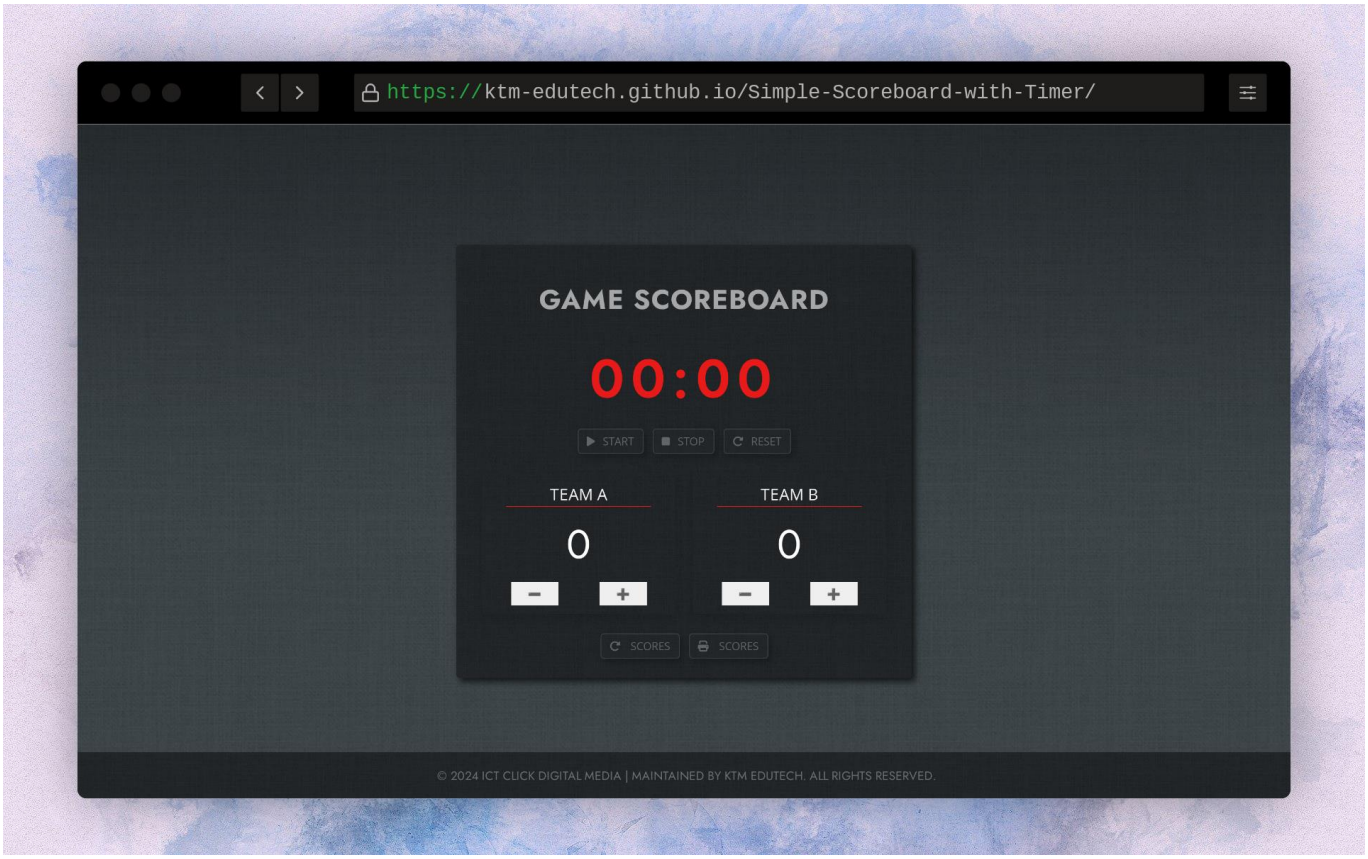


Pare-feu ZyXEL → Commutateur réseau (LAN 192.168.10.0/24) → [Borne WiFi Aruba "HANDBALL"] + [Raspberry Pi 4 192.168.10.34] + [PC Arbitre 192.168.10.35]

Le Raspberry Pi est connecté en HDMI à l'écran TV (mode kiosque Chromium). L'arbitre contrôle l'application depuis son PC via le réseau WiFi HANDBALL. Les spectateurs voient l'affichage en temps réel sur l'écran TV.

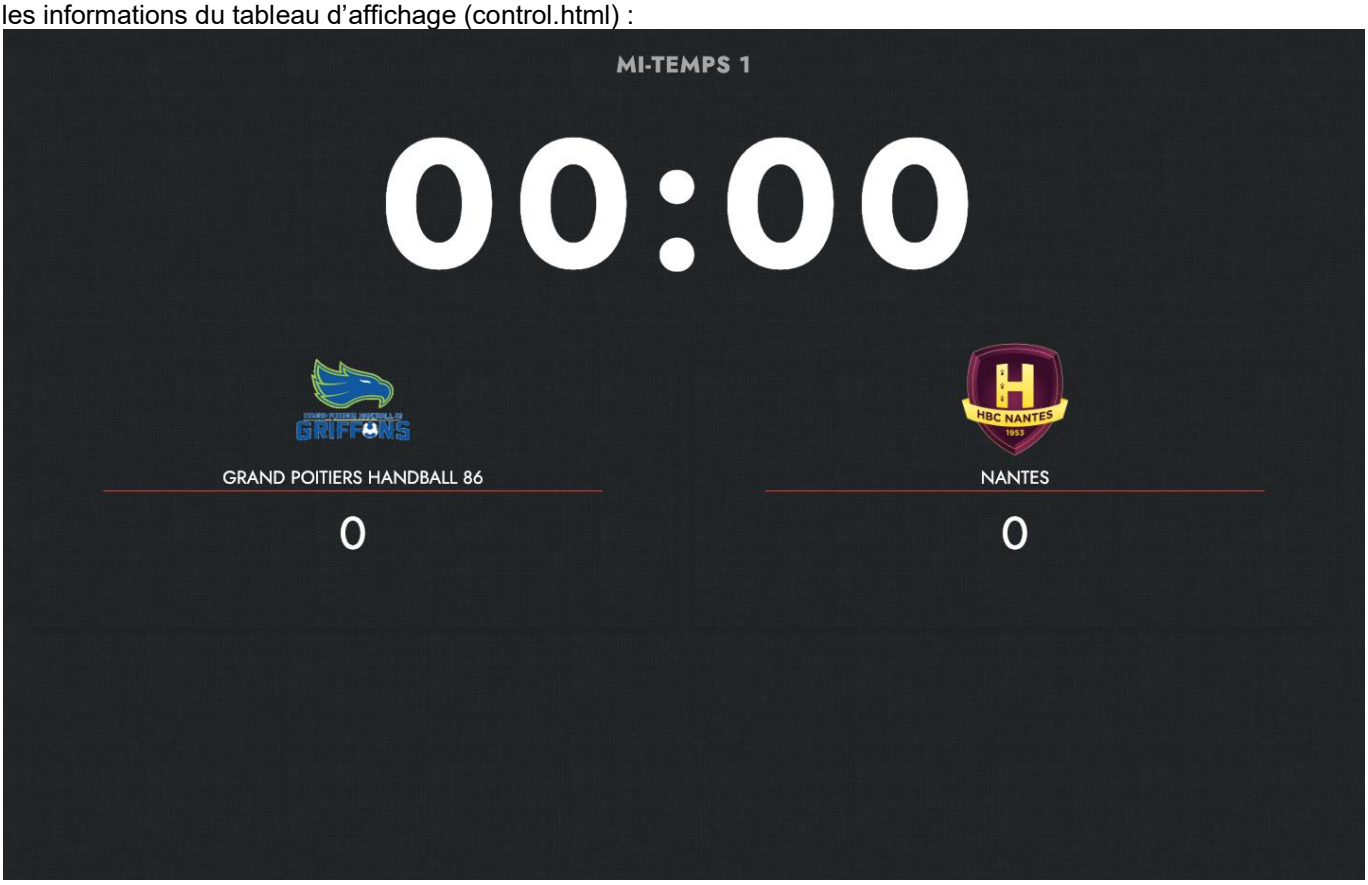
3. Solution technique déployée

Pour réaliser ce projet j'ai utilisé comme base le projet open source Simple-Scoreboard-with-Timer de Kyd Tantano Masong (GitHub : ktm-edutech).

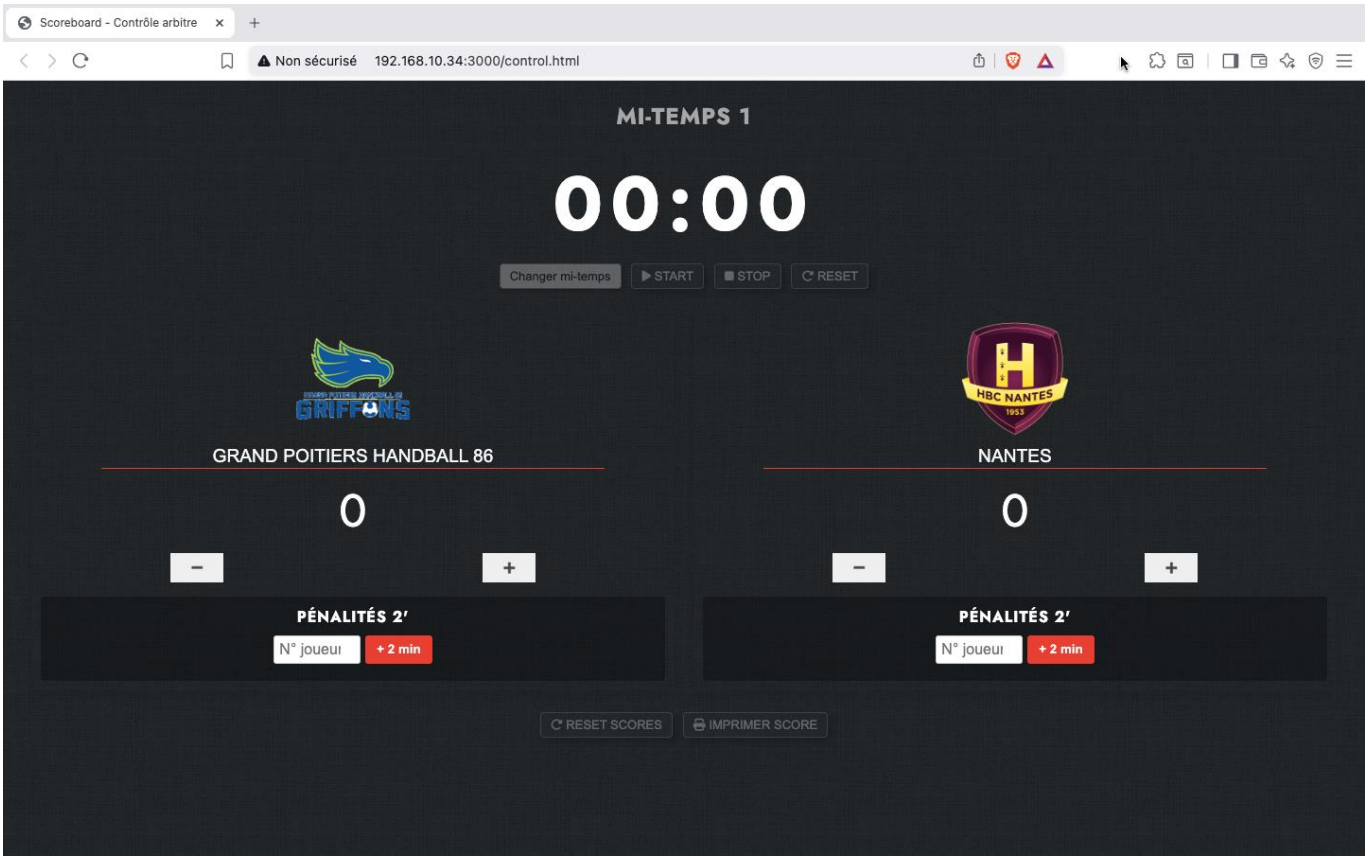


(Voici à quoi ressemble le projet initial de Kyd Tantano Masong)

Je l'ai adapté à ma manière afin que le tableau d'affichage puisse répondre plus précisément aux besoins des clubs de Handball en créant une vue spectateur (index.html) ainsi qu'une vue pour l'arbitre de table qui met à jour les informations du tableau d'affichage (control.html) :



(Vue des spectateurs sur l'écran TV)



(Vue de l'arbitre sur l'ordinateur portable)

De plus j'ai complètement modifié la structure initiale : migration de localStorage vers une architecture API REST, ajout du serveur Express/Node.js, ajout du système de pénalités 2 minutes, ajout et gestion des logos d'équipes, déploiement sur Raspberry Pi en mode kiosque.

Serveur web Express/Node.js (port 3000) hébergé sur le Raspberry Pi, géré par pm2 (démarrage automatique au boot). L'application expose une API REST :
- GET /api/state → lecture de l'état du match (scores, chrono, pénalités, logos)
- POST /api/state → mise à jour de l'état du match
- GET /api/logos → liste des logos d'équipes disponibles

Les données sont persistées dans data.json sur le Pi.

4. Fonctionnalités réalisées

- Affichage temps réel (polling 1 seconde) : scores, chronomètre, mi-temps, logos des équipes
- Chronomètre 0 → 30:00 (bloqué à 30 min, réinitialisation par mi-temps)
- Pénalités 2 minutes : saisie du numéro de joueur, compte à rebours visible, max 3 simultanées/équipe
- Gestion des logos d'équipes : sélection via modal dans l'interface arbitre
- Mode kiosque Chromium (démarrage automatique au boot du Pi, plein écran TV)
- Démarrage automatique du serveur via pm2 + systemd au boot

5. Tests réalisés

- Test de synchronisation : modification d'un score sur control.html → affichage mis à jour sur index.html (TV) en moins d'une seconde ✓
- Test pénalités : ajout/suppression de pénalités, vérification du compte à rebours, limite à 3 ✓
- Test persistance : redémarrage du Pi → données conservées dans data.json, serveur redémarré automatiquement ✓
- Test kiosque : boot du Pi → Chromium lance automatiquement index.html après 10 secondes ✓
- Test réseau : accès depuis PC arbitre via WiFi HANDBALL → latence < 1 seconde ✓

6. Difficultés rencontrées et solutions apportées

• Migration localStorage → API REST : le stockage local (localStorage) ne permettait pas la synchronisation entre les appareils. Migration vers une architecture client-serveur avec API REST et stockage persistant JSON.
• Arrêt du serveur Apache2 préexistant (conflit port 80) → désactivation via systemctl.
• Pérennisation du service : utilisation de pm2 + systemd pour garantir le redémarrage automatique d'Express après chaque reboot.

BTS Services Informatiques aux Organisations
Option SISR — Administration des systèmes et des réseaux
SESSION 2026

DOCUMENTATION TECHNIQUE RÉALISATION PROFESSIONNELLE N°1

Conception et déploiement d'un tableau d'affichage de score en temps réel pour matchs de handball

Raspberry Pi 4 — Node.js/Express — API REST — Kiosque Chromium

Candidat	EMILIEN Noam
N° candidat	02428526184
Période	Mars 2026 – Mai 2026
Lieu	Domicile
Serveur	Raspberry Pi 4 Model B 4Go — Raspberry Pi OS 64-bit

Compétences évaluées : Concevoir une solution réseau | Installer, tester et déployer une solution réseau

Sommaire

1. Contexte et problématique

Le **Grand Poitiers Handball 86**, club évoluant en Nationale 3, ne disposait d'aucun système d'affichage numérique lors de ses matchs à domicile. Les scores et le chronomètre étaient gérés manuellement, sans affichage en temps réel pour les spectateurs.

L'objectif de cette réalisation est de concevoir et déployer une **solution économique, autonome et professionnelle**, déployable dans un gymnase, permettant à un arbitre de contrôler l'affichage depuis son poste, pendant que les spectateurs visualisent le score en temps réel sur un écran TV.

Contraintes identifiées

- Budget limité (club amateur) → solution basée sur matériel existant/économique
- Déploiement en gymnase → réseau WiFi privé dédié, indépendant d'internet
- Facilité d'utilisation → interface arbitre simple, accessible via navigateur
- Fiabilité → démarrage automatique au boot, pas d'intervention manuelle

2. Architecture réseau et infrastructure

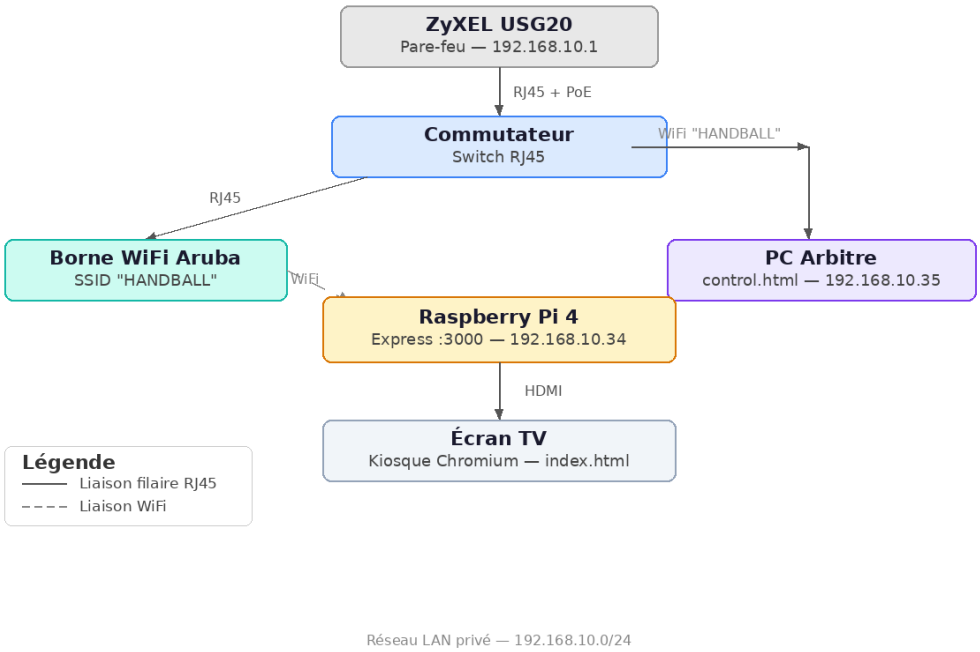


Figure 1 — Architecture réseau du système de tableau d'affichage handball

Équipements

Équipement	Modèle	Rôle	IP
Pare-feu	ZyXEL USG20	Sécurité réseau + alimentation PoE de la borne	192.168.10.1
Commutateur	Switch RJ45	Interconnexion des équipements	-
Borne WiFi	Aruba Networks	Diffusion SSID "HANDBALL"	192.168.10.10
Raspberry Pi 4	Model B — 4Go RAM	Serveur Express + affichage kiosque	192.168.10.34
PC Arbitre	PC Windows	Contrôle via control.html (WiFi)	192.168.10.35
Écran TV	HDMI	Affichage spectateurs (kiosque Chromium)	-

Limite identifiée : Le Raspberry Pi se connecte au réseau via la borne WiFi (contrainte physique : le Pi est positionné derrière l'écran TV à 5-10m de hauteur, loin du commutateur). En cas de panne WiFi, l'affichage TV est interrompu. **Solution envisagée :** prolonger un câble Ethernet depuis le commutateur jusqu'au Pi (câble RJ45 long en goulotte).

3. Stack technique

Couche	Technologie	Rôle
Système	Raspberry Pi OS 64-bit (Debian Trixie)	OS du serveur
Runtime	Node.js 20.19.2	Environnement d'exécution JavaScript
Framework	Express.js	Serveur web et API REST
Processus	pm2	Gestionnaire de processus (démarrage auto)
Persistance	data.json	Stockage de l'état du match
Frontend	HTML5 / CSS3 / JavaScript	Interfaces arbitre et spectateurs
Affichage	Chromium (mode kiosque)	Affichage plein écran TV
Versioning	Git	Gestion du code source

4. Arborescence du projet

Structure des fichiers sur le Raspberry Pi :

```
~/Simple-Scoreboard-with-Timer/
■■■ index.html ← Vue spectateurs (TV kiosque)
■■■ control.html ← Vue arbitre (PC)
■■■ script.js ← Logique JavaScript (API, timer, pénalités)
■■■ styles.css ← Feuille de style
■■■ server.js ← Serveur Express + API REST
■■■ data.json ← État du match (persistance)
■■■ logos/ ← Logos des équipes
■ ■■■ grand-poitiers.png
■ ■■■ nantes-hbc.png
■ ■■■ ...
■■■ package.json ← Dépendances Node.js
■■■ node_modules/ ← Express et dépendances
```

```
pi@raspberrypi:~/Simple-Scoreboard-with-Timer $ ls -la ~/Simple-Scoreboard-with-Timer/
total 116
drwxr-xr-x  5 pi pi  4096 25 mai   20:44 .
drwx------ 17 pi pi  4096 25 mai   17:08 ..
-rw-r--r--  1 pi pi  4129 23 mai   17:49 control.html
-rw-rw-r--  1 pi pi   365 25 mai   20:38 data.json
-rw-r--r--  1 pi pi  6148 14 mai   23:05 .DS_Store
drwxr-xr-x  7 pi pi  4096 25 mai   18:08 .git
-rw-r--r--  1 pi pi  1367 23 mai   17:49 index.html
-rw-r--r--  1 pi pi  1075 14 mai   23:05 LICENSE
drwxrwxr-x  2 pi pi  4096 23 mai   19:12 logos
drwxrwxr-x 67 pi pi  4096 23 mai   00:18 node_modules
-rw-rw-r--  1 pi pi    52 23 mai   00:18 package.json
-rw-rw-r--  1 pi pi 27850 23 mai   00:19 package-lock.json
-rw-r--r--  1 pi pi  3914 14 mai   23:05 README.md
-rw-r--r--  1 pi pi  6943 23 mai   00:54 script_avant_express.js
-rw-r--r--  1 pi pi  9991 23 mai   17:50 script.js
-rw-rw-r--  1 pi pi   891 23 mai   17:44 server.js
-rw-r--r--  1 pi pi  7605 23 mai   17:51 styles.css
pi@raspberrypi:~/Simple-Scoreboard-with-Timer $
```

Figure 2 — Arborescence du projet sur le Raspberry Pi (ls -la)

5. Serveur Express et API REST

Le serveur **Express.js** tourne sur le port 3000 du Raspberry Pi. Il expose trois endpoints :

Méthode	Endpoint	Description
GET	/api/state	Lecture de l'état du match (scores, chrono, pénalités, logos)
POST	/api/state	Mise à jour de l'état du match
GET	/api/logos	Liste des logos disponibles dans le dossier logos/
GET	/*	Serveur de fichiers statiques (HTML, CSS, JS, images)

La commande **curl** permet de tester l'API directement depuis le terminal. La réponse est identique au contenu de **data.json**, confirmant la cohérence entre l'API et le fichier de persistance :

```
$ curl http://localhost:3000/api/state
```

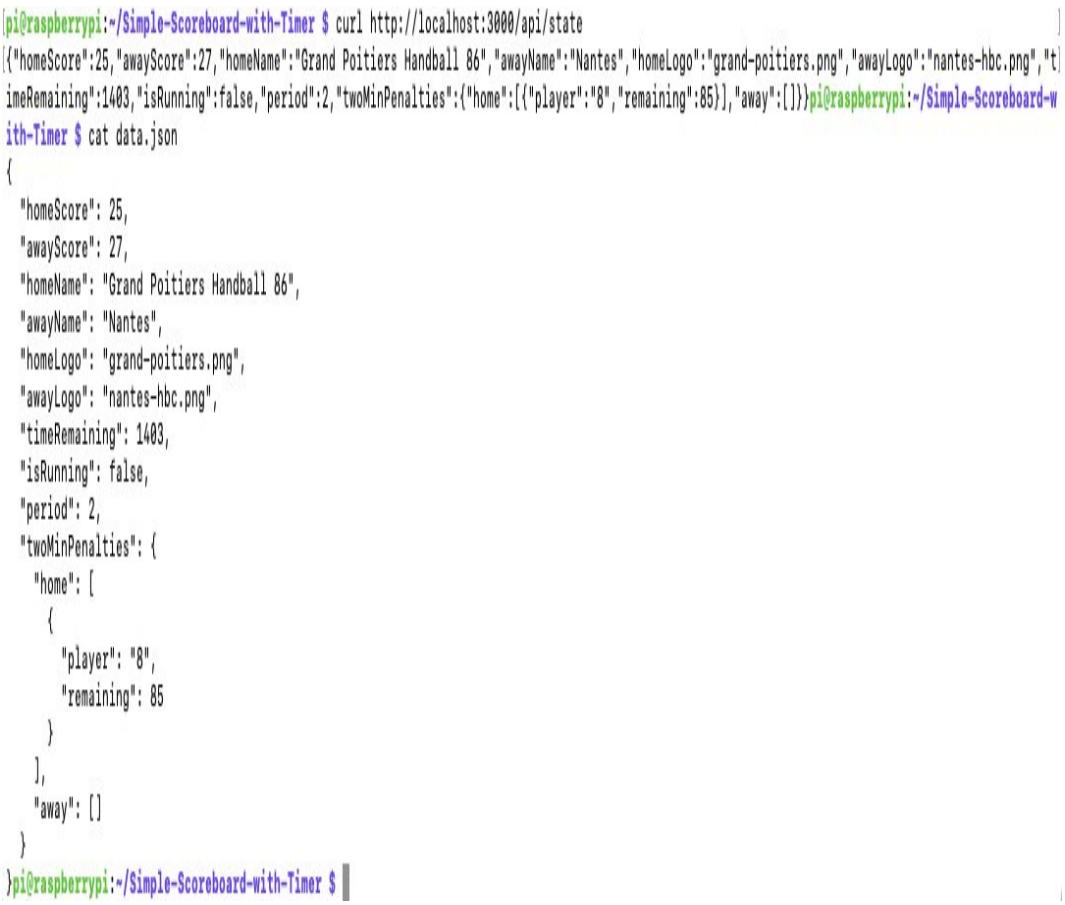


Figure 3 — Réponse de l'API /api/state (curl) et data.json — corrélation confirmée

Note : La réponse curl apparaît tronquée dans le terminal (affichage sur une seule ligne). C'est un comportement d'affichage uniquement — toutes les données sont bien transmises. Le fichier data.json ci-dessous confirme la complétude des données.

```
$ pm2 list
```

```
pi@raspberrypi:~/Simple-Scoreboard-with-Timer $ pm2 list
```

id	name	namespace	version	mode	pid	uptime	↺	status	cpu	mem	user	watching
0	scoreboard	default	N/A	fork	3365	2h	0	online	0%	69.5mb	pi	disabled

Figure 4 — pm2 list : service scoreboard online, uptime 2h, 0 restart

6. Structure des données — data.json

Toutes les données du match sont persistées dans **data.json**. Ce fichier est lu à chaque requête GET et écrasé à chaque requête POST. Il survit aux redémarrages du serveur.

Figure 5 — Structure de data.json avec exemple de match en cours (pénalité N°8 active)

Champ	Type	Description
`homeScore / awayScore`	`integer`	Scores des équipes domicile et visiteur
`homeName / awayName`	`string`	Noms des équipes
`homeLogo / awayLogo`	`string`	Nom du fichier logo dans logos/
`timeRemaining`	`integer`	Temps écoulé en secondes (0 → 1800)
`isRunning`	`boolean`	État du chronomètre (en cours / arrêté)
`period`	`integer`	Numéro de mi-temps (1 ou 2)
`twoMinPenalties`	`object`	Pénalités actives {home: [], away: []}

7. Interface arbitre — control.html

L'interface arbitre est accessible depuis n'importe quel navigateur sur le réseau HANDBALL. Elle permet de contrôler tous les éléments du match en temps réel :

- Démarrer / Arrêter / Réinitialiser le chronomètre
- Changer de mi-temps
- Incrémenter / Décrémenter les scores
- Ajouter une pénalité 2 minutes (numéro de joueur requis)
- Changer le logo d'une équipe via le modal de sélection
- Imprimer le score final

```
URL : http://192.168.10.34:3000/control.html
```

Figure 6 — Interface arbitre (control.html) : mi-temps 2, score 25-27, pénalité N°8 active

8. Interface spectateurs — index.html (kiosque TV)

L'interface spectateurs s'affiche en plein écran sur la TV connectée au Raspberry Pi via HDMI. Elle se rafraîchit automatiquement **toutes les secondes** via un polling sur l'API. Elle n'affiche aucun bouton de contrôle — uniquement les informations du match :

- Mi-temps en cours
- Chronomètre (taille XXL pour lisibilité à distance)
- Logos des équipes
- Noms des équipes
- Scores
- Badges rouges des pénalités 2 minutes en cours

```
URL : http://192.168.10.34:3000/index.html
```

Figure 7 — Vue spectateurs (index.html) : affichage temps réel, score 25-27, badge pénalité N°8

Synchronisation temps réel : toute modification effectuée sur control.html (score, chrono, pénalité) est visible sur index.html en moins d'une seconde grâce au polling JavaScript coté client.

Configuration kiosque Chromium

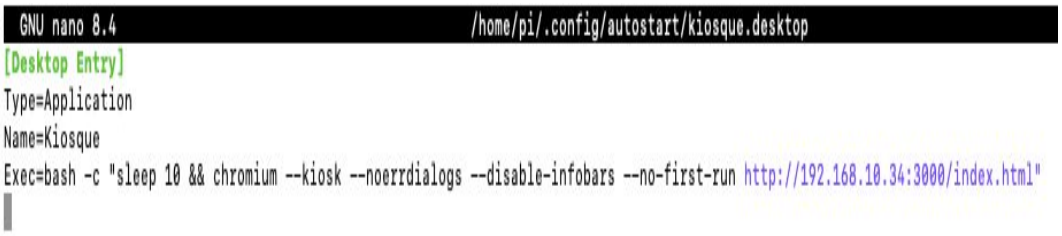


Figure 8 — Fichier kiosque.desktop : démarrage automatique de Chromium au boot du Pi

9. Pénalités 2 minutes

Le règlement du handball prévoit des exclusions temporaires de 2 minutes. Le système gère jusqu'à **3 pénalités simultanées par équipe**. Chaque pénalité affiche un compte à rebours en temps réel.

Fonctionnalité	Description
Saisie	L'arbitre entre le numéro du joueur exclu et clique "+ 2 min"
Affichage arbitre	Liste des pénalités avec compte à rebours et bouton suppression (✕)
Affichage TV	Badge rouge "N°X - 1:25" visible sous le score de l'équipe
Décompte automatique	Le chronomètre pénalité décrémente chaque seconde (synchronisé avec le timer)
Expiration	La pénalité disparaît automatiquement quand le compte atteint 0
Limite	Maximum 3 pénalités simultanées par équipe (alerte si dépassement)

Figure 9 — Pénalité N°8 active (1:25 restant) côté arbitre avec bouton suppression

Figure 10 — Badge pénalité "N°8 - 1:25" affiché en rouge sur la vue spectateurs TV

10. Gestion des logos d'équipes

Les logos des équipes sont stockés dans le dossier **~/Simple-Scoreboard-with-Timer/logos/** sur le Raspberry Pi. Ils sont téléchargés en début de saison depuis le web directement sur le Pi via la commande **wget**.

Téléchargement des logos

La commande wget permet de télécharger un logo et de le renommer directement :

```
$ mkdir -p ~/Simple-Scoreboard-with-Timer/logos
$ cd ~/Simple-Scoreboard-with-Timer/logos
$ wget -O grand-poitiers.png "https://upload.wikimedia.org/wikipedia/fr/thumb/6/6e/
GrandPoitiersHandball86.png/120px-GrandPoitiersHandball86.png"
$ wget -O nantes-hbc.png "https://upload.wikimedia.org/wikipedia/fr/..."
```

Option -O : permet de spécifier le nom du fichier de sortie (Output). Sans cette option, wget conserve le nom original de l'URL (souvent long et peu pratique). Formats supportés : .png, .jpg, .jpeg, .svg

Sélection du logo via modal

Dans l'interface arbitre, un clic sur le logo d'une équipe ouvre une **fenêtre modale** listant tous les logos disponibles dans le dossier logos/. L'arbitre sélectionne le logo souhaité et il s'affiche immédiatement sur la TV.

Figure 11 — Modal de sélection de logo : 6 équipes disponibles, Grand Poitiers sélectionné

11. Démarrage automatique — pm2 + systemd

pm2 est le gestionnaire de processus Node.js utilisé pour garantir que le serveur Express redémarre automatiquement après chaque reboot du Pi, sans intervention manuelle.

```
$ pm2 start server.js --name scoreboard
$ pm2 save
$ pm2 startup → génère la commande systemd à exécuter
```

La commande **pm2 startup** crée un service systemd **pm2-pi.service** qui démarre pm2 au boot. pm2 restaure ensuite automatiquement le process *scoreboard* sauvegardé.

Commande	Description
`pm2 list`	Affiche l'état de tous les processus gérés
`pm2 monit`	Dashboard temps réel (CPU, RAM, logs)
`pm2 logs scoreboard`	Consultation des logs d'activité
`pm2 describe scoreboard`	Détails complets (uptime, restarts, chemins logs)
`pm2 restart scoreboard`	Redémarrage du service
`pm2 stop scoreboard`	Arrêt du service

12. Kiosque Chromium — démarrage automatique

Chromium est configuré pour démarrer automatiquement en mode kiosque (plein écran, sans barre de navigation, sans contrôles) via le fichier d'autostart du bureau graphique labwc :

```
~/.config/autostart/kiosque.desktop
```

```
Exec=bash -c "sleep 10 && chromium --kiosk --noerrdialogs
--disable-infobars --no-first-run http://192.168.10.34:3000/index.html"
```

sleep 10 : délai de 10 secondes pour laisser le temps au réseau et à pm2 de démarrer avant que Chromium essaie de charger l'URL.

--kiosk : mode plein écran, sans barre d'adresse, sans bouton fermer. L'utilisateur ne peut pas quitter l'application.

13. Tests réalisés

Test	Procédure	Résultat
Synchronisation temps réel	Modifier un score sur control.html, observer index.html	✓ **Mise à jour < 1 seconde**
Pénalité 2 minutes	Ajouter pénalité N°8, observer décompte TV	✓ **Badge rouge + décompte correct**
Limite 3 pénalités	Ajouter une 4ème pénalité	✓ **Alerte bloquante affichée**
Persistance données	Redémarrer le Pi, vérifier data.json	✓ **Données conservées**
Démarrage automatique	Reboot Pi, attendre 15s	✓ **Kiosque + Express démarrés**
Changement logo	Cliquer logo → modal → sélectionner	✓ **Logo mis à jour sur TV < 1s**
API REST	curl http://localhost:3000/api/state	✓ **JSON retourné correct**
Mi-temps	Cliquer "Changer mi-temps"	✓ **Timer reset + MI-TEMPS 2 affiché**

14. Limites identifiées et évolutions envisagées

Limite	Impact	Évolution envisagée
Pi connecté via WiFi (contrainte physique)	Panne WiFi = plus d'affichage	Câble Ethernet long en goulotte
Stockage JSON (pas de base de données)	Pas d'historique des matchs	Base de données SQLite ou historique par fichier horodaté
Curseur souris visible (Wayland/labwc)	Aspect peu professionnel sur la TV	Solution Wayland pour cacher le curseur au boot

15. Bilan

Cette réalisation répond pleinement à la problématique initiale : le Grand Poitiers Handball 86 dispose désormais d'un **tableau d'affichage numérique professionnel**, déployable en gymnase, économique et autonome.

Sur le plan technique, la migration de l'architecture **localStorage** (statique, mono-appareil) vers une **architecture client-serveur avec API REST** a été la décision clé du projet. Elle permet à plusieurs clients (arbitre, spectateurs) d'accéder aux mêmes données en temps réel, quel que soit leur appareil ou leur système d'exploitation.

L'utilisation de **pm2 + systemd** garantit la résilience du service : le serveur Express redémarre automatiquement après tout incident ou reboot, sans intervention humaine. Le mode kiosque Chromium assure un affichage permanent et professionnel sur l'écran TV.

Ce projet a également une dimension **open source** : le code sera publié sur GitHub pour être réutilisé par d'autres clubs de handball amateurs qui n'ont pas les moyens d'un affichage professionnel. Il n'existe pas, à ce jour, de solution équivalente librement disponible et spécifiquement conçue pour le handball (pénalités 2 minutes incluses).

Compétence SISR	Mise en œuvre dans ce projet
Concevoir une solution d'infrastructure réseau	Architecture LAN 192.168.10.0/24, choix des équipements, schéma réseau détaillé, choix technologiques justifiés
Installer, tester et déployer une solution réseau	Installation Node.js/Express/pm2, configuration kiosque Chromium, déploiement sur Raspberry Pi 4, tests fonctionnels complets