# Serrure de la porte du local (Raspberry Pi « gatepi »)

Un Raspberry Pi lit les badges sur un lecteur **Wiegand** placé à la porte et pilote la **serrure motorisée**. Il gère aussi un **bouton** qui ouvre le garage en passant par l'atom. Voir aussi ouverture du garage et réseau du local.

Relevé du 13/10/2026.

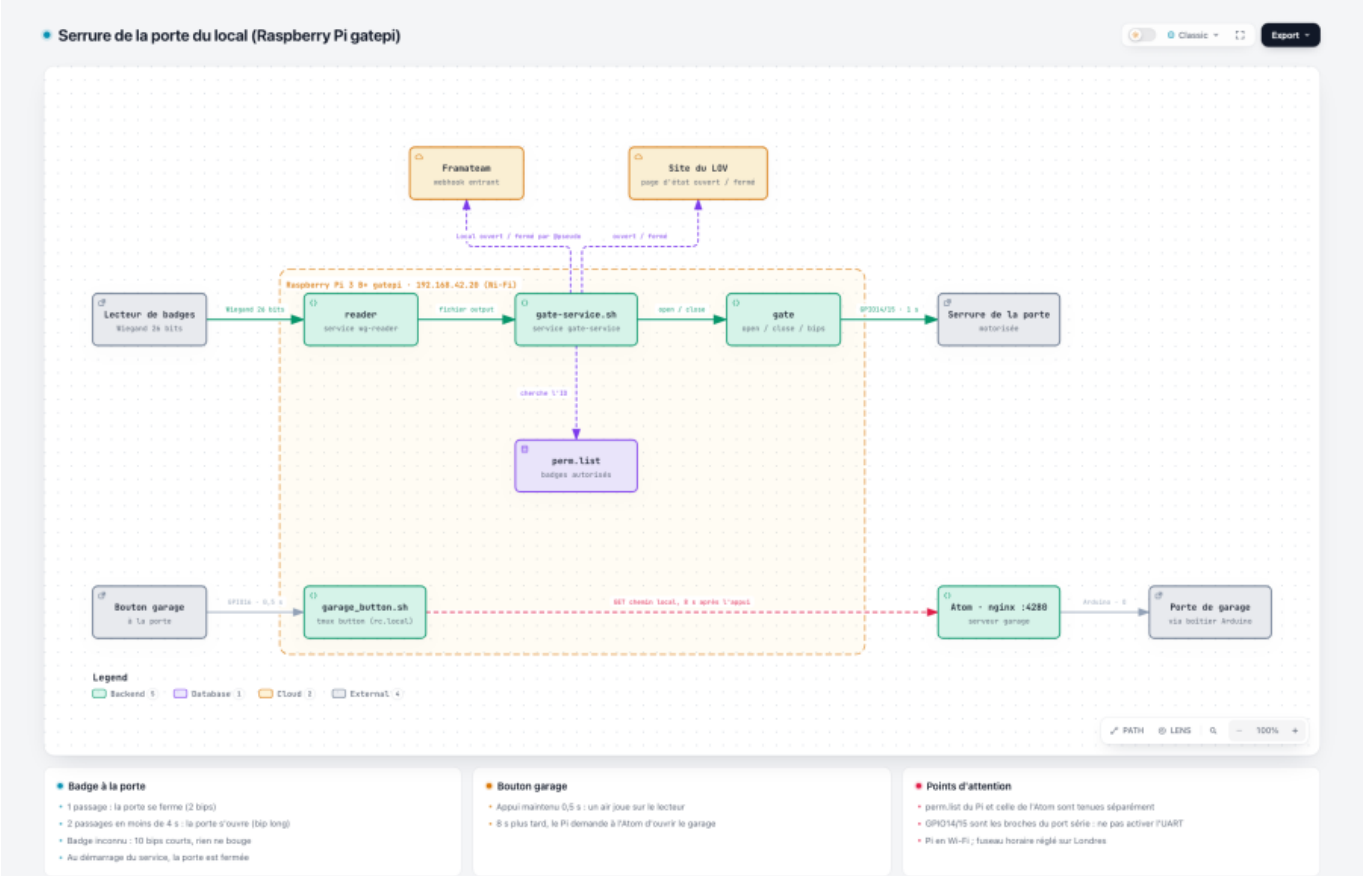



*Schéma réalisé avec Archify ; source : `docs/serrure/archify/serrure.architecture.json` dans le dépôt AdminLova.*

## Utilisation

- **Passer son badge une fois** : la porte se **ferme** (2 bips).
- **Repasser le même badge dans les 4 secondes** : la porte s'**ouvre** (bip long).
- **Badge inconnu** : 10 bips courts, rien ne bouge.
- À chaque ouverture ou fermeture, un message « Local ouvert / fermé par @pseudo » est posté sur Framateam et la page d'état du site passe à « ouvert » ou « fermé ».
- **Bouton garage** (à la porte) : le maintenir appuyé 0,5 s ; un air joue sur le lecteur et, **8 secondes plus tard**, la porte de garage s'ouvre (ou se ferme).

# Matériel

| Élément | Valeur |
|---|---|
| Carte | Raspberry Pi 3 Model B+ |
| Système | Debian 12 (bookworm) |
| Nom / adresse | `gatepi` · `192.168.42.20` (réservée dans `/etc/dnsmasq.d/static.conf` de l'atom) |
| Réseau | **Wi-Fi** (l'Ethernet n'est pas branché) |
| Accès | `ssh pi@gatepi` ; mot de passe dans le keepass du bureau |
| Lecteur | Lecteur de badges Wiegand 26 bits (fils rouge, noir, vert, blanc, jaune) |
| Serrure | **HUA XING QIANG**, serrure électronique en applique, avec antenne (récepteur radio intégré) |
| Boîtier | Boîtier imprimé en 3D (rouge) fixé à côté de la porte ; le Pi est monté sur une plaque à pastilles |
| Alimentation | Convertisseur DC-DC réglable **LM2596** (bornier à vis en entrée) qui fournit le 5 V du Pi |
| Bouton garage | Petit bouton-poussoir sur le flanc droit du boîtier (fils bleu et violet) |
| Connexions | Deux connecteurs sur le côté gauche du boîtier pour les câbles du lecteur et de la serrure |

L'ancienne adresse `192.168.42.128` n'est plus valable depuis que l'atom distribue les adresses.

## Photos





### Brochage GPIO

Relevé dans le code (numérotation wiringPi convertie avec `gpio readall`).

| Fonction | Programme | BCM | Broche physique |
|---|---|---|---|
| Lecteur : Data0 (fil vert) | `reader` | GPIO17 | 11 |
| Lecteur : Data1 (fil blanc) | `reader` | GPIO18 | 12 |
| Lecteur : bip à chaque lecture (fil jaune) | `reader` | GPIO26 | 37 |
| Bips d'état (ouvert, fermé, erreur, garage) | `gate` | GPIO4 | 7 |
| Serrure : **fermer** (impulsion de 1 s) | `gate close` | GPIO14 (TxD) | 8 |
| Serrure : **ouvrir** (impulsion de 1 s) | `gate open` | GPIO15 (RxD) | 10 |
| Bouton garage (vers la masse) | `garage_button` | GPIO16 | 36 |

Les fils Data0 et Data1 du lecteur sont en 5 V : ils doivent être ramenés à 3,3 V avant les GPIO du Pi.

⚠ **Ne pas activer le port série du Pi** (`enable_uart=1` ou `raspi-config` → Serial) : la serrure est branchée sur les broches du port série (GPIO14/15), et les messages de démarrage la feraient basculer.

## Logiciel

Dossier `/home/pi/rpi-wiegand-reader`, dérivé de rpi-wiegand-reader.

| Fichier | Rôle | Lancé par |
|---|---|---|
| `reader` (`reader.c`) | Décode les badges et ajoute une ligne au fichier `output` | service `wg-reader` |
| `gate-service.sh` | Vérifie le badge dans `perm.list`, ouvre ou ferme, prévient Frameteam et le site | service `gate-service` |
| `gate` (`gate.c`) | `gate open`, `gate close`, `gate beepo / beepc / beepe / beepg` | `gate-service.sh`, `garage_button.sh` |

| Fichier | Rôle | Lancé par |
|---|---|---|
| `garage_button` et `garage_button.sh` | Attend l'appui sur le bouton, joue un air, attend 8 s et demande à l'atom d'ouvrir le garage | `/etc/rc.local` (session tmux `button`) |
| `perm.list` | Badges autorisés, une ligne `ID pseudo` | — |

## Code source

Version en service sur le Pi (`/home/pi/rpi-wiegand-reader`). Les adresses sensibles sont masquées : `<url-webhook-framateam>` (webhook entrant Framateam), `<url-etat-ouvert>` / `<url-etat-ferme>` (page d'état du site) et `<chemin-local>` (route secrète du garage sur l'atom). Les vraies valeurs sont dans les fichiers sur le Pi.

`reader.c` (décodage Wiegand) est identique au dépôt d'origine.

### gate-service.sh : décision ouvrir / fermer

```
#!/bin/bash

cd $(dirname "$0")
LOG=./output
OPEN_DELAY=4

validate_id() {
    awk "/^$1 / { print \$2 }" ./perm.list
}

gate_error() {
    ./gate beepe
}

gate_open() {
    ./gate beepo
    ./gate open
    curl --connect-timeout 3 <url-etat-ouvert> || true
    curl --connect-timeout 3 -i -X POST -H 'Content-Type: application/json'
-d '{"text": "Local ouvert par @'"$1"' :)"}' <url-webhook-framateam> || true
}

gate_close() {
    ./gate beepc
    ./gate close
    curl --connect-timeout 3 <url-etat-ferme> || true
    curl --connect-timeout 3 -i -X POST -H 'Content-Type: application/json'
-d '{"text": "Local fermé par @'"$1"' :)"}' <url-webhook-framateam> || true
}

# at boot make sure we are closed
```

```
./gate close

while true; do
    inotifywait -qq -e modify "$LOG"
    date
    l=$(tail -n 1 "$LOG")
    if ! grep -q 'Read 26 bits' <<< "$l" ; then
        echo "invalid read: $l"
        continue
    fi
    c=$(awk '{ print $7 }' <<< "$l")
    if [ -z "$c" ] ; then
        echo "empty code"
        continue
    fi
    name=$(validate_id "$c")
    if [ -z "$name" ] ; then
        echo "unauthorized code $c"
        gate_error
        continue
    fi
    echo "identified $name"
    if inotifywait -qq -t "$OPEN_DELAY" -e modify "$LOG" ; then
        l=$(tail -n 1 "$LOG")
        if ! grep -q 'Read 26 bits' <<< "$l" ; then
            echo "invalid read 2: $l"
            gate_error
            continue
        fi
        c2=$(awk '{ print $7 }' <<< "$l")
        if [ "$c" != "$c2" ] ; then
            echo "code 1 doesn't match code 2"
            continue
        fi
        echo "calling open"
        gate_open "$name"
    else
        echo "calling close"
        gate_close "$name"
    fi
done
```

- Une ligne est valide si elle contient `Read 26 bits` ; l'identifiant du badge est le 7e champ.
- Après un badge autorisé, le script attend une 2e lecture pendant `OPEN_DELAY` (4 s) : même badge → ouverture, rien → fermeture.

### gate.c : commande de la serrure et des bips

Compilation : `gcc gate.c -o gate -lwiringPi -lpthread -lrt`. Les numéros de broches sont en numérotation **wiringPi** (15 = GPIO14, 16 = GPIO15, 7 = GPIO4).

```
/*
 * linked with -lpthread -lwiringPi -lrt
 */

#include <stdio.h>
#include <stdlib.h>
#include <wiringPi.h>
#include <time.h>
#include <unistd.h>
#include <memory.h>

#define PIN_C 15
#define PIN_O 16
#define PIN_B 7

int gate_close() {
    pinMode(PIN_C, OUTPUT);
    digitalWrite(PIN_C, 1);
    delay(1000);
    digitalWrite(PIN_C, 0);
    pinMode(PIN_C, INPUT);
    pullUpDnControl(PIN_C, PUD_OFF);
}

int gate_open() {
    pinMode(PIN_O, OUTPUT);
    digitalWrite(PIN_O, 1);
    delay(1000);
    digitalWrite(PIN_O, 0);
    pinMode(PIN_O, INPUT);
    pullUpDnControl(PIN_O, PUD_OFF);
}

/* close sound */
int gate_beepc() {
    pinMode(PIN_B, OUTPUT);
    digitalWrite(PIN_B, 0);
    delay(150);
    digitalWrite(PIN_B, 1);
    pinMode(PIN_B, INPUT);
    pullUpDnControl(PIN_B, PUD_OFF);
    delay(100);
    pinMode(PIN_B, OUTPUT);
    digitalWrite(PIN_B, 0);
    delay(150);
    digitalWrite(PIN_B, 1);
    pinMode(PIN_B, INPUT);
    pullUpDnControl(PIN_B, PUD_OFF);
}

/* open sound */
```

```
int gate_beepo() {
    pinMode(PIN_B, OUTPUT);
    digitalWrite(PIN_B, 0);
    delay(1000);
    digitalWrite(PIN_B, 1);
    pinMode(PIN_B, INPUT);
    pullUpDnControl(PIN_B, PUD_OFF);
}

/* error sound */
int gate_beepe() {
    for (int i = 0; i < 10; i++) {
        pinMode(PIN_B, OUTPUT);
        digitalWrite(PIN_B, 0);
        delay(50);
        digitalWrite(PIN_B, 1);
        pinMode(PIN_B, INPUT);
        pullUpDnControl(PIN_B, PUD_OFF);
        delay(50);
     }
}

/* garage sound */
int gate_beepg() {
    pinMode(PIN_B, OUTPUT);

#define TADA(on, off)  digitalWrite(PIN_B, 0); delay(on);
digitalWrite(PIN_B, 1); delay(off);
#define C TADA(80, 120)
#define CA TADA(120, 80)

    for(int i = 0; i < 4; i++) {
        CA C C
        CA C C
        CA C C
        CA C CA C
    }


    pinMode(PIN_B, INPUT);
    pullUpDnControl(PIN_B, PUD_OFF);
}

void main(int argc, char* argv[]) {
    wiringPiSetup() ;

    if (argc >=2 && strncmp(argv[1], "open", 4) == 0) {
        gate_open();
    } else if (argc >=2 && strncmp(argv[1], "beepc", 5) == 0) {
        gate_beepc();
    } else if (argc >=2 && strncmp(argv[1], "beepo", 5) == 0) {
```

```
        gate_beepo();
    } else if (argc >=2 && strncmp(argv[1], "beepe", 5) == 0) {
        gate_beepe();
    } else if (argc >=2 && strncmp(argv[1], "beepg", 5) == 0) {
        gate_beepg();
    } else {
        gate_close();
    }
}
```

### garage_button.c et garage_button.sh : bouton garage

garage_button utilise la numérotation **BCM** (wiringPiSetupGpio) : la broche 16 est GPIO16. Le programme se termine dès que le bouton est maintenu 0,5 s ; le script joue alors l'air beepg, attend 8 s et appelle l'atom.

```
/*
 * linked with -lpthread -lwiringPi -lrt
 */

#include <stdio.h>
#include <stdlib.h>
#include <wiringPi.h>
#include <time.h>
#include <unistd.h>
#include <memory.h>

#define PIN_BUTTON 16

void main(int argc, char* argv[]) {
    wiringPiSetupGpio() ;
    pinMode(PIN_BUTTON, INPUT);
    pullUpDnControl(PIN_BUTTON, PUD_UP);

    /* wait for the button to be released */
    while(digitalRead(PIN_BUTTON) == 0) {
    delay(100);
    }
    //puts("waiting for the button...");

    while(1) {
        if(digitalRead(PIN_BUTTON) == 0) {
            delay(500);
        if(digitalRead(PIN_BUTTON) == 0) {
            exit(0);
        }
        }
        delay(100);
    }
```

```
}
```

```
#!/bin/bash

cd /home/pi/rpi-wiegand-reader

while true; do
    ./garage_button
    ./gate beepg
    echo "button pressed, wwaiting for 15s..."
    sleep 8
    echo "opening garage..."
    curl -v http://192.168.42.1:4280/<chemin-local> || true
done
```

### Services systemd

```
# /etc/systemd/system/gate-service.service
[Unit]
Description=LOV gate service

[Service]
Type=simple
User=root
WorkingDirectory=/home/pi/rpi-wiegand-reader
ExecStart=/home/pi/rpi-wiegand-reader/gate-service.sh
Restart=on-failure

[Install]
WantedBy=multi-user.target

# /etc/systemd/system/wg-reader.service
[Unit]
Description=Wiegand badge reader

[Service]
Type=simple
User=root
WorkingDirectory=/home/pi/rpi-wiegand-reader
ExecStart=/home/pi/rpi-wiegand-reader/reader
Restart=on-failure

[Install]
WantedBy=multi-user.target
```

Le bouton garage n'est pas un service : il est lancé par `/etc/rc.local` avec `tmux new -s button -d /home/pi/rpi-wiegand-reader/garage_button.sh`.

### Commandes utiles

```
ssh pi@gatepi
sudo systemctl status gate-service wg-reader
journalctl -u gate-service -f        # suivre les badges en direct
cd ~/rpi-wiegand-reader
sudo ./gate open                     # ouvrir la porte (attention : réel)
sudo ./gate close                    # fermer la porte
sudo tmux attach -t button           # bouton garage (sortir : Ctrl-B puis
D)
```

### Ajouter un badge

1. Passer le badge sur le lecteur (10 bips courts : il est inconnu).
2. Sur le Pi : `journalctl -u gate-service | tail` → relever l'identifiant sur la ligne `unauthorized code <ID>`.
3. Ajouter la ligne `<ID> <pseudo>` dans `~/rpi-wiegand-reader/perm.list`. C'est pris en compte immédiatement.
4. Pour que le badge ouvre aussi le garage, ajouter la **même ligne** dans `~/garage/perm.list` sur l'atom (voir ouverture du garage) : les deux listes sont séparées.

## Points d'attention

- **Deux listes de badges** : celle du Pi (porte) et celle de l'atom (garage) sont tenues à la main. Le 13/10/2026, un badge n'était présent que sur le Pi.
- **Le Pi dépend du Wi-Fi** pour Frameteam, la page d'état et le bouton garage. Les badges, eux, fonctionnent sans réseau.
- **Code non sauvegardé** : plusieurs fichiers modifiés ou ajoutés ne sont pas versionnés. Avant de les pousser sur un dépôt, retirer l'URL du webhook Frameteam de `gate-service.sh`.
- Le fichier `output` grossit sans fin (1,6 Mo) : prévoir une rotation.
- Le Pi est réglé sur l'heure de **Londres** : les journaux ont une heure de retard. Correction : `sudo timedatectl set-timezone Europe/Paris`.

## À vérifier sur place

- Modèle du lecteur Wiegand et tension de l'alimentation en entrée du convertisseur DC-DC.
- Ce que commandent GPIO14/15 (entrées filaires de la serrure, relais ou transistors). La serrure a une antenne : l'Arduino du boîtier garage, qui commande une « porte arrière » par la télécommande de la serrure (D10/D11), pilote probablement cette même serrure par radio. À confirmer.
- Présence de l'adaptation 5 V → 3,3 V sur Data0/Data1.
- Fil jaune (bip) : sur GPIO26 (broche 37), GPIO4 (broche 7), ou les deux ?
- Confirmer que le bouton du flanc droit est bien relié à GPIO16 (broche 36).

From:
https://labovilleurbanne.fr/dokuwiki/ - **DokuWiki du LOV**

Permanent link:
**https://labovilleurbanne.fr/dokuwiki/asso:si:serrure?rev=1791843357**

Last update: **2026/10/12 22:15**