

Système d'ouverture du garage et de la porte arrière

Le boîtier rose fixé près du garage contient un **Arduino Uno** (clone AZ-Delivery), une **télécommande 433 MHz** de la porte de garage et le **lecteur de badges RFID**. Il est branché en USB sur le serveur **Atom** (192.168.42.1), qui le pilote de deux façons :

- **Framateam (Mattermost)** : commandes slash dans les canaux autorisés → ouvrir le garage, ouvrir / fermer la porte arrière ;
- **Badge RFID** : un badge de la liste ouvre le **garage** uniquement.

```

Framateam —HTTP :4280→ nginx → server.py (Flask, 127.0.0.1:8001) —USB
série 9600→ Arduino → télécommande garage (D9)
Badge → lecteur RFID USB → rfid.py → rfid.log → gate-service.sh
(perм.list) └┐ └┐ télécommande serrure (D10 / D11)

```

Électronique

LOV garage opener v3 — boîtier porte de garage + serrure
 Arduino Uno (clone AZ-Delivery, CH340) relié en USB au serveur Atom (série 9600 bauds).
 Commandes : O = garage [D9 à la masse 800 ms] · E = ouvrir porte (D10 HIGH 2,5 s) · L = fermer porte (D11 HIGH 2,5 s).
 Câblage reconstitué (firmware + photos, 10/10/2026). Valeurs de R1–R3 et étage serrure à vérifier.

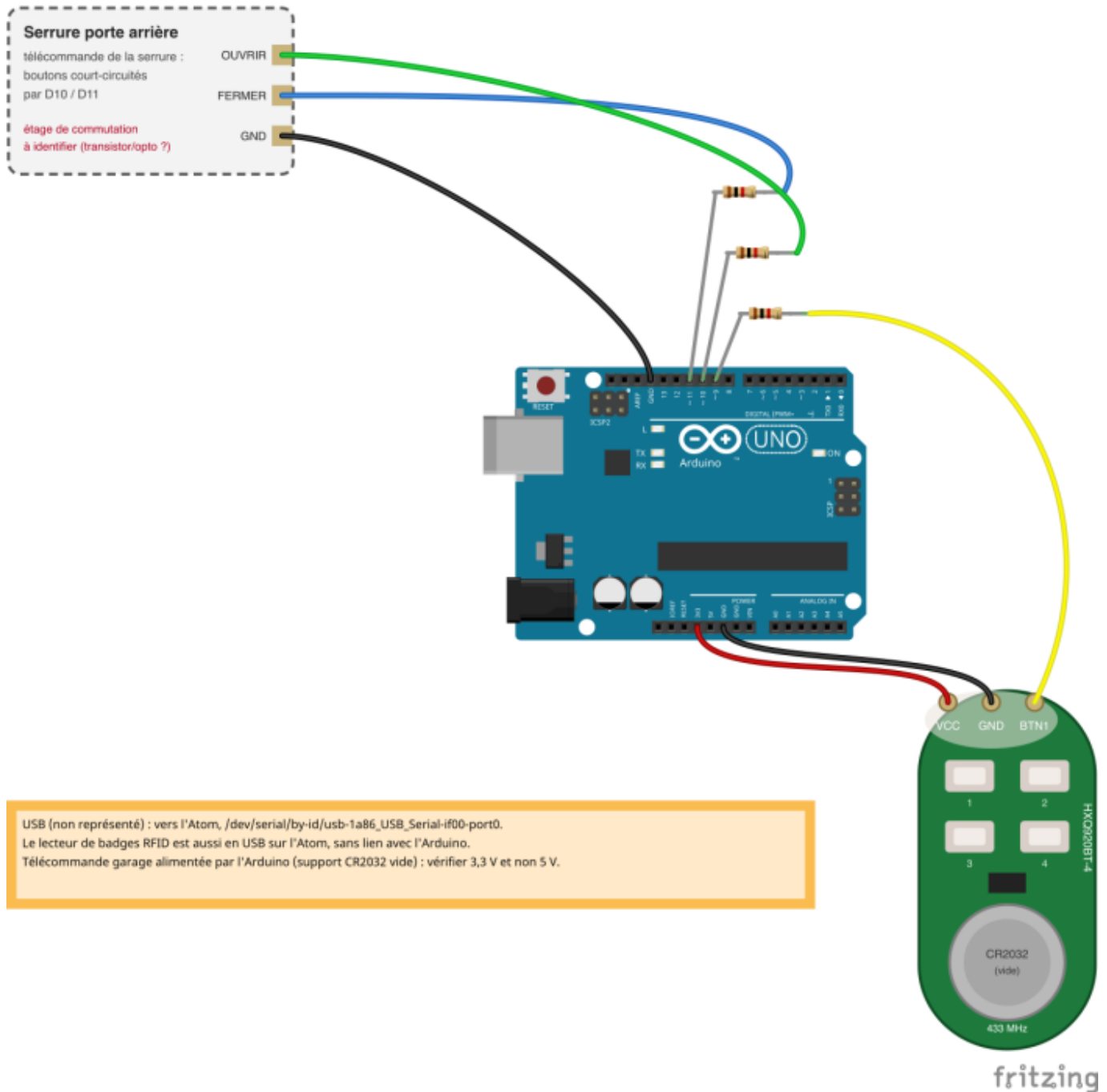
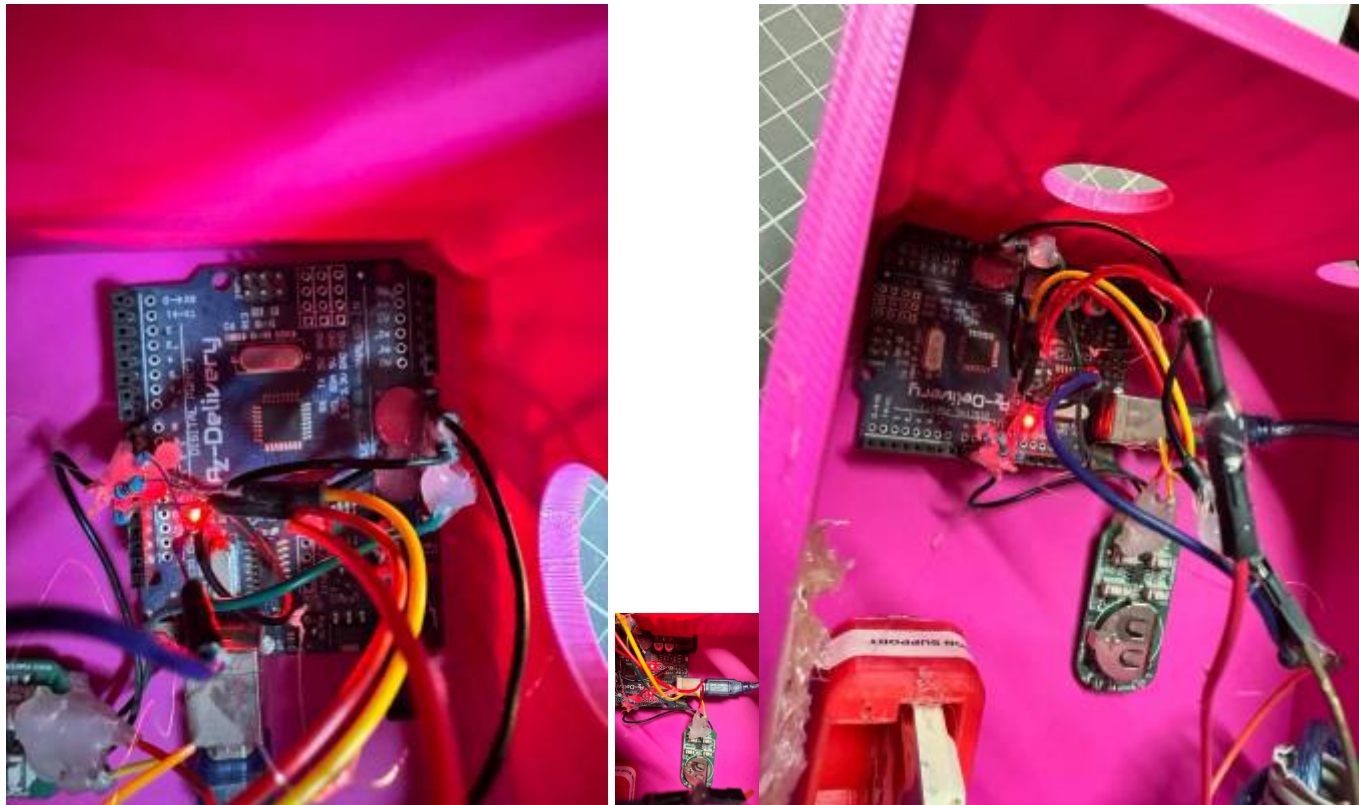


Schéma Fritzing modifiable : [boitier_garage_fritzing.zip](#) (contient boitier-garage.fzz, à ouvrir avec Fritzing)



Broche Arduino	Fonction	Fonctionnement
D9	Bouton de la télécommande du garage	Mise à la masse pendant 800 ms = appui (ouvre ou ferme)
D10	Télécommande de la serrure : ouvrir	Impulsion 5 V de 2,5 s
D11	Télécommande de la serrure : fermer	Impulsion 5 V de 2,5 s
USB	Liaison avec l'Atom	Série 9600 bauds, puce CH340

La télécommande du garage (PCB vert HXQ920BT-4) n'a plus de pile : elle est alimentée par l'Arduino. Les boutons de la télécommande de la serrure sont court-circuités par D10 / D11.

Anciennement, un servomoteur appuyait physiquement sur le bouton de la télécommande ; il a été remplacé par la commande électrique de D9 (code servo conservé en commentaire ci-dessous).

Firmware Arduino (LOV garage opener v3)

Protocole : un caractère sur le port série (9600 bauds). 0 = garage, E = ouvrir la porte arrière, L = fermer la porte arrière. Tout autre caractère est ignoré. Au démarrage, l'Arduino affiche LOV garage opener v3.

```
#include <Servo.h>

Servo servo;

void setup() {
  pinMode(10, OUTPUT);
  digitalWrite(10, LOW);
  pinMode(11, OUTPUT);
}
```

```
digitalWrite(11, LOW);

Serial.begin(9600);
Serial.println("LOV garage opener v3");

/*servo.attach(9);
servo.write(0);
delay(400);
servo.detach();*/
pinMode(9, INPUT);
digitalWrite(9, HIGH);
}

void loop() {
  int c = 0;
  c = Serial.read();
  if (c == 'O') {
    Serial.println("opening garage...");
    /*servo.attach(9);
    servo.write(30);
    delay(1000);
    servo.write(0);
    delay(400);
    servo.detach();*/
    pinMode(9, OUTPUT);
    digitalWrite(9, LOW);
    delay(800);
    pinMode(9, INPUT);
    digitalWrite(9, HIGH);
    Serial.println("done");
  } else if (c == 'E') {
    //Serial data for shorting the remote controller of backdoor
    Serial.println("opening back door...");
    digitalWrite(10, HIGH);
    delay(2500);
    digitalWrite(10, LOW);
    Serial.println("done");
  } else if (c == 'L') {
    //Serial data for shorting the remote controller of backdoor
    Serial.println("closing back door...");
    digitalWrite(11, HIGH);
    delay(2500);
    digitalWrite(11, LOW);
    Serial.println("done");
  }
}
```

Vérifié le 10/10/2026 : ce code correspond au firmware présent sur la carte.

Serveur (Atom)

Pré-requis :

```
apt install python3-flask python3-serial python3-evdev inotify-tools tmux
nginx
adduser lov dialout
```

Fichiers dans /home/lov/garage :

Fichier	Rôle
server.py	Serveur Flask (127.0.0.1:8001) qui envoie O / E / L à l'Arduino (/dev/serial/by-id/usb-1a86_USB_Serial-if00-port0)
gate-service.sh	Surveille rfid.log, cherche le badge dans perm.list et ouvre le garage
rfid.py	Lit le lecteur RFID USB et ajoute chaque badge lu à rfid.log
perm.list	Badges autorisés, une ligne ID pseudo
rfid.log	Journal de tous les badges passés

nginx écoute sur le port **4280** et ne transmet au serveur Flask que les URL secrètes (voir /etc/nginx/sites-enabled/default sur l'Atom — ne pas les recopier sur le wiki) :

Route	Déclenchée par	Action	Contrôle
ouverture locale (GET)	gate-service.sh (badge)	garage	aucun : réservé à l'usage local
garage (POST)	commande slash Framateam	garage	canal garage ou canal porte
entrer (POST)	commande slash Framateam	ouvre la porte arrière + message « X a ouvert le local »	canal porte
sortir (POST)	commande slash Framateam	ferme la porte arrière + message « X a fermé le local »	canal porte

Code du serveur (server.py)

Version en service sur l'atom (/home/lov/garage/server.py). Les chemins d'URL et les identifiants de canaux Framateam sont **secrets** : ils sont remplacés ci-dessous par <chemin-...> et <id-canal-...>. Les vraies valeurs sont dans le fichier sur l'atom.

```
from flask import Flask, request, jsonify
app = Flask(__name__)
import serial

ARDUINO = '/dev/serial/by-id/usb-1a86_USB_Serial-if00-port0'
GARAGE_CHANNEL = '<id-canal-garage>'
PORTE_CHANNEL = '<id-canal-porte>'

def send_command(c):
    try:
```

```
    with serial.Serial(port=ARDUINO, baudrate=9600) as f:
        banner = f.readline().decode('ascii')
        if 'LOV' not in banner:
            return "didn't find the garage opener banner"
        f.write(c)
        f.flush()
        resp = f.readline().decode('ascii')
        return "\n".join((banner, resp))
except serial.SerialUtil.SerialException as e:
    return str(e)

# unconditionnally open the garage (for local usage)
@app.route('/<chemin-local>', methods=['GET'])
def local_open():
    return send_command(b'O')

@app.route('/<chemin-garage>', methods=['POST'])
def open():
    user_name = request.form.get('user_name', 'unknown user')
    channel_id = request.form.get('channel_id', '')
    if channel_id != GARAGE_CHANNEL and channel_id != PORTE_CHANNEL:
        return "this is not the command you are looking for"
    return send_command(b'O')

@app.route('/<chemin-entrer>', methods=['POST'])
def enter():
    user_name = request.form.get('user_name', 'unknown user')
    channel_id = request.form.get('channel_id', '')
    if channel_id != PORTE_CHANNEL:
        return "this is not the command you are looking for"
    r = send_command(b'E')
    return jsonify({
        "response_type": "ephemeral",
        "text": r,
        "extra_responses": [{
            "response_type": "in_channel",
            "text": "%s a ouvert le local" % user_name,
            "channel_id": "<id-canal-annonces>"
        }]
    })

@app.route('/<chemin-sortir>', methods=['POST'])
def leave():
    user_name = request.form.get('user_name', 'unknown user')
    channel_id = request.form.get('channel_id', '')
    if channel_id != PORTE_CHANNEL:
        return "this is not the command you are looking for"
    r = send_command(b'L')
    return jsonify({
        "response_type": "ephemeral",
        "text": r,
```

```

        "extra_responses": [{
            "response_type": "in_channel",
            "text": "%s a fermé le local" % user_name,
            "channel_id": "<id-canal-annonces>"
        }]
    })

if __name__ == '__main__':
    app.run(host='127.0.0.1', port=8001)

```

Fonctionnement :

- send_command ouvre le port série de l'Arduino. L'ouverture **redémarre l'Arduino**, qui envoie sa bannière LOV garage opener v3 : le serveur la lit pour vérifier qu'il parle bien au boîtier, envoie la lettre (O, E ou L) puis lit la réponse (opening ...). Compter environ 2 secondes par commande.
- local_open (GET) : ouvre le garage **sans aucune vérification**. Elle n'est utilisée que par le lecteur de badges, sur l'atom lui-même.
- open, enter, leave (POST) : appelées par les commandes slash de Framateam. Mattermost envoie user_name et channel_id ; la commande n'est acceptée que depuis les canaux prévus.
- enter et leave répondent en JSON : un message visible seulement par l'auteur (ephemeral) et une annonce publique « X a ouvert / fermé le local » dans le canal d'annonces.
- Le serveur n'écoute que sur 127.0.0.1:8001 : c'est **nginx** (port 4280) qui le rend joignable, uniquement pour les quatre chemins prévus.

Code du lecteur de badges (gate-service.sh)

```

#!/bin/bash

cd $(dirname "$0")
LOG=./rfid.log

(while true ; do python3 rfid.py ; sleep 1 ; done) &

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

while true; do
    inotifywait -qq -e modify "$LOG"
    date
    l=$(tail -n 1 "$LOG")
    if ! egrep -q '^[0-9A-Z]{8}' <<< "$l" ; then
        echo "invalid read: $l"
        continue
    fi
    c=$(cut -d ' ' -f 2 <<< "$l")
    name=$(validate_id "$c")
    if [ -z "$name" ] ; then

```

```
    echo "unauthorized code $l"
    continue
fi
echo "identified $name, opening garage..."
curl http://192.168.42.1:4280/<chemin-local> || true
done
```

Fonctionnement :

- lance `rfid.py` en boucle (il redémarre tout seul si le lecteur est débranché puis rebranché) ;
- `rfid.py` lit le lecteur USB, qui se comporte comme un clavier, convertit le numéro du badge au format de l'ancien lecteur (8 caractères) et ajoute une ligne à `rfid.log` ;
- à chaque nouvelle ligne de `rfid.log` (surveillée par `inotifywait`), le script cherche l'identifiant dans `perm.list` ;
- si le badge est autorisé, il appelle la route locale du serveur (`<chemin-local>`) : le garage s'ouvre.

Lancement au démarrage, dans `/etc/rc.local` :

```
tmux new -s garage -d 'sudo -u lov -g lov /usr/bin/python3
/home/lov/garage/server.py'
tmux new -s rfid -d /home/lov/garage/gate-service.sh
```

Pour voir ce qui se passe : `sudo tmux attach -t garage` ou `sudo tmux attach -t rfid` (sortir avec `Ctrl-B` puis `D`).

Ajouter un badge

1. Passer le badge sur le lecteur.
2. Sur l'Atom : `tail ~/garage/rfid.log` → noter le 2e champ (8 caractères, ex. A1B2C3D4).
3. Ajouter la ligne `<ID> <pseudo>` dans `~/garage/perm.list`. C'est pris en compte immédiatement.

Dépannage

- `didn't find the garage opener banner` : l'Arduino ne répond pas → vérifier le câble USB, puis `screen /dev/serial/by-id/usb-1a86_USB_Serial-if00-port0 9600` doit afficher la bannière.
- `could not open port` : Arduino débranché (`ls /dev/serial/by-id/`).
- Le badge ne fait rien : lecteur débranché (`lsusb`), ou badge absent de `perm.list`.
- L'Arduino répond `done` mais la porte ne bouge pas : vérifier la télécommande (alimentation, bouton) — `done` veut dire « impulsion envoyée », pas « porte ouverte ».

Documentation technique détaillée (rétro-ingénierie, schéma, sauvegarde du firmware) : dépôt AdminLova, `docs/boitier-garage/`.

From:

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

Permanent link:

https://labovilleurbanne.fr/dokuwiki/asso:si:si_garage?rev=1791841029

Last update: **2026/10/12 21:37**

