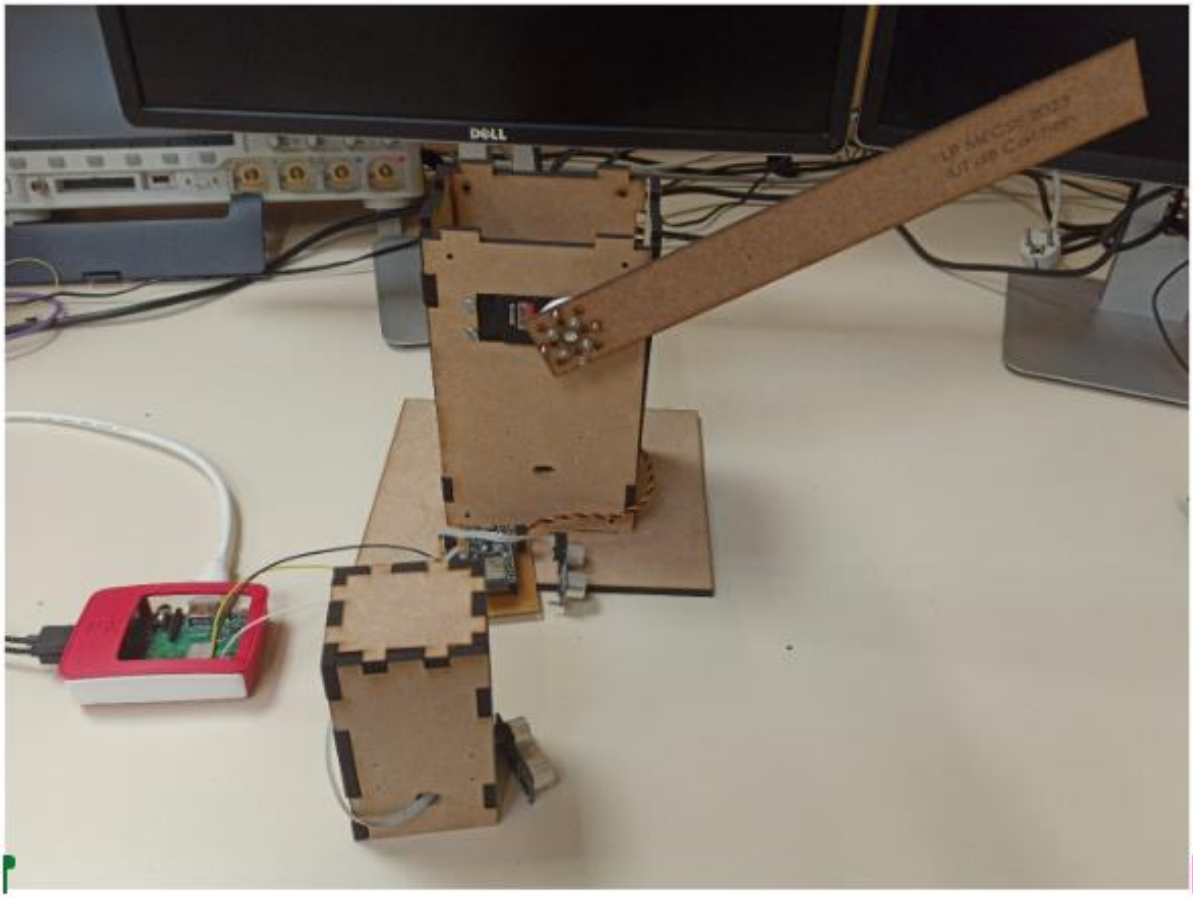
université
PARIS-SACLAY

IUT DE CACHAN

Rapport de projet

Gestion de parking

Promotion 2022/2023



Établissement : IUT de Cachan

Nom des étudiants : Alexis POUILLAT, Simon WARIN

université
PARIS-SACLAY

Table des matières

université
PARIS-SACLAY

Remerciements

Tout d'abord, nous tenons fortement à remercier Olivier CLAMENS, Xavier MININGER, Patrick RUIZ et Emile MARTINCIC, nos tuteurs de projet qui nous ont encadrés et aidés, tout en partageant leurs connaissances sur ce projet passionnant qu'est la barrière automatique.

Nous voulons également remercier Marc ARDILLIER pour son aide et ses conseils dans la découpe des pièces 3Ds modélisées sur SolidWorks et des cartes électroniques.

Pour finir, nous voulons également remercier Nathalie BRISSARD, enseignante de culture communication pour sa disponibilité, son implication dans la réussite de ces travaux, ses conseils pertinents et pour ses encouragements tout au long de cette année.

Introduction

L'objet de notre projet a porté sur la conception d'une barrière de parking automatisé miniature à l'échelle 1/10. Le fonctionnement de la barrière est simple nous disposons d'un capteur à ultrasons placé en amont de la barrière qui permet de détecter la présence d'un véhicule. Une fois le véhicule détecté notre caméra prend une photo de la barrière puis à l'aide d'une bibliothèque OCR on vient lire la plaque d'immatriculation de la barrière pour la comparer à notre base de donnée pour savoir si ce véhicule rentre ou sort du parking. Si le véhicule rentre dans le parking, nous retenons son heure d'entrée afin de lui facturer le bon prix au moment de sa sortie. La barrière s'ouvre à l'aide d'un servomoteur. Nous disposons d'un second capteur à ultrasons sous la barrière pour refermer la barrière une fois que le véhicule est rentré.

I. Présentation du projet

1. Schéma explicatif

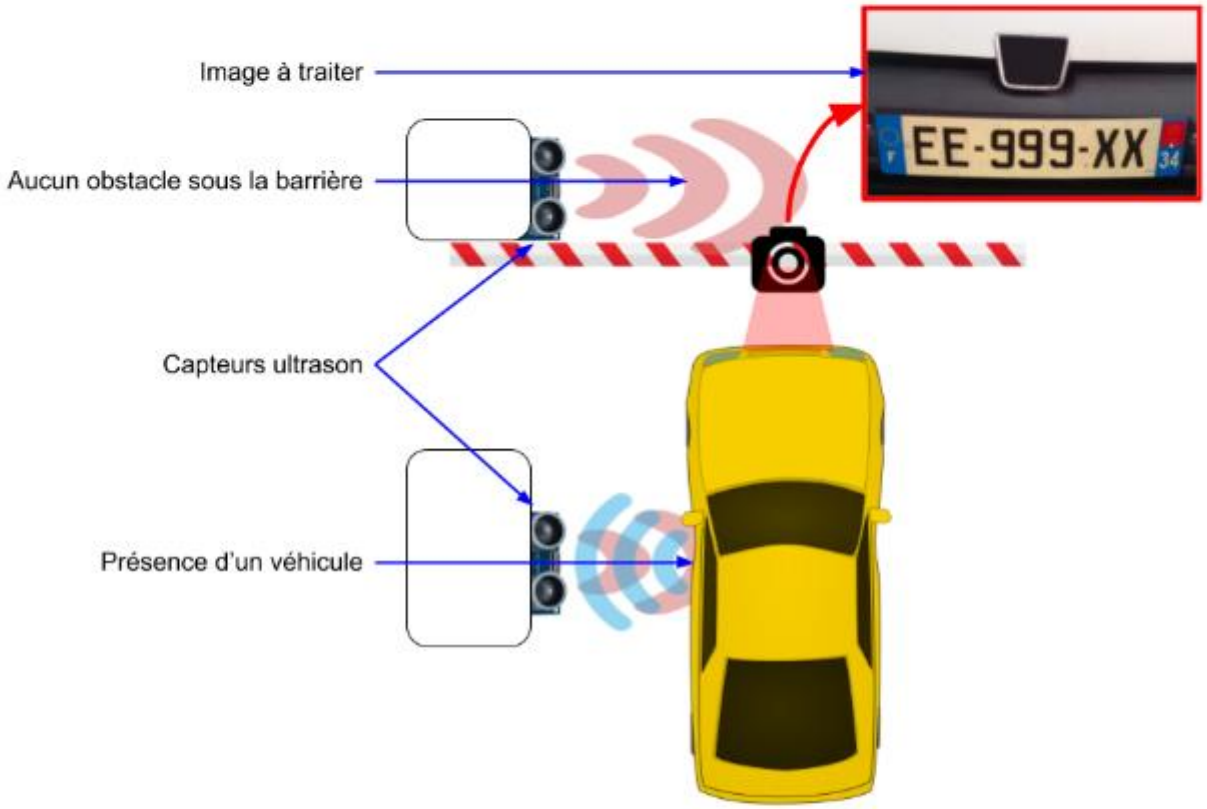


Schéma explicatif du projet

Sur ce schéma, nous pouvons voir un véhicule essayant de passer la barrière. Le véhicule est détecté par le capteur ultrason qui est devant la barrière. Cela déclenche la prise d'une photo par la caméra qui est sur la barrière. Sur cette photo, on doit trouver une plaque d'immatriculation d'un véhicule autorisé. Si c'est ce que l'on détecte, la barrière s'ouvre. La barrière se fermera seulement après qu'un obstacle ait été détecté sous la barrière (la voiture en train de passer) puis qu'il ait disparu. Enfin, le système attend qu'un véhicule essaye de passer la barrière pour recommencer le cycle.

2. Schéma synoptique

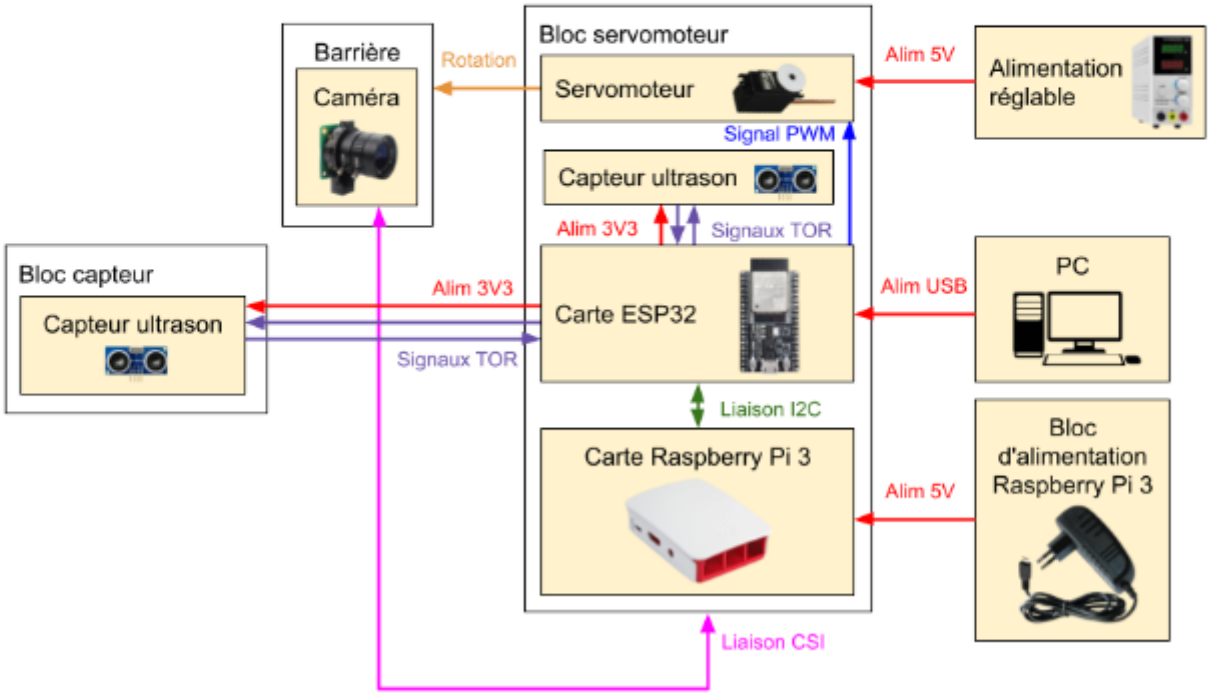


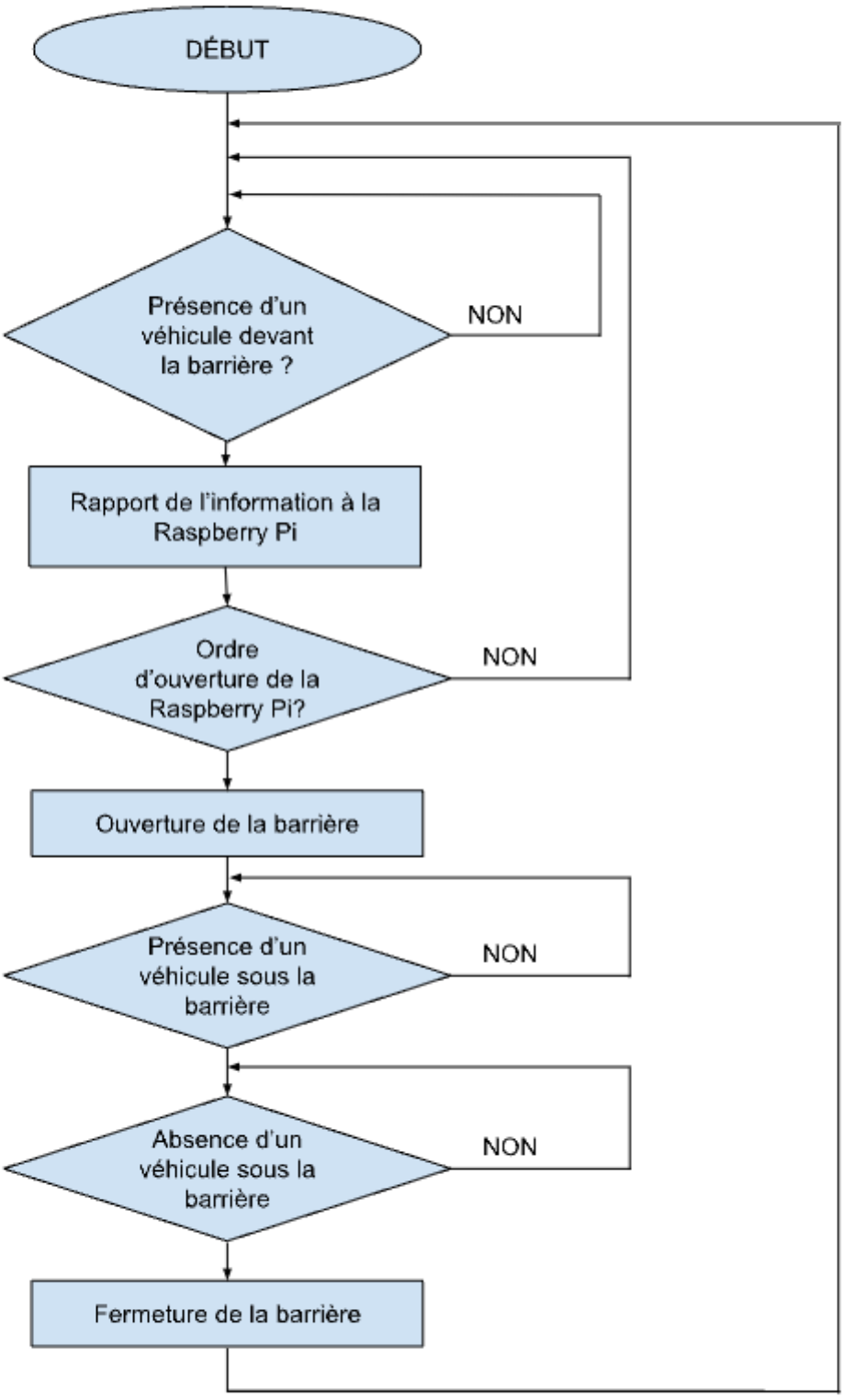
Schéma synoptique du projet

On observe sur ce schéma que tous les signaux sont reliés à la carte ESP32 ou la carte Raspberry Pi, à l'exception de l'alimentation du servomoteur qui est gérée par une alimentation réglable. Nous pouvons donc diviser le schéma en deux parties : la partie ESP32 et la partie Raspberry Pi.

La partie ES32 est responsable de la gestion et de l'alimentation des deux capteurs ultrason. Chacun des deux capteurs ultrason est géré en envoyant un signal TOR (le signal *trig*) et en attendant un autre (le signal *echo*). Le servomoteur est commandé avec un signal PWM. La carte ESP32 est alimentée grâce à un câble micro USB relié à un PC, permettant de gérer un terminal série sur ce PC pour afficher des informations de l'ESP 32 (par exemple s' il y a un ordre d'ouverture de la barrière). Enfin, il y a une liaison I2C avec la carte Raspberry Pi permettant d'envoyer des informations comme la détection d'un véhicule devant la barrière.

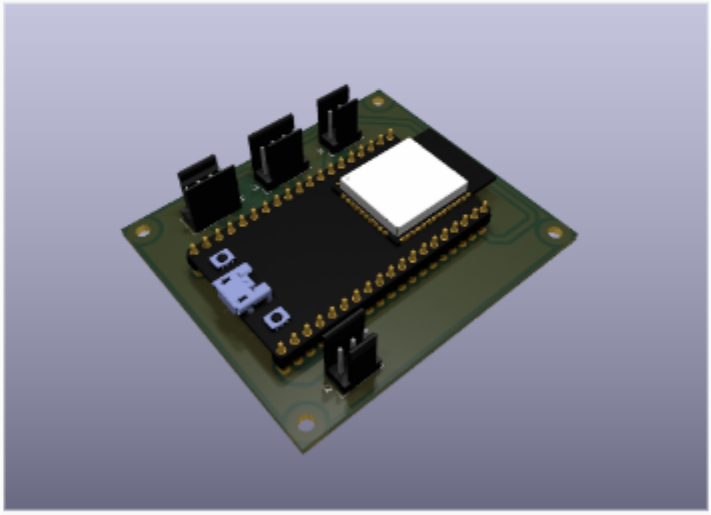
La partie Raspberry Pi est responsable de la gestion et de l'alimentation de la caméra. À l'aide d'une liaison CSI, la RaspBerry demande à la caméra de prendre une photo puis récupère l'image et la traite. La Raspberry Pi est alimentée par un bloc d'alimentation Raspberry Pi.

II. Partie gestion du la barrière et des capteurs ultrasons (ESP32)



Algorigramme de la partie gestion du la barrière et des capteurs ultrasons

Nous avons dû concevoir une carte de cuivre à une couche afin d'avoir de port mâles permettant de relier l'ESP32 aux deux capteurs ultrasons, au servomoteur et à la Raspberry Pi. Nous avons utilisé le logiciel KiCad pour concevoir les schéma de cette carte puis nous l'avons imprimée et soudée. Les schéma de cette carte se trouvent en annexe.

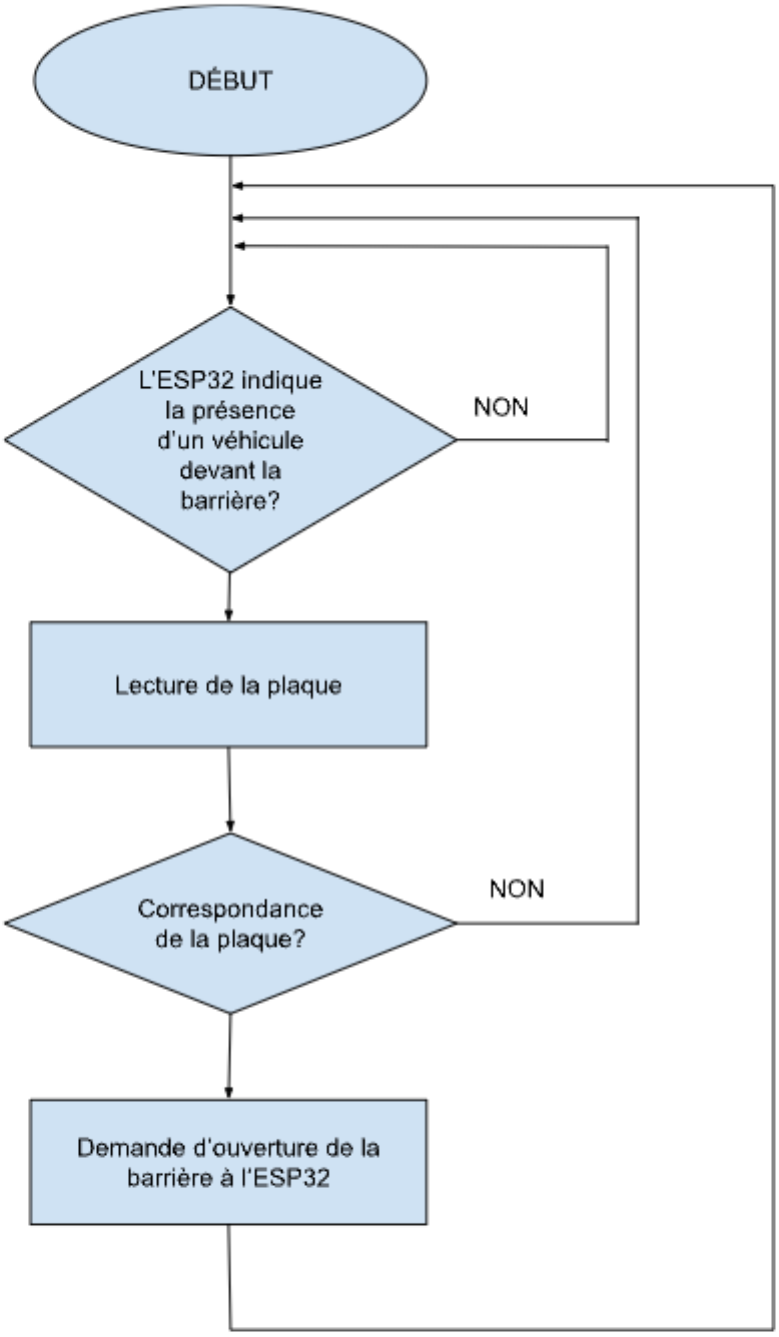


Rendu 3D de la carte concue pour accompagner l'ESP32

Nous avons codé la carte en langage C avec le logiciel Visual Studio Project afin de pouvoir exécuter le fonctionnement décrit par l'algorigramme. Le programme commenté se trouvé sur le lien GitHub du projet qui est en annexe.

Cependant, à cause de retard sur le projet sur la mise en place de la caméra et la Raspberry Pi, nous n'avons pas pu terminer la partie demandant à la RaspBerry Pi si la plaque d'immatriculation est correcte avant d'ouvrir la barrière. À la place, la barrière s'ouvre dès qu'elle détecte un véhicule devant elle.

III. Partie gestion de la caméra (Raspberry Pi 3)



Algorigramme de la partie gestion de la caméra

```python
from PIL import Image
import pytesseract
import re
import smbus2

bus = smbus2.SMBus(1)
bus.write_byte(0x54, 0xA)
data = bus.read_byte(0x54)
print(data)
```

code python I2C

Mise en place de l'I2C côté raspberry pi. Pour mettre en place l'I2C nous avons décidé d'utiliser la bibliothèque smbus2 qui permet d'envoyer un octet à l'esclave souhaité à l'aide de son adresse puis on attend son retour d'informations.

```python
import cv2
from PIL import Image
import pytesseract
import re

# filtre les caracteres non souhaité
def filtre(chaine):
    chaine_filtrer = re.sub(r'[^A-Z0-9]','',chaine)
    return chaine_filtrer


#lis les caracteres sur l'image
def lecture(image_path):
    text = pytesseract.image_to_string(image_path)
    return text


cap = cv2.VideoCapture(0) # video capture source camera (Here webcam of laptop)
ret,frame = cap.read() # return a single frame in variable `frame`


while(True):
    cv2.imshow('plaque',frame) #display the captured image
    if cv2.waitKey(1) & 0xFF == ord('y'): #save on pressing 'y'
        cv2.imwrite('plaque.png',frame)
        cv2.destroyAllWindows()
        break

cap.release()

img = Image.open('plaque.png')
result = lecture(img)
print(result)
plaque = filtre(result)
print(plaque)
```

code python lecture plaque

Ci-dessus nous pouvons voir le code d'acquisition de la plaque d'immatriculation. Nous utilisons la bibliothèque cv2 pour prendre la photo de la plaque. Puis nous lisons les caractères de plaque d'immatriculation à l'aide de la bibliothèque OCR pytesseract.

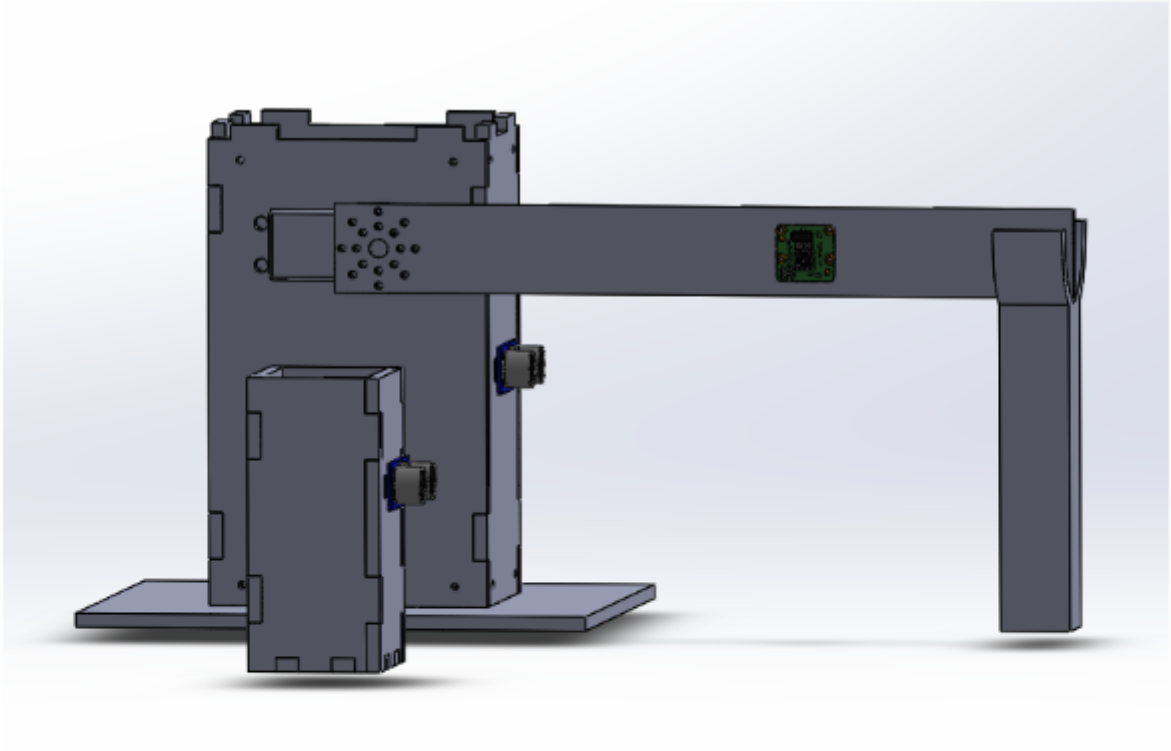

```
>>> %Run test2.py
PL-123-AKiz
▫
PL123AK
```

retour code lecture plaque

Nous pouvons voir ce que nous renvoie ce code sur la première ligne on peut observer qu'il y a des caractères non désirée à cause du département que nous avons filtré à l'aide de la fonction "filtre" ce qui nous donne PL123AK a la 3ème ligne.

IV. Conception mécanique

Pour modéliser notre barrière nous avons décidé d'utiliser le logiciel SOLIDWORKS et de la réaliser en bois découpé au laser afin de réaliser un assemblage robuste.



Rendu 3D de la carte concue pour accompagner l'ESP32

Voici un aperçu final de la barrière: cet assemblage permet de voir si les pièces s'emboîtent bien toutes ensemble et si la place à l'intérieur de la barrière est suffisante pour la carte électronique et la raspberry pi.

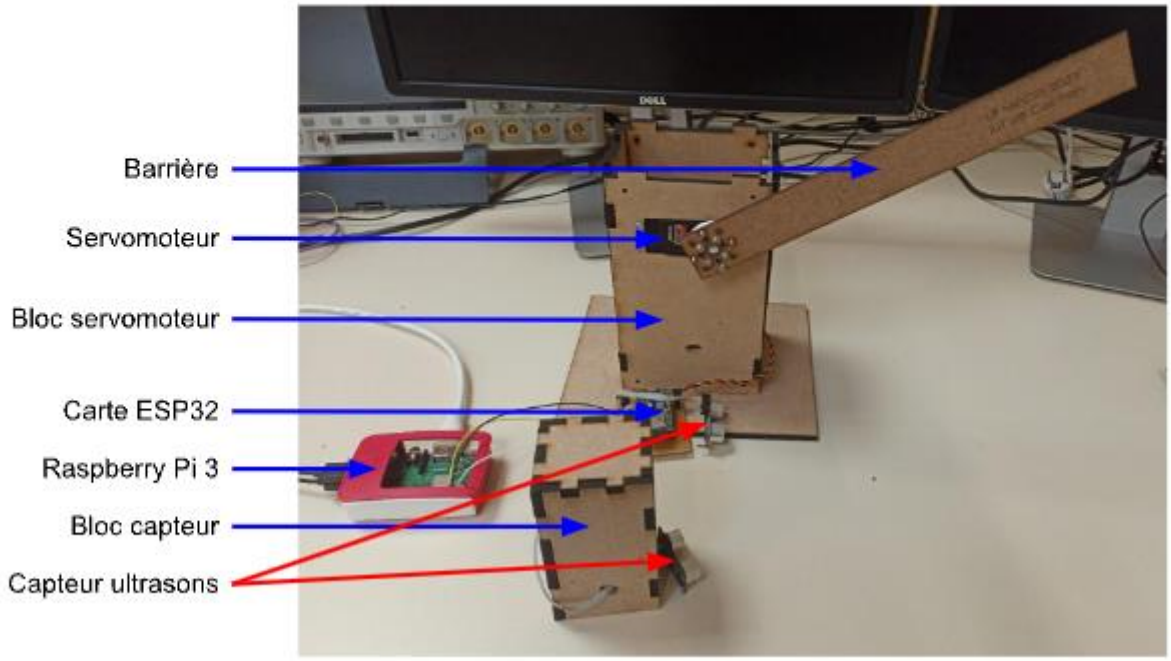


Photo de l'assemblage mécanique

On peut voir le rendu de la barrière final une fois découpé assemblé avec les différents capteur et actionneur placé il faut seulement cacher la raspberry pi et la carte électronique dans la barrière.

université
PARIS-SACLAY

Conclusion

Nous avons réussi à créer une barrière pouvant s'ouvrir à la détection d'un véhicule s'arrêtant devant et se fermer une fois qu'il est passé grâce à une carte ESP32. Nous avons aussi réussi à créer un système pouvant lire ce qui est écrit sur une plaque d'immatriculation à l'aide d'une caméra et d'une Raspberry Pi et pouvant dire s' il s'agit d'une plaque d'immatriculation dont nous voulons ou non.

Ce projet nous a permis de gagner en expérience sur les logiciels KiCad, SolidWorks et Visual Studio Project mais aussi sur la programmation des cartes ESP32 et sur la programmation Python des cartes Raspberry Pi.

Cependant, il nous reste encore à assembler les deux systèmes réalisés de manière à ce que la carte ESP32 demande à la Raspberry Pi si la plaque d'immatriculation du véhicule devant la barrière est correcte avant de l'ouvrir.

Annexe

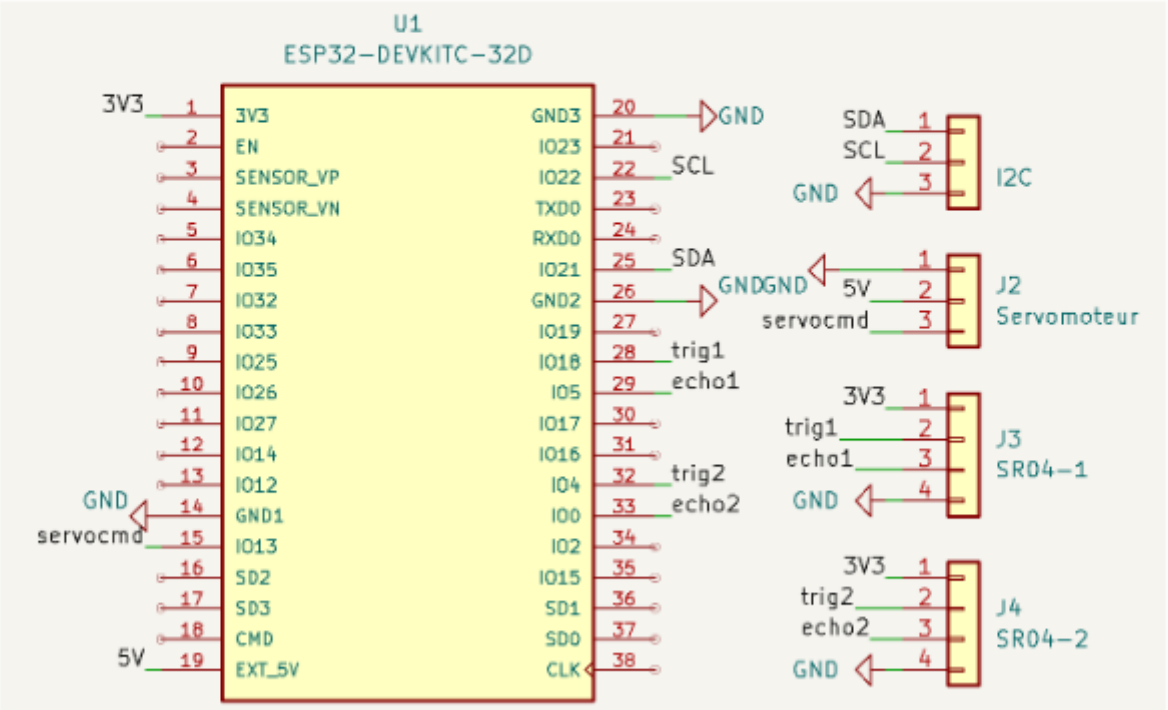


Schéma de symbole de la carte associée à l'ESP32 avant son impression

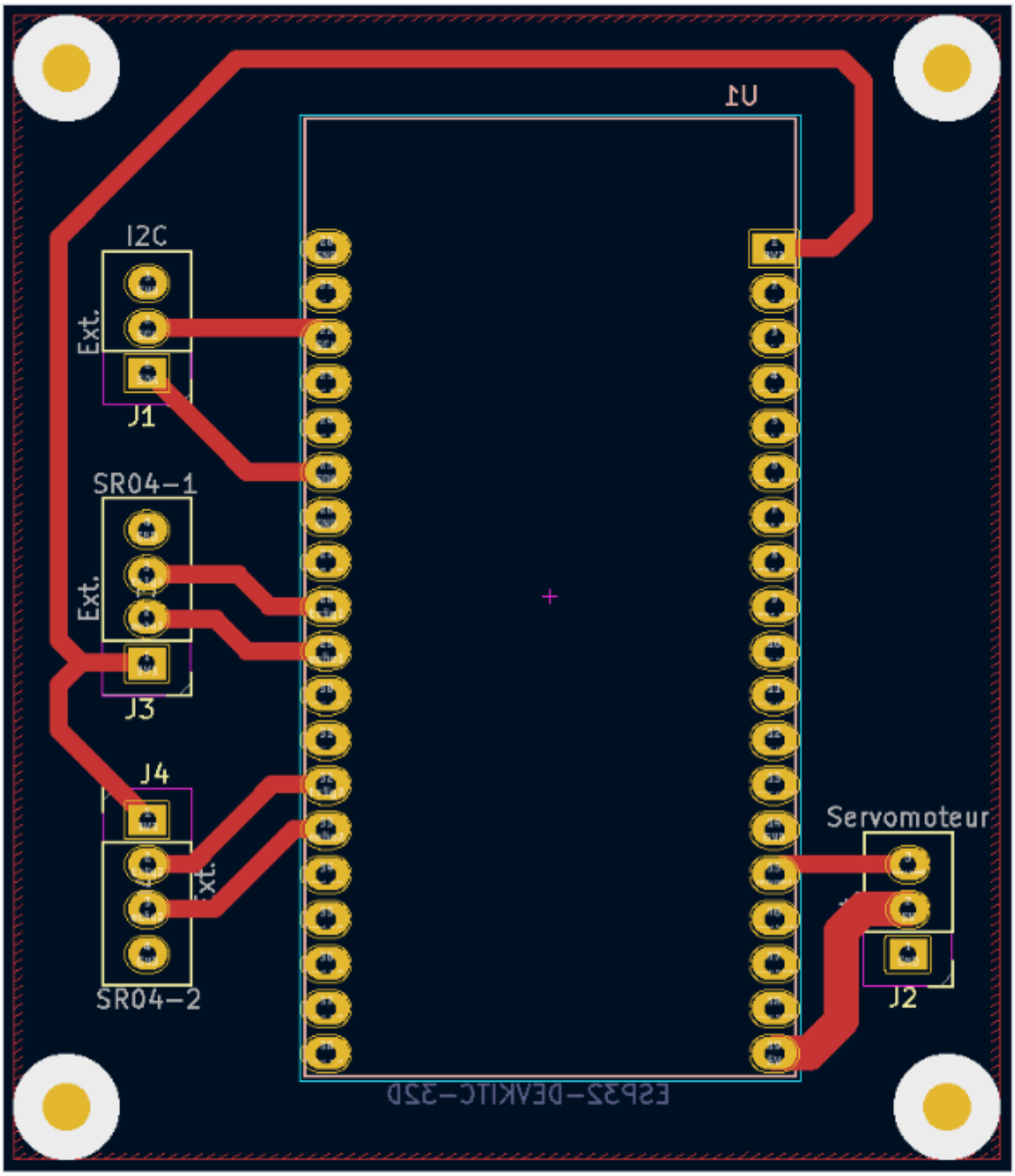


Schéma de la carte associée à l'ESP32 avant son impression

Lien GitHub :

https://github.com/SimonWARIN/gestion_de_parking_POUILLAT_WARIN_2023.git