Smart Parking

Group 4: Sacha Katharine Côme Nans

Summary

- Project introduction
  - Architectures
  - Functions
- Implementation
  - Scheduling
  - Sensors
  - Licence plate reader
  - Server
- Conclusion

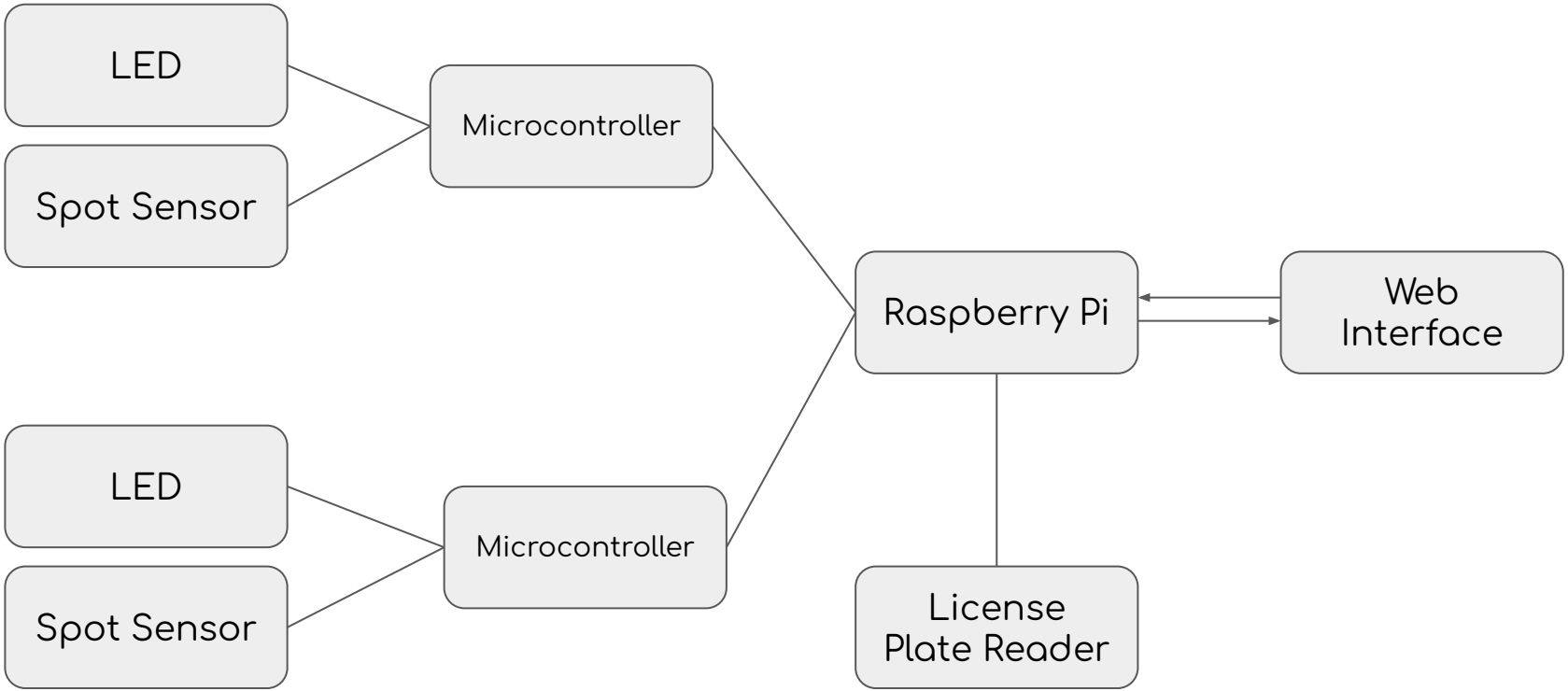
Demo

Project introduction

Smart Parking allowing users to :

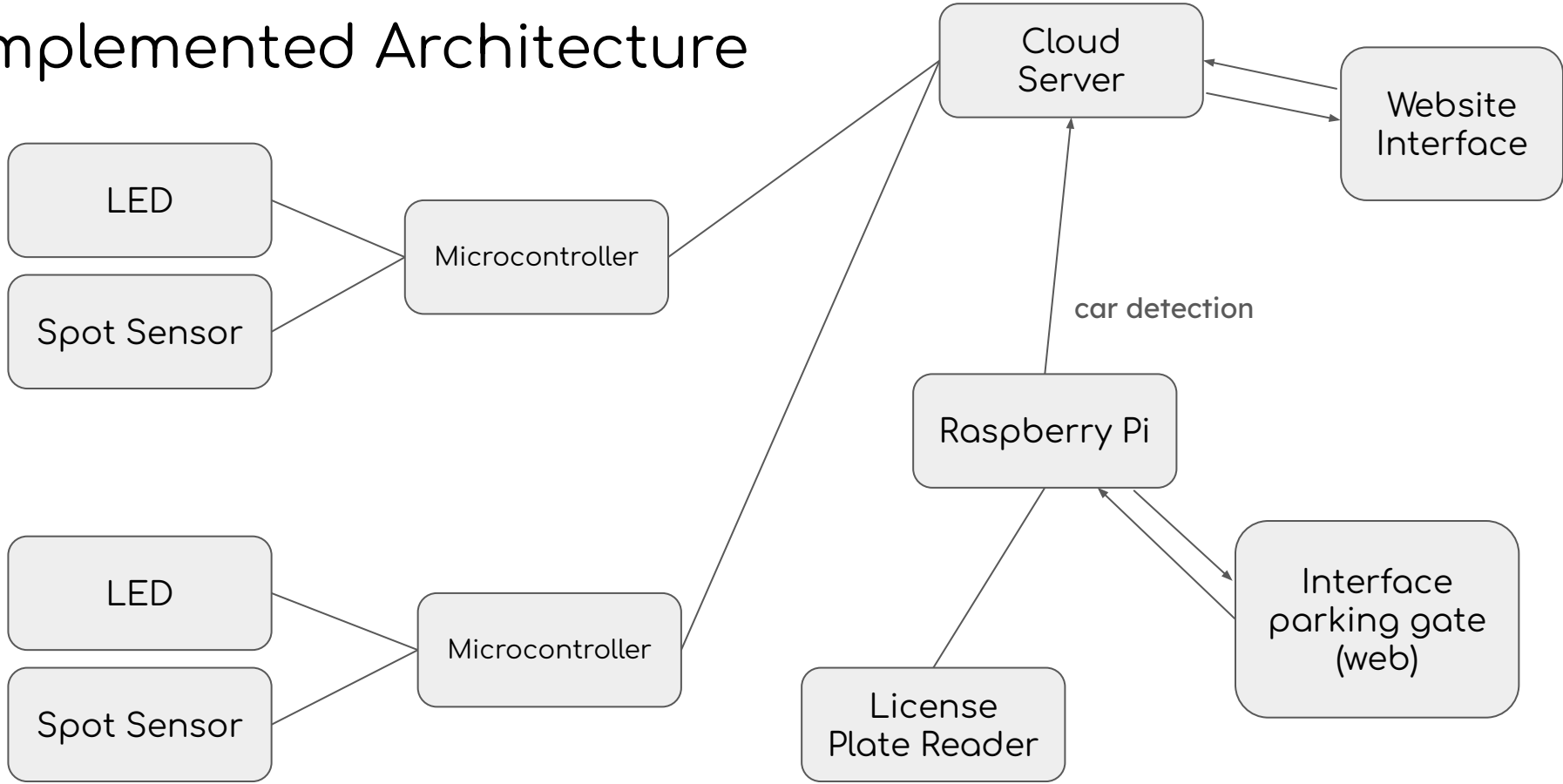
- Check parking availability
- Reserve a spot in advance
- Get guidance to the selected spot
- License plate reader for seamless entry and billing

Proposal Architecture

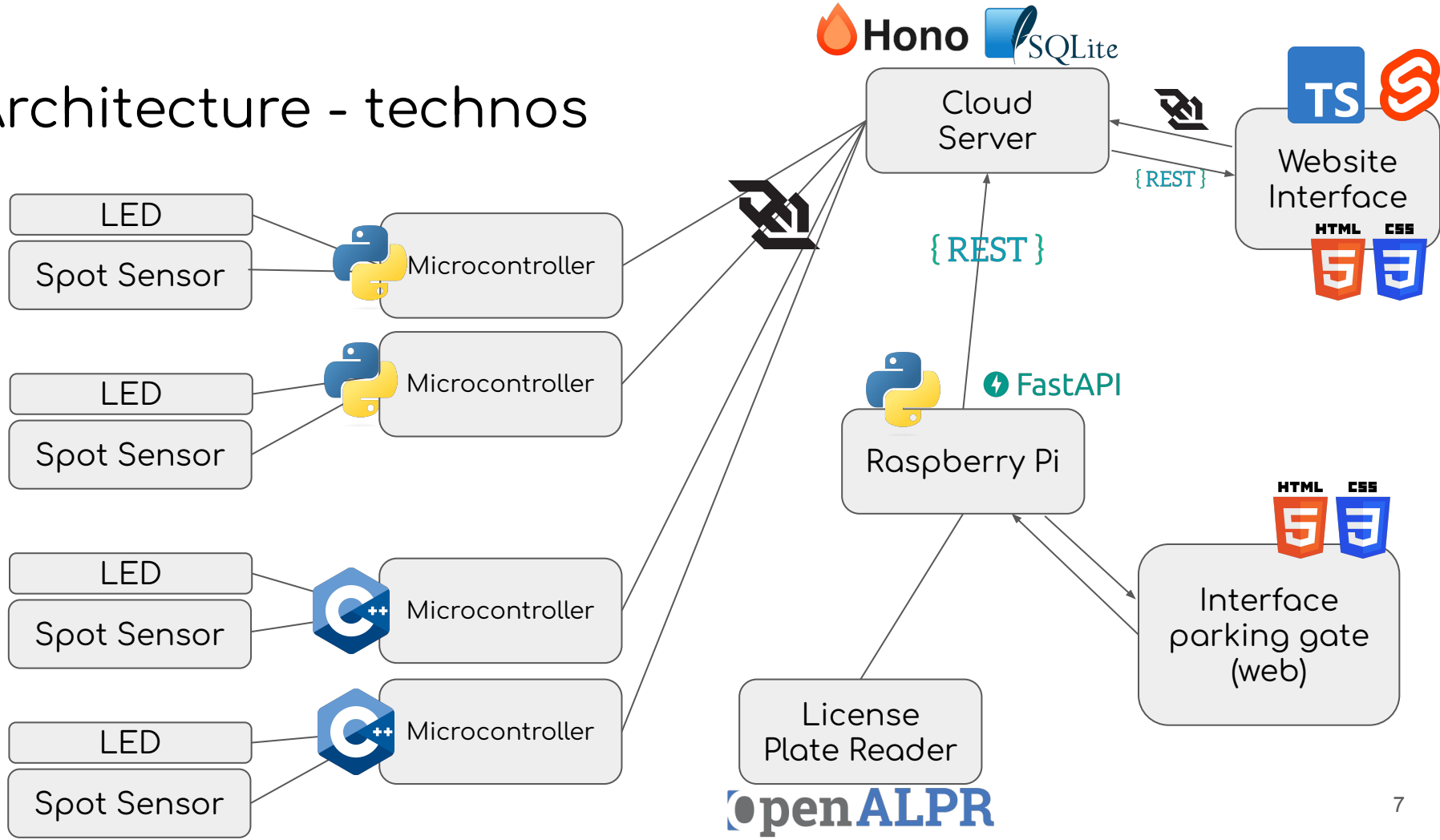


Cloudification

Implemented Architecture



Architecture - technos

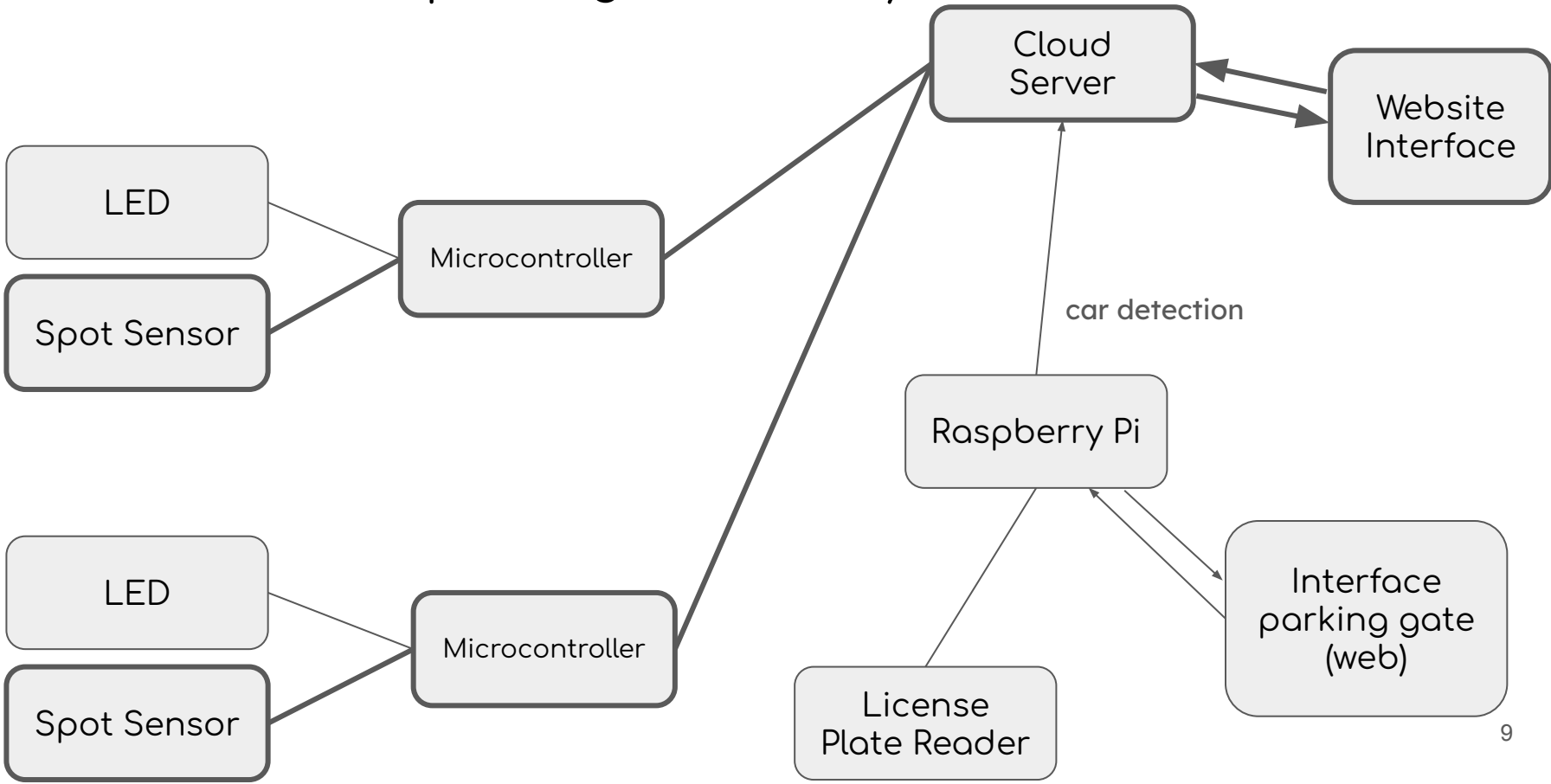


Function 1 : Check parking availability

Before entering a parking, users want to know if they will be able to park there and potentially where.

- Know the number of available spots in real-time
- Display this number on the website / application
- Find spots for electric vehicles

Function 1 : Check parking availability

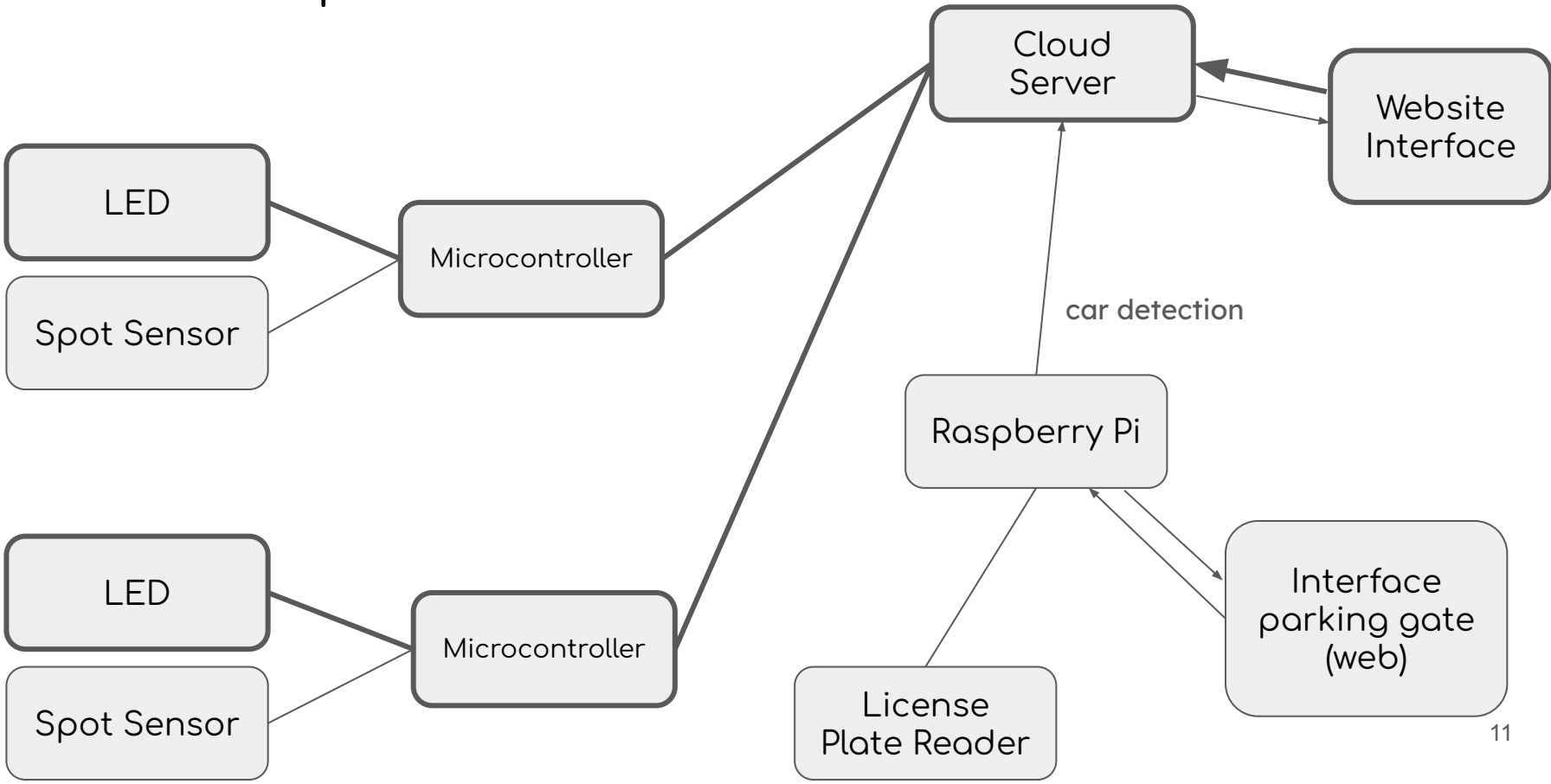


Function 2 : Spot reservation in advance

If a user is in a hurry or if the user regularly parks at the same parking, they might want to reserve a spot in advance.

- Using the website / application, visualise all available spots
- Select a spot
- Tag the spot as reserved on the website / application and in the parking lot

Function 2 : Spot reservation in advance



Function 3 : guide to you place

The parking has a lot of spots, a lot of floors but where are the available spots?

- Get guidance to the selected parking spot on your smartphone.

Function 3 : guide to you place

The parking has a lot of spots, a lot of floors but where are the available spots?

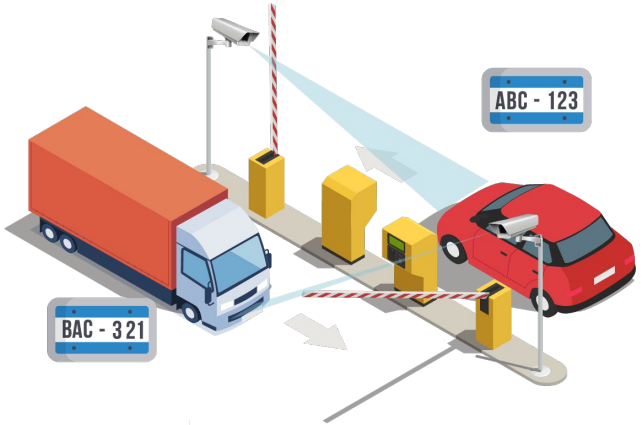
- Get guidance to the elevator or stairs and your spot on your smartphone.

Already done in INF103 !

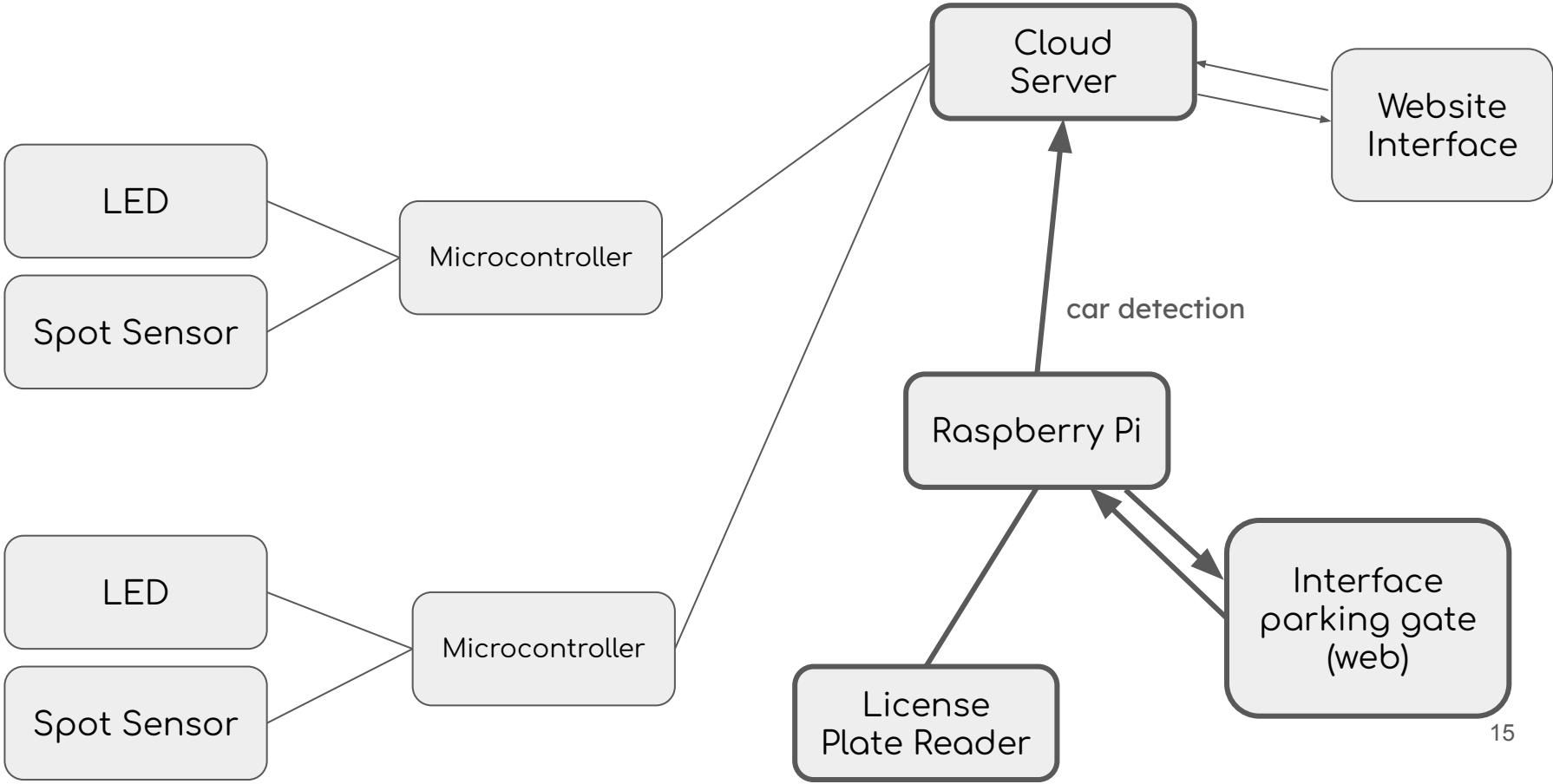
Function 4 : License plate reader for seamless entry and billing

Having an no-ticket parking is great for multiple reasons :

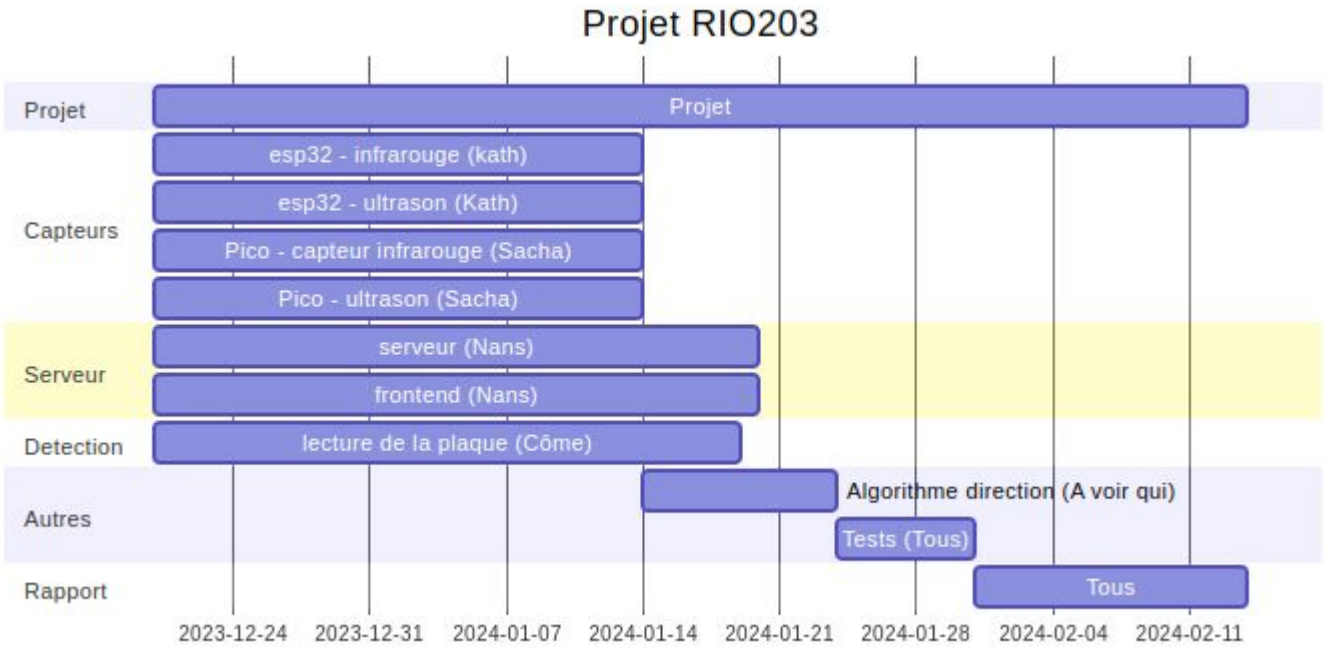
- Faster entry, no need for physical tickets.
- Users can exit and enter on foot without a physical ticket.
- You can pay on your mobile for your stay



Function 4 : License plate reader for seamless entry and billing



Implementation



Sensors - Introduction

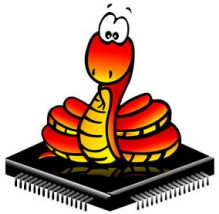
What we wanted ?

- High availability & responsivity (real time communication)

- High compatibility
  - multi-sensors
  - multi-controller

ARDUINO

MicroPython



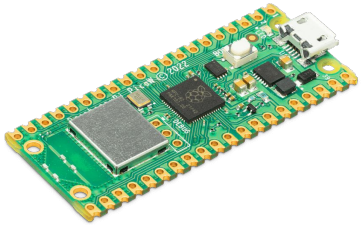
Sensors WOKWI

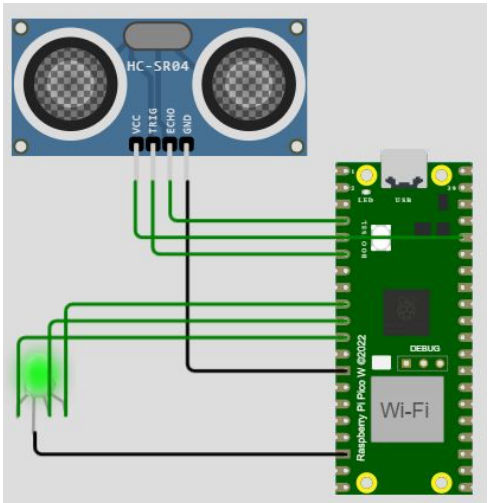
Implementation by Sacha

- Python and C++
- (Raspberry Pico and ESP8266)

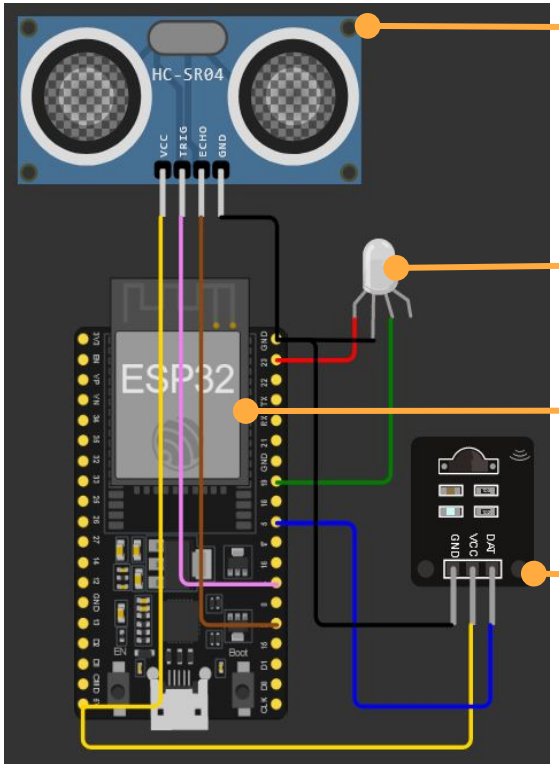
- Real hardware then Simulator

- Websocket protocol





Sensors



Ultrasound sensor

Dual Colour LED

Microcontroller ESP32

Infrared sensor

ARDUINO

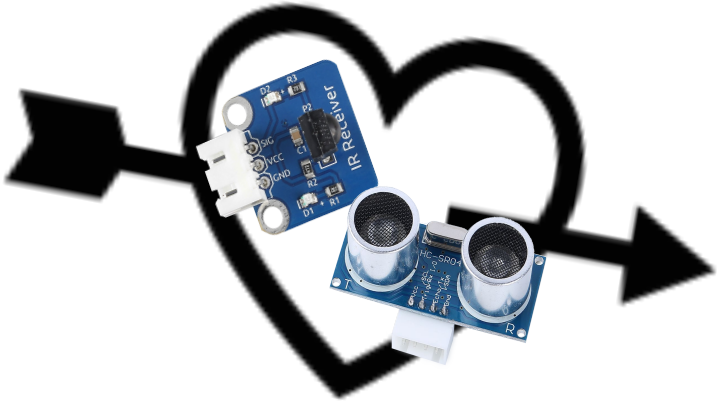
- Real Hardware
- C++ programming
- WebSocket

Multisensor solution

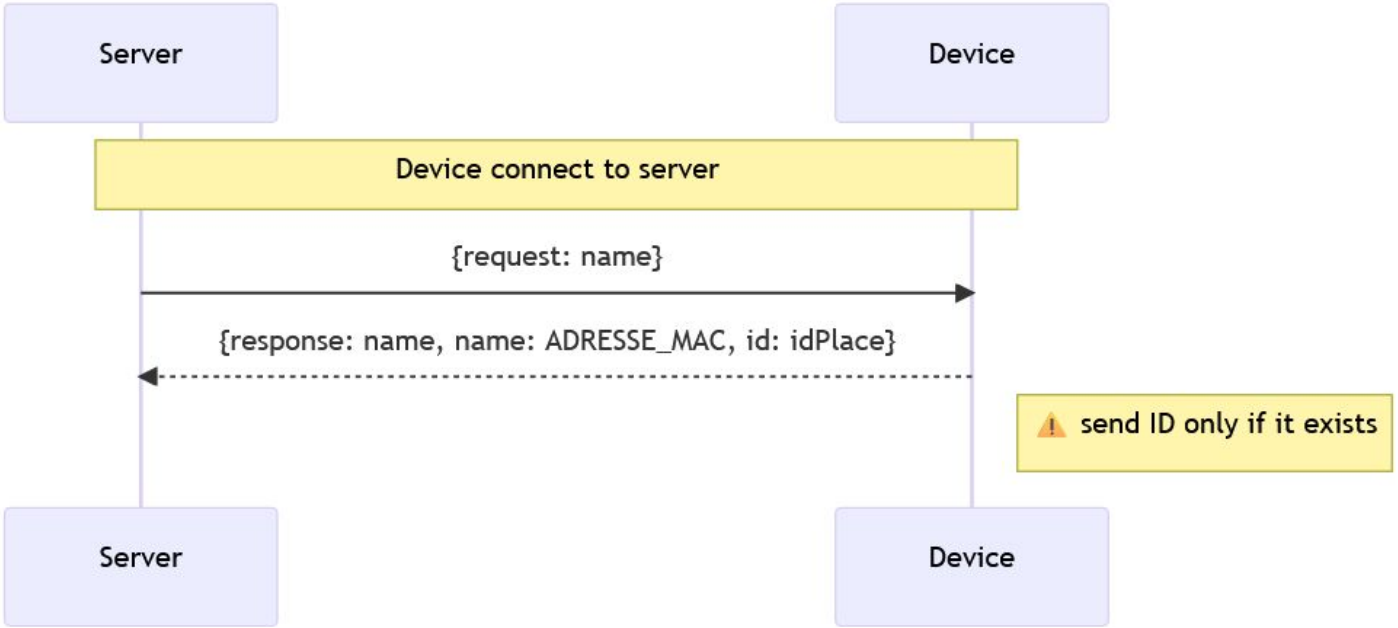
- Ultrasound sensor
- Infrared sensor

→ High compatibility

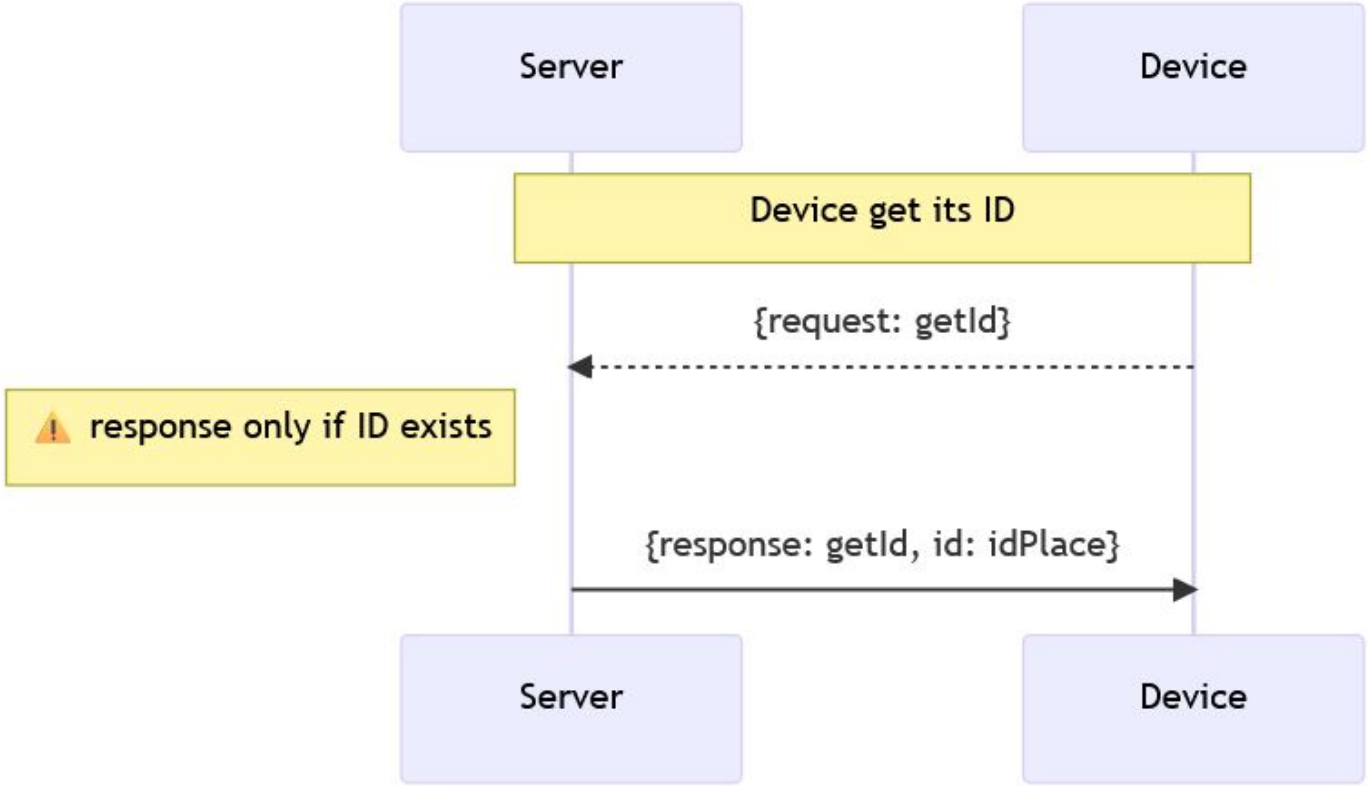
→ Adaptive solution



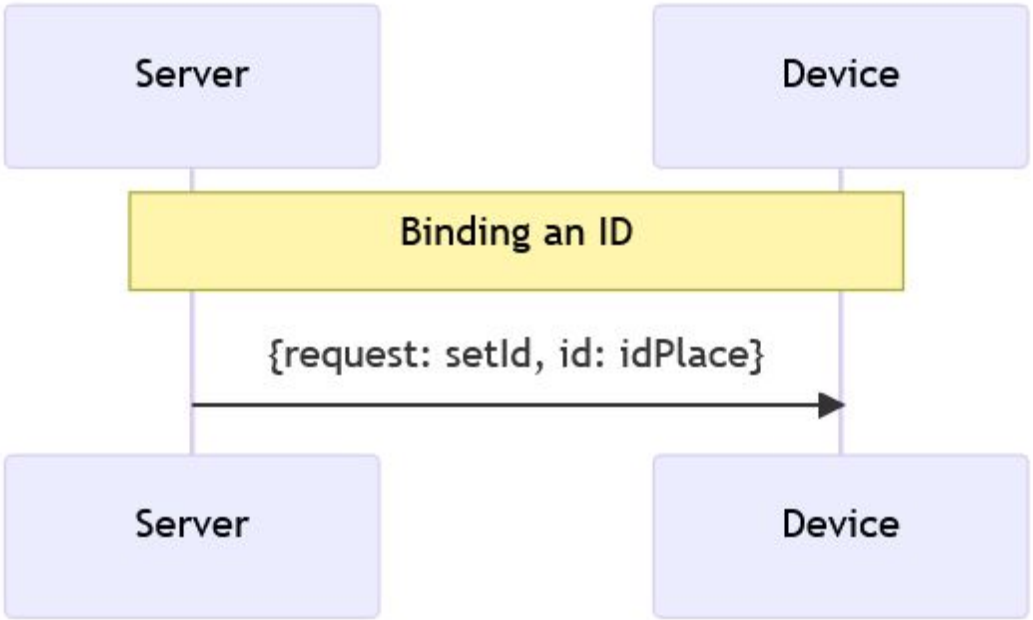
Sensors requests - Device info



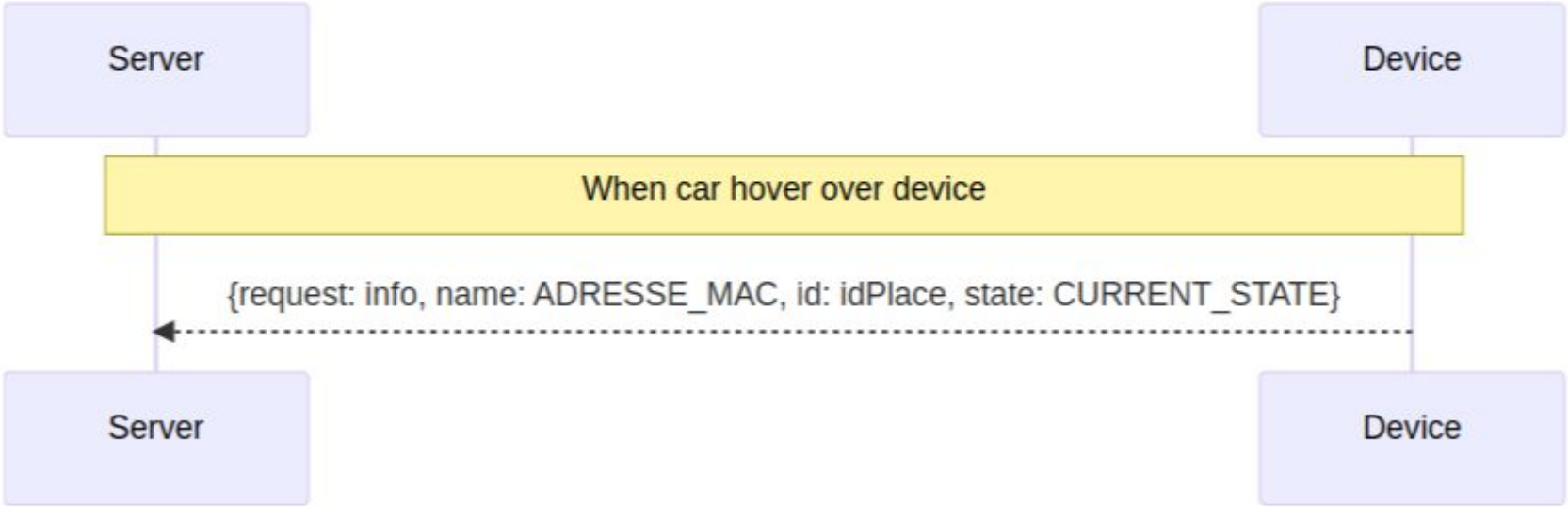
Sensors requests - Device ID getter



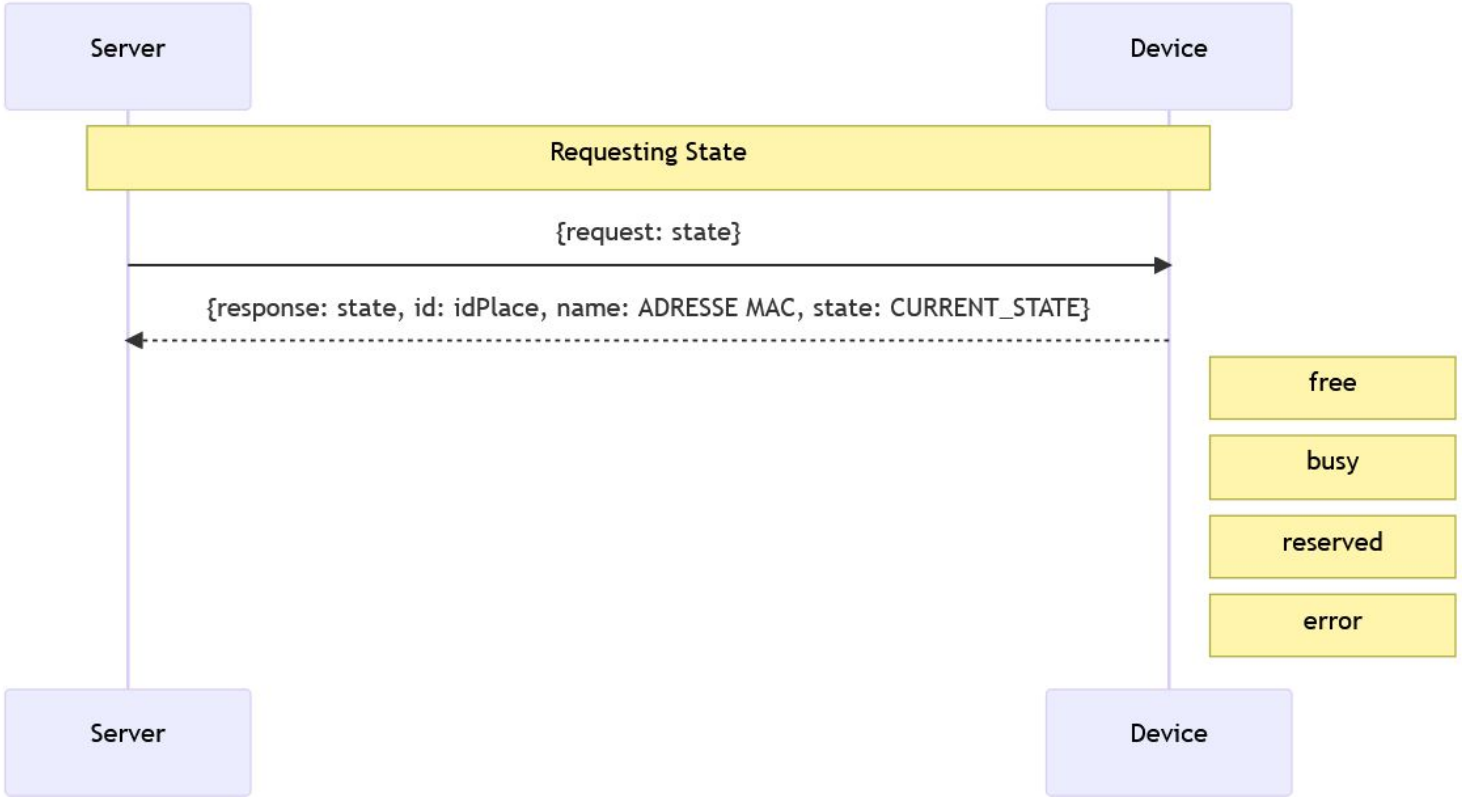
Sensors requests - Device ID setter



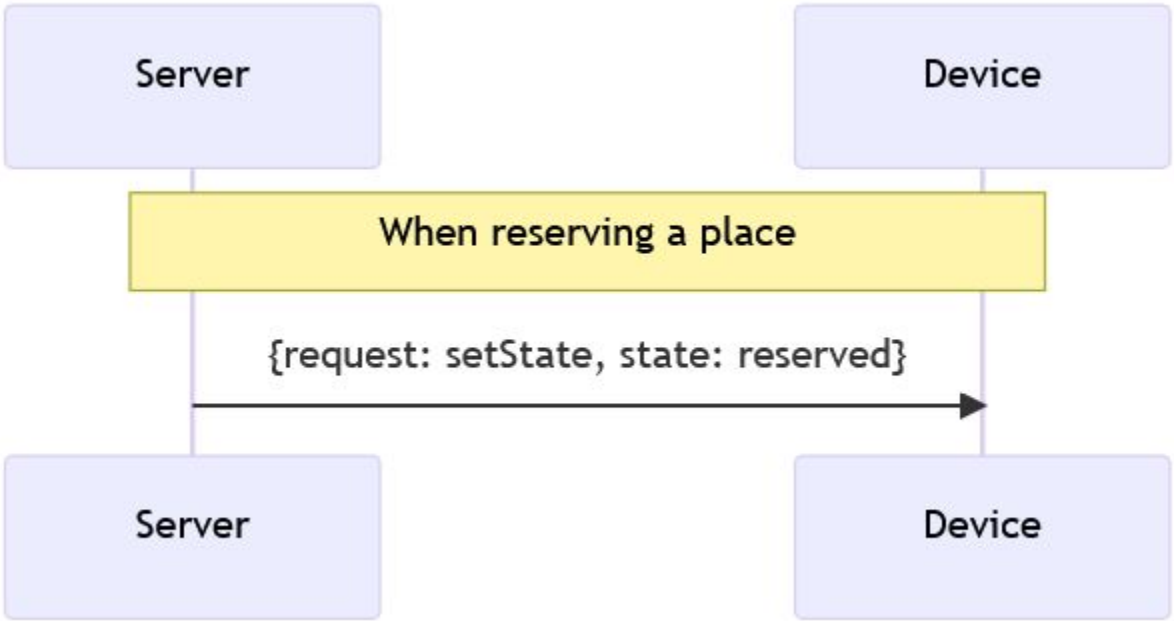
Sensors requests - Device hover notification



Sensors requests - Device state info



Sensors requests - Device reservation



Sensors requests - Websocket

Requirements :

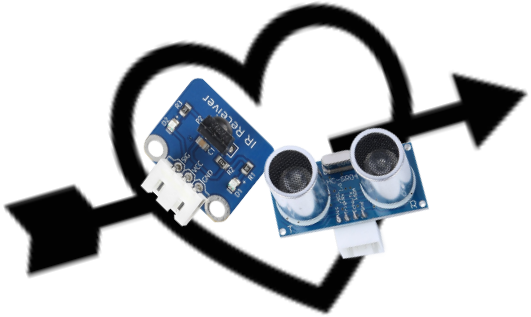
- Real-time transmission
- Bidirectional transmission

We started by using HTTP requests then moved on to Websocket due to limitations.

Why websocket ?

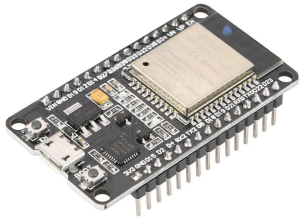
- Bypass NAT/PAT & firewall issues
- Real-time transmission

Sensors - conclusion



Implementation

- C++
- Python



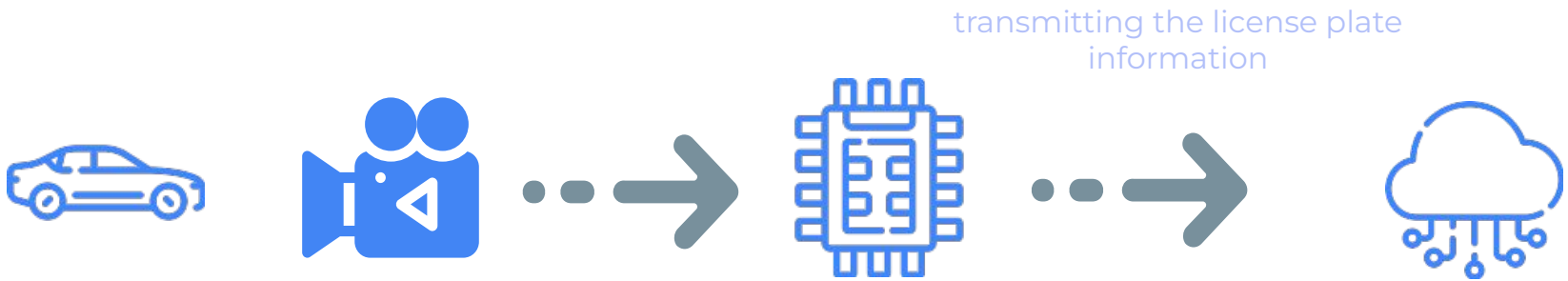
- Real hardware (ESP32)
- Simulator (ESP8266)

} Ultrasound & Infrared sensors

- Websocket protocol

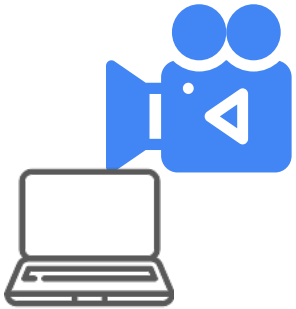
Licence plate reader

Base idea



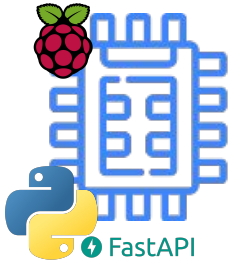
Licence plate reader

Actual Implementation



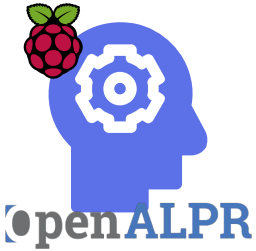
Laptop webcam as camera

HTTP



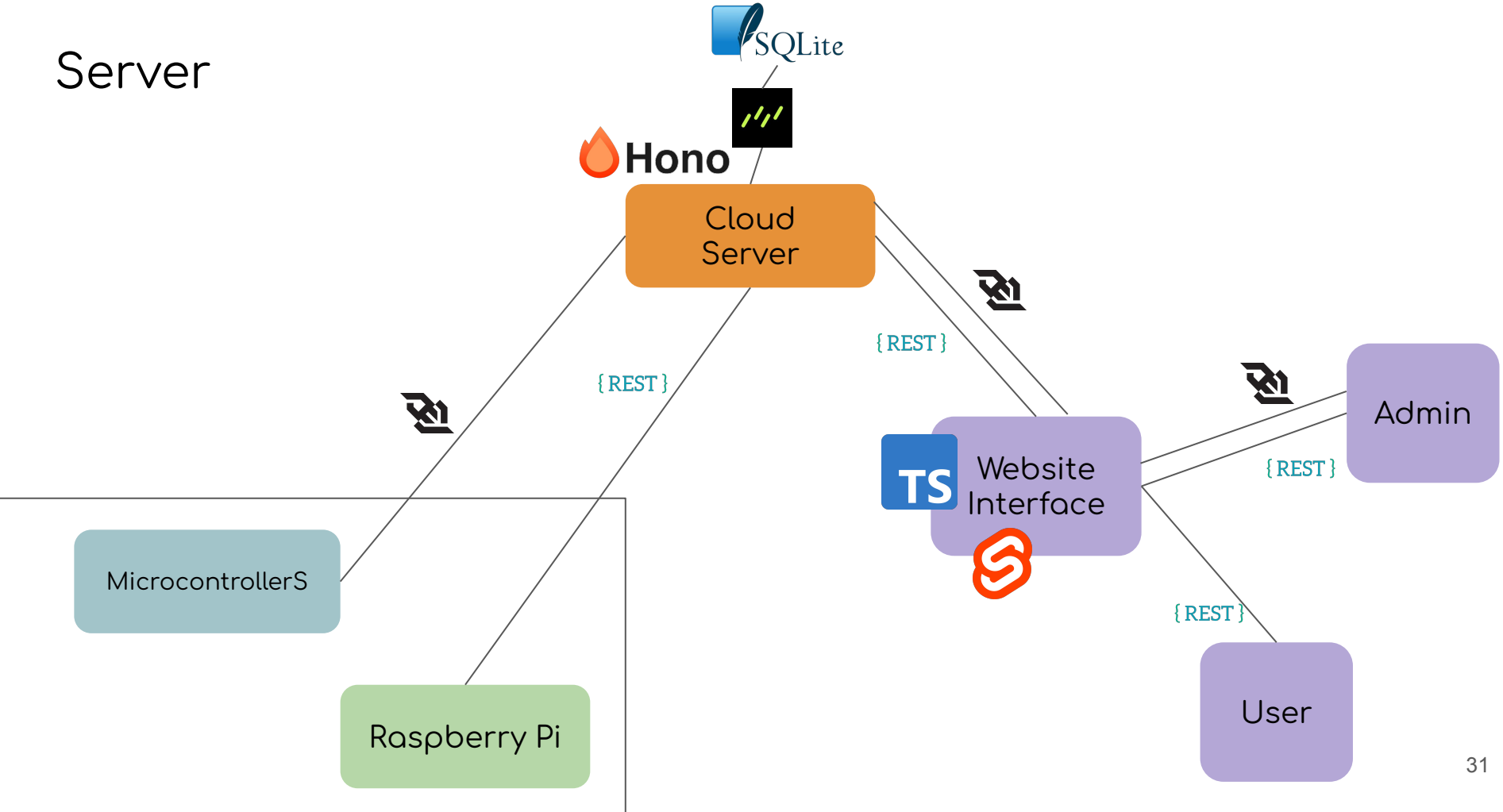
HTTP Server to receive the image

OpenALPR as License Plate Detection Service on the Raspberry Pi



HTTP to the main server

Server


SQLite
Hono
Cloud Server
{REST}
{REST}
Admin
{REST}
TS
Website Interface
{REST}
MicrocontrollerS
Raspberry Pi
User

Conclusion

Questions ?