Università di Genova | DIPARTIMENTO DI INFORMATICA, BIOINGEGNERIA, ROBOTICA E INGEGNERIA DEI SISTEMI

SCUOLA DI SCIENZE MATEMATICHE, FISICHE E NATURALI

Corso di laurea in Computer Science

Anno accademico 2021/2022

Internet of Things ~ VP-Dibris Smart Parking

Simone Cella S4334970
Matteo Spinaci S4338765
Davide Raffo S4510233

Main Objectives:

The project goal is to build a system able to monitor the different parking lots of VP-Dibris.

...more in details…

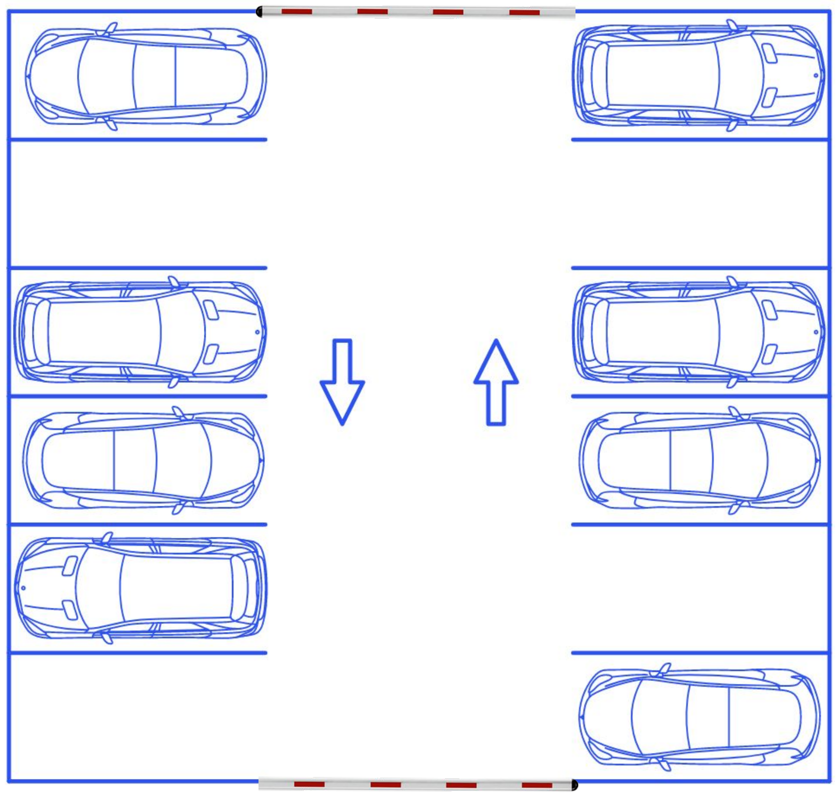
The parking is composed by different parking slots, each of which is controlled by the sensors which are managed by the monitoring system.

People who can enter to park must be registered in advance to the system, otherwise they are not recognized.

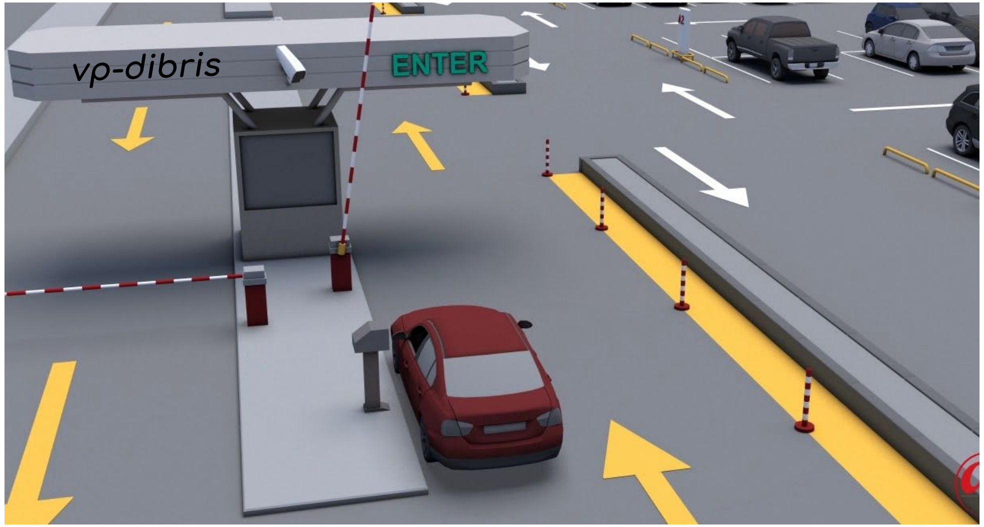
Parking Description:

There are two gates, one for entry and one for exit.

Both upon arrival and at the exit of a car, the system is able to check from the database whether the machine is registered, if not the gate remains closed.



Gates System:

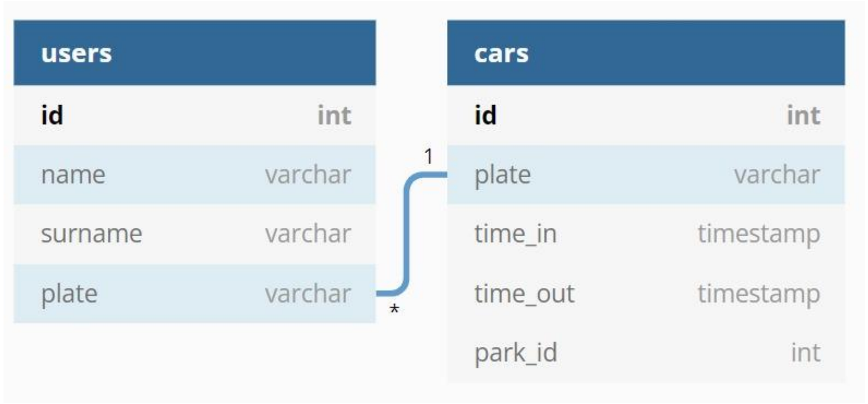


Each gates has:

- Two pairs of photocells
- A Webcam IP (for plate recognition)
- A physical gate
- Panel for user interaction

Database details:

We decided to store all the information in a Mysql Database.
This database will work in local mode on the Linux system, from node-red we can access and manipulate it by using the right port (*localhost@3306*)



users	
id	int
name	varchar
surname	varchar
plate	varchar

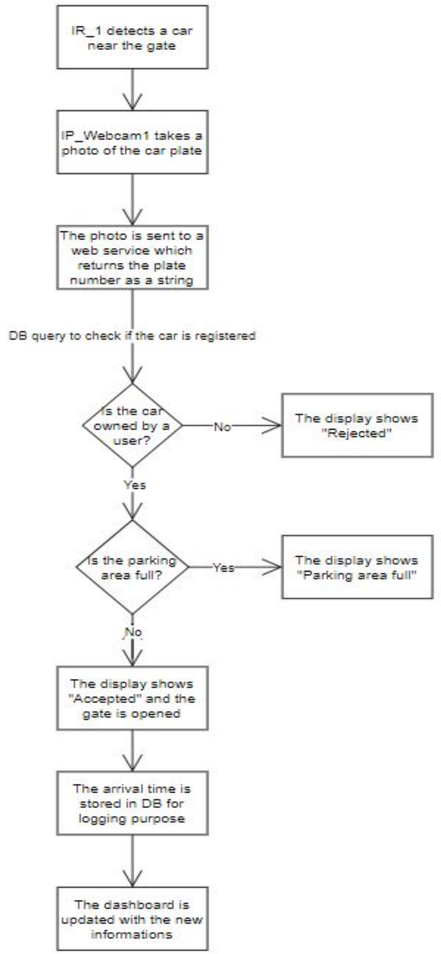
cars	
id	int
plate	varchar
time_in	timestamp
time_out	timestamp
park_id	int

As mentioned before the system is able to check from the database whether the car is registered. Indeed we have two tables in our database, we take the picture of the arriving car, and then we check if it register in the users.table.

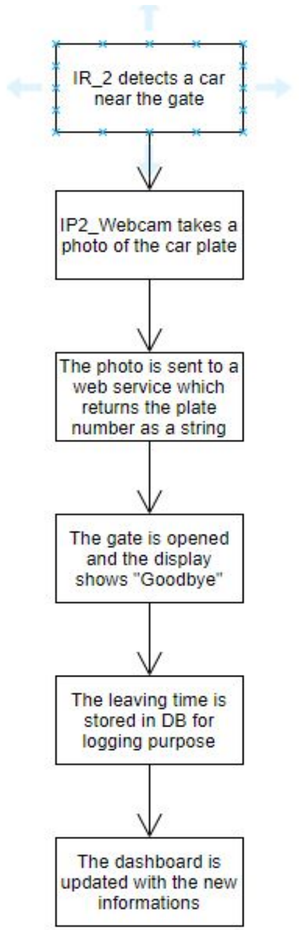
The plate is strictly linked to a single person, allowed to enter the parking.

All the users are registered before enter the parking.

You can make many actions on db (create,drop table,insert….) by using the middleware.js that allows us to interact with the db.


IR_1 detects a car near the gate
IP_Webcam1 takes a photo of the car plate
The photo is sent to a web service which returns the plate number as a string
DB query to check if the car is registered
Is the car owned by a user?
No
The display shows "Rejected"
Yes
Is the parking area full?
Yes
The display shows "Parking area full"
No
The display shows "Accepted" and the gate is opened
The arrival time is stored in DB for logging purpose
The dashboard is updated with the new informations

Car enter event


IR_2 detects a car near the gate
IP2_Webcam takes a photo of the car plate
The photo is sent to a web service which returns the plate number as a string
The gate is opened and the display shows "Goodbye"
The leaving time is stored in DB for logging purpose
The dashboard is updated with the new informations

Car leave event

Flow explanation in node-red:

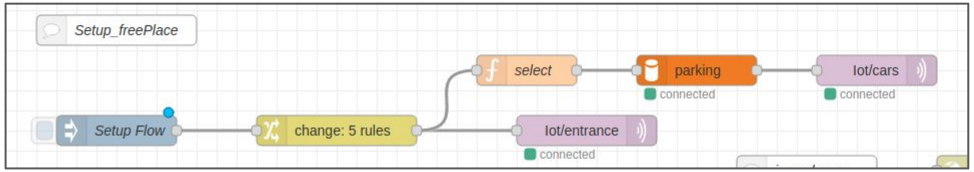
At the beginning, in the first subset of linked-nodes "***Setup Flow***" we're going to define the size of our parking, the flow variables that are used in order to update the available places of the parking.

Moreover, in order to simulate the arriving of a new car, we provided the link of many plates pictures.

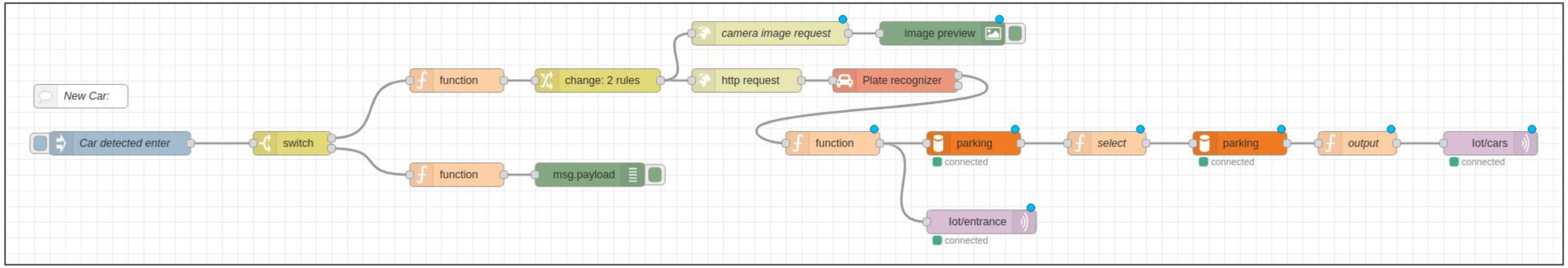
Then, in the "***Car detected enter***" subset we simulate the entrance of a new car, so we have to check the number of free places in the parking, and then if the plate of the car is present in the user table.If everything is correct, we proceed by assigning a place where the user have to park the car, and we'll write all the necessary informations in the database, such as:

- Id of the user (surrogate key)
- Plate
- Time_in (time of arrival)
- Time_out (Null for this moment...)
- Park_id

Setup Flow:

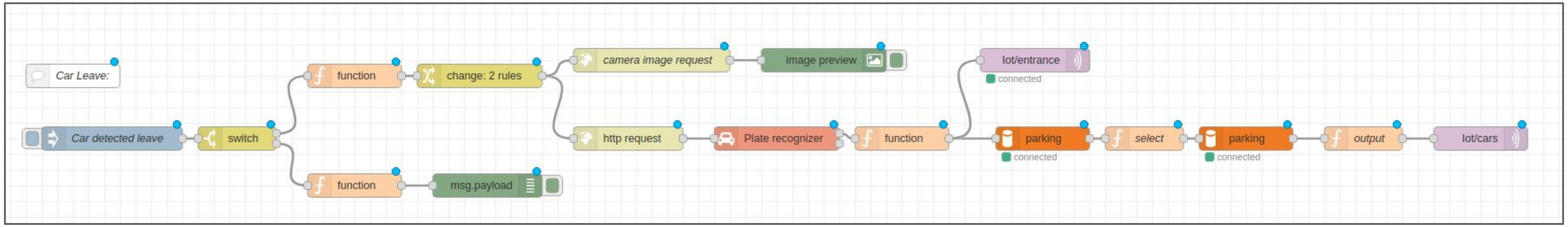


Car detected enter:



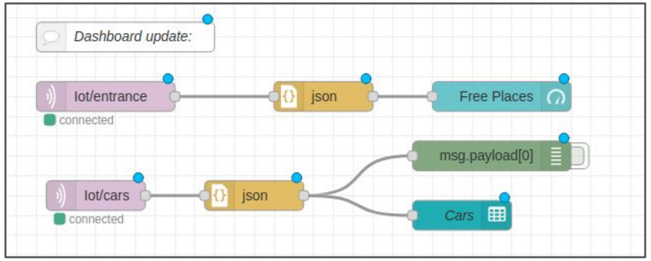
Then, we proceed to simulate the exit of a certain car, in the "***Car detected leave***" flow, the first thing to do is to check if there still are cars in the parking, if yes the IP-Webcam will take a picture of the plate and the system add to the database the leaving time, by updating the corresponding record.

Car detected leave flow:



In the end, we have to send all these information to the Node-red Dashboard, where we can visualize the data about all the cars in the parking, also the complete history of the traffic in the parking.

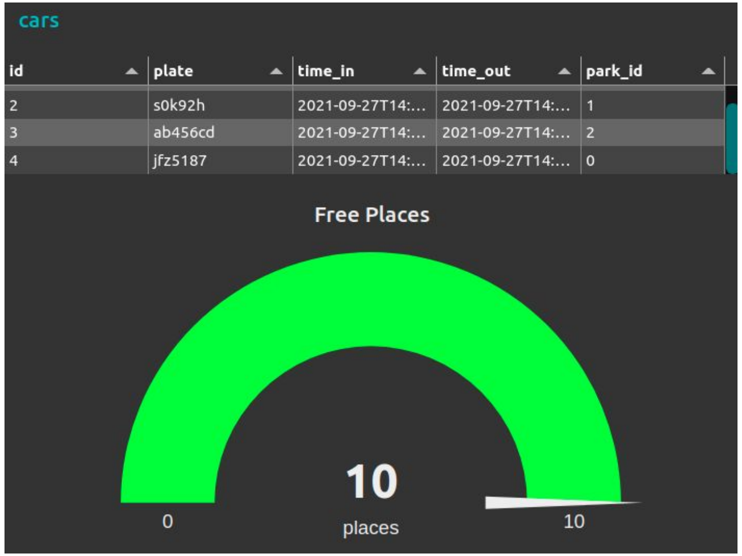
Dashboard update flow:



Mqtt broker:

We use Mqtt broker in order to *send&receive* data across the application.

In the details we use it to update constantly the number of free places in the parking and to send the all the informations about the cars to the Node-red Dashboard where we can see it.



cars

id	plate	time_in	time_out	park_id
2	s0k92h	2021-09-27T14:...	2021-09-27T14:...	1
3	ab456cd	2021-09-27T14:...	2021-09-27T14:...	2
4	jfz5187	2021-09-27T14:...	2021-09-27T14:...	0

This is how the dashboard is able to show all the information

Software Components:

- ***Node-red****:*
  Applications used to handle the back-end of our system.
- ***Node-red Dashboard****:*
  Applications used to handle the front-end of our system.
- ***Platerecognizer****:*
  API used to convert the plate photo into a String of characters.
- ***Mysql****:*
  Used to store data acquired from sensors.
- ***Node.js****:*
  Used to handle all the database actions.

Hardware Requirements:

- ***Raspberry Pi 4***:
  Used to manage the front-end logic.
- ***2 Pairs of Photocells***:
  For each gate we need 2 photocells to detect car and ensure it went through the gate.
- ***2 IR Raspberry Pi cameras***:
  To recognize the plate of the cars.
- ***2 LCD display***:
  to give all the information to the user (enter permission, free places…)

Live Demo