# IoT Technical Report – Team MaPaY

*Marion Burg - Pierre Corbay - Aymeric Fröhlicher*

## Description of the project:

Everybody has once faced the situation where you go to a shop and you have to find a slot in the parking. But the parking is huge and you don't know where there is an aivailable slot. Thanks to our solution, this won't happen anymore! You will know how many slots are available in the parking before entering, and you will be directed to the available slots thanks to some LEDs.

## State-of-the-art:

In order to propose the most adapted solution, we have studied many already existing solutions.

One of thoses solutions is proposed by TTS Parking. It consists in an interface which shows the number of places in the parking.

Accor Solutions also proposes a solution where you have a light above the available slots, which indicates to the driver where he can park.

All the other solutions are pretty similar from the ones explained above. We didin't find an exact price but it varies according to the number of slots in the parking. The bigger the parking is, the higher the price will be.

To distinguish ourselves from the existing propositions, we decided to add a light system which will show in which part of the parking are the available slots. It will avoid driving across all the parking to find a slot.

## Description of our IoT-based solution and what makes it better than other solutions:

For our solution, we will create a smart parking. At the entrance of the parking, a digital display will be placed to inform car drivers about the remaining slots.

If the parking is full, the parking barrier will stay closed. In the other case, the LCD display at the parking entrance will show the slot number which is available and the driver will be guided by our led system.

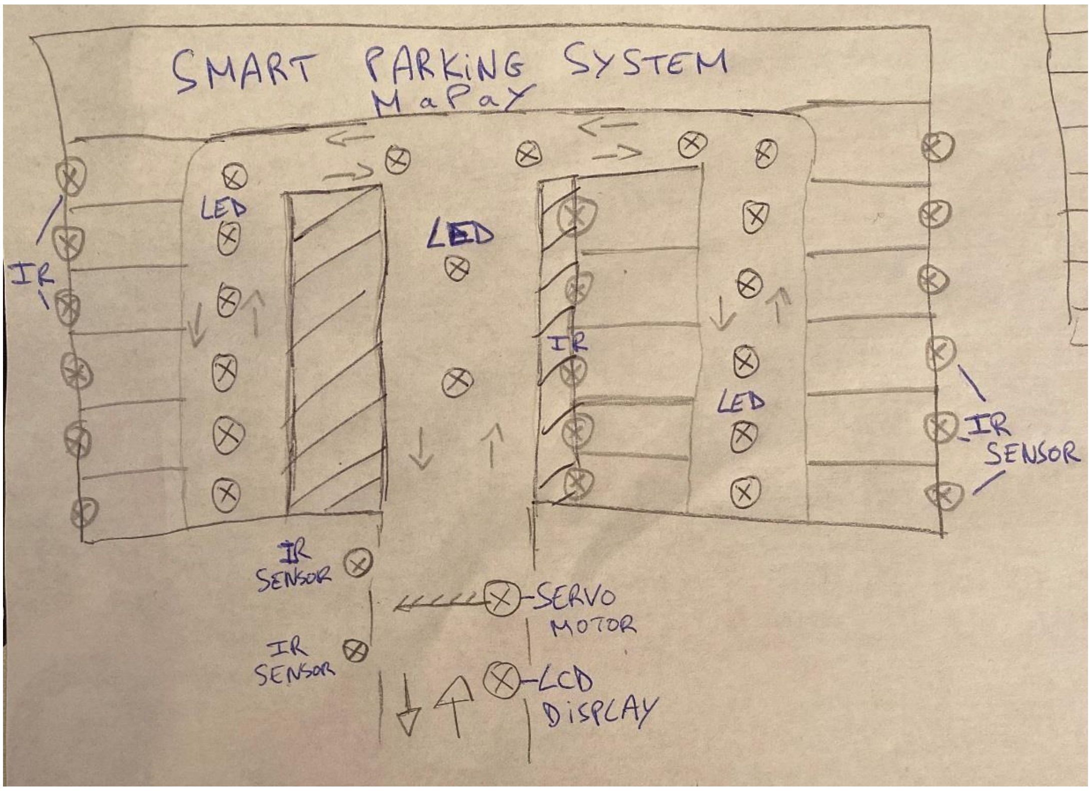
Drivers will save time because we all know big parkings where we're looking for a slot for a long time.

When a car approaches the entrance of the parking, an IR sensor / ultrasonic sensor will detect it. Only if some parking slots remain free, the parking barrier will open (servo motor) and let the car enter.

We will count the number of cars entering and going out of the parking. IR sensors will be placed beside each slot to know if the slot is available or not.

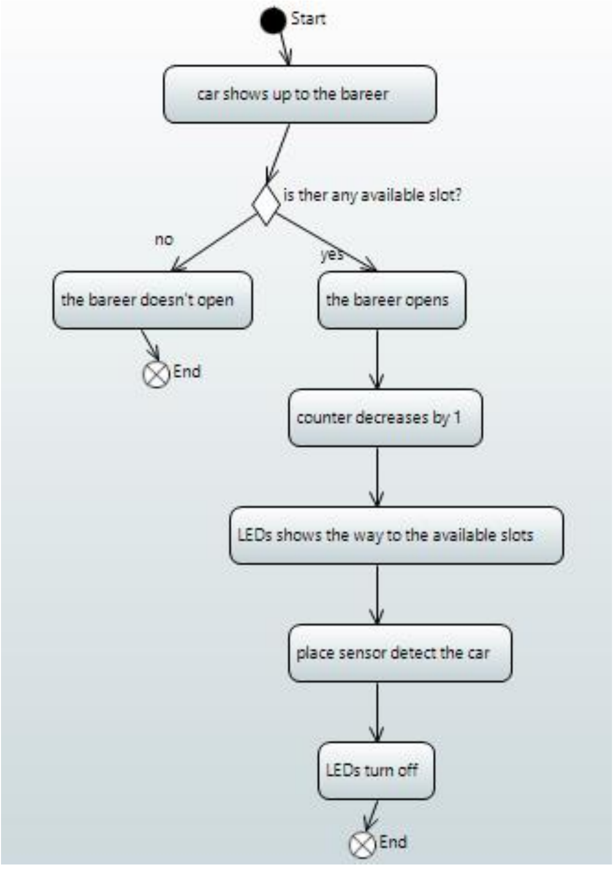
We are going to connect everything with the Arduino UNO board.

A global schema:

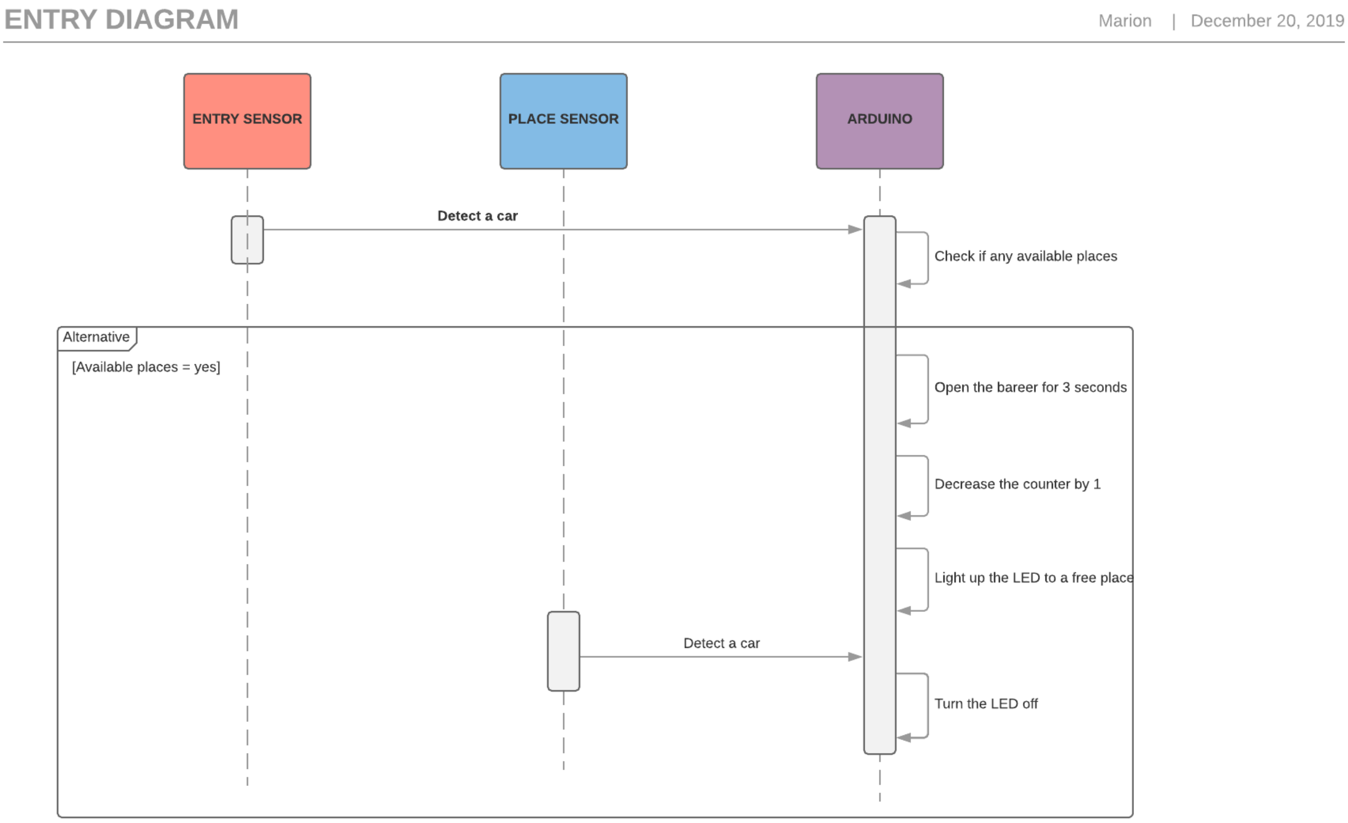


UML diagrams:

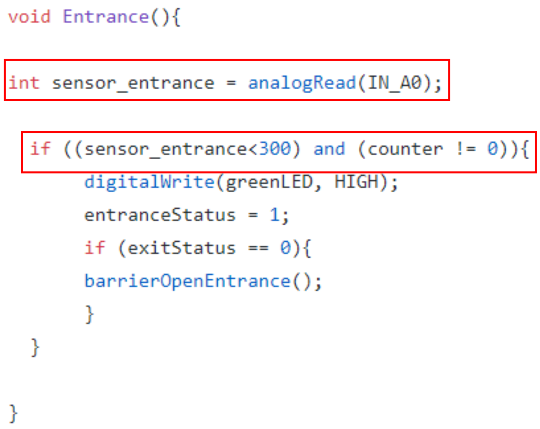
In order to help us realizing our solution and to define it precisely, we did some activity and sequence diagrams. The first ones explain the scenario when a car wants to enter in the parking, and the second ones describe the situation where a car exits the parking.


Start
car shows up to the bareer
is ther any available slot?
no
yes
the bareer doesn't open
End
the bareer opens
counter decreases by 1
LEDs shows the way to the available slots
place sensor detect the car
LEDs turn off
End

*Activity diagram for the situation where a car enters the parking*

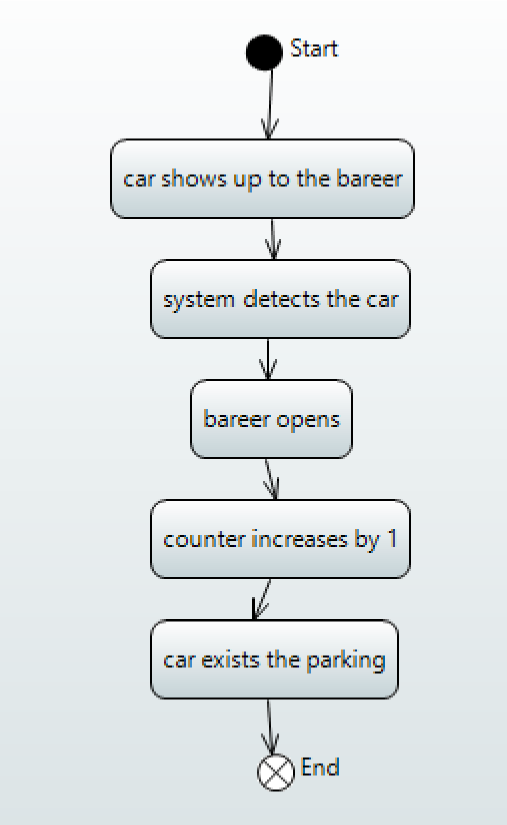

ENTRY DIAGRAM
Marion | December 20, 2019
ENTRY SENSOR
PLACE SENSOR
ARDUINO
Detect a car
Check if any available places
Alternative
[Available places = yes]
Open the bareer for 3 seconds
Decrease the counter by 1
Light up the LED to a free place
Detect a car
Turn the LED off

*Activity diagram for the situation where a car enters the parking*



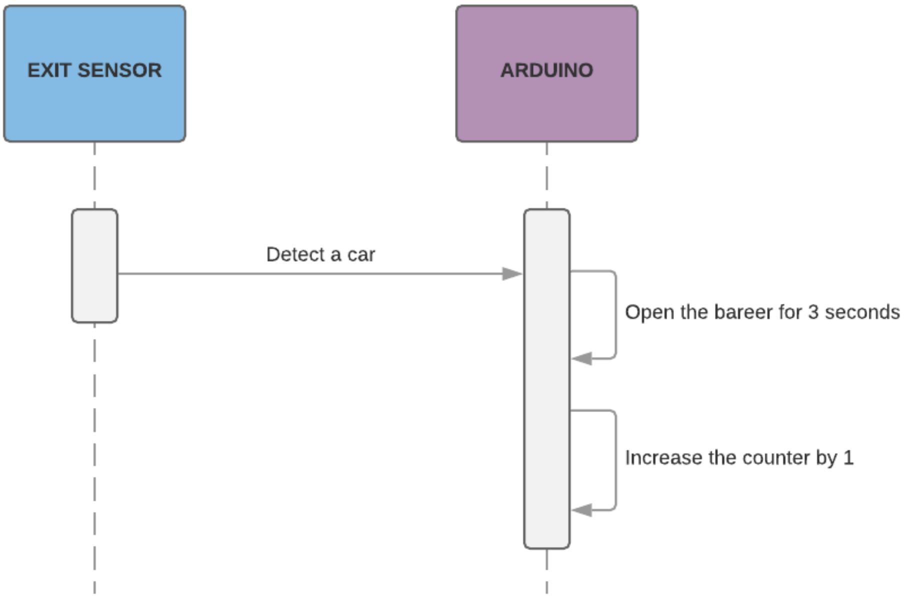
```
void Entrance(){

int sensor_entrance = analogRead(IN_A0);

  if ((sensor_entrance<300) and (counter != 0)){
        digitalWrite(greenLED, HIGH);
        entranceStatus = 1;
        if (exitStatus == 0){
        barrierOpenEntrance();
        }
  }

}
```

The detection of the car (“Detect a car” in the diagram above) is done in the code with the function *analogRead()*. The condition to open the barrier is that the value of the entrance sensor is under 300, which means that a car is above it, and that there still are some places available (the counter must not be equal to 0).

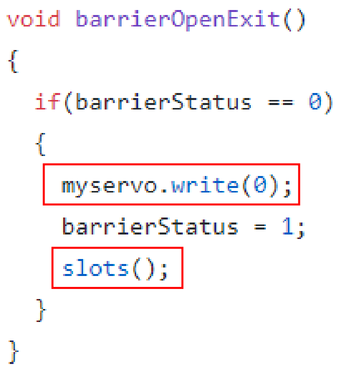


*Activity diagram for the situation where a car exits the parking*

# EXIT DIAGRAM



*Sequence diagram for the situation where a car exits the parking*



```
void barrierOpenExit()
{
  if(barrierStatus == 0)
  {
    myservo.write(0);
    barrierStatus = 1;
    slots();
  }
}
```

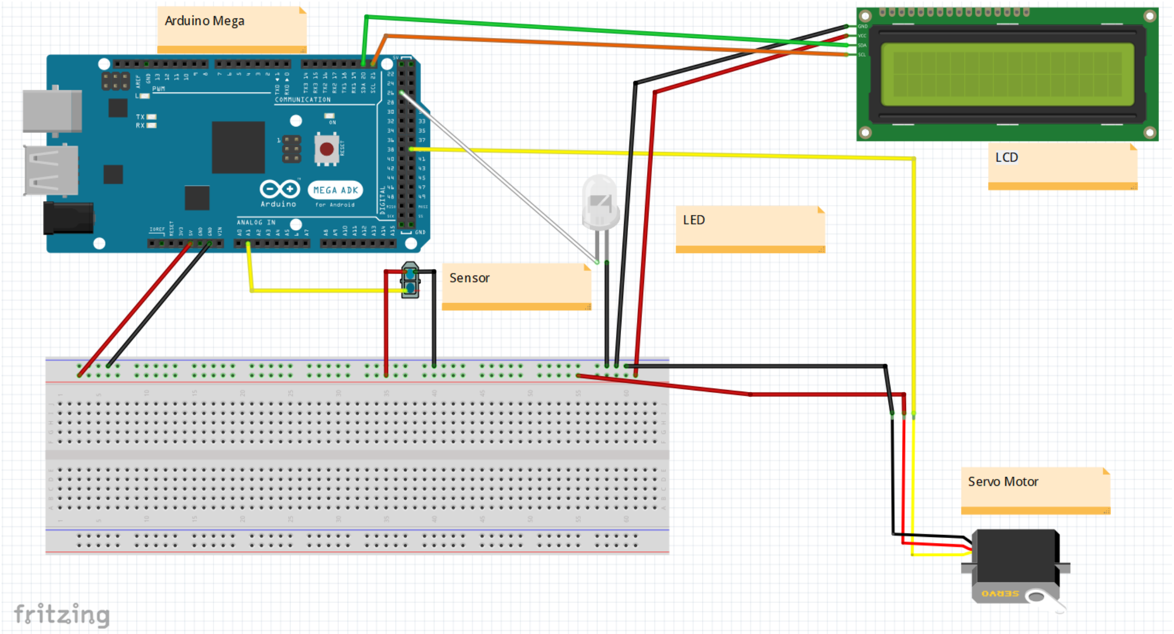
The function above describes the two messages “Open the barrier for 3 seconds” and “Increases the counter by 1” shown in the sequence diagram. We can see that we use the *write()* function on the motor, which will open the barrier, then we call the function *slots()*, which will increase the counter by 1.

## Components list:

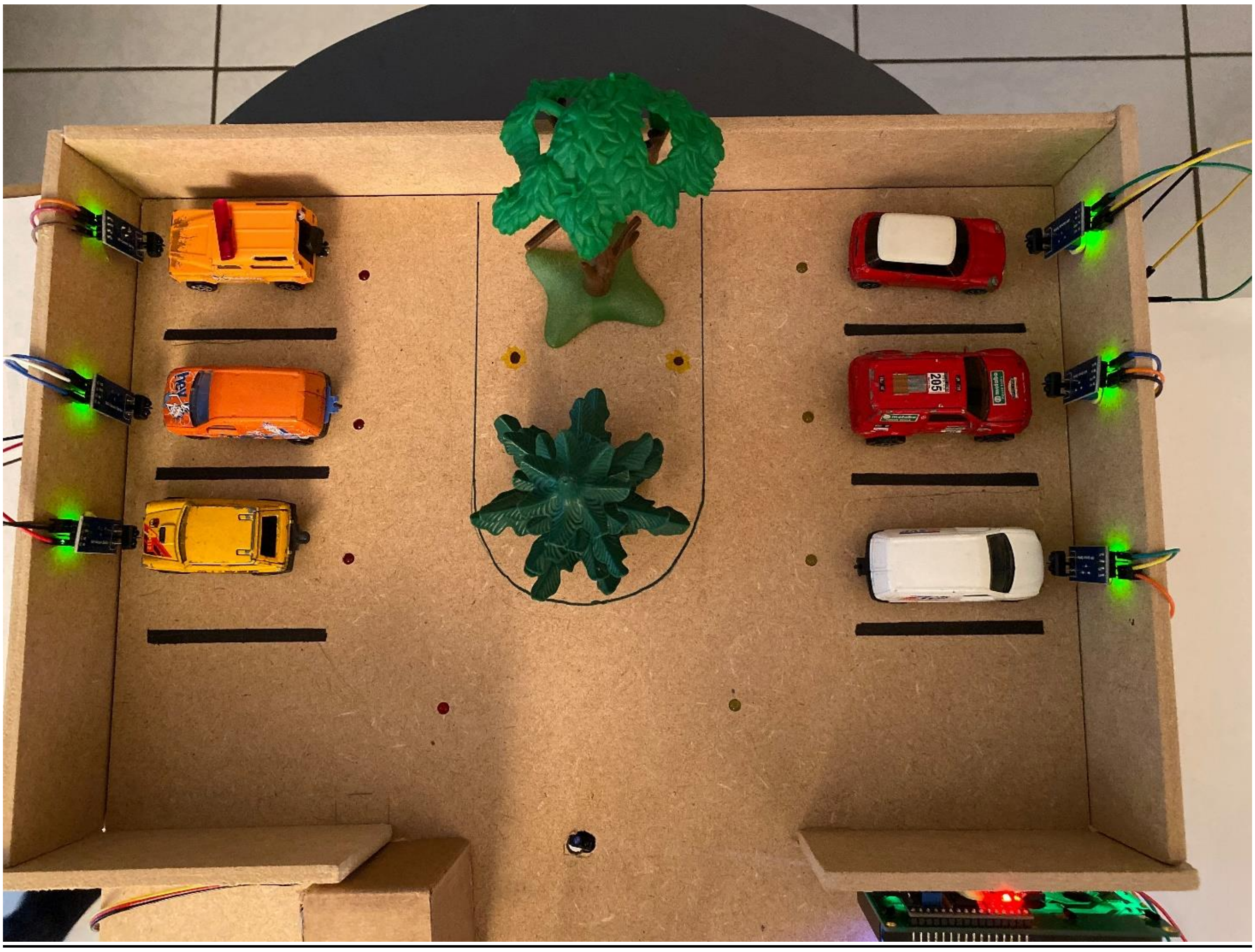
- Arduino Mega board used to link all the component together. We had one.
- IR/ultrasonic/proximity sensors used to detect cars on each slots of the parking. https://www.amazon.fr/gp/product/B071WG3F4P/ref=ppx_yo_dt_b_asin_title_o03_s00?ie=UTF8&psc=1&fbclid=IwAR0HyV3xO98TulG1i_aCTM3CiORLJe9W27jIOdBhRpgv0skZLW7-yQaYRgg
- LED used to indicate the way where there are free slots. Provided in the Arduino box.
- Resistors used with the led and the servo motor. Provided in the Arduino box.
- Breadboard used to connect all the element easily. Provided in the Arduino box.
- Servo Motor used to open the fence on the enter or exit of the parking. Provided in the Arduino box.
- LCD display used to display the number of remaining places in the parking.
- For the model we will use some paper or wood or cardboard to make the fence and the slots.

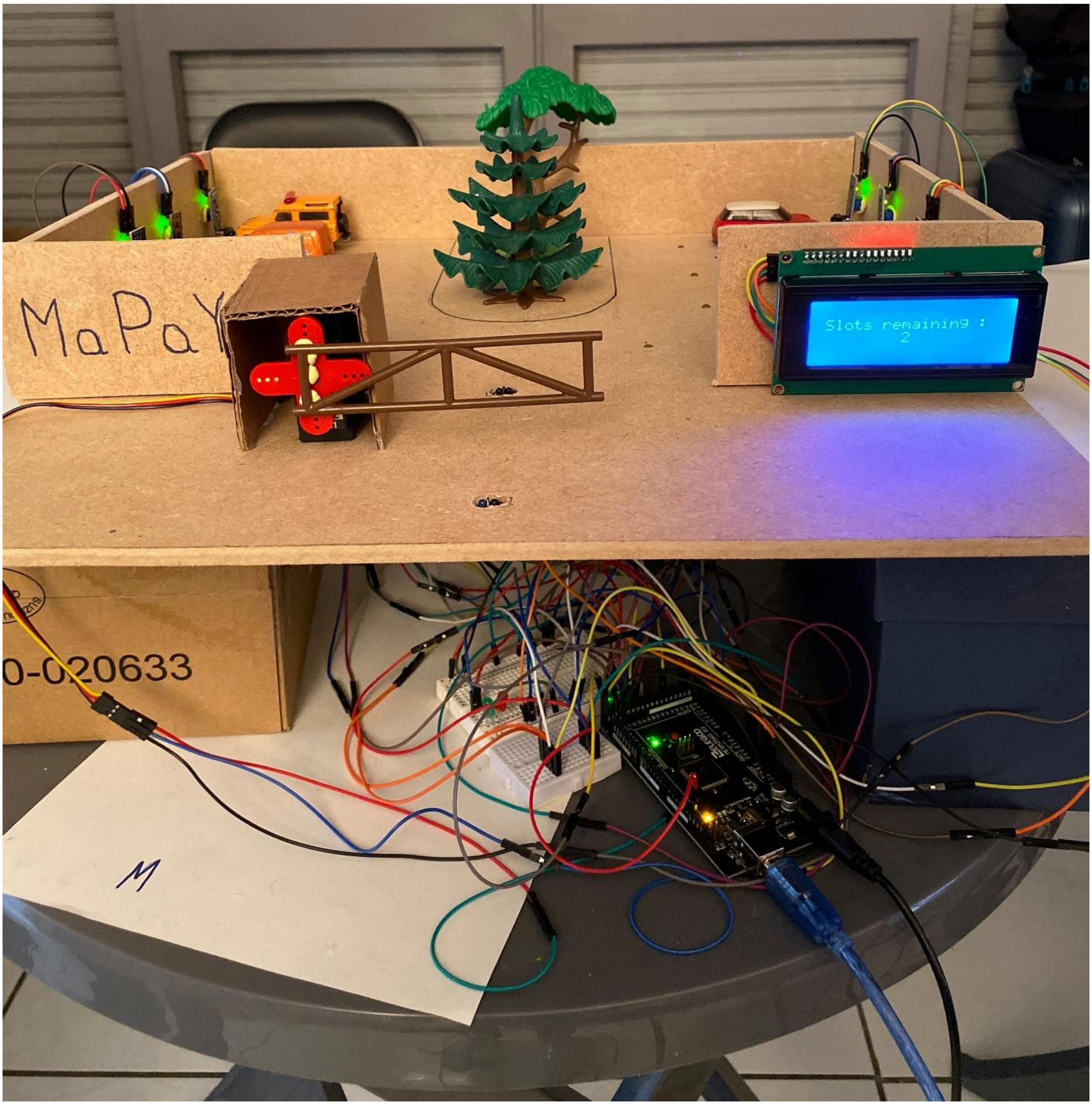
## Hardware design:

We realized a Fritzing schema with the components we used before doing the model. We only represented one item for each component but for the model we used 8 LED and 8 IR sensors (6 for the places, 1 for the entry, and 1 for the exit).

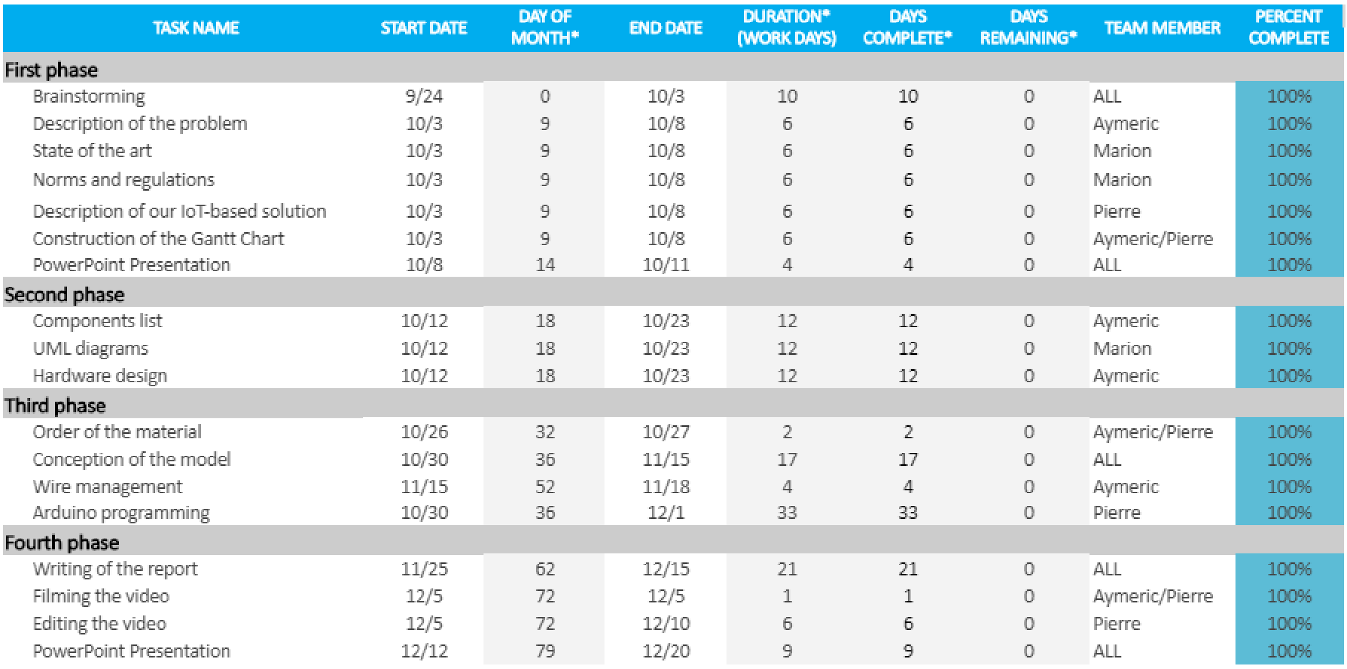


Final model:

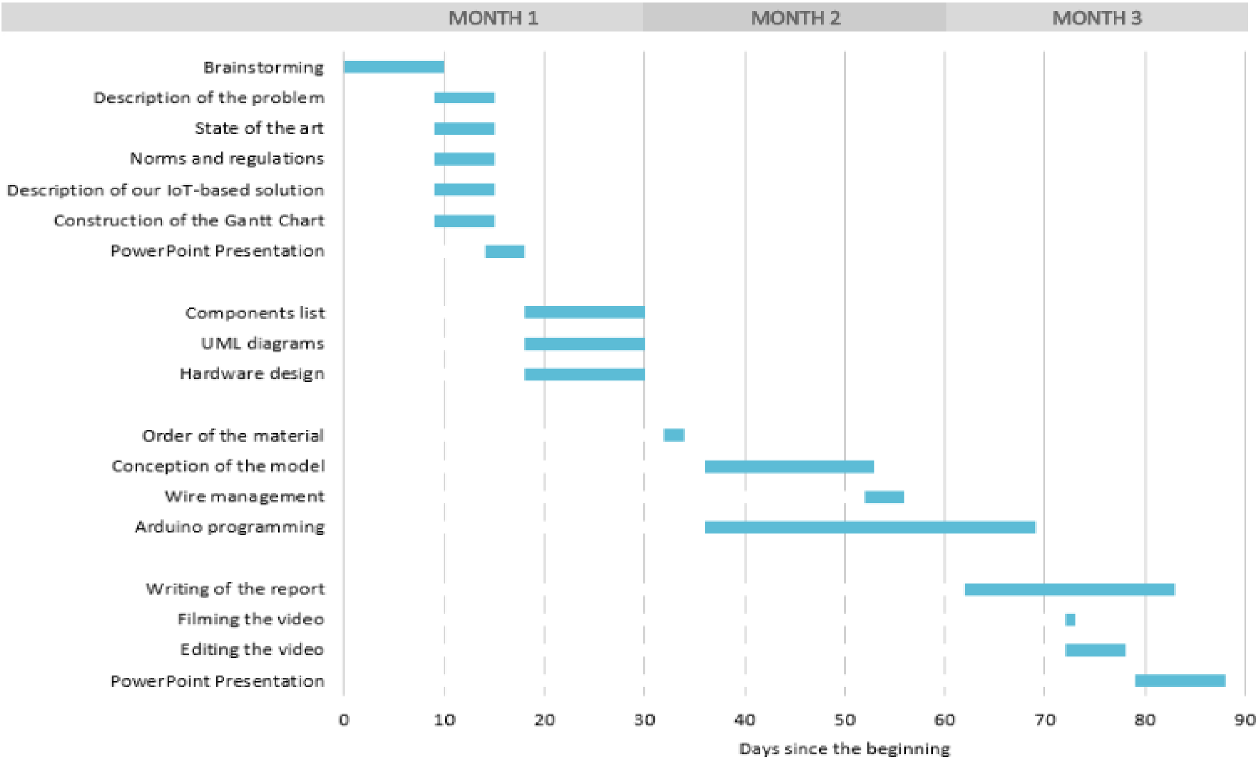



MaPaY
Slots remaining :
2
0-020633
M

## Gantt diagram:



| TASK NAME | START DATE | DAY OF MONTH* | END DATE | DURATION* (WORK DAYS) | DAYS COMPLETE* | DAYS REMAINING* | TEAM MEMBER | PERCENT COMPLETE |
|---|---|---|---|---|---|---|---|---|
| **First phase** | | | | | | | | |
| Brainstorming | 9/24 | 0 | 10/3 | 10 | 10 | 0 | ALL | 100% |
| Description of the problem | 10/3 | 9 | 10/8 | 6 | 6 | 0 | Aymeric | 100% |
| State of the art | 10/3 | 9 | 10/8 | 6 | 6 | 0 | Marion | 100% |
| Norms and regulations | 10/3 | 9 | 10/8 | 6 | 6 | 0 | Marion | 100% |
| Description of our IoT-based solution | 10/3 | 9 | 10/8 | 6 | 6 | 0 | Pierre | 100% |
| Construction of the Gantt Chart | 10/3 | 9 | 10/8 | 6 | 6 | 0 | Aymeric/Pierre | 100% |
| PowerPoint Presentation | 10/8 | 14 | 10/11 | 4 | 4 | 0 | ALL | 100% |
| **Second phase** | | | | | | | | |
| Components list | 10/12 | 18 | 10/23 | 12 | 12 | 0 | Aymeric | 100% |
| UML diagrams | 10/12 | 18 | 10/23 | 12 | 12 | 0 | Marion | 100% |
| Hardware design | 10/12 | 18 | 10/23 | 12 | 12 | 0 | Aymeric | 100% |
| **Third phase** | | | | | | | | |
| Order of the material | 10/26 | 32 | 10/27 | 2 | 2 | 0 | Aymeric/Pierre | 100% |
| Conception of the model | 10/30 | 36 | 11/15 | 17 | 17 | 0 | ALL | 100% |
| Wire management | 11/15 | 52 | 11/18 | 4 | 4 | 0 | Aymeric | 100% |
| Arduino programming | 10/30 | 36 | 12/1 | 33 | 33 | 0 | Pierre | 100% |
| **Fourth phase** | | | | | | | | |
| Writing of the report | 11/25 | 62 | 12/15 | 21 | 21 | 0 | ALL | 100% |
| Filming the video | 12/5 | 72 | 12/5 | 1 | 1 | 0 | Aymeric/Pierre | 100% |
| Editing the video | 12/5 | 72 | 12/10 | 6 | 6 | 0 | Pierre | 100% |
| PowerPoint Presentation | 12/12 | 79 | 12/20 | 9 | 9 | 0 | ALL | 100% |


MONTH 1
MONTH 2
MONTH 3
Brainstorming
Description of the problem
State of the art
Norms and regulations
Description of our IoT-based solution
Construction of the Gantt Chart
PowerPoint Presentation
Components list
UML diagrams
Hardware design
Order of the material
Conception of the model
Wire management
Arduino programming
Writing of the report
Filming the video
Editing the video
PowerPoint Presentation
Days since the beginning