**11th January 2023**

**Document Version 0.1**

UML Document

Industry project

|  |  |
| --- | --- |
|  | **Teacher:** |
|  | Andova, Suzana |
|  |  |
|  | **Students:** |
|  | - Mitchell van ‘t Kruys (4058151)  - Daniel García López (4976215)  - Nick Mulders (4523504)  -  Andre Sanao (4425553)  - Sean Brokke - (4208501)  - Luka Aerts (4202317) |
|  |  |

Table of Contents

[1. UML Diagram 2](#_Toc1332288840)

[2. State Diagrams 4](#_Toc143676420)

[2.1 Parking cluster module 5](#_Toc111019082)

[2.2 Entrance Module 6](#_Toc192261270)

[2.3 Node Red 8](#_Toc481560953)

[2.4 Gateway Module 9](#_Toc842585510)

[3. Flow Charts 10](#_Toc1679429699)

[3.1 Parking cluster / entrance module connection 10](#_Toc1768501923)

[3.2 Node Red message handling 11](#_Toc1827592439)

[3.3 LCD display protocol 12](#_Toc904563092)

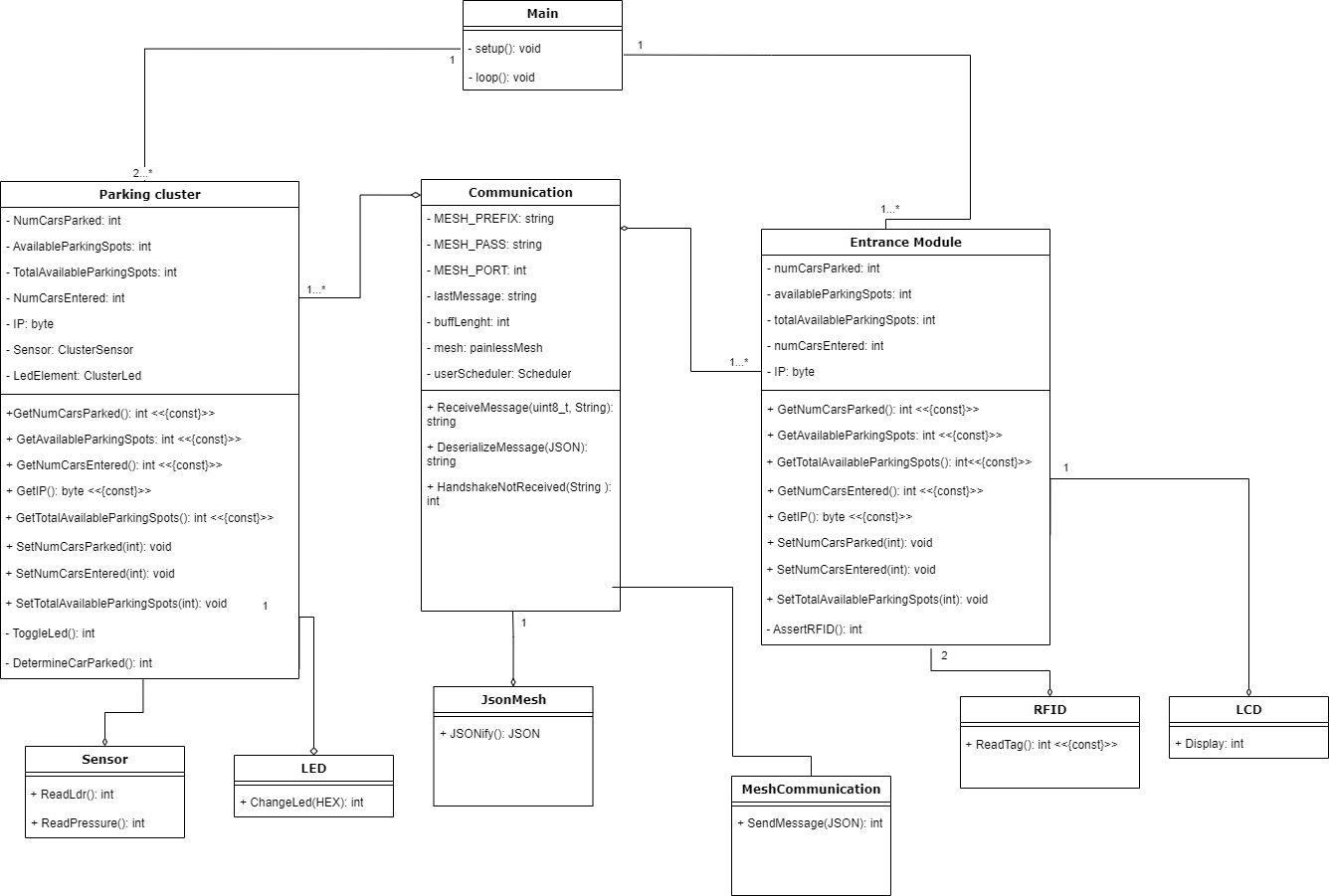
[3.4 LCD display emptiest cluster 13](#_Toc1694621937)

[3.5 LCD display nearest cluster 14](#_Toc1470391692)

# UML Diagram

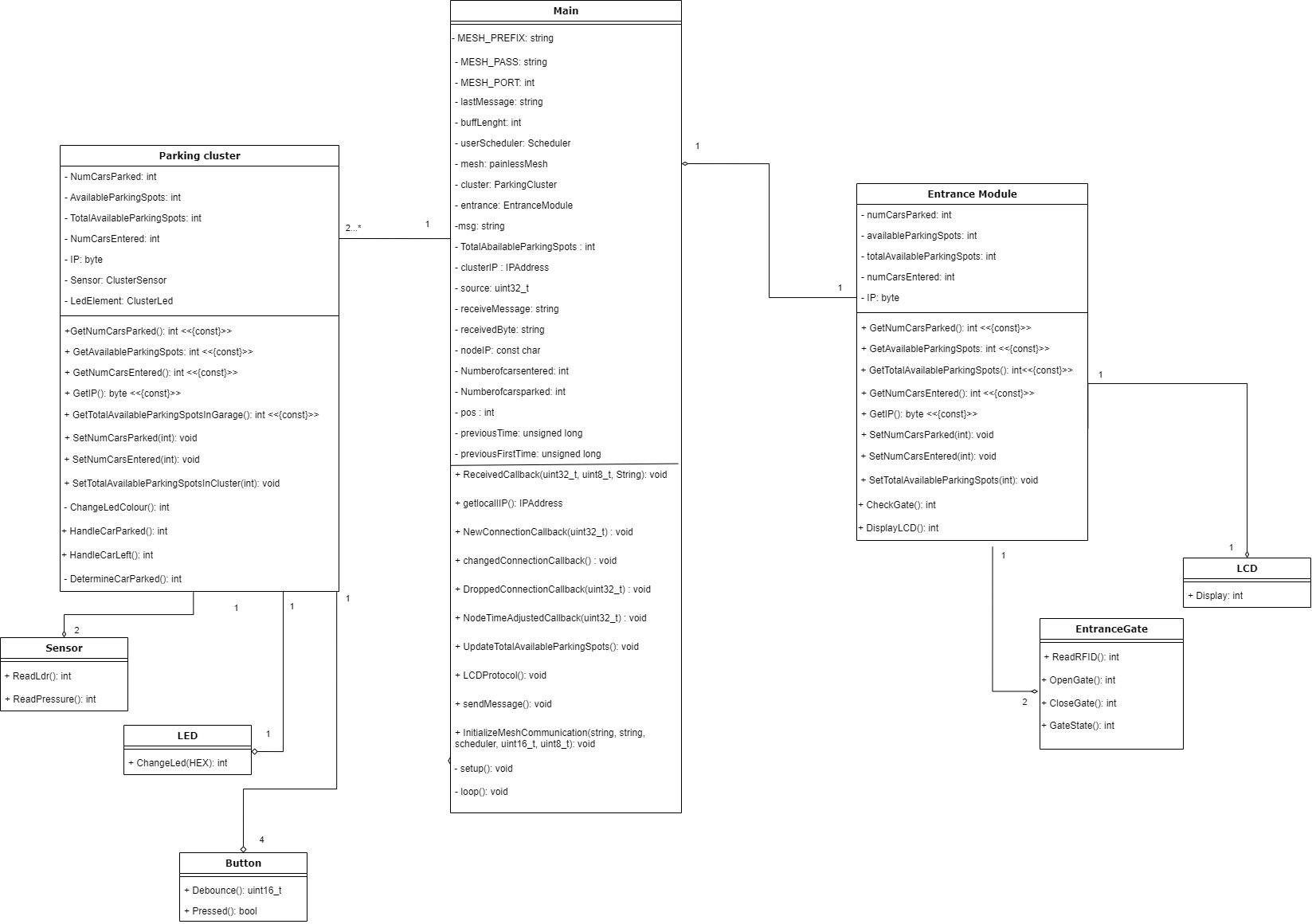
This is the first iteration of the project's UML design. The goal was to divide functionality over as many classes as seen fit. The main was to be kept as clean and compact as possible. Under the main all the functionality gets taken care of by its designated class.

Version 1:



After trying to code the project true to the design, a few problems were faced. The biggest problem was the fact that the communication could only work If the functionality was handled by the main. Going against the initial design. This meant that a revamp of the UML diagram was needed. During this time other things were also discovered to not be optimal, so those had to be changed as well.

Version 2:

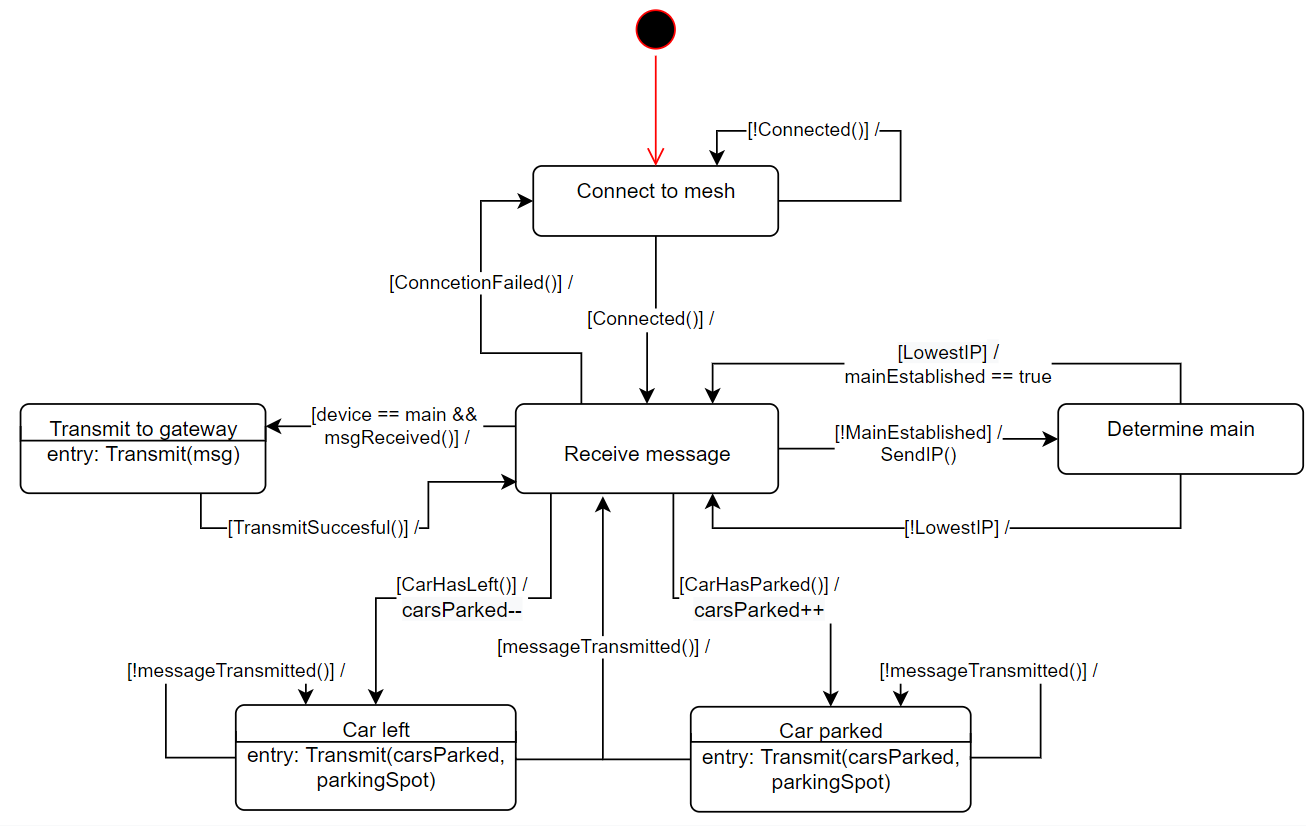


# State Diagrams

## 2.1 Parking cluster module

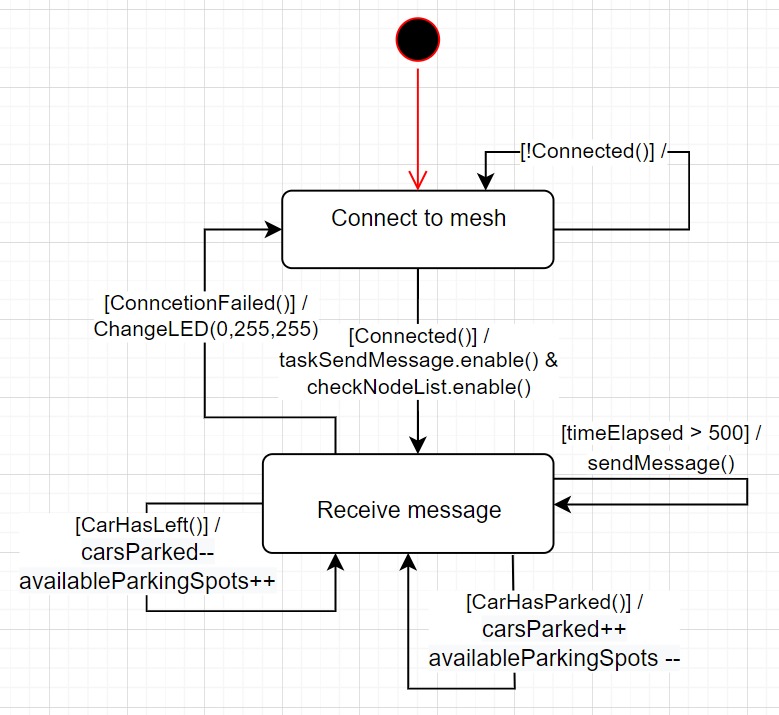
This diagram displays the messaging system of the mesh-network. It shows how the network reacts to messages and how it handles them. It also shows how the picking order of the clusters is decided using the IP addresses. Besides this it shows how the occupancy of the garage is being managed.

Version 1:



For the second iteration of this state diagram, it was heavily simplified, as it was felt that some states were not necessary. So, it was dialed down to two states. This while maintaining the same functionality.

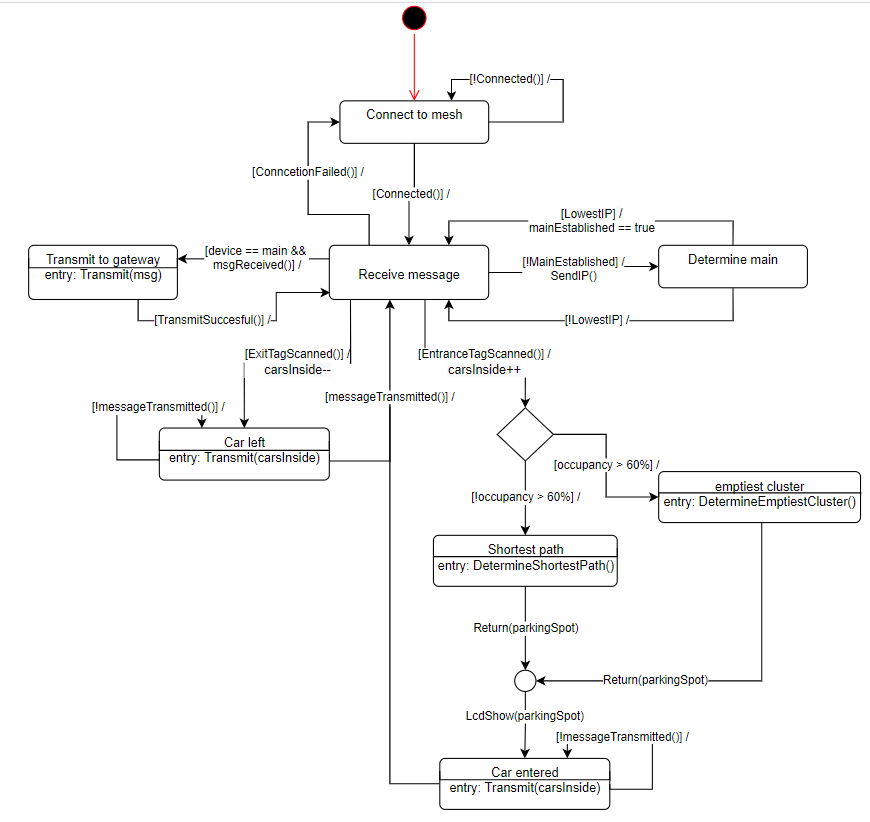
Version 2:



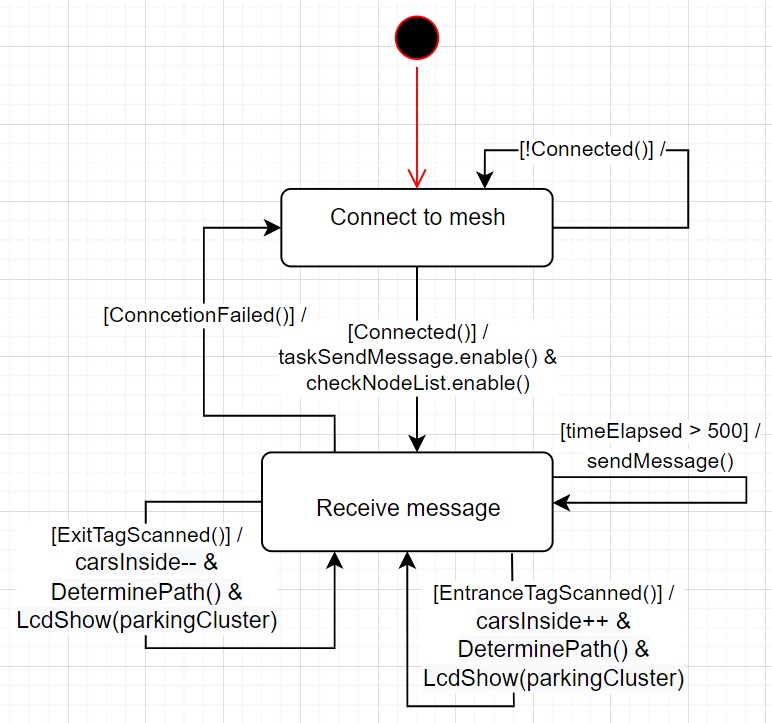
## 2.2 Entrance Module

The entrance module has the same functionality as the parking cluster module. But on top of this the module also has extra functionality that manages the flow of cars that enter and leave the garage. The module can help cars with finding the emptiest cluster to park in or help them find the shortest path to a parking spot.

Version 1:

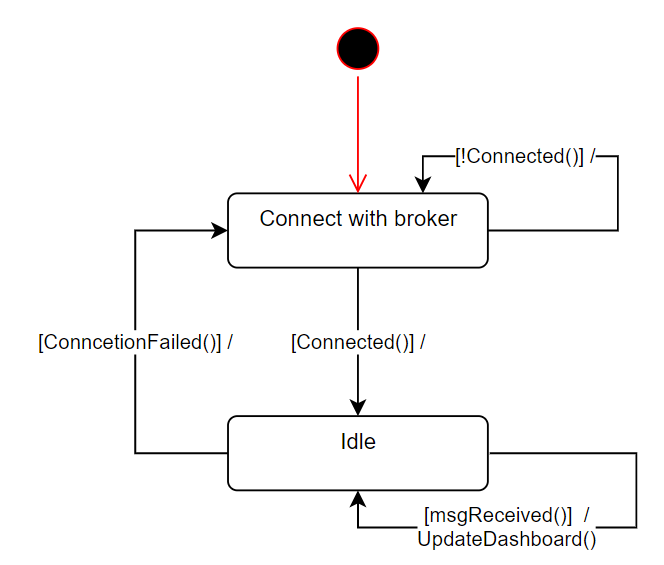
  
For this state machine the same type of changes was made as in the parking cluster state machine. This means that once again the state machine has been simplified to a design with fewer states. There also has been some functionality added that supports the RFID scanner and the LCD screen. A part of the previous iteration is missing, but this part has been moved to a flowchart that will follow later in the document ([3.3 - 3.5](#_3.3_LCD_display)).

Version 2:



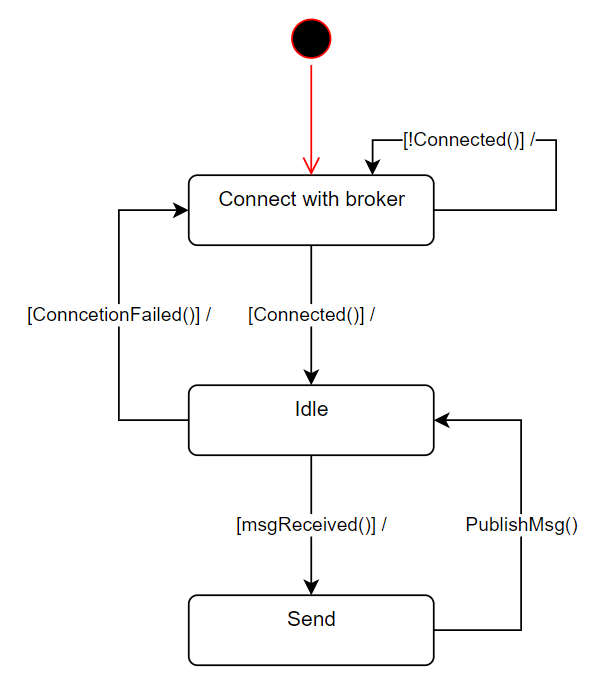
## 2.3 Node Red

This diagram shows how the connection between the module with the broker which is Node-Red. When a connection has been established, the module will go into idle state and wait for incoming messages.



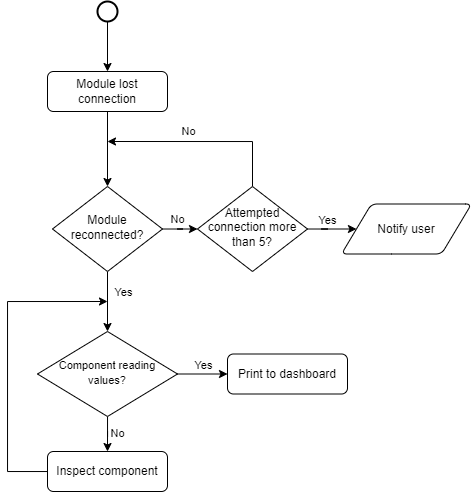
## 2.4 Gateway Module

The Gateway module shows how the system connects with the broker and waits in the idle state once it's connected. When it receives a message, it will make sure it gets sent and when this is done it will wait in the idle state until it receives a new message, or the connection gets lost.



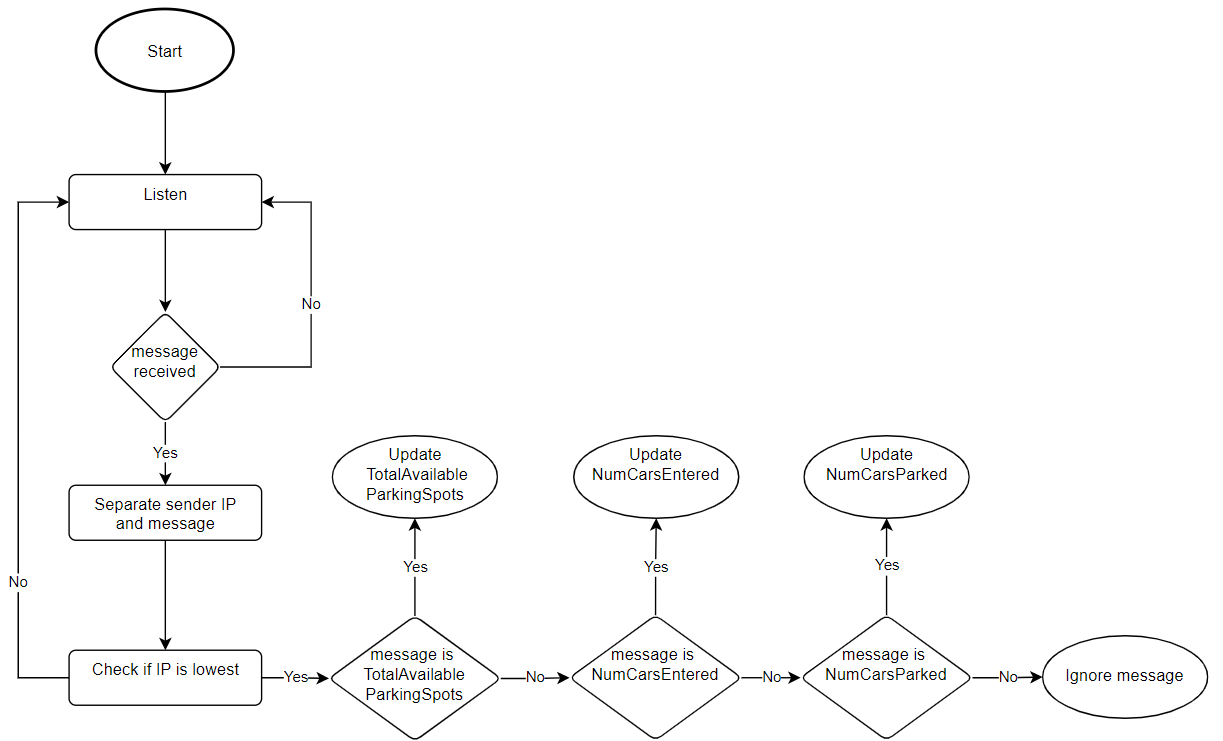
# Flow Charts

## 3.1 Parking cluster / entrance module connection



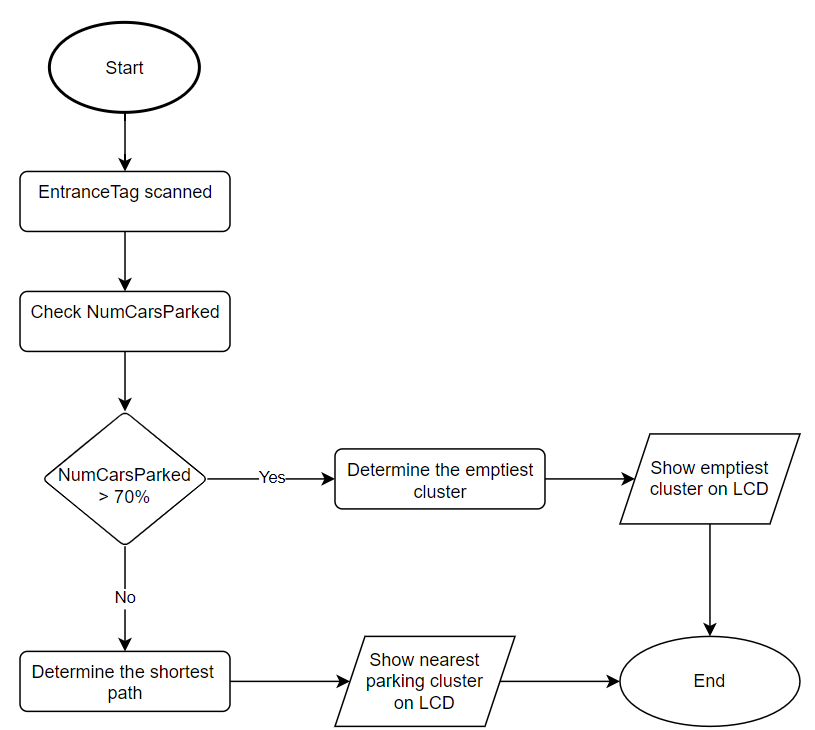
The parking cluster module connection and the entrance module connection flowchart show the process of how the module interacts with the dashboard (See figure 9). When the module is disconnected, the system will attempt to establish the connection again and if it has not found after a certain attempt, then it will notify the user. However, if the connection is to be found then it will go into another scan to check if the hardware component is reading its values. If the component detects a value, then it will print it to the dashboard otherwise the component should be inspected.

## 3.2 Node Red message handling



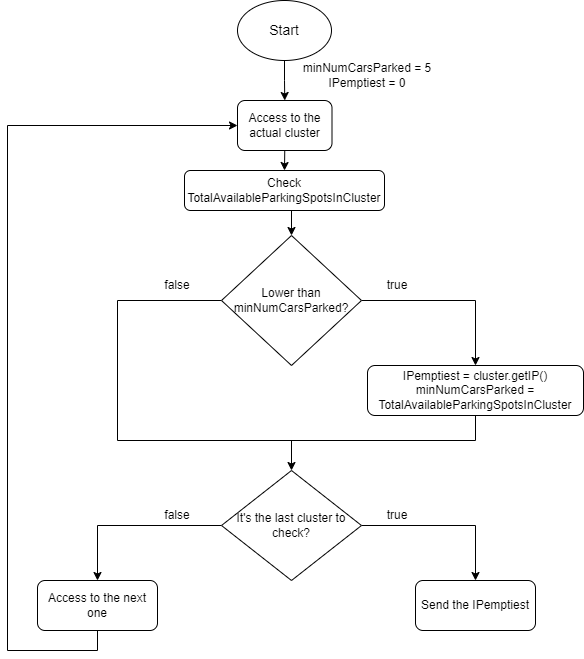
The Node-Red message handling Flowchart gives a clear view on how messages should be handled. This Flowchart appears in the Node-Red state machine in Figure 13. The Flowchart first listens for messages and when a message is received the ID that is sent with the value/state is being separated from the message. Then the Node-Red looks at how to handle the message and how to display it. If the message is not recognized, e.g., the message has been corrupted, the message is ignored because a new message will be sent within a few seconds.

## 3.3 LCD display protocol



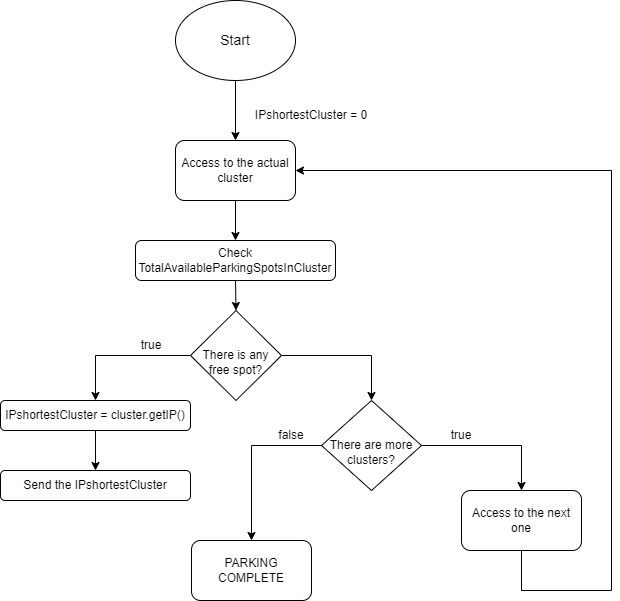
The LCD display protocol flowchart shows how the protocol of determining the advised parking spot for the user must look. In case it is busy in the parking garage (Occupancy > 70%) we want to prevent traffic jams inside the parking facility. Therefore, we advise the user to park in the parking cluster which is the emptiest.

## 3.4 LCD display emptiest cluster



The LCD must indicate to the driver of the car which enters in the parking which cluster is the emptiest (in the case of the parking is busy), so the system will check every cluster, and once all the clusters have been checked, it sends the IP of the emptiest cluster, and the LCD will display it.

## 3.5 LCD display nearest cluster



The LCD must indicate to the driver of the car which enters in the parking which cluster is the nearest (in the case of the parking isn’t busy), so the system will check every cluster until it finds one with free spots. Once it has found the first one, it sends the IP of the nearest cluster, and the LCD will display it and some indications to arrive there.