**Decision Support System of MONICA project H2020**

**1. Purpose of the document**

The purpose of this document is to present the functionality, the installation and the execution of the Decision Support System (DSS), which has been implemented in the context of the Horizon 2020 MONICA project.

**2. Description**

The Decision Support System (DSS) is characterised as the intelligence layer to the MONICA platform, which can solve problems and make interactive decisions by analysing massive streams of data in real time. DSS receives inputs from cameras, microphones and IoT sensors (such as wristbands, environmental sensors) in JSON form and depends on the input, it applies the appropriate intervention strategies and feeds COP with results/alerts.

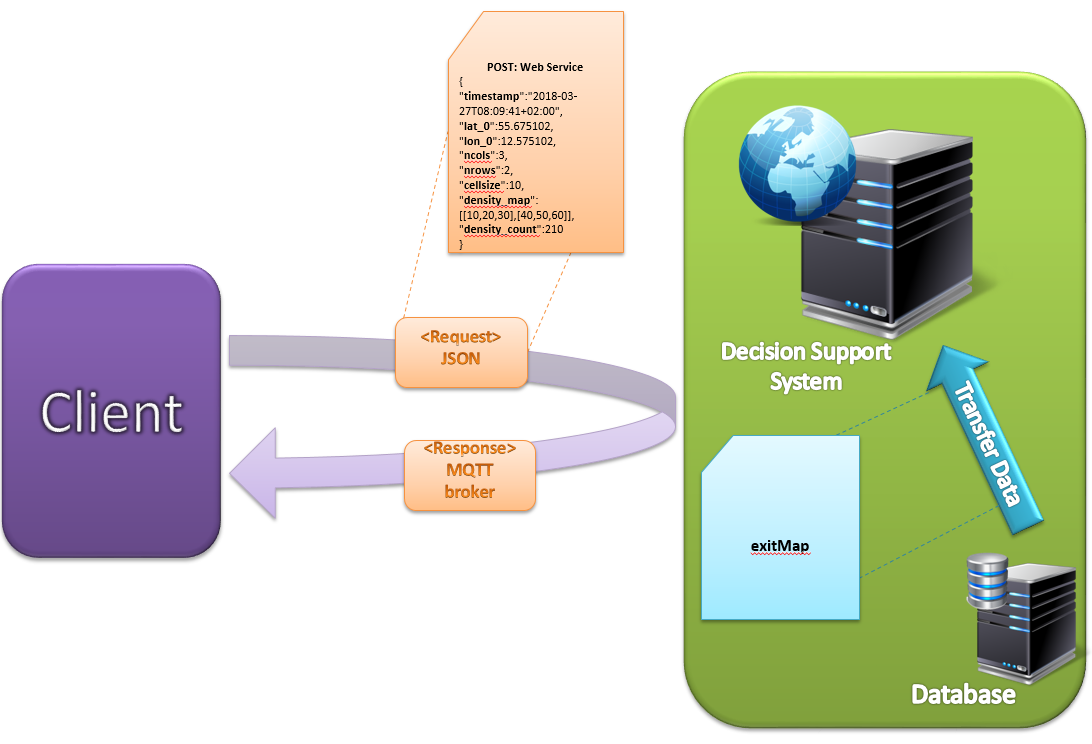
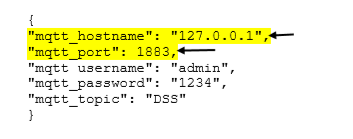


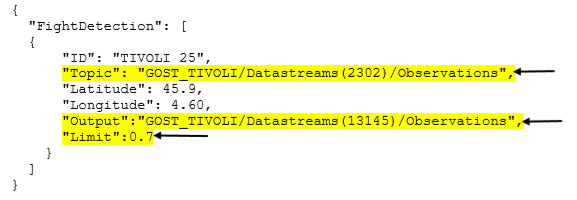
Fig 1. Operation of Decision Support System

**3. Installation**

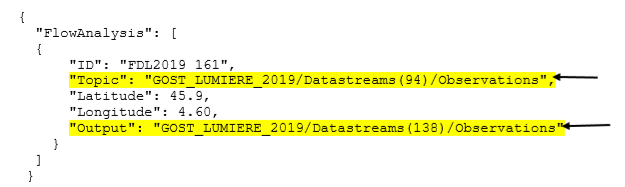
* The user has to download the folder with the binary of the DSS in Ubuntu environment and to store it on the computer. Then, he has to reach it through the command line by typing cd (name\_of\_the\_folder).
* The user has to adjust the parameters of the mqtt server on which DSS will connect to to start receiving and sending data. For achieving that, he has to reach the input folder by typing “cd input”, to open the mqttServerParameters json file by typing “gedit mqttServerParameters. json” and to declare the ip and port of the mqtt server to the fields “**mqtt\_hostname**” (line 2) and “**mqtt\_port**” (line 3) .



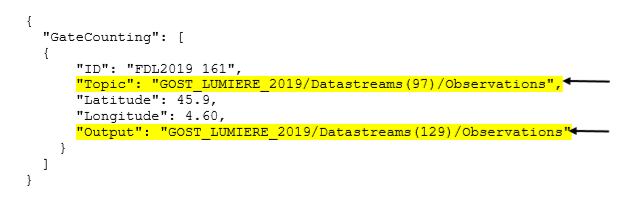
* The user has also to determine the name of the topics, from which DSS will receive and send data. Depends on the functionality there is the appropriate json file in the input folder. He has to open it and declare the input and output datastreams. In addition, he has to adjust the limits if it is needed. The .json files included in the DSS are described below:
* fightdetection.json: It includes the datastreams come from the fight detection module. The user has to declare in the fields “**Topic**” and “**Output**” the names of input and output datastreams relatively. In addition, in the field “Limit”, he has to determine the limit of fight detection confidence. If this limit has been exceeded, DSS will start sending alert messages.



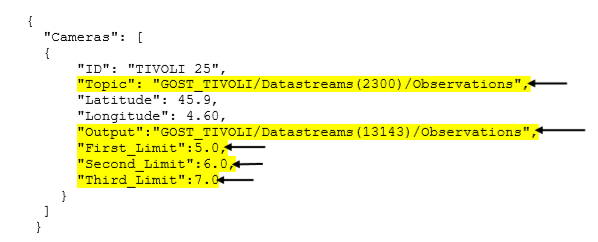
* flowanalysis.json: It includes the datastreams come from flow detection module. The user has to declare in the fields “**Topic**” and “**Output**” the names of input and output datastreams relatively.



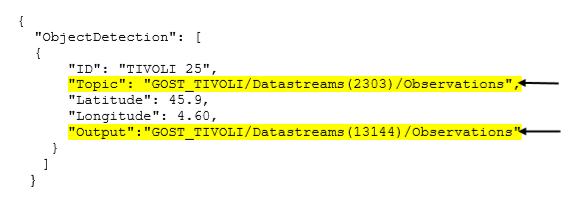
* gatecounting.json: It includes the datastreams come from gate counting module. The user has to declare in the fields “**Topic**” and “**Output**” the names of input and output datastreams relatively.



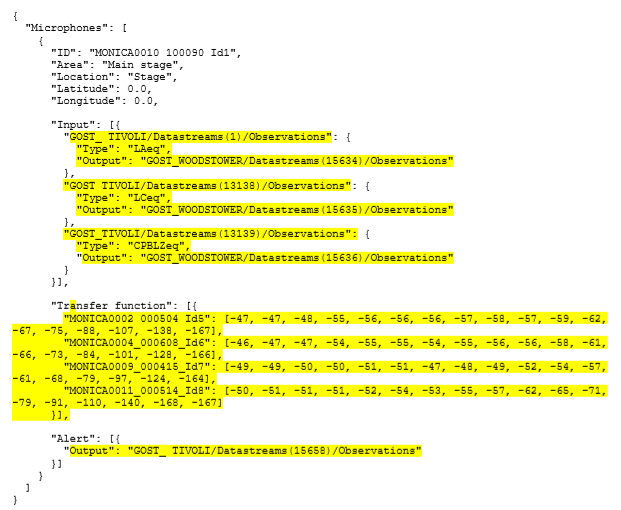
* cameras.json: It includes the datastreams come from cameras. The user has to declare in the fields “**Topic**” and “**Output**” the names of input and output datastreams relatively. The user has also to declare the limits of the number of people per square meter before DSS starts sending warning, alert, high alert, so he has to adjust the fields “**First\_Limit**”, "**Second\_Limit**" and "**Third\_Limit**" respectively.



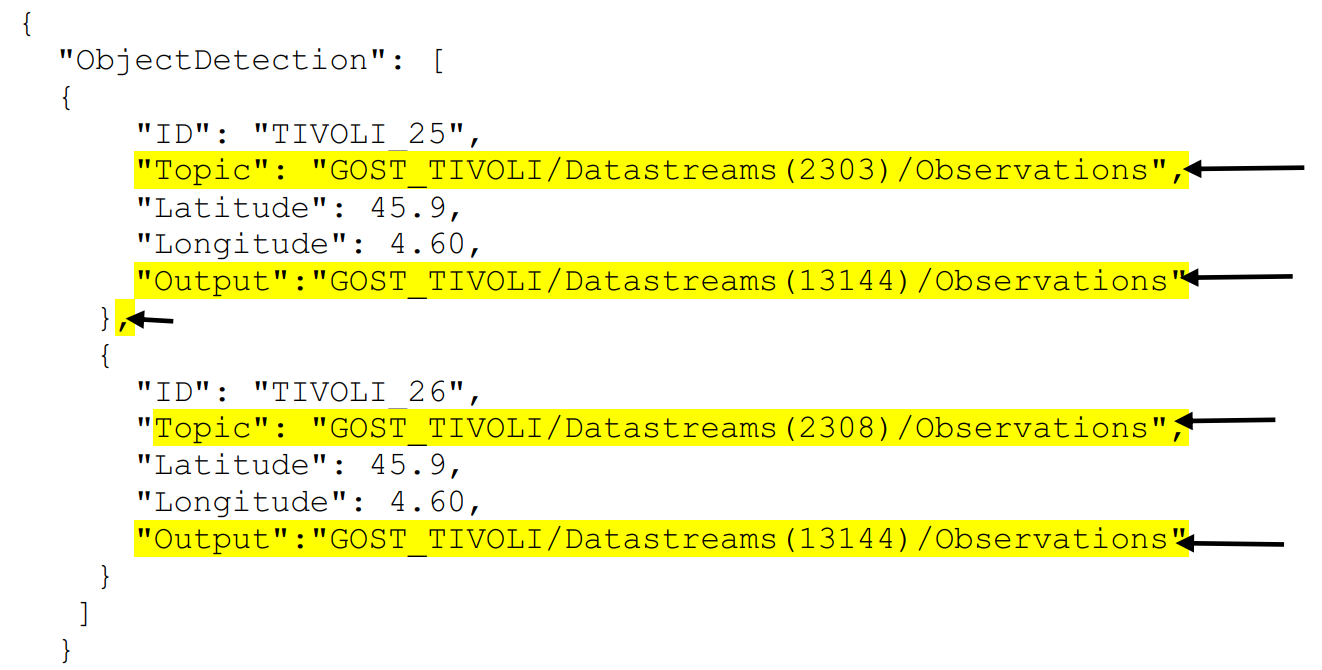
* objdetection.json: It includes the datastreams come from object detection module. The user has to declare in the fields “Topic” and “Output” the names of input and output datastreams relatively.



* microphones.json: It includes the datastreams come from SLMs. The user has to declare in the fields “**Topic**” and “**Output**” the names of input and output datastreams relatively. The user should also, declare to the field “Alert” → “**Output**” the name of the datastream on which DSS will start sending alert messages. Finally, he ought to determine in the field “**Type**” the type of the SLM and in the case that there are more than one SLMs, the name and the transfer function of the other in the field “Transfer function”.

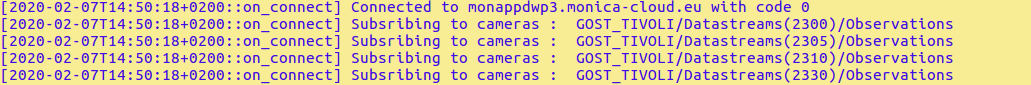


* In the case that the user needs to add new input datastreams, it can easily be done by adding a same structure of the json format, declaring the new input datastream and separating the two structures by a comma.



**4. Excecution**

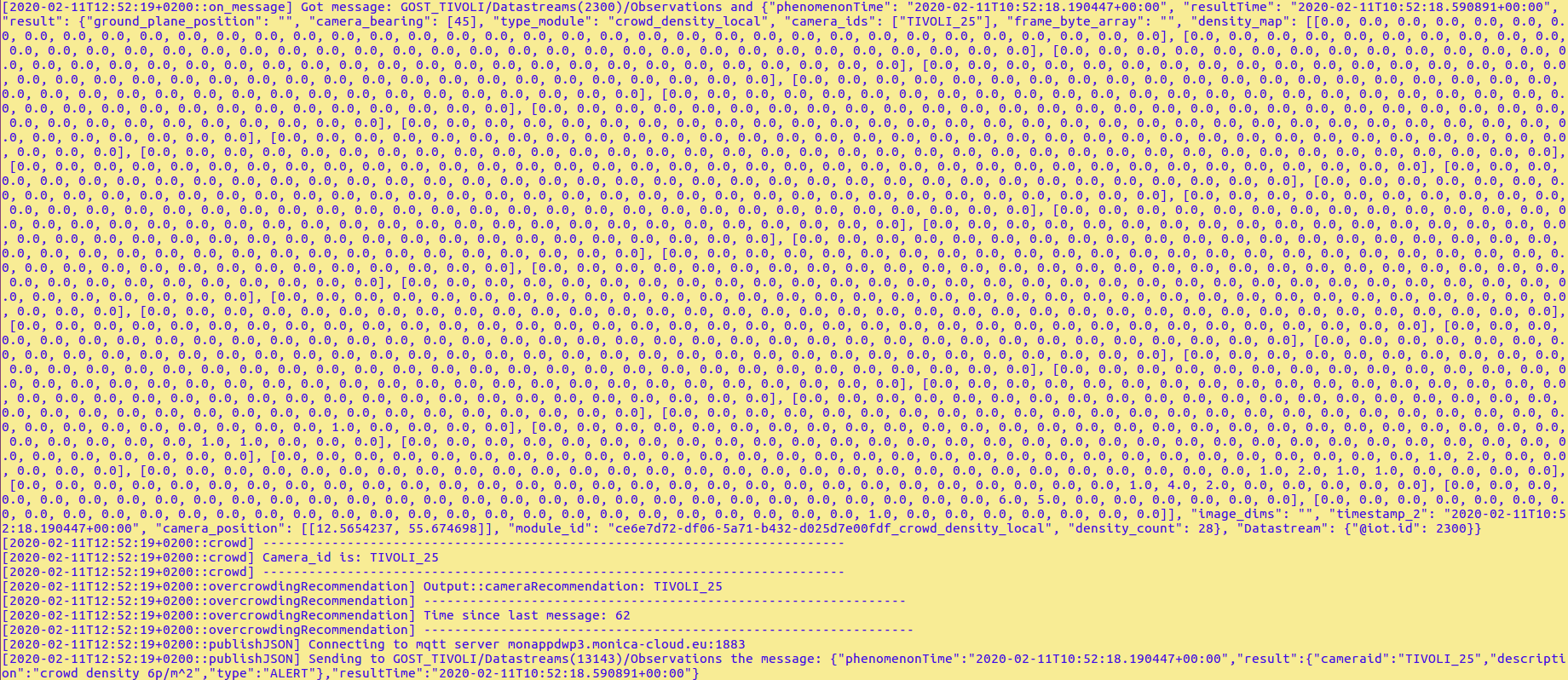
For the execution of the Decision Support System the user has to reach through terminal the folder on which is stored the binary of the dss and then type ./dss. If dss has connected properly to the mqtt server and has subscribed to all topics, the terminal will look like the picture below and soon DSS will start receiving and sending data.



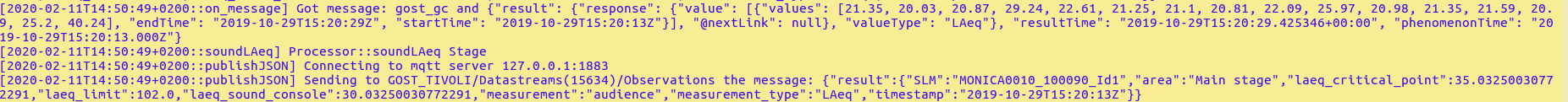
**5. Functionality of DSS**

Depends on the input of dss, it can execute many functionalities.

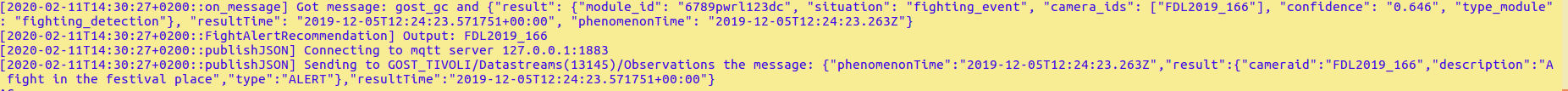
* If it receives the crowd heat map from cameras, it can export alert messages in the case that the number of people per square meter exceeds a limit.



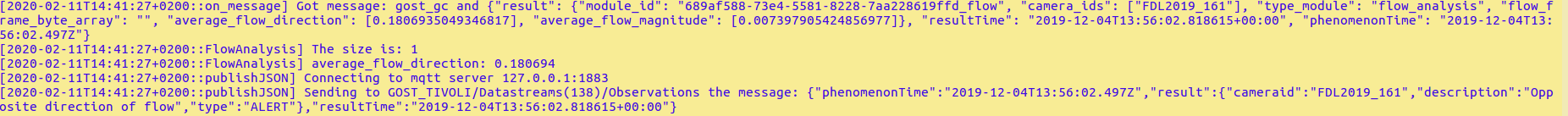
* If it receives the sound heat map from SLMs (types LAeq, Lceq), it can export alert messages in the case that the volume of the sound exceeds a limit. Furthermore, it can export the moving average of the sound per 15 minutes.



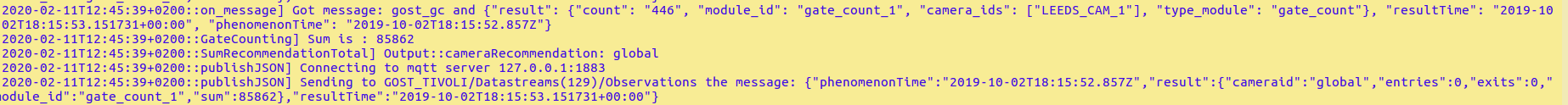
* If it receives data from the fight detection module, it can export alert messages in the case of the confidence exceeds a limit.

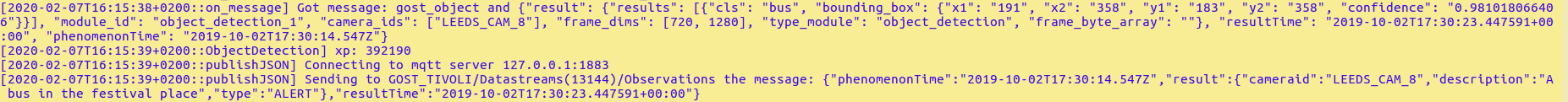


* If it receives data from the flow detection module, it can export alert messages in the case that the direction is the opposite of that it should be.



* If it receives data from the gate counting module it can export the number of entries and exits.



* If it receives data from the object detection module it can export alert messages in the case of unwanted detection.

**7. Usage**

DSS is highly adaptable and configurable, using JSON configuration files. The code base is written in a generic, pilot-agnostic way and has been tested in a lot of pilots, some of them are Fete des Lumieres, Woodstower and Nuit Sonores festival. DSS can handle large amounts of data without any issue. It is written efficiently in C++ and uses multithreading to simultaneously handle data from different MQTT streams using the mosquittopp library.