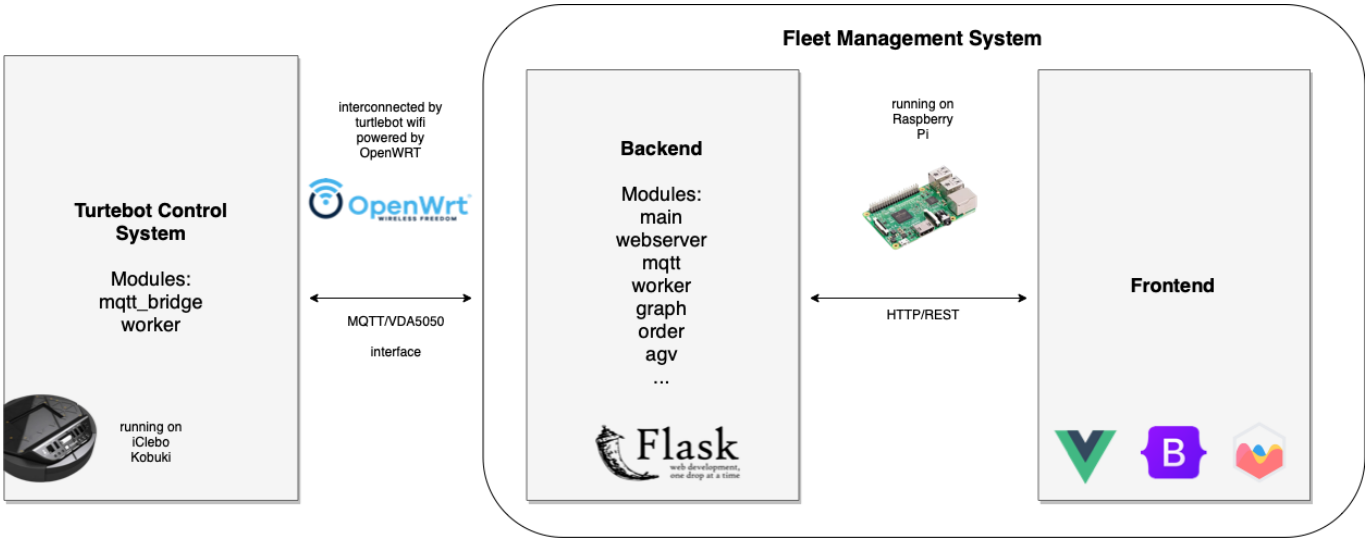
# Design Documentation

Our Design Documentation can be found at https://github.com/amosproj/amos2022ss03-turtlebot-fleet-management/wiki/Design-Documentation.
*The conent below was generated from the wiki. [v_2022_07_26]*




The project consists of two main components, the Fleet Management System (FMS) running on a Raspberry Pi and the Turtlebot Control System running directly on the robots. The two systems are interconnected by a wifi which is powered by OpenWRT and communicate which each other over MQTT by using VDA5050 messages.

# Fleet Management

The Fleet Management System is built as a web application and is divided into a frontend and a backend.

## Backend

The Fleet Management backend coordinates a fleet of up to five turtlebots by:

- Receiving driving requests from a user
- Planning global routes from station A to B
- Avoiding collisions between two robots
- Sending order updates to the turtlebots
- Scheduling turtlebots to fulfill tasks and recharge
- Keeping track of the turtlebots positions and current state

The Fleet Management backend is completely written in Python3. All the additional used python packages are listed in the requirements.txt.

The most important modules are:

- **Main** (main.py)
  Starting point of the whole Fleet Management. Reads in the config file, creates the graph using

vmap_importer, creates the AGVs based on the config and starts all the threads.

- **Webserver** (webserver.py)
  Starts the webserver and receive and executes requests from the frontend. Uses Flask for building the webserver.

- **MQTT** (mqtt.py)
  Establishes the MQTT connection with the broker, receives and handles incoming VDA5050 packets and sends outgoing ones. Uses Paho MQTT for building up the MQTT connection.

- **Worker** (worker.py)
  Distributes pending orders to the AGVs and handles the correct execution of the assigned orders for an agv.

- **Graph** (TurtleGraph.py)
  Provides the structure for the driving course and all the required functions which are related to that.

- **Order** (Order.py)
  Provides the structure for a driving order and does the main part of avoiding collisions between different orders.

## Frontend

The Fleet Management frontend displays the driving course with the locations of the different AGVs and their routes to drive. Also it shows two tables. One with all the Orders and associated informations and one with all the AGVs and their statuses.
Also the Frontend gives the user the options to create an order for an AGV and to cancel orders.

The Frontend is build with HTML and JavaScript. It also uses Vue.js as a framework, Bootstrap for a good looking UI and Chart.js for the graph.

# Turtlebot

The Turtlebot part of the software is build using ROS2 nodes which are written in Python and consists of two main modules:

- **MQTT Module** (mqtt_bridge)
  The MQTT module establishes a MQTT connection with the broker using mqtt_bridge and de/encode messages using vda5050_msgs.

- **Worker Module** (worker_node)
  The worker module fulfills incoming orders and sends position and status updates back to the FMS.