Event based control
Project report

Biernat Damian	226 309
Brzozowy Martin	226 339
Maliszewski Marcin	227 347
Mróz Sebastian	218 559
Sil Aleksander	218 576

October 2019

1 Project description

Idea of the project is to design the system able to control model of the AGV's in task of transporting cargo from dock to the warehouse in the common space. Space between origin and destination is divided into grid of known size. Controlling system has the knowledge about the state and the position of the AGV's based on their communicates. It is assumed that AGV is connected directly to the management system with some link. Robot has finite acceleration and deceleration values. AGV will proceed to travel between the cells of the grid only if system allows for that. Basic functionalities of the system is to avoid collisions between AGV's. For extended task system has also manage time of arrival AGV to the warehouses and react to the changing environment. In summary, project is divided into 2 parts. Designing the AGV model and designing the controlling system.

1.1 Team responsibilities

Damian Biernat AGV modeling in Matlab/Simulink

Martin Brzozowy Matlab-Python link & Matlab modeling

Marcin Maliszewski Visualization in Python

Sebastian Mróz Visualization in Python

Aleksander Sil Controller modeling in Matlab/Simulink

1.2 Controller

The controller's goal is to provide such a path that allows a collision-free move and correct delivery, assuming the delivery time has a higher priority than path length. System components presented in figure 1 consists of:

- AGV Robots
  - Robot id
  - Current state
  - Next state
  - Path - list of states
- Master Controller
  - State evaluation
  - Logic controller
  - Effectiveness controller

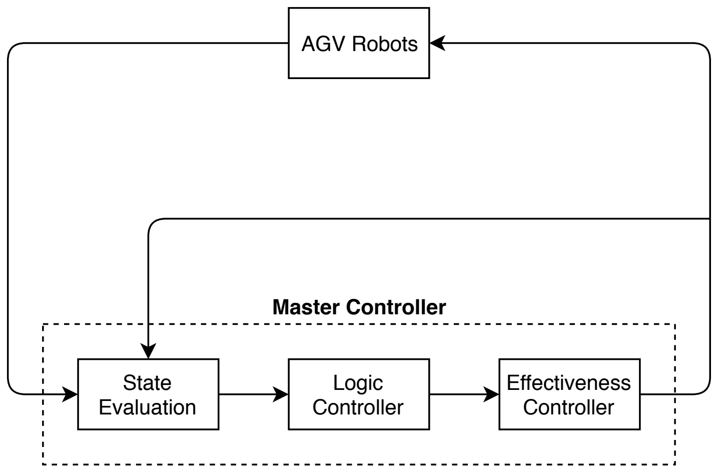

Figure 1: Diagram of control system

As an input controller gets the actual robot position (actual state). The master controller assigns the robots with different paths to deliver cargo from ship's cargo hold to cargo storages. The paths are assigned at the beginning of the script, when robots are in the base and when robots successfully deliver cargo to the storage.

During runtime, robots send passage request when they want to pass a crossroad. The crossroads are nodes with designator $C*$ on figure 2. The controller checks for collision and decide which robot should pass the crossroad first. For detection of collision and other forbidden states, the occupancy grid is used. The occupancy will be a 3D matrix:

$$M \in \mathbb{N}^{SxSxR}$$

where S - number of possible states, R - number of robots

1.3 Workflow - delivery map

The robot workspace (fig.2) describes a possible paths to deliver packages. Each robot starts from *Base*. Tasks are initialized in ST position and finished in *SH*. The start and goal of the task are described by the list of order which is assigned to the robot and the robot holds this information. (fig.1).

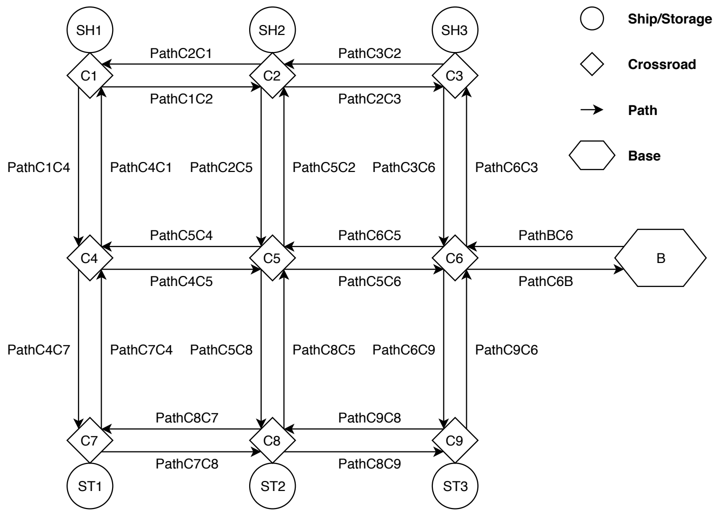

Figure 2: Workflow diagram

1.4 Visualization

Visualisation will be performed using python. It will be based on fig. 2 and will show occupancy of the system by the robots as well as their paths to the target. Each agent (and his path) will be color coded and his actual location will also highlight number of the robot. Visualization will also show points of intersect of robots paths as potential points of accident. Additionally information about type of cargo that arrives on given dock and how much of it is still left on it. Idea for the visualising localization and path of the robot is to better understand algorithm steering robots and possibly faster identify and resolve problems with evaluation of the state and/or controllers. It will provide needed simplification for purpose of making quick and accurate observations. This will be possible by the possibility of slowing down visualization in critical points (robots intersecting path of each other) and going step-by-step on the history of the recorded situation.

2 Tools

The following tools have been chosen to complete the project:

- Python
- Matlab & Simulink
- Visualization libraries
- Git

- Trello
- (opt.) State-flow for Simulink

The project will be developed using Matlab with Simulink and Python. Additional libraries for visualising operation of the system are taken into consideration. Using them will simplify the debugging process.

For purpose of managing the project, proper workflow and splitting the tasks Git and Trello solutions will be used. Git will be used as a version control system and will allow segregation of duties. Trello will be used as sudo-scrum project managing tool.

3 Milestones

1.10.2019—31.10.2019	Generate first report, finish model diagram, picking proper tools, initialize repositories and create Trello workpage, split the duties.
1.11.2019—15.11.2019	Create basic functionalities in python/matlab, adding visualization to the script. Decision of tool used for this project.
16.11.2019—30.11.2019	Further developement of the system, extending the system's capabilities, debugging
01.12.2019—20.12.2019	Project evaluation, decision of expanding system capabilities.
21.12.2019—6.01.2020	Christmas break, filling batteries, preparing mentally for the end of the project.
07.01.2020—31.01.2020	To be decided according to the state of the project: adding new functionalities or finishing basic capabilities of the system