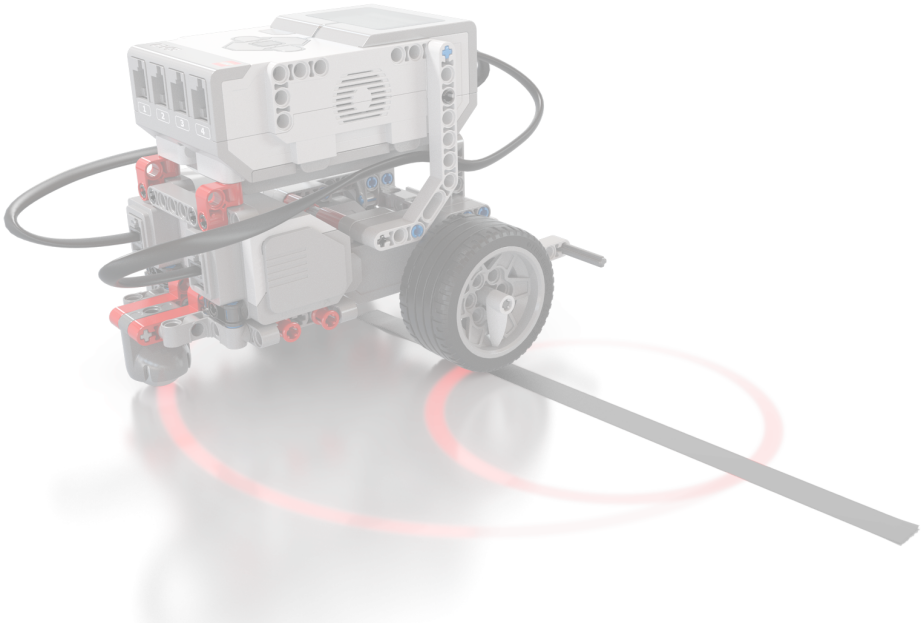

Autonomous Vehicle Network using Lego Mindstorms EV3

DT0227 Software Engineering for Autonomous Systems

Academic Year 2022-23

Prof. Davide Di Ruscio

Members:
Aswin Palathumveetil Jagadeesan (288721)
Rahul Pankajakshan (288722)

Table of Contents

Goal

The aim is to create and deploy an autonomous vehicular network utilising Lego Mindstorms EV3, which allows multiple vehicles to communicate their state, motion statistics, and emergency situations to each other through a global planner. This will facilitate intelligent decision-making based on the received information, leading to a seamless driving experience. The architecture employed is decentralised, meaning that each vehicle and server has its own logic implemented within a MAPE-K loop system.

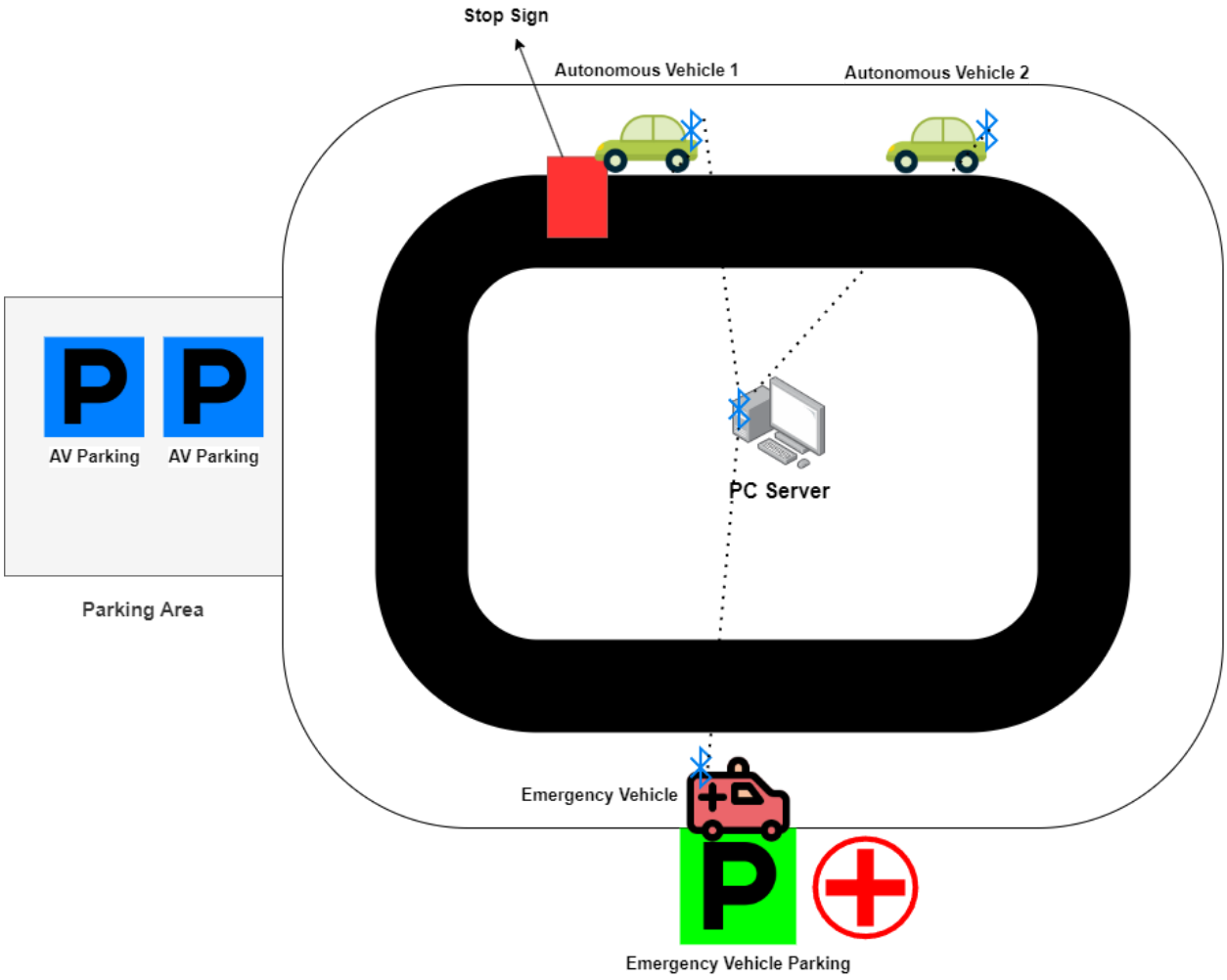

Fig. 1: Layout of the environment.

Sensors & Effectors

The Lego Mindstorms come with different kinds of sensors and effectors. Following are the sensors that we have used on the Lego Mindstorms EV3 brick to create an autonomous vehicle:

- The *ultrasonic sensor* measures the distance of the obstacles ahead of the vehicle. It can read from 1 to 250 cm.
- The *colour sensor* monitors the colour as well as the intensity of reflected light. Capable of detecting seven colours (black, white, red, blue, green, yellow, brown) plus the absence of colour.
- The *touch sensor* acts as a switch with a True value when pressed and a False when released.

The effectors on the vehicle and their purposes are as follows:

- The vehicle is manoeuvred along the track by the motion of the *servo motors* connected to the wheels.
- The EV3 brick has built-in programmable *LED lights* on its top that indicate various states of the vehicle.
- When there is a change in state, the built-in *speaker* in the brick will produce appropriate sounds.

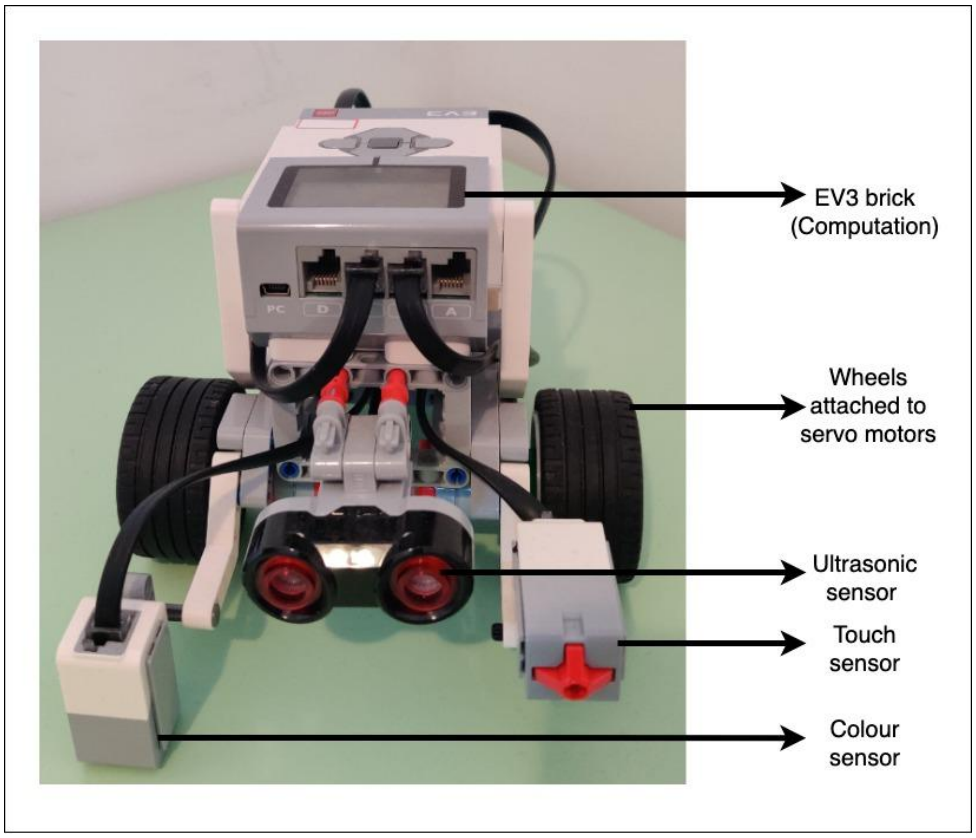

Fig. 2: Sensors and effectors of the autonomous vehicle

Monitor

The monitor is responsible for observing and collecting information about the system's state from various sources, such as sensor readings and messages from the server. The monitor then analyses this information and passes it to the analyser for further processing. Specifically, in the Lego Mindstorms EV3 Autonomous Vehicle Network, the monitor receives messages from the server and combines them with the sensor readings to form a complete picture of the system's current state. The monitor updates the knowledge base with sensor readings as well as commands received from the server.

Analyser

The information gathered by the monitor is analysed to identify and understand the change in the environment and the information received from the server. Based on these, the planner component is notified to take the appropriate plan of action.

Normal Autonomous Vehicle:

- By analysing the modifications in state variables and flags within the knowledge base, vehicle state changes are identified. These changes can cause the vehicle to transition into different states, such as stop, emergency, and parking.
- Stop sign identification (red colour) by analysing the colour on the track.
- Emergency state identification by analysing specific messages from the server.
- Parking spot identification (blue colour) by analysing the colour on the track as well as messages from the server
- Checking the occupancy of the parking spot by analysing the distance monitored using the ultrasonic sensor.
- The vehicle can return to the track from a parking area by analysing messages received from the server
- Monitored ultrasonic readings are also analysed to detect obstacles on the lane.
- EV3 analyses the touch sensor value to determine whether the vehicle has crashed. Messages from the server are analysed to determine whether the crash is critical or not.

Emergency Vehicle:

- It is analysing alert instructions received from the server for moving to track.
- It is using ultrasonic sensors to identify the location of the emergency spot along the track.
- Upon returning from the emergency spot, it analyses colour sensor values to determine the appropriate parking spot for the emergency vehicle.

- If there is a current emergency situation, another emergency situation will only be addressed after the current situation has been resolved.

Planner

The analysis from the analyser as well as the data from the knowledge base is used to derive the plan of action. It analyses the system's current state, the goals to be achieved, and the available resources to determine the best course of action. The Planner then generates a plan and this plan is communicated to the executor component, which carries out the necessary steps to implement the plan. The decisions are taken based on a rule based approach.

Plans of Action for a Normal Vehicle

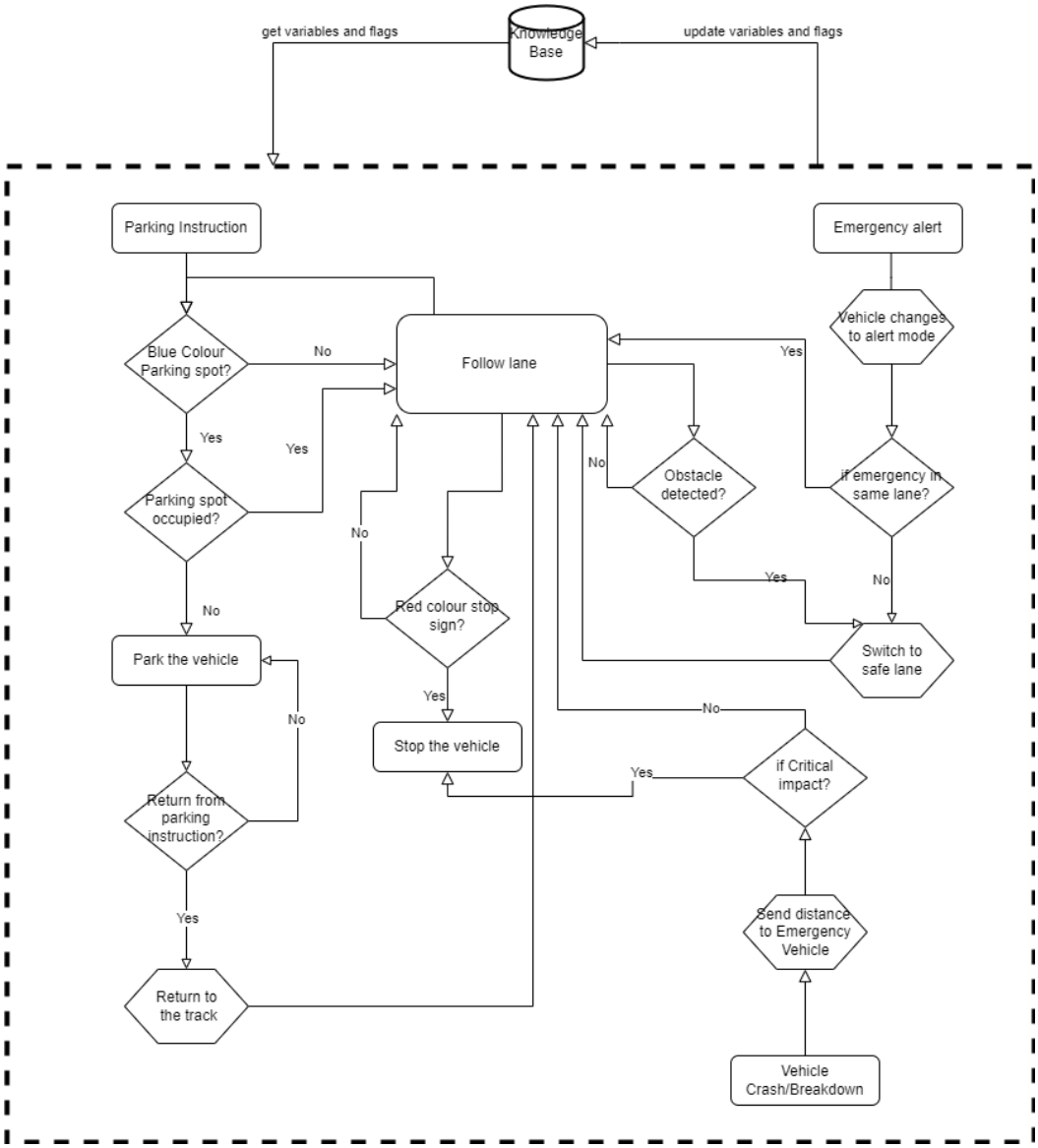

Fig. 3: Flowchart of Normal AV

Plans of Action for an Emergency Vehicle

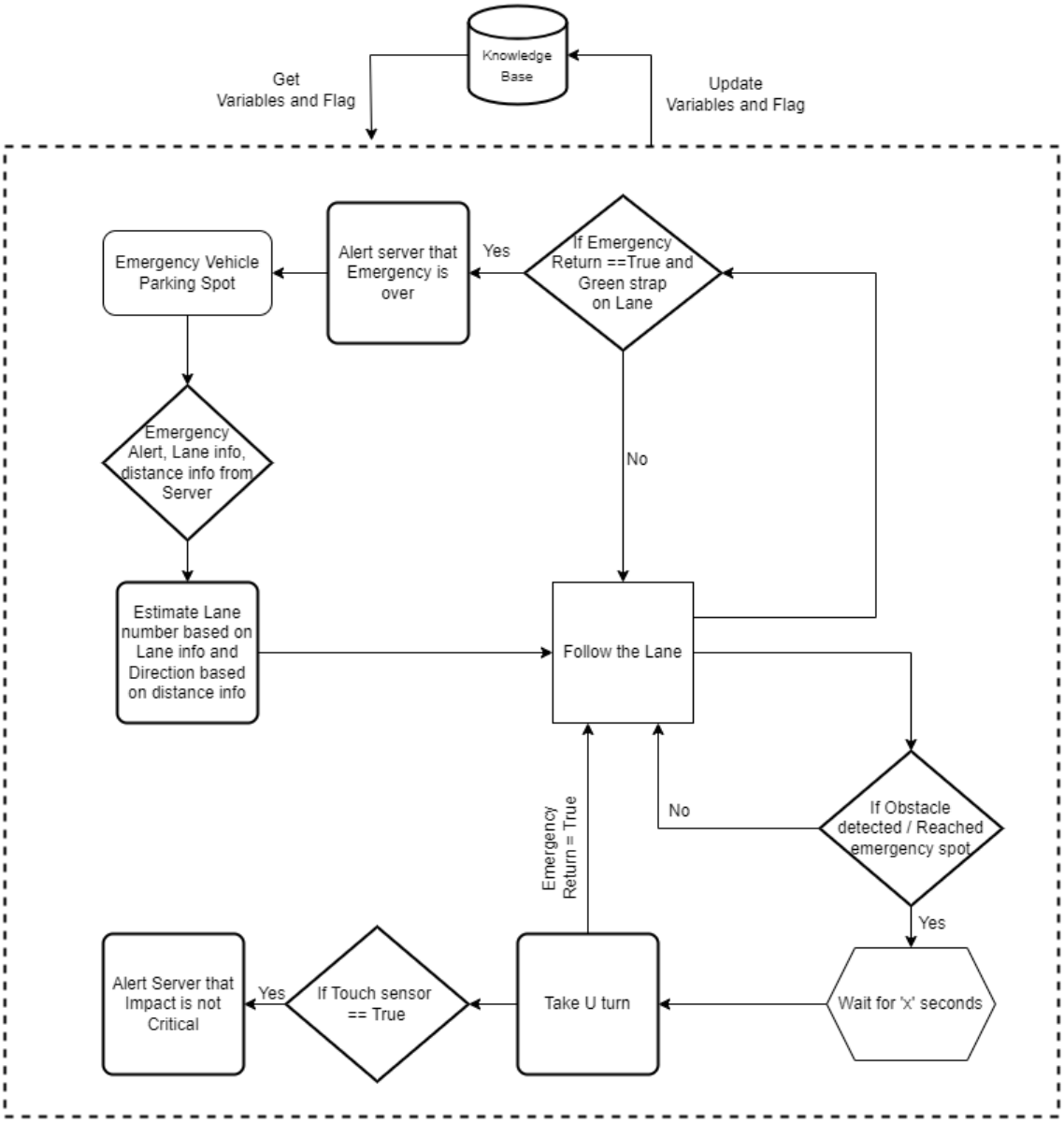

Fig. 4: Flowchart of Emergency AV

The knowledge base is updated by the planner in accordance with the decisions made.

Constraints

- The vehicles/clients need to be connected to the server in a certain order.
- The vehicles are programmed to travel in one direction, except for the emergency vehicle. It will take the shortest path back to its starting point.

- It is assumed that when the emergency vehicle is in motion, the only obstacle in its lane is the crashed vehicle.

Executor

In the MAPE-K feedback loop, the executor is tasked with executing the actions determined by the planner. This includes controlling the movements of the robot, activating indicator lights, and playing sounds based on the planner's instructions. When communicating with a server, the executor is also responsible for transmitting information and acknowledgements to the server.

Sequence Diagrams

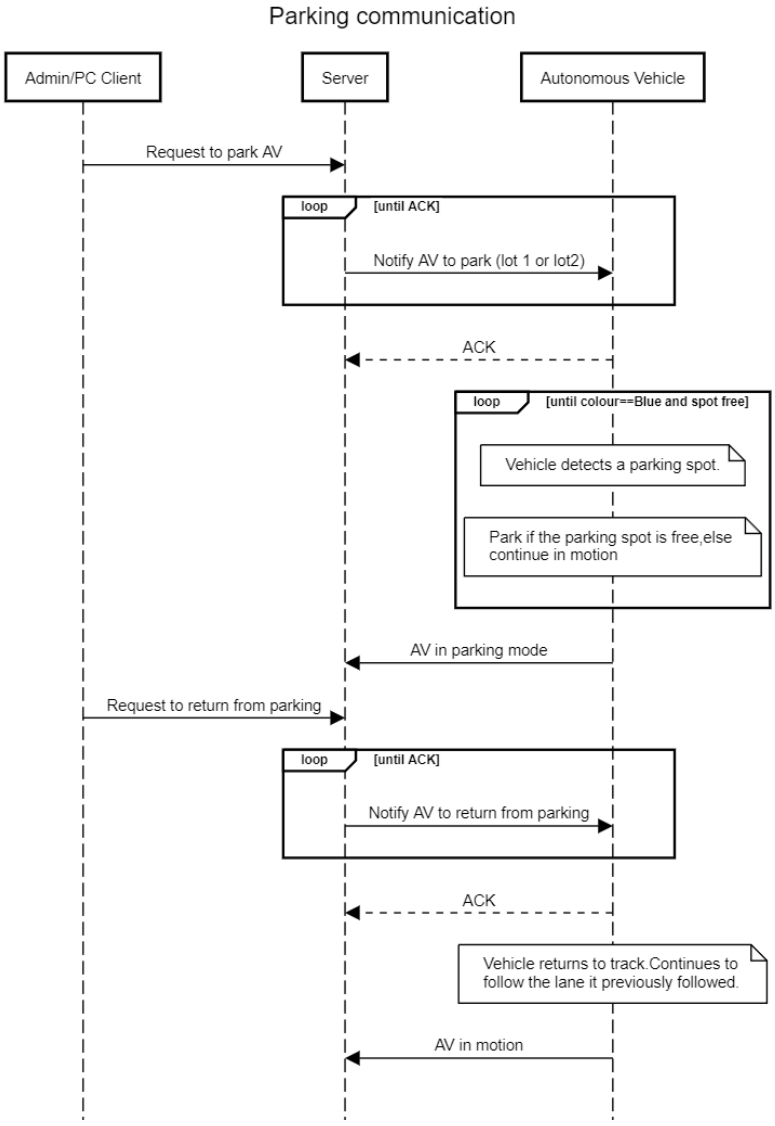

Fig. 5: Parking communication sequence diagram

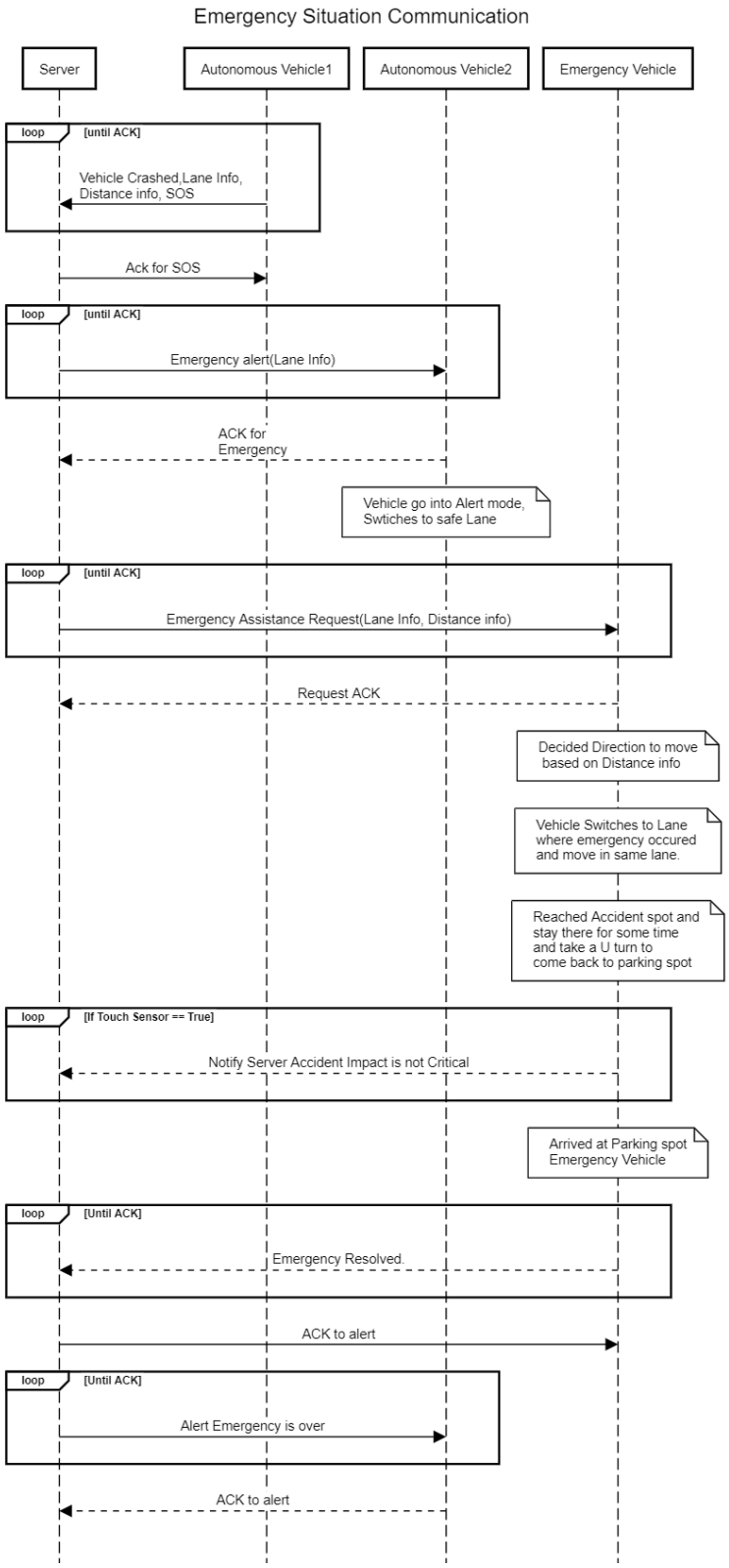

Fig 6: Emergency situation sequence diagram

Knowledge

The knowledge base comprises different variables, flags, sensor parameters, and configurations that define the behaviour of the vehicle in the network. The log of change in state can also be stored in the knowledge base. By continually updating the knowledge base with new data and information, the system can adapt and respond to changing conditions in its environment.

Communication

As there are no message brokers for communicating between multiple EV3 bricks, we used Bluetooth mailboxes for communication. The documentation to establish the connection between multiple EV3 bricks and PC client/server can be found here.

Message format

The following are the various types of message categories, along with their respective status codes.

Message type	Message Direction	Description	Status code
Crash/SOS	Crashed AV => Server	Crash/SOS alert to server.	10+lane number (0 or 2)
Crash ACK	Server => Crashed AV	Crash ACK from Server to crashed AV when the 'SOS' alert is received.	200
Emergency Alert to Emergency Vehicle	Server => Emergency AV	Emergency alert to the Emergency vehicle from the server.	10+lane number (0 or 2)
Emergency Alert ACK	Normal AV, Emergency AV => Server	Emergency Alert ACK from both AVs and emergency vehicle to server.	100
Emergency Alert to Normal AVs	Server => Normal AV	Alert from server to Normal AVs in an emergency.	10+lane number (0 or 2)
Emergency Over	Emergency AV => Server	The emergency vehicle reached the parking after the emergency resolution.	0

Emergency Over ACK	Server => Emergency AV	ACK from server to emergency AV when ‘Emergency Over’ message is received.	200
Emergency Over alert to Normal AV	Server => Normal AV	Alert Normal AVs when the emergency is resolved.	0
Emergency Over ACK - Normal AV	Normal AV => Server	Emergency over ACK from AVs to the server, when the server sends ‘Emergency Over	300
Parking Instruction	Server => Normal AV	Server command to AV for parking	999
Parking Instruction ACK	Normal AV => Server	ACK from AV to server for Parking Instruction.	200
Return from parking instruction	Server => Normal AV	Server command to AV to exit the parking	666
Exit parking instruction ACK	Normal AV => Server	ACK from AV to the server for Return from parking instruction	222

Additional Details

Sensor Noise:
During the vehicle's traversal around the track, it was observed that the sensors detected a significant number of colours, specifically yellow, green, and blue, in addition to white and black which are the colours of the track. This faulty colour detection resulted in the vehicle making erroneous decisions, as each colour held different purposes. The recorded colour sensor values of the track were compared and evaluated against the colour sensor values of the intended reference colours (strips of colour papers) for our project. The vehicle was programmed to take action only if the colours detected matched the reference colours.

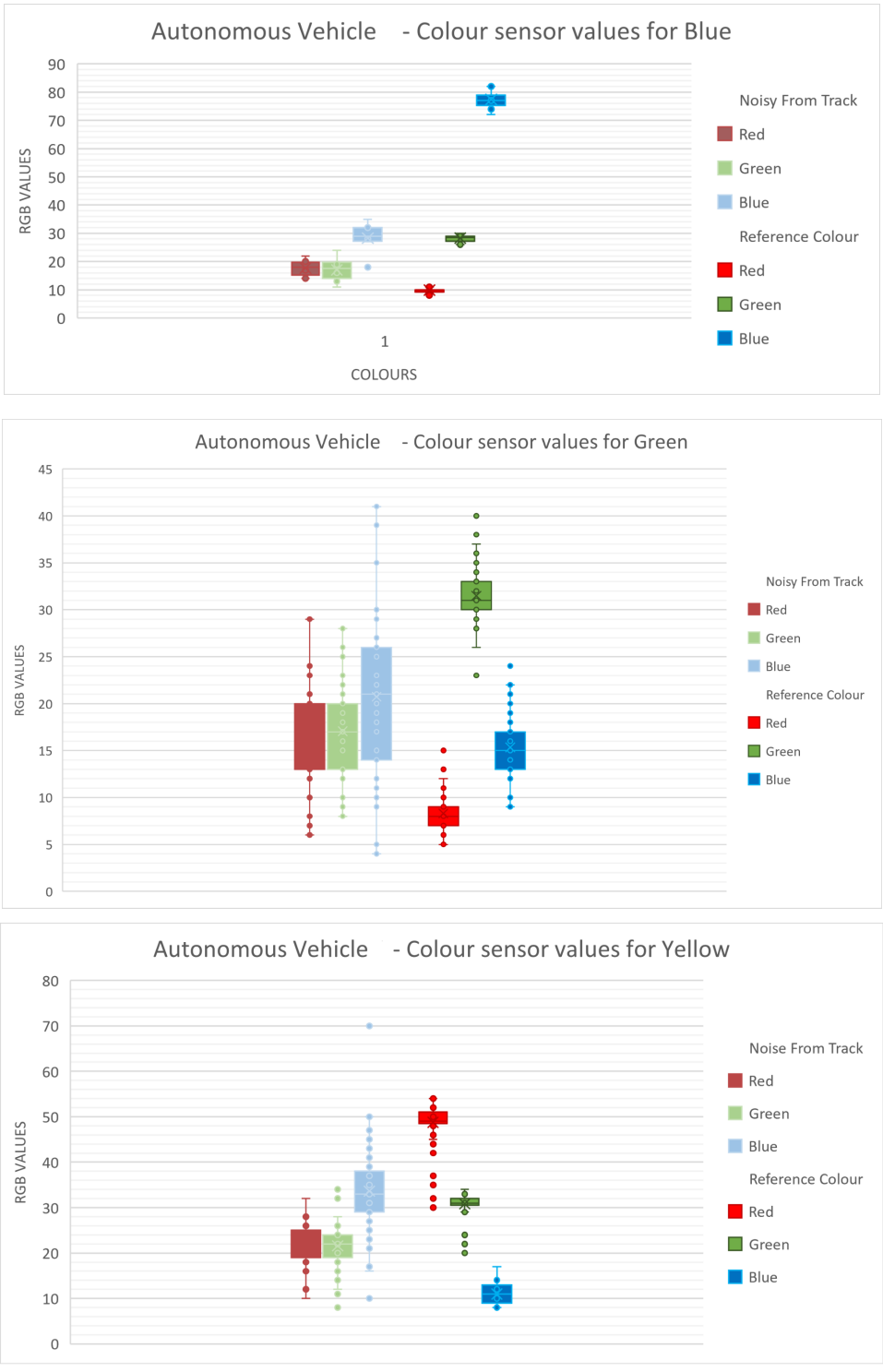

Fig. 8: Colour sensor values comparison graph

Track for the experiment:

A rectangular shape track with outer lanes in white colour and inner lanes in black colour (Fig.7).

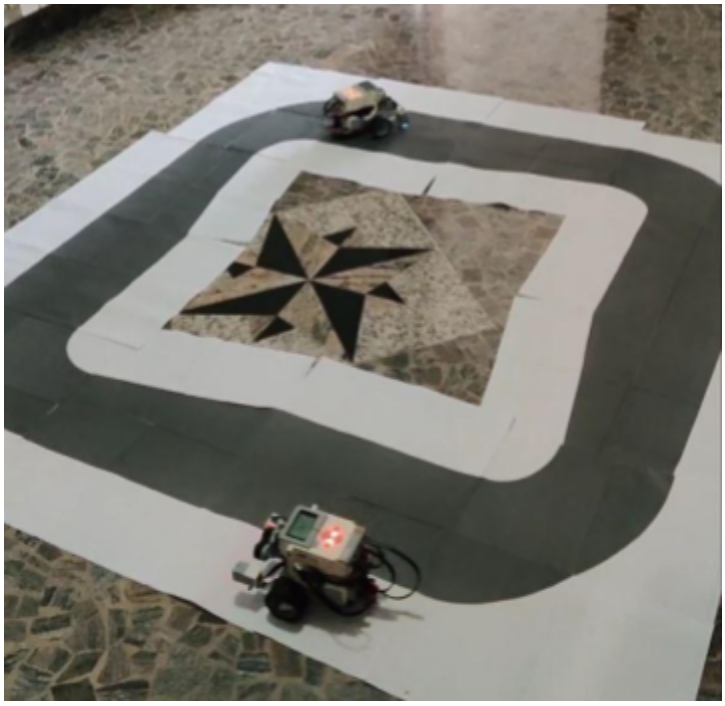

Fig. 9: Track used for the project

Technologies used - EV3 and PyBricks:

EV3 is a robotics platform produced by LEGO that allows users to build and program robots. The EV3 platform consists of programmable brick, motors, sensors and other hardware components that can be combined to create a variety of robots.

Pybricks is a Python library that provides an easy-to-use API for programming LEGO robots using the EV3 platform. It is designed to simplify the process of programming robots by providing a high-level interface for accessing sensors, motors and other hardware components. Pybricks is built on top of the ev3dev platform, which is a Debian-based Linux distribution that runs on the EV3 bricks.