Controlling an AGV using an AR interface

Robert Breugelmans, Industrial Design Engineering, University of Twente, The Netherlands

This project revolved around creating a working Augmented Reality (AR) interface that is able to control an Automatic Guided Vehicle (AGV) and was done for and in cooperation with the Virtual Reality Lab at the University of Twente. The VR Lab has been busy with developing a system where physical objects can be moved by manipulating things in virtual space, and vice versa. This is done by connecting different devices like AGVs, lamps and robot arms with 'The Internet of Things' (IoT) and developing programs and interfaces to control them. These interfaces can be screens and phones, but also VR or AR glasses, essentially mixing the real world with the virtual world.

A problem for the VR Lab was that there was no simple way of controlling the AGVs with other IoT devices. If this were possible, it could be used to for example scale up the Industry 4.0 Testbed prototyping table that is also being developed. Then, if you would move an object on that table, an AGV would also move according to the movement of that object.

IoT devices need to be able to communicate with each other to make the whole system work, for this project, the MQTT internet communication protocol was chosen to do this for the system. This protocol works via a Publish/Subscribe model, where the different IoT devices, 'Clients', can publish/subscribe to different topics on a central broker, to give/get information from that topic. Figure 1 shows a simple MQTT system with 2 clients, where Client A publishes it's coordinates to the topic "Coordinates/ClientA" and subscribes to the topic "Coordinates/ClientB", to get the coordinates Client B publishes.

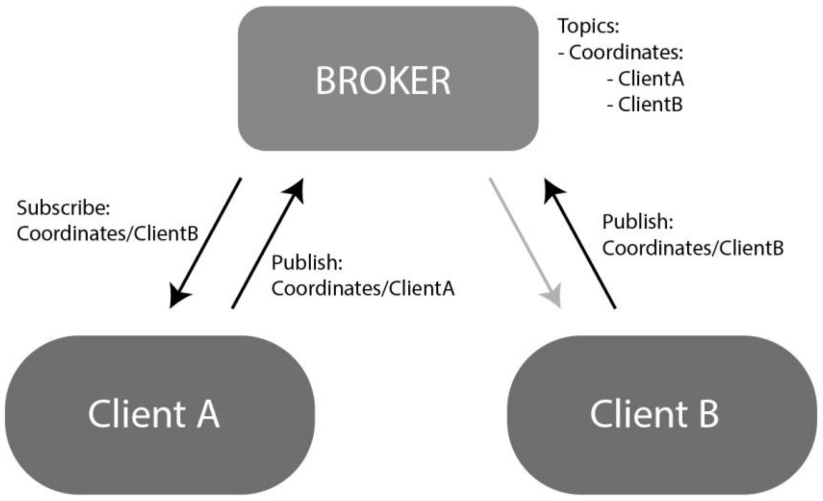
BROKER
Topics:
- Coordinates:
- ClientA
- ClientB
Subscribe: Coordinates/ClientB
Publish: Coordinates/ClientA
Publish: Coordinates/ClientB
Client A
Client B

Figure 1: MQTT broker example

An AGV is a driving robot that can autonomously move from one point to another and is for example used on manufacturing grounds to move supplies around. The type of AGV that was chosen for this assignment is a Turtlebot3 Waffle Pi, see figure 2.

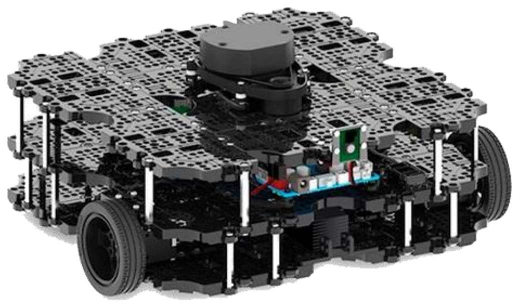

Figure 2: Turtlebot3 Waffle Pi

A Turtlebot3 works with the Robot Operating System (ROS), which controls different programs, 'nodes', that each control a different part/function of the robot.

These nodes communicate with each other by using a Publish/Subscribe protocol, much like the MQTT protocol. Figure 3 shows such a system for a moving robot with a distance detection sensor, Node B handles the calculations of the movement.

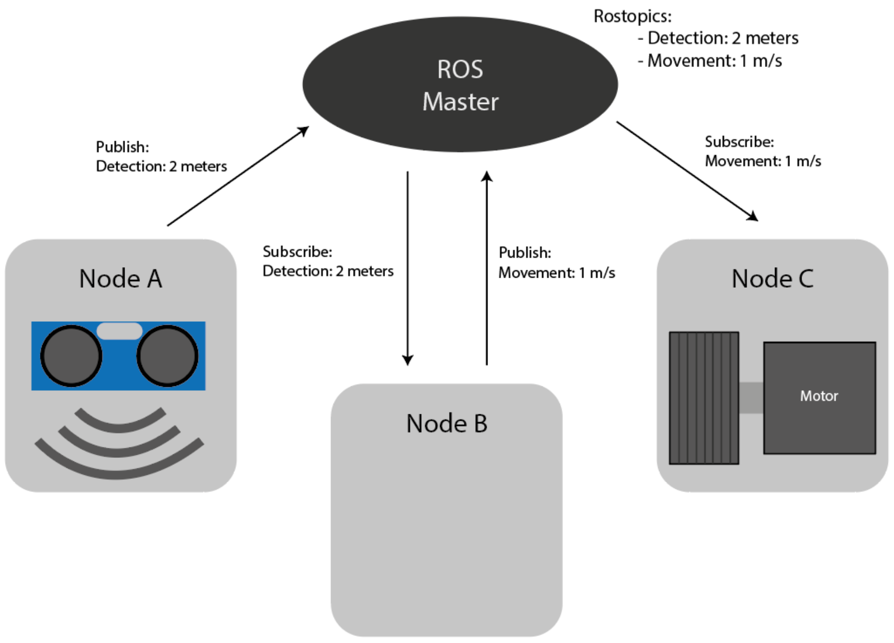

Figure 3: ROS communication example

To make the Turtlebot3 move, you send a specific command including the velocities it needs to move/rotate.

To make the Turtlebot3 move to a specific point, pointed out by the user of the AR application, the Turtlebot3's location was gained through a tracking image at the back and by placing a 'pointer' on the ground where it needs to move towards. The application then calculates the distance and rotation between the Turtlebot3 and that point, after which it sends the right messages to the MQTT broker to rotate and move the Turtlebot3. Figure 4 shows the Turtlebot3 moving from it's initial (red dot) position to the pointer.

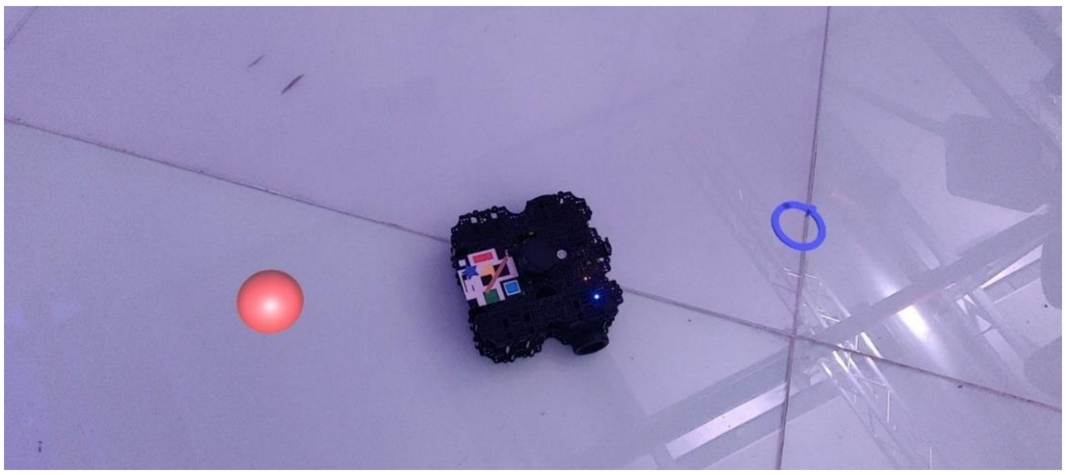

Figure 4: Point to Move interface

To make it clear that MQTT could be used for more things than moving a Turtlebot3, another system was created. This system shows the location of the phone that runs the AR application, by sending it's coordinates to the MQTT broker.

This program worked by calibrating the local coordinate system of the AR application (which is generated each time on start up), to the global coordinate system of the VR Lab. This was also done by using a tracking image placed at a known position in the VR Lab, which made this position clear for the AR application to convert the coordinates with. Figure 5 shows a screenshot of the program that shows the location of the phone with a red sphere.

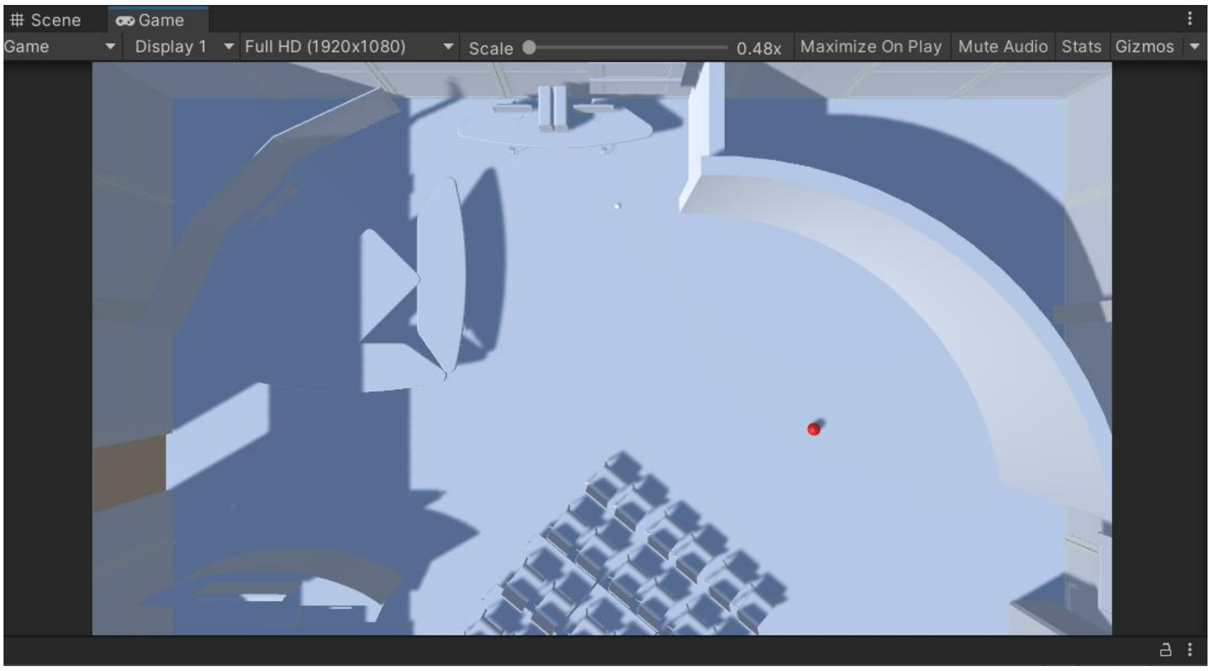

Figure 5: Screenshot localization system

In the end, these results show how to control an AGV using an AR interface and how to create a system for coordinate sharing, however, there are limitations to both. The AR application sometimes shows trouble with tracking the environment, which causes the AGV to move to a different spot. For the coordinate sharing system, this causes that the coordinates are not correct. An external tracking system would be advised to solve this, this would also remove the calibration process, eliminating possible calibration errors.