Establishing a wireless communication between BOCART, cloud, and tablet/mobile[*]

Ekkart Kindler (ekki@dtu.dk) and Hubert Baumeister (huba@dtu.dk)

01.08.2019 — 31.12.2019

BOCART is a small Danish Company producing go-carts. They wish to gamify some teaching material by using their "bocarts". For amplifying a lesson, groups of students could compete against each other not only in driving the BOCARTs, but also in applying the skills or knowledge from the lesson. A team would consist of a bocart driver and one other student. The other student would need to solve some quizzes on some topic; when the other student solves a quiz correctly, this will provide the team's bocart with "more power" (allowing it to drive faster for a period of time).

In order to realize this idea, BOCART needs some teaching and learning software to be able to communicate and control the bocarts. This requires some extra hardware and some software controller on the bocarts (we call them BoCart Controllers for now) as well as some cloud based solution for the communication between the learning software and the bocart controllers. DTU Compute helped with designing such a cloud-based solution. Since neither the hardware on the bocarts nor the BoCart Controllers existed at the time when the project started, and since it also was not clear which learning software would be used, BOCART and DTU Compute had agreed to focus on the design of the cloud software and implement a proof-of-concept for this design. For the proof-of-concept, the bocart controllers are replaced by Bocart Simulators and the learning software is replaced by a simple BoCart (Web) Client. The cloud software and its interfaces, however, are designed in such a way that it could eventually be used for the complete BoCart Solution.

Figure 1 shows an overview of the main components of the BoCart solution, where the BoCart controller is replaced with a simple BoCart Simulator which mimics the interface of the BoCart controller. Likewise, the learning software is replaced by a simple Web Client (BoCart Client), which can be used to send commands to a bocart and to see information on the bocart's current position and speed.

The main component is the BoCart Backend, which provides a REST API, which can be used by other applications for communicating with the bocarts (the BoCart Controllers and the Simulator in our case). In our case, the "other application" is a simple Web Client that allows an end user to see the current position and speed of a bocart and to control it. The control, in our case, will be sending some commands to start and stop the BoCart Simulator and to increase and decrease the bocart's speed. Since the REST API follows a request-response pattern, this interface cannot be used for pushing information from the bocart to the client. For pushing information to the client, the BoCart Backend also provides an endpoint which is a WebSocket (WS); once opened, this WebSocket can be used to send information from the BoCart Backend to the con-

[*]funded by Region Sjælland as part of "ViiRS Vækstaftale 2"

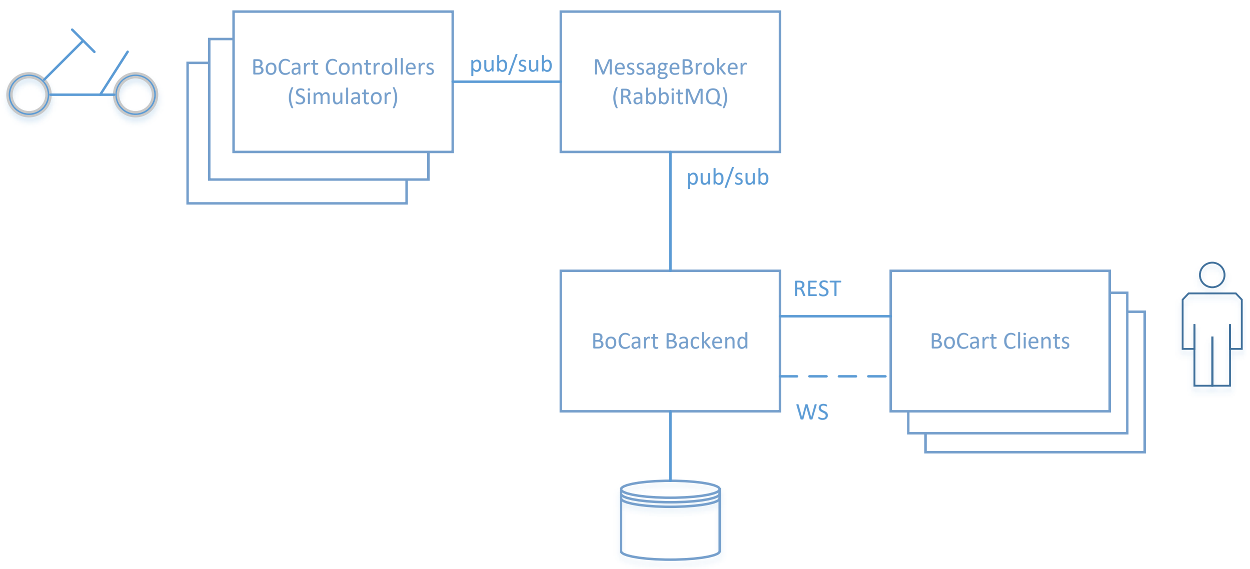

Figure 1: Overview of architecture

nected clients. This interface is graphically represented as a dashed line since it is used in our BoCart Client, but it might not be necessary for the communication with the actual learning software.

The communication between the BoCart Backend and the BoCart Controllers (Simulator) is not via a REST API or via WebSockets, but it is based on lightweight messaging via a message broker using the publish/subscribe mechanism (pub/sub). The BoCart Controllers as well as the BoCart Backend can publish messages via this message broker, and both components can subscribe to receive some messages. The messages are organized in "topics", which allows the different components to subscribe only to the topics that they are concerned in. For the communication with the bocarts, we have chosen the following structure for the topics:

- ⟨gocart name⟩
  - data
  - commands

The top-level topic is the individual name of the respective bocart. This way, it is possible to subscribe to messages concerning a specific Bocart only. On the next level, there are two topics: "data" and "commands". The sub-topic "data" is used by the specific bocart to send its data on the current speed and position to the BoCart Backend. The sub-topic "commands" is used by the BoCart Backend to send commands to a specific bocart; as of now, there are four commands: "start", "stop", "faster" and "slower". But, it would be very easy to replace that by whatever will be needed to control the real bocarts once the real BoCart Controllers are in place. The remaining two components in this overview are the BoCart (Web) Client and the BoCart Simulator. The Web Client allows the user to set up and start races, to create teams of one driver and one player, which can then join a race; then, the player on the Web Client will be able to see the current speed and position of the bocart. This is mostly to illustrate that the communication back and forth between an application and a bocart via the BoCart Backend is working. Our BoCart Client is realized as a simple Elm application, and the BoCart Simulator is a simple Java application, which can be run on any computer; or it can be run on a Raspberry Pi with a GPS module installed. In the latter case, it will send the actual position and speed data from the device. The Simulator software could be used as a template for implementing the real BoCart Controller later—this applies in particular to the component taking care of receiving and sending messages via the message broker.