activation is important, and so is the different intensities. For instance, a higher intensity is easily felt by everyone yet a lower intensity is not.

To go back to the motor control option the user has to press the "Switch to Motor Control" button.

The process of associating the buttons and intensity boxes of the interface and the message to be sent is done using callbacks. Every time a button in the interface is clicked a callback is called giving values to the variables in the message.

If a given motor is selected, then in the location array its value is gonna be 1. If a motor is not selected then its value is 0. The same is done for the intensity array. The value of the intensity selection box is given to the intensity array in the motor position. For instance, if the motor 1 button is clicked and an intensity of 5 is selected (and assuming no other motor is active), then the variables are $numberOfMotors = 1$, $intensity[0] = 5$ and $location[0] = 1$.

When the "Send" button is pressed the *SendMessage()* explained previously is called and the message is then sent to the defined topic.

3.5 Different Approach

Sometimes, due to the huge amount of data being sent between the nodes, Rosserial loses the connection and the system also has some delay.

With the aim of solving this problem, the communication node was eliminated, leaving only the Interface and the Controller node, as shown in Figure 3.26.

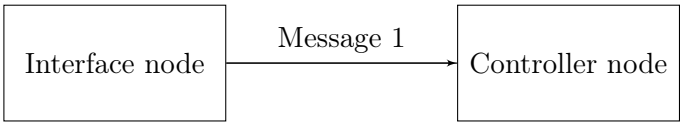

Figure 3.26: New ROS nodes flowchart

In this approach the information from the interface goes straight to the controller, and the mask is created in the controller itself instead of in the communication node.

The message sent from the interface node to the controller node is only the duty cycle and the location of the motors (instead of the motor mask), and this message in sent only when some change on the interface occurs.

Besides making the system faster, this also allows for, in the future, the ROS platform to be dropped.

Figure 3.27 shows the new ROS nodes layout made with the *rqt_graph* software, where *qtguinode* is the interface node, *serial_node* is the controller node and *guichattergui* is the topic where the message is published.

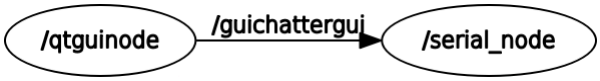

Figure 3.27: New ROS nodes layout made with *rqt_graph*