1 Task Controlling the Robot

Control the robot via an OPC UA client.

Before you can program the fourth task, you should have the UAExpert program installed. With this OPC UA client you can search the namespace of the puzzle bot. You also have the possibility to control the puzzlebot and follow it via the camera. To program an OPC UA client you need the namespace and the node identification number (Node-ID) of the respective object, variable and method. You can read these using the UAExpert program.

To program the client, start by creating auxiliary variables.

- Variable for the username
- Variable for the password
- variable for the hostname
- Variable for the port

To initialize the client use the function `client("path")` The function returns a client. The path in the function must be specified in the format `"opc.tcp://username:password@hostname:port"` The hostname of the puzzlebot is `engine.ie.technikum-wien.at` and the port has the number 4840.

With the function `client.connect()` you establish a connection to the OPC UA server of the puzzlebot. Now you have created a session with the name `client` which is available for the next steps. With the function `client.get_namespace_index("path")` you can define the namespace and save it into a variable. This function returns the namespace. For the path to work in the namespace of the puzzlebot please enter `opc.tcp://engine.ie.technikum-wien.at/PuzzleBot` For the connection via UA Expert you will be provided with a file which you have to open. When you have established the connection, you will see the namespace of the server on the left side under the tab Address Space. In the object Puzzlebot you will find all important variables and methods to control the Puzzlebot. Click on a variable or method, then you will see the item Node-ID on the right side under the Attributes tab. With this node-identification number you can read and write the respective variable in the program or call the method. Figure 1 shows parts of the UAExpert program when connected to a test server.

The node identification number is in theory always the same for the same variable or method. However, if a change is made on the OPC UA Server, a variable or method is added or removed,

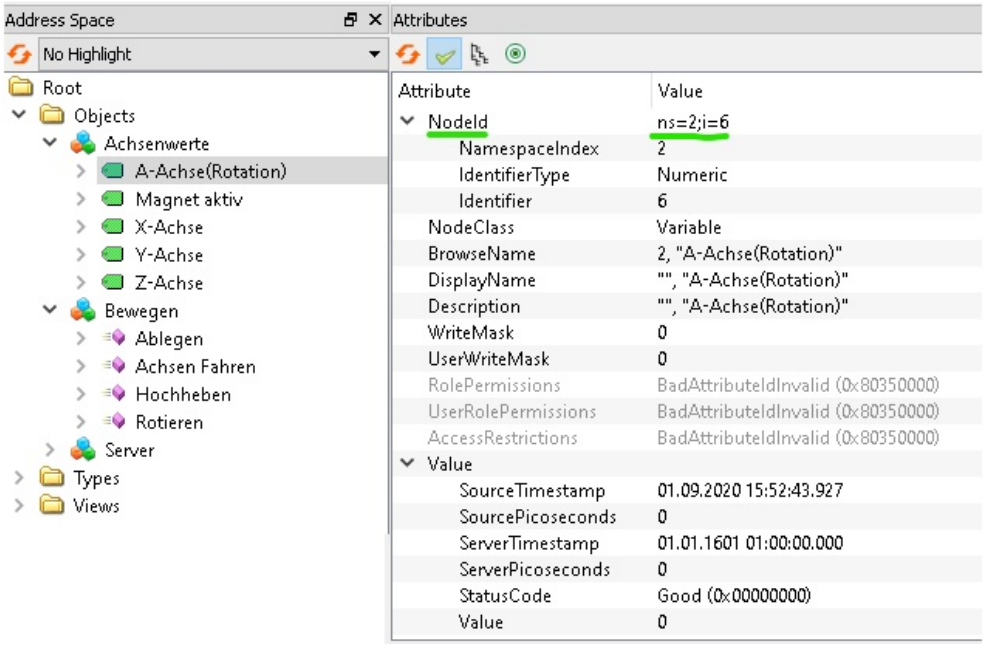

Figure 1: Part of UAExpert version 1.5.1.331 connected to a test server

it is no longer guaranteed that the node identification number still matches. For this task read out all node IDs you need.

The function `Client.get_node("Node-ID")` returns a node.

Hinweis: Keep in mind that a node can also be an object containing several variables and methods. You can address an object as well as variables and methods directly via the node ID.

With the function `Node.get_value()` you can read the variable value. If a node has subordinate nodes, then this node must be designated as parent. With the function `Parent.get_children()` and a loop you can read the child variables and methods.

Next call with `call_method(node,value)` the function of the puzzle bot, which is responsible for its axis movement. Pass the values from task 3 to the function and the puzzlebot will move to the position where the center of gravity of the selected object is located. By calling the `PICK` method, objects can be lifted with the magnet. If you want to place the objects at a different position, you can do this with the `PLACE` method of the Puzzlebot.

Note: Please keep in mind that the puzzlebot needs some time to execute your instructions. It is therefore advisable to implement a delay with the function `time.sleep(seconds)`

Finally, be sure to terminate the connection to the Puzzlebot with `CLIENT.disconnect()` The OPC UA server of the puzzlebot is configured in such a way that a maximum of one session per user can be created.

This completes task 4. Use the libraries and function listed in chapter 1.1 to program task 4. Please note that functions from tasks 1, 2 and 3 can be used in the same way.

1.1 Functions and Libraries Exercise 4

Libraries:

- `from opcua import ua, uamethod, Client`
- `import time`
- `(import cv2)`

Functions:

- `Client()` Defining a connection/client
- `connect()` Connect to the server
- `get_namespace_index()` Gets the index for a namespace
- `get_node()` Addressing a specific node
- `get_value()` Read a value
- `set_value()` Writing a value
- `call_method()` Calling a method
- `disconnect()` Terminate connection with the server