THE METHOD OF NETWORK CONTROL OF ROBOTIC SYSTEMS OF DIFFERENT MODELS AND MANUFACTURERS

Victor Andreev\textsuperscript{a,b}, Pavel Pletenev\textsuperscript{a}, Yury Poduraev\textsuperscript{a}

\textsuperscript{a}MSTU “STANKIN”, Vodkaovsky lane 1, Moscow, 127994, Russia
\textsuperscript{b}International Laboratory “Sensorika”, Miusskaya sq. 4, Moscow, 125047, Russia

Abstract

The method of organization of network management for robotic systems from different manufacturers without modification of their low-level code is described. The method is based on the “concept of drivers”. It is similar to the external device drivers for various operating systems. The requirements for the hardware and software architecture of the interaction of robotic systems of different models and manufacturers were defined. The development of a “driver” for mechatronic gripper module of AMUR (ML “Sensorika”) educational robot is shown as a specific example of driver's development stages.

Keyword: mobile robots; remote control; control systems; network control; robotic systems; gripper; Internet-laboratory

This Publication has to be referred as: Andreev, Victor; Pletenev, Pavel & Poduraev, Yury (2016). The Method of Network Control of Robotic Systems of Different Models and Manufacturers, Proceedings of the 26th DAAAM International Symposium, pp.0727-0730, B. Katalinic (Ed.), Published by DAAAM International, ISBN 978-3-902734-07-5, ISSN 1726-9679, Vienna, Austria DOI:10.2507/26th.daaam.proceedings.101
1. Introduction

This work is a part of a big scientific and applied work on creation of geographically distributed educational Internet laboratory. As part of this laboratory, a full-featured Internet access to specific robotic systems of different models and manufacturers is organized [1].

The idea of remote control of robotic systems is not new and it had been realized in previous century. As an example one can present the result of collaboration of Microsoft and Massachusetts Institute of Technology (MIT) – "iLab Project", which was created in 1998 [2]. This project shared resources (training stands) of different laboratories using networking technologies. Software of this project runs on a dedicated server connected to training stand under Windows operation system. Thus, this solution involves additional technical resources for computer hardware and software to adapt to a particular stand. For mobile robots this problem was also researched in other works [3, 4]. Unlike "iLab Project" this work propose use of the “driver concept”.

2. Driver concept

«Driver conception» was introduced first in the work [5]. Minimum controllable object is any electronic or mechatronic device, which can be a part of a robotic system. This conception was realized as a finite set of programs and network protocols – an application programming interface (API) [6]. This software will be called “Internet laboratory” API further in text. This interface provides transparent integration of elements of a mechatronic device and its software in the information space of information measurement and control system (IMCS) of the robotic complex. In other words, “driver concept” involves the creation for each of mechatronic devices unified control network protocol based on existing low-level APIs interaction. Manufacturers of robotic systems can relatively easy add new devices to the network using this concept through the creation of an additional driver that implements software instructions and networking protocols, determined by the protocol.

3. Driver creation technology

To organize network access to mechatronic (or electronic) device using “driver concept” such device must be an edge device. In this work, an edge device is a set of hardware and software that make up the control system of mechatronic device as a part of robotic systems that allow it to perform a full range of functions of “Internet laboratory” API including the ability to dynamically re-programming the control system. An edge device must meet the following requirements:

- the device must have the LAN interface;
- the device must have a microcontroller and the ability to install the software implementing the minimum necessary subset used in the system of the protocol.

“Internet laboratory” API is written in the Python programming language which provides portability to different hardware and software platforms. It is based on message exchange library ZeroMQ, which is also cross-platform. Ergo an edge device must support installation and proper operation of Python interpreter and ZeroMQ library. Examples of edge devices are microcomputer Raspberry Pi B+, nanocomputer Black Swift and many others, based on ARM, MIPS and x86 processor technologies.

“Internet laboratory” is written using object-oriented programming and contains classes responsible for both drivers and remote control applications. Each driver is described as a set of RoboWorker objects, which control and describe different mechatronic devices or their submodules as components of a robotic system. RoboWorker objects register component in “Internet laboratory” network, which is done using _start() method which takes name of a component as argument. Method add() and get() are responsible for data exchange and send and receive data respectively. Data can be arbitrary objects, which can be serialized and deserialized using JSON.

The aim of this work is to develop procedures for creating a "driver" for mechatronic device that provides automatic integration with software and hardware infrastructure of the "Internet-laboratory" without any corrections of its low-level code.

To achieve this, the authors have created a manual for mechatronic module developer and they put themselves in the place of such developer, offering an example of realization of this manual creating a driver for a mechatronic gripper module.

The following manual on how to create a “driver” using “Internet laboratory” API is based on its requirements and characteristics:
1) Connect and configure the edge device to work in a local area network (LAN).
2) Install into edge device Python interpreter of version 2.7 and pyzmq library.
3) Create the device control module.
4) Create an interface to device control module in Python.
5) Write the driver code that is responsible for sending data back and forth between the interface that created on the previous step and the “Internet laboratory” API.
An edge device must be connected to a LAN through a variety of techniques, such as the EtherNet, Wi-Fi, as well as through advanced mobile networks. If the operator of edge device is physically located in a place with no access to the LAN of edge device, the operator may combine this LANs through Internet using VPN technology.

In 2008, the Python development split into two incompatible languages: subversions of 2.7 and subversions of 3.0 [7]. However, at the moment the most common version of the language and the Python interpreter is version 2.7: many GNU/Linux distributions select as a default Python interpreter version 2.7. Version 3.0 has not yet been ported to some operating systems. Therefore, the primary development language is Python version 2.7.

The device control module must meet the challenges of supervisory control and create an array of data on the current state of the device. This module must include a maximum number of “intelligent” control functions of robotic system. The main languages for development of such software are C and C++. Their advantages are undeniable – they are compiled both high- and low-level languages with a high barrier to entry. To accelerate the development of this module can be created in Python. However, its main advantage and at the same time its main disadvantage is dynamic typing, that, on the one hand, accelerates the development process and reduces the barrier to entry, but on the other hand, slows down the implementation of programs and makes it extremely difficult to optimize their performance. In addition to the rich standard library of Python one can also install third party libraries to work with various communication protocols. However, these disadvantages can be partially circumvented by using the library for scientific calculations for Python - numpy and scipy [8], which respectively accelerate arithmetic operations and add support for symbolic computation and many special mathematical functions.

If there is an existing infrastructure based on control modules written in C or C++ or requirements on speed the internal control systems are such that Python because of its low speed is not suitable as the primary programming language for the control module, it is necessary to create an interface for these modules using libraries such as Cython [9], ctypes [10], boost::python [11] or by using the Python CAPI [12].

Driver is created in Python and should be controlled by a number of RoboWorker objects of “Internet laboratory” API. Driver developer must decide which data from the array, given by the device control module, will be published on the network, and what data the device will receive as control signals. To initiate the transmission and reception of data one must call add() and get() methods respectively on RoboWorker objects.

Fig. 1 is a schematic representation of the steps towards the development of the driver. The base of development is a mechatronic device. Each step is based on the results of the previous one, and the final stage is the driver.

![Diagram](image)

Fig. 1. Schematic representation of driver developer manual.

4. Driver development example

Drivers for robots Robotino (Festo, Germany) [5] and AMUR (ML “Sensorika”, Russia) of different modifications were created using the manual described above. Authors developed mechatronic gripper module for autonomous mobile university robot AMUR-10. Its control system has I2C input. The driver for this module was created. Microcomputer Raspberry Pi B+ was selected as an edge device, because control system of the gripper module is not an edge device. Raspberry Pi B+ has both Ethernet and I2C interfaces thus it can communicate both with LAN and gripper module. Microcomputer runs under Raspbian operating system, which was additionally configured to work with I2C devices. Python interpreter (default 2.7 version), pyzmq library and “Internet laboratory” API were installed in this OS. Next a Python control module was created for gripper using the python-smbus library. Finally all parts of the system were united in a driver, which send data back and forth between Python gripper control module and “Internet laboratory” API. As a result, a full-fledge edge device was created.
5. Conclusions

Proposed “driver concept” solves a problem of distributed control of a robotic systems. It introduces the unified “Internet laboratory” API to create both drivers and client application for supervisory control. One of the best parts of proposed concept is capability to integrate mechatronic modules from different manufacturers into control system over the Internet without modifying low-level code of control system. This concept can be used not only in robotic systems, but also in manufacturing, for example in bionic systems, described in [13, 14]. The only restriction of this concept is necessity for mechatronic device to be an edge device. For robotic systems, which consist of a variety of mechatronic modules, one must create separate drivers. A real laboratory, consisting of several robots, can be represented as a tree, where the root is the laboratory; branches are the addresses of the individual of robotic systems and leaves are drivers of mechatronic devices. There are some parts of the concept that can be enhanced. For example: for greater clarity and a clear presentation of data, developers must be able to create groups of drivers. Also at the moment, the “Internet laboratory” API does not provide the developer with methods of access restriction. The implementation of this function is entrusted to individual mechatronic device developers. Therefore it is necessary to develop such mechanism.

6. References


