“BAXTER” INDUSTRIAL ROBOT ONLINE CONTROL

Y. Yordanov, O. Nakov, V. Mladenov

Technical University of Sofia (BULGARIA)

Abstract

The "Baxter" robot [1] is a robot that has many different applications including the education, research, industrial tasks, etc. Its height is more than 1.8m and he weighs 138kg - along with its stand. The robot has two hands, each with a gripper and a built-in camera. With them it can perform pick & place operations, recognition of different objects [2] (he can recognize their shape, color and so on), which then sorts. It has sensors that allow it to detect when it’s made contact with a person, so it can stop and also they give the robot the ability to adapt to its environment. Based on the data from the sensors, the robot can sense potential collision events early and can reduce the force before the impact. This is due to a motor driving a spring which drives Baxter's arms instead of just a driving its arms motor. Another feature that distinguishes Baxter from other robots is that instead of writing complicated code, users can grab the robot’s arms and physically guide them through the required motions – this way the robot can even fold clothes. Its face is a flat screen that can express “feelings”. This also makes it able to participate in studies about human-robot interaction (HRI).

Keywords: “Baxter” robot, Control, Webpage, Python, Rubi, Linux, Windows, Android

1 INTRODUCTION

The robot has a built-in operating system - the Robotic Operation System (ROS) [3]. One of the primary tasks of the ROS is to manage the communication with the robot via an Ethernet connection. It provides access to the robot’s motors in order to create various specific movements, reading robot’s sensors data (in particular the sonar) - to avoid unintentional contact with a person, processing camera’s data – for performing facial and object recognition and others. The connection with the ROS is established through a dedicated Linux [4] system responsible for communicating with the robot. The Linux system serves as a platform to write and test the code that is sent for execution on the robot. The primary programming language used to write the script is Python. However, even with the help of online libraries and resources many hardware and software problems and limitations still exist and need to be thought out and resolved using custom methods, script or programs. For example, some of the main problems are the need for Linux, as well as a permanent Ethernet connection, which when combined with its large robot size, limit the potential of the functionality and features that can be developed and implemented for future devices and platforms. Since the Linux operating system is the recommended way to control the robot, it needs to be used as the primary tool to develop custom interfaces that can create a bridge for controlling the robot using other operating systems and devices, such as Windows, Android [5], iOS, which are not initially supported for communicating and interacting with the robot. Without this interface bridge, interacting with the robot away from the Linux system – for example from places like expos, university laboratories for demonstration and/or training purposes using modern devices has been seemingly impossible.

The objective of the paper is to present the developed python software and ruby website that solves these problems. With them now is possible to control remotely the robot through all the operation systems and from any devices connected to the internet [6], albeit with a limited number of commands. This paper describes not only the hardware and software that was used to create the system, but also the methodologies for the tests and adjustments made to it.

This paper is structured so that the developed website is described in details following the introduction. The next section presents the robot and devices used for the system's additional tests. The experiments themselves are described in a special section devoted to the testing. In conclusion it is worth to point out that the test results show that they are successful - the Baxter robot is successfully controlled by Android and Windows devices as well as all those connected to the Internet. The last section also presents the future development and possible improvements of the system.

2 SOFTWARE

2.1. Website creation

For the creation of the site is used programming language “Ruby” [7] version 2.6.1. The site can be found here and can be accessed from all operation systems:

http://baxtercontroller.krastev.org/

These are the gems (or libraries), used in the project. The first one is Web Framework “rails” version 5.2.2 - the entire site is based on it - for example links between the used libraries in the project, processing the pages and their connection to the server, recording of the information in the database, etc. The 'bundler', version 1.17.2 is used for the proper operation of the other libraries. The "puma", version 3.11, is used for runing a local web server and test the site locally. When the site is uploaded, the servers of the hosting side are used. To record the actions of the user, the "device" library is used - to track log-in or log-out actions of the users, to recover forgotten passwords and other user-related activities. The 'rolify' library is used to create and manage roles in the site - admin, user, hierarchy, and so on. The admin page uses "rails_admin". At start up, the information that is generated each time is saved by hashing from the “bootsnap” library, version 1.1.0. This gem allows faster startup of the server. The"rack-cors" library is used for handling Cross-Origin Resource Sharing (CORS), making cross-origin AJAX possible

The frontend gems are, 'bootstrap-sass’, it allows the developer to use icons, drop-down menus, navigation fields, different styles, and more in the project. The gem 'sass-rails', version 5.0 is an extension of CSS that adds power and elegance to the basic language. It allows the developer to use variables, nested rules, mixins, inline imports, and more, all with a fully CSS-compatible syntax. Sass helps keep large stylesheets well-organized and get small stylesheets up and running quickly, especially with the help of the Compass style library. The gem 'uglifier', version 1.3.0 is a JavaScript parser, minifier, compressor and beautifier toolkit. It helps to compress the code so it is smaller - to load the file faster, as well as complicate reading of the code by outsiders. The gem 'coffee-rails', version 4.2 is adapter for the Rails asset pipeline. Also adds support to use CoffeeScript to respond to JavaScript requests.

The backend gems are, gem 'figaro', used for creating variables in the working enviroment. It is simple, Heroku-friendly Rails app configuration using ENV and a single YAML file. The gem 'sqlite3', version 1.3.6 allows the usege of database SQL lite version 3. The gem 'web-console’ is a debugging tool for the Ruby on Rails project. The gem 'listen', version 3.0.5 listens to file modifications and notifies the developer about the changes. The gem 'spring' is a Rails application preloader. It speeds up development by keeping the application running in the background so the developer don't need to boot it every time he runs a test, rake task or migration. The gem 'spring-watcher-listen’, version 2.0.0 makes Spring watch the filesystem for changes using Listen rather than by polling the filesystem. On larger projects this means spring will be more responsive, more accurate and use less cpu on local filesystems. The gem 'better_errors' replaces the standard Rails error page with a much better and more useful error page. It is also usable outside of Rails in any Rack app as Rack middleware. The gem 'rails_layout' is used for setting up layout files for developer’s choice of front-end framework, supports, Zurb Foundation 5.3 Bootstrap 4.0 Bootstrap 3.3. In this project bootstrap 4.0 is used. The gem 'byebug' is a simple to use and feature rich debugger for Ruby. It uses the TracePoint API for execution control and the Debug Inspector API for call stack navigation. Therefore, Byebug doesn't depend on internal core sources. Byebug is also fast because it is developed as a C extension and reliable because it is supported by a full test suite. The debugger permits the ability to understand what is going on inside a Ruby program while it executes and offers many of the traditional debugging features such as: Stepping: Running a program one line at a time. Breaking: Pausing the program at some event or specified instruction, to examine the current state. Evaluating: Basic REPL functionality. Tracking: Keeping track of the different values of developer’s variables or the different lines executed by the program The gem 'pg' allows the usege of PostgreSQL [8] – a powerful, open source object-relational database system with over 30 years of active development with strong reputation for reliability, feature robustness, and performance.

2.2. Server creation

The server is deployed on the Linux workstation. It reads the information from the site and then controls the robot’s movements. The programming language, used for writing the server, is Python

3.5. The main library, used for the server, is “requests”. It is an Apache2 Licensed HTTP library, written in Python. It is designed to be used by humans to interact with the language. This means that the developer don’t have to manually add query strings to URLs, or form-encode the POST data. Requests allow the developer to send HTTP/1.1 requests. With it, it is posible to add content like headers, form data, multipart files, and parameters. It also allows the developer to access the response data of Python in the same way.

3 HARDWARE

3.1. Introduction

Three types of hardware are used in the system - “Baxter robot” and two specialized internet devices.

3.1.1 Baxter robot

3.1.1.1. Physical specifications

The robot height is 3’ 1” without pedestal and 5’ 10” – 6 ’3” with adjustable pedestal. It’s maximum reach is 1210 mm and the torso mounting plate diameter is 13.3” (it is used for mounting on table). The body weight of the robot is 165 lbs. without pedestal and 306 lbs. with the pedestal. “Baxter” has 14 Degrees of Freedom - 7 per arm. The Pedestal Footprint is 36” × 32”. The robot has max payload 5 lb / 2.2 kg and it’s gripping torque is 10 lb /4.4 kg.

3.1.1.2. Computer and sensor specifications

The CPU of the robot is 3rd Gen Intel Core i7-3770 Processor (8MB, 3.4GHz) w/HD4000 Graphics. It has 4GB, NON-ECC, 1600MHZ DDR Memory and 128GB Solid State Drive storage. “Baxter” has a build in front camera with max resolution - 1280 x 800 pixels with effective resolution 640 x 400 pixels. The Camera’s frame rate is 30 frames per second and the focal length is 1.2 mm. The front screen has resolution 1024 x 600 pixels. The robot has also build-in infrared sensor, which range varies between 4 and 40 cm.

3.1.1.3. Electrical specifications

The supply voltage of the robot is 120 volts alternating current. It’s rated current is 6 amps. The interface of the robot to the power grid is standard 120VAC power. Robot power bus and internal PC both have “universal” power supplies and support 90 – 264V AC (47 – 63Hz). The maximum consumption of “Baxter” is 6A at 120V AC - 720W max per unit. It’s maximum efficiency is 87% to 92%. The power supply is medical-grade DC switching. Sags are tolerated to 90V. Sustained interruption will require manual power-up. The Voltage Flicker has a holdup time of 20mS and the voltage unbalance is reduced to single phase operation only.

3.1.2. Specialised internet devices

Both devices have Internet connectivity capabilities, but they work with different operating systems, allowing more diverse tests.

3.1.2.1. Specialised Android platform for software development H3/5 main specifications

The device has build-in operation system Android 4.2. It supports HDMI and USB OTG. It has 2 GB DDR2 of random access memory as well as a USB 2.0 ports. The device is also equipped with WiFi and Bluetooth module. Audio and AV output is also available, as well as SD card slot.

3.1.2.2. Specialised Windows device with wireless communication for online control 802.11. main specifications:

The second device is equipped with the MS Windows 10 operation system, as well as a Bluetooth 4.1 module, WiFi, a 100Mb/s Ethernet port, 8GB DDR2 of RAM (Random Access Memory) and a 400GB HDD (Mechanical Hard Drive).

4 TESTS AND RESULTS

Numerous tests were performed to check the loading time of site with different Internet browsers. It is important to specify that the tests are performed with a Wireless connection at 25Mbps.

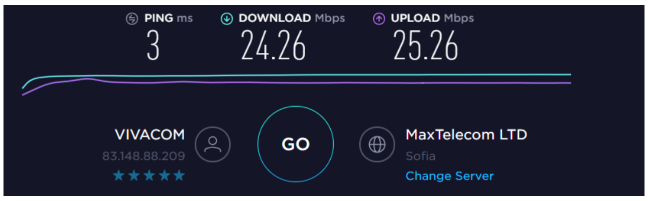

Figure 1: Speedtest results

4.1. Tests of the site's performance

The table below shows the time the site needs to load completely with the Mozilla Firefox web browser [9], Google Chrome browser [10] and Safari browser [11] (Reaction time). Five attempts (Number of the test) with OS Windows and five with Android OS were made and measured.

Table 1: Site’s load time performance with different internet browsers

Firefox browser			Chrome browser			Safari browser		
Number of the test	Reaction time (s) Android	Reaction time (s) Windows	Number of the test	Reaction time (s) Android	Reaction time (s) Windows	Number of the test	Reaction time (s) Android	Reaction time (s) Windows
1	1.35	1.22	1	1.13	1.10	1	1.40	1.35
2	1.19	1.15	2	1.15	1.03	2	1.80	1.48
3	1.32	1.25	3	1.20	1.14	3	2.5	1.50
4	1.41	1.32	4	1.32	1.17	4	2.32	1.95
5	1.21	1.22	5	1.21	1.15	5	1.52	1.44

4.1 Testing the time for completion of the commands on the robot - executed from the website.

The robot is capable to perform 6 different commands. He can move his head left and right as well as going back to the center. His arms can go up, wide and go back to neutral position.

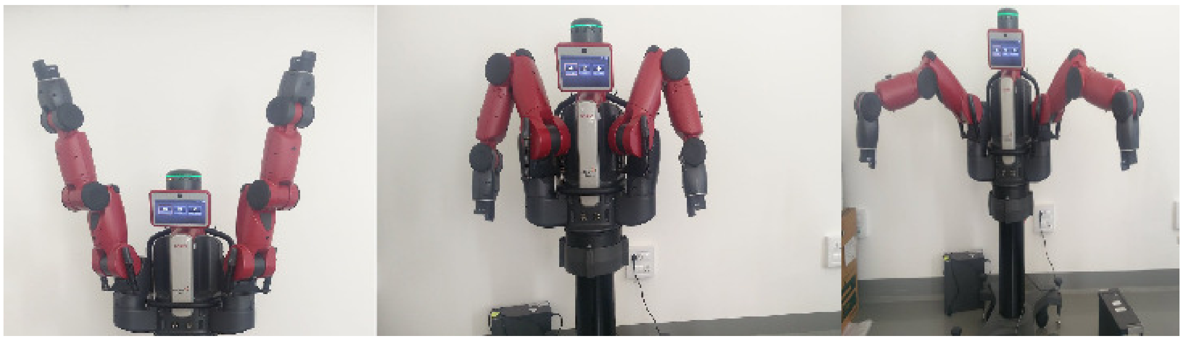

Figure 2: Arm movements – up, neutral, wide

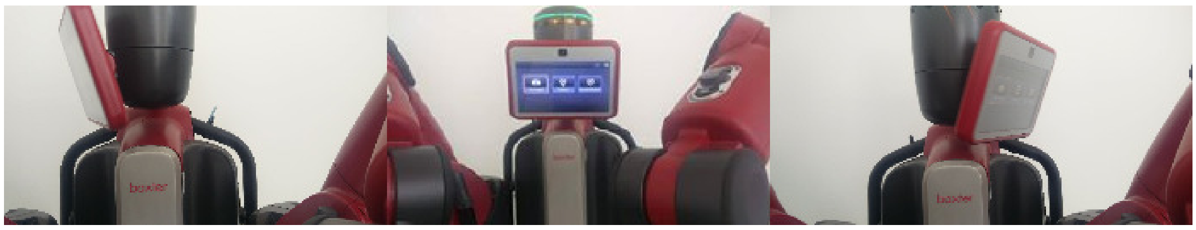

Figure 3: Head movements – left, neutral, right

The fastest web browser of the three - Google Chrome is selected for the tests. The table 4 below describes:

- Command - the command sent from the website to the robot for execution
- Reaction time - the time between the initialization of the command and its completion - measured in seconds
- Status - a column for whether the command has been executed

Table 2: Reaction time of the robot

Command	Reaction time (s) Android	Reaction time (s) Windows	Status	Command	Reaction time (s) Android	Reaction time (s) Windows	Status
Head position neutral	1.75	1.62	OK	Arm position neutral	1.48	1.48	OK
Head position left	1.83	1.67	OK	Arm position straight up	1.71	1.50	OK
Head position right	1.92	1.77	OK	Arm position wide	1.68	1.43	OK

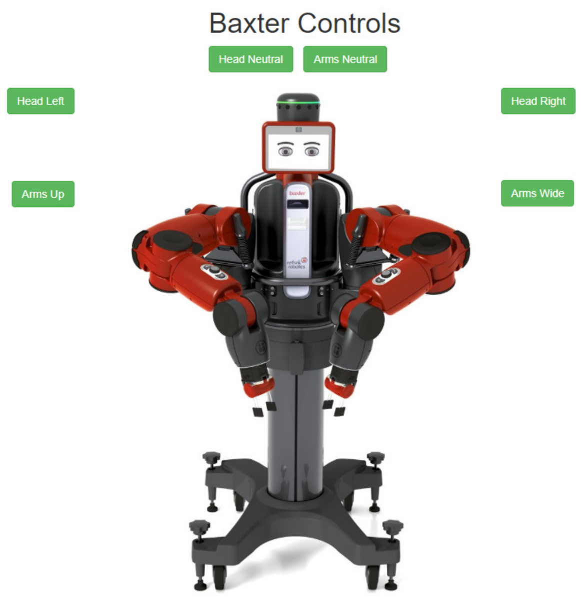

Figure 4: View of the developed website and control buttons

5 CONCLUSIONS

The main problems of controlling the Baxter robot remotely have been resolved – the website and python script have been developed and fully tested. The test results show that the Baxter robot can be successfully controlled not only by Linux system, but also by Android and Windows devices as well as all those connected to the Internet, with minimal latency between the systems. The future of this project envisions additions of a real-time camera stream, and additional commands to the developed website.

ACKNOWLEDGEMENTS

This work is supported with project “Development of mobile android application for control of industrial robot Baxter” № 182ПД0015-08 / session 2018 of the Research and Development Sector of the Technical University of Sofia.

REFERENCES

[1] A. Smith, C. Yang, C. Li, H. Ma and L. Zhao, "Development of a dynamics model for the Baxter robot," 2016 IEEE International Conference on Mechatronics and Automation, Harbin, 2016, pp. 1244-1249.

[2] Xiuwen Liu, A. Srivastava and K. Gallivan, "Optimal linear representations of images for object recognition," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 26, no. 5, pp. 662-666, May 2004.

[3] D. Whitney, E. Rosen, D. Ullman, E. Phillips and S. Tellex, "ROS Reality: A Virtual Reality Framework Using Consumer-Grade Hardware for ROS-Enabled Robots," 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, 2018, pp. 1-9.

[4] M. Kong, J. Li and Wang Fengming, "Study on educational mode of Linux majors in colleges," 2010 International Conference on Artificial Intelligence and Education (ICAIE), Hangzhou, 2010, pp. 623-626.

[5] I. Khokhlov and L. Reznik, "Android system security evaluation," 2018 15th IEEE Annual Consumer Communications & Networking Conference (CCNC), Las Vegas, NV, 2018, pp. 1-2.

[6] G. Wood, "IPv6: Making Room for the World on the Future Internet," in IEEE Internet Computing, vol. 15, no. 4, pp. 88-89, July-Aug. 2011.

[7] C. Chunling, "Construction of the Individualized College English Learning Management System Using Ruby on Rails," 2015 International Conference on Service Science (ICSS), Weihai, 2015, pp. 160-163.

[8] M. Jung, S. Youn, J. Bae and Y. Choi, "A Study on Data Input and Output Performance Comparison of MongoDB and PostgreSQL in the Big Data Environment," 2015 8th International Conference on Database Theory and Application (DTA), Jeju, 2015, pp. 14-17.

[9] Y. Cai and Y. Yuan, "A comparative Study of the Safety between Internet Explorer and Firefox," 2012 Fourth International Symposium on Information Science and Engineering, Shanghai, 2012, pp. 165-168.

[10] S. Lal and A. Sureka, "Comparison of Seven Bug Report Types: A Case-Study of Google Chrome Browser Project," 2012 19th Asia-Pacific Software Engineering Conference, Hong Kong, 2012, pp. 517-526.

[11] G. Alor-Hernandez et al., "RIA Optimization to Support Web Service Invocations on Mobile Safari for iPhone," 2009 Second International Conference on Advances in Human-Oriented and Personalized Mechanisms, Technologies, and Services, Porto, 2009, pp. 39-44.