# System for monitoring and control of the Baxter robot

Yasen Yordanov
*Faculty of Automatics*
*Technical University of*
*Sofia,* Bulgaria
valerim@tu-sofia.bg

Ognyan Nakov
*Faculty of Computer Systems and*
*Technologies*
*Technical University of*
*Sofia*, Bulgaria
nakov@tu-sofia.bg

Valeri Mladenov
*Faculty of Automatics*
*Technical University of*
*Sofia*, Bulgaria
valerim@tu-sofia.bg

*Abstract*—**This scientific report details how a software and hardware system can be created to allow the surveillance, access and control of a Baxter industrial robot from any location connected to the Internet. Baxter is being aimed at the nearly 270,000 small to midsize manufacturers, which have five hundred or fewer employees - companies of that size are unlikely to be able to invest hundreds of thousands of dollars into robots that require a redesign of their workspace and IT personnel to run them. Commonly used, the robot can handle many different industry tasks – co-packing and end-of-line packaging, pick and place operations, plastic injection, printed circuit board handling and many others. It's important to noted, that the platform on which the control software works is Microsoft Windows. The programming language used to write the code for the system is Python 2.7. For monitoring a robot when executing a command, a high quality camera - AUSDOM AW615 1080p - is also used in the system.**

***Keywords— Baxter, Python, Software, WebCam***

## I. INTRODUCTION

The Baxter robot [1] has many and different applications. Thanks to its embedded cameras, a team from New Zealand has managed to develop a project in which the robot plays autonomously chess against human [2]. This is achieved with the camera, positioned in one of the robot's arms, that perceive the game state, an open-source chess engine [3] that computes the next move, and a mechatronics subsystem with a 7-DOF arm [4] that handles the pieces. Also there are many different models of the robot, that are developed to ease the it's use. The dynamic model for example is important, because he helps to find the relationship between the joint actuator torques and the resulting motion in the robot. The models is developed by a team from the United Kingdom and uses Lagrange Euler (L-E) equations [5]. In addition, the team has also developed and presented the kinematic model of the robot. In general, the Baxter robot is required to have a separate computer on which the Robotic Operation System (ROS) [6] is installed. It accesses the robot and its main functions. The problem is that it is Linux [7] based, so Windows[8] computers can not control the robot. Another problem is that the computer and the robot must be connected in a common network so that they can send and receive commands to each other. Of course, difficulties in working with the robot appear because of its size - 182 cm height and weight 138 kg. with the tray. With these dimensions, dragging the robot for demonstrations and training becomes a difficult task. The software developed & described in the report solves these problems. It is written for MS Windows operating systems. A socket is used to connect to the computer running the robot. The only other necessary hardware is a camera through which the operator monitors the proper execution of the commands provided by the software. Thanks to the code developed, the robot can perform different movements with its arms - stretched up, stretched to the side and then back to neutral position, and can also move its head - left, right and return to neutral position. Below, the report describes both the software packages involved in writing the system code and the hardware used. There is also a special insight into testing the system - in the chapter Tests and Validation.

## II. SOFTWARE

### 2.1. Intoduction

The Baxter robot is only compatible with Python [9] version 2.7 - due to restrictions associated with the use of "ROS". ROS, in addition to managing the robot, is only compatible with Linux based operating systems. In order to manage the robot through Windows, we use the python WebSocket library. In order to communicate between the two operating systems, the command is first sent through the websocket from the user interface developed by the Tkinter library, and then processed by the Ubuntu machine, which, depending on the information, sends the required execution command to the robot. The Windows machine in the system plays a role as a client, and the Linux one on a server. Before the software is used, it is necessary to create a working environment. This is done by using the command catkin make. The necessary packages can then be used. This command is also used to configure the path to the code so that it can be imported. In case other ROS packages are used, the paths to the libraries that will be used must be dynamically attached. Different classes are used to wrap the envelopes.

### 2.3. Libraries used in the system

#### 2.3.1. Python "tkinter"

This is the library that was used to create the user's graphical interface, which is deployed on the Windows personal computer. Some of the key functions used to write the code are:

• button_connect ()

• button_config ()

• button.grid ()

#### 2.3.2. Python "sys"

The library is used to operate the system with the interpreter. For example, to get out of a function with exit (), we need this library.

#### 2.3.3. Python "time"

Provides various time-related features. For the proper operation of the library, it's important to get to know the operating system (OS) well, because each OS has a different implementation of the time-processing functions. The function, used in the system, to create delay is time.sleep().

#### 2.3.4. ROS "Baxter_interface"

This is a library that initializes the particular part of the Baxter robot that will be used. It is possible to initiate more than one part of the robot, which allows for more complex operations.

#### 2.3.5. ROS "Rospy"

This is the library that makes it possible to interact with the properties of ROS faster and easier.

#### 2.3.6. WebSocket Library

WebSocket is a library that we use to create a communication protocol for a persistent, bi-directional, full duplex TCP connection from a client to a server. A WebSocket connection is initiated by sending a WebSocket handshake request from a HTTP connection to a server to upgrade the connection. Along with the upgrade request header, the handshake request includes a 64-bit Sec-WebSocket-Key header. The server responds with a hash of the key in a Sec-Websocket-Auth header. This header exchange prevents caching proxy from resending previous WebSocket exchanges. From that point, the connection is binary and does not conform to HTTP protocol. The WebSocket API is an advanced technology that makes it possible to open a two-way interactive communication session between a client and a server.

### 2.4. ROS

Robot Operational System (ROS) is used to communicate with the robot. Once a command is sent from the user interface through the WebSocket, this information is processed, and with ROS the necessary connection with the robot's hardware is created. Then the commands set by the software are executed. ROS works with python version 2.7, scripts can be written interactively via a terminal or in separate files - as in the case of system development. ROS is sometimes called a "meta operating system" because it performs many of the functions of an operating system. One of its main purpose is to provide communication between the user, the Ubuntu OS and, of course, Baxter. As with any operating system, the benefit of ROS is the hardware abstraction and low-level control of Baxter without the Baxter user knowing all the details of the robot. The robot, as well as the control station, must be connected to the same internet network so they can communicate with each other. ROS has several visualization and easy-to-read controls. For example, MoveIt is a planning framework for movements and operations. It makes it easier to take some of the necessary coordinates needed for proper positioning of the robot's hands and head.

## III. HARDWARE

### 3.1. Introduction

Two types of hardware are used in the system - “Baxter robot” and a high quality camera AUSDOM AW615 1080p. The camera is used used to track the robot state by an operator.

### 3.2. Baxter robot [3]

#### 3.2.1. Physical specifications

The robot has 3’ 1” height without pedestal and between 5’ 10” and 6 ’3” with the adjustable pedestal. It’s maximum reach is1210 mm. The torso mounting plate diameter is 13.3” and it‘s used for mounting on table. It has body weight 165 lbs. without pedestal and 306 lbs. with pedestal. The robot has 14 Degrees of Freedom (DOF) - 7 per arm.

The pedestal footprint is with dimentions 36” × 32”. Baxter’s max Payload is 5 lb / 2.2 kg and he has gripping torque maximum 10 lb /4.4 kg.

#### 3.2.2. Computer and sensor specifications

The robot has 3rd Gen Intel Core i7-3770 (8MB, 3.4GHz) processor [10] w/HD4000 Graphics, 4GB memory (NON-ECC, 1600MHZ DDR), 128GB Solid State Drive Storage. Baxter’s camera has max resolution 1280 x 800 pixels, it’s effective resolution is 640 x 400 pixels and has 30 frames per second frame rate. The camera’s focal length is 1.2 mm. The robot has screen on his head with screen resolution 1024 x 600 pixels. On his head he has also infrared sensors [11] with range between 1.5 – 15 in / 4 – 40 cm.

#### 3.2.3. Electrical specifications

The supply voltage of the robot is 120 volts alternating current with rated current - 6 amps. The robot can operate on battery – he has internal DC-to-120V AC Inverter [12] It‘s important to note that the Baxter robot has an internal PC, which cannot be powered directly off of 24V DC. Baxter has standard 120VAC power interface. Robot power bus and internal PC both have “universal” power supplies and support 90 – 264V AC (47 – 63Hz). It’s maximum consumption is 6A at 120V AC, 720W max per unit, the maximum efficiency varry between 87% to 92%. For power supply the robot uses medical-grade DC switching power supply for robot power bus. Baxter has also tolerance to Sags – they are tolerated to 90V but sustained interruption will require manual power-up. The Voltage Flicker holdup time is 20mS and the voltage unbalance supports single phase operation only.

### 3.3. Camera AUSDOM AW615 1080p

#### 3.3.1. Details

The camera has 123g. weight and dimensions 66*35*135mm

#### 3.2.2. Lens Spec

They have focus from 30cm to infinity. The material, used for create them is PC-6100D1, 4-layer film-coated glass lens group. They have 65° horizontal viewing angle and support zooming.

#### 3.2.3. Microphone Spec

The microphone is built-in. It has S.P.L 32dB±2Db and it's omni-directional

#### 3.2.4. Hardware Spec

The camera's chipset is P269+MA1080, the control IC is P269(REALTEK RTS5822)QFN46 and the sensor is MA1080(5B3)CSP-48 1/4.5 CMOS "SAMSUNG". The user can choose between USB3.0 [13]/USB2.0/1.1 for connection interface. The camera's power consumption is ≤220MA and it supports different Operating Systems: Windows 7, Windows XP2, Windows 8 and Windows Vista

#### 3.2.5. Video/Image Spec

The video resolution is maximum 1920*1080 5fps [14] and minimum - 160*120 30fps. There are 2 video modes: the default one - YUY2 with max resolution 1920*1080 30fps and MJPG with min. resolution 160*120 30fps. The photo resolution is 1920*1080 and the photo format is JPG [15].

## IV. Experiments and Results

### 4.1. Robot tests

The system allows the user to execute up six different commands - three for movement of the head and three for movement of the robot's arms:

- Rotate the robot's head to the left
- Rotate the head of the robot to the right
- Neutral position of the robot's head
- Hands stretched upwards
- Hands stretched sideways
- Neutral hand position

The test set-up is show on figure 1

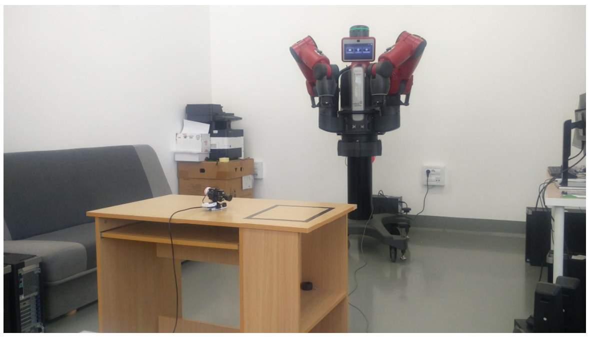

Fig. 1. – View of the system's test set-up

The figures below show the movements of the hands of Baxter and the movement of the robot's head

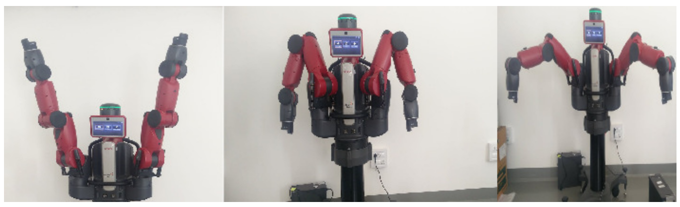

Fig. 2. - Examples of execution of a command for arm movement set by the GUI

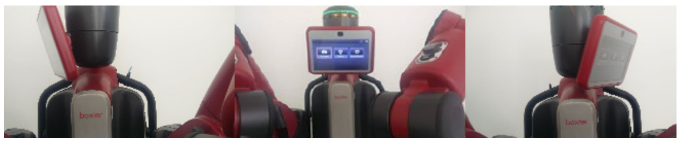

Fig. 3. - Examples of execution of a command for head movement set by the GUI

### 4.2. Software testing

The developed management software has seven buttons - one for each movement performed by the robot, and one for the connection between the individual workstations. The software also displays the information received from the camera.

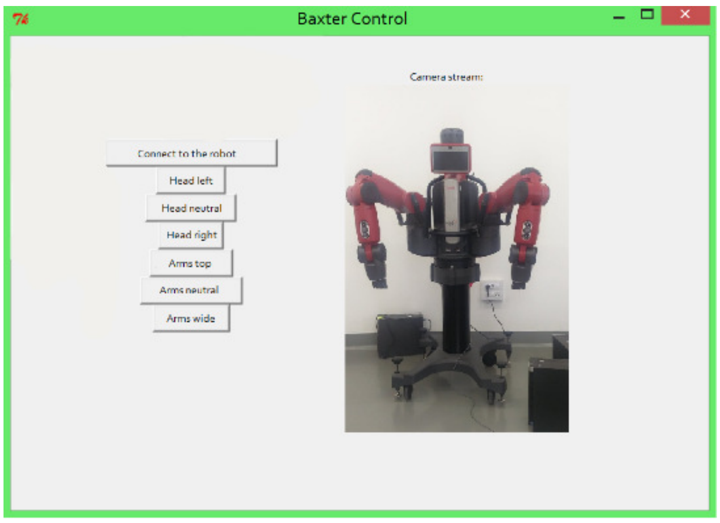

Fig. 4 – View of robot's control software

### 4.3. Testing the performance of the system

The system has been tested for its performance. Table 1 below describes:

- Command - the command sent from the user interface 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 1 - REACTION TIME OF THE SYSTEM

| Command | Reaction time (s) | Status |
|---|---|---|
| Head position neutral | 1.35 | OK |
| Head position neutral 2 | 1.19 | OK |
| Head position left | 1.32 | OK |
| Head position left 2 | 1.41 | OK |
| Head position right | 1.21 | OK |
| Head position right | 1.23 | OK |
| Arm position neutral | 3.21 | OK |
| Arm position neutral 2 | 3.32 | OK |
| Arm position straight up | 2.53 | OK |
| Arm position straight up 2 | 2.46 | OK |
| Arm position wide | 2.37 | OK |
| Arm position wide 2 | 2.39 | OK |

## V. CONCLUSION

As a result from the tests, it is clear that the system performs all of the required tasks - running on Windows OS, connecting to the Linux workstation, controlling the robot's actions and displaying real-time performance. Also the overall reaction time of the system is very fast. Still the reaction time depends more or less on the internet connection of the user, so the results received by different users may differ from the experimental data given in this paper. The future development of the system includes improvements to the graphical interface - modernizing the view as well as adding more execution commands to be performed by the robot. Furthermore, the camera will be replaced with Microsoft Kinect in order for the robot to be controlled by the actions of a person standing in front of the robot's sensor by following the person's movements.

## ACKNOWLEDGMENT

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] S. Cremer, L. Mastromoro and D. O. Popa, "On the performance of the Baxter research robot," 2016 IEEE International Symposium on Assembly and Manufacturing (ISAM), Fort Worth, TX, 2016, pp. 106-111.

[2] A. T. Chen and K. I. Wang, "Computer vision based chess playing capabilities for the Baxter humanoid robot," 2016 2nd International Conference on Control, Automation and Robotics (ICCAR), Hong Kong, 2016, pp. 11-14.

[3] M. Levene and J. Bar-Ilan, "Comparing Typical Opening Move Choices Made by Humans and Chess Engines," in The Computer Journal, vol. 50, no. 5, pp. 567-573, Sept. 2007.

[4] T. Zhao, J. Yuan, M. Zhao and D. Tan, "Research on the Kinematics and Dynamics of a 7-DOF Arm of Humanoid Robot," 2006 IEEE International Conference on Robotics and Biomimetics, Kunming, 2006, pp. 1553-1558.

[5] 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.

[6] H. Wei, Z. Huang, Q. Yu, M. Liu, Y. Guan and J. Tan, "RGMP-ROS: A real-time ROS architecture of hybrid RTOS and GPOS on multi-core processor," 2014 IEEE International Conference on Robotics and Automation (ICRA), Hong Kong, 2014, pp. 2482-2487.

[7] 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.

[8] H. Upadhyay, H. A. Gohel, A. Pons and L. Lagos, "Windows Virtualization Architecture For Cyber Threats Detection," 2018 1st International Conference on Data Intelligence and Security (ICDIS), South Padre Island, TX, 2018, pp. 119-122.

[9] B. A. Malloy and J. F. Power, "Quantifying the Transition from Python 2 to 3: An Empirical Study of Python Applications," 2017 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM), Toronto, ON, 2017, pp. 314-323.

[10] L. Dake, C. Zhaoyun and W. Wei, "Trends of communication processors," in China Communications, vol. 13, no. 1, pp. 1-16, Jan. 2016.

[11] K. Kudlaty, A. Purde and A. W. Koch, "Development of an infrared sensor for on-line analysis of lubricant deterioration," SENSORS, 2003 IEEE, Toronto, Ont., 2003, pp. 903-908 Vol.2.

[12] K. Rahimi, A. N. Motlagh and M. Pakdel, "A new soft-switched ZCZVT DC-AC inverter," 2009 IEEE Vehicle Power and Propulsion Conference, Dearborn, MI, 2009, pp. 1338-1344.

[13] P. Wüstner et al., "The use of USB 3.0 for fast data transfer in a PET detector," 2014 19th IEEE-NPSS Real Time Conference, Nara, 2014, pp. 1-2.

[14] M. Sultanoff, "A 100,000,000 Frame per Second Camera," in Journal of the Society of Motion Picture and Television Engineers, vol. 55, no. 2, pp. 158-166, Aug. 1950.

[15] P. B. Pawar, P. K. Kawadkar, M. Nagle and P. K. Ambare, "Analysis of signature and signature free bufferoverflow detection for gif and jpg format," 2013 Tenth International Conference on Wireless and Optical Communications Networks (WOCN), Bhopal, 2013, pp. 1-5.