Contactless Controlled Low-cost Robotic System

Srđan A. Milojević

***Abstract*—This paper presents and depicts a realization of a robotic arm with four degrees of freedom based on cheap components and software for its control with hand gestures. A position of the arm is determined by processing the acquired data from the sensor. The application is developed using free software packages. Results of this project could find use in industrial robotics, therapeutic biofeedback, telemanipulation systems, etc.**

***Key words* — acquisition, LeapMotion, Python, rehabilitation, robot, control.**

I. Introduction

Research of robotic systems was initially mainly oriented towards the application in industry. However, modern technology has led to the expansion of other fields of research [1], too. As a result, household and therapeutic robots [2] appeared along with the more advanced robots useful in the care of elderly people with severe dementia [3] or even subacute stroke [4] and finally telemanipulation systems that assist coronary bypass surgeries [5] - [7] etc.

A very important factor in the development are contactless Human Machine Interfaces (abbr. HMI) which simplify the control of robotic systems [8]. Leap Motion controller has proved itself as an efficient and reliable device for implementation of HMI.

In this paper, the author describes the main topics of the four degrees of freedom robotic arm constructed with low-cost components and the software created for its operation and real time control.

Chapter II is dedicated to the construction of the arm and the components needed for its realization. Additionally, the use and signal processing of Leap Motion sensor are thoroughly explained. Finally, control algorithm and inverse kinematics equations of two segment manipulator are discussed.

In chapter III the graphical user interface of the control software is shown, while chapter IV points out the vast variety of possible applications and implementations of the created robotics system.

Paper received April 24, 2018; revised June 22, 2018; accepted July 29, 2018. Date of publication December 25, 2018. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Nataša Nešković.

This paper is a revised and expanded version of the paper presented at the 25th Telecommunications Forum TELFOR 2017 [17].

Srđan A. Milojević, School of Electrical Engineering, Bulevar kralja Aleksandra 73, 11120 Belgrade, Serbia (telephone: 381-63-7558982, e-mail: srdjanmilojevic195@gmail.com).

II. Material and Method of Work

A block diagram of the control system for converting signals acquired from Leap Motion sensor to angular movement of joint actuators of the robotic arm is shown in Fig. 1. In subsection A. the author discusses the hardware configuration and the specific reason for choosing the robot arm's design and all the other used parts.

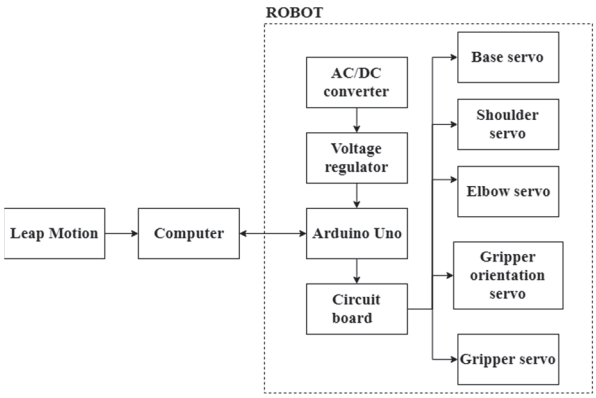

Fig. 1. Block diagram of the robotic system.

A. Hardware design

The constructed robotic system is depicted in Fig. 2.

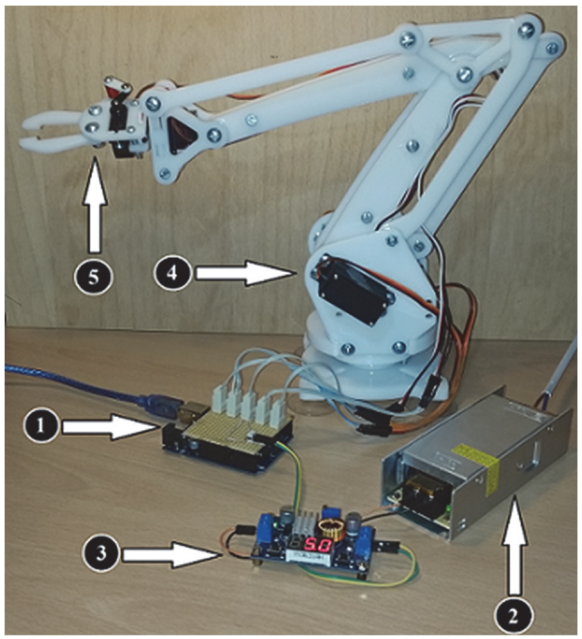

Fig. 2. Main components: Arduino Uno controller with circuit board (1), AC/DC modulator (2), buck down converter (3), base of the arm with the actuators (4), gripper (5).

A robot represents a kinematic chain with four degrees of freedom. The first three joints position the end effector in reference (*x, y, z*) coordinate system while keeping the gripper parallel to the floor and the fourth rotates the gripper in a horizontal plane. All joints are revolute kinematic pairs which means they have only one degree of freedom, and act as a hinged joint.

The robotic arm is designed with reference to the model of ABB IRB 460 (ABB, Zurich, Switzerland) industrial robot [9], constructed for the purpose of palletizing. Five-bar linkage mechanism of this robot enables that the TCP (*Tool Center Point*) coordinate system stays level to the surface during all movements. Another benefit of this mechanical design is that all the actuators are displaced from rotational axis to the mutual base of the arm. This results in the minimization of static load which compensates the low torque of MG996R [10] servo motors. Because of this feature, such a configuration was used in the other project too, such as [11].

For the purpose of designing the robotic arm, the software package AutoCAD (Autodesk, San Rafael, California, USA) was used. Afterwards, it was important to simulate the mechanism and a reliable system for that was the software package SolidWorks (Dassault Systèmes, Vélizy-Villacoublay, France). The assembly of all the parts in this project, later also used for 3D graphical visualization, is shown in Fig. 3.

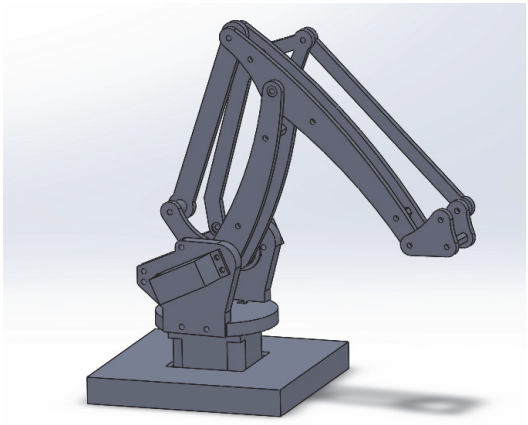

Fig. 3. Solidworks assembly of the robotic arm.

After minor structural and visual adjustments the final layout of the desired parts is presented in Fig. 4. The building material chosen for this project is 5mm thick Plexiglas because of its balance between price and durability. It was then laser cut into the corresponding shapes. Other, more complex parts were 3D printed, while flange shaft bearings are placed in each joint in order to reduce friction. The end effector is a horizontal gripper, but it can easily be changed due to clever design of the robotic arm. Consequently, the user can create its own end effectors that best suit their needs.

The realized low-cost robotic system is constructed mainly of cheap components with the total expenses summing up to around one hundred Euros (Table 1). This leads to a conclusion that just by upgrading these low-quality actuators, controllers and sensors the end result would be greatly improved.

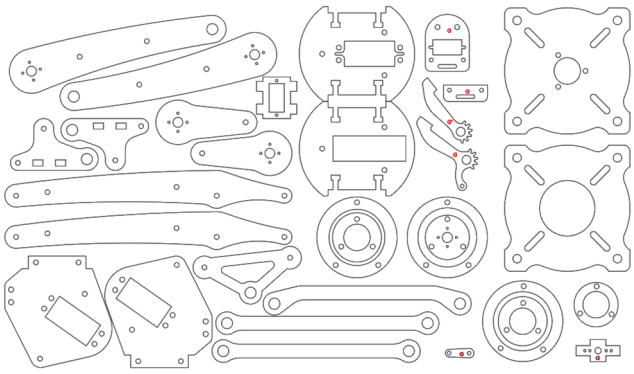

Fig. 4. CAD drawing of the needed parts.

TABLE 1: LIST OF COMPONENTS AND PRICES

Parts	***Price (€)***
1 x Arduino Uno (Arduino, Somerville, SAD)	20,00
3 x MG996R servo motor (GREAT WALL Electronics Co., Ltd., Kina)	15,66
2 x MG90S micro servo motor (HappyBuy Store, China)	6,51
12 x MF84ZZ ball bearings (Tecla&Chiara Store, Kina)	7,49
1 x Power cord	3,20
1 x McIgIcM 5A, 75W, DC-DC *buck-down*converter (Professional semiconductor suppliers, Kina)	5,86
1 x LED power supply12V, 5A, 60W (Tingtop Trading CO Ltd., Kina)	10,19
1 x Plexiglas (30x50cm) and laser cutting	18,00
Others (nuts, bolts, cables etc.)	15,00
Sum	**101,91**

Robot is powered from mains electricity (alternating-current, 230V) which is then converted with an AC/DC converter to the output of 12V and 5A. The next step is to lower the voltage down to 5V using buck-down McIgIcMconverter. External power supply with a high value of direct-current was important in order to achieve smooth robotic movements with simultaneous activity of actuators (MG996R operate with stall current of 2.5A, and running current of 0.9A). The next subsection introduces the reader to the Leap Motion (Leap Motion Inc., San Francisco, California, USA) sensor and the data acquired from it.

B. Sensor system

Leap Motion sensor is a small peripheral USB device which uses two monochromatic infra-red cameras and three infer-red LED to very accurately synthesize a 3D representation of user's hand by combining two 2D images. Apart from its original application in the development of virtual reality and games, this sensor has come across large popularity in other fields of research, such as biomedicine and systems control. This resulted from Leap Motion's simple and straightforward implementation [12]-[13].

The information of interest which the sensor provides are

the position (*x,y,z*) and the angles of yaw φ, roll ψ and pitch θ of the normal vector which points perpendicularly out of the hand as shown in Fig. 5. Processing of these data enables the contactless control of the robotic arm, i.e. translation of user's gestures to robot's movements [14].

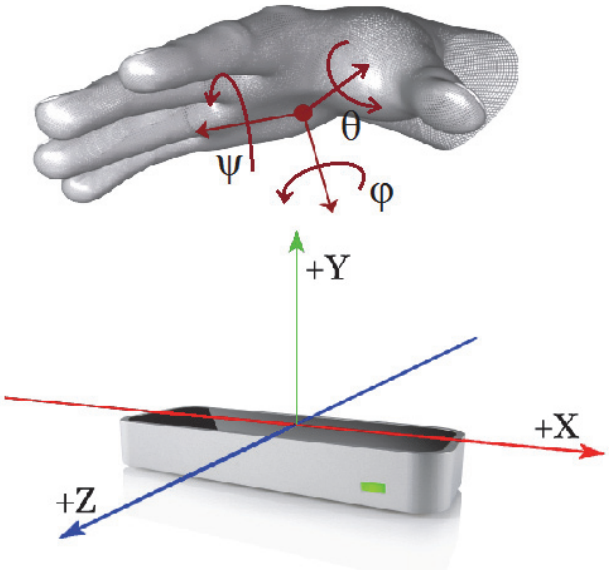

Fig. 5. Leap Motion sensor and its referent coordinate systems.

The way these signals are used and the communication between the computer and the Arduino UNO controller is thoroughly explained in the following subsections.

C. *Control algorithm of the robotic arm*

Leap Motion sensor sends the needed information about user's hand to the computer via a USB connection. A program written in the software package Python 2.7.15. processes the received signals of the position and rotation and forms control signals (see subsection D. for inverse kinematics) which are then transferred to Arduino Unocontroller (Arduino, Somerville, USA).A diagram of the application flow is illustrated in Fig. 6.

The communication between Arduino Uno controller and the computer is being operated through a serial port which transfers coded commands as char string variables. The received commands are then decoded in the controller itself. For example, with the program running on Arduino Uno, a code "WS3:95" is translated into "set the angle of servo number three to 95 degrees".

A graphical user interface (GUI) of the application was developed with Python PyQt4 library (Riverbank Computing Limited, Dorchester, UK) and designed using the software package Qt Designer (The Qt Company, Oslo, Norway). Apart from that, the program uses additional libraries such as NumPy (Open Source), Leap (Leap Motion Inc., San Francisco, California, USA), MatPlotLib (Open Source), PyGame (PyGame Community), PyOpenGL (Source Forge), etc. However, using these many different libraries comes with a price. The issue with running this program on other computers is the need for all these and some other Python packages. Before using the created system, the operator will have to follow detailed instructions on how to install the Python package manager better known as pip (Python Software Foundation, Wilmington, Delaware, USA). Therefore, these installations can be conducted in matter of minutes, given that the user has followed the instructions thoroughly.

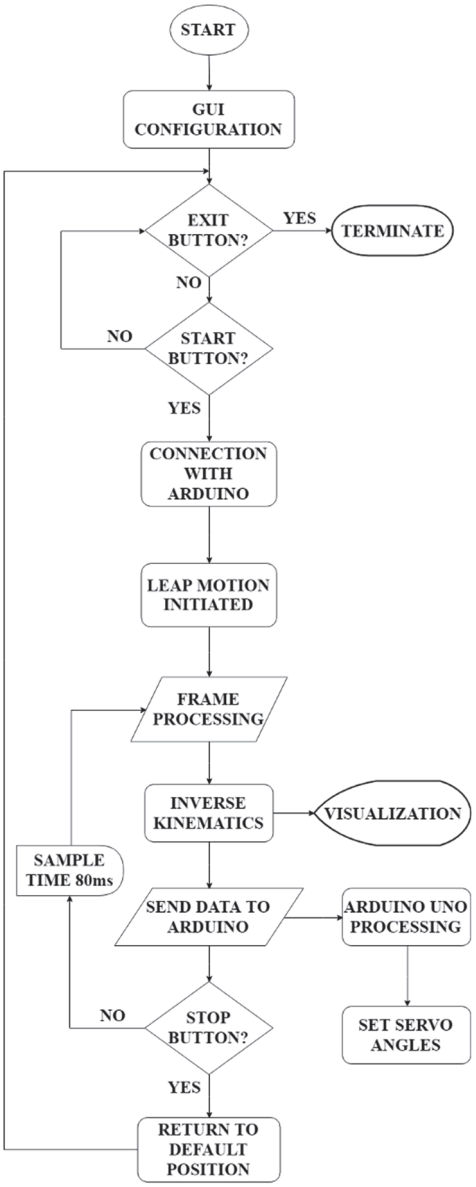

Fig. 6. Diagram of the application.

D. *Inverse kinematics*

Relation of the desired (*z, y)* position and the inner coordinates θ_1 and θ_2 (Fig.7.) for the example of two-segment begins with the following equations (1) and (2):

$$z = l_1 \cos\theta_1 + l_2 \cos(\theta_1 + \theta_2) \quad (1)$$

$$y = l_1 \sin\theta_1 + l_2 \sin(\theta_1 + \theta_2) \quad (2)$$

The final forms of the transformation equations for position and angle are:

$$\theta_1 = \arccos\frac{l_1^2 + y^2 + z^2 - l_2^2}{2l_1 l_2} + \arctan\frac{y}{z} \quad (3)$$

$$\theta_2 = 180^\circ - \arccos\frac{l_1^2 + l_2^2 - y^2 - z^2}{2l_1 l_2} \quad (4)$$

where l_1 and l_2 are the lengths of arm's segments, z and y

are position of the end effector. Additionally, *x* axis motion is implemented in such a way that it enables easy and intuitive control. It is dictated by the velocity of user's palm in *x* axis direction in Leap Motion's reference system. The default angle state is either incremented or decremented according to the amount of speed.

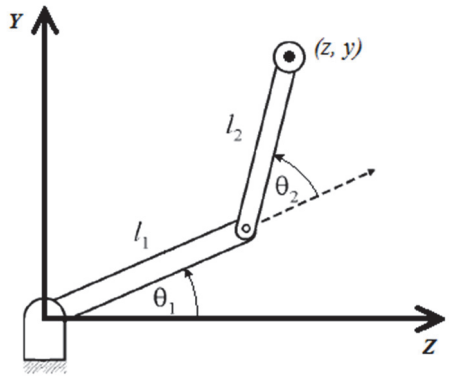

Fig. 7. Two dimensional representation of the robot arm.

In the Table 2 we see available robot arm movements and the corresponding user's gestures.

TABLE2: HAND GESTURES FOR THE CONTROL

Hand gesture	***Robot's motion***
Opening of the fist	Gripper opens
Closing of the fist	Gripper closes
Upward movement	End effector raises
Downward movement	End effector lowers
Movement to the left	End effector moves to left
Movement to the right	End effector moves to right
Palm rotation	End effector rolls

E. Software development

The robotic arm needs a convenient and easy to use environment for the operator. Therefore, a combination of software packages was needed for the development of GUI. First of all, the previously explained control algorithm was written in Python 2.7 because of its versatility and ability to communicate with other third party software packages. Because SolidWorks files tend to describe parametric geometry, such as NURBs, and most game engines prefer to work with raw polygons, the parts needed conversion. To begin with, using Free CAD (Jürgen Riegel, Werner Mayer, Yorik van Havre) the author opened each SOLID file and exported it as Mash format (OBJ). The output files consist of vectors for vertexes and faces but there are no materials for normals. Current OBJ files would also work but the visualization would lack shadows and lightings so the next step utilizes Blender 2.79 (Blender Foundation, Amsterdam, Netherlands). After editing color and material, the files are exported with a custom setting in OBJ file format. Secondly, the author uses an open source code that provides a function for loading a model from a wave frontOBJ file into an OpenGL display list. It additionally loads any referenced material and texture files. The code can be found on PyGame (PyGame Community) Wiki page [15]. With a PyGame engine the author was able to create 3D graphical visualization (see chapter III. Results).

III. RESULTS

Upon opening the application, the user distinguishes three segments: a command part with buttons, indicators of current angular positions of servo motors and real time graphical visualization of the position of robotic arm (Fig. 8). By choosing the button "Start Leap", the connection with Arduino Uno controller is initiated, followed by Leap Motion's turning on. To pause tracking, user places his or her other arm in the sensor's field of view and the control resumes when only one arm is present. Finally, choosing the option "Stop Leap" returns the robotic arm to its default position and the program waits for a new connection. Pressing the button "Quit" terminates the program.

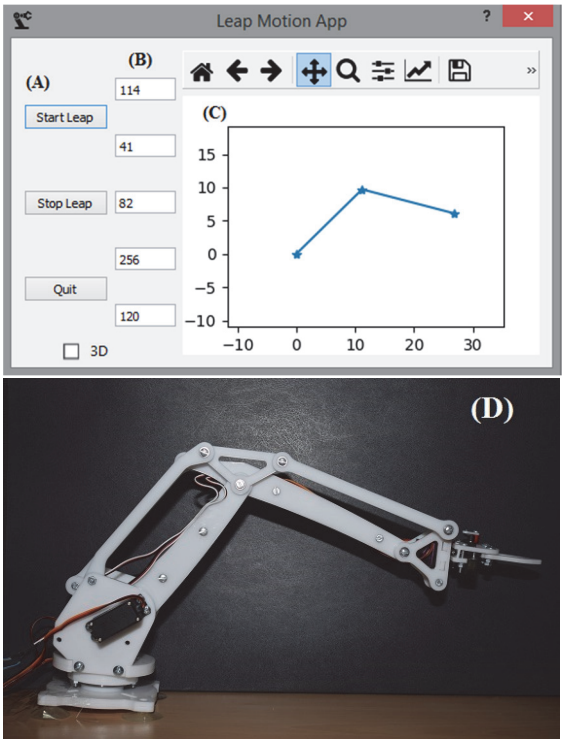

Fig. 8. Graphical user interface with its segments: (A) controls, (B) indicators, (C) visualization, (D) current position of the arm.

On the very bottom of the command part of GUI there is a checkbox whose checking opens up an additional window. The PyGame window (Fig. 9.) is a three dimensional real time representation of robot's current pose.

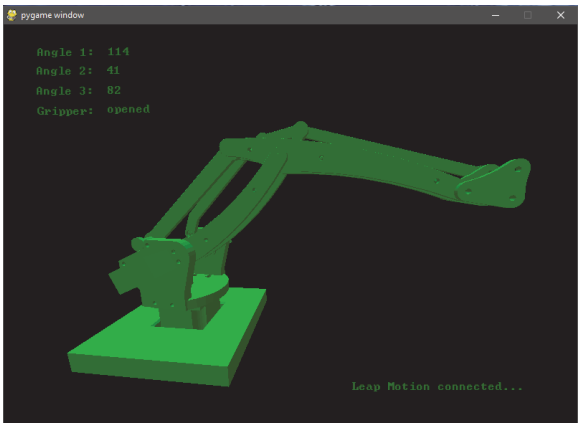

Fig. 9. Three dimensional model in real time.

In the upper left hand corner, a user sees indicators about the angle of robot's joints and in the bottom corner there is a notification line next to the base of the robot. The user can adjust the orientation of the perspective, zoom in and out, and pan the robot across the screen. Adding features such as recording of this 3D visualization will be helpful for analyzing robot's and user's behavior. Also, this would allow an implementation of algorithms for robot's offline programming. These possibilities, along with a few others, will be discussed in the next chapter.

IV. Discussion and conclusion

The created system has a fast response rate and is efficient in real time operation. The software is written in a modular and organized way and other developers can easily modify or continue the project.

Concept of the described system can find its use in industrial robotics where it could replace traditional methods of path programming. The old and outdated teach pendants [16] are rather difficult to operate and too time consuming. This contactless control is more intuitive, minimizes room for error and is more time efficient. With its use, paths and targets would be memorized faster and cheaper, which is exactly what the industry always strives for. Along with the 3D visualization comes an opportunity to program robot's paths and tasks offline, without the presence of the actual robot. Afterwards, the user could simply upload the code created in offline editor to the robot's controller and watch it operate.

Another interesting possibility for implementation of the created system is in biofeedback systems aimed towards the rehabilitation of children suffering from cerebral palsy. Adapting the interface of the application for this purpose and implementing interesting tasks, the therapy would get less stressful character and the patient would be occupied with the control of the robotic arm while the doctor uninterruptedly observes the rate of improvement. What is more, the doctor could choose to record 3D visualization of each session and hence get a better insight into the patient's problem. Also, the software could keep a track of all the therapies and suggest a better approach to the treatment.

Moreover, an obvious idea that comes to mind with this contactless control is real time telemanipulation control of robotic systems. This product could save lives of people working in hazardous areas. For example, placing this robotic arm on a movable platform, a soldier could navigate it to a place where a bomb is planted and demount it quickly, easily and, most importantly, from a safe distance. Upgrading the 3D model viewer with a bomb defusing simulation would enable soldiers to practice on this platform before coming in contact with real bombs and save equipment while getting used to operating this system.

In addition to the mentioned implementations, the created hardware along with the Python program makes a good platform to study and develop a machine learning algorithm. This is perfect for students since the whole robotic system is very affordable and desktop ready. Moreover, its fun usage is a clever way to interest teenagers to study robotics as it is becoming a more and more important branch in technology and industry.

There are numerous ways to improve the created system. First of all, simply by replacing these low-cost components with high-end products the performance would be significantly enhanced. To be precise, switching from servo to stepper motors with feedback, the control regulation would be upgraded. Motors with a stronger torque would increase pay load of the robotic arm. Finally, replacing Arduino Uno controller with a more sophisticated one (more processing power and memory) would help with the control system.

Acknowledgements

The author expresses his gratitude to the laboratory for Biomedical Instrumentation and Technologies, Laboratory for electrical measures and ETF Robotics team at the School of Electrical Engineering, University of Belgrade for the support during the research. The author owes special credit to dr Milica Janković for great dedication and mentorship.

References

[1] Branislav Borovac, Goran Đorđević, MirkoRaković, i Milan Rašić, *Industrijska robotika*,University of Novi Sad, Faculty of TechnicalSciences, FTN publishing, Novi Sad, 2017, ch. 1.

[2] Kazuyoshi Wada, Takanori Shibata, *Robot Therapy in a Care House - Its Sociopsychological and Physiological Effects on the Residents,* National Institute of Advanced Industrial Science and Technology 1-1-1 Umezono, Tsukuba, Ibaraki, Japan, 2006.

[3] Toshiyo Tamura, Satomi Yonemitsu, Akiko Itoh, *Is an Entertainment Robot Useful in the Care of Elderly People With Severe Dementia?*, Department of Gerontechnology, National Institute for Longevity Sciences, Aichi, Japan, 2004.

[4] M. Iosa, G. Morone, A. Fusco, M. Castagnoli, FR Fusco, L. Pratesi, S. Paolucci, "Leap motion controlled videogame-based therapy for rehabilitation of elderly patients with subacute stroke: a feasibility pilot study," *Top Stroke Rehabilitation*. 2015 Aug;22(4):306-16.

[5] Garcia-Ruiz A, Gagner M, Miller JH, et al: "Manual vs robotically assisted laparoscopic surgery in the performance of basic manipulation and suturing tasks." *Arch Surg* 133: 957– 960, 1998.

[6] H. Shennib, A. Bastawisy, MJ. Mack, et al: "Computer-assisted telemanipulation: an enabling technology for endoscopic coronary bypass." *Ann ThoracSurg* 66: 1060–1063, 1998.

[7] Tooher R, Pham C. *The da Vinci Surgical Robotics System: Technology Overview ASERNIP-S Report No. 45*. Adelaide, South Australia: ASERNIP-S; 2004. ISBN: 0909844658.

[8] D. Bassily, C. Georgoulas, J. Güttler, T. Linner, T. Bock, *Intuitive and Adaptive Robotic Arm Manipulation using the Leap Motion Controller*, TU München, Germany, 2014

[9] ABB IRB 460 Datasheet, ABB, Zurich, Switzerland http://new.abb.com/products/robotics/industrial-robots/irb-460

[10] http://www.electronicoscaldas.com/datasheet/MG996R_Tower-Pro.pdf

[11] https://github.com/mimeindustries/MeArm

[12] Leap Motion SDK (software development kit) for Python - https://developer.leapmotion.com/documentation/python/devguide/Leap_SDK_Overview.html

[13] Mischa Spiegelmock, *Leap Motion Development Essentials*, Packt Publishing, Birmingham, 2013.

[14] Chang Chen, Liang Chen, Xuefeng Zhou, Wu Yan, "Controlling a robot using leap motion," *2nd International IEEE Conference on Robotics and Automation Engineering (ICRAE)*, 2017

[15] https://www.pygame.org/wiki/OBJFileLoader

[16] ABB Flex Pendant SDK Application manual, ABB Robotics https://library.e.abb.com/public/b60548090c886120c1257b5f002f8250/3HAC036958-en.pdf

[17] S. A. Milojević, "Beskontaktno upravljani jeftini robotski sistem," *2017 25th Telecommunication Forum (TELFOR)*, Belgrade, 2017, pp. 1-4.