Master of Science

A Study on the Design and Control of Low-Cost Humanoid Arm using 3D Printing

The Graduate School

of The University of Ulsan

Department of Mechanical Engineering

Dakarimov Sayat

A Study on the Design and Control of Low-Cost Humanoid Arm using 3D Printing

Supervisor: Soon Yong Yang

A Dissertation

Submitted to

the Graduate School of the University of Ulsan

In partial Fulfillment of the Requirements

for the Degree of

Master of Science

by

Dakarimov Sayat

Department of Mechanical Engineering

Ulsan, Korea

April 2019

A Study on the Design and Control of Low-Cost Humanoid Arm using 3D Printing

This certifies that the master's thesis

of Dakarimov Sayat is approved

[illegible]

Committee Chair Prof. Byung Ryong Lee

[illegible]

Committee Member Prof. Kyoung Kwan Ahn

[illegible]

Committee Member Prof. Soon Yong Yang

Contents

List of Figures

List of Tables

Chapter 1. Introduction

1.1 Purpose of Study

Recently, a lot of research has been carried out in the field of humanoid robot development. The reason for the high interest shown in the humanoid robot development relationship is the functional characteristics of humanoid robots and its wide industrial application. However, major developments are underway in the field of creating upright erect bipedal robots or high-level robots, such as talking robots using artificial intelligence or androids that mimic human facial expressions. Meanwhile, fully functional robotic arms are also an equally unimportant factor in the development of humanoid robots. Therefore, in this research we will try to study the methods of control of the robotic arm produced using 3D printing technology and methods for its implementation, also to create a controller for the humanoid robot arm based on the data obtained during the study.

The purpose of this study is to develop a control system for humanoid robot arms. Up to now, research on the development of humanoid robot arm control systems has rarely been done. Most industrial robots have only one manipulator unit, which, in turn, is the reason for the limited ability to work, the number of simultaneously performed operations and the narrow workspace. However, increased number of manipulators requires more complex control system. Thus, exploring the controls based on the robotics kinematics and the development of the control system of one humanoid robot arm including a hardware-software complex, we will develop a control system for a two-handed robot, that is, a dual arm robot based on the data obtained during conducted study.

Purpose of Study:

- Construction of low-cost humanoid robot arm using 3D printing technology
- Study on hardware and control of humanoid robot arm
- Study on control system

1.2 Research problems

In accordance with the purpose of this study, development of low-cost humanoid robot arm using 3D printing technology, the following problems should be solved:

1) Design, production using 3D printing technology and assemble of humanoid robot arm

2) Development of control system hardware using Arduino

3) Development of control system software using Arduino IDE and MATLAB/Simulink

Since the main goal of this study is to create a manipulator control system, the main emphasis will be placed on this.

1.3 Research content and scope

To achieve the goals of this study, the following work must be completed:

1) Printing necessary parts of humanoid robotic arm using 3D printer
2) Assembly of printed robotic arm hardware in laboratory conditions
3) Development of control system hardware using Arduino and compatible microcontrollers
4) Forward and inverse kinematics calculation of humanoid arm. Numerical analysis of obtained results using MATLAB
5) Software development based on manipulator kinematics using MATLAB / Simulink software

The scope of humanoid robots is very wide; hence, it seems appropriate to narrow scope of study by researching humanoid robots available for producing using 3D printing technology.

Since the time of the first research and significant results in the field of humanoids, many robots and robotic platforms have been developed for various purposes, which will be discussed in the next chapter. Most of these robots were designed to demonstrate the latest achievements of the company, as well as to advance the trends of future developments, and are therefore not available for a wide range of developers and students. And those that are more or less available are very expensive and not everyone can afford to purchase them. A significant exception to these rules is robots and robots developed with the possibility of production using 3D printing technology.

The ability to produce and assemble the using 3D printing technology also assumes that the robot will be in the public domain, that is, a robotic platform with an open source.

Thus, we narrowed the field of research to open-source robots, with the possibility of production using 3D printing technology.

Considering all of the above, we have chosen a robotic platform as the object of study.

Chapter 2. Theoretical background

2.1 Humanoid Robot

Humanoid robot - is a robot with its body shape built to resemble human body. The design may be for functional purposes, such as interacting with human tools and environments, for experimental purposes, such as the study of bipedal locomotion, or for other purposes. In general, humanoid robots have a torso, a head, two arms, and two legs, though some forms of humanoid robots may model only part of the body, for example, from the waist up. Some humanoid robots also have heads designed to replicate human facial features such as eyes and mouths.

Prefix "humanoid" means a robot with its body shape designed to resemble human body. Using robots in production line and in other fields of industry helps to reduce human factor related error. Most industrial robots have only one manipulator unit, which, in turn, is the reason for the limited ability to work, the number of simultaneously performed operations and the narrow workspace. One of the most effective solutions to this problem are dual arm robots. However, increased number of manipulators requires more complex control system.

Recent researches on robot development show that humanoid robots are being developed a lot in the world. The reason for this is the functional characteristics of humanoid robots and its wide industrial application. The most common design purposes are functional - interacting with human tools and environments, experimental - study of locomotion of human body and robot design improvement, and research - to develop computational model of human behavior, to build better orthosis and prosthesis for people with physical disabilities.

2.1.1 Types of Humanoid Robots

Humanoid robots are complex system, which consist of mechanical structure, control system and control algorithms. The robot's size may vary from small to human sized depending on the purpose of building and application field.

In general, there are two application categories, hence two functional design differences between humanoid robots (Stasse, O., and T. Flayols., 2018).

1) Physical performance humanoid robots
2) Biological and/or cognitive models

As an example of a robot from the first category one can take the ATLAS robot from Boston Dynamics, and for the second category, the Kenshiro robot [43] from Tokyo University is good example.

According to [51] when the goal is to have a walking robot the mass disruption and undesirable mechanical resonance should be considered. When the goal is robot-human interruption the control precision of the robot is not the main objective.

Common design purposes or humanoid robot types [12].

a) Human assistance and entertainment, hence “living with human”. Example, QRIO by Sony [8] [9].
b) Working in human environment and using of human tool Example, HRP-4 by Kawada Industries [48]
c) Studying human locomotion and development of robot capable of performing a wide range of tasks. Example, ASIMO by Honda [48]
d) Entertainment and assistance in house environment. Example, robot PaPeRo [7]

2.1.2 Application of Humanoid Robots in various fields

Humanoid robots have been significantly used in healthcare and education (A. Choudhury et. al, 2018). However, there are many other application fields of humanoid robots such as home assistance, service robots, scientific research, entertainment, automotive manufacturing line, greeting guests and visitors.

The role of humanoid robots in healthcare could be divided in two categories: clinical and non-clinical. For example, in clinical setting humanoid robots have been used to assist patients with different diseases: cerebral palsy and pediatric cancer. There are also many more examples of the use of humanoid robots in medicine for different purposes, but we will not delve into this topic.

The development of technology has led to the development of multiple robot platforms for teaching robotics to children and university students. Education is one the fields of humanoid robot application. Mainly in various educational institutions humanoid robot were used in education along with human teacher or educator. There are many open source humanoid robot platforms, which will be discussed deeply in paragraph 4 of this chapter. For example, InMoov [5] robot from designer *Gaël Langevin*, Reachy [10] from Pollen Robotics and THORMANG3 [52]. These robot platforms can be purchased as ready-made samples for assembly or as drawings for a 3D printer. The second option is non-compliant, except for the cost of acquiring material for the printer and various activators.

One of the main humanoid robot application field is socially assistive robots. Social robots of socially assistive robots (SAR) is the robots developed to help and to assist people, especially elder people or people with physical disabilities. Robot can be developed as a universal assistant robot or especially for children and old people separately. Children SARs also have ability to entertain or play games.

Scientific research is a crucial field of humanoid robot application. Despite the fact that robotics technology is developing fast in recent years there are still many boundaries or in challenges for humanoid robots. For example, development of artificial intelligence for humanoid robots is still at the beginning stage since there is no fully operational complex intelligent control system for humanoid robots. However, there are some exciting examples of intelligent robots such as Sophia developed by Hanson Robotics [11]. Sophia can talk and answer to the various questions in real time.

Main challenges in studying and development of humanoid robots (Humanoid Robots, 2019):

a) Proprioceptive sensors. Sensors used to sense the position, the orientation and the speed of the humanoid robot's body and joints
b) Exteroceptive sensors. These sensors are divided into several types: tactile - can be used to provide data on what has been touched; vision - refers to processing data from any camera. In humanoid robots it is used to recognize objects and determine their properties; sound - allow robots to hear and recognize speech or environmental sounds, and perform as the ears of the human being
c) Actuators - electric devices that are used to drive robots. Main challenge in humanoid robot study is development of actuators or actuator sequences that will give mimic and performance results similar to human.
d) Planning and control. The difficulty of planning and controlling movement of humanoid robot is that the movement of robot has to be human-like, using legged locomotion, especially biped gait. The question of walking biped robot's stabilization on the surface is of great importance. Maintenance of the robot’s gravity center over the center of bearing area for providing a stable position can be chosen as a goal of control [6].

On top of everything else, robots are also used in production and allow you to automate the production process. Of course, this is nothing new, robots began to be used in industry since the 60s of the last century to complete various tasks such as welding, paint and assembling product parts. Humanoid robots were not actually used in industry, but dual arm robots developed on their basis are widely used in the production process [54][39]. Dual arm robots allow to widen workspace of robot as well as increasing the number of possible tasks.

Humanoid or dual arm robots can also be used as an assistant in laboratories or product assembly. One the latest examples of robot co-worker is YuMi developed by ABB [1]. Using robot in assistance with human called robot-human collaboration, which is completely new field or robotics technology development.

2.1.3 Future trends in Humanoid Robot development

Compared to traditional robots that move in space with the help of wheels, humanoid robots have a great advantage in the movement in different terrains, for example, climbing stairs. In addition, humanoid robots with high walking precision could be sent to hazardous environments instead of human, it would help to increase safety in the workplace for humans.

According to (Denny et al, 2016) despite the good development of walking robots have following problems: backlashes and frictions in joints, walking instability, limited payloads during motion execution, low precision motion sensors, therefore non-accurate localization.

Nowadays, various studies have been conducted in humanoid robot localization. For example, (A. Hornung et al) presented method for robot localization navigating in complex indoor environment using onboard sensors and using 3D representation. As a result, this method is able to estimate 6D pose (3D position and rotation) of the humanoid robot. There are also other studies in humanoid robot localization [30], however since it is not goal of this paper, we won’t dive deep into it.

Another trend toward humanoid robot research is mainly focused on development of humanoid robot using cognitive systems including artificial intelligence. Embedding complex control software and algorithms will help to apply studies in neurobiological science into the of humanoid robots, which includes various tasks like actuation, multiple sensor control and distributed architecture control. The development of an energy-efficient power supply system is also one of the priorities not only in the development of humanoid robots but also of whole robotics industry. Last but not least trend in humanoid robot study is focused towards making it more human-like. There are already some exciting results in this area, for example, Sophia or Albert HUBO developed in conjunction of KAIST and Hanson Robotics.

2.1.4 Open source robot platforms

According to (Techopedia, 2010) open source is a philosophy that promotes the free access and distribution of an end-product, usually software or a program, although it may extend to the implementation and design of other objects. Open source revolves around the concept of freely sharing technological information so that it may be improved through multiple insights and viewpoints. Since the technology is open source, the amount of work that needs to be done is reduced because multiple contributions are added by many individuals.

Various robot development platforms have also become part of this philosophy. An open source allows students and researchers around the world to contribute to the development of a certain technology or field. It also reduces the price of various components and make them possible to produce on a 3D printer. One of such open source robot platforms is the object of study of this paper.

The platform used is open source 3D printed life-size robot - InMoov [5]. There are other open source robots as well such as Reachy [10] from Pollen Robotics or THORMANG3 [52]. More complete technical information about is given in Chapter 3.

2.2 Precedent research

According to [63] interest in built-in type humanoid robot research is rising since first presentation of robots P2 [29], P3 and ASIMO in 1996 [49]. Robots P2 and P3 from Honda there are 7 DoF on both arms and 2 DoF on hand. In case of robot Asimo there are 6 DoF: 5 DoF on arm and 1 DoF on hand. One of the examples of study started after first development of humanoid robots is a project HRP (Humanoid Robot Project) that lasted five years and conducted by METI as a result presented a humanoid robot HRP-2 [34]. Comparing to previous robots HRP-2 has 6 DoF on arm and 2 DoF on hand, 8 DoF in total. Many such studies were under way in the world [33][25] but most of them were focused on developing biped robots. For example, walking robot LOLA [55] from Technical University of Munich or BHR (Beijing Humanoid Robot) [60] and etc.

Since first humanoid robot named HUBO developed by KAIST professor Jun-Ho Oh’s team in 2004 many studies conducted in this field in Korea as well [64]. The platform for this robot was another humanoid robot KHR-3 (KAIST Humanoid Robot-3) developed by same institution [46]. The KHR-3 has 6 DoF on arm and 5 DoF on hands. Before KHR-3 was designed considering the shortcomings of previous robots KHR-1 [36] and KHR-2 [37] After that, in 2008, robot HUBO 2 was introduced. Comparing to first generation main difference was that robot HUBO 2 also can run with speed 3.6 km. In addition to this, the research team of the HUBO developed the robot Albert HUBO [44] with Albert Einstein's head in combination with the US manufacturers of artificial intelligence robots Hanson Robotics and participated in the DRC (DARPA Robotics Challenge) with robot DRC-HUBO 2.

There are other domestic humanoid robots such as Mahru and Ahra [59], KIBO, THORMANG [52] and CHARLI-2. Robot KIBO was developed as a robotics learning kit [19] for 4-5-year-old kids. Although, domestic humanoid robots had dual-arm body structure and studies showed quite good results most of them basically had a goal of developing a bipedal walking system for humanoid robots.

The modern history of humanoid robots started from development of robot ASIMO by HONDA, as it mentioned earlier (Hirose et al., 2001). Other examples of humanoid robots are iCub, the robot developed by The RobotCub Consortium, which consists of 11 European institutions in (Tsakarakis et al., 2007) for cognitive and neuroscience research. The iCub is the only robot, which is used hand for walking, because it could only crawl till 2010, thus requiring a complex control system. General Motors and NASA developed second version of NASA's robot named Robonaut. Robonaut can use many astronaut tools and can work in the same tight corridors as astronauts (Ambroseet al., 2000). The most recognizable representative of humanoid robots at the moment is the Boston Dynamics' robot called PetMan (Petman, 2011). The main difference of PetMan from other humanoid robots is the use of hydraulic and pneumatic actuators, which resulted in showing much faster and most human-like gait performance. Biped robot Dexter from AnyBot uses the same actuator principle, which is also able to jump (Anybots, 2008). In addition, to using pneumatic and hydraulic actuators there is also growing interest to the use of pneumatic artificial muscles or air muscles [24]. Shadow Robot's pneumatic hand (Shadow Robot Company, 2003) and Festo's BionicCobot [4] or Bionic Handling Assistant show quite accurate performing and highly dexterous movements of robotic arm.

However, most studies on humanoid robots mentioned above mostly aiming to develop human-like gait and creating bipedal robots. This tendency is caused by the fact that control systems for manipulators that have been developed in different variations for use in industries can be used to control various tools and implement the assembly process. Control systems that used in industrial manipulators can be applied to control humanoid robot arm with some changes in kinematics calculation.

Despite the fact that nowadays humanoid robot is fast growing field of robotics industry it’s still on the beginning level. Creating intelligent and constructively correct robot which will be able to interact with human beings, tools and environment needs more time and study.

Chapter 3. Kinematic Model of Humanoid Robot Arm

3.1 Object of Study. Open source Humanoid Robot Platform

To achieve goal of this study open source 3D printed life-size humanoid robot platform **InMoov** (figure 1) were used. InMoov is the first Open Source 3D printed life-size robot. Replicable on any home 3D printer with a 12x12x12 cm area, it is conceived as a development platform for Universities, Laboratories, Hobbyist, but first of all for Makers. It's concept, based on sharing and community [5].

This robot has 5 DoF in each arm, 16 DoF in each hand and in additional 10 finger sensors, 10 independent motorized fingers with silicone grip. Robot can be built by using 3D printing technology as it mentioned earlier or by simply purchasing pre produced parts from seller. The reason of choosing InMoov robot before other open source humanoid robot platforms for our study is open source platform, low-cost due to self-production and inexpensive hardware components (sensors, actuators, etc.).

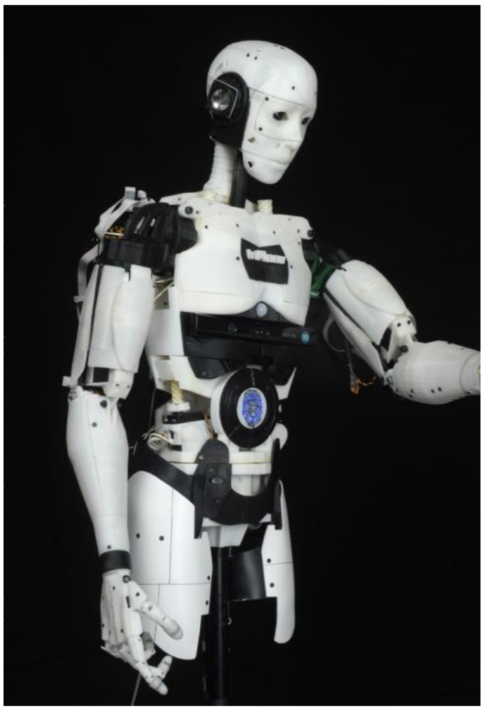


Figure 1. InMoov robot platform (Source - inmoov.blogspot.com)

This robot has articulated configuration [40], which in turn means that the connection of links is revolute type (figure 2).

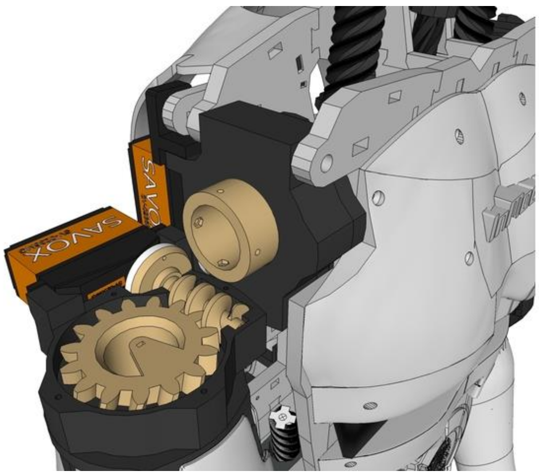


Figure 2. Robotic arm joint (Source: https://www.thingiverse.com/)

Since our main goal is to develop a control system for the hands, namely, to get the angle values which will be used as joint input in order to position the end of the arm in a certain position, the entire robot body was not assembled, but only the hands.

3.2 Kinematics calculation of robotic arm

Kinematics is the science of motion that treats the subject without regard to the forces that cause it. Within the science of kinematics, one studies the position, the velocity, the acceleration, and all higher order derivatives of the position variables (with respect to time or any other variable(s)). Hence, the study of the kinematics of manipulators refers to all the geometrical and time-based properties of the motion. Thus, its subject is the description of the spatial position of the manipulator as a function of time, and, in particular, the relationship between the space of the attached variables of the manipulator - generalized coordinates, position and orientation of the gripper. In this chapter, we consider the two main problems of the kinematics of a manipulator.

First, determine the position and orientation of the end point of manipulators grip relative to the absolute coordinate system, using known input values of manipulator’s joints, generalized coordinates and given geometrical parameters of the links. This problem is called **Forward Kinematics**.

Second, with known geometric parameters of the links, find all possible joint input values of the manipulator, which ensure the position and orientation of the gripper with respect to the absolute coordinate system. This is **Inverse Kinematics** problem.

3.2.1 Forward Kinematics

Mechanical manipulator consists of links connected by joints. There are basically 6 types of joints, but in general only two of them are mostly used in manipulator construction: revolute joints and prismatic joints. Each pair consisting of a link and a joint provides one degree of freedom. Consequently, a manipulator with **N** degrees of freedom contains **N** pairs of link hinge, and link 0 is connected to the base where the inertial coordinate system of this dynamic system is usually placed, and the last link is equipped with a working tool (figure 3). Links and joints are numbered in ascending order from the stand to the gripper of the manipulator; Thus, the junction 1 is considered the junction point of the link 1 and the support. Each link is connected to no more than two others so that no closed chains are formed.

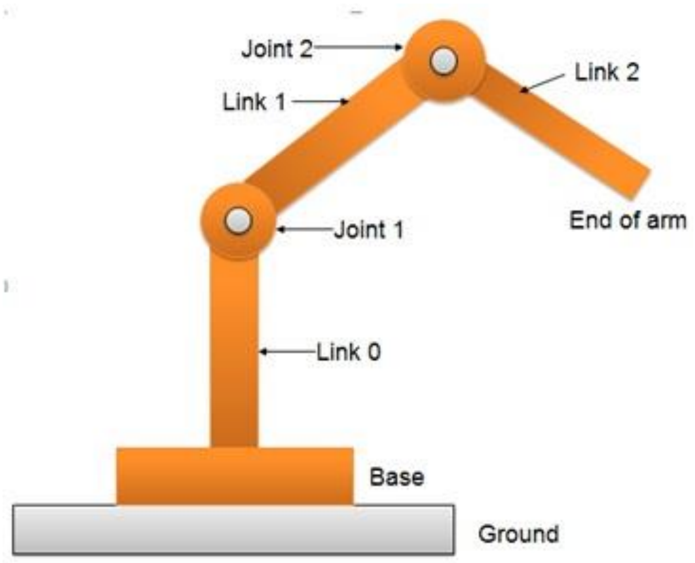


Figure 3. Joint-link scheme for robot manipulator (Source: https://nptel.ac.in/courses/)

In case of our robotic arm there are 5 DoF on arm and additional 5 DoF on hand.

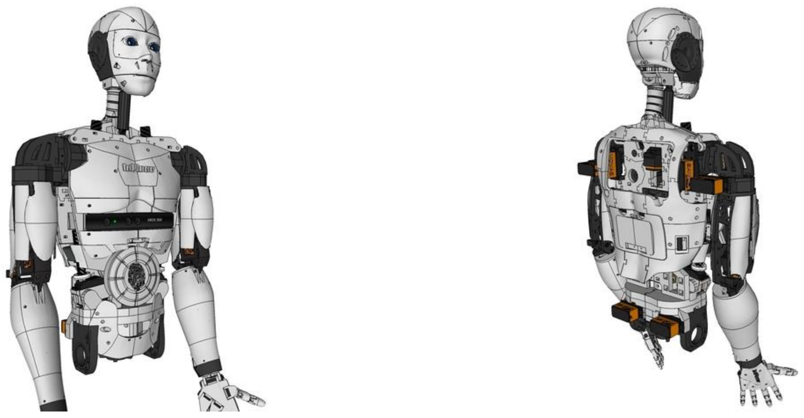


Figure 4. Schematic view of Humanoid robot (Source: https://www.thingiverse.com/)

In order to carry out kinematic calculations of manipulator each joint in the serial link should have individual link frame assignment (Figure 5). These frames will be used to obtain parameters for Denavit-Hartenberg technique [40]. This technique has become standard way of representing robot and modelling their motions. The Denavit-Hartenberg (D-H) model of representation is very simple way of modelling robot link and joints and can be used for any robot configuration, regardless its sequence or complexity (Saeed, 2011).

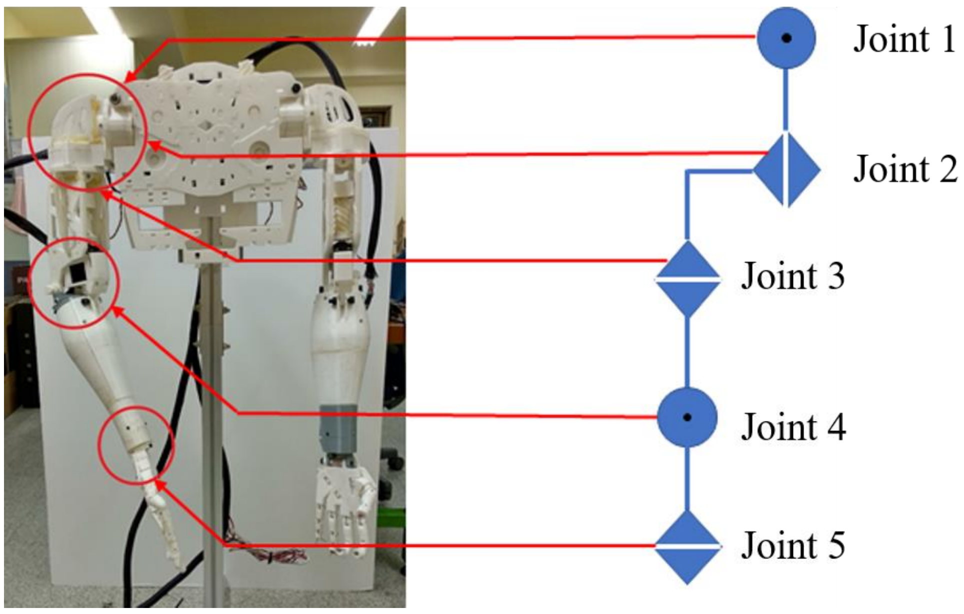


Figure 5. Robotic arm joint configuration

As it can be observed from figure 5 robot arm has 3 degrees of freedom are located together and intersect, but not in same place as it usually dose in case of other manipulators.

Based on robotic arm joint configuration link frame assignment started from first shoulder joint, which is first joint from the top. Z axis is placed on rotation axis and X is perpendicular to it, Y axis is assigned using right-hand rule (figure 6).

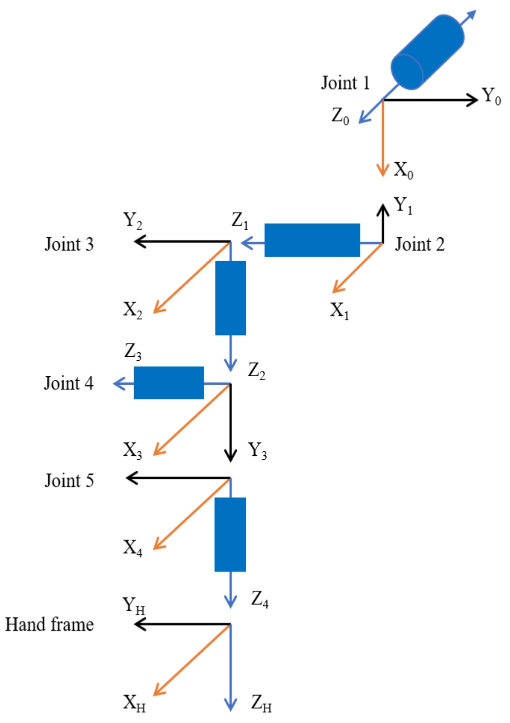


Figure 6. Link frame assignment of right arm

Since, configuration of right arm and left arm is same, there is no need to process link frame assignment again for left arm. Same link-frames can be used for left arm as well, however with slight changes in DH parameters (table 1). For example, parameter di, which is link offset, that is measured along axis z1 will be negative in case of left arm (figure 7, b). Of course, in accordance with this, kinematic calculations will change a little, but in general it is very convenient, since you can use the same method for both hands.

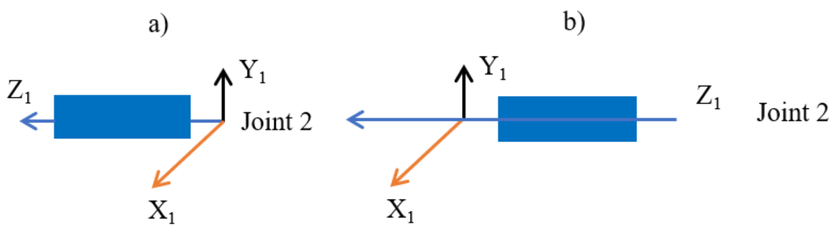


Figure 7. Joint 2 link frame. a) Right arm, b) Left arm

There are 4 parameters should be obtained: a_i, α_i, d_i and Θ:

a_i - is link length, the distance from Z_i to Z_{i+1} measured along X_i;

α_i - is link twist, the angle from Z_i to Z_{i+1} measured about X_i;

d_i - is link offset, the distance from X_{i-1} to X_i measured along Z_i;

Θ_i - is joint angle, the angle from X_{i-1} to X_i measured about Z_i;

According conventions mentioned above following DH parameters were obtained for right arm.

Table 1. Denavit-Hartenberg parameter of right arm

Joint number	a_i, (m)	α_i, (rad)	d_i, (m)	θ_i, (deg)
1	0.044	$\pi/2$	0	-5+θ_1
2	0	$\pi/2$	0.086	90+θ_2
3	0	- $\pi/2$	0.283	0
4	0	$\pi/2$	0	30+θ_4
5	0	0	0.31	0

As you can see from table θi parameters for joints 1, 2 and 4 has are not zero. This corresponds to their initial positions. Using DH parameters and general formula (1) for obtaining transformation A matrix below transformation for each consecutive frame were obtained (2 - 6).

$$A = \begin{bmatrix} \cos\theta_i & -\sin\theta_i * \cos\alpha_i & \sin\theta_i * \sin\alpha_i & a_i * \cos\theta_i \\ \sin\theta_i & \cos\theta_i * \cos\alpha_i & -\cos\theta_i * \sin\alpha_i & ai * \sin\alpha_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (1)$$

Individual joint transformation matrices.

$$A_{01} = \begin{bmatrix} c_1 & 0 & -s_1 & a_1 * c_1 \\ s_1 & 0 & c_1 & a_1 * s_1 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2)$$

$$A_{12} = \begin{bmatrix} c_2 & 0 & s_2 & 0 \\ s_2 & 0 & -c_2 & 0 \\ 0 & 1 & 0 & d_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3)$$

$$A_{23} = \begin{bmatrix} c_3 & 0 & -s_3 & 0 \\ s_3 & 0 & c_3 & 0 \\ 0 & -1 & 0 & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (4)$$

$$A_{34} = \begin{bmatrix} c_4 & 0 & s_4 & 0 \\ s_4 & 0 & -c_4 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (5)$$

$$A_{45} = \begin{bmatrix} c_5 & s_5 & 0 & 0 \\ s_5 & -c_5 & 0 & 0 \\ 0 & 0 & 1 & d_5 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad (6)$$

where c_i and s_i are $\cos\theta_i$ and $\sin\theta_i$ respectively.

By multiplying transformation of each link-frame final transformation matrix can be found. However, this matrix is so big, that is difficult to show it here. Interested reader should refer to Appendix A at the end of this paper.

$$A = A_{01} * A_{12} * A_{23} * A_{34} * A_{45} \quad (7)$$

Transformation matrix contains information about tool frame (in our case robotic arm hand) orientation, upper left 3 by 3 matrix and position upper right 3 by 1 vector (8).

$$T = \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (8)$$

For robotic application it is not convenient to use matrix multiplication due to unnecessary and time-consuming operations of multiplication by zero. Therefore, formula for each element of transformation matrix can be found using symbolic operator multiplication in MATLAB. As you will see this is simpler way of forward kinematics calculation, which excludes multiplication by zero.

$r_{11} = c_5 * (c_4 * (s_1 * s_3 + c_1 * c_2 * c_3) - c_1 * s_2 * s_4) + s_5 * (c_3 * s_1 - c_1 * c_2 * s_3);$

$r_{12} = c_5 * (c_3 * s_1 - c_1 * c_2 * s_3) - s_5 * (c_4 * (s_1 * s_3 + c_1 * c_2 * c_3) - c_1 * s_2 * c_4);$

$r_{13} = s_4 * (s_1 * s_3 + c_1 * c_2 * c_3) + c_1 * c_4 * s_2;$

$r_{21} = -c_5 * (c_4 * (c_1 * s_3 - c_2 * c_3 * s_1) + s_1 * s_2 * s_4) - s_5 * (c_1 * c_3 + c_2 * s_1 * s_3);$

$r_{22} = s_5 * (c_4 * (c_1 * s_3 - c_2 * c_3 * s_1) + s_1 * s_2 * s_4) - c_5 * (c_1 * c_3 + c_2 * s_1 * s_3);$

$r_{23} = c_4 * s_1 * s_2 - s_4 * (c_1 * s_3 - c_2 * c_3 * s_1);$

$r_{31} = c_5 * (c_2 * s_4 + c_3 * c_4 * s_2) - s_2 * s_3 * s_5;$

$r_{32} = -s_5 * (c_2 * s_4 + c_3 * c_4 * s_2) - c_5 * s_2 * s_3;$

$r_{33} = c_3 * s_2 * s_4 - c_2 * c_4;$

$p_x = a_1 * c_1 + d_2 * s_1 + d_5 * (s_4 * (s_1 * s_3 + c_1 * c_2 * c_3) + c_1 * c_4 * s_2) + c_1 * d_3 * s_2;$

$p_y = a_1 * s_1 - c_1 * d_2 - d_5 * (s_4 * (c_1 * s_3 - c_2 * c_3 * s_1) - c_4 * s_1 * s_2) + d_3 * s_1 * s_2;$

$p_z = -c_2 * d_3 - d_5 * (c_2 * c_4 - c_3 * s_2 * s_4);$

However, each formula should be checked to make sure that the calculations are correct. There is very simple way of checking forward kinematics. First, need to take initial conditions (joint angles and link length) of robotic arm, put them as input to the formula shown above and multiply them. The output should result in full length of robotic arm.

```
position = [px py pz]';
```

However, there is slight change of input values. The movement range of joint 4 is from 30 to 90 degrees, but in order to be able to check our calculation input value was taken as 0. This gives us arm length in full stretch.

The input values are following:

```
theta1 = 0;
theta2 = 90;
theta3 = 0;
theta4 = 0;
theta5 = 0;
```

Simple multiplication in MATLAB gives us following result.

```
position =

    0.6370
   -0.0860
   -0.0000
```

Based on that 0.044 + 0.283 + 0.31 is equal to 0.6370 which is arm length pointing down and its position on x axis it is clear that calculations are right. However, there is another way to check if calculations are correct. The input value of joint 4 is taken as 90 degrees, so as a result we must get the arm bent at the elbow.

```
position =

    0.3270
   -0.0860
    0.3100
```

Based on that 0.044 + 0.283 is equal to 0.3270 which is length of arm from shoulder to elbow and 0.31 is length from elbow to the point on the hand it is clear that forward kinematics are correct.

3.2.2 Inverse Kinematics

Forward Kinematics is the problem of describing robotic arm in cartesian space with known joint input values i.e. converting from joint space to cartesian space. However, in order to be able to position the arm in certain position inverse kinematics problem must be solved. Inverse Kinematics is the problem of finding joint angles with known end position and orientation of robotic arm tool frame i.e. hand relative to the base.

Minimum 6 DoF are needed for full orientation and position of manipulator end effector in 3D space, however, 4 and 5 DoF robotic arms are useful in most industrial application, hence need inverse kinematics solution as well. As it mentioned earlier the robotic arm taken as the object of study in this paper has 5 DoF. According to (R. Manseur and K. L. Doty, 1992) unless the robotic arm has a structural characteristic that allows closed-form solutions such as three intersecting joint-axes (Pieper, 1968) or three parallel joint axes, solving the inverse kinematics problem of robots with more than 4 DoF requires the use of iterative techniques [15][18]. However, 4 DoF robot arms of any complexity is can always be solved by closed form solution. There are many researches conducted to study inverse kinematics problem of 5 DoF robotic arms. For example, (R. Manseur and K. L. Doty, 1992) solved inverse kinematics using closed form solution [41] of 4 DoF robotic arm and applied obtained method to 5 and 6 DoF manipulators. The disadvantage of this method is that the angle of first joint in link chain is found by initial guess which makes it very non-convenient for real world application, since it is impossible to develop manipulator for real task application. (Yang Si et al, 2013) also used closed form solution to find inverse kinematics of 4 DoF manipulator. Masayuki Shimizu [50] used analytical method for solving inverse kinematics of 5 DoF manipulator.

In addition to these many researches has been conducted to solve inverse kinematics problem of 5 and 6 DoF robotic arms. The main problem of applying those methods to robotic arm used in our study is joint configuration difference. Most of the studied humanoid manipulators have 3 shoulder joint axes intersecting at one point which is different comparing to one we have. In case of our manipulator shoulder joints do not intersect at one point and specification is different, which makes it quite difficult to find suitable method for solving inverse kinematics problem.

Considering all of the above we decided to use Damped Least-Squares method to solve inverse kinematics of 5 DoF humanoid robot arm. This method of solving the inverse kinematics of the manipulator involves finding the Jacobian matrix. According to (C. W. Wampler, 1986) previous formulations based on Jacobian matrix, are inefficient and fail near kinematic singularities. To avoid this problem the inverse kinematics was reformulated as a damped least-squares method, which reduces error. This method was first proposed by [42] and [57]. This method involves reducing accuracy for feasibility.

3.2.2.1 Damped Least-Square method Development

According to (Buss, 2004), if there are k end-effectors, their positions are denoted by $s_1, \ldots, s_k$. Every end-effector position is the function of joint angles. The manipulator is controlled by specifying target positions of end-effectors. Target positions given by a vector $\vec{t} = (t_1, \ldots t_k)^T$, where t_i is the target position for the ith end-effector. The joint angles a given by column vector as $\theta = (\theta_1, \ldots, \theta_n)^T$. The inverse kinematics problem is to find values for the θ, so that:

$$t_i = s_i(\theta), \text{ for all } i. \qquad (9)$$

However, there are may not always be a solution to the equation above, nor may not be a unique solution. Also, there are may not be a closed form solution as well.

Therefore, using iterative method to approximate a good solution can be the way of calculating the inverse kinematics (IK). For this purpose, the s_i functions are linearly approximated using the Jacobian matrix.

$$J(\theta) = \left(\frac{\partial s_i}{\partial \theta_i}\right)_{i,j}$$

The Jacobian implies the use of an iterative method to solve the equation (9). Assuming that we have current values for $\theta, \vec{s}$ and $\vec{t}$, the Jacobian can be calculated: $J = J(\theta)$. After that we can seek an update values for $\Delta\theta$, for the purpose of incrementing the joint angles θ by $\Delta\theta$.

$$\theta := \theta + \Delta\theta$$

The idea is that the value of $\Delta\theta$ should be chosen so that $\Delta\vec{s}$ is approximately equal to $\vec{e}$. Where:

$$\vec{e} = \vec{t} - \vec{s}$$

Although it is also common to choose $\Delta\theta$ so that the approximate movement $\Delta\vec{s}$ in the end-effectors (partially) matches the velocities of the target positions.

The change in end-effector positions caused by this change in joint angles can be estimated as:

$$\Delta\vec{s} \approx J\Delta\theta$$

There are basically two methods of calculating the Jacobian matrix. In our case we use an Alternate method.

$$J(\theta) = \left(\frac{\partial t_i}{\partial \theta_j}\right)_{i,j}$$

where the meaning of $\partial t_i / \partial \theta_j$ is that the target position is thought of as being attached to the same link as the *i*th end-effector. The intuition is that with this formulation of the Jacobian, we are trying to move the target positions towards the end-effector, rather than the end-effector towards the target positions (Buss, 2009).

So, the Damped Least-Squares method avoids many of the pseudoinverse method's problems with singularities (Buss, 2009) and can give a numerically stable method of selecting $\Delta\theta$. Rather than just finding the minimum vector $\Delta\theta$ that gives a best solution to equation:

$$\vec{e} = J\Delta\theta \qquad (10)$$

it needs the value of $\Delta\theta$ that minimizes the quantity:

$$\|J\Delta\theta - \vec{e}\|^2 + \lambda^2\|\Delta\theta\|^2$$

where, λ is non-zero damping constant. This is equivalent to minimizing the quantity:

$$\left\| \begin{pmatrix} J \\ \lambda I \end{pmatrix} \Delta\theta - \begin{pmatrix} \vec{e} \\ 0 \end{pmatrix} \right\|$$

The corresponding normal equation is:

$$\begin{pmatrix} J \\ \lambda I \end{pmatrix}^T \begin{pmatrix} J \\ \lambda I \end{pmatrix} \Delta\theta = \begin{pmatrix} J \\ \lambda I \end{pmatrix}^T \begin{pmatrix} \vec{e} \\ 0 \end{pmatrix}$$

This can be equivalently rewritten as:

$$(J^T J + \lambda^2 I)\Delta\theta = J^T \vec{e}$$

It can be shown that $J^T J + \lambda^2 I$ is non-singular. Thus, the damped least squares solution is equal to:

$$\Delta\theta = (J^T J + \lambda^2 I)^{-1} J^T \vec{e} \qquad (11)$$

Now $J^T J$ is an n x n matrix, where n is the number of degrees of freedom. It is easy to show that $(J^T J + \lambda^2 I)^{-1} J^T = J^T (J^T J + \lambda^2 I)^{-1}$. Thus,

$$\Delta\theta = J^T (J J^T + \lambda^2 I)^{-1} \vec{e} \qquad (12)$$

where, J is Jacobian matrix, λ is non-zero damping constant, e is the position error.

3.2.2.2 Inverse Kinematics calculation using Damped Lease-Square method

The main formula for Damped Least-Square method is as following:

$$\Delta\theta = J^T (J J^T + \lambda^2 I)^{-1} \vec{e}$$

This is iterative method for solving inverse kinematics of manipulator.

As it can be seen from formula the only problem is to find the Jacobian matrix. After the Jacobian matrix was found the only thing is to form iteration to find the solution. The way to find Jacobian matrix and end-effector position is analyzing homogeneous transformation matrix of robot arm.

Calculation steps:

1. Obtaining transform matrix
2. Obtaining formula for target position vector
3. Calculating an alternate Jacobian
4. Defining merge of error, number of iteration and non-zero damping constant

In order to obtain final transform matrix all, transform matrices of each joint should be multiplied. Multiplication result gives as matrix A with position and orientation information as shown in formula (1).

After we have transformation matrix A, we need to multiply it by vector $[0 \quad 0 \quad 0 \quad 1]^T$ in order to obtain position vector of robot arm. That will be used in a loop to check correctness of inverse kinematics calculation. So:

$$P_{new} = A * \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \qquad (13)$$

In order to find an alternate Jacobian, derivative of each element of position vector P_{new} should be calculated.

$$J(\theta) = \left[\frac{\partial P_{newi}}{\partial \theta_i}\right] \qquad (14)$$

Since, this method is numerical, all calculation is conducted in MATLAB. Interested reader should refer to the Appendix A.

After we have, Jacobian matrix, we need to define merge of error. As it was said earlier the Damped Least-Square method implies small inaccuracy for feasibility. Therefore, next step is to define merge of error and number of iterations.

$$esp = [0.05\ 0.05\ 0.05]$$

$$errT = [0\ 0\ 0]^T$$

$$i = 1000$$

As you can see above the merge of error is shown by variable esp. The variable $errT$ is needed for a loop calculation in MATLAB. Finally, the number of iterations is taken to be 1000. Of course, the merge of error can be changed in order to get more accurate results but in that case, number of iterations should be changed as well. In our case 1000 iterations is enough, since it’s suitable for real world application.

When we have all these values, the loop for calculation is started. First position input is given to the system by vector $P_d = [Px\ Py\ Pz]^T$. This vector shows desired position for robot hand that should be reached after inverse kinematics calculation and used to check the accuracy of calculation. When loop is started, first, system calculates formula (12). After, elements of vector $\Delta\theta$ that has values of each angle is used to calculate vector P_{new}. Then the accuracy is checked by:

$$q_{new} = P_{new} - P_d \qquad (15)$$

If the values of the variable q_{new} in formula (15) is smaller than merge of error, loop is terminated and obtained values of $\Delta\theta$ is saved.

3.2.2.3 Inverse Kinematics analysis

To check the correctness of the calculations of inverse kinematics, several analyzes were performed. First, several connections were selected. Further, by assigning data within the joint ranges of movement, forward kinematics calculations were carried out. The obtained data were used to calculate inverse kinematics. The results of these calculations were compared with the original data.

In the second analysis, data of the positions and orientations of the hand were taken as the basis. The principle of this analysis is similar to the first. To begin with, forward kinematic calculations with arbitrary data were carried out. The resulting data of these calculations were used in the calculations of inverse kinematics. And the results were used for repeated calculations of forward kinematics. The results of the latest data were used to compare with the original data. Thus, a complete analysis of inverse kinematics was carried out.

In order to conduct first analysis of inverse kinematics calculation, need to analyze angles of several joints. For this purpose, first 2 joints of robot arm are chosen. Continuous angle input will be given separately to each joint and forward kinematics calculation output will be obtained. This output later will be used in inverse kinematics calculation. After that inputs of forward kinematics and inverse kinematics will be compared. However, it is important to mention that in out inverse kinematics calculation we did not included last joint, since it’s only effects on orientation of robot hand. First, we will start from joint 1. In table 1 you can see input and output, as well as difference between them.

Table 2. Joint 1 input and output data

Sample time	theta4 input for F.K. calculation, deg	theta4 output from I.K. calculation, deg	Difference, deg
1.	-10	-9.6502	0.3498
2.	-20	-18.9710	1.029
3.	-30	-28.2230	1.777
4.	-40	-37.3987	2.6013
5.	-50	-46.4738	3.5262
6.	-60	-55.4151	4.5849

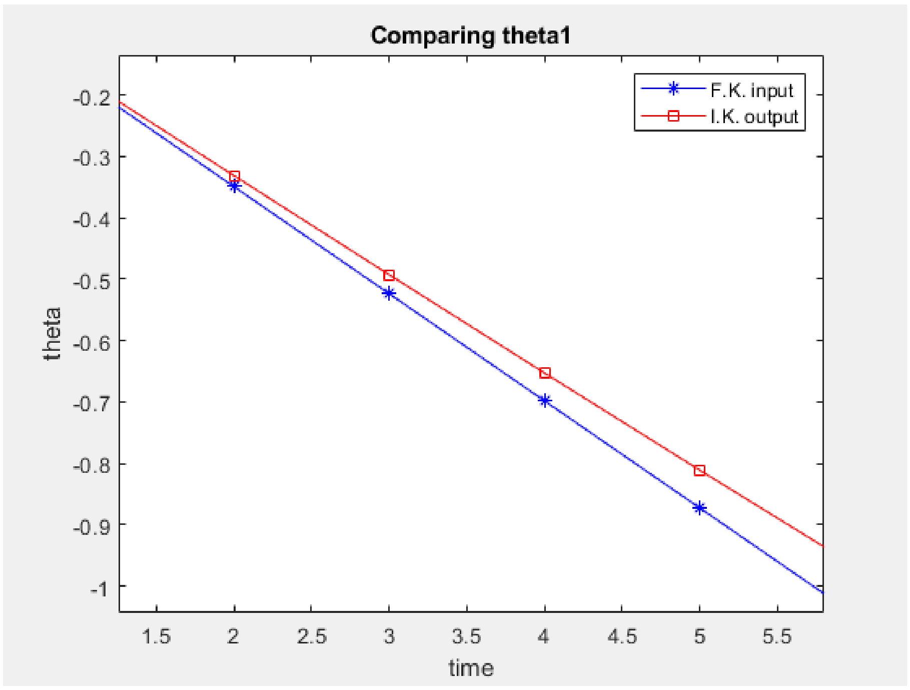


Figure 8. Comparing theta 1

As you can see there is a slight change between original data and data obtained during inverse kinematics calculation. However, since the calculation of the inverse kinematics is carried out considering the error in the positional data, minor differences in the data are permissible. Especially considering the features of chosen method. The reasons for the admissibility of such minor differences will be explained in the following analyzes.

Table 3. Joint 2 input and output data

Sample time	theta4 input for F.K. calculation, deg	theta4 output from I.K. calculation, deg	Difference, deg
1.	60	60.9773	-0.9773
2.	70	70.9287	-0.9287
3.	80	80.8418	-0.8418
4.	90	90.6973	-0.6973
5.	100	100.4921	-0.4921
6.	110	110.0958	-0.0958

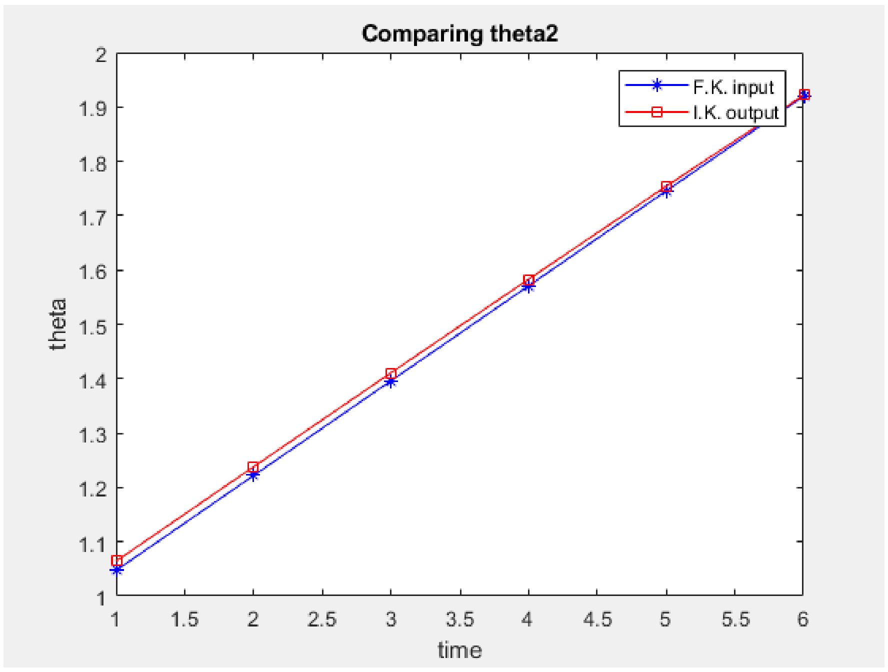


Figure 9. Comparing theta 2

As you can see from inverse kinematics analysis data there some small errors in calculations. It is important to mention that Damped Least-Squares method reducing accuracy for feasibility. But, since calculation of forward kinematics with inverse kinematics output gives us quite accurate results, it is assumed to be correct. Also, it is accurate enough for control system proposed in this paper. However, in the future work inverse kinematics calculation with orientation will be conducted. It can give ability to project more accurate and difficult control systems for various applications.

For the position and orientation analysis of inverse kinematics five calculations with arbitrary data were conducted. This is one of the common ways to conduct calculation analysis. First, performing the calculation of forward kinematics with different angles and getting the position coordinates of the end-effector of the robotic arm (table 4). Before starting calculations, it is also necessary to determine the joint limits, hence joint angle range in which each joint can move (see table 5). In this table you also can see name of each joint of robotic arm given by its functional characteristic, starting from first joint in robotic arm shoulder all the way to the wrist.

Table 4. Joint angle range

Joint, num	Joint name	Max, (deg)	Min, (deg)
Joint 1	Shoulder abduction-adduction	-5	-60
Joint 2	Shoulder flexion-extension	135	25
Joint 3	Upper arm rotation	60	-40
Joint 4	Elbow	90	30
Joint 5	Wrist pronation-supination	-45	45

Table 5. Input and output parameters for Forward Kinematics calculation

Input, (deg)						Output, m		
Number	Joint 1	Joint 2	Joint 3	Joint 4	Joint 5	Px	Py	Pz
1	-5	135	0	90	0	0.0173	-0.0878	0.4193
2	-5	135	60	90	0	0.1031	-0.3648	0.3097
3	-5	135	-40	90	0	0.0858	0.1062	0.3680
4	-5	25	0	30	0	0.4085	-0.1221	-0.4343
5	-60	110	60	30	0	0.0771	-0.5741	0.2614

Since joint five, i.e. wrist pronation-supination only affects hand orientation it hasn’t given any value during this calculation except zero, which corresponds to the direct position of the hand in space.

Now using obtained position coordinates conduct inverse kinematics in order to compare resulting angle values for each joint with input of forward kinematics calculation (see table 6).

Table 6. Inverse Kinematics calculation

Input, (m)				Output, (deg)				
Num	Px	Py	Pz	Joint 1	Joint 2	Joint 3	Joint 4	Joint 5
1	0.0173	-0.0878	0.4193	-16.7296	132.6945	-1.2758	90.0000	0
2	0.1031	-0.3648	0.3097	-49.6453	91.7832	13.4260	84.9380	0
3	0.0858	0.1062	0.3680	-5.0000	134.9956	-40.0000	90.0000	0
4	0.4085	-0.1221	-0.4343	-5.5070	25.0645	-1.3776	30.0000	0
5	0.0771	-0.5741	0.2614	-60.0000	102.9889	42.1772	37.8551	0

As you can see in table 6 calculation results 3 to 5 is close to the input data of the first calculations, especially number 3 is quite accurate, but number 1 and 2 is strikingly different from the input data of the forward kinematics calculation. However, the data obtained do not go beyond certain limits of joint range and considering the fact that inverse kinematics can have many possible solutions for one curtain problem one can assume that results is correct. However, it is always better show calculation data in a graphical view. From figure 10 to 12 you can check analysis of data of each coordinate axes.

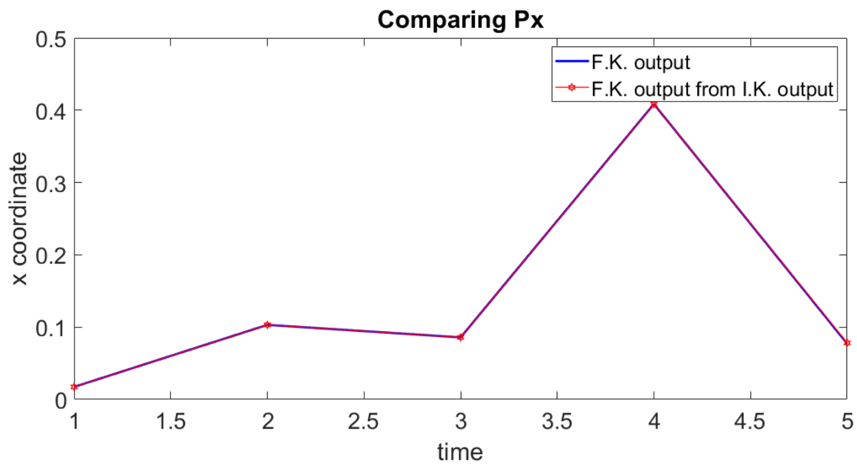


Figure 10. Comparing coordinates on x-axis

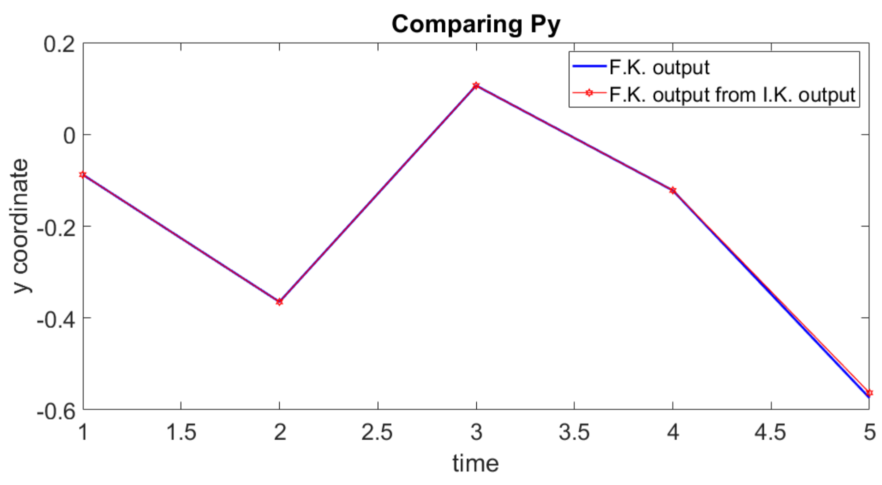


Figure 11. Comparing coordinates on y-axis

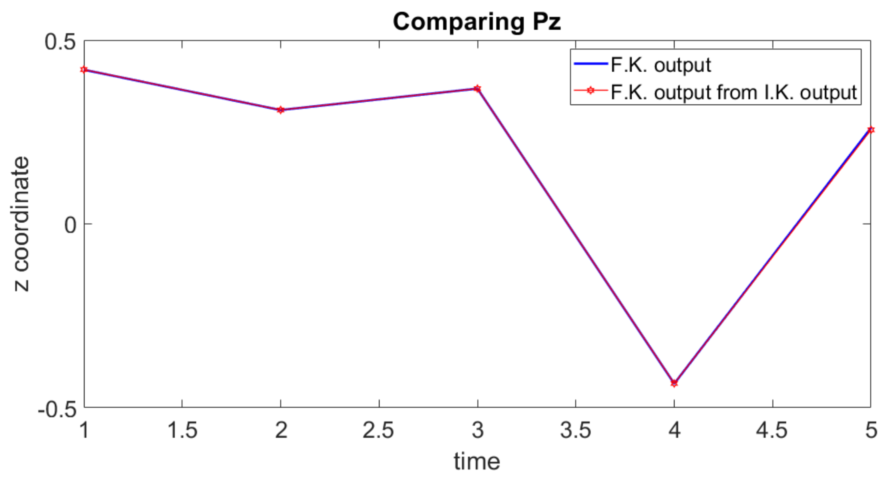


Figure 12. Comparing coordinates on z-axis

Judging by the results of graphical analysis, the results of inverse kinematics are rather accurate. Based on this, it is safe to conclude that inverse kinematics calculations are correct.

The fact that the orientation of manipulator end-effector can be shown by fewer numbers, rather than nine numbers, is a well-known fact. Since our robot arm doesn't have spherical joint, Euler angles can be used in order to conduct orientation analysis of inverse kinematics calculations. Thus, angle-axis representation was chosen for this purpose.

In order to conduct orientation analysis, first we take information from homogeneous transformation matrix and calculate angle-axis representation of orientation of manipulators end-effector.

Chapter 4. Development of Hardware and Software of Robotic arm

In this chapter we will develop the hardware part of the robot. The robot has two main parts - mechanical and electronic. The mechanical design of the robot contains a manipulator. The electronic design of the robot contains a computer, a microcontroller and peripheral electronic devices.

4.1 Control unit Hardware

The manipulator has 5 links and each link is controlled by a servo drive. Computer is used to calculate inverse kinematics and trajectory generation for robot arm microcontroller, which in this particular case is Arduino board, that controls all servo motors through PWM servo driver device. Computer and microcontroller are connected using I2C communication [16]. As it mentioned earlier computer is used to process kinematics calculation for robot arm and to send angular data to Arduino board, since Arduino IDE is not capable for high level kinematics calculation. MATLAB and Simulink are main software used to develop control unit for robotic arm. MATLAB provides software package for Arduino and it allows to control Arduino directly through MATLAB or Simulink.

4.1.1 Microcontrollers and peripheral electronic devices

The Arduino Mega 2560 - is a microcontroller board (figure 20) based on the ATmega2560 (see table 6). It has 54 digital input/output pins (of which 15 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery. The Mega 2560 board is compatible with most shields designed for the Uno and the former boards Duemilanove or Diecimila. In our case two Arduino Mega 2560 boards is used in control unit. First one is as “Master Read” device and second is as “Slave Write” device.

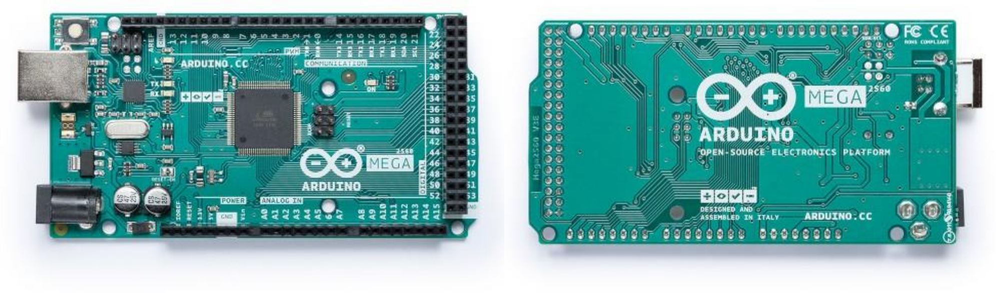


Figure 13. Arduino Mega 2560 microcontroller (source: www.arduino.cc)

Table 7 contains summary of information about specification of Arduino Mega 2560 board.

Table 7. Specifications summary

Microcontroller type	ATmega2560
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	54 (of which 14 provide PWM output)
Analog Input Pins	16
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	256 KB of which 8 KB used by bootloader
SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz

Driving servo motors with Arduino Servo library is easy, but each servo requires pin and Arduino processing power. Due to this reason we cannot use Arduino board to control lot of servos. The Adafruit 16-Channel 12-bit PWM/Servo Driver [2] allows to drive up to 16 servos over I2C connection with only 2 pins. The on-board PWM controller will drive all 16 channels simultaneously with no additional Arduino processing overhead. In addition to this there is a possibility for connection for up to 62 servo drivers to control up to 992 servos with same board and same 2 pins. Therefore, two Adafruit 16-Channel 12-bit PWM/Servo Drivers, one for each arm, were used to control robot arms (figure 21).

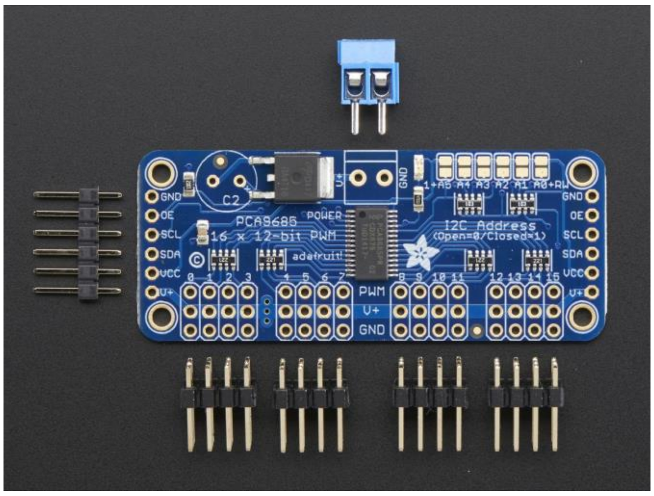


Figure 14. Adafruit 16-Channel 12-bit PWM/Servo Driver (source: www.learn.adafruit.com)

The Adafruit servo driver is controlled via the I2C bus. This board has two groups of connectors for the I2C bus on both sides, which allows to connect several boards in series to the bus or connect other devices to the I2C bus.

The power supply of the controller and the outputs of the PWM channels is divided and can be from 3 to 5 volts. For PWM channels, a maximum voltage of 6 volts is allowed. Power for the PWM channels can be supplied to the pins (V+) or through the terminal. There is a place on the board for a filtering capacitor (see table 8).

Table 8. I/O ports of Adafruit 16-Channel 12-bit PWM/Servo Driver

Size	54 x 70 x 3 mm
Weight	9 g
GND	ground
OE	switching on outputs, active level low
SCL	I2C interface clock signal
SDA	I2C interface data
VCC	module power supply
V+	load power supply

There are two types of servo motors used in robotic arm: HITEC HS-805BB for robot arm joint control and another type for hand control. More specific information can be obtained from table 8.

Table 9. Servo motor specification

Brand	HITEC
Motor type	3 pole
Bearing type	Dual Ball Bearing
Speed (4.8 V / 6.0 V)	0.19 / 0.14
Torque kg/cm (4.8 V / 6.0 V)	19.8 / 24.7

4.1.2 I2C connection between MATLAB \ Simulink and Arduino

MATLAB support package for Arduino allows I2C communication between computer and Arduino. Data is transferred by Arduino USB cable. For I2C communication two Arduino boards is used. One of them is Master board, second one is Slave board. Since computer and

MATLAB is used as master device second Arduino board is programmed as Slave write device, but we still need to use two boards. First board receives angular data from MATLAB and sends it to second board which controls servo boards.

I2C communication protocol uses two wires to transfer data between devices SDA and SCL (figure 22). There are master and slave devices. I2C connection allows to connect multiple slave devices with only one master devices controlling them.

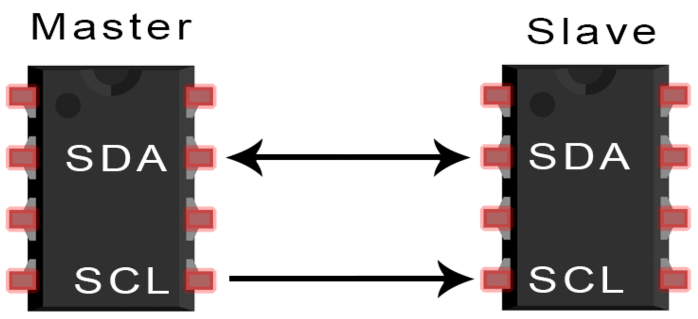


Figure 15. Introduction to I2C Single Master Single Slave (Source - www.circuitbasics.com)

SDA (Serial Data) – the line for the master and slave to send and receive data. **SCL** (Serial Clock) – the line that carries the clock signal. I2C is a serial communication protocol, so data is transferred bit by bit along a single wire (the SDA line). I2C is synchronous connection, so the output of bits is synchronized to the sampling of bits by a clock signal shared between the master and the slave. The clock signal is always controlled by the master [3]. Interested reader should refer to [3][31].

4.1.3 Driver control subsystem

Driver control subsystem has two Arduino microcontrollers and two peripheral PCA9685 servo driver devices. The PCA9685 chip controls 10 servos to move both robotic arm joint servos. Arduino controller will communicate with PCA9685 chips via I2C interface. The controller will convert angular data to PWM and send angular data over the I2C communication interface to PCA9685 and servo driver will move robotic arm.

First, need to connect two Arduino boards by I2C communication [13]. Only three wires are used for this purpose (see figure 11).

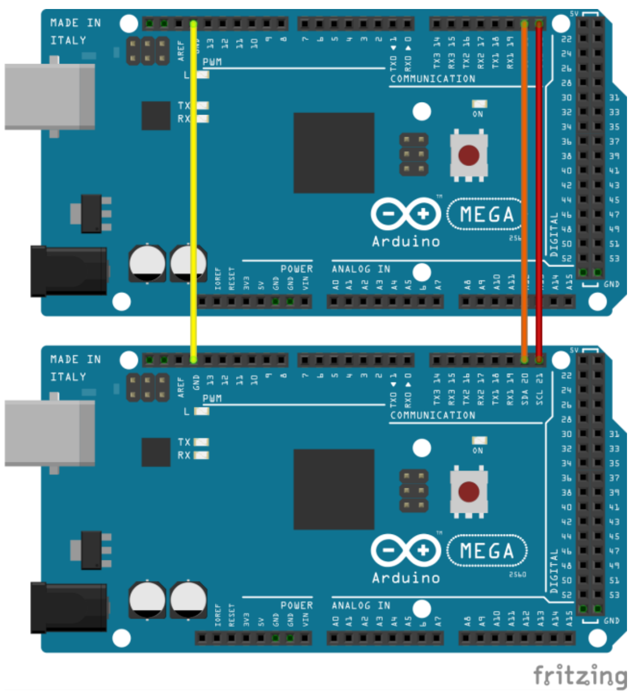


Figure 16. Arduino Mega 2560 I2C communication

The reason of choosing Arduino Mega 2560 board as a Slave device is that it has 2 I2C pins. This is important, because Servo Drive [2] boards are also connected through I2C protocol (figure 24).

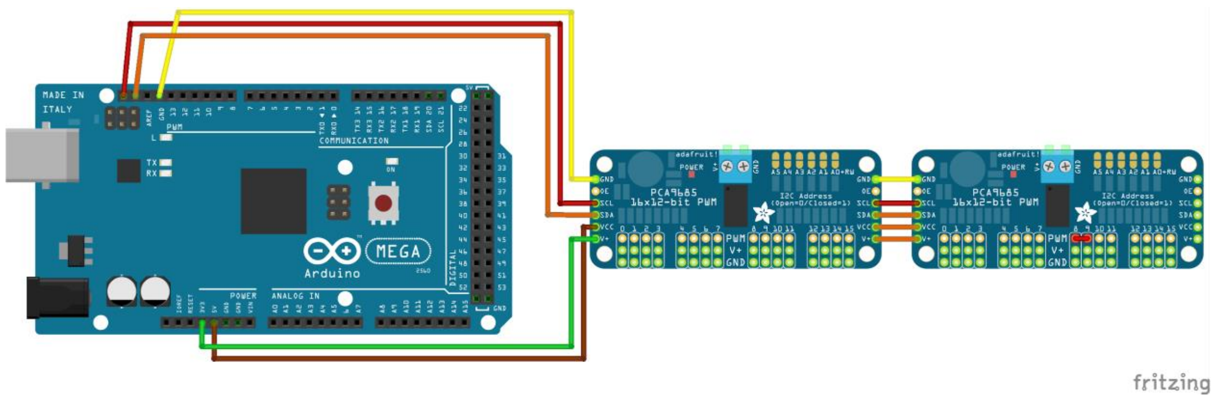


Figure 17. I2C connection Arduino Mega and Adafruit Servo Driver boards

In figure 4.5 there five wires that connect Arduino Mega board and two servo driver boards. There are three wires of I2C communication on top: yellow wire is ground, red wire SCL and orange is SDA. There are two more wires: brown cable is connected to VCC pin of servo driver board. This is logic power pin. Green cable is connecting 3.3 V pin on Arduino board and to V+ pin on servo driver board. This is an optional power pin that will supply distributed power to servos (see figure 25). Due to the fact that we use 5 servo motors in each servo controller, 3.3 V is not enough. In this example, this is done only to show that you need to add a separate power source for the servos. Additional power could be injected through the 2-pin terminal block at the top of the board.

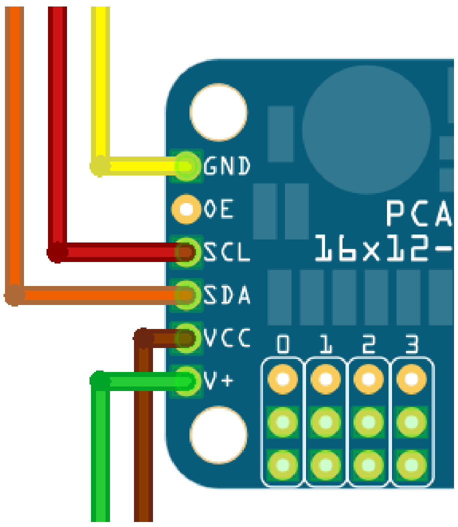


Figure 18. Close up view

It is not enough just to connect two servo controller boards. It is necessary to determine an individual address for each board. There are separate address pins on each controller for this purpose (figure 26). The first controller in the series will have a default address (0x40) and need to allocate a special address for the second board that is different from the first. It could be done simply by soldering the pin A0 on the address pin, which will correspond to the address (0x41).

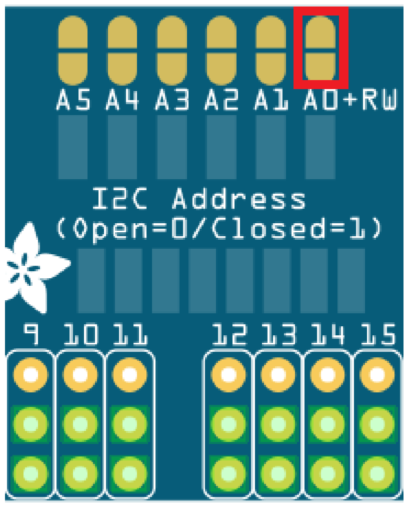


Figure 19. Adafruit PCA9685 PWM servo driver address pins

After all the preparatory work is done, servo motors can be connected to the servo drivers (figure 27).

Note that each Adafruit board will control 5 servo motors, one servo driver per robotic arm. Also note the changed color of the wires. This was done to keep the reader from getting confused. The black wire is ground, the green connects the SDA pins, the blue SCL pins, the red wire is VCC, and the orange wire connects V+ pin, the additional power pin to servos.

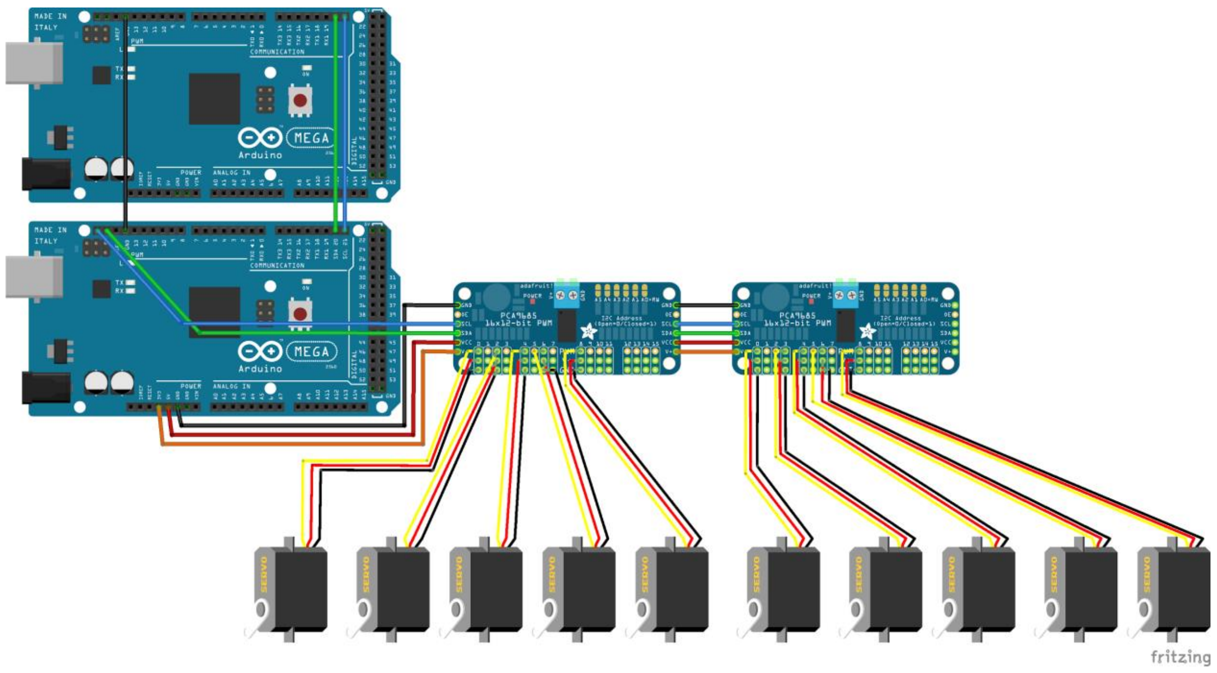


Figure 20. General view of Hardware connection

The energy supplied to the servos through the pin V+ is not enough, as was said earlier, so it’s needed an additional power source. For this purpose, a separate pin is provided in the servo controller (figure 28).

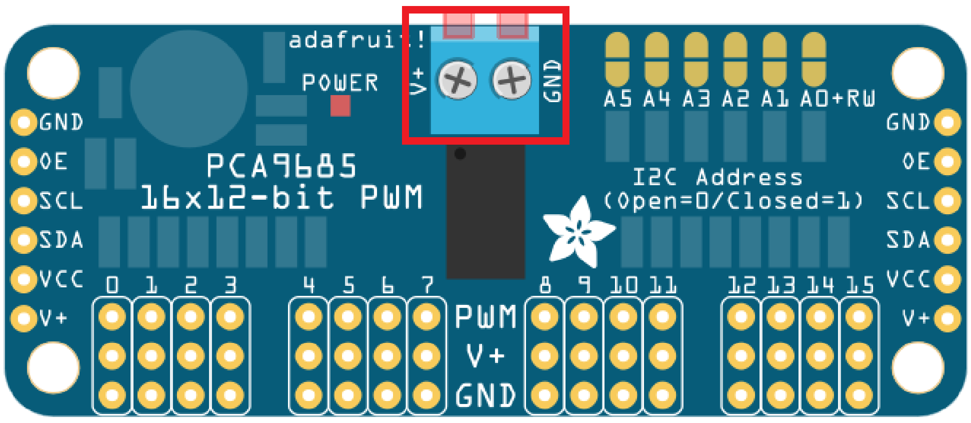


Figure 21. Additional V+ power pin

4.2 Control unit Software

The main software used is MATLAB \ Simulink with Arduino support hardware. The reason for choosing this program is simplicity and ease of use. First, MATLAB is a high-level programming language. Secondly, the system of blocks in Simulink simplifies working with mathematical functions and algorithms. Simulator also has add-ons for Arduino, which allows to create projects on Simulink using Arduino boards. Thirdly, the MATLAB is multifunctional program and no need to use additional programs.

4.2.1 Simulink model of control algorithm

Only three parts from Simulink library needed to control Slave Write board through I2C communication in Simulink (see figure 26).

1) Angular data
2) Data conversion to uint8 datatype
3) Arduino Slave Write

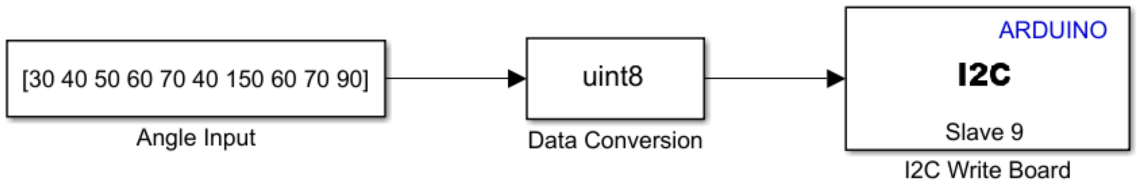
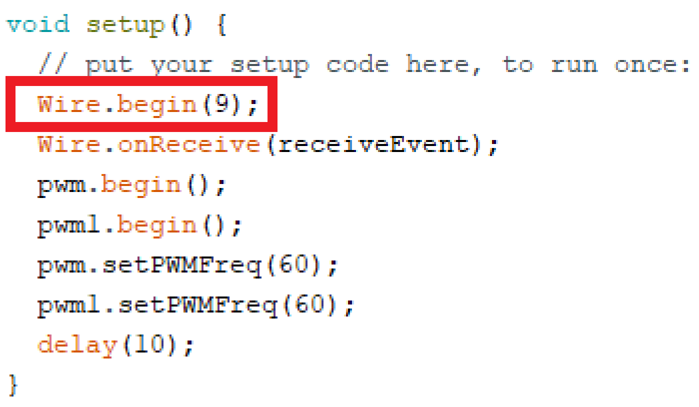


Figure 22. Simulink model of control algorithm

As you can see it is quite simple. First, we obtain angular input data for each joint and place them in form of string array or [1x10] matrix form. First five numbers is for right arm and last five numbers is for left arm of robot. Then input data is sent to Data conversion block. In this block all input values is converted to uint8 type. After, all converted data is sent to Arduino board. Pay attention to the address of the Slave Write board. Board address can be set manually, but it must match the one defined in the code for I2C communication of Slave board (figure 30). In this case the address is “9”.



```
void setup() {
  // put your setup code here, to run once:
  Wire.begin(9);
  Wire.onReceive(receiveEvent);
  pwm.begin();
  pwm1.begin();
  pwm.setPWMFreq(60);
  pwm1.setPWMFreq(60);
  delay(10);
}
```

Figure 23. Setting Slave board address

Figure 18 shows one-time setting block for the Arduino I2C code. The link marked with red block shows address setting of the board and function “Wire” means I2C communication. On the line below is the function for definition of purpose of the board, which in this particular case is receiving incoming data.

Also, below you can see the code line **pwm.setPWMFreq(60)** that sets the PWM frequency for the servo driver board. The fact is that the frequency of the driver board cannot change in accordance with the pin, all pins must have same PWM frequency. This means that you cannot use one controller for servo motors and LED at the same time, because for LEDs usually required frequency PWM 1.0 KHz, but servos need 60 Hz.

4.2.2 - Simulink model for Right arm of the Robot

Once we have decided on the basic structure of the Simulink model, to control robotic arms, we need to make a complete model with the calculation of inverse kinematics. To begin, we will discuss the model for only one hand - the right hand, since the structure of Simulink model is the same for both hands (figure 31). Then look at the full model for both hands.

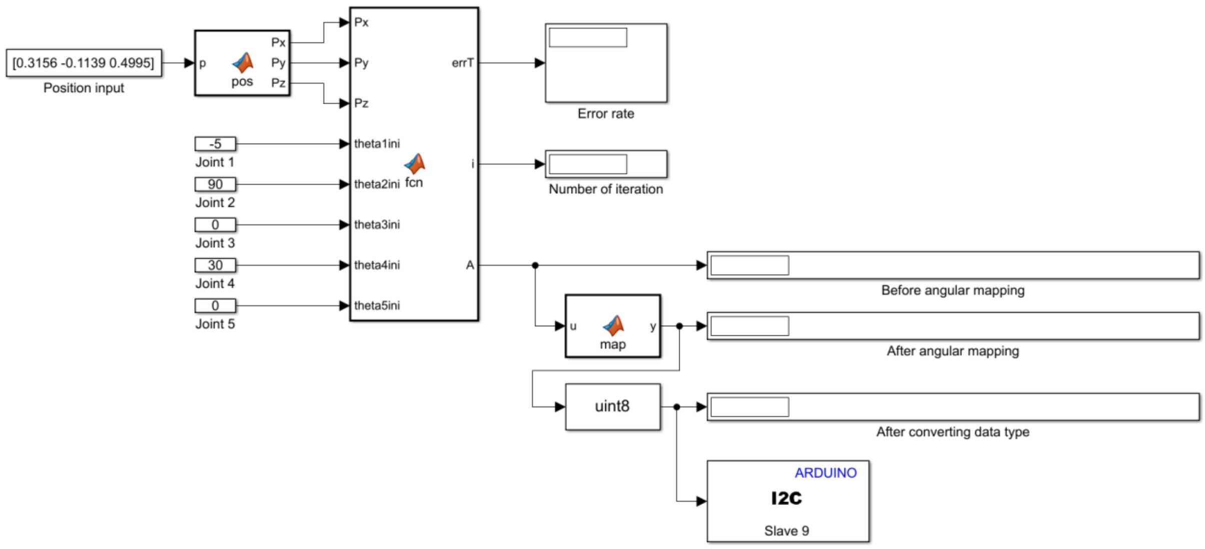


Figure 24. Full Simulink model for Right arm

As you can see, this is quite simple model at first look. Now we begin to discuss each part of this model separately.

First, important thing is inputs to the inverse kinematics block of the robotic arm, which implemented by MATLAB function (figure 32).

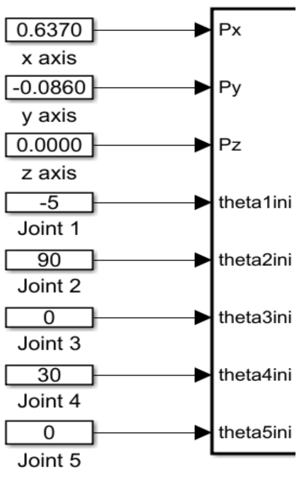


Figure 25. Input of the Simulink model

There are eight main inputs for inverse kinematics block, which are three position values of the hand of robot arm, and five initial values of each joint on robot arm. It can clearly be seen on the picture above. For inverse kinematics no need to change initial values inputs for joint angle settings, but position. Next block is MATLAB function, that calculates inverse kinematics based on position inputs. The source code for function is to massive to put here, so interested reader should refer to the Appendix files.

It is very important to track the results of the calculations, so that after the MATLAB function block, there are 2 display blocks, which allow to track error and number of iteration (figure 33).

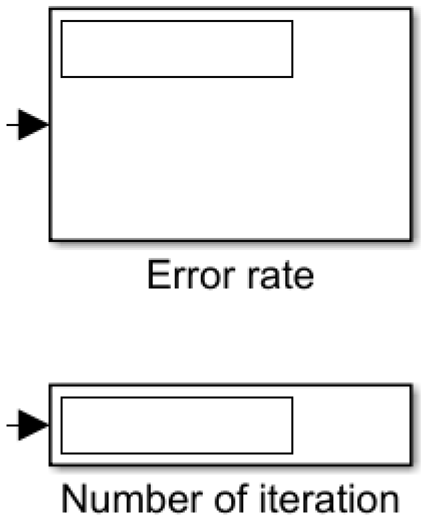


Figure 26. Error rate and number of iteration display blocks

Changes of parameters in each coordinate axis also can be tracked through graphical block, but it’s not crucial.

As mentioned earlier, the input data for the Arduino board should be in the form of a 1 by 10 matrix. Since there are two hands, need to somehow collect the data in one matrix. For this purpose, there is another MATLAB function (figure 34. The output from the block of calculations of inverse kinematics is obtained in the form of a 1 by 5 matrix. These data are sent to the input block and form a 1 by 10 matrix with data obtained from the inverse kinematics calculations of the left hand.

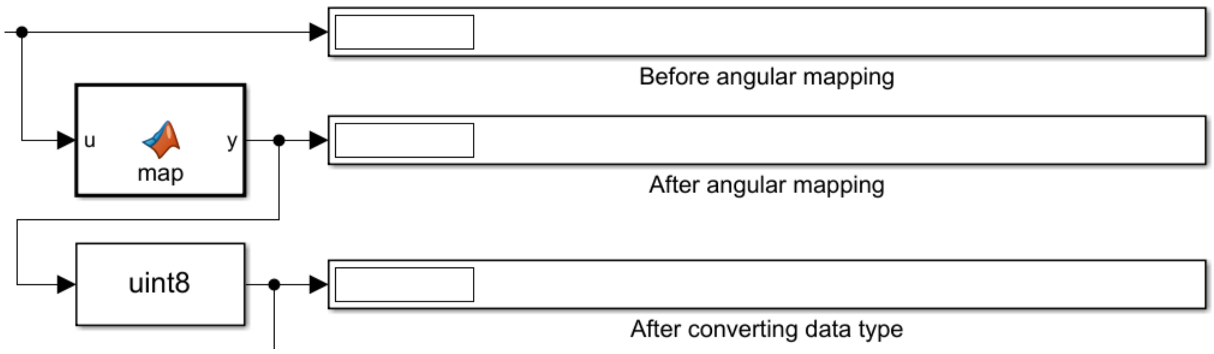
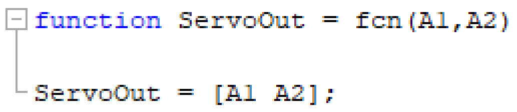


Figure 27. Angular mapping, data type conversion and result overview blocks

This block corresponds to the Angle input block shown in Figure 29. Angle input data for right arm goes to the input A1 and for left arm to the input A2 respectively. In expanded form, the function is as follows:



```
function ServoOut = fcn(A1,A2)

ServoOut = [A1 A2];
```

Figure 28. MATLAB function for matrix (expanded)

After all data from inverse kinematics calculation of left and right hand is collected into one matrix in order to be sent through I2C it should be converted into uint8 data type, as you can see in figure 34.

4.3 Experimental part

Since robot arm was built and control system was created hardware and control unit requires verification through experiment. Before starting experiment need to prepare control computer and experiment area. For this purpose, an experimental environment was created for working on the robot.

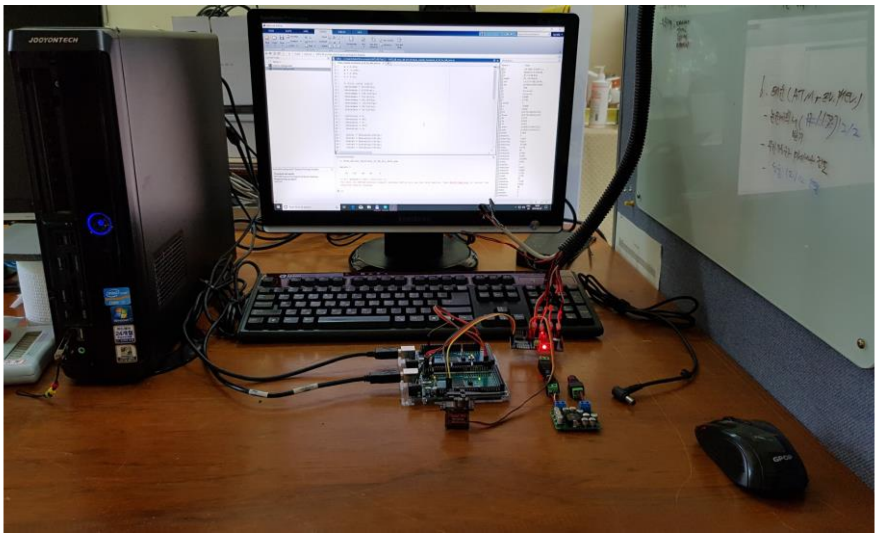


Figure 29. Experimental control unit

As you can see on figure 24 controlling unit of robot arm is computer for inverse kinematics calculation and preprocessing, 2 Arduino Mega 2560 boards for I2C communication and one Adafruit servo driver board for servo motor controlling. In addition to this there is a step-down regulator, and power source, respectively.

4.3.1 Point to Point control on each robotic arm

First, point to point control (P2P) experiment is conducted on both hands. Each arm is tested separately. Experimental data is shown in tables 11 and 12. There are 5 position inputs for each hand. 5 positions were transferred alternately. Thus, the ability of the control system to calculate the trajectory of movement after each input was tested.

In order to be able to control robot arm the range of servo motors should be known as well (see table 10).

Table 10. Servo motor motion range

Joint number	max, (deg)	min, (deg)	initial position, (deg)
1	120	70	70
2	130	30	70
3	130	30	70
4	80	20	20

Table 11. Left arm input data

Experiment	**Px, (m)**	**Py, (m)**	**Pz, (m)**
1	0.0964	0.1043	0.4811
2	0.2077	-0.1027	0.4075
3	-0.0176	0.2705	0.4535
4	0.4107	0.2779	0.0237
5	0.5857	0.1376	0.1550

Table 12. Right arm input data

Experiment	**Px, (m)**	**Py, (m)**	**Pz, (m)**
1	0.0964	-0.1043	0.4811
2	0.2077	0.1027	0.4075
3	-0.0176	-0.2705	0.4535
4	0.4107	-0.2779	0.0237
5	0.5857	-0.1376	0.1550

In this experiment, our main goal was to position the robotic arm, so that we excluded link number 5 from the experiment, since it only affects the orientation of the hand of robot arm.

The output was sent to Arduino Slave Write board after each step of calculation is finished. Summarizing the above, first step is kinematics calculation and obtaining joint angle values. Next step is converting angles into servo angles and finally sending obtained data through I2C communication to control robot arm. The initial position of robot arm shown in figure 25.

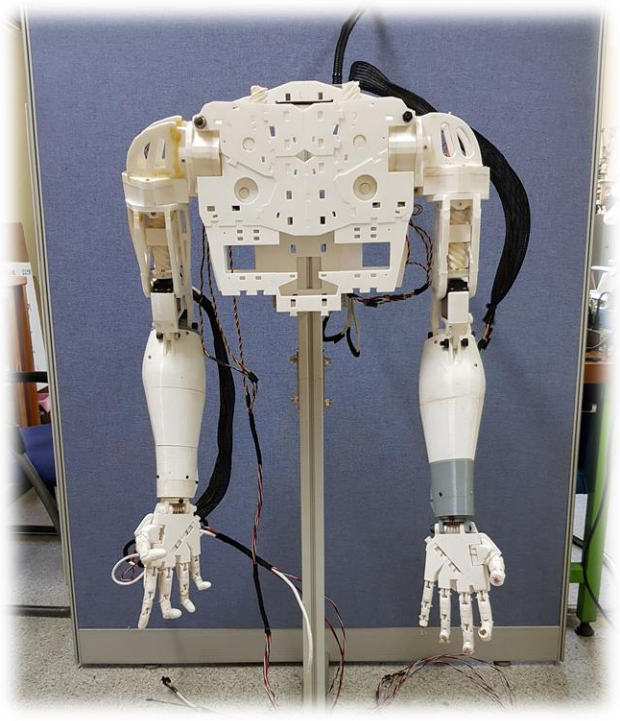


Figure 30. Initial position of robot arm

Experiment starts at initial position of robot arm and end also at initial position. The results of point to point experiment are displayed in figures 26 and 27.

In order to understand point to point experiment algorithm see figure 28. As you can see on the picture, first the input values of position are given to the control system, then, inverse kinematics calculation was conducted. After calculation is over, obtained data, which is angle values, are mapped onto the servo angles. Since, movement angle range is different from the on obtained by inverse kinematics calculation. Mapped data is converted onto uint8 data type, because I2C communication protocol is only works with iont8 or uint16 data types. Converted data is sent to Slave Write board through I2C. Finally, PWM mapping of accepted data is conducted by microcontroller and servo motors are controlled.

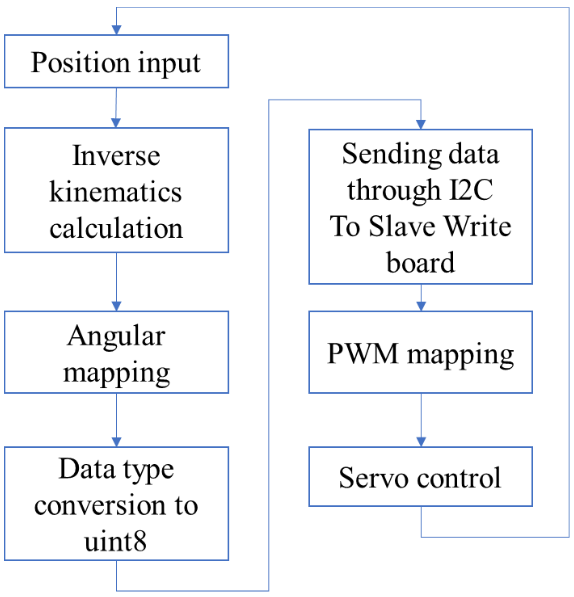
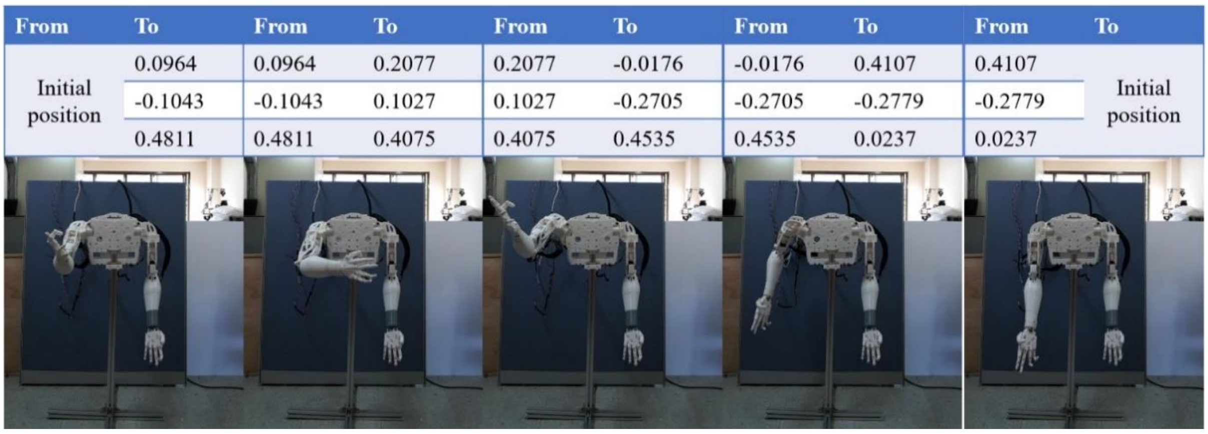
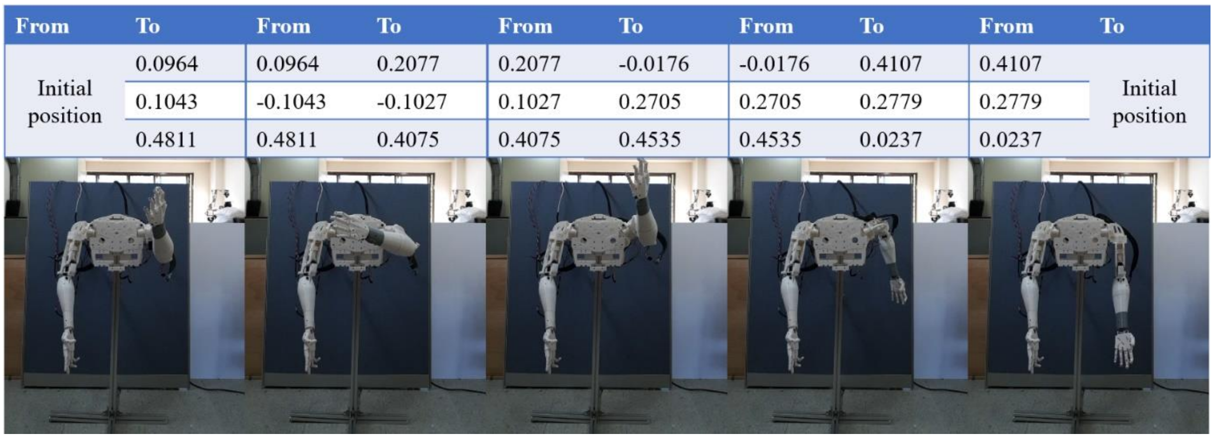

Position input
Inverse kinematics calculation
Angular mapping
Data type conversion to uint8
Sending data through I2C To Slave Write board
PWM mapping
Servo control

Figure 31. P2P experiment algorithm



From	To	From	To	From	To	From	To	From	To
Initial position	0.0964	0.0964	0.2077	0.2077	-0.0176	-0.0176	0.4107	0.4107	Initial position
	-0.1043	-0.1043	0.1027	0.1027	-0.2705	-0.2705	-0.2779	-0.2779	
	0.4811	0.4811	0.4075	0.4075	0.4535	0.4535	0.0237	0.0237	

Figure 32. Right arm P2P experiment



From	To	From	To	From	To	From	To	From	To
Initial position	0.0964	0.0964	0.2077	0.2077	-0.0176	-0.0176	0.4107	0.4107	Initial position
	0.1043	-0.1043	-0.1027	0.1027	0.2705	0.2705	0.2779	0.2779	
	0.4811	0.4811	0.4075	0.4075	0.4535	0.4535	0.0237	0.0237	

Figure 33. Left arm P2P experiment

As you can see here, both arms are placed in desired position correctly, which means that inverse kinematics calculation is correct and control system of single hand work is also correct.

Next experiment is dual arm point to point experiment. Algorithm for this experiment is the same as for experiment conducted earlier. Position input data is shown on table 13 and 14. The results of experiment is shown in figure 29.

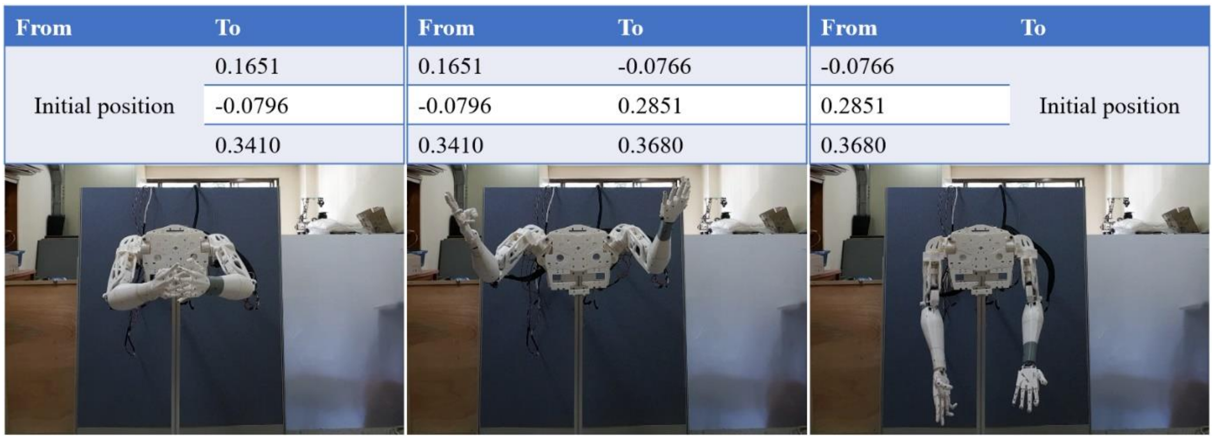
Table 13. Left arm input data

Experiment	**Px, (m)**	**Py, (m)**	**Pz, (m)**
1	0.1651	-0.0796	0.3410
2	-0.0766	0.2851	0.3680
3	0.5955	0.0860	0.1550

Table 14. Right arm input data

Experiment	**Px, (m)**	**Py, (m)**	**Pz, (m)**
1	0.1651	0.0796	0.3410
2	-0.0766	-0.2851	0.3680
3	0.5955	-0.0860	0.1550

As you can see here experiment was successful as well. Both arm correctly positioned is desired position.



From	To	From	To	From	To
Initial position	0.1651	0.1651	-0.0766	-0.0766	Initial position
	-0.0796	-0.0796	0.2851	0.2851	
	0.3410	0.3410	0.3680	0.3680	

Figure 34. Dual arm P2P experiment

4.4 Discussion

A total of three tests were conducted that show the capabilities of robotic arms and also confirms that the developed method of control of the robot arm is fully functional. Of course, the system still requires adjustments, improvements, and some additional elements, such as a vision system for autonomous operation of robot arm. It is also necessary to create a control box that will help locate and connect all component parts in one place, which in turn will solve the problem of mobility. At the moment, a personal computer is used as the main component of the monitoring system.

Results:

- Hardware and software parts were successfully developed
- According to experiment control system is fully functional

Need improvement:

- Servo speed control
- Force control
- Robot hand (gripper) control

Solved research problems:

- Design, production using 3D printing technology and assemble of humanoid robot arm
- Development of control system hardware using Arduino
- Development of control system software using Arduino IDE and MATLAB / Simulink

Achieved study purpose:

- Construction of low-cost humanoid robot arm using 3D printing technology
- Study on hardware and control of humanoid robot arm
- Study on control system

Chapter 5. Conclusion

In this article, a method was described for creating a low-cost humanoid robot and control system. Since the main goal was to create a control method for a robotic arm, an open resource robot was taken as a hardware. This allowed us to exclude the design, projection and creation of a manipulator. The robot was based on the InMoov robotic platform with an open source. First, the left and right hands of the robot were printed on a 3D printer and assembled, including activators and sensors. Further, the forward kinematics and inverse kinematics of the robot arm were calculated. The inverse kinematics method used in this paper is Damped Least-Square method. This method implies a reduction in accuracy to suitability. Hence, it is not quite accurate. However, as we could see during analysis of this method error rate is quite low.

Kinematic analysis included performing forward kinematics calculations and obtaining output data. Further, these data were used in the calculations of inverse kinematics. The data thus obtained during inverse kinematics calculation were compared with the initial input data for calculating the forward kinematics. Then, they were used to carry out another forward kinematics calculation in order to compare the output data with those obtained during the first forward kinematics calculation. Summarizing the above, a three-step test was performed on the method of calculating inverse kinematics. Based on the analysis, the method turned out to be extremely accurate. In fact, merge of acceptable error level can be controlled, therefore user can define lower level of merge, but in this case number of iterations increases.

After all the calculations were carried out and analyzed, an algorithm was developed to calculate the inverse kinematics and data transmission via the I2C communication protocol. This protocol was chosen because it allows user to control multiple devices connected in series using one master device. Then a control system based on kinematic calculations was created. It includes two Arduino boards, which were used to receive a signal from the main unit, which in this case is a personal computer; and Adafruit servo drivers used to control servo motors. Hardware test showed that the control system is working properly. The gap between the data transfer after the calculation and the movement of the servomotors is very small. The servo driver also allows servo motors to operate simultaneously.

In this article, in the first place, there was a general introduction to humanoid robots, which are the main trend in the development of robots these days. secondly, a preliminary study was conducted of domestic and foreign work in the development of humanoid robots. In the main part of the article, methods of kinematic calculations, the creation of a robot hardware unit and a control algorithm were presented. Then in the experimental part a standard analysis was carried out: definition of input data, carrying out calculations, controlling the device based on calculations and recording the results. Despite the amount of work done, the creation of a control system has not yet received a complete look. Using the test, the robotic arm can be controlled, but the control system lacks a user interface and additional sensors that will make the operation more convenient.

In the future, there will be a work carried out on the design and creation of the user interface, as well as connecting the vision system. There will also be work on combining the entire hardware of the control system into one small control unit. Also, most importantly, the use of robot hands to determine the application and the list of works performed by the humanoid robot.

References

1. "ABB's Collaborative Robot -YuMi - Industrial Robots from ABB Robotics." ABB's Collaborative Robot -YuMi - Industrial Robots from ABB Robotics. https://new.abb.com/products/robotics/industrial-robots/irb-14000-yumi.
2. "Adafruit PCA9685 16-Channel Servo Driver." Overview | Adafruit PCA9685 16-Channel Servo Driver | Adafruit Learning System. https://learn.adafruit.com/16-channel-pwm-servo-driver.
3. "Basics of the I2C Communication Protocol." Circuit Basics. April 11, 2017. http://www.circuitbasics.com/basics-of-the-i2c-communication-protocol/.
4. "BionicCobot." BionicCobot | Festo Corporate. https://www.festo.com/group/en/cms/12746.htm.
5. "Build Yours." InMoov. http://inmoov.fr/build-yours/.
6. "Humanoid Robot." Wikipedia. April 16, 2019. https://en.wikipedia.org/wiki/Humanoid_robot.
7. "PaPeRo." Wikipedia. February 25, 2018. https://en.wikipedia.org/wiki/PaPeRo.
8. "Qrio - ROBOTS: Your Guide to the World of Robotics." ROBOTS. https://robots.ieee.org/robots/qrio/.
9. "QRIO." Wikipedia. February 03, 2019. https://en.wikipedia.org/wiki/QRIO.
10. "Reachy." Pollen Robotics. https://www.pollen-robotics.com/reachy/.
11. "Sophia." Hanson Robotics. https://www.hansonrobotics.com/sophia/.
12. "Types of Robots - Humanoid, Japanese, Autonomous Robots, ASIMO, AIBO, HRP, QRIO." Science Kids - Fun Science & Technology for Kids! http://www.sciencekids.co.nz/sciencefacts/technology/typesofrobots.html.
13. "Wire." Arduino. https://www.arduino.cc/en/reference/wire.
14. Ambrose, R.o., H. Aldridge, R.s. Askew, R.r. Burridge, W. Bluethmann, M. Diftler, C. Lovchik, D. Magruder, and F. Rehnmark. "Robonaut: NASA's Space Humanoid." *IEEE Intelligent Systems* 15, no. 4 (07 2000): 57-63. doi:10.1109/5254.867913.
15. Angeles, Jorge. "Iterative Kinematic Inversion of General Five-Axis Robot Manipulators." *The International Journal of Robotics Research* 4, no. 4 (01 1986): 59-70. doi:10.1177/027836498600400405.
16. Arduino Mega 2560 Rev3. https://store.arduino.cc/usa/mega-2560-r3.
17. Bazylev, D.n., A.a. Pyrkin, A.a. Margun, K.a. Zimenko, A.s. Kremlev, D.d. Ibraev, and M. Cech. "Approaches for Stabilizing of Biped Robots in a Standing Position on Movable

Support." *Scientific and Technical Journal of Information Technologies, Mechanics and Optics*, 05, 2015, 418-25. doi:10.17586/2226-1494-2015-15-3-418-425.

18. Benhabib, B., A. A. Goldenberg, and R. G. Fenton. "A Solution to the Inverse Kinematics of Redundant Manipulators." *Journal of Robotic Systems* 2, no. 4 (1985): 373-85. doi:10.1002/rob.4620020404.
19. Bers, Marina Umaschi. "Coding, Playgrounds and Literacy in Early Childhood Education: The Development of KIBO Robotics and ScratchJr." *2018 IEEE Global Engineering Education Conference (EDUCON)*, 04 2018. doi:10.1109/educon.2018.8363498.
20. Cho, Gun Rae, Mun-Jik Lee, Min-Gyu Kim, and Ji-Hong Li. "Inverse Kinematics for Autonomous Underwater Manipulations Using Weighted Damped Least Squares." *2017 14th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, 06 2017. doi:10.1109/urai.2017.7992822.
21. Choudhury, Avishek, Huiyang Li, and Christopher M. Greene. "Humanoid Robot: Application and Influence." *International Journal of Applied Science - Research and Review* 05, no. 04 (2018). doi:10.21767/2394-9988.100082.
22. Donald L. Pieper, The kinematics of manipulators under computer control. PhD, 1968.
23. Dsouza, Royson & Denny, Joe & Angel D’costa, Shannon & Elyas, Mohamed. (2016). Humanoid Robots – Past, Present and the Future. 3. 8-15.
24. Durán,sta Boris, and Serge Thill. *Rob's Robot: Current and Future Challenges for Humanoid Robots*. INTECH Open Access Publisher, 2012.
25. Fujita, M., Y. Kuroki, T. Ishida, and T.t. Doi. "A Small Humanoid Robot SDR-4X for Entertainment Applications." *Proceedings 2003 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM 2003)*. doi:10.1109/aim.2003.1225468.
26. Hammam, Ghassan Bin, Patrick M. Wensing, Behzad Dariush, and David E. Orin. "Kinodynamically Consistent Motion Retargeting for Humanoids." *International Journal of Humanoid Robotics* 12, no. 04 (12 2015): 1550017. doi:10.1142/s0219843615500176.
27. Han, Jeakweon. (2012). Bipedal Walking for a Full-sized Humanoid Robot Utilizing Sinusoidal Feet Trajectories and Its Energy Consumption.
28. Heo JW., Lee J., Lee IH., Lim J., Oh JH. (2017) History of HUBO, Korean Humanoid Robot. In: Goswami A., Vadakkepat P. (eds) Humanoid Robotics: A Reference. Springer, Dordrecht
29. Hirai, Kazuo. "The Honda Humanoid Robot: Development and Future Perspective." *Industrial Robot: An International Journal* 26, no. 4 (06 1999): 260-66. doi:10.1108/01439919910277431.

30. Hornung, A., K. M. Wurm, and M. Bennewitz. "Humanoid Robot Localization in Complex Indoor Environments." *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 10 2010. doi:10.1109/iros.2010.5649751.
31. I2C. https://learn.sparkfun.com/tutorials/i2c/all.
32. Jung, Yang-Geun, O-Duek Lim, Min-Seong Kim, Ki-Hoon Do, and Sung-Hyun Han. "A Study on Kinematics Analysis and Motion Control of Humanoid Robot Arm with Eight Joints." *Journal of the Korean Society of Industry Convergence* 20, no. 1 (03, 2017): 49-55. doi:10.21289/ksic.2017.20.1.049.
33. Kagami, Satoshi, Koichi Nishiwaki, James Kuffner, Kei Okada, Yasuo Kuniyoshi, Masayuki Inaba, and Hirochika Inoue. "Low-level Autonomy of the Humanoid Robots H6 & H7." *Springer Tracts in Advanced Robotics Robotics Research*: 83-97. doi:10.1007/3-540-36460-9_6.
34. Kaneko, K., F. Kanehiro, S. Kajita, H. Hirukawa, T. Kawasaki, M. Hirata, K. Akachi, and T. Isozumi. "Humanoid Robot HRP-2." *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*, 2004. doi:10.1109/robot.2004.1307969.
35. Kim, Joohyung, Younbaek Lee, Sunggu Kwon, Keehong Seo, Hoseong Kwak, Heekuk Lee, and Kyungsik Roh. "Development of the Lower Limbs for a Humanoid Robot." *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 10 2012. doi:10.1109/iros.2012.6385728.
36. Kim, Jung-Hoon & Park, Seo-Wook & Park, Ill-Woo & Oh, Jun-Ho. (2002). Development of a humanoid biped walking robot platform KHR-1-initial design and its performance evaluation.
37. Kim, Jung-Yup, Ill-Woo Park, Jungho Lee, Min-Su Kim, Baek-Kyu Cho, and Jun-Ho Oh. "System Design and Dynamic Walking of Humanoid Robot KHR-2." *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*. doi:10.1109/robot.2005.1570316.
38. Kim, Sanghyun, Mingon Kim, Jimin Lee, Soonwook Hwang, Joonbo Chae, Beomyeong Park, Hyunbum Cho, Jaehoon Sim, Jaesug Jung, Hosang Lee, Seho Shin, Minsung Kim, Nojun Kwak, Yongjin Lee, Sangkuk Lee, Myunggi Lee, Sangyup Yi, Kyong-Sok K.c. Chang, and Jaeheung Park. "Approach of Team SNU to the DARPA Robotics Challenge Finals." *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, 11 2015. doi:10.1109/humanoids.2015.7363458.
39. Krüger, J., G. Schreck, and D. Surdilovic. "Dual Arm Robot for Flexible and Cooperative Assembly." *CIRP Annals* 60, no. 1 (2011): 5-8. doi:10.1016/j.cirp.2011.03.017.

40. Manseur, Rachid, and Keith L. Doty. "A Complete Kinematic Analysis of Four-revolute-axis Robot Manipulators." *Mechanism and Machine Theory* 27, no. 5 (09 1992): 575-86. doi:10.1016/0094-114x(92)90046-k.

41. Manseur, Rachid, and Keith L. Doty. "Fast Inverse Kinematics of Five-revolute-axis Robot Manipulators." *Mechanism and Machine Theory* 27, no. 5 (09 1992): 587-97. doi:10.1016/0094-114x(92)90047-l.

42. Nakamura, Yoshihiko, and Hideo Hanafusa. "Inverse Kinematic Solutions with Singularity Robustness for Robot Manipulator Control." *Journal of Dynamic Systems, Measurement, and Control* 108, no. 3 (1986): 163. doi:10.1115/1.3143764.

43. Nakanishi, Yuto, Shigeki Ohta, Takuma Shirai, Yuki Asano, Toyotaka Kozuki, Yuriko Kakehashi, Hironori Mizoguchi, Tomoko Kurotobi, Yotaro Motegi, Kazuhiro Sasabuchi, Junichi Urata, Kei Okada, Ikuo Mizuuchi, and Masayuki Inaba. "Design Approach of Biologically-Inspired Musculoskeletal Humanoids." *International Journal of Advanced Robotic Systems* 10, no. 4 (01 2013): 216. doi:10.5772/55443.

44. Oh, Jun-Ho, David Hanson, Won-Sup Kim, Young Han, Jung-Yup Kim, and Ill-Woo Park. "Design of Android Type Humanoid Robot Albert HUBO." *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 10 2006. doi:10.1109/iros.2006.281935.

45. Oscar Efrain Ramos Ponce. Generation of the whole-body motion for humanoid robots with the complete dynamics. Robotics [cs.RO]. Universite Toulouse III Paul Sabatier, 2014. English.

46. Park, Ill-Woo, Jung-Yup Kim, Jungho Lee, and Jun-Ho Oh. "Mechanical Design of Humanoid Robot Platform KHR-3 (KAIST Humanoid Robot - 3: HUBO)." *5th IEEE-RAS International Conference on Humanoid Robots, 2005.* doi:10.1109/ichr.2005.1573587.

47. Park, Ill-Woo, Jung-Yup Kim, Jungho Lee, and Jun-Ho Oh. "Mechanical Design of the Humanoid Robot Platform, HUBO." *Advanced Robotics* 21, no. 11 (01 2007): 1305-322. doi:10.1163/156855307781503781.

48. Robotics - Humanoid Robot HRP-4 | KAWADA INDUSTRIES, INC. http://global.kawada.jp/mechatronics/hrp4.html.

49. Shigemi, Satoshi. "ASIMO and Humanoid Robot Research at Honda." *Humanoid Robotics: A Reference*, 2017, 1-36. doi:10.1007/978-94-007-7194-9_9-2.

50. Shimizu, Masayuki. "Analytical Inverse Kinematics for 5-DOF Humanoid Manipulator under Arbitrarily Specified Unconstrained Orientation of End-effector." *Robotica* 33, no. 4 (03, 2014): 747-67. doi:10.1017/s0263574714000538.

51. Stasse, O., and T. Flayols. "An Overview of Humanoid Robots Technologies." *Springer*

Tracts in Advanced Robotics Biomechanics of Anthropomorphic Systems, 08, 2018, 281-310. doi:10.1007/978-3-319-93870-7_13.

52. THORMANG3. "Introduction." Manual. http://emanual.robotis.com/docs/en/platform/thormang3/introduction/.
53. Tsagarakis, N. G., G. Metta, G. Sandini, D. Vernon, R. Beira, F. Becchi, L. Righetti, J. Santos-Victor, A. J. Ijspeert, M. C. Carrozza, and D. G. Caldwell. "ICub: The Design and Realization of an Open Humanoid Platform for Cognitive and Neuroscience Research." *Advanced Robotics* 21, no. 10 (01 2007): 1151-175. doi:10.1163/156855307781389419.
54. Tsarouchi, Panagiota, Sotiris Makris, George Michalos, Michael Stefos, Konstantinos Fourtakas, Konstantinos Kaltsoukalas, Dimitris Kontrovrakis, and George Chryssolouris. "Robotized Assembly Process Using Dual Arm Robot." *Procedia CIRP* 23 (2014): 47-52. doi:10.1016/j.procir.2014.10.078.
55. Ulbrich, Heinz, T. Buschmann, and S. Lohmeier. "Development of the Humanoid Robot LOLA." *Applied Mechanics and Materials* 5-6 (10 2006): 529-40. doi:10.4028/www.scientific.net/amm.5-6.529.
56. Chiaverini, S., B. Siciliano, and O. Egeland. "Review of the Damped Least-squares Inverse Kinematics with Experiments on an Industrial Robot Manipulator." *IEEE Transactions on Control Systems Technology* 2, no. 2 (06 1994): 123-34. doi:10.1109/87.294335.
57. Wampler, Charles. "Manipulator Inverse Kinematic Solutions Based on Vector Formulations and Damped Least-Squares Methods." *IEEE Transactions on Systems, Man, and Cybernetics* 16, no. 1 (01 1986): 93-101. doi:10.1109/tsmc.1986.289285.
58. Wang, Hongfei, Yuan F. Zheng, Youngbum Jun, and Paul Oh. "DRC-hubo Walking on Rough Terrains." *2014 IEEE International Conference on Technologies for Practical Robot Applications (TePRA)*, 04 2014. doi:10.1109/tepra.2014.6869151.
59. You, Bum-Jae, Doik Kim, Changhwan Kim, Yong-Hwan Oh, Mun-Ho Jeong, and Sang-Rok Oh. "Network-based Humanoid 'MAHRU' as Ubiquitous Robotic Companion." *IFAC Proceedings Volumes* 41, no. 2 (2008): 724-29. doi:10.3182/20080706-5-kr-1001.00124.
60. Yu, Zhangguo, Qiang Huang, Gan Ma, Xuechao Chen, Weimin Zhang, Jing Li, and Junyao Gao. "Design and Development of the Humanoid Robot BHR-5." *Advances in Mechanical Engineering* 6 (01 2014): 852937. doi:10.1155/2014/852937.
61. Zhang, Yajia, Jingru Luo, Kris Hauser, H. Andy Park, Manas Paldhe, C. S. George Lee, Robert Ellenberg, Brittany Killen, Paul Oh, Jun Ho Oh, Jungho Lee, and Inhyeok Kim. "Motion Planning and Control of Ladder Climbing on DRC-Hubo for DARPA Robotics Challenge." *2014 IEEE International Conference on Robotics and Automation (ICRA),* 05

2014. doi:10.1109/icra.2014.6907139.

62. 김바울, 송명진, 이근주, 김용덕, 김상욱, 김경덕, (2011). 휴머노이드 로봇의 실시간 드로잉을 위한 윤곽선의 궤적 좌표 집합 추출 알고리즘. 한국정보과학회 학술발표논문집,38(1C),413-416.

63. 오준호, (2004). [테마기획 - Intelligent Robotics 현황 및 발전 방향] 휴머노이드 로봇의 현황과 발전 방향. 기계저널,44(4),44-52.

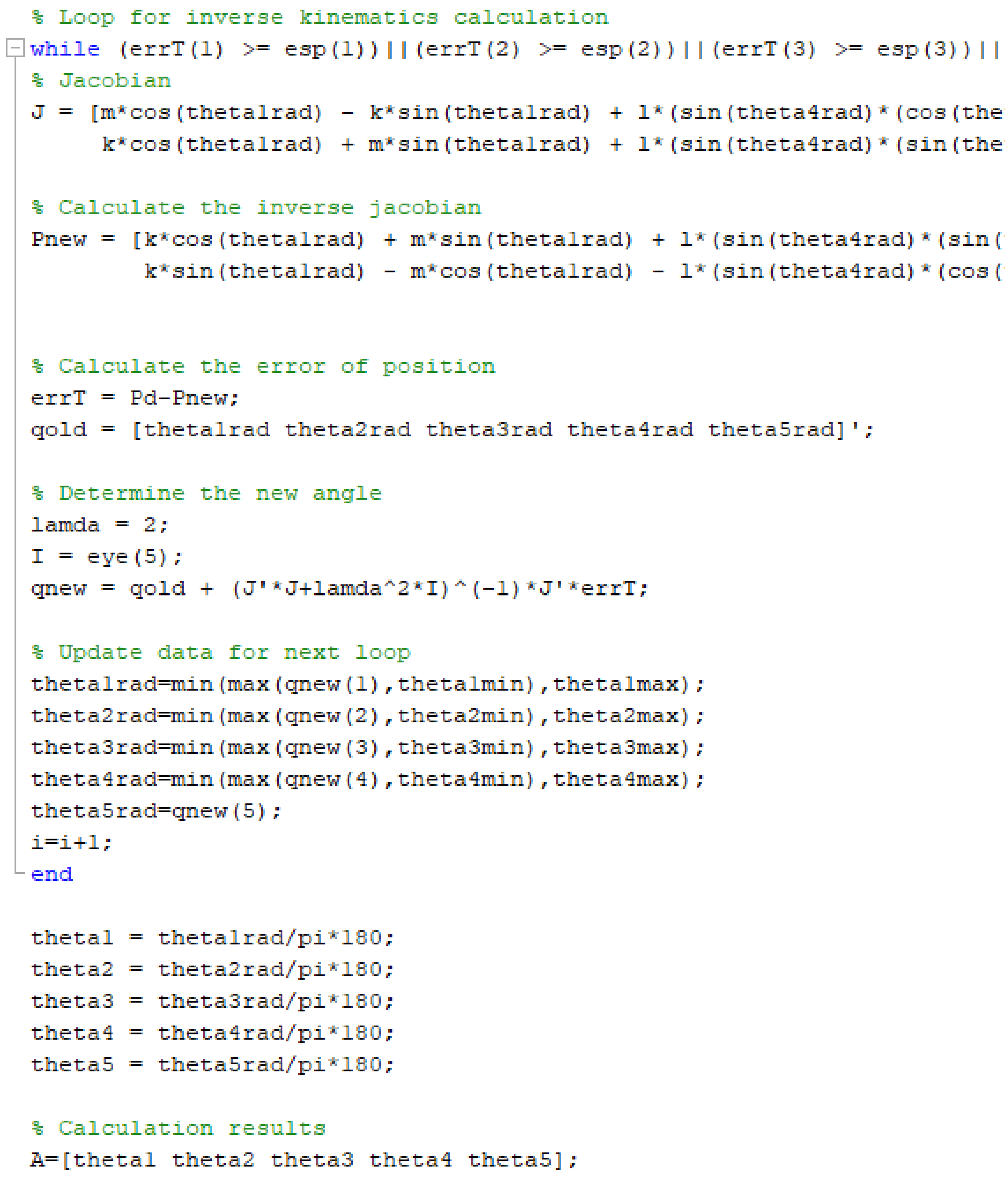
64. 유범재, 오용환, 최영진, (2004). 휴머노이드 연구동향. 한국정밀공학회지, 21(7), 15-21. 전상원, (2015). 국내의 휴머노이드 로봇. 전자공학회지,42(12),18-25.

65. Samuel R. Buss, “Introduction to Inverse Kinematics with Jacobian Transpose, Pseudoinverse and Damped Least Squares Methods”, April 17, 2004 Unpublished.

APPENDIX A

MATLAB code for Inverse Kinematics

```
% DH parameters
k = 0.044;
m = 0.086;
n = 0.283;
l = 0.31;

% Max and Min values of joint range
theta1max = -5/180*pi;
theta1min = -60/180*pi;
theta2max = 135/180*pi;
theta2min = 25/180*pi;
theta3max = 60/180*pi;
theta3min = -40/180*pi;
theta4max = 90/180*pi;
theta4min = 30/180*pi;

% Initial position
theta1ini = 0;
theta2ini = 90;
theta3ini = 0;
theta4ini = 30;
theta5ini = 0;

% Converting to radian
theta1rad = theta1ini/180*pi;
theta2rad = theta2ini/180*pi;
theta3rad = theta3ini/180*pi;
theta4rad = theta4ini/180*pi;
theta5rad = theta5ini/180*pi;

% Defining the merge of error
esp = [0.05 0.05 0.05]'; % margin of error
errT = [0 0 0]';

% Input data
Px = 0.0173;
Py = -0.0878;
Pz = 0.4193;

Pd = [Px Py Pz]';
i = 1; % Number of iterations: 1000
```



```
% Loop for inverse kinematics calculation
while (errT(1) >= esp(1))||(errT(2) >= esp(2))||(errT(3) >= esp(3))||
% Jacobian
J = [m*cos(theta1rad) - k*sin(theta1rad) + 1*(sin(theta4rad)*(cos(the
     k*cos(theta1rad) + m*sin(theta1rad) + 1*(sin(theta4rad)*(sin(the

% Calculate the inverse jacobian
Pnew = [k*cos(theta1rad) + m*sin(theta1rad) + 1*(sin(theta4rad)*(sin(
        k*sin(theta1rad) - m*cos(theta1rad) - 1*(sin(theta4rad)*(cos(


% Calculate the error of position
errT = Pd-Pnew;
qold = [theta1rad theta2rad theta3rad theta4rad theta5rad]';

% Determine the new angle
lamda = 2;
I = eye(5);
qnew = qold + (J'*J+lamda^2*I)^(-1)*J'*errT;

% Update data for next loop
theta1rad=min(max(qnew(1),theta1min),theta1max);
theta2rad=min(max(qnew(2),theta2min),theta2max);
theta3rad=min(max(qnew(3),theta3min),theta3max);
theta4rad=min(max(qnew(4),theta4min),theta4max);
theta5rad=qnew(5);
i=i+1;
end

theta1 = theta1rad/pi*180;
theta2 = theta2rad/pi*180;
theta3 = theta3rad/pi*180;
theta4 = theta4rad/pi*180;
theta5 = theta5rad/pi*180;

% Calculation results
A=[theta1 theta2 theta3 theta4 theta5];
```

APPENDIX B

Arduino code for "Slave Write" board

```
#include <Adafruit_PWMServoDriver.h>
#include <Wire.h>

// Defining max and min values of pulse length
// and setting initial position for the the robot
int servomin = 180;
int servomax = 600;
int angleSet1 = 0, x1;
int angleSet2 = 0, x2;
int angleSet3 = 0, x3;
int angleSet4 = 0, x4;
int angleSet5 = 0, x5;

double pulselength1, pulselength2, pulselength3;
double pulselength4, pulselength5;

// Declaration of servo driver board. "0x40" is the address
Adafruit_PWMServoDriver pwm = Adafruit_PWMServoDriver(0x40);

void setup() {
  // put your setup code here, to run once:
  Wire.begin(9);
  Wire.onReceive(receiveEvent);
  pwm.begin();
  pwm.setPWMFreq(60);
  delay(10);
}

// Saving received angle values
void receiveEvent(int byte) {
  x1 = Wire.read();
  x2 = Wire.read();
  x3 = Wire.read();
  x4 = Wire.read();
  x5 = Wire.read();
}
```

```
// Assinging received values
void loop() {
  // put your main code here, to run repeatedly:
  angleSet1 = x1;
  angleSet2 = x2;
  angleSet3 = x3;
  angleSet4 = x4;
  angleSet5 = x5;

  // Converting angular values into pulse signals (mapping)
  int pulselength1 = map(angleSet1, 0, 180, servomin, servomax);
  pwm.setPWM(0, 0, pulselength1);
  int pulselength2 = map(angleSet2, 0, 180, servomin, servomax);
  pwm.setPWM(2, 0, pulselength2);
  int pulselength3 = map(angleSet3, 0, 180, servomin, servomax);
  pwm.setPWM(4, 0, pulselength3);
  int pulselength4 = map(angleSet4, 0, 180, servomin, servomax);
  pwm.setPWM(6, 0, pulselength4);
  int pulselength5 = map(angleSet5, 0, 180, servomin, servomax);
  pwm.setPWM(8, 0, pulselength5);
  delay(10);
}
```