Table of contents

Summary

The aim of this thesis is to design, build and analyze the kinematics of an articulated robot with four degrees *of freedom* (4DOF), equipped with rotary joints. The paper presents the next stages of the project. Chapter 2 presents the theoretical foundations of direct and inverse kinematics along with the derivation of the effector position equations in space. Chapter 3 provides an overview of existing methods for compensating for position errors that are designed to increase the precision of the manipulator. Chapter 4 discusses in detail the mechanical design of the arm, the integration of the electronic system, and the configuration of the power supply using a laboratory stabilized power supply. Chapter 5 is devoted to practical tests and experimental verification of the system's operation. The project developed two independent control interfaces – a PC app and a mobile app using Bluetooth communication. The implementation also includes the functionality of activating the error compensation algorithm, which corrects deviations in the mapping of the position of the robot tip in relation to the set coordinates. In addition, the functionality of the designed manipulator was compared with selected commercial solutions with four degrees of freedom.

The aim of this engineering thesis is to design, build, and analyze the kinematics of an articulated robot with four degrees of freedom (4DOF), equipped with rotary joints. The thesis presents the successive stages of the project implementation. Chapter 2 introduces the theoretical foundations of forward and inverse kinematics, along with the derivation of equations describing the end-effector's position in space. Chapter 3 provides an overview of existing position error compensation methods aimed at improving the precision of the manipulator's operation. Chapter 4 discusses in detail the mechanical construction of the robotic arm, integration of the electronic system, and the power supply configuration using a stabilized laboratory power supply. Chapter 5 is dedicated to practical testing and experimental validation of the system's performance. As part of the project, two independent control interfaces were developed – a desktop application and a mobile application using Bluetooth communication. The implementation also includes functionality for activating the error compensation algorithm, which corrects deviations in the end-effector's position relative to the target coordinates. Additionally, a comparison of the designed manipulator's functionality with selected commercial 4DOF solutions was conducted.

Chapter 1

1.1 Aim of the study

The aim of the work is to design and manufacture an electronic controller for a spherical manipulator with four degrees of freedom, enabling precise control of the robotic arm and remote communication via Bluetooth interface. An additional goal is to develop and implement algorithms for compensation of position errors resulting from design imperfections, calibration and external forces.

1.2 Thesis of the thesis

The design and implementation of kinematics and positioning error compensation algorithms in the 4DOF robot allows to increase the accuracy of its operation, while maintaining low prototyping costs thanks to the use of open hardware and programming platforms.

1.3 Operating range

The scope of this thesis includes the development and implementation of the design of a robotic arm with four degrees of freedom, starting from theoretical analysis, through mechanical and electronic design, to the implementation of control algorithms and functional tests.

In particular, the following elements were developed and implemented:

- analysis of the kinematics of the designed spherical robot,
- selection of mechanical and electronic components and design of the control system,
- modification of the robotic arm kit to version 4 DOF,
- implementation of control and compensation algorithms in the Arduino environment,
- creation of a mobile application and a desktop user interface for remote control of the keypad,
- conducting tests and experimental evaluation of the accuracy and effectiveness of the system.

Introduction

The robotic arm is a key component in automation, capable of performing precise and repetitive tasks in a variety of industries, such as manufacturing, packaging, and medical applications. The ability of the arm to perform complex movements depends primarily on the number of degrees of freedom, which determine the number of independent joint movements, allowing the positioning and orientation of the end effector. Although industrial robots typically have six or more degrees of freedom to mimic the dexterity of a human arm, robotic arms with four degrees of freedom represent a practical compromise between mechanical complexity and functionality.

A typical 4 DOF robotic arm contains joints that allow the base to rotate, move in the shoulder and elbow joints (in the vertical plane) and rotate the wrist, which allows it to move in three-dimensional space with additional rotation of the effector. This configuration is sufficient for many tasks, such as sorting and moving objects, operating machines or stacking goods on pallets, where full spatial orientation control is not necessary. The reduced complexity of the 4 DOF keypads simplifies design and control and allows for cost-effective implementation, making them popular in education and small and medium-sized enterprises.

The basis for the operation of each robotic arm is kinematic analysis, which includes solving problems of simple and inverse kinematics. Simple kinematics calculates the position and orientation of the effector from known joint angles, while inverse kinematics determines which joint angles are needed to achieve a given effector position. Although simple kinematics calculations are not particularly complex, inverse kinematics often involves the need to solve nonlinear systems of equations and the analysis of many possible solutions, which is a significant challenge in the design of control systems. Accurate kinematic modeling is crucial for motion trajectory planning, ensuring smooth and precise manipulator operation.

Recent advances in microcontrollers, such as the Arduino platform, have enabled low-cost control systems for 4 DOF arms, integrating servos and sensor feedback to reliably control motion. In addition, the development of communication technologies such as USB and Bluetooth interfaces allows flexible control of the device from computer and mobile applications. These innovations support both manual and automatic operation modes, increasing the range of applications of the 4 DOF arms in research and practice.

This thesis focuses on the design, modeling, and control of the 4 DOF robotic arm, with a focus on the implementation of simple and inverse kinematics algorithms and the method of positioning error compensation to increase accuracy. The project uses open hardware and software tools to demonstrate effective, low-cost prototyping and to verify the impact of compensation algorithms on positioning precision.

Examples of popular 4-DOF articulated robots:

1. RoArm-M2-S [10]

Four-axis robot designed for education, research and light automation applications. Thanks to its lightweight carbon fiber and aluminum construction, it provides high precision and ease of integration into various control systems.

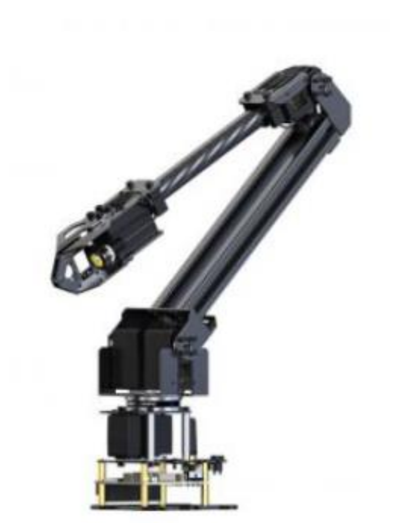


Figure 1.1: RoArm-M2-S robot.

Table 1.1. Basic parameters of the RoArm-M2-S robot.

Parameter	**Value**
Number of axles	4
Maximum load capacity	0.5 kg
Arm reach	1 m
Repeatability	±0.1 mm
Scales	0.83 kg
Controller	ESP32, ROS2

2. 4-DOF Multifunction Robotic Arm – IADIY [11]

Compact robotic arm for education and hobby automation projects. Featuring a lightweight design and USB/Bluetooth interface, it allows you to quickly deploy motion control projects in environments such as Arduino and Python. It supports functions such as gripper, camera or vacuum suction cup.

Rysunek 1.2: Robot 4-DOF Multifunction Robotic Arm – IADIY.

Tabela 1.2. Podstawowe parametry robota 4-DOF Multifunction Robotic Arm – IADIY.

Parameter	**Value**
Number of axles	4
Maximum load capacity	approx. 0.2–0.3 kg
Arm reach	approx. 35–50 cm
Repeatability	~±0.5 mm (est.)
Scales	approx. 1.2 kg
Controller	Arduino

3. Igus® robolink® DP – 4DOF [12]

Modular industrial plastic manipulator with high durability, designed for light automation applications. It enables you to work with ROS and the free igus® Robot Control software. Also available in IP44 (splash resistant) version.

Rysunek 1.3: Robot Igus® robolink® DP – 4DOF.

Table 1.3. Basic parameters of the Igus® robolink® DP robot – 4DOF.

Parameter	**Value**
Number of axles	4
Maximum load capacity	2-3 kg
Arm reach	790 mm
Repeatability	±0.5 mm
Scales	~5–6 kg (depending on version)
Controller	igus® Robot Control, ROS

The presented robots are characterized by a specific arrangement of joint axes, enabling the implementation of a wide range of positions and orientation of the effector in the working area. This allows them to successfully perform a variety of manipulative tasks. In the further part of the work, a table containing its technical parameters will be presented. This will enable direct comparison with the previously described models and assessment of its functionality in the context of solutions available on the market.

Chapter 2

Robot kinematics

The kinematics of an articulated robot is the basic stage in the design of a control system that allows you to determine the position, orientation, and trajectory of the robot's tip relative to the base system [1]. It describes the movement of individual manipulator members without taking into account the forces and moments acting, focusing only on geometric relations resulting from the given hinged angles.

In the case of articulated manipulators, two main issues are of particular importance: forward kinematics and inverse kinematics. Their correct development allows for precise determination of the position of the effector in the working area and allows to control the movement of the robot in a manner consistent with application requirements.

2.1 Basic concepts

An articulated manipulator consists of a series of components that are connected to each other by means of rotary joints. Each joint allows one degree of freedom, and the complete configuration of the robot is determined by the number of such joints.

In the case of the designed spherical robot with four degrees of freedom, only rotary joints (type R) are used, which enables the movement of the effector in three-dimensional space.

The basic tool used to analyze the motion of the manipulator is a homogeneous transformation matrix , which allows the description of the position and orientation of a given cell in relation to the base system. In this paper, the commonly used Denavit-Hartenberg (D-H) method was used, which defines the geometry of the manipulator using four parameters describing the spatial relationships between successive hinges [2].

2.2 Simple kinematics

The aim of the simple kinematics analysis is to determine the position and orientation of the end effector on the basis of known values of the angles of the individual robot joints. In the case of an articulated manipulator, this involves transforming a set of input values (rotation angles for each hinge) into a single transformation matrix that describes the total displacement from the underlying system.

The Denavit-Hartenberg method consists in assigning local coordinate systems to each manipulator term and describing the transformations between them using four parameters:

- θ_i— the angle of rotation around the axis,z_{i-1}
- d_i— Offset along axis,z_{i-1}
- a_i — the length of the member along the axis,x_i
- α_i — the angle of rotation about the x_i axis, between the .z_{i-1} and z_i axes

Coordinate systems in the D-H method

In order to correctly assign the D-H parameters, each member should be assigned a local coordinate system according to the following rules:

- The axis z_i is the axis of the joint and indicates the direction of movement (for rotary joints – the axis of rotation),
- The axis x_i is the common normal between the z_{i-1} and z_i axes — it is perpendicular to both,
- The origin of the system is at the intersection of the z_i and x_i axis,
- The y_i axis completes the right-hand system.

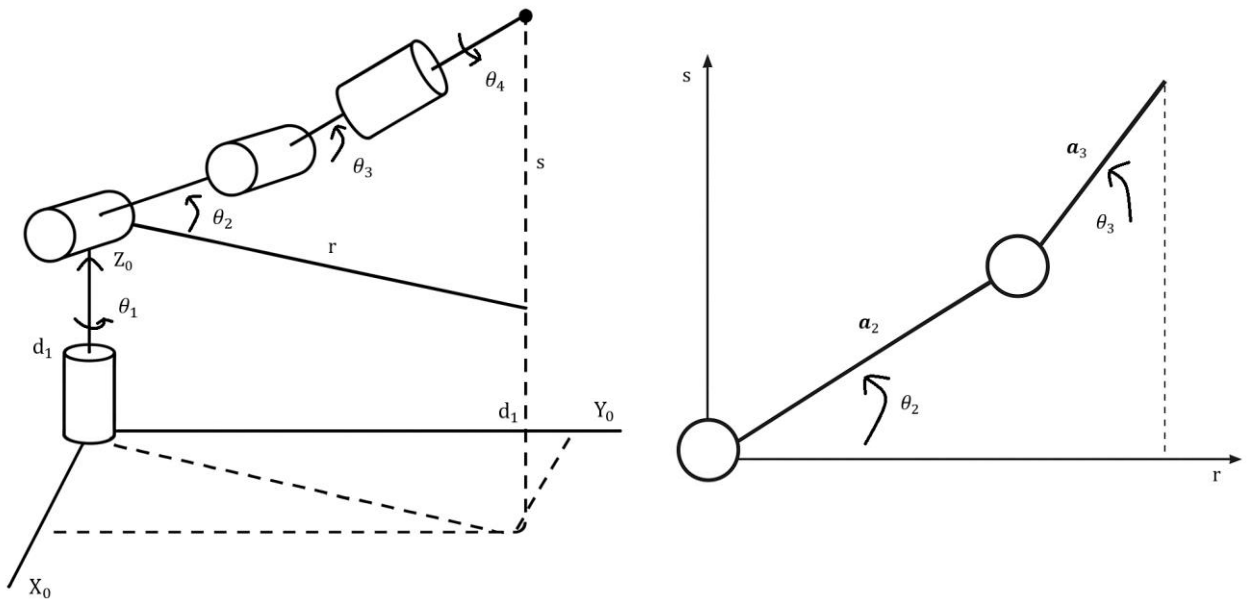


Figure 2.1: Diagram of the coordinate systems and DH parameters for the 4DOF of the keypad. [3]

r **-** corresponds to the horizontal distance (along the X axis) between the base of the manipulator and the final effect of the operation,

s **-** indicates the vertical distance (along the Z-axis) of the tonearm tip from the base.

These parameters are used to calculate the position of the robot tip and to determine the required rotation angles for the individual joints.

D-H map matrix

The transformation between successive coordinate systems is described by a homogeneous matrix [4]:

$$^{i-1}T_i = \begin{bmatrix} \cos\theta_i & -\sin\theta_i \cos\alpha_i & \sin\theta_i \sin\alpha_i & \alpha_i \cos\theta_i \\ \sin\theta_i & \cos\theta_i \cos\alpha_i & -\cos\theta_i \sin\alpha_i & \alpha_i \sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The final transformation of the effector position with respect to the robot's base system is expressed as the product of all matrices:

$$T_0^n = T_0^1 \times T_1^2 \times T_2^3 \times \cdots \times T_{n-1}^n$$

In the case of a robot with four degrees of freedom, the above equation takes the form:

$$T_0^4 = T_0^1 \times T_1^2 \times T_2^3 \times T_3^4$$

The designed robotic system with four degrees of freedom has only rotary joints (type R), which classifies it as an articulated robot with a spherical configuration. The structure consists of:

- the swivel joint of the base (q_1), responsible for changing the direction of the arm in the horizontal plane (rotation around the Z axis),
- two bending joints (q_2, q_3), which carry out movements in the vertical plane XZ,
- end joint (Q_4), which allows the gripper to rotate around its axis.

Table 2.1: Denavit-Hartenberg parameters. [3]

and	$\boldsymbol{\theta_i}$ *(variable)*	$\boldsymbol{d_i}$ **[mm]**	$\boldsymbol{a_i}$ **[mm]**	$\boldsymbol{\alpha_i}$
1	θ_1	d_1	0	90°
2	θ_2	0	L_2	0
3	θ_3	0	L_3	0

The actual length parameters of the designed manipulator are equal: d_1 = 96 mm, L_2 = 66 mm, L_3= 178 mm.

2.3 Inverse kinematics

Inverse kinematics consists in determining the values of the angles of the robot's joints, which will allow to achieve the given position of the manipulator (effector) tip in space [1]. In contrast to simple kinematics, in which the position of the tip is determined based on the preset angles of rotation, in inverse kinematics the user provides the desired coordinates of the target point and the algorithm calculates the appropriate values for each joint.θ_i

The solution to inverse kinematics is much more mathematically complex and often ambiguous — for a single tip position, there may be more than one possible joint alignment (or no solution at all if the point is outside the robot's workspace).

Simplifying the assumptions of the 4 DOF model

In the case of the designed robot with four degrees of freedom, a simplified kinematic model was adopted, in which the orientation of the effector is not taken into account (joint 4 is only responsible for the rotation of the gripper around its axis). This means that inverse kinematics is limited to determining the angles , $\theta_1\theta_2$ θ_3 , needed to position the tip in the working space. [4]

Calculation procedure

Assuming that the coordinates of the target point (x,y,z) are known, the following approach can be taken:

1. Determination of the angle θ_1 – rotation of the base:

$$\theta_1 = arctan2(y, x)$$

2. Calculation of the projected distance in the XY plane and the displacement in the Z axis:

$$r = \sqrt{x^2 + y^2} \qquad z' = z - d_{pdstawy}$$

3. Application of the cosine theorem to determine the angle of flexion of the "elbow":

$$\cos\theta_3 = \frac{r^2 + z'^2 - L_2^2 - L_3^2}{2L_2L_3}$$

4. Angle calculation θ_2 from geometric relationships:

$$\theta_2 = arctan2(z', r) - arctan2(L_3 sin\theta_3, L_2 + L_3 cos\theta_3)$$

5. Angle θ_4 – any or imposed depending on your needs.

Example of inverse kinematics calculation:

For the specified effector coordinates in the workspace:

x=28,58 mm, y=16,50 mm, z=331,16 mm

and known base and arm lengths:

L1=96 mm, L2=66 mm, L3=178 mm

inverse calculations were performed to determine the values of the hinge angles , , .$\theta_1\theta_2\theta_3$

1. Determination of the angle of rotation of the base:

θ_1, = arctan2(y, x)=arctan2(16,50, 28,58) ≈ 30,00°

2. Calculation of the projected distance in the XY plane and the shift in the Z axis:

$$r = \sqrt{x^2 + y^2} = \sqrt{28{,}58^2 + 16{,}50^2} \approx 33{,}00\,mm$$

$$z' = z - d_{pdstawy} = 331{,}16 - 96 = 235{,}16\,mm$$

3. Calculation of the angle of flexion of the elbow based on the theorem of cosines:

$cos\ \theta_3\ =\ \frac{r^2 + z\prime^2 - L_2^2 - L_3^2}{2L_2L_3} = cos\ \theta_3\ =\ \frac{33^2 + 235{,}16^2 - 66^2 - 178^2}{2\times66\times178} \approx 0{,}867$

$\theta_3 =\ arccos(0{,}867) \approx\ 29{,}99°$

4. Determination of the angle of the arm θ_2:

$\theta_2 = arctan2(z', r) - arctan2(L_3sin\theta_3, L_2 + L_3cos\theta_3)$

$arctan2(235{,}16, 33) \approx\ 82{,}00°$

$arctan2(178 \times sin(29{,}99°), 66 + 178 \times cos(29{,}99°))) \approx\ 22{,}00°$

$\theta_2 = 82{,}00°\ -\ 22{,}00°\ =\ 60{,}00°$

Final results: $\theta_1\ = 30{,}00°$, , .$\theta_2 = 60{,}00°\theta_3 = 29{,}99°$

Verification of inverse kinematics using simple kinematics:

1. Conversion of angles to radians:

$\theta_1 = 30{,}00° = 0{,}5236$ rad, , .$\theta_2 = 60{,}00° = 1{,}0472$ rad$\theta_3 = 29{,}99° = 0{,}5232$ rad

2. X,y,z coordinates:

$x = (L_2 \times cos\theta_2 + L_3 \times cos(\theta_2 + \theta_3)) \times cos\theta_1$

$x = (66 \times cos(1{,}0472) + 178 \times cos(1{,}5704)) \times cos(0{,}5236) \approx 28{,}58\ mm$

$y = (L_2 \times cos\theta_2 + L_3 \times cos(\theta_2 + \theta_3)) \times sin\theta_1$

$y = (66 \times cos(1{,}0472) + 178 \times cos(1{,}5704)) \times sin(0{,}5236) = 16{,}50\ mm$

$z = L_1 + L_2 \times sin\theta_2 + L_3 \times sin(\theta_2 + \theta_3)$

$z = 96 + 66 \times sin(1{,}0472) + 178 \times sin(1{,}5704) = 331{,}16\ mm$

Final Results (FK): *x,* $\approx 28{,}58\ mm\ y\ = 16{,}50\ mm, z\ \ = 331{,}16\ mm$

Coordinates calculated from the values of the hinge angles $\theta_1\theta_2\theta_3$, , . completely coincide with the desired position of the effector. This means that the developed inverse kinematics algorithm works correctly and is compatible with simple kinematics.

Chapter 3

Algorithms for compensating for errors in the movement of an articulated robot

Articulated robots, which are widely used in industry and research applications, are characterized by high flexibility of movement and a wide range of applications. However, due to design inaccuracies, mechanical deformations, assembly errors and limitations of control algorithms, the actual trajectory of the manipulator tip often deviates from the expected one. In order to minimize these deviations, algorithms for compensating motion errors are used [6]

Classification of error compensation methods

In the literature and engineering practice, several main groups of error compensation methods are distinguished:

- **Model compensation** – consists in correcting the parameters of the kinematic model (e.g. arm length) based on calibration.
- **Data-Driven Compensation** – Uses experimental data in the form of a Look-Up Table (LUT) or interpolation. [5]
- **Error regression** – approximation of errors with mathematical functions (e.g. linear regression).
- **Feedback compensation** – dynamic position correction based on sensor data (e.g. cameras, encoders, strain gauges).
- **Adaptive and learning methods** – advanced AI-based approaches that learn errors in real time.

An overview of error compensation methods:

1. Model compensation (correction of kinematic parameters)

One of the simplest methods is to correct the length of the tonearm members in the kinematic model, resulting from e.g. inaccurate measurement or mechanical wear. At the calibration stage, correction factors a, b are determined:

*L1_corr = L1 * a;*
*L2_corr=L2*b;*

The corrected values are then used in inverse kinematics calculations, which allows the mathematical model to be better fitted to the actual robot.

2. Array compensation

The Look-Up Table (LUT) **method** consists in creating a grid of calibration points in the manipulator's workspace, in which the difference between its actual and expected position of the robotic tip (effector) is determined for each point [5]. The resulting table contains **correction errors Δx(x,y),Δy(x,y),** which are stored e.g. in the EEPROM memory of the microcontroller and used to correct the results of inverse kinematics calculations:

$$x_{poprawione} = x + dx(x,y),\ y_{poprawione} = y + dy(x,y)$$

Interpolation between points can be linear, bilinear or based on basic functions. This makes it possible to partially eliminate systematic positioning errors resulting from e.g. arm flexibility, clearance in gears.

Example of a correction calculation:

It was assumed that the expected position of the robot tip (effector): *x = 180 mm, y = 130 mm*

And the actual position of the robot tip (effector): *x = 182.4 mm, y = 128.4 mm*

From the LUT for the point (x,y)=(180,130) positioning errors were read:

dx(180,130) = -2.4mm, dy(180,130) = 1.6 mm

Application of correction:

$x_{poprawione}$ *=182.4+(-2.4)=180mm,* $y_{poprawione}$ *=128.4+1.6=130mm*

3. Error Regression (Math Correction)

In the case where positioning errors change systematically in space (e.g. increase with distance from the center of the system), they can be described by a mathematical function. For example, for linear regression:

$$dx = a_1 x + a_2 y + a_3\ ,\ dy = b_1 x + b_2 y + b_3$$

Coefficients a_i b_i *can* be determined on the basis of measurement data, e.g. in Excel, and then implemented in the Arduino code as position corrections.

Example of calculation for regression adjustment:

Examples of coefficients (e.g. determined from measurements and linear regression in Excel):

$a_1 = 0.01,,$ $a_2 = -0.005$ $a_3 = -1.2$

$b_1 = 0.002,,$ $b_2 = -0.008$ $b_3 = 0.6$

For the expected point x=150 mm, y=120 mm - calculation of corrections:

$dx = 0.01 \times 150 + (-0.005) \times 120 + (-1.2) = -0.3\, mm$

$dy = 0.002 \times 150 + (-0.008) \times 120 + 0.6 = 1.86\, mm$

Application of correction (x,y)=(150.3, 118,14):

$x_{poprawione} = 150.3+(-0.3) = 150\ mm$, $y_{poprawione}$=118.14+ 1.86 = 120 mm

4. Compensation with feedback loop

An advanced method based on real-time measurement of the effector position. It requires the use of sensors such as:

- cameras with image analysis (e.g. OpenCV),
- strain gauges detecting contact with the object,
- additional encoders on the effector.

The sensor data is used for dynamic position correction by comparing the preset and current tip position.

5. Adaptive/learning algorithms (e.g., neural networks)

These methods use data from multiple robot cycles, learning the relationship between position and error. This allows them to correct non-linear and time-varying errors as well. They require a lot of computing power (e.g. PC, GPU), so they are mainly used in industry or scientific research.

Table 3.1: Comparison of methods.

Method	**Implementation**	**System requirements**	**Type of errors**	**Use**
Model compensation	Very easy	Lack	global, systemic	Prototypes, rapid tests
LUTs (Correction Table)	Easy	EEPROM / RAM	local, nonlinear	Calibration + Arduino
Mathematical regression	Average	Lack	Systematic	Calibration in Excel
Feedback	Difficult	Sensors	Dynamic	Industry, Camera Applications
Machine learning algorithms	Very difficult	PC / GPU	non-linear, variables	Industry, research robots

Chapter 4

Description of the articulated robot

4.1 Mechanical Design

The designed articulated robot has four degrees of freedom (4DOF) and was made on the basis of a set of commercially available metal arm elements. The manipulator consists of a base, a main arm, a forearm and a gripper. The movement is carried out exclusively by rotary joints (type R), which classifies the structure as an articulated robot with a spherical configuration.

The main components of the robot are shown in the following pictures:

The base q1 (figure 4.1) is made of bonded flat aluminum plates. The base is fitted with the first PowerHD LF-20MG servo, which allows the entire arm to rotate around a vertical axis.

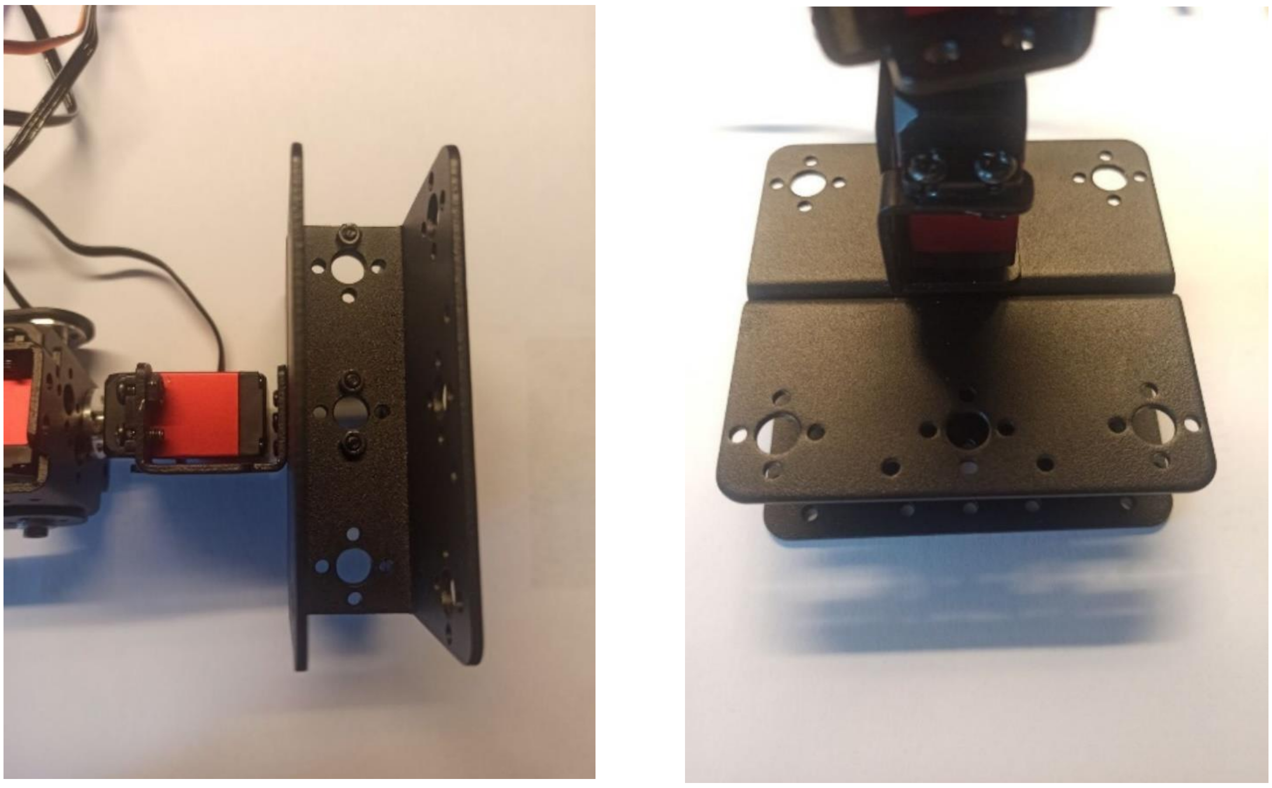


Figure 4.1: Pictures of the robot base.

The main arm q2 (Figure 4.2) is attached to the base and rotated by the second LF-20MG servo, set in portrait orientation. The design is based on rectangular arms with mounting holes. The photos also show the fasteners and screws connecting the load-bearing elements.

The gripper forearm connects to the main arm via the q3 elbow joint (Figure 4.2), driven by another LF-20MG servo.

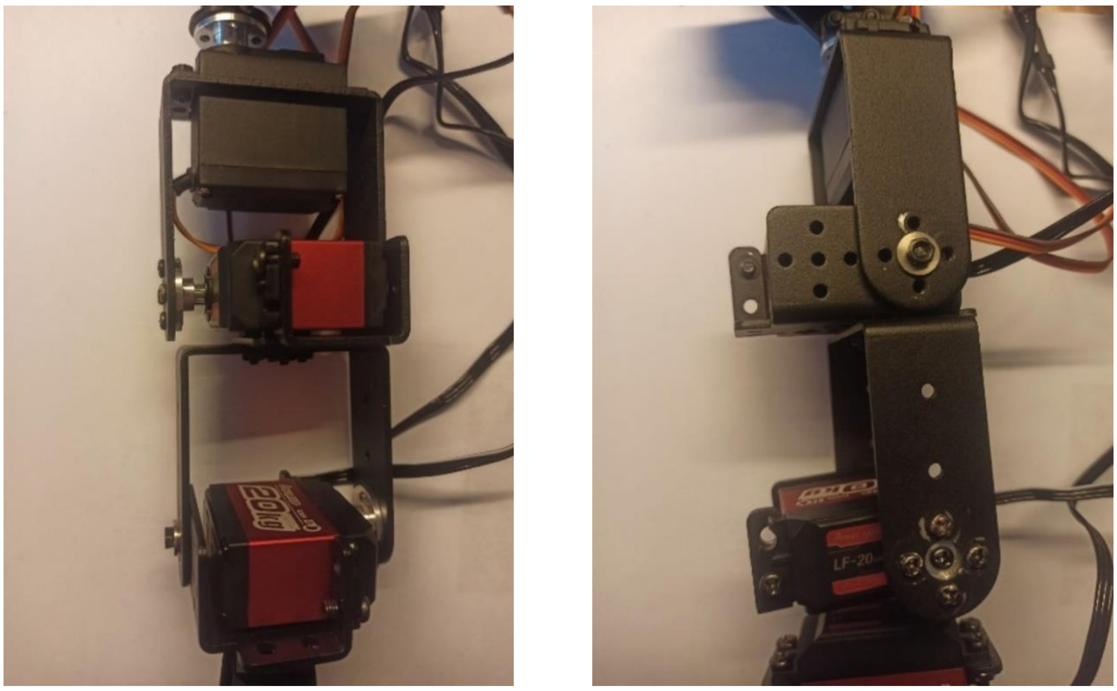


Figure 4.2: View of the main arm and elbow joint with servos mounted.

The gripper (Figure 4.3) is the end of the robot and is mounted on the axis of the joint q4. The rotation is carried out by the MG996R servo, while the grip itself is carried out by a mechanism with two jaws with a limited angle of movement.

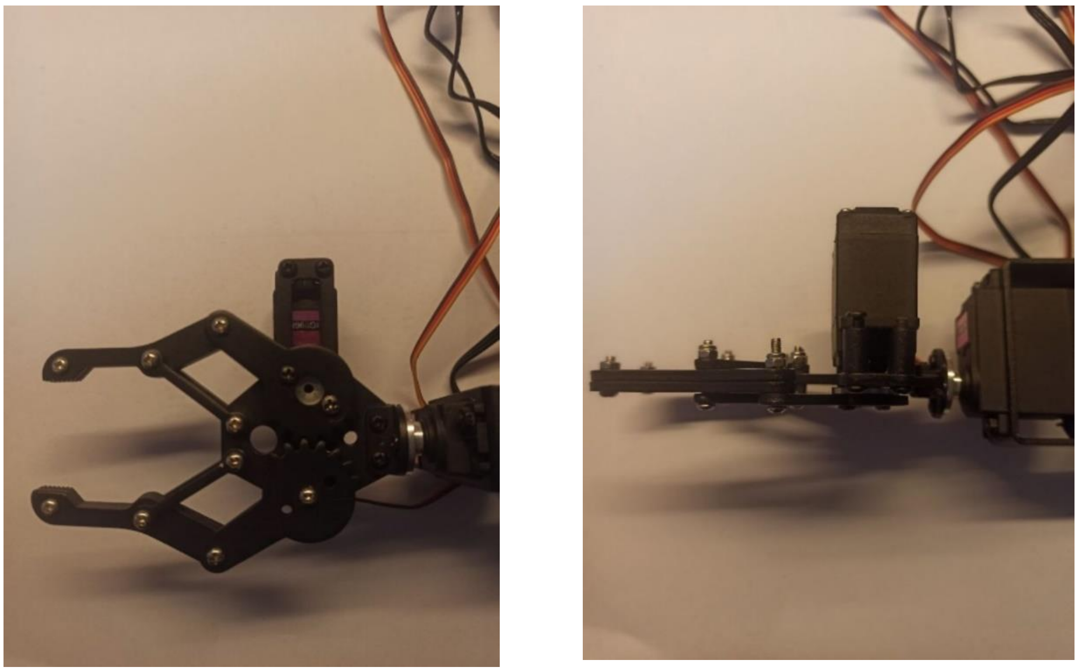


Figure 4.3: Gripper of an articulated robot.

Each joint has independent control by means of PWM signals. Three LF-20MG and two MG996R servos were used for the drive.

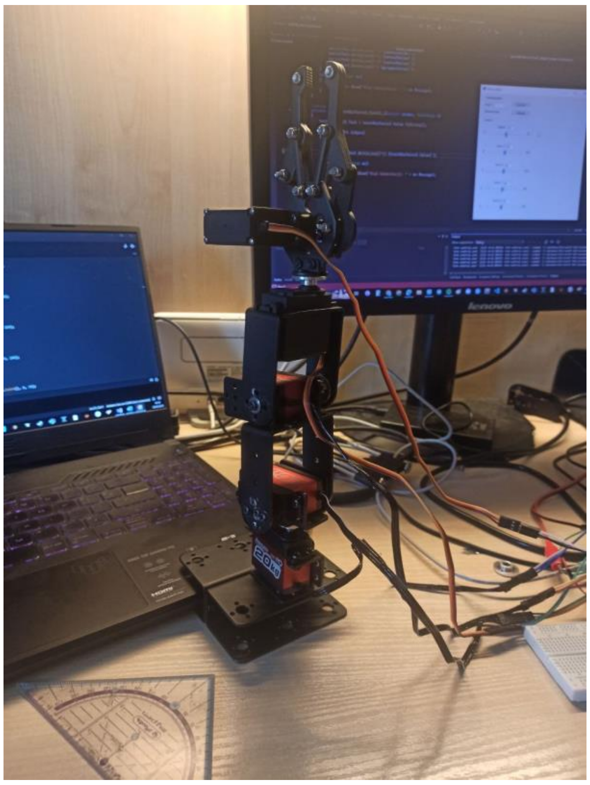


Figura 4.4: A photo of a robot's structure.

4.2 Drive

The first version of the articulated robot design was equipped exclusively with MG996R servos, due to their wide availability and low cost. However, tests have shown that these servos do not provide enough torque in the joints with the most stressed loads – especially in the base and joint of the arm. During the tests, there were drops in performance, stops in the middle of the movement and a characteristic "buzzing" resulting from the overload of the engine. For this reason, it was decided to replace the MG996R with more powerful PowerHD LF-20MG servos in three main joints: q_1 (base rotation), q_2 (arm) and q_3 (elbow). The LF-20MG servos provide much higher torque (up to 20 kg·cm), which allows stable movements with a full load on the arm. Two MG996R servos are used in the end joint – one for turning the gripper, the other for the jaw clamp – due to its lower torque requirements and limited range of travel.

Table 4.1: shows an overview of the servos used and their parameters.

Wrist	Servo Model	Range of motion	Torque	Power
Q1	LF-20MG	0–270°	20 kg·cm	6.0–7.4 V
Q2	LF-20MG	0–270°	20 kg·cm	6.0–7.4 V
Q3	LF-20MG	0–270°	20 kg·cm	6.0–7.4 V
Q4 (turnover)	MG996R	0–180°	10 kg·cm	5.0–6.0 V
Q4 (Clamp)	MG996R	0–180°	10 kg·cm	5.0–6.0 V

4.3 Robot Electronics

The robot is controlled by an Arduino Uno R3 microcontroller system, extended by the Sensor Shield V5.0 module, which provides convenient connection of servos and a communication module. The entire robot system is powered by two independent sources. The first is the LongWei PS3010DF [9] laboratory power supply, set to 6.1 V DC, which supplies power to the Sensor Shield V5.0 – an expansion board that distributes voltage to all servos and a Bluetooth communication module. The power cables are routed directly to the screw power connector on the Sensor Shield. The second source is a computer that supplies 5 V via the USB interface, directly powering the board of the Arduino Uno R3 microcontroller. Such a division of sources allows for the separation of the power supply of the logic microcontroller from the power supply of the executive part (servos and Bluetooth module), which translates into the stability of the entire system.

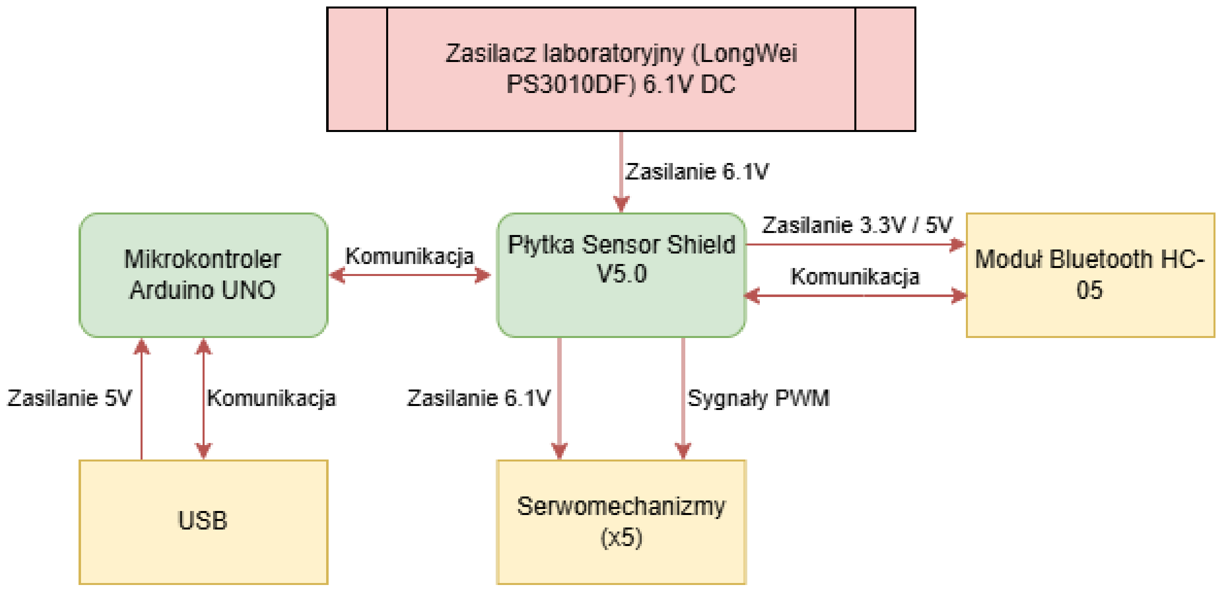


Figure 4.5: Schematic diagram of a robot's electronic circuit.

The system uses two communication interfaces:

- **USB**, which allows the robot to be directly connected to a computer for programming and testing,
- **Bluetooth**, implemented by the HC-05 module, which is connected to the TX/RX line on the Sensor Shield and allows wireless control of the robot from the mobile application.

4.3.1 Power Supply

From the Sensor Shield, the voltage of 6.1 V goes to:

- PWM servo connectors (which require a voltage in the range of 5–7.2 V),
- HC-05 Bluetooth module, which can be powered by up to 6 V (VCC),
- common ground (GND), shared with the microcontroller.

To ensure proper communication between the Arduino Uno (5 V logic) and the HC-05 Bluetooth module (3.3 V logic), a voltage divider made of two 1 kΩ resistors is used, connected to the TX line from the Arduino and to the RX input of the Bluetooth module. This divider lowers the signal level from 5 V to approximately 2.5 V, which is within the safe input voltage range of the HC-05 module and prevents damage to it. Although the voltage divider used causes a small power loss (approx. 12.5 mW), it is sufficient for a communication signal with low current.

4.3.2 Communication-

The keypad control system uses two basic communication interfaces: wired (USB) and wireless (Bluetooth). This configuration allows flexible control of the robot arm from both a computer and a mobile device.

Microcontroller – Arduino UNO R3 [7]

To perform the tasks of controlling the keypad, the popular Arduino UNO R3 development board, based on the ATmega328P microcontroller, was used. This device was chosen because of its ease of use, universality and wide availability of libraries and documentation.

Basic parameters of Arduino UNO R3:

- 1-core ATmega328P microcontroller (16 MHz frequency),
- 32 KB Flash, 2 KB SRAM, 1 KB EEPROM,
- 14 digital inputs/outputs (including 6 PWM),
- 6 analog inputs,
- interfaces: UART, I2C, SPI,
- power supply: 5 V (with USB or external via Vin),
- built-in USB-UART converter (ATmega16U2).

The Arduino UNO serves as the main controller of the system in this project, receiving data via the UART and controlling the servos via PWM outputs

Wired communication – USB

During configuration and testing, the microcontroller communicates with the PC via a USB-B interface, providing a 5 V logic power supply and a serial connection (UART) at the same time. This allows for communication with a user interface created as a computer application, as well as programming the system from the Arduino IDE environment.

Wireless communication – Bluetooth HC-05 [8]

To control the keypad from the mobile application, the HC-05 Bluetooth module was used, operating in slave mode. This module communicates with Arduino using a UART interface (RX/TX), enabling wireless data exchange.

Basic features of the HC-05 Bluetooth module:

- communication interface: **UART** (3.3 V logic),
- supply voltage: **3.6–6 V** (powered by Sensor Shield),
- operating mode: **slave** or **master** (slave by default),
- **Bitrate: 9600 bps** (default)
- Built-in voltage regulator and connection status LED.

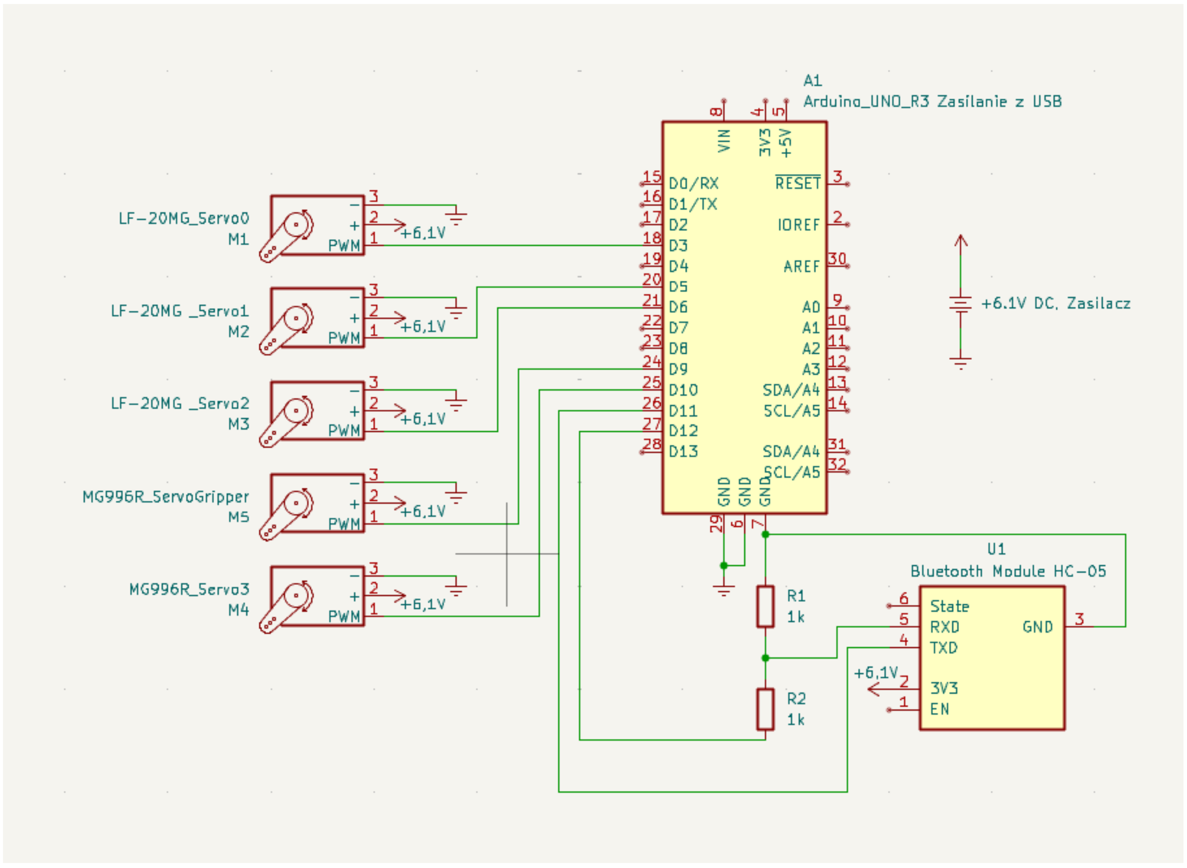


Figure 4.6: Electronic diagram of the robot system.

Table 4.2 of the Arduino UNO microcontroller pins.

Pin Arduino UNO	I/O	Functionality
D3	About	PWM signal – joint servo 0
D5	About	PWM signal – joint servo 1
D6	About	PWM signal – joint servo 2
D10	About	PWM signal – joint servo 3
D9	About	PWM signal – gripper servo
D11	About	RX Arduino ← TX Bluetooth (Direct)
D12	And	TX Arduino RX Bluetooth (via voltage divider)→

4.4 Software

4.4.1 Software structure and libraries used

The keypad control software has been implemented in the Arduino IDE environment using the C++ language and the standard library for the Arduino platform. The main task of the code was to process control commands, perform kinematic calculations (simple and inverse), control servos and optionally compensate for errors in the position of the manipulator tip.

Two libraries were used for hardware operation:

- Servo.h – a library used to control PWM servos,
- SoftwareSerial.h – a library that allows for additional serial communication (in this case to support the HC-05 Bluetooth module).

The entire software consists of five main components:

1. **System initialization (setup())** – configuration of servo pins, activation of communication interfaces and setting the initial position of the arm.
2. **Main program loop (loop())** – monitoring communication channels (USB and Bluetooth) and responding to received commands.
3. **HandleCommand(String command) function** – analyzes the received control command, reads the appropriate values of the set angles and passes them to the appropriate servos.
4. **Kinematics Features**:
    - forwardKinematicsWithoutQ4() – calculates the position of the robot tip based on the given angles of the joints,
    - inverseKinematicsWithoutQ4() – determines the angles of rotation of the joints needed to achieve the given position of the tip.

5. **The applyCompensation() function** – optionally modifies the calculated joint angles, taking into account manually selected corrections to compensate for actual positioning errors.

All functions have been integrated with each other in a way that allows both manual control of the arm (using sliders in the interface or commands via Bluetooth) and automatic control based on spatial coordinates.

4.4.2 Communication and command handling

The robot is controlled by a simple text protocol transmitted via a serial interface. The controller program is equipped with the ability to receive data from two communication sources:

- USB (Serial) – used for communication with a computer and for local tests,
- Bluetooth (SoftwareSerial) – implemented using the HC-05 module, enabling remote control of the robot from the mobile application.

The code implements support for both channels as part of the loop() function. The read lines of text are analyzed by the handleCommand(String command) function, which is responsible for recognizing the content of the command and performing appropriate actions.

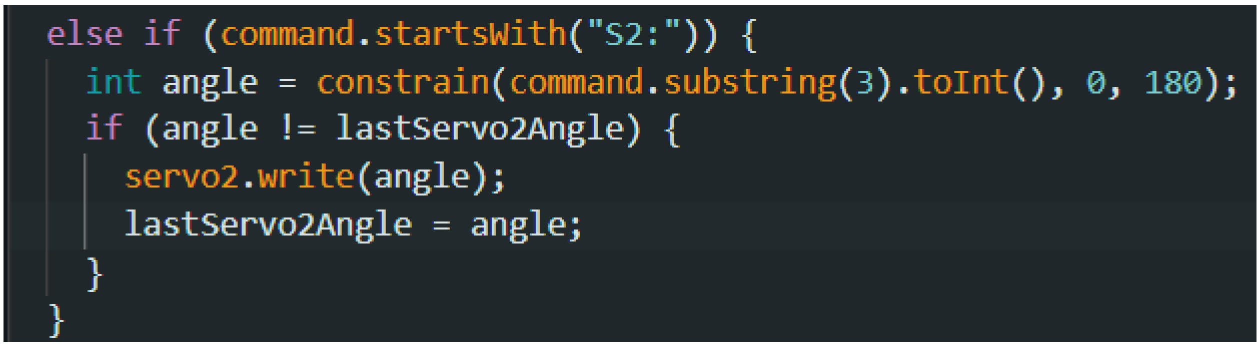
Format komend

The commands sent are in the form of an abbreviation denoting a given joint and a colon and angle values:

Table 4.3: Description of commands.

Command	**Description**	**Value Range**
S0:90	Base rotation	0–180
S1:90	Joint angle 1	0–180
S2:90	Joint angle 2	55–150 (limited range)
S3:90	Joint angle 3	0–180
G:45	Setting up the gripper (gripper)	0-70 (limited range)

Example: The command S2:120 sets the second servo (arm) to 120 degrees.



```
else if (command.startsWith("S2:")) {
  int angle = constrain(command.substring(3).toInt(), 0, 180);
  if (angle != lastServo2Angle) {
    servo2.write(angle);
    lastServo2Angle = angle;
  }
}
```

Figure 4.7: Arduino code snippet – S2 command support for joint 2.

Operation and security

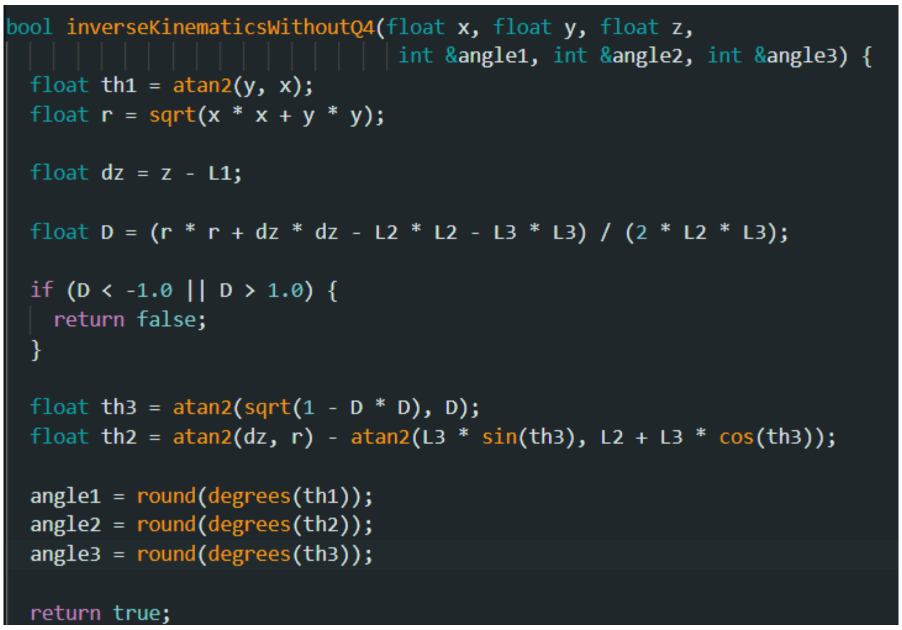
Each command is analyzed for syntax and value correctness. The program includes a constrain(...) function that limits the range of possible servo angles to safe values to avoid mechanical damage:

int angle = constrain(command.substring(3).toInt(), 0, 180);

In addition, lastServoXAngle variables have been used, which prevent unnecessary sending of commands to the servo if the angle value has not changed. This prevents arm vibration and extends the service life of the servos.

Positional Commands (XYZ)

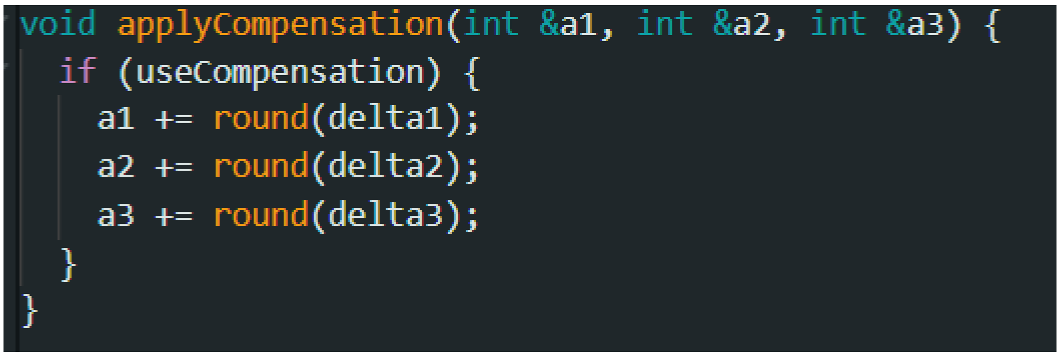
In addition to the direct setting of angles, the system also allows you to send spatial coordinates (x, y, z), on the basis of which the appropriate joint angles are calculated using the inverseKinematicsWithoutQ4() function. The calculated angles can then be automatically transferred to the servos.



```
bool inverseKinematicsWithoutQ4(float x, float y, float z,
                                int &angle1, int &angle2, int &angle3) {
  float th1 = atan2(y, x);
  float r = sqrt(x * x + y * y);

  float dz = z - L1;

  float D = (r * r + dz * dz - L2 * L2 - L3 * L3) / (2 * L2 * L3);

  if (D < -1.0 || D > 1.0) {
    return false;
  }

  float th3 = atan2(sqrt(1 - D * D), D);
  float th2 = atan2(dz, r) - atan2(L3 * sin(th3), L2 + L3 * cos(th3));

  angle1 = round(degrees(th1));
  angle2 = round(degrees(th2));
  angle3 = round(degrees(th3));

  return true;
```

Figure 4-8: Arduino code snippet implementing the inverseKinematicsWithoutQ4 function.

If error compensation is active, the additional applyCompensation() function makes adjustments to the calculated values, ensuring higher positioning precision.



```
void applyCompensation(int &a1, int &a2, int &a3) {
  if (useCompensation) {
    a1 += round(delta1);
    a2 += round(delta2);
    a3 += round(delta3);
  }
}
```

Figure 4.9: Arduino code snippet implementing error compensation.

4.4.3 Developed control applications

Desktop application (.NET Framework)

In order to enable convenient control of the keypad from a computer, a desktop application in C# using .NET Framework technology was designed. The app allows the user to intuitively issue commands and control the position of servos via a graphical interface.

Main functionalities:

- control of five servos (S0-S3 and gripper) with sliders (TrackBar) and text fields (TextBox),
- dynamic synchronization of values between sliders and text fields,
- selection of the communication port (e.g. COM3) from the drop-down list and initialization of the connection with the Arduino microcontroller,
- Refresh button to refresh the list of available ports in real time,
- information about the connection status (Connected/Disconnected),
- the ability to change all angles at once by entering values in the text fields and clicking the Update button (after connecting to the device, the Refresh button changes to Update, allowing all set values from the text fields to be sent at the same time.).

The application is designed with simplicity and clarity in mind, which makes it useful both for experimental testing and for demonstrating the system in laboratory conditions.

The following is a screenshot of the running desktop application (Figure 4-10):

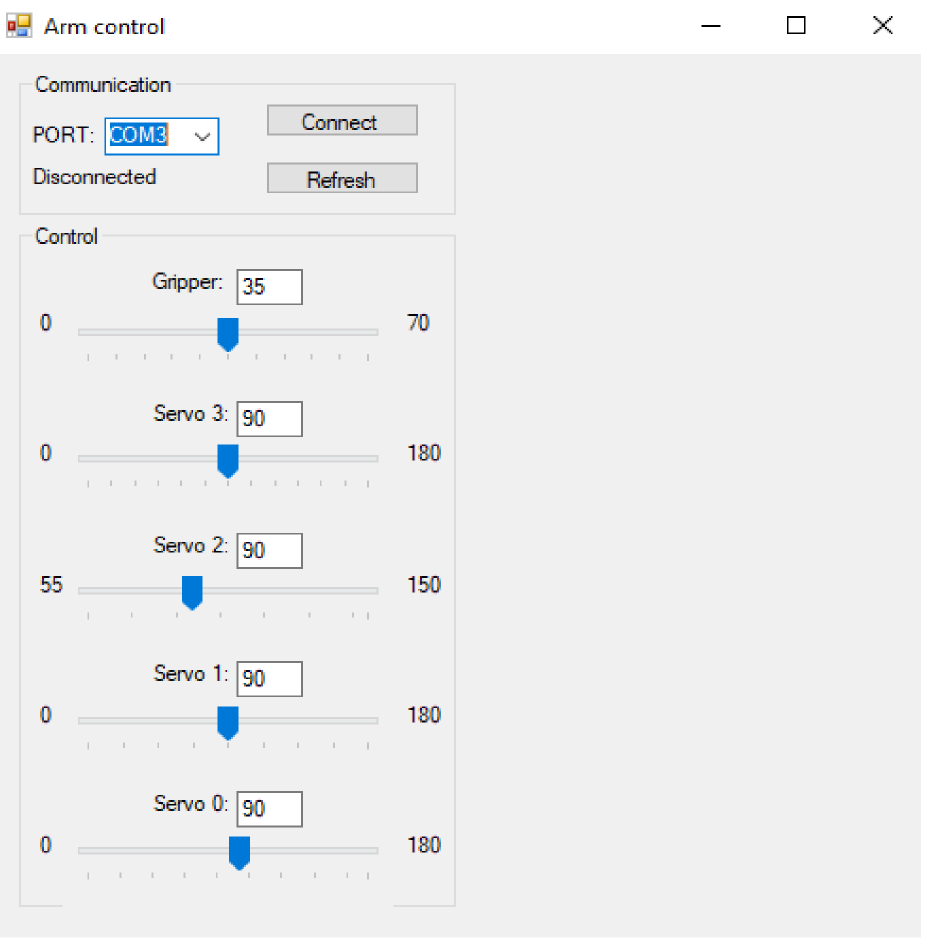


Figure 4.10: Desktop application (.NET Framework).

The application logic was placed in the Form1 class, inheriting from Form. Serial communication has been implemented using the SerialPort class, which allows opening, closing and sending data to the COM port.

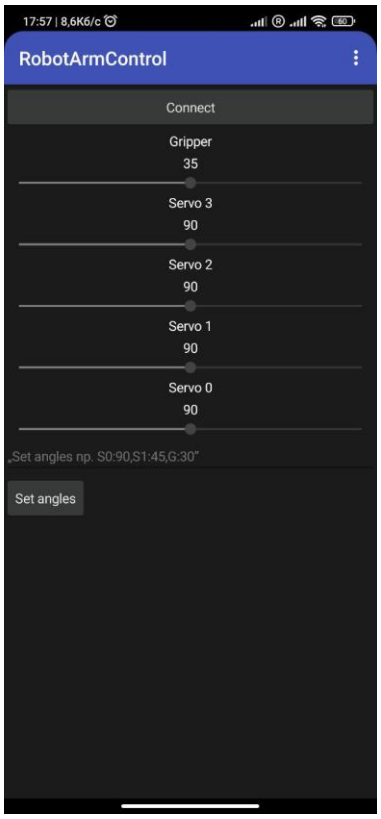
Functions and elements of the application:

- COM port initialization: The LoadAvailablePorts() method automatically reads the available ports and populates the comboBoxPorts drop-down list.
- Connection to Arduino: when you click Connect, the program opens the selected COM port with a baud rate of 9600 baud. The connection status is displayed in the form of a colored message.
- Servo control: each of the five robot joints (including the gripper) is controlled by:
    - slider (TrackBar) – by moving the slider, the user changes the angle value,
    - TextBox – by typing the angle value directly from the keyboard. Assigned events (Scroll, KeyPress) synchronize the values of both controls and send the appropriate command through the serial port.

All serial transmission operations are covered by try-catch blocks, which allows you to handle potential transmission errors, incorrect data, and port disconnections.

Mobile app (MIT App Inventor)

In order to enable wireless control of the manipulator from a mobile device, a mobile application was developed using the **MIT App Inventor** environment. The interface allows commands to be transmitted via the HC-05 Bluetooth module and allows the user to control five robot servos (four axes + gripper).



Gripper 35

Servo 3 90

Servo 2 90

Servo 1 90

Servo 0 90

Figure 4.11: Mobile app (MIT App Inventor).

Application structure and operation logic

The application uses two ways to transfer data to the microcontroller:

- **Direct slider control** – each movement of the slider causes an immediate text command (e.g. S1:90) to be sent via BluetoothClient.SendText.
- **Text field and "Set angles" button** – the user can manually enter values in protocol format (e.g. S0:90; S1:90;...) and send the entire setup sequence at once.

Key components and their functions:

- **ListPicker + BluetoothClient:** Used to search for and connect to available Bluetooth devices (previously paired with Android).
- **Sliders + Labels** – are responsible for sending current values of servo angles and their visualization.
- **TextBox** – allows you to manually enter a sequence of commands in a given format.

- **"Set angles" button** – parses the entered data, sets the values of the sliders and sends them via Bluetooth.
- **Split, compare texts, get item, and select list item blocks** – implement a text command parser to extract the values assigned to a given servo.

Arduino reads data from the serial port, parses it, and sets the angles of the servos.

Chapter 5

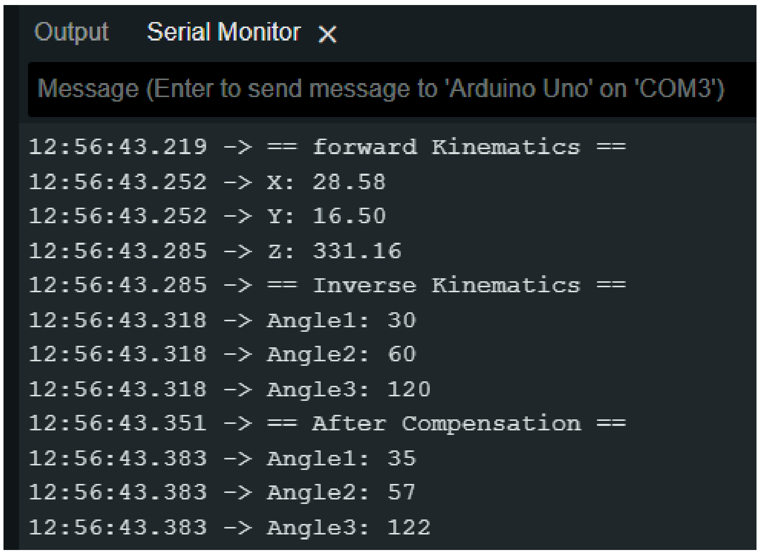
Commissioning and testing

5.1 Verification of kinematics calculations

In order to check the correctness of the implemented functions of simple and inverse kinematics, simulation and experimental tests were carried out. Below is an example of a verification case in which a full cycle of recalculation of the coordinates of the manipulator tip was performed. In contrast to the theoretical calculations for the robot's kinematic model, in the implementation of the code, 90° was added to the third angle (angle3) at the end of the calculation, because in the real system the zero position of this joint corresponds to the value of 90°, additionally taking into account the reversed direction of rotation.

The following screenshot (Figure 5.1) shows the program logs from the test run:

- Tip position calculations,
- Reverse kinematics,
- Transformed angles after compensation.



```
12:56:43.219 -> == forward Kinematics ==
12:56:43.252 -> X: 28.58
12:56:43.252 -> Y: 16.50
12:56:43.285 -> Z: 331.16
12:56:43.285 -> == Inverse Kinematics ==
12:56:43.318 -> Angle1: 30
12:56:43.318 -> Angle2: 60
12:56:43.318 -> Angle3: 120
12:56:43.351 -> == After Compensation ==
12:56:43.383 -> Angle1: 35
12:56:43.383 -> Angle2: 57
12:56:43.383 -> Angle3: 122
```

Figure 5.1: Fragment of the controller log illustrating the correct operation of kinematic calculations.

For the given joint angles: **Angle1 = 30°**, **Angle2 = 60°** and **Angle3 = 120°**, forward kinematics calculations were performed, obtaining the position of the robot tip in space: **X = 28.58 mm**, **Y = 16.50 mm**, **Z = 331.16 mm**. Then, on the basis of this position, inverse kinematics calculations were performed, which returned exactly the same values of angles: **Angle1 = 30°**, **Angle2 = 60°**, **Angle3 = 120°**, which confirms the correctness of the implemented functions forwardKinematicsWithoutQ4() and inverseKinematicsWithoutQ4().

In order to increase the precision of motion reproduction, the operation of an error compensation algorithm is also taken into account. The corrected values of the joints were: **Angle1 = 35°**, **Angle2 = 57°**, **Angle3 = 122°**, which corresponds to the application of the corrections: +5°, −3°, +2°, respectively. This reduces the impact of actual deviations due to mechanical clearances, servo inaccuracies or miscalculations. The experiment confirms that in the simulation environment, deviations are marginal, but in a real system, the use of a compensation algorithm can significantly improve the positioning accuracy of the robot effector.

5.2 Experimental verification of the error compensation algorithm

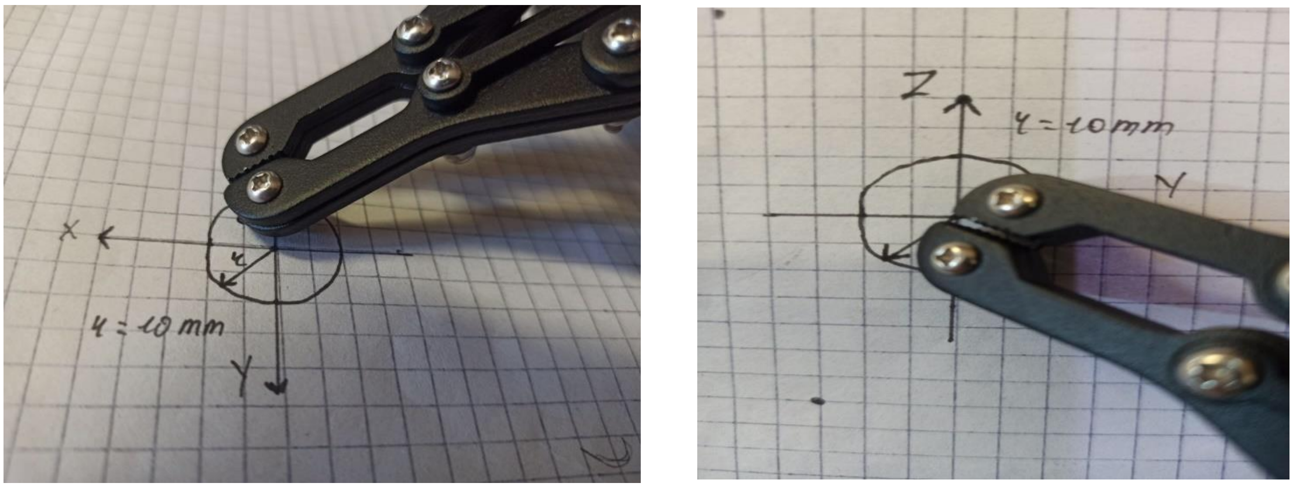
The purpose of this chapter is to present the process of experimental verification of the effectiveness of the implemented algorithm for compensating for errors in the position of the manipulator tip. This compensation was accomplished by adding appropriate angular corrections to the values determined on the basis of inverse kinematics. **Methodology:**

Test PointsA set of TCP targets in the workspace is selected.

Two measurement modesFor each point, two arm positioning is performed: without compensation and
with compensation enabled.

Measurements of the actual position of the TCPThe position of the tip was verified using a reference circuit (Figure 5.2)

Error measurementFor each position, the deviation of the actual TCP position relative to the



destination point was measured.

Figure 5.2: Measurements of position inaccuracies.

Table 5.1: Results of measurements of position inaccuracy.

Point	P1	P2	P3
Preset (x, y, z) [mm]	(100, 130, 0)	(50, 210, 10)	(0, 180, 130)
Actual Uncompensated (x, y, z)	(104, 135, 0)	(55, 216, 7)	(4, 185, 133)
Error [mm]	(4, 5, 0)	(5, 6, 3)	(4, 5, 3)
Relative error [%]	[4.0, 3.85, 0]	[10.0, 2.86, 30.0]	[0, 2.78, 2.31]
Actual compensated (x, y, z)	(101, 133, 0)	(52, 213, 9)	(2, 183, 132)
Error [mm]	(1, 3, 0)	(2, 3, 1)	(2, 3, 2)
Relative error [%]	[1.0, 2.31, 0]	[4.0, 1.43, 10.0]	[0, 1.67, 1.54]

Based on the results in Table 5.1, it can be seen that the use of the positional error compensation algorithm effectively reduces inaccuracies in the mapping of the given robot tip coordinates.

For all three endpoints tested, there was a clear reduction in errors in each of the spatial directions (x, y, z). For example, in the case of point P1, the error in the X-axis decreased from 4 mm to 1 mm, in the Y-axis from 5 mm to 3 mm, while in the Z-axis the position was maintained without deviation. Similar improvements were also observed at points P2 and P3, where the total spatial error was reduced by up to half.

5.3 System Power Measurements

In order to determine the energy demand of the robot and to check the correctness of the power supply selection, a series of measurements of the voltage and current consumed by the system in various operating states were carried out.

The measurements were carried out using a LongWei PS3010DF laboratory power supply with an adjustable output voltage in the range of 0–30 V and a maximum current of 10 A. The measurement was carried out at an operating voltage of 6.1 V, using an external meter and the indications of the built-in display.

Measurement conditions:

- Supply voltage: 6.1 V (manually set),
- Load: 5 servos (LF-20MG and MG996R),
- Current measurement at different states: idle, single servo movement, gripper full load.

Table 5.2: Measurement of current and power consumption by the robot system.

Operating Mode	**Voltage [V]**	**Current ± ΔI [A]**	**Relative error [%]**	**Power [W]**	**Comments**
Idle state (all servos stopped)	6.1	0.18 ± 0.01	5,56	1.042	Hold position only
Single servo movement (q1)	6.1	1.35 ± 0.01	0,74	4.578	No mechanical load
Movement of three servos at the same time	6.1	1.71 ± 0.01	0,58	10.255	Loaded
Movement of the gripper (gripper) with a load	6.1	0.36 ± 0.01	2,78	2.098	Gripping a small object
Locked joint + motion test	6.1	1.06 ± 0,01	0,94	6.025	Peak current

The maximum measured power consumed by the robot was about 10,255 W at 6.1 V. The power supply system must provide a stable power supply even with temporary increases in power consumption. Capable of delivering up to 10 amps, the PS3010DF power supply meets the requirements of the system and provides an adequate safety margin.

5.4 Comparison of the parameters of the designed robot with existing solutions

Compared to the compared designs, the designed robot is distinguished by the lowest curb weight (0.578 kg), compact arm reach (~230 mm) and good repeatability (estimated ±.3 mm). Despite using a simple Arduino UNO R3 controller, the robot allows remote control and performs operations comparable to more expensive educational models. The design parameters and performance confirm that with the right design and implementation, it is possible to create a functional 4DOF keypad for educational and demonstration applications at low cost.

Table 5.3: Basic parameters of the designed 4DOF robot.

Parameter	**Designed 4DOF robot**	**RoArm-M2-S**	**Multifunction Robotic Arm – IADIY**	**Igus® Robolink® DP – 4DOF**
Number of axles	4	4	4	4
Maximum load capacity	0.15 kg	0.5 kg	0.2-0.3kg	2-3 kg
Arm reach	Approx. 230 mm	210 mm	350-500mm	790 mm
Repeatability	~±3 mm (est.)	~±1 mm (est.)	~±0.5 mm (est.)	~±0.3 mm (est.)
Scales	0.578 kg	0.83 kg	approx. 1.2 kg	~5–6 kg
Controller	Arduino UNO R3	ESP32, ROS2	Arduino	igus® Robot Control, ROS

Chapter 6

Summary

The aim of this thesis was to design, manufacture and program a manipulator with four degrees of freedom (4 DOF), enabling the control of the effector in 3D space, using the developed inverse kinematics algorithm and error compensation mechanism. The assumptions of the work have been fully implemented, and the final effect confirms the functionality of the designed system.

The project modified the off-the-shelf 6DOF robotic arm to four degrees of freedom, greatly simplifying the mathematical model and enabling the successful application of a simplified approach to inverse kinematics. One of the challenges was the need to select the right drive components – the initially used MG996R servos were replaced by more powerful and precise LF-20MG models. At the same time, unstable voltage converters were abandoned in favor of a professional LongWei PS3010DF laboratory power supply, which significantly improved the reliability of the system.

The designed system was based on the Arduino UNO microcontroller, to which all servos and the Bluetooth module (HC-05) were connected, enabling wireless communication with the user. Two control applications were also developed – mobile and computer. The desktop interface was made in .NET Framework technology, which allowed for intuitive operation of the keypad and enabled testing of serial communication in real time.

As part of the work, simple and inverse kinematics algorithms were implemented, using the geometric model of the manipulator, and an error compensation function based on the measurement results was designed. The experiments showed that the use of compensation significantly improves the positioning accuracy – in the analysed cases, the error decreased from 5–6 mm to about 2–3 mm.

The measurement of current consumption showed that the designed system is characterized by low power consumption – the maximum consumption was about 10.26 W with simultaneous movement of three axes with a load in the form of a small object weighing about 150 g, while in the idle state the power consumption was limited to about 1 W.

Literature

[1] Kozłowski Krzysztof, Dutkiewicz Piotr, Wróblewski Waldemar; Modeling and control of robots. Wydawnictwo Naukowe PWN, 2016
[2] Krzysztof Tchoń, Robert Muszyński, Robotics, Notes for lectures in the field of automation and robotics, Department of Cybernetics and Robotics, Wrocław University of Science and Technology, Wrocław 2018
[3] Addison, A. (2020). *How to Find DenavitHartenberg Parameter Tables*. Automatic Addison.
[4] Mohammed, A. A. M., & Sunar, M. (2015). *Kinematics Modeling of a 4DOF Robotic Arm* [Conference paper]. King Fahd University of Petroleum & Minerals.
[5] Górski, F., Płuciennik, M., & Wichniarek, R. (2019). *Modeling and compensation of robot errors using LUTs and interpolation functions*. Measurements Automation Robotics, 23(3), 123–129.
[6] Dutka, A., & Winiarski, J. (2016). Factors affecting the accuracy and repeatability of manipulator positioning. *Measurements Automation Robotics*, 4, 55–61.
[7] Technical documentation of the Arduino UNO R3 microprocessor chip https://docs.arduino.cc/hardware/uno-rev3
[8] HC-05 Bluetooth Technical Documentation https://components101.com/sites/default/files/component_datasheet/HC-05%20Datasheet.pdf
[9] LongWei PS3010DF Laboratory Power Supply Manufacturer's website https://longweielec.com/product/ps-3010df
[10] RoArm-M2-S website https://www.waveshare.com/wiki/RoArm-M2-S
[11] Strona internetowa robota 4DOF Multifunction Robotic Arm for Desktop od IADIY https://www.iadiy.com/4-DOF-Multifunction-Robotic-Arm-for-Desktop
[12] Website of the Igus® robolink® DP – 4DOF https://www.igus.com/product/20232?srsltid=AfmBOor4RHztOVnHyrTYBwJVae3khTI9mcCTffbJ2TVxGjUfwJZMzGpU&utm_source=chatgpt.com&artNr=RL-DP-4