Sofia University "St. Kliment Ohridski"
Faculty of mathematics and informatics
Department of Computer informatics

Modelling and Control of an Anthropomorphic Robot Arm

Thesis Abstract

Lyubomira Lachezarova Miteva

for the educational and scientific degree of Doctor of Philosophy in
4.6. Informatics and Computer Science

Doctoral program:
"Information systems" – Embedded and autonomous systems

Supervised by:

Prof. Evgeniy Krastev, PhD
Assoc. Prof. Ivan Chavdarov, PhD

Sofia, 2023

The dissertation was discussed and directed to the defense of an extended meeting of the Department of Computer Informatics at the Faculty of Mathematics and Informatics of Sofia University "St. Kliment Ohridski", held on 06.02.2023.

The author is a PhD student at the Faculty of Mathematics and Informatics of Sofia University "St. Kliment Ohridski".

The thesis has a volume of 132 pages, which contain 75 figures, 11 tables, 98 sources are cited. The list of publications of the author on the essence of the dissertation includes 6 titles.

The public defense of the dissertation will take place at an open meeting on …….. from …….. hours in …….

The required materials are available in …….. of FMI at SU (Sofia, James Boucher Blvd. 5).

Contents

General characteristics of the dissertation 4
Relevance of the problem and motivation 4
Purpose and objectives of the dissertation 5
Structure and content of the dissertation....5
Chapter 1. Overview of the subject area....5
Chapter 2. Modelling of a planar anthropomorphic robot arm....9
Chapter 3. Anthropomorphic robot arm control 15
Chapter 4. Experimental verification using a 3D printed robot arm 24
Conclusion 29
References....30
Contributions to the dissertation 33
Applied scientific contributions....33
Applied contributions....33
Publications, reports and participation in projects related to the topic of the dissertation 34
Publications....34
Conference presentations 34
Scientific projects....35
Declaration of originality....35
Biography of the author of the dissertation....35
Acknowledgments....35

General characteristics of the dissertation

Industrial robots usually perform predefined tasks such as welding, assembling, transportation, etc. Each robot has a different number of independent joints that uniquely determine the position and orientation of its end effector. The number of joints of a robot determines the number of degrees of freedom of this robot. Robots are required to perform precisely and accurately the tasks assigned to them, observing a certain trajectory, and considering the obstacles in their workspace. For this reason, some robots are designed in such a way that they have more degrees of freedom than are required to perform the desired task. This makes them more flexible and increase their productivity [1]. The dissertation discusses and proposes approaches for executing motion for optimal time and trajectory planning in the presence of static and dynamic obstacles in the workspace of a redundant robot.

Relevance of the problem and motivation

The motivation to introduce additional degrees of freedom to the mechanical structure of the robot comes from the goal to increase the reliability of the robotic system and to reduce the probability of error. The presence of additional degrees of freedom allows movements of the manipulator that do not displace the end effector from a certain position. That means that the same position and orientation of the end effector can be performed with different joint configurations, which makes it possible to avoid regions with obstacles and leads to greater flexibility of the manipulator [2]. This characteristic is important to using robotic systems in dynamic and non-deterministic environments. Most redundant robots are used for industrial purposes, such as painting, welding, etc. [3].

The redundant robots can avoid obstacles in their workspace, while performing their main task with their end effector. Despite all the advantages that this type of robot has over robots without additional degrees of freedom, there are some peculiarities to them. The redundant robots have multiple joint configurations that are solutions to the inverse kinematics problem. Having multiple solutions provide an opportunity to choose the most appropriate joint configuration. But this makes more difficult solving the inverse kinematics problem and motion planning. The choice of the most appropriate joint configuration depends on the task assigned to the robot. The solutions of the inverse kinematics problem can be classified into several types depending on the values of the joint angles. The workspace of the robot can also be divided into several zones depending on the existing types of solutions to the inverse kinematics problem. When performing a movement, such as moving the robot's end effector from one point to another, the robot can go from one joint configuration to another in order to make a transition between two zones. But this change of the configuration type can lead to a deviation from the desired trajectory or an increased execution time. Therefore, it is necessary to explore the different areas in the robot's workspace and to propose an approach for finding points at which the robot can change its joint configuration type without deviation from the desired trajectory. In addition, in the presence of obstacles, the transition between some of the zones may be impossible. If the task of the robot is assembly and it is necessary at a certain point in its workspace to change the orientation of its end effector, it is possible that in case of improper planning of its movement, the end effector of the robot to move away from the desired point.

The possible limitations of redundant robots in performing a given motion in the presence of obstacles motivated the creation of this dissertation in order to explore different methods of trajectory planning in an environment with available static and dynamic obstacles. The motion planning is a problem studied for many years and by many scientists, but there is a lack of research in the literature on the classification of solutions to the inverse kinematics problem depending on the values of joint angles and the use of this classification in motion planning.

Purpose and objectives of the dissertation

The aim of this thesis is to create a mathematical model and prototype of a planar anthropomorphic redundant robot arm, as well as to research and create approaches for the motion control. Conducting this research should lead to the creation of algorithms for trajectory planning in the presence of static or dynamic obstacles in the workspace of the robot. To achieve the desired objective, the following tasks can be formulated:

1. Classification of different types of solutions of the inverse kinematics problem depending on the values of the joint coordinates of the anthropomorphic redundant robot.
2. Research and development of trajectory planning algorithms for redundant anthropomorphic robots to avoid static and/or dynamic obstacles and reach a desired position.
3. Creation of a prototype of anthropomorphic redundant robot using the methods of 3D printing.
4. Analyzing and selecting appropriate hardware components and designing a suitable software control system for the created redundant anthropomorphic robot.
5. Verification of the proposed hardware and software control system and trajectory planning algorithms, by computer simulation and experiment with the designed robotic system.

Structure and content of the dissertation

The structure of the dissertation consists of Introduction, 4 chapters and Conclusion and has 132 pages. The results presented in these chapters have been reported at 4 international conferences. The publications are referenced in Scopus.

Chapter 1. Overview of the subject area

The progress of various scientific fields such as mechanics, electronics, informatics, and mathematics make possible the development of robotic systems. Among the advantages of using robots are reduced labor costs, better precision and productivity, better flexibility compared to specialized machines, and replacing humans in repetitive or in dangerous tasks [4]. This chapter discusses the main characteristics of robots with additional degrees of freedom and different methods of motion planning in the presence of obstacles.

Anthropomorphic robot arms with additional degrees of freedom

The number of robots used in an unstructured and dynamic environment where they must work together with other robots or humans has significantly increased. For this reason, robots are being designed in such a way that they have more degrees of freedom than are necessary to perform the tasks assigned to them. The difference of the dimension of the joint space and that of the task space determines the degree of redundancy (additional degrees of freedom) for a given robotic system [5]. Robots with additional degrees of freedom can reach a given position and orientation of the end effector using different joint configurations [2]. These robotic systems are characterized by greater performance and accuracy in the performance of the assigned task [1, 6]. The presence of multiple joint configurations for a given position and orientation of the end effector allows easier avoiding of obstacles than other robotic systems [7]. Robots that possess human characteristics in one way or another are called anthropomorphic robots.

Motion planning of redundant robots is an interesting topic, as a criterion is needed to select an appropriate joint configuration depending on the assigned task. Various methods for selecting an appropriate joint configuration have been researched and proposed in the literature. Hollerbach proposes the selection of such a joint configuration where torque optimization can be achieved [8], Hirakawa and Kawamura present a method for selecting joint configurations that can optimize the energy consumed by the robot [9], Chembuly and Voruganti study a method for obstacle avoidance [10] and Yoshikawa

introduced the manipulability coefficient metric for joint configuration selection [11]. The methods proposed in the literature for choosing an appropriate joint configuration are applicable to the specific objectives such as optimizing the energy consumed by the robot or applying a larger force. The solutions to the inverse problem of such a type of robot can be classified into several types according to the values of the joint angles. Additionally, based on the different types of solutions, the robot's workspace can be divided into different zones. As the robot changes its joint configuration, it may need to move from one zone to another. The proposed methods in the literature do not consider the fact that when the robot needs to move from one joint configuration to another, there is a possibility that it will pass through a singular configuration and deviate from the desired trajectory. This would lead to more time to complete the assigned task, or incorrect execution of the task. Therefore, it is necessary to explore and find the points (zones) in which the robot can make the change in the joint configuration without causing a deviation from the desired movement.

Motion planning methods

The end effector of a robotic system must perform a movement from a given initial position to a certain target position in the workspace, by avoiding static or dynamic obstacles and considering the joint constraints of the robot. Different methods of motion planning have been investigated and presented in the literature. Some of the most popular and used are Dijkstra, A*, Rapidly exploring random tree and probabilistic roadmap methods [12].

The Dijkstra method was proposed in 1959 by Edsger Dijkstra and is used to find the shortest paths between nodes in a given graph [13]. It is considered an effective and efficient method even with graphs with many nodes. The algorithm finds the shortest path from one vertex to another given vertex [14].

The **A*** algorithm is based on the best-first search algorithm and uses a heuristic function to find the shortest path by estimating the total costs [14]. It is suitable for planning movement with static obstacles avoidance, but if the graph with possible paths is too large, the algorithm will lose its effectiveness [15].

The **Rapidly exploring random tree** (RRT) and **Probabilistic roadmap method** (PRM) algorithms are also some of the most popular methods of motion planning [12]. The RRT method is commonly used in various commercial and industrial tasks. The method creates a tree through which the robot's workspace is explored by randomly searching for appropriate joint configurations [16]. The algorithm is suitable for dynamic environments. The PRM method is used more often in a static environment with previously known obstacles [17].

Kinematics

Kinematics studies the position, speed, and acceleration of joint coordinates. The forward and inverse kinematics problem are used to design any robotic system [18].

The goal of the **forward kinematics problem** is to find the position and orientation of the end effector of a robotic system according to given joint coordinates. To solve it, it is necessary to place local coordinate systems in each link of the robotic system. The most common method for selecting coordinate systems for open kinematics chain manipulators is the Denavit-Hartenberg convention (Fig. 1.6), which defines the relational position and orientation of two adjacent links. It is necessary to define the coordinate systems of two adjacent links and to find the coordinate transformation between them. A fixed coordinate system $O_0x_0y_0z_0$ is connected to the center of the base of the manipulator. The axis z_0 coincides with the axis of rotation and/or the translation axis of the first link relative to the base, and the axes x_0 and y_0 are selected so that it is a right oriented coordinate system. The links from the base to the end effector are numbered with $i = 1,2,\dots,N$ and a fixed coordinate system $O_ix_iy_iz_i$ is connected to each link. The Denavit-Hartenberg convention defines the coordinate system of the i-th link. The axis $z_i, i = 1,2,\dots,N-1$ is selected in the direction of the link axis. The axis $x_i, i = 1,2,\dots,N-1$ is determined by the common perpendicular between the axes z_{i-1} and z_i with the positive direction from link i to link $i + 1$. The axis $y_i, i = 1,2,\dots,N-1$ is selected, so that it complements a right oriented

coordinate system. The last coordinate system $O_n x_n y_n z_n$ is fixedly connected to the end effector of the manipulator, with the z_n axis defining its orientation and usually being its axis of symmetry. The remaining axes are determined in the described manner [19].

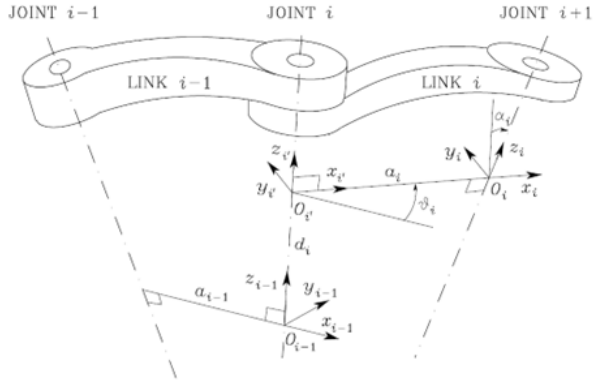

Fig. 1.6 Denavit-Hartenberg parameters [19].

After all local coordinate systems are set, the position and orientation of coordinate system i relative to coordinate system $i-1$ is completely determined by the following parameters: a_i is the distance between O_i and $O_{i'}$ along the x_i axis, d_i is the distance between $O_{(i-1)}$ and $O_{i'}$ along the z_{i-1} axis, α_i is the angle between the z_{i-1} and z_i axes relative to the x_i axis, θ_i is the angle between the x_{i-1} axes and x_i relative to the z_{i-1} axis. To find the coordinate transformation between the coordinate systems $O_{i-1}x_{i-1}y_{i-1}z_{i-1}$ and $O_i x_i y_i z_i$ the following steps consisting of incremental displacements and rotations are performed:

- Rotate an angle θ_i along the z_{i-1} axis until the x_{i-1} and $x_{i'}$ axes become parallel.
- Displacement along the x_{i-1} axis at a distance d_i until the x_{i-1} and $x_{i'}$ axes coincide.
- Displacement along the $x_{i'}$ axis at a distance a_i until the coordinate origin $O_{i'}$ coincides with O_i.
- Rotation of an angle α_i around the axis $x_{i'}$ until all coordinate axes coincide.

The transformation matrix between two coordinate systems can be expressed as a product of four transformations:

$$\boldsymbol{T}_i^{i-1} = \boldsymbol{R}(Z_{i-1}, \theta_i)\boldsymbol{Tr}\left(Z_{i-1,} d_i\right)\boldsymbol{Tr}(X_i, a_i)\boldsymbol{R}(X_i, \alpha_i) \tag{1.1}$$

$$\boldsymbol{T}_i^{i-1} = \begin{bmatrix} cos(\theta_i) & -sin(\theta_i) & 0 & 0 \\ sin(\theta_i) & cos(\theta_i) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos(\alpha_i) & -sin(\alpha_i) & 0 \\ 0 & sin(\alpha_i) & cos(\alpha_i) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1.2}$$

After performing the necessary calculations for the transformation matrix T_i^{i-1} we obtain:

$$\boldsymbol{T}_i^{i-1} = \begin{bmatrix} cos(\theta_i) & -sin(\theta_i)\,cos(\alpha_i) & sin(\theta_i)sin\,(\alpha_i) & a_i\,cos(\theta_i) \\ sin(\theta_i) & cos(\theta_i)\,cos(\alpha_i) & -cos(\theta_i)sin\,(\alpha_i) & a_i\,sin(\theta_i) \\ 0 & sin(\alpha_i) & cos(\alpha_i) & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1.3}$$

The solution of the forward kinematics problem is expressed as a product of all the transformation matrices that represent the coordinate transformations between the individual links. The Denavit-Hartenberg convention is also applicable when solving the forward kinematics problem for redundant robots.

When planning a movement along a given trajectory, it is necessary for the robot to perform certain positions and orientations with its end effector. For this purpose, it is necessary to find the values of the joint coordinates for a given position and orientation of the end effector of the manipulator. This problem

is known as the **inverse kinematics problem** [4]. The configuration of the robot is expressed by a column vector $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_n)^T$, which describes the position of each joint of the robotic system. We say that each manipulator has n number of links and θ_i is called the joint angle. The desired position and orientation of the end effector can be described by a homogeneous transformation matrix $\boldsymbol{H}$ of dimension 4x4, according to Equation (1.4):

$$\boldsymbol{H} = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{p} \\ 0 & 1 \end{bmatrix} \tag{1.4}$$

where $\boldsymbol{R}$ is a 3x3 rotation matrix, $\boldsymbol{p}$ is a column vector that describes the translation, $\boldsymbol{H}$ is a homogeneous transformation matrix that describes the position and orientation of the actuator. The task is reduced to finding the joint variables $\theta_i, \ldots, \theta_n$, such that $\boldsymbol{T}_n^0(\theta_1, \ldots, \theta_n) = \boldsymbol{H}$ [4].

Unlike the forward kinematics problem, there is no generally accepted algorithm for the solution of the inverse kinematics problem. The inverse kinematics problem is usually said to be solved for a given manipulator if the joint variables are determined by an algorithm that allows finding all configurations of the joint variables for a given position and orientation [20]. In robots with additional degrees of freedom, it is possible to have infinitely many solutions to the inverse kinematics problem. Therefore, methods for finding solutions to the inverse kinematics problem of robots without additional degrees of freedom are not applicable to redundant robots.

Iterative methods based on the Jacobi matrix are most often used to solve the inverse kinematics problem for redundant robots. These methods provide an approximate solution and at each step to minimize the difference between the current position and orientation of the end effector and the desired position and orientation. Jacobian solutions are linear approximations to the inverse kinematics problem. They model linearly the end effector movements relative to instantaneous changes in joint angles. The Jacobian matrix is a function of the joint angles $\boldsymbol{\theta} = (\theta_1, \theta_2, \ldots, \theta_n)$. Orin and Schrader describe in [21] how to calculate the Jacobi matrix for different types of joints. The simplest method based on the Jacobian matrix is the **transposed Jacobian matrix** method. The method uses the transposed Jacobian matrix. It was first used for the solution of the inverse kinematics problem in [22]. The change in joint velocities $\Delta\boldsymbol{\theta}$ is expressed with Equation (1.7):

$$\Delta\boldsymbol{\theta} = \alpha \boldsymbol{J}(\boldsymbol{\theta})^T \boldsymbol{e} \tag{1.7}$$

where α is a scalar constant that affects the change of $\boldsymbol{\theta}$ at each iteration [23] and $\boldsymbol{e}$ is an error vector $(6x1)$ representing the position and orientation deviation between the current position and orientation $\boldsymbol{s}$ and the target position and orientation $\boldsymbol{t}$: $\boldsymbol{e} = \boldsymbol{t} - \boldsymbol{s}$. The value of α is chosen such that the new value of the vector $\boldsymbol{e}$ is minimal. The advantages of this method are minimal computation time and easy implementation [24], but the disadvantage of the method is its poor results near singular configurations [25].

The **inverse Jacobi method** uses the inverse Jacobi matrix to calculate the change in joint variables. The inverse Jacobi matrix exists only under the condition that the Jacobi matrix has full rank. The dimensionality of the Jacobian matrix grows with increasing degrees of freedom of the robot, therefore the method will require much more time to find the inverse matrix for robots with additional degrees of freedom [26]. Nearly to the singular configuration, this method is inapplicable because the rank of the Jacobian matrix is not complete [27].

A good approximate solution to the inverse problem of kinematics is given by the **pseudo-inverse Jacobi matrix** method, also known as the inverse Moore-Penrose matrix. It is denoted by $\boldsymbol{J}^{\dagger}$ and is an $n \times m$ matrix. For robotic systems with additional degrees of freedom, the pseudo-inverse Jacobi matrix is expressed with Equation (1.9):

$$\boldsymbol{J}^{\dagger} = \boldsymbol{J}^T (\boldsymbol{J}\boldsymbol{J}^T)^{-1} \tag{1.9}$$

The method of the pseudo-inverse Jacobi matrix is relatively fast, but with a not very good approximation [46]. The method results in a very high velocity of the end effector near a singular

configuration or when the point where the end effector of the robot needs to be positioned is outside its reachable area.

The main common problem of the methods discussed so far is their poor performance near singular configuration. This problem can be dealt with the method of Levenberg-Marquardt – **method of damping least squares** (DLS) with a damping factor. The damping factor can reduce the large velocity near a singularity. The method was used to solve the inverse kinematics problem for the first time by Wampler [28]. It was determined to be the best for finding a solution to the inverse problem of robot kinematics with additional degrees of freedom compared to other methods based on the Jacobian matrix [24, 29]. Its disadvantage is that it is slower compared to the other considered methods, requiring more computing time [24].

Chapter 2. Modelling of a planar anthropomorphic robot arm

In this chapter, the functional requirements for the researched robot are presented and basic notations are introduced. The methods used to solve the forward and inverse kinematics problem for the considered robot are described. No classification by type of solutions to the inverse kinematics problem for robots with additional degrees of freedom is found in the literature. This classification is important in motion planning since a change in the solution type can lead to a deviation from the desired trajectory or an incorrect execution of the assigned task. The scientific applied contributions in this chapter are the application of a method for classifying by type the solutions of the inverse kinematics problem for the considered robot, performing an analysis of the workspace depending on the obstacles for a planar anthropomorphic robot with additional degrees of freedom and the study of the service angle in the different zones (defined by the solution types) in the robot's workspace. This chapter presents the results obtained from the completion of the first task from the tasks of the dissertation.

Functional requirements

The main task of robots is to perform a given movement from point to point or along a predetermined trajectory. For this purpose, it is necessary to consider the structural limitations of the robot, as well as the limitations caused by the environment in which it manipulates. Typically, these constraints are obstacles in the robot's workspace. They can be static or dynamic. The robot designed in the dissertation must be able to perform a given task in both a static and a dynamic environment. For this purpose, it must meet the following requirements:

- The robot must have additional degrees of freedom when performing a plane motion, which means it must have at least three links and 4 rotational joints to perform motion in the plane.
- The lengths of the links must be consistent with the dimensional parameters of the actuators that are used. The robot must be able to perform a spatial movement.
- It is necessary for the designed robot to be able to perform a given trajectory precisely, without deviation from the assigned path and in a minimum time.
- The robot must have an appropriate hardware and software system with the help of which it can locate and avoid the obstacles in its workspace.

Anthropomorphic robot arm description

Let m denotes the number of independent parameters describing the execution of a given task in the robot's workspace, and n is the number of degrees of freedom of the robot. In order the robot to have additional degrees of freedom in respect to a planar motion in the O_{xyz} plane, n must be greater than m. To perform an arbitrary planar motion, 3 degrees of freedom are required, ie. $m = 3$. Therefore, the robot must have more than 3 degrees of freedom.

Let $l_i, i = 1, \dots, 4$ denote the lengths of the robot's links. The designed robot will have 4 rotational joints. Let $\theta_i, i = 1, \dots, 4$ denote the joint coordinates of the robot. We will call the final link of the robot

the end effector or gripper. We will denote the position of the end effector in the plane O_{xyz} by (x, y, z). The robot will perform tasks where the axis z is assumed to be constant and one of the orientation vectors remains parallel to O_z. The rotational joints will have the following joint constraints:

$$\theta_i^{min} \leq \theta_i \leq \theta_i^{max}, \text{за } i = 1, \dots, 4 \quad (2.1)$$

Forward kinematics

In robotic systems with an open kinematic chain, the joint coordinates uniquely determine the position and orientation of the end effector [30]. The considered robot with additional degrees of freedom has 4 rotational joints and corresponding joint coordinates – $\theta_1, \theta_2, \theta_3$ and θ_4. When the fourth link is of zero length, the fourth joint coordinate θ_4 changes only the orientation of the end effector and does not affect its position. In such a case, the joint coordinates $\theta_1, \theta_2, \theta_3$ determine the position of the end effector in the O_{xy} plane. We are only interested in the position of the end effector since the orientation is achieved by changing the joint coordinate θ_4. Fig. 2.1 presents the kinematic diagram of the considered planar redundant robot.

Let us denote the position of the end effector in the workspace by (x, y) and the joint coordinates in the generalized space by $\boldsymbol{\theta} = (\theta_1, \theta_2, \theta_3)$.

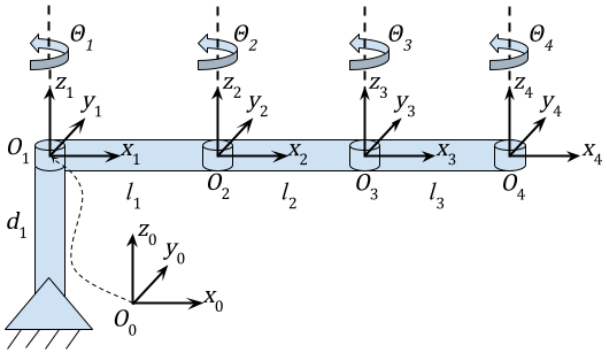

Fig. 2.1. Kinematics scheme of a planar redundant robot.

We will use the Denavit-Hartenberg convention to solve the forward kinematics problem. A coordinate system $O_i x_i y_i z_i$ is placed at each joint $i, i = 0,1,2, \dots, n$, and the coordinate system $O_0 x_0 y_0 z_0$ is placed to the base of the robot. The z_i axis coincides with the axis of rotation of joint $i+1$ relative to joint i. For convenience, the z_0 axis is chosen to coincide with the z_1 axis and $O_0 \equiv O_1$. The z_n axis is chosen to represent the axis of rotation of the actuator. The x_i axis is perpendicular to the z_{i-1} and z_i axes. The y_i axis complements $O_i x_i y_i z_i$ to a right oriented coordinate system. After a fixed local coordinate system $O_i x_i y_i z_i$ is attached to each joint, the position and orientation of coordinate system $O_i x_i y_i z_i$ can be determined relative to coordinate system $O_{i-1} x_{i-1} y_{i-1} z_{i-1}$. The following parameters are used for this purpose. The distance between O_{i-1} and O_i along the x_i axis is denoted by a_{i-1}, the distance between two adjacent coordinate systems along the z_i axis is represented by d_i, α_{i-1} is the angle between the axes z_{i-1} and z_i relative to the x_i axis. The generalized coordinates θ_i represent a rotation relative to the z_i axis, with joint i being a rotational joint respectively [31]. The Denavit-Hartenberg parameter values for the considered robot are presented in Table 2.1

Table 2.1. Denavit-Hartenberg parameters.

i	α_{i-1} [rad]	a_{i-1} [mm]	d_i [mm]
1	0	0	d_1
2	0	l_1	0
3	0	l_2	0
4	0	l_3	0

The Denavit-Hartenberg convention allows us to express any transformation matrix between two coordinate systems by performing the following translations and rotations. Rotation around the z_{i-1} axis at an angle θ_i until the x_{i-1} and x_i axes become parallel. Movement along the z_{i-1} axis by a distance d_i until the x_{i-1} and x_i axes coincide. Movement along the x_i axis at a distance a_i until the coordinate origin O_{i-1} coincides with O_i. Rotation of an angle α_i around the x_i axis until all coordinate axes coincide. These transformations are represented as a product of four matrices (1.1). After multiplication of the given matrices for the transformation matrix $\boldsymbol{T}_i^{i-1}$, we get the matrix from (1.3).

To solve the forward kinematics problem, it is necessary to find a transformation matrix $\boldsymbol{T}_4^0$, which is expressed as a product of all transformation matrices:

$$\boldsymbol{T}_4^0 = \boldsymbol{T}_1^0 \boldsymbol{T}_2^1 \boldsymbol{T}_3^2 \boldsymbol{T}_4^3 \tag{2.2}$$

After expression and substitution in (1.3) and (2.2) for the transformation matrix $\boldsymbol{T}_4^0$, we obtain the following result:

$$\boldsymbol{T}_4^0 = \begin{bmatrix} c_{1234} & -s_{1234} & 0 & a_1c_1 + a_2c_{12} + a_3c_{123} \\ s_{1234} & c_{1234} & 0 & a_1s_1 + a_2s_{12} + a_3s_{123} \\ 0 & 0 & 1 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.3}$$

Where for $1 \leq i, j, k, l \leq 4$:

- $s_i = \sin(\theta_i), c_i = \cos(\theta_i)$,
- $s_{ij} = \sin(\theta_i + \theta_j), c_{ij} = \cos(\theta_i + \theta_j)$,
- $s_{ijk} = \sin(\theta_i + \theta_j + \theta_k), c_{ijk} = \cos(\theta_i + \theta_j + \theta_k)$,
- $s_{ijkl} = \sin(\theta_i + \theta_j + \theta_k + \theta_l), c_{ijkl} = \cos(\theta_i + \theta_j + \theta_k + \theta_l)$.

For the x, y and z coordinates defining the position of the end effector of the robot, we get:

$$x = a_1c_1 + a_2c_{12} + a_3c_{123}, y = a_1s_1 + a_2s_{12} + a_3s_{123}, z = d_1 \tag{2.4}$$

Inverse kinematics problem

The goal of the inverse kinematics problem is to find the required joint coordinates for a given position and orientation of the robot's end effector. The solution of the inverse kinematics problem for robots with an open kinematic structure is most often used in motion planning to determine the set of points that can be used to reach a certain point in the configuration space. Since robots with additional degrees of freedom can reach a given position in their workspace with multiple joint configurations, the solutions to the inverse kinematics problem are also infinitely many. Geometric approach is used in this dissertation for solving the inverse kinematics problem [20]. Fig. 2.3 shows a planar redundant manipulator with n links. With $l_1, \ldots, l_n$, where $i = 1, \ldots, n$ are denoted the lengths of the links. The end effector of the robot should be positioned at point P_n (Fig. 2.3). For this purpose, it is necessary to find the values of the joint coordinates that position the end effector in the target position. The approach starts from the end effector (from the last joint coordinate) and searches for a possible solution for each joint coordinate until the first joint coordinate is reached. A set $\boldsymbol{S}_n$ of points uniformly distributed on a circle with center P_n and radius $r_n = l_n$ is generated. For each point P_{n-1} of the set $\boldsymbol{S}_n$, it is checked whether there exists a configuration to position the end of link $n-1$ at point P_{n-1}. For this purpose, a set $\boldsymbol{S}_{n-1}$ of points uniformly distributed on a circle with center P_{n-1} and radius $r_{n-1} = l_{n-1}$ is generated in an analogous way. These actions are repeated for each joint coordinate from the end effector to the base of the robot. When a set $\boldsymbol{S}_3$ of points uniformly distributed on a circle with center P_3 and radius $r_3 = l_3$ is reached, a solution to the inverse kinematics problem of a two-link mechanism is sought. For each point P_2 of the set $\boldsymbol{S}_3$ is checked whether there exists a valid configuration for the two-link mechanism consisting of units l_1 and l_2. If at least one solution exists, it is checked whether the found

joint coordinates satisfy the joint constraints. The joint angles that satisfy the joint constraints define the joint configurations that can be executed by the robot. Then the configurations defined by points $P_0, P_{1r}, \dots, P_n$ and $P_0, P_{1l}, \dots, P_n$ are saved as the solution of the inverse kinematics problem.

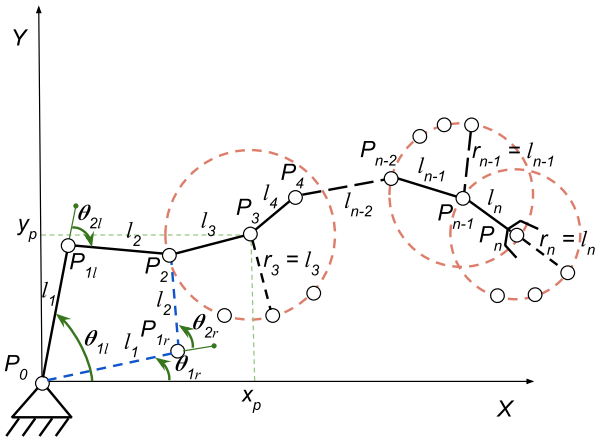

Fig. 2.3 Solution of the inverse kinematics problem for planar redundant robot.

Classification of different types of solutions of the inverse kinematics problem

Let us consider a planar manipulator with three links of non-zero lengths (Fig. 2.5.). Depending on the sign of the second and third joint coordinates, four different types of solutions to the inverse kinematics problem can be defined: Solution RR when $\theta_2 > 0$ and $\theta_3 > 0$. Solution RL when $\theta_2 > 0$ and $\theta_3 < 0$. LR solution when $\theta_2 < 0$ and $\theta_3 > 0$. Solution LL when $\theta_2 < 0$ and $\theta_3 < 0$. The different types of solutions are shown in Fig. 2.5.

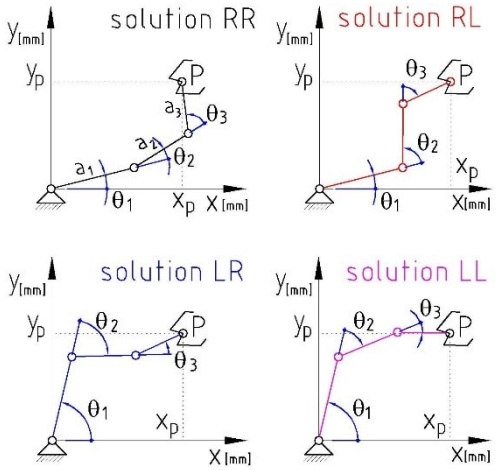

Fig. 2.5. The four different solution types of the inverse kinematics problem.

The four types of solutions are not possible for every point of the workspace. Passing from one solution type to another is important when planning a robot motion.

Let $l_i, i = 1, \dots, 4$ denote the lengths of the robot's links and assume that the investigated robot has the following link lengths (2.10) and joint constraints (2.11):

$$l_1 = 150\ mm, l_2 = 100\ mm, l_3 = 100\ mm, l_4 = 0\ mm, \tag{2.10}$$

$$-\frac{\pi}{2} \le \theta_i \le \frac{\pi}{2}, \text{за } i = 1, \dots, 4. \tag{2.11}$$

Considering the joint constraints, the robot workspace can be divided into different zones, depending on the different types of solutions of the inverse kinematics problem. Ten different zones are available for the considered robot (Fig. 2.6).

The union of all zones represents the entire workspace of the manipulator (Fig. 2.6.a). The intersection of all zones represents an area of the workspace in which all four different types of joint configurations are found (Fig. 2.6.b). In the workspace of the robot there are trajectories, for the execution of which it is required to change the type of solution of the inverse kinematics problem. Fig. 2.6 shows such a trajectory. The robot cannot move from point 1 to point 15 without changing from one type of joint configuration to another. This is because point 1 is in zone 10 and point 15 in zone 1 and zones 1 and 10 do not intersect. Of course, if the desired motion is entirely in zone 1 or zone 10, then the motion will be performed without changing the solution type, since only one type of joint configuration exists in these zones.

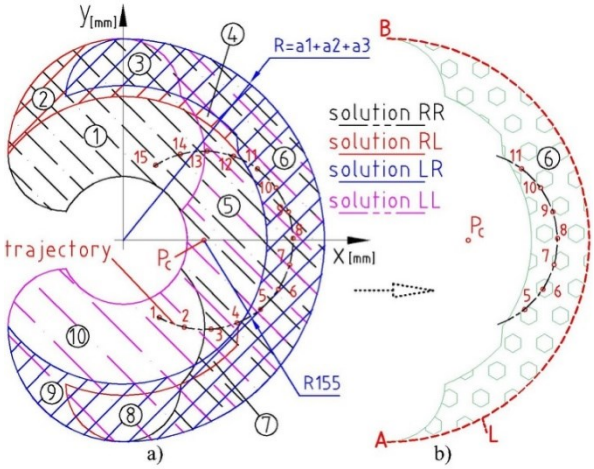

Fig. 2.6. The different zones in the workspace of the robot a) the union of all zones b) the intersection of all zones.

In addition to the joint constraints defined in 2.11, which define the workspace of the manipulator, there may be obstacles in the workspace. Depending on their position and geometry, the workspace of the manipulator can be changed. Let's assume that we have an obstacle with a square shape and coordinates (195, 90). Side length 31 mm and an additional 49 mm for the width of the joints to avoid collision with the obstacle (Fig. 2.8.a.).

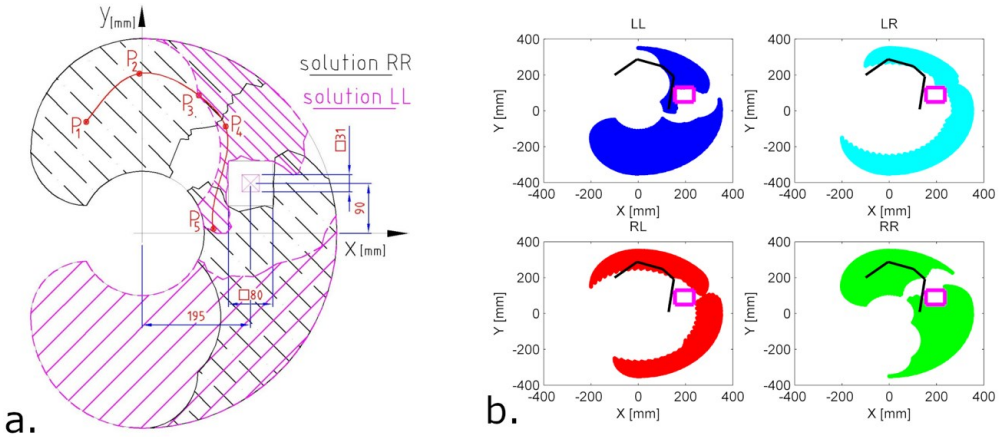

Fig. 2.8. Workspace of the robot in the presence of an obstacle: a. robot workspace and obstacle position; b. change in the four decision types and desired movement trajectory.

As can be seen from the figure, the zones containing solution type RR and LL are divided into two separate parts. The obstacle changes the zones of the four types of solutions of the inverse kinematics problem (Fig. 2.8.b.). If the robot must perform a movement with a starting position from zone number 1 and an end point in zone number 5 (Fig. 2.8), then this movement would not be possible without changing the solution type. Because of joint constraints or the presence of obstacles in the workspace, the robot must change the type of solution to the inverse kinematics problem to be able to realize the desired movement.

Service angle

The term service angle was introduced by Vinogradov in 1971. The service angle is a measure of the multiple orientations of the end effector that the manipulator can realize when the end effector is positioned at a given point in the workspace [32].

The four types of solutions divide the robot's workspace into ten zones (Fig. 2.6). In some of the zones, such as the first and tenth, only one type of solution to the inverse kinematics problem exists. Therefore, the possible orientations of the fourth joint can be fulfilled by only one type of solution. In the other zones, there are points that the robot can reach with its end effector with the same orientation, but with a different type of solution of the inverse kinematics problem. At these points, if the robot's end effector needs to change its orientation, it is possible for it to deviate from the point where it is positioned when switching from one decision type to another. For this reason, it is necessary to analyze the service angle in the different zones. Due to the symmetry of the robot's workspace, it is sufficient to examine the service angle for the first six zones.

Fig. 2.10 shows all the possible orientations for positioning the end effector at point P of zone 1. The third link can be rotated through an angle α from position 1 to position 9. The fourth link has a length equal to zero and has joint constraints given in (2.11). Fig. 2.10.a presents the service angle of the fourth joint. Point P can only be reached with solutions of type RR.

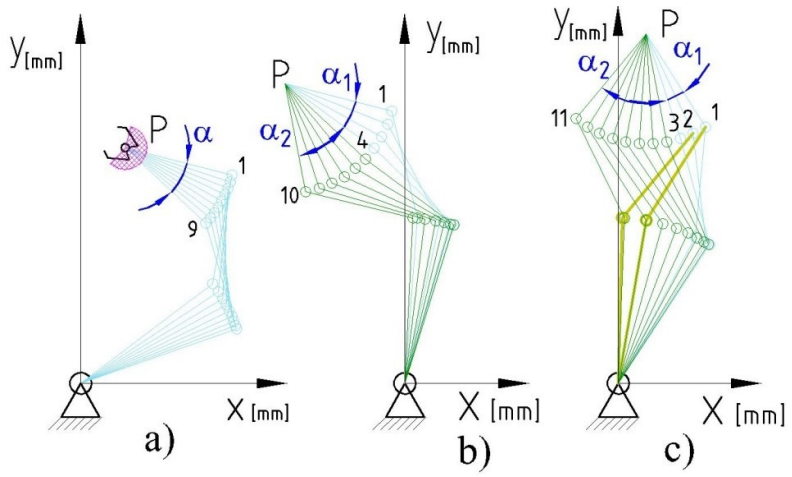

Fig. 2.10. Different joint configurations with which the end effector of the robot can reach point P: a. in zone 1; b. in zone 2; c. in zone 3.

In Fig. 2.10.b. the possible configurations for reaching point P in zone 2 are illustrated. In this zone, two types of solutions to the inverse kinematics problem exist, RR and RL. The transition between the two types of solutions occurs around configuration 4. The service angle can be represented as the sum of the angles α_1 and α_2. In this area, the robot's end effector can change its orientation without the need to move from point P.

The orientation in zone 3 is shown in Fig. 2.10.c. There are three types of solutions in this area: RR, RL and LR. The robot's end effector can change its orientation without deviation from point P.

If point P is in zone 4, 3 types of solutions to the inverse kinematics problem are possible: RR, LL and RL. As can be seen from Fig. 2.11.a., the various angles of the end effector are grouped into two zones. Here, to move from one zone to another, it is necessary to leave point P. This is an important condition that must be considered when planning movements. In zone 5 (Fig.2.11.b.) the situation is analogous to zone 4. Here there are two types of solutions: LL and RR.

In zone 6, all types of solutions exist (Fig. 2.11.c). Depending on the position of point P in zone 6, it is possible that all possible configurations form a continuous interval. In this way, a change in the orientation of the end effector without displacement from point P would be possible. Also, in this area

there are points for which the solutions of the inverse kinematics problem do not form a continuous interval. In this case, the robot's end effector must move from point P to change the type of solution.

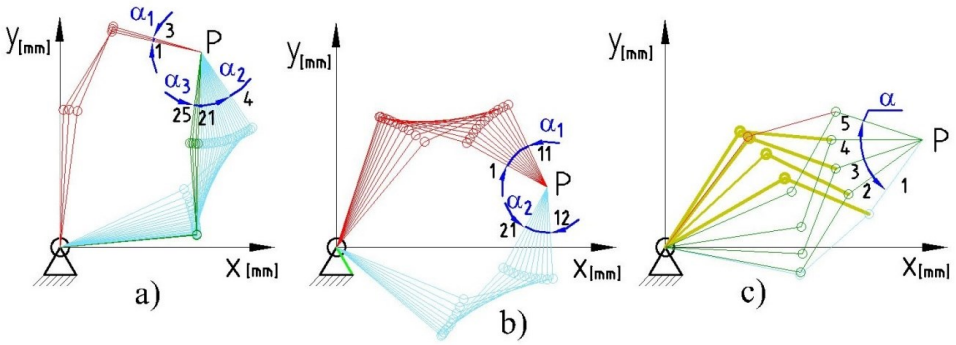

Fig. 2.11. Different joint configurations with which the end effector of the robot can reach point P: a. in zone 4; b. in zone 5; c. in zone 6.

If the end effector of the robot is positioned in a certain position and needs to change its orientation, then the reasoning made should be considered and such a joint configuration should be chosen that changes the orientation and minimizes the deviation from the target point.

Chapter 3. Anthropomorphic robot arm control

The main purpose of the control is to find and assign appropriate commands (positions) to the motors. The task of the actuators is to position the joint coordinates in such a way that the end effector of the robot reaches the desired position and orientation. For this purpose, a hardware and software system are required to perform the necessary calculations and control the robot, as well as a suitable motion planning algorithm. The proposed methods in the literature for motion planning do not consider suitable transition points at which to make the change in solution type of the inverse kinematics problem. In the motion planning algorithms proposed in this chapter, appropriate transition points are considered so that the robot can complete its assigned task in minimum time and without deviation from the assigned motion. In this chapter, the following scientific applied contributions are achieved: a trajectory planning algorithm using graph theory for a planar robot arm with additional degrees of freedom and limited joint space is proposed, motion planning approaches in the presence of static and dynamic obstacles for a planar robot are also proposed for a robot with additional degrees of freedom. The applied contribution in this chapter is the design of a hardware and software control system for an anthropomorphic robot with additional degrees of freedom. In this chapter are described the obtained results from the completion of the second and third tasks from the dissertations tasks.

Hardware requirements

It is necessary for the robot to be able to perform a desired movement or sequence of movements. For this purpose, the robot needs a hardware system to control the motors of the robot. The requirements for hardware components are as follows:

- The main controller must have the necessary computational resources to calculate the forward and inverse kinematics problems.
- The control electronics need to have sufficient memory to allow the loading of a sequence of positions that the robot must perform with its end effector.
- The control electronics need to be based on open-source platforms and a convenient development environment.
- It is necessary for the robot to allow control via both a USB port and a Wi-Fi connection.

- The main controller must have at least 3 serial UART ports. One for communication with the actuators, one for the module that will provide wireless communication with the robot and one for the UART communication with the operator's computer.
- The control electronics must have light indicators to indicate the status of the motors (on/off), whether the control controller has power, and whether a wireless connection to the robot is established.
- The actuators must allow movement of rotational joints of 180 degrees and position control.
- Since the redundant anthropomorphic robot arm will be used for research and teaching purposes, the designed hardware system for its control must be easily expandable and upgradeable.

Hardware components

The selected control electronics of the robot are based on the Arduino platform and the WeMos D1 ESP8266 Wi-Fi module. It is necessary to control the rotational joints of the robot by position, therefore HerkuleX DRS-0101 smart servo motors were selected for their control. FeeTech FT90M servomotors are used to drive the actuator and translation mechanism.

Smart servo motors **HerkuleX DRS-0101** are controlled via serial communication and can return information about their current position and speed. They allow smooth control of their position. They support UART communication, which allows easy change of the position, the color of the LED indicator, and control of up to 254 servo motors at the same time. They also support up to seven different error types. The presence of any of these errors is reported by the LED indicator [33]. The translation mechanism and the end effector of the robot are controlled by **FeeTech FT90M** mini servo motors. They have no feedback well as speed control and are controlled by pulse-width signals (PWM). Their operating voltage is 4.8-6 V [34]. The **Arduino Mega 2560** microcontroller and the WeMos D1 mini-Wi-Fi module were chosen to control the robot. The Arduino Mega 2560 is a microcontroller based on the ATmega2560 microcontroller. It has 54 digital input/output pins, 16 analog inputs, and allows USB connectivity. It has 4 hardware serial UART ports, which makes it suitable for controlling the anthropomorphic robot [35]. The microcontroller has all the necessary computational resources to calculate the forward and inverse kinematics task, which are necessary to perform a given movement and correctly position the executive unit. The **WeMos D1 mini**-Wi-Fi module allows the robot to receive commands via WebSocket, as well as providing a convenient programming and demonstration web-based graphical user interface. The WeMos D1 Mini is a small Wi-Fi device based on the ESP8266EX chip [36]. It can be reprogrammed over a Wi-Fi connection. Control electronics have three main light indicators.

Hardware architecture and communication

The communication method of the described hardware components is shown on Fig. 3.4. Three of the Arduino microcontroller's hardware UART ports are used to communicate with the DRS-0101 servo motors, the Wi-Fi module, and the operator's computer. UART interfaces allow two-way communication. Smart servos are connected on the same communication channel. They are responsible for controlling the four rotational joints and returning information about the current position. The small servo motors are controlled by the PWM signals generated by the Arduino microcontroller.

The Arduino Mega microcontroller is responsible for calculating the forward and inverse kinematics problem, controlling the motors, and handling the communication protocol. The Wi-Fi module provides both a WebSocket and an HTTP server. The WebSocket acts as a proxy and forwards requests and responses to and from the Arduino board. The HTTP server serves the developed graphical web-based interface. This interface uses the WebSocket connection to control the robot. Also, the robot can be controlled directly, through a USB connection between a computer and the Arduino microcontroller.

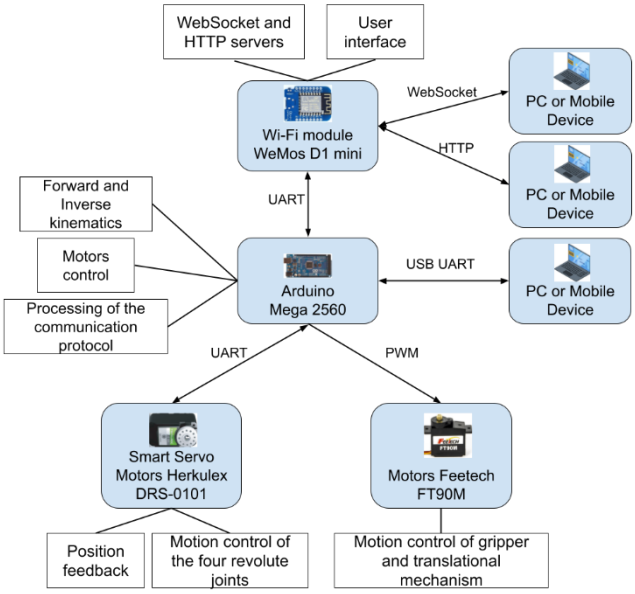

Fig. 3.4. Hardware architecture.

Software requirements

The main purpose of the robot is to be able to perform a predefined movement or set of movements. For this purpose, software system must be developed that meets the following requirements:

- The software system must be able to solve the forward and inverse kinematics problems correctly and quickly.
- It allows the user to control the robot, turn the motors on or off, set a desired position in Cartesian or generalized coordinates, set a sequence of points for a certain movement.
- The software system must provide a communication protocol for communicating with the robot and the ability to monitor the responses returned by the robot.
- The software system must also provide a user interface which can be both graphical and textual.
- The designed software system should allow easy addition of new components and functionalities. This would enable the developed robotic system to be used for various educational, research and industrial tasks.

Software system control

A software system has been developed that meets the described requirements. The software system consists of three main modules: kernel module, service module and graphical user interface. The first module runs on the Arduino microcontroller, and the other two on the WeMos D1 mini-Wi-Fi module. Fig. 3.6 presents the architecture of the designed software system.

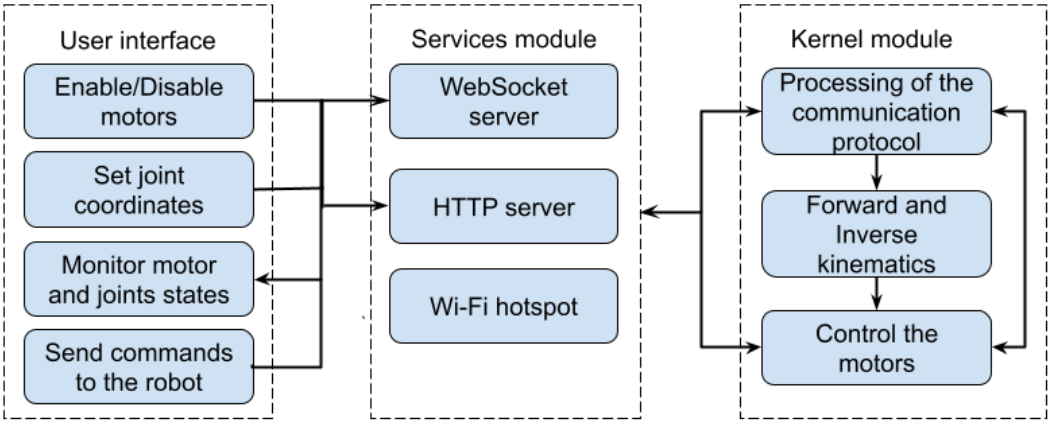

Fig. 3.6. Architecture for software system control of a planar robot.

The **kernel module** is responsible for processing the commands sent through the communication protocol, as well as solving the forward and inverse kinematics problem. The module is also responsible for sending the necessary commands to the motors. The kernel module software is implemented using the C++ programming language and the Arduino IDE programming environment. Commands to the robot's motors can be sent via USB connection between the computer and the microcontroller or via WebSocket communication, which is handled by the service module.

The **service module** runs independently of the kernel module and provides the ability for WebSocket and HTTP communication, as well as a Wi-Fi access point for the robot. This module is responsible for making the connection between the user interface and the kernel module. The Wi-Fi module supports an HTTP server that allows the development of a graphical user interface. The **user interface** can be accessed from any web browser after connecting to the robot via a Wi-Fi access point. Only JavaScript and HTML/CSS were used to implement the user interface.

An application of graph theory to trajectory planning

The task of industrial manipulators is usually to perform a predefined movement without deviation from the desired trajectory and in a minimum time. Therefore, the time to execute a given movement or predefined trajectory is an important part of the control of robotic systems. The faster and more accurately the robot can complete the trajectory, the greater its productivity will be.

Greedy approach

Depending on the values of the joint coordinates, four types of solutions to the inverse kinematics problem can be classified. When executing a desired trajectory, the robot may need to change the solution type. To execute a movement, the motion can be planned as follows. The motion starts from an arbitrary solution of the inverse kinematics problem. This solution type is saved and used for subsequent points of the desired trajectory until a change in the solution type of the inverse kinematics problem is required. If a solution type change is required, the motion must stop, and the joint configuration type of the robot must be changed. After that, the execution of the movement can continue. This method is called the greedy approach. It is necessary to note that when applying this method to a redundant robot and with limited joint space, when changing the solution type of the inverse kinematics problem, the robot will deviate from the desired trajectory.

Algorithm for trajectory planning with minimal deviation from the desired path

It is essential for a robotic system to be able to perform a given task with maximum precision and in minimum time. Since most robots follow a predefined trajectory, this means that during their movement they must not deviate from this trajectory. The algorithm must determine the solution types of the inverse kinematics problem. The desired trajectory must be divided into a certain number of points so that a discrete structure can be used as a directed graph for trajectory planning. The approach should find the possible solutions of the inverse kinematics problem for each point and classify them according to their type, then construct a weighted graph and find a minimum cost path for the desired trajectory. This trajectory planning algorithm can be presented in several phases:

1. Kinematic analysis of a planar redundant robot

The first phase of the algorithm consists in performing a kinematic analysis of the robot, considering its joint constraints. Kinematic analysis will provide information on the inverse kinematics solution types.

2. Trajectory analysis

The second phase consists of performing an analysis of the desired trajectory for which the robot's motion must be planned. This trajectory must be divided into a set of points equidistant from each other.

3. Generation of solutions

In the third phase, it is necessary to calculate the different solutions of the inverse kinematics problem for each of the selected points of the trajectory from the previous step. Since the considered robot arm has additional degrees of freedom, there exists a set of solutions to the inverse kinematics problem for each point. Therefore, the solutions must be limited and equidistant from each other.

4. Classification of the solutions

In this phase, all generated solutions must be classified according to the solution types of the inverse kinematics problem found in the kinematic analysis of the first phase.

5. Construction of weighted directed graph

The next phase consists in constructing a directed graph with weights. Each of the vertices of the graph corresponds to a solution generated from step 3. Between two vertices v and u there will be an edge (v,u) if v and u correspond to two adjacent points p_v and p_u of the desired trajectory and p_v is before p_u. The weight $\omega(v,u)$ for each edge (v,u) can be defined as follows:

$$\omega(v,u) = \sum_{i=1}^{k} |\theta_i^v - \theta_i^u| \tag{3.1}$$

Where k is the number of joint coordinates and $(\theta_1^v, \theta_2^v, \dots, \theta_k^v)$ and $(\theta_1^u, \theta_2^u, \dots, \theta_k^u)$ are the solutions corresponding to vertices v and u. For the considered robot $k = 3$. Equation (3.1) does not consider the time required to accelerate or stop the motion. These times can be ignored because the planned trajectory is considered to have acceleration only at the beginning and deceleration at the end of the movement. Equation (3.1) considers the difference in the joint coordinates of all k vertices. It should be noted that passing through a singular configuration may require a longer execution time.

6. Finding the path with minimum cost

The final phase of the algorithm is to find an optimal path from each vertex that represents a start position solution to each vertex that represents a goal position solution of the desired trajectory. Two of the most used methods in graph theory are Floyd-Warshall and Dijkstra. The Floyd-Warshall algorithm has high computational complexity and is suitable for graphs with a small number of vertices. Another popular algorithm is the heuristic method A*. When applying the Dijkstra and A* algorithms to traversal and search in a graph, there is only one start and one end vertex. In the considered trajectory planning case for a planar robot with additional degrees of freedom, there are more than one start and goal vertices, depending on the number of solutions generated. All initial vertices found must be considered. Therefore, the Floyd-Warshall algorithm will be used for graph theory-based trajectory planning for the anthropomorphic robot with additional degrees of freedom. Fig. 3.14 presents a diagram including the individual steps of the algorithm for planning a time-optimal trajectory.

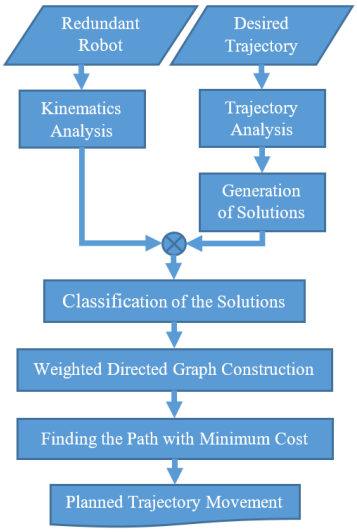

Fig. 3.14. Schematic representation of the trajectory planning algorithm.

The result of the algorithm will be a path from start to target vertex with minimum weight. These will be the solutions of the inverse kinematics problem for successive points of the desired trajectory. Between every two consecutive points representing a different solution type, a stop point must be planned to pass through a singular configuration and change the current solution type. In this way, the movement along the trajectory can be planned with a minimum execution time.

Comparison of trajectory planning algorithms

To compare the presented trajectory planning algorithms, we will use the trajectory from Fig. 2.6.a. and the redundant robot. Again, we will recall that the robot has link lengths (2.10) and joints constraints, which are defined in (2.11).

The trajectory that the robot must execute is an arc movement. The arc is part of a circle whose center has coordinates (140, 0) mm and radius 155 mm. The desired trajectory cannot be executed by the robot using only one type of solution to the inverse kinematics problem. Therefore, at least one solution type change will be required. The desired trajectory is divided into 15 equally spaced points. Up to 100 solutions of the inverse kinematics problem are generated for each point. These solutions are categorized according to the 4 possible solution types of the inverse kinematics problem. Since the robot arm has constrained joint space, there exist points with a small number of solutions for the selected trajectory. The total number of solutions found is 408.

If we plan the movement with the greedy approach, it can be performed with a single change of the decision type (at point 12). But the constrained joint space will require a deviation from the desired trajectory when performing this change. The motion can be planned without deviating from the desired trajectory if the decision type change occurs at an earlier position. When executing the proposed trajectory planning algorithm with minimum execution time, the total number of solutions found to the inverse kinematics problem is 408. This means that the constructed graph has 408 vertices. Therefore, the Floyd-Warshall algorithm can be directly applied since the number of vertices of the graph is less than 1000. The planned optimal path consists of a trajectory that requires two changes of the solution type of the inverse kinematics problem. The first change needs to be made in point 5, and the second in point 8. A comparison of the execution time of the desired move using the three approaches: greedy approach, one solution change move, and the proposed algorithm is shown in Fig. 3.18.

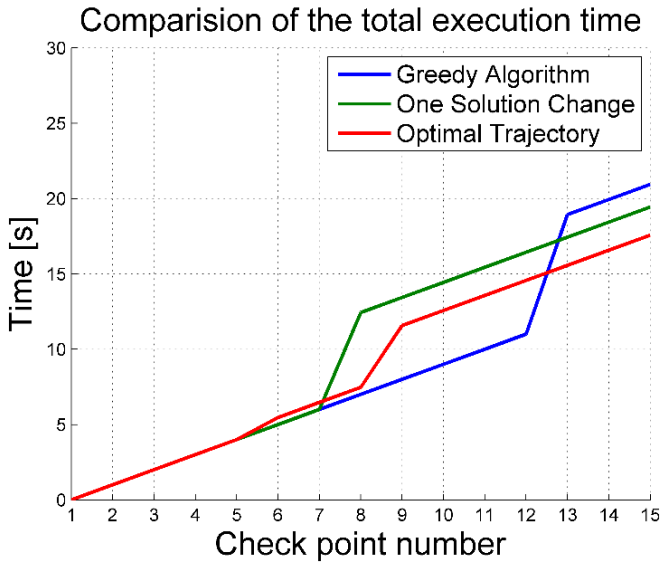

Fig. 3.18. Comparison of the desired trajectory execution time using the considered approaches.

As it can be seen from the figure, the greedy algorithm is fastest until the solution type changes, at point 12. It takes more time to make a change in the solution type of the inverse kinematics problem because the change at point 12 causes a deviation from the desired trajectory. The one solution change

planned motion of the solution type in point 7 has less time to execute because the change is performed in a position that requires less joint displacement. As we can see, the proposed algorithm has the least execution time. Two changes of the solution type are performed, and after the second change, the algorithm takes the lead over the other two algorithms.

Motion planning in the presence of static obstacles

When a manipulator operates in an environment with obstacles whose geometric features are known and their position remains constant during the movement of the robot, we say that the robot is in an environment with static obstacles. In this case, a movement must be planned that can be executed without the risk of colliding with an available obstacle. For this purpose, a point-to-point obstacle avoidance motion planning algorithm for a planar robot will be created, considering the robot's kinematic characteristics as well as the appropriate transition points. Since planar motion will be considered, the robot has additional degrees of freedom. Its joint constraints are defined in (2.11).

The robot workspace can be divided into a finite number of square segments of side length a and a finite number of points. Additionally, we can consider a finite set $\boldsymbol{A}$ of valid joint coordinates for which there is no collision with obstacles. The solution of the forward kinematics problem for the considered set will result from a finite set $\boldsymbol{B}$ of coordinates in the robot workspace. Then, an undirected unweighted graph $\boldsymbol{G}$ can be created as follows.

The vertices of the graph correspond to every single point of the set $\boldsymbol{B}$. For every two distinct vertices u, v there exists edges (u, v) and (v, u) if:

- the norm of the vector difference of the corresponding u and v coordinates in the workspace is less than a predefined value α;
- the norm of the vector difference of the corresponding joint coordinates of u and v is less than a predetermined value β;
- there is no possibility of collision with an available obstacle when the robot moves from u to v;
- the corresponding coordinates in the workspace lie within the same segment of the discretized workspace (this allows the robot to change its configuration from one solution type of the inverse kinematics problem to another) or if both configurations have the same solution type of the inverse kinematics problem. This allows the robot to move to an adjacent position without changing the current solution type.

The breadth first search (BFS) algorithm can be used to find a path between two vertices in the graph $\boldsymbol{G}$. To construct the graph $\boldsymbol{G}$ and perform BFS, it is sufficient to solve only the forward kinematics problem. This is a computationally efficient task for open kinematic loop manipulators. If the desired start or target position of the point-to-point motion does not match any of the vertices in the graph $\boldsymbol{G}$, then the motion planning algorithm must find the closest matching vertex to the start and/or target position. After that, the path between the two vertices is planned to use the BFS algorithm. Movement between two positions is planned as a simultaneous movement of all joint coordinates of equal duration. When planning a point-to-point motion, using graph $\boldsymbol{G}$ and BFS ensures that there is no danger of collision with an available obstacle in the robot's workspace. The sequence of the motion planning algorithm is shown in Fig. 3.19.

The algorithm can be divided into two phases: construction of graph $\boldsymbol{G}$ and point-to-point motion planning. The graph $\boldsymbol{G}$ creation phase is performed only once before point-to-point motion planning and only known static obstacles in the robot's workspace are considered. In this phase, appropriate algorithm parameters are chosen, and the robot workspace is divided into a finite number of square segments. A finite set of joint coordinates and their corresponding coordinates in the workspace is defined, and an undirected unweighted graph is constructed. The creation of the graph considers the defined rules for the adjacency of two vertices and the information about available obstacles.

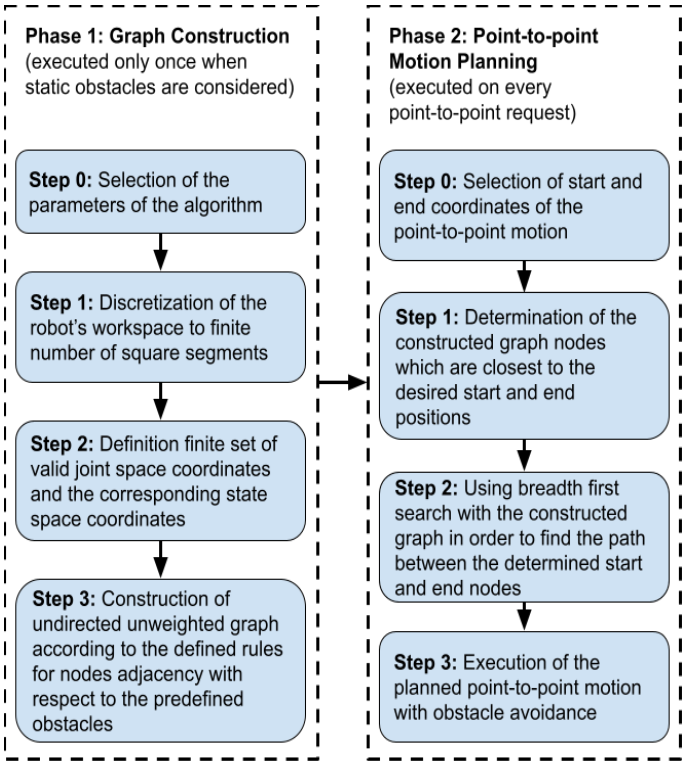

Fig. 3.19. Algorithm for motion planning in the presence of static obstacles.

The second phase of the algorithm consists of point-to-point movement planning and is executed for each movement request to a certain point. In this phase, a start and target position are selected to perform a point-to-point movement. The vertices from the graph that are closest to the desired start and target positions are determined. The BFS algorithm is used on the constructed graph in the first phase to find a path between the start and target vertices. The phase ends with the execution of the planned movement from point to point with static obstacles avoidance.

Robot motion control in a presence of dynamic obstacles

When a manipulator works together with other robots or humans, we say that it is in an environment with dynamic obstacles. The motion of the robot cannot be considered as a point-to-point motion since we are considering a case where the robot is in a dynamic environment. At any moment, a robot, person, or other object may enter his workspace. For this purpose, the control system of the robot must dynamically plan its trajectory depending on the available obstacles and successfully overcome them. The robot control system must plan the new trajectory in such a way that the deviation from the predetermined trajectory is as small as possible.

Let's consider a robotic system with n links that must execute a predefined planar trajectory $\boldsymbol{\theta}_d(t)$, where $t = [0, T]$ is the time interval. The trajectory $\boldsymbol{\theta}_d$ is planned in such a way that the robot will avoid every single static obstacle known in advance. The control system must detect each newly appeared obstacle and change the robot's movement so that the obstacle is avoided without the risk of colliding with it. Obstacle detection and new planning must be done in real time. The end effector of the robot needs to be positioned at the desired target position, avoiding the available obstacles in the robot's workspace.

Obstacles are represented in Cartesian coordinates. When a new obstacle appears, the robot's control system must detect the part of the trajectory that will not be able to be executed safely and block it. After that, the control system needs to replan the trajectory in such a way that the robot avoids the obstacle. The trajectory planning algorithm in the presence of dynamic obstacles is based on graph theory and uses a breadth-first search algorithm. In this way, the execution of the change of the already planned

movement is computationally efficient. This makes it possible to implement the algorithm in real time. The goal is for the robot to position its end effector in a predefined final position at the end of the movement. For the considered robot arm the number of different joint configurations for the same position of the robot's actuator can be infinite. This needs to be considered by the control system when planning the trajectory.

Obstacles are detected when they appear in the robot's workspace. For real-time trajectory planning, the robot workspace can be represented as a graph with vertices corresponding to each possible set of joint coordinates corresponding to the position of the robot's end effector in the workspace. The edges of a graph connect two vertices depending on their corresponding position in the workspace. Two vertices from the graph are connected if their corresponding positions are the same or adjacent and if the required movement is below a predefined threshold parameter. Then, the robot workspace can be divided into a small number of positions to reduce the number of vertices and edges in the graph. This separation is determined by the resolution parameter of the algorithm.

During the execution of the trajectory, obstacles are monitored, and if an obstacle occurs, some edges of the graph are marked as blocked. When the robot enters a near-collision situation with an available obstacle, the control system finds the part of the trajectory that needs to be replanned and applies a breath first search (BFS) algorithm to plan a trajectory according to the created graph . The algorithm has three main steps: **graph creation**, **obstacle tracking**, **trajectory execution and trajectory replanning**. Information about the created graph, the planned trajectory and the cells into which the workspace is divided is kept in shared memory. During each of the steps, a change in the graph and the planned trajectory are monitored and the corresponding information in the shared memory is updated.

The creation of the graph depends on the following parameters of the algorithm: the number of links n, the length of the links a_i , where $1 \leq i \leq n$, the *threshold* value that determines when two different joint configurations are considered adjacent and the resolution parameter that divides the workspace into the robot in a certain number of positions. The robot workspace is represented as a grid of cells (*workspace_grid*). Each cell represents the position of the robot's workspace. Then, a graph is created, with each vertex corresponding to a solution to the inverse kinematics problem that positions the robot's end effector at the center of the cell. For each vertex of the graph, there is the corresponding grid cell, its neighboring cell, and the corresponding vertices of the graph. The algorithm adds an edge in the graph, connecting those vertices for which the corresponding grid cell is the same or adjacent and the required movement corresponds to the *threshold* parameter**.**

Obstacle tracking depends on the following parameters: *obstacles* - description of obstacles found, *workspace_grid* - workspace grid created during the graph creation phase, *desired_trajectory* - initially desired and planned trajectory, *current_time* - current time and *reaction_time* - predefined constant, which accounts for the robot's ability to stop its motion, which determines how close to the obstacle the new trajectory can be planned. When an obstacle is detected, it is marked in the grid of cells (*workspace_grid*) and the edges from the graph that connect the vertices corresponding to the cells are marked as unusable. The algorithm also checks whether the obstacle is new or has changed its position. Obstacle detection is done in real time**.**

During the tracking of the trajectory execution and obstacle movement, it is possible for the control system to replan the movement due to newly appearing obstacles. **Replanning of the trajectory** can be done using the generated graph and the standard BFS algorithm and depends on the following parameters: *workspace_grid* – created in the first phase workspace grid from the generated graph, *graph* – also generated in the first phase, *start_node* and *end_node* – start and final node found during the monitoring phase. These vertices correspond to the beginning and end of the blocked part of the set trajectory. The algorithm checks whether the next position of the planned trajectory is safe to execute). If there is no danger of collision with an obstacle, the robot performs the next position of the trajectory. Otherwise, the algorithm finds a position from the planned trajectory that is not blocked by an obstacle.

Using the created graph and the BFS algorithm, a path is found between the current position of the robot's end effector and the next safe to execute position. If such a trajectory cannot be found, the next point of the trajectory is searched for. Once found, the new trajectory replaces the part of the planned trajectory between the current position and the newfound position, and the robot continues its movement by executing the next point of the trajectory. The planning of the new trajectory is done in real time. Fig. 3.24 shows all the steps of the algorithm for planning a trajectory and overcoming dynamic obstacles.

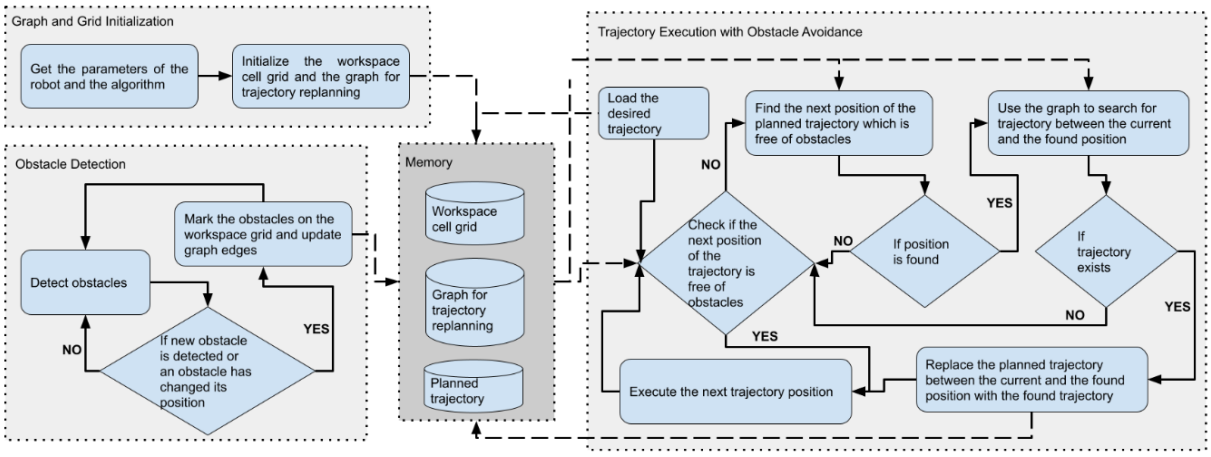

Fig. 3.24. Algorithm for trajectory planning and dynamic obstacles avoidance.

Chapter 4. Experimental verification using a 3D printed robot arm

This chapter verifies the applicability of the proposed algorithms for motion planning of an anthropomorphic robot with additional degrees of freedom in the presence of static and dynamic obstacles in its workspace, through computer simulation in Webots [37] and a real experiment with the 3D printed model of the robot. The experiments with the 3D printed robot also show whether the selected hardware components and the designed software system are suitable for controlling this type of robot. This chapter presents the obtained results from tasks 4 and 5 from the tasks of this dissertation.

Design of 3D printed prototype of robot

3D printing technology was used to create a real model of planar redundant robot. The AutoCAD software program was used to create the 3D model of the desired robot (Fig. 4.1).

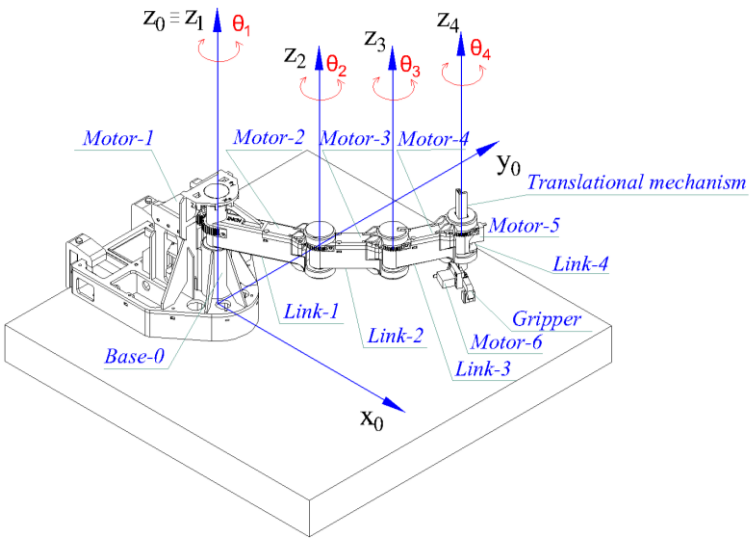

Fig. 4.1. 3D model of planar redundant robot.

One of the requirements for the robot is to have additional degrees of freedom when performing planar motions. For this purpose, it must have a minimum of 4 degrees of freedom. Therefore, the designed robot will have 4 rotational parallel axes. It is also necessary for the robot to be able to perform tasks related to moving various objects, as well as perform spatial movements. Grasping of these objects will be done using an end effector that is attached to a translation mechanism.

The robot consists of Base-0 - base, Link-1 - first link, Link-2 - second link, Link-3 - third link, Lin-4 - fourth link, Translational mechanism, and Gripper. At the base of the robot is the control electronics, which is based on Arduino Mega and W-Fi module WeMos D1 mini. The first motor Motor-1 is mounted in the base of the robot and controls the first link. To perform the translational mechanism movement along the z-axis, the first link is fixed to the base at a height of 150 mm along the z-axis. The second motor is positioned at the end of link 1 and controls the second link. Similarly, the third motor is mounted at the end of the second link and drives the third link. The fourth motor is located at the end of the third link and changes the orientation of the end effector. The fifth motor, Motor-5, is responsible for controlling the translation mechanism, and the sixth motor, Motor-6, for opening and closing the fingers of the end effector. The created 3D printed robot is shown in Fig. 4.5. The material chosen to make the model is PLA.

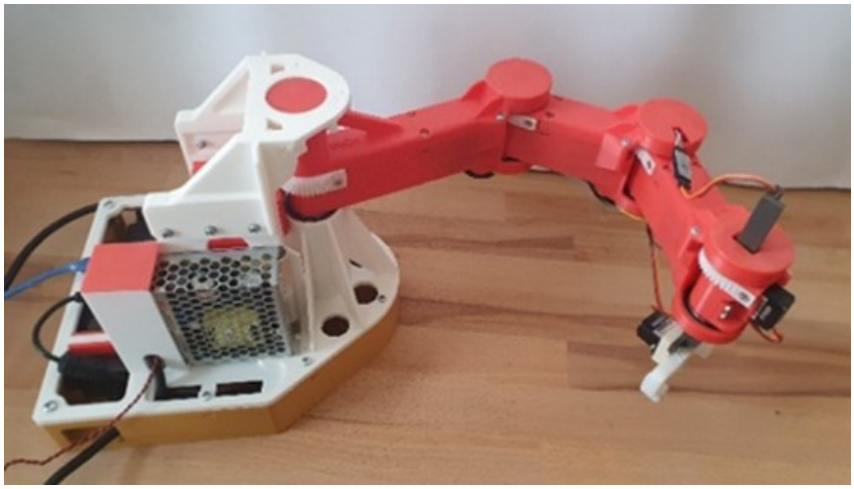

Fig. 4.5. 3D printed robot.

Experimental verification in a Webots simulation environment

A model of the physical 3D printed robot was created in the Webots simulation environment. For this purpose, the 3D model of the robot created in AutoCAD, was used. The model of each link of the robot is saved as .stl file. A .proto file is then created in Webots, where the robot's files are imported and its joint constraints and geometry are defined. Different sensors, cameras as well as obstacles or other necessary objects can also be added to the simulation environment. To this experiment, a camera, and an obstacle with a side length of 31 mm and a height of 120 mm were added. The purpose of the experiment is to prove that the proposed algorithm for motion planning with dynamic obstacles avoidance is effective and suitable for this type of robot. The robot must reach the final position of the predefined trajectory with its end effector, avoiding the dynamic obstacles in the workspace. The robot has length of the links and joint constraints defined in (2.10) and (2.11), respectively. The algorithm represents the workspace as a grid of cells, the *resolution* parameter is chosen to be 10 mm. The desired trajectory is (4.3):

$$\theta_1(t) = 0.4 - \frac{0.7t}{5}, \theta_2(t) = 0.5 - \frac{0.9t}{5}, \theta_3(t) = 0.5 - \frac{0.12t}{5}, t = [0,5] \quad (4.3)$$

During the movement of the robot, the obstacle will change its position, and the robot must successfully go around it. When the simulation camera recognizes the obstacle, the positions in the graph

that are blocked by the obstacle are marked as inaccessible. The part of the trajectory that is blocked by the obstacle is marked in red (Fig. 4.8.a.).

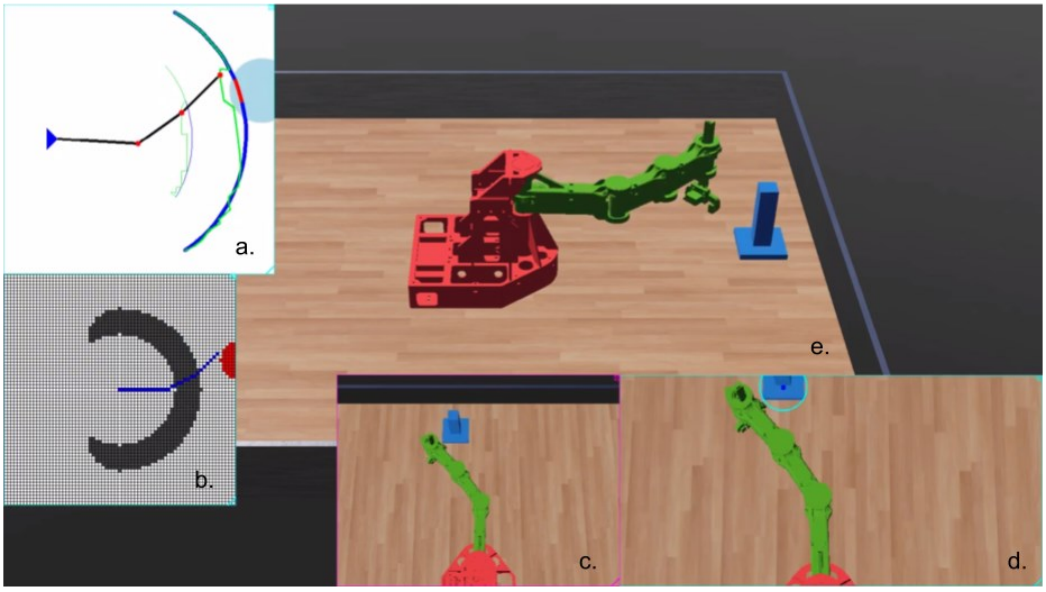

Fig. 4.8. Part of the trajectory is blocked by a dynamic obstacle.

The control system must plan in real time a new trajectory to avoid the moving obstacle. In addition, the new trajectory must be planned in such a way as to return the robot's end effector to the desired trajectory as soon as possible. The green line represents the newly planned trajectory (Fig. 4.8.a.). During the replanning of the trajectory, it is possible for the obstacle to again in such a way that it again overlaps the desired trajectory. Therefore, the control system must constantly monitor during the movement of the robot for changes in the position of already detected obstacles, as well as for newly appearing ones. When an obstacle appears, the third link begins to fold. The purpose of the movement of the third link is to reduce the distance between it and the base of the robot. The purpose of the movement of the third link is to reduce the distance between it and the base of the robot. For each new position that the third rotary joint performs, it is checked whether a safe trajectory can be created, and if there is one, the robot continues its movement. The trajectory is successfully replanned, and the robot manages to reach the desired final position with the desired joint configuration.

Obstacle avoidance

For the experimental verification of the control and trajectory planning algorithms, the 3D printed robot is used. To validate the algorithm proposed in Chapter 3 for motion planning in the presence of static obstacles, the following task is defined. The robot must grasp and carry a certain object while performing point-to-point movement and obstacles avoidance in the workspace.

The obstacle is placed at a position with coordinates (195,90) in the O_{xy} plane. The obstacle has a side length of 31 mm and is 120 mm high. The robot starts its movement from a position with coordinates (-100, 210) in the plane and must reach an object placed at a position with coordinates (120, 10) in the plane. The robot has joint constraints defined in (2.11). The first step of the proposed approach is to create a finite set $\boldsymbol{A}$ of valid joint coordinates. The possible range of valid joint coordinates is discretized linearly, and 10 values are considered for each coordinate. This will result in a maximum score of 1000 points for set $\boldsymbol{A}$. After checking all possible configurations for a possible collision with an obstacle, we get sets $\boldsymbol{A}$ and $\boldsymbol{B}$, each with 811 points. The side length of the workspace segments α is chosen to be 10 mm. Fig. 4.15 shows the different types of segments denoted by the points of the set $\boldsymbol{B}$ with their corresponding type of inverse kinematics solution.

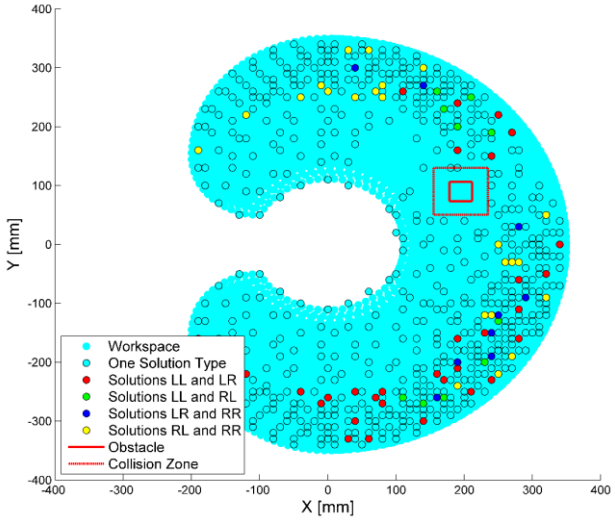

Fig. 4.15. The points of the workspace that are in the graph.

For the construction of graph $\boldsymbol{G}$, the norm of the workspace α is set to 50 (coordinates in the workspace are in mm) and the norm of joint coordinates β is 0.8 (joint coordinates are in radians). With these parameters, the created undirected graph $\boldsymbol{G}$ consists of 811 vertices and 5104 edges. The motion planning algorithm determined the position (-97.18, 210.19) as the closest to the desired start position and the position (111.07, 12.78) as the closest to the desired end position. Fig. 4.17 shows several joint configurations that are close to the obstacle.

The BFS algorithm is used to find a path between the two positions. It finds a path with 22 vertices from the starting position to the final position. The algorithm was executed on a single thread on an Intel® Core™ i7-4710HQ CPU. The graph was created in 19.8 seconds. It took 0.019 seconds for the BFS algorithm to plan the robot's motion. The algorithm is also tested with an unreachable point (100, -300) for motion planning. It took the BFS algorithm 0.24 seconds to determine that it could not plan a move. This is also the maximum time required to plan a point-to-point movement.

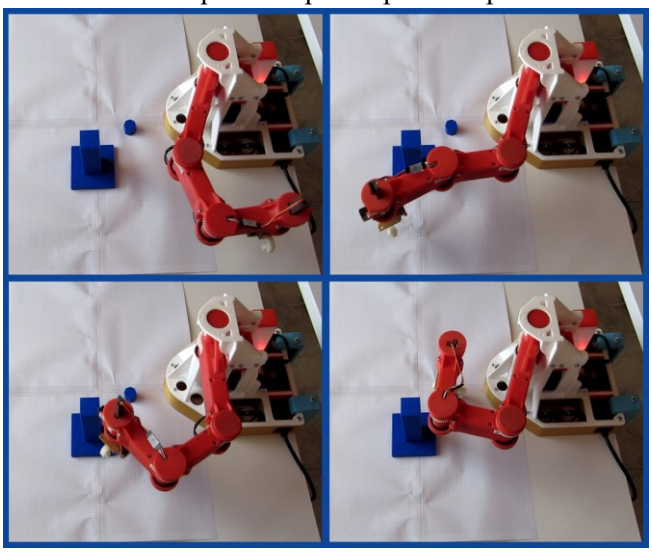

Fig.4.17. Selected joint configurations from the execution of the movement.

Real time obstacle avoidance

After validating the proposed algorithm for motion planning in the presence of static obstacles in the robot's workspace, one can proceed to conduct an experiment to execute a predefined trajectory in the presence of dynamically moving obstacles in the robot's workspace. The algorithm for avoiding dynamic

obstacles is a continuation of the approach for static obstacles avoidance and is presented in the third chapter and validated in a simulation environment Webots.

The purpose of the experiment is to show that for the designed robot this algorithm can work in real time. Unlike the previous experiment, in which the position of the obstacle was predefined, in this experiment we have no prior information about the position of the obstacle. For the algorithm to work in real time, it is necessary at any moment to be able to obtain information about the position of the obstacle.

For the validation of the algorithm can be considered an obstacle that is monochromatic and that will move only in the plane O_{xy}. Its position on z will not change, and its height will be fixed and constant. Such an obstacle can be detected in real time, through a simple color camera and image processing algorithms. The camera does not need to be in a specific position, because we can make an initial calibration. The important thing is not to change its position during robots' movement. A Canyon color camera was used for the purpose of the experiment. The camera is connected via USB to a computer, where it sends the recorded data, and their processing is carried out. To the experiment, the 3D printed robot, obstacle and camera are used to recognize dynamically moving objects. Since the experiment requires dynamically moving objects, it is necessary to periodically change the position of the obstacle. For this purpose, it will be manually moved. The problems that need to be solved to be able to calculate the position of the object are the follows: **object detection**, **calibration method** and **finding the coordinates of the spatial coordinates** in a calibrated image. The Python programming language and the OpenCV computer vision library were used to implement these tasks [38].

The first task is to detect the object and determine the coordinates of its center in the image. This can be done by filtering the color of the object in the HSV space, binarization and finding the center of the largest region in the binary image.

Once the coordinates in the frame can be determined, it can be proceeded to the calibration. It consists of determining the coordinates of the object in the frame when it is placed at four preselected positions in a space. The positions in m are the follows: $(0.5, 0.4, 0.0), (0.0, 0.4, 0.0), (0.0, -0.4, 0.0)$ and $(0.5, -0.4, 0.0)$. They were chosen as such because the total length of the robot's links is 0.35 m. When the object is further away, there is no possibility of a collision and no replanning of the trajectory is required. Once the correspondences of the positions have been determined, it is now possible to proceed to the actual operation of the algorithm.

When we have the correspondences of the positions, at each step only the part of the frame that falls into the quadrangle formed by the coordinates that correspond to the calibration positions can be taken. Depending on the position of the camera, a quadrilateral will be obtained, which is not a rectangle. We can, by a perspective change operation, correct the image and reduce it to a rectangle with the correct proportions. The goal is to convert it from an arbitrary rectangle to 500 by 800 pixels rectangle. This can be done with a warpPerspective function in OpenCV. The considered four positions in space used in the calibration have the following coordinates as correspondences in the untransformed image: (580, 219), (521, 51), (180, 45), (126, 208) and their corresponding coordinates in the transformed image are: (0, 0), (499, 0), (499, 799), (0, 799).

Once we have a 500 by 800 pixel frame, the coordinates of the obstacle in it can be easily found. Again, we filter by color and search for the center of the largest region. This center will have coordinates that can now easily be spatially mapped. The center (0.0, 0.0) in the O_{xy} plane of the robot will have coordinates (0, 400) in the image. The coordinates of the obstacle in the image can be converted to coordinates in the robot workspace using the Equation (4.4) and Equation (4.5):

$$obstacle_x = \frac{(500.0 - object_x)}{500} * 0.5 \tag{4.4}$$

$$obstacle_y = \frac{(400.0 - object_y)}{400} * 0.4, \tag{4.5}$$

where *obsacle_x* and *obstacle_y* are the searched x and y coordinates in the O_{xy} plane, respectively, *object_x* and *object_y* are the x and y coordinates of the obstacle in the image.

Fig. 4.19 shows 4 positions of the robot during the execution of the experiment. The position of the obstacle in the plane and the corresponding position in the image for the sample shown is: In the first position (Fig. 4.19.a.) the coordinates of the obstacle in the plane are (0.34, 0) and the coordinates in the image will be (157, 403). In the second position (Fig. 4.19.b.) the coordinates of the obstacle in the robot's workspace are (0.29, 0.02), and in the image: (203, 380). In the third (Fig. 4.19.c.) and fourth position (Fig. 4.19.d.) the coordinates in the plane are: (0.27, 0.11) and (0.35, -0.1), and the coordinates in the image are: (221, 288) and (148, 505).

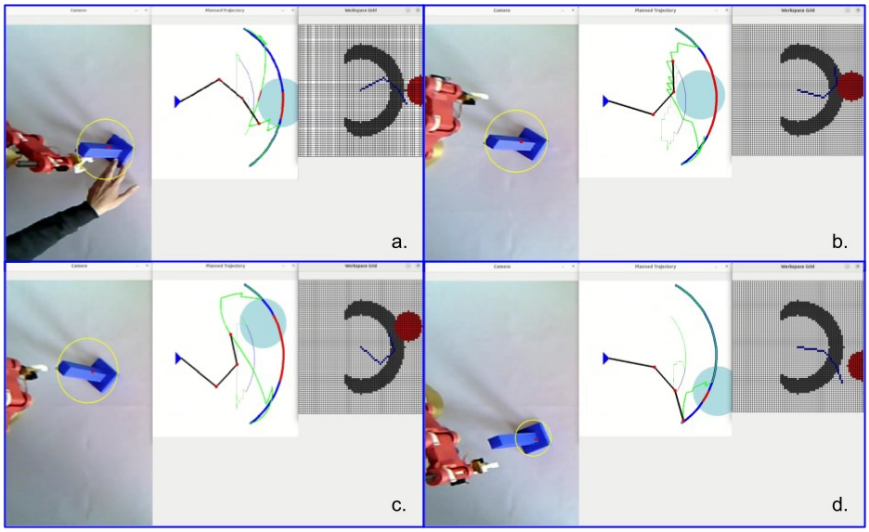

Fig. 4.19. Selected positions of the experiment.

Each of the displayed positions consists of three fields. In the first field, an image from the camera used to detect the obstacle (the camera of the control system) is given. The yellow circle on the obstacle means that it has been detected by the control system and all the described steps for its recognition have been successfully performed. The second field of each position illustrates the target trajectory (the blue arc), the replanned path (the green line), the part of the desired trajectory that is blocked by the obstacle (indicated in red), and the position of the obstacle (the blue circle).

The experiment was conducted successfully. The control system successfully detects an obstacle in the robot's workspace and plans a new trajectory to avoid collision with it and return the robot's end effector to the desired trajectory as soon as possible. During planning, it is possible for the obstacle to change its position, the control system monitors this and, if necessary, plans a new trajectory. A breadth-first search algorithm is used for trajectory planning, making the task computationally efficient and successful to run in real time.

Conclusion

The dissertation analyzes the advantages and disadvantages of redundant robots and motivates the choice to use a robot with additional degrees of freedom in tasks related to overcoming obstacles in the robot's workspace. A mathematical model of a robot with additional degrees of freedom was prepared. The forward kinematics problem is solved using the Denavit-Hartenberg convention, and the inverse kinematics problem is solved using a geometric solution method. The solutions of the inverse kinematics problem of such type of robot can be classified into several types, depending on the sign of the second and third joint coordinates. The thesis examines these types of solutions and how they divide the robot's workspace into different zones. Transitioning from one zone to another may require a change of solution type, which may lead to a deviation from the desired movement. Therefore, transition points where the

change of solution type can be performed without deviating from the desired path have been investigated.

A prototype of a robot arm with 4 rotational joints was designed and realized, using 3D printing methods. A hardware and software control system for the robot was created, which is based on the Arduino Mega 2560 open-source platforms and ESP8266 Wi-Fi module. A user interface is created through which the operator can monitor the robot's status and send commands to it, and a motion planning algorithm is proposed. The algorithm considers the transition points and is based on graph theory.

Algorithms for trajectory planning in the presence of static and dynamic obstacles are proposed and developed in the dissertation. The algorithms are validated by creating computer experiments and conducting real experiments with the 3D printed prototype. When performing a given task in the presence of static obstacles, the robot's control system has prior information about the position of the obstacles. The task becomes more complex when the robot is manipulating an environment with dynamic obstacles. In this case, the control system must monitor the position of the obstacles in real time, and when an obstacle blocks the planned movement, it is necessary to replan the robot's movement and return to the target trajectory as quickly as possible. The conducted and described experiments proved that the proposed algorithms for trajectory planning in the presence of static and dynamic obstacles are suitable for controlling this type of robot, they are computationally efficient, and this allows them to be applied in tasks requiring real-time execution.

All the tasks that were defined at the beginning of the dissertation have been completed. The goal of creating a mathematical model and prototype of a planar anthropomorphic robot with additional degrees of freedom and creating methods for its control has been achieved.

Development prospects

Research on trajectory planning algorithms in the presence of static or dynamic obstacles will continue in the future. It can be investigated how the choice of parameters of the trajectory planning algorithm in the presence of static obstacles affects the performance of the algorithm. Also, how the graph construction phase can be implemented with higher computational efficiency or even done in real-time in the presence of dynamic obstacles.

The conducted experiments show that the designed hardware system is suitable for controlling this type of robot. But the internal memory of the Arduino controller is limited. Therefore, it would be good, for the future, to add external memory to the robot's hardware system.

References

[1] V. Potkonjak, M. Popović, M. Lazarević and J. Sinanović, "Redundancy problem in writing: From human to anthropomorphic robot arm," in *Systems and Cybernetics, Part B: Cybernetics, IEEE Transactions*, 1999.

[2] T. Petric, A. Gams, N. Likar and L. Zlajpah, "Obstacle avoidance with industrial robots," in *Springer International Publishing*, 2015.

[3] St. Chiaverini, G. Oriolo and A. Maciejewski, "Redundant Robots," 2016.

[4] M. W. Spong, S. Hutchinson and M. Vidyasagar, Robot Modeling and Control, 2020.

[5] G. Oriolo, M. Cefalo and M. Vendittelli, "Repeatable motion planning for redundant robots over cyclic tasks," *IEEE Transactions on Robotics,* pp. 1-14, 2017.

[6] Ch. A. Klein and Br. E. Blaho, "Dexterity Measures for the Design and Control of Kinematically Redundant Manipulators," *The International Journal of Robotics Research,* vol. 6, pp. 72-83, 1987.

[7] V. Chembuly and H. Voruganti, "Trajectory planning of redundant manipulators moving along constrained path and avoiding obstacles," in *Procedia Computer Science*, 2018.

[8] J. Hollerbach and Ki Suh, "Redundancy resolution of manipulators through torque optimization," *IEEE Journal on Robotics and Automation,* vol. 3, pp. 308-316, 1987.

[9] A. R. Hirakawa and A. Kawamura, "Trajectory planning of redundant manipulators for minimum energy consumption without matrix inversion," in *Proceedings of International Conference on Robotics and Automation*, Albuquerque, 1997.

[10] H. Voruganti and V. Chembuly, "An optimization based inverse kinematics of redundant robots avoiding obstacles and singularities," in *Advances in robotics Conference*, 2017.

[11] T. Yoshikawa, "Manipulability of robotic mechanisms," in *IEEE International Conference on Robotics and Automation*, 1985.

[12] Ch. Zhou, B. Huang and P. Fränti, "A review of motion planning algorithms for intelligent robots," *Journal of Intelligent Manufacturing,* vol. 33, pp. 387-424, 2022.

[13] R. Venkat, "Path Finding - Dijkstra's Algorithm," 2014.

[14] S. Skiena, "Dijkstra's algorithm," in *Implementing Discrete Mathematics: Combinatorics and Graph Theory with Mathematica*, Addison-Wesley, Boston, Mass, USA, 1990, pp. 225-227.

[15] Z. -c. Du, G. -Y. Ouyang, J. Xue and Y. -b. Yao, "A Review on Kinematic, Workspace, Trajectory Planning and Path Planning of Hyper-Redundant manipulators," in *10th Institute of Electrical and Electronics Engineers International Conference on Cyber Technology in Automation, Control, and Intelligent Systems (CYBER)*, 2020.

[16] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," *The International Journal of Robotics Research,* 1999.

[17] L. E. Kavraki, P. Svestka, J. C. Latombe and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," in *IEEE Transactions on Robotics and Automation*, 2002.

[18] A. Gupta, S. Agrawal, A. Deshmukh and P. Bhargava, "A Geometric Approach to Inverse Kinematics of a 3 DOF Robotic Arm," 2018.

[19] Br. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, Robotics: Modelling, Planning and Control (1st. ed.), Springer Publishing Company, Incorporated., 2009.

[20] I. Chavdarov, V. Nikolov, B. Naydenov and G. Boiadjiev, "Design and Control of an Educational Redundant 3D Printed Robot," in *International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, Split, Croatia, 2019.

[21] D. E. Orin and W. W. Schrader, "Efficient computation of the Jacobian for robot manipulators," *International Journal of Robotics Research,* pp. 66-75, 1984.

[22] J. Baillieul, "Kinematic programming alternatives for redundant manipulators," in *IEEE International Conference on Robotics and Automation*, 1985.

[23] Buss, R. Samule , "Introduction to Inverse Kinematics with Jacobian Transpose, Pseudoinverse and Damped Least Squares methods," 2009.

[24] M. Kalasariya, V. Patel, A. Thakkar, "Comparative Study of Iterative Inverse Kinematics Methods for Serial Manipulators," *International Journal of Engineering and Technical Research,* 2018.

[25] R. B. L. Barinka, "Inverse Kinematics - Basic Methods".

[26] M. R. Gier, "Control of a robotic arm: Application to on-surface 3Dprinting," 2015.

[27] E. Krustev, "Passing Through Jacobian Singularities in Motion Path Control of Redundant Robot Arms," in *Proceedings of the 27th International Conference on Robotics in Alpe-Adria Danube Region (RAAD 2018)*, 2018.

[28] Wampler and W. Charles, "Manipulator Inverse Kinematic Solutions Based on Vector Formulations and Damped Least-Squares Methods," in *IEEE Transactions On Systems, Man, And Cybernetics*, 1986.

[29] I. Duleba, M. Opalka, "A Comparison Of Jacobian–Based Methods Of Inverse Kinematics For Serial Robot Manipulators," *International Journal of Applied Mathematics in Computer science,* vol. 23, pp. 373-382, 2013.

[30] A. Mueller, "Modern Robotics: Mechanics, Planning, and Control," *IEEE Control Systems Magazine,* vol. 39, no. 6, pp. 100-102, 2019.

[31] Boiadjiev G, Krastev E, Chavdarov I, Miteva L, "A Novel, Oriented to Graphs Model of Robot Arm Dynamics," *Robotics,* vol. 10, no. 128, 2021.

[32] Vinogradov, A. Kobrinski, Y. Stephenko, L. Tives, "Details of Kinematics of Manipulators with the Method of Volumes," *Mekanika Mashin,* vol. I, no. 5, pp. 5-16, 1971.

[33] "Smart Servo Motors HerkuleX DRS-0101," [Online]. Available: https://www.robotshop.com/en/herkulex-drs-0101-robot-servo.html. [Accessed 01. 11. 2022].

[34] "Homepage FeeTech,," [Online]. Available: http://www.feetechrc.com/. [Accessed 01 11 2022].

[35] "Arduino Mega 2560," [Online]. Available: https://store.arduino.cc/arduino-mega-2560-rev3. [Accessed 01. 11. 2022].

[36] "Wemos D1 mini," [Online]. Available: https://docs.zerynth.com/latest/official/board.zerynth.wemos_d1_mini/docs/index.html. [Accessed 01. 11. 2022].

[37] "Webots," [Online]. Available: https://cyberbotics.com/doc/guide/index. [Accessed 25. 03. 2022].

[38] "OpenCV," [Online]. Available: https://opencv.org/. [Accessed 20. 11. 2022].

Contributions to the dissertation

Considering the work on the dissertation and the results obtained from the conducted research and presented in the thesis, it can be formulated the following contributions:

Applied scientific contributions

- An approach is created to classify by type the solutions of the inverse kinematics problem for a planar robot with additional degrees of freedom (Chapter 2, [П1], [Д3]).
- Analysis of the workspace of a planar robot with additional degrees of freedom depending on the available obstacles was performed (Chapter 2, [П4], [Д4]).
- The service angle in the workspace of a planar robot with additional degrees of freedom was investigated (Chapter 2, [П2]).
- A graph theory trajectory planning algorithm is developed for a planar robot with additional degrees of freedom and constrained joint space (Chapter 3, [П1], [Д3]).
- A motion planning approach in the presence of static obstacles for a planar robot with additional degrees of freedom is developed (Chapter 3, [П4], [Д4]).
- A real-time dynamic obstacle avoidance algorithm is implemented in the workspace of a planar robot with additional degrees of freedom (Chapter 3, [П5]).

Applied contributions

- A hardware and software system was designed to control a planar robot with additional degrees of freedom (Chapter 3, [П3], [Д2]).
- A computer experiment of the proposed trajectory planning methods was created using Webots simulation software (Chapter 4, [П6]).
- A real experiments with 3D printed prototype are conducted in order to verify the proposed algorithms for trajectory planning with static or dynamic obstacle avoidance. (Chapter 4, [П4], [П5], [Д4]).

Publications, reports and participation in projects related to the topic of the dissertation

Publications

П1. **Lybomira Miteva**, Ivan Chavdarov, Kaloyan Yovchev, *Trajectory Planning for Redundant Robotic Manipulators with Constrained Joint Space*, 2020 International Conference on Software, Telecommunications and Computer Networks (SoftCOM), 2020, DOI: 10.23919/SoftCOM50211.2020.9238296, ISBN: 978-953-290-099-6, ISSN: 1847-358X, Ref Scopus, Ref IEEE Xplore, https://ieeexplore.ieee.org/document/9238296.

П2. **Lyubomira Miteva**, Galya Pavlova, Roumen Trifonov, Kaloyan Yovchev, *Manipulability Analysis of Redundant Robotic Manipulator*, ACM International Conference Proceeding Series, 21st International Conference on Computer Systems and Technologies, CompSysTech 2020, Pages: 135-140, DOI: 10.1145/3407982.3407987, ISBN 978-145037768-3, Ref Scopus, SJR (0.182 – 2020), https://dl.acm.org/doi/10.1145/3407982.3407987

П3. **Lyubomira Miteva**, Kaloyan Yovchev, *Hardware and Software Design for Redundant Robotic Manipulators*, CEUR Workshop Proceedings, 2020, Volume: 2656, Pages: 167-180, ISSN: 1613-0073, Ref Scopus, SCOPUS, SJR (0.177 – 2020), https://ceur-ws.org/Vol-2656/paper17.pdf.

П4. **Lyubomira Miteva**, Kaloyan Yovchev, Ivan Chavdarov, *Point-to-point Motion Planning with Obstacle Avoidance for Hard Constrained Redundant Robotic Manipulators*, 2021 XXX International Scientific Conference Electronics (ET), 2021, DOI: 10.1109/ET52713.2021.9579820, ISBN: 978-166544518-4, Ref Scopus, Ref IEEE Xplore, https://ieeexplore.ieee.org/document/9579820.

П5. Kaloyan Yovchev, **Lyubomira Miteva**, *Real-time Trajectory Replanning for Dynamic Obstacles Avoidance for Robotics Manipulators*, ACM International Conference Proceeding Series, 23rd International Conference on Computer Systems and Technologies, 2022, Pages: 45–50, DOI: 10.1145/3546118.3546135, ISBN 978-145039644-8, Ref Scopus, SJR (0.232 – 2021), https://dl.acm.org/doi/abs/10.1145/3546118.3546135.

П6. **Lyubomira Miteva**, Kaloyan Yovchev, Denis Chikurtev, *Software and Hardware Infrastructure for Research and Development of Intelligent Control for Robotic Manipulators*, 2022 XXXI International Scientific Conference Electronics (ET), 2022, DOI: 10.1109/ET55967.2022.9920270, ISBN: 978-166549878-4, Ref Scopus, Ref IEEE Xplore, https://ieeexplore.ieee.org/document/9920270.

Conference presentations

Д1 **Manipulability Analysis of Redundant Robotic Manipulator**, international conference CompSysTech 2020. http://www.compsystech.org/_cst20

Д2 **Hardware and Software Design for Redundant Robotic Manipulators**, international conference ISGT 2020. https://isgt.fmi.uni-sofia.bg/

Д3 **Trajectory Planning for Redundant Robotic Manipulators with Constrained Joint Space**, international conference SoftCom 2020. http://softcom2020.fesb.unist.hr/

Д4 **Point-to-point motion planning with obstacle avoidance for hard constrained redundant robotic manipulator**, international conference Electronics - ET 2021, Sozopol. https://e-university.tu-sofia.bg/e-conf/?konf=24

Scientific projects

- **Research and Development of Learning Algorithms for Interaction Between Industrial Robots and Objects in the Workspace**, № KP-06-M47/5 from 27.11.2020, Competition for financial support for projects of junior basic researchers and postdocs – 2020.
- **Research and Modelling of New Robots by Using Innovative Technologies and Materials**, NSF № 17/10 from 12.12.2017, Competition for financial support of scientific research – 2017.
- **Development of Methods and Algorithms for Robotic Manipulators**, NSF 80-10-23/2020
- **Development of Methods and Algorithms for Robotic Manipulators**, NSF 80-10-89/25.03.2021.
- **Development of Methods and Algorithms for Robotic Manipulators**, NSF № 80-10-70/10.05.2022.

Declaration of originality

I declare that the presented in connection with the procedure for the acquisition of the educational and scientific degree Doctor of Philosophy at Sofia University "St. Kliment Ohridski" dissertation on the topic: "Modeling and control of an anthropomorphic robot arm" is my work.

Citations of all sources of information, text, illustrations, tables, images, and others are indicated according to the standards.

The results and contributions of the dissertation research conducted are original and are not borrowed from research and publications in which I have no participation.

Signature: ……
/Lyubomira Miteva/

Biography of the author of the dissertation

Lyubomira Miteva was born in 1994 in Pazardzhik. In 2017, she graduated with a Bachelor's degree in Software Engineering at the Faculty of Mathematics and Informatics (FMI) of Sofia University "St. Kliment Ohridski". In 2019, she graduated with a Master's degree in informatics and computer science, specifically "Mechatronics and Robotics" at FMI. The thesis is in the field of anthropomorphic robots. In 2020, she was enrolled as a full-time PhD student at the Department of Computer Informatics of FMI, Sofia University "St. Kliment Ohridski". The topic of the thesis is "Modelling and control of an anthropomorphic robot arm".

Since 2020, she has been a part-time lecturer at FMI and leads exercises for the courses "Introduction to Programming" and "3D Modeling, Printing and Applications in Robotics".

Lyubomira Miteva's scientific interests are related to the design, creation and control of robotic manipulators and mobile robots.

Acknowledgments

I would like to thank my supervisors Prof. Evgeniy Krastev, PhD and Assoc. Prof. Ivan Chavdarov, PhD for constructive advice, comments, and remarks.

I am also grateful to the members of the Departments of Computer Informatics and Mechatronics, Robotics and Mechanics for their advice and guidance.

I also thank the Institute of Robotics – BAS for the support and the opportunity to participate in scientific projects.