TECHNISCHE HOCHSCHULE LÜBECK

[Abteilung/FB]
Datum: 15.11.2022

INDUSTRTIAL ROBOTICS – PRACTICAL WORK

Objectives:

In this practical course, the position of the gripper tip of the Tinkerbot Braccio will be calculated from the joint angles in a Python program. The basis for this is the kinematic model of the robot and the determination of the Denavit-Hardenberg transformation parameters. The forward kinematics of the Tinkerbot Braccio will be implemented in Python and the robot will be controlled.

Preparation:

- Denavit-Hardenberg process
- Solve the exercises for the Denavit-Hardenberg procedure
- Python:
    - Dealing with SymPy (definition of variables, substitution of variables)
    - Definition of functions

Required parts:

1. Robot attached to plates (best fixed with adhesive tape in case of a crash)
2. Adhesive tape / Edding to draw in base coordinate system
3. Caliper for measuring the position, or folding rule / ruler
4. Small model coordinate system that can be pressed into the robot's hand

Task 1: Control of the robot

Write a small Python script that allows you to control the Tinkerkit Braccio robot.

- To do this, first connect the robot to the USB interface and find out which COM interface is assigned to the robot.
- For USB/serial communication, use the "pyserial" library under Python (https://pyserial.readthedocs.io/en/latest/). Use Jupyter Notebook for programming with Python
- Send movement commands to the robot. The description of the communication protocol can be found here: https://github.com/stefangs/arduino-library-braccio-robot/blob/master/examples/serialBraccio/serialBraccio.ino
    - The numbers entered in the string indicate the rotation angles of the base, shoulder, elbow, wrist joint, wrist rotation, gripper opening and movement speed.
    - The specified numerical values refer to the absolute values of the drive angles and have a range of 0 to 180°.
    - Tip: Reduce the movement speed from 100 to 30.
- For each axis, find out what the positive direction of rotation is. Attach a label to the axes.
- Move the robot to the position shown in Fig. 1. The base should be set to a rotation angle of 0°. Use tape strips on the base plate to mark the x and y axes of the robot base.

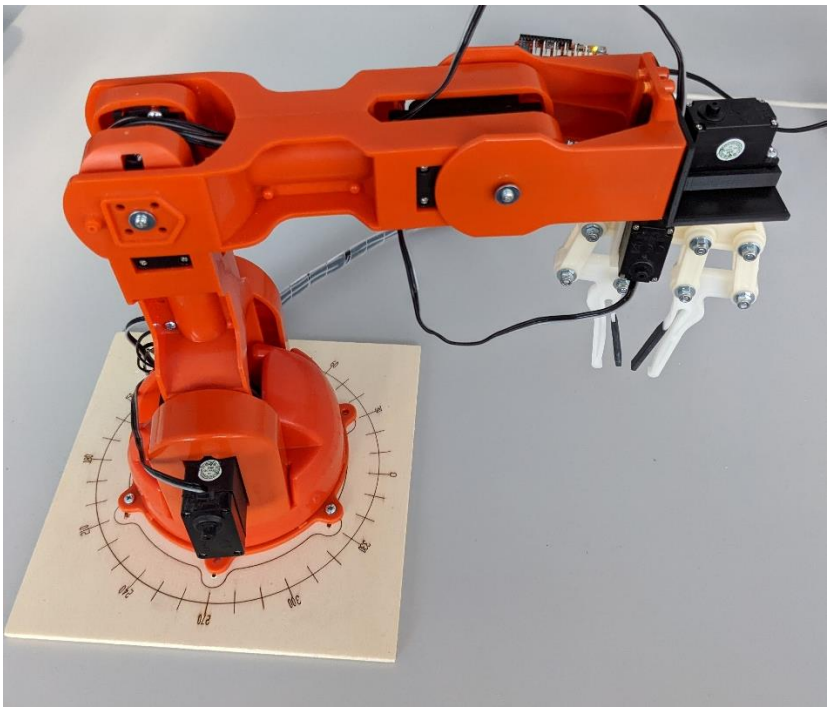

Fig.1: Starting position of the robot

Task 2: Modelling

Create the kinematic model of the robot:

1. Move the robot to the position shown in Fig. 1 (the labels on the wooden panel are irrelevant). The arm should then be aligned parallel to the x-axis on the base plate, which we refer to as the home position. The joint angles are then:
   - Base: 0°
   - Shoulder: 90°
   - Elbow: 180°
   - Wrist: 90°
   - Wrist rotation: 0°
2. The coordinate system 5 at the gripper tip should be aligned so that the z-axis is pointing downwards and the x-axis into the image plane (in relation to Fig. 1).
3. Make a sketch with the robot's kinematic chain for the position shown in Fig. 1.
   1. Name the links and joints.
   2. Sketch the z-axes. Note that the direction of rotation of the servo motor corresponds to the respective orientation of the z-axis.
   3. Measure the link lengths using a ruler or caliper and enter the dimensions in the sketch.
4. Abstract the kinematic chain further into a concatenation of the joint coordinate systems. Using the Denavit-Herdenberg method, draw the x and z axes for each joint. Also draw the intermediate coordinate systems.
5. Determine the Denavit-Hardenberg parameters and enter them in Table 1 below.
6. Determine the offset values of the joint angles for their programming. Enter the determined values in Table 2. These are required in subtasks 3.

Table 1: DH parameters

Glied	d_i	δ_i	a_i	$\propto_i$

Table 2: Determining the offset for programming

Link	δ_i-parameters from the DH table	Joint angle of the robot in the basic position according to Fig. 1	Programmier-Offset

Task 3: Programming the forward kinematics

Create a Python program for calculating forward kinematics. The program should calculate the position of a point in the TCP coordinate system in the coordinates of the robot base.

1. First, set up the matrices in Python for the individual transformations of the DH transformation (rotation around x, rotation around z, translation along x, translation along z). From this, determine (symbolically) the DH transformation matrix *TM*.
2. Define a transformation function for each member of the robot using the under DH transformation matrix. Use the substitution function of SymPy to replace variables with function parameters.
3. Check your program by comparing the target position according to forward kinematics with the actual positioning of the robot.