Project Assignment: Educational Robotic Arm

Robotics 34753

Department of Electrical and Photonics Engineering
Department of Civil and Mechanical Engineering
Technical University of Denmark

PURPOSE

The purpose of this project assignment is to practice the topics learned in the Robotics course 34753 on a realistic robotic application.

INTRODUCTION

In this project assignment, a DIY 4-joint robotic arm is considered, made of 4 Dynamixel servos. The robot is shown in the picture on the front page of this document and is used for educational and research purposes. It is equipped with a camera on its end-effector, and a stylus can be attached in order for the robot to perform a variety of tasks. The project assignment treats tasks which involve direct and inverse kinematics, dynamics, singularities, simulation, trajectory planning and computer vision.

ABOUT THIS MATERIAL:

This document defines a project assignment which consists of **11 problems** organized in 5 different parts. Parts 1-4 are theoretical and can be solved without access to the actual hardware, while part 5 is practical and requires access to a robot arm in the lab and programming the actual robotic arm. Annex A contains some derivations from the velocity kinematics theory that are useful for solving problems 5 and 6. Annex B contains practical information about access to the hardware and Annexes C and D contain useful information about setting up the software for controlling the robots.

CONTENTS:

Part 1 contains problems 1-5, all concerning different aspects of forward, inverse and velocity kinematics. Part 2 contains problems 6 and 7 which deal with trajectory planning. In part 3 you are asked to solve problems 8 an 9, which concern singularities and static loads, respectively. Part 4 contains problem 10 which is about the dynamics modelling of the robotic arm.
The last part 5, contains four different choices for problem 11, but you have to choose only one. This last problem is much more open than the previous 10, and requires a significant amount of effort and research for solving.

SOLVING AND REPORTING THE PROJECT ASSIGNMENT:

A project assignment hand-in is considered complete if all 11 problems are solved. The answers shall be supported by a sufficient amount of intermediate calculations and explanatory text, so that the principles and methods used are clear. All plots, tables, and equations must be numbered consecutively.

The report is assessed as a whole based on the quality of the explanatory text and the correctness of the answers. The last page of the report should be signed by the participant(s). Remember your 'study registration number'. The project assignment needs to be carried out by a team of five students and the individual

contribution to the project work must be clearly indicated for each of the 11 Problem solutions contained in the handed in report. Specify the work contribution to each problem by percentages, e.g. Elisabeth 30% / Peter 30% / Jack 40%.

DEADLINES:

The project report must be submitted via DTU Learn, NO LATER THAN 5pm, 5.Dec.2023.

THE PROJECT ASSIGNMENT

PART 1: KINEMATICS

Figure 1 shows a schematic of the robot with a set of coordinate systems following the Denavit-Hartenberg convention. The dimensions given in the schematic are indicative and you might need to revisit them in PART 5 of this assignment, in order to better represent the actual robot that you will work with.

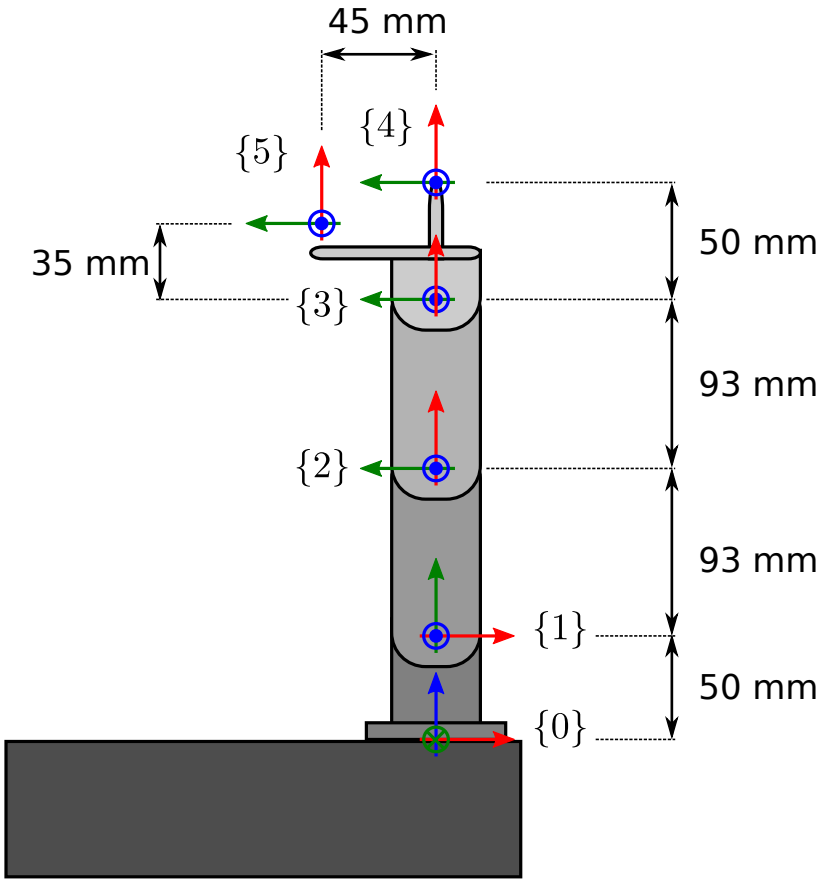


Figure 1: Robot model in its default configuration and fundamental dimensions.

•**Problem 1:** Find, by the use of Figure 1, the direct kinematic transformations, T_4^0 for the robot stylus, and T_5^0 for the robot camera, as function of all joint angles.

•**Problem 2:** Determine the inverse kinematic transformation

$$q = [q_1, q_2, q_3, q_4]^T = f\left(x_4^0, o_4^0\right)$$

where x_4^0 are the first 3 components of the first column of T_4^0, and o_4^0 are the first 3 components of the last column of T_4^0, respectively. Satisfy all position components in o_4^0 and only the last component of x_4^0.

The robot manipulator is now supposed to track, with the stylus tip, a circle with center p_c and radius R. The circle is defined by the equation:

$$p^0(\varphi) = p_c^0 + R \begin{bmatrix} 0 \\ \cos(\varphi) \\ \sin(\varphi) \end{bmatrix}$$

for $0 \leq \varphi \leq 2\pi$.

•**Problem 3:** Find a sequence of 37 robot configurations

$$q^{(j)} = [q_1^{(j)}, q_2^{(j)}, q_3^{(j)}, q_4^{(j)}]^T = f\left(x_4^0 = [?, ?, 0]^T,\, o_4^0 = p^0(\varphi_j)\right), \quad j = 0, 1, \ldots, 36$$

that are necessary for the stylus tip to track 36 equidistant points on a circle with $R = 32\,\text{mm}$ and $p_c^0 = [150, 0, 120]^T\,\text{mm}$. The tracked points start at $\varphi_0 = 0$ and end at $\varphi_{36} = 2\pi$, while the stylus remains horizontal at all configurations.

•**Problem 4:** Determine the Jacobian of the manipulator for the robot end-effector and the Jacobian for the robot camera (as a function of the joint configuration q). Report the numerical results for the two Jacobians at $\varphi = 0$, $\varphi = \pi/2$, $\varphi = \pi$, and $\varphi = 3\pi/2$ along the path studies in Problem 3.

•**Problem 5:** Compute the joint velocities $\dot{q}$ at $\varphi = \pi/2$, along the path from Problem 3, so that the stylus tip velocity is $v_4^0 = [0, -3, 0]\,\text{mm/s}$ and $\dot{x}_4 = [?, ?, 0]$.
Hint: the last quantity $\dot{x}_4$, you have not seen it as such in the course before, so you need to think about how to interpret it in terms of angular velocities.

PART 2: TRAJECTORY PLANNING

In this part, the goal is to plan a trajectory which approximates the circular path from Problem 3 by means of 5 knot-points at φ_0, φ_9, φ_{18}, φ_{27}, φ_{36}.

•**Problem 6:** Use the inverse computed joint configurations $q^{(0)}$, $q^{(9)}$, $q^{(18)}$, $q^{(27)}$, $q^{(36)}$ from Problem 3, to find suitable interpolation polynomials for the following segments:

Segment A: $q^{(0)} \rightarrow q^{(9)}$, $0 \leq t_A \leq 2$ s

$$\begin{aligned} q_1(t_A) &= A_{15} \cdot t_A^5 + A_{14} \cdot t_A^4 + A_{13} \cdot t_A^3 + A_{12} \cdot t_A^2 + A_{11} \cdot t_A + A_{10} \\ q_2(t_A) &= A_{25} \cdot t_A^5 + A_{24} \cdot t_A^4 + A_{23} \cdot t_A^3 + A_{22} \cdot t_A^2 + A_{21} \cdot t_A + A_{20} \\ q_3(t_A) &= A_{35} \cdot t_A^5 + A_{34} \cdot t_A^4 + A_{33} \cdot t_A^3 + A_{32} \cdot t_A^2 + A_{31} \cdot t_A + A_{30} \\ q_4(t_A) &= A_{45} \cdot t_A^5 + A_{44} \cdot t_A^4 + A_{43} \cdot t_A^3 + A_{42} \cdot t_A^2 + A_{41} \cdot t_A + A_{40} \end{aligned}$$

Segment B: $q^{(9)} \rightarrow q^{(18)}$, $0 \leq t_B \leq 2$ s

$$\begin{aligned} q_1(t_B) &= B_{15} \cdot t_B^5 + B_{14} \cdot t_B^4 + B_{13} \cdot t_B^3 + B_{12} \cdot t_B^2 + B_{11} \cdot t_B + B_{10} \\ q_2(t_B) &= B_{25} \cdot t_B^5 + B_{24} \cdot t_B^4 + B_{23} \cdot t_B^3 + B_{22} \cdot t_B^2 + B_{21} \cdot t_B + B_{20} \\ q_3(t_B) &= B_{35} \cdot t_B^5 + B_{34} \cdot t_B^4 + B_{33} \cdot t_B^3 + B_{32} \cdot t_B^2 + B_{31} \cdot t_B + B_{30} \\ q_4(t_B) &= B_{45} \cdot t_B^5 + B_{44} \cdot t_B^4 + B_{43} \cdot t_B^3 + B_{42} \cdot t_B^2 + B_{41} \cdot t_B + B_{40} \end{aligned}$$

Segment C: $q^{(18)} \rightarrow q^{(27)}$, $0 \leq t_C \leq 2$ s

$$\begin{aligned} q_1(t_C) &= C_{15} \cdot t_C^5 + C_{14} \cdot t_C^4 + C_{13} \cdot t_C^3 + C_{12} \cdot t_C^2 + C_{11} \cdot t_C + C_{10} \\ q_2(t_C) &= C_{25} \cdot t_C^5 + C_{24} \cdot t_C^4 + C_{23} \cdot t_C^3 + C_{22} \cdot t_C^2 + C_{21} \cdot t_C + C_{20} \\ q_3(t_C) &= C_{35} \cdot t_C^5 + C_{34} \cdot t_C^4 + C_{33} \cdot t_C^3 + C_{32} \cdot t_C^2 + C_{31} \cdot t_C + C_{30} \\ q_4(t_C) &= C_{45} \cdot t_C^5 + C_{44} \cdot t_C^4 + C_{43} \cdot t_C^3 + C_{42} \cdot t_C^2 + C_{41} \cdot t_C + C_{40} \end{aligned}$$

Segment D: $q^{(27)} \rightarrow q^{(36)}$, $0 \leq t_D \leq 2$ s

$$\begin{aligned} q_1(t_D) &= D_{15} \cdot t_D^5 + D_{14} \cdot t_D^4 + D_{13} \cdot t_D^3 + D_{12} \cdot t_D^2 + D_{11} \cdot t_D + D_{10} \\ q_2(t_D) &= D_{25} \cdot t_D^5 + D_{24} \cdot t_D^4 + D_{23} \cdot t_D^3 + D_{22} \cdot t_D^2 + D_{21} \cdot t_D + D_{20} \\ q_3(t_D) &= D_{35} \cdot t_D^5 + D_{34} \cdot t_D^4 + D_{33} \cdot t_D^3 + D_{32} \cdot t_D^2 + D_{31} \cdot t_D + D_{30} \\ q_4(t_D) &= D_{45} \cdot t_D^5 + D_{44} \cdot t_D^4 + D_{43} \cdot t_D^3 + D_{42} \cdot t_D^2 + D_{41} \cdot t_D + D_{40} \end{aligned}$$

Determine the coefficients A_{ij}, B_{ij}, C_{ij}, D_{ij} so that

$$\begin{aligned} v(t_A{=}0) = [0,0,0]^T \text{mm/s} \quad & \ddot{q}(t_A{=}0) = [0,0,0,0]^T \text{ rad/s}^2 \\ v(t_A{=}2) = v(t_B{=}0) = [0,-27,0]^T \text{mm/s} \quad & \ddot{q}(t_A{=}2) = \ddot{q}(t_B{=}0) = [0,0,0,0]^T \text{ rad/s}^2 \\ v(t_B{=}2) = v(t_C{=}0) = [0,0,-27]^T \text{mm/s} \quad & \ddot{q}(t_B{=}2) = \ddot{q}(t_C{=}0) = [0,0,0,0]^T \text{ rad/s}^2 \\ v(t_C{=}2) = v(t_D{=}0) = [0,27,0]^T \text{mm/s} \quad & \ddot{q}(t_C{=}2) = \ddot{q}(t_D{=}0) = [0,0,0,0]^T \text{ rad/s}^2 \\ v(t_D{=}2) = [0,0,0]^T \text{mm/s} \quad & \ddot{q}(t_D{=}2) = [0,0,0,0]^T \text{ rad/s}^2 \end{aligned}$$

Hint #1: End-effector velocities v need to be converted to joint velocities $\dot{q}$
Hint #2: Requiring zero joint acceleration at knot points is an arbitrary condition used just for simplicity

•**Problem 7:** Plot the actual path of the end-effector for the entire period from $t = 0$ to 8 s for the interpolated trajectory from Problem 6 and compare it to the exact desired circular path. Try to improve the approximation either by using more knot-points or by using different interpolation functions than those found in Problem 6.

PART 3: SINGULARITIES AND STATICS

•**Problem 8:** Plot the condition number of the Jacobian matrix of the manipulator along the path from Problem 3 as well as along the actual path from Problem 6 or 7, and evaluate if the path includes any singularities.
Hint: The condition number of a matrix is the ratio of the largest to the smallest eigenvalue. The larger the value the closest the matrix to being singular. You can use the "cond" function in Matlab or "numpy.linalg.cond" in Python.

•**Problem 9:** Neglecting the own mass of the robot arm, and assuming a weight of 1 N acting on the end-effector along the negative z_0 direction, calculate and plot all joint torques τ_1, τ_2, τ_3, τ_4 as a function of the position $\varphi \in [0, 2\pi]$. Hint: Neglect friction and other losses.

PART 4: DYNAMICS

Figure 2 shows the positions of the centers of mass for the four links and defines the corresponding principal axes of inertia.

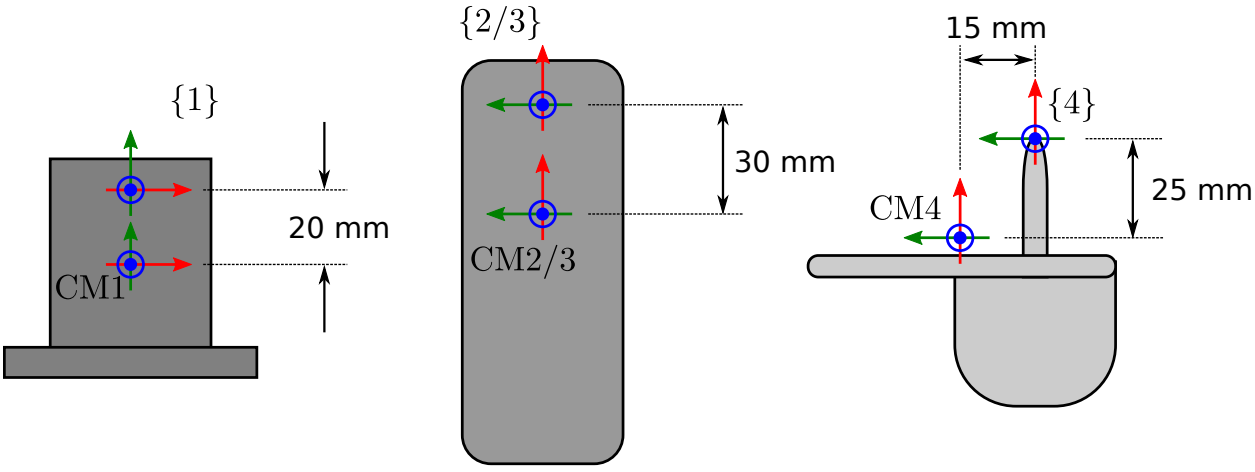


Figure 2: Centers of mass and principal axes of inertia for links 1, 2, 3 and 4.

The mass of link 1 is 60 g and its matrix of inertia in the frame CM1 is in the form:

$$\bar{D}_1 = \begin{bmatrix} I_0 & 0 & 0 \\ 0 & 0.4I_0 & 0 \\ 0 & 0 & 0.9I_0 \end{bmatrix}$$

The masses of links 2 and 3 are equal to 80 g and their matrices of inertia respectively in the frames CM2 and CM3 are in the form:

$$\bar{D}_2 = \bar{D}_3 = \begin{bmatrix} 0.45I_0 & 0 & 0 \\ 0 & 1.4I_0 & 0 \\ 0 & 0 & 1.2I_0 \end{bmatrix}$$

Finally, the mass of link 4 (including the camera) is 40 g and its matrix of inertia in the frame CM4 is in the form:

$$\bar{D}_4 = \begin{bmatrix} 0.5I_0 & 0 & 0 \\ 0 & 0.5I_0 & 0 \\ 0 & 0 & 0.5I_0 \end{bmatrix}$$

•**Problem 10:** Provide a rough but realistic estimate for I_0 based on the dimensions and masses of the links. Then derive the dynamic system for the robot arm in its standard form

$$D(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = \tau$$

and plot the required joint torques τ for the trajectory of Problem 6 or 7.

PART 5: COMPUTER VISION AND CONTROL

The robot arm is implemented with 4 robotis servos of type AX-12A which can be controlled from Matlab or Python through a usb port serial communication protocol. The servo specifications are available at: `https://emanual.robotis.com/docs/en/dxl/ax/ax-12a/` and the communication API is based on a Port Handler object for Matlab) or Python and a Packet Handler object for Matlab) or Python. You will receive some code template for communicating with the robot.
In this part of the assignment you are expected to use the actual robot arm, and a camera mounted at the position of frame 5, in order to perform some complex tasks that involve image recognition.
Choose and implement one of Problems 11a, 11b, 11c, 11d, or a problem of similar complexity after agreement with the course responsibles.

•**Problem 11a:** Create a software which enables the robot arm to recognize the keys of a keyboard that is placed in front of it and it types "hello world".

•**Problem 11b:** Create a software which enables the robot arm to recognize all red "smarties" among a cluster of colourful "smarties" on a surface in front of it, and to probe all of them with its stylus.

•**Problem 11c:** Create a software which enables the robot arm to track a black line drawn on an A4 page, placed in front of the robot, with the tip of its stylus.

•**Problem 11d:** Create a software which enables the robot arm to enter its stylus through a ring that you place in front of it.

For any of the problems that you choose to solve, you must provide details about the theory behind all necessary calculations, the code implementation (in Matlab or Python) and a video recording demonstrating how well the robot is performing the desired task.

Hint #1: Feel free to refine the dimensions and frame positions that you have received in Figure 1, in order to match the actual hardware you are using.

Hint #2: For extracting depth information with only 1 camera, you can combine the information from images from multiple poses of the end-effector.

Annex A: Rotation matrix rate to angular velocities

For a frame oriented according to a rotation matrix

$$R(t) = \begin{bmatrix} R_{xx} & R_{xy} & R_{xz} \\ R_{yx} & R_{yy} & R_{yz} \\ R_{zx} & R_{zy} & R_{zz} \end{bmatrix}$$

and rotating with an angular velocity

$$\omega(t) = \begin{pmatrix} \omega_x \\ \omega_y \\ \omega_z \end{pmatrix}$$

the temporal rate of the rotation matrix is

$$\frac{dR}{dt} = \dot{R} = S(\omega) \cdot R = \begin{bmatrix} R_{zx}\omega_y - R_{yx}\omega_z & R_{zy}\omega_y - R_{yy}\omega_z & R_{zz}\omega_y - R_{yz}\omega_z \\ R_{xx}\omega_z - R_{zx}\omega_x & R_{xy}\omega_z - R_{zy}\omega_x & R_{xz}\omega_z - R_{zz}\omega_x \\ R_{yx}\omega_x - R_{xx}\omega_y & R_{yy}\omega_x - R_{xy}\omega_y & R_{yz}\omega_x - R_{xz}\omega_y \end{bmatrix}$$

Recall that the columns of the rotation matrix R correspond to the frame axes unit vectors x, y and z. Therefore, the temporal rate of the x-axis unit vector is

$$\dot{x} = \begin{pmatrix} R_{zx}\omega_y - R_{yx}\omega_z \\ R_{xx}\omega_z - R_{zx}\omega_x \\ R_{yx}\omega_x - R_{xx}\omega_y \end{pmatrix}$$

expressed in terms of R and ω components.

Annex B: Use and availability of robots

You can use the robots in the databars during the scheduled time (we will bring the robots to the databars), and outside of the scheduled time (in B326, Room 111).

Please reserve the robots for maximum of 2 hours at a time, using the spreadsheet at this link:
`https://ethercalc.net/w14brd6w4hwn`

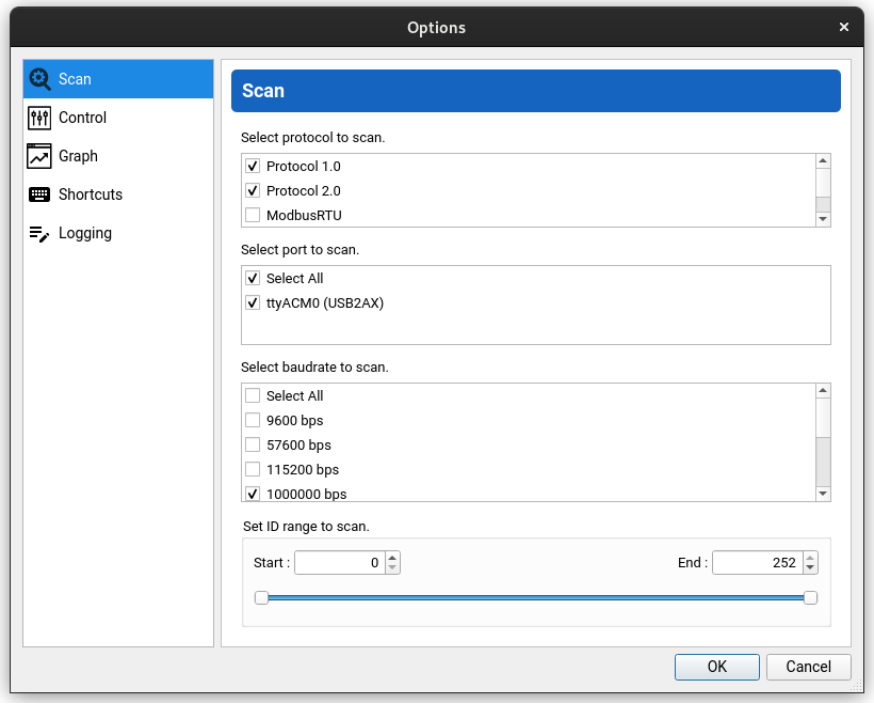
Fill in the sheet corresponding to the correct week number, when you want to book a robot. Write your group number in the preferred time slot.

Annex C: Robot setup in Matlab and Python

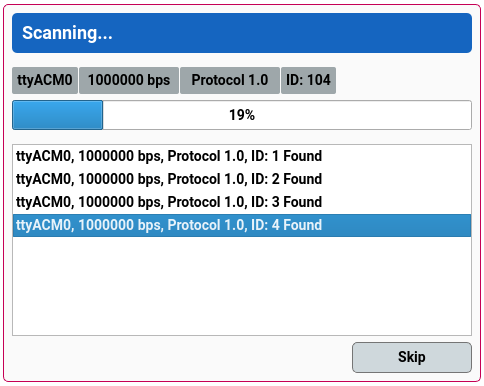
Download and install the Dynamixel Wizard:
`https://emanual.robotis.com/docs/en/software/dynamixel/dynamixel_wizard2/`

Turn on the CourseBot and connect its usb cable to a usb2/3 port of your laptop.

Start the Dynamixel Wizard, open the software 'Options' to make sure that Protocol 1.0 is enabled, the relevant usb port is selected, and in order to select a communication baudrate.



Then select 'Scan' from the menu and make sure that Dynamixel Wizard detects the 4 servos of the robot.



Note down the name of the port that the software has detected, e.g. 'ttyACM0' in Linux or 'COM1' in Windows. You will need this to connect to the servos from your Python or Matlab script.

Go to
https://emanual.robotis.com/docs/en/software/dynamixel/dynamixel_sdk/download/
and download a Dynamixel SDK version that includes support for Python or Matlab.

Then follow the installation instructions for Windows
https://emanual.robotis.com/docs/en/software/dynamixel/dynamixel_sdk/library_setup/python_windows/

or Linux
https://emanual.robotis.com/docs/en/software/dynamixel/dynamixel_sdk/library_setup/python_linux/

Matlab

You can start your Matlab script with library import, definition of constants, and initialization taking inspiration from the following code snippet

```
[~, ~] = loadlibrary('dxl_x64_c', 'dynamixel_sdk.h', 'addheader',...
                     'port_handler.h', 'addheader', 'packet_handler.h');

ADDR_MX_TORQUE_ENABLE         = 24;
ADDR_MX_CW_COMPLIANCE_MARGIN  = 26;
ADDR_MX_CCW_COMPLIANCE_MARGIN = 27;
ADDR_MX_CW_COMPLIANCE_SLOPE   = 28;
ADDR_MX_CCW_COMPLIANCE_SLOPE  = 29;
ADDR_MX_GOAL_POSITION         = 30;
ADDR_MX_MOVING_SPEED          = 32;
ADDR_MX_PRESENT_POSITION      = 36;
ADDR_MX_PUNCH                 = 48;
PROTOCOL_VER                  = 1.0;
DXL_IDS                       = [1,2,3,4];
DEVICENAME                    = '/dev/ttyACM0';
BAUDRATE                      = 1000000;
TORQUE_ENABLE                 = 1;
TORQUE_DISABLE                = 0;
port_num = portHandler(DEVICENAME);
packetHandler();
openPort(port_num)
setBaudRate(port_num, BAUDRATE)
for DXL_ID=DXL_IDS
  write1ByteTxRx(port_num, PROTOCOL_VER, DXL_ID, ADDR_MX_TORQUE_ENABLE, TORQUE_ENABLE);
  write2ByteTxRx(port_num, PROTOCOL_VER, DXL_ID, ADDR_MX_CW_COMPLIANCE_MARGIN, 0)
  write2ByteTxRx(port_num, PROTOCOL_VER, DXL_ID, ADDR_MX_CCW_COMPLIANCE_MARGIN, 0)
  write1ByteTxRx(port_num, PROTOCOL_VER, DXL_ID, ADDR_MX_CW_COMPLIANCE_SLOPE, 32)
  write1ByteTxRx(port_num, PROTOCOL_VER, DXL_ID, ADDR_MX_CCW_COMPLIANCE_SLOPE, 32)
  write2ByteTxRx(port_num, PROTOCOL_VER, DXL_ID, ADDR_MX_MOVING_SPEED, 100)
end
```

For using the useb camera you need to detect the camera with the Matlab command:
`webcamlist()`
Please make sure to identify and select the correct camera device. Then, you can create the webcam object using the correct identifier, e.g. with
`cam = webcam(2);`

You can find more inspiration about how to use the Robotis servos and the camera in Matlab in the following code repository:
`https://github.com/MonsisGit/MyRobot`
Please note that you are not supposed to use the code from the above repository as is, but just as an inspiration. Please also ignore the "python" folder in this repository, it does not contain any functioning code.

Python

You can start your Python script with library import, definition of constants, and initialization taking inspiration from the following code snippet

```
import dynamixel_sdk as dxl

ADDR_MX_TORQUE_ENABLE         = 24
ADDR_MX_CW_COMPLIANCE_MARGIN  = 26
ADDR_MX_CCW_COMPLIANCE_MARGIN = 27
ADDR_MX_CW_COMPLIANCE_SLOPE   = 28
ADDR_MX_CCW_COMPLIANCE_SLOPE  = 29
ADDR_MX_GOAL_POSITION         = 30
ADDR_MX_MOVING_SPEED          = 32
ADDR_MX_PRESENT_POSITION      = 36
ADDR_MX_PUNCH                 = 48
PROTOCOL_VERSION              = 1.0
DXL_IDS                       = [1,2,3,4]
DEVICENAME                    = '/dev/ttyACM0'
BAUDRATE                      = 1000000
TORQUE_ENABLE                 = 1
TORQUE_DISABLE                = 0
portHandler = dxl.PortHandler(DEVICENAME)
packetHandler = dxl.PacketHandler(PROTOCOL_VERSION)
portHandler.openPort()
portHandler.setBaudRate(BAUDRATE)
for DXL_ID in DXL_IDS:
  packetHandler.write1ByteTxRx(portHandler, DXL_ID, ADDR_MX_TORQUE_ENABLE, TORQUE_ENABLE)
  packetHandler.write2ByteTxRx(portHandler, DXL_ID, ADDR_MX_CW_COMPLIANCE_MARGIN, 0)
  packetHandler.write2ByteTxRx(portHandler, DXL_ID, ADDR_MX_CCW_COMPLIANCE_MARGIN, 0)
  packetHandler.write1ByteTxRx(portHandler, DXL_ID, ADDR_MX_CW_COMPLIANCE_SLOPE, 32)
  packetHandler.write1ByteTxRx(portHandler, DXL_ID, ADDR_MX_CCW_COMPLIANCE_SLOPE, 32)
  packetHandler.write2ByteTxRx(portHandler, DXL_ID, ADDR_MX_MOVING_SPEED, 100)
```

For the computer vision part you can base your implementation on the OpenCV libary:
https://en.wikibooks.org/wiki/Applied_Robotics/Sensors_and_Perception/Open_CV/Basic_OpenCV_Tutorial