University of Belgrade
School of Electrical Engineering

Robotics and Automatization (13E053RA)

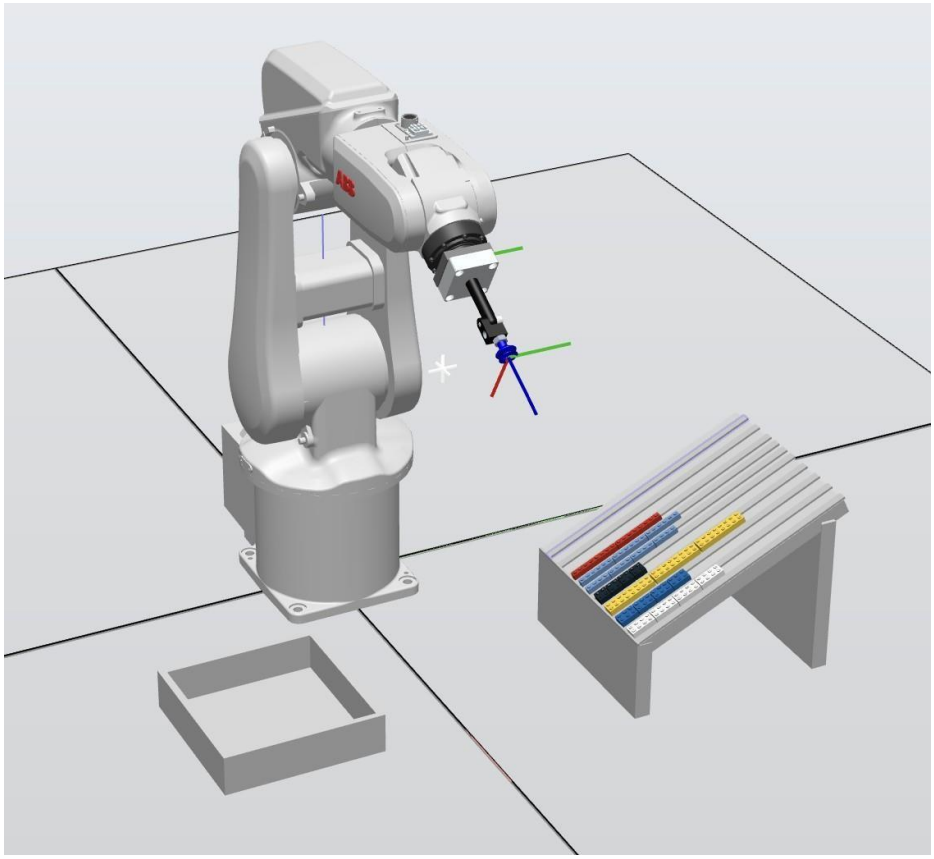
First homework assignment

Students: Cvetković Dositej 20/0224
Matea Koković 19/0004

Belgrade, September 2024

Task

It is necessary to implement an application in which the ABB IRB120/Denso VS6556 robot will perform the packaging process for Lego bricks. The layout of the working environment is shown in Figure 1. In close proximity to the robot, there is a ramp with Lego bricks, consisting of 7 different types of bricks. The Lego bricks need to be packed into a box in front of the robot.



Picture 1.

The dimensions of the box are → (W × L × H) = (200 × 200 × 50 mm).

Table 1 contains a list of bricks with their dimensions. An unlimited number of Lego bricks of each type is available.

The user has the option to select the toys that the robot should pack into the corresponding boxes. The user is offered three types of toys, which consist of different numbers of Lego bricks. Table 2 shows the composition of all the toys.

It is possible to pack three toys of type 1 into the box, or two toys of type 2 and one toy of type 3, or two toys of type 3.

The user can select a maximum of 10 toys of different types. The user enters their selection of toys and their quantity using the Teach Pendant or through the appropriate interface.

Table 1: List of Lego bricks

Type of Lego brick	Dimensions (H x W x D)
Lego 2x4	$11.4 \times 15.8 \times 31.8\ mm$
Lego 2x3	$11.4 \times 15.8 \times 23.8\ mm$
Lego 2x8	$11.4 \times 15.8 \times 63.8\ mm$
Lego 2x2	$11.4 \times 15.8 \times 15.8\ mm$
Lego 1x4	$11.4 \times 7.8 \times 31.8\ mm$
Lego 1x6	$11.4 \times 7.8 \times 47.8\ mm$
Lego 1x8	$11.4 \times 7.8 \times 63.8\ mm$

Tabela 2: Spisak Lego kockica

Type of toy	Lego 2x4	Lego 2x3	Lego 2x8	Lego 2x2	Lego 1x4	Lego 1x6	Lego 1x8
Toy 1	0	0	0	0	2	1	1
Toy 2	2	3	0	1	1	0	0
Toy 3	2	2	3	2	2	0	1

(the student chooses the method they want to implement communication with the user).

The robot needs to execute the packing process of the selected types of toys. The packing process begins after the user selects the types and number of toys to be packed and verifies that new empty boxes are placed in front of the robot. When everything is ready for operation, the user gives permission for the robot to start the Lego brick packing process.

During the robot's operation in the packing process, it is necessary to activate the corresponding signal indicating that the robot is in active mode by setting the digital output RobotActive to a high logical level. Before and after the packing is completed, this signal should be set to a low logical level as a notification to the operator that they can take the packed box or set a new empty box. When the box is packed, it is necessary to stop the robot (the robot is not in operation) and signal to the user that they need to replace the corresponding box. When the user replaces the box, it is essential to notify the robot so that it can continue packing. During the box replacement, the robot should be in its home position to allow the operator to replace the boxes without interruption.

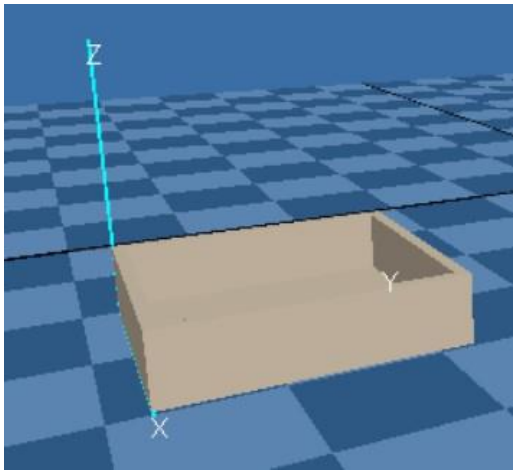
The maximum speed at which the robot can move must not exceed v500, while the robot's speed for approaching objects it needs to manipulate should be v100, as well as when placing bricks in boxes.

Care should be taken regarding how the bricks are placed in the box to ensure the box is evenly filled with the corresponding bricks.

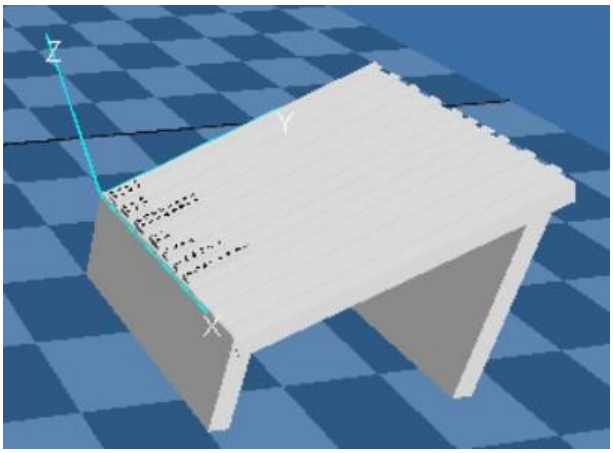
Setting Up Working Coordinate Systems

The coordinate system that defines the position of the workpiece is called the object coordinate system or, in Denso terminology, the work coordinate system. By introducing such coordinate systems defined in relation to the robot's base, programming the robot becomes easier, and it allows for simple adaptation to a new working environment if the robot itself is moved.

Since there are two positions from which the robot picks/packages the Lego bricks, we will define two working coordinate systems.



Picture 2. – Work 1



Picture 3. – Work 2

The positions of the working coordinate systems in relation to the robot's base coordinate system are presented in the table.

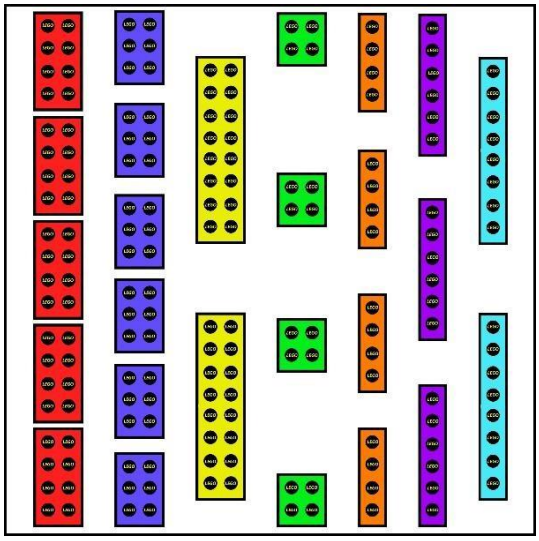
work	x	y	z	R_x	R_y	R_z
1	200	-575	0	0	0	0
2	300	300	100	25	0	0

The positions where the vacuum gripper is brought when lifting Lego bricks are defined by points in the work 2 coordinate system:

Lego brick 2x4: $P[1] = (20, 20, 10, 180, 0, -155)$
Lego brick 2x3: $P[2] = (45, 16, 10, 180, 0, -155)$
Lego brick 2x8: $P[3] = (75, 35, 10, -180, 0, -155)$
Lego brick 2x2: $P[4] = (100, 10, 10, -180, 0, -155)$

Lego brick 1x4: $P[5] = (125, 20, 10, 180, 0, -155)$
Lego brick 1x6: $P[6] = (140, 25, 10, 180, 0, -155)$
Lego brick 1x8: $P[7] = (160, 35, 10, \text{-}180, 0, -155)$

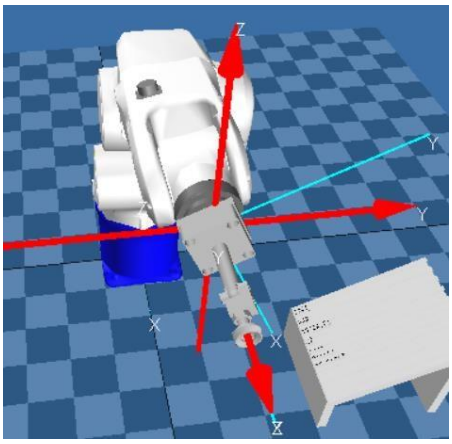
Within the work 1 coordinate system, it is necessary to define the positions of the Lego bricks as they are packed. Our solution is illustrated in Figure 4, where each brick has its own position. If a position for a specific Lego brick is no longer available, stacking occurs by placing one brick on top of another.



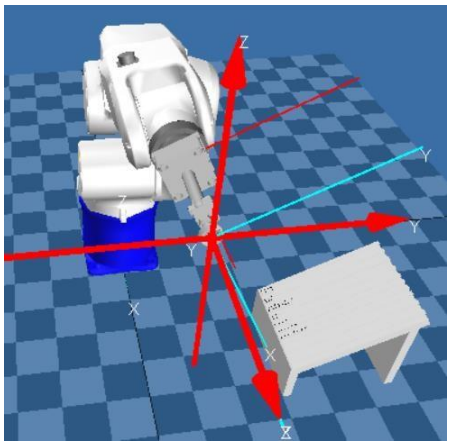
Picture 4.

Setting Up Tool Coordinate Systems

The position of the robot's tool (gripper) is defined in relation to the tool coordinate system, whose coordinate origin is set at the TCP (Tool Center Point) located at the end of the mechanical configuration and is independent of the tool that the robot carries. The coordinate system at the characteristic point of the tool is defined in relation to the basic TCP coordinate system.



Picture 5. – basic TCP coordinate system – tool 0



Picture 6. – Tool 1

Based on Figure 6, it is easy to determine which homogeneous transformations need to be performed to establish the coordinate systems of the vacuum gripper (tool 1).

Tool	x	y	z	R_x	R_y	R_z
1	0	0	130	0	0	0

Robot control

At the beginning of the program, after gaining control over the robot and starting the motors in its joints, we define the initial point where the robot should be located at the beginning and at the end of each cycle.

$$P[0]=(560,0,790,0,90,0)$$

Depending on the action the robot performs (picking or packing), we change its working coordinate system. To ensure the robot operates with minimal idle motion and unnecessary time consumption, each picking/packing point for the Lego bricks is approached using the command approach. The result of this command is positioning the tool vertically above the working point at a height of 100 mm, significantly faster (50%) compared to the speed at which it should approach the Lego brick/box. Contact with the Lego brick is achieved by vertically lowering the tool at a slow speed (10%), facilitated by the move command. After the robot completes the packing, it gently moves away from the box vertically upwards to a safe distance using the depart command.

Activating and deactivating the vacuum gripper actually represents controlling the corresponding pneumatic valve. The gripper is controlled using signals 25 and 26, alternating between activation and deactivation. When output 25 is active and 26 is inactive, the gripper is closed or activated. Conversely, the gripper opens or deactivates with the opposite output combination.

Each time the robot packs a box, a signal must be given to change the box by setting Digital Output 27 to a high logical level for 20 seconds. Additionally, each time the robot is packing/working, Digital Output 24 (RobotActive) is active.

The process of selecting the number of toys of different types utilizes Digital Outputs 9-12, where each output represents one bit, with Digital Output 12 as the highest bit and Digital Output 9 as the lowest. These binary outputs are converted into a decimal number representing the number of toys. The user has 10 seconds to choose the number of toys for each of the three types: the first 10 seconds for type 1, the next 10 seconds for type 2, and the last 10 seconds for type 3. Finally, the program checks whether the total number of toys exceeds 10. If so, the program terminates.

After selecting the number of toys, the user needs to give permission to the robot by setting Digital Output 8 to a high logical level to allow the robot to start working. Once the robot finishes packing one box, Digital Output 8 is automatically set to a low logical level. The user then reactivates Digital Output 8 to a high logical level after replacing the box, with a 20-second window, for the robot to continue working.

The robot only packs when there are enough toys to fill the entire capacity of the box. Partial packing is not possible.

The code just for packing toy 1

```
'!TITLE "<Title>"
PROGRAM domaci_zadatak

takearm
motor on
changetool 1

dim toy1 as integer
dim toy2 as integer
dim toy3 as integer
dim count as integer



'choose number of toy1
delay 10000
if io12 = 0 then
        if io11 = 0 then
                if io10 = 0 then
                        if io9 = 0 then
                                toy1 = 0
                        else
                                toy1 = 1
                        end if
                else
                        if io9 = 0 then
                                toy1 = 2
                        else
                                toy1 = 3
                        end if
                end if
        else
                if io10 = 0 then
                        if io9 = 0 then
                                toy1 = 4
                        else
                                toy1 = 5
                        end if
                else
                        if io9 = 0 then
                                toy1 = 6
                        else
                                toy1 = 7
                        end if
                end if
```

```
        end if
else
        if io11 = 0 then
                if io10 = 0 then
                        if io9 = 0 then
                                toy1 = 8
                        else
                                toy1 = 9
                        end if
                else
                        if io9 = 0 then
                                toy1 = 10
                        end if
                end if
        end if
end if

'choose number of toy2
delay 10000
if io12 = 0 then
        if io11 = 0 then
                if io10 = 0 then
                        if io9 = 0 then
                                toy2 = 0
                        else
                                toy2 = 1
                        end if
                else
                        if io9 = 0 then
                                toy2 = 2
                        else
                                toy2 = 3
                        end if
                end if
        else
                if io10 = 0 then
                        if io9 = 0 then
                                toy2 = 4
                        else
                                toy2 = 5
                        end if
                else
                        if io9 = 0 then
                                toy2 = 6
                        else
                                toy2 = 7
                        end if
```

```
                end if
        end if
else
        if io11 = 0 then
                if io10 = 0 then
                        if io9 = 0 then
                                toy2 = 8
                        else
                                toy2 = 9
                        end if
                else
                        if io9 = 0 then
                                toy2 = 10
                        end if
                end if
        end if
end if

'choose number of toy3
delay 10000
if io12 = 0 then
        if io11 = 0 then
                if io10 = 0 then
                        if io9 = 0 then
                                toy3 = 0
                        else
                                toy3 = 1
                        end if
                else
                        if io9 = 0 then
                                toy3 = 2
                        else
                                toy3 = 3
                        end if
                end if
        else
                if io10 = 0 then
                        if io9 = 0 then
                                toy3 = 4
                        else
                                toy3 = 5
                        end if
                else
                        if io9 = 0 then
                                toy3 = 6
                        else
                                toy3 = 7
```

```
                end if
            end if
        end if
else
        if io11 = 0 then
            if io10 = 0 then
                if io9 = 0 then
                    toy3 = 8
                else
                    toy3 = 9
                end if
            else
                if io9 = 0 then
                    toy3 = 10
                end if
            end if
        end if
end if

count = toy1 + toy2 + toy3
if count > 10 then
        goto *toomany
end if

*idle:
reset io8
delay 20000
reset io27
if io8 then

speed 50
        set io24
        delay 100

        if toy1 > 2 then
            changework 2
            'first piece 1x4
            approach p, p5, 100
            set io26
            reset io25
            delay 1000
            speed 10
            move l, @e p5
            speed 50
            set io25
            reset io26
            delay 1000
```

```
depart l, 100

changework 1
approach p, p37, 100
speed 10
move l, @e p37
set io26
reset io25
delay 1000
depart l, 100
speed 50

changework 2
'second piece 1x4
approach p, p5, 100
set io26
reset io25
delay 1000
speed 10
move l, @e p5
speed 50
set io25
reset io26
delay 1000
depart l, 100

changework 1
approach p, p38, 100
speed 10
move l, @e p38
set io26
reset io25
delay 1000
depart l, 100
speed 50

changework 2
'piece  1x6
approach p, p6, 100
set io26
reset io25
delay 1000
speed 10
move l, @e p6
speed 50
set io25
reset io26
```

```
delay 1000
depart l, 100

changework 1
approach p, p44, 100
speed 10
move l, @e p44
set io26
reset io25
delay 1000
depart l, 100
speed 50

changework 2
'piece  1x8
approach p, p7, 100
set io26
reset io25
delay 1000
speed 10
move l, @e p7
speed 50
set io25
reset io26
delay 1000
depart l, 100

changework 1
approach p, p48, 100
speed 10
move l, @e p48
set io26
reset io25
delay 1000
depart l, 100
speed 50

'second toy1
changework 2
'first piece 1x4
approach p, p5, 100
set io26
reset io25
delay 1000
speed 10
move l, @e p5
speed 50
```

```
set io25
reset io26
delay 1000
depart l, 100

changework 1
approach p, p39, 100
speed 10
move l, @e p39
set io26
reset io25
delay 1000
depart l, 100
speed 50

changework 2
'second piece 1x4
approach p, p5, 100
set io26
reset io25
delay 1000
speed 10
move l, @e p5
speed 50
set io25
reset io26
delay 1000
depart l, 100

changework 1
approach p, p40, 100
speed 10
move l, @e p40
set io26
reset io25
delay 1000
depart l, 100
speed 50

changework 2
'piece  1x6
approach p, p6, 100
set io26
reset io25
delay 1000
speed 10
move l, @e p6
```

```
speed 50
set io25
reset io26
delay 1000
depart l, 100

changework 1
approach p, p45, 100
speed 10
move l, @e p45
set io26
reset io25
delay 1000
depart l, 100
speed 50

changework 2
'piece  1x8
approach p, p7, 100
set io26
reset io25
delay 1000
speed 10
move l, @e p7
speed 50
set io25
reset io26
delay 1000
depart l, 100

changework 1
approach p, p49, 100
speed 10
move l, @e p49
set io26
reset io25
delay 1000
depart l, 100
speed 50

'third toy1
changework 2
'first piece 1x4
approach p, p5, 100
set io26
reset io25
delay 1000
```

```
speed 10
move l, @e p5
speed 50
set io25
reset io26
delay 1000
depart l, 100

changework 1
approach p, p41, 100
speed 10
move l, @e p41
set io26
reset io25
delay 1000
depart l, 100
speed 50

changework 2
'second piece 1x4
approach p, p5, 100
set io26
reset io25
delay 1000
speed 10
move l, @e p5
speed 50
set io25
reset io26
delay 1000
depart l, 100

changework 1
approach p, p42, 100
speed 10
move l, @e p42
set io26
reset io25
delay 1000
depart l, 100
speed 50

changework 2
'piece  1x6
approach p, p6, 100
set io26
reset io25
```

```
delay 1000
speed 10
move l, @e p6
speed 50
set io25
reset io26
delay 1000
depart l, 100

changework 1
approach p, p46, 100
speed 10
move l, @e p46
set io26
reset io25
delay 1000
depart l, 100
speed 50

changework 2
'piece  1x8
approach p, p7, 100
set io26
reset io25
delay 1000
speed 10
move l, @e p7
speed 50
set io25
reset io26
delay 1000
depart l, 100

changework 1
approach p, p50, 100
speed 10
move l, @e p50
set io26
reset io25
delay 1000
depart l, 100
speed 50

changework 0
'home position
move p, p0
reset io26
```

```
                reset io24
                'indicator to change the box
                set io27

                toy1 = toy1 - 3

                goto *idle
        end if

        motor off
        givearm

else
        motor off
        givearm
end if

*toomany:
END
```