Guidelines to use SCORBOT-ER VII

Robotics

2022-2023 (P2 / S1)

Alberto Vale, João Sequeira

DEEC/IST

References of text, tables and images

1. **SCORBOT-ER VII User Manual**
   2nd Edition, Eshed Robotec, December 1998, ISBN 965-291-033-3

2. **ACL – Advanced Control Language**, Version 1.43, F.44, Reference Guide, 4th edition, Eshed Robotec, January 1995, Catalog #100083 Rev.A

3. **Introduction to the Scorbot ER VII and the Eshed Robotec Pty. Ltd.**
   Advanced Control Language (ACL)
   R. Mahony, Dep. Engineering, ANU, ACT, 0200, Australia.

SCORBOT-ER VII

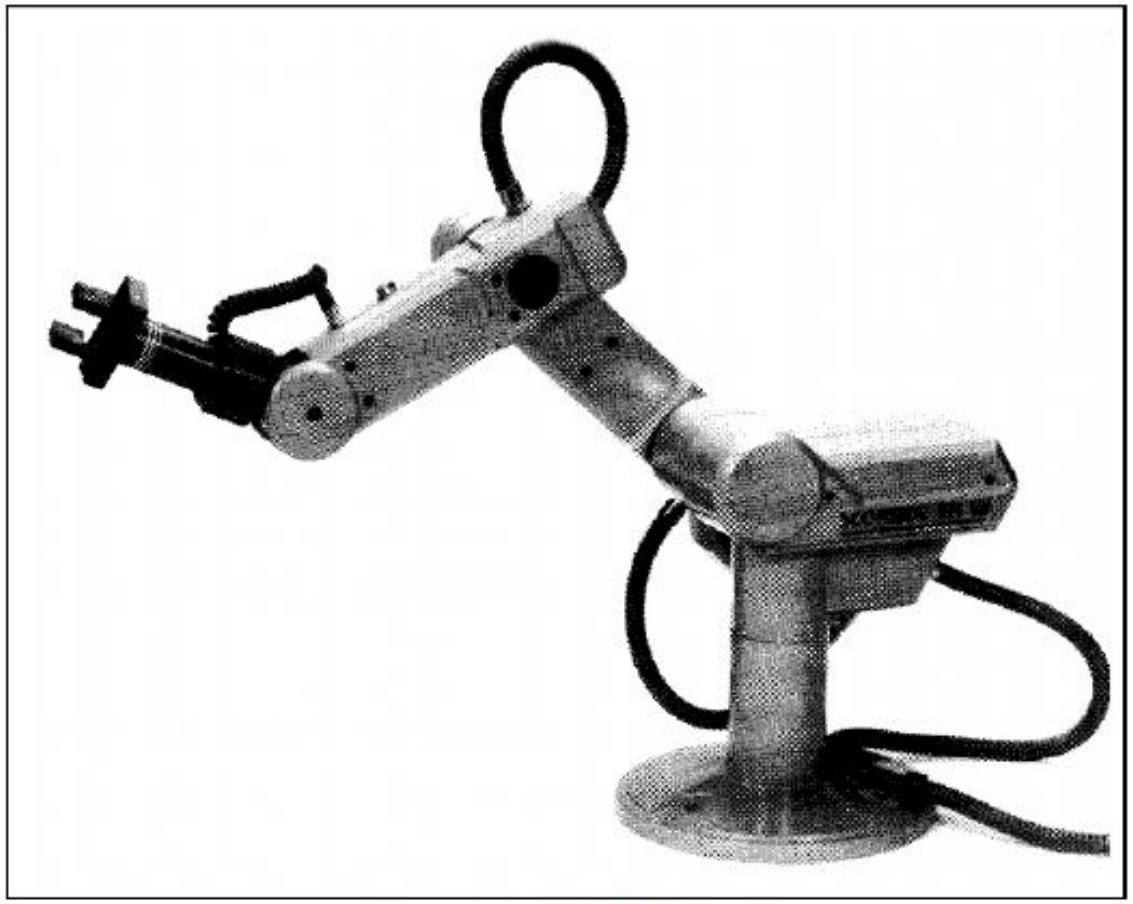


Figure 2-1: SCORBOT-ER VII Robot Arm

SCORBOT-ER VII Setup

- Power button
- Motors enable button
- Teach Pendant

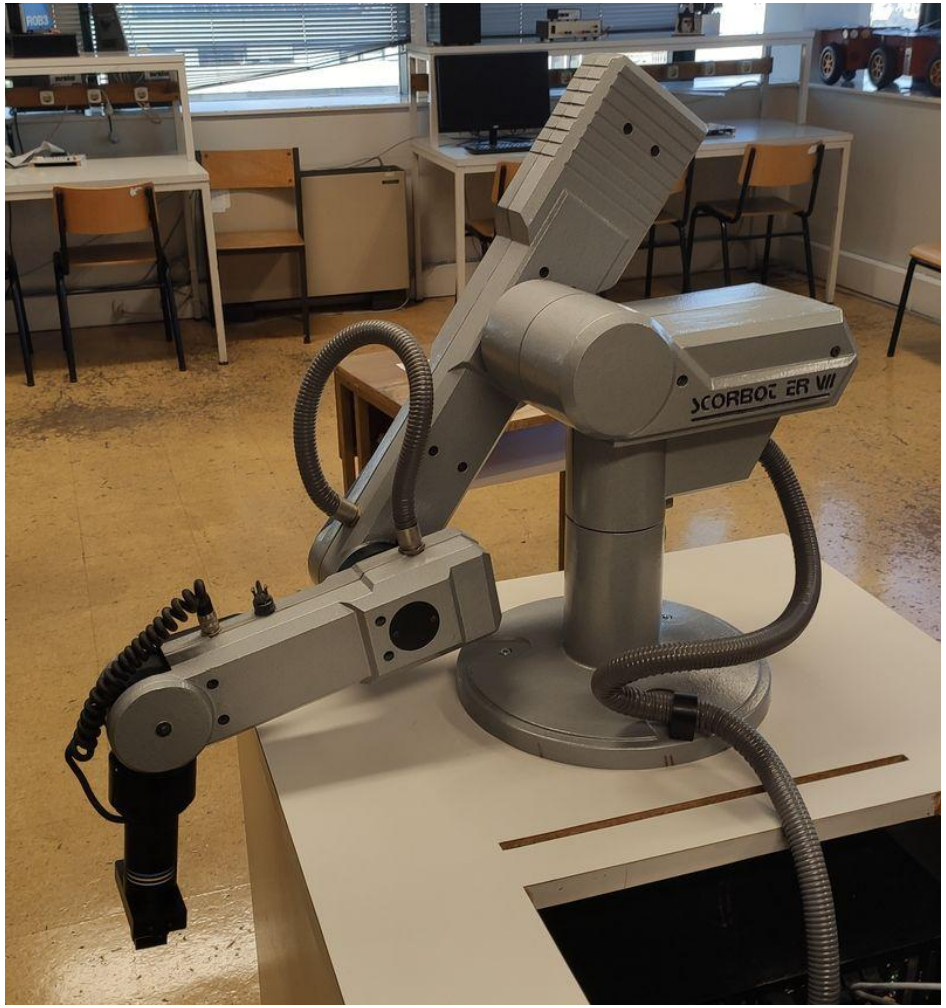
Instructions

1. Check that the manipulator workspace is free of obstacles.
2. Do not enter the robots safety range or touch the robot during operation.
3. Verify that you can reach the red emergency button on the controller. One person should always be in a position to abort control using the emergency switch during operation.
4. Switch on the controller. Activate the motors.
5. Home the robot using the command HOME

The SCORBOT-ER VII is dangerous and can cause severe injury.

USE WITH EXTREME CAUTION.

Risk of clash (examples)



Arm parts, axes and main features

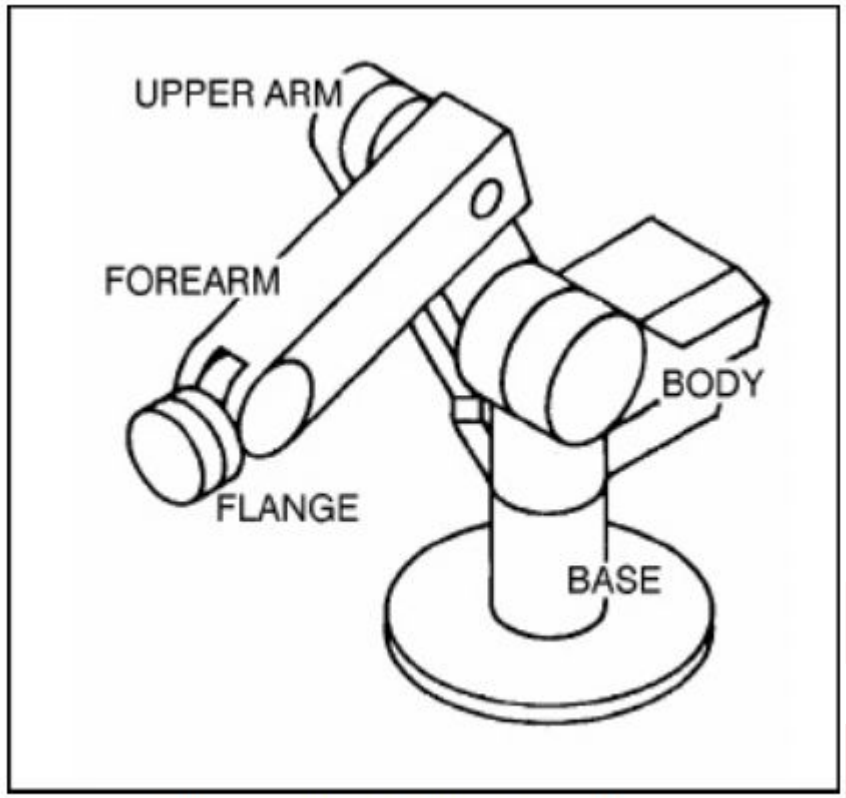


Figure 1-2: Robot Arm Parts

Axis Movement	
Axis 1: Base rotation	250°; 310° user programmable
Axis 2: Shoulder rotation	170°
Axis 3: Elbow rotation	225°
Axis 4: Wrist pitch	180°
Axis 5: Wrist roll	360°

Encoders max values (**experimental values!**)
-31960 to 31950
-16960 to 25972
-28480 to 28942
-27048 to 28133
-31929 ro 31956

Attention: some combinations results in clash!

Maximum Payload	2 kg (4.4 lb.), including gripper
Position Repeatability	±0.2 mm (0.008")
Weight	30 kg (66 lbs)
Maximum Path Velocity	1000 mm/sec (39.4"/sec)

Arm joints and motor locations

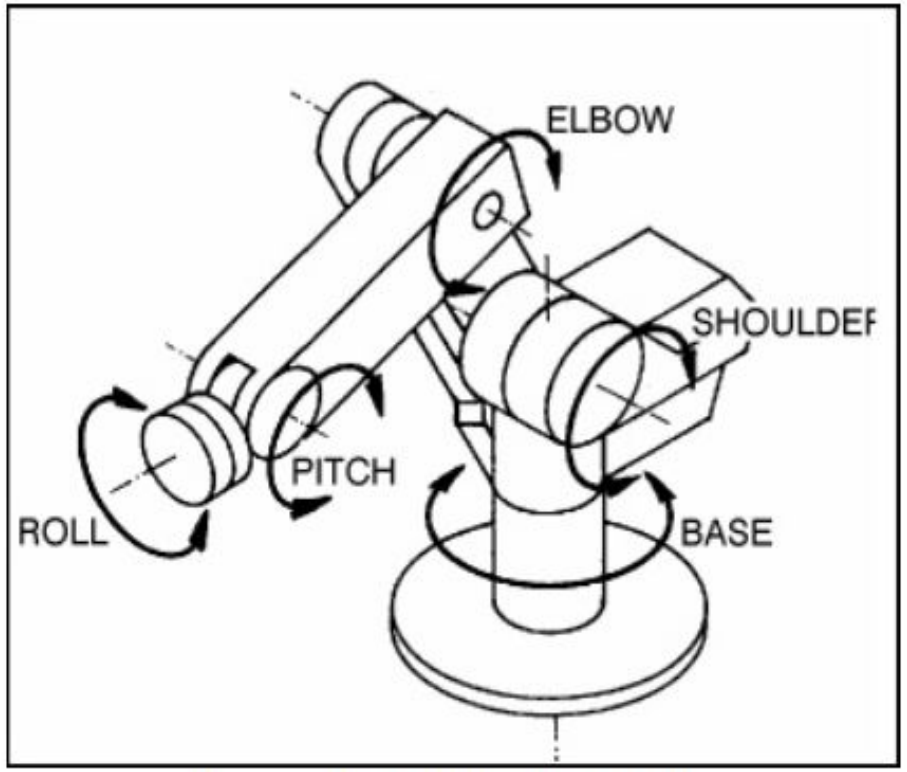


Figure 2-3: Robot Arm Joints

Axis No.	Joint Name	Motion	Motor No.
1	Base	Rotates the body.	1
2	Shoulder	Raises and lowers the upper arm.	2
3	Elbow	Raises and lowers the forearm.	3
4	Wrist Pitch	Raises and lowers the end effector	4
5	Wrist Roll	Rotates the end effector	5

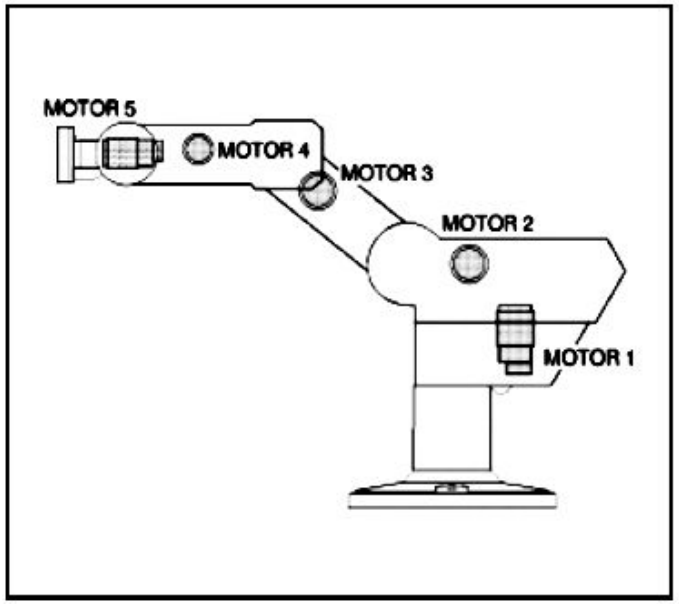


Figure 2-7: Motor Locations

Coordinates and home configuration

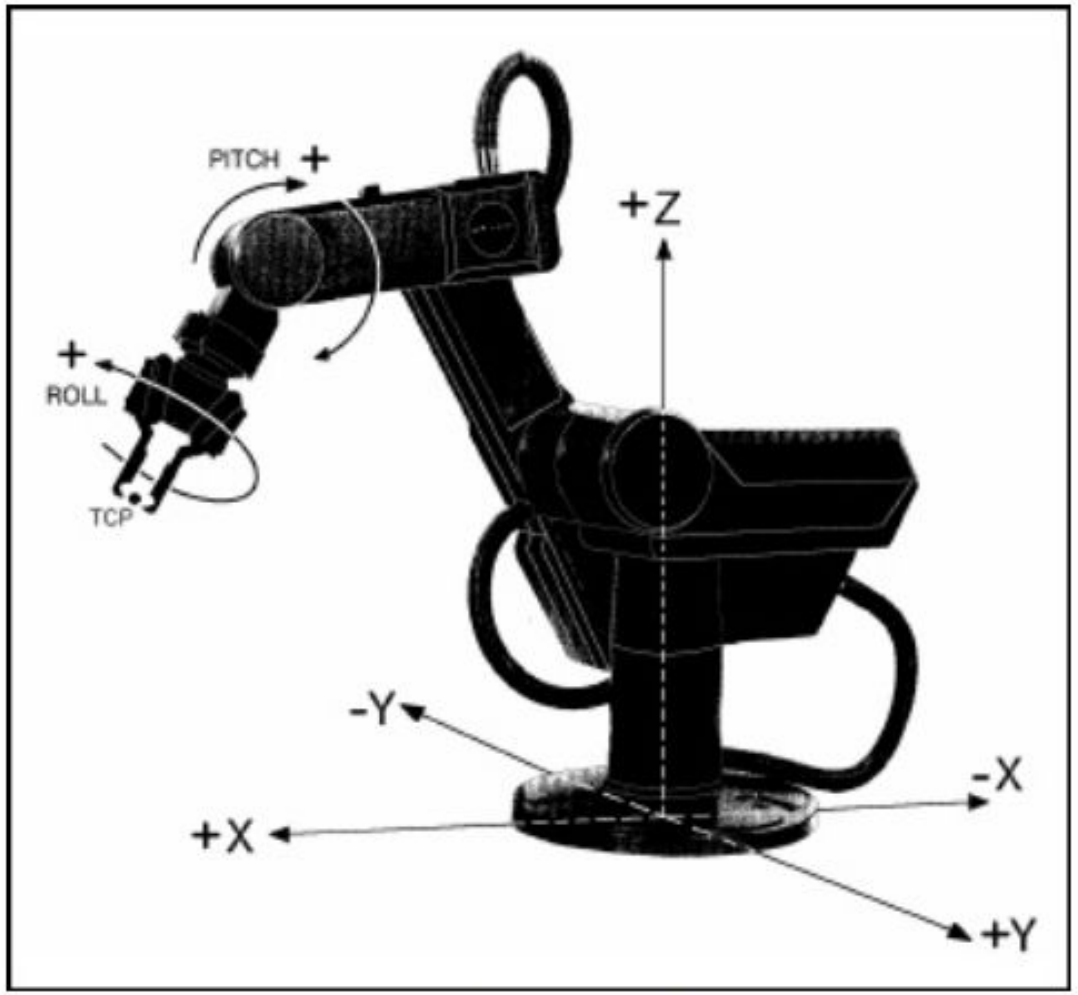
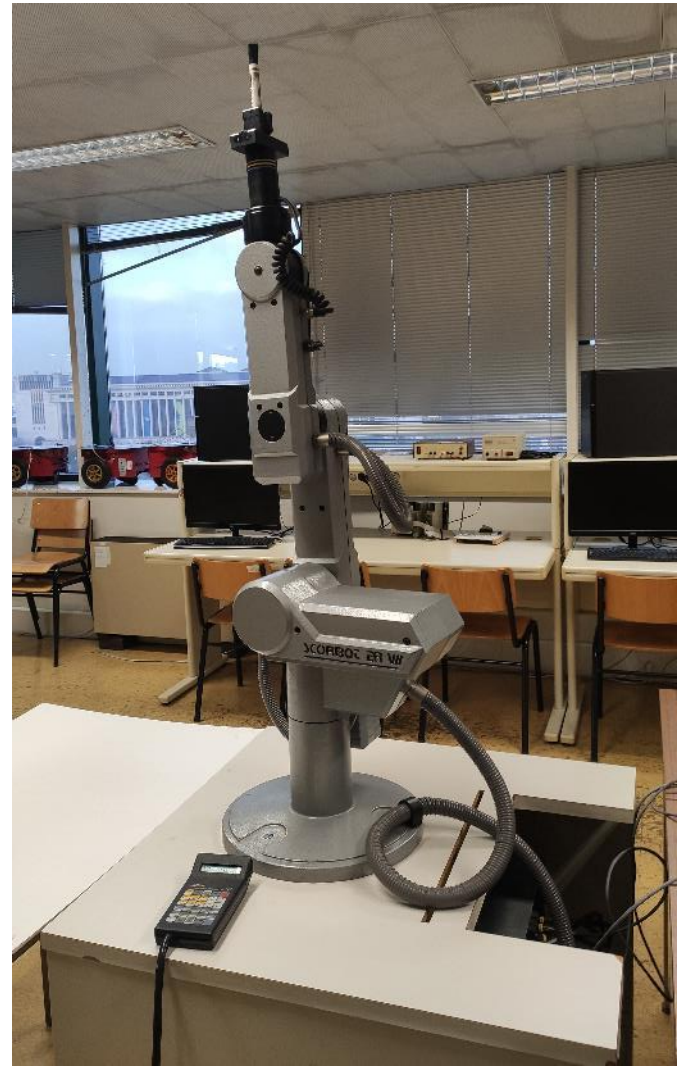


Figure 6-1: Cartesian Coordinates

Home configuration



Dimensions and operation range

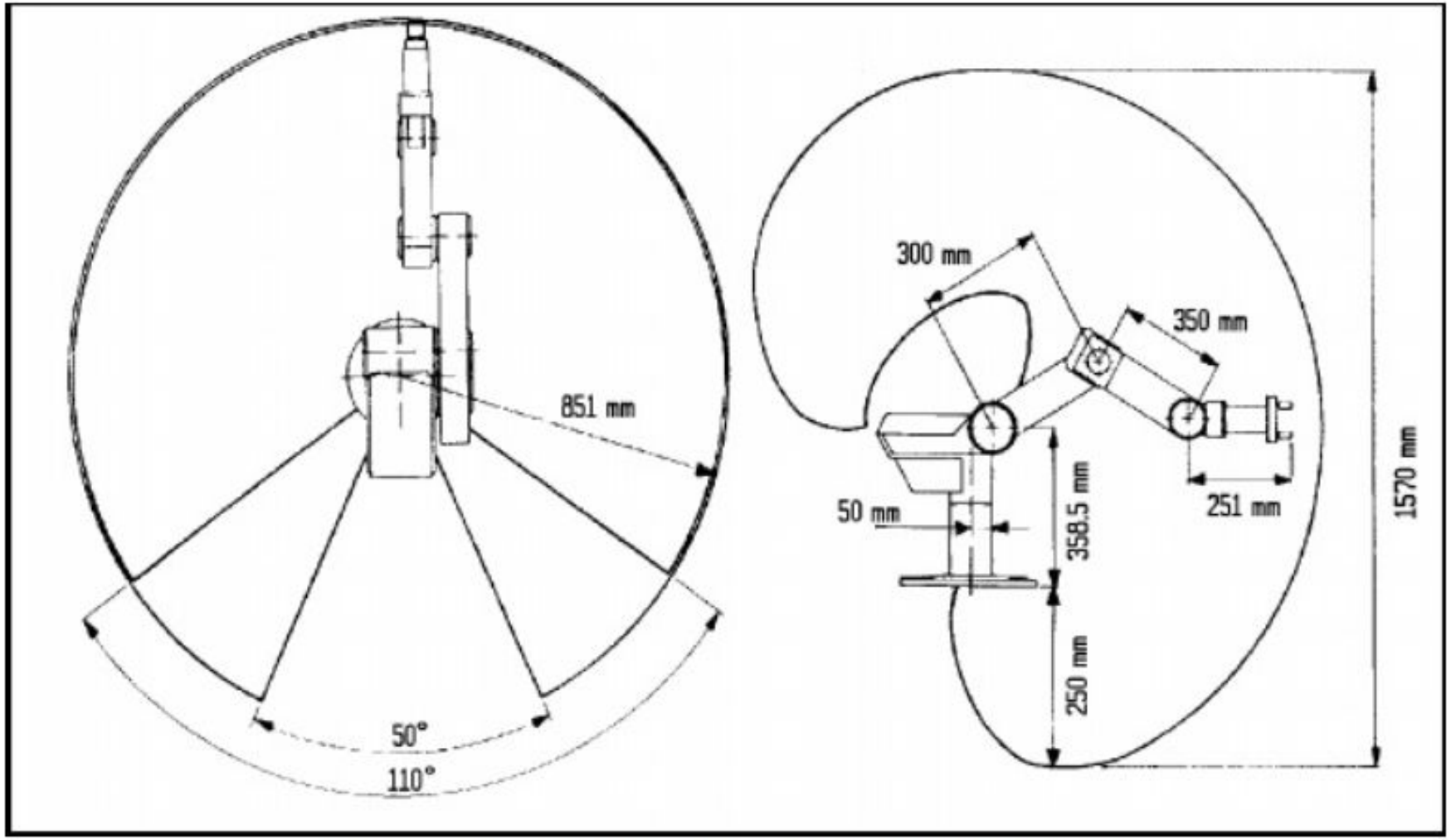
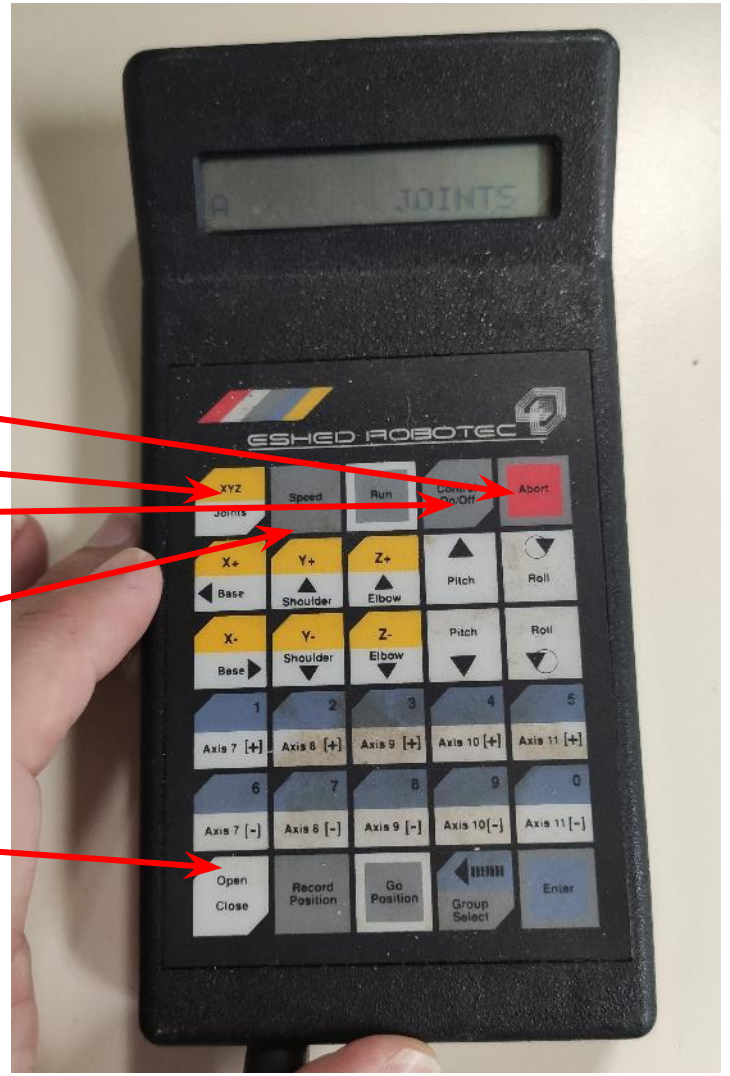


Figure 2-4: Operating Range With Gripper Attached

Teach Pendant/Keypad/Controller

The most important keys:

- **Abort**
- XYZ/Joints mode
- Control on/off
- Speed + number (1%-100%) + Enter
- Open/Close gripper



ACL: Advanced Control Language

ACL has two types of commands:

- **DIRECT** commands are executed as soon as they are entered at the terminal/computer keyboard **[recommended]**
- **EDIT**, or indirect, commands are executed during the running of the programs and routines in which they are used.

- About ACL, see reference 2 in slide 1.

Main DIRECT commands

- **CON:** Control ON for all axes.
- **HELP**: Displays current coordinates of robot arm.
- **HOME**: Searches for microswitch home position, for all robot axes, or specific axis. (takes 1-2 mins)
- **OPEN/CLOSE**: opens/closes the gripper
- **SPEED 50**: sets speed movements of 50% maximum speed.
- **HERE *pos***: records a position, in joint coordinates, according to the current location of the axes.

Main DIRECT commands (cont.)

- **LISTP:** To see a list of the defined positions
- **LISTPV A31**: To view the coordinates of position A31

```
1:0          2:1791       3:2746       4:0          5:-1
X:1690       Y:0          Z:6011       P:-636       R:-1
```

601.1 mm

- 63.6º

  - The first line shows the **joint coordinates**; defined in **encoder counts**.
  - The second line shows the **Cartesian (XYZ) coordinates**.
    X, Y and Z are defined in tenths of millimeters; P (Pitch) and R (Roll) are defined in tenths of degrees.

- **LISTPV POSITION**: Displays current coordinates of robot arm.

Main DIRECT commands (cont.)

- **DEFP *pos***: defines position *pos* for the robot.
- **DIMP *vect[n]***: defines (creates) a vector of n positions. n = 1, …, N
- **DELP *pos***: deletes the position.
- **HERE *pos***: records joint coordinates for current position of axes.
- **SETPV *pos axis var***: changes one joint coordinate of a previously recorded position
- **SETPVC *pos coord var***: Changes one Cartesian coordinate of a previously recorded robot position.
- **SETP *pos2=pos1***: Copies the coordinates and type of pos1 to pos2.

Main DIRECT commands (cont.)

- **DELP A99**: to delete position A99.
- **MOVE A31**: to send the robot to position A31. The robot moves at the current speed setting.
- **MOVE A32 1000**: to send the robot to position A32 in 10 seconds.
- **MOVED A32:** similar to move, but moved ensures that the robot will accurately reach the target position before continuing to the next command.
- **MOVES *POS n1 n2*:** Moves axes smoothly through all consecutive vector positions between position n1 and position n2, at current joint speed. Constant speed between consecutive positions.

Main DIRECT commands (cont.)

- **SHOW ENCO**: Displays the values of all encoders every 0.5 seconds
- **SHOW SPEED**: Displays the current speed settings

(to be updated with more commands…)

Program SCORBOT-ER VII

- See Chapter 7-1 of reference 1 in slide 1.

How to connect with SCORBOT-ER VII

- The communication with SCORBOT-ER VII is done by a **Serial Port (RS232)**.
- A Serial Port is required on the laptop or, instead, a USB <> serial converter
  - [suggestion] buy a USB <> serial converter per group (costs 10 - 20 EUR)

Attention: not all USB <> serial converters are recognized in Windows/Linux

How to operate with SCORBOT-ER VII

1. Communicate with SCORBOT-ER VII to send and receive commands in real time **[recommended]**
2. Program the SCORBOT-ER VII to run offline

Send and receive commands in real time

- Windows software (e.g.: PuTTY)
- Linux shell commands

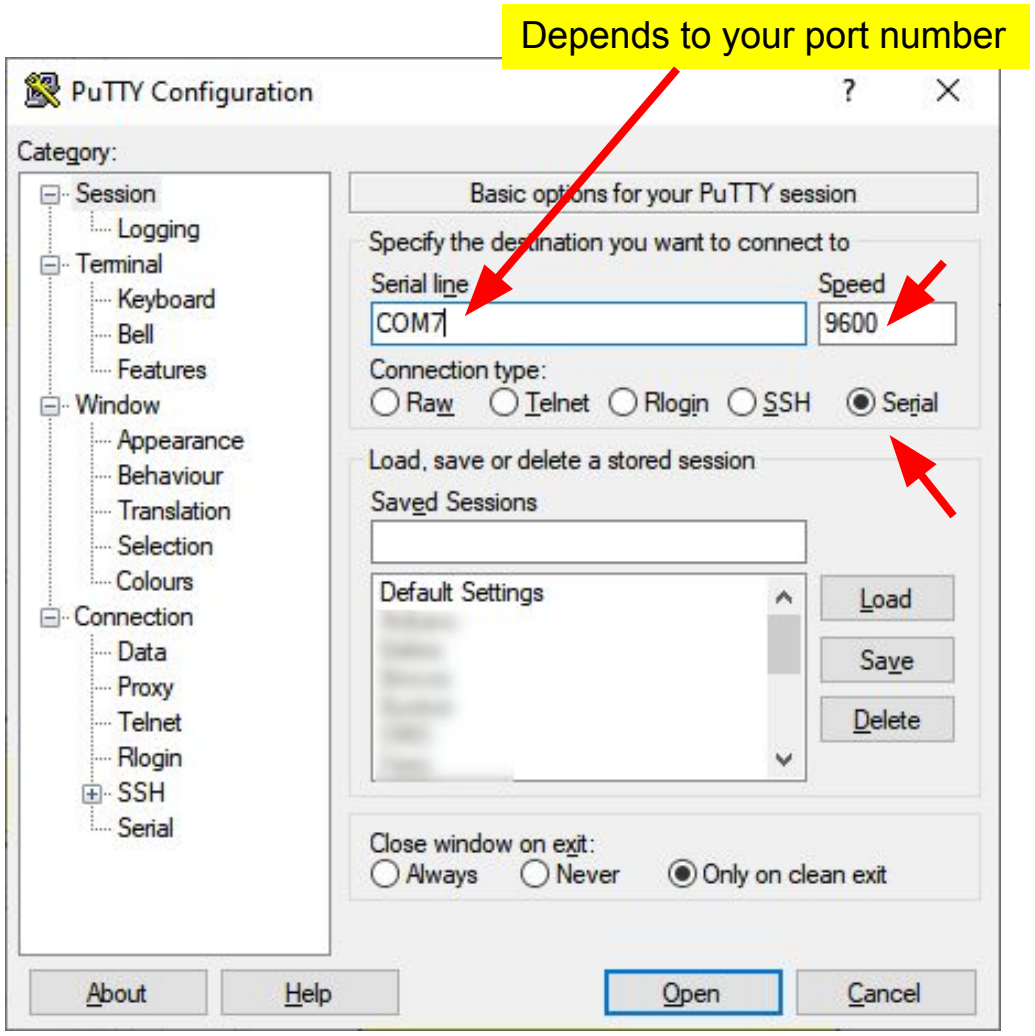
Recommended for quick testing

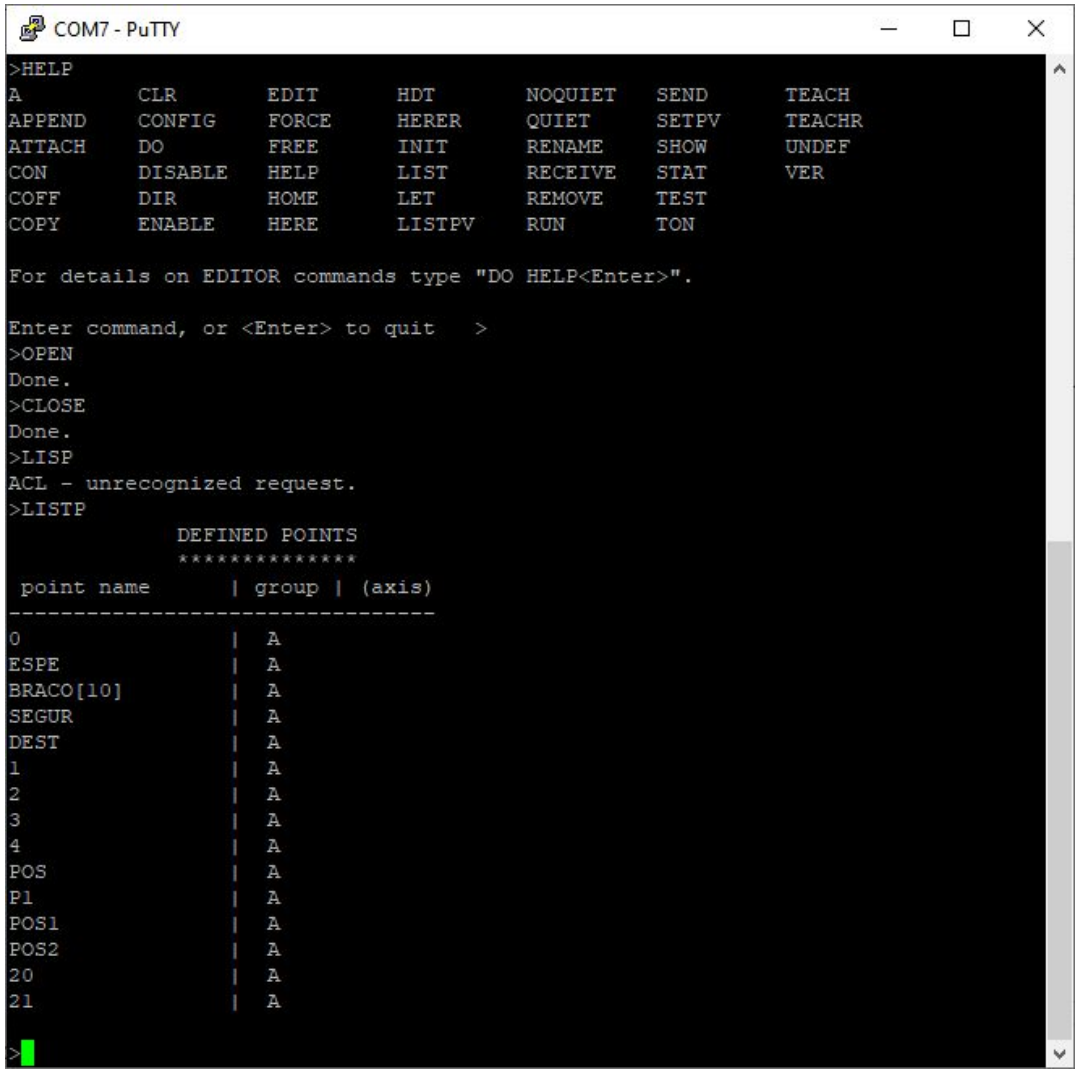
- Popular programming languages:
  - MATLAB
  - Python
  - C/C++
  - other

Recommended for development

Attention: the messages are strings and must end with a “carriage return” (\r)

Example with PuTTY





Example with a shell in Linux

- Suggestion in Linux:

```
chmod o+rw /dev/ttyUSB0

stty 9600 < /dev/ttyUSB0

echo 'MOVE P1\r' > /dev/ttyUSB0

cat < /dev/ttyUSB0
```

Video demo

Also available here:
https://tinyurl.com/2j9eyuq3

Special notes

- To create a vector, for instance, PVECT, it is necessary to created it by "DIMP PVECT[n]", where n is the n is the number of positions (n>=1). Then it is not possible to define each axis of each PVECT[i] by "SETPVC PVECT[i] coord var", for instance "SETPVC PVECT[1] X 5000". To bypass it, run "HERE PVECT[1]" and then "SETPVC PVECT[1] X 5000".

- When it is necessary to modify more than one coordinate of a point, the intermediate modifications must result in a point also inside the workspace of the manipulator. For instance, to modify X and Y of PVECT[1], it is necessary to do "SETPVC PVECT[1] X var_X" and then "SETPVC PVECT[1] Y var_Y". However, when changing the X first, the resulted temporarily point must fit in the workspace.

- After creating a vector, for instance PVECT[...] with 10 positions, it is not possible to run "MOVES PVECT 1 10". The result is an error about the trajectory. A possible solution is to run for each i "TEACH PVECT[i]", then "MOVE PVECT[i]" and finally HERE PVECT[i]" to record again VECT[i] (it seems silly...). Then run again "MOVES PVECT 1 10" and it works! By the way, we should use "MOVES PVECT 1 10 --time_to_execute--".