Міністерство освіти і науки України
Державний вищий навчальний заклад
"Національний гірничий університет"

Електротехнічний
(факультет)

Кафедра Електропривода
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА

дипломної роботи

магістра

(назва освітньо-кваліфікаційного рівня)

галузь знань 14 Електрична інженерія
(шифр і назва галузі знань)

спеціальність 141 Електроенергетика, електротехніка та електромеханіка
(код і назва спеціальності)

(освітня програма «Електромеханічні системи автоматизації та електропривод»)

освітній рівень магістр
(назва освітнього рівня)

Кваліфікація 2151.2 (Int) Інженер-електромеханік
(код і назва кваліфікації)

на тему: Стенд остаточного складання автомобільних дверних панелей

Виконавець:

Студент 6 курсу, групи 141м-16-4

____________ **Герман Е.С.**

(підпис) **(прізвище та ініціали)**

Керівники	Прізвище, ініціали	Оцінка	Підпис
проекту	Азюковський О.О.		
розділів:			
Аналітична частина	Азюковський О.О.		
Основна частина	Азюковський О.О.		

Рецензент			

Нормоконтроль	Казачковський М.М.		

Дніпропетровськ
2018

Міністерство освіти і науки України
Державний вищий навчальний заклад
"Національний гірничий університет"

ЗАТВЕРДЖЕНО:

завідувач кафедри
Електроприводу
(повна назва)

______________ Казачковський М.М.

(підпис) (прізвище, ініціали)

«_____»_______________2018 року

ЗАВДАННЯ
на виконання кваліфікаційної роботи магістра
спеціальності 141 Електроенергетика, електротехніка та електромеханіка
(код і назва спеціальності)
(освітня програма «Електромеханічні системи автоматизації та електропривод»)

студенту 141м-16-4 Герман Е.С.
(група) (прізвище та ініціали)

Тема дипломної роботи Стенд остаточного складання автомобільних дверних панелей

1 ПІДСТАВИ ДЛЯ ПРОВЕДЕННЯ РОБОТИ

Наказ ректора ДВНЗ "НГУ" від _________________№_____

2 МЕТА ТА ВИХІДНІ ДАНІ ДЛЯ ПРОВЕДЕННЯ РОБІТ

Об’єкт досліджень Стенд остаточного складання автомобільних дверних панелей

Предмет досліджень програмне і апаратне забезпечення стенду остаточного складання автомобільних дверних панелей, контроль якості і передпродажна підготовка автомобільних дверний панелей.

Мета роботи: Розробити оптимальну систему програмного і апаратного забезпечення стенду остаточного складання автомобільних дверних панелей, якабуде відповідати представленим вимогам швидкості, якості та безпеки виробничого процесу.

Вихідні дані для проведення роботи Вимоги замовника до програмного і апаратного забезпечення, технічні дані використаного обладнання, стандарти розробки технічного і апаратного забезпечення виробника.

3 ОЧІКУВАНІ НАУКОВІ РЕЗУЛЬТАТИ

Наукова новизна полягає у розробці нового стенду остаточного складання автомобільних дверних панелей з використанням колабораційного робота, системи комп’ютерного зору, підготовкою стенду до стандартів “Industry 4.0”.

Практична цінність полягає у підвищенні швидкості передпродажної підготовки та контролю якості виготовленою продукції.

4 ВИМОГИ ДО РЕЗУЛЬТАТІВ ВИКОНАННЯ РОБОТИ

Результати повинні бути достовірними та обґрунтованими

5 ЕТАПИ ВИКОНАННЯ РОБІТ

Найменування етапів робіт	**Строки виконання робіт (початок-кінець**
Огляд технології автономного і комбінованого електрозабезпечення	01.09.2017 – 17.09.2017
Розробка технічних рішень щодо програмного і апаратного забезпечення стенду остаточного складання автомобільних дверних панелей та дослідження їх роботи	17.09.2017 – 18.10.2017

6 РЕАЛІЗАЦІЯ РЕЗУЛЬТАТІВ ТА ЕФЕКТИВНІСТЬ

Виробничий ефект для споживача очікується позитивним завдяки оптимізації виробництва, підвищенні ефективності роботи, покращенню контролю якості та строренню бази даних виготовленої продукції. Підвищення умов праці, покращенню безпеки роботи на етапі передпродажної підготовки.

7 ДОДАТКОВІ ВИМОГИ

Відповідність оформлення ДСТУ 3008-95. Документація. Звіти у сфері науки і техніки. Структура і правила оформлення

Завдання видав ______________ Азюковський О.О.
(підпис) (прізвище, ініціали)

Завдання прийняв до виконання ______________ Герман Е.С.
(підпис) (прізвище, ініціали)

Дата видачі завдання: __________________

Термін подання дипломної роботи до ЕК ______________

Abstract

The object of investigation: – End-of-line (EOL) workbench of car door panels.

The object of investigation: software and hardware for the EOL workbench of car door panels, quality control and pre-sale preparation of automotive door panels.

The purpose of the work: Develop an optimal system of software and hardware for the EOL workbench of car door panels, which will meet the requirements of the speed, quality and safety of the production process.

During the work has been developed the system of software and hardware for the stand of final assembly of automotive door panels, which will meet the requirements of the speed and quality of the production process. Were succeeded all technical and safety requirements of customer. Developed effective communication and monitoring system for workbench.

Workbench developed according to rules and requirements of collaborative work between human workers and robots. Satisfied safety requirements of movable parts and integrated system of errors avoiding during workflow.

Workbench has been prepared for IoT usage and can be easily integrated into Industry 4.0 environment.

Реферат

Об'єкт дослідження: - Стенд остаточного складання автомобільних дверних панелей.

Предмет досліджень: програмне і апаратне забезпечення стенду остаточного складання автомобільних дверних панелей, контроль якості і передпродажна підготовка автомобільних дверних панелей.

Мета роботи: Розробити систему програмного і апаратного забезпечення стенду остаточного складання автомобільних дверних панелей, яка буде відповідати представленим вимогам швидкості, якості та безпеки виробничого процесу.

В ході роботи була розроблена система програмного та апаратного забезпечення для стенду остаточного складання автомобільних дверних панелей, що відповідає вимогам швидкості та якості виробничого процесу. Були виконані всі технічні та технічні вимоги замовника. Розроблена ефективна система комунікації та моніторингу стенду.

Стенд розроблений відповідно до правил і вимог спільної роботи між працівниками та роботами. Задоволені вимоги безпеки рухомих частин та інтегрована система уникання помилок при роботі.

Стенд був підготовлений для використання як IoT пристрій і може бути легко інтегрований в середовище Industry 4.0.

Реферат

Объект исследования: - Стенд окончательной сборки автомобильных дверных панелей.

Предмет исследований: программное и аппаратное обеспечение стенда окончательной сборки автомобильных дверных панелей, контроль качества и предпродажная подготовка автомобильных дверных панелей.

Цель работы: Разработать систему программного и аппаратного обеспечения стенда окончательной сборки автомобильных дверных панелей, которая будет соответствовать представленным требованиям скорости, качества и безопасности производственного процесса.

В ходе работы была разработана система программного и аппаратного обеспечения для стенда окончательной сборки автомобильных дверных панелей, соответствующая требованиям скорости и качества производственного процесса. Были выполнены все технические и технические требования заказчика. Разработана эффективная система коммуникации и мониторинга стенда.

Стенд разработан в соответствии с правилами и требованиями совместной работы между работниками и работами. Удовлетворены требования безопасности подвижных частей и интегрированная система ухода ошибок при работе.

Стенд был подготовлен для использования в качестве IoT устройства и может быть легко интегрирован в среду Industry 4.0.

Table of Contents

1 Introduction **5**

2 Technical specifications **6**

3 Product specifications **8**

4 WORK CYCLE **10**

4.1 Timing table *12*

5 Hardware specifications **16**

5.1 Hardware description and network configurations. *19*

5.2 Robot *22*

5.2.1 Forward kinematic 26

5.2.2 Inverse Kinematics 28

5.3 Robot Settings *29*

5.3.1 Inputs, outputs of the robot controller 29

5.3.2 Configuring Security with PolyScope 35

5.3.3 General limits 36

5.4 Network structure. *37*

6 Software **38**

6.1. Robot program *38*

6.2 Spervisor communication *41*

7 Robot Safety Standards for Collaborative Operation ISO 10218 **44**

8 Conclusions **47**

9 Bibliography **48**

Addition 1 **49**

Addition 2 **79**

1 Introduction

Modern humans are distinguished from other species by their extensive use of tools to control and adapt to their surroundings. Early stone tools such as anvils had no holes and were not designed as interchangeable parts. Mass production established processes for the creation of parts and system with identical dimensions and design, but these processes are not uniform and hence some customers were unsatisfied with the result. Quality control separates the act of testing products to uncover defects from the decision to allow or deny product release, which may be determined by fiscal constraints.[6] For contract work, particularly work awarded by government agencies, quality control issues are among the top reasons for not renewing a contract.

The simplest form of quality control was a sketch of the desired item. If the sketch did not match the item, it was rejected, in a simple Go/no go procedure. However, manufacturers soon found it was difficult and costly to make parts be exactly like their depiction; hence around 1840 tolerance limits were introduced, wherein a design would function if its parts were measured to be within the limits. Quality was thus precisely defined using devices such as plug gauges and ring gauges. However, this did not address the problem of defective items; recycling or disposing of the waste adds to the cost of production, as does trying to reduce the defect rate. Various methods have been proposed to prioritize quality control issues and determine whether to leave them unaddressed or use quality assurance techniques to improve and stabilize production.

The End-of-line test station is located at the end of the door panel assembly and it permits checking of quality-relevant parameters. The scope of testing includes parts presence, visual check-in of all variable parameters and overall picture of assembled door panel, including the final classification or good/bad decision.

2 Technical specifications

- 457 car sets/day;
- Takt time: max. 75 sec includes handling
- 3 shift/day.
- N°2 Workstations with 4 nests on 4 separated pallets (Quick change over concept)
- Front Right pallet with nest
- Front Left pallet with nest
- Rear Right pallet with nest
- Rear Left pallet with nest

The workstations must be equipped with the following materials as below described:

- Bosch profiles in aluminum (or equivalent).
- The implicit part of the workstation must be designed with the station properly dimensioned and adjustable in height and rotation 180°(to satisfy safety/ergonomics issues) respecting Cycle time, and meet quality final assembly
- The work surface are nests built on pallets with 180°rotation (one for every single door panel). In addition, the pallet must be equipped with, all necessary to respect quick changeover concept (max 10 min. from part to part). The nests must also consider the following characteristics, suitable for the containment of parts in order to conduct the

activities in the best way to preserve the integrity/quality of the final preassembled door panel.

- It also must be equipped with sensors presence for the door panel components.

- Check the correct version/ type directly interfaced with PLC.

- Make picture from “A” surface (includes side areas as well). Avoid any shadow area on the picture. Picture name should be identically to the running serial numbers of the Door Panel assy. “A” side picture resolution should be min. 13 MPixel. The pictures should be transferred to the plant’ server immediately. In case of plant’s server shut down, pictures quantity for 1 shift should be stored on the station.

- Electrical tightening system, equipped with unit control that checks torque and angle control directly interfaced with PLC.

- PLC Siemens series S7

- The machines parameters need to be protect by password and EKS system needs to install.

- Touch screen panel 19” built on a articulated and movable frame

- INDUSTRIAL COMPUTER needed to manage information regarding the process (ex.ODS Quality information). Industry 4.0 specifications needs to apply.

- Interface WIFI/Ethernet with the plant system able to display all the info above mentioned.

- Barcode reader (model 128, Datamatrix reader 2D)

- Printer for barcode labels (model Intermec PM 43; type of barcode to print: Code 39)

- Adequate lighting system (energy saving lamps).

- All the material needed for the workstations provided and quoted by the supplier.

- Ambient light / non-ambient light version of Beltline (all 4 doors affected) à no electrical test, only vision if light is installed

3 Product specifications

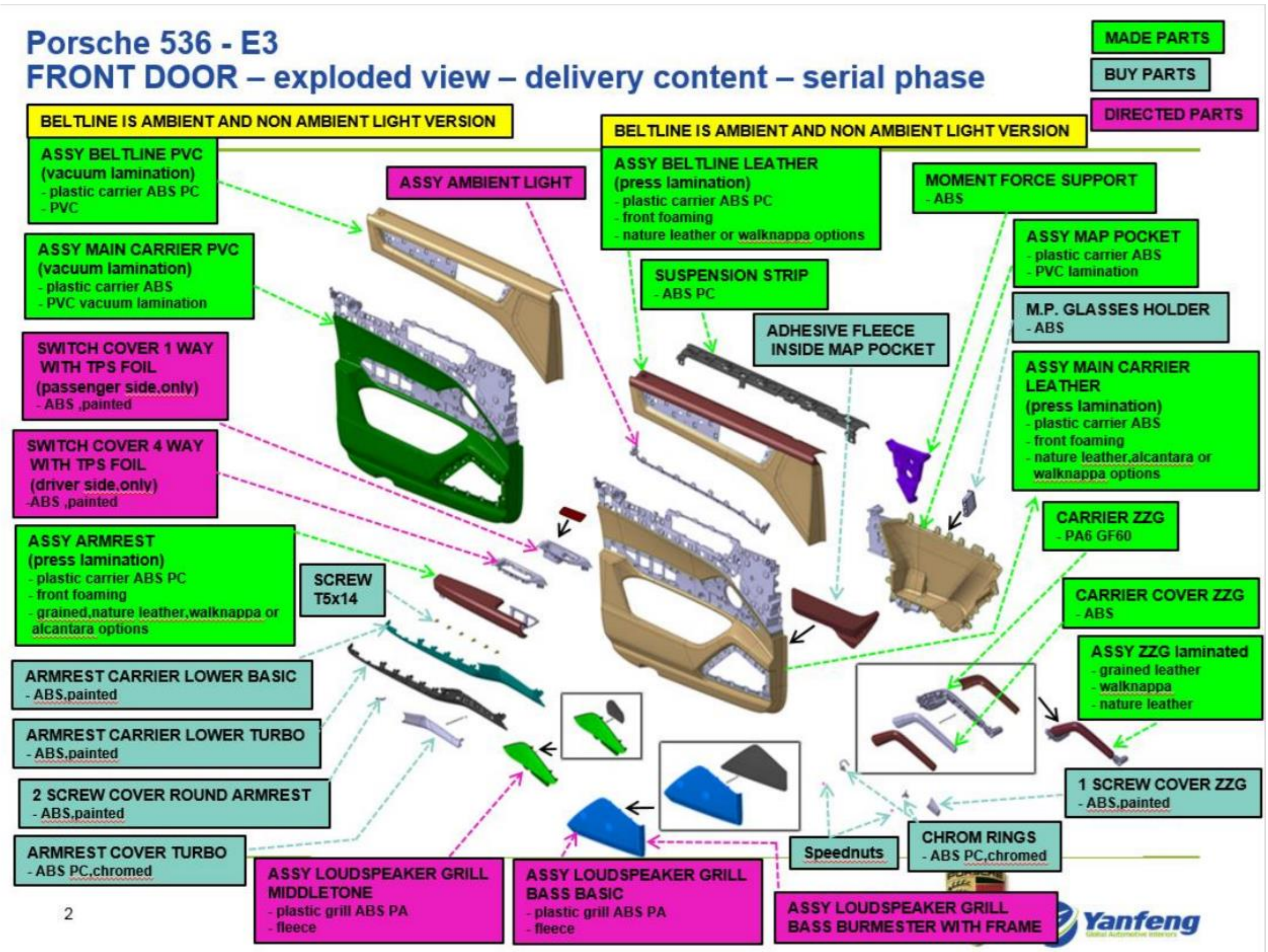


Figure 3.1. Door panel configuration.

Versions can vary based on: o Door side (Front Left/ Rear Left; Front Right/ Rear Right)

- Cover types: PVC, Embossed leather, Walknappa leather, Club leather, Alcantara. check all type of material
- Color of cover: leather/PVC (see below table) Cover types: PVC, Embossed leather, Walknappa leather, Club leather, Alcantara. check all colors o Ambient light / non-ambient light version of Beltline (all 4 doors affected)
- Rollo / non-Rollo version of Rear Beltline (only rear doors affected) o Beltline leather version can have 2 different colors of cut-parts a
- Burmester / Basic Bass loudspeaker grill (5 colors/ each) Needs to check colors.
- Burmester / Basic Bass loudspeaker grill shape differentiate needs to clarify, yet.
- Check presence and color of Decrational sew line on Armrest and Beltline leather parts.
- Safety Led cut out verification on Front DP.

4 WORK CYCLE

- Read the card by barcode of the version to produce.
- Check by camera the presence suspension strip welded in the Beltline at “B” side.
- Tuning the ASSY Door Panel by 180°.
- Scan the sub-assembly labels on the door panel (Main Carrier, Beltline, Map pocket). Double check the color and cover type appropriate by camera check.
- Check Ambient light version functional/ optical test
- Reading presence of loud speaker Low bass and loud speaker middle tone (9 colors) or Burmester version.
- If all controls done of the above operations and if ok, the system will allow the screwing operation
- Assembly Main carrier and Beltline with 1 screws (Only rear doors both sides). Check torque control + angle control.
- Picture of the aesthetical side A of the door panel to certify the status and store in a data base.

- Additional 2 pictures from the edges opposite sides (most probably damaging areas), maybe takes 5-6 sec more takt time.

- Beltline safety LED cut out verification for Front Doors.

- Automatic printing of the final label (EOL) if and only if all previous operations are ok. In case of NOK parts, the station must catch the DP ASSY and release can be allowed by higher (min 4.) level EKS key log in.

- Unload the assembled door panel and place it in the final container. In case that the screwing operation is not tight correctly, the system must allow to screw and unscrew N times the same screw, (N to be define) and then give KO final, or continue with the next screws.

- Torque, angle and their tolerances will be determined later.

4.1 Timing table

Panel position- Side B: front operator, Side A: ground							
Operation	**Panel Side**	**Robot: Vision, Photo**	**Handle Barcode reader**	**Sensors**	**Screw contoller**	**Printer**	**Cycle Time exlcuding handling sec.**
Read the card by barcode	Side B		x				
Reading presence of door panel with 2 sensors	Side B			x			
Scan the sub-assembly labels (to be confirmed)	Side B		x				
Reading presence of suspension strip welded in the Beltline (No check for Rollo version)	Side B			x			

Safety led on beltline Front L/R: the sensor check the presence of Hole	Side B			x			10
Puddle lamp: the sensor checks the presence of Hole	Side B			x			
If ok: Start Robot operations (protect by safety beans)							
Reading presence of Ambient light (OPT)	Side A	x					
Reading presence and colour of loud speaker Low bass	Side A	x					
Reading presence and colour of loud speaker middle tone or Burmester (see logo??) version – Only Front	Side A	x					

Reading if Beltline is Rollo or No Rollo (No Rollo Rear)	Side A	x					
Reading presence and colour/material of Beltline upper part	Side A	x					
Reading presence and colour/material of Beltline lower part (if exist)	Side A	x					
Reading presence and colour of sew line Beltline OPT	Side A	x					
Reading presence and colour of Map Pocket (only PVC)	Side A	x					
Reading presence and colour/material of Main carrier	Side A	x					
If ok: Start Picture							

Picture of the aesthetical. Make more photo also if possible on the side	Side A	x					8
							35
If ok: Start screwing							
Assembly Main carrier and Beltline with 1 screws (torque, angle) – Front L/R	Side B				x		5
Print EOL barcode	Side B					x	5
Total Time							**63**
							65 sec

Table 4.1 Timing table.

5 Hardware specifications

According to the customer requirements has been developed software-hardware complex, which is shown on Fig. 5.1.

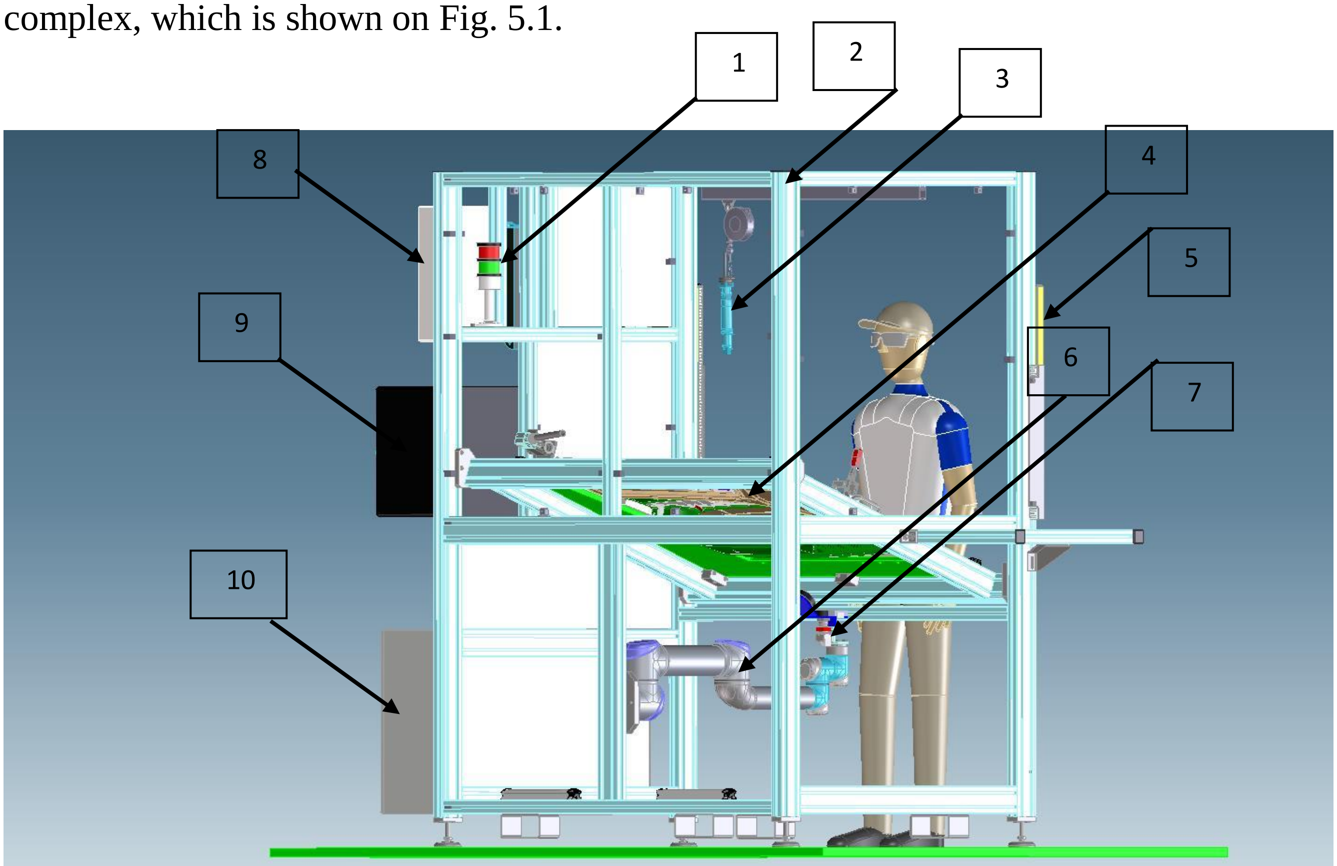


Fig. 5.1. Mechanical render of the developed workbench (Back side).

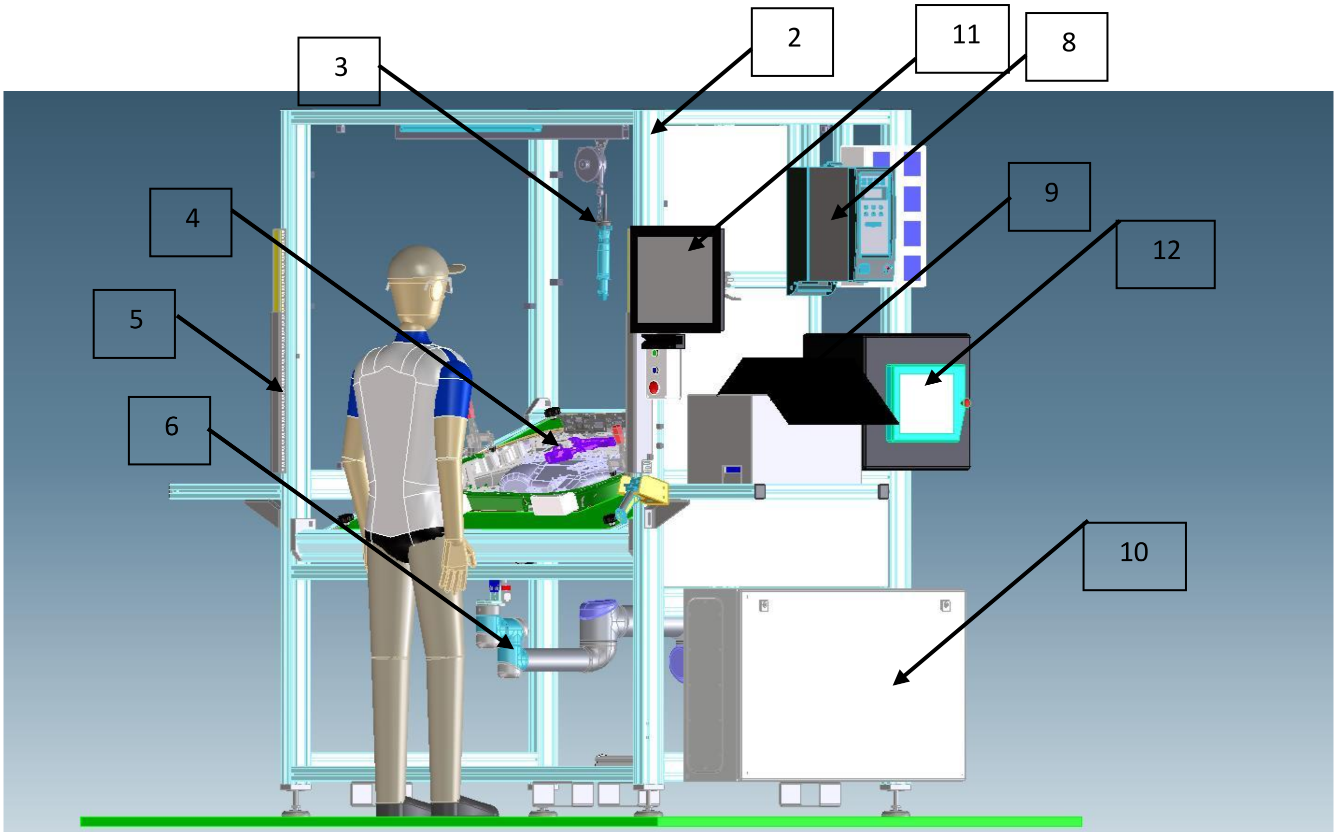


Fig. 5.2. Mechanical render of the developed workbench (Front side).

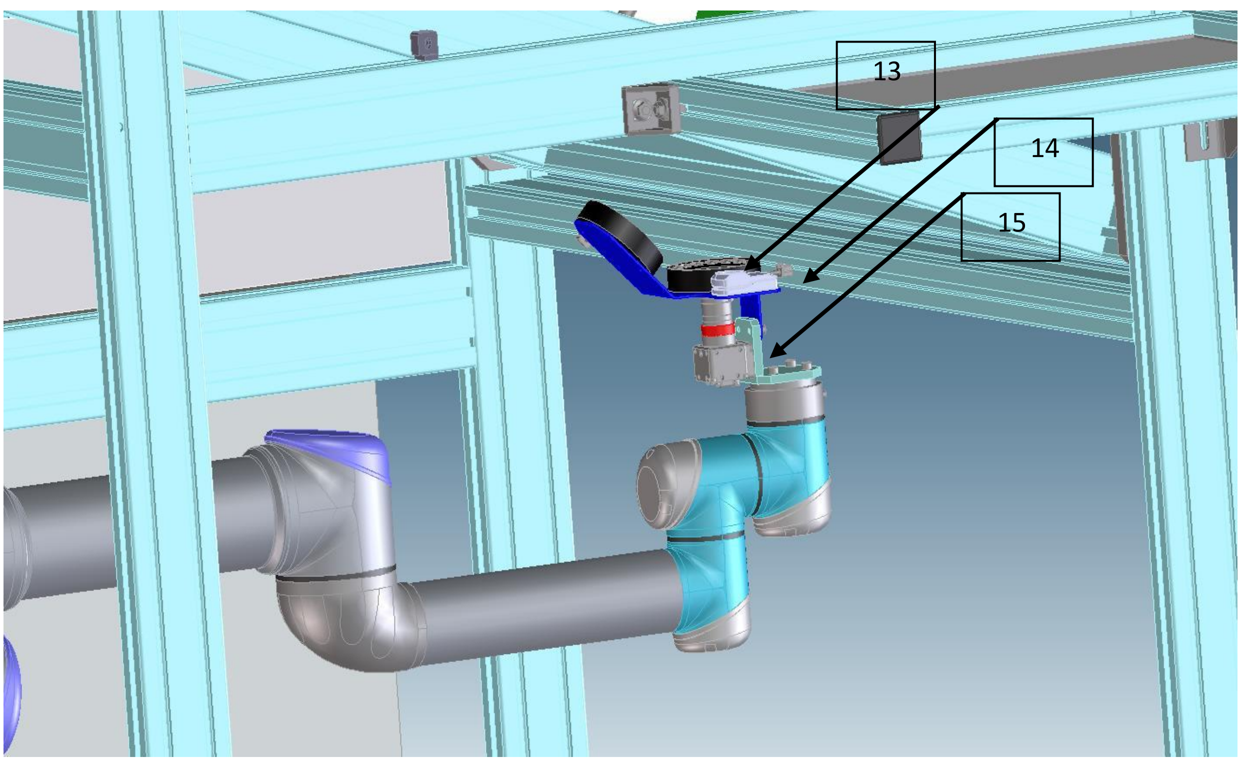


Fig. 5.3. Mechanical render of the developed workbench (End-effector).

Description for figures 5.1.,5.2.,5.3.:

1 – Safety indicator;

2 – Aluminum frame;

3 – Atlas CopCo screw driver;

4 – Panel holder with installed panel;

5 – Safety curtain;

6 – Collaborative robot UR5;

7 – Omron camera;

8 – Atlas CopCo screw driver controller;

9 – Bar code printer;

10 – Electrical box;

11 – Panel PC;

12 – Robot controller;

13 – Ambient ligth;

14 – Snapshot camera;

15 – Omron camera;

5.1 Hardware description and network configurations.

Name	Characteristic
Model	Axiomtec
Operational system	Windows 7
Hostname	Server
IP address	192.168.5.21
Port	21

Table 5.1 PC data.

Name	Characteristic
Model	Siemens Simantic S7-1500
IP address	192.168.5.4
Dashboard server port	24

Table 5.2 PLC data.

Name	Characteristic
Model	Omron FH-1050
Number of connected cameras	2
Hostname	Vision Controller
IP address	192.168.5.3
Dashboard server port	23
Name	Characteristic
Model	Omron FH-SC02
Image elements	CMOS image elements (2/3-inch equivalent)
Color/Monochrome	Color
Effective pixels	2040 (H) × 1088 (V)
Imaging area	11.26 × 5.98 (12.76 mm)
Pixel size	5.5 (μm) × 5.5 (μm)
Shutter function	Electronic shutter; 25 μs to 100 ms.
Partial function	2 to 1088 lines
Frame rate	219 fps (4.6 ms) *
Lens mounting	C mount
Weight	Approx.110 g

Table 5.4 Omron Camera data.

Name	Characteristic
Model	iDS UI-3013XC
Image elements	CMOS image elements (1/2.45-inc)
Color/Monochrome	Color

Effective pixels	4192 (H) × 3104 (V)
Resolution	13.01 MPix
Imaging area	11.26 × 5.98 (12.76 mm)
Pixel size	1.4 (μm) × 1.4 (μm)
Shutter function	Electronic shutter; 25 μs to 100 ms.
Frame rate	1 fps

Table 5.4. Snapshot camera data.

Name	Characteristic
Model	Intermec PM-43
IP address	
Dashboard server port	

Table 5.5. Barcode printer data.

Name	Characteristic
Model	Atlas Copco Power focus 4002-C-HW
IP address	192.168.5.6
Dashboard server port	26

Table 5.6. Screwdriver data.

5.2 Robot

A manipulator is a controlled device or a robot for performing motor functions, similar to the functions of a human hand when moving objects in space, equipped with a working body. The manipulator consists of links connected by movable kinematic pairs. The latter move in a certain program with the help of controlled drives

Manipulation system of manipulators and their motor capabilities are determined by the type and arrangement of kinematic pairs. The choice of a specific version of the kinematic scheme of the manipulation system is determined by the specific condition and requirement. This scheme should provide a sufficient degree of universality of the robot's operation taking into account the operations that it must perform in the external environment. The robot can be programmed to move the tool and communicate with other machines via electrical signals. The robot is a co-driver, consisting of stamped aluminum tubes and joints. With the help of the PolyScope programming interface, is possible to program the robot to move the tool along a specified path.

Industrial robots are usually large, heavy and expensive machines, however, UNIVERSAL ROBOTS has developed a flexible and easy-to-use manipulator. As a result, the company introduced the first own robot UR5, which will be used in this course project.

In the development of this system, special requirements have been determined by the safety of use in close proximity to humans. Therefore, a set of protective and safety mechanisms will be developed that will meet these requirements.

Positioning precision	-+0.1 mm
Working temperature	0-50 C
Power consumption	90-325 Wt
Collaborativity	15 safety functions Functionality agreeded by TUV NORD Tested according to EN ISO 13849:2008 Pl d
Specifications	
Carry load	5 kg
Maximum distance	850 mm
Degrees of freedom	6 swing points
Programming	Graphical interphase PolyScope, sensor tablet 12 inches with holders
Physical parameters	
Base diameter	149 mm
Working tool connection	M8
Manipulator connecting cable	6 m
Weight, including connecting cable	18.4 kg

Table 5.2.1 Robot physical parameters.

Dependence category	IP20
Noise level	Less than 65 dB(A)
Input-output ports	Number of digital inputs: 16 Number of digital outputs: 16 Number of analog inputs: 2 Number of analog outputs: 2
Connection protocols	Profinet, Ethernet IP
Feeding	100-240 V, 50-60 Hz
Dimensions (WxHxD)	475 x 423 x 268, mm
Weight	15 kg
Material	Steel

Table 5.2.2. Robot controller parameters.

Defendence class	IP20
Physical parameters	
Materials	Aluminum, polypropylene
Weight	1.5 kg
Connection cable length	4.5 m

Table 5.1.3. Control tablet parameters.

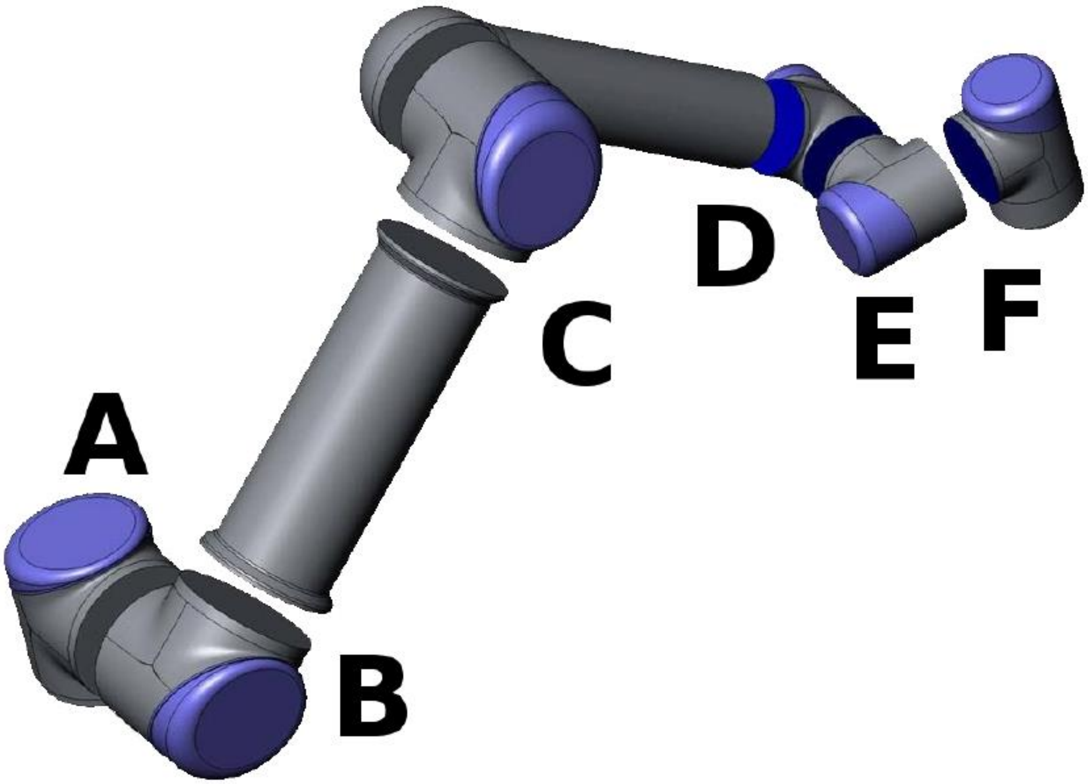


Fig. 5.2.1 Schematic representation of the Universal Robots UR5

Articulation of the robot:

A is the base;

B - Shoulder joint;

C - Elbow joint;

D, E, F - wrists 1, 2, 3, respectively;

5.2.1 Forward kinematic

A 4-axis SCARA (Selective Compliance Assembly Robot Arm) robot has parallel shoulder, elbow, and wrist rotary joints, and a linear vertical axis through the center of rotation of the wrist. This type of manipulator is very common in light-duty applications such as electronic assembly. Most users want to program this device as an XYZ Cartesian system with a rotary angle (C) for the tool, leaving the controller to compute the required joint angles and lengths. Mechanism Description In this example, the upper-arm length (L1) is 400 mm, and the lower-arm length (L2) is 300 mm. The shoulder joint (S), the elbow joint (E), and the wrist joint (W) have resolutions of 1000 counts per degree. Rotation in the positive direction for all 3 joints is counter-clockwise when viewed from the top. The vertical axis (V) has a resolution of 100 counts per millimeter, and movement in the positive direction goes up. When the shoulder, elbow, and wrist joints are at their zero-degree positions, the two links are both extended along the X-axis and the tool orientation C is at zero degrees. When the vertical axis is at its home position, it is 250 mm above the Z-axis zero point. Due to wiring constraints, rollover of the rotary axes is not permitted. This diagram shows a top view of the mechanism. Z-axis motion is into and out of the page.

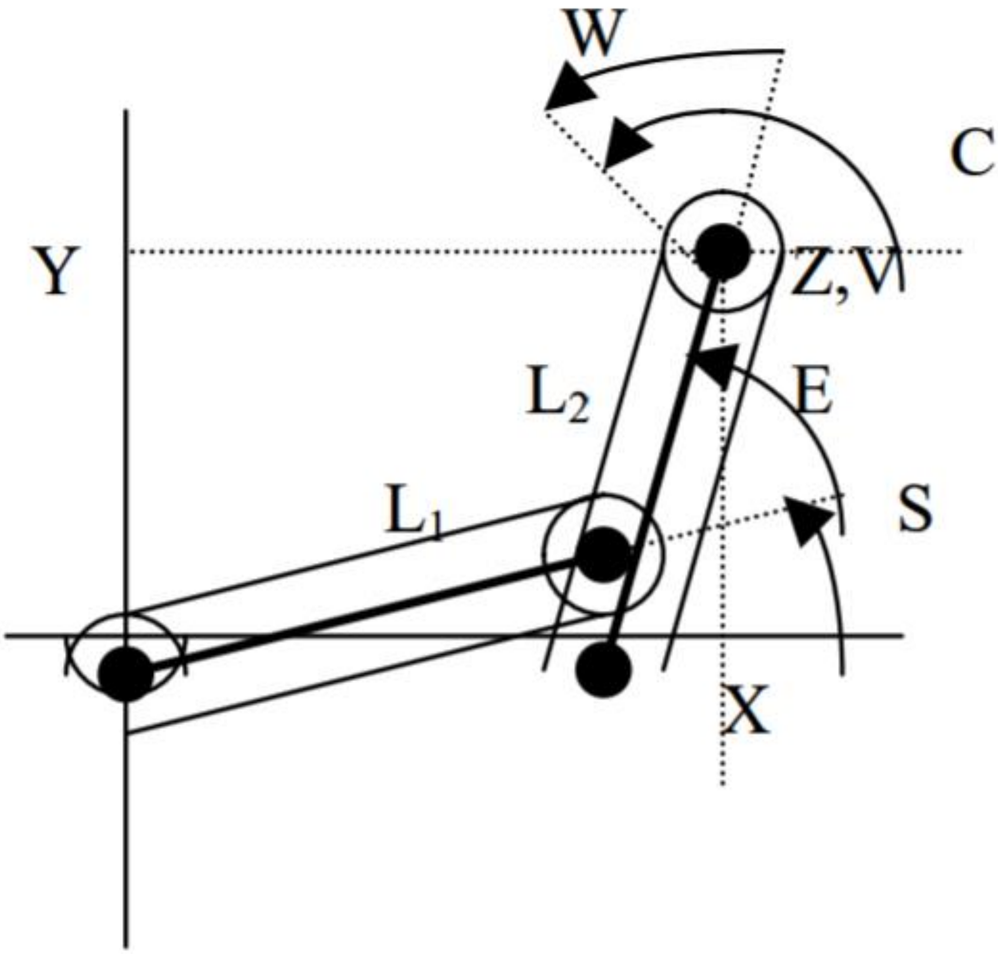


Fig. 5.2.1.1 Scheme of the top view of the mechanism

The forward-kinematic equations are:

$$X = L_1 \cos(S) + L_2 \cos(S + E) \quad (5.2.1.1)$$

$$Y = L_1 \sin(S) + L_2 \sin(S + E) \quad (5.2.1.2)$$

$$C = S + E + W \quad (5.2.1.3)$$

$$Z = V + Z_0 \quad (5.2.1.4)$$

5.2.2 Inverse Kinematics

Limiting ourselves to positive values of the elbow (E) angle, producing the right-armed case (done by selecting the positive arc-cosine solutions), we can write our inverse kinematic equations as follows: To implement these equations inverse-kinematic program for Coordinate System 1 that converts the X, Y, and Z tip coordinates in millimeters and the tip angle C in degrees to the shoulder

$$E = +\cos^{-1}\left(\frac{X^2 + Y^2 - L_1^2 - L_2^2}{2L_1L_2}\right) \quad (5.2.2.5)$$

$$S + Q = \arctan 2(Y, X) \quad (5.2.2.6)$$

$$Q = +\cos^{-1}\left(\frac{X^2 + Y^2 + L_1^2 - L_2^2}{2L_1\sqrt{X^2 + Y^2}}\right) \quad (5.2.2.7)$$

$$S = (S + Q) - Q \quad (5.2.2.8)$$

$$W = C - S - E \quad (5.2.2.9)$$

$$V = Z - Z_0 \quad (5.2.2.7)$$

5.3 Robot Settings

5.3.1 Inputs, outputs of the robot controller

This I/O is extremely flexible and can be used for wide range of different equipment; including pneumatic relays, PLCs and emergency stop buttons. The illustration below shows the layout of electrical interface inside the control box.

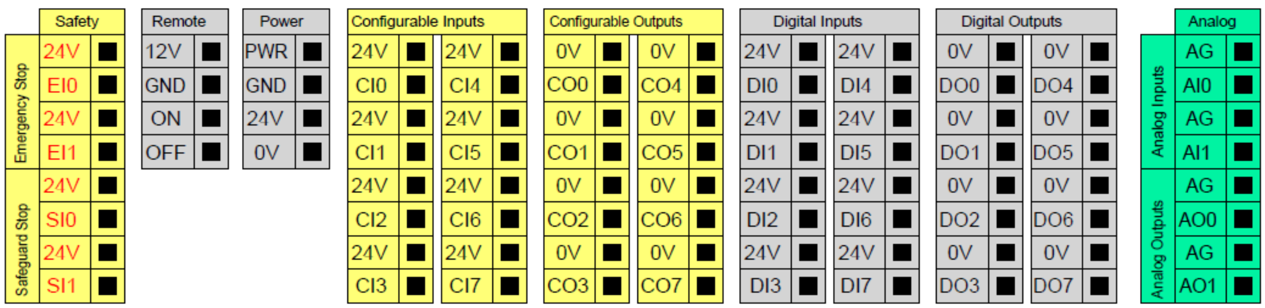


Figure 5.3.1.1. Schematic drawing of controller I/O.

The meaning of the different colors must be observed, see in table 5.2.1.

Yellow with red text	Dedicated safety signals
Yellow with black text	Configurable for safety
Gray with black text	General purpose digital I/O
Green with black text	General purpose analog I/O

Table 5.3.1.1. Common specifications for all digital I/O

This section define electrical specifications for the following 24V digital I/O of the control box.

- Safety I/O.
- Configurable I/O.

• General purpose I/O.

It is very important that UR robots are installed according the electrical specifications, which are the same for all three different kinds of inputs.

It is possible to power the digital I/O from an internal 24V power supply or from an external power source by configuring the terminal block called “Power”. This block consists of four terminals. The upper two (PWR and GND) are 24V and ground from the internal 24V supply. The lower two terminals (24V and 0V) in the block are the 24V input to supply the I/O. The default configuration is to use the internal power supply, see below.

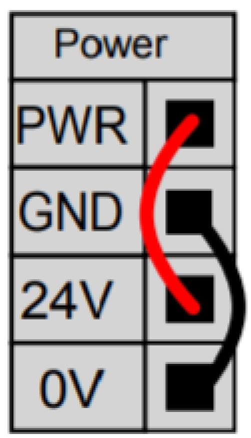


Figure 5.3.1.2. Schematic drawing of the default supply configuration.

The electrical specifications for both the internal and an external power supply are shown below.

Terminals	Parameter	Min	Type	Max	Unit
Internal 24V power supply					
[PWR - GND]	Voltage	23	24	25	V
[PWR - GND]	Current	0	-	2	A
External 24V input requirements					
[24V - 0V]	Voltage	20	24	25	V
[24V - 0V]	Current	0	-	6	A

Table 5.3.1.2. Electrical specifications of internal and external power supply

The digital I/O are constructed in compliance with IEC 61131-2. The electrical specifications are shown below.

Terminals	Parameter	Min	Type	Max	Unit
Digital outputs					
[COx / DOx]	Current	0	-	1	A
[COx / DOx]	Voltage drop	0	-	0.5	V
[COx / DOx]	Leakage current	0	-	0.1	mA
[COx / DOx]	Function	-	PNP	-	Type

[COx / DOx]	IEC 61131-2	-	1A	-	Type
Digital inputs					
[EIx/SIx/CIx/DIx]	Voltage	-3	-	30	V
[EIx/SIx/CIx/DIx]	OFF region	-3	-	5	V
[EIx/SIx/CIx/DIx]	ON region	11	-	30	V
[EIx/SIx/CIx/DIx]	Current (11-30V)	2	-	15	mA
[EIx/SIx/CIx/DIx]	Function	-	PNP	-	Type
[EIx/SIx/CIx/DIx]	IEC 61131-2	-	3	-	Type

Table 5.3.1.3. The electrical specifications of controller digital inputs

In most cases of using the robot, one or more additional emergency stop buttons must be connected. The figure below shows how you can connect one or more emergency stop buttons.

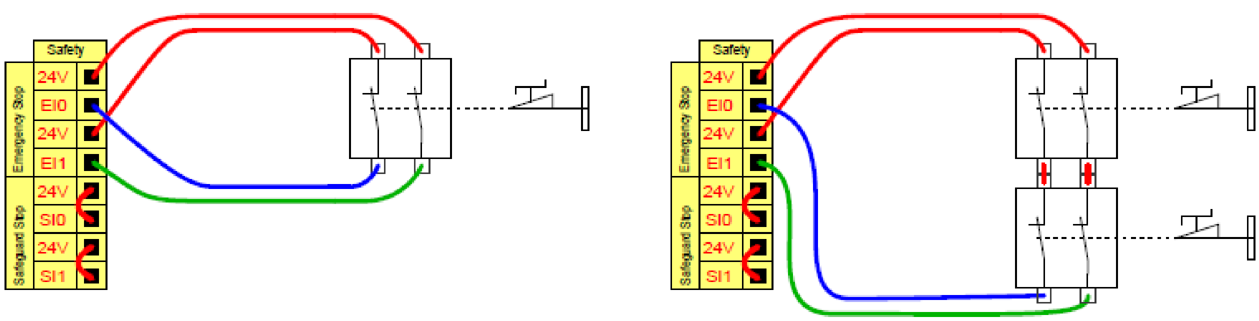


Figure 5.3.1.3. Connection diagram of the stop buttons

An example of a basic safety stop device is a switch that allows the robot to stop when it is triggered, see the figure below.

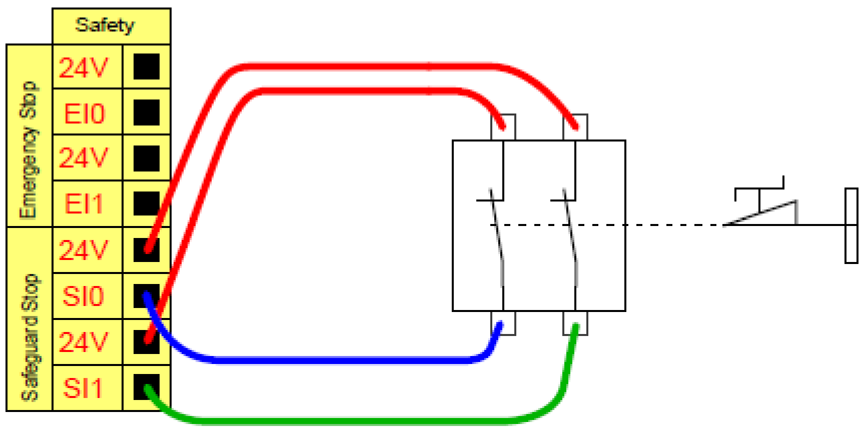


Figure 5.3.1.4. Wiring diagram of the basic switch

This configuration is intended only for use in cases where the operator can not pass through the door and close it behind him. Customizable inputs can be used to configure the reset button located behind the door to reactivate the robot's movement.

If the safety interface is used to exchange data with a light curtain, activation of the reset function outside the perimeter of the safety is required. The reset button must be a two-channel type. In this example, the input with the output configured for performing the reset function is OnC10-S11Off.

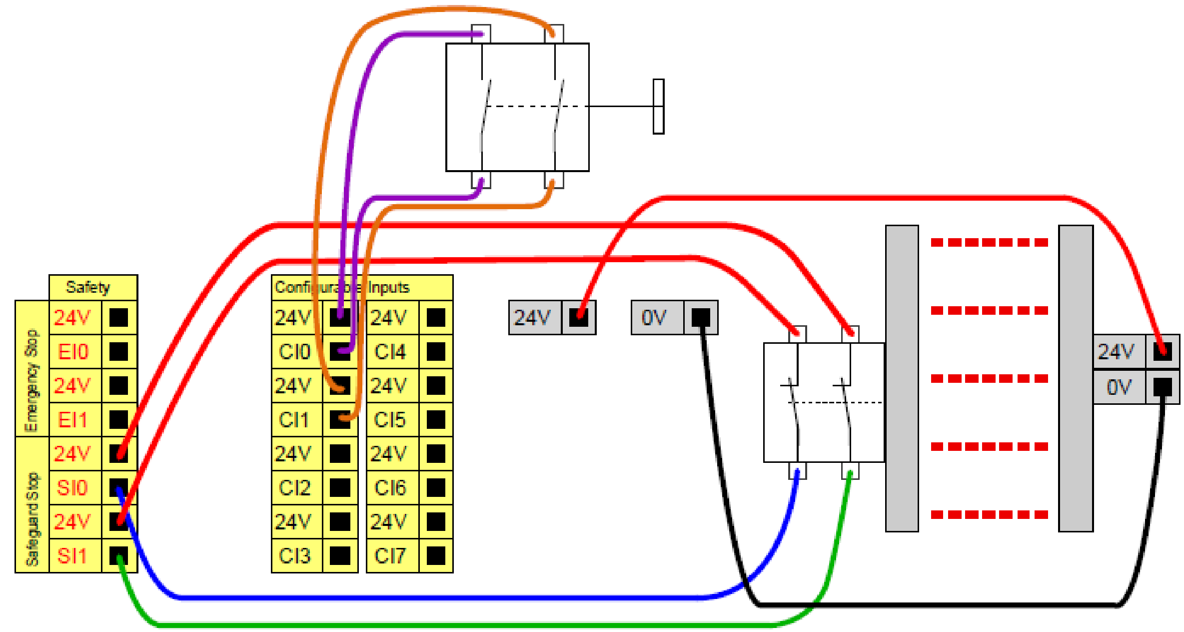


Figure 5.3.1.5. Connection diagram for the stop and reset buttons

Ethernet

ETHERNET-connection is located at the bottom of the control unit, figure.

The Ethernet interface can be used for the following purposes:

• Modules of I / O expansion of MODBUS. For more information, see Part II.

• Remote access and management.

5.3.2 Configuring Security with PolyScope

The robot is equipped with an integrated security system. Depending on the characteristics of the working area of the robot, it is necessary to perform the installation of security settings to ensure the safety of all employees and neighboring equipment. The application of the settings determined by the risk assessment is the very first thing that the system builder should do

The use and configuration of security-related functions and interfaces must be performed in accordance with the assessment that the collector performs for a particular robot application, see the Hardware Installation Guide.

The security configuration settings for installation and training should be applied according to the risk assessment performed by the system assembly and before the first power supply to the robot arm.

All security configuration settings on this screen and its tabs must be installed in accordance with the risk assessment performed by the system integration specialist.

The system collector must ensure that all changes to the configuration of the security configuration are performed in accordance with its risk assessment.

The system builder should prevent unauthorized persons from changing the security configuration. To do this, use, for example, password protection.

5.3.3 General limits

Using general limits configurations limit of linear speed of TCP can be served as well as force and maximum power limitation.

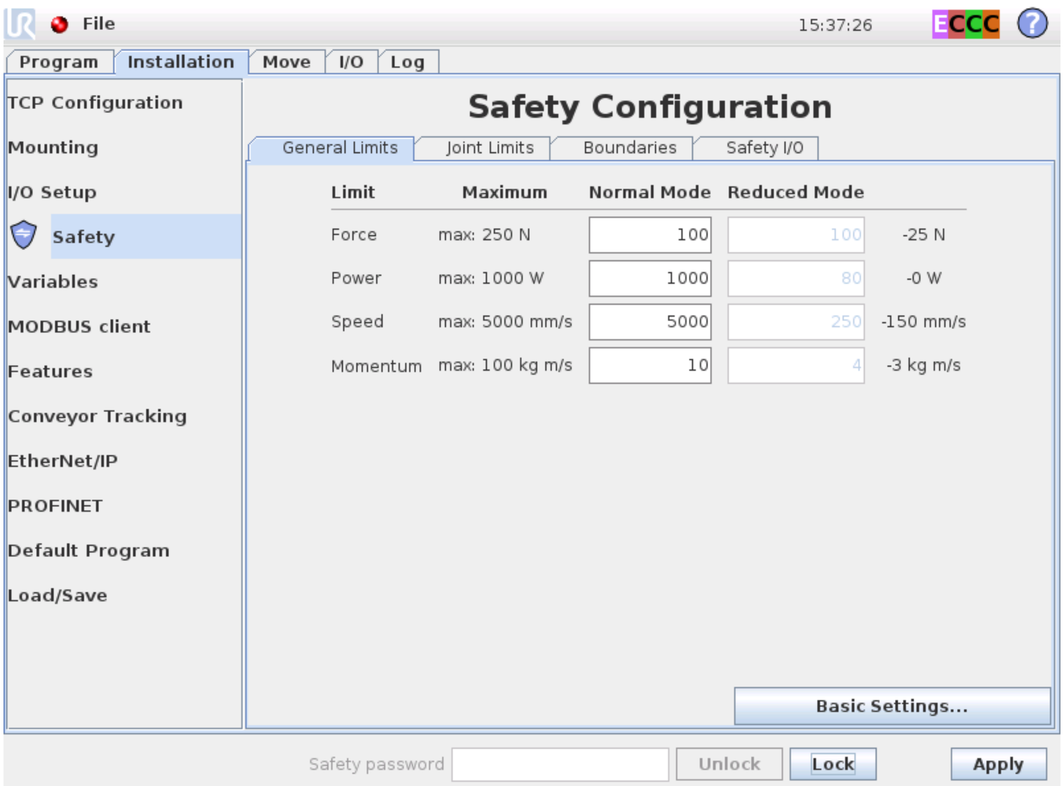


Figure 5.3.3.1. Safety configuration.

5.4 Network structure.

A schematic image is provided below:

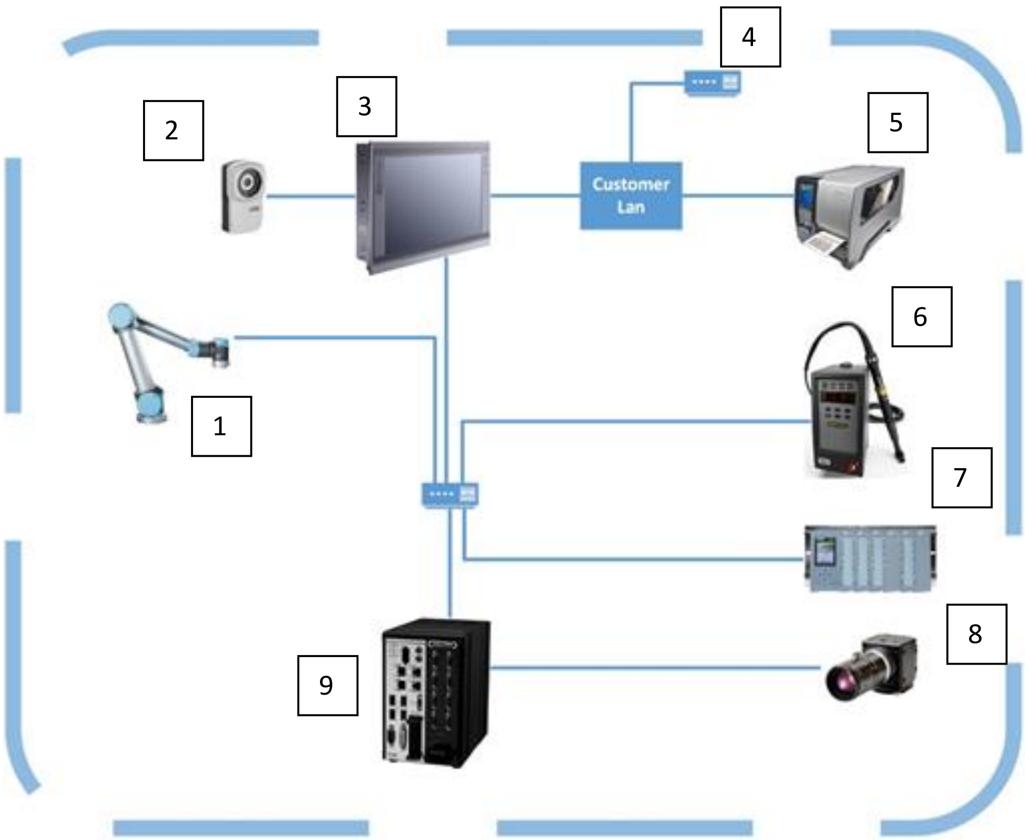


Fig. 5.3.1 Schematic representation of communication between devices.

1 – Universal Robots UR5;

2 – Snapshot camera;

3 – Axiomtec Panel PC;

4 – Cloud;

5 – Intermec PM 43;

6 – Atlas Copco Power focus 4002-C-HW;

7 – Siemens Simantic S7-1500;

8 – Omron iDS UI-3013XC;

9 – Omron FH-SC02;

6 Software

6.1. Robot program

To implement the task, you must connect to the robot controller server to manage the basic functions, such as:

- initialization;

- loading the program into the controller memory;

- program execution;

- response to security messages;

- completion of the program;

- turn off the robot;

In turn, a client program that implements these functions will be launched on the personal computer. The messages are exchanged according to ASII. C # and URscript programming languages are used.

In parallel with this task, a server program was created on the personal computer and a client program on the robot controller. This design allows you to implement the movement of the working body in accordance with the coordinates generated by the personal computer.

An algorithm for interpolating the movement of the robot's working body was implemented. It is presented in Fig.6.1.1.

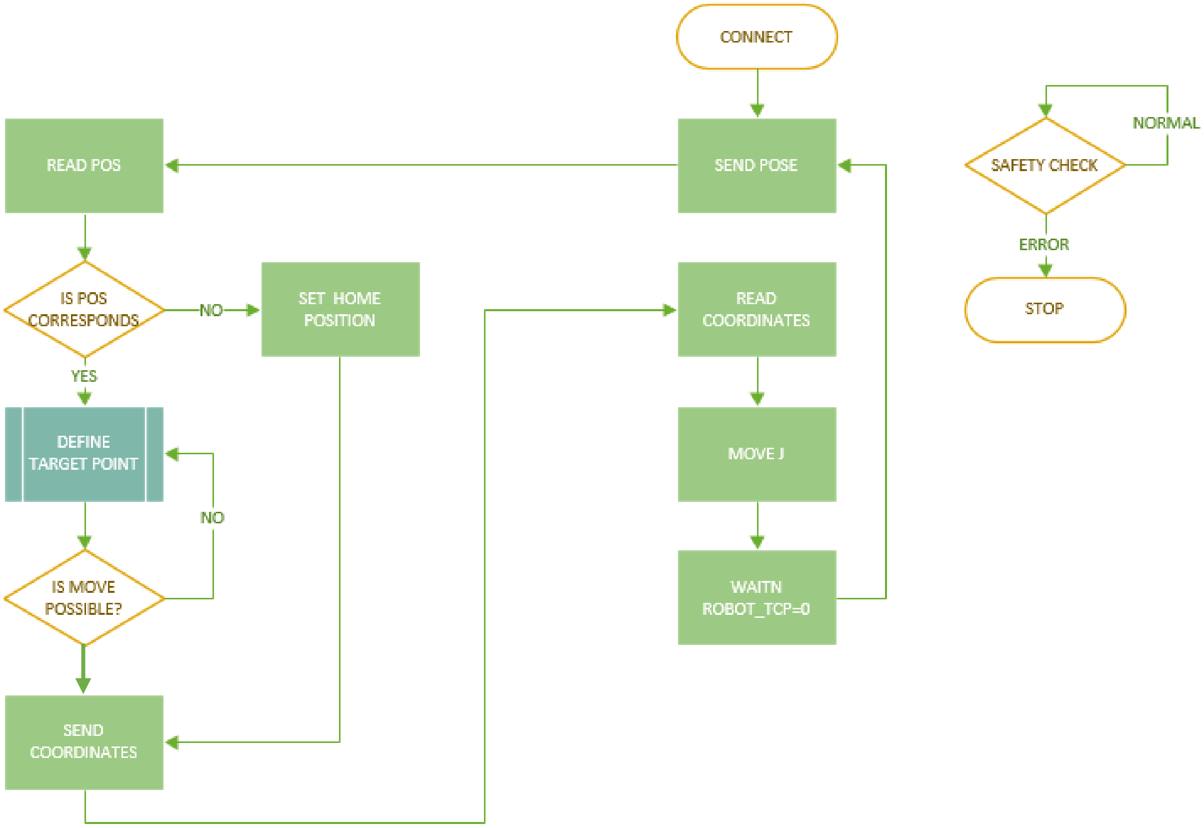


Fig. 6.1.1. The algorithm of the server communication.

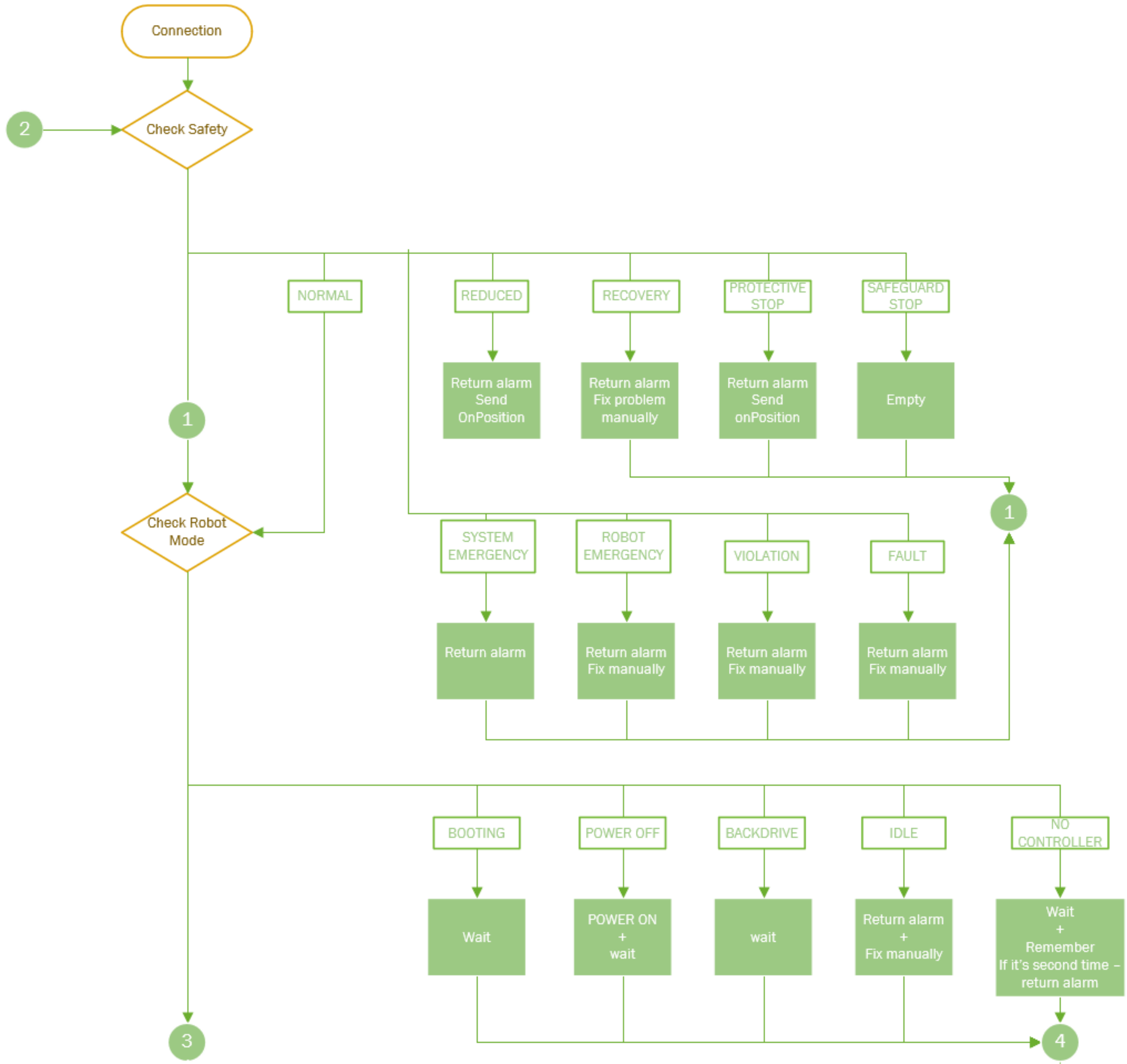
Due to robot specifications and restrictions of installed camera as end-effector has been created matrix of truth for all possible movements to perform without damages for robot joints, camera and lightning, other equipment. Matrix consist of bool values and its always square matrix. According this, certain movements are impossible to perform.

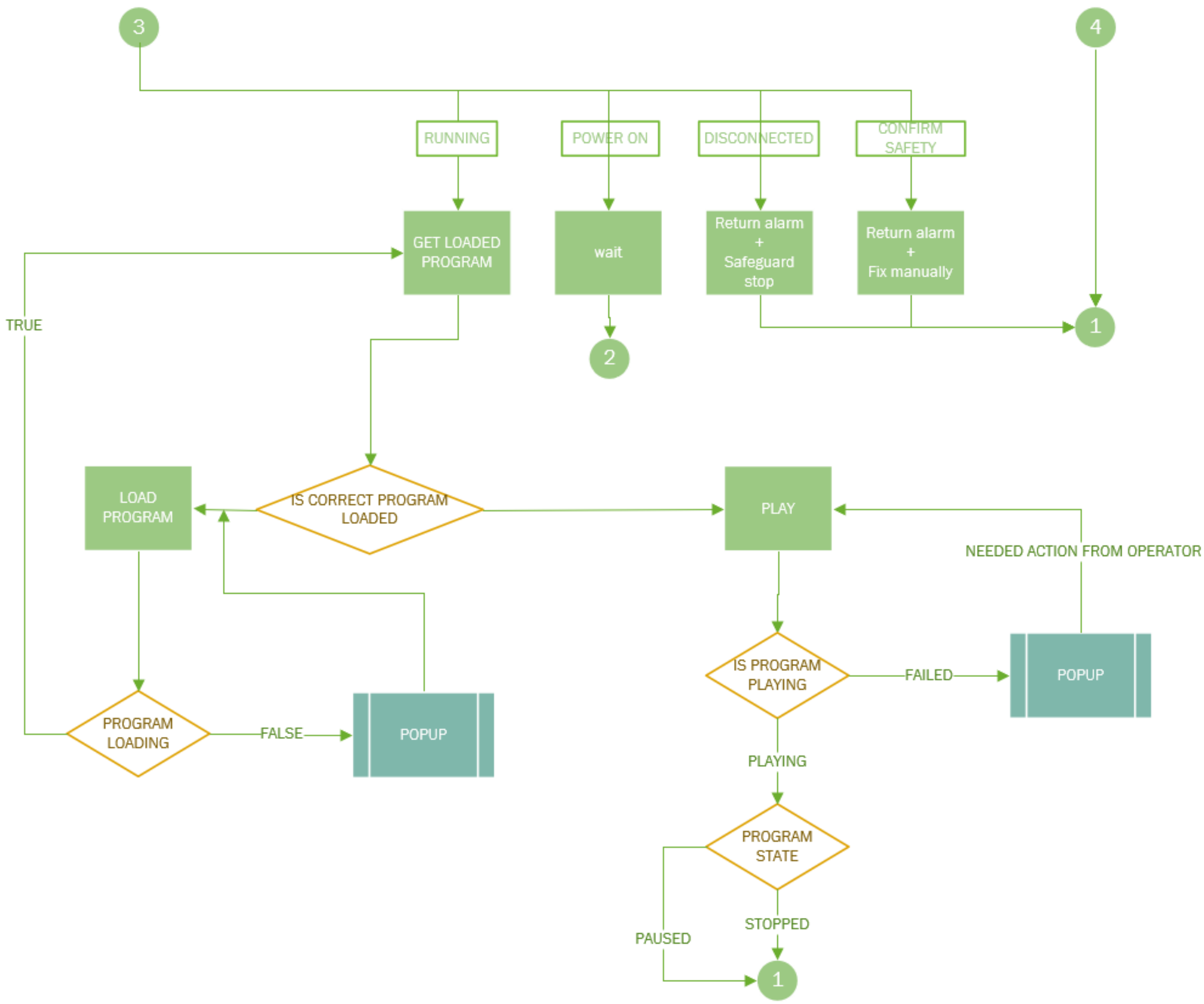
```
        //   0    1     2     3    4    5      6     7     8     9     10     11
  /*0*/  {true, true, true,false,false,false,false, true, true,false,false,false},  //0 low bass + main carrier
  /*1*/  {true, true, true, false,false,false,false, true, true,false,false,false},  //1 map pocket
  /*2*/  {true, true, true,false,false,false,false, true, true,false,false,false},  //2 middle tone + beltline lower part
  /*3*/  {false,false,false,true,false,false,false, false,true,false,false,false},  //3 ambient light
  /*4*/  {false,false,false, false, true,false,false, true, true,false,false,false},  //4 sew line + beltline upper part (rollo??)
  /*5*/  {false,false,false,false,false, true,false, true, true,false,false,false},  //5 snapshot1
  /*6*/  {false,false,false,false,false,false, true, true, true,false,false,false},  //6 snapshot2
  /*7*/  {true, true, true, false, true, true, true, true,true,false,false,false},  //7 trans snapshot 2
  /*8*/  {true, true, true, true, true, true, true, true, true, true,true,false},  //8 onpos
  /*9*/  {false,false,false,false,false,false,false,false, true, true,false,false},   //9 Calibration
  /*10*/ {false,false,false,false,false,false,false,false,true, false,true,true} ,  //10 transitio da MapPocket divider a home
  /*11*/ {false,false,false,false,false,false,false,false,false, false,true,true}   //11 MapPocket Divider
```

Fig. 6.1.2. Matrix of the robot possible movements.

6.2 Supervisor communication

The server communication program has been created in order to provide stable work of the workbench under different conditions, avoid operator mistakes and defend operator from the mechanical and electrical dangers which can be experienced during the workflow. It predicts all possible robot states and reacts to violations of the working process. According to created communication program is impossible to return false values to the database. In case of work interruption or violation of working cycle whole system returns failure, which can be fixed just with administrator authentication.


Connection
2
Check Safety
NORMAL
REDUCED
RECOVERY
PROTECTIVE STOP
SAFEGUARD STOP
Return alarm Send OnPosition
Return alarm Fix problem manually
Return alarm Send onPosition
Empty
1
1
Check Robot Mode
SYSTEM EMERGENCY
ROBOT EMERGENCY
VIOLATION
FAULT
Return alarm
Return alarm Fix manually
Return alarm Fix manually
Return alarm Fix manually
BOOTING
POWER OFF
BACKDRIVE
IDLE
NO CONTROLLER
Wait
POWER ON + wait
wait
Return alarm + Fix manually
Wait + Remember If it's second time – return alarm
3
4


3
4
RUNNING
POWER ON
DISCONNECTED
CONFIRM SAFETY
GET LOADED PROGRAM
wait
Return alarm + Safeguard stop
Return alarm + Fix manually
TRUE
2
1
LOAD PROGRAM
IS CORRECT PROGRAM LOADED
PLAY
NEEDED ACTION FROM OPERATOR
IS PROGRAM PLAYING
FAILED
POPUP
PROGRAM LOADING
FALSE
POPUP
PLAYING
PROGRAM STATE
PAUSED
STOPPED
1

7 Robot Safety Standards for Collaborative Operation ISO 10218

Due to the manufacturing requirements, used robotics equipment has to be chosen according to International Standard Organization requirements. Following standarts will be described further.

Global Industrial Robots Safety Standard:

ISO 10218-1: Manufacturer of robots.

ISO 10218-2: Integrator of robot systems/applications.

ISO/TS 15066: NOT a standard, but a Technical Specification with additional guidance on collaborative robots.

ISO 13849-1: Provides safety requirements and guidance on the principles for the design and integration of safety related parts of control systems (SRP/CS), including the design of software.

The European Machinery Directive 2006/42/EC is the machinery law relevant for all installations in Europe. It can be found on the European Commission’s official homepage

ISO 10218-1 Section 5.10 says: “Robots designed for collaborative operation shall provide a visual indication when the robot is in collaborative operation and shall comply with one or more of the requirements in 5.10.2 to 5.10.5”

ISO 10218-1 Section 5.10.5 Power and Force limiting by inherent design and control “The power and force limiting function of the robot shall be in compliance with 5.4. If any parameter limit is exceeded, a protective stop shall be issued. The robot is only a component in a final collaborative robot system and alone is not sufficient for a safe collaborative operation. The collaborative operation application shall be determined by the [risk assessment] performed during the application system design. Information for use shall include details for setting established parameter limits in the controlled robot

ISO 10218-2 shall be used for designing collaborative operations. Additional information will be contained in the [ISO/TS 15066]”.

Risk Assessment: A risk assessment is the overall process comprising a risk analysis and a risk evaluation. This means identifying all risks and reducing them to an appropriate level (See ISO 12100).

ISO/TS 15066 published February 15, 2016. It contains valuable guidance on risk assessment for the integrators of collaborative robots. It also includes a presentation of a research study on pain thresholds that an integrator voluntarily can choose to make use of. Universal Robots is part of ISO committee and has been an active participant in the ISO/TS 15066 development.

Regarding Robot Safety: All machines installed within EU shall comply with the essential health and safety requirements listed in ANNEX I of the Machine Directive 2006/42/EC (MD). It is not required by any law to comply with any standard, however, the standards ISO 10218-1, ISO 10218-2 and ISO 13849-1 are harmonized under the MD. If a machine complies with such a harmonized standard, it also complies with the essential requirements of the MD. The MD additionally requires the following documentation for the complete robot installation: • Clear instructions for the operator • Make a CE Declaration Of Conformity according to ANNEX II, 1., A • Put a CE mark on the complete robot installation according to ANNEX III including manufacturer contact information, installation date, TypeName and/or serial number. • Collect all information in one big technical file and store it for at least 10 years.

ISO 10218-1 Section 5.4: Safety related parts of control systems shall be designed so that they comply with PL=d with structure Category 3 as described in ISO 13849-1 (ISO 10218-1 section 5.2.2), or so that it complies with the PL determined by the risk assessment (ISO 10218-1 section 5.2.3). Performance Level (PL) is a discrete level used to specify the ability of safety-related parts of a control system to perform a safety function under foreseeable conditions. In other words it is a defined measure of how likely it is for a system to fail. In a PL=d system it is very unlikely that a dangerous failure will occur. (See ISO 13849-1 for more details) Category 3 is a term used about a system when it is designed as a dual-channel system. It is pretty common to construct Safety related systems as dual channel systems. Safety Category 3 means that a single fault does not lead to the loss of the safety function. Furthermore, most single faults are detected and well-tested safety principles have been applied. (See ISO 13849-1 for more details) The above is compliant with Universal Robots (UR) robots. The safety system on all UR Robots is PL=d with Category 3 for all safety I/O's.

Summary:

• It is not the robot alone that makes an application safe. It is the entire application that makes the application safe

• EVERY application needs a proper risk assessment

• Universal Robots comply with the current global safety standards for collaborative operation

• The safety system on all Universal Robots (UR) robots is designed to comply with a PL=d , Category 3 safety Interface

• The Technical Specification (TS) for collaborative robots (ISO/TS 15066) provides additional guidance for conducting risk assessment.

8 Conclusions

During the work has been developed the system of software and hardware for the stand of final assembly of automotive door panels, which will meet the requirements of the speed and quality of the production process. Were succeeded all technical and safety requirements of customer. Developed effective communication and monitoring system for workbench.

Workbench developed according to rules and requirements of collaborative work between human workers and robots. Satisfied safety requirements of movable parts and integrated system of errors avoiding during workflow.

Workbench has been prepared for IoT usage and can be easily integrated into Industry 4.0 environment.

9 Bibliography

1. ISO/TC 184/SC 2 “Robots and robotic devices”; working group WG3 “Industrial safety” is one of five active working groups in this subcommittee.
2. ISO 10218 “Robots and robotic devices – Safety requirements for industrial robots”, with parts 1 (“Robots”) and 2 (“Robot systems and integration”), ISO Copyright Office, Geneva, 2011.
3. A. De Santis, B. Siciliano, A. De Luca, and A. Bicchi, “Atlas of physical human-robot interaction,” Mechanism and Machine Theory, Vol. 43, No. 3, March 2008, p. 253-270.
4. O. Ogorodnikova, “How Safe the Human-Robot Coexistence Is? Theoretical Presentation,” Acta Polytechnica Hungarica, Vol. 6, No.4, p. 51-74 (2009).

Supervisor communication code

```
using System;
using System.Net;
using System.Net.Sockets;
using System.Text;

namespace URServer
{
  class Program
  {
        private float SafePos(float Position)
        {
            float x = 0, y = 0, z = 0, rx = 0, ry = 0, rz = 0;



           float[] JPos = new float [] { x,y,z,rx,ry,rz};

            return 0;
        }

        static void Main(string[] args)
    {

      // The IP address of the server (the PC on which this program is running)
```

```
// Standard port number
int nPort = 21;
      int a =1;// signature of element in the array ASSIGN ACCORDING TO THE PREVIOUS POINT
      int b = 0;// signature of element in the array ASSIGN ACCORDING TO THE NEXT POINT
      int c =0;
      bool Availability = false;
      bool[,] TR;

      // Create the IP address
      Console.WriteLine("Starting to listen on port: " + nPort);
TcpListener tcpListener = new TcpListener(nPort);  // Create the tcp Listener
tcpListener.Start();                                // Start listening

// Keep on listening forever
while (true)
{

  TcpClient tcpClient = tcpListener.AcceptTcpClient();      // Accept the client
  Console.WriteLine("Accepted new client");
```

```
NetworkStream stream = tcpClient.GetStream();             // Open the network stream
while (tcpClient.Client.Connected)
{
  // Create a byte array for the available bytes
  byte[] arrayBytesRequest = new byte[tcpClient.Available];
  // Read the bytes from the stream
  int nRead = stream.Read(arrayBytesRequest, 0, arrayBytesRequest.Length);
        if (nRead > 0)
        {
           // Convert the byte array into a string
           string sMsgRequest = ASCIIEncoding.ASCII.GetString(arrayBytesRequest);
           Console.WriteLine("Received message request: " + sMsgRequest);



           string sMsgAnswer = string.Empty;

           TR = new bool[,]
        {       { true, false,false,false,false,true, false}, //0
                { false,true, true, true, false,true, true},  //1
                { false,true, true, true, false,true, true},  //2
                { false,true, true, true, false,true, true},  //3
                { false,false,false,false,true, false,true},  //4
                { true, true, true, true, false,true, true},  //5
                { false,true, true, true, true, true, true}   //6
```

```
};



string Point0 = "0.230285,-1.10077,1.95491,-0.750319,-
0.579063,0.0156387";

string Point1 = "0.463885,-1.51588,2.28166,-2.3451,-1.57855,0.0155907";

string Point2 = "0.59939,-0.839762,1.26926,-2.03362,-
1.57858,0.0155787";

string Point3 = "0.803725,-0.839701,1.26925,-2.03361,-
1.57859,0.0155787";

string Point4 = "0.980461,-0.553382,0.858255,-0.626309,-
2.88645,0.0156027";

string Point5 = "0.226977,-1.40376,1.80796,-0.723061,-
0.57911,0.0156506";

string Point6 = "1.00714,-0.716358,0.857476,-0.630289,-
2.92502,0.0155668";

string Point99 = "0,0,0,0,0,0";


Console.WriteLine("a= " + a);

Console.WriteLine("b= " );

string LineInput = Console.ReadLine();

c = Int32.Parse(sMsgRequest);

b = Int32.Parse(LineInput);

Availability = TR[a, b];

if (TR[a, b] == true)

{
```

```
{
    switch (b)
    {
        case 0:
            sMsgAnswer = "("+Point0 + "," + Point99 + ")";
            break;
        case 1:
            sMsgAnswer ="("+ Point1 + "," + Point99 + ")";
            break;
        case 2:
            sMsgAnswer = "("+ Point2 + "," + Point99 + ")";
            break;
        case 3:
            sMsgAnswer = "("+ Point3 + "," + Point99 + ")"; ;
            break;
        case 4:
            sMsgAnswer = "("+ Point4 + "," + Point99 + ")";
            break;
        case 5:
            sMsgAnswer = "(" + Point5 + "," + Point99 + ")";
            break;
        case 6:
            sMsgAnswer = "(" + Point6 + "," + Point99 + ")";
            break;
```

```
                        default: break;
                    }


                }
            }
            else
            {
                switch (b)
                {
                    case 0:
                        if (a == 4 && b == 0) // ADD THIRD POINT
                        {
                            sMsgAnswer = "(" + Point6 + "," + Point5 + ")";


                            byte[] arrayBytesAnswer =
ASCIIEncoding.ASCII.GetBytes(sMsgAnswer);
                            stream.Write(arrayBytesAnswer, 0, arrayBytesAnswer.Length);
                            {
                                System.Threading.Thread.Sleep(2500);
                            }
                            sMsgAnswer = "(" + Point5 + "," + Point0 + ")";
                        }
                        else
                        {
```

```
        sMsgAnswer = "(" + Point5 + "," + Point0 + ")";
    }
    break;

case 1:
    if (a == 0 && b == 1)
    {
        sMsgAnswer = "(" + Point5 + "," + Point1 + ")";
    }
    if (a == 4 && b == 1)
    {
        sMsgAnswer = "(" + Point6 + "," + Point1 + ")";
    }
    break;

case 2:
    if (a == 0 && b == 2)
    {
        sMsgAnswer = "(" + Point5 + "," + Point2 + ")";
    }
    if (a == 4 && b == 2)
    {
        sMsgAnswer = "(" + Point6 + "," + Point2 + ")";
    }
    break;
```

```
case 3:
    if (a == 0 && b == 3)
    {

        sMsgAnswer = "(" + Point5 + "," + Point3 + ")";
    }
    if (a == 4 && b == 3)
    {

        sMsgAnswer = "(" + Point6 + "," + Point3 + ")";
    }
    break;


case 4:
    if ( a == 0 && b == 4) // ADD THIRD POINT
    {
        sMsgAnswer = "(" + Point5 + "," + Point6 + ")";

        byte[] arrayBytesAnswer =
ASCIIEncoding.ASCII.GetBytes(sMsgAnswer);
        stream.Write(arrayBytesAnswer, 0, arrayBytesAnswer.Length);
        {
            System.Threading.Thread.Sleep(2500);
```

```
                }
                sMsgAnswer = "(" + Point6 + "," + Point4 + ")";
            }
            else
            sMsgAnswer = "(" + Point6 + "," + Point4 + ")";
            break;

        case 5:
            if (a == 4 && b == 5)
                sMsgAnswer = "(" + Point6 + "," + Point5 + ")";
            break;

        case 6:
            sMsgAnswer = "(" + Point5 + "," + Point6 + ")";
            break;
        default: break;
    }
}

a = b;



// Check which workpoint is requested


 if (sMsgAnswer.Length > 0)
```

Mea.	Sheet	№ doc.	Sign	D.	ED.MD.18	Page 57

```
            {
              Console.WriteLine("Sending message answer: " + sMsgAnswer);



                    Console.WriteLine("Availability= " + Availability);
                    // Convert the point into a byte array
                    byte[] arrayBytesAnswer = ASCIIEncoding.ASCII.GetBytes(sMsgAnswer);
              // Send the byte array to the client
              stream.Write(arrayBytesAnswer, 0, arrayBytesAnswer.Length);
            }
          }
          else
          {
            if (tcpClient.Available == 0)
            {
              Console.WriteLine("Client closed the connection.");
              // No bytes read, and no bytes available, the client is closed.
              stream.Close();
            }
          }
        }
      }
    }
  }
}
```

Код программы – сервера ПК

```
using System;

namespace RobotComm
{
    public class robot_interface
    {
        public enum states
        {
            not_connected,
            connecting,
            connected,
            ready,
        }

        public class log
        {
            public enum types
            {
                @event,
                recv,
                send,
            }
```

```
        public readonly types type;
        public readonly string text;

        public log(types type, string text)
        {
            this.type = type;
            this.text = text;
        }
    }

    private System.Threading.Thread m_thread = null;
    private string m_last_exception = string.Empty;
    private string m_address = null;
    private int m_port = 0;
    private readonly System.Collections.Generic.Queue<string> m_send_queue = new
System.Collections.Generic.Queue<string>();
    private states m_state = states.not_connected;
    private bool m_connect = false;
    private readonly System.Collections.Generic.List<log> m_log = new
System.Collections.Generic.List<log>();

    public states state
    {
        get { return m_state; }
    }
```

```
~robot_interface()
{
   stop();
}

public bool start()
{
   // Stop thread first, if already running
   stop();

   try
   {
      m_thread = new System.Threading.Thread(new
System.Threading.ThreadStart(thread_function_socket));
      m_thread.Start();
      return true;
   }
   catch (Exception ex)
   {
      Program.ErrorHandler(ex,
System.Reflection.MethodBase.GetCurrentMethod());
      return false;
   }
}

public void stop()
```

```
        {
            if (m_thread != null)
            {
                try
                {
                    m_thread.Abort();
                    m_thread.Join();
                }
                catch (Exception ex)
                {
                    Program.ErrorHandler(ex,
System.Reflection.MethodBase.GetCurrentMethod());
                }
                m_thread = null;
            }
        }

        private void ErrorHandler(Exception ex, System.Reflection.MethodBase method)
        {
            // evita di visualizzare continuamente lo stesso messaggio
            if (ex.Message != m_last_exception)
            {
                m_last_exception = ex.Message;
                Program.ErrorHandler(ex,
System.Reflection.MethodBase.GetCurrentMethod());
            }
```

```
}

public void connect(string address, int port)
{
    m_address = address;
    m_port = port;
    m_connect = true;
}

public void disconnect()
{
    m_connect = false;
}

private void thread_function_socket()
{
    byte[] buffer = new byte[1024000];

    while (true)
    {
        try
        {
            m_state = states.not_connected;
            while (!m_connect)
            {
```

```
            System.Threading.Thread.Sleep(500);
        }

        while (m_connect)
        {
            m_state = states.connecting;
            add_log(log.types.@event, "connecting...");
            // create the socket
            using (var socket = new
System.Net.Sockets.Socket(System.Net.Sockets.AddressFamily.InterNetwork,
System.Net.Sockets.SocketType.Stream, System.Net.Sockets.ProtocolType.Tcp))
            {
                bool connected = false;

                // connect to server
                while (m_connect && !connected)
                {
                    var ar = socket.BeginConnect(m_address, m_port, null, null);
                    ar.AsyncWaitHandle.WaitOne();
                    try
                    {
                        socket.EndConnect(ar);
                        connected = true;
                        add_log(log.types.@event, "connected");
                    }
                    catch (ObjectDisposedException)
```

```
                {
                }
                catch (Exception ex)
                {
                    ErrorHandler(ex,
System.Reflection.MethodBase.GetCurrentMethod());
                }
            }

            if (m_connect)
            {
                m_state = states.connected;

                // imposta il timeout per poter leggere
                socket.ReceiveTimeout = 500;

                int step = 0;
                var received = new System.Text.StringBuilder();

                while (m_connect && connected)
                {
                    // receive data
                    try
                    {
                        int count = socket.Receive(buffer);
```

```
if (!OnReceive(received, buffer, count, ref step))
{
    connected = false;
}
}
catch (System.Net.Sockets.SocketException ex)
{
    if (ex.SocketErrorCode !=
System.Net.Sockets.SocketError.TimedOut)
    {
        ErrorHandler(ex,
System.Reflection.MethodBase.GetCurrentMethod());
        connected = false;
    }
}
catch (Exception ex)
{
    ErrorHandler(ex,
System.Reflection.MethodBase.GetCurrentMethod());
    connected = false;
}

// send data
if (connected)
{
    if (m_send_queue.Count > 0)
```

```
{
    var command = m_send_queue.Peek();
    if (send(socket, command + '\n'))
    {
        m_send_queue.Dequeue();
    }
    else
    {
        connected = false;
    }
}
else
{
    var command = get_command_to_send(step);
    if (command != null)
    {
        if (send(socket, command))
        {
            step = step;
        }
        else
        {
            connected = false;
        }
    }
```

```
                    }
                  }


                  {
                    System.Threading.Thread.Sleep(500);
                  }
                }
              }
            }
          }
        }
        catch (System.Threading.ThreadAbortException)
        {
          add_log(log.types.@event, "aborted");
          m_state = states.not_connected;
          return;
        }
        catch (Exception ex)
        {
          ErrorHandler(ex, System.Reflection.MethodBase.GetCurrentMethod());
        }

        add_log(log.types.@event, "disconnected");
        m_state = states.not_connected;
```

```
    }
}

private string get_command_to_send(int step)
{
    try
    {
        switch (step)
        {
            case 0:
                return "safetymode\n";
            case 1:
                return "robotmode\n";
            case 2:
                return "power on\n";
            case 4:
                return "brake release\n";
            case 6:
                return "load /programs/URClientMOVEJ.urp\n";


            case 8:
                return "play\n";
            default:
                return null;
        }
```

```
    }
    catch (Exception ex)
    {
        ErrorHandler(ex, System.Reflection.MethodBase.GetCurrentMethod());
        return null;
    }
}



private int calculate_next_step(string robot_answer)
{
    switch (robot_answer)
    {
        case "Safetymode: NORMAL":
            return 1;   // Check the robot mode

        case "Safetymode: REDUCED":
            return 1;   // fffffFFFF

        case "Safetymode: PROTECTIVE_STOP":
            return 1;   // fffffFFFF

        case "Safetymode: RECOVERY":
            return 1;   // fffffFFFF
```

```
case "Safetymode: SAFEGUARD_STOP":
    return 1;   // fffffFFFF

case "Safetymode: SYSTEM_EMERGENCY_STOP":
    return 1;   // fffffFFFF

case "Safetymode: ROBOT_EMERGENCY_STOP":
    return 1;   // fffffFFFF

case "Safetymode: VIOLATION":
    return 1;   // fffffFFFF

case "Safetymode: FAULT":
    return 1;   // fffffFFFF



case "Robotmode: IDLE":
    return 0;   // Check the safety mode

case "Robotmode: BACKDRIVE":
    return 0;   // Check the safety mode

case "Robotmode: CONFIRM_SAFETY":
    return 0;   // Check the safety mode
```

```
case "Robotmode: NO_CONTROLLER":
    return 0;   // Check the safety mode

case "Robotmode: POWER_OFF":
    return 2;   // Powering on

case "Robotmode: RUNNING":
    return 6;   // Loading the program

case "Robotmode: POWER_ON":
    System.Threading.Thread.Sleep(1000);
    return 1;   // Check the robot mode



case "Powering on":
    return 4;

case "Brake releasing":
    return 1;



case "Loading program: /programs/URClientMOVEJ.urp":
    return 8;
```

```
                default:
                    return 1;
            }
        }

        private bool OnReceive(System.Text.StringBuilder received, byte[] buffer, int
count, ref int step)
        {
            try
            {
                if (count <= 0)
                {
                    return false;
                }

                string text = System.Text.Encoding.UTF8.GetString(buffer, 0, count);
                System.Diagnostics.Trace.WriteLine("recv: " + text);

                // aggiunge al buffer
                received.Append(text);

                // spacchetta i barcode, nel caso siano più di uno
                while (received.Length > 0)
                {
                    text = received.ToString();
```

```
if (text.Length == 0)
{
    break;
}

int end = text.IndexOf('\n');
if (end < 0)
{
    break;
}

int len = end + 1;
text = text.Substring(0, len - 1);
add_log(log.types.recv, text);

step = calculate_next_step(text);



if (len >= received.Length)
{
    System.Diagnostics.Debug.Assert(len == received.Length);
    received.Clear();
}
else
{
```

```
                received.Remove(0, len);
              }
            }

            return true;
          }
          catch (Exception ex)
          {
            ErrorHandler(ex, System.Reflection.MethodBase.GetCurrentMethod());
            return false;
          }
        }

        private bool send(System.Net.Sockets.Socket socket, string text)
        {
          try
          {
            System.Diagnostics.Trace.WriteLine("send: " + text);
            add_log(log.types.send, text);
            var buffer = System.Text.Encoding.UTF8.GetBytes(text);
            var ar = socket.BeginSend(buffer, 0, buffer.Length,
System.Net.Sockets.SocketFlags.None, null, null);
            ar.AsyncWaitHandle.WaitOne();
            int count = socket.EndSend(ar);
            System.Diagnostics.Debug.Assert(count > 0);
```

```
            return true;
        }
        catch (ObjectDisposedException)
        {
            return false;
        }
        catch (Exception ex)
        {
            ErrorHandler(ex, System.Reflection.MethodBase.GetCurrentMethod());
            return false;
        }
    }

    public void send_command(string text)
    {
        try
        {
            m_send_queue.Enqueue(text);
        }
        catch (Exception ex)
        {
            ErrorHandler(ex, System.Reflection.MethodBase.GetCurrentMethod());
            throw;
        }
    }
```

```
private void add_log(log.types type, string text)
{
   try
   {
      lock (m_log)
      {
         m_log.Add(new log(type, text));
      }
   }
   catch (Exception ex)
   {
      ErrorHandler(ex, System.Reflection.MethodBase.GetCurrentMethod());
      throw;
   }
}

public System.Collections.Generic.List<log> get_log()
{
   try
   {
      lock (m_log)
      {
         var list = new System.Collections.Generic.List<log>(m_log);
         m_log.Clear();
```

```
                return list;
            }
        }
        catch (Exception ex)
        {
            ErrorHandler(ex, System.Reflection.MethodBase.GetCurrentMethod());
            throw;
        }
    }
  }
}
```

Client program for the UR5

```
Program
  BeforeStart
    open:=socket_open("192.168.0.138",21)
    Loop open≟ False
      open:=socket_open("192.168.0.138",21)
    targetPos:=p[0,0,0,0,0,0]
    counter:=6
  Robot Program
    sendToServer:='send to server'
    socket_send_string(sendToServer)
    receiveFromServ:=socket_read_ascii_float(6)
    Loop receiveFromServ[0]≠6
      Wait: 0.3
      receiveFromServ:=socket_read_ascii_float(6)
    Loop counter<6
      targetPos[counter]=receiveFromServ[counter+1]
      counter:=counter+1
    MoveJ
      targetPos
```