SELECTED ENGINEERING PROBLEMS
NUMBER5
INSTITUTE OF ENGINEERING PROCESSES AUTOMATION
AND INTEGRATED MANUFACTURING SYSTEMS

Piotr MICHALSKI[1],Krzysztof TKOCZ[1]

[1] Institute of Engineering Processes Automation and Integrated Manufacturing Systems, Faculty of Mechanical Engineering, Silesian University of Technology, Gliwice
* piotr.michalski@polsl.pl

THE ALGORITHMS SUPPORT FOR VISUALISATION OF INDUSTRIAL ROBOT MOVEMENTS

Abstract: This paper presents a practical implementation of algorithms for correctly visualize the movements of an industrial robot in the SCADA type visualization. As an example, the authors chose the three-arm robot with end effector.

1. Introduction

Modern industrial process visualization systems allow you to view images with high resolution. With processors for high performance computing, it is possible to implement algorithms to support accurate visualization of selected elements of the process. This article describes the algorithms necessary to perform professional visualization of an industrial robot, consisting of 3 arms and gripper. As a sample SCADA system the CodeSys 3S has been chosen.

2. Industrial robot movements

The movement of the robot will be controlled by forcing the displacement of one of the four arms. Authors created a model of an industrial robot shown in Figure 1a, just to perform the visualization. When modeling a robot arms, users should pay attention to the pivot point of the arm, it has to be located in the same position as the center of rotation axis. One of example robot arm in Figure 2b has been presented. Assuming that for the changes in the rotation angle will be responsible such a variable as: rRR1up to rRR4, as well as taking the RX1 up to X4 and RY1 up to RY4 as auxiliary variables to determine the displacement, simple function code has been developed:

```
(* position point – axis no 2, arm 2 *)
rX1:=rRR1*0.01745329;    (* conversion of an angle *)
rX2:=SIN(rX1)*126;              (* determine the horizontal position X *)
rY1:=COS(rX1)*126;              (* additional help variable *)
rY2:=126-rY1;                   (* determine the vertical position Y *)
```

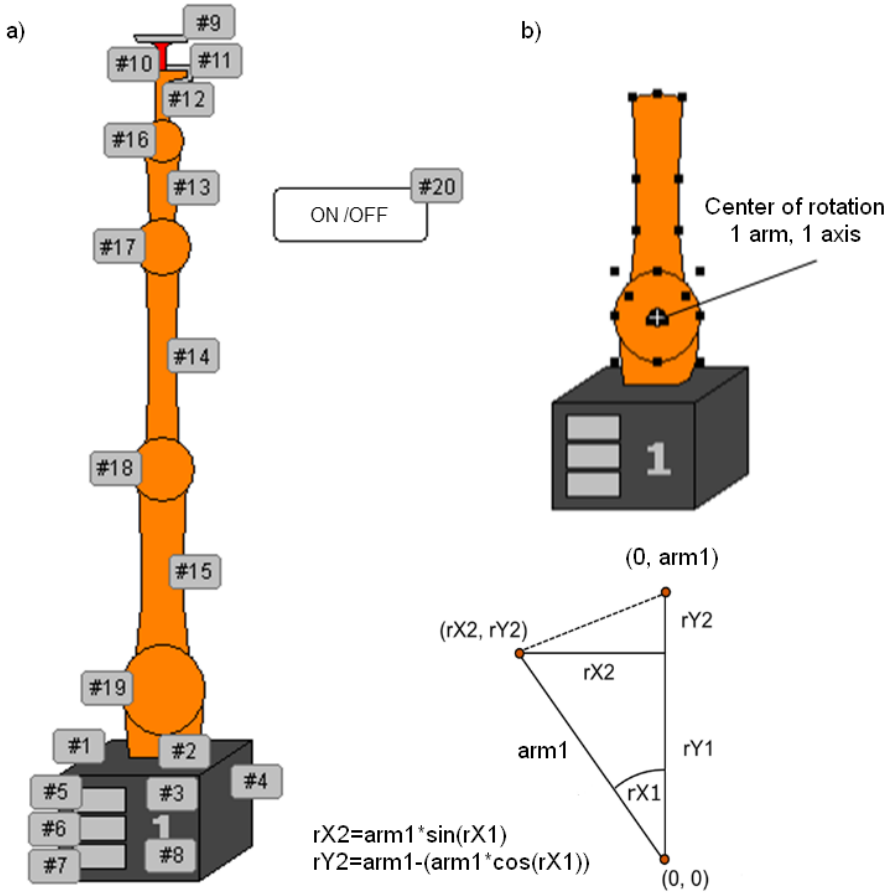

Fig. 1. Model of industrial robot: a) view with part numbers, b)

In the CoDeSys software the function of SIN and COS are calculated in radian measure, therefore, is required for the conversion. The length of the first arm (126 pix) measured from the pivot point to the last point of the upper - see Figure 1b. The moving parts are at the end of the first arm (# 14, # 18) is required to determine the position of X and Y as changing its angle of rotation. For this purpose, the function using simple trigonometric transformations determines the value of the displacement in the vertical and horizontal axes see the equation on Figure 1b. The moving parts at the ends of the arm of the second and third proceed similarly, add additional displacement determined by means of further parts of programs.

```
(* position point – axis no 3, arm 3 *)
  rX3:=rRR2*0.01745329;
  rX4:=SIN(rX3)*126;
  rY3:=COS(rX3)*126;
  rY4:=126-rY3;
(* position point – axis no 4, arm 4, and additional elements *)
  rX5:=rRR3*0.01745329;
  rX6:=SIN(rX5)*58;
  rY5:=COS(rX5)*58;
  rY6:=58-rY5;
```

Transformation of the trigonometric functions used to designate the position of subsequent visualization elements of an industrial robot shown on Figure 2.

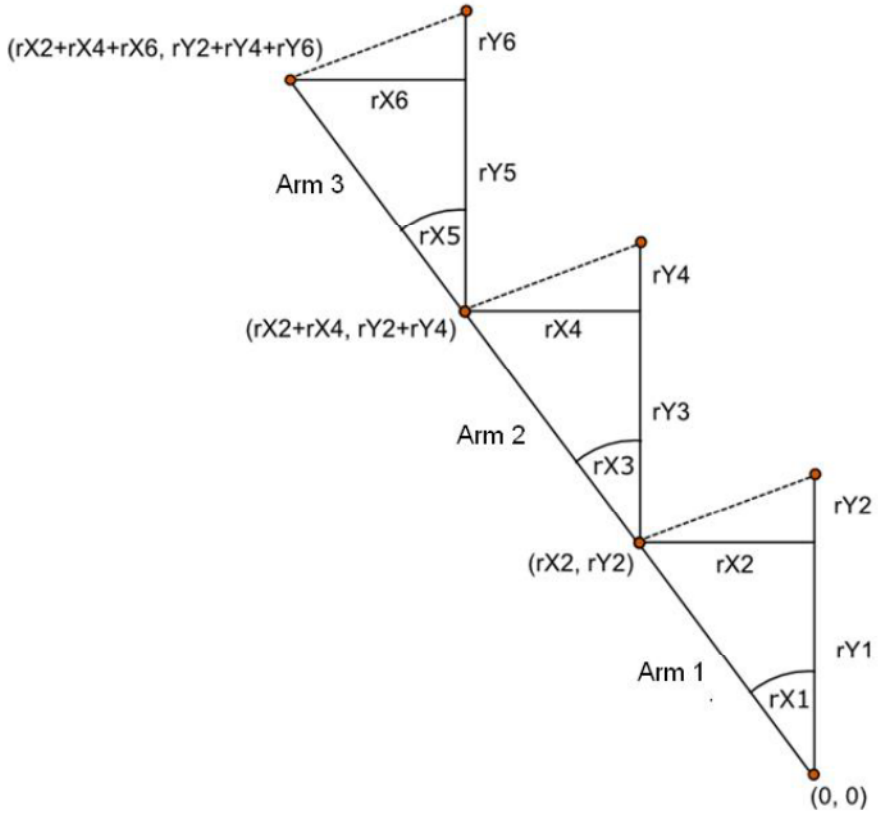

Fig. 2. Determination of the position of the items placed on the end of the third arm

The algorithm of the control program is based on forcing the successive movements of the arms of an industrial robot in order to reach the set-point position. In the example shown in this article, a program responsible for the movement of the robot from base position, in which the values of the angles are equal to zero to the position of pickup the item detail (movement of each arm mowing sequence: 20, -100, -150, -180). Global variable defined in the program and used by the robot are as follows:

```
(* main program variables *)
bStartRobot1:BOOL:=FALSE;                (* START/STOP of the robot *)

(* robot status variables *)
bWlaczonyRobot1:BOOL:=TRUE;  (* robot is ON *)
bPracaRobot1:BOOL:=FALSE;                (* robot is BUSY *)
bCzekaRobot1:BOOL:=FALSE;                (* robot is WHITING FOR A JOB *)


(* robot speed control variables *)
rPredkoscRobot1Ramie1:REAL:=2; (* speed of arm 1 *)
rPredkoscRobot1Ramie2:REAL:=2; (* speed of arm 2 *)
rPredkoscRobot1Ramie3:REAL:=2; (* speed of arm 3 *)
rPredkoscRobot1Ramie4:REAL:=2; (* speed of arm 4 *)
rPredkoscRobot1Dojazd:REAL:=1;  (* additional speed of base *)
```

Creation of an additional movement simulation program for proper operation of the visualization has been required. In our example, as the PLC programming language the ST language has been selected. The code automatically forces the variables, so we are able to see the robot movements in the visualization. The proper code looks like:

```
PROGRAM VARIABLE_GEN
VAR
        blinkGen: BLINK;
        ctuLicznik: CTU;
        rSygnal: REAL;
END_VAR
    blinkGen(ENABLE:=bWylacznikAwaryjny, TIMELOW:=T#50ms,TIMEHIGH:=T#50ms);
    ctuLicznik(CU:=blinkGen.OUT, RESET:=rSygnal=1);
    rSygnal:=ctuLicznik.CV;
```

The motion simulation used a pulse generator (Blink-like element located inside the Util.lib), which in defined intervals gives on the output the high logical state (BOOL variable). Since the simulation will need variable of type REAL, in addition to the output of the generator the up counter CTU (library Standard.lib) has been connected, which reset condition was set as CV> 1. RSygnal variable defined at the output of the counter. Thus prepared, the system allows you to generate a variable of REAL type in equal to 1 specified intervals.

The next step is to create a robot motion control program, including to insert a new module called Robot_1 (as the PLC language we choose SFC), then declare local variables used in the control program. The code looks like:

```
PROGRAM Robot_1

VAR
        (* Condition to pass step 1 to 8 *)
        bWarunek1: BOOL;
        ...
        bWarunek8: BOOL;

        (* variable of displacement *)
        rX1: REAL; rX2: REAL;
        rX3: REAL; rX4: REAL;
        rX5: REAL; rX6: REAL;
        rY1: REAL; rY2: REAL;
        rY3: REAL; rY4: REAL;
        rY5: REAL; rY6: REAL;

        (* variables responsible for movements of each arm *)
        rRR1: REAL;
        rRR4: REAL;
        rRR3: REAL;
        rRR2: REAL;
END_VAR
```

In order to achieve smooth motion simulation function of displacement of the robot to be placed in each block of shares, which are associated with the movement instructions. You should also be aware of the negation of the previous transition conditions placed on top of all the blocks. Below are the contents of each block of shares robot control program. Boot (the INIT) block is the first element of the program, which provides condition for the start of the robot and the control variables statuses.

```
Start:
        bWarunek8:=FALSE;
    IF bStartRobot1=TRUE THEN          (* condition of starting *)

                (* statuses variables *)
                bWlaczonyRobot1:=FALSE;
                bPracaRobot1:=FALSE;
                bCzekaRobot1:=TRUE;

                bWarunek1:=TRUE; (* transition condition = open the gate *)
    END_IF
```

In the next container block (Akcja_1) provided instructions for changing the displacement of each robot arms. The principle of operation is the addition of a variable pulse motion simulation variables corresponding to the displacement of each of the arms. The condition of leaving the block is the movement of the first arm by an angle of 20 degrees. Conditions for the other arms ensure you do not exceed the target values.

```
Akcja_1:
        bWarunek1:=FALSE; (* transition condition = close the gate *)

        (* variables for movements *)
        rRR1:=rRR1+rPredkoscRobot1Ramie1*Variable_gen.rSygnal;
        rRR2:=rRR2-rPredkoscRobot1Ramie2*Variable_gen.rSygnal;
        rRR3:=rRR3-rPredkoscRobot1Ramie3*Variable_gen.rSygnal;
        rRR4:=rRR4-rPredkoscRobot1Ramie4*Variable_gen.rSygnal;

        (* function of displacements *)
        rX1:=rRR1*0.01745329;
        rX2:=SIN(rX1)*126;
        rY1:=COS(rX1)*126;
        rY2:=126-rY1;
        rX3:=rRR2*0.01745329;
        rX4:=SIN(rX3)*126;
        rY3:=COS(rX3)*126;
        rY4:=126-rY3;
        rX5:=rRR3*0.01745329;
        rX6:=SIN(rX5)*58;
        rY5:=COS(rX5)*58;
        rY6:=58-rY5;

        (* END conditions *)
        IF rRR4<=-180 THEN rRR4:=-180; END_IF
        IF rRR3<=-150 THEN rRR3:=-150; END_IF
        IF rRR2<=-100 THEN rRR2:=-100; END_IF
        IF rRR1>=20 THEN bWarunek2:=TRUE; END_IF
```

Instructions for enabling positioning of the other arms are located in action blocks Akcja_2 up to Akcja_4. Terms transitions are fulfilled by reaching preset positions for the next industrial robot arms.

```
Akcja_2:
        bWarunek2:=FALSE;
        rRR2:=rRR2-rPredkoscRobot1Ramie2*Variable_gen.rSygnal;
```

```
        rRR3:=rRR3-rPredkoscRobot1Ramie3*Variable_gen.rSygnal;
        rRR4:=rRR4-rPredkoscRobot1Ramie4*Variable_gen.rSygnal;



        (*function of displacements *)
        IF rRuchRamie4<=-180 THEN rRR4:=-180; END_IF
        IF rRuchRamie3<=-150 THEN rRR3:=-150; END_IF
        IF rRuchRamie2<=-100 THEN bWarunek3:=TRUE; END_IF

Akcja_3
        bWarunek3:=FALSE;
        rRR3:=rRR3-rPredkoscRobot1Ramie3*Variable_gen.rSygnal;
        rRR4:=rRR4-rPredkoscRobot1Ramie4*Variable_gen.rSygnal;

        (*function of displacements *)
        IF rRuchRamie4<=-180 THEN rRR4:=-180; END_IF
        IF rRuchRamie3<=-150 THEN bWarunek4:=TRUE; END_IF

Akcja_4:
        bWarunek4:=FALSE;
        rRR4:=rRR4-rPredkoscRobot1Ramie4*Variable_gen.rSygnal;

        (*function of displacements *)
        IF rRuchRamie4<=-180 THEN bWarunek5:=TRUE; END_IF
```

On the screen we should be able to see the visualization of fluid motion robot moving to the target position.

3. Conclusion

The example can be freely expanded by adding the value of movements for each arm in successive blocks of control. After we create the control program we should assign a variable displacement to each arm, also keeping in mind the account of the previous arms movements, as shown in Figure 2. The final result visualization looks very realistic.

References

1. Legierski T., Kasprzyk J.: Programowanie sterowników PLC, WNT, Gliwice, 2008.
2. Sałat R., Korpysz K., Obstawski P.: Wstęp do programowania sterowników PLC, WNT, Warszawa, 2010.
3. Kasprzyk J.: Programowanie sterowników przemysłowych, WNT, Warszawa, 2006.
4. Programing manual for CoDeSys, Warszawa, 2010.