Spot Welding Simulation Report
Assignment 4

Théo de Morais // Jesús Sánchez

1. Introduction

This report documents the process and results of the spot welding simulation using RobotStudio and the TATEM tool on the ABB robot Rudolf. The objective is to demonstrate the operation, record simulation results, and analyze performance improvements.

3. RobotStudio Simulation

3.2 Modified Simulation

- Modify testPeg00() to request a peg position.
- Implement looping to select multiple pegs.
- Add rotation options between -180° and 180° in 45° increments.

```
PROC testPeg00()
    VAR num pegNum;
    VAR num dx;
    VAR num dy;
    VAR num rotation;
    VAR num numTime := 0;
    VAR bool isValid;

    WHILE TRUE DO
    ! Ask for peg number
    TPReadNum pegNum, "Enter peg number (1-16) or invalid to quit:";

    ! Convert peg number to (dx, dy)
    isValid := TRUE;
    IF pegNum = 1 THEN
          dx := 0;   dy := 0;
    ELSEIF pegNum = 2 THEN
          dx := 0;   dy := 60;
    ELSEIF pegNum = 3 THEN
```

```
            dx := 0;   dy := 120;
      ELSEIF pegNum = 4 THEN
            dx := 0;   dy := 180;
      ELSEIF pegNum = 5 THEN
            dx := 60;  dy := 0;
      ELSEIF pegNum = 6 THEN
            dx := 60;  dy := 60;
      ELSEIF pegNum = 7 THEN
            dx := 60;  dy := 120;
      ELSEIF pegNum = 8 THEN
            dx := 60;  dy := 180;
      ELSEIF pegNum = 9 THEN
            dx := 120; dy := 0;
      ELSEIF pegNum = 10 THEN
            dx := 120; dy := 60;
      ELSEIF pegNum = 11 THEN
            dx := 120; dy := 120;
      ELSEIF pegNum = 12 THEN
            dx := 120; dy := 180;
      ELSEIF pegNum = 13 THEN
            dx := 180; dy := 0;
      ELSEIF pegNum = 14 THEN
            dx := 180; dy := 60;
      ELSEIF pegNum = 15 THEN
            dx := 180; dy := 120;
      ELSEIF pegNum = 16 THEN
            dx := 180; dy := 180;
      ELSE
            TPWrite "Invalid peg number!";
            EXIT;
      ENDIF

      ! Ask for rotation
      TPReadNum rotation, "Enter rotation (-180 to 180, steps of 45):";

      IF (rotation MOD 45 <> 0) OR (rotation < -180) OR (rotation > 180)
THEN
            TPWrite "Invalid rotation! Try again.";
            isValid := FALSE;
      ENDIF

      ! Check reachability
      IF isValid THEN
```

```
            isValid := isReachable(dx, dy, rotation);
            IF NOT isValid THEN
                  TPWrite "Target point is not reachable! Choose another.";
            ENDIF
      ENDIF

      ! Execute peg test if valid
      IF isValid THEN
            !TPWrite "Testing peg at (" \Num:=dx, ", " \Num:=dy, "),
Rotation: " \Num:=rotation;

            MoveAbsJ jCalibPos,vFast,z10,tool0;
            ClkReset clock1;
            ClkStart clock1;

            initTatemTool;
            doPeg00 dx, dy, rotation;

            MoveAbsJ jCalibPos,vFast,z10,tool0;
            numTime := ClkRead(clock1);
            TPWrite "Time used on testPeg00() [s] = " \Num:=numTime;
      ENDIF
      ENDWHILE
ENDPROC

   FUNC bool isReachable(num dx, num dy, num rotation)
        VAR bool reachable;

        CONST num X_MIN := -200;
        CONST num X_MAX := 200;
        CONST num Y_MIN := -200;
        CONST num Y_MAX := 200;
        CONST num ROT_MIN := -180;
        CONST num ROT_MAX := 180;

        ! Check if dx, dy are within reachable limits
        reachable := (dx >= X_MIN) AND (dx <= X_MAX) AND
                     (dy >= Y_MIN) AND (dy <= Y_MAX);

        ! Check if rotation is valid
        reachable := reachable AND (rotation MOD 45 = 0) AND
                     (rotation >= ROT_MIN) AND (rotation <= ROT_MAX);
```

```
    RETURN reachable;
ENDFUNC
```

- Document reachable positions in a table.

No.	Dx,Dy	-180	-135	-90	-45	0	45	90	135	180
1	0,0	✅	✅	✅	✅	✅	✅	✅	❌	✅
2	0,60	✅	✅	✅	✅	✅	✅	✅	❌	✅
3	0,120	✅	✅	✅	✅	✅	✅	✅	❌	✅
4	0,180	✅	✅	✅	✅	✅	✅	✅	❌	✅
5	60,0	✅	✅	✅	✅	✅	✅	✅	❌	✅
6	60,60	✅	✅	✅	✅	✅	✅	✅	❌	✅
7	60,120	✅	✅	✅	✅	✅	✅	✅	❌	✅
8	60,180	✅	✅	✅	✅	✅	✅	✅	❌	✅
9	120,0	✅	✅	✅	✅	✅	✅	✅	❌	✅
10	120,60	✅	✅	✅	✅	✅	✅	✅	❌	✅
11	120,120	✅	✅	✅	✅	✅	✅	✅	❌	✅
12	120,180	✅	✅	✅	✅	✅	✅	✅	❌	✅
13	180,0	✅	✅	✅	✅	✅	✅	✅	❌	✅
14	180.60	✅	✅	✅	✅	✅	✅	✅	❌	✅
15	180,120	✅	✅	✅	✅	✅	✅	✅	❌	✅
16	180,180	❌	✅	✅	✅	✅	✅	✅	❌	❌

3.3 Extended Simulation

- Implement test7pegs() function.

- Maintain original test board layout and observe movement behavior.
- Record used rotations and total time from the FlexPendant.

```
        PROC test7pegs()
        VAR num numTime := 0;

        IF useFlexPendant THEN
                TPWrite "test7pegs() started";
        ENDIF

        MoveAbsJ jCalibPos,vFast,z10,tool0;
        ClkReset clock1;
        ClkStart clock1;

        initTatemTool;
        doPeg00 0, 180, 90; !1
        doPeg00 0, 0, 0;
        doPeg00 180, 0, 90;
        doPeg00 60, 180, -45;
        doPeg00 60, 60, 45;
        doPeg00 120, 120, 0;
        doPeg00 180, 120, -45;

        MoveAbsJ jCalibPos,vFast,z10,tool0;
        numTime := ClkRead(clock1);

        IF useFlexPendant THEN
                TPWrite "Time used on test7pegs() [s] = " \Num:=numTime;
        ENDIF
        ENDPROC


        PROC doPeg00(num dx, num dy, num rotz)
        ! arguments here should be given relative to work object
        ! but RelTool adjust position reltive to tool coordinate system
        ! Below sign of dx is kept, sign of dy (and dz) and rotation around
z-axis are reversed,
        ! this will, for the cases here, make arguments dx, dy, and rotz as
if they were
        ! related to the work object.
        !
        VAR num t1 := 0.025;    ! at peg wait t1 and the activate tool
```

```
      VAR num t2 := 0.450;   ! then wait t2, staying calm on peg
      VAR num t3 := 0.125;   ! deactivate tool, and wait t3 until moving
from peg
      !
      IF doSlow THEN
            MoveL RelTool(Peg00, dx, -dy, -50, \Rz:= -rotz), vFast, z5,
TatemTool1\WObj:=wobjTestBoard;
            MoveL RelTool(Peg00, dx, -dy,   0, \Rz:= -rotz), vSlow, fine,
TatemTool1\WObj:=wobjTestBoard;
            WaitTime t1;
            SetDO AirValve, 1;  ! activate tool
            WaitTime t2;
            SetDO AirValve, 0;  ! deactivate tool
            WaitTime t3;
            MoveL RelTool(Peg00, dx, -dy, -50, \Rz:= -rotz), vSlow, z5,
TatemTool1\WObj:=wobjTestBoard;
      ELSE
            ! here try to do a faster 'weld simulation', using TriggL
            ! Moving to above wanted position (dx, dy) from Peg00 and tool
rotated rotz degrees clockwise
            MoveL RelTool(Peg00, dx, -dy, -50, \Rz:= -rotz), vFast,
      z10, TatemTool1\WObj:=wobjTestBoard;
            ! the signal is turned on tAdelay second before (=above) the
target point
            TriggL RelTool(Peg00, dx, -dy,  0, \Rz:= -rotz),  vSlow,
PGunOn, fine, TatemTool1\WObj:=wobjTestBoard;
            WaitTime tWait;   ! should be the minimum time to wait
            ! move up again
            MoveL RelTool(Peg00, dx, -dy, -50, \Rz:= -rotz), vFast,
      z10, TatemTool1\WObj:=wobjTestBoard;
      ENDIF
      ENDPROC

      ! The last two functions for using trigger to activate and and
interrupt to deactivate tool
      PROC initTatemTool()
      ! just return IF doSlow
      IF NOT doSlow THEN
            ! initialize the TATEM tool, for using trigger and interrupt
            ! Connect the triggdata variable PGunOn to the DO signal
AirValve
            ! and set the startup time of the tool as tA_dalay seconds
before reaching the point
```

```
            ! the last argument, 1, is the value to assign to the DO signal when triggered.
            TriggIO PGunOn, tAdelay\Time \DOp:=AirValve, 1;
            IDelete igun_on;
            CONNECT igun_on WITH resetSignal;
            ISignalDO AirValve, 1, igun_on;
      ENDIF
      ENDPROC
```