CodeSys modeling of an industrial cell, and linear drive control with PLC & ATmega2560 microcontrollers for testing and improvement purposes

T Pajkos, T I Erdei, G Husi

Department of Mechatronics Engineering, University of Debrecen, Faculty of Engineering, Debrecen, Hungary

tiborc97@gmail.com

Abstract. In this document, the upgrades of the robotics lab at the University of Debrecen, faculty of engineering will be detailed. As the first part of the task, the conversion and upgrade of the KUKA KR5's cube holder were done. As a result, the cube holder (feeder) can count the added and removed cubes, while also displays their current amount in the holder. The completion of the other task enriched the existing robot cell with the addition of a liner drive powered system, complete with a test panel, which can even be used for educational purposes in the future. And lastly, as a summary, the simulation of a manufacturing process in the robot cell was also completed.

1. Introduction

The upgrades and research tasks were completed in the Building-mechatronics Research Centre, robotics lab of University of Debrecen Faculty of Engineering [1]. The robotics lab, complete with an industrial robot cell, is able to satisfy all challenges set by the upcoming industry 4.0 [2].By the end of the 20th century, mass production became a wide-spread phenomenon. As the demand for mass production grew, so did the need for automation. As a result, in 1968 the first PLC was created. This new controller had to satisfy the following criteria [3]: to be simple and freely reprogrammable, to have galvanically isolated I/O modules, to be safer than electromechanical solutions, and lastly, to have a better price/value ratio than pre-wired logic control circuits. In the case of switching for the manufacturing of a new item, it's easier to reprogram them, than to create new machines for the changed workflow. I chose the enhancement of the robotics lab's robot cell as my topic (and the subject of the completed tasks), because I find automation an interesting field.

2. The cube counter

My first task was the upgrade of the KUKA KR-5's cube holder and feeder. I will start by showing the steps and methods used during the conversion and upgrading process.

2.1. Used methods and equipment

The original cube holder was only capable of storing the cubes, and, due to the fact that the construction mainly uses hollow sections, the user could not see how many items are currently available in the storage. To solve this issue, my proposal was to design a counting unit with a connected display

The first task was to realize the sensing of the cubes. For this, reflective object sensing was used, as only one side of the holder was usable for placing sensors, and this seemed the most suitable option. These sensing points were created using optical cable. To process the sensors' digital signals, an Arduino Nano was chosen as a controller. With this choice, it was possible to not only process the incoming signals, but to control the display as well. As a numeric display, an 8 by 8 LED matrix screen [4] was chosen, which is based on a MAX7219 controller. These appliances required both 5 volts and 24-volt power; to solve the issue of different voltage levels, a voltage divider was built (complete with pulldown resistors), making the 24-volt sensors compatible with the Arduino Nano.

2.2. Assembly of the counter

A solution was needed to fasten every component to the holder, using the least amount of modification. During assembly, two holes were drilled on the top and bottom parts of the holder, to place the optical sensors. On the side of the holder, a DIN rail was secured to provide mounting points for the equipment. Two power supplies, and a 5-connector terminal block was placed here. Since the 24 volt sensors and the 5 volts Arduino was used at the same time, the two power supplies were connected to a common ground. The box, which houses the display, the voltage divider, and the Arduino, has been placed at the upper part of the cube holder. The "clear" button has been placed on the side of said box.

2.3. Creating the program

Two libraries have been used in the program's creation (not counting the Arduino IDE's built-in functions): EEPROM.h, and LedControl.h. During programming, my main goals were to keep track of the added and removed cubes' number, and the display and storage (archival) of said number. The storage of the number of cubes was necessary, to ensure that the values are remembered after the system returns from a non-powered state. In addition, a "clear" button was included in the program.

In the first half of the program, the variables and their starting value are initialized, the inputs and outputs are defined, the used functions are called, and the hexadecimal forms of the shown images are declared, to be used by the screen. This is followed by the "setup", where the screen is set up, the number of cubes is loaded from the EEPROM, and the lower sensor's value is checked, whether there's a cube in front of it, or not. This is necessary, because in the program, the changes of the sensor's output are used, and if there's a cube in front of the sensor in the starting state, it must not be interpreted as a cube being taken out, to avoid erroneous down-counting.

```
if ((vs_1  == 1) && (folyamat_1 == false)) {
  counter++;
  folyamat_1 = true;
  EEPROM.update(1, counter);
  Serial.println( counter );
  update_Display();
  delay(200);
} else if ((vs_1  == 0) && (folyamat_1 == true)) {
  folyamat_1 = false;
}
```

Figure 1. The upper sensor's program block

```
void printByte(byte character [])
{
  int i = 0;
  for (i = 0; i < 8; i++)
  {
    lc.setRow(0, i, character[i]);
  }
}
```

Figure 2. Displaying characters

On the contrary, the "loop" part of the program repeats from beginning to end every time it's fully executed. This is where the current states of the sensors are read, and where the up and down counting happens, with the usage of the assigned variable. If the number of cubes changes, the screen and the value in the EEPROM are both updated. The "clear" button's state is also observed in this portion. Pressing this button causes the number of cubes to return to zero.

The pictures on the LED-matrix screen are achieved by multiplexing. During this process, every row and its matching columns are switched on only for a short time. The numbers used in this case were created in a program called "Pixel to Matrix"[5]. The process of this is as follows: first, the desired shapes are created in an eight by eight matrix, then, after pressing the "Generate" button, the program provides the necessary binary, and hexadecimal form of said characters. In this chase the hexadecimal form was chosen, because it takes up less space, and improves the code's readability. After creating the

code to be displayed, a variable was created for it, to be able to point back to it every time there's a need to display it. In the program, a case structure was used to assign each number's picture to the number itself. After the code is assigned, the function draws it on the display, row by row.

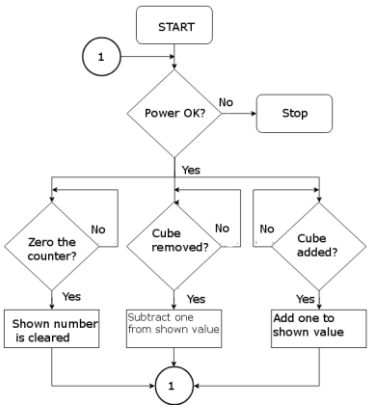

Figure 3. Cube counter's flowchart

Figure 4. The finished cube counter

3. Industrial linear drives

3.1. Designing the process

In my other task, I wished to use linear drives. The idea was to create a process using linear drives, which would cooperate with a KUKA KR-5 robot, by moving cubes in a specific way, after the industrial robot placed one on it. After working out a draft of the task its details had to be worked on. First, the KUKA KR-5 places a cube on one of the sliders residing on the linear drive, which will be sensed by a reflective optical sensor. Following that, the slider starts moving forward its other end position (limit), and after reaching it, another drive would activate, and push the cube into a box. This simulates the workpiece leaving the cell. During this task, I have used two ISEL LEZ-1 drive [6], which are driven by an MS045 HT motor [7]. The chosen optical sensors are Schneider Electric XUB0BKSNM12T [8].

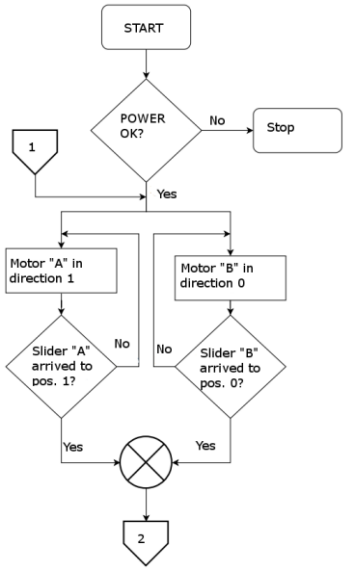

Figure 5. Linear drives' flowchart (1/2)

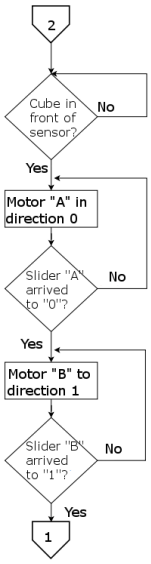

Figure 6. Linear drives' flowchart (2/2)

3.2. Programming the drives in CodeSys

As the next step of the task, I have created the model of the unit in CodeSys, and later simulated the entire process within CodeSys. First, options were created for both manual and automatic execution. A

TON timer was used in series with a TOF timer. This works in a way, that a button, which activates a bell, must be held for 3 seconds, and only after this can the program start. After releasing the "bell" button, seven seconds are given to start the process itself.

In automatic mode, the process was realized by using a form of the sequential method, in which a cycle consisting of steps was created, which are executed one after another. This solution is relatively easy to read, and making the modification of the existing program relatively simple. The content of the steps themselves also follows a modular, "5 ladder step" pattern, these 5 are: starting, safety, command, complete, and error. Commands are given in these steps, and their successful completion is later verified.

3.3. CodeSys simulation, the motor controller unit

As the first step of the modeling, the three-dimensional models of the drives had to be found, which were re-colored to improve their visibility, and later imported to CodeSys and assigned an ID in there, so that it's possible later to reference them during programming the animation. For the buttons, the build in CodeSys resources was used, or customized to suit the needs. To achieve the visual movements on the HMI, it was necessary to re-create the virtual linear drives, limit switches, and sensors. The position changes and their observation were achieved by a code segment, which detects which direction the motor gets the signal from at a given moment. When this detection and reading is complete, "Lin A pos" variable is checked and verified. If the limit is not yet reached, or in this particular case, the limit switches are not yet activated, then one unit is added, or deducted from the variable, depending on the movement's current direction. The limit switches function was realized by using a case structure.

```
IF Leptetes.Lin_A_0 THEN
    IF Lin_A_pos > 0 THEN
        Lin_A_pos := Lin_A_pos - 1;
    END_IF
END_IF

IF Leptetes.Lin_A_1 THEN
    IF Lin_A_pos < 1019 THEN
        Lin_A_pos := Lin_A_pos + 1;
    END_IF
END_IF

CASE Lin_A_pos OF
       0: GVL.Vegallas_Lin_A_0 := TRUE;
       1: GVL.Vegallas_Lin_A_0 := FALSE;
    1018: GVL.Vegallas_Lin_A_1 := FALSE;
    1019: GVL.Vegallas_Lin_A_1 := TRUE;
END_CASE
```

Figure 7. Programming of the virtual linear drive

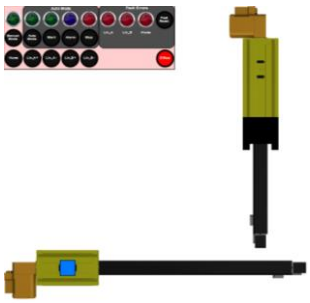

Figure 8. Test HMI

In the making of the motor controller unit, two Polulu A4988 bipolar stepper motor controllers were used. An Arduino Nano serves as a controller for these components. In addition, two capacitors were added to smoothen out the power spikes caused by the motors. For testing purposes, a test panel was created, which is compatible with an Arduino Mega, and with the help of an external program, can be programmed in ladder diagram. The test panel has eight inputs, and eight outputs, which are galvanically separated using optical couplers. To visualise the working channels, LEDs were added. Using the completed test panel, and the "PLC Ladder Simulator Pro" android application, the PLC program operating the linear drives was created. After uploading the necessary additional program, the assembled unit could be successfully tested.

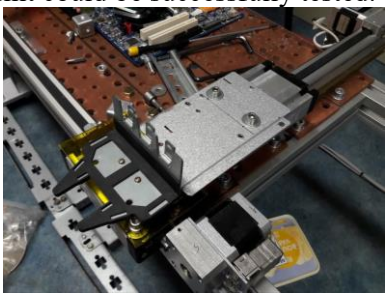

Figure 9. The completed tool

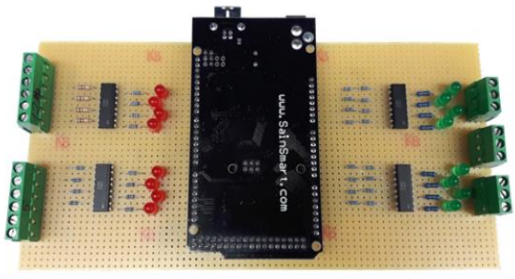

Figure 10. Completed test panel

3.4. PLC and synchronized control of the linear drives

After making certain that that both the motor controller unit and the ladder diagram program work without error, it was deemed important to operate the linear drives using the PLC found in the robotics lab. The already designed and tested program was implemented in the Unity Pro XL software.

During operation, the system first returns to its starting position. Then, it checks the optical sensor's signal, and when it senses an object, the process starts. When that happens, the drive with the cube starts moving towards its other end position. When it reaches the limit switch, the other drive perpendicular to it starts moving, and eventually pushes off the cube, into the box. When this step is complete, the system returns to its starting position again. As the last task, the KUKA KR-5 industrial robot was integrated into this process, utilising the finished linear drives' program.

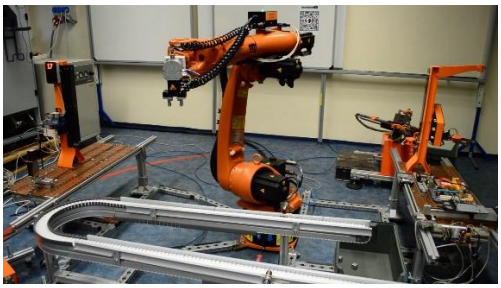

Figure 11. The completed robot cell

4. Conclusion

During the completion of these tasks, I have successfully applied my previously learned knowledge. As such, I can declare that not only I have acquired the necessary theoretical knowledge, but I only gained valuable practical skills as well. In my time spent at the robotics lab, I became acquainted with industrial devices and components, and I was able to use them in my project.

Acknowledgment

I would like to express my thanks to Erdei Timotei István, for his support during my work on the task. In addition, I'd also like to thank Dr. habil. Husi Géza, for making it possible for me to use the tools and devices found in the robotics lab.

References

[1] Erdei T I, Molnár Zs and Husi G 2016 *Robot visual and virtual control technology In industrial environment* WoS publication, International Symposium on Small-Scale Intelligent Manufacturing Systems, Narvik, NORWAY- IEEE

[2] Erdei T I, Molnár Zs, Obinna N C and Husi G 2017 *Cyber-physical systems in mechatronic research centre* Volume 126 MATEC Web Conf.

[3] Kulcsár B 2012 *Control and automation in material handling* (*Az anyagmozgatás irányítás- és automatizálástechnikája)* (Typotex Press)

[4] „*MAX7219*," [Online]. Available: https://datasheets.maximintegrated.com/en/ds/MAX7219-MAX7221.pdf [Accessed: 25. 11. 2018.]

[5] „*Pixel To Matrix*," [Online]. Available: http://educ8s.tv/arduino-8x8-led-matrix-tutorial/. [Accessed: 25. 11. 2018.]

[6] *ISEL* [Online]. Available: http://www.isel.hu/index.php/klub-7/piac/1886/mechanika/bordasszijas-linearis-egyseg/lez1-bordasszijas-linearis-egyseg.html [Accessed: 25. 11. 2018.]

[7] *ISEL* MS 045 HT [Online]. Available: https://www.isel.com/hu/dsh-s.html [Accessed: 25. 11. 2018.]

[8] *OsiSense* [Online]. Available: https://www.schneider-electric.com/en/product/XUB0BKSNM12T/photo-electric-sensor---xub---emitter---12..24vdc---m12/ [Accessed: 25. 11. 2018.]