

**IMAGE BASED SERVO CONTROL SYSTEM  
USING PLC AND SCADA**

**2017**  
**M. Sc. Thesis**  
**ELECTRICAL & ELECTRONICS ENGINEERING**

**ABDULKAREM MOFTAH AMAR MOHAMED**

**IMAGE BASED SERVO CONTROL SYSTEM USING PLC AND SCADA**

**A THESIS SUBMITTED TO  
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES OF  
KARABUK UNIVERSITY**

**BY**

**ABDULKAREM MOFTAH AMAR MOHAMED**

**IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR  
THE DEGREE OF MASTER OF SCIENCE IN  
DEPARTMENT OF  
ELECTRICAL & ELECTRONICS ENGINEERING**

**April 2017**

I certify that in my opinion the thesis submitted by Abdulkarem Moftah Amar MOHAMED titled "IMAGE BASED SERVO CONTROL SYSTEM USING PLC AND SCADA" is fully adequate in scope and in quality as a thesis for the degree of Master of Science.

Assist. Prof. Dr. Hüseyin ALTINKAYA

Thesis Advisor, Department of Electrical & Electronics Engineering



This thesis is accepted by the examining committee with a unanimous vote in the Department of Electrical Electronics Engineering as a master thesis. April 18, 2017

Examining Committee Members (Institutions)

Signature

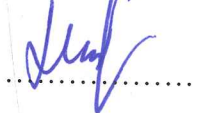
Chairman: Assoc. Prof. Dr. Ziyadulla YUSUPOV (KBU)



Member : Assist. Prof. Dr. Burhan BARAKLI (SAU)



Member : Assist. Prof. Dr. Hüseyin ALTINKAYA (KBU)



..... / ..... / 2017

The degree of Master of Science by the thesis submitted is approved by the Administrative Board of the Graduate School of Natural and Applied Sciences, Karabük University.

Prof. Dr. Nevin AYTEMİZ

Head of Graduate School of Natural and Applied Sciences





*“I declare that all the information within this thesis has been gathered and presented in accordance with academic regulations and ethical principles and I have according to the requirements of these regulations and principles cited all those which do not originate in this work as well.”*

Abdulkarem Mofteh Amar MOHAMED

## **ABSTRACT**

**M. Sc. Thesis**

### **IMAGE BASED SERVO CONTROL SYSTEM USING PLC AND SCADA**

**Abdulkarem Moftah Amar MOHAMED**

**Karabük University**

**Graduate School of Natural and Applied Sciences**

**The Department of Electrical & Electronics Engineering**

**Thesis Advisor:**

**Assist. Prof. Dr. Hüseyin ALTINKAYA**

**April 2017, 119 pages**

Nowadays, automation and robotics are rapidly advancing. In recent years, image processing and state analysis have become an important part of industrial automation processes. In this thesis, the sorting of plastic bottle caps according to their colors and geometric shapes has been carried out on image processing basis. The prototype built for this aim consists of a conveyor belt, a camera, an Arduino card, a robotic arm and a PLC. System control is provided by PLC (Programmable Logic Controller), SCADA (Supervisory Control and Data Acquisition) and Arduino card. By using image processing method on the prototype, the robustness and colors of the caps are determined. The C# programming language has been used for this aim. The caps are taken by the robotic arm and stacked according to their geometric shapes and colors. The system can be monitored and controlled from the SCADA screen. The prototype can be used as an R&D tool for such industrial automation processes. In addition to that it can also be used as a training material in schools.

**Key Words** : Image processing, robotic arm, PLC, SCADA, servo systems,  
automation.

**Science Code** : 905.1.096



## ÖZET

**Yüksek Lisans Tezi**

### **PLC VE SCADA İLE GÖRÜNTÜ TABANLI SERVO KONTROL SİSTEMİ**

**Abdulkarem Moftah Amar MOHAMED**

**Karabük Üniversitesi**

**Fen Bilimleri Enstitüsü**

**Elektrik-Elektronik Mühendisliği Anabilim Dalı**

**Tez Danışmanı:**

**Yrd. Doç. Dr. Hüseyin ALTINKAYA**

**Nisan 2017, 119 sayfa**

Günümüzde otomasyon ve robot teknolojileri hızlı bir şekilde ilerlemektedir. Son yıllarda görüntü işleme ve durum analizi, endüstriyel otomasyon proseslerinin önemli bir parçası olmaya başlamıştır. Bu tezde pet şişe kapaklarının renklerine ve geometrik şekillerine göre tasnifi görüntü işleme tabanlı olarak gerçekleştirilmiştir. Bunun için yapılan prototip, bir konveyör bant, bir kamera, bir Arduino kart, bir robot kol ve bir PLC'den oluşmaktadır. Sistemin kontrolü PLC (Programmable Logic Controller), SCADA (Supervisory Control and Data Acquisition) ve Arduino kart ile sağlanmaktadır. Prototip üzerinde görüntü işleme metoduyla kapakların sağlamlık ve renk tespiti yapılmaktadır. Bunun için C# programlama dili kullanılmıştır. Kapaklar robot kol ile alınarak geometrik şekillerine ve renklerine göre istiflenmektedir. SCADA ekranından system izlenebilmekte ve kontrol edilebilmektedir. Prototip, bu tür endüstriyel otomasyon prosesleri için bir Ar-Ge aracı olarak kullanılabilirliği gibi okullarda bir eğitim metaryeli olarak da kullanılabilir.

**Anahtar Kelimeler :** Görüntü işleme, robot kol, PLC, SCADA, servo sistemler,  
otomasyon.

**Bilim Kodu** : 905.1.096





## **ACKNOWLEDGMENT**

First of all, I would like to give thanks to my advisor, Assist. Prof. Dr. Hüseyin ALTINKAYA, for his great interest and assistance in preparation of this thesis.

I would like to thank my parents, for their never-ending encouragement, support, and unwavering love, which makes my dream come true. I appreciate my wife and children for their love and support, and for their patience.

This work supported by Research Fund of Karabuk University. Project Number: KBU-BAP-16/1-YL-172.

## CONTENTS

	<b><u>Page</u></b>
APPROVAL.....	ii
ABSTRACT.....	iv
ÖZET.....	vi
ACKNOWLEDGMENT.....	viii
CONTENTS.....	ix
LIST OF FIGURES.....	xiii
LIST OF TABLES.....	xv
SYMBOLS AND ABBREVIATIONS INDEX.....	xvi
CHAPTER 1.....	1
INTRODUCTION.....	1
CHAPTER 2.....	5
IMAGE PROCESSING.....	5
2.1. BASIC CONCEPTS REGARDING IMAGE PROCESSING.....	5
2.1.1. Sight, Appearance and Image.....	5
2.1.2. Analog Image.....	6
2.1.3. Digital Image.....	6
2.1.4. Pixel.....	6
2.1.5. Light.....	6
2.2. DIGITAL IMAGE PROCESSING.....	6
2.2.1. Knowledge Base.....	7
2.2.2. Acquisition.....	7
2.2.3. Preprocessing.....	7
2.2.4. Segmentation.....	8
2.2.5. Inference.....	8
2.2.6. Interpretation.....	8
2.3. PROCESS ON DIGITAL IMAGE.....	9

	<u>Page</u>
2.3.1. Point Processes .....	9
2.3.2. Local Processes.....	9
2.3.3. Local Processes.....	9
2.3.4. Digitization of Analog Image .....	10
2.3.5. Image Sampling .....	10
2.3.6. Image Quantization.....	11
2.3.7. Gray Image .....	12
2.3.8. Color Image .....	13
2.3.9. Converting Color Picture Into Gray Scale.....	14
2.3.10. Resolution.....	15
2.3.11. Area Sensitivity .....	15
2.3.12. Brightness Sensitivity .....	15
2.3.13. Demonstration of 2-Dimensional Digital Images as a Matrix.....	16
2.4. IMAGE PROCESSING AND ACQUISITION .....	17
2.5. MODEL FOR CREATING AN IMAGE.....	19
2.6. IMAGE INTERPOLATION .....	21
2.7. PRINCIPLES OF SPATIAL FILTERING .....	22
CHAPTER 3 .....	25
ROBOTIC SYSTEMS .....	25
3.1. DEFINITIONS RELATED TO ROBOTIC SYSTEM .....	26
3.1.1. Position .....	26
3.1.2. Orientation .....	27
3.1.3. Resolution.....	29
3.1.4. Sensitivity .....	29
3.1.5. Repeatability .....	29
3.1.6. Degree of Freedom .....	29
3.1.7. Manipulator.....	30
3.1.8. Servo Motor .....	30
3.1.9. Working Volume .....	30

	<u>Page</u>
3.1.10. Workspace .....	30
3.2. CLASSIFICATION OF ROBOTS.....	30
3.3. ROBOTS ACCORDING TO THEIR DEGREE OF FREEDOM .....	32
3.3.1. Cartesian (PPP) Robot.....	32
3.3.2. Cylindrical (RPP) Robot.....	33
3.3.3. SCARA (RPP) Robot .....	34
3.3.4. Spherical (RRP) Robot .....	34
3.3.5. Rotary (RRR) Robot.....	35
3.3.6. Articulated Robot.....	35
3.4. ROBOT KINEMATICS .....	36
3.4.1. Forward Kinematics.....	38
3.4.1.1. Denavit-Hartenberg Method .....	38
3.4.1.2. Placing Coordinate Systems on Joints .....	41
3.4.1.3. Joint Transformation Matrix .....	43
3.4.1.4. General Rules Applied to Define Robot Kinematics.....	44
3.4.2. Inverse Kinematics of Robots.....	46
3.4.2.1. Structure of Inverse Kinematics.....	46
3.4.2.2. Inverse Kinematics Solution Methods .....	49
CHAPTER 4 .....	53
PLC AND SCADA .....	53
4.1. PLC.....	54
4.1.1. Operation and Function of PLC.....	54
4.1.2. PLC Components.....	55
4.1.2.1. Input Unit .....	55
4.1.2.2. Output Unit .....	56
4.1.2.3. CPU (Central Process Unit) .....	56
4.1.2.4. Programming Unit .....	56
4.1.2.5. Power Supply .....	57
4.1.2.6. Communication Modules.....	57
4.1.3. Programming Software.....	57

	<u>Page</u>
4.2. SCADA .....	58
4.2.1. Control Units of the SCADA System .....	59
4.2.1.1. MTU (Master Terminal Unit) .....	59
4.2.1.2. RTU (Remoto Terminal Unit) .....	60
4.2.2. Network Topology.....	60
 CHAPTER 5 .....	 62
IMAGE BASED CONVEYER BELT APPLICATION .....	62
5.1. MATERIALS USED IN THE SYTEM .....	62
5.1.1. Webcam .....	62
5.1.2. Bunker.....	62
5.1.3. Arduino Control Card .....	63
5.1.4. Sensors .....	63
5.1.5. Robotic Arm .....	64
5.1.6. Conveyor Belt.....	64
5.1.7. 1215C DC/DC/DC PLC .....	65
5.2. PROTOTYPE IMPLEMENTATION .....	66
 CHAPTER 6 .....	 75
CONCLUSION .....	75
REFERENCES.....	76
 APPENDIX A. C# CODES FOR IMAGE PROCESSING .....	 79
APPENDIX B. ARDUINO CODES FOR ROBOTIC ARM CONTROL .....	100
APPENDIX C. PLC LADDER DIAGRAM.....	111
 RESUME .....	 119

## LIST OF FIGURES

	<u>Page</u>
Figure 1.1. Industrial automation pyramid.....	2
Figure 2.1. The basic steps of digital image processing.....	7
Figure 2.2. Demonstration of operations on digital image.....	10
Figure 2.3. Digitization of analog image by taking proper samples. ....	11
Figure 2.4. Basic structure of a 2-D digital image with the size of N x M .....	12
Figure 2.5. The representation of 256 different gray levels on a grid of 16x16 .....	13
Figure 2.6. Demonstration of 3 bands in N x M dimension that constitute the RGB image .....	14
Figure 2.7. Display of 2B digital images as a matrix .....	16
Figure 2.8. Three basic sensors used for converting illumination to digital images .....	18
Figure 2.9. An example of the digital image acquisition process .....	19
Figure 2.10. The mechanics of spatial filtering.....	24
Figure 3.1. Defining P point in respect to {A} coordinate system.....	27
Figure 3.2. Defining P point in respect to both {A} and {B} coordinate systems.....	27
Figure 3.3. Defining the orientation of a body with respect to the reference coordinate.....	28
Figure 3.4. Kuka industrial robot with serial arm structure .....	31
Figure 3.5. ABB industrial robot with parallel arm structure .....	31
Figure 3.6. Industrial robot with cartesian structure .....	33
Figure 3.7. Cylindrical robot (Seiko RT3300).....	33
Figure 3.8. SCARA robot.....	34
Figure 3.9. Spherical robot.....	35
Figure 3.10. Revolute robot (Motoman SK16) .....	35
Figure 3.11. Articulated robot .....	36
Figure 3.12. Joint structures performing translation and rotation movements.....	37
Figure 3.13. Robots consisting of prismatic and revolute joints .....	37
Figure 3.14. Layout of (I-1), i links, and (i-1), i and (i+1) axes.....	39
Figure 3.15. Placing coordinate systems on (i-1) and i axes.....	39

	<u>Page</u>
Figure 3.16. $a_{i-1}$ link length lying on the direction of $X_{i-1}$ between $Z_{i-1}$ and $Z_i$ .....	40
Figure 3.17. $d_i$ link deviation on $Z_i$ between $X_{i-1}$ ile $X_i$ .....	40
Figure 3.18. $\alpha_{i-1}$ link angle between $Z_{i-1}$ ile $Z_i$ rotation axes .....	41
Figure 3.19. $\theta_i$ joint angle between $a_{i-1}$ and $a_i$ links .....	41
Figure 3.20. Calculation of $a_{i-1}$ , $\alpha_{i-1}$ , $d_i$ and $\theta_i$ joint variables .....	43
Figure 3.21. Schematic illustration of forward and reverse kinematics problem .....	46
Figure 3.22. The relation between mathematical statement and physical solution.....	47
Figure 3.23. Impact of the joint structure on the number of solutions .....	47
Figure 3.24. Four different solutions for PUMA-560 robot.....	48
Figure 5.1. HD web-cam .....	62
Figure 5.2. Bunker.....	63
Figure 5.3. Arduino nano ATmega328 .....	63
Figure 5.4. MZ80 object detection sensor.....	64
Figure 5.5. Robotic arm.....	64
Figure 5.6. Conveyor belt.....	65
Figure 5.7. PLC .....	65
Figure 5.8. Model study .....	66
Figure 5.9. Flow card .....	67
Figure 5.10. Block diagram of the system .....	69
Figure 5.11. RGB hue cubes .....	70
Figure 5.12. Image processing interface program - determination of geometric shape .....	70
Figure 5.13. Image processing interface program - determination of the color blue.....	71
Figure 5.14. Image processing interface program - determination of the color red .....	71
Figure 5.15. System overview .....	72
Figure 5.16. SCADA screen 1-initial state.....	73
Figure 5.17. SCADA screen 1-after placed caps .....	74
Figure 5.16. SCADA screen 2.....	74

## LIST OF TABLES

	<b><u>Page</u></b>
Table 2.1. RGB values of some basic colors .....	13
Table 3.1. Determination of D-H variables.....	45





## SYMBOLS AND ABBREVIATIONS INDEX

### SYMBOLS

$a_{i-1}$  : link length

$\alpha_{i-1}$  : link angle

$d_i$  : joint offset

$\theta_i$  : joint angle

${}^i_0T$  : D-H transformation matrix

$l$  : intensity of a monochromatic image

$g(x, y)$ : linear spatial filtering

$v(x, y)$ : linear interpolation

### ABBREVIATIONS

*DoF* : Degree of Freedom

*PLC* : Programmable Logic Controller

*SC ADA* : Supervisory Control and Data Acquisition

## **CHAPTER 1**

### **INTRODUCTION**

The place and importance of industrial automation in our daily life is constantly growing. Nowadays, it is not even possible to imagine operating neither small/simple facilities/systems nor big/complex ones without automation. The objective of industrial automation systems is to minimize the human factor and to produce the highest number of products at the highest quality level in the least possible time period. Furthermore, enabling intervention to possible breakdowns in the shortest time period by following-up and monitoring the system on a 24/7 basis is another objective.

An Industrial Automation System consists of numerous elements that perform a variety of functions related to Instrumentation, Control, Supervision and Operations Management related to the industrial process. These elements may also communicate with one another to exchange information necessary for overall coordination and optimized operation of the plant/factory/process.

Industrial automation systems are very complex having large number of devices with confluence of technologies working in synchronization. In order to know the performance of the system we need to understand the various parts of the system. Industrial automation systems are organized hierarchically as shown in Figure 1.1. [1,2].

Classically, the Industrial Automation consists of five main levels. The prototype built and the software conducted include studies belonging to the first three levels of industrial automation.

The robotic arm and image processing which have an important place in the industrial automation systems constitute the two main components of this thesis.

The primary purpose of the prototype is to provide a training material for students and hence, two different types of controllers (Arduino and PLC) are used. Arduino controller, which is very popular among students nowadays, is used together with PLC to ensure that both logic controllers are seen on the same prototype at the same time. Furthermore, the process of establishing connection/communication between different devices which is very frequent in industrial automation systems is also displayed.

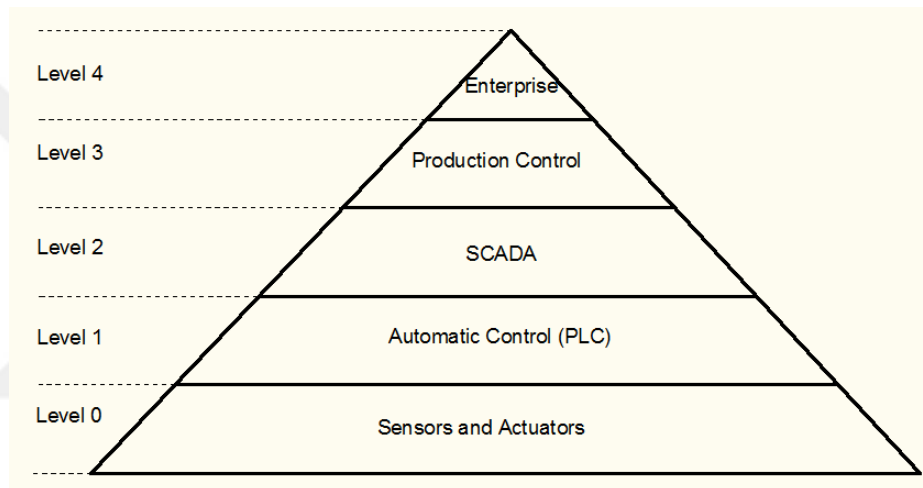


Figure 1.1. Industrial automation pyramid.

Some of the studies conducted on the image based servo control systems are presented below.

For the standard mechanical systems, image based visual servo control strategy has been developed. Hamiltonian port has been used. An interactive matrix including both the changes in image plane and depth information has been used. The proposed approach has been applied to the two-level robot arm problem and the simulation results have been given. It has been demonstrated that the resulting control strategy played an important role in the system for providing the desired image properties based on the change of variables by momentum adaptation generalized with camera system [3].

For the visual servo control of a dynamic system, an image based strategy has been proposed. Dynamic modeling of semi-stationary flight capacities of unmanned aerial vehicles has also been included in the system. They have benefited from Lyapunov algorithm by using back stepping techniques for dynamic modeling. They have defined the image errors in terms of image properties obtained from the camera input. They have proved the local exponential stability of the system and introduced a closed cycle system for the prediction of the attraction basin [4]. The position and orientation of an industrial robot with six degrees of freedom while moving to the operation points has been provided without using any teaching method and by processing the data obtained from a camera according to robot kinematic model [5]. By using the 6 degrees of freedom arm and Kinect depth sensor of U MAY which is a humanoid robot project, visual servo control has been conducted in simulation environment. There are two approaches in camera robot integration mechanism. These define the positioning of the camera as attached to the robot arm (eye-in-hand) or in a position where it sees both the robot arm and target objects(eye-to-hand) [6]. An image based algorithm predicting the position and movement of different targets in real-time has been introduced. Visual servo robotic manipulators have been used in the system. An adaptive extended Kalman filter and a hybrid photogrammetry approach for real-time position and movement predictions and a visual closed cycle circuit for robotic manipulation have been developed. The feasibility, effectiveness and soundness of the experimental results have been shown [7].

In serial production lines where industrial robots are heavily used, the most important and prioritized issues are quality and low production cycle time. In order to increase the product quality and decrease the production cycle time various control and automation applications are developed in the production lines. Recently, within the scope of these applications, certain solutions are produced by using visual control technique [8-10]. In visual control technique, two different methods, i.e. model based or image based are usually used [10-13]. Different solutions are tried to be produced by incorporating visual based control into the control and automation applications conducted for on-band or independent take-and-leave operations which is one of the most intensively used applications in production lines [14,17]. In the industry and with regard to the reliability of the product, smart cameras are preferred for the

control systems which include image processing applications. Both in smart camera applications and a standard camera application, the data obtained from the camera should be transformed into a meaningful data format for the kinematic of the robot in a robotic application [15-17].

The thesis consists of six chapters. In the first chapter, importance of the industrial automation and the target of the thesis is explained and literature review is presented. In the second chapter, brief information on basic concepts and algorithms regarding image processing is given. In the third chapter, explanations regarding robots and the use of robotic arms, their types and robot kinematic take place. In the fourth chapter, brief information on PLC and SCADA is presented. In the fifth section, the conducted prototype practice is explained in detail. In the sixth and last chapter, the conclusion and recommendations are explained.

In this study, a prototype which controls the robustness and colors of plastic bottle caps has been developed. Robustness control is conducted according to the geometrical shape of caps. The caps which do not fit the ideal dimensions and geometric shape of a robust cap that is predefined and introduced to the system are defined as faulty (disordered). Color detection and robustness control are performed by the same camera. Instead of smart camera, a simple web cam has been used. The C# programming language has been used for image processing. Sorted caps are put into their respective boxes by the robotic arm using take-leave process. Instead of using a manufactured industrial robot, a robotic arm which was produced by us using a 3D printer has been used. The aim of the prototype which is a combination of camera - Arduino - Robot - PLC is both to provide a different approach for industrial applications and particularly to create an effective training material which will be used in industrial automation lessons.

## **CHAPTER 2**

### **IMAGE PROCESSING**

In this thesis, instead of giving unnecessary details, necessary information related with the scope of this study will be given on image processing which constitutes a crucial stage of the prototype that has been produced. Image Processing is used in numerous fields such as military, industry, underwater imaging, robotics, astronomy, physics, art, biomedical, geographic information systems, remote sensing, observation applications, animal husbandry, oil exploration, newspaper and photo industry, traffic, radar, medicine and security. In this context, various tools such as artificial neural networks, wavelet transform, directional filters, fuzzy logic, Markov random field filters and image transmission are used together with it. Image processing refers to digital image processing. Image processing which has a wide range of application areas is related to many disciplines. Today, almost all image processing operations are carried out by digital image processing techniques.

#### **2.1. BASIC CONCEPTS REGARDING IMAGE PROCESSING**

##### **2.1.1. Sight, Appearance and Image**

Perceiving objects by way of rays which hit their surface or pass through them is called sight and the content of the objects which can be perceived by this way is called appearance. A two-dimensional (2D) picture of the appearance obtained in any form is called image. Image can also be defined as a two-dimensional map of the three-dimensional (3D) appearance.

### **2.1.2. Analog Image**

If  $x$  and  $y$  values which constitute the function  $(f(x,y))$  in a certain max and min range in analog image continuously take changing real values (including decimal values), this image is an analog image. In an analog image, no matter how close we look at the picture (for instance by using a microscope) colors that make up the image are still there. Digital computers cannot process continuous functions/parameters. These functions should be digitized.

### **2.1.3. Digital Image**

It is the expression of continuous image which is represented as  $f(x, y)$  (analog image) in discrete examples and it is displayed as  $f[x, y]$ .

### **2.1.4. Pixel**

Each element  $f [x, y]$  in the form of a 2-dimensional array of a digital image is called a pixel or pel "picture element".

### **2.1.5. Light**

It is a kind of energy. There are various theories about light. It is proved that light is both substance (mass) and energy (wave). Light is the small wavelets that are emitted by the light source. These wavelets are in the form of radiation energy (electromagnetic radiation). It spreads in straight lines. The visible light is placed in a very small range in the electromagnetic spectrum (around 2%). We cannot see light waves which are beyond this range with bare eyes [18,21].

## **2.2. DIGITAL IMAGE PROCESSING**

Digital image processing deals with transforming an analog image into a digital form and processing this image by using digital computers for various purposes

(improvement, repairment, classification, compression, understanding and interpretation, etc.). The following figure shows its basic steps:

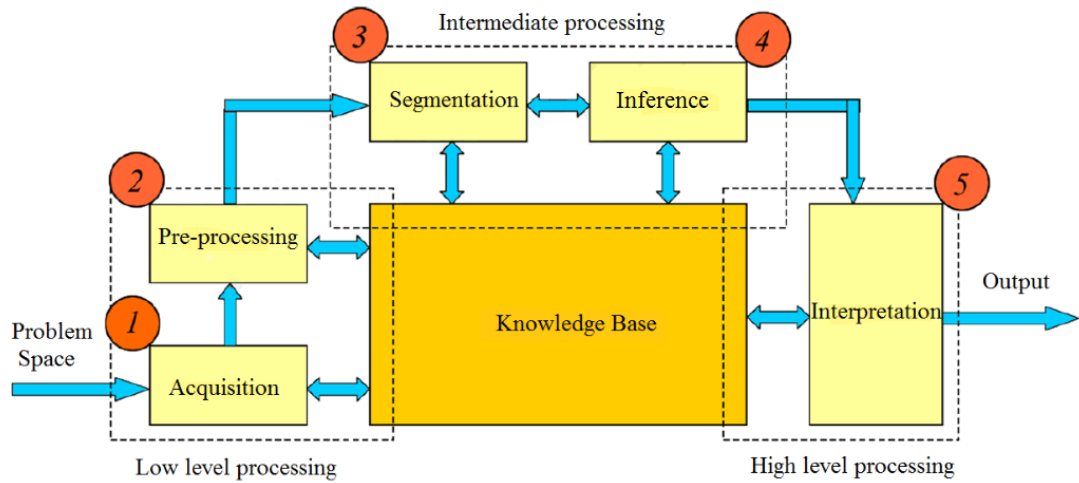


Figure 2.1. The basic steps of digital image processing.

### 2.2.1. Knowledge Base

A knowledge base is depicted, where knowledge concerning a problem domain is kept in a knowledge database and encoded into the image processing system. Interaction occurs with the knowledge base in the steps due to a perceived prior knowledge on what the result will be.

### 2.2.2. Acquisition

The first step in image processing is the acquisition of the digital image by a digital camera.

### 2.2.3. Preprocessing

The next step following the acquisition of digital picture is preprocessing. Pre-processing is the act of performing some prior processes to the image in order to obtain a more successful result before using the obtained digital image. These processes can basically be grouped under the following subtitles:



- image enhancement,
- image restoration, and
- image compression.

Image acquisition and preprocessing is called low level image processing.

#### **2.2.4. Segmentation**

After preprocessing, segmentation phase begins. Segmentation which is the hardest stage of image processing is the process of separating the object and background or any other relevant sections within the image which have different properties from each other in an image. By determining the borders and areas of an object in an image, segmentation produces raw information on the shape of that object. If we are interested in the shapes of objects, then segmentation provides us information on the edges, corners and borders of that object. On the other hand, if the main concern is the internal features of objects such as surface coverage, area, color, skeleton, then regional segmentation should be conducted. In order to solve complicated problems such as character or general pattern recognition, it may be necessary to use both segmentation methods (borders and areas) together.

#### **2.2.5. Inference**

Inference is foregrounding the raw information and details of interest obtained from the image. In other words, it is the separation of specific areas from the background and from each other.

#### **2.2.6. Interpretation**

At his stage which is within the high-level image processing group, various decision making mechanisms (such as artificial intelligence algorithms) are used to label and classify objects or regions removed from the background of the image.

## **2.3. PROCESSES ON DIGITAL IMAGE**

All the processes during the image processing phase are performed on the pixels that generate the image, that is, on the on the color information that these pixels possess. These processes can be categorized into three groups: point processes, local (regional) processes and global processes.

### **2.3.1. Point Processes**

These are the processes applied on one pixel of the input picture in order to create one pixel of the output picture. They are conducted by changing the pixel's own color information independently of its neighboring pixels. That is to say, the process on each pixel in the input image creates its counterpart pixel in the output image.

### **2.3.2. Local Processes**

Here, the color of the point which will form the output picture also depends on the color properties of its neighbors in the input picture. To which neighbors' colors it will depend will be determined by the size of the determined mask. By scrolling these masks over all the pixels in the image, the image is filtered. In this sense; sweeping blur in the image, cleaning noise, determining edge and area characteristics can be given as examples of local operations. In short, value of a pixel in the output image depends on the value of various pixels in the input image in this process.

### **2.3.3. Global Processes**

In global processes, color properties of all the pixels of the picture affect the pixel of the output picture which will be created. That is to say, value of a pixel in the output image depends on the values of all pixels in the input image in this process.

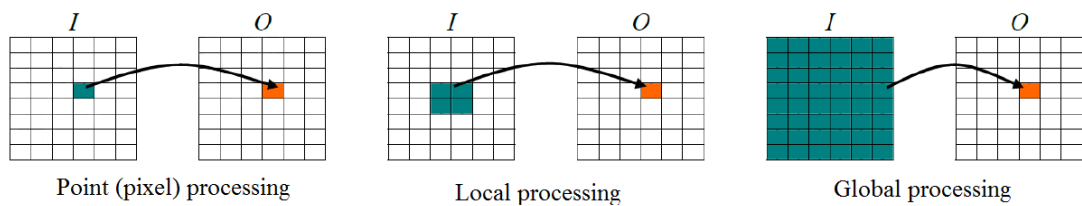


Figure 2.2. Demonstration of operations on digital image.

### 2.3.4. Digitization of Analog Image

In order to process them in the computer environment, the analog images should be digitized. First thing to do for digitization is sampling which is followed by quantization. In order to process the function in computer environment, it should be digitized both spatially and in amplitude (in color information). The digitization of the coordinates regarding the image function is called image sampling, and digitization of amplitude values is called quantization.

### 2.3.5. Image Sampling

A digital image can be generated over  $N$  samples along the  $x$ -axis and  $M$  samples along the  $y$ -axis at even intervals over a continuous image function. Thus, the transition from the continuous-time image function to the discrete-time image function takes place. An analogue image can be roughly expressed with a total of  $N \times M$  end-point sample values consisting of  $N$  samples horizontally and  $M$  samples vertically in 2-D discrete-time. In this process, the analogue image function is properly sampled. That is to say, proper sampling is made by taking samples from the analog image at equal intervals in both horizontal and vertical directions. The resulting digital (numerical) image is actually a matrix consisting of  $N$  rows and  $M$  columns. There is a loss of information in this process.

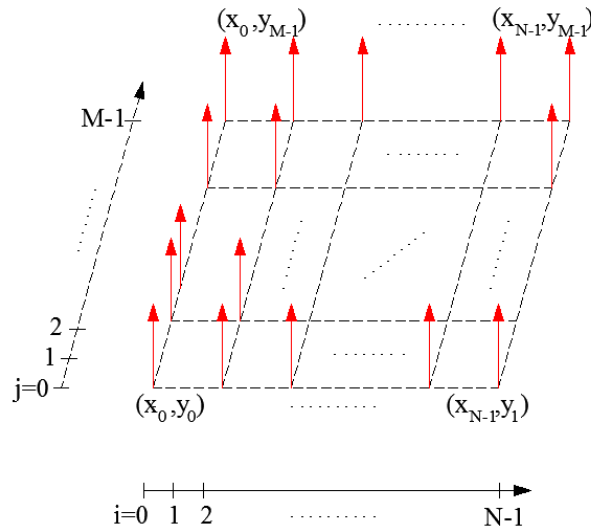


Figure 2.3. Digitization of analog image by taking proper samples [18].

### 2.3.6. Image Quantization

The positive integer value indicating the brightness intensity of each element (pixel) of the image is determined by quantization. The minimum and maximum amplitude values range of the image element is separated into steps and the element obtains the image value closest to the corresponding step value.

As a result of these two processes, a digital image that can be processed by the computers is obtained. One of the example devices that digitize an analog image by performing these two processes is a scanner. Images obtained from the browser in certain formats are digital and they can be processed with software on a computer. The digital image obtained after sampling and quantization is in a two-dimensional matrix structure with components which consist of positive integer values. Each element of the matrix representing the digital image is called a pixel. A pixel is the smallest element that creates a digital image, and the value of a pixel defines the magnitude of brightness of the corresponding image element. The positive integer value associated with brightness intensity is determined by quantization.

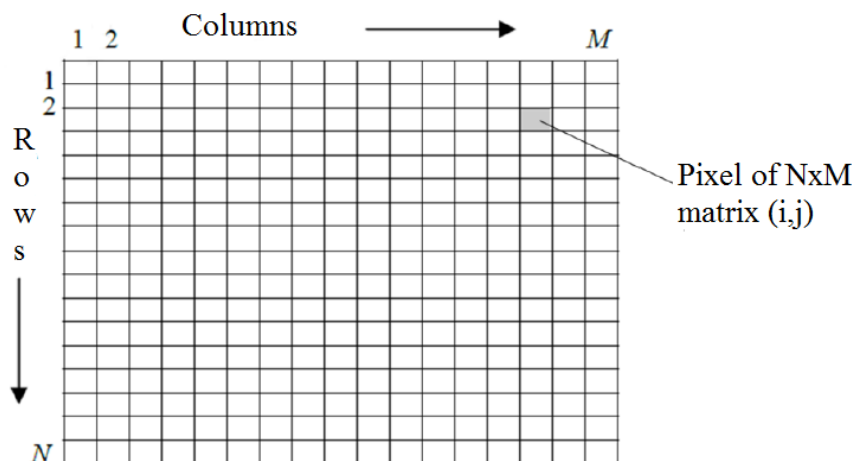


Figure 2.4. Basic structure of a 2-D digital image with the size of  $N \times M$ .

### 2.3.7. Gray Image

In the digitization process, the image size and the brightness value that each pixel can have must be specified. The brightness value of each pixel of the digital image is called gray levels. The gray level range is determined by the number of bits that the brightness value of each pixel is encoded.

Two colors exist at the borders of the gray level, black and white. The images encoded between these two are called gray scale (monochromatic) images. In practice, each most commonly used pixel is encoded with 8 bits. In such images, each pixel consists of  $2^8 = 256$  different gray tone contrast values (brightness level), and the gray value range is expressed as  $G = \{0, 1, 2, \dots, 255\}$ . As a rule; 0 gray level corresponds to black color, 255 gray level corresponds to white color, and gray levels between these values correspond to gray tones. The figure shows the representation of 256 different gray levels on a grid of  $N \times M = 16 \times 16$ .

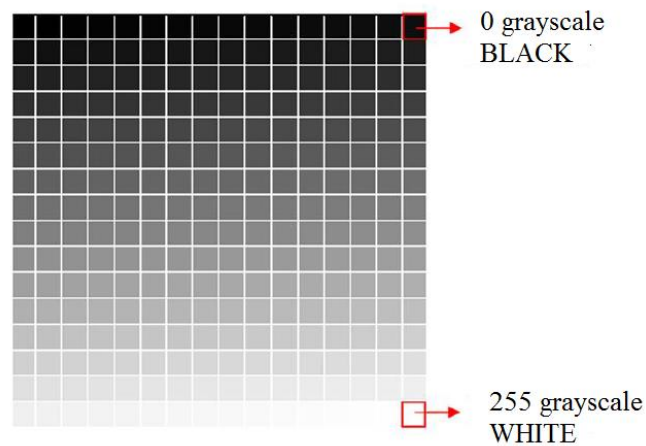


Figure 2.5. The representation of 256 different gray levels on a grid of  $16 \times 16$ .

### 2.3.8. Color Image

Color images consist of three gray-scale images of the same object encoded in R(Red), G(Green),and B(Blue), all superimposed on the screen. These three colors which constitute the color image are called band. Each pixel of the color images is displayed as a 24-bit datum on computer screens. That is to say, since each color will be encoded with 8 bits ( $2^2 \cdot 2^2 \cdot 2^2 \cdot 2^2 = 2^8 = 256$ ) three colors (RGB) will be encoded with  $3 \times 8 = 24$  bits. In this case, each pixel of the RGB images may have a different color value of  $2^8 \cdot 2^8 \cdot 2^8 = 2^{24} = 16.777.216$  (about 17 million), and the value range of the combination of these colors is shown as  $RGB = (0, 0, 0), \dots, (255, 255, 255)$ . Some sample colors and their values are given in the table below.

Table 2.1. RGB values of some basic colors.

Colors	R	G	B	Appearance
Red	255	0	0	
Green	0	255	0	
Blue	0	0	255	
White	255	255	255	
Black	0	0	0	
Light Grey	200	200	200	
Dark Grey	100	100	100	
Yellow	255	255	0	
Turquoise	0	255	255	
Magenta	255	0	255	

In terms of matrix, a natural colored 2 Dimensional RGB image consists of a combination of three matrices (grids) each with the size of (N x M) and it is represented as  $\{I(i, j, k) \mid i = 1, 2, \dots, N; j = 1, 2, \dots, M; k = 1, 2, 3\}$  in matrix demonstration. In general, each of these matrices represent the image with pixels having one of the 256 levels.

$I(i, j, 1); i = 1, 2, \dots, N, j = 1, 2, \dots, M \rightarrow$  The matrix related to the red band

$I(i, j, 2); i = 1, 2, \dots, N, j = 1, 2, \dots, M \rightarrow$  The matrix related to the green band

$I(i, j, 3); i = 1, 2, \dots, N, j = 1, 2, \dots, M \rightarrow$  The matrix related to the blue band

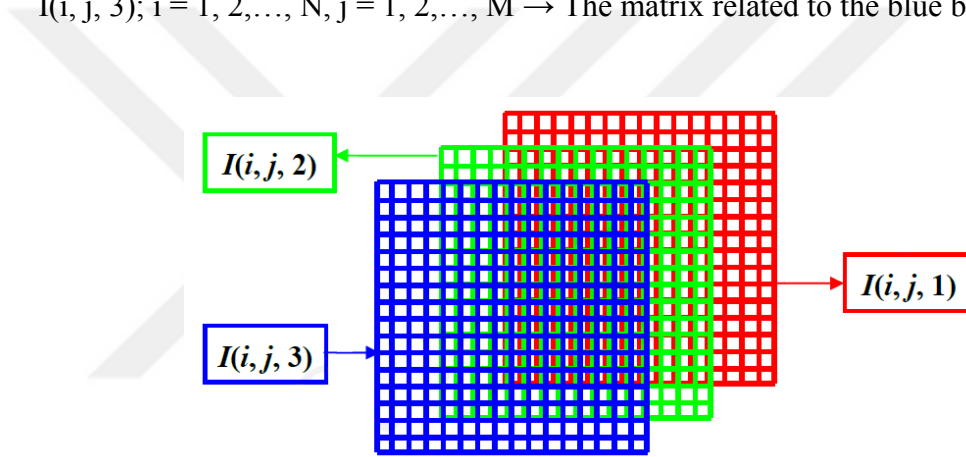


Figure 2.6. Demonstration of 3 bands in N x M dimension that constitute the RGB image.

### 2.3.9. Converting Color Picture Into Gray-Scale

Converting a color digital image into a gray-scaled image is nothing more than scaling gray-toned images that correspond to each color band specified in the RGB color model. In this sense, the scaling process conducted by sticking to the brightness values of the color image is given by the formula

$$Gray = 0.299xR + 0.587xG + 0.114xB \quad (2.1)$$

using the aforementioned demonstration. Gray-scaled equivalent of the color image is obtained by using this equation.

### **2.3.10. Resolution (Area Sensitivity and Color Brightness Sensitivity)**

The resolution of an image is the degree to which details within the image can be discerned. This concept includes both the Area resolution and the Brightness resolution. The Area resolution is the number of samples taken from the surface ( $N \times M$ ) and the Brightness resolution specifies the gray level counterpart of the brightness ( $m$ ) of the color on the pixel. The greater the increase in the values of these parameters, the closer the digitized image to its original. However, the memory area of the image increases quickly.

### **2.3.11. Area Sensitivity (Pixel resolution)**

Area sensitivity refers to the number of samples collected during horizontal/vertical scanning of the analogue image to obtain a digital image. As the number of pixels ( $N \times M$ ) forming the digital image increase, the image comes closer to the original (analog) image. On the other hand, if the number of pixels is reduced, then the spatial resolution of the image falls and the details in the image begin to disappear. This is called the checkerboard effect. This effect results in the creation of artificial squares in the image. 8-bit grayscale images with various area resolutions are given in the figure below. It can be seen that as the number of pixels decreases, the checkerboard effect increases.

### **2.3.12. Brightness Sensitivity (Gray-Level Color Resolution)**

The brightness sensitivity indicates the corresponding number of the brightness of each pixel on gray level scale. That is to say, while whitening colors from black to white in gray level scale, 256 steps (8 bits) are used. Instead of the gray color on this scale, equivalent of any color will show the brightness of that color.

As it has been previously mentioned, gray-scaled images which are widely used in practice have a full brightness value of 255 in the sense of gray level, and this value drops down up to white color. That is, the pixels of such images can have 256 different values in  $G = \{0, 1, 2, \dots, 255\}$  gray level range and as a result each pixel is



represented by  $m = 8$  bits. When the number of gray levels decreases, artificial linear lines start appearing in the image. The figure below shows gray-scaled images with the same area resolution ( $N \times M = 256 \times 256$ ) in 256 colors (8 bits), 128 colors (7 bits), 64 colors (6 bits), 32 colors (5 bits), 16 colors (4 bits), 8 colors (3 bits), 4 colors (2 bits) and 2 colors (1 bit).

When the images are analyzed, it can be seen that the difference between 8 bits and 5 bits are not fully grasped by the naked eye (if it was a color picture it would be more visible). However, decrease of the quality between 4 bits and 1 bit can be seen clearly and artificial boundaries are noticeable.

### 2.3.13. Demonstration of 2-Dimensional Digital Images as a Matrix

As it has been previously mentioned, a 2-D digital image with  $(N \times M)$  pixels can be thought of as a matrix consisting of  $N$  rows and  $M$  columns. The components of the matrix, each point where the rows and columns intersect -which represents the smallest part of the image, i.e. pixel- have positive integer value. For example, for images with 256 gray levels, this value consists of integers between 0 and 255.

The components of a matrix representing a digital image can never consist of negative and non-integer values. However, it is possible to encounter such results in the new image which is obtained after processing the image. In such cases, proper scaling and rounding processes on the matrix values must be performed to display the new image correctly on the computer screen.

$$A = \begin{array}{c} \xrightarrow{256 \text{ columns}} \\ \left[ \begin{array}{cccc} 0 & 1 & \dots & 255 \\ 0 & 1 & \dots & 255 \\ \vdots & \vdots & \dots & \vdots \\ 0 & 1 & \dots & 255 \end{array} \right] \\ \downarrow \begin{array}{l} 2 \\ 5 \\ 6 \\ \text{r} \\ \text{o} \\ \text{w} \\ \text{s} \end{array} \end{array}$$

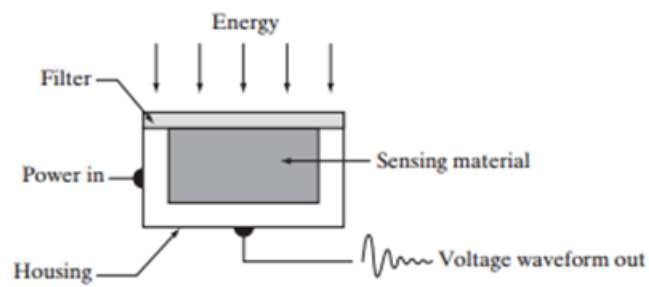
Figure 2.7. Display of 2B digital images as a matrix.

The basic terms related to image processing are defined and a general framework is given above. On the other hand, some of the topics of digital image processing, which is a very wide and complex issue, will be explained below limited to the scope of this thesis [18-21].

## **2.4. IMAGE PROCESSING AND ACQUISITION**

Most of the images we are interested in are produced by the combination of a "lighting" source and a "place", elements of which either reflect or absorb the energy coming from this source. We enclose the words lighting and place in order to emphasize the fact that they are much more general than the known situation of the illumination of a usual 3-D (three-dimensional) place.

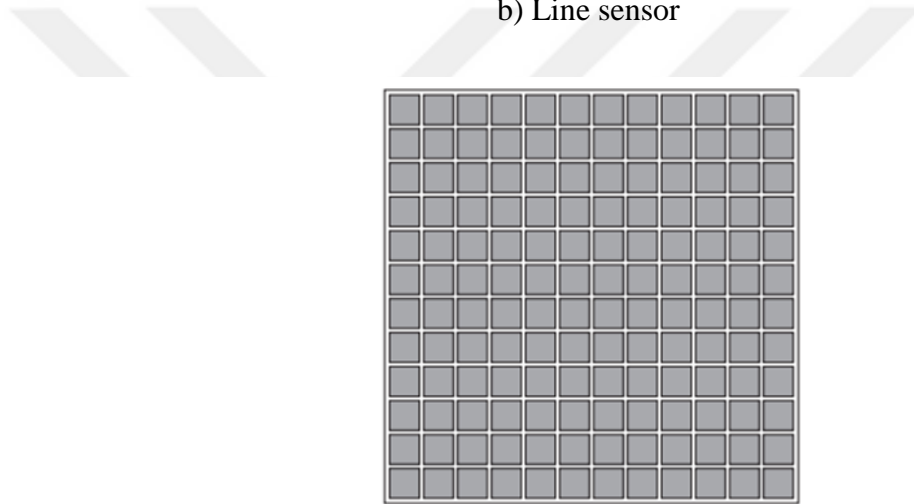
Figure 2.8 shows three basic sensor equipment which are used to convert the lighting energy into digital images. The idea is straightforward: Incoming energy is converted into a tension by a combination of a detector material which is sensitive to a certain type of energy which will be perceived and input electrical power. The waveform of the output voltage is the response of the sensors, and a numerical quantity is obtained by digitizing the response of each sensor.



a) Single imaging sensor



b) Line sensor



c) Array sensor

Figure 2.8. Three basic sensors used for converting illumination to digital images [18].

Figure 2.8 (c) shows discrete detectors arranged in the form of a 2-D array. This mechanism is used in digital cameras. A typical sensor for these cameras is a CCD array manufactured with wide sensing range features and packaged as arrays with 4000 x 4000 or more elements. The response of each sensor is directly proportional to the integral of the reflected light energy, and this property is used in astronomical and other applications that require low noise images. Noise reduction is accomplished by the sensor by picking up the input light signal for minutes or even hours. Since the sensor array in Figure 2.8 (c) is two-dimensional, obtaining a full

image by focusing the energy pattern to the surface of the array is the basic advantage of this sensor array.

The basic form in which array sensors are used is shown in Figure 2.9. This figure shows the energy coming from a lighting source reflected from a stage element (energy can also be transmitted through stage elements). The first function performed by the imaging system in Figure 2.9 is to collect the incoming energy and focus it on an image plane. If the illumination is light, the front of the imaging system is an optical lens which reflects the displayed scene to the lens' focal plane, as shown in Figure 2.9. At each sensor, the sensor array which is coincident with the focal plane produces output proportional to the integral of the light received. Digital and analog circuit systems scan these outputs and then convert them to analog signals which are digitized by another part of the imaging system. As shown schematically in Figure 2.9, the output is a digital image.

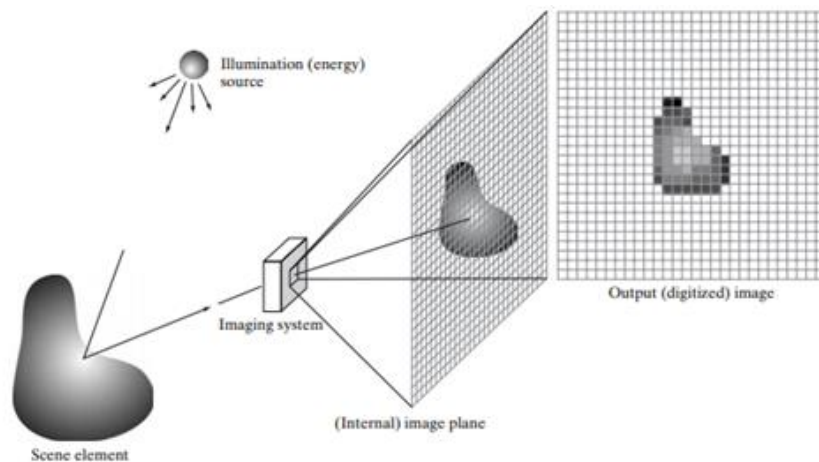


Figure 2.9. An example of the digital image acquisition process [18].

## 2.5. MODEL FOR CREATING AN IMAGE

We display images with two dimensional functions in the form of  $f(x,y)$ . In the spatial  $(x,y)$  coordinates, the value or amplitude of  $f$  is a positive numerical quantity, physical meaning of which is determined by the source of the image. When an image is created by a physical process, its intensity values are directly proportional to the energy emitted by the physical source (e.g. electromagnetic waves).

Hence,  $f(x,y)$  should be different than zero and finite; that is

$$0 < f(x,y) < \infty \quad (2.2)$$

The function  $f(x,y)$  can be characterized by two components. (1) the amount of the source illumination on the scene being displayed, (2) the amount of illumination reflected by the objects on the scene. Naturally, these are called the illumination and reflection components and denoted by  $i(x,y)$  and  $r(x,y)$ , respectively. The two functions combine in multiplication to form  $f(x,y)$ .

$$f(x,y) = i(x,y) r(x,y) \quad (2.3)$$

Here

$$0 < i(x,y) < \infty \quad (2.4)$$

and

$$0 < r(x,y) < 1 \quad (2.5)$$

Equation (2.5) shows that the reflection is limited to 0 (full absorption) and 1 (full reflection). It is determined by the natural lighting source of  $i(x,y)$ , and  $r(x,y)$ , the characteristics of the displayed objects.

Values included in equations (2.4) and (2.5) are theoretical boundaries. The following average numerical values represent some typical ranges of  $i(x,y)$  for visible light. On a clear day, the sun can produce over 90,000  $\text{lm/m}^2$  of illumination on the earth's surface. This value is drops to less than 10,000  $\text{lm/m}^2$  on a cloudy day. On an open evening, the full moon gives illumination of approximately 0.1  $\text{lm/m}^2$ . Typical lighting level of a commercial office is about 1,000  $\text{lm/m}^2$ . Similarly, the following are typical values of  $r(x,y)$ . 0.01 for black velvet, 0.65 for stainless steel, 0.80 for plain white wall paint, 0.90 for silver plated metal and 0.93 for snow.

Assuming that the intensity of a monochromatic image at any  $(x_0,y_0)$  coordinate is shown by

$$l = f(x_0, y_0) \quad (2.6)$$

it is obvious that  $L$  from (2.3) to (2.5) is between

$$L_{min} \leq l \leq L_{max} \quad (2.7)$$

range. In theory, the only condition on  $L_{min}$  is that it is positive, and finite in  $L_{max}$ . In practice,  $L_{min} = i_{min} r_{min}$  and  $L_{max} = i_{max} r_{max}$ . If we use the aforementioned average office lighting and reflection range values as a guide, we can expect  $L_{min}=10$  and  $L_{max}=1000$  as typical limits for indoor values without any additional lighting.

The range  $[L_{min}, L_{max}]$  is called gray (intensity) scale. The common practice is to shift this range numerically to the range  $[0, L-1]$ . Here, in gray scale  $l=0$  is assumed to be black and  $l=L-1$  is assumed to be white. All intermediate values are gray tones varying from black to white.

## 2.6. IMAGE INTERPOLATION

Image interpolation is a tool which is extensively used in operations such as enlargement, reduction, rotation and geometrical correction. Basically, intermediate valuation is a process of using known data to estimate values in unknown places. This can be explained by the following example. Assume that an image with 600x600 pixel size is enlarged twice to a size of 1200x1200. One way to visualize the enlargement is to create a 1200x1200 virtual grid with the same pixel pitch as its original and then shrink it to fit the original view. The pixel pitch of the reduced 1200x1200 grid will be less than the pixel pitch of the original image. In order to assign intensity level to each point in the superimposition, we look for the closest pixel to that point and assign the intensity of this pixel to the new pixel in 1200x1200 grid. When we have finished all the intensity assignments of all the points in the superimposition grid, we expand the grid to its original size to get the enlarged image.

This method is called the nearest neighbor interpolation. Because the method assigns the intensity of its nearest neighbor in the original image to each new place. This

approach is simple, but prone to create undesirable artifacts such as severe degradation of flat edges. For this reason, it is not often used in practice. A more appropriate approach is the double linear interpolation. In this interpolation, the nearest four neighbors are used to estimate the intensity at a given area. If  $(x,y)$  indicates the coordinates of the area to which an intensity value is intended to be assigned and  $v(x,y)$  indicates its intensity value; the value assigned for double linear interpolation is calculated by using:

$$v(x, y) = ax + by + cxy + d \quad (2.8)$$

equation. Four coefficients that are used here are found from four equations with four unknowns which are written by using the four nearest neighbors of  $(x,y)$ . Although its calculation load is a little more than the nearest neighbor interpolation, the double linear interpolation yields much better results.

The next level of complexity is the bicubic interpolation which involves the nearest sixteen neighbors of a point. Intensity value that is assigned to point  $(x,y)$  is obtained from the equation

$$v(x, y) = \sum_{i=0}^3 \sum_{j=0}^3 a_{ij} x^i y^j \quad (2.9)$$

Sixteen coefficients that are used here are found from sixteen equations with sixteen unknowns which are written by using the sixteen nearest neighbors of  $(x,y)$ . In general, bicubic interpolation works better in terms of fine detail protection than its double linear counterpart.

## 2.7. PRINCIPLES OF SPATIAL FILTERING

Filter name refers to accepting (permeating) or rejecting certain frequency components of filtering. For example, a filter that permeates low frequencies is called a low-pass filter. The net effect created by a low-pass filter is the blurring (softening) of the image. We can also do a similar softening on the image itself by using spatial filters (also called spatial masks, cores, and windows). In fact, there is a similarity between the linear spatial filters and the filters in the frequency domain.

A spatial filter consist of predefined process performed on image pixels of a spatial filter from 1 neighborhood zone (usually a small rectangle) and surrounded by 2 neighborhood regions. The filter creates a new pixel whose coordinates are equal to the coordinates of the center of the neighborhood and whose value is the result of filtering. A filtered image is created when each of the pixels in the central input image of the filter is visited one by one.

Figure 2.10 shows the operation of the linear spatial filter using a 3x3 neighborhood area. At any (x,y) point in the image,  $g(x,y)$  which is the output of the filter is the sum of the multiplication of the filter coefficients and the image pixels surrounded by the filter.

$$g(x,y)=w(-1, -1)f(x-1,y-1)+w(-1,0)f(x-1,y)+\dots+w(0,0)f(x,y)+\dots+w(1,1)f(x+1,y+1)$$

The center coefficient of the filter,  $w(0,0)$  is at the same level with the pixel in (x,y). For a mask of size  $m \times n$ , where  $a$  and  $b$  are positive integers, it is assumed that  $m=2a + 1$  and  $n=2b + 1$ . Linear spatial filtering of an  $M \times N$  image with an  $m \times n$  filter is expressed as:

$$g(x, y) = \sum_{s=-a}^a \sum_{t=-b}^b w(s, t) f(x + s, y + t) \quad (2.10)$$

Here,  $x$  and  $y$  are changed so that each pixel in  $w$  stops by at every pixel in  $f$ .



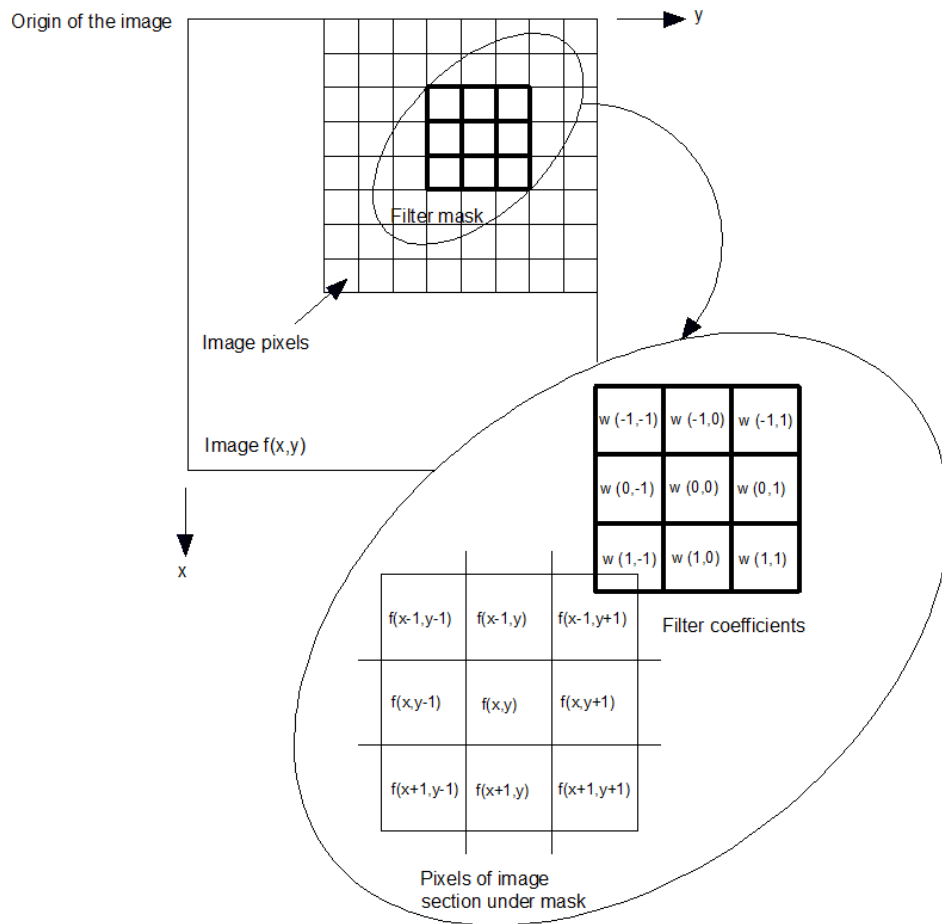


Figure 2.10. The mechanics of spatial filtering [18].

Filtering in the frequency domain comes from changing the Fourier transformation of the image and then obtaining the processed result by taking the inverse transformation. Thus, the basic filtering equation for a given  $M \times N$  dimensional digital image  $f(x,y)$  is as follows:

$$g(x,y) = \mathfrak{F}^{-1}[H(u,v)F(u,v)] \quad (2.11)$$

While  $\mathfrak{F}^{-1}$  is the IDFT of the image,  $F(u,v)$  is its DFT.  $H(u,v)$  is the filter function (it can also simply be called filter transfer function) and  $g(x,y)$  is the filter(output) image. Just like the input image,  $F$ ,  $H$ , and  $g$  functions are arrays in  $M \times N$  dimensions [18-22].

## CHAPTER 3

### ROBOTIC SYSTEMS

In this section, information on robotic systems which is one of the two main parts of the research (image processing and robotic arm) will be given. As in image processing, only necessary information related to the scope of this thesis will be given without going into details and too many theoretical information.

A robot is an electro-mechanical device capable of performing autonomous or pre-programmed tasks. According to its current definition, robots are programmable devices with an ability to sense that consist of electronic and mechanical units. According to another definition, robots are engineering products that contain interdisciplinary items with physical abilities and artificial intelligence which can mimic the functions and behaviors of living things.

A robot is a machine that behaves intelligently depending on computer algorithms with its mechanical systems and associated control and sensing systems. A robot is a reprogrammable, multi-functional machine which carries or processes material, parts and tools with programmed movements for the performance of a related task. (Robot Institute of America 1979) A robot is a controlled mechanical device equipped with at least one arm, holding organs (usually clamps, vacuum lifters, or electromagnets), pneumatic, hydraulic or electrical sensors, position and pressure sensors, and data processing organs which are placed on a base.

Today, the biggest usage area of robots is industrial production. A large number of robots are used in various areas of the industry. Most of them are arm-formed robots. These robots carry, mount, assemble, weld and paint the parts [23,24].

### 3.1. DEFINITIONS RELATED TO ROBOTIC SYSTEM

Robots move in a three-dimensional space in which objects in their surroundings also take place. Hence, in order to determine the location and orientation of the object around the robot with respect to each other, a coordinate system is set over each object including the robot itself in the three-dimensional space. Definitions of some important terms including position, orientation, and coordinate system are given below.

#### 3.1.1. Position

A point can be located anywhere in the universal framework by defining a coordinate system. Many coordinate systems can be placed in the universal coordinate framework. In the three-dimensional space, a point can be represented by a 3x1 dimensional vector defined according to the center of these coordinate systems. These vectors are named according to the coordinate system to which they are defined. For instance, the location of a P point which is in the universal framework in respect to {A} coordinate system is represented by a vector in the form of  $A_P$ .  $A_P$ , numerically defines the distance of the P point to the center of {A} coordinate system on (x,y,z) axes. The  $A_P$  vector is mathematically represented as in the 3.1 equation.

$$A_P = \begin{bmatrix} A_{p_x} \\ A_{p_y} \\ A_{p_z} \end{bmatrix} \quad (3.1)$$

Figure 3.1 shows {A} coordinate system which has three units perpendicular to each other and P point together.

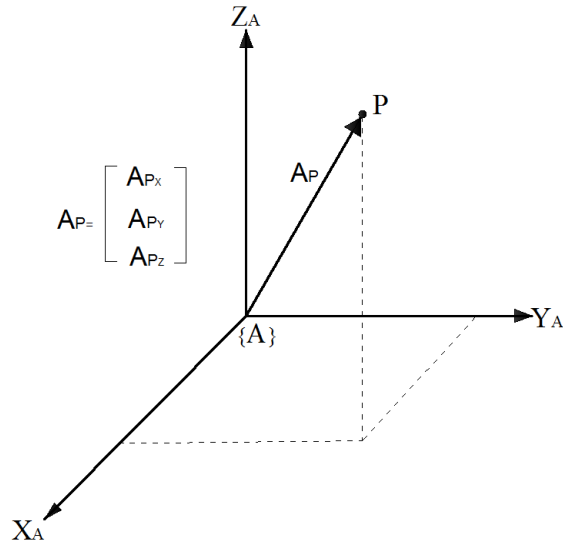


Figure 3.1. Defining P point in respect to {A} coordinate system.

As shown in Figure 3.2, the same P point can be defined according to both the {A} and {C} coordinate systems. The distance of the P point to the center of {A} coordinate system and the center of {C} coordinate system does not have to be equal.

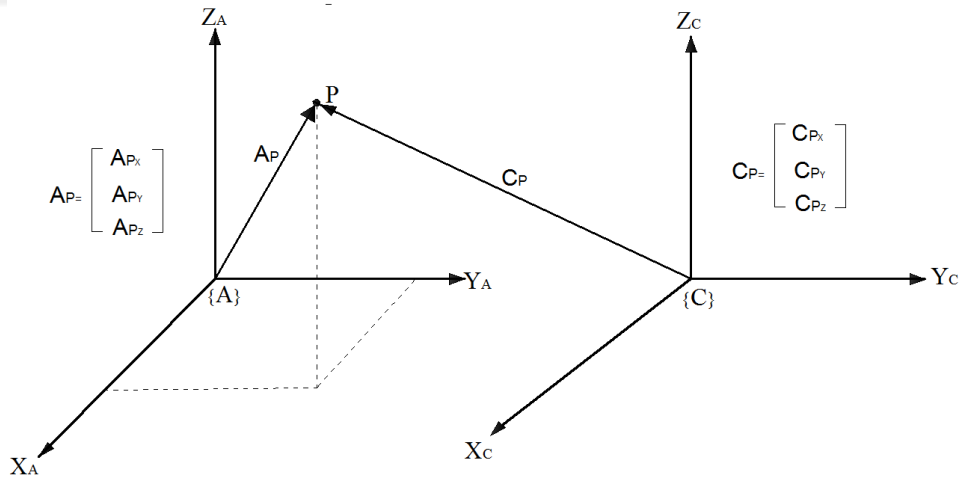


Figure 3.2. Defining P point in respect to both {A} and {B} coordinate systems.

### 3.1.2. Orientation

In the three-dimensional space, the orientation of a point is also defined together with its position in respect to any coordinate system. Orientation is the amount of rotation

of one coordinate system with respect to another coordinate system and it is expressed by a 3x3 dimensional matrix. A coordinate system is set over a rigid body in order to define the orientation of this rigid body according to another reference coordinate system. As it can be seen in Figure 3.3, by putting {B} coordinate system over end function, its orientation according to {A} reference coordinate system is defined.

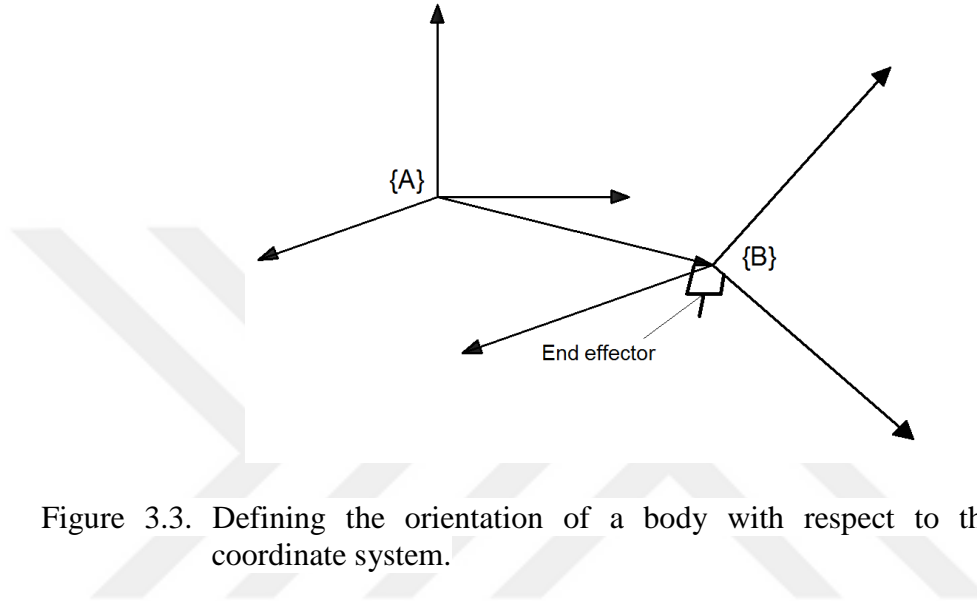


Figure 3.3. Defining the orientation of a body with respect to the reference coordinate system.

Unit vectors are used to express the {B} coordinate system, which is placed in the end function, in terms of the {A} reference coordinate system. The vectors of the {B} coordinate system can be represented as follows [23,29].

$$\{B\} = \dot{X}_B, \dot{Y}_B, \dot{Z}_B \tag{3.2}$$

On the other hand, the vectors of {B} coordinate system can be shown in terms of {A} coordinate system as follows:

$$\{A\} = {}^A\dot{X}_B, {}^A\dot{Y}_B, {}^A\dot{Z}_B \tag{3.3}$$

The expression in the equation 3.3 can be expressed as in the equation 3.4 by using a 3x3 dimensional matrix. This matrix which can be written as 3x3 is called the rotation matrix. Since it explains the orientation of the {B} coordinate system according to the {A} coordinate system, this matrix is denoted as  ${}^A_B R$ . It indicates

the amount of the rotation of the {B} coordinate system with respect to the {A} coordinate system on X, Y, Z axes.

$${}^A_B R = [A_{\dot{X}_B} \quad A_{\dot{Y}_B} \quad A_{\dot{Z}_B}] = \begin{bmatrix} r_{1_1} & r_{1_2} & r_{1_3} \\ r_{2_1} & r_{2_2} & r_{2_3} \\ r_{3_1} & r_{3_2} & r_{3_3} \end{bmatrix} \quad (3.4)$$

### 3.1.3. Resolution

It is the smallest amount of displacement that can be done with the robot. The resolution of the robot is defined as its smallest displacement capability with a load determined at its end-function movement and it depends on the resolution of the sensors that measure the position, velocity and force at the robot's joints.

### 3.1.4. Sensitivity

It can be defined as the ability of the end function of the robot to follow the desired trajectory with zero error during its movement with a specific load. The robot precision is based on the high-quality data transmitted from the sensors, the power transmission mechanism, the actuator structure and the motion control algorithms.

### 3.1.5. Repeatability

It can be defined as the capability of the robot to provide the same position information every time.

### 3.1.6. Degree of Freedom

It is the number of the independent variables required to fully determine the position of an object or a system with respect to a fixed point. The degree of freedom in robots can be easily found by counting joints and actuators. The degree of freedom of a mechanism is the number of parameters required to determine the position of all the limbs in that mechanism.

In the field of robotics the “axis” should be thought as the Degree of Freedom (DoF). If a robot has 3 Dof, it means that it can move on x-y-z axis. Bu it cannot tilt or turn. When you increase the DoF, that is to say the axis of the robot, you will have more workspace.

### **3.1.7. Manipulator**

They are mechanical systems that form an open-ended kinematic chain connected by joints.

### **3.1.8. Servo Motor**

They are custom designed motors for precise adjustment of the position of the motor with hardware such as a sensor that measures the amount of movement and a brake system. Step motors are also used together with servo motors in robots.

### **3.1.9. Working Volume**

It is the cluster of points that the robot’s end point can reach.

### **3.1.10. Workspace**

It is the area that is formed by the projection of the points that the robot’s end point can reach to the horizontal plane.

## **3.2. CLASSIFICATION OF ROBOTS**

Robots can be categorized in a wide variety of forms according to their degree of freedom, control methods, actuator structures, sensitivity and sharpness ratings, and joint structures. However, the most preferred way of categorization is classifying the robots according to joint configuration and/or degree of freedom.

Apart from these, industrial robots are basically divided into two groups as serial and parallel robots. The serial robots generally consist of a number of joints and links connecting these joints. The serial robots have a wider workspace and fewer mechanical components. On the other hand, parallel robots consist of multiple parallel bars gathered between the main frame and the end function (TCP). Since one arm motion is dependent to the motion of the other arm, kinematic modeling of these robots is more difficult. However, parallel robots have a very strong mechanical structure compared to serial robots. In Figure 3.4 serial robot structure and in Figure 3.5 parallel robot structure is shown:



Figure 3.4. Kuka industrial robot with serial arm structure.



Figure 3.5. ABB industrial robot with parallel arm structure.



### **3.3. ROBOTS ACCORDING TO THEIR DEGREE OF FREEDOM**

When the robots are classified according to degree of freedom, joint structure of the first three arms is generally taken into account. If the first three arm structures are prismatic (Prismatic-Prismatic-Prismatic) then this robot is called a Cartesian (PPP) robot. If the first arm structure has revolute joint, and the structures of the second and third arms have prismatic joints (Revolute Prismatic Prismatic), then this robot is called a Cylindrical (RPP) robot. If the structures of the first two arms have revolute joints, the structure of the third arm has prismatic joint and if all these joints are parallel to each other, then this robot is called SCARA (Selective Compliant Articulated Robot Assembly); If the structures of the first two arms have revolute joints and the structure of the third arm has prismatic joint, then this robot is called a Spherical (RRP) robot; if the first three structures all have revolute joints, then this robot is called a Jointed robot. Robots used in the robotic production lines are generally jointed robots with six degrees of freedom [20].

#### **3.3.1. Cartesian (PPP) Robot**

Mechanical structures whose linkage structure of the first three arms is designed as prismatic joints are called Cartesian robots. This robot type has the simplest kinematic structure. Although they have a strong mechanical structure, their movement abilities in workspace are rather limited. The cartesian robots which only have the ability to hold and carry can make linear movements on X ,Y ,Z axes. As a result of their simple structure, they are easier to plan. Position calculations in such robots are simpler than other robot types because the position of the robot end element (TCP) is at the point where the joints currently are at that moment.

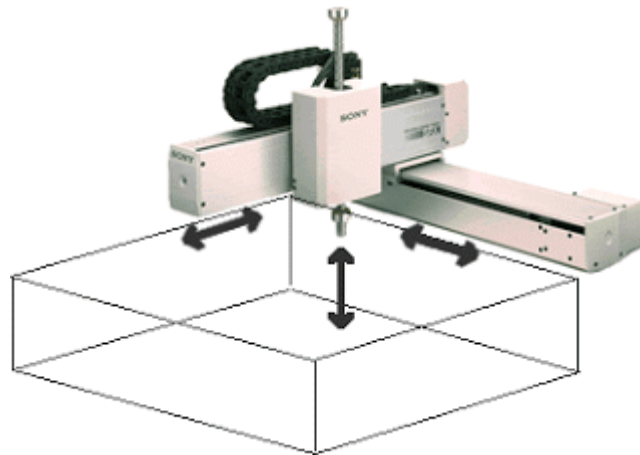


Figure 3.6. Industrial robot with cartesian structure.

### 3.3.2. Cylindrical (RPP) Robot

The first joint of a robot with a cylindrical joint configuration is rotational, the second joint is parallel to the first joint and prismatic, and the third joint is perpendicular to the second joint and prismatic. The cylindrical robot consists of a joint that can rotate around itself and arms which can make linear movements in X, Y, Z plane, placed on it. A cylindrical robot that is used in the industry can be seen in Figure 3.7.



Figure 3.7. Cylindrical robot (Seiko RT3300).

### 3.3.3. SCARA (RRP) Robot

Even though it looks like a spherical configuration thanks to its configured robot joint structure, SCARA (Selective Compliant Articulated Robot for Assembly) is totally different from the spherical robot in terms of the joint geometry. All of the first three joints are parallel to each other. It is widely used in the industry thanks to high position and velocity performance. In general terms, the robot consists of an electric motor at the junction points of the arms, and a pneumatic arm ensuring vertical (up and down) movement. Electric motors at the junction points of the arms provide rotation of the axes.



Figure 3.8. SCARA robot.

### 3.3.4. Spherical (RRP) Robot

The first two joints of this robot are revolute, while the third one is prismatic. The structure consists of waist, shoulder and elbow hinges. Hinges at the first and second axes can rotate around themselves, while the hinge at the third axis provide extension and contraction movement. As seen in Figure 3.9, the elbowroom has a spherical coordination system. Their kinematic structure is more complicated compared to Cartesian and cylindrical robot arms, and their solutions are harder.

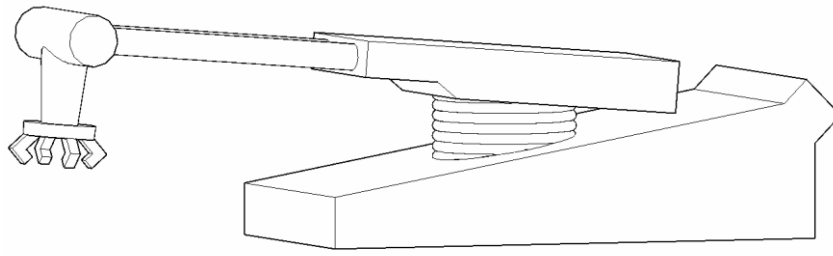


Figure 3.9. Spherical robot.

### 3.3.5. Rotary (RRR) Robot

All three joints of these robots are revolute. Figure 3.10 shows the revolute robot itself.



Figure 3.10. Revolute robot (Motoman SK16).

### 3.3.6. Articulated Robot

Having the greatest movement capabilities in the working space, this robot is the closest one to imitate human arm movements. As the robots with other arm structures have limited mobility in the production systems, robots with 5 to 6 joints at the junction points of the arms are needed. These robots, which may also have more than six joints, are called redundant robots. Figure 3.11 gives an example of an articulated robot.



Figure 3.11. Articulated robot.

Apart from the degree of freedom, robots are divided into two groups based on the methods of control: “point robots” and “continued orbit controlled robots”. Point robots have generally low degree of freedom and they are mainly used to remove and place an object (hold and place). It does not matter how a robot moves between two points. The only thing that matters is reaching the final point. On the other hand, continued orbit controlled robots are controlled by the user to follow a certain orbit.

Furthermore, the robots can be classified under three groups based on the power supply used by the robot actuators: robots moving with electric motors, robots moving with hydraulic servo motors and robots moving with pneumatic servo motors.

### **3.4. ROBOT KINEMATICS**

Robot kinematics is used to evaluate the movement in terms of position, orientation, velocity and acceleration. In robot studies, kinematic connections are used to model the duties of the arms and the interaction between neighboring arms. A relation between end-effector and joints is identified. In structural terms, a robot consists of independent joints ensuring translation (prismatic-P) and rotation (revolute-R) movements, and links that connect those joints. Figure 3.12 shows the joint structures performing translation and rotation movements. The change in location due to rotation movement is called joint angle; while the translation caused by the change of

location between the links is called joint offset. Figure 3.13.a shows a robot consisting of prismatic joints; while Figure 3.13.b is an illustration of a robot consisting of revolute joints.

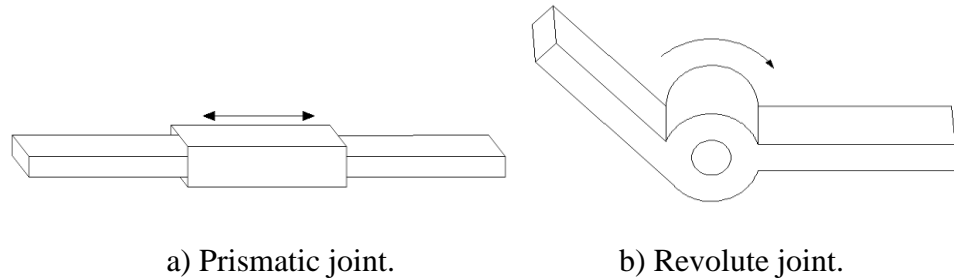


Figure 3.12. Joint structures performing translation and rotation movements.

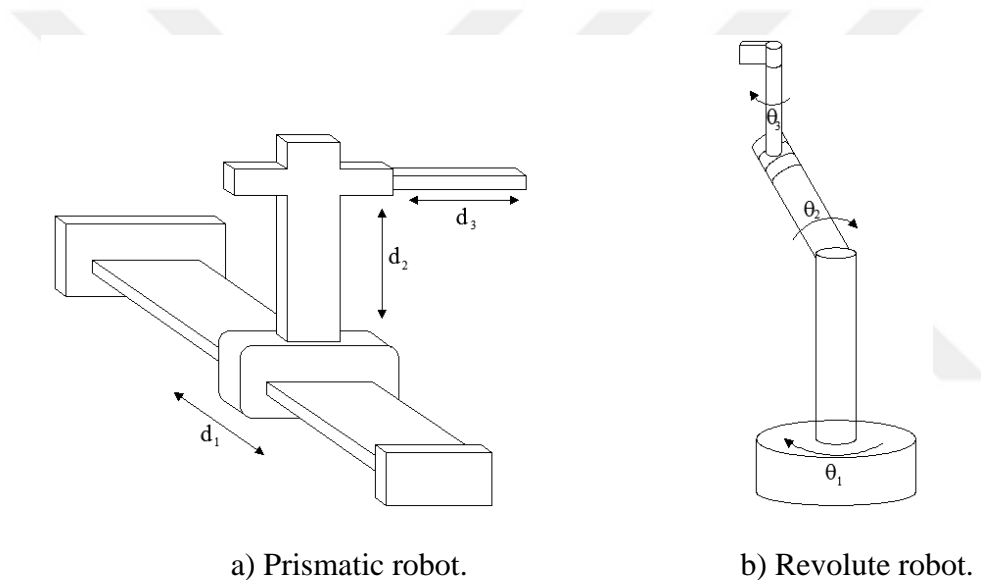


Figure 3.13. Robots consisting of prismatic and revolute joints [23].

Position and orientation of each robot joint can be expressed according to previous and next one. This consecutive relation is called open kinematic chain. The statements creating such relation consist of 4x4 homogeneous transformation matrices containing position and orientation data of the robot. The relation between each joint is expressed with a homogeneous transformation matrix. The number of such matrices is determined by the degree of freedom of the robot. Six degrees of freedom are adequate to reach a point in a three dimensional space through any orientation. On the other hand, the robots with more than six degrees of freedom are in a state of redundancy. In this case, an overlap is observed in an area scanned by a joint. Such robots are designed for special conditions and are used for special

purposes. Robotic system kinematics is addressed under forward kinematics and inverse kinematics.

### **3.4.1. Forward Kinematics**

Forward kinematics of a robot means finding the position and orientation of the end-effector based on the main frame of the robot. A robot consists of serial links connected through prismatic and/or revolute joints from the main frame to the tool frame. The relation between those two links is expressed through a homogeneous transformation matrix. Successive multiplication of joint transformation matrix gives the relation between the main frame and tool frame. This relation expresses the orientation and position of the tool frame according to the main frame. In other words, forward kinematic equation is used to calculate the position and orientation of the end-effector according to the main frame, using the joint variables given. The matrix is used to calculate joint angles and fixed physical quantities of the robot. These variables are placed in the transformation matrix to calculate the orientation and position of the robot from the main frame to the tool frame.

Several kinematic methods were developed to define joint variables of the robots. Kinematic problems are solved in two different spaces: three dimensional Cartesian and four dimensional Quaternion space. Several methods are used in Cartesian space, including exponential method and Pieper-Roth method. However, Denavit-Hartenberg (D-H) method is the most popular one used to determine joint variables of a robot.

#### **3.4.1.1. Denavit-Hartenberg Method**

In Denavit-Hartenberg method, robot kinematics is determined based on four main variables. These variables are the link length between two axes ( $a_{i-1}$ ), link twist between (i-1) and i axes ( $\alpha_{i-1}$ ), joint offset between overlapping links ( $d_i$ ), and the joint angle between two links ( $\theta_i$ ). These four variables are called D-H variable. To define those variables, a robot's rotation axes are determined as shown in Figure

3.14, and rotation axes are enumerated in a way that the axes outnumber the links [23,25,26].

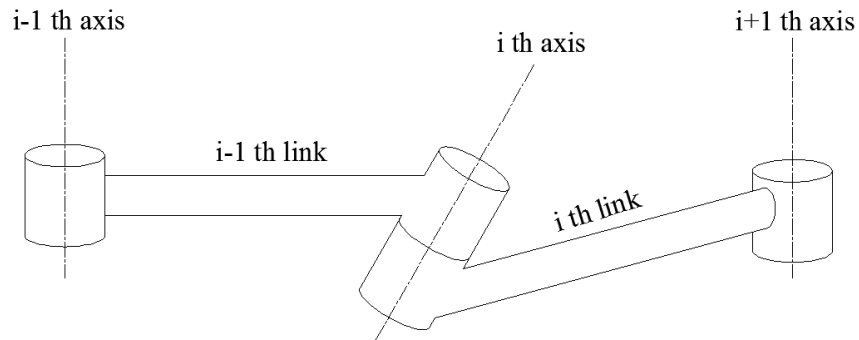


Figure 3.14. Layout of (I-1), i links, and (i-1), i and (i+1) axes [23].

Then, a coordinate system is placed on each of these axes. Link rotation axis is deemed to be the Z axis of the coordinate system, as seen in Figure 3.15.

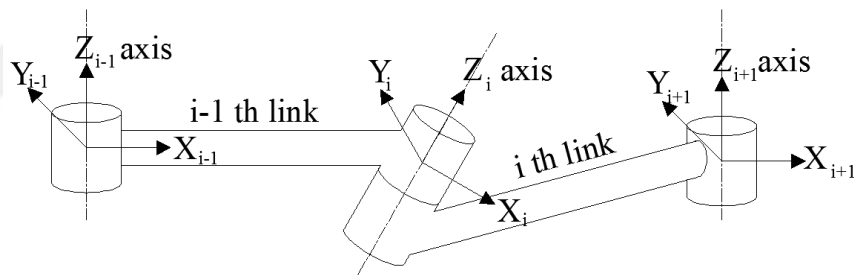


Figure 3.15. Placing coordinate systems on (i-1) and i axes.

Then, as seen in Figure 3.15, the vertical distance between  $Z_{i-1}$  and  $Z_i$  lying on the direction of  $X_{i-1}$  is defined as the  $a_{i-1}$  link length.



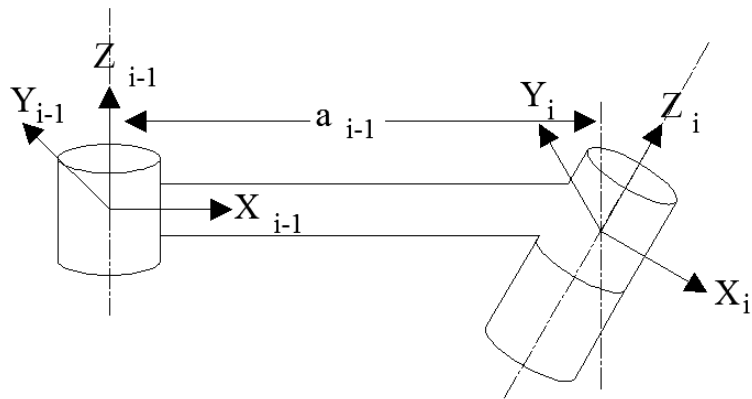


Figure 3.16.  $a_{i-1}$  link length lying on the direction of  $X_{i-1}$  between  $Z_{i-1}$  and  $Z_i$

On the other hand, the distance between overlapping links on  $Z_i$  between  $X_{i-1}$  and  $X_i$  is defined as the link deviation ( $d_i$ ) as seen in Figure 3.17.

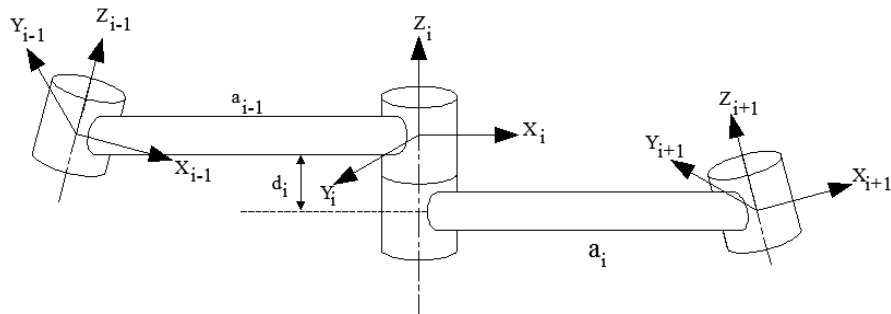


Figure 3.17.  $d_i$  link deviation on  $Z_i$  between  $X_{i-1}$  and  $X_i$  [23].

As can be seen in Figure 3.18, the angle between  $Z_{i-1}$  and  $Z_i$  rotation axes is  $\alpha_{i-1}$  link angle.

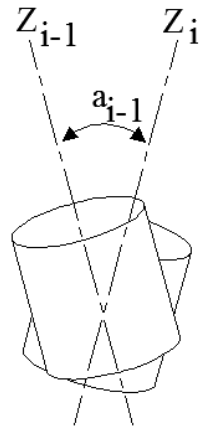


Figure 3.18.  $\alpha_{i-1}$  link angle between  $Z_{i-1}$  and  $Z_i$  rotation axes.

As can be seen in Figure 3.19,  $\theta_i$  joint angle is the angle on X axis between  $a_{i-1}$  and  $a_i$  links.

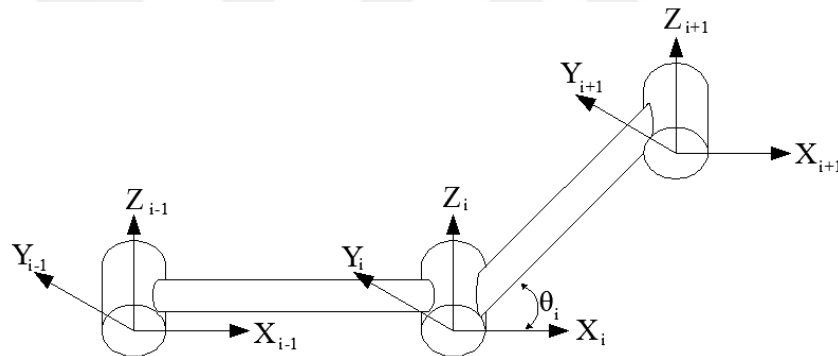


Figure 3.19.  $\theta_i$  joint angle between  $a_{i-1}$  and  $a_i$  links [23].

### 3.4.1.2. Placing Coordinate Systems on Joints

The following procedures are respectively applied to place coordinate system on joints.

- 1- Rotation and sliding directions of the joint axes are determined, and a line is drawn in parallel to this axis.
- 2- Rotation direction is deemed to be Z for joint axes and revolute axes, while sliding direction is accepted to be Z axis for prismatic joints.

3- Link length that is vertical to the Z axis and lying along the arm is accepted as X axis.

4- After determining Z and X axes, Y axis is determined based on right hand rule.

5- In cases where the rotation and sliding directions of two consecutive joints are the same, X axis is determined throughout the arm after identifying the Z axis. Finally, the Y axis is determined based on the right hand rule.

6- Axes no 0 and 6 may be deemed overlapping.

7- While placing coordinate systems on a serial robot, after defining the rotation direction of the first axis as Z axis, and upon rotating an X axis to this joint, an X axis placed so that two neighboring Z axes overlap.

After defining the coordinate systems to be placed on the joints in accordance with the above-mentioned rules, joint variables are denominated considering the following statements.

1-  $a_{i-1}$  is the length on  $\dot{X}_{i-1}$  between  $\dot{Z}_{i-1}$  and  $\dot{Z}_i$ .

2-  $\alpha_{i-1}$  is the angle on  $\dot{X}_{i-1}$  between  $\dot{Z}_{i-1}$  and  $\dot{Z}_i$ .

3-  $d_i$  is the length on  $\dot{Z}_i$  between  $\dot{X}_{i-1}$  and  $\dot{X}_i$ .

4-  $\theta_i$  is the angle on  $\dot{Z}_i$  between  $\dot{X}_{i-1}$  and  $\dot{X}_i$ .

$a_{i-1}$  and  $d_i$ , which are harder to define compared to  $a_{i-1}$ ,  $\alpha_{i-1}$ ,  $d_i$  and  $\theta_i$  variables explained above, can be calculated as follows.

$a_{i-1}$  length ( $i=1,2,3, \dots$ )

1-  $a_0$  is the length on  $\dot{X}_0$  between  $\dot{Z}_0$  and  $\dot{Z}_1$  ( $i=1$ )

2-  $a_1$  is the length on  $\dot{X}_1$  between  $\dot{Z}_1$  and  $\dot{Z}_2$  ( $i=2$ )

3-  $a_2$  is the length on  $\dot{X}_2$  between  $\dot{Z}_2$  and  $\dot{Z}_3$  ( $i=3$ )

4-  $a_3$  is the length on  $\dot{X}_3$  between  $\dot{Z}_3$  and  $\dot{Z}_4$  ( $i=4$ )

5-  $a_4$  is the length on  $\dot{X}_3$  between  $\dot{Z}_4$  and  $\dot{Z}_5$  ( $i=5$ )

$d_i$  length ( $i=1,2,3, \dots$ )

- 1-  $d_1$  is the length of  $\dot{Z}_1$  between  $\dot{X}_0$  and  $\dot{X}_1$  ( $i=1$ )
- 2-  $d_2$  is the length on  $\dot{Z}_2$  between  $\dot{X}_1$  and  $\dot{X}_2$  ( $i=2$ )
- 3-  $d_3$  is the length on  $\dot{Z}_3$  between  $\dot{X}_2$  and  $\dot{X}_3$  ( $i=3$ )
- 4-  $d_4$  is the length on  $\dot{Z}_4$  between  $\dot{X}_3$  and  $\dot{X}_4$  ( $i=4$ )

### 3.4.1.3. Joint Transformation Matrix

Based on the coordinate system placed on joint  $i$   $\{i\}$  and the coordinate system placed on joint  $i-1$   $\{i-1\}$ ;  $a_{i-1}$ ,  $\alpha_{i-1}$ ,  $d_i$  and  $\theta_i$  joint variables defined in Figure 3.20 are used and the transformation matrix of a joint of a robot is calculated as illustrated in equation 3.5.

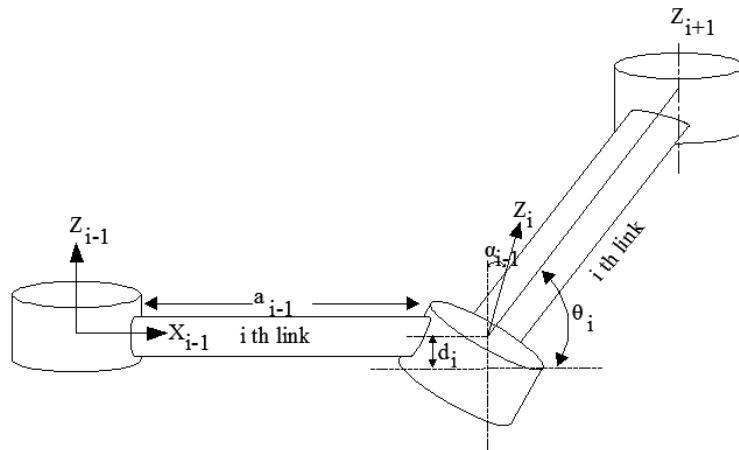


Figure 3.20. Calculation of  $a_{i-1}$ ,  $\alpha_{i-1}$ ,  $d_i$  and  $\theta_i$  joint variables.

Multiplication of matrices comprising of these four variables will give the transformation matrix of only one joint of a motor having an  $n$  degree of freedom. This transformation matrix consists of a  $3 \times 3$  rotation matrix and  $3 \times 1$  position vector.

$${}^i - {}^1 T = R_x(\alpha_{i-1})D_x(a_{i-1})R_z(\theta_i)D_z(d_i)$$

$${}^i - {}^1 T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha_{i-1} & -\sin\alpha_{i-1} & 0 \\ 0 & \sin\alpha_{i-1} & \cos\alpha_{i-1} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_{i-1} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} x$$

$$\begin{aligned}
& \begin{bmatrix} \cos\theta_i & -\sin\theta_i & 0 & 0 \\ \sin\theta_i & \cos\theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
&= \begin{bmatrix} 1 & 0 & 0 & a_{i-1} \\ 0 & \cos\alpha_{i-1} & -\sin\alpha_{i-1} & 0 \\ 0 & \sin\alpha_{i-1} & \cos\alpha_{i-1} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta_i & -\sin\theta_i & 0 & 0 \\ \sin\theta_i & \cos\theta_i & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
&= \begin{bmatrix} \cos\theta_i & -\sin\theta_i & 0 & a_{i-1} \\ \sin\theta_i \cos\alpha_{i-1} & \cos\theta_i \cos\alpha_{i-1} & -\sin\alpha_{i-1} & -\sin\alpha_{i-1} d_i \\ \sin\theta_i \sin\alpha_{i-1} & \cos\theta_i \sin\alpha_{i-1} & \cos\alpha_{i-1} & \cos\alpha_{i-1} d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.5)
\end{aligned}$$

In the equation,  $c\theta_i$  represents  $\cos \theta_i$ , and  $s\theta_i$  represents  $\sin \theta_i$ .

#### 3.4.1.4. General Rules Applied to Define Robot Kinematics

Forward kinematics of a robot manipulator at zero position is determined following the below-mentioned rules.

- 1- First of all, each joint of the robot is determined and a coordinate system is placed on each of these joints.
- 2- D-H variables are determined for each joint, considering Table x.x. Using the variables given in each line of the table, transformation matrix of a joint is determined. The number of lines or the number of transformation matrix defines the degree of freedom.  $\alpha_{i-1}$  and  $a_{i-1}$  parameters on the table are fixed parameters that do not change with the movement of the robot; while  $d_i$  and  $\theta_i$  parameters change based on the movement. It should be noted that only one of the  $d_i$  and  $\theta_i$  parameters determined for each joint is variable. The last column of the D-H table is determined accordingly.

Table 3.1. Determination of D-H variables [23].

Axis No	D-H Variables				$i^{\text{th}}$ Joint Variable
$i$	$\alpha_{i-1}$	$a_{i-1}$	$d_i$	$\theta_i$	$d_i$ or $\theta_i$
1	$\alpha_0$	$a_1$	$d_1$	$\theta_1$	$d_1$ or $\theta_1$
2	$\alpha_1$	$a_0$	$d_2$	$\theta_2$	$d_2$ or $\theta_2$
3	$\alpha_2$	$a_2$	$d_3$	$\theta_3$	$d_3$ or $\theta_3$
4	$\alpha_3$	$a_3$	$d_4$	$\theta_4$	$d_4$ or $\theta_4$

3- Upon determining the D-H variables in Table 3.1, the following general joint transformation matrix is used for each joint. (Equation 3.5)

$${}^{i-1}T_i = \begin{bmatrix} \cos\theta_i & -\sin\theta_i & 0 & a_{i-1} \\ \sin\theta_i \cos\alpha_{i-1} & \cos\theta_i \cos\alpha_{i-1} & -\sin\alpha_{i-1} & -\sin\alpha_{i-1} d_i \\ \sin\theta_i \sin\alpha_{i-1} & \cos\theta_i \sin\alpha_{i-1} & \cos\alpha_{i-1} & \cos\alpha_{i-1} d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.5)$$

4- Multiplication of the above-mentioned transformation matrices will provide forward robot kinematics from the main frame to the tool frame.

$${}^0T_N = {}^0T_1 {}^1T_2 {}^2T_3 \dots {}^{N-1}T_N \quad (3.6)$$

Where  ${}^0T_N$ , the transformation matrix is a function of N joints, while  ${}^{i-1}T_i$ , each joint matrix is a function of a joint variable.

5- Multiplication of the transformation matrices will provide a general transformation matrix that contains the position and orientation of the end effector and that constitutes a function for joint variables. This transformation matrix has a total of 12 components comprising of 9 rotations ( $r_{11}, r_{12}, r_{13}, r_{21}, r_{22}, r_{23}, r_{31}, r_{32},$  and  $r_{33}$ ) and 3 positions ( $p_x, p_y$  and  $p_z$ ).

### 3.4.2. Inverse Kinematics of Robots

Reverse kinematics problem for robots means calculation of angle sets required for a robot to reach certain orientation and position, based on the orientation and position of tool frame according to the main frame. In other words, reverse kinematics problem is transformation of the orientation and position of the end effector pertaining to a robot manipulator from Cartesian coordinate system to joint coordinate system. As the problems faced in reverse kinematics are not linear, such equations may be very complicated to solve. Figure 3.21 provides schematic illustration of forward and reverse kinematics problem. Reverse kinematic problem solving is extremely important for several applications such as real time control, calculation of joint torques of the actuators, welding, dyeing and orbit planning.

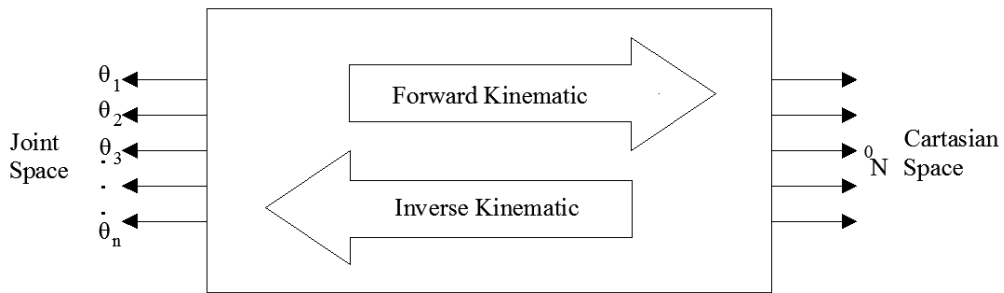


Figure 3.21. Schematic illustration of forward and reverse kinematics problem.

#### 3.4.2.1. Structure of Inverse Kinematics

Reverse kinematic problem is very hard to solve due to the following characteristics.

- 1- It contains analytically complex and non-linear equations.
- 2- It depends on joint structure. The greater the number of prismatic joints the easier to produce reverse kinematics solutions; while the greater the number of revolute joints the harder to produce a solution.
- 3- Mathematical solution does not necessarily represent the physical solution. As seen in Figure 3.22,  $\theta=A \tan 2(-k,p_z)$  provides total match between mathematical and physical solutions; while  $\theta=A \tan 2(k,-p_z)$  gives a solution that cannot be applied in physical terms.

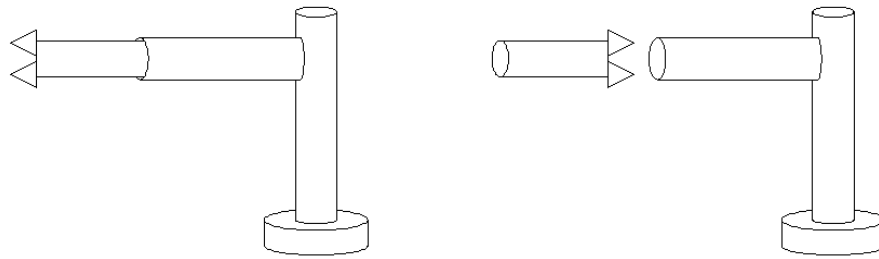


Figure 3.22. The relation between mathematical statement and physical solution.

4- There may be more than one solution for the same end effector configuration. The number of reverse kinematics solution depends on both the degree of freedom of the robot and the joint variables. Having both a and d parameters in each joint will increase the number of solution. For instance in Figure 3.23, when there is at least one a or d variable in each joint of 6R robot, the number of reverse kinematics solution is  $2^6=64$ . However, some of these solutions may be real, while others are virtual.

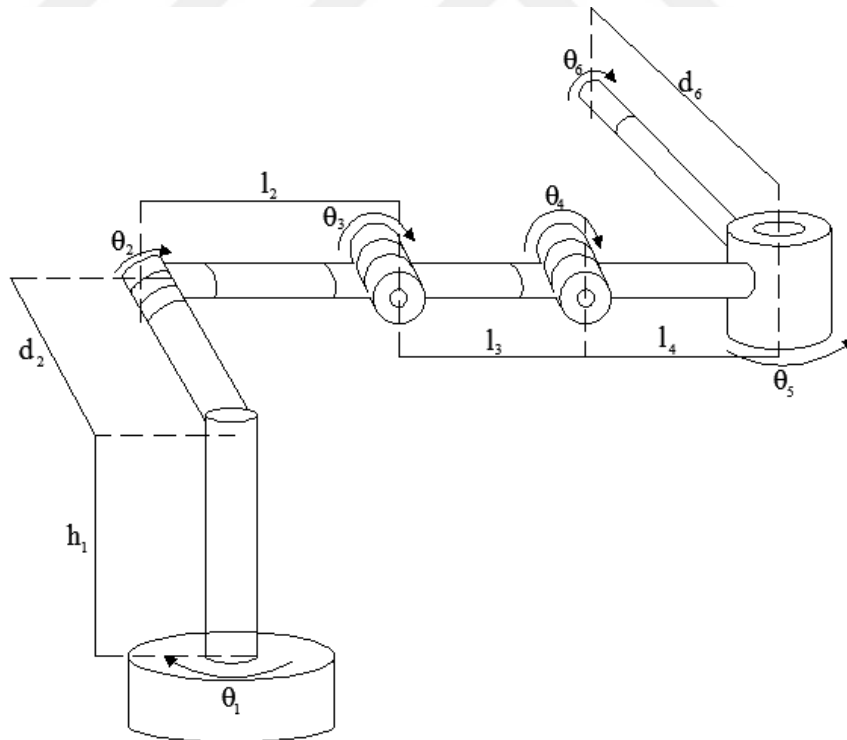


Figure 3.23. Impact of the joint structure on the number of solutions.

Joint structure of the robots have a significant impact on the number of reverse kinematics solution. Prismatic joints reduce the number of solution, while revolute



joints increase this number. Furthermore, having greater number of physical solutions in robots consisting of revolute joints provides the opportunity to reach a point in three dimensional space in several different ways. Figure 3.24 demonstrates that PUMA-560 robot can reach the same point in four different ways.

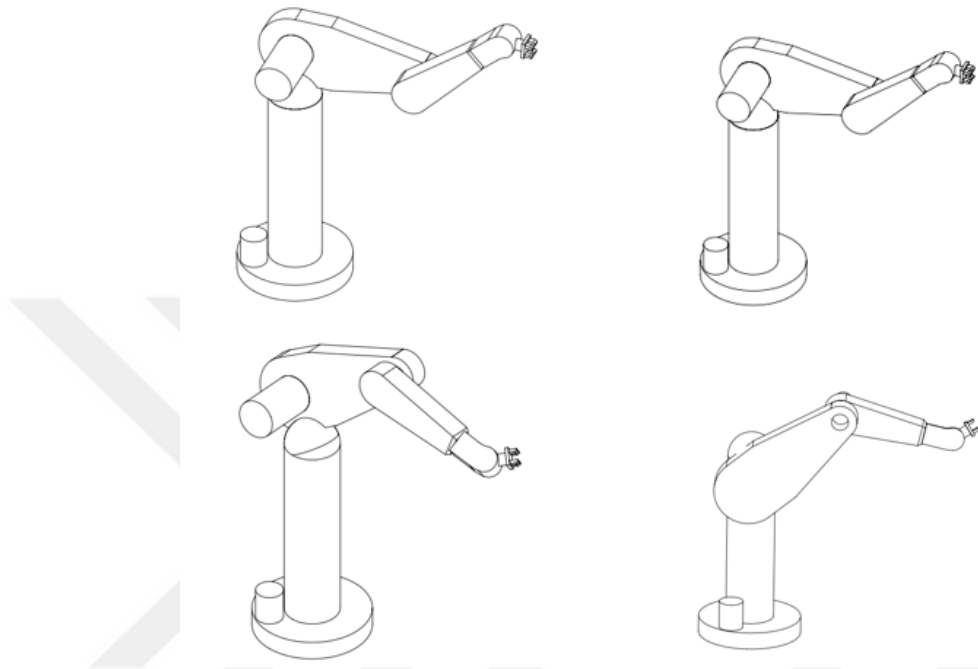


Figure 3.24. Four different solutions for PUMA-560 robot.

5- Reverse kinematic problem can be solved completely analytically (closed form) for the robot configuration given; however, numerical methods can also be used in cases where analytical solution is not suitable. Nevertheless, the equations of analytic solution producing a precise result work very rapidly in computer environment; while the numerical solution solving the joint angles iteratively is slower in the same environment compared to the analytical solution. Furthermore, the structure of the algorithm developed to calculate joint angles numerically (solution time and initial conditions) is also very significant.

As the numeric solution is slower in the computer environment compared to the analytic solution, robot designers usually prefer the designs offering analytical solution. Today, industrial robots are generally produced in a simple structure that can be solved analytically. As can be seen in PUMA and SCARA robots, if three

consecutive joints are parallel, reverse kinematics solution of the robots with 6 degrees of freedom can be produced analytically [27-31].

### 3.4.2.2. Inverse Kinematics Solution Methods

#### Analytical Solution Approach

Forward kinematics of a robot having six degrees of freedom as defined by Craig can be written as equation 3.7.

$${}^0T_6 = {}^0T_1 {}^1T_2 {}^2T_3 {}^3T_4 {}^4T_5 {}^5T_6 \quad (3.7)$$

${}^0T_6$  forward kinematics matrix is written in the form of equation 4.2 in terms of the matrix components containing position and orientation, and the equity in equation 3.9 is ensured. First column of the matrix given in equation 3.8 illustrates the standard vector of the end effector ( $n=[n_x \ n_y \ n_z]^T$ ); while the second and third columns show sliding vector ( $s=[s_x \ s_y \ s_z]^T$ ) and approaching vector ( $a=[a_x \ a_y \ a_z]^T$ ), respectively.

$${}^0T_6 = \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.8)$$

$$[{}^0T_1]^{-1} {}^0T_6 = [{}^0T_1]^{-1} {}^0T_1 {}^1T_2 {}^2T_3 {}^3T_4 {}^4T_5 {}^5T_6 \quad (3.9)$$

As,  $[{}^0T_1]^{-1} {}^0T_1 = I$ , equation 3.9 can be simplified as follows.

$$[{}^0T_1]^{-1} {}^0T_6 = {}^1T_2 {}^2T_3 {}^3T_4 {}^4T_5 {}^5T_6 \quad (3.10)$$

Reverse kinematics problem is solved by equalizing the matrix components at either side of the equation 3.10. Other equations are given below.

$$[{}^0T_1 {}^1T_2]^{-1} {}^0T_6 = {}^2T_3 {}^3T_4 {}^4T_5 {}^5T_6 \quad (3.11)$$

$$[{}^0_1T {}^1_2T {}^2_3T]^{-1} {}^0_6T = {}^3_4T {}^4_5T {}^5_6T \quad (3.12)$$

$$[{}^0_1T {}^1_2T {}^2_3T {}^3_4T]^{-1} {}^0_6T = {}^4_5T {}^5_6T \quad (3.13)$$

$$[{}^0_1T {}^1_2T {}^2_3T {}^3_4T {}^4_5T]^{-1} {}^0_6T = {}^5_6T \quad (3.14)$$

Another analytical approach for reverse kinematics problem solution was developed by Raghavan and Roth. Raghavan and Roth degraded a multiple-variable system into a sixteen-degree polynomial in  $\tan(\theta_3/2)$ . Third joint angle ( $\theta_3$ ) is calculated based on these equations. The other joint angles are calculated based on other interim equations.

Raghavan and Roth modified the matrix equation previously used in reverse kinematics solutions of robot manipulators as seen in equation 3.15.

$${}^2_3T {}^3_4T {}^4_5T = {}^1_2T^{-1} {}^0_1T^{-1} {}^0_6T {}^5_6T^{-1} \quad (3.15)$$

$\theta_3$ ,  $\theta_4$  and  $\theta_5$  joint variations are on the left side of the equation, while the right side shows  $\theta_1$ ,  $\theta_2$  and  $\theta_6$  joint variations. The researchers reduced the polynomial degrees of the statements produced based on the equation 3.15, decreased symbolic complexity, and linearly obtained the equation 3.16.

$$Q \begin{bmatrix} \sin\theta_1 \sin\theta_2 \\ \sin\theta_1 \cos\theta_2 \\ \cos\theta_1 \sin\theta_2 \\ \cos\theta_1 \cos\theta_2 \\ \sin\theta_1 \\ \cos\theta_1 \\ \sin\theta_2 \\ \cos\theta_2 \end{bmatrix} = P \begin{bmatrix} \sin\theta_4 \sin\theta_5 \\ \sin\theta_4 \cos\theta_5 \\ \cos\theta_4 \sin\theta_5 \\ \cos\theta_4 \cos\theta_5 \\ \sin\theta_4 \\ \cos\theta_4 \\ \sin\theta_5 \\ \cos\theta_5 \end{bmatrix} \quad (3.16)$$

In equation 3.16, Q is a 14x8 dimensional matrix containing the variables regarding the position of end effector; while, P is a 14x9 matrix containing the linear functions of  $\sin \theta_3$  and  $\cos \theta_3$ , coefficients thereof, manipulator variables and positions. Eight out of fourteen equities mentioned in Q matrix in equation 4.10 were used,  $\theta_1$  and  $\theta_2$

variables at the left side of the equation were removed from the equation which was created based on  $\theta_3, \theta_4$  and  $\theta_5$  functions. Furthermore, for the solution

$$\sin\theta_i = \frac{2X_i}{1+X_i^2} \quad \text{and} \quad \cos\theta_i = \frac{1-X_i^2}{1+X_i^2} \quad (3.17)$$

They used equation transformation. In the equation,  $X_i = \tan\left(\frac{\theta_i}{2}\right)$ . Power of the system mentioned in equation 3.16 is calculated to obtain equation 3.18.

$$\begin{bmatrix} A_{11} & A_{12} & A_{13} & 0 \\ A_{21} & A_{22} & A_{23} & 0 \\ 0 & A_{11} & A_{12} & A_{13} \\ 0 & A_{21} & A_{22} & A_{23} \end{bmatrix} \begin{bmatrix} X_4^3 X_5^2 \\ X_4^3 X_5 \\ X_4^3 \\ X_4^2 X_5^2 \\ X_4^2 X_5 \\ X_4^2 \\ X_4 X_5^2 \\ X_4 X_5 \\ X_4 \\ X_5^2 \\ X_5 \\ 1 \end{bmatrix} = 0 \quad (3.18)$$

In the equations,  $A_{ij}$  is 3x3 dimensional matrix, while 0 is 3x3 dimensional 0 matrix. Components of  $A_{ij}$  matrix are the third degree polynomial of  $x_3$ . The matrix having  $A_{ij}$  and 0 as components is a 12x12 dimensional matrix. The determinant of this matrix is a 24<sup>th</sup> degree polynomial of  $x_3$  that can be divided to  $(1+x_3^2)^4$ . By equalizing this polynomial to zero, roots of  $\theta_3$  (third joint angle) is found. These roots are also the reverse kinematics problem solution of the third joint angle,  $\theta_3$ . After finding  $\theta_3$ , other joint angles can easily be found by solving linear equation system. Some of the trigonometric equations used to produce reverse kinematic solution are given below.

$$\cos\theta = a, \quad \theta = \text{Atan2}\left(\pm\sqrt{1-a^2}, a\right) \quad (3.19)$$

$$\sin\theta = a, \quad \theta = \text{Atan2}\left(a, \pm\sqrt{1-a^2}\right) \quad (3.20)$$

$$\cos\theta=a \text{ ve } \sin\theta=b, \quad \theta = A \tan2(b, a) \quad (3.21)$$

$$a \sin\theta + b \cos\theta = 0, \theta = A \tan 2(-b, a) \text{ or } \theta = A \tan 2(b, -a) \quad (3.22)$$

$$a \sin\theta + b \cos\theta = c, \theta = A \tan 2(a, b) + A \tan 2(\pm \sqrt{a^2 + b^2 - c^2}, c) \quad (3.23)$$

### **Numerical Solution Methods**

During the real time controls, precise solution of reverse kinematic problem pertaining to the robot is preferred. Precise solution is obtained via analytical solution of the problem. More often than not, such analytical solution can be obtained when three successive axes intersect at a point (Euler wrist) or when they are parallel. Most of the industrial robots are designed in such a way to ensure analytical solution. And that is only possible when Euler wrist is used. However, in certain cases, industrial robots are required to lift heavier loads. And in those robots, instead of Euler wrist, joint deviated wrist that does not connect three joints is used. Some of the industrial robots having six degrees of freedom, which are produced with joint deviated wrist, can be solved analytically, while it is not possible to produce analytical solutions for some of them. That is because reverse kinematic problem consists of non linear equation systems containing two or three unknowns. These equation systems that cannot be solved analytically are solved with certain iterative methods. Pashkevich solved the reverse kinematics of robots that cannot be solved analytically using the numerical method he developed. Contrary to the well known numerical solution methods, Pashkevich's method solves reverse kinematic problems using effective one dimensional iterative research without using Jacobian matrix.

## **CHAPTER 4**

### **PLC AND SCADA**

Since certain part of the system control has been conducted using PLC (Programmable Logic Controller) and as the system has been monitored with SCADA (Supervisory Control and Data Acquisition), this section was designed to provide general information on PLC and SCADA.

Industrial automation can be defined as a process that ensures automatic performance of the procedures required for an industrial production system to operate exactly as desired and intended. In the most general sense, industrial automation systems consist of certain sections used to perform command, control and data communication functions. Industrial automation systems are the systems that regulate and verify the operating conditions of production units according to logical rules (such as activation and deactivation). Control systems are installed to ensure that a production process operates at a desired value, without any disruptive impact. The main role of the control system is to eliminate the difference between the controlled and desired quantities, which arises for any reason, as soon as possible and considering certain criteria. Data communication systems are used to ensure reliable and rapid flow of real time information between units. Benefiting from the opportunities offered by data communication systems; real time process monitoring, remote control and control processes can be performed using special-purpose software such as SCADA. Programmable Logic Controller is the most significant device that can take essential roles in verifying all three parts of the industrial automation systems.

## **4.1. PLC**

In general terms PLC is an electronic device that performs the function written in accordance with the digital principles designed to be used in industry, ensures data exchange with the system thanks to its input / output units, and offers a general control thanks to its timing, counting, storage and arithmetic operation functions. Using certain measurement devices, PLC determines the physical events in the system controlled, assesses information according to the program written by the user, and reflects the results of the assessment on the system through the components it controls. Information obtained from the system is the electrical signals of the changes that occur during operation. Such information may be analog or digital. If the information is analog, the inquiry is made on certain range of the value, while for digital information the inquiry is shaped based on whether or not there is a signal. PLC can be used to control only one machine or to command the whole factory. Today, PLCs are widely used in all areas of industry. PLC is an essential control component in industrial processes, and several PLC brands / models are produced.

### **4.1.1. Operation and Function of PLC**

Even if the structures and operation principles of all PLCs are similar, from this point of the paper on, details of the PLC brand / model used in the project (thesis), Siemens S7 1200, are given.

A PLC basically consists of three parts. First one is the input part (input card), second one is the central processor unit, and the third one is the output part (output card) where the components to be controlled are connected.

Generally all PLCs have a module to reduce its own supply voltage (24V DC or 220V AC) to 5V for CPU. DC 24 V is the most popular voltage for input and output parts, however, this value may reach up to 220 V.

AC inputs are electrically isolated from CPU using opto-coupler. Similarly, the outputs are isolated from CPU in electrical terms using opto-coupler or relay. Output currents are defined in the producer company catalogs.

Status of the signals from the sensors (0/1) is entered on a table, which looks like a big signboard PII (Process-Image Input Table). Similarly, outputs obtained in the program are recorded on another table PIQ (Process-Image Output Table).

Operating system takes the information on PII, applies the logic operations we ordered, and enters the final outputs to PIQ. After processing all commands, the outputs on the PIQ are written on the output card, and a new cycle is initiated with the data on PII.

Furthermore, thanks to its microprocessor, SYSTEM BUS (communication protocol of the components comprising the CPU) communicates with certain components such as time component, numerator, process input image and status evaluation marks etc. and uses the same in the process of the program.

The method used to write the program (LAD, FBD, SCL, GRAPH) does not matter, because it is transformed into a form that can be understood by CPU once it has reached the CPU.

## **4.1.2. PLC Components**

### **4.1.2.1. Input Unit**

They are the units that receive electrical values (analog and/or digital) from certain components such as the pressure, level, and temperature sensors, buttons and limit switches, and transfer such values to CPU upon turning them into logic values (0-1). Depending on their brands and models, PLCs may have only digital or both digital and analog inputs.



#### **4.1.2.2. Output Unit**

These units turn the logic values written on the output image memory (PIQ) by CPU (i.e. digital or analog output signals), into electrical marks. It is located on the hardware that is suitable to drive control components on the system, such as contactor, relay, and solenoid. Output unit may have relay, transistor or triac output. Relay is used to drive the components having a different voltage rate from the PLC output, where output signals are not frequent (DC/AC 2A). Transistor is used in DC outputs that require very high number of on/off (DC 0.5 A). Triac is used in AC outputs that require very high number of on/off (AC 1A).

#### **4.1.2.3. CPU (Central Process Unit)**

The unit reads input values. It runs the program according to the data uploaded to its memory, and transfers the results to the output. S7 1200 CPUs are produced in a compact form. In other words, certain number of digital inputs/outputs, analog input, rapid numerator modules are in the same body with the CPU. If required, signal modules (SM) and communication modules (CM) may be placed on the right and left side of the CPU, respectively; furthermore, signal boards (SB) acting as the signal module may be placed on the body of the CPU.

#### **4.1.2.4. Programming Unit**

To upload a program to PLC memory and to run it, a programming unit is needed. Programming unit may be a microprocessor based special hand device or a software that has been uploaded to a general-purpose personal computer. This unit is also used to write the program, transfer it to PLC, and monitor and change the statuses of input/output, numerator, timer and certain recorders on the data memory, while operating. Today, it is common to use special software uploaded to general-purpose personal computers. This software is a special program that can easily be used and adapted by the persons who are engaged in control circuits.

#### **4.1.2.5. Power Supply**

This module provides the energy needed for PLC to operate. It reduces external source feeds to PLC's internal voltage levels. Output current of the power supply may vary based on the system to be controlled by the PLC. Some PLCs do not need an external power supply.

#### **4.1.2.6. Communication Modules**

Communication modules such as Profibus, Profinet, RS232, RS485 and GPRS are used to communicate with other PLCs or devices.

#### **4.1.3. Programming Software**

Software means a set of commands that notify the hardware (electronic components comprising the PLC) about how to react under certain situations and ensure communication between the programming device and hardware. Each hardware has a unique software. S7 1200 PLCs are programmed via TIA PORTAL programming software. Today, the same software can be used to program S7-300, S7-400 and S7-1500 type CPUs, Operator Panels, SCADA and drivers as well.

There are two types of programming: linear and structural. In linear programming, commands are entered based on their sequence in the program memory. After processing the command in the final line, the program proceeds with the first line. It means that there is a continuous loop. The cycle time means the time required to process all commands once. This time depends on the number and types of commands. It should be as short as possible. Linear programming is generally used for simple and non-extensive programs where the whole program is written on a single program block.

While processing, the structural program is divided into small logical blocks (sub-programs such as FB, FC) according to the functions. An organization program

(MAIN\_OB1) is created to call all of these sub-programs in an order. Fixed data to be used in the program is stored in the data blocks.

TIA PORTAL programming software offers miscellaneous programming options. Its programming languages include LAD (ladder), FBD (Function Block Diagram), STL (Statement List), and SCL( Structured Control Language) [32,33].

## **4.2. SCADA**

SCADA is the abbreviation of Supervisory Control and Data Acquisition. SCADA system is a general name for systems that supervise and monitor the devices distributed in a wide area, process them using a predetermined logic, and store old data, from a single center using a computer.

In short, SCADA system provides Monitoring, Consultation, Control, and Data Collection functions. Thanks to SCADA control system, which is a control and supervision system using an extensive and integrated database, all equipment belonging to a facility or organization can be controlled, production can be planned; furthermore, all units from environment control units to auxiliary organizations can be automatically controlled and supervised. SCADA provides the opportunity to gradually meet all kinds of control-related needs of various organizations.

In simple terms, the SCADA system structure contains a server with the software; PLC or RTU supervising the actuators in the field and receiving information from the servers; sensors and actuators in the field; and finally cards and cables ensuring supervision of and communication between field and PLC/RTU and PLC and server. Such communication function varies according to the communication environment and the needs of SCADA system that uses cables and cards. It should be noted that the devices will communicate with each other within the system installed in the field. Hence, the system components should be compatible.

SCADA basically consists of three parts:

1- Remote Terminal Unit (RTU or PLC): This is the system that form data collection and control terminal units.

2- Communication System: This system sends data or message from one region to another.

3- Master Terminal Unit (MTU): It can be defined as the unit where the facilities distributed in a wide geographical area are remotely controlled, monitored and managed via a computer based structure.

What to expect from SCADA System:

- To monitor electrical and industrial parameters of the system on PC;
- To run the alarm for the values set;
- To record desired values at predetermined intervals;
- To monitor Graphical Trend;
- To records the trends according to date and time;
- An energy saver database;
- Product-oriented energy cost;
- To invoice electric consumption;
- To control the loads of shops, offices, groups and buildings from a single center;
- Priority load rejection and load input;
- To track malfunctions;
- To control each point of the system using PC.

#### **4.2.1. Control Units of the SCADA System**

##### **4.2.1.1. MTU (Master Terminal Unit)**

MTU can be defined as the unit where the facilities distributed in a wide geographical area are remotely controlled, monitored and managed via a computer based structure. MTUs are generally established in a central place between SCADA systems and the facilities to be controlled. This is significant in terms of the

communication performance of the SCADA system. Single or multiple MTUs can be installed. In major systems, sub-MTUs are also installed under the main MTU.

SCADA System Master Terminal Unit ensure coordination of RTUs distributed to a large area in the SCADA system, interpretation of the information given by RTUs and delivery of the same to the users. The unit further transmits the user statistics to RTUs, and perform central control functions

#### **4.2.1.2. RTU (Remote Terminal Unit)**

RTU is the unit that undertake data collection and supervision tasks. RTU is a SCADA unit that collects and stores information on the system variables of the center, sends such information to the control center via a certain communication environment, and implements the commands sent by the MTU [4].

RTUs that form local measurement and control points in the SCADA system can be used to control various interconnected devices, cutters and jigs. The tasks of RTU, a unit acting as an Information Collection and Supervision Unit, are given below.

- 1- To collect and store data
- 2- To control and command, when required
- 3- Monitoring
- 4- To determine and isolate the place where malfunction is observed

#### **4.2.2. Network Topology**

Communication infrastructure, which has a significant role in the development process of SCADA systems, has reached to third generation thanks to the technological opportunities. First generation SCADA systems with no ethernet technology had a monolithic and centralized supervision/control structure; while the second generation had communication, supervision/control and process equipment connected to each other through local area network (LAN). On the other hand, in the

third generation, various SCADA functions are totally distributed and created in line with the open system architecture.

Topology covers the connection between the equipment creating the network, the devices to be used, cabling standards, selection of communication protocol and applicability of such protocols to the network structure.

Linear Line Topology: All components forming the network is aligned on a single line. Sent signal travels through all stations. Each station controls the signal address and processes only the one that belongs to it.

Star Topology: All components of the system are allowed to communicate over a central station. Central processor also serves as a signal amplifier.

Ring Topology: All components are connected to the ring either directly or indirectly through transfer cable. Signals are transferred from a component to the other. Each component receives the signal coming from the line, processes the one belonging to it and gives back to the line amplifying them.

Hierarchical Topology: Components here are located as hierarchical segments. Each segment has a star topology within itself, and the segments are at linear topology with the central processor [34-37].

## CHAPTER 5

### IMAGE BASED CONVEYER BELT APPLICATION

In the prototype, plastic bottle caps proceeding on a conveyor band are classified according to their geometrical shapes (crushed/intact) and colors (blue, red, and yellow); and the robotic arm is used to drop the caps to right boxes. This section will introduce the materials used, and explain the operating scenario and algorithm of the system.

#### 5.1. MATERIALS USED IN THE SYSTEM

The materials used in the system are briefly explained below.

##### 5.1.1. Webcam

An HD webcam was used to process the image. Resolution is 1080p.



Figure 5.1. HD web-cam.

##### 5.1.2. Bunker

This system drops the caps used in the system to conveyor belt, using 1 nema 17 step motors. Plastic pieces of the bunker were drawn on Solidworks as a solid material, and then produced with a 3D printer (100 micron, 25% density).

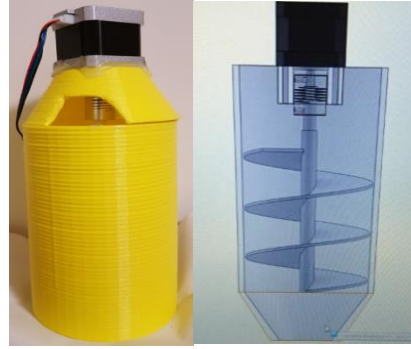


Figure 5.2. Bunker.

### 5.1.3. Arduino Microcontroller

To control the system, Arduino Mega 2560 brand/model microcontroller was used. This card is used to compare the signals received from the image processing interface, and to control conveyor belt and robotic arm movements. A DC motor with reducer, two MZ80 object detection sensors and four servo motors were connected to the Arduino card.

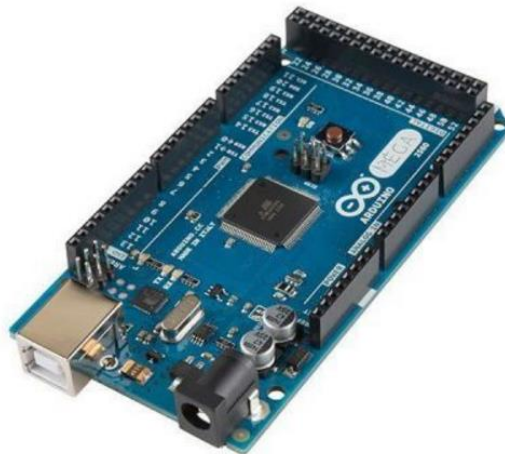


Figure 5.3. Arduino Mega 2560.

### 5.1.4. Sensors

In order to detect the places where the caps lay, MZ80 object detection sensor was placed in front of the camera and robotic arm, respectively. MZ80 is a digital sensor which sends signal 1 when there is an object detected, and 0 when there is no object.



It is an industrial, infrared sensor with a digital output and 80 cm range.



Figure 5.4. MZ80 Object detection sensor.

### 5.1.5. Robotic Arm

Robotic arm has three axes. The movement is ensured through four servo motor. In the light of the information given in chapter 3, The robotic arm is manufactured with 6 degree of freedom, articulated, kuka type. It was produced with a 3D printer (100 micron, 25% density). Its duty is to drop the caps to the correct box.



Figure 5.5. Robotic arm.

### 5.1.6. Conveyor Belt

Movement of the conveyor belt (1400x350x10 mm) is provided by a DC motor with a reducer. It is white and made of PVC. The color white was preferred because it produces less flash while processing the image.

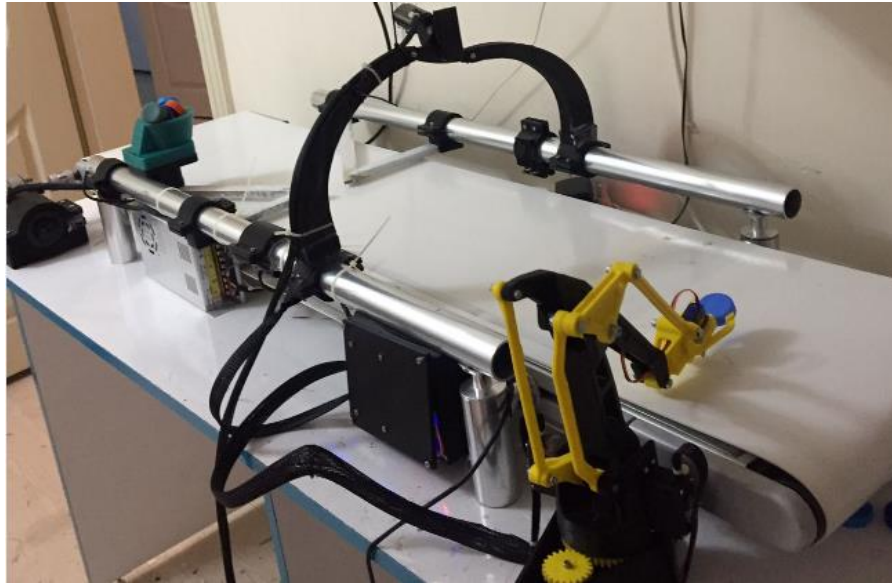


Figure 5.6. Conveyor belt.

### 5.1.7. PLC Used in the System

Brand/model of the PLC used is Siemens S7-1200 CPU 1215C DC/DC/DC. Its task is to control the step motor on the bunker, and track and control the system through SCADA screen.

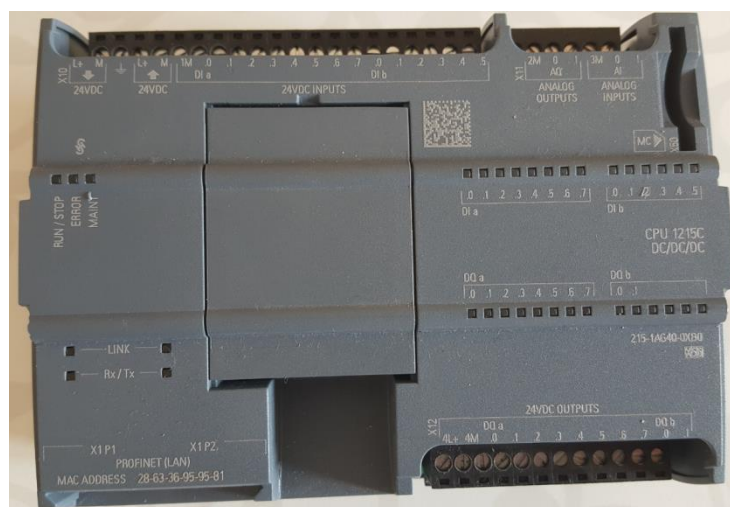


Figure 5.7. PLC used in the system.

## 5.2. PROTOTYPE IMPLEMENTATION

Operation scenario of the prototype developed to classify plastic bottle caps is as follows: The caps are randomly taken from the bunker, and individually placed on the conveyor belt. Provided that several caps drop at the same time, the caps are lined up on the conveyor using the mechanical device. When the lid proceeding on the conveyor belt is detected by the sensor, the conveyor belt is stopped. The lid is scanned by the camera and assessed according to the geometric shape and color. Data is sent to Arduino card. Caps that have a different shape compared to the geometry of a predefined intact lid are defined as crushed, and placed into the “crushed lid” box by the robotic arm without considering the color. Intact caps are grouped according to their colors and placed into the box bearing the same color. The conveyor is stopped again by the second sensor before robotic arm places the caps into concerned boxes. The sensors are required to determine a reference position point for the camera and robotic arm. After completing the procedures regarding the first lid, the same process continues periodically for the other caps. This study used three different colors of caps: blue, red and yellow.

Figures 5.8, 5.9 and 5.10 illustrate the system model, flow card and block diagram.

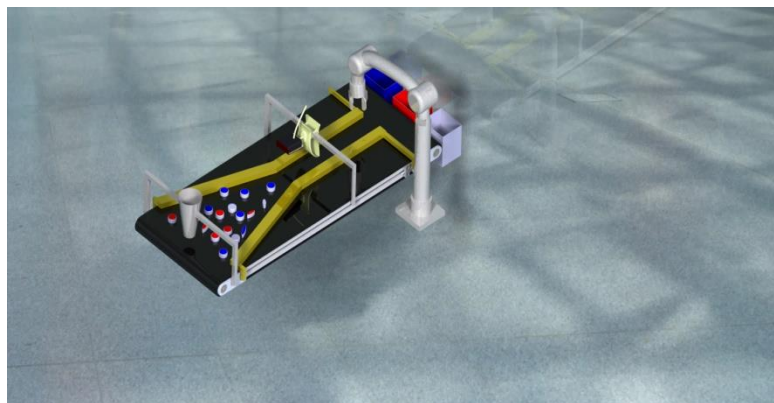


Figure 5.8. Model study.

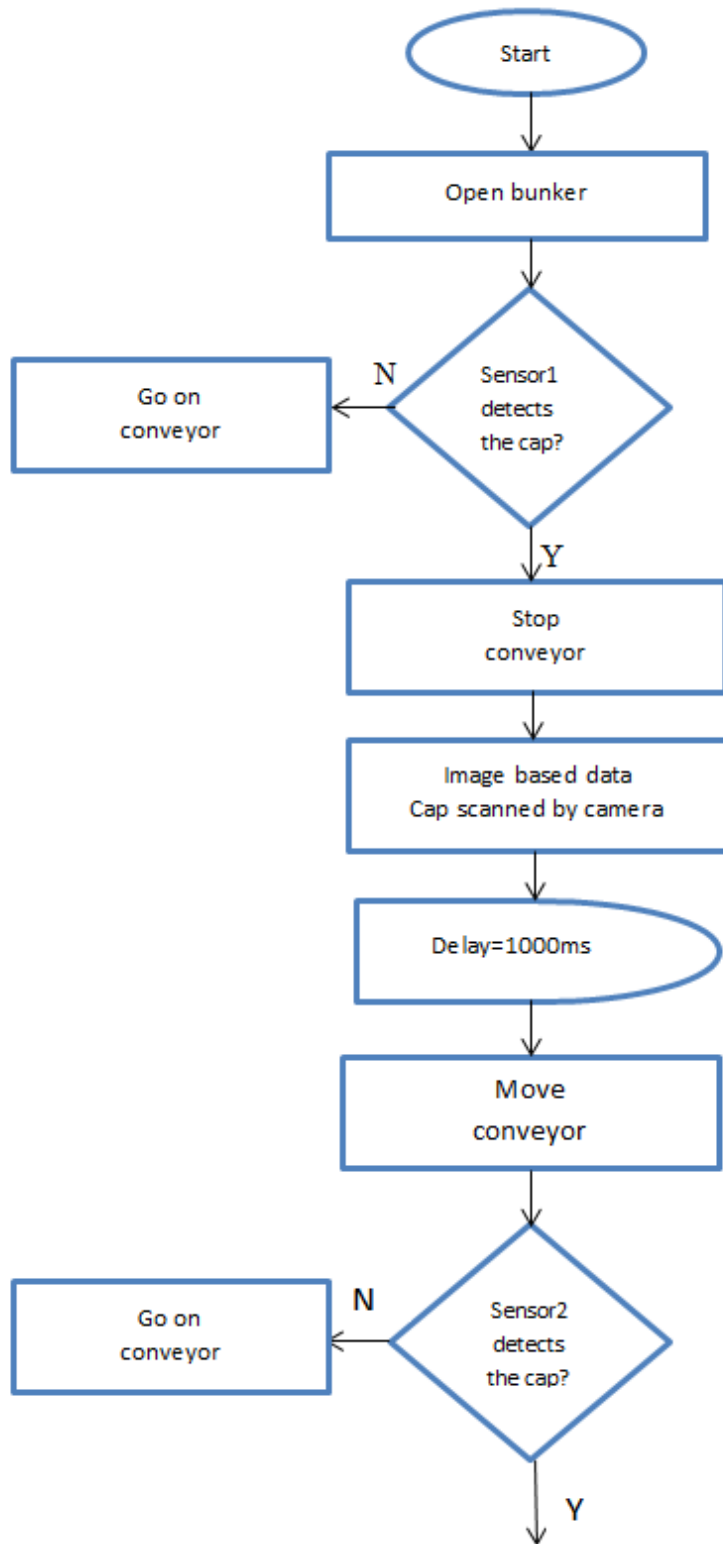


Figure 5.9. Flow card.

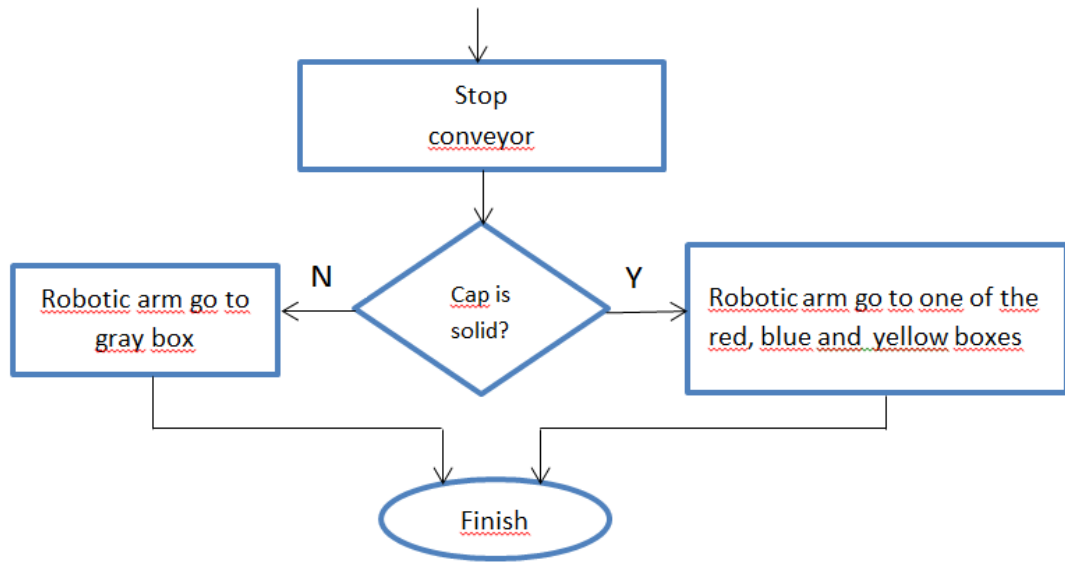


Figure 5.9. (continuing).

There are two object detection sensors in the system. These sensors provide digital outcome (1 when an object is detected and 0 when there is no object). First sensor controls the area in front of the camera that processes the images. Second sensor controls the area in front of the robotic arm. The task of the sensors is to detect the critical positions of the caps laying on the conveyor belt. Accordingly, the first sensor sees in front of the camera on the conveyor belt. When the first sensor detects an object, the conveyor belt stops. The camera assigns dimension of the system, then it calculates the mean value and applies color space filtration. Then through a shape analysis, it is determined whether the object is a circle. If the object is not a circle, color is not taken into consideration. For circle-shaped objects, color analysis is initiated based on the average of pixels on the lid. Then the data on the lid, which has been analyzed in terms of color and shape, is transferred to Arduino. Based on the lid data, Arduino turns the output pins into high and informs the PLC. At this moment conveyor belt resume movement. When the caps on the conveyor belt reach the robotic arm, the belt is stopped by the second object detection sensor. Based on the data it received, Arduino directs the robotic arm and the caps are piled in related boxes.

Bunker and robotic arm used in the prototype are printed with a 3D printer by the research team. The system has a total of six motors: one DC motor with reducer, one

step motor and four servo motors. DC motor with reducer moves the conveyor, step motor opens/closes the bunker, and servo motors move the robotic arm. Step motor is controlled by the PLC; while, Arduino is used to control DC motor and servo motors.

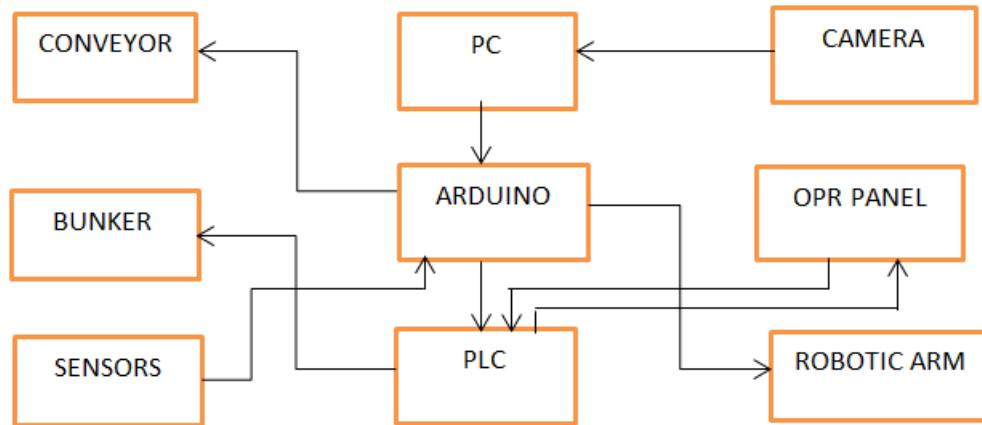


Figure 5.10. Block diagram of the system.

Image processing, which is a significant part of the system, is conducted as follows: Photos of the caps on the conveyor are taken thanks to an interface program written in C# programming language, and two basic filters are applied. These filters are used to transform the image into a gray hue and eliminate noises. After these two basic filters, color detection is applied; and the object selectivity is ensured based on the color codes attached by the library to three basic colors and hues. On the other hand, while determining the size and shape of the objects, clustering of the image filtered, i.e. the pixels filtered, should be taken into consideration. Clustering should be determined using matrix method.

Industrial smart cameras or PC based webcams are used to process the images. For this project, webcam was preferred as it is very cost effective compared to the other option.

The procedures are conducted as follows in the interface program:

- 1- Data sent by the camera was transferred to the interface. (Image Source)

- 2- Pre-filter was applied on the image source through resizing, median and HSL filters. The objective was to minimize the probable handicaps on the image.
- 3- Shape Checker method was used to detect whether there was any circle (lid geometry) in the image.
- 4- If there was a lid geometry, average color density of the pixels in the lid were calculated to find the lid color, this color was compared to HSL color model, and the lid color was determined based on hue value.
- 5- Lid color was anticipated by comparing the average hue value to certain value ranges.

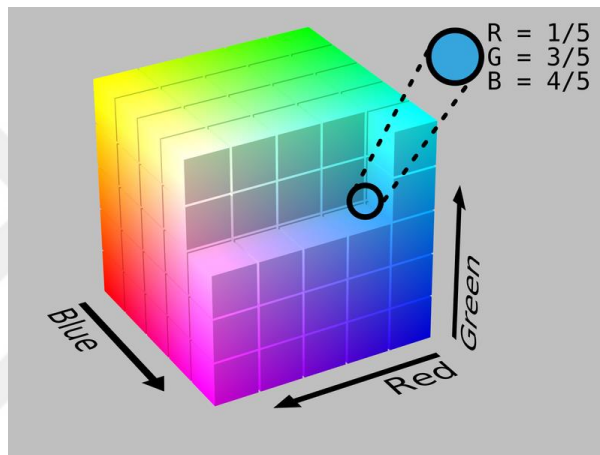


Figure 5.11. RGB hue cube.

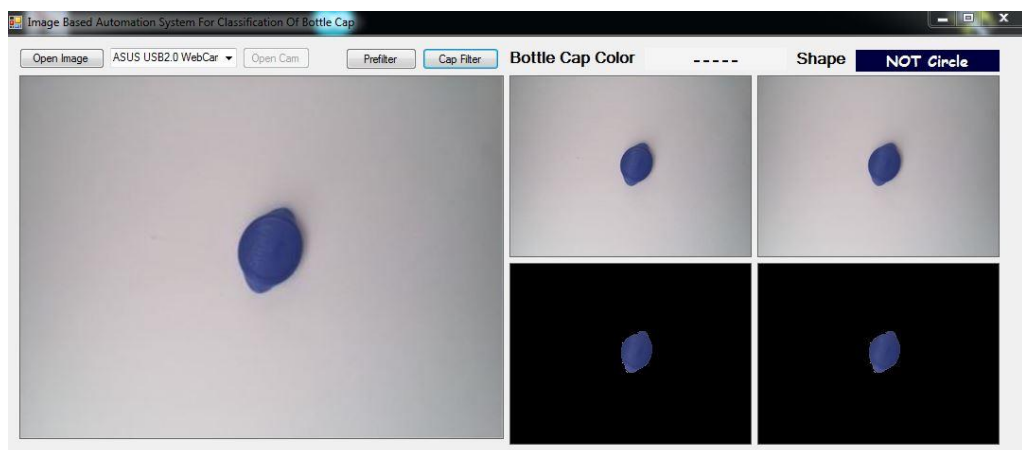


Figure 5.12. Image processing interface program - determination of geometric shape.

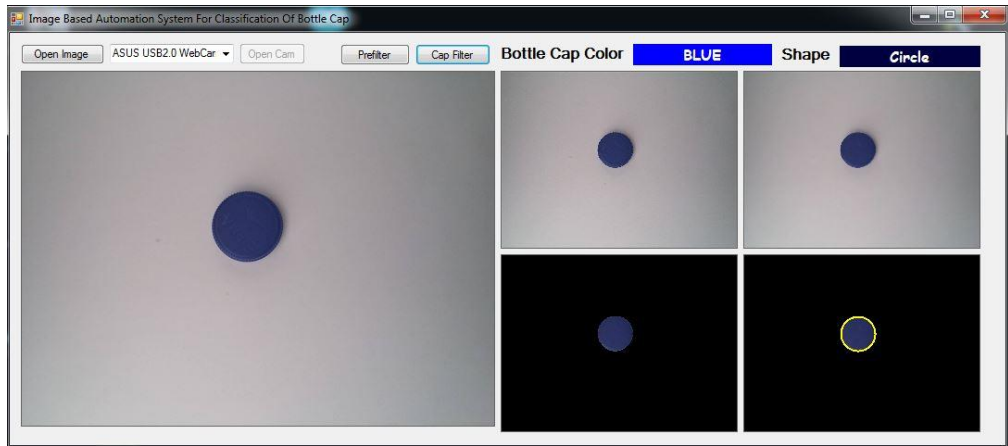


Figure 5.13. Image processing interface program - determination of the color blue.

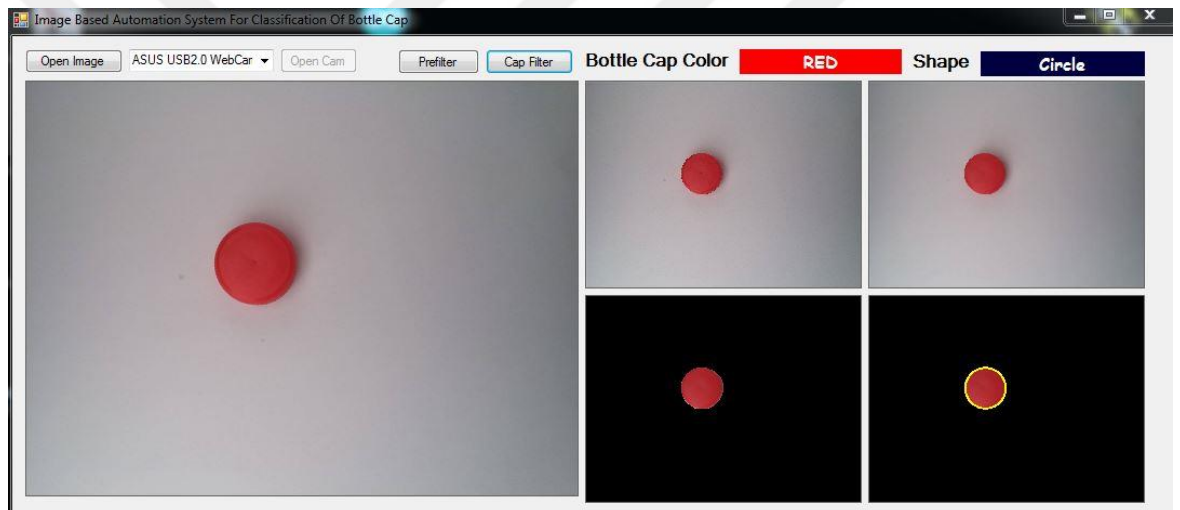


Figure 5.14. Image processing interface program - determination of the color red.

As seen in Figures 5.12, 5.13 and 5.14, the interface program received images from the camera, processes the images in line with the algorithm determined above, and produces an accurate output. This output is sent to Arduino card via usb.

Servo motors on the robotic arm are controlled by Arduino. Robotic arm has three MG 995 and one SG 90 type servo motors. SG90 provides jaw movements; while MG995s offer axis movements. Camera and C# program determine the robustness and color of the caps, and the data is sent to Arduino, which sends the data to PLC. Based on digital signals taken from Arduino, on the SCADA screen, it can be monitored all of movement of the system. When the caps are detected by the second



sensor, the conveyor belt is stopped. This point where caps lay is taken as the reference point for location information. Robotic arm movements are always assigned according to this reference point.

PLC software was developed in ladder programming language. PTO function is used to control step motor. The reference point from which the lid has been taken and the location of the box where the lid is to be placed are steady. Hence there are 4 different pre-defined routes to be taken based on the durability and color of the lid (crushed caps, blue caps, red caps, and yellow caps). Robotic arm constantly move through one of these four routes. Prototype overview is shown in Figure 5.15.



Figure 5.15. System overview.

SIMATIC HMI Application/WinCC RT Advanced was used as the operator panel. No real (physical) operator panel was used.

The system can be seen on the SCADA screen. SCADA screen provides the opportunity to monitor system operation online. It can be used to track real time bunker start-up/shut down, plastic lid movements (proceeding), robotic arm movements, and placement of plastic caps into boxes. It further gives the opportunity to start and stop the system.

Two SCADA screens are designed. The first screen shows the working statements of the elements on the prototype and the number of caps piled in the boxes. The running of the system elements are shown in green and the going off in red. In addition, the system is started and stopped and reset from this screen. On the second screen, on-line animation of the movements of the caps on the conveyor belt and placing it to the boxes by the robot arm is shown. R1, R2, R3 and R4 represent the four servo motors on the robotic arm.

In Figure 5.16 and Figure 5.17, the screen 1 of the SCADA are shown in the initial state and after the caps are piled in the boxes, while in Figure 5.18, the screen 2 of SCADA is displayed.

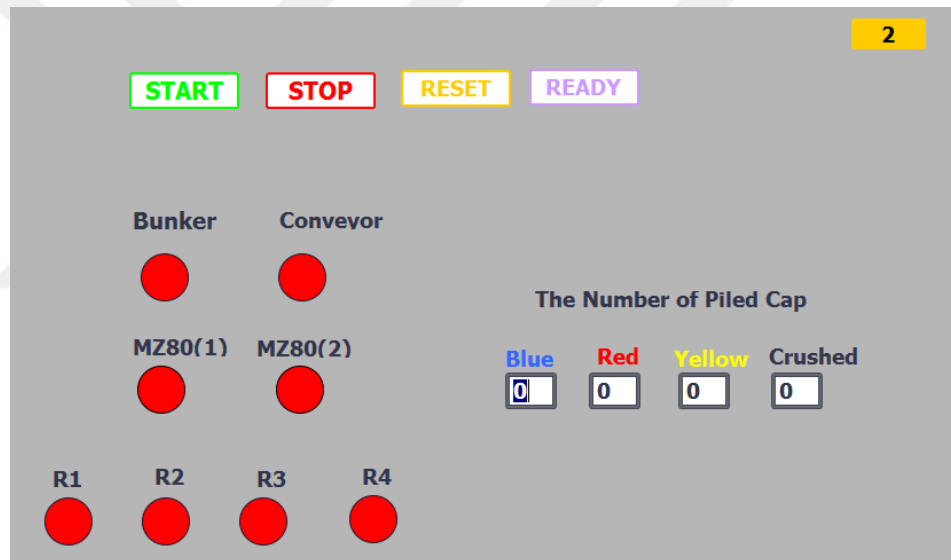


Figure 5.16. SCADA screen 1-initial state.

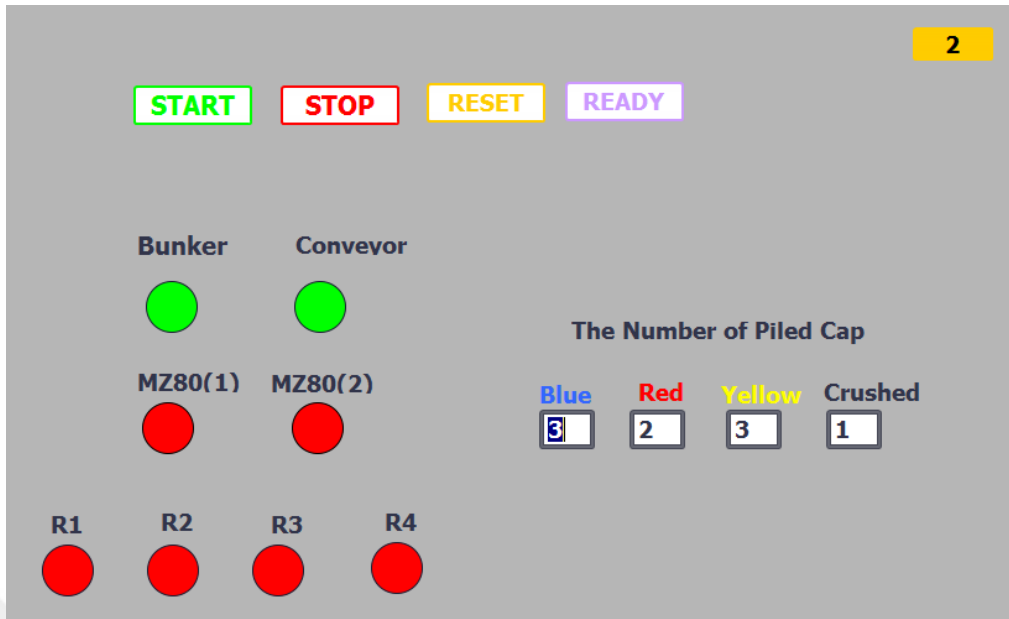


Figure 5.17. SCADA screen 1-after placed caps.

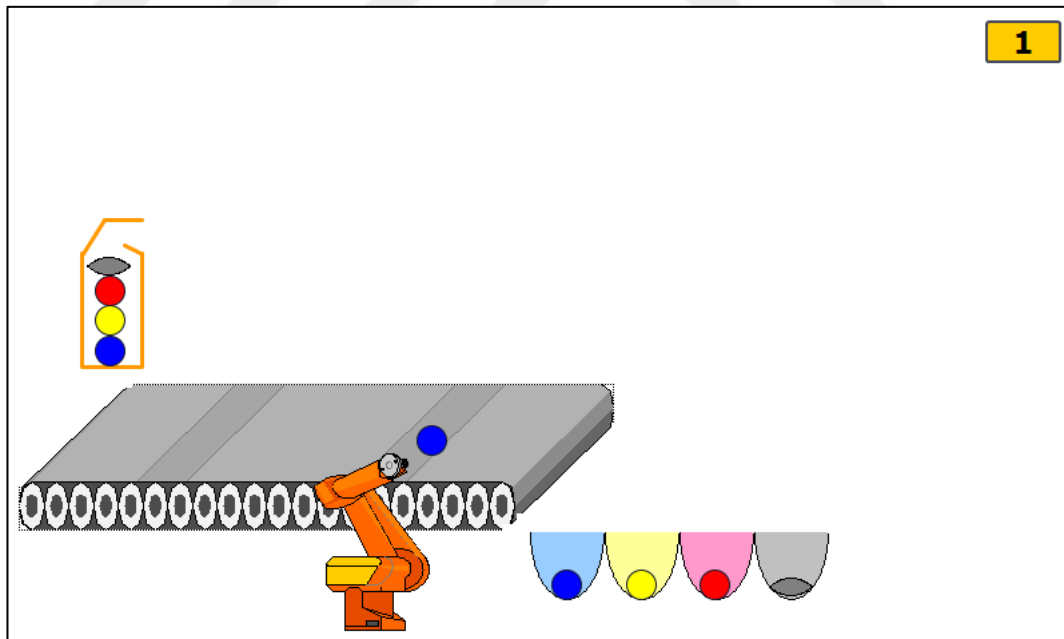


Figure 5.18. SCADA screen 2.

## **CHAPTER 6**

### **CONCLUSION**

Within the scope of this thesis, an image-based industrial automation system prototype was created. To run the system, C# program was used to process images; while, Arduino and PLC microcontrollers were used for other controls, both for hardware and software purposes. Both Arduino and PLC were used as a controller because the prototype will be used as a training material. The objective is to combine PLC system with the Arduino embedded system, which is highly popular among university students, with the aim of providing an opportunity to make a comparison and producing a different viewpoint. It was observed that optimization of study scenario (story) and algorithm was very important for rapid and effective operation of industrial automation systems.

This project will be used at universities and other schools that offer vocational - technical education, and will make great contributions to the laboratory training of students as it will provide the opportunity to observe operating logic of an image-based industrial automation system. As a training tool, the prototype can be used in certain classes such as industrial automation, special electric machines etc.

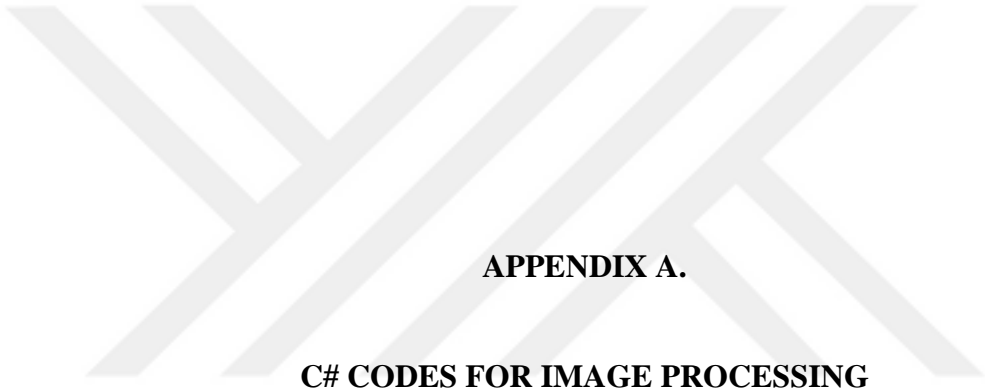
The prototype is open to development in terms of hardware and software. System limitations are expected to be addressed in the further studies. Different image processing procedures will be tested and the results will be observed. The robot used in this study moves in a linear direction and holds/drops the objects. Feedback visual servo control applications are intended to be conducted in the future studies.

## REFERENCES

1. Abdu, I. A. and Taleb M. M., “Arhitecture of industrial automation systems”, *European Scientific Journal*, 10 (3): 273–283 (2014).
2. Internet: EE IIT, Kharagpur, “Arhitecture of Industrial Automation Systems”, [http:// nptel.ac.in/courses/108105063/pdf/L-02\(SM\)\(IA&C\)%20\(\(EE\)NPTEL\).pdf](http://nptel.ac.in/courses/108105063/pdf/L-02(SM)(IA&C)%20((EE)NPTEL).pdf) (2017).
3. Arias M. M., Hawwary M. I. and Scherpen J. M. A., “Image-based visual servo control using the port-Hamiltonian approach”, *IFAC-PapersOnLine*, 48 (13): 105–110 (2015).
4. Dong G. and Zhu Z. H., “Autonomous robotic capture of non-cooperative target by adaptive extended Kalman filter based visual servo”, *Acta Astronautica*, 122, 209–218 (2016).
5. Hamel T. and Mahony R., “Image based visual servo control for a class of aerial robotic systems”, *Automatica*, 43, 1975 – 1983 (2007).
6. Deniz C., “Image processing based control in conventional robotic lines in automotive industry”, M Sc Thesis, *Kocaeli University Graduate School of Natural and Applied Sciences*, Kocaeli, (2011).
7. Rahimi A., “ Visual servo-control application in a humanoid robot using depth-camera information”, M Sc Thesis, *ITU University Graduate School of Natural and Applied Sciences*, İstanbul, (2014).
8. Lippiello, V., Siciliano, B. and Villani, L., “Position-based visual servoing in industrial multirobot cells using a hybrid camera configuration”, *IEEE Transactions On Robotics*, 23 (1):73-86 (2007).
9. Motai, Y., Kosaka, A., “Hand–eye calibration applied to viewpoint selection for robotic vision”, *IEEE Transactions on Industrial Electronics*, 55 (10): 3731-3741 (2008).
10. Xie, W., Li, Z., Tu, X-W., Peron, C., “Switching control of image-based visual servoing with laser pointer in robotic manufacturing systems”, *IEEE Transactions on Industrial Electronics*, 56 (2): 520-529 (2009).
11. Deng, F.L., Janabi-Sharifi, W., and Wilson, J., “Hybrid motion control and planning strategies for visual servoing”, *IEEE Transactions On Industrial Electronics*, 52 (4): 1024-1041 (2005).

12. Golnabi, H., and Asadpour, A., "Design and application of industrial machine vision systems", *Robotics and Computer-Integrated Manufacturing*, 23 (6): 630- 637, (2007).
13. Sulzer, J. and Kova C., I., "Enhancement of positioning accuracy of industrial robots with a reconfigurable fine-positioning module", *Precision Engineering*, 34 (2): 201–217, (2010).
14. Selver, M., Akay, O., Alim, F., Bardak, S. and Olmez , M., "An automated industrial conveyor belt system using image processing and hierarchical clustering for classifying marbleslabs", *Robotics And Computer-Integrated Manufacturing*, 27 (1): 164–176 (2011).
15. Ryberg, A., Christiansson, A. K., Eriksson, K. and Lennartson, B., "A new camera model and algorithms for higher accuracy and better convergence in vision-based pose calculations", *Proceedings Of The 2006 IEEE International Conference On Mechatronics And Automation*, Luoyang, China. June 25 – 28, (2006).
16. Fawaz, K., Merzouki, R. and Ould-Bouamama, B., "Model based real time monitoring for collision detection of an industrial robot", *Mechatronics*, 19 (5): 695–704 (2009).
17. Rousseau, P., Desrochers, A. and Krouglicof, N., "Machine Vision System for the Automatic Identification of Robot Kinematic Parameters", *IEEE Transactions on Robotics and Automation*, 17 (6): 972-978 (2001).
18. Gonzales R. C. and Woods R. E., "Digital Image Processing 3rd ed.", *Pearson Education Inc. Publishing Prentice Hall*, New Jersey, 48-96, 105-158, 395-454 (2008).
19. Pratt W. K., "Digital Image Processing 3rd ed.", *A Wiley-Interscience Publication*, New York, 3-39, 91-136, 401-435 (2001).
20. Uçan N. O., Osman O. and Albora A. M. "Image Processing Techniques and Engineering Applications", *Nobel Yayın Dağıtım*, Ankara, 1-6 (2006).
21. [http://www.ibrahimcayiroglu.com/Dokumanlar/GoruntuIsleme/Goruntu\\_Isleme\\_Ders\\_Notlari-1.Hafta.pdf](http://www.ibrahimcayiroglu.com/Dokumanlar/GoruntuIsleme/Goruntu_Isleme_Ders_Notlari-1.Hafta.pdf)
22. Internet: Ford A. and Robberts A., "Colour Space Conversions" <http://sites.biology.duke.edu/johnsenlab/pdfs/tech/colorconversion.pdf> (2017).
23. Bingül Z. and Küçük S. "Robot Kinematics", *Umuttepe Yayınları*, Kocaeli, 1-41, 104-179 (2015).
24. Gün A. and Gök K. "Basics of Automation and Robot Control", *Seçkin Yayıncılık*, Ankara, 77-97 (2011).

25. Rocha C.R., Tonetto C.P. and Dias A., “A comparison between the Denavit-Hartenberg and the screw-based methods used in kinematic modeling of robot manipulators”, *Robotics and Computer-Integrated Manufacturing*, 27 (4): 723-728 (2011).
26. Li Z. and Schicho J., “Denavit-Hartenberg parameters of 6R linkages that are necessary for movability” *Mechanism and Machine Theory*, 94: 1-8 (2015).
27. Raghavan M. and Roth B., “Kinematic analysis of the 6R manipulator of general geometry” *International Symposium Robotics*, 314-320 (1989).
28. Raghavan M. and Roth B., “Inverse kinematic of the general 6R manipulator and related linkages”, *ASME Transaction, Journal of Mechanical Design* 502-508 (1993).
29. Küçük S., “Modelling and Off-Line programming of industrial robots”, Ph D Thesis, *Kocaeli University Graduate School of Natural and Applied Sciences*, Kocaeli, (2004).
30. Sugiarto I. and Conradt J., “A model-based approach to robot kinematics and control using discrete factor graphs with belief propagation”, *Robotics and Autonomous Systems*, 91, 234-246 (2017).
31. Wei Y., Jian S., He S. and Wang Z., “General approach for inverse kinematics of nR robots”, *Mechanism and Machine Theory*, 75, 97-106 (2014).
32. Eminoglu Y., “PLC Programming and S7 1200”, *Birsen Yayınevi*, İstanbul, 15-25, 403-405 (2015).
33. Internet: Siemens, “Simatic”, <https://support.industry.siemens.com/cs/document/39710145/simatic-s7-1200-easy-book?dti=0&lc=en-WW> (2017).
34. Internet: DHS, “Guide”, <https://www.dhs.gov/sites/default/files/publications/csd-nistguidetosupervisoryanddataacquisitionscadaandindustrialcontrolsystemssecurity-2007.pdf> (2017).
35. Choi D., Lee S., Won D. And Kim S. “Efficient secure group communications for SCADA”, *IEEE Trans. Power Del.* 25 (2): 714–722 (2010).
36. Kang D., Lee J., Kim B. and Hur D. “Proposal strategies of key management for data encryption in SCADA network of electric power systems”, *Int. J. Electr. Power Energy Syst.* 33 (9):1521–1526 (2011).
37. Rezai A., Keshavarzi P. and Moravej Z. “Advance Hybrid Key Management Architecture for SCADA Network Security”, *Secur. Commun. Netw.* (2016).



**APPENDIX A.**  
**C# CODES FOR IMAGE PROCESSING**



```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Drawing.Imaging;
using System.Collections;
using System.Reflection;

// add api[DllImport("user32.dll")]
using System.Runtime.InteropServices;

// aforge image processing dll
using AForge.Video;
using AForge.Video.DirectShow;
using AForge;
using AForge.Imaging;
using AForge.Imaging.Filters;
using AForge.Imaging.Textures;
using AForge.Math.Geometry;

namespace Cap1
{
    public partial class Form1 : Form
    {

        string RxString;

        // Aforge image processing codes

```

```

// Image Capture Devices in Video
private FilterInfoCollection CaptureDevices;
private VideoCaptureDevice FinalFrame;

// Bitmap operations
private System.Drawing.Bitmap image;
private System.Drawing.Bitmap sourceImage;
private System.Drawing.Bitmap filteredImage;

private String CapColour = "no";
private Boolean CapSolid = false;

sc_operation can_operation= new sc_operation();

public Form1()
{
InitializeComponent();
}

// Loading form
private void Form1_Load(object sender, EventArgs e)
{
// Aforge image processing codes - Video input control
CaptureDevices = new FilterInfoCollection(FilterCategory.VideoInputDevice);

foreach (FilterInfo Device in CaptureDevices)
{
comboBox1.Items.Add(Device.Name);
}
try
{

```

```

comboBox1.SelectedIndex = 0;
button1.Enabled = true;
}
catch
{
MessageBox.Show("There is no Camera!");
}
}

```

```

// Aforge image processing codes - New frame
void FinalFrame_NewFrame(object sender, NewFrameEventArgs eventArgs)
{
pictureBox1.Image = (Bitmap)eventArgs.Frame.Clone();
}

```

```

private void button1_Click(object sender, EventArgs e)
{
FinalFrame = new
VideoCaptureDevice(CaptureDevices[comboBox1.SelectedIndex].MonikerString);
FinalFrame.NewFrame += new NewFrameEventHandler(FinalFrame_NewFrame);
FinalFrame.Start();
button2.Enabled = true;
button4.Enabled = true;
}

```

```

private void ApplyFilter(IFilter filter)
{
// apply filter
filteredImage = filter.Apply(sourceImage);
}

```

```

}

private void button2_Click(object sender, EventArgs e)
{
if(button1.Enabled == true) if (FinalFrame.IsRunning) FinalFrame.Stop();

sourceImage = (Bitmap)pictureBox1.Image.Clone();
ApplyFilter(new ResizeBicubic(256,192));
pictureBox2.Image = filteredImage;

sourceImage = filteredImage;
ApplyFilter(new Median());
pictureBox3.Image = filteredImage;

sourceImage = filteredImage;
// create filter
HSLFiltering HSLfilter = new HSLFiltering();
// set color ranges to keep
HSLfilter.Hue = new IntRange(0, 400);
HSLfilter.Saturation = new Range(0.2f, 1);
HSLfilter.Luminance = new Range(0.1f, 1);
// apply the filter
ApplyFilter(HSLfilter);

sourceImage = filteredImage;
// and to picture box
pictureBox4.Image = sourceImage;

// UpdatePictureBoxPosition();

if (button1.Enabled == true) FinalFrame.Start();

}

```

```

private void Form1_FormClosed(object sender, FormClosedEventArgs e)
{
    if (button1.Enabled == true) if (FinalFrame.IsRunning) FinalFrame.Stop();

}

private void button3_Click(object sender, EventArgs e)
{
    button1.Enabled = false;
    openFileDialog1.Filter = "Tüm Resim Dosyaları| +
    *.bmp;*.jpg;*.gif;*.wmf;*.tif;*.png";
    openFileDialog1.DefaultExt = ".jpg";
    openFileDialog1.Title = "Resim Açma Ekranı";
    if (openFileDialog1.ShowDialog() == DialogResult.OK)
    {
        sourceImage = (Bitmap)Bitmap.FromFile(openFileDialog1.FileName);
        // check pixel format
        if ((sourceImage.PixelFormat == PixelFormat.Format16bppGrayScale) ||
            (Bitmap.GetPixelFormatSize(sourceImage.PixelFormat) > 32))
        {
            MessageBox.Show("The demo application supports only color images.", "Error",
            MessageBoxButtons.OK, MessageBoxIcon.Error);
            // free image
            sourceImage.Dispose();
            sourceImage = null;
        }
        else
        {
            // make sure the image has 24 bpp format
            if (sourceImage.PixelFormat != PixelFormat.Format24bppRgb)
            {

```

```

Bitmap temp = AForge.Imaging.Image.Clone(sourceImage,
PixelFormat.Format24bppRgb);
sourceImage.Dispose();
sourceImage = temp;
}
}

```

```

pictureBox1.Image = null;

```

```

// display image

```

```

pictureBox1.Image = sourceImage;
button2.Enabled = true;
button4.Enabled = true;
}
}

```

```

// Conver list of AForge.NET's points to array of .NET points

```

```

private AForge.Point[] ToPointsArray(List<IntPoint> points)

```

```

{
AForge.Point[] array = new AForge.Point[points.Count];

```

```

for (int i = 0, n = points.Count; i < n; i++)

```

```

{
array[i] = new AForge.Point(points[i].X, points[i].Y);
}

```

```

return array;

```

```

}

```

```

private void button4_Click(object sender, EventArgs e)

```

```

{
CapColour = "no";

```

```
CapSolid = false;
```

```
if (DaireMi((Bitmap)pictureBox4.Image.Clone()))
{
    textBox2.Text = "Circle";
    int Hue=Convert.ToInt32(textBox1.Text);
    textBox1.ForeColor = System.Drawing.Color.White;
    if ((Hue >= 330 && Hue < 361) || Hue < 30 ) {textBox1.Text = "RED";
    textBox1.BackColor = System.Drawing.Color.Red;}
    else if (Hue >= 30 && Hue < 90) { textBox1.Text = "YELLOW";
    textBox1.BackColor = System.Drawing.Color.Yellow; textBox1.ForeColor =
    System.Drawing.Color.Black; }
    else if (Hue >= 90 && Hue < 150) { textBox1.Text = "GREEN";
    textBox1.BackColor = System.Drawing.Color.Green; }
    else if (Hue >= 150 && Hue < 270) { textBox1.Text = "BLUE";
    textBox1.BackColor = System.Drawing.Color.Blue; }
    else { textBox1.Text = Hue.ToString(); textBox1.BackColor =
    System.Drawing.Color.Black; }

}

else
{
    textBox1.ForeColor = System.Drawing.Color.Black;
    textBox1.BackColor = System.Drawing.Color.WhiteSmoke;
    textBox1.Text = "-----";
    textBox2.Text = "NOT Circle";
}

}
```

```

private Boolean DaireMi(Bitmap bitmap)
{
    Boolean Daire = false;
    Bitmap bitmap1 = bitmap;

    // lock image
    BitmapData bitmapData = bitmap1.LockBits(
        new Rectangle(0, 0, bitmap1.Width, bitmap1.Height),
        ImageLockMode.ReadWrite, bitmap1.PixelFormat);

    // step 2 - locating objects
    BlobCounter blobCounter = new BlobCounter();

    blobCounter.FilterBlobs = true;
    blobCounter.MinHeight = 20;
    blobCounter.MinWidth = 20;

    blobCounter.ProcessImage(bitmapData);
    Blob[] blobs = blobCounter.GetObjectsInformation();
    bitmap1.UnlockBits(bitmapData);

    // step 3 - check objects' type and highlight
    SimpleShapeChecker shapeChecker = new SimpleShapeChecker();

    Graphics g = Graphics.FromImage(bitmap1);
    Pen yellowPen = new Pen(Color.Yellow, 2); // circles

    if (blobs.Length.ToString() != "0")
    {
        for (int i = 0, n = blobs.Length; i < n; i++)
        {
            List<IntPoint> edgePoints = blobCounter.GetBlobsEdgePoints(blobs[i]);
            AForge.Point center;

```



```

float radius;
// is circle ?
if (shapeChecker.IsCircle(edgePoints, out center, out radius))
{
g.DrawEllipse(yellowPen,
(float)(center.X - radius), (float)(center.Y - radius),
(float)(radius * 2), (float)(radius * 2));
// MessageBox.Show("x:" + center.X.ToString() + "y:" + center.Y.ToString() + "r:"
+ radius.ToString());

```

```

byte red = blobs[i].ColorMean.R ;
byte green = blobs[i].ColorMean.G;
byte blue = blobs[i].ColorMean.B;
int Hue = 0;
int max;
int min;
int ort;

```

```

ort = (red + green + blue) / 3;

```

```

max = Math.Max(red, Math.Max(green, blue));
min = Math.Min(red, Math.Min(green, blue));
if (max != min)
{
if (max == red && green >= blue) Hue = 60 * (green - blue) / (max - min) + 0;
if (max == red && green < blue) Hue = 60 * (green - blue) / (max - min) + 360;
if (max == green) Hue = 60 * (blue - red) / (max - min) + 120;
if (max == blue) Hue = 60 * (red - green) / (max - min) + 240;
}

```

```

textBox1.Text = Hue.ToString();

```

```

Daire = true;
}
}
}

yellowPen.Dispose();
g.Dispose();

// put new image to clipboard
//Clipboard.SetDataObject(bitmap);

// and to picture box
pictureBox5.Image = bitmap1;
// UpdatePictureBoxPosition();

return Daire;
}

private void SariCapAra(Bitmap bitmap)
{

Bitmap bitmap1 = bitmap;

// lock image
BitmapData bitmapData = bitmap1.LockBits(
new Rectangle(0, 0, bitmap1.Width, bitmap1.Height),
ImageLockMode.ReadWrite, bitmap1.PixelFormat);

// create filter
HSLFiltering HSLfilter = new HSLFiltering();
// set color ranges to keep
HSLfilter.Hue = new IntRange(0, 120);

```

```

HSLfilter.Saturation = new Range(0.2f, 1);
HSLfilter.Luminance = new Range(0.1f, 1);
// apply the filter
HSLfilter.ApplyInPlace(bitmapData);

// step 2 - locating objects
BlobCounter blobCounter = new BlobCounter();

blobCounter.FilterBlobs = true;
blobCounter.MinHeight = 20;
blobCounter.MinWidth = 20;

blobCounter.ProcessImage(bitmapData);
Blob[] blobs = blobCounter.GetObjectsInformation();
bitmap1.UnlockBits(bitmapData);

// step 3 - check objects' type and highlight
SimpleShapeChecker shapeChecker = new SimpleShapeChecker();

Graphics g = Graphics.FromImage(bitmap1);
Pen yellowPen = new Pen(Color.Yellow, 2); // circles

if (blobs.Length.ToString() != "0")
{
for (int i = 0, n = blobs.Length; i < n; i++)
{
List<IntPoint> edgePoints = blobCounter.GetBlobsEdgePoints(blobs[i]);

AForge.Point center;
float radius;
CapColour = "Yellow";
// is circle ?
if (shapeChecker.IsCircle(edgePoints, out center, out radius))

```

```

{
g.DrawEllipse(yellowPen,
(float)(center.X - radius), (float)(center.Y - radius),
(float)(radius * 2), (float)(radius * 2));
// MessageBox.Show("x:" + center.X.ToString() + "y:" + center.Y.ToString() + "r:"
+ radius.ToString());

CapSolid = true;
}
else
{
CapSolid = false;
}
}
}

yellowPen.Dispose();
g.Dispose();

// put new image to clipboard
//Clipboard.SetDataObject(bitmap);

// and to picture box
pictureBox4.Image = bitmap1;
// UpdatePictureBoxPosition();
}

private void RedCapAra(Bitmap bitmap)
{

Bitmap bitmap1 = bitmap;

// lock image

```

```

BitmapData bitmapData = bitmap1.LockBits(
new Rectangle(0, 0, bitmap1.Width, bitmap1.Height),
ImageLockMode.ReadWrite, bitmap1.PixelFormat);

// create filter
HSLFiltering HSLfilter = new HSLFiltering();
// set color ranges to keep
HSLfilter.Hue = new IntRange(300, 30);
HSLfilter.Saturation = new Range(0.2f, 1);
HSLfilter.Luminance = new Range(0.1f, 1);
// apply the filter
HSLfilter.ApplyInPlace(bitmapData);

// step 2 - locating objects
BlobCounter blobCounter = new BlobCounter();

blobCounter.FilterBlobs = true;
blobCounter.MinHeight = 20;
blobCounter.MinWidth = 20;

blobCounter.ProcessImage(bitmapData);
Blob[] blobs = blobCounter.GetObjectsInformation();
bitmap1.UnlockBits(bitmapData);

// step 3 - check objects' type and highlight
SimpleShapeChecker shapeChecker = new SimpleShapeChecker();

Graphics g = Graphics.FromImage(bitmap1);
Pen yellowPen = new Pen(Color.Yellow, 2); // circles

if (blobs.Length.ToString() != "0")
{
for (int i = 0, n = blobs.Length; i < n; i++)

```

```

{
List<IntPoint> edgePoints = blobCounter.GetBlobsEdgePoints(blobs[i]);

AForge.Point center;
float radius;
CapColour = "Red";
// is circle ?
if (shapeChecker.IsCircle(edgePoints, out center, out radius))
{
g.DrawEllipse(yellowPen,
(float)(center.X - radius), (float)(center.Y - radius),
(float)(radius * 2), (float)(radius * 2));
// MessageBox.Show("x:" + center.X.ToString() + "y:" + center.Y.ToString() + "r:"
+ radius.ToString());

CapSolid = true;
}
else
{
CapSolid = false;
}
}
}

yellowPen.Dispose();
g.Dispose();

// put new image to clipboard
//Clipboard.SetDataObject(bitmap);

// and to picture box
pictureBox4.Image = bitmap1;
// UpdatePictureBoxPosition();

```

```

}

private void BlueCapAra(Bitmap bitmap)
{

    Bitmap bitmap1 = bitmap;

    // lock image
    BitmapData bitmapData = bitmap1.LockBits(
    new Rectangle(0, 0, bitmap1.Width, bitmap1.Height),
    ImageLockMode.ReadWrite, bitmap1.PixelFormat);

    // create filter
    HSLFiltering HSLfilter = new HSLFiltering();
    // set color ranges to keep
    HSLfilter.Hue = new IntRange(0,400);
    HSLfilter.Saturation = new Range(0.2f, 1);
    HSLfilter.Luminance = new Range(0.1f, 1);
    // apply the filter
    HSLfilter.ApplyInPlace(bitmapData);

    // step 2 - locating objects
    BlobCounter blobCounter = new BlobCounter();

    blobCounter.FilterBlobs = true;
    blobCounter.MinHeight = 20;
    blobCounter.MinWidth = 20;

    blobCounter.ProcessImage(bitmapData);
    Blob[] blobs = blobCounter.GetObjectsInformation();
    bitmap1.UnlockBits(bitmapData);

    // step 3 - check objects' type and highlight

```

```

SimpleShapeChecker shapeChecker = new SimpleShapeChecker();

Graphics g = Graphics.FromImage(bitmap1);
Pen yellowPen = new Pen(Color.Yellow, 2); // circles

if (blobs.Length.ToString()!="0")
{
for (int i = 0, n = blobs.Length; i < n; i++)
{
List<IntPoint> edgePoints = blobCounter.GetBlobsEdgePoints(blobs[i]);

AForge.Point center;
float radius;
CapColour = "Blue";

// is circle ?
if (shapeChecker.IsCircle(edgePoints, out center, out radius))
{
g.DrawEllipse(yellowPen,
(float)(center.X - radius), (float)(center.Y - radius),
(float)(radius * 2), (float)(radius * 2));
// MessageBox.Show("x:" + center.X.ToString() + "y:" + center.Y.ToString() + "r:"
+ radius.ToString());

CapSolid = true;
}
else
{
CapSolid = false;
}
}
}
}

```



```
yellowPen.Dispose();
g.Dispose();

// put new image to clipboard
//Clipboard.SetDataObject(bitmap);

// and to picture box
pictureBox4.Image = bitmap1;
// UpdatePictureBoxPosition();
}
```

```
private void label1_Click(object sender, EventArgs e)
{
}
}
```

```
private void textBox1_TextChanged(object sender, EventArgs e)
{
}
}
```

```
private void textBox2_TextChanged(object sender, EventArgs e)
{
}
}
```

```
private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
{
}
}
```

```

private void button5_Click(object sender, EventArgs e)
{

try
{
if (serialPort1.IsOpen)
{
MessageBox.Show("Serial port is already opened!");

}
else
{
serialPort1.PortName = textBox3.Text;
serialPort1.BaudRate = 9600;
serialPort1.Open();
}
button5.Enabled = false;
button6.Enabled = true;
textBox5.ReadOnly = false;
serialPort1.Write("O");
}
catch
{
MessageBox.Show("Serial port open error!");
}
}

private void button6_Click(object sender, EventArgs e)
{
try
{
if (!serialPort1.IsOpen)
{

```

```

MessageBox.Show("Serial port is already closed!");
}
else
{
serialPort1.Close();
}
button5.Enabled = true;
button6.Enabled = false;
textBox5.ReadOnly = true;
}
catch
{
MessageBox.Show("Serial port close error!");
}
}

private void DisplayText(object sender, EventArgs e)
{
textBox5.AppendText(RxString);

if (RxString == "M")
{
button2.PerformClick();
button4.PerformClick();
if (textBox2.Text == "NOT Circle")
{
serialPort1.Write("X");
}
}
else
{
if (textBox1.Text == "RED") serialPort1.Write("R");
if (textBox1.Text == "GREEN") serialPort1.Write("G");
if (textBox1.Text == "BLUE") serialPort1.Write("B");
}
}
}

```

```
if (textBox1.Text == "YELLOW") serialPort1.Write("Y");  
}  
}  
  
}
```

```
private void Form1_FormClosing(object sender, FormClosingEventArgs e)  
{  
if (serialPort1.IsOpen) serialPort1.Close();  
}
```

```
private void serialPort1_DataReceived(object sender,  
System.IO.Ports.SerialDataReceivedEventArgs e)  
{  
RxString = serialPort1.ReadExisting();  
this.Invoke(new EventHandler(DisplayText));  
}
```

```
private void textBox5_TextChanged(object sender, EventArgs e)  
{  
  
}  
}  
}
```



**APPENDIX B.**

**ARDUINO CODES FOR CONVEYER BELT CONTROL**

```
//mz80 relay 1
//RED relay 2
//YELLOW relay 3
//BLUE relay 4
//CRUSHED relay 5
//NULL□ relay 6
//mz80_2 relay 7
//JAW relay 8
//UPDOWN relay 9
//RIGHTLEFT relay 10
//NULL□ relay 11
//NULL□ relay 12
//NULL□ relay 13
//NULL□ relay 14
#include <Servo.h>
int pos = 0;
int i;
int belt=23;

//int capArrivalTime = 2800;
int led=25;
int output1=53;
int output2=51;
int output3=49;
int output4=47;
int output5=45;
int output6=43;
int output7=41;
int output8=39;
int output9=37;
int output10=35;
int output11=33;
int output12=31;
```

```

int output13=29;
int output14=27;

int mz80_1=0;
int mz80_2=1;
const int mz80_1pin = 4;
const int mz80_2pin = 6;
Servo JAW; //bottom
Servo upDown; //body1
Servo RIGHTLEFTaReturn; //body2
Servo forwardReverse; //Jaw

char incomingData;
void setup() {

pinMode(mz80_1pin, INPUT);
pinMode(mz80_2pin, INPUT);
//servo motor frist control//
JAW.attach(9);//Jaw
upDown.attach(8);
RIGHTLEFTaReturn.attach(2);
forwardReverse.attach(3);
JAW.write(60);
upDown.write(80);
RIGHTLEFTaReturn.write(0);//Joint
delay(30);
//relay control unit//
pinMode(belt, OUTPUT);
pinMode(led, OUTPUT);
pinMode(output1, OUTPUT);
pinMode(output2, OUTPUT);
pinMode(output3, OUTPUT);

```

```
pinMode(output4, OUTPUT);
pinMode(output5, OUTPUT);
pinMode(output6, OUTPUT);
pinMode(output7, OUTPUT);
pinMode(output8, OUTPUT);
pinMode(output9, OUTPUT);
pinMode(output10, OUTPUT);
pinMode(output11, OUTPUT);
pinMode(output12, OUTPUT);
pinMode(output13, OUTPUT);
pinMode(output14, OUTPUT);
pinMode(belt, OUTPUT);
```

```
digitalWrite(belt,HIGH);
digitalWrite(led,HIGH);
```

```
digitalWrite(output1,HIGH);
digitalWrite(output2,HIGH);
digitalWrite(output3,HIGH);
digitalWrite(output4,HIGH);
digitalWrite(output5,HIGH);
digitalWrite(output6,HIGH);
digitalWrite(output7,HIGH);
digitalWrite(output8,HIGH);
digitalWrite(output9,HIGH);
digitalWrite(output10,HIGH);
digitalWrite(output11,HIGH);
digitalWrite(output12,HIGH);
digitalWrite(output13,HIGH);
digitalWrite(output14,HIGH);
```

```
///step motors pins out//
```



```

pinMode(14, OUTPUT);
pinMode(15, OUTPUT);
digitalWrite(14,LOW);
digitalWrite(15, LOW);
////////////////////
Serial.begin(9600);
bunker();
}

/* Variable in which the read data is stored */
void loop()
{
mz80_1 = digitalRead(mz80_1pin);

digitalWrite(belt, LOW);
digitalWrite(output1, LOW);

if(mz80_1==0)
{
digitalWrite(output3,LOW);
delay(200);
digitalWrite(led,LOW);
digitalWrite(belt,HIGH);
digitalWrite(output1, HIGH); delay(1000);
Serial.write("M");
delay(1000);
digitalWrite(led,HIGH);
digitalWrite(output1,HIGH);
digitalWrite(output3,HIGH);
while(Serial.available(>0)
{ /* any new data */

```

```

incomingData = Serial.read();
Serial.flush();

/* Save new data to read character */
switch(incomingData)
{ /* Process according to the readed character*/
case 'R': /* Comparison of incoming character */
//digitalWrite(output2,LOW);
FirstMovement();
Replace(120,150);
digitalWrite(output14,LOW);
delay(500);
digitalWrite(output14,HIGH);
bunker();
// digitalWrite(output2,HIGH);
break;
case 'Y':
//digitalWrite(output3,LOW);
FirstMovement();
Replace(180,90);
digitalWrite(output13,LOW);
delay(500);
digitalWrite(output13,HIGH);
//digitalWrite(output3,HIGH);
bunker();
break;
case 'B':
// digitalWrite(output4,LOW);
FirstMovement();
Replace(150,105);
digitalWrite(output11,LOW);
delay(500);

```

```

digitalWrite(output11,HIGH);
bunker();
// digitalWrite(output4,HIGH);
break;
case 'X':
FirstMovement();
Replace(170,5);
digitalWrite(output12,LOW);
delay(500);
digitalWrite(output12,HIGH);
bunker();

break;
}
}
Serial.flush();
}
}

void FirstMovement()
{ digitalWrite(belt,LOW);
digitalWrite(output1,LOW);
//delay(capArrivalTime);
while(digitalRead(mz80_2pin) == 1){ }
digitalWrite(output1,HIGH);
digitalWrite(output4,LOW);
delay(200);
digitalWrite(belt,HIGH);

JAW.write(80);
upDown.write(70);
delay(100);

```

```

for(int z=70;z<=180;z++)
{
upDown.write(z);
delay(10);
}
digitalWrite(output6,LOW);

delay(1000);
digitalWrite(output6,HIGH);
digitalWrite(output4,HIGH);
for(int k=80;k>=20;k--)
{
JAW.write(k);
delay(50);
}
digitalWrite(output5,LOW);
delay(100);

delay(1000);
digitalWrite(output5,HIGH);
digitalWrite(output7,LOW);
delay(100);
for(int z=140;z>=20;z--)
{
upDown.write(z);
delay(20);
}

upDown.write(20);
delay(1000);
digitalWrite(output7,HIGH);

}

```

```

void Replace(int Return, int uzaklik)
{
digitalWrite(output8,LOW);
RIGHTLEFTaReturn.write(Return);
delay(1000);
digitalWrite(output8,HIGH);
digitalWrite(output6,LOW);
upDown.write(uzaklik);
delay(750);
digitalWrite(output6,HIGH);

JAW.write(80);
delay(500);
digitalWrite(output4,HIGH);
digitalWrite(output7,LOW);
upDown.write(20);
delay(750);
digitalWrite(output7,HIGH);
digitalWrite(output9,LOW);
RIGHTLEFTaReturn.write(0);
delay(1000);
digitalWrite(output9,HIGH);

}

void bunker()
{
digitalWrite(output2,LOW);
for (int i = 0; i<=1200; i++)
{
digitalWrite(14,HIGH);
digitalWrite(15,LOW);
delay(1);
digitalWrite(15,HIGH);
}
}

```

```

delay(1); }

for (int i = 0; i<=300; i++)
{
digitalWrite(14,LOW);
digitalWrite(15,LOW);
delay(1);
digitalWrite(15,HIGH);
delay(1); }
for (int i = 0; i<=800; i++)
{
digitalWrite(14,HIGH);
digitalWrite(15,LOW);
delay(1);
digitalWrite(15,HIGH);
delay(1); }
for (int i = 0; i<=300; i++)
{
digitalWrite(14,LOW);
digitalWrite(15,LOW);
delay(1);
digitalWrite(15,HIGH);
delay(1); }
for (int i = 0; i<=800; i++)
{
digitalWrite(14,HIGH);
digitalWrite(15,LOW);
delay(1);
digitalWrite(15,HIGH);
delay(1); }
digitalWrite(15,LOW);
digitalWrite(14,LOW);
digitalWrite(output2,HIGH);

```

}





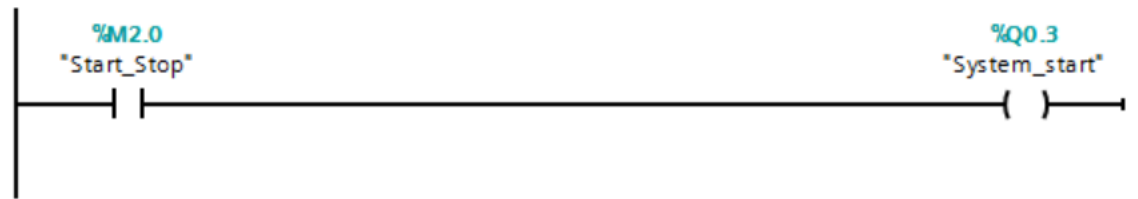
**APPENDIX C.**

**PLC LADDER DIAGRAM**



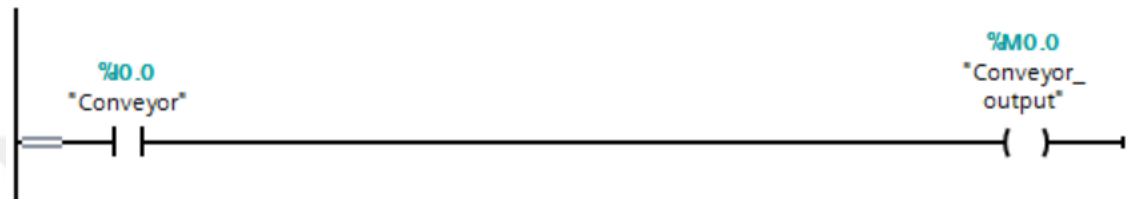
**Network 1: .....**

Comment



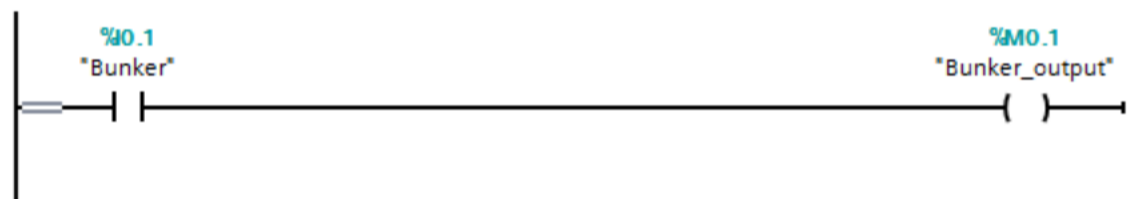
**Network 2: .....**

Comment



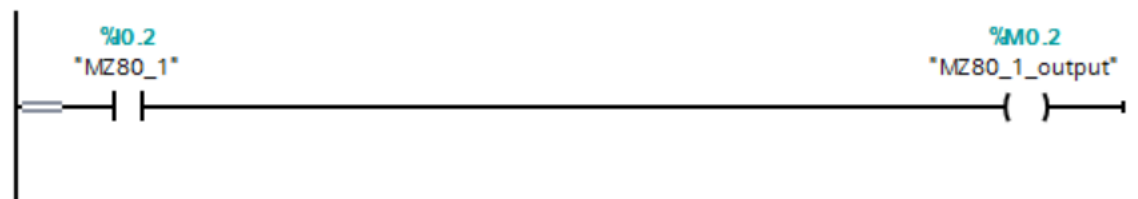
**Network 3: .....**

Comment



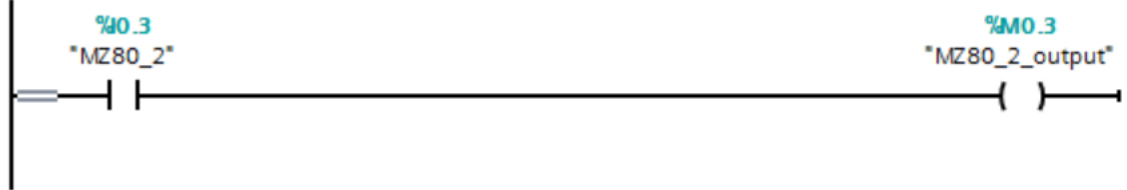
**Network 4: .....**

Comment



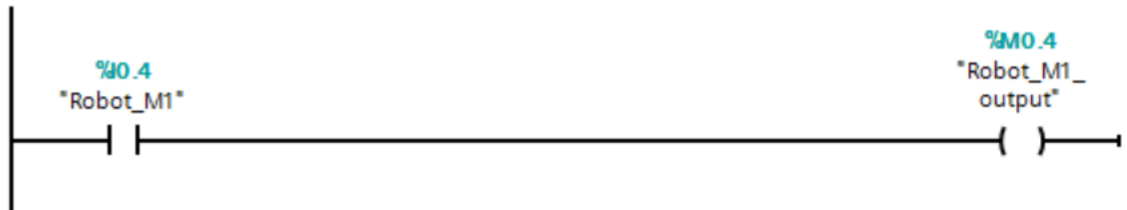
### Network 5: .....

Comment



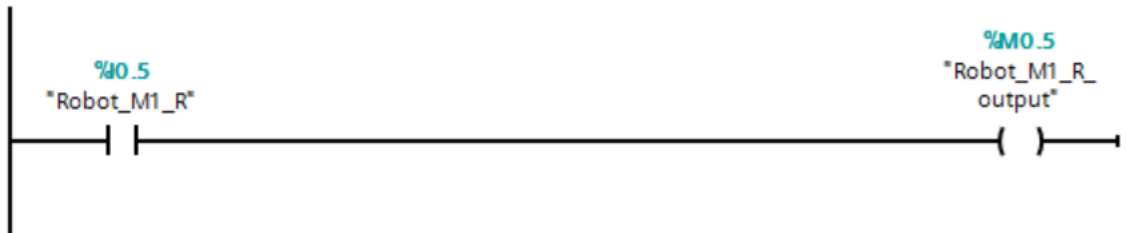
### Network 6: .....

Comment



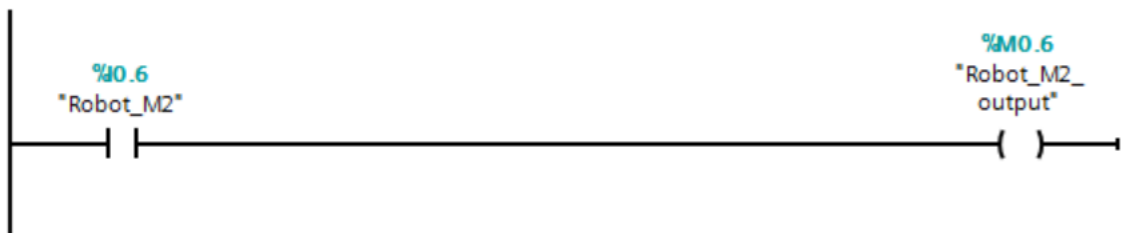
### Network 7: .....

Comment



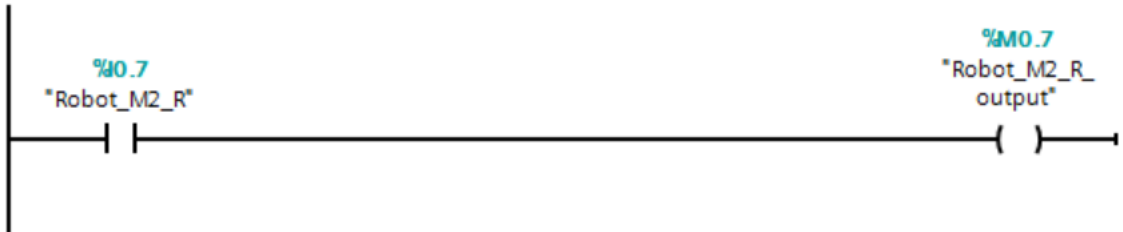
### Network 8: .....

Comment



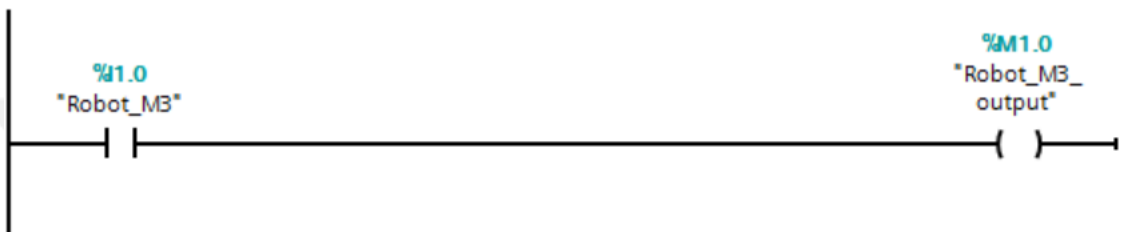
**Network 9:** .....

Comment



**Network 10:** .....

Comment



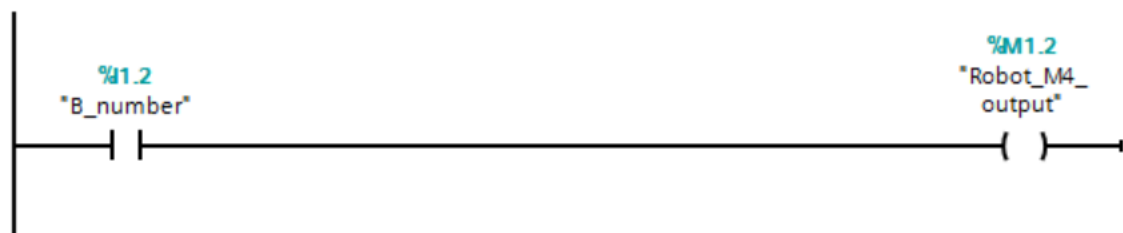
**Network 11:** .....

Comment



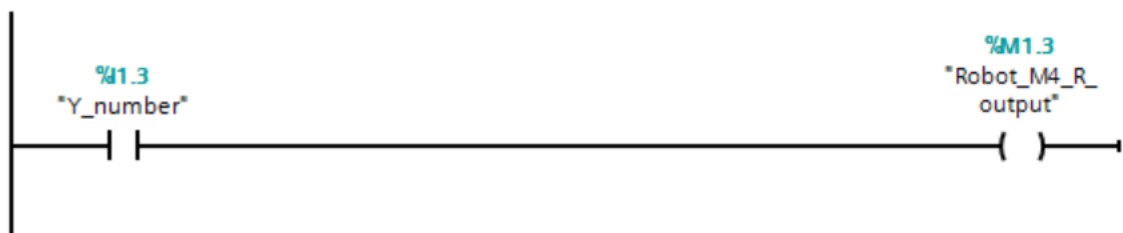
**Network 12:** .....

Comment



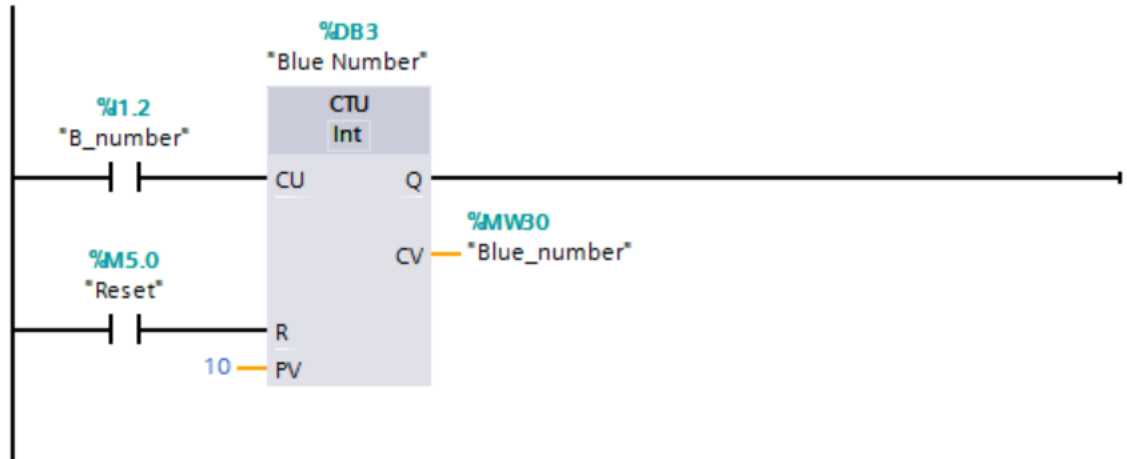
**Network 13:** .....

Comment



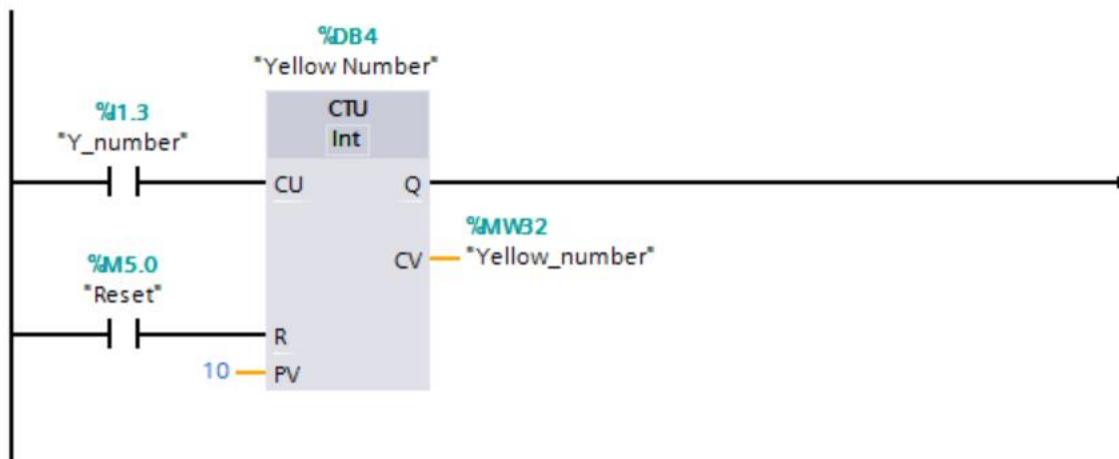
**Network 14:** .....

Comment



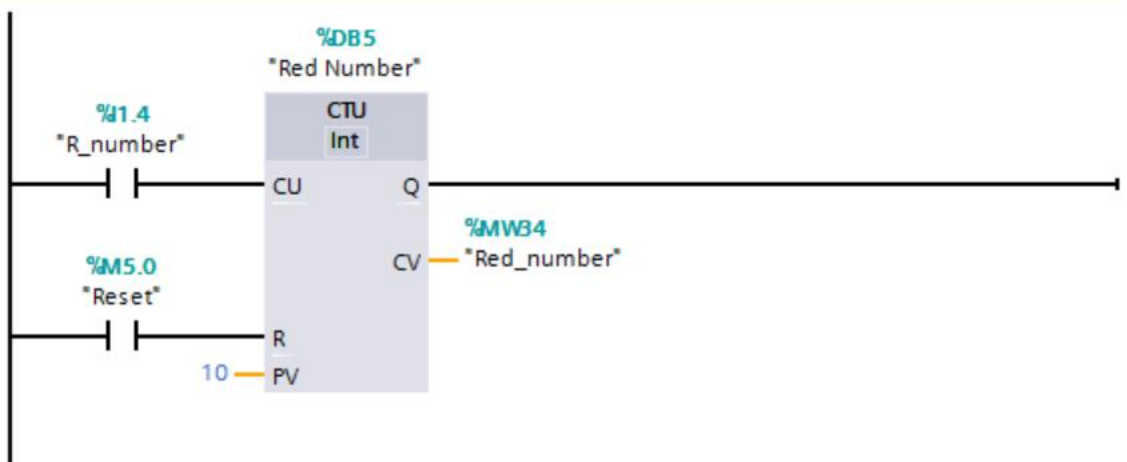
**Network 15:** .....

Comment



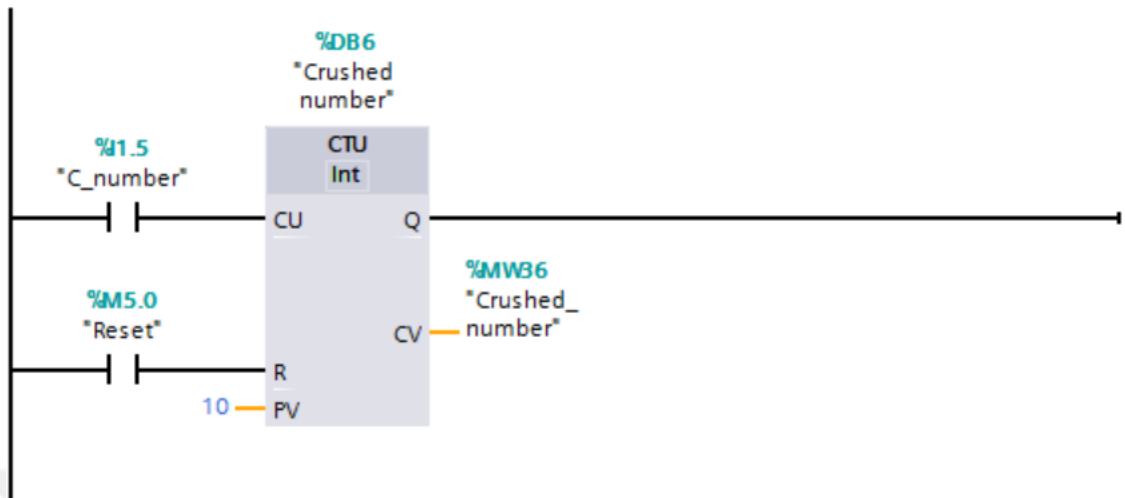
**Network 16:** .....

Comment



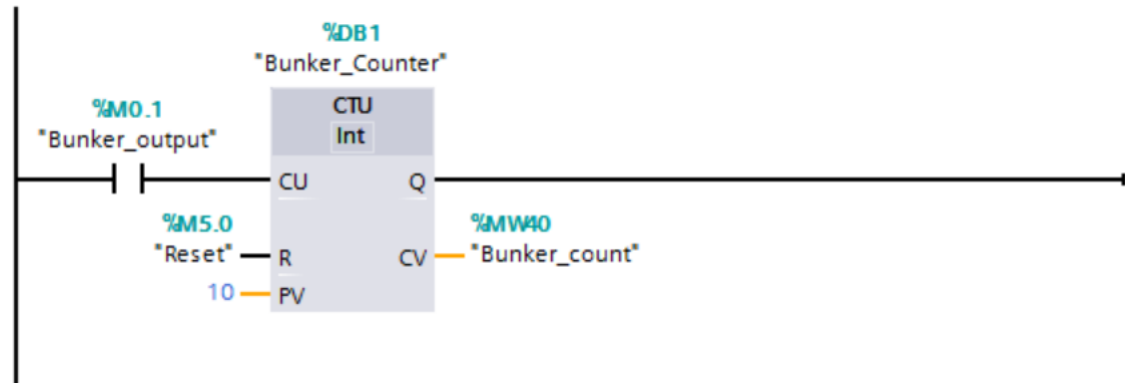
### Network 17: .....

Comment



### Network 18: .....

Comment



### Network 19: .....

Comment



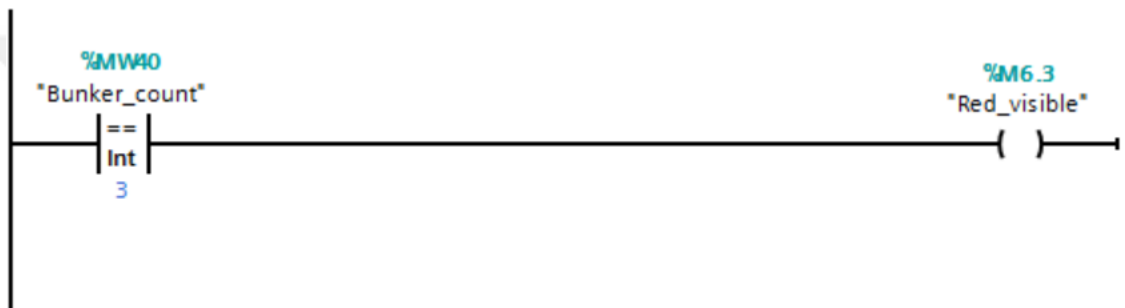
### Network 20: .....

Comment



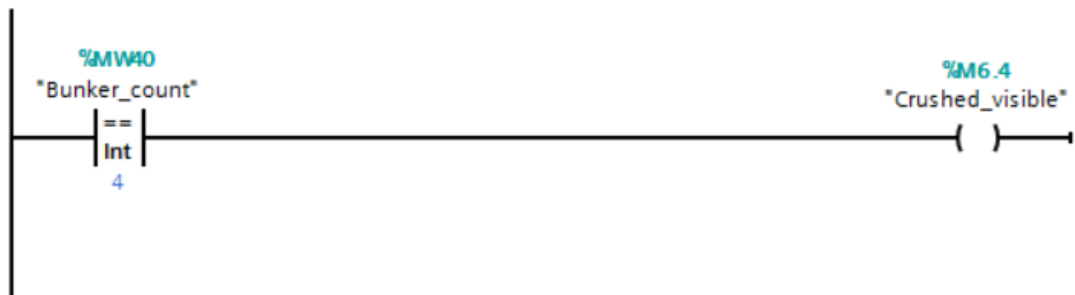
### Network 21: .....

Comment



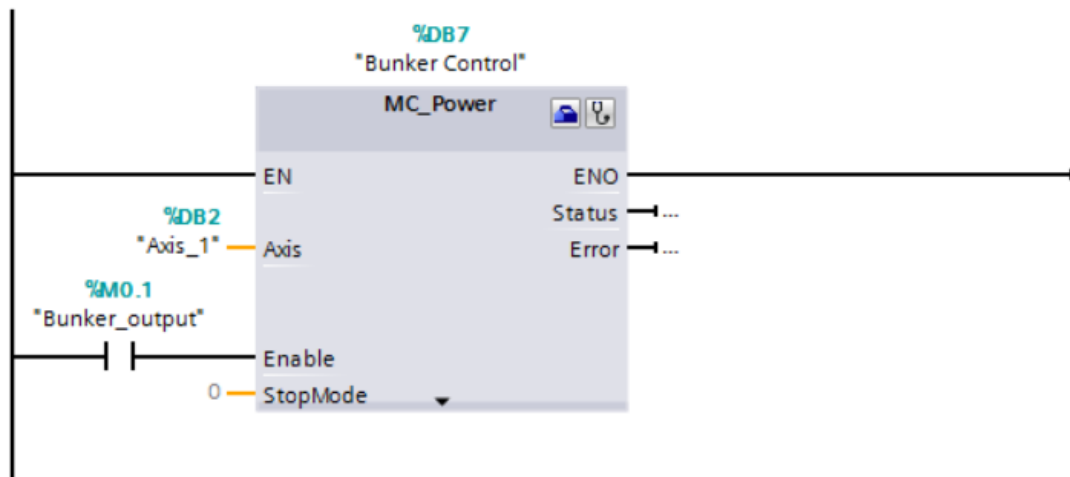
### Network 22: .....

Comment



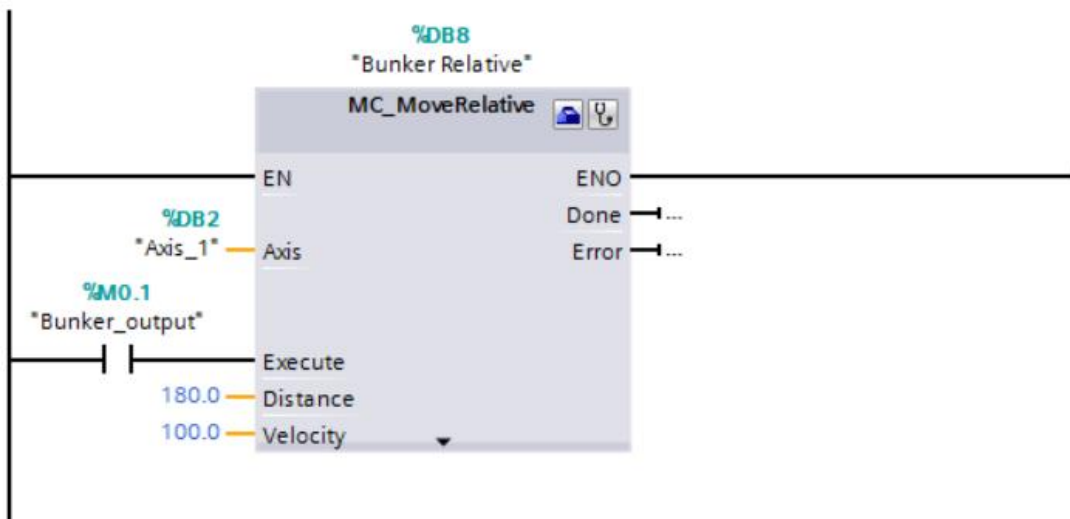
### Network 23: .....

Comment



### Network 24: .....

Comment



## **RESUME**

Abdulkarem Moftah Amar MOHAMED was born in Libya in 1986 and he graduated first, elementary and high school education in this country. After that, he finished undergraduate program in Bani Walid University Department of Auto Control in 2007. In 2015, he started graduate study in Karabuk University Department of Electrical & Electronics Engineering. He has been still continuing his education in this section.

### **CONTACT INFORMATION**

Address: Karabük University  
Graduate School of Natural & Applied Science  
Demir-Çelik Campus/KARABUK

E-mail: [kareem\\_oky@yahoo.com](mailto:kareem_oky@yahoo.com)