ACROME

myCONTROL

DELTA ROBOT

USER MANUAL

For more information on the solutions ACROME Inc. Offers, please visit the web site at:
http://www.acrome.net

Acrome Inc.
ITU ARI4 Science Park
Maslak, Istanbul Turkey
info@acrome.net
Phone: +90 212 807 04 56
Fax: +90 212 285 25 94

Printed in Maslak, Istanbul

CONTENTS

1 OVERVIEW 4

1.1 System Description 4

2 COMPONENTS 5

2.1 Smart Servo Motors 6

2.2 Round Electromagnet 6

2.3 Raspberry Pi 7

2.4 ACROME Power Distribution Box 7

3 TECHNICAL SPECIFICATIONS 8

4 WIRING 9

4.1 Cable Names 9

4.2 Connections 10

5 SETTING UP THE SYSTEM 11

5.1 Software Installation 11

5.2 Python Code : 11

6 Troubleshooting 16

1 OVERVIEW

1.1 System Description

The ACROME Delta Robot is composed of three identical arms in parallel between the base plate and the traveling end-effector plate. The combination of the constrained motion of these arms results in the three translational degrees of freedom. The upper robot arms are directly attached to the actuators. The three actuators are rigidly mounted on the base plate with 120° in between. Each of the lower arms has two parallel bars. They connect the upper arm with the traveling end-effector plate. The system was integrated with a round electromagnet and a USB camera for pick and place applications. The general structure of ACROME Delta Robot is shown in Figure 1.1.

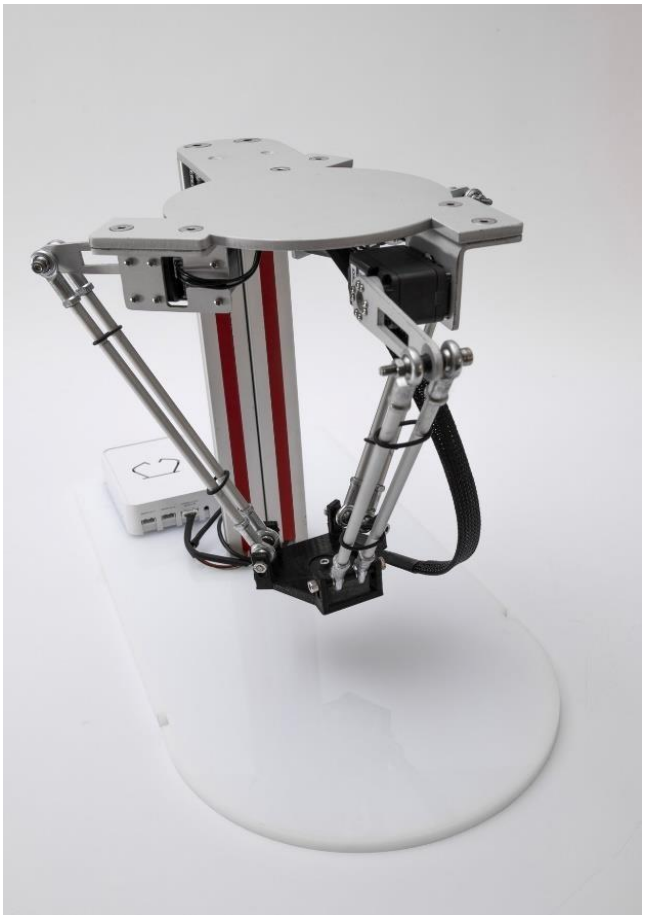


Figure 1.1: Delta Robot System

2 COMPONENTS

All the main components of "**ACROME Delta Robot**" can be seen in Figure 2.1. The numbers and the names of elements are listed in Table 2.1 below.

Table 2.1: Descriptions and Numbers of Components

Numbers	Description
1	Round Electromagnet
2	Smart Servo Motors
3	ACROME Power Distribution Box
4	Acrome Controller
5	Camera

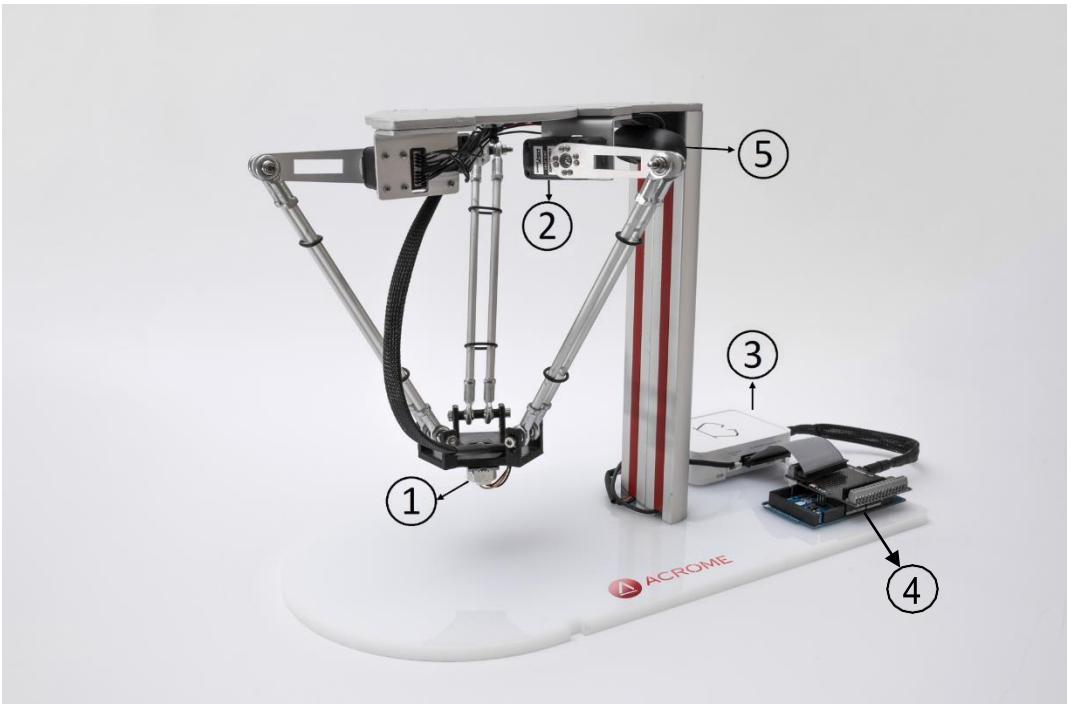


Figure 2.1: Components of the Delta Robot System

2.1 Smart Servo Motors

HerkuleX DRS 0101 includes a motor, a gear reducer, a control circuitry, and a communications capability in one single package. These servos can be easily assembled with simple wiring. If required, each servo allows serial or parallel connections. In the Delta Robot system, three servo motors are connected serially.

Figure 2.2: HerkuleX Smart Servo Motor

2.2 Round Electromagnet

The round electromagnet is used for its ability to attract and hold ferromagnetic materials with varying degrees of force, through the application of controlled DC electrical current. Electromagnets also have the ability to release the item when required.

ZYE1-P20/15 type is chosen for pick and place operations of the delta robot. The holding force of a round electromagnet is determined as 25 N.

Figure 2.3: The Round Electromagnet

In this version of the Delta Robot, the round electromagnet is used to make a stylus pen tip. With this usage, the tablet screen can detect the end effector of the Delta robot. Delta Robot can thus perform various operations in tablet applications.

2.3 Raspberry Pi

Raspberry Pi is a series of small single-board computers (SBCs) developed in the United Kingdom by the Raspberry Pi Foundation in association with Broadcom. The Raspberry Pi has a 1.4GHz quad-core ARM Cortex-A53 CPU. Also, the wireless capabilities has been upgraded to dual band (2.4GHz and 5GHz) 802.11ac and Bluetooth 4.2 with BLE support. Besides all these wireless features, Raspberry Pi 3 Model B+ also has three times more speed in Ethernet performance than the older models.

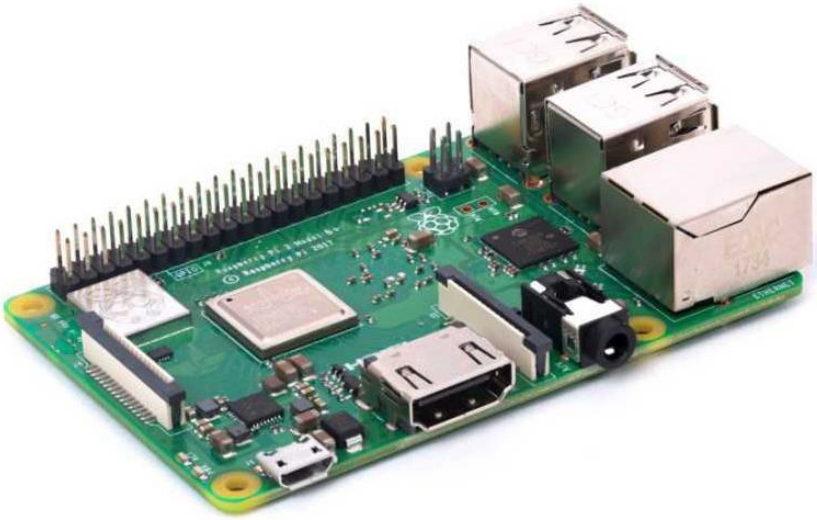


Figure 2.4: Raspberry Pi

2.4 ACROME Power Distribution Box

Three smart servo motors, sensors (round electromagnet), and Arduino connections are located on the ACROME power distribution box which is shown in Figure 2.5. It also has an RGB led and switch-mode regulators.

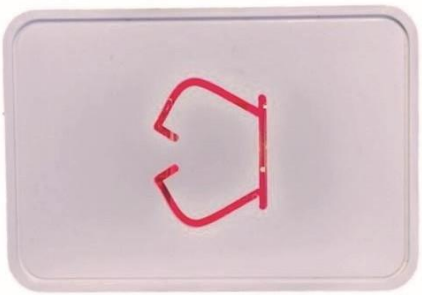


Figure 2.5: ACROME Power Distribution Box

3 TECHNICAL SPECIFICATIONS

ACROME Delta Robot system has dimensions and parameters which are given in Table 3.1. The schematic representation of the Delta Robot system is shown in Figure 3.1.

Table 3.1: Dimensions and Parameters of the System

Symbols	Definition	Value	Unit
f	Side of the base plate triangle	230.59	[mm]
e	Side of the traveling plate triangle	112.96	[mm]
r_f	Length of the upper arm	64.20	[mm]
r_e	Length of the lower arm	200.00	[mm]
l_s	Induction chrome shaft length	240.00	[mm]
mm	Maximum payload	100.00	[g]

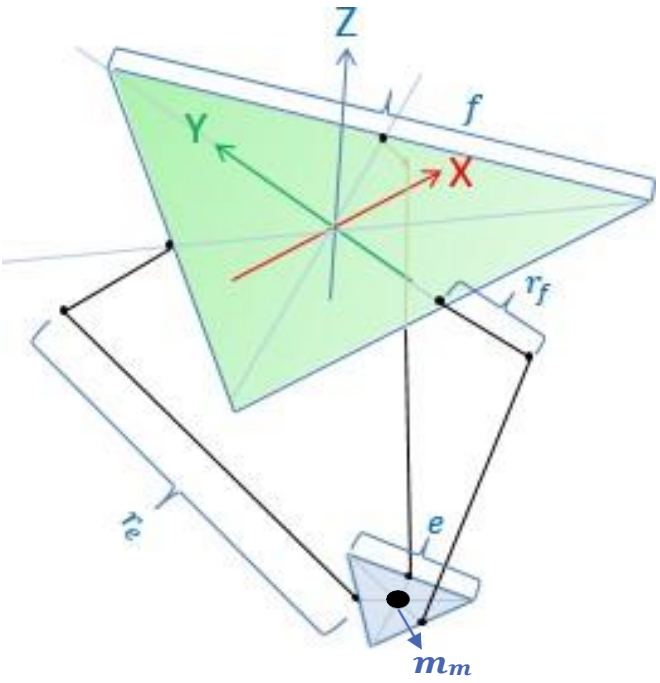
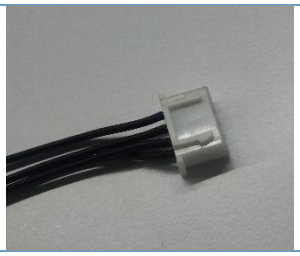


Figure 3.1: Delta Robot Schematics

4 WIRING

4.1 Cable Names

Cables which are used in the Delta Robot system are defined and denominated as seen in Table 4.1 below.

Cable Names	View of the Cables	Definitions
1. Electromagnet Cable		Transmits power and signal to electromagnet
2. Smart Servo Motor Cable		Connects Smart servo motor to power distribution box to transmit UART signals.
3. Flat Cable		Transmits signals between Acrome Controller and the power distribution box.
4. Camera Cable		Transmits signals from camera to PC.
5. Power to power Cable		Transmits the power from the adapter to raspberry and Acrome Controller.

4.2 Connections

In order to use the system, all cables mentioned above should be connected properly. All the essential connections among the components are shown in Figure 4.1.

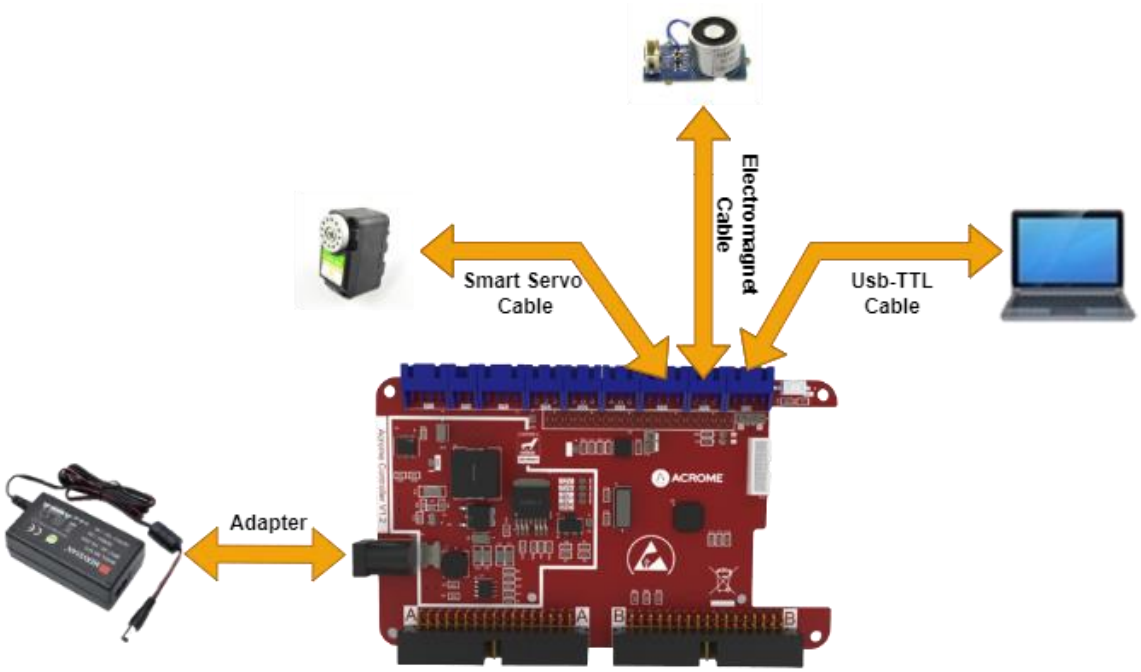


Figure 4.2: Delta Robot System Connections

5 SETTING UP THE SYSTEM

5.1 Software Installation

In order to run the Delta Robot Python Pick and Place application, python3.5 must be installed on your computer.

The pip installation must be completed in order to download the necessary python libraries. For pip installation, the steps of the article in the link should be followed.

Required python libraries are mentioned below.

-OpenCV

-acrome

-numpy

To install the required libraries, select the directory where the requirements.txt file is located and run the following terminal command:

pip install -r .\requirements.txt

5.2 Python Code :

The following files must be in the same directory for the Python script to work properly:

-main.py

-Delta_Robot.py

-calibrationimage.JPG

Run the main.py file to run the Pick and Place application.

5.2.1 Delta_Robot.py

DeltaRobot.py > ...

```
1  import time
2  import math
3  import cv2
4  import numpy as np
5
6  class AcromeDelta(object):
7      _f = 230.59   # Distance from center of machine base to center of each motor sha
8      _e = 112.96  #  Distance from wrists to tool
9      _rf= 64.2    #   Distance from motor shaft to elbow
10     _re= 200     #   Distrance from elbow to the wrist
```

Adding the required libraries and defining the mechanical parameters of the Delta Robot

```
def inverse_kin(self,x0,y0,z0):
    for i in range(3):
        if  i==0:
            x0_1 = x0
            y0_1 = y0
        elif i==1 :
            x0_1 = x0 * math.cos(math.pi/3*2) + y0 * math.sin(math.pi/3*2)
            y0_1 = y0 * math.cos(math.pi/3*2) - x0 * math.sin(math.pi/3*2)
        else:
            x0_1 = x0 * math.cos(math.pi/3*2) - y0 * math.sin(math.pi/3*2)
            y0_1 = y0 * math.cos(math.pi/3*2) + x0 * math.sin(math.pi/3*2)

        y1 = -0.5 * self.__class__._f * math.tan(math.pi / 6)
        y0_1 =y0_1 -(0.5 * math.tan(math.pi / 6)* self.__class__._e)

        a = (x0_1 * x0_1 + y0_1 * y0_1 + z0 * z0 + self.__class__._rf ** 2 - self
        b = (y1 - y0_1) / z0

        #discriminant
        d = -( a + b * y1) * (a + b * y1) + self.__class__._rf * (b * b * self.__
```

This function calculates the inverse kinematic solution of the Delta robot. The x, y, z values of the end effector are added to the function as arguments and the function returns the calculated motor angles.

```
def forward_kin(self,theta1,theta2,theta3):
    theta1= theta1 * math.pi/180
    theta2= theta2 * math.pi/180
    theta3= theta3 * math.pi/180

    t = (self.__class__._f-self.__class__._e)*math.tan(math.pi/6)/2

    y1 = -(t + self.__class__._rf*math.cos(theta1))
    z1 = -self.__class__._rf*math.sin(theta1)

    x2 = (t + self.__class__._rf*math.cos(theta2))*math.sin(math.pi/6)*math.tan(m
    y2 = (t + self.__class__._rf*math.cos(theta2))*math.sin(math.pi/6)
    z2 = -self.__class__._rf*math.sin(theta2)

    x3 = -((t + self.__class__._rf*math.cos(theta3))*math.sin(math.pi/6))*math.ta
    y3 = (t + self.__class__._rf*math.cos(theta3))*math.sin(math.pi/6)
    z3 = -self.__class__._rf*math.sin(theta3)
    #   Define the deminator parameter
    d = (y2-y1)*x3-(y3-y1)*x2
```

This function calculates the forward kinematic solution of the Delta robot. The motor angles are added to the function as arguments and the function returns the calculated x,y,z values of the end effector.

```
def detect_coins(self):
    camera = cv2.VideoCapture(1+cv2.CAP_DSHOW)
    return_value, frame = camera.read()
    img = cv2.imread("calibrationimage.JPG")
    gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
    gray = np.float32(gray)
    dst = cv2.cornerHarris(gray,2,3,0.04)
    #result is dilated for marking the corners, not important
    dst = cv2.dilate(dst,None)
    ret, dst = cv2.threshold(dst,0.01*dst.max(),255,0)
    dst = np.uint8(dst)
    # find centroids
    ret, labels, stats, centroids = cv2.connectedComponentsWithStats(dst)
```

The newly captured image is calibrated by detecting the edges from the sample calibration image. Afterwards, the coins are detected with the Hough Circle detection method and their pixel values are returned.

```
def trajectory(self,startTime,Ipos,Fpos,tf):
    #Higher(5th) order trajectory function
    endTime = time.time()
    t=(endTime-startTime) #ElapsedTime
    Final_Velocity=0
    Final_Acceleration=0
    Initial_Velocity=0
    Initial_Acceleration=0
    PosTraj = [0,0,0]  # Initialize the position trajectory
    for i in range(3):
        a0 = Ipos[i]
        a1 = Initial_Velocity
        a2 = Initial_Acceleration/2
        a3 = (20*Fpos[i] - 20*Ipos[i] - ((8*Final_Velocity + 12*Initial_
        a4 = (30*Ipos[i] - 30*Fpos[i] + ((14*Final_Velocity + 16*Initia
        a5 = (12*Fpos[i] - 12*Ipos[i] - ((6*Final_Velocity + 6*Initial_
        PosTraj[i] = a5*t**5 + a4*t**4 + a3*t**3 + a2*t**2 + a1*t + a0
    return PosTraj
```

Trajectory equations are created between 2 points. Values return depending on time. The end-effector will be in the desired position at the end of the desired time.

```
156     def postoAngle(self,pos):
157         theta=-np.multiply(pos-512,0.32612)     #Converts the position to angle
158         return theta
159
160     def angletoPos(self,theta):
161         pos = np.divide(theta,-0.32612) + 512   #Converts the angle to position
162         return pos
```

Angle-motor position conversions for writing angles to the Delta robot

5.2.2 main.py

```python
from DeltaRobot import AcromeDelta
from acrome.controller import *
import time
import numpy as np

dev = Delta('COM5') # Default port is COM5

robot=AcromeDelta()
trajTime=2
initialPos=[-10,-50,-145]
placingPos=[0,-55,-220]

#If the end effector is not grabbing the coin, set the X-Y-Z offsets
calc_Z=-229     #Z value of the coins
x_offset=15
y_offset=0

theta=robot.inverse_kin(initialPos[0],initialPos[1],initialPos[2])
pos=robot.angletoPos(theta)
dev.set_motors(np.int_(pos)) # Go to initial position
dev.update()
```

- Run the main.py file to run the application.
- The default communication port is COM5. Change this value according to your own COM port.
- The Z value of the end-effector's coin to grab is -229. If it cannot grab, you need to adjust the calc_Z , x_offset, y_offset values once.
- If you want to change the movement speed of the Acrome Delta Robot, change the trajTime value.
- The sequence of operations in the pick and place application of the Delta robot is as follows:

1. Take a photo
2. Calibrate photo according to the calibration image
3. Detect coins.
4. Move to the coin location according to the desired trajectory time.
5. Grab the coin
6. Return to initial position
7. Go to Place point
8. Drop the coin
9. Move to the initial position.
10. Go back to step 4 to catch 2nd and 3rd coins

```python
circlePoint=robot.detect_coins()
if circlePoint is not None:
    for x in circlePoint:
        for i in range(4):
            startTime = time.time()
            endTime = time.time()
            coinPos= [x[0]/5-55,x[1]/5-55,calc_Z] #Convert the pixel to the real world coordinates
            """
            i=0 picking
            i=1 go to initial
            i=2 placing
            i=3 go to initial
            """
            if i==0:
                print("Initial Point")
                Ipos=initialPos
                Fpos=coinPos
            if i==1:
                print("Picking Point")
                dev.pick(True)
                Ipos=coinPos
                Fpos=initialPos
            if i==2:
                print("Initial Point")
                Ipos=initialPos
                Fpos=placingPos
            if i==3:
                print("Placing Point")
                dev.pick(False)
                Ipos=placingPos
                Fpos=initialPos
            while(endTime-startTime)<trajTime:
                traj_pos=robot.trajectory(startTime,Ipos,Fpos,trajTime)
                print(traj_pos)
                theta=robot.inverse_kin(traj_pos[0]+x_offset,traj_pos[1]+y_offset,traj_pos[2])
                pos=robot.angletoPos(theta)
                dev.set_motors(np.int_(pos))
                dev.update()
                endTime = time.time()
else:
    print("not detected")
```

If coins are detected, the robot's main loop where moves are performed.

6 Troubleshooting

Motors Error:

If motors blink red color like the below picture, you unplug and plug of the adapter.

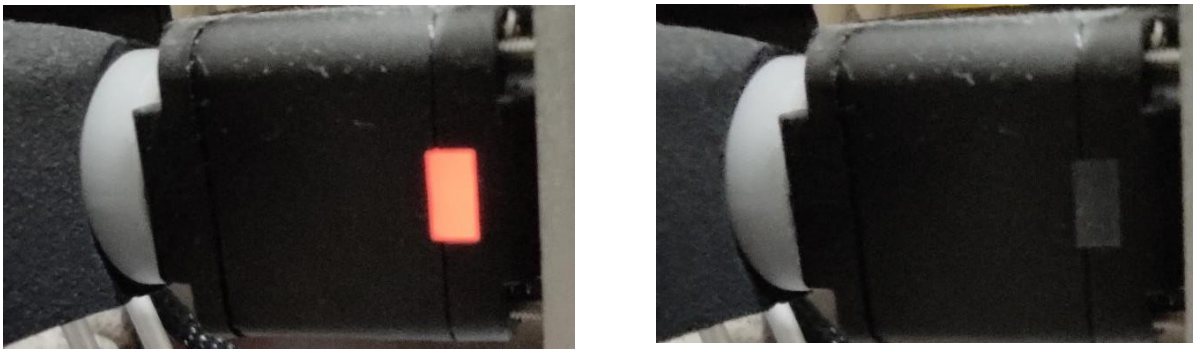


Figure 7.2: Motors Error

ACROME Robotik Mekatronik Sistemleri San. ve Tic. Ltd. AŞ.

İTÜ Ayazağa Kampüsü

Koru Yolu ARI 4 Binası B204

For further information on ACROME equipment please contact.

Website: http://www.acrome.net/

e-mail: info@acrome.net

Telephone: 0212 807 0456