International Journal of Engineering and Information Systems (IJEAIS)
ISSN: 2643-640X
Vol. 7 Issue 2, February - 2023, Pages: 84-97

Programming and Experimental Testing of A 6 DOF Pick-And-Place Robot with Vision Measurement System for Industrial Application at ALUMIL Aluminium Industry SA

Panagiotis Rizos

Robotics Engineer at ALUMIL Aluminum Industry S.A, Kilkis Industrial Area, Greece.

Abstract— *This paper presents the development and laboratory experiments on a six-degree-of-freedom robotic arm model that does a specific pick and place process and could be used at low cost in a specific industrial application at ALUMIL Aluminium Industry SA. The robotic arm cooperates with an optical vision system which measures the cutting length dimension of the objects. The robotic arm first picks up a saw-cut object and transports it in front of the vision system's camera to be measured. If the object is found out of tolerance, then the arm receives the measured object to transport it to a scrap point. If the object is found within bounds, then the arm receives the object to move it to another point to continue its process in the next stage. Laboratory experiments showed that the model of the robot-vision system works satisfactorily in terms of cooperation with vision system and execution of ten cycles, but future improvement in measurement accuracy is needed for the camera to recognize millimeter differences. Also, vibrations occurred in the robot body during the ten cycle paths due to the low speeds and the material it is made of, resulting bad accuracy and repeatability.*

Keywords— Robot, Pick and Place, Vision Measurement System

1. INTRODUCTION

Pick and place robots belong to the category of industrial robots that can be programmed to move in three or more axes. The importance of these robots can be understood by realizing how big the industry is. The sales volume of industrial robots has tripled in the last decade, reaching a figure of nearly 4,00,000 units in 2018. About 2,000 doses of robots can replace 10,000 jobs in the industry [1].

Pick and place robots take the object from one location and place it at another coordinate that we have programmed. Incorporating a vision system would allow the robot to visualize the exact location of the object to grasp, pick it up, and move it on a conveyor line. The benefits of such robots are many in industry. One of them is speeding up production and reducing business operating costs. The use of a pick and place robot can greatly increase the speed of the processes involved in various operations. This saves a lot of time for businesses by moving repetitive tasks at a steady pace due to the extent of synchronization it brings to the process. In Fig 1 we see an application of a General Motors pick and place robot in an aluminum die casting foundry [2].

Accuracy and consistency are among the most reliable characteristics of a robot. This is because there is no possibility of human error at all. Robots are programmed to do specific tasks and minimize the chances of any human error to zero. Integrating robots into a system is one of the major benefits for businesses [3].

This paper presents the programming and experimental results of a Tinkerkit Braccio-type 6 DOF robot coupled with an optical vision system to measure parts from the output of a saw. The application of the system concerns a specific production process of parts carried out at the ALUMIL Aluminium Industry SA. ALUMIL specialized in the research, development, and production of aluminum architectural systems. Initially, the materials used, the characteristics of the robotic arm used for pick and place, the programmable Arduino microcontroller, and the camera of the optical measurement system connected to the computer are presented. Then the ideal implementation of the system in the production process is presented to understand the process we wish to do. Then the codes written to control the robotic arm and the optical vision system are presented. Finally, there are the results of the ten laboratory experiments corresponding to ten robot cycles and future improvements that could be made [4].

Fig 1. General Motors pick and place robot in aluminum die casting foundry. Source [5].

2. MATERIALS AND METHODS

2.1 Pick and Place Robot

The Tinkerkit Braccio robot model was used for our experiments. This particular robot is small in size and made of plastic material. This particular robot was used to demonstrate that the system can be developed, improved and evolved based on low-cost materials that can eventually be replaced by other materials such as an aluminum body or higher power motors [6].

Tinkerkit Braccio body characterized by Weight 792g, Maximum operating range 80cm, Maximum height 52cm, Base width 14cm, Handle opening 90mm Cable length 40 cm, Load capacity Maximum weight at 32 cm operating distance: 150 g, Maximum weight in minimum Braccio configuration: 400 g. Regarding to servo motors, robot characterized by Spring RC SR431 Dual Output Servo Motor, Control Signal: Analog PWM, Torque: 4.8V 169.5 oz-in (12.2 kg-cm) and 6.0V: 201.4 oz-in (14.5 kg-cm), Weight 62.0 g, Dimensions 1.65 × 0.81 × 1.56 in (42.0 × 20.5 × 39.5 mm), Speed: 4.8 V: 0.20 sec / 60 ° and 6.0V: 0.18 sec / 60 °, Rotation Support: Double Bearings, Gear Material: Metal, Rotation Range: 180 °, Connection Type: J (also known as Futaba).

Also, another type of servo that robot has, characterized by Spring RC SR311, Control signal Analog: PWM, Torque: 4.8V: 43.13 oz-in (3.1 kg-cm) and 6.0V: 52.86 oz-in (3.8 kg-cm), Weight: 27.0 g, Dimensions: 1.23 × 0. 65 × 1.13 inches (31.3 × 16.5 × 28.6 mm), Speed: 4.8V: 0.14 sec / 60 ° and 6.0V: 0.12 sec / 60 °, Rotation support: Double bearings, Gear material: Metal, Rotation range: 180 °, Connection type: J (also known as Futaba). The Arduino TinkerKit Braccio Robotic Arm is a fully functional robotic arm, controlled via an Arduino processor ideal for modeling processes. For our experiments we use the specific robot to do the procedure of pick and place [7].

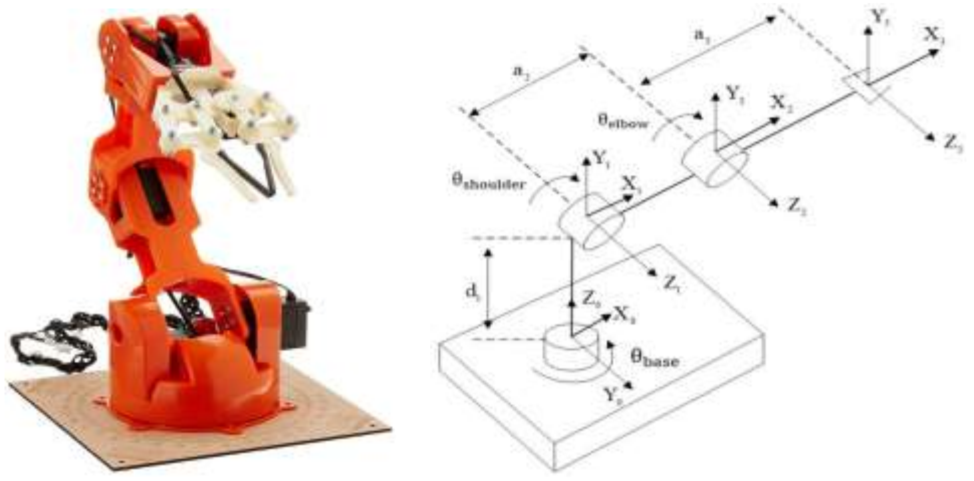


Fig 2. Tinkerkit Braccio (left) and drawing of the robotic arm and joint frames (right).

2.2 Arduino Mega 2560 Microcontroller and Robot Shield

Arduino Mega 2560 programmable microcontroller (Fig 3. left) is necessary for operate the robot model. The basis of this board will be the programming of the robot in C++ language. The features of the board are operating voltage 5V, recommended input voltage will range from 7v to 12V, input voltage ranges from 6v to 20V, Digital input/output pins are 14, Analog i/p pins are 6, DC Current for each input/output pin is 40 mA, DC Current for 3.3V Pin is 50 mA, Flash Memory is 32 KB.

On top of the Arduino microcontroller, it is necessary to place the Robot Shield (Fig 3. right). It comes with a 5V 4A DC switching power supply (wall-wart), and circuit for powering the servos from the arduino. The shield connectors labeled M1 through M6 are connected to the PWM capable outputs of the Arduino board and are used to drive the six servo motors of the robotic arm [8].

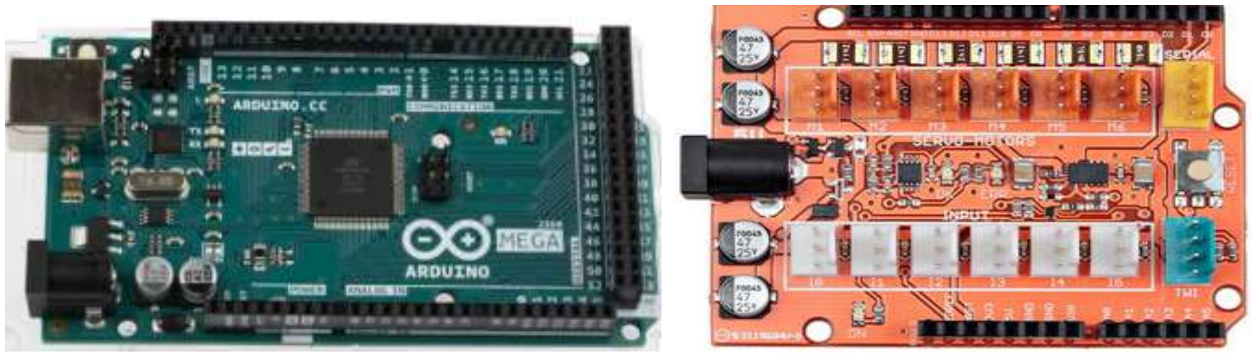


Fig 3. Arduino Mega 2560 Microcontroller (left) and Robot Shield (right).

2.3 Optical Measurement System

The HP 320 Webcam (Fig 4.) was used to check the cut length dimension of the part. This particular camera was placed in a vertical position with visual targeting at the space of a single object and connected to the computer. Based on this, the code was written in Python language, to recognize the cutting length of the part. The camera is characterized by Cable length 59 in Compatible operating systems Windows 11, Windows 10, macOS and Chrome OS Minimum system requirements USB-A What's in the box Webcam, Quick Start guide and Warranty Compatibility Compatible with USB-A devices. Dimensions (W x D x H) 2.83 x 2.11 x 2.09 in Weight 0.24 lb Warranty HP standard one-year limited warranty [9].

Fig 4. HP 320 Webcam for optical measurement system.

3. IDEAL SYSTEM APPLICATION

Before we proceed to present the robot and vision code, it is necessary to see the ideal implementation of the system in production. In the following Fig 5 we see the space where our system could be placed. To the left of the picture is the saw outlet. Then comes the parts chute through which the cut parts end up in a bin ready for the next process The saw cuts aluminum rods into small parts. All parts cut from the rod have the same length dimensions. Each time the saw cuts the bar a new cut part is created which drags the one before it. Let's say that the specific part has a cutting length of 60mm. When the first part from the queue reaches the opening of the saw and slide then the robot can pick it up with its gripper. The robot can be placed at the junction between the saw exit and the beginning of the slide. The length of the arm is sufficient to go from an initial position to the position of the part and pick it up.

The robot has now received the cut part. Visual inspection must now be done via the camera system. The camera can be placed horizontally on the slide next to the robot. The camera will aim perpendicular to the horizontal surface. On this surface the part will be left for measurement. So, when the robot receives the cut part from the saw outlet it will place it in the horizontal plane under the camera so that it can be measured. If the camera system finds it out of tolerance i.e., above 60mm or below 60mm (tolerance will be shown in the codes) the part will grab it again and go to place it in another spot for scrap. The scrap point could be to the right and behind the robot. If the camera system finds the part within tolerances i.e., 60mm (the tolerance will be shown in the codes) then the robot will grab the part from the horizontal plane it is on and place it on the slide to go to the ready basket and the next process that is needed.

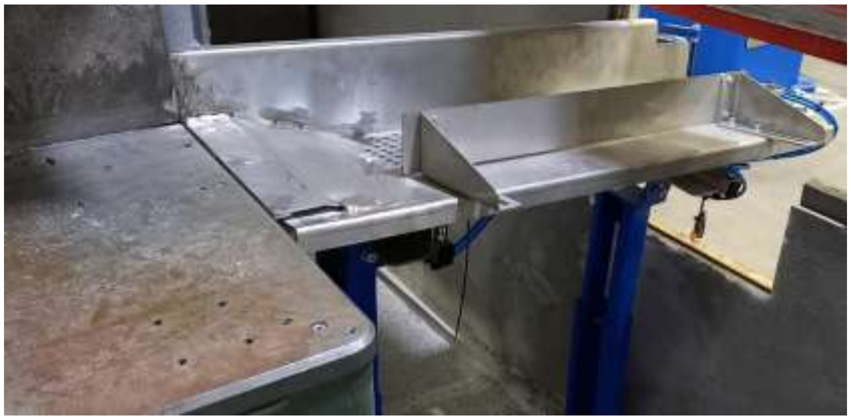


Fig 5. Ideal system application for saw in ALUMIL Aluminium Industry S.A.

4. Robot and vision meassurement system programming

4.1 Robot Programming

```
#include <BraccioRobot.h>
#include <Servo.h>
#define GRIPPER_HALF_CLOSED 64
#define LED 13
Position myInitialPosition(87, 135, 0, 57, 90,  GRIPPER_CLOSED);
Position readyPosition(60, 135, 0, 57, 90, 10);
Position pickUpPosition(180, 60, 50, 0, 60, 10);
Position putDownPosition;
void setup() {
Serial.begin (9600);
pinMode(LED, OUTPUT);
BraccioRobot.init(myInitialPosition);
putDownPosition = pickUpPosition;
}
void moveBetweenPositions(Position& fromPosition, Position& toPosition) {
BraccioRobot.moveToPosition(readyPosition.setGripper(GRIPPER_OPEN), 100);
BraccioRobot.moveToPosition(fromPosition.setGripper(GRIPPER_OPEN), 80);
BraccioRobot.moveToPosition(fromPosition.setGripper(GRIPPER_HALF_CLOSED), 100);
BraccioRobot.moveToPosition(readyPosition.setGripper(GRIPPER_HALF_CLOSED), 100);
BraccioRobot.moveToPosition(toPosition.setGripper(GRIPPER_HALF_CLOSED), 100);
BraccioRobot.moveToPosition(toPosition.setGripper(GRIPPER_OPEN), 100);
BraccioRobot.moveToPosition(readyPosition.setGripper(GRIPPER_OPEN), 100);
}
void loop() {
```

```
char RxedByte = 0;
Move from pickUpPosition to putDownPosition
moveBetweenPositions(pickUpPosition, putDownPosition);
delay(2000);
moveBetweenPositions(putDownPosition, pickUpPosition);
if (Serial.available())
{
RxedByte = Serial.read();
switch(RxedByte)
{
case 'A':
digitalWrite(Play, HIGH);
moveBetweenPositions(putDownPosition, pickUpPosition);
moveBetweenPositions(putDownPosition, pickUpPosition);
putDownPosition.setBase(100);
delay(1000);
moveBetweenPositions(pickUpPosition, putDownPosition);
putDownPosition.setBase(10);
moveBetweenPositions(putDownPosition, pickUpPosition);
putDownPosition.setBase(180);
putDownPosition.setBase(10);
moveBetweenPositions(putDownPosition, pickUpPosition);
putDownPosition.setBase(30);
}
case 'B':
{
digitalWrite(Play, LOW);
delay(1000);
moveBetweenPositions(pickUpPosition, putDownPosition);
putDownPosition.setBase(10);
moveBetweenPositions(putDownPosition, pickUpPosition);
putDownPosition.setBase(30);
}
default:
break;
}//end of switch()
}//endof if
}
```

4.2 Vision Measurement System Programming

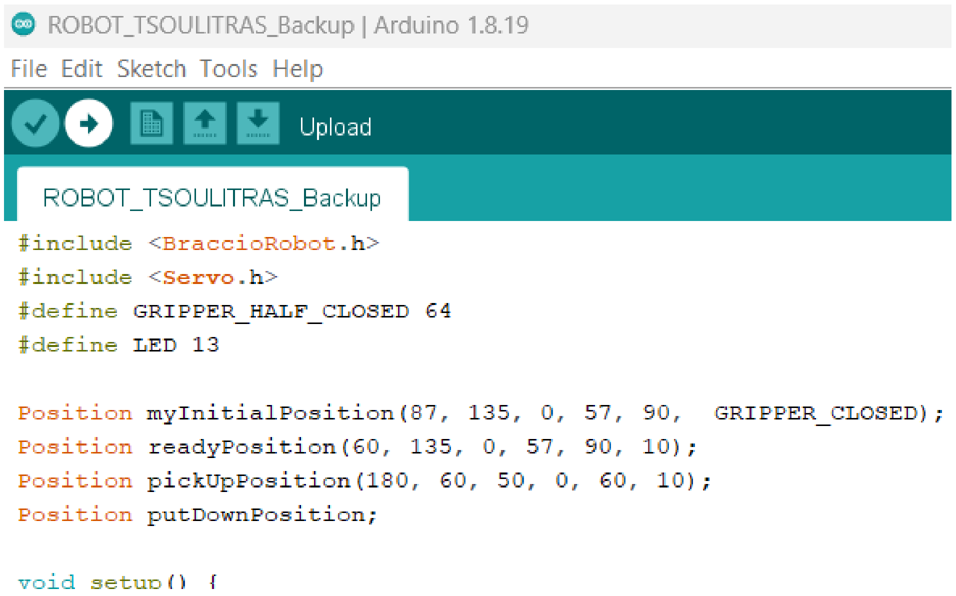
```
import cv2
from object_detector import *
import numpy as np
import serial
import time
SerialObj = serial.Serial('COM5')
SerialObj.baudrate = 9600
SerialObj.bytesize = 8
SerialObj.parity   ='N'
SerialObj.stopbits = 1
time.sleep(3)
print('-------AUTONOMOUS OPTICAL MEASUREMENT SYSTEM FOR ROBOT MANIPULATOR------')
parameters = cv2.aruco.DetectorParameters_create()
aruco_dict = cv2.aruco.Dictionary_get(cv2.aruco.DICT_5X5_50)
detector = HomogeneousBgDetector()
cap = cv2.VideoCapture(0)
cap.set(cv2.CAP_PROP_FRAME_WIDTH, 1280)
cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 720)
while True:
_, img = cap.read()
corners, _, _ = cv2.aruco.detectMarkers(img, aruco_dict, parameters=parameters)
if corners:
int_corners = np.int0(corners)
cv2.polylines(img, int_corners, True, (0, 255, 0), 5)
aruco_perimeter = cv2.arcLength(corners[0], True)
pixel_cm_ratio = aruco_perimeter / 20
contours = detector.detect_objects(img)
for cnt in contours:
rect = cv2.minAreaRect(cnt)
(x, y), (w, h), angle = rect
object_width = w / pixel_cm_ratio * 10
object_height = h / pixel_cm_ratio * 10
box = cv2.boxPoints(rect)
box = np.int0(box)
cv2.circle(img, (int(x), int(y)), 5, (0, 0, 255), -1)
cv2.polylines(img, [box], True, (255, 0, 0), 2)
cv2.putText(img, "LENGTH {} mm OK!".format(round(object_width, 1)), (int(x - 100), int(y - 20)),
cv2.FONT_HERSHEY_PLAIN, 2, (100, 200, 0), 2)
if object_width > 60.2:
cv2.putText(img, "BIG CUT {} mm SCRAP!".format(round(object_width, 1)), (int(x - 100), int(y - 20)),
cv2.FONT_HERSHEY_PLAIN, 2, (300, 400,
900), 3)
BytesWritten = SerialObj.write(b'A')
print('SERVO ON! BIG SCRAP: ', BytesWritten)
time.sleep(0)
elif object_width < 59.8:
```

```
cv2.putText(img, "SMALL CUT {} mm SCRAP!".format(round(object_width, 1)), (int(x - 100), int(y - 20)), cv2.FONT_HERSHEY_PLAIN, 2, (300, 400,
900), 3)
BytesWritten = SerialObj.write(b'B')
print('SERVO ON! SMALL SCRAP: ', BytesWritten)
time.sleep(0)
cv2.imshow("Image", img)
key = cv2.waitKey(1)
if key == 27:
break
cap.release()
cv2.destroyAllWindows()
```

5. EXPERIMENTAL RESULTS

Experiments of model robot system and optical vision system were done in laboratory. To start the experiments, the robot shield was placed on the Arduino Mega 2560 microcontroller. The microcontroller was connected to the computer via USB port. The prey used is COM5. The camera was also connected to the computer to perform the visual measurement of the object. The prey used is COM6 [10]. The robot model was placed on the lab bench with the computer in a similar arrangement as it might be placed on the saw of the Ideal System Application example. The robot when placed on the bench was left in a random position because when the code starts to run serially then the robot will take a safe position from which to start its cycle. At the SAW point a piece of fixture like the one the saw would cut is placed at its exit.

To start the experiments, we need to load the code we wrote for the robot model to the Arduino microcontroller through the Arduino program with version 1.8.19. In Fig 6 we see the upload of the code to the microcontroller. After the code is passed we can close the program. The code will now be in the microcontroller and when we turn on the Arduino it can run autonomously without the support of the computer.


ROBOT_TSOULITRAS_Backup | Arduino 1.8.19
File Edit Sketch Tools Help
Upload
ROBOT_TSOULITRAS_Backup

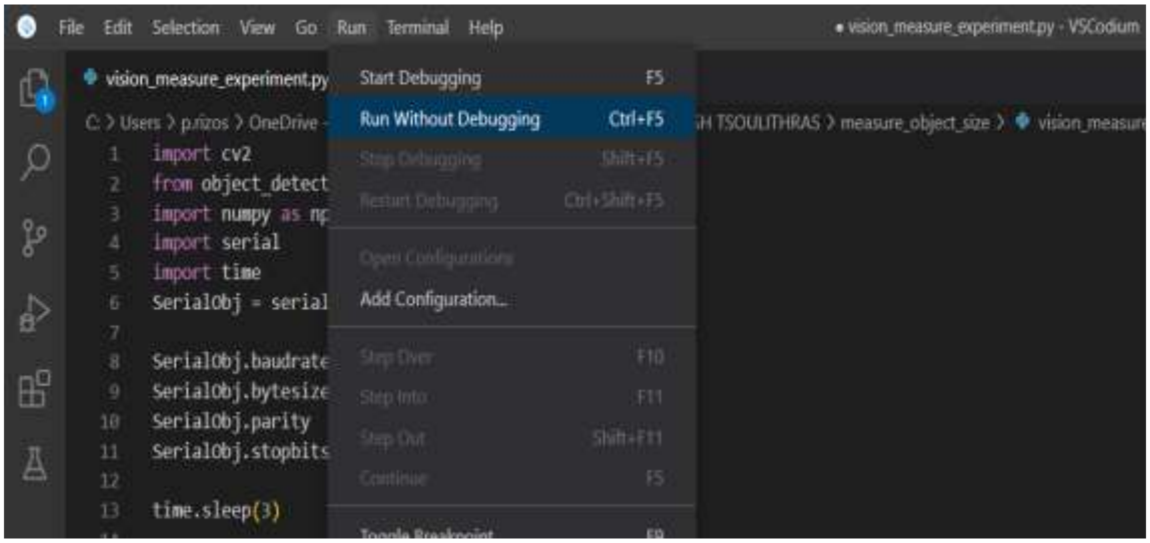
```
#include <BraccioRobot.h>
#include <Servo.h>
#define GRIPPER_HALF_CLOSED 64
#define LED 13

Position myInitialPosition(87, 135, 0, 57, 90,  GRIPPER_CLOSED);
Position readyPosition(60, 135, 0, 57, 90, 10);
Position pickUpPosition(180, 60, 50, 0, 60, 10);
Position putDownPosition;

void setup() {
```

Fig 6. Robot model program upload

Next stage is to run the visual measurement code of section 4.2, as we see in Fig 7. We open the VSCodium program and run the code we wrote in Python [11]. In the camera field next to where the robot will leave the part for measurement, we placed a 5x5 aruco symbol. Arucos are binary square fiducial markers that can be used for camera pose estimation [12]. Their main benefit is that their detection is robust, fast and simple. The code will compare the dimensions of the aruco with the part currently viewed by the camera. An important role in this is played by the dimension of the Aruco, the lighting of the space and the distance of the camera from the component.


File Edit Selection View Go Run Terminal Help
• vision_measure_experiment.py - VSCodium
vision_measure_experiment.py
C: > Users > p.rizos > OneDrive -
H TSOULITHRAS > measure_object_size > vision_measure
Start Debugging F5
Run Without Debugging Ctrl+F5
Stop Debugging Shift+F5
Restart Debugging Ctrl+Shift+F5
Open Configurations
Add Configuration...
Step Over F10
Step Into F11
Step Out Shift+F11
Continue F5
Toggle Breakpoint F9

```
import cv2
from object_detect
import numpy as np
import serial
import time
SerialObj = serial

SerialObj.baudrate
SerialObj.bytesize
SerialObj.parity
SerialObj.stopbits

time.sleep(3)
```

Fig 7. Vision measure program run.

Pressing the optical measurement code to run, the robot model is activated from the safety position and heads to the saw point with the gripper open as shown in Fig 8. After a second time it closes the gripper, rises straight up to go to place the component under the optical measurement camera. The slow speed of the robot model greatly affects the gripping of the part by the gripper. We noticed that on the route we described, the robot trembles. The lower the speed we put in the robot's code, the more it will vibrate during the specific route. The higher the speed we put in the robot code, the smoother it moves between points and the smoother the stops along the way. In a pick and place application as in any other application of robotic systems, accuracy and repeatability play a big role. Accuracy measures how close results are to the true or known value. Precision, on the other hand, measures how close results are to one another. Also, the speed should not be slow. At the specific stage of the experiment, we conclude that the robot model does not show good accuracy and repeatability. This is due to the material of the robot, the 6 servos, the shape of the gripper, and the code that runs serially on the microcontroller and gives the signals to the servos in turn.

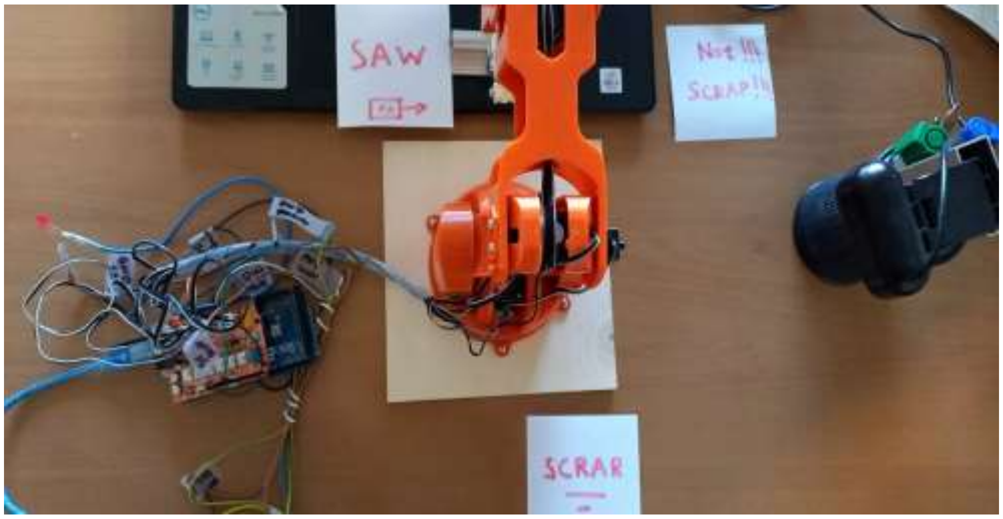


Fig 8. Robot from safe position to pick part from saw.

After the robot model receives the part from the saw and rises straight up it heads towards the camera point. After arriving in front of the camera, it approaches vertically downwards and leaves the part by opening the gripper as we see in Fig 9. After the gripper opens and leaves the part, the robot rises straight up and goes to a safety position until the measured camera accessory. The time the robot will wait in the safety position is not more than 2 seconds and we can set it. Also, the safety position can be placed in another part of the space even closer to the component without the camera seeing it and affecting the measurement of the component. In the path taken by the robot we observed vibrations due to the slow speed of the robot as in the previous path.

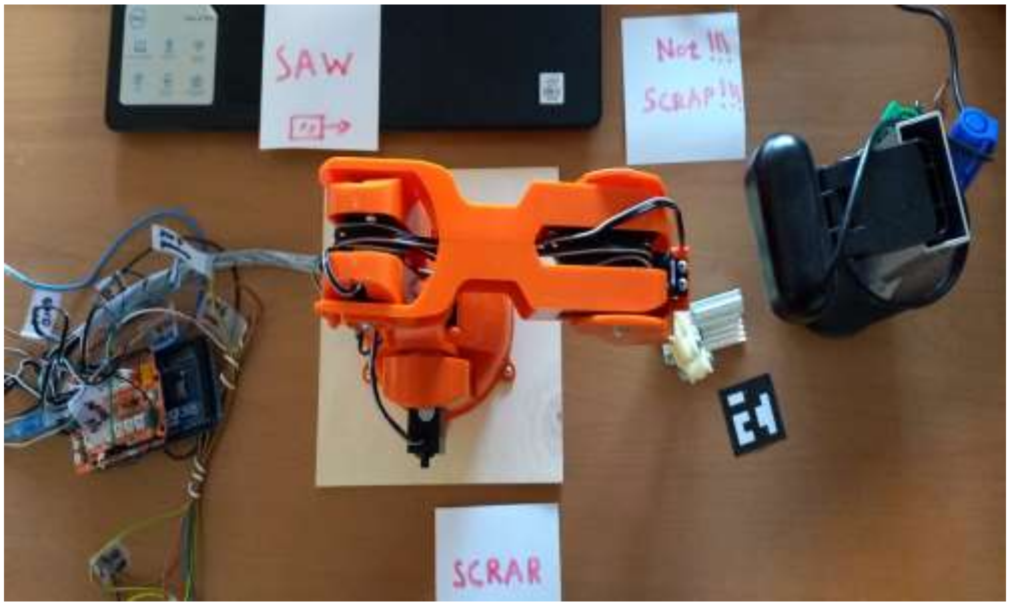


Fig 9. Robot from pick part to camera vision system.

The robot model has left the part to be measured in front of the camera and is directed to the safety position for 2 seconds as we see in Fig 10.

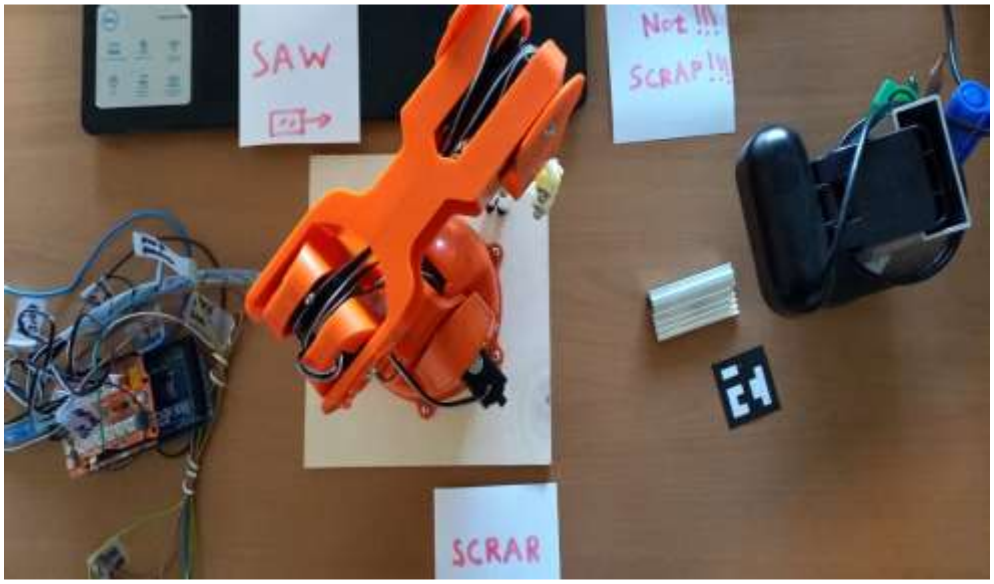
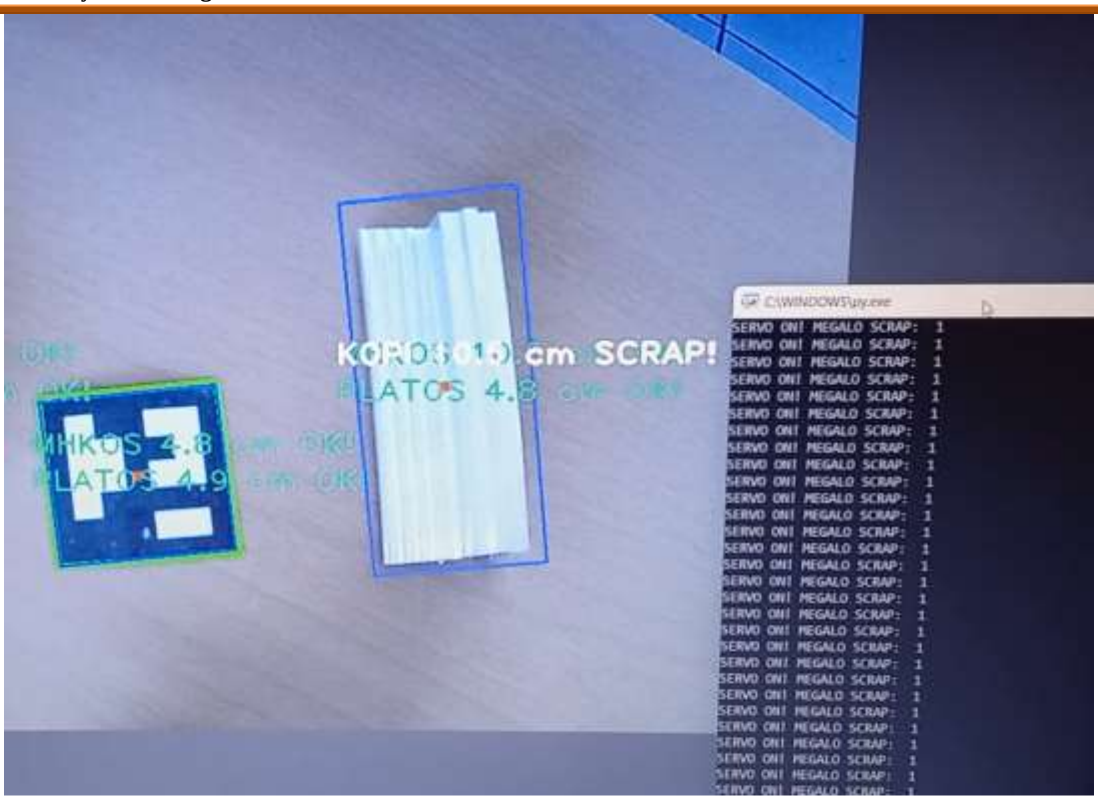


Fig 10. Robot from camera vision system to safe position.

The fixture at this stage is in front of the camera and at 20 cm vertically from it. At the moment the measurement result we get on our computer is for a part with a longer cutting length. This case is scrap and is shown in Fig 11. The program instructs the robot to move towards the camera to grab the part. The robot descends towards the component in a straight line with the gripper open and when it reaches the component, the gripper closes as shown in Fig 12.



```
SERVO ON! MEGALO SCRAP:  1
SERVO ON! MEGALO SCRAP:  1
SERVO ON! MEGALO SCRAP:  1
SERVO ON! MEGALO SCRAP:  1
SERVO ON! MEGALO SCRAP:  1
SERVO ON! MEGALO SCRAP:  1
SERVO ON! MEGALO SCRAP:  1
SERVO ON! MEGALO SCRAP:  1
SERVO ON! MEGALO SCRAP:  1
SERVO ON! MEGALO SCRAP:  1
SERVO ON! MEGALO SCRAP:  1
SERVO ON! MEGALO SCRAP:  1
SERVO ON! MEGALO SCRAP:  1
SERVO ON! MEGALO SCRAP:  1
SERVO ON! MEGALO SCRAP:  1
SERVO ON! MEGALO SCRAP:  1
SERVO ON! MEGALO SCRAP:  1
SERVO ON! MEGALO SCRAP:  1
SERVO ON! MEGALO SCRAP:  1
SERVO ON! MEGALO SCRAP:  1
SERVO ON! MEGALO SCRAP:  1
SERVO ON! MEGALO SCRAP:  1
SERVO ON! MEGALO SCRAP:  1
SERVO ON! MEGALO SCRAP:  1
SERVO ON! MEGALO SCRAP:  1
SERVO ON! MEGALO SCRAP:  1
SERVO ON! MEGALO SCRAP:  1
SERVO ON! MEGALO SCRAP:  1
SERVO ON! MEGALO SCRAP:  1
```

Fig 11. Vision system result.

In the path taken by the robot we noticed vibrations due to the slow speed of the robot as in the previous paths. After the measurement is done and the robot's gripper is closed, we don't care about how it will grip the part if it is Scrap, as long as it doesn't slip along the way. In this case, when the part is scrap, after catching it, the robot will get up to go and leave it at the scrap point.

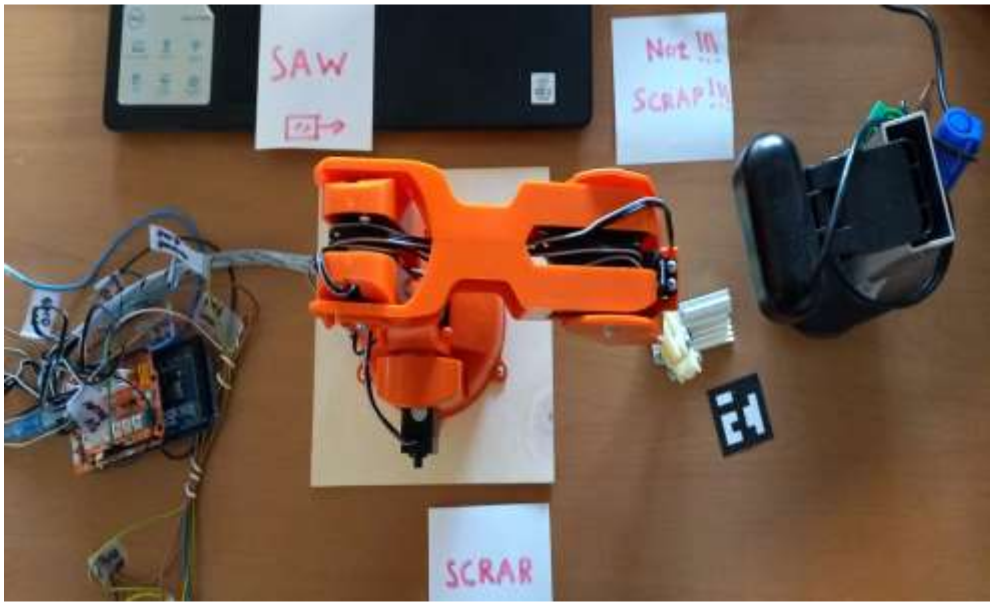


Fig 12. Robot from safe position to camera vision system.

When the robot approaches the scrap point it descends vertically and opens the gripper to drop the part as shown in Fig 13. When the part falls to the scrap point the robot rises vertically up and goes to the safety position and from there back to its point saw to make a pick part of a new part. The new part that the robot will catch will follow the same route and procedure as before, i.e. it will make a pick part, leave the part in the camera and wait 2 seconds in a safe position. If the measurement finds the part OK i.e. it has the correct cut length from the saw, then the program instructs the robot to go pick up the part from the camera plane and place it on the slide at the not scrap point to continue its journey for the next process.

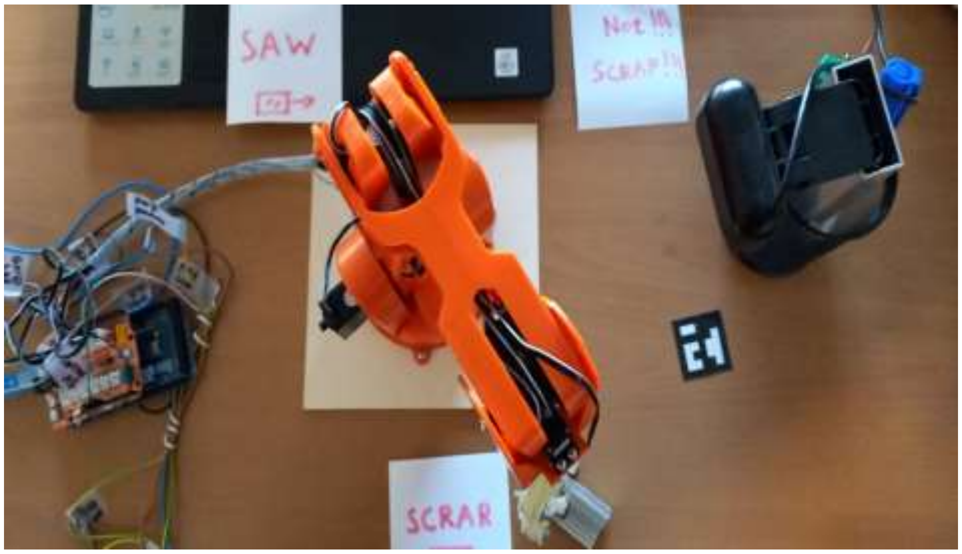


Fig 13. Robot from camera vision system to scrap.

In Fig 14. we see the result of the visual measurement of the component made by the camera program. The part was found to be within cut tolerances and so the scrap note does not appear on the part. The robot at that moment is instructed to leave the safety position and go get the part to take to the slide. And in this path made by the robot we noticed vibrations due to the slow speed of the robot as in the previous paths.

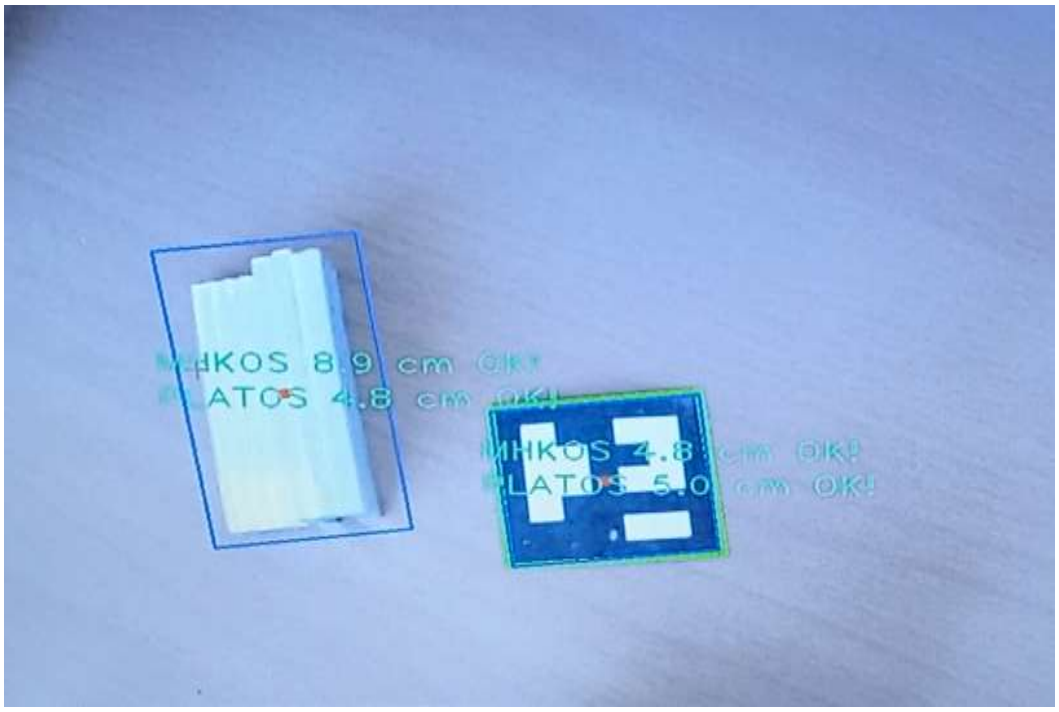


Fig 14. Vision system result.

The robot closes the part on the camera with the gripper, rises vertically upwards and goes to the not scrap point of the slide. Then it descends vertically downwards holding the part with the gripper so that the length of the part is parallel to the slide as shown in Fig 15. After the part reaches the point at a height of 2 cm the gripper opens, and the part falls and rolling down the slide. The robot rises straight up and, passing through a safety position, is directed to re-pick part of a new component that has been cut by the saw. And in this path made by the robot we noticed vibrations due to the slow speed of the robot as in the previous paths.

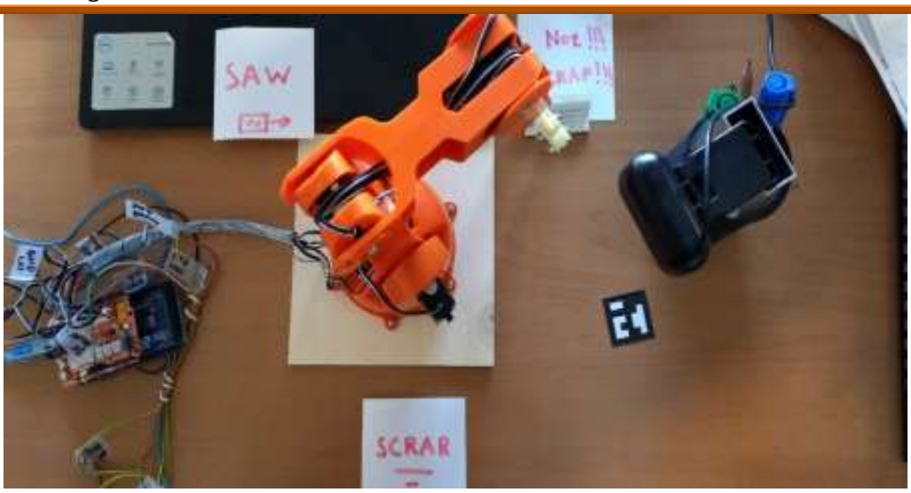


Fig 15. Robot from camera vision system to not scrap.

We know that the absolute position accuracy is the ability of the robot to reach a specific programmed position with a minimum of error. To assess the static accuracy of the robot movement, the position measurements are carried out after a complete stop of the end-effector's movement. Repeatability can be defined as the closeness of agreement between several positions reached by the robot's end-effector for the same controlled position, repeated several times under the same conditions [13].

In the robot model and visual measurement system, 10 experiments corresponding to 10 cycles of the robot were run. The 5 experiments were with a component out of tolerances (scrap), and the other 5 experiments were with a component within tolerances (not scrap). In the 5 robot cycles with part out of tolerance the robot did the path safe position, pick part, camera vision system, safe position, camera vision system, scrap, safe position. In the 5 cycles with a part within tolerances, the robot ran the path safe position, pick part, camera vision system, safe position, camera vision system, not scrap, safe position. The cycle time in the case of a non-scrap part was calculated to be 8 seconds. The same cycle time of 8 seconds was also in the case of an in-tolerance part. In Fig 16 we see the result of 10 cycles of experiments in repeatability and accuracy. The result was that we had bad Repeatability and Accuracy in all 10 cycles.

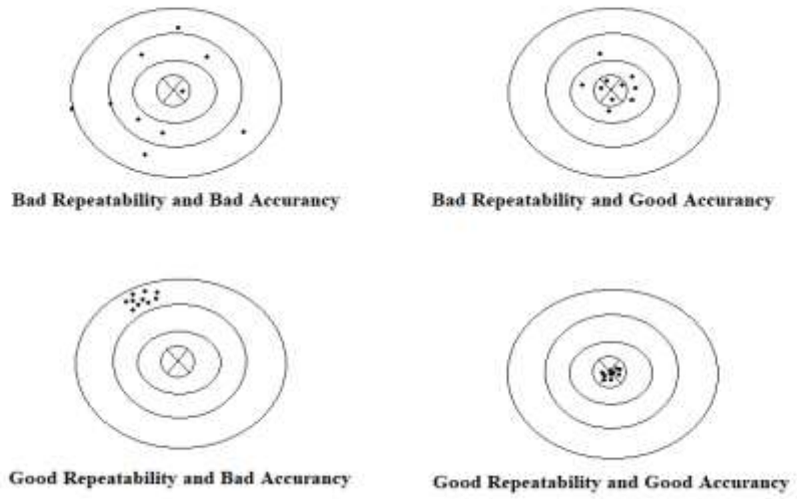

Bad Repeatability and Bad Accurancy
Bad Repeatability and Good Accurancy
Good Repeatability and Bad Accurancy
Good Repeatability and Good Accurancy

Fig 16. Experimental results.

From the results of Fig 16 we understand that the robot system does not perform well. This is due to the construction material of the robot which is plastic. Also, the low speeds needed in the specific case of the saw application create vibrations in the robot body due to the pulse signals received by the servo motors in all six joints. The component, although it generally seems to be left in one place, if we look at it in detail, it is not in the same place in the following cycles. Changing the position of the robot can create a problem in the next cycle and the robot cannot pick the part with its gripper from the pick part position. Even when the robot grabs the part and leaves it in the measurement system with a changed position from the one, we have programmed it can create a wrong measurement from the camera due to the angle of view it will have. To solve these problems, it would be good in future experiments to replace the material of the robot body with a stronger and lighter material such as aluminum. The robot should be doubled in size and the six joint servomotors should be replaced with more powerful ones.

6. CONCLUSION

In this paper, we have presented the laboratory experiments performed on a six-degree-of-freedom robot for an industrial application at ALUMIL Aluminium Industry SA. The robotic arm cooperates with an optical vision system which measures the cutting length dimension of objects. The robotic arm first picks up a saw-cut object and transports it in front of the vision system's camera to be measured. If the object is found out of tolerance, then the arm receives the measured object to transport it to a scrap point. If the object is found within bounds, then the arm receives the object to move it to another point to continue its process in the next stage. 10 experiments were performed corresponding to 10 cycles of the robot. The 5 experiments were with a component out of tolerances (scrap), and the other 5 experiments were with a component within tolerances (not scrap). The experiments showed that the robotic model works well in cooperating with the vision system and performing ten cycles, but future improvement in measurement accuracy is needed for camera to recognize millimeter differences. Also, vibrations occurred in the robot's body during the ten cycles, resulting poor accuracy and repeatability.

7. REFERENCE

[1] Prabhakar, Meenakshi, et al. "Remote Controlled Pick and Place Robot." *IOP Conference Series: Materials Science and Engineering*. Vol. 1012. No. 1. IOP Publishing, 2021.

[2] Wong, Ching-Chang, et al. "Generic Development of Bin Pick-and-Place System Based on Robot Operating System." *IEEE Access* (2022).

[3] Guagliumi, Luca, et al. "Design Optimization of a 6-DOF Cable-Driven Parallel Robot for Complex Pick-and-Place Tasks." *Symposium on Robot Design, Dynamics and Control*. Springer, Cham, 2022.

[4] Appiah, Frank. "An Introduction to Robotic Entertainment." (2021).

[5] https://www.dbusiness.com/daily-news/general-motors-to-invest-51m-at-indiana-foundry/

[6] Reyes, Abel A., et al. "Gesture Controlled Collaborative Robot Arm and Lab Kit." *Conference for Industry and Education Collaboration 2022*. 2022.

[7] Uçar, Kürşad, and Hasan Erdinç Koçer. "Determination of Angular Status and Dimensional Properties of Objects for Grasping with Robot Arm." *IEEE Latin America Transactions* 100.XXX (2022).

[8] Butuner, Resul, and Yusuf Uzun. "Arduino Microcontroller Card." *S. Kocer, O. Dundar* (2021).

[9] Ye, Chen, et al. "Pilot Feasibility Study of a Multi-View Vision Based Scoring Method for Cervical Dystonia." *Sensors* 22.12 (2022): 4642.

[10] Meje, Kelebogile Confidence, et al. "Real-time power dispatch in a standalone hybrid multisource distributed energy system using an Arduino board." *Energy Reports* 7 (2021): 479-486.

[11] Bessai, Jan, George T. Heineman, and Boris Düdder. "Covariant Conversions (CoCo): A Design Pattern for Type-Safe Modular Software Evolution in Object-Oriented Systems (Artifact)." Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2021.

[12] Tocci, Tommaso, Lorenzo Capponi, and Gianluca Rossi. "ArUco marker-based displacement measurement technique: uncertainty analysis." *Engineering Research Express* 3.3 (2021): 035032.

[13] Vocetka, Michal, et al. "Influence of Drift on Robot Repeatability and Its Compensation." *Applied Sciences* 11.22 (2021): 10813.

Author

Panagiotis Rizos,

Robotics Engineer

ALUMIL Aluminum Industry S.A,

Kilkis Industrial Area, Greece.