Ann. For. Res. 66(1): 3664-3673, 2023
ISSN: 18448135, 20652445

ANNALS OF FOREST RESEARCH
www.e-afr.org

DYNAMIC RECONFIGURATION SYSTEM FOR CONTROLLERS OF A CARTESIAN ROBOT

Mantilla Alex
Grupo de investigación ciencias económicas financieras y administrativas CEFIAD, Escuela Superior Politécnica de Chimborazo, Riobamba, Chimborazo 060106, Ecuador, alex.mantilla@espoch.edu.ec ORCID: https://orcid.org/0000-0001-7047-7072

Mejía Nora
Grupo de Investigación y Desarrollo en Agroindustria (IDEA), Escuela Superior Politécnica de Chimborazo, Riobamba, Chimborazo 060106, Ecuador, nora.mejia@espoch.edu.ec
ORCID: https://orcid.org/0000-0002-0308-5412

Haro Diego
Grupo de Investigación EGASIV , Escuela Superior Politécnica de Chimborazo, Riobamba, Chimborazo 060106, Ecuador, diego.haro@espoch.edu.ec
ORCI ID: https://orcid.org/0000-0002-6018-1267

Abstract—This research presents a dynamic reconfiguration based on intelligent agents applied to a Cartesian robot with
2 Degrees of Freedom. This reconfiguration has been carried out using two agents, a main agent and a backup agent, which run on two Raspberry Pi 3 B and communicate through an ethernet network. The tests carried out have shown that the reconfiguration is 100% effective, with an average reconfiguration time of 0.7 seconds, regardless of the time it takes to complete the process. This research demonstrates that dynamic reconfiguration can prevent process disruption with an experimental method.
Index Terms—Robot, Degrees of freedom, Raspberry, Ether- net, Time.

I. INTRODUCTION

The latest advances in technology have led to the develop- ment of many automated and robotic systems that require equipment with high processing power. Therefore, centralized systems have been created. However, as these systems are no longer able to achieve higher levels of performance due to lack of scalability, their use has been surpassed by distributed systems, which have resulted in many advantages and have allowed their expansion worldwide, becoming the main players in the control of automated systems for processes, robots, etc.

Distributed systems refer to a connection of several processing units that behave as a single system to achieve a goal.

This type of system allows users to access resources without needing to know their exact location. This idea was described by Tanenbaum [1] and further developed by [2].

The importance of distributed systems is experiencing sig- nificant changes due to the influence of widespread network technologies, the increasing demand for multimedia services, and the vision of distributed systems as a single unit [3]. This has led to significant advances in distributed systems architecture, improving scalability, enhancing response times, enabling interoperability, and increasing reliability. For ex- ample, various network architectures have been developed, such as those based on containers or microservices, as tools to provide greater efficiency and scalability. Additionally, various communication protocols, such as the User Datagram Protocol (UDP), have been developed and improved to address limitations such as latency. These protocols facilitate commu- nication between distributed nodes, enabling interoperability between systems. Finally, fault tolerance mechanisms, such as replication systems, have been developed to improve the reliability and availability of distributed systems.

Raspberry Pi, with its small dimensions but high features, is an excellent tool for development and prototyping [4]. This platform offers a variety of programming languages for interaction with the outside world. That is why it is used as a controller for Cartesian robotic prototypes or parallel kinematic machines [5]. These robots can be used to carry out technological and scientific projects [6].

The processes involving robots are critical, so it is important to have a dynamic reconfiguration system defined by Han- nebauer Hannebauer [7]. This system is based on distribut- ing problems and solving them through the implementation of agents. Each individual agent is responsible for solving problems, reconfiguring itself autonomously, and adapting to the specific problem. The main things to optimize in dynamic reconfiguration are communication between agents and the quality with which problems are solved. The main idea of dynamic reconfiguration is to adapt the configuration of each autonomously and dynamically of the knowledge, goals, and abilities for problem resolution.

II. METODOLOGY

A. Cartesian Robot Prototipe

The prototype used as a case study is a two-degree-of- freedom Cartesian robot. It has a physical structure of 4 links connected by three joints and two servomotors as actuating elements. Inverse kinematic control has been implemented on two pocket computers, Raspberry Pi 3, which are equipped with an ARM processor and are connected on an Ethernet network with the Arduino Mega 2560, with the help of the W5100 Ethernet shield. The Arduino serves as an interface between the Raspberry Pi and the servomotors.

B. Intelligent Agents

Although the term agent does not have a concrete definition, it is commonly used to refer to a computer system based on software and hardware that must have its control structure to work autonomously, be capable of perceiving changes in its environment, and react to them; another property that an agent must have been the ability to communicate with other agents

2) Inverse Kinematcis Left Arm: If we square the equations 1 and 2

Px2 = l2 cos2 θ1 + 2l1l2 cos θ1 cos(θ1 + θ2) + l2 cos2(θ1 + θ2) (3)

to coordinate actions[8]. 1 2
Communication protocols are those that allow message exchange and understanding, while protocols established for interaction between agents are responsible for maintaining conversations for structured message exchange. The following types of messages can be exchanged between agents.

Py2 = l2 sin2 θ1 + 2l1l2 sin θ1 sin(θ1 + θ2) + l2 sin2(θ1 + θ2) (4)
Adding the equations 3 and 4

- Propose a course of action

Px2 + Py2 − l2 − l2(5)

- Accept the proposed course of action

cos θ2 =1 2 2l1l2

- Reject the proposed course of action
- Withdraw the proposed course of action
- Disagree with the proposed course of action
- Counterpropose a course of action.

Thus, an example of the use of these messages is proposed by Weiss Weiss [9] in which communication between two agents is carried out as follows:

- Agent 1 proposes a course of action to Agent 2.
- Agent 2 evaluates Agent 1's proposal and can proceed in several ways.
  – Send an acceptance to Agent 1 or
  – Send a counterproposal to Agent 1 or
  – Send a disagreement to Agent 1 or
  – Send a rejection to Agent 1.

The equation 5 let stablish left arm's reach, where we can stand out the next cases
|cos θ2| > 1 The point is inaccessible
|cos θ2| = 1 The arm is outstretched
|cos θ2| < 1 It has two solutions to reach the point θ2 y − θ2
Developing the equations 1 and 2 we have:
Px = l1 cos θ1 + l2 (cos θ1 cos θ2 − sin θ1 sin θ2) (6)
Py = l1 sin θ1 + l2 (sin θ1 cos θ2 + sin θ2 cos θ1) (7)
Solving sin θ1 from equation 6
The development of the agents was carried out in Python, an object-oriented programming language that also has easy-to-
sin θ1

= − Px + l1 cos θ1 + l2 cos θ1 cos θ2
l2 sin θ2
use command syntax due to its high level of abstraction, which in turn facilitates its interpretation. All of this was carried out based on the example proposed earlier.
Replacing in the equation 7 we have cos θ1
Pyl2 sin θ2 − Px (l1 + l2 cos θ2) (8)
cos θ1 =(l2 + 2l l cos θ + l2)
1 1 2 2 2 2

C. kinematics

For testing the functioning of the agents, a two-degree-of-
Solving cos θ1 from equation 7
Py − l1 sin θ1 − l2 sin θ1 cos θ2
freedom Cartesian robotic arm was implemented, studying its behavior in solving inverse kinematics.
cos θ1 =l2 sin θ2
Replacing in the equation 6 we have sin θ1
sin θ= Py (l1 + l2 cos θ2) − Pxl2 sin θ2 (9)
1 (l2 + 2l l cos θ + l2)
1 1 2 2 2 2
3) Direct Kinematics of the Right Arm:
Px = lh + l3 cos θ3 + l4 cos(θ3 + θ4)

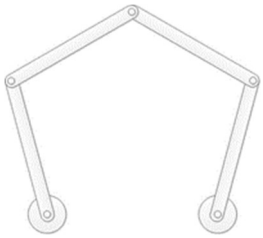

Figure 1: Cartesian Robot 2DOF
1) Direct Kinematics Left Arm :
Px = l1 cos θ1 + l2 cos(θ1 + θ2) (1)
Py = l3 sin θ3 + l4 sin(θ3 + θ4)
Dx = Px − lh Dy = dy
Dx = l3 cos θ3 + l4 cos(θ3 + θ4) (10)
Py = l1 sin θ1 + l2 sin(θ1 + θ2) (2)
Dy = l3 sin θ3 + l4 sin(θ3 + θ4) (11)
4) Inverse Kinematics Left Arm: If we square the equations 10 and 11
Dx2 = l2 cos2 θ3 +2l3l4 cos θ3 cos(θ3 +θ4)+l2 cos2(θ3 +θ4) (12)

D. Simulation of the arm in Matlab

Prior to the physical implementation of the arm, a simu- lation was carried out using CAD tools, in this case, Matlab software, in which the behavior of each of the elements that

3 4 make up the robotic arm can be simulated. All the elements for the simulation are found in the SimMechanics library of Simulink, as shown in Figure 2

Dy2 = l2 sin2 θ3 + 2l3l4 sin θ3 sin(θ3 + θ4) + l2 sin2(θ3 + θ4) (13)

3 4

Adding the equations12 and 13

Dx2 + Dy2 − l2 − l2

cos θ4 =3 42l3l4

Going back to the original variables we have:

(Px − lh)2 + Py2 − l2 − l2 (14)

cos θ4 =3 42l3l4

Equation 14 allows establishing the reach of the left arm, from which the following cases can be highlighted.

|cos θ4| > 1 The point is inaccessible

|cos θ4| = 1 The arm is outstretched

(a) Blocks

|cos θ4| < 1 It has two solutions to reach the point θ2 y − θ2

Developing equations 10 and 11 we have the following equations:

Dx = l3 cos θ3 + l4 (cos θ3 cos θ4 − sin θ1 sin θ4) (15)

Dy = l3 sin θ3 + l4 (sin θ3 cos θ4 + sin θ4 cos θ3) (16)

Solving sin θ1 from equation 15

(b) Links

Figure 2: Simulation in Matlab

Once the prototype was simulated, the robotic arm was constructed with direct drive, since the axis of the actuators,

sin θ

= − Dx + l3 cos θ3 + l4 cos θ3 cos θ4

in this case servo motors, are connected directly to two links,

l4 sin θ4

Replacing in the equation 16 we find cos θ1

while the other two are connected by rotary joints. In Figure 3, the completed prototype can be seen.

cos θ

= Dy4 sin θ4 − Dx (l3 + l4 cos θ4)17)

3 (l2 + 2l l cos θ + l2)

3 3 4 4 4

Solving cos θ3 from the equation 16 and replacing Dy

cos θ

= Py − l3 sin θ3 − l4 sin θ3 cos θ4

l4 sin θ4

Replacing in the equation 15 we find sin θ3

sin θ

= Dy (l3 + l4 cos θ4) − Dxl4 sin θ4

3 (l2 + 2l l cos θ + l2)

3 3 4 4 4

Replacing Dy and Dx we obtain

Py (l3 + l4 cos θ4) − (Px − lh) l4 sin θ4

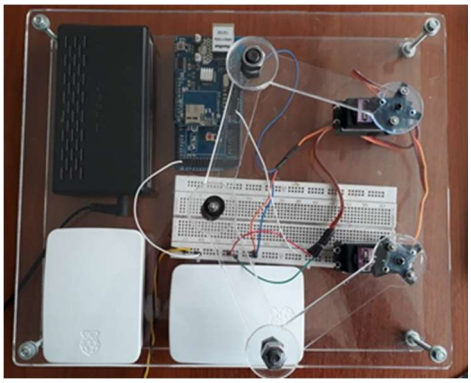

Figure 3: Cartesian Robot

III. RESULTS AND DISCUSSION

For the development of the research, trajectories were

sin θ3 =(l2 + 2l l cos θ + l2)

proposed to be followed by the Robotic Arm, which will be

3 3 4 4 4

defined by the user and will be shown below:

Trajectory 1. This trajectory is composed of 2 points in which the arm will move in a loop 25 times.

Table I: Result with 2 points

	Trajectory 1		
N°	Point 1	Point 2	Reset Time
1	(0, 0)	(2, 2)	0.765 seg.
2	(-2, -2)	(2, 2)	0.703 seg.
3	(0, 0)	(3, 3)	0.647 seg.
4	(0, 4)	(0, -3)	0.713 seg.
5	(-2,	(2, -	0.700

2) 2) seg.

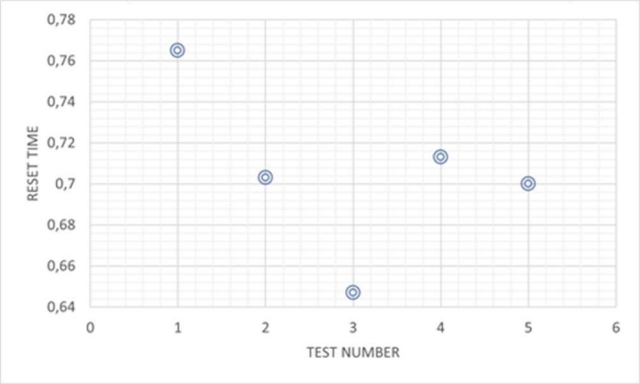

Figure 4: Test 1 vs Time

Trajectory 2.This trajectory is composed of 3 points in which the arm will move in a loop 20 times.

Table II: Result with 2 points

Trajectory 2				
N°	Point 1	Point 2	Punto 3	Reset Time
1	(2, 2)	(0, 0)	(-2, 2)	0.609 seg.
2	(3, 3)	(0, -2)	(0, 4)	0.611 seg.
3	(2, -3)	(0, 0)	(-2, 3)	0.631 seg.
4	(0, -3)	(2, 0)	(0, 3)	0.655 seg.
5	(0, 3)	(-2, 0)	(0, -3)	0.889 seg.

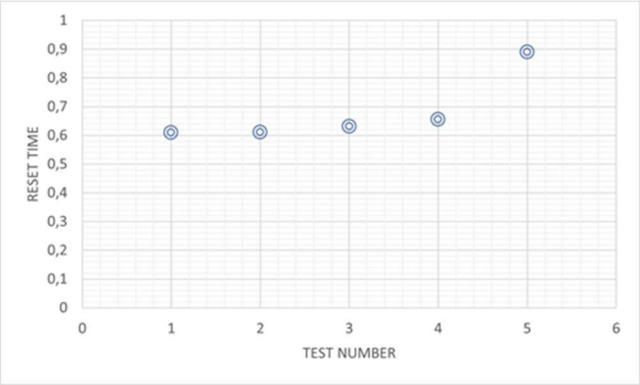

Figure 5: Test 2 vs Time

Table III: Result with 2 points

Trajectory 3					
N°	Point 1	Point 2	Point 3	Point 4	Reset Time
1	(2, -2)	(-2, -2)	(-2, 2)	(2, 2)	0.680 seg.
2	(2, -2)	(0, 2)	(-2, -2)	(2, 2)	0.696 seg.
3	(3, 0)	(3, 3)	(-3, 0)	(3, -3)	0.809 seg.
4	(0, 4)	(0, -3)	(0, 4)	(-2, 0)	0,699 seg.
5	(3, 3)	(0, 0)	(0, -3)	(0, 4)	0.647 seg.

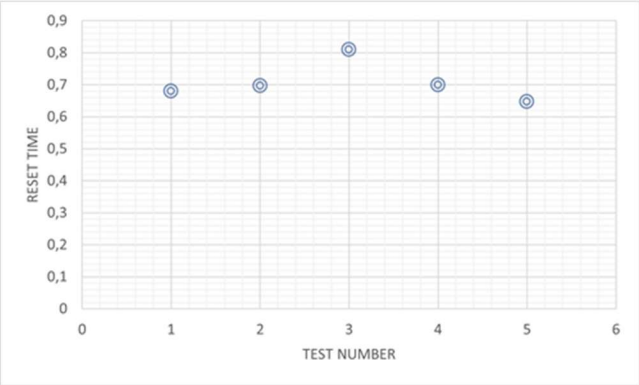

Figure 6: Test 3 vs Time

Trajectory 4.This trajectory is composed of 5 points in which the arm will move in a loop 10 times.

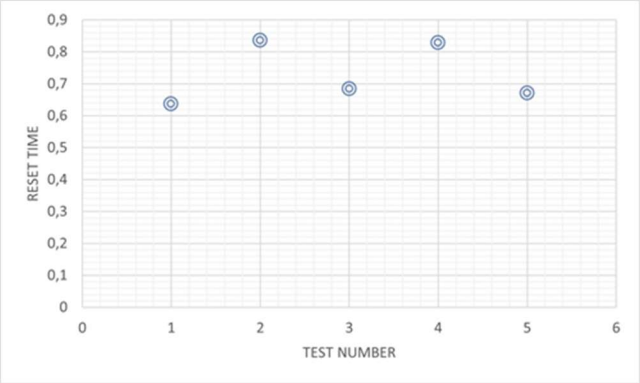

Table IV: Result with 4 points

Table V: Descriptive Statistics

Statistics	T. 1	T. 2	T. 3	T. 4
Mean	0,705	0,679	0,7062	0,7308
Standard Error	0,018	0,0531	0,0273	0,0418
Median	0,703	0,631	0,696	0,684
Standard Deviation	0,041	0,1188	0,0610	0,0936
Sample Variance	0,00176	0,0141	0,0037	0,0087
Kurtosis	1,741	4,5025	3,1886	-3,0289
Skewness	0,0481	2,1043	1,5728	0,4475
Range	0,118	0,28	0,162	0,199
Minimum	0,647	0,609	0,647	0,636
Maximum	0,765	0,889	0,809	0,835
Sum	3,528	3,395	3,531	3,654
Count	5	5	5	5

Trajectory 1 In this trajectory there is greater homogeneity of the data since its standard deviation is the lowest, which means that the data do not deviate much from the arithmetic mean, so there is an average of 0.04 seconds of dispersion.

Trajectory 2 In this trajectory, homogeneity decreases, although a better arithmetic mean is

obtained, but the dataare very dispersed, with a standard deviation of 0.118.

Trajectory 3 In this trajectory, the arithmetic mean is like to the arithmetic mean of trajectory 1, however, the data are more dispersed since the standard deviation is 0.061.

Trajectory 4 In this trajectory, the arithmetic mean is high, which means that it is unfavorable since it takes more time to reconfigure, and it presents a high dispersion of data since the standard deviation is 0.093.

IV. CONCLUSIONES

A structure has been implemented to examine the inverse kinematics of a Cartesian robotic arm with two degrees of freedom, which is controlled by a Raspberry Pi 3 both as the main controller and the backup.

- The designed system allows evaluating if a dynamic reconfiguration is possible without compromising the process and avoiding it to stop.
- The algorithms for the agents make the best decision in assuming control of the process, simulating human behavior in terms of the communication used.
- The system has the capacity to have two agents on each Raspberry, allowing the main controller to be replaced by the backup and vice versa.
- The use of distributed control systems on low-cost plat- forms improves the development of new applications due to their characteristics for both discrete and continuous processes.
- Robots are often used in critical processes, so it is im- portant to have robust control systems, and this research presents a backup control system.
- According to the results, the average reconfiguration time of the control system to solve the inverse kinematics of the Cartesian robot with two degrees of freedom is

0.7 seconds, and according to the statistical analysis (ANOVA), it has been determined that it is within an acceptable range.

REFERENCES

[1] A. S. Tanenbaum, Sistemas Operativos Distribuidos (Spanish Edition). Prentice Hall, 1996.

[2] A. Lafuente, “Introducción a los sistemas distribuidos,”2009.

[3] G. Coulouris, J. Dollimore, and T. Kindberg, Distributed Systems: Concepts and Design, 2012, vol. 4.

Maker Media, Inc, 2013.

[5] J. Rojas, I. Mahla, and G. Muñoz, “Diseño de un sistema robótico cartesianopara aplica- ciones industriales,” Revista Facultad de Ingenieía, vol. 11, pp. 11–16, 2003. [Online]. Available: http://www.redalyc.org/articulo.oa?id=11411102

[6] J. P. Merlet, Parallel Robots. Springer-Verlag GmbH, 2005.

[7] M. Hannebauer, Autonomous Dynamic Reconfiguration in Multi-Agent Systems: Improving the Quality and Efficiency of Collaborative Problem Solving. Springer, 2002.

[8] B. Hill, IP Network-based Multi-agent Systems for Indus- trial Automation British Library Cataloguing in Publica- tion Data. Liverpool: Springer-Verlag GmbH, 2007.

[9] G. Weiss, Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence. The MIT Press, 1999.