ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

"DISEÑO DE RECONFIGURACIÓN DINÁMICA BASADA EN AGENTES INTELIGENTES PARA SISTEMAS DISTRIBUIDOS DE BAJO COSTO"

ALEX SANTIAGO MANTILLA MIRANDA

Trabajo de Titulación modalidad: Proyectos de Investigación y Desarrollo, presentado ante el Instituto de Postgrado y Educación Continua de la ESPOCH, como requisito parcial para la obtención del grado de:

MAGISTER EN SISTEMAS DE CONTROL Y AUTOMATIZACIÓN INDUSTRIAL

Riobamba - Ecuador

Marzo 2019

ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

CERTIFICACIÓN:

EL TRIBUNAL DEL TRABAJO DE TITULACIÓN CERTIFICA QUE:

El **Trabajo de Titulación Modalidad Proyectos de Investigación y Desarrollo**, denominado: "DISEÑO DE RECONFIGURACIÓN DINÁMICA BASADA EN AGENTES INTELIGENTES PARA SISTEMAS DISTRIBUIDOS DE BAJO COSTO", de responsabilidad del señor Alex Santiago Mantilla Miranda, ha sido minuciosamente revisado y se autoriza su presentación.

DR. JUAN VARGAS GUAMBO; MSc.
PRESIDENTE

ING. PABLO LOZADA YANEZ; MSc.
DIRECTOR

ING. FAUSTO CABRERA AGUAYO; MSc.
MIEMBRO

ING. MARCELO GARCIA SANCHEZ; PhD.
MIEMBRO

Riobamba, Marzo 2019

DERECHOS INTELECTUALES

Yo, Alex Santiago Mantilla Miranda soy responsable de las ideas, doctrinas y resultados expuestos en este Trabajo de Titulación y el patrimonio intelectual del mismo pertenece a la Escuela Superior Politécnica de Chimborazo.

ALEX SANTIAGO MANTILLA MIRANDA

No. Cédula 180464206-2

DECLARACIÓN DE RESPONSABILIDAD

Yo, Alex Santiago Mantilla Miranda, declaro que el presente proyecto de investigación, es de mi autoría y que los resultados del mismo son auténticos y originales. Los textos constantes en el documento que provienen de otras fuentes están debidamente citados y referenciados.

Como autor, asumo la responsabilidad legal y académica de los contenidos de este Trabajo de Titulación de Maestría.

ALEX SANTIAGO MANTILLA MIRANDA

No. Cédula 180464206-2

DEDICATORIA

A la mujer que incondicionalmente me ha apoyado, mi madre.

AGRADECIMIENTO

Al ser supremo, por brindarme la fortaleza para no desfallecer en los momentos difíciles de la vida, hacer que las cosas tomen forma en el momento y lugar preciso, A Gisselle León, por su infinita paciencia comprensión y apoyo incondicional, a la señora Carmita Terán por su ayuda incondicional a lo largo de estos años, a mi familia por ser el pilar de apoyo.

A la Escuela Superior Politécnica de Chimborazo, donde he tenido la fortuna de formarme como profesional y la suerte de conocer grandes personas.

A Pablo Lozada mi estimado tutor quien ha estado predispuesto a colaborar en el desarrollo de esta investigación sin objeción alguna.

Alex Mantilla

CONTENIDO

RESUMEN XIII

ABSTRACT XIV

CAPÍTULO I

1 INTRODUCCIÓN 1

1.1 ANTECEDENTES 1

1.2 FORMULACIÓN DEL PROBLEMA 2

1.3 PREGUNTAS DIRECTRICES 2

1.4 JUSTIFICACIÓN 2

1.5 OBJETIVOS DE LA INVESTIGACIÓN 3

1.5.1 Objetivo general 3

1.5.2 Objetivos específicos 3

1.6 HIPÓTESIS 3

CAPÍTULO II

2 MARCO TEÓRICO 4

2.1 SISTEMAS DISTRIBUIDOS. 4

2.1.1 Ventajas y Desventajas de los sistemas distribuidos 5

2.1.2 Elementos de diseño de sistemas distribuidos 6

2.2 RECONFIGURACIÓN DINÁMICA 7

2.3 AGENTES 8

2.3.1 Agente Inteligente 8

2.3.2 Arquitectura 9

2.3.3 Sistema Multiagente 12

2.3.4 Cooperación entre agentes en un Sistemas Multiagente 14

2.3.5 Comunicación entre Agentes 14

2.4 PLATAFORMAS HARDWARE DE DESARROLLO 15

2.4.1 BeagleBone 16

2.4.2 MyRio 17

2.4.3 Raspberry Pi 18

2.5 ROBOT PLANAR 20

CAPÍTULO III

3 DESARROLLO DE LA INVESTIGACIÓN 21

3.1 METODOLOGÍA. 21

3.1.1 Cinemática 21

3.1.2 Fuerzas en el Brazo 28

3.1.3 Simulación del brazo en Matlab 29

3.2 CONSTRUCCIÓN DEL BRAZO ROBÓTICO 30

3.2.1 Eslabón 31

3.2.2 Actuador 31

3.2.3 Articulación 32

3.3 RED DE COMUNICACIÓN 32

3.4 TARJETA DE DESARROLLO 33

3.4.1 Instalación del sistema operativo 34

3.5 DESARROLLO DEL CÓDIGO DE RECONFIGURACIÓN 36

3.5.1 Agente Principal 37

3.5.2 Agente de Respaldo 38

3.5.3 Arduino 40

3.6 PRUEBA DE FUNCIONAMIENTO DE LOS AGENTES 40

3.6.1 Sistema sin Respaldo 41

3.6.2 Sistema con Respaldo 41

3.7 RENDIMIENTO DEL RASPBERRY 43

CAPÍTULO IV

4 ANÁLISIS Y RESULTADOS 46

4.1 DETALLES DEL EXPERIMENTO 46

4.2 ANÁLISIS DE RESULTADOS. 50

4.3 PLANTEAMIENTO DE LA HIPÓTESIS. 51

4.4 PRUEBA DE HIPÓTESIS 52

CONCLUSIONES 54

RECOMENDACIONES 55

BIBLIOGRAFÍA

ANEXOS

ÍNDICE DE TABLAS

Tabla 1-2: Características de Ambientes Multiagentes 14

Tabla 2-4: Resultado con 2 puntos 46

Tabla 3-4: Resultado con 3 puntos 47

Tabla 4-4: Resultado con 4 puntos 48

Tabla 5-4: Resultado con 5 puntos 49

Tabla 6-4: Estadística Descriptiva 50

Tabla 7-4: Descriptivos..... 52

Tabla 8-4: Análisis de Varianza 52

ÍNDICE DE FIGURAS

Figura 1-2: Sistema Distribuido 5

Figura 2-2: Agente BDI 10

Figura 3-2: Agente Reactivo 11

Figura 4-2: Agente Reactivo 12

Figura 5-2: BeagleBone 16

Figura 6-2: NI myRIO 17

Figura 7-2: NI myRIO 19

Figura 8-3: Robot planar 2DOF 21

Figura 9-3: Robot planar 2DOF 29

Figura 10-3: Diagrama de bloques del brazo 30

Figura 11-3: Diagrama de bloques del brazo 30

Figura 12-3: Robot Planar 31

Figura 13-3: Eslabón 31

Figura 14-3: Servo Motor Mg996R 31

Figura 15-3: Articulación 32

Figura 16-3: Arquitectura del sistema distribuido 33

Figura 17-3: Sistema Operativo 34

Figura 18-3: Raspberry y sus Periféricos 34

Figura 19-3: Escritorio Raspbian 35

Figura 20-3: Configuración de IP 35

Figura 21-3: IP Estática de los Agentes 35

Figura 22-3: Arquitectura del sistema distribuido 36

Figura 23-3: Proceso sin Respaldo 41

Figura 24-3: Agente de Respaldo 42

Figura 25-3: Agente Principal 42

Figura 26-3: Reconfiguración 43

Figura 27-3: RPi-Monitor 44

Figura 28-3: Carga del CPU 45

Figura 29-3: Carga del CPU 45

Figura 30-4: Resultado con 2 puntos 47

Figura 31-4: Resultado con 3 puntos 48

Figura 32-4: Resultado con 4 puntos 49

Figura 33-4: Resultado con 5 puntos 50

Figura 34-4: Distribución de ANOVA 53

RESUMEN

En este trabajo se investigó la reconfiguración dinámica basada en agentes inteligentes para sistemas distribuidos de bajo costo, que consiste en evitar la detención de un proceso, mediante la coordinación de los agentes. Los agentes son entidades de software, los mismos que fueron desarrollados en Python un lenguaje de programación orientado a objetos, de código abierto y de alto nivel cuya sintaxis es de fácil interpretación, características que brindan un gran potencial en el desarrollo de los agentes. Los agentes están corriendo en dos Raspberry Pi 3 modelo B, tanto el agente principal, así como el agente de respaldo, en esta investigación se estudia el comportamiento de la reconfiguración dinámica aplicada a procesos discretos, y el caso de estudio es la resolución de trayectorias de un brazo Robótico planar de dos grados de libertad. Se implemento el brazo Robótico el mismo que se compone de dos servomotores como actuadores y 4 eslabones de 10cm de longitud, los servomotores reciben la información de los ángulos un Arduino mega, el mismo que tiene una shield ethernet W5500 para la comunicación en red con los agentes, los ángulos que deben girar los servomotores se basa el estudio de la cinemática inversa. Para verificar el proceso de reconfiguración se evaluó diferentes trayectorias, en los que se somete a un fallo al agente principal evidenciando el comportamiento del agente de respaldo el mismo que es satisfactorio ya que en cualquier fallo retoma el proceso hasta su culminación, con una media de tiempo de 0.7 segundo. El estudio estadístico de análisis de varianza (ANOVA) el valor calculado de Fisher es de 0,4835 mientras que el valor critico de Fisher es de 3.2389 de modo que se acepta la hipótesis nula. Se recomienda Linux como sistema operativo para las Raspberry debido a que es de código abierto.

Palabras Clave: <TECNOLOGIA Y CIENCIAS DE LA INGENIERIA, CONTROL AUTOMATICO, <PROCESOS DISCRETOS>, <RASPBERRY PI 3>, PYTHON (SOFTWARE), <BRAZO ROBÓTICO>, <CINEMÁTICA INVERSA>, <SERVOMOTOR>, <ETHERNET>, <ARDUINO MEGA>

ABSTRACT

In this work, we researched the dynamic reconfiguration based on intelligent agents for distributed systems of low cost, which consists of avoiding the stoppage of a process, through the coordination of the agents. The agents are software entities, the same ones that were developed in Python, a programming language oriented to open source and high-level objects, whose syntax is easy to interpret, characteristics that offer great potential in the development of agents. Both the main agent and the backup agent are running on two Raspberry Pi 3 model B. This research studies the behavior of dynamic reconfiguration applied to discrete processes, and the case study is the trajectory resolution of a planar Robotic arm of two degrees of freedom. The Robotic arm was implemented, which consists of two servomotors as actuators, and 4 links of 10cm in length. The servomotors receive the information of the angles, a mega Arduino, the same one that has an Ethernet shield W5500 for the communication in a network with the agents. The angles that the servomotors must turn are based on the study of inverse kinematics. To verify the reconfiguration process, different trajectories were evaluated, in which the main agent is subjected to failure, evidencing the behavior of the backup agent, which is satisfactory, since in any failure it restarts the process until its completion with an average of 0.7 seconds. In the statistical analysis of variance (ANOVA), Fisher's calculated value is 0.4835, while Fisher's critical value is 3.2389, so the null hypothesis is accepted. Linux is recommended as an operating system for Raspberry because it is open source.

Keywords: <ENGINEERING TECHNOLOGY AND SCIENCE>, <AUTOMATIC CONTROL>, <DISCREET PROCESSES>, <RASPBERRY PI 3>, <PYTHON (SOFTWARE)>, <ROBOTIC ARM>, REVERSE KINEMATICS>, SERVOMOTOR>, <ETHERNET>, <ARDUINO MEGA>.

CAPÍTULO I

1 INTRODUCCIÓN

1.1 Antecedentes

Los avances tecnológicos y el desarrollo de un sin número de sistemas automatizados, robóticos, requieren de equipos con gran capacidad de procesamiento, por lo que se han desarrollado sistemas de tipo centralizado, estos han cumplido con su función y no pueden llegar a un nivel superior.

Los sistemas para la automatización de procesos continuos y discretos tienen una gran gama de soluciones para cubrir las necesidades existentes, de ahí que uno de los principales actores en el mundo de la automatización es el PLC (Controlador Lógico Programable), sin embargo, al trabajar con estos aparatos se concentra todo el proceso en su memoria, dando lugar a inconvenientes con el mismo cuando falla, no es común que suceda ya que son construidos con robustez, pero existe la posibilidad. En vista de aquello, se han diseñado los sistemas de control distribuido (DCS) por sus siglas en inglés, como su nombre lo dice distribuyen el proceso por lo que tienen la capacidad de ser tolerante a fallos.

Todas esas características y prestaciones que brinda un sistema de control distribuido hacen que su costo sea elevado, siendo asequible para las grandes industrias ya sean las petroquímicas, cementeras, farmacéuticas, hidroeléctricas, etc.

En la actualidad se hace uso de sistemas distribuidos; en vista de sus grandes prestaciones se han expandido por todo el mundo llegando a tener un rol importante en el control de los sistemas de automatización de procesos, de sistemas robotizados, etc.

Por esta razón surge la necesidad de crear un sistema distribuido que pueda ser asequible. Finalmente, en el presente trabajo de titulación, se pretende realizar un diseño de reconfiguración dinámica para sistemas distribuidos de bajo costo, brindando una herramienta para el control de procesos críticos.

1.2 Formulación del problema

- ¿Cómo diseñar reconfiguración dinámica basada en agentes inteligentes para sistemas distribuidos de bajo costo?

1.3 Preguntas Directrices

- ¿Cuáles son las ventajas del uso de sistemas distribuidos en procesos discretos?
- ¿Qué nivel de confianza tiene la reconfiguración dinámica basada en agentes inteligentes?
- ¿Qué nivel de potencialidad en tiempos de respuesta provee la reconfiguración dinámica sistemas distribuidos de bajo costo ante una eventual parada del proceso?

1.4 Justificación

Cada vez la electrónica y la automatización de proceso se relaciona con muchas otras disciplinas por lo que la necesidad de obtener controladores robustos que eviten que los procesos cuyo funcionamiento es crítico, se vea comprometido ante una eventual falla del sistema de control, en ese sentido, el presente trabajo tiene como objetivo el estudio de las diferentes configuraciones de agentes inteligentes de software y como sus características pueden ser usadas para procesos de reconfiguración de controladores en sistemas distribuidos, mismos controladores que estarán destinados al control de procesos de tipo discreto, para cubrir las necesidades de dichos procesos críticos.

La investigación está enfocada al aporte de una nueva forma de controlar procesos de carácter discreto, enfocado a la utilización de plataformas de bajo costo que permiten la investigación y desarrollo y debido a las prestaciones que brindan, por lo que se ha selección de la plataforma ira enfocado a la, potencialidad tanto en software así como en hardware, en la misma se implementaran los algoritmos de control del proceso y de reconfiguración del controlador de modo que ante una eventual falla del controlador principal el controlador secundario será capaz de retomar las acciones de control, desde el mismo punto donde surgió la falla del primer controlador.

Por lo tanto, la reconfiguración dinámica de controladores es vital para asegurar que el proceso se ejecute de principio a fin sin la necesidad de que existan grandes pausas de ejecución de los mismos.

Al culminar la investigación aplicando los conocimientos teóricos prácticos se realizará un aporte para todos aquellos que se dediquen al desarrollo y automatización de procesos discretos con carácter crítico.

1.5 Objetivos de la Investigación

1.5.1 Objetivo general

Diseñar la reconfiguración dinámica basada en agentes inteligentes para sistemas distribuidos de bajo costo.

1.5.2 Objetivos específicos

- Describir las ventajas del uso de sistemas distribuidos de bajo costo para el control de procesos discretos.
- Determinar el nivel de confianza de la de reconfiguración dinámica basada en agentes inteligentes.
- Evaluar el nivel de potencialidad en el tiempo de respuesta que provee la reconfiguración dinámica en sistemas distribuidos de bajo costo, frente a una parada del proceso.

1.6 Hipótesis

El número de puntos que conforman las trayectorias inciden significativamente en el tiempo reconfiguración dinámica basada en agentes inteligentes para sistemas distribuidos en plataformas de bajo costo.

CAPÍTULO II

2 MARCO TEÓRICO

2.1 Sistemas Distribuidos.

Los sistemas distribuidos, definido por Tanenbaum (1996) se entiende como la ínter conexión de varios CPU's trabajando como uno solo de forma del conjunto, realice una tarea específica, los en los sistemas distribuidos la ubicación de los recursos es transparente para el acceso de los usuarios(Lafuente, 2009).

La inter conexión entre cada uno de los dispositivos se establece a través de un medio conocido como bus en la actualidad las redes para los sistemas distribuidos pueden ser LAN´s o WAN´s debido a las altas velocidades de transferencia de datos que poseen puesto que van en el orden de los mega bits a la giga bits por segundo, toda esta evolución en los sistemas informáticos computacionales ha aportado en el desarrollo de muchas otras disciplinas del conocimiento.

En ese sentido hay que considerar dos aspectos importantes el hardware y el software, en el primer caso cuando cada CPU, es una unidad autónoma y el segundo se ve al sistema como un único cpu. La coordinación tanto de hardware como de software se realiza a través del paso de mensajes (Coulouris, Dollimore, & Kindberg, 2012).

Las tendencias de los sistemas distribuidos experimentan cambios significativos por la influencia de tecnologías de redes generalizadas, la creciente demanda de servicios multimedia, y la visión de los sistemas distribuidos como una unidad (Coulouris et al., 2012)

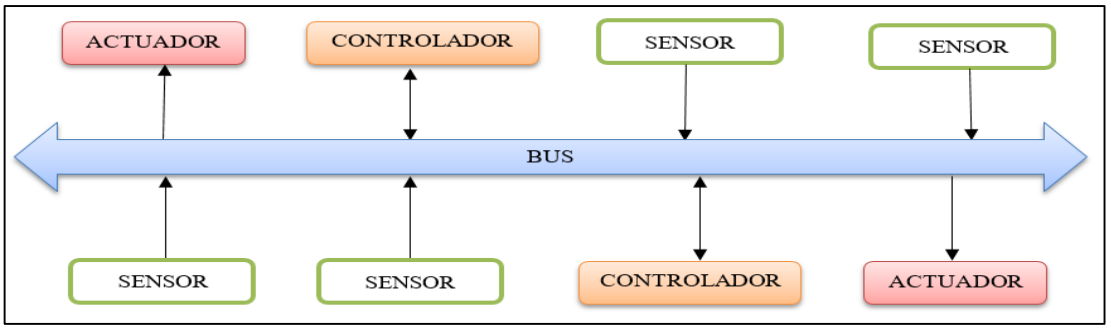


Figura 1-2: Sistema Distribuido
Fuente: Realizado por: Alex Mantilla, 2019

2.1.1 Ventajas y Desventajas de los sistemas distribuidos

Ventajas de los Sistemas Distribuidos

Los sistemas distribuidos debido a sus aplicaciones, las cuales van desde un sistema en un auto, aviones a sistemas mundiales que poseen un gran número de terminales; brindan ciertas ventajas como son:

- De tipo económico debido a que los precios en los procesadores son asequibles en comparación con el rendimiento que brindan.
- En función de la velocidad debido a la mayor capacidad de procesamiento en comparación con un sistema centralizado.
- La distribución de las aplicaciones en varias terminales remotas hace que sea una ventaja inherente.
- La confiabilidad es una de sus mejores ventajas debido a que si falla uno de los terminales el sistema en conjunto no se ve comprometido.
- El crecimiento del sistema se puede efectuar cada vez que se requiera una mayor capacidad de computo
- Compartir datos, hace que sea posible que desde los diferentes terminales del sistema distribuido se pueda acceder a repositorios de conocimiento en común.

Desventajas de los Sistemas Distribuidos

A pesar de las ventajas ya expuestas, existen desventajas, aunque no podríamos hablar de desventajas como tal sino de retos por resolver.

- El software dedicado a los sistemas distribuidos aún se encuentra en desarrollo.
- En las redes existen varios tópicos por solucionar como la transferencia de grandes volúmenes de información.
- La seguridad necesita ser mejorada para asegurar la confidencialidad de la información.
- Las fallas operativas aún son frecuentes.

2.1.2 Elementos de diseño de sistemas distribuidos

Para diseñar un sistema distribuido hay varios elementos que se deben considerar, de modo que se asegura el correcto desempeño del sistema distribuido; de los que se destacan:

1. **Transparencia**

   Por la propia definición de sistema distribuido, la cual hace hincapié en la separación de los dispositivos, por lo que se presenta una desventaja para los usuarios y programadores de aplicaciones, por la complejidad añadida en los sistemas distribuidos, por lo que resulta imperioso ocultar esta complejidad, de modo que, se pueda presentar como un sistema centralizado (Arno Puder Kay Rarmer, 2005).

2. **Flexibilidad**

   La flexibilidad en los sistemas distribuidos hace referencia a la capacidad de soportar cambios, actualizaciones y mejoras.

3. **Confiabilidad**

   Este término de confiabilidad ya da por si solo una descripción a lo que se hace referencia sin embargo en sistemas distribuidos, se enfoca en el garantizar la ejecución de los procesos de modo que, si un cpu fallase, algún otro cpu sea capaz de ejecutar el proceso hasta su culminación.

4. **Desempeño**

   El desempeño es un concepto que está implícito, a pesar de haber cumplido con los elementos anteriores, no se garantiza que el desempeño sea adecuado del sistema

distribuido, en ese sentido existen diversas métricas del desempeño; como es el tiempo de respuesta, aunque también se puede evaluar por el número de tareas ejecutadas por una unidad de tiempo a la cual se requiera hacer referencia, la cantidad de consumo de la red, entre otros.

5. **Escalabilidad**

   Este es uno de los elementos más importantes dentro de todo sistema distribuido de cualquier índole a la que se haga referencia debió a la facilidad que brinda en el crecimiento del mismo sistema al incorporar nuevos dispositivos, aunque no todo resulta perfecto en vista que la evolución tecnológica de los nuevos dispositivos pueden ser compatibles para la ejecución de ciertas tareas, mientras que para otras no lo sean, por lo que es uno de los puntos que hay que tomar en cuenta al momento de incluir nuevos dispositivos en el sistema (Tanenbaum, 1996).

2.2 Reconfiguración Dinámica

La reconfiguración por si sola se entiende como la variación de parámetros, variables o estados iniciales para que la correcta ejecución de un proceso sea la idónea, en cuanto se refiere a procesos informáticos, electrónicos, robóticos, etc.; Por la naturaleza de este tipo de procesos se requiere disminuir el tiempo de modificar dichos parámetros variables o estados por lo que se introduce el concepto de reconfiguración dinámica.

Una arquitectura reconfigurable es aquella que puede alterar las funcionalidades de sus componentes y la estructura que conecta estos componentes. Cuando la reconfiguración es rápida con poca o ninguna sobrecarga, se dice que es dinámica (Vaidyanathan & Trahan, 2004).

La reconfiguración dinámica definida por Hannebauer (2002) se basa en la distribución de los problemas y dar solución a los mismos, por medio de la implementación de agentes, por lo que cada agente individual está encargado de resolver problemas, re configurarse de forma autónoma y adaptarse al problema en cuestión, las principales cosas que se deben optimizar en la reconfiguración dinámica es la comunicación entre agentes y la calidad con la que se resuelven los problemas. La idea principal de la reconfiguración dinámica consiste en adaptar de forma autónoma y dinámica la configuración de cada uno de los conocimientos, metas y habilidades para la resolución del problema.

La reconfiguración dinámica tiene varias ventajas como la que presenta al mejorar la utilización de recursos, gracias a la adaptación de la funcionalidad del hardware, para la resolución de tareas o problemas. Por otro lado, está la facilidad de adaptar su arquitectura para explorar las características de un problema. Además, plantea nuevos hitos en cuanto a la escalabilidad algorítmica.

La reconfiguración dinámica es muy prometedora para una computación rápida y eficiente. Como un coprocesador dedicado, una arquitectura reconfigurable puede entregar velocidades mucho más allá de la capacidad de los enfoques convencionales. Para un entorno más general, puede establecer un equilibrio entre velocidad y eficiencia mediante la reutilización de hardware para adaptarse a la computación. De hecho, varias aplicaciones de procesamiento de imágenes y video, criptografía, procesamiento de señales digitales y redes utilizan esta característica (Vaidyanathan & Trahan, 2004).

2.3 Agentes

Aunque el termino agente no esté posea una definición concreta el termino agente es comúnmente utilizado para denotar un sistema informático basado en software y hardware los mismos que deben tener su estructura de control, para trabajar de manera autónoma, ser capaces de percibir los cambios de su entorno y reaccionar ante los mismos; otra de las propiedades con las que debe contar un agente es la capacidad de comunicación con otros agentes para poder coordinar las acciones (Hill, 2007).

El trabajo con agentes como tal se ha convertido en un paradigma para el desarrollo y moldeamiento de sistemas con un grado de complejidad alto, por lo que su campo de aplicación es bastante amplio que va desde el campo de las fianzas pasando por la medicina hasta la electrónica y Robótica.

2.3.1 Agente Inteligente

La definición más acertada de agente inteligente es la expuesta por (Michael Wooldridge, 1995) en la que destaca a un agente como una entidad de software que debe cumplir con las siguientes características o propiedades.

- **Autonomía**

  Un agente debe ser capaz de trabajar con independencia y sin la intervención de un humano u otro, en la solución de problemas.

- **Sociabilidad**

  Un agente inteligente para poder ser llamado como tal debe ser capaz de comunicarse con otros agentes, además de mantener sino una comunicación con los humanos una interacción.

- **Reactividad**

  Los agentes deben ser capaces de percibir su entorno, el cual puede ser el mundo real, así como una colección de otros agentes, la mayor red mundial como es Internet o también se puede ser un entorno híbrido.

- **Iniciativa**

  Los agentes no deben actuar simplemente en respuesta a su entorno, sino también ser capaces de exponer un comportamiento dirigido a objetivos tomando la iniciativa.

2.3.2 Arquitectura

La arquitectura de agentes define la manera en la que se interceptaran los componentes del sistema tanto en hardware, así como en software de modo que se siga manteniendo la definición de característica de agente. Por lo que encontrar una gran cantidad de arquitecturas es posible debido a la gran cantidad de avances en dispositivos existentes como expone Jose Molina (2005); en tal virtud se presentan los tipos de arquitecturas más relevantes.

- **Arquitectura Deliberada**

  Las arquitecturas arquitectura deliberadas están basadas en la teoría de una planificación, además de tener un punto inicial y un objetivo que cumplir, uno de los ejemplos de arquitectura deliberada es la arquitectura BDI cuya abreviatura se encuentra en el idioma ingles y su traducción es Intención - Deseo - Creencia, conocida por que el desempeño

del agente hace referencia a su nombre, debido a que la base del conocimiento está basado en un conjunto de creencias que es aquello que considera el agente que es verdad, los deseos son las condiciones desea cumplir y finalmente las intenciones que son las acciones que está comprometido a cumplir el agente (Michael Wooldridge, 1995).

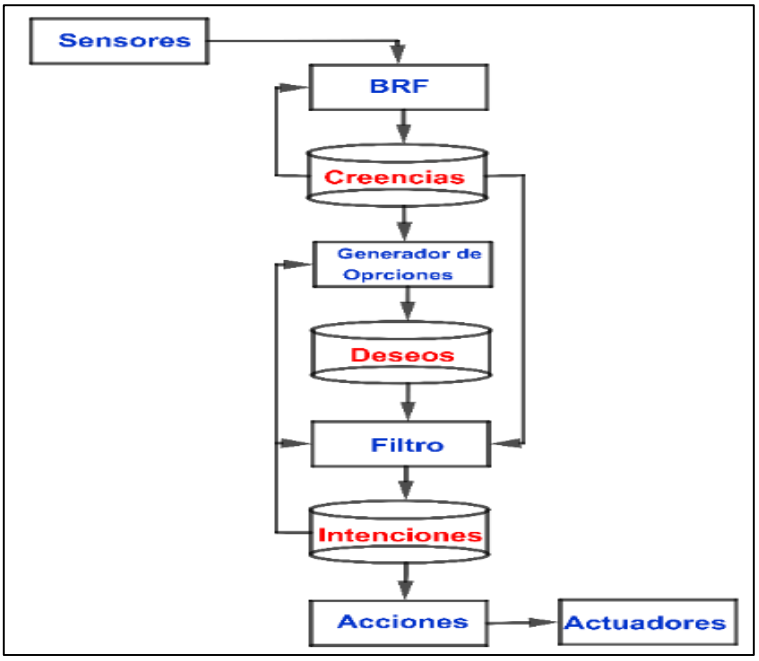


Figura 2-2: Agente BDI
Realizado por: Alex Mantilla, 2019

- **Arquitectura Reactiva**

Esta arquitectura está caracterizada por su razonamiento descentralizado, además de poseer un razonamiento simbólico sencillo, además de operar en forma rápida y efectiva. Los agentes reactivos toman sus decisiones basadas en el presente, sin tomar en cuenta los eventos del pasado, debido a que no poseen información del mismo. Los agentes que se encuentran en esta jerarquía tienen la capacidad de reaccionar a un entorno sin razonar al respecto.

Un ejemplo de arquitectura reactiva es la arquitectura de subsunción, la cual fue desarrollada por Rodney Brooks cuya característica es la toma de decisiones a través de un conjunto de comportamientos al realizar de tareas; sin embargo cada comportamiento puede ser considerado una acción individual (Weiss, 1999).

Un agente está conformado por módulos, los mismos que son capaces de interactuar con el entorno, además cada módulo posee competencias específicas que se encarga de una tarea bien definida, el conjunto de módulos resuelve la tares completa, sin embargo no

puede resolver tareas para las cuales no exista un módulo de competencias, a diferencia de las arquitecturas deliberadas (Group, 2000).

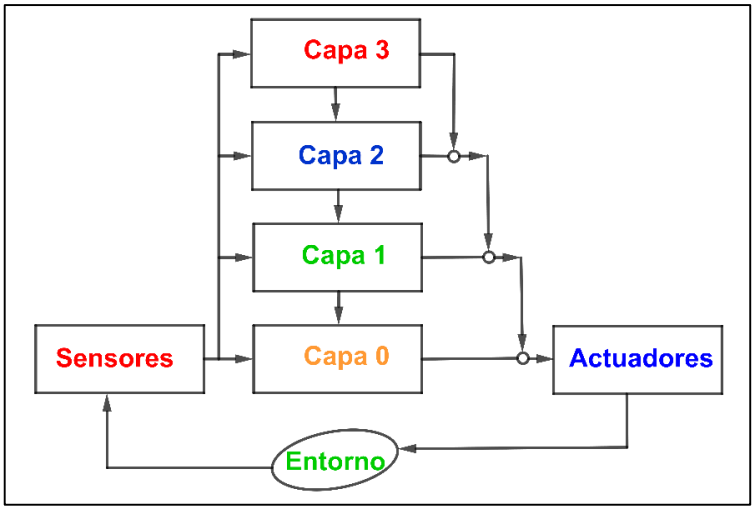


Figura 3-2: Agente Reactivo
Realizado por: Alex Mantilla, 2019

- **Arquitecturas Híbridas**

Las arquitecturas híbridas combinan las características de una arquitectura de tipo deliberada y del tipo reactivo es por esta razón que los elementos de la arquitectura reactiva interactúa con el entorno de modo que los módulos con competencias específicas resuelven los eventos o tareas que se presentan, acortando el tiempo ya que no utilizan razonamiento para hacerlo; de esta manera la parte restante de la composición de una arquitectura híbrida es la parte deliberada encargada de planificar y tomar decisiones, y por esta razón se encuentra en un nivel de abstracción superior.

La forma de composición de una arquitectura híbrida es ventajosa debido a que no todos los problemas pueden ser tratados de forma deliberada o de forma reactiva, al realizar la combinación se puede aprovechar el potencial de cada una de las arquitecturas para mejorar la resolución de los problemas. la arquitectura híbrida se diseña en capas siguiendo su conceptualización cono se muestra en la Figura 4-2.

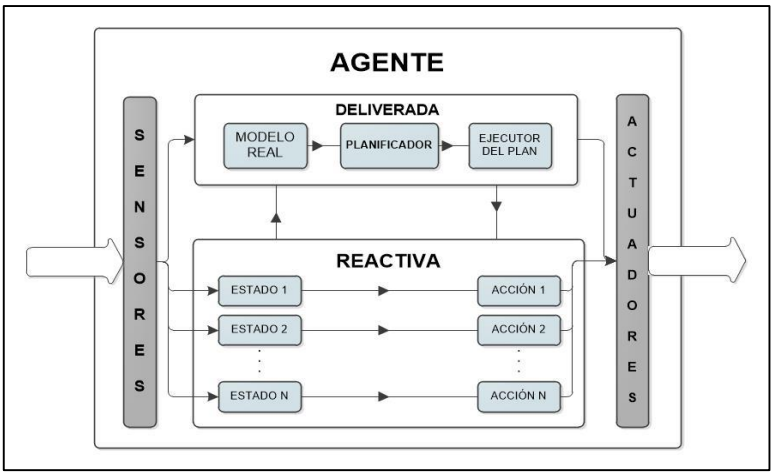


Figura 4-2: Agente Reactivo
Realizado por: Alex Mantilla, 2019

2.3.3 Sistema Multiagente

Los agentes trabajan en los entornos informáticos tanto en software como en hardware, en muchas ocasiones un agente es capaz de resolver tareas sin presentar mayor dificultad, el constante crecimiento de la interconexión y la evolución de las redes informáticas, ha dado lugar a que los agentes interactúen entre sí, debido a que la cantidad de agentes en ocasiones suele ser muy numerosa se habla de una sociedad de agentes.

Las sociedades de agentes se desarrollan en entornos, los mismos que permiten que los agentes puedan operar adecuadamente e interactuar entre sí de manera más productiva: dichos entornos proporcionan la infraestructura adecuada para que las interacciones entre agentes se lleven a cabo, por lo que es necesario que existan o se incluyan protocolos tanto para poder establecer la comunicación entre agentes, así como para que la interacción sea óptima.

Los protocolos de comunicación son aquellos que permiten el intercambio de mensajes y la comprensión de los mismos, mientras que aquellos protocolos establecidos para la interacción entre agentes se encargan de mantener conversaciones para el intercambio estructurado de mensajes.

Los siguientes tipos de mensajes pueden ser intercambiados entre agentes.

- Proponer una acción en curso
- Aceptar la acción en curso
- Rechazar la acción en curso
- Retirar la acción en curso
- Desacuerdo con la acción en curso propuesta

- Contra proposición de la acción en curso

De este modo un ejemplo de la utilización de estos mensajes es el propuesto por Weiss (1999) en el que la comunicación entre dos agentes se lleva a cabo de la siguiente manera:

- Un Agente 1 propone una acción en curso al Agente 2.
- El Agente 2 evalúa la propuesta de Agente 1 y puede proceder de varias formas.
    - Envía una aceptación al Agente 1

      o
    - Envía una contra propuesta al Agente 1

      o
    - Envía un desacuerdo al Agente 1

      o
    - Envía un rechazo al Agente 1

Los sistemas de agentes centralizados suelen ser más eficientes por la concentración de información y tareas, sin embargo, de todos los cálculos que se pueden efectuar en los sistemas distribuidos de agentes también se puede optimizar en un solo cpu para ser tan eficiente como un sistema centralizado.

El enfoque centralizado no siempre es posible debido a la arquitectura de la organización ya que los datos pueden estar en entidades aisladas e independientes. Los sistemas distribuidos se dan lugar en vista de que la información es distribuida y reside en sistemas grandes y complejos de manejo de información como pueden ser:

1. Geográficamente distribuidos
2. Pueden tener muchos componentes
3. Pueden tener un contenido enorme, tanto en el número de conceptos como en la cantidad de datos sobre cada concepto
4. Pueden tener un alcance amplio.

Por esta razón hay cuatro técnicas para el tratamiento del tamaño y complejidad de los sistemas de información como son:

- Modularidad
- Distribución
- Abstracción
- Inteligencia

Mientras más inteligente se aun agente mayor será su capacidad de búsqueda para el manejo de información, en tal virtud el uso de módulos inteligentes distribuidos puede combinar de cualquier forma las técnicas imitando la inteligencia humana para dar lugar a la inteligencia artificial.

Los sistemas multiagentes son la mejor manera de caracterizar o diseñar sistemas de computación distribuidos. El procesamiento de la información es omnipresente. Hay procesadores de computadora aparentemente en todas partes, incrustados en todos los aspectos de nuestro entorno.

Tabla 1-2: Características de Ambientes Multiagentes

Propiedad	Rango de Valores
Plataforma de Diseño de Autonomía	Protocolo de Interacción / Idioma / Arquitectura Interna
Infraestructura de comunicación	Memoria compartida pizarra) o basada en mensajes Conectado o sin conexión (correo electrónico) Punto a punto, Multicast o Difusión Empujar o tirar Sincrónico o asincrónico
Servicio de directorio	Páginas blancas, Páginas amarillas
Protocolo de mensajes	KQML, HTTP y HTML OLE, CORBA, DSOM
Servicios de Mediación	¿Basada en Ontología? ¿Actas?
Servicios de seguridad	Sellos de tiempo / Autenticación
Servicios de Remesas	Facturación / Moneda
Soporte de Operaciones	Archivo / Redundancia / Restauración / Contabilidad

Realizado por: Alex Mantilla, 2019

2.3.4 Cooperación entre agentes en un Sistemas Multiagente

Para que la cooperación de agentes sea eficiente en un sistema multiagente, se debe establecer criterios lógicos y físicos en la comunicación y existencia del agente, así como la inclusión de protocolos para interacción, por otro lado, es imperioso la escalabilidad de agentes y una adecuada infraestructura tecnológica para el desarrollo del sistema multiagente y lograr la reconfiguración.

2.3.5 Comunicación entre Agentes

La comunicación es la parte más importante de sistema multiagentes por lo que se debe cumplir con los siguientes criterios.

- El lenguaje en que se desarrollen los agentes debe ser lo suficientemente claro para poder interactuar frente a restricciones y para la resolución del problema.
- Los protocolos de comunicación desarrollada en el lenguaje de programación de agentes deben ser formales de modo que permita verificar su idoneidad y su corrección, debido a que en gran medida la reconfiguración del sistema depende de los protocolos empleados.
- La comunicación debe ser exclusiva para los agentes y ser capaz de ser funcional con un pequeño número de agentes, así como para un número elevado de agentes con mayor complejidad.

La comunicación entre agentes está basada en el intercambio de proposiciones, reglas y acciones, por lo que la comunicación entre agentes se basa en el concepto lingüístico de las acciones que pretende realizar.

2.4 Plataformas Hardware de Desarrollo

Las plataformas de desarrollo juegan un papel importante en el diseño, prueba y creación de nuevos prototipos electrónicos, los mismos que están compuestos de elementos de hardware y software, en ese sentido estas plataformas permiten reducir el tiempo de desarrollo, debido a que el investigador se enfoca en las prestaciones y objetivos a cumplir de su prototipo ya que de los detalles de bajo nivel ya están cubiertos por los fabricantes de las plataformas, por otro lado ayuda verificar la confiabilidad del prototipo, entre otras cosas.

En un proyecto se ve involucrado el hardware y el software que se encuentra ejecutándose en el hardware, por lo que en la construcción del hardware se recurre a varios componentes de diversos fabricantes para consolidarse y trabajar como uno solo, además es imperioso que dichos componentes establezcan una comunicación adecuada, para su correcto desempeño, es por esto que iniciar un proyecto desde cero tratando de elaborar el hardware y el software llevaría mucho tiempo, pudiendo incrementar considerablemente el costo del prototipo a desarrollar, por tal razón se recurre a plataformas de desarrollo electrónico de las que destacaremos algunas plataformas con microprocesador haciendo una comparación en costo beneficio.

2.4.1 BeagleBone

Es una placa de desarrollo, la cual trae una distribución de Linux incrustada, la misma que es de software y código abierto y utilizan los procesadores de Texas Instrument, con un núcleo de la serie ARM Cortex-A, diseñados y desarrollados para dispositivos móviles de bajo consumo.

BeagleBone tiene una conexión Ethernet integrada, además posee las herramientas básicas que vienen empaquetadas en Linux, por lo que el acceso remoto es posible permitiendo el manejo de proyectos desde una red remota, o Internet. Esta placa tiene la capacidad de realizar un seguimiento de la hora y fecha o actualizarla accediendo a servidores de Internet, como un ordenador personal está en la posibilidad de almacenar, organizar y recuperar datos, tiene la posibilidad de escribir código en diferentes lenguajes de programación como: C, C ++, Python, Perl, Ruby o incluso un script de Shell.

El posible realizar muchas tareas en BeagleBone debido a que la mayoría del software desarrollado para Linux, se puede ejecutar en la tarjeta. Es así que se puede hacer uso de una cámara web, realizar procesamiento de imágenes entre otras cosas, Por estar basada en Linux existe gran documentación para casi cualquier tema en el que esté interesado (Richardson, 2013).

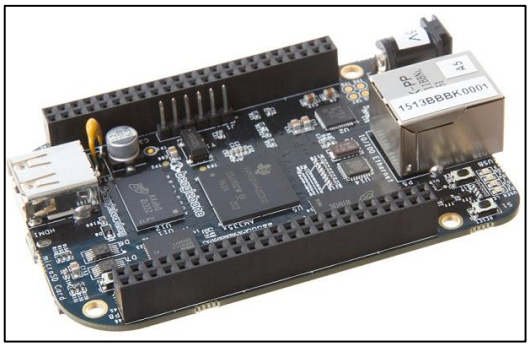


Figura 5-2: BeagleBone
Fuente: http://elinux.org/Beagleboard:BeagleBoneBlack, 2019

Las características que posee esta tarjeta y que hacen que BeagleBone sea llamativa son las siguientes:

- Su procesador ARM Cortex-A8 de 1Ghz, además posee una memoria sdram de 512 MB DDR3L.
- La tarjeta necesita una fuente de alimentación de 5Voltios y 500 mA de corriente directa, en caso de suministrar un mayor nivel de voltaje posee un regulador de voltaje, la

alimentación se realiza por medio de un puerto mini USB, común en muchos de los dispositivos electrónicos.

- El puerto Ethernet es a través de un RJ45, para conectarse directamente a un router y acceder a Internet.
- BeagleBone está equipada con puertos USB de modo que pueden conectar periféricos como teclados, mouse, adaptadores bluethoot, o wifi.
- Puertos de entrada y salida que pueden ser configurados para ser analógicos o digitales.
- Un slot para una tarjeta Micro SD que contiene el sistema operativo.
- Indicadores led para dar a conocer el estado de funcionamiento de la tarjeta, así como el estado del puerto Ethernet.
- Un micro puerto HDMI para mostrar la imagen en un monitor o un televisor
- Botones para iniciar y reiniciar la tarjeta.

2.4.2 MyRio

La tarjeta de desarrollo denominada myRIO es un dispositivo desarrollado por la National Instrument, enfocado para proceso de enseñanza aprendizaje, y desarrollo de proyectos embebidos que requieran ser ejecutados en tiempo real, cuenta con un procesados ARM Cortex-A9 de 667 MHz, además cuenta con una FPGA incrustada, para el desarrollo de de problemas de diseño complicado, también incluye un chip Zynq-7010 para poder utilizar todo el potencial que brinda Labview, ya que mediante este software se pueden desarrollar las aplicaciones.

En Labview se puede realizar una imitación de la apariencia de instrumentos reales, como generadores de señales, osciloscopios, multímetros entre otros, por lo que los objetos de Labview se denominan instrumentos virtuales (VIs), la programación es orientada a objetos utilizando bloques de funciones los mismos que interceptados mediante líneas de flujo de datos generan las aplicaciones (Instruments, 2016).

Figura 6-2: NI myRIO
Fuente: http://www.ni.com/es-cr/shop/select/myrio-student-embedded-device, 2019

Las características de esta tarjeta de desarrollo para el desarrollo de prototipos son las siguientes:

- Utilizando herramientas de tiempo real, FPGA y capacidades integradas de Wi-Fi, junto con la memoria integrada.
- Tres conectores (dos puertos NI myRIO expansión [MXP] y un puerto de NI miniSystems [MSP] que es idéntico al conector NI myDAQ) envían y reciben señales desde los sensores y circuitos que los estudiantes necesitan en sus sistemas.
- Cuarenta líneas de E/S digitales, con el apoyo de SPI, PWM, entrada de codificador de cuadratura, UART e I2C.
- Ocho entradas analógicas de una sola terminal.
- Dos entradas analógicas diferenciales.
- Cuatro salidas analógicas unipolares.
- Dos salidas analógicas con referencia a tierra permiten la conectividad a un sinnúmero de sensores y dispositivos y control de programación de sistemas.
- Acelerómetro incluido para tres ejes.

En última instancia, estas características ayudan a los estudiantes hacer la ingeniería del mundo real en este momento, desde los vehículos de radio control hasta la creación de dispositivos médicos independientes.

2.4.3 Raspberry Pi

La idea general que viene a la mente en cuanto se habla de Raspberry es la de una tarjeta de reducidas dimensiones, con acceso a todo su contenido y orientado a la educación, sin embargo, su potencial es mucho mayor, ya que a simple vista parece una tarjeta basada en microcontroladores, pero está basada en un microprocesador además de poseer conectores para establecer comunicación con otros periféricos.

Raspberry tiene una distribución de Linux propia para esta tarjeta basada en Debían, y combinada con el nombre propio de la tarjeta da lugar a Raspbian, misma distribución que incorpora varios lenguajes de programación para poder establecer una interacción con el mundo exterior siendo así una gran herramienta y plataforma de desarrollo de prototipos (Richardson & Wallace, 2012).

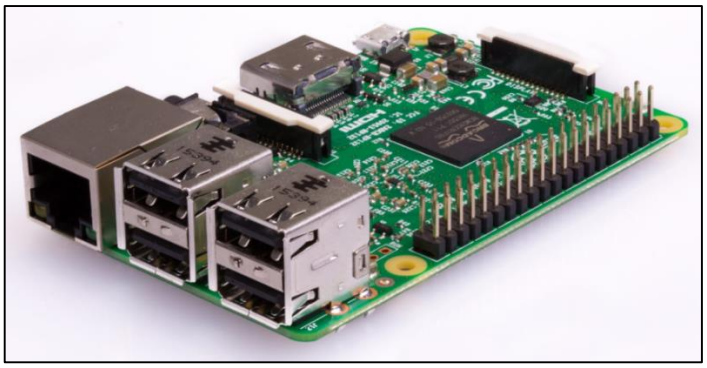


Figura 7-2: NI myRIO
Fuente: https://www.raspberrypi.org/app/uploads/2017/05/Raspberry-Pi-3-Ports-1-1833x1080.jpg, 2019

Las ventajas de usar esta tarjeta giran en torno a sus características las mismas que se muestran a continuación.

- El procesador de la Raspberry Pi 3 está a la par de los procesadores utilizados para dispositivos móviles como tabletas o celulares, el mismo que es ARM Cortex-A53 64bit Quad Core, además posee memoria de 1Gb SDRAM.
- Un slot o ranura para tarjeta micro (SD), puesto que no cuenta con un disco dura para su almacenamiento, en la tarjeta micro SD se almacena el Sistema operativo que puede ser el mismo Debian o Cualquier distribución de Linux así como otros sistemas operativos desarrollados por Microsoft entre otros.
- Este nuevo modelo de la familia de Raspberry trae cuatro puertos USB 2.0 para la conexión de periféricos.
- Puerto Ethernet. En este modelo es un puerto RJ45 Ethernet estándar, puede conectarse a una red cableada mediante un adaptador Ethernet.
- Integra la conectividad Bluetooth Low Energy (BLE) yWi-Fi, ya no necesita dispositivos adicionales.
- Conector HDMI. El puerto HDMI proporciona salida de vídeo y audio digital, a un monitor externo o a un televisor.
- Entrada de alimentación. Una de las primeras cosas que te darás cuenta es que no hay interruptor de encendido en el Pi. Este conector micro USB se utiliza para suministrar energía soportando un máximo de 2.5 amperios y 5 voltios.
- El conector de la interfaz serie de la cámara (CSI). Este puerto permite una cámara módulo a conectar directamente a la placa.
- Leds indicadores del estado de la tarjeta, así como del estado de la conexión en el puerto Ethernet.

Todas las características presentadas, hacen que esta tarjeta sea popularizada y muy empleada en el desarrollo de prototipos y ha sido seleccionada para el desarrollo de esta investigación.

2.5 Robot Planar

Un cuerpo rígido en el espacio puede moverse de varias maneras, en movimiento de traslación o rotación, los movimientos capaces de realizar se denominan grados de libertad, el alcance del actuador final se denomina posición y se describe por coordenadas generalizadas; y la orientación está definida por ángulos; De modo que cuando se logra controlar los grados de libertad del actuador final por medio de un sistema mecánico, electrónico, neumático, etc., el sistema puede ser llamado robot.

Los robots planares se encuentra dentro de una clasificación de robots paralelos, también conocidas como maquinas cinemáticas paralelas, los mismos que definidos por Merlet, (2005) son mecanismos de bucle cerrado que presentan muy buenos resultados en términos de precisión, rigidez y capacidad para manipular grandes cargas, donde los elementos mecánicos que facilitan el movimiento de un cuerpo rígido respecto a una base denomina actuador final.

Los Robots paralelos son mecanismos con las siguientes características:

- Tienen por las menos dos cadenas que sostienen al actuador final, cada una de esas cadenas contiene al menos un actuador simple.
- Tiene un sensor apropiado para medir el valor de las variables asociadas con la la actuación ya sea en rotación o movimientos lineales
- El número de actuadores define el número de grados de libertad del actuador final
- La movilidad del manipulador es cero cuando los actuadores están bloqueados

Un robot planar completamente paralelo tiene tres grados de libertad, sin embargo, existen robots planares con menos grados de libertad como es el caso dentro de esta investigación, dos cadenas soportan al actuador final, las cadenas están unidas al actuador final en un solo punto (Merlet, 2006) .

CAPÍTULO III

3 DESARROLLO DE LA INVESTIGACIÓN

3.1 Metodología.

En este capítulo se describe cada uno de los pasos a seguir para la realización de la investigación, en el cual se desarrolla el modelo matemático de la cinemática directa, así como la inversa del brazo robótico planar, el diseño de los agentes inteligentes para controlar las trayectorias del brazo robótico, con la finalidad de cumplir los objetivos planteados y demostrar la hipótesis planteada.

3.1.1 Cinemática

El estudio de la cinemática, hace referencia al seguimiento de trayectorias del robot, sin importar el tipo de actuadores para efectuar dicho movimiento, para el estudio de este caso se utilizó el método geométrico, realizando el análisis de dos brazos, de modo que al final se pueda obtener el conjunto de ecuaciones que gobiernan el movimiento.

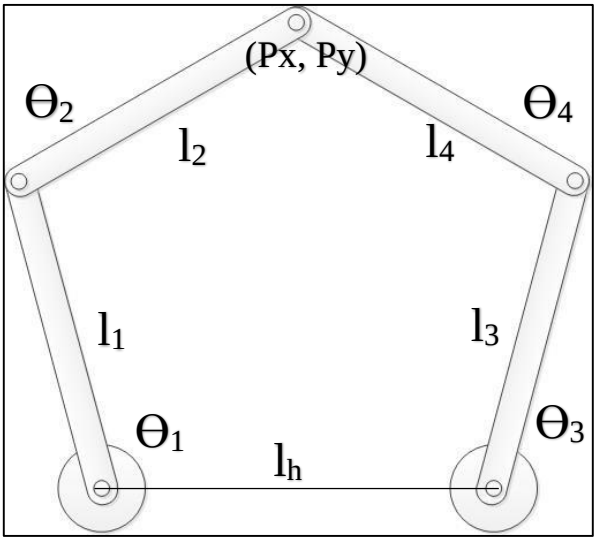


Figura 8-3: Robot planar 2DOF
Realizado por: Alex Mantilla, 2019

- Cinemática Directa Brazo Izquierdo

  Para el estudio de la cinemática directa del brazo izquierdo se tomó como punto de referencia el actuador izquierdo siendo este el origen de coordenadas, de modo que para hallar el punto (Px,Py) se obtienen las ecuaciones 3.1 y 3.2

$$Px = l_1 \cos\theta_1 + l_2 \cos(\theta_1 + \theta_2) \tag{3.1}$$

$$Py = l_1 \sin\theta_1 + l_2 \sin(\theta_1 + \theta_2) \tag{3.2}$$

- Cinemática Inversa del Brazo Izquierdo

  La cinemática inversa del brazo parte de elevar al cuadrado las ecuaciones 3.1 y 3.2, obteniendo así las ecuaciones 3.3. y 3.4, para luego sumarlas y así obtener el coseno del ángulo Θ_2

$$Px^2 = l_1^2 \cos^2\theta_1 + 2l_1 l_2 \cos\theta_1 \cos(\theta_1 + \theta_2) + l_2^2 \cos^2(\theta_1 + \theta_2) \tag{3.3}$$

$$Py^2 = l_1^2 \sin^2\theta_1 + 2l_1 l_2 \sin\theta_1 \sin(\theta_1 + \theta_2) + l_2^2 \sin^2(\theta_1 + \theta_2) \tag{3.4}$$

$$\begin{aligned} Px^2 + Py^2 = {} & l_1^2 \cos^2\theta_1 + 2l_1 l_2 \cos\theta_1 \cos(\theta_1 + \theta_2) + l_2^2 \cos^2(\theta_1 + \theta_2) + l_1^2 \sin^2\theta_1 \\ & + 2l_1 l_2 \sin\theta_1 \sin(\theta_1 + \theta_2) + l_2^2 \sin^2(\theta_1 + \theta_2) \end{aligned}$$

$$\begin{aligned} Px^2 + Py^2 = {} & l_1^2 \cos^2\theta_1 + 2l_1 l_2 \cos\theta_1 \cos(\theta_1 + \theta_2) + l_2^2 \cos^2(\theta_1 + \theta_2) + l_1^2 \sin^2\theta_1 \\ & + 2l_1 l_2 \sin\theta_1 \sin(\theta_1 + \theta_2) + l_2^2 \sin^2(\theta_1 + \theta_2) \end{aligned}$$

$$\begin{aligned} Px^2 + Py^2 = {} & l_1^2 \cos^2\theta_1 + 2l_1 l_2 \cos\theta_1 (\cos\theta_1 \cos\theta_2 - \sin\theta_1 \sin\theta_2) + l_2^2 \cos^2(\theta_1 + \theta_2) \\ & + l_1^2 \sin^2\theta_1 + 2l_1 l_2 \sin\theta_1 (\sin\theta_1 \cos\theta_2 + \sin\theta_2 \cos\theta_1) + l_2^2 \sin^2(\theta_1 + \theta_2) \end{aligned}$$

$$\begin{aligned} Px^2 + Py^2 = {} & l_1^2 \cos^2\theta_1 + l_1^2 \sin^2\theta_1 + 2l_1 l_2 \cos\theta_1 (\cos\theta_1 \cos\theta_2 - \sin\theta_1 \sin\theta_2) \\ & + 2l_1 l_2 \sin\theta_1 (\sin\theta_1 \cos\theta_2 + \sin\theta_2 \cos\theta_1) + l_2^2 \cos^2(\theta_1 + \theta_2) \\ & + l_2^2 \sin^2(\theta_1 + \theta_2) \end{aligned}$$

$$Px^2 + Py^2 = l_1^2\left(\cos^2\theta_1 + \sin^2\theta_1\right) + 2l_1l_2\left[\cos^2\theta_1\cos\theta_2 - \cos\theta_1\sin\theta_1\sin\theta_2 + \sin^2\theta_1\cos\theta_2 + \sin\theta_1\sin\theta_2\cos\theta_1\right] + l_2^2\left[\cos^2(\theta_1+\theta_2) + \sin^2(\theta_1+\theta_2)\right]$$

$$Px^2 + Py^2 = l_1^2 + 2l_1l_2\left[\cos\theta_2\left(\cos^2\theta_1 + \sin^2\theta_1\right)\right] + l_2^2$$

$$Px^2 + Py^2 = l_1^2 + 2l_1l_2\cos\theta_2 + l_2^2$$

$$\cos\theta_2 = \frac{Px^2 + Py^2 - l_1^2 - l_2^2}{2l_1l_2} \tag{3.5}$$

La ecuación 3.5 permite establecer el alcance del brazo izquierdo, de donde se puede destacar los siguientes casos.

$$\begin{cases} |\cos\theta_2| > 1 & El\ punto\ es\ inaccesible \\ |\cos\theta_2| = 1 & El\ brazo\ está\ extendido \\ |\cos\theta_2| < 1 & Tiene\ dos\ soluciones\ para\ llegar\ al\ punto\ \theta_2\ y\ -\theta_2 \end{cases}$$

Desarrollando las ecuaciones 3.1 y 3.2 tenemos las siguientes de ecuaciones:

$$Px = l_1\cos\theta_1 + l_2\left(\cos\theta_1\cos\theta_2 - \sin\theta_1\sin\theta_2\right) \tag{3.6}$$

$$Py = l_1\sin\theta_1 + l_2\left(\sin\theta_1\cos\theta_2 + \sin\theta_2\cos\theta_1\right) \tag{3.7}$$

Despejando sin Θ_1 de la ecuación 3.6

$$\sin\theta_1 = \frac{-Px + l_1\cos\theta_1 + l_2\cos\theta_1\cos\theta_2}{l_2\sin\theta_2}$$

Remplazando en la ecuación 3.7 hallamos el cos Θ_1

$$Py = \left[\frac{-Px + l_1\cos\theta_1 + l_2\cos\theta_1\cos\theta_2}{l_2\sin\theta_2}\right](l_1 + l_2\cos\theta_2) + l_2\sin\theta_2\cos\theta_1$$

$$Py = \frac{-Px(l_1 + l_2\cos\theta_2) + l_1^2\cos\theta_1 + l_1 l_2\cos\theta_1\cos\theta_2 + l_1 l_2\cos\theta_1\cos\theta_2}{l_2\sin\theta_2}$$
$$\frac{+l_2^2\cos\theta_1\cos^2\theta_2 + l_2^2\sin^2\theta_2\cos\theta_1}{l_2\sin\theta_2}$$

$$Py = \frac{-Px(l_1 + l_2\cos\theta_2) + \cos\theta_1\left(l_1^2 + 2l_1 l_2\cos\theta_2 + l_2^2\right)}{l_2\sin\theta_2}$$

$$\cos\theta_1 = \frac{Py l_2\sin\theta_2 - Px(l_1 + l_2\cos\theta_2)}{\left(l_1^2 + 2l_1 l_2\cos\theta_2 + l_2^2\right)} \tag{3.8}$$

Despejando cos Θ_1 de la ecuación 3.7

$$\cos\theta_1 = \frac{Py - l_1\sin\theta_1 - l_2\sin\theta_1\cos\theta_2}{l_2\sin\theta_2}$$

Remplazando en la ecuación 3.6 hallamos el sin Θ_1

$$Px = \left[\frac{Py - l_1\sin\theta_1 - l_2\sin\theta_1\cos\theta_2}{l_2\sin\theta_2}\right](l_1 + l_2\cos\theta_2) - l_2\sin\theta_1\sin\theta_2$$

$$Px = \frac{Py(l_1 + l_2\cos\theta_2) - l_1^2\sin\theta_1 - l_1 l_2\sin\theta_1\cos\theta_2 - l_1 l_2\sin\theta_1\cos\theta_2}{l_2\sin\theta_2}$$
$$\frac{-l_2^2\sin\theta_1\cos^2\theta_2 - l_2^2\sin\theta_1\sin^2\theta_2}{l_2\sin\theta_2}$$

$$Px = \frac{Py(l_1 + l_2\cos\theta_2) - \sin\theta_1\left(l_1^2 + 2l_1 l_2\cos\theta_2 + l_2^2\right)}{l_2\sin\theta_2}$$

$$\sin\theta_1 = \frac{Py(l_1 + l_2\cos\theta_2) - Px l_2\sin\theta_2}{\left(l_1^2 + 2l_1 l_2\cos\theta_2 + l_2^2\right)} \tag{3.9}$$

- Cinemática Directa Brazo Derecho

  Para el análisis de la cinemática directa del brazo derecho se hace referencia al mismo actuador izquierdo únicamente considerando el desplazamiento horizontal lh, de modo que para alcanzar el punto (Px, Py), se utilizan las ecuaciones 3.10 y 3.11.

$$Px = lh + l_3\cos\theta_3 + l_4\cos(\theta_3 + \theta_4)$$

$$Py = l_3\sin\theta_3 + l_4\sin(\theta_3 + \theta_4)$$

$$Dx = Px - lh$$
$$Dy = dy$$

$$Dx = l_3\cos\theta_3 + l_4\cos(\theta_3 + \theta_4) \tag{3.10}$$

$$Dy = l_3\sin\theta_3 + l_4\sin(\theta_3 + \theta_4) \tag{3.11}$$

- Cinemática Inversa Brazo Derecho

  En el estudio de la cinemática inversa del brazo derecho se procede de la misma forma, que, con el brazo izquierdo, obteniendo así el coseno de Θ4.

$$Dx^2 = l_3^2\cos^2\theta_3 + 2l_3l_4\cos\theta_3\cos(\theta_3 + \theta_4) + l_4^2\cos^2(\theta_3 + \theta_4) \tag{3.12}$$

$$Dy^2 = l_3^2\sin^2\theta_3 + 2l_3l_4\sin\theta_3\sin(\theta_3 + \theta_4) + l_4^2\sin^2(\theta_3 + \theta_4) \tag{3.13}$$

$$Dx^2 + Dy^2 = l_3^2\cos^2\theta_3 + 2l_3l_4\cos\theta_3\cos(\theta_3 + \theta_4) + l_4^2\cos^2(\theta_3 + \theta_4) + l_3^2\sin^2\theta_3 + 2l_3l_4\sin\theta_3\sin(\theta_3 + \theta_4) + l_4^2\sin^2(\theta_3 + \theta_4)$$

$$Dx^2 + Dy^2 = l_3^2\cos^2\theta_3 + 2l_3l_4\cos\theta_3\cos(\theta_3 + \theta_4) + l_4^2\cos^2(\theta_3 + \theta_4) + l_3^2\sin^2\theta_3 + 2l_3l_4\sin\theta_3\sin(\theta_3 + \theta_4) + l_4^2\sin^2(\theta_3 + \theta_4)$$

$$Dx^2+Dy^2 = l_3^2\cos^2\theta_3+2l_3l_4\cos\theta_3(\cos\theta_3\cos\theta_4-\sin\theta_3\sin\theta_4)+l_4^2\cos^2(\theta_3+\theta_4)$$
$$+l_3^2\sin^2\theta_3+2l_3l_4\sin\theta_3(\sin\theta_3\cos\theta_4+\sin\theta_4\cos\theta_3)+l_4^2\sin^2(\theta_3+\theta_4)$$

$$Dx+Dy^2 = l_3^2\cos^2\theta_3+l_3^2\sin^2\theta_3+2l_3l_4\cos\theta_3(\cos\theta_3\cos\theta_4-\sin\theta_3\sin\theta_4)$$
$$+2l_3l_4\sin\theta_3(\sin\theta_3\cos\theta_4+\sin\theta_4\cos\theta_3)+l_4^2\cos^2(\theta_3+\theta_4)$$
$$+l_4^2\sin^2(\theta_3+\theta_4)$$

$$Dx^2+Dy^2 = l_3^2(\cos^2\theta_3+\sin^2\theta_3)+2l_3l_4[\cos^2\theta_3\cos\theta_4-\cos\theta_3\sin\theta_3\sin\theta_4+\sin^2\theta_3\cos\theta_4$$
$$+\sin\theta_3\sin\theta_4\cos\theta_3]+l_4^2[\cos^2(\theta_3+\theta_4)+\sin^2(\theta_3+\theta_4)]$$

$$Dx^2+Dy^2 = l_3^2+2l_3l_4[\cos\theta_4(\cos^2\theta_3+\sin^2\theta_3)]+l_4^2$$

$$Dx^2+Dy^2 = l_3^2+2l_3l_4\cos\theta_4+l_4^2$$

$$\cos\theta_4 = \frac{Dx^2+Dy^2-l_3^2-l_4^2}{2l_3l_4}$$

Regresando a las variables originales tenemos:

$$\cos\theta_4 = \frac{(Px-lh)^2+Py^2-l_3^2-l_4^2}{2l_3l_4} \qquad (3.14)$$

La ecuación 3.14 permite establecer el alcance del brazo izquierdo, de donde se puede destacar los siguientes casos.

$$\begin{cases} |\cos\theta_4|>1 & El\ punto\ es\ inaccesible \\ |\cos\theta_4|=1 & El\ brazo\ está\ extendido \\ |\cos\theta_4|<1 & Tiene\ dos\ soluciones\ para\ llegar\ al\ punto\ \theta_2\ y\ -\theta_2 \end{cases}$$

Desarrollando las ecuaciones 3.10 y 3.11 tenemos las siguientes de ecuaciones:

$$Dx = l_3\cos\theta_3 + l_4(\cos\theta_3\cos\theta_4 - \sin\theta_1\sin\theta_4) \tag{3.15}$$

$$Dy = l_3\sin\theta_3 + l_4(\sin\theta_3\cos\theta_4 + \sin\theta_4\cos\theta_3) \tag{3.16}$$

Despejando sinq1 de la ecuación 3.15

$$\sin\theta_3 = \frac{-Dx + l_3\cos\theta_3 + l_4\cos\theta_3\cos\theta_4}{l_4\sin\theta_4}$$

Remplazando en la ecuación 3.16 hallamos el cos Θ_3

$$Dy = \left[\frac{-Dx + l_3\cos\theta_3 + l_4\cos\theta_3\cos\theta_4}{l_4\sin\theta_4}\right](l_3 + l_4\cos\theta_4) + l_4\sin\theta_4\cos\theta_3$$

$$Dy = \frac{-Dx(l_3 + l_4\cos\theta_4) + l_3^2\cos\theta_3 + l_3l_4\cos\theta_3\cos\theta_4 + l_3l_4\cos\theta_3\cos\theta_4}{l_4\sin\theta_4}$$
$$\frac{+ l_4^2\cos\theta_3\cos^2\theta_4 + l_4^2\sin^2\theta_4\cos\theta_3}{l_4\sin\theta_4}$$

$$Dy = \frac{-Dx(l_3 + l_4\cos\theta_4) + \cos\theta_3(l_3^2 + 2l_3l_4\cos\theta_4 + l_4^2)}{l_4\sin\theta_4}$$

$$\cos\theta_3 = \frac{Dy4\sin\theta_4 - Dx(l_3 + l_4\cos\theta_4)}{(l_3^2 + 2l_3l_4\cos\theta_4 + l_4^2)} \tag{3.17}$$

Despejando cos Θ_3 de la ecuación 3.16 y remplazando Dy

$$\cos\theta_3 = \frac{Py - l_3\sin\theta_3 - l_4\sin\theta_3\cos\theta_4}{l_4\sin\theta_4}$$

Remplazando en la ecuación 3.15 hallamos el sin Θ_3

$$Dx = \left[\frac{Dy - l_3 \sin\theta_3 - l_4 \sin\theta_3 \cos\theta_4}{l_4 \sin\theta_4}\right](l_3 + l_4 \cos\theta_4) - l_4 \sin\theta_3 \sin\theta_4$$

$$Dx = \frac{Dy(l_3 + l_4 \cos\theta_4) - l_3^2 \sin\theta_3 - l_3 l_4 \sin\theta_3 \cos\theta_4 - l_3 l_4 \sin\theta_3 \cos\theta_4}{l_4 \sin\theta_4}$$
$$\frac{- l_4^2 \sin\theta_3 \cos^2\theta_4 - l_4^2 \sin\theta_3 \sin^2\theta_4}{l_4 \sin\theta_4}$$

$$Dx = \frac{Dy(l_3 + l_4 \cos\theta_4) - \sin\theta_3 \left(l_3^2 + 2 l_3 l_4 \cos\theta_4 + l_4^2\right)}{l_4 \sin\theta_4}$$

$$\sin\theta_3 = \frac{Dy(l_3 + l_4 \cos\theta_4) - Dx l_4 \sin\theta_4}{\left(l_3^2 + 2 l_3 l_4 \cos\theta_4 + l_4^2\right)}$$

Remplazando Dy y Dx obtenemos

$$\sin\theta_3 = \frac{Py(l_3 + l_4 \cos\theta_4) - (Px - lh) l_4 \sin\theta_4}{\left(l_3^2 + 2 l_3 l_4 \cos\theta_4 + l_4^2\right)}$$

3.1.2 Fuerzas en el Brazo

Para el cálculo de las fuerzas que generan el movimiento, de modo que se pueda dimensionar los actuadores capaces de ejecutar las trayectorias calculadas a través de la cinemática inversa del brazo es necesario hacer un análisis dinámico del brazo robótico.

El análisis de la dinámica del brazo se puede realizar utilizando el método del Lagrange, el mismo que consiste en analizar la relación energía cinética y potencial, sin embargo, no se ha desarrollado la matemática que implica el análisis dinámico ya que no es el caso de estudio de esta investigación.

El procedimiento del cálculo de los actuadores se lo realizo de manera más sencilla realizando el análisis del torque necesario para mantener el equilibrio del brazo sometido a efectos de la gravedad, de este modo se selecciona un actuador con mayor torque.

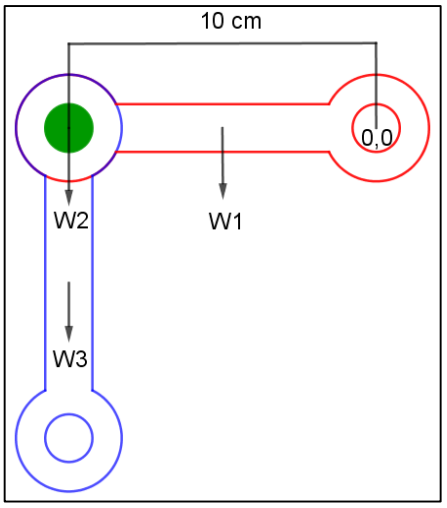


Figura 9-3: Robot planar 2DOF
Realizado por: Alex Mantilla, 2019

Desarrollando las ecuaciones para el cálculo de torque.

$$\sum \tau_{(0,0)} = 0$$

$$\sum \tau_{(0,0)} = r_1 W_1 + r_2 W_2 + r_3 W_3$$

$$\sum \tau_{(0,0)} = r_1 m_1 g + r_2 m_2 g + r_3 m_3 g$$

$$\sum \tau_{(0,0)} = 0,05m \times 0,05kg \times 9,8 m/s^2 + 0,1m \times 0,1kg \times 9,8 m/s^2 + 0,1m \times 0,05kg \times 9,8 m/s^2$$

$$\sum \tau_{(0,0)} = 0,0245\,[N]\,m + 0,098\,[N]\,m + 0,049\,[N]\,m$$

$$\sum \tau_{(0,0)} = 0,392\,[N]\,m$$

3.1.3 Simulación del brazo en Matlab

Previo a la implementación física del brazo se realizó la simulación con el uso de herramientas CAD, en este caso el software Matlab, en el mismo que se puede simular el comportamiento de cada uno de los elementos de los que está conformado el brazo robótico, todos los elementos para la simulación se encuentran en la librería de simmechanics de simulink, es así que en el bloque denominado body, se detallan todas las dimensiones de los eslabones, las articulaciones son simuladas por el bloque denominado revoluta y a su vez permite simular los actuadores, los actuadores van conectados a otro bloque el mismo en el que se establece el punto de referencia

para el eje coordenado, todos esto facilita evidenciar el movimiento que tendrá el brazo y todo lo mencionado se muestra en la siguiente figura.

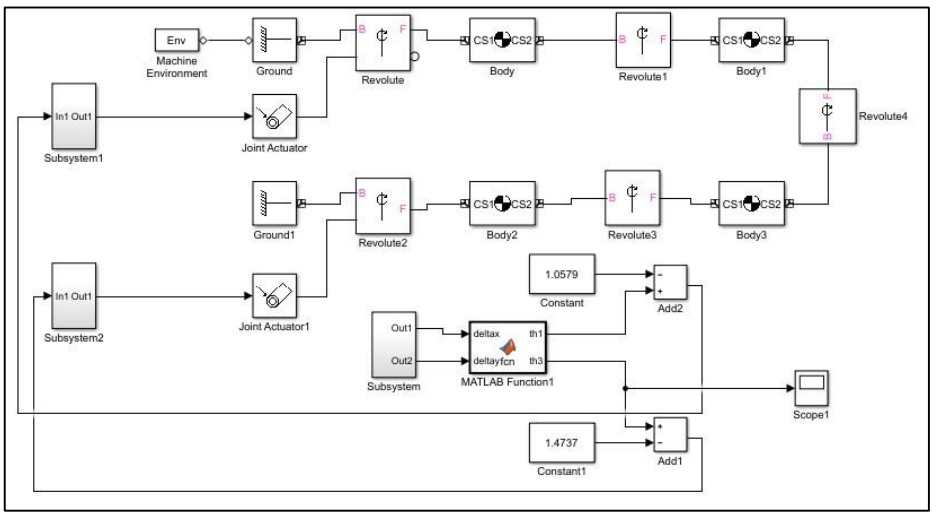


Figura 10-3: Diagrama de bloques del brazo
Realizado por: Alex Mantilla, 2019

Las trayectorias fueron comprobadas gracias a que permite ejecutar las ecuaciones de la cinemática inversa gracias al bloque de funciones.

Al ejecutar el programa desarrollado para la simulación nos presenta el modelo del brazo y a su vez se mueve en tiempo real ejecutando la trayectoria establecida.

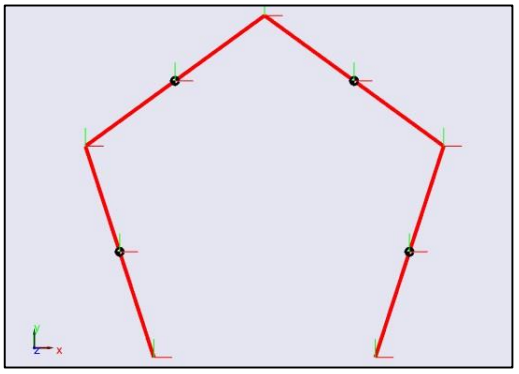


Figura 11-3: Diagrama de bloques del brazo
Realizado por: Alex Mantilla, 2019

3.2 Construcción del Brazo Robótico

El brazo Robótico es uno de tipo planar de dos grados de libertad, de accionamiento directo debido a que el eje de los actuadores en este caso los servo motores van conectados directo a dos eslabones, y los otros dos están unidos por articulaciones rotativas.

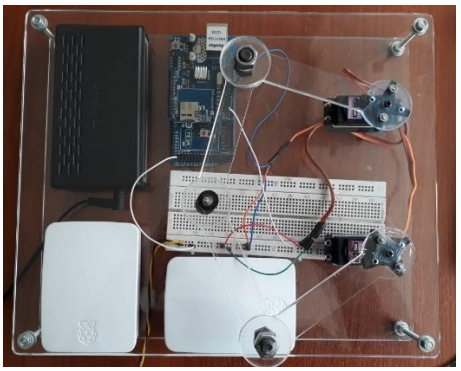


Figura 12-3: Robot Planar
Realizado por: Alex Mantilla, 2019

3.2.1 Eslabón

Es un elemento mecánico rígido que se desplaza en el área de trabajo del robot, los mismos que pueden estar unidos por articulaciones de diferentes tipos para lograr cubrir una mayor área.

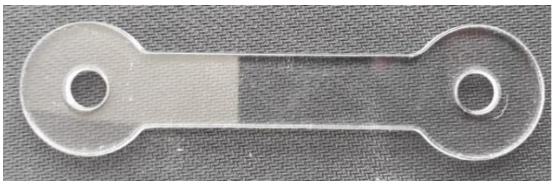


Figura 13-3: Eslabón
Realizado por: Alex Mantilla, 2019

3.2.2 Actuador

Es el mecanismo proporciona la fuerza motriz para el movimiento de los eslabones, y de este modo alcanzar el punto deseado, en este caso el actuador es eléctrico, el mismo que trabaja en un rango de 4.8 voltios a 7.2 voltios y consume una corriente que va desde los 500mA a 900mA, generando así un torque de 9.4Kgf.cm a 11Kgf.cm y su velocidad de 0˚ a 180˚ en 0,5 segundos; aunque pueden existir actuadores de otro tipo.

Figura 14-3: Servo Motor Mg996R
Fuente: https://www.nextiafenix.com/producto/servo-motor-mg995/, 2019

3.2.3 Articulación

Es el elemento que permite la unión de varios eslabones, estas articulaciones pueden ser de tipo prismática, cilíndrica planar, esférica y rotacional la misma que es utilizada en este brazo Robótico.

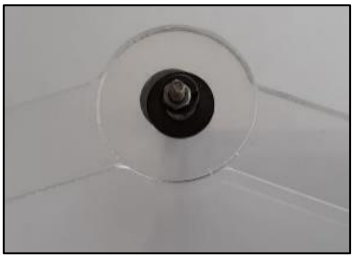


Figura 15-3: Articulación
Realizado por: Alex Mantilla, 2019

3.3 Red de Comunicación

La comunicación es fundamental en la implementación de los sistemas distribuidos de modo que la elección de la tecnología influye directamente en los costos del proyecto, es así que no existe una tecnología única para todos los escenarios, ya que se puede implementar la solución de diferentes maneras.

La red de comunicación para esta investigación está basada en ethernet, y el protocolo elegido es el protocolo de datos no confiables UDP por sus siglas en inglés, Cuando hablamos de los protocolos de Internet industrial, a menudo se sugiere UDP como el protocolo de bajo costo y eficiencia para IIoT, debido a su baja relación de carga útil a encabezado (Gilchrist, 2016).

Únicamente es aceptable para escenarios en los maneja un flujo de datos no críticos como mientras que en transferencia de datos críticos se puede utilizar TCP/IP sin embargo por el direccionamiento en las cabeceras hacen que el tiempo de comunicación sea mayor.

3.4 Tarjeta de desarrollo

Para el desarrollo de esta investigación se eligió la Raspberry Pi, debió a las prestaciones que presta frente a las otras tarjetas descritas en el capítulo anterior.

La Raspberry Pi tiene una versión para trabajar en entornos industriales, pero la ventaja radica en que se puede realizar el trabajo en una versión básica y luego exportarlo, por otro lado frente a la Beagle bone black, tienen las mismas características su única ventaja radica en el precio ya que la Raspberry Pi cuesta un 20 % menos, y frente a la tarjeta My RIO, sus ventajas son mayores tanto en capacidad de procesamiento, así como en el desarrollo debió a que la tarjeta My RIO es de National Instrumens por lo que hay que adquirir licencias, razón por la cual su costo es elevado frente a Raspberry que es Open Sourse.

La arquitectura distribuida está compuesta por dos Raspberries en las que se implementa los agentes inteligentes, además tiene un Arduino Mega 2560 el mismo que sirve de interfaz entre los agentes y los actuadores para llevar a cabo ejecución de las trayectorias programadas para el robot, y toda la información es intercambiada a través de una red Ethernet, la representación de la arquitectura se muestra en la siguiente figura.

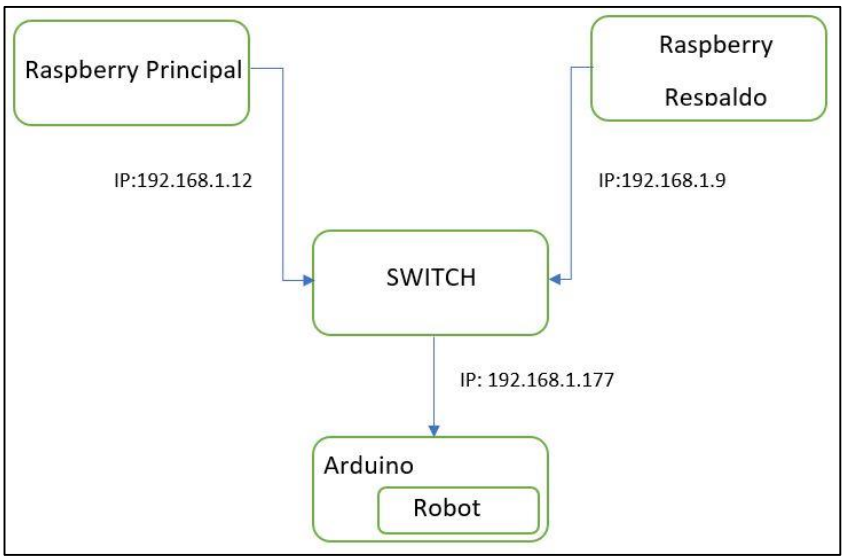


Figura 16-3: Arquitectura del sistema distribuido
Realizado por: Alex Mantilla, 2019

3.4.1 Instalación del sistema operativo

El sistema operativo a instalar en la Raspberry es Raspbian una distribución de Linux basada en Debian exclusiva para la Raspberry. este sistema operativo puede ser descargado de la página oficial de la Raspberry, y ser instalado en una tarjeta micro SD, capacidad de almacenamiento dependerá de las necesidades del usuario.

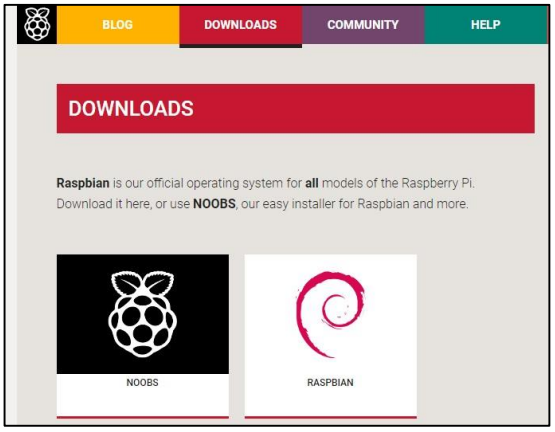


Figura 17-3: Sistema Operativo
Fuente: https://www.raspberrypi.org/downloads/, 2019

Una vez ya preparada la tarjeta micro SD con el sistema operativo conectamos se inserta la en la Raspberry, además se conecta todos los periféricos necesarios para iniciar el proceso de instalación, para lo cual hay que seguir una serie de pasos de acuerdo como se vaya completando el proceso.

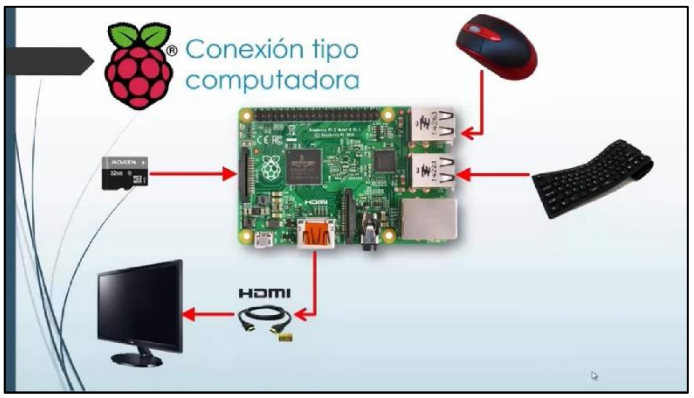


Figura 18-3: Raspberry y sus Periféricos
Realizado por: Alex Mantilla, 2019

Al finalizar la instalación del sistema operativo se muestra la pantalla del escritorio con su logo característico de la distribución de Raspbian.

Figura 19-3: Escritorio Raspbian
Realizado por: Alex Mantilla, 2019

Al iniciar la ejecución de los Raspberries hay que establecer las direcciones IP tanto para el Raspberry en el que se ejecutara el agente de respaldo, así como en el Raspberry que ejecutara el agente principal por lo que se necesita escribir las siguientes instrucciones en el terminal.

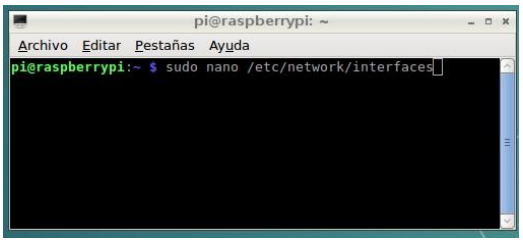
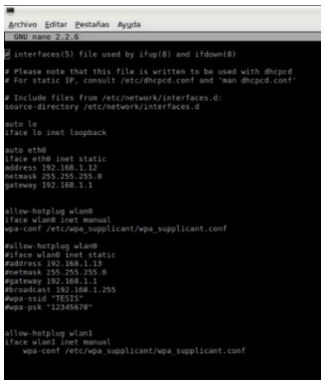
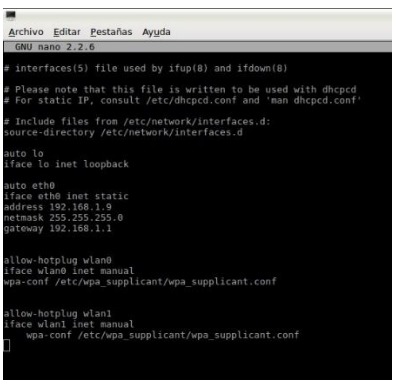


Figura 20-3: Configuración de IP
Realizado por: Alex Mantilla, 2019

Al ejecutar el comando mostrado en la figura 20-3 se abre el archivo que permite establecer las IP de manera estática, el proceso se lo realiza tanto como para el agente principal, así como para el agente de respaldo.



(a) IP Agente Principal



(b) IP Agente de Respaldo

Figura 21-3: IP Estática de los Agentes
Realizado por: Alex Mantilla, 2019

3.5 Desarrollo del código de Reconfiguración

Para el desarrollo de los agentes inteligentes, los mismos que serán encargados de llevar a cabo la reconfiguración dinámica, se establece la arquitectura de todo el sistema, y los elementos que lo componen.

Una vez determinada la arquitectura, se toma en cuenta el papel que debe desempeñar cada uno de elementos dentro del sistema, y la forma de trabajo queda establecido por el siguiente diagrama de flujo.

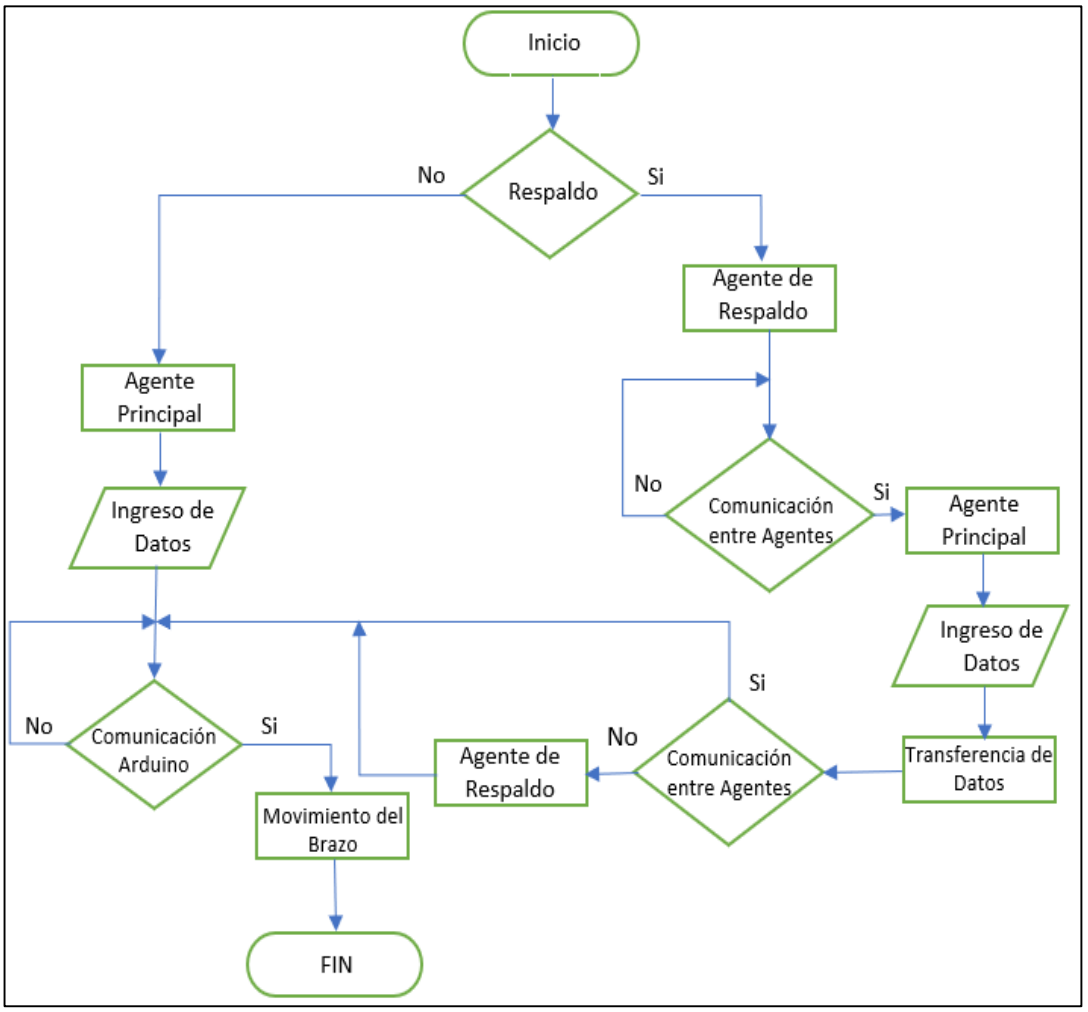


Figura 22-3: Arquitectura del sistema distribuido
Realizado por: Alex Mantilla, 2019

El diagrama de flujo facilita el desarrollo del código de reconfiguración de los agentes el mismo que será desarrollado en Python, tanto del Agente principal, así como el Agente de Respaldo, así como el código desarrollado en el Arduino.

3.5.1 Agente Principal

```
from math import cos,sin,tan,acos,asin,atan,pi
from socket import *
import time
```

En primer lugar, hay que importar las librerías para poder usar funciones específicas de Python, es así que la línea from math import, permite usar funciones matemáticas como son las trigonométricas del seno, coseno, tangente además la constante de pi (π).

Por otro lado la función principal en esta investigación es la de comunicación, la misma que utiliza el protocolo UDP por lo que la línea de from socket import *, permite hacer uso de todos los elementos de la librería, tales como la configuración de direcciones ip, creaciones de Socket y los puertos para la comunicación.

Además, hay que importar la librería para el manejo de tiempo necesaria en la toma de datos para la reconfiguración, mediante la línea time import.

Las siguientes líneas son de inicialización de variables para el cálculo de la cinemática inversa del brazo Robótico.

```
c=[] NumPuntos=0 bandera=0 i=0 InicioProceso=0
l1=10.0 l2=10.0 l3=10.0 l4=10.0 ld=10.0
X2=0.0 Y2=0.0 Num1=0.0 Den1=0.0 Num2=0.0 Den2=0.0 Num3=0.0 Den3=0.0 #Angulos
AnguloSend_1=0 AnguloSend_2=0
#Punto cero
CeroX=5.0 CeroY=13.38 AnguloInit1=113; AnguloInit2=66; NumeroProcesos=0
ConectBand=0
```

El agente principal debe establecer comunicación tanto con el agente de respaldo así como con el Arduino, por lo que es necesario crear una instancia para la dirección IP de los mismos ademas se utiliza un puerto diferente, todo esto se logra mediante:

```
# DIRECCION ARDUINO
address= ( "192.168.1.177", 5000) #define la ip del Arduino y el puerto de comunicación
client_socket = socket(AF_INET, SOCK_DGRAM) #Configura el Socket
client_socket.settimeout(2) #Tiempo de espera para responder
# DIRECCION RESPALDO
server_address = ('192.168.1.12', 5000) #Apunta al agente de Respaldo
client_socket2 = socket(AF_INET, SOCK_STREAM) #Configura el Socket
```

El sistema puede funcionar únicamente con el Agente Principal así como con el Agente de Respaldo por lo que hay que diferenciar el modo de trabajo esto se logra de la siguiente forma.

```
try:
client_socket2.connect(('192.168.1.12', 9995)) #Trata de conectar con el respaldo
ConectBand=1
print ("Inicio de trabajo Con Respaldo")
except:
#Si no hay Respaldo se ejecuta unicamente el Agente Principal
print ("Inicio de trabajo Sin Respaldo")
pass
```

Las siguientes líneas únicamente se ejecutan cuando el sistema trabaja con el Agente principal, así como con el Agente de Respaldo, en la comunicación para informar el estado del proceso.

```
try:
rec_data, addr = client_socket.recvfrom(2048) #Recibe la respuesta desde el arduino

bandera = int (rec_data)
print(bandera)
while (bandera!= 41):
print ("No llega 41")
time.sleep(1)
except:
pass
# Comunicacion con el agente de respaldo para siga esperando
if (ConectBand==1):
client_socket2.sendto( str(bandera), server_address) #Send the data request

print ("Enviando :" + str(bandera))

#Espero confirmacion de agente de respaldo seguir con el calculo del siguiente punto
rec_data2, addr2 = client_socket2.recvfrom(2048) #Lee la respuesta de agente de Respaldo
bandera2 = int (rec_data2)
print(bandera2)
while (bandera2!=51):
print ("No llega data")
time.sleep(1)
```

3.5.2 Agente de Respaldo

El agente de Respaldo, es el encargado de llevar a cabo el proceso hasta culminar siempre que hay fallado el agente principal, por esta razón su algoritmo es bastante similar al Agente principal, sin embargo, se detalla el código a continuación.

Al igual que con el agente principal se necesita importar las librerías de matemática, de comunicación, y de tiempo.

```
from math import cos,sin,tan,acos,asin,atan,pi
socket import socket
import time
```

En el agente de Respaldo debe escuchar las conexiones entrantes por lo que utiliza otros recursos de la librería Socket.

```
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM) #Configura el socket
s.bind(("", 9995)) #Con el metodo bind le indicamos que puerto debe escuchar y de que servidor esperar conexiones, la ip mejor dejarlo en blanco y hace referencia al equipo local
s.listen(1) #Aceptamos conexiones entrantes con el metodo listen, y ademas aplicamos como parametro , al recibir datos este
sc, addr = s.accept()#Instanciamos un objeto sc (socket cliente) para recibir datos

address= ( "192.168.1.177", 5000) # Define la dirección del arduino y el puerto de comunicación
client_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
```

La comunicación se da siempre y cuando el sistema tenga funcionando el agente de Respaldo, además se evalúa el tiempo que tarda en reconfigurar el sistema, tomando medidas desde que deja de responder el Agente Principal hasta que responde el Arduino luego de que el Agente de Respaldo haya Tomado las riendas del proceso.

```
try:
recibido = sc.recv(1024)
print (recibido)
if (recibido == "41"):
print ("Llego 41")
sc.send(str(51)) #Respondo al Agente Principal con bandera para ejecutar el siguiente punto
else:
BanderaRepetir=1
inicio_tiempo=time.time()
except:
pass
#Retom el proceso para evital la detención del mismo
if (BanderaRepetir==1):
client_socket.sendto( Trama, address) #Envia los datos salvados al arduino
try:
rec_data, addr = client_socket.recvfrom(2048) #Lee la respuesta desde el arduino
bandera = int (rec_data)
print("paso: " + str(index)+" tiempo de reconfiguracion "+str(time.time()-inicio_tiempo))
print(bandera)
while (bandera!= 41):
print ("No llega 41")
time.sleep(1)
```

3.5.3 Arduino

El Arduino también juega un papel importante ya que es el encargado de interpretar los datos de la cinemática inversa calculados en Python, mismos datos que se traducirán en ángulos de giro efectuados por los servomotores.

```
packetSize =Udp.parsePacket(); //Lee el tamaño de los datos
if(packetSize>0)
{
Udp.read(packetBuffer, UDP_TX_PACKET_MAX_SIZE); //Lee los datos del buffer
//Serial.println(packetBuffer);
char Rxchar1[10];
/**********************************************Numero de paso
*******************************************/
strcpy(Rxchar1, strtok(packetBuffer,","));
Serial.print ("Paso: ");
Serial.println(atoi(Rxchar1));
for (int i=0;i<10;i++){Rxchar1[i]=0;}
/**********************************************ANgulo
1*******************************************/
strcpy(Rxchar1, strtok(NULL,","));
myservo1.write(atoi(Rxchar1));
Serial.println(myservo1.read());
Serial.println(atoi(Rxchar1));
for (int i=0;i<10;i++){Rxchar1[i]=0;}
/**********************************************Angulo
2*******************************************/
strcpy(Rxchar1, strtok(NULL,","));
myservo2.write(atoi(Rxchar1));
Serial.println(myservo2.read());
Serial.println(atoi(Rxchar1));
for (int i=0;i<10;i++){Rxchar1[i]=0;}
delay(300);
Udp.beginPacket(Udp.remoteIP(), Udp.remotePort()); //Inicializa el paquete para enviar
Udp.print(41); //Envia aviso de ejecución
Udp.endPacket();
```

El código Completo de los agentes, así como del Arduino se encuentra detallado en los Anexos.

3.6 Prueba de funcionamiento de los Agentes

El potencial de la investigación radica en que el agente principal y el agente de respaldo pueden compartir el mismo hardware, de modo que en los dos Raspberries se alojan los dos agentes.

El agente principal se ejecuta en el Raspberry principal, mientras que el agente secundario se ejecuta en el Raspberry secundario, con el potencial que se puede conmutar invirtiendo los papeles en la ejecución de los agentes, siendo así que el Raspberry secundario puede ejecutar al agente principal y el Raspberry principal puede ejecutar al agente secundario.

3.6.1 Sistema sin Respaldo

Al ejecutar solo en agente principal se muestra un mensaje de aviso que el sistema no tiene respaldo, como se muestra en la figura 3.16, además se seleccionaron 4 puntos como trayectoria para el brazo Robótico los mismos que son (0,3); (3,0); (0,-3); (-3,0).

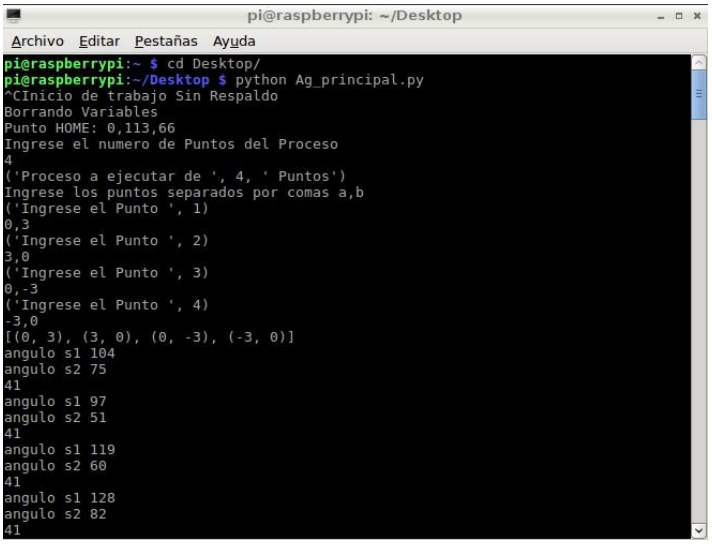


Figura 23-3: Proceso sin Respaldo
Realizado por: Alex Mantilla, 2019

3.6.2 Sistema con Respaldo

Cuando el sistema tiene el sistema de respaldo se ejecuta en primer lugar el agente de Respaldo de modo que empieza a escuchar si existe alguna petición de comunicación, y al establecer comunicación recibe los la cantidad de puntos de la trayectoria ejecutada por el brazo Robótico y cuáles serán esos puntos.

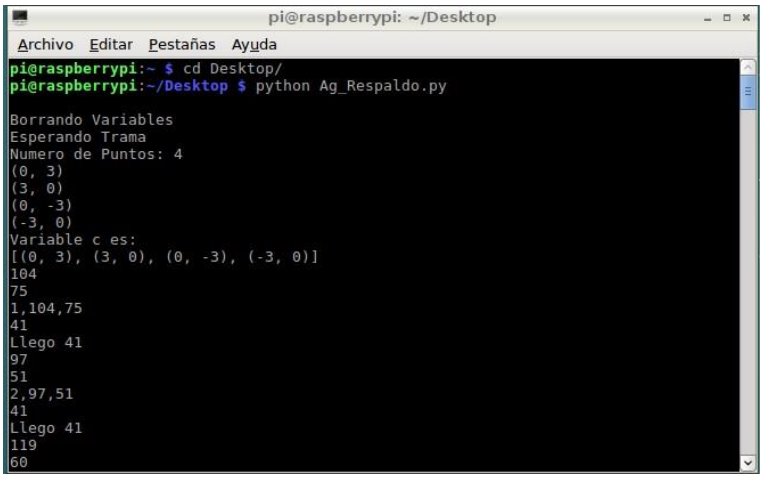


Figura 24-3: Agente de Respaldo
Realizado por: Realizado por: Alex Mantilla, 2019

Es así que al ejecutar el agente principal muestra un mensaje de aviso que el sistema tiene su respaldo respectivo, y se ingresa la información requerida.

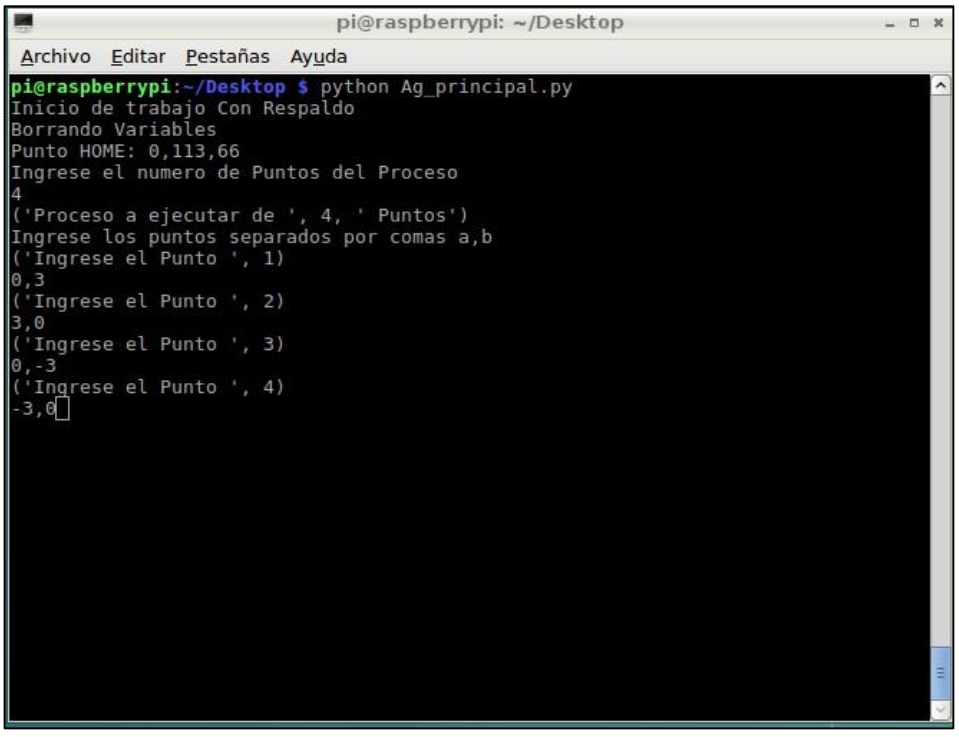


Figura 25-3: Agente Principal
Realizado por: Alex Mantilla, 2019

Una vez iniciado el proceso con su respectivo respaldo se somete a un fallo intencional al agente principal para evidenciar el comportamiento del agente del respaldo, de modo que, si se reconfigura el sistema, siendo así que el agente de respaldo toma el proceso desde el punto en el que ocurrió el fallo de agente principal, hasta culminarlo.

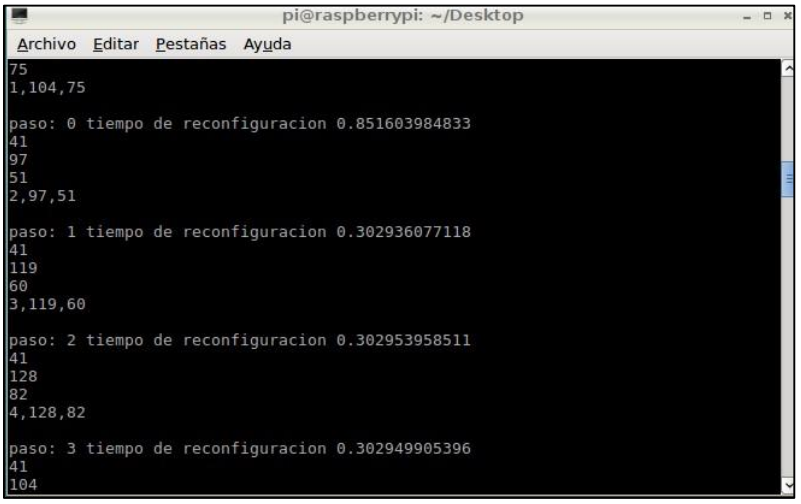


Figura 26-3: Reconfiguración
Realizado por: Alex Mantilla, 2019

En el caso de fallo el agente de respaldo informa de cual fu el punto en el que se interrumpió la comunicación, además se muestra el tiempo que tarda en reconfigurarse.

3.7 Rendimiento del Raspberry

Al ejecutar el proceso de reconfiguración del sistema, se evalúa el desempeño del procesador del raspberry, así como la carga que sufre la red, para llevar a cabo este proceso, es necesario instalar un monitor de procesos, en tal virtud se ha instalado el RPi-Monitor, esta aplicación monitorea todos los parámetros a través de la red utilizando la dirección IP asignada, la misma que es 192.168.1.12.

Para instalar el RPi-Monitor hay que ejecutar los siguientes comandos en el terminar del raspberry

- sudo wget https://github.com/XavierBerger/RPi-Monitor-deb/raw/master/packages/rpimonitor_2.7-1_all.deb
  Esta instrucción descarga el RPi-Monitor, en un archivo comprimido

- sudo dpkg -i rpimonitor_2.7-1_all.deb

Una vez descargado es necesario instalarlo por lo que se ejecuta la línea antes mencionada, una vez instalado, abrimos un navegador en cualquier equipo que se encuentre en la red y se especifica la dirección y el puerto IP 192.168.1.12:8888, mostrando los parámetros a ser monitorizados.

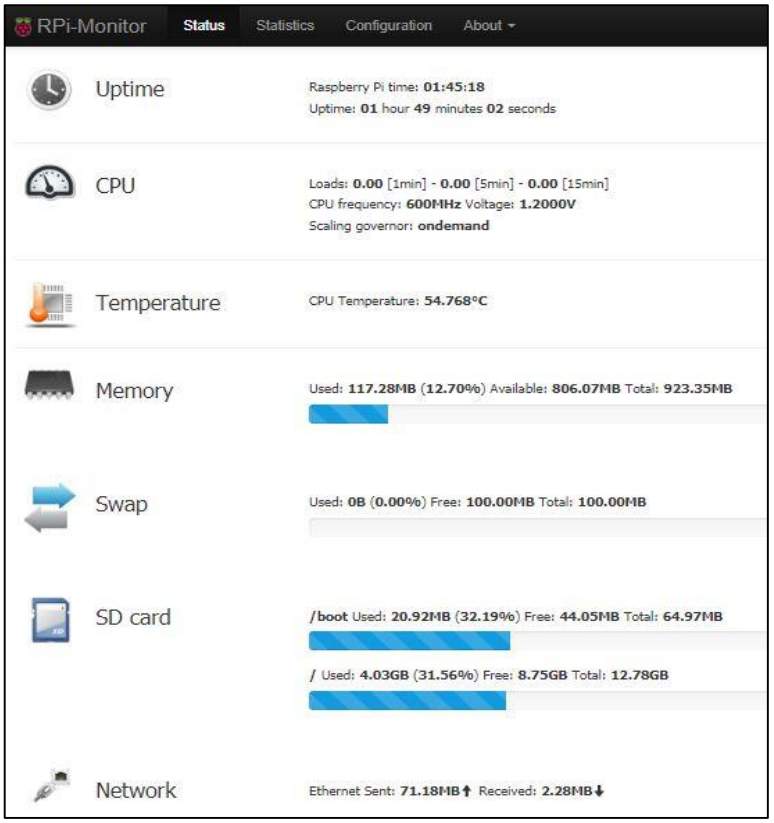


Figura 27-3: RPi-Monitor
Fuente: Rpi-MOnitor, 2019

En primer lugar, se evalúa el rendimiento del CPU con respeto al tiempo de ejecución que invierte en la ejecución del proceso de reconfiguración, como se muestra en la siguiente figura.

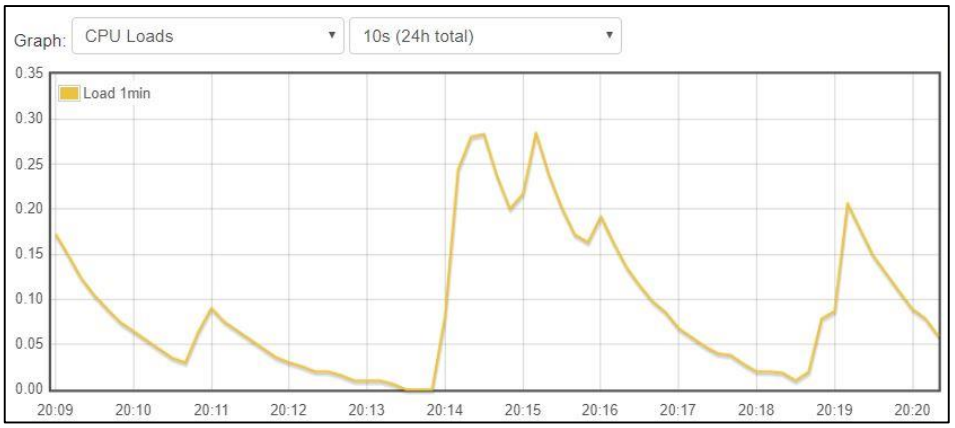


Figura 28-3: Carga del CPU
Fuente: RPi-Monitor, 2019

Por otro lado debido a la arquitectura del sistema distribuido, es necesario evaluar el desempeño de la red sometiéndola a un tráfico normal de datos, es así que mientras se ejecuta el proceso de reconfiguración se pudieron obtener los siguientes datos teniendo un pico de 146.5 KBs, dato obtenido en el instante de la transferencia de información del agente principal con el agente de respaldo.

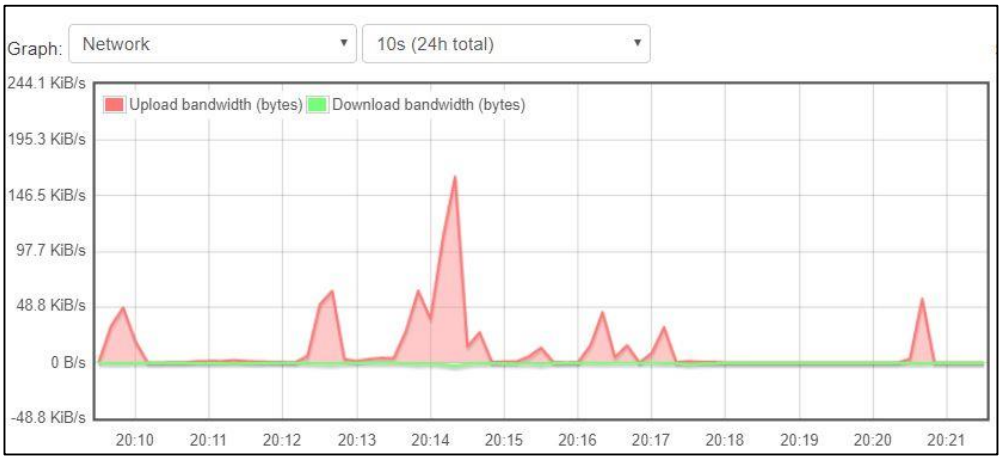


Figura 29-3: Carga del CPU
Fuente: RPi-Monitor, 2019

CAPÍTULO IV

4 ANÁLISIS Y RESULTADOS

Para llevar a cabo la reconfiguración dinámica, se realizó el desarrollo de los agentes inteligentes y la resolución de la cinemática inversa del brazo Robótico en Python, estos algoritmos se encuentran implementados en dos Raspberry Pi 3, además se utiliza un Arduino Mega para recibir datos de las Raspberries y Mover los servos motores.

Para cumplir con lo establecido en el planteamiento en la investigación se realizó varios ajustes a los algoritmos de los agentes inteligentes de modo que el proceso de comunicación sea el más idóneo para lograr la reconfiguración.

4.1 Detalles del Experimento

Para el desarrollo de la investigación se plantearon trayectorias a seguir por el brazo Robótico, las mismas que serán definidas por el usuario, las cuales serán evidenciadas a continuación:

Trayectoria 1. Esta trayectoria está compuesta por 2 puntos en que el moverá en bucle por 25 veces.

Tabla 2-4: Resultado con 2 puntos

N°- Prueba	Punto 1	Punto 2	Tiempo de Reconfiguración
1	(0, 0)	(2, 2)	0.765 segundos
2	(-2, -2)	(2, 2)	0.703 segundos
3	(0, 0)	(3, 3)	0.647 segundos
4	(0, 4)	(0, -3)	0.713 segundos
5	(-2, 2)	(2, -2)	0.700 segundos

Realizado por: Alex Mantilla, 2019

La siguiente figura evidencia el tiempo que tarda en reconfigurarse el sistema con una trayectoria formada por dos puntos, de modo que existen dos puntos de tiempo que se encuentran dispersos, sin embargo, estos puntos están a una décima de segundo de dispersión lo cual no influye significativamente en el desempeño del sistema.

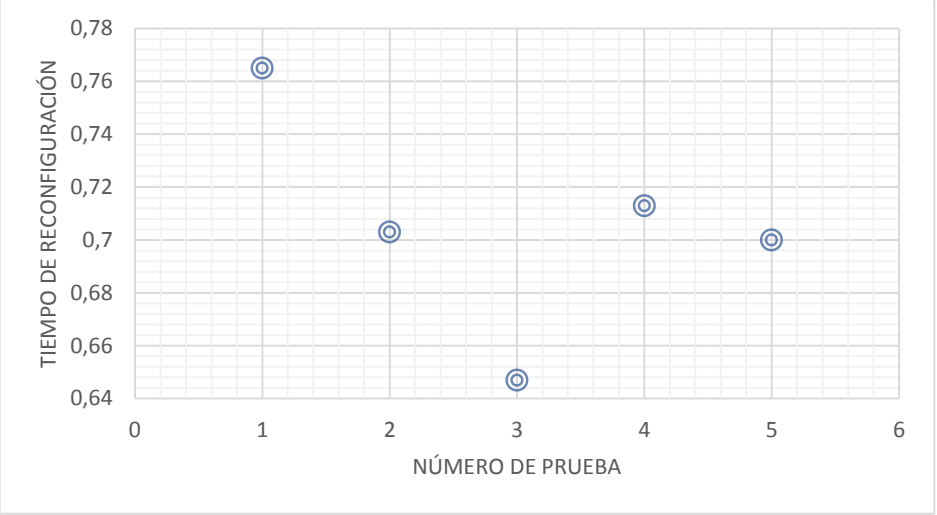


Figura 30-4: Resultado con 2 puntos
Realizado por: Alex Mantilla, 2019

Trayectoria 2. Esta trayectoria está compuesta por 3 puntos en que el brazo se moverá en bucle por 20 veces.

Tabla 3-4: Resultado con 3 puntos

N°- Prueba	Punto	Punto	Punto	Tiempo de Reconfiguración
1	(2, 2)	(0, 0)	(-2, 2)	0.609 segundos
2	(3, 3)	(0, -2)	(0, 4)	0.611 segundos
3	(2, -3)	(0, 0)	(-2, 3)	0.631 segundos
4	(0, -3)	(2, 0)	(0, 3)	0.655 segundos
5	(0, 3)	(-2, 0)	(0, -3)	0.889 segundos

Realizado por: Alex Mantilla, 2019

La siguiente figura evidencia el tiempo que tarda en reconfigurarse el sistema con una trayectoria formada por tres puntos, y solo un punto se encuentra disperso en dos décimas de segundo, el mismo que no influye significativamente en el desempeño del sistema.

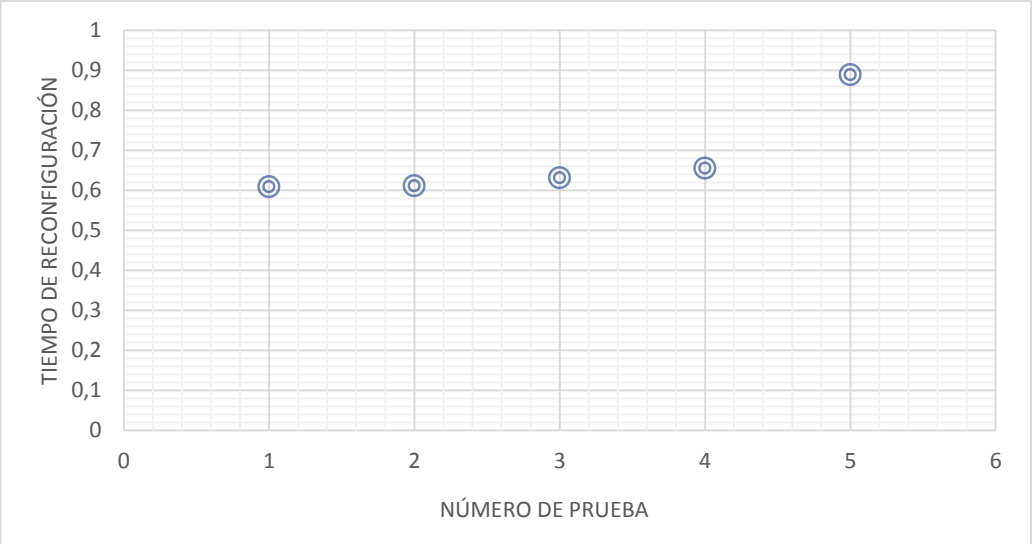


Figura 31-4: Resultado con 3 puntos
Realizado por: Alex Mantilla, 2019

Trayectoria 4. Esta trayectoria está compuesta por 5 puntos en que el brazo se moverá en bucle por 10 veces.

Tabla 4-4: Resultado con 4 puntos

N°- Prueba	Punto 1	Punto 2	Punto 3	Punto 4	Tiempo de Reconfiguración
1	(2, -2)	(-2, -2)	(-2, 2)	(2, 2)	0.680 segundos
2	(2, -2)	(0, 2)	(-2, -2)	(2, 2)	0.696 segundos
3	(3, 0)	(3, 3)	(-3, 0)	(3, -3)	0.809 segundos
4	(0, 4)	(0, -3)	(0, 4)	(-2, 0)	0,699 segundos
5	(3, 3)	(0, 0)	(0, -3)	(0, 4)	0.647 segundos

Realizado por: Alex Mantilla, 2019

La siguiente figura evidencia el tiempo que tarda en reconfigurarse el sistema con una trayectoria formada por cuatro puntos, y la distribución del tiempo es bastante uniforme a comparación de las anteriores.

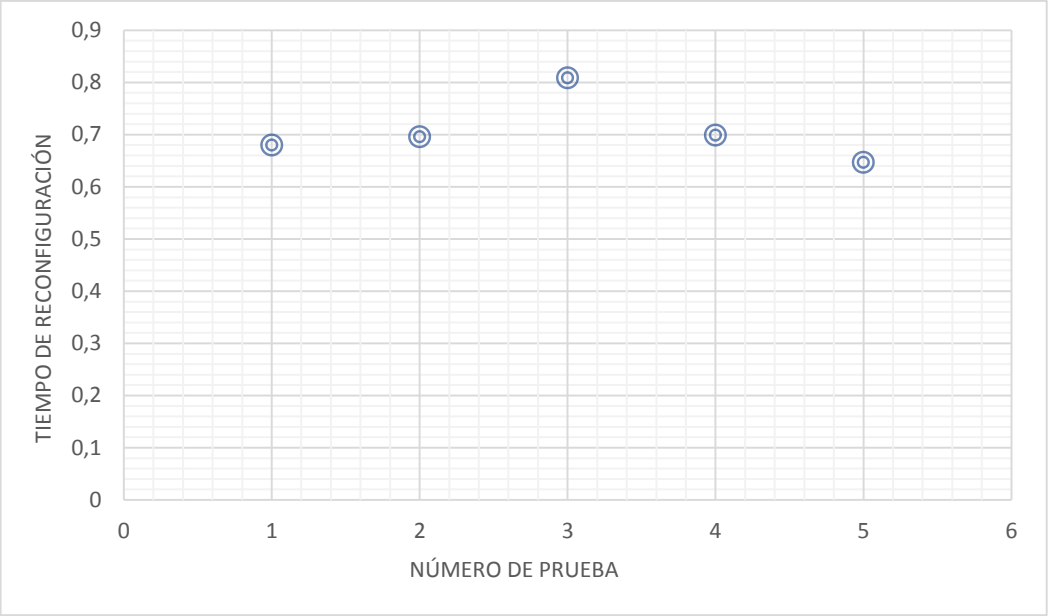


Figura 32-4: Resultado con 4 puntos
Realizado por: Alex Mantilla, 2019

Trayectoria 4. Esta trayectoria está compuesta por 5 puntos en que el brazo se moverá en bucle por 10 veces.

Tabla 5-4: Resultado con 5 puntos

N°- Prueba	Punto 1	Punto 2	Punto 3	Punto 4	Punto 5	Tiempo de Reconfiguración
1	(0, 0)	(-2, -2)	(-2, 2)	(2, 2)	(0, 4)	0.636 segundos
2	(-2, -2)	(0, 2)	(-2, -2)	(2, 2)	(-2, 0)	0.835 segundos
3	(0, 0)	(3, 3)	(-1, 0)	(3, 3)	(-2, -2)	0.684 segundos
4	(0, 4)	(3, 0)	(0, 4)	(-3, 0)	(-2, -2)	0.828 segundos
5	(-2, 2)	(0, 0)	(0, -3)	(2, -2)	(3, 3)	0.671 segundos

Realizado por: Alex Mantilla, 2019

La siguiente figura evidencia el tiempo que tarda en reconfigurarse el sistema con una trayectoria formada por cinco puntos, de modo que existen dos puntos de tiempo que se encuentran dispersos, sin embargo, estos puntos están a una décima de segundo de dispersión lo cual no influye significativamente en el desempeño del sistema.

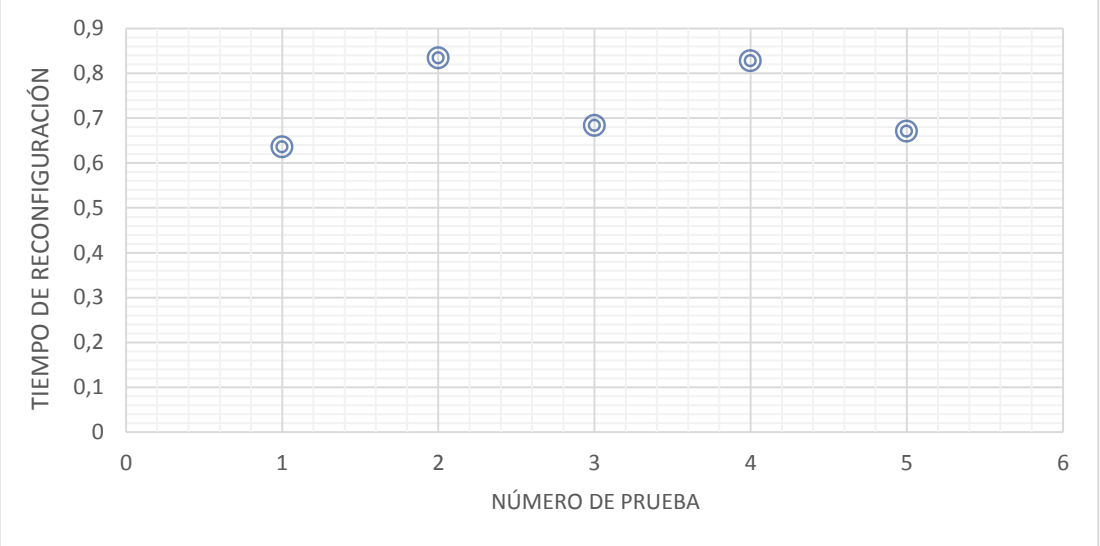


Figura 33-4: Resultado con 5 puntos
Realizado por: Alex Mantilla, 2019

4.2 Análisis de Resultados.

Tabla 6-4: Estadística Descriptiva

Estadísticos	Trayectoria 1	Trayectoria 2	Trayectoria 3	Trayectoria 4
Medi	0,7056	0,679	0,7062	0,7308
Error típico	0,018781906	0,05315261	0,027308241	0,041868126
Median	0,703	0,631	0,696	0,684
Desviación estándar	0,041997619	0,11885285	0,061063082	0,093619977
Varianza de la muestra	0,0017638	0,014126	0,0037287	0,0087647
Curtosi	1,741846626	4,502523305	3,188649227	-3,028926985
Coeficiente de asimetría	0,048186037	2,104322874	1,572853995	0,447552067
Rang	0,118	0,2	0,162	0,199

Mínim	0,647	0,609	0,647	0,636
Máxim	0,765	0,889	0,809	0,835
Sum	3,528	3,395	3,531	3,654
Cuent	5	5	5	5

Realizado por: Alex Mantilla, 2019

Trayectoria 1 En esta trayectoria hay una mayor homogeneidad de los datos puesto que su desviación estándar es la más baja esto quiere decir que los datos no se distancian mucho de la media aritmética, entonces hay un promedio de 0.04 segundos de dispersión.

Trayectoria 2 En esta trayectoria, disminuye la homogeneidad, aunque se obtiene una mejor media aritmética, pero los datos son muy dispersos evidenciándose una desviación estándar de 0.118.

Trayectoria 3 En esta trayectoria la media aritmética se asemeja a la media aritmética de la trayectoria 1, sin embargo, los datos se encuentran más dispersos puesto que la desviación estándar es de 0.061.

Trayectoria 4 En esta trayectoria la media aritmética es alta lo cual significa que es desfavorable ya que toma más tiempo en re configurarse, y presenta una alta dispersión de los datos puesto que la desviación estándar es de 0.093.

4.3 Planteamiento de la Hipótesis.

El número de puntos que conforman las trayectorias inciden significativamente en el tiempo reconfiguración dinámica basada en agentes inteligentes para sistemas distribuidos en plataformas de bajo costo.

Hipótesis Nula: $\mathbf{H_0}$ "El número de puntos que conforman las trayectorias NO inciden significativamente en el tiempo reconfiguración dinámica basada en agentes inteligentes para sistemas distribuidos en plataformas de bajo costo."

Hipótesis Alterna: $\mathbf{H_1}$ "El número de puntos que conforman las trayectorias SI inciden significativamente en el tiempo reconfiguración dinámica basada en agentes inteligentes para sistemas distribuidos en plataformas de bajo costo."

4.4 Prueba de Hipótesis

Tabla 7-4: Descriptivos

Grupos	Cuent	Sum	Promedi	Varianza
Trayectoria 1	5	3,52	0,7056	0,001763
Trayectoria 2	5	3,39	0,679	0,014126
Trayectoria 3	5	3,53	0,7062	0,003728
Trayectoria 4	5	3,65	0,7308	0,008764

Realizado por: Alex Mantilla, 2019

Tabla 8-4: Análisis de Varianza

Origen de las variaciones	*Suma de cuadrados*	*Grados de libertad*	*Promedio de los cuadrados*	*F*	*Probabilidad*	*Valor crítico para F*
Entre grupos	0,0184	3	0,0061	0,8566	0,4835	3,2389
Dentro de los grupos	0,1145	16	0,0072			
Total	0,1329	19				

Realizado por: Alex Mantilla, 2019

El cálculo del análisis de varianza (ANOVA) evidencia que no hay diferencia significativa en la duración de tiempos de reconfiguración dinámica en la ejecución de las diferentes trayectorias, como lo expresa el *p-value* de 0.4835, que es muy superior al p-value de 0.05 ampliamente aceptado en las decisiones estadísticas. Esto significa que el número de trayectorias no inciden significativamente en el tiempo empleado en la ejecución de las mismas.

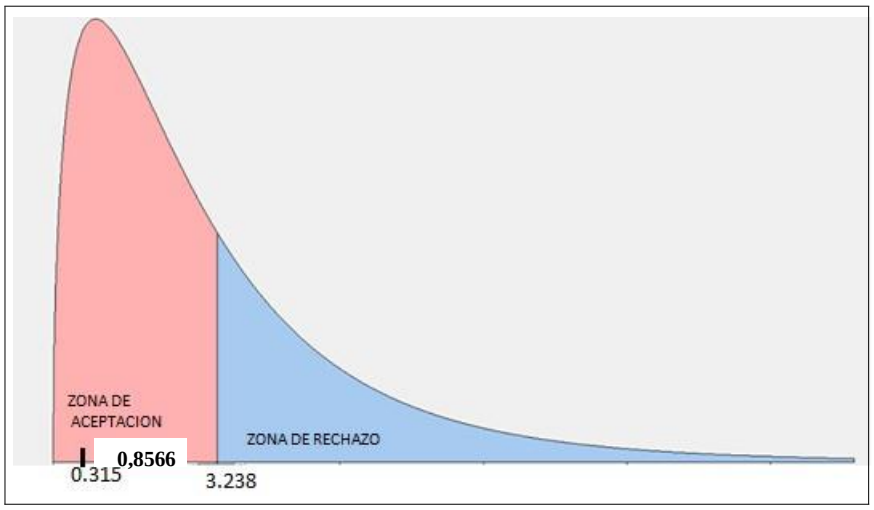


Figura 34-4: Distribución de ANOVA
Realizado por: Alex Mantilla, 2019

Una vez realizado el análisis de los datos mediante Análisis de los datos (ANOVA), se evidencia que no hay diferencia estadística significativa, por lo tanto, ; por lo tanto, el investigador tiene la certeza del 81.4% para aceptar la hipótesis nula esto quiere decir "El número de puntos que conforman las trayectorias NO inciden significativamente en el tiempo reconfiguración dinámica basada en agentes inteligentes para sistemas distribuidos en plataformas de bajo costo." concluyéndose que el número de puntos incluidos en la trayectoria no modifican el comportamiento del sistema concluyendo la ejecución de las trayectorias con éxito.

CONCLUSIONES

- Se implementó una estructura para el estudio de la cinemática inversa de un brazo Robótico planar de dos grados de libertad cuyo controlador es un Raspberry pi 3, tanto el principal, así como el Secundario o de respaldo.

- El sistema desarrollado permite evaluar si es posible o no una reconfiguración dinámica, de modo que le proceso no se vea comprometido y no se detenga.

- Los algoritmos desarrollados para los agentes toman la mejor decisión en la toma del control del proceso, imitando el comportamiento humano en el sentido del nivel de comunicación que utilizan.

- El sistema tiene la potencialidad de tener en cada Raspberry los dos agentes, de modo que el controlador principal puede ser respaldo y viceversa.

- Los sistemas de control distribuido en plataformas de bajo costo benefician en el desarrollo de nuevas aplicaciones, por las prestaciones que brindan tanto para procesos discretos, así como continuos

- Los sistemas Robóticos muchas veces son utilizados para la ejecución de procesos críticos por lo que contar sistemas de control robustos es imperioso, y en esta investigación se plantea un sistema de control con respaldo.

- Los tiempos de reconfiguración del sistema de control para la resolución de la cinemática in- versa del robot planar con dos grados de libertad, tiene un promedio de 0,7 segundos, y según el análisis estadístico (ANOVA) se determina que se encuentra dentro del rango de aceptación.

RECOMENDACIONES

- Se recomienda utilizar motores paso a paso como actuadores del brazo Robótico ya que los servos son susceptibles a realizar pequeños cambios en los ángulos de giro.

- Se debería usar una red cableada para la interconexión de los dispositivos ya que mejora la velocidad de comunicación para llevar a cabo el proceso de reconfiguración.

- Establecer un tamaño adecuado de los eslabones del brazo, para tener mayor cantidad de puntos a alcanzar.

- Si se ejecuta los agentes en Windows hay que tener en cuenta el comportamiento del Socket en la comunicación ya que es diferente al de Linux.

- Es importante hacer la elección de un buen lenguaje de programación para el desarrollo de los agentes ya que se puede realizar en varios lenguajes, sin embargo, el nivel de complejidad varía de acuerdo al lenguaje seleccionado.

- La tarjeta Raspberry debe ser la ultima versión del mercado, por la mejora en sus características de hardware.

- La red de comunicación debe ser exclusiva para la comunicación de los agentes, para disminuir el tiempo de reconfiguración.

BIBLIOGRAFÍA

Ajay D. Kshemkalyani, M. S. (s/f). *Distributed computing: principles, algorithms, and systems*.

Arno Puder Kay Rarmer, F. P. (2005). *Distributed Systems Architecture: A Middleware Approach*. MORGAN KAUFMANN PUBL INC. Recuperado de 58000/35fa8c71b1b31dab417d2db06eaf57ca

Coulouris, G., Dollimore, J., & Kindberg, T. (2012). *Distributed Systems: Concepts and Design*. *Computer* (Vol. 4). Recuperado de http://www.amazon.com/dp/0321263545

Creus, A. (2011). *INSTRUMENTACIÓN INDUSTRIAL* (Octava). México.

Gilchrist, A. (2016). *Industry 4.0 The Industrial Internet of Things*. Nonthaburi: Springer. https://doi.org/10.1007/978-1-4842-2047-4

Group, I. S. R. (2000). *Diseño y Aplicación de Modelos Multiagente para Ayuda a la Decisión*.

Hannebauer, M. (2002). *Autonomous Dynamic Reconfiguration in Multi-Agent Systems: Improving the Quality and Efficiency of Collaborative Problem Solving*. Springer. Recuperado de 184000/493482215500bcdae90278df771eb4ed

Harrington, W. (2015). *Learning Raspbian*. Recuperado de https://books.google.com/books?hl=en&lr=&id=O6HNBgAAQBAJ&oi=fnd&pg=PP1&dq=raspbian&ots=ZG2PnhATWF&sig=0QwaUR3tOkvYgRWReEo5sgesoHo

Higuera, A. G. (2005). *El control automático en la industria*. Ediciones de la Universidad de Castilla-La Mancha.

Hill, B. (2007). *IP Network-based Multi-agent Systems for Industrial Automation British Library Cataloguing in Publication Data*. Liverpool.

Hurtado, J. (s/f). Introducción a las redes de Comunicación Industrial. *Comunicaciones Industriales*, 19. Recuperado de http://www.proatec.com.mx/profinet.pdf

Instruments, N. (2016). Desarrollo de Sistemas con NI myRIO y CompactRIO.

Jennings, N. R. (2000). On agent-based software engineering. *Artificial Intelligence*, *117*(2), 277–296. https://doi.org/10.1016/S0004-3702(99)00107-1

Jennings, N. R. (2001). Building Complex Software Systems Why Agent - Oriented Approaches Are Well Suited for. *Communications of the ACM*, *44*(4), 35–41.

Lafuente, A. (2009). Introducción a los sistemas distribuidos. Recuperado de http://www.sc.ehu.es/acwlaroa/SDI/Apuntes/Cap1.pdf

Langtangen, H. P. (2009). *A Primer onScientific Programming with Python*. (Timothy J. Barth, Michael Griebel, D. E. Keyes, R. Nieminen, D. Roose, & T. Schlick, Eds.), *Media*. Oslo. https://doi.org/10.1007/978-3-642-02475-7

Mancilla Espinosa, L. E. (2008). ¿ Qué son los Agentes Inteligentes de Software ? *Gaceta ideas CONCYTEG*, *31*, 25–47.

Merlet, J.-P. (2005). *Parallel Robots*. Springer-Verlag GmbH. Recuperado de http://www.ebook.de/de/product/5298605/jean_pierre_merlet_parallel_robots.html

Merlet, J.-P. (2006). *Parallel Robots*. *Civil Engineering* (Vol. 208). https://doi.org/10.1007/1-4020-4133-0

Michael Wooldridge, N. J. (1995). *Intelligent agents:theory and practice* (Vol. 10). The Knowledge Engineering Review.

Molina Jose, G. J. (2005). Agentes y Sistemas Multiagente. *Tecnologías y Servicios para la Sociedad de la Información*, 278.

Richardson, M. (2013). *Getting Started with BeagleBone: Linux-Powered Electronic Projects With Python and JavaScript*. Maker Media, Inc. Recuperado de https://www.amazon.com/Getting-Started-BeagleBone-Linux-Powered-Electronic-ebook/dp/B00FNC57GC?SubscriptionId=0JYN1NVW651KCA56C102&tag=techkie-20&linkCode=xm2&camp=2025&creative=165953&creativeASIN=B00FNC57GC

Richardson, M., & Wallace, S. (2012). *Getting Started with Raspberry Pi (Make: Projects)*. O'Reilly Media. Recuperado de https://www.amazon.com/Getting-Started-Raspberry-Pi-Make/dp/1449344216?SubscriptionId=0JYN1NVW651KCA56C102&tag=techkie-20&linkCode=xm2&camp=2025&creative=165953&creativeASIN=1449344216

Tanenbaum, A. S. (1996). *Sistemas Operativos Distribuidos (Spanish Edition)*. Prentice Hall.

Vaidyanathan, R., & Trahan, J. (2004). *DYNAMIC RECONFIGURATION 2003/E*. SPRINGER VERLAG GMBH. Recuperado de http://www.ebook.de/de/product/3696969/ramachandran_vaidyanathan_jerry_trahan_dynamic_reconfiguration_2003_e.html

Weiss, G. (1999). *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. The MIT Press. Recuperado de https://www.amazon.com/Multiagent-Systems-Distributed-Artificial-Intelligence/dp/0262232030?SubscriptionId=0JYN1NVW651KCA56C102&tag=techkie-20&linkCode=xm2&camp=2025&creative=165953&creativeASIN=0262232030

ANEXOS

ANEXO A
Código de agente principal

```
from math import cos,sin,tan,acos,asin,atan,pi
from socket import *
import time
c=[]
NumPuntos=0
bandera=0
i=0
InicioProceso=0
l1=10.0
l2=10.0
l3=10.0
l4=10.0
ld=10.0
X2=0.0
Y2=0.0
Num1=0.0
Den1=0.0
Num2=0.0
Den2=0.0
Num3=0.0
Den3=0.0
#angulos
AnguloSend_1=0
AnguloSend_2=0
#varphyBuff=0.0
#Punto cero
CeroX=5.0
CeroY=13.38
AnguloInit1=113;
AnguloInit2=66;
NumeroProcesos=0
ConectBand=0
# DIRECCION ARDUINO
address= ( "192.168.1.177", 5000) #define server IP and port
client_socket = socket(AF_INET, SOCK_DGRAM) #Set up the Socket
client_socket.settimeout(2) #Only wait 1 second for a response
# DIRECCION python 2
#address2= ( "192.168.1.9", 5000) #define server IP and port
#server_address = ('192.168.6.138', 5000)
server_address = ('192.168.1.12', 5000) #APUNTA AL RESOALDO
#server_address = ("192.168.1.9", 5000)
client_socket2 = socket(AF_INET, SOCK_STREAM) #Set up the Socket
#client_socket2.settimeout(1) #Only wait 1 second for a response
try:
client_socket2.connect(('192.168.1.12', 9995)) #APUNTA AL RESPALDO
ConectBand=1
print ("Inicio de trabajo Con Respaldo")
except:
```

```
print ("Inicio de trabajo Sin Respaldo")
pass
while(1):
print ('Borrando Variables')
i=0
del c[:]
bandera=0
NumPuntos=0
i=0
X2=0.0
Y2=0.0
Num1=0.0
Den1=0.0
Num2=0.0
Den2=0.0
Num3=0.0
Den3=0.0
#angulos
AnguloSend_1=0
AnguloSend_2=0
AnguloInit1=113;
AnguloInit2=66;
Trama = "0," + str(AnguloInit1) + "," + str(AnguloInit2)
print("Punto HOME: "+ str (Trama))
client_socket.sendto( Trama, address) #Send the data request
#trama
Trama = 0
NumeroProcesos=0
if (ConectBand==1):
client_socket2.send('Nuevo Proceso')
print ("Ingrese el numero de Puntos del Proceso")
NumPuntos=input()
print ("Proceso a ejecutar de ", NumPuntos, " Puntos")
if (ConectBand==1):
client_socket2.send(str(NumPuntos))
print ("Ingrese los puntos separados por comas a,b ")
while (i < NumPuntos):
print ("Ingrese el Punto ",i+1)
h=input()
c.append(h)
i=i+1
if (ConectBand==1):
client_socket2.send(str(h))
print (c)
while (NumeroProcesos<=20):
for index in range(NumPuntos):
#calculos angulos brazo
izquierdo************************************************************
#calculo de angulos
PuntoFinalX=c[index][0]+CeroX
PuntoFinalY=c[index][1]+CeroY
X2=(PuntoFinalX)**2
Y2=(PuntoFinalY)**2
Num1=X2+Y2-(l1**2)-(l2**2)
```

```
Den1=2*l1*l2
#Control del angulo
**********************************************************************
if (abs(Num1/Den1)>1):
varphyBuff=(-float(acos(1)))
else:
varphyBuff=(-float(acos(Num1/Den1))) #Brazo izquierdo
#print ("Angulo 1 Phy es "+ str(varphyBuff))
#calculo de Coseno de Thetha
Num2=((PuntoFinalY)*l2*sin(varphyBuff)+(l1+(l2*cos(varphyBuff))) * PuntoFinalX)
Den2=((l1**2)+((2*cos(varphyBuff))*l1*l2)+l2**2)
Valor1=float(Num2/Den2)
#calculo de Seno de Thetha
Num3=(PuntoFinalY*(l1+(l2*cos(varphyBuff))))-(PuntoFinalX*l2*sin(varphyBuff))
Den3=Den2
Valor2=Num3/Den3
try:
Valor3=(Valor2/Valor1)
except ValueError:
raise ValueError, "x e y deben poder convertirse a enteros"
except ZeroDivisionError:
raise ZeroDivisionError, "y no puede ser cero"
AnguloSend_1=int((pi-abs(atan(Valor3)))*180/pi)
Valor3=0
print ("angulo s1 " +str(AnguloSend_1))
#cálculos ángulos brazo derecho
**********************************************************
#calculo de angulos
X2=((PuntoFinalX-ld))**2
Y2=(PuntoFinalY)**2
Num1=X2+Y2-(l3**2)-(l4**2)
Den1=2*l3*l4
#Control del angulo ************************************************************
if (abs(Num1/Den1)>1):
varpsiBuff=(float(acos(1)))
else:
varpsiBuff=(float(abs(acos(Num1/Den1))))
#print ("Angulo 2 Psi es "+ str(varpsiBuff))
#calculo de Coseno de Thetha
Num2=((PuntoFinalY)*l4*sin(varpsiBuff)+(l3+(l4*cos(varpsiBuff))) * (PuntoFinalX-ld))
Den2=((l3**2)+((2*cos(varpsiBuff))*l3*l4)+l4**2)
Valor1=float(Num2/Den2)
#calculo de Seno de Thetha
Num3=(PuntoFinalY*(l3+(l4*cos(varpsiBuff))))-((PuntoFinalX-ld)*l4*sin(varpsiBuff))
Den3=Den2
Valor2=Num3/Den3
#print ("Num1= "+ str(Num1))
#print ("Den1= "+ str(Den1))
#print ("Num2= "+ str(Num2))
#print ("Den2= "+ str(Den2))
#print ("Num3= "+ str(Num3))
#print ("Den3= "+ str(Den3))
#print ("Valor1= "+ str(Valor1))
#print ("Valor2= "+ str(Valor2))
```

```
try:
Valor3=Valor2/Valor1
except ValueError:
raise ValueError, "x e y deben poder convertirse a enteros" #poner un pass o print o ambos
except ZeroDivisionError:
raise ZeroDivisionError, "y no puede ser cero"
AnguloSend_2=int(atan(Valor3)*180/pi)
print ("angulo s2 " +str(AnguloSend_2))
#Trama = "PA" + str(NumPuntos) + "=" + str(AnguloSend_1) + "," + "PB" + str(NumPuntos) +
"=" + str(AnguloSend_2
Trama = str(index + 1) + "," + str(AnguloSend_1) + "," + str(AnguloSend_2)
#print("t"+str(Trama))
#envio trama a arduino
client_socket.sendto( Trama, address) #Send the data request
#trato de recivir respuesta de confirmacion de punto alcanzaod oaccion desde el arduino
try:
rec_data, addr = client_socket.recvfrom(2048) #Read response from arduino
bandera = int (rec_data)
print(bandera)
while (bandera!= 41):
print ("No llega 41")
time.sleep(1)
except:
pass
# aviso a Python 2 que arduino alcanzo el punto que se envio y que no debe entrara a aslavar el
proceso
if (ConectBand==1):
client_socket2.sendto( str(bandera), server_address) #Send the data request
print ("Enviando :" + str(bandera))
#espero confirmacion de Python 2 para seguir con el calculo del siguiente punto
rec_data2, addr2 = client_socket2.recvfrom(2048) #Read response from
bandera2 = int (rec_data2)
print(bandera2)
while (bandera2!=51):
print ("No llega data")
time.sleep(1)
#client_socket2.close()
NumeroProcesos=NumeroProcesos+1
```

ANEXO B

Código de agente de respaldo

```
import os
#os.system('clear')
from math import cos,sin,tan,acos,asin,atan,pi
#importamos el modulo socket
import socket
import logging
#importamos el modulo socket
import time
#instanciamos un objeto para trabajar con el socket
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
#Server_address= ( "", 9998) #define server IP and port
```

```
#Con el metodo bind le indicamos que puerto debe escuchar y de que servidor esperar conexiones
#Es mejor dejarlo en blanco para recibir conexiones externas si es nuestro caso
s.bind(("", 9995))
#Aceptamos conexiones entrantes con el metodo listen, y ademas aplicamos como parametro
#El numero de conexiones entrantes que vamos a aceptar
s.listen(1)
#Instanciamos un objeto sc (socket cliente) para recibir datos, al recibir datos este
#devolvera tambien un objeto que representa una tupla con los datos de conexion: IP y puerto
sc, addr = s.accept()
# DIRECCION ARDUINO
address= ( "192.168.1.177", 5000) #define server IP and port
client_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
#client_socket.settimeout(1) #Only wait 1 second for a response
VarSwitchCase=0
BandSendToARduino=0
recibido=0
c=[]
NumPuntos=0
bandera=0
i=0
InicioProceso=0
l1=10.0
l2=10.0
l3=10.0
l4=10.0
ld=10.0
X2=0.0
Y2=0.0
Num1=0.0
Den1=0.0
Num2=0.0
Den2=0.0
Num3=0.0
Den3=0.0
#angulos
AnguloSend_1=0
AnguloSend_2=0
BanderaRepetir=0
#Punto cero
CeroX=5.0
CeroY=13.38
AnguloInit1=113;
AnguloInit2=66;
NumeroProcesos=0
#tiempo de reconfiguracion
inicio_tiempo=0
while (1):
print ('Borrando Variables')
i=0
del c[:]
bandera=0
NumPuntos=0
i=0
```

```
X2=0.0
Y2=0.0
Num1=0.0
Den1=0.0
Num2=0.0
Den2=0.0
Num3=0.0
Den3=0.0
#angulos
AnguloSend_1=0
AnguloSend_2=0
#trama
Trama = 0
NumeroProcesos=0
#Recibimos el mensaje, con el metodo recv recibimos datos y como parametro
#la cantidad de bytes para recibir
print ("Esperando Trama")
recibido = sc.recv(1024)
if recibido == "Nuevo Proceso":
VarSwitchCase=0
BandSendToARduino=0
recibido=0
if recibido == "close":
sc.close()
s.close()
recibido=0
recibido = sc.recv(1024)
if recibido == "close":
sc.close()
s.close()
try:
NumPuntos= int(recibido)
except ValueError:
break
except ZeroDivisionError:
break
recibido=0
print ("Numero de Puntos: " + str (NumPuntos))
while (i < NumPuntos):
try:
recibido=sc.recv(1024)
except:
pass
c.append(eval(recibido))
i=i+1
print (recibido)
if recibido == "close":
sc.close()
s.close()
recibido=0
print ("Variable c es:")
print (c)
#sc.settimeout(3) #Only wait 1 second for a response
while(NumeroProcesos<=20):
```

```
for index in range(NumPuntos):
#calculos angulos brazo
izquierdo**********************************************************
#calculo de angulos
PuntoFinalX=c[index][0]+CeroX
PuntoFinalY=c[index][1]+CeroY
X2=(PuntoFinalX)**2
Y2=(PuntoFinalY)**2
Num1=X2+Y2-(l1**2)-(l2**2)
Den1=2*l1*l2
#Control del angulo
*************************************************************************
if (abs(Num1/Den1)>1):
varphyBuff=(-float(acos(1)))
else:
varphyBuff=(-float(acos(Num1/Den1))) #Brazo izquierdo
#print ("Angulo 1 Phy es "+ str(varphyBuff))
#calculo de Coseno de Thetha
Num2=((PuntoFinalY)*l2*sin(varphyBuff)+(l1+(l2*cos(varphyBuff))) * PuntoFinalX)
Den2=((l1**2)+((2*cos(varphyBuff))*l1*l2)+l2**2)
Valor1=float(Num2/Den2)
#calculo de Seno de Thetha
Num3=(PuntoFinalY*(l1+(l2*cos(varphyBuff))))-(PuntoFinalX*l2*sin(varphyBuff))
Den3=Den2
Valor2=Num3/Den3
try:
Valor3=(Valor2/Valor1)
except ValueError:
raise ValueError, "x e y deben poder convertirse a enteros"
except ZeroDivisionError:
raise ZeroDivisionError, "y no puede ser cero"
AnguloSend_1=int((pi-abs(atan(Valor3)))*180/pi)
Valor3=0
print ("angulo s1 " +str(AnguloSend_1))
#calculos angulos brazo derecho
**************************************************************
#calculo de angulos
X2=((PuntoFinalX-ld))**2
Y2=(PuntoFinalY)**2
Num1=X2+Y2-(l3**2)-(l4**2)
Den1=2*l3*l4
#Control del angulo
*************************************************************************
if (abs(Num1/Den1)>1):
varpsiBuff=(float(acos(1)))
else:
varpsiBuff=(float(abs(acos(Num1/Den1))))
#print ("Angulo 2 Psi es "+ str(varpsiBuff))
#calculo de Coseno de Thetha
Num2=((PuntoFinalY)*l4*sin(varpsiBuff)+(l3+(l4*cos(varpsiBuff))) * (PuntoFinalX-ld))
Den2=((l3**2)+((2*cos(varpsiBuff))*l3*l4)+l4**2)
Valor1=float(Num2/Den2)
#calculo de Seno de Thetha
Num3=(PuntoFinalY*(l3+(l4*cos(varpsiBuff))))-((PuntoFinalX-ld)*l4*sin(varpsiBuff))
```

```
Den3=Den2
Valor2=Num3/Den3
#print ("Num1= "+ str(Num1))
#print ("Den1= "+ str(Den1))
#print ("Num2= "+ str(Num2))
#print ("Den2= "+ str(Den2))
#print ("Num3= "+ str(Num3))
#print ("Den3= "+ str(Den3))
#print ("Valor1= "+ str(Valor1))
#print ("Valor2= "+ str(Valor2))
try:
Valor3=Valor2/Valor1
except ValueError:
raise ValueError, "x e y deben poder convertirse a enteros" #poner un pass o print o ambos
except ZeroDivisionError:
raise ZeroDivisionError, "y no puede ser cero"
AnguloSend_2=int(atan(Valor3)*180/pi)
print ("angulo s2 " +str(AnguloSend_2))
#Trama = "PA" + str(NumPuntos) + "=" + str(AnguloSend_1) + "," + "PB" + str(NumPuntos) +
"=" + str(AnguloSend_2
Trama = str(index + 1) + "," + str(AnguloSend_1) + "," + str(AnguloSend_2)
recibido=0
#Comunicacion
# trato de recibir la bandera que envio el arduino al Python Cliente
try:
recibido = sc.recv(1024)
print (recibido)
if (recibido == "41"):
print ("Llego 41")
sc.send(str(51)) #Respondo a Python 1 con bandera para sieguiente punto
else:
BanderaRepetir=1
inicio_tiempo=time.time()
#print(str(inicio_tiempo))
except:
pass
#Entramos a salvar el proceso xq se levando la bandera
if (BanderaRepetir==1):
client_socket.sendto( Trama, address) #Send the data request
try:
rec_data, addr = client_socket.recvfrom(2048) #Read response from arduino
bandera = int (rec_data)
print("paso: " + str(index)+" tiempo de reconfiguracion "+str(time.time()-inicio_tiempo))
print(bandera)
while (bandera!= 41):
print ("No llega 41")
time.sleep(1)
except:
pass
NumeroProcesos=NumeroProcesos+1
while (BanderaRepetir==1):
Trama = "0," + str(AnguloInit1) + "," + str(AnguloInit2)
print("Punto HOME: "+ str (Trama))
client_socket.sendto( Trama, address) #Send the data request
```

```
print ("Proceso salvado Revisar el Sistema")
time.sleep(2)
#Cerramos la instancia del socket cliente y servidor Usar break para salir a este nivel
```

ANEXO C

Código de Arduino

```
#include <Ethernet.h> //Load Ethernet Library
#include <EthernetUdp.h> //Load the Udp Library
#include <SPI.h> //Load SPI Library
#include "Wire.h" //imports the wire library
#include <Servo.h>
Servo myservo1; // create servo object to control a servo
Servo myservo2; // create servo object to control a servo
byte mac[] ={ 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xEE}; //Assign mac address
IPAddress ip(192, 168, 1, 177); //Assign the IP Adress
unsigned int localPort = 5000; // Assign a port to talk over
char packetBuffer[UDP_TX_PACKET_MAX_SIZE]; //dimensian a char array to hold our data
packet
int packetSize; //Size of the packet
EthernetUDP Udp; // Create a UDP Object
void setup() {
Serial.begin(38400); //Initialize Serial Port
Ethernet.begin( mac, ip); //Inialize the Ethernet
Udp.begin(localPort); //Initialize Udp
delay(1500); //delay
myservo1.attach(5); // attaches the servo on pin 9 to the servo object
myservo2.attach(6 ); // attaches the servo on pin 10 to the servo object
Serial.println("INIT");
}
void loop() {
packetSize =Udp.parsePacket(); //Reads the packet size
if(packetSize>0)
{
Udp.read(packetBuffer, UDP_TX_PACKET_MAX_SIZE); //Read the data request
//Serial.println(packetBuffer);
char Rxchar1[10];
/**********************************************Numero de
paso*****************************************/
strcpy(Rxchar1, strtok(packetBuffer,","));
Serial.print ("Paso: ");
Serial.println(atoi(Rxchar1));
for (int i=0;i<10;i++){Rxchar1[i]=0;}
```

```
/**********************************************ANgulo
1*****************************************/
strcpy(Rxchar1, strtok(NULL,","));
myservo1.write(atoi(Rxchar1));
Serial.println(myservo1.read());
Serial.println(atoi(Rxchar1));
for (int i=0;i<10;i++){Rxchar1[i]=0;}
/**********************************************Angulo
2*****************************************/
strcpy(Rxchar1, strtok(NULL,","));
myservo2.write(atoi(Rxchar1));
Serial.println(myservo2.read());
Serial.println(atoi(Rxchar1));
for (int i=0;i<10;i++){Rxchar1[i]=0;}
delay(300);
Udp.beginPacket(Udp.remoteIP(), Udp.remotePort()); //Initialize packet send
Udp.print(41); //Send the temperature data
Udp.endPacket(); //End the packet
}
memset(packetBuffer, 0, UDP_TX_PACKET_MAX_SIZE); //clear out the packetBuffer array
}
```