REPÚBLICA DEL ECUADOR
Escuela Politécnica Nacional
"E SCIENTIA HOMINIS SALUS"

La versión digital de esta tesis está protegida por la Ley de Derechos de Autor del Ecuador.

Los derechos de autor han sido entregados a la "ESCUELA POLITÉCNICA NACIONAL" bajo el libre consentimiento de los autores.

Al consultar esta tesis deberá acatar con las disposiciones de la Ley y las siguientes condiciones de uso:

- Cualquier uso que haga de estos documentos o imágenes deben ser sólo para efectos de investigación o estudio académico, y usted no puede ponerlos a disposición de otra persona.

- Usted deberá reconocer el derecho del autor a ser identificado y citado como el autor de esta tesis.

- No se podrá obtener ningún beneficio comercial y las obras derivadas tienen que estar bajo los mismos términos de licencia que el trabajo original.

El Libre Acceso a la información, promueve el reconocimiento de la originalidad de las ideas de los demás, respetando las normas de presentación y de citación de autores con el fin de no incurrir en actos ilegítimos de copiar y hacer pasar como propias las creaciones de terceras personas.

Respeto hacia sí mismo y hacia los demás.

ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

DISEÑO E IMPLEMENTACIÓN DE UN BANCO DE PRUEBAS DE EXPERIMENTACIÓN CON ROBOTS MÓVILES TERRESTRES DE TAMAÑO REDUCIDO PARA EL CONTROL DE POSICIÓN Y SEGUIMIENTO DE TRAYECTORIA

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN "ELECTRÓNICA Y CONTROL"

GUSTAVO ISRAEL JÁCOME MORETA

gustavo.jacome@epn.edu.ec

YESENIA MARISOL SIERRA JÁCOME

yesenia.sierra@epn.edu.ec

DIRECTOR: ING. PATRICIO JAVIER CRUZ DÁVALOS, PhD

patricio.cruz@epn.edu.ec

Quito, marzo 2019

AVAL

Certifico que el presente trabajo fue desarrollado por Gustavo Israel Jácome Moreta y Yesenia Marisol Sierra Jácome, bajo mi supervisión.

Ing. Patricio Javier Cruz Dávalos, PhD
DIRECTOR DEL TRABAJO DE TITULACIÓN

DECLARACIÓN DE AUTORÍA

Nosotros, Gustavo Israel Jácome Moreta, Yesenia Marisol Sierra Jácome, declaramos bajo juramento que el trabajo aquí descrito es de nuestra autoría; que no ha sido previamente presentada para ningún grado o calificación profesional; y, que hemos consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedemos nuestros derechos de propiedad intelectual correspondientes a este trabajo, a la Escuela Politécnica Nacional, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

Gustavo Israel Jácome Moreta	Yesenia Marisol Sierra Jácome

DEDICATORIA

A mi Dios por colmarme de muchas bendiciones a lo largo de mi vida y haberme permitido culminar este proyecto.

A mi mami Laura y a mi papi Fabián, por ese amor incondicional que siempre me han brindado.

A mis hermanos Andrés y Laura, por toda la confianza depositada y el cariño que me han demostrado.

A mi novia Eri, por quererme mucho, apoyarme en cada decisión y por llenar mi vida de felicidad.

Gustavo Israel

DEDICATORIA

El resultado de este trabajo se lo dedico a mi mami. Esto es resultado de tu esfuerzo y cariño que me has dado.

A Cris, por ser mi ejemplo de inspiración y perseverancia.

A Walter, por ser mi ejemplo de esfuerzo y paciencia.

Marisol

AGRADECIMIENTO

Agradezco a Dios por ayudarme a culminar este proyecto, por todas las bendiciones derramadas sobre mí, y sobre todos mis seres queridos. Gracias por guiar mis pasos en todo momento.

A mis padres, por su incondicional apoyo a lo largo de mi vida, por sus consejos y enseñanzas que me han servido para formar mi carácter.

A mis hermanos, con los cuales hemos compartido buenos y malos momentos, en especial gracias Andrés por la confianza y aprecio que siempre me has tenido.

A mi novia, muchas gracias por la paciencia, respeto y amor que me brindas día a día.

A mi compañera de tesis y amiga Marisol, por todo el apoyo y perseverancia demostrada para llevar a cabo nuestro proyecto; especialmente gracias por tu amistad desinteresada y sincera.

Gracias a mi director Dr. Patricio Cruz por sus consejos, empeño y ayuda brindada a lo largo del proyecto.

Gracias Escuela Politécnica Nacional, por haberme acogido y permitirme conocer a excelentes maestros, en particular a aquellos que enseñan no solo materias de ingeniería, sino que también se preocupan del crecimiento personal y profesional de sus alumnos. Muchas gracias a mis maestros Dr. Franklin Quilumba, Dr. Marcelo Pozo, Ing. Nelson Sotomayor, Ing. Cristian Tasiguano e Ing. Silvana Gamboa.

Gustavo Israel

AGRADECIMIENTO

Primeramente, agradezco a Dios por permitirme acabar esta larga etapa.

Este largo proceso lo he culminado gracias a la mejor persona de mi vida, mi mami. Gracias mami por todo, no solo por darme un hogar lleno de alegría sino por ser ese ejemplo de fortaleza para mí. Este triunfo no es solo mío, es tuyo también. Te quiero mami.

Cris, tú has sido mi mayor y mejor ejemplo por seguir, gracias por tanto apoyo y cariño. Gracias por los consejos, por siempre estar ahí siendo mi mejor amiga. Berni, gracias por brindarme una idea, consejo y animo cuando lo necesité.

A mi familia, gracias por su apoyo y guía. Gracias ñaña y ñaño Marce. A mis chiquitas, Nichi y Sami por siempre confiar en mí.

El inicio de este proyecto fue gracias a mi amigo Gus. Gus, gracias por el apoyo, empeño, y paciencia que dedicaste a lo largo de este tiempo. No sólo eres un excelente profesional, también eres una excelente persona.

A lo largo de la carrera conoces personas que hacen de la vida universitaria una excelente experiencia, ellos fueron mis amigos Mary, Vic, Andrés V., Gabo, Marito, Andre, Jeampool, y Chris G. A mis dos mejores amigas del cole, gracias Anita y Eri por escucharme siempre y apoyarme.

Este es el primero de muchos retos que se cumplirán a tu lado, Walter. Gracias mi Flaco, por ser la sonrisa de todos los días. Por apoyarme siempre que lo necesito y por ser mi mejor amigo y mi pareja. Te amo.

Gracias Dr. Patricio Cruz, por la confianza y dedicación brindada a lo largo del proyecto. Este es el resultado de varias horas de trabajo y consejos que usted nos brindó.

Gracias a la Escuela Politécnica Nacional, por abrirme las puertas para superarme. Un agradecimiento especial al Ing. Cristian Tasiguano por enseñarme que las cosas se deben hacer bien y con tiempo y a la Ing. Silvana Gamboa por su excelente método de enseñar y valorar a las mujeres dentro de la facultad.

Marisol

ÍNDICE DE CONTENIDO

AVAL I
DECLARACIÓN DE AUTORÍA II
DEDICATORIA III
DEDICATORIA IV
AGRADECIMIENTO V
AGRADECIMIENTO VI
ÍNDICE DE CONTENIDO VII
RESUMEN XII
ABSTRACT XIII
1. INTRODUCCIÓN 1
1.1 Objetivos 2
1.2 Alcance 2
1.3 Marco Teórico 3
Robótica móvil 3
Clasificación de la robótica 4
Clasificación de robots móviles con ruedas 5
Bancos de pruebas 6
Clasificación de bancos de pruebas según el rango de experimentos 8
Clasificación de bancos de pruebas según su arquitectura 8
Clasificación de bancos de pruebas según el usuario objetivo 9
Modelo matemático del robot uniciclo 11
Modelo cinemático con restricción no holonómica mejorada 14
Control de robots móviles 15
Control de posición 15

Seguimiento de caminos 15
Seguimiento de trayectorias 16
Métodos de control 17
Controlador tipo PID 18
Controlador SMC 18
Controlador basado en álgebra lineal 19
Visión por computadora 20
Técnicas de visión por computadora 21
Calibración de cámara 24
Modelo de la cámara 25
Tipos de marcadores de referencia 30
Marcadores de referencia circular 31
Marcadores de referencia cuadrados sin borde 31
Marcadores de referencia cuadrados con borde 32
Marcador de referencia matricial ArUco 32
Detección de marcadores de referencia matricial 35
Procesamiento de Imágenes 35
Filtrado de contorno 36
Extracción de bits 36
Protocolo de comunicación ZigBee 38
Topología ZigBee 39
Trabajos similares 40
2. METODOLOGÍA 42
2.1 Robots móviles de tamaño reducido 44
Componentes electrónicos 44
Minimotores DC 44
Módulo DRV8835 45

XBee Serie 1 46
Fio V3- Atmega 32U4 46
Baterías de Litio 47
Componentes mecánicos 48
Ruedas 48
Bola de castor 48
Estructura mecánica de polímero 49
Implementación de los robots móviles 51
2.2 Sistema de monitoreo 53
Cámara de video 53
Generación de Marcador ArUco 56
Calibración de cámara 60
Modelo de la cámara 62
Detección de marcadores de referencia ArUco 63
Conversión a escala de grises 63
Binarización de imagen 63
Dibujo de contornos 64
Detección de Contornos 65
Filtrado de Contornos 65
Búsqueda de posibles marcadores 66
Estimación de posición y orientación del marcador 70
Matlab Engine 75
2.3 Sistema de control 77
Diseño del algoritmo de control 77
Implementación del controlador 80
Generación de trayectorias 83
2.4 Sistema de comunicación 84

Configuración de los módulos XBee 85
Configuración del coordinador 87
Configuración de los dispositivos finales 88
2.5 Sistema de visualización 89
Ventanas Principales 90
Ventana de selección de parámetros 92
Ventana de Ingreso de Datos 94
Ventanas emergentes 95
Ventanas de ayuda 96
Ventana de visualización y resultados 96
Ventana de Análisis de gráficos 98
3. RESULTADOS Y DISCUSIÓN 99
3.1 Pruebas de identificación de marcadores ArUco 99
3.2 Pruebas de comunicación entre robots y ordenador 101
Prueba de envío de comandos a los robots 101
Prueba de lectura de estado de batería 102
3.3 Calibración del tiempo de adquisición de datos 102
Pruebas de tiempo de retardo del sistema 102
Prueba de tiempo de muestreo 104
3.4 Prueba de diferentes ganancias de control en trayectoria circular 105
3.5 Prueba de seguimiento de trayectorias ante diferentes valores de velocidad lineal 109
Trayectoria Lineal con pendiente 0° 110
Trayectoria Lineal con pendiente de 45° 113
Trayectoria Doble frecuencia 117
3.6 Resultados obtenidos del control implementado para dos robots 120
Prueba con dos robots trayectoria lineal con pendiente de 45° 120

3.7 Análisis de perturbaciones aplicadas al seguimiento de trayectoria de doble frecuencia 124

4. CONCLUSIONES 127

RECOMENDACIONES 128

5. REFERENCIAS BIBLIOGRÁFICAS 130

6. ANEXOS 137

RESUMEN

En la actualidad muchas de los proyectos realizados dentro de la Escuela Politécnica Nacional en el área de robótica se han enfocado principalmente en el diseño y simulación de algoritmos de control para el manejo de diferentes tipos de robots. En particular, se han desarrollado nuevas técnicas de control para robots móviles a partir de estudios y simulaciones con la finalidad de implementar dichos algoritmos en plataformas robóticas. Una de las plataformas utilizadas generalmente para el estudio de esta clase de robots es la de tipo diferencial, cuyo modelo matemático puede ser analizado a través del robot tipo uniciclo. El proyecto que se presenta a continuación tiene como finalidad la creación de un banco de pruebas enfocado al control de posición y seguimiento de trayectorias de dos robots móviles tipo diferencial de tamaño reducido (small sized robots). Este banco de pruebas es accesible tanto para docentes como estudiantes de la Escuela Politécnica Nacional, contribuyendo a futuros trabajos de investigación y proyectos en coordinación de robots móviles, evasión de obstáculos, e implementación de algoritmos avanzados de control, entre otros. El banco de pruebas implementado cuenta con un sistema de monitoreo mediante una cámara web y un sistema de control corriendo en una computadora central que se comunica con los robots mediante radio frecuencia. Además, se dispone de una interfaz gráfica que permite el manejo y la obtención de resultados del comportamiento de los robots para el seguimiento de diferentes trayectorias establecidas.

PALABRAS CLAVE: Banco de pruebas, robot tipo uniciclo, seguimiento de trayectorias, sistema de monitoreo, sistema de control.

ABSTRACT

Currently, many of the projects that have been carried out at Escuela Politécnica Nacional, in the robotics area have focused mainly in the design and simulation of control algorithms for the coordination of different types of robots. In particular; new control techniques have been developed for mobile robots from previous studies and simulations in order to implement these algorithms on robotic platforms. One of the platforms that has been used in many studies is the differential type whose mathematical model can be analyzed by assuming the unicycle robot. The purpose of the project presented below is to create a testbed focused on the position control and trajectory tracking of two differential small sized robots. This testbed is accessible to both teachers and students of Escuela Politécnica Nacional, helping to future research projects focused on coordination of mobile robots, obstacle avoidance, and implementation of advanced control algorithms. The testbed implemented consist of a monitoring system composed by a web cam and a control system running in a main computer that communicates with the robots throughout radiofrequency. Beside, this testbed has an user graphic interface that allows to manage and get results of the robots behavior for the tracking of different pre-defined trajectories.

KEYWORDS: Testbed, unicycle type robot, trajectory tracking, monitoring system, control system.

1. INTRODUCCIÓN

El desarrollo tecnológico en las últimas décadas ha permitido extender el área de investigación del campo de la robótica, mediante mecanismos inteligentes que contribuyen en diferentes aplicaciones. Un ejemplo de esto es la plataforma de aprendizaje Duckietown [1] que permite trabajar con robots móviles mediante el uso de sensores para la detección de objetos, percepción y seguimiento de trayectorias. Este sistema ha sido utilizado para crear prototipos de vehículos autónomos con la colaboración de CSAIL y TOYOTA.

Otra de las técnicas para realizar pruebas de investigación de robótica móvil es Robotarium [2], un banco de pruebas (testbed) que permite probar algoritmos de control para grupos de robots y tiene como principal objetivo la mínima intervención y mantenimiento del operador. Se basa en el seguimiento de posición y ejecución de rutas de movimiento sin colisiones probables para robots prediseñados llamados GRITSBOTS.

Por esto, el presente proyecto plantea diseñar e implementar un banco de pruebas para el control de posición y seguimiento de trayectorias individuales de robots móviles de tamaño reducido, que consta de dos componentes principales hardware y software.

El primer componente está representado por la electrónica de los robots móviles y del sistema de visión. El segundo está constituido por un sistema de monitoreo que permite el rastreo individual de cada robot móvil mediante un marcador de referencia matricial dentro del área de trabajo y un sistema de control, el cual, en base al procesamiento de información, definirá las acciones para cada robot permitiéndoles moverse independientemente en sus trayectorias preestablecidas.

Disponer de un sistema que permita el control de un conjunto de robots móviles representa un papel importante en trabajos de investigación. Si bien el proyecto se enfoca en el control de posición y de seguimiento de trayectorias individuales para un grupo de robots de tamaño reducido, cabe resaltar que la implementación de dicho banco de pruebas servirá para futuros proyectos de investigación, como por ejemplo el estudio del comportamiento colectivo de sistemas multiagentes; así como, el desarrollo y comprobación experimental de nuevas técnicas de control para formación de robots.

1.1 Objetivos

El objetivo general de este proyecto es:

Diseñar e implementar un banco de pruebas de experimentación con robots móviles terrestres de tamaño reducido para el control de posición y seguimiento de trayectoria.

Los objetivos específicos de este proyecto son:

- Realizar la revisión bibliográfica del modelamiento y control de un robot móvil tipo uniciclo, así como de ejemplos de bancos de pruebas en base a robots móviles de este tipo considerando un tamaño reducido.
- Ensamblar mecánica y electrónicamente prototipos de robots móviles tipo uniciclo de tamaño reducido.
- Integrar un banco de pruebas que conste de dos robots móviles, un sistema de monitoreo/posicionamiento y una interfaz gráfica en una computadora.
- Diseñar e implementar un algoritmo de control para el control de posición y seguimiento de trayectorias individuales, para cada uno de los robots.
- Verificar el comportamiento del algoritmo de control desarrollado, utilizando el testbed implementado, en base a pruebas individuales y grupales que permitan determinar errores de la técnica de control escogida.

1.2 Alcance

- Se realizará la correspondiente revisión bibliográfica para establecer el modelo matemático del robot tipo uniciclo, así como los parámetros físicos considerados dentro del modelo.
- Se realizará la correspondiente revisión bibliográfica de técnicas de control de posición y seguimiento de trayectoria del robot tipo uniciclo, para el diseño e implementación de un algoritmo de control de la posición y seguimiento de trayectorias individuales, sin considerar el control de formación de robots.
- Se realizará el estudio de diferentes plataformas conformadas por robots móviles tipo uniciclo y un sistema de visión artificial para comparación y selección de un banco de pruebas accesible.
- Se escogerá, las partes de la estructura mecánica y electrónica de los prototipos de robots móviles (motores, drivers, ruedas, estructura, tarjetas embebidas, módulos de comunicación inalámbrica, baterías, etc.) garantizando que los

robots cumplan con las características para que sean catalogados como de tamaño reducido (mini sized robots).

- Considerando el prototipo seleccionado se implementarán dos robots móviles tipo uniciclo de tamaño reducido, tomando en cuenta las dimensiones específicas. (mini sized robots) [3] [4].
- Se adquirirán la cámara web de alta definición y su soporte para conformar el sistema de posicionamiento basado en visión artificial, posteriormente se implementará el sistema de monitoreo mediante el uso de una cámara web colocada a una altura adecuada para la detección de los robots dentro del área de trabajo al menos de 120 cm x 80 cm.
- Se seleccionarán y establecerán los marcadores de referencia matricial [5] colocados en cada robot uniciclo para adquirir la posición de los robots móviles dentro del área de trabajo.
- Se diseñará la interfaz gráfica en una computadora, desarrollada con un lenguaje de programación orientado a objetos que permite visualizar al conjunto de robots móviles.
- Se utilizará el protocolo Zigbee para la comunicación inalámbrica entre el ordenador y los robots móviles tipo uniciclo.
- Se diseñará e implementará un algoritmo de control de posición y seguimiento de trayectorias individuales sin considerar el control de formación de robots.
- Se integrará el banco de pruebas para el monitoreo y control del grupo de robots móviles tipo uniciclo de tamaño reducido.
- Se realizarán las pruebas correspondientes del sistema de control y monitoreo para comprobar el correcto funcionamiento del sistema implementado a través de pruebas individuales de los robots móviles sometidos a diferentes trayectorias, verificando la técnica de control escogida mediante el cálculo de errores.

1.3 Marco Teórico

Robótica móvil

Se puede definir a la robótica móvil como el diseño y empleo de robots que presenta algún mecanismo de locomoción que les permiten realizar tareas específicas, sustituyendo la intervención directa del hombre. Actualmente, el diseño e

implementación de nuevas técnicas de control han permitido una mejor interacción entre los robots móviles y el medio en donde realizan tareas específicas, en actividades industriales, hospitalarias, militares, investigativas y educativas. [6]

Sin embargo, al hablar acerca de robots móviles hacemos referencia a una parte dentro de la clasificación de la robótica, por lo cual es necesario conocer esta clasificación de manera general con la finalidad de tener una visión más completa al momento de identificar el modelo del robot usado en el presente proyecto.

Clasificación de la robótica

Dentro del campo de la robótica existen diferentes características en base a las cuales se puede clasificar a los robots, tales como: [4]

- Posicionamiento.
- Aplicaciones.
- Tamaño.
- Locomoción.

En la Figura 1.1 se puede observar un esquema de clasificación de la robótica.

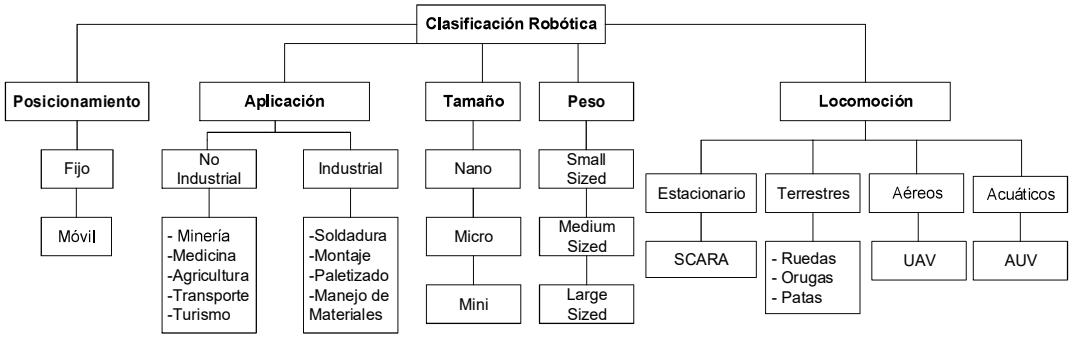


Figura 1.1. Esquema de clasificación de la robótica.

De acuerdo con la clasificación, los robots usados en el proyecto se los puede catalogar como:

- ***Robots Móviles:*** Su estructura permite el movimiento entero del robot.
- ***No Industriales:*** Existen robots que ayudan a la producción de bienes materiales, como en el sector de agricultura y minería. Además, los que brindan un soporte al operario, utilizados en el área de medicina, transporte, entre otros.
- ***Mini:*** Son robots que su volumen alcanza hasta $1dm^3$. [3]
- ***Small Sized*****:** Los robots que no excedan los 6 Kg de masa.

- ***Robots Terrestres con ruedas:*** Son robots que poseen una estructura de movilidad con ruedas, que les permite desplazarse en superficies duras y lisas.

A partir de lo expuesto en esta sección a los robots implementados en este proyecto se los clasifica según su aplicación, locomoción, posicionamiento y tamaño en la sección de Diseño de la estructura de los prototipos.

Debido a la amplia gama de estos robots se indica a continuación la clasificación de los robots móviles con ruedas, para posteriormente seleccionar el modelo del robot móvil a ser utilizado en nuestro proyecto.

Clasificación de robots móviles con ruedas

Dependiendo del tipo de superficie por la cual se debe transportar el robot móvil permite establecer el mecanismo de locomoción adecuado. El caso más sencillo es dotar al robot de ruedas siempre y cuando la superficie sea lo suficientemente estable. Los robots móviles con ruedas pueden presentar uno o más tipos de ruedas: [7]

- ***Rueda motriz:*** Proporciona la fuerza de traslación del robot móvil.
- ***Rueda directriz:*** Es controlable permitiendo la orientación del robot.
- ***Rueda fija:*** No tiene tracción y gira en torno a su eje.
- ***Rueda de castor:*** Conocida como rueda loca, no es controlable y sirve como apoyo del robot.

Por otro lado, en base a las variaciones del diseño del sistema de traslación de los robots móviles con ruedas podemos tener [7,8]:

Ackerman: Este diseño brinda mayor estabilidad, se aplica en los automóviles convencionales.

Triciclo: El diseño tradicional presenta dos ruedas traseras pasivas que brindan estabilidad al robot móvil, y una rueda delantera motriz que cumple al mismo tiempo la tarea de ser directriz.

Omnidireccional: Esta configuración permite que el robot móvil se desplace en cualquier dirección, sin necesidad de una orientación.

Síncrono: Consta de tres ruedas orientadas en una misma dirección, las ruedas son motrices y a la vez directrices.

Diferencial: Se considera el diseño más simple, conformado de dos ruedas motrices perpendiculares a la dirección del robot móvil y para generar mayor estabilidad se colocan una o dos ruedas de castor o bola loca que sirven de apoyo (Figura 1.2).

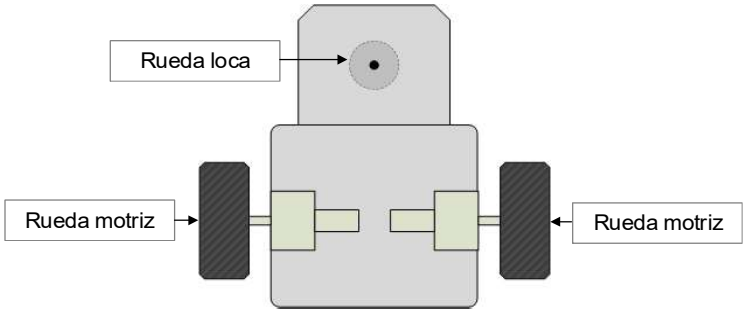


Figura 1.2. Configuración tipo diferencial.

A partir de la clasificación presentada en la sección de Clasificación de robots móviles con ruedas, se elige el diseño del robot móvil diferencial como modelo de los prototipos de nuestro banco de pruebas, considerando que los robots móviles se desplazarán en una superficie lo más homogénea posible.

Una vez elegido el tipo de robot a implementar, el presente proyecto busca crear un banco de pruebas que permita la integración de los robots con diferentes dispositivos fundamentales para su correcto funcionamiento. Existen varios tipos de banco de pruebas que reúnen diferentes funcionalidades; a continuación, se detallan generalidades de los bancos de pruebas robóticos.

Bancos de pruebas

La necesidad de ubicar robots móviles en red, con sus respectivos sensores, cámaras, y demás dispositivos inteligentes, ha contribuido en el desarrollo y utilización de bancos de pruebas o testbeds. El diseño, comportamiento, y nivel de interoperabilidad entre diferentes tecnologías, son varios criterios tomados en cuenta al momento de construir dichos testbeds. Además, la validación de algoritmos y métodos en la robótica se ve limitada por el uso únicamente de simulaciones lo cual no proporcionan el grado de experimentación como lo hace una plataforma física.

Por esto, actualmente existe una gran variedad de bancos de pruebas que integran un número variable de robots, redes de sensores, cámaras, tabletas e inclusive teléfonos inteligentes. Basados en [9], de acuerdo con sus componentes los bancos de pruebas pueden clasificarse en:

Bancos de pruebas no integrados. – Generalmente este tipo de banco de pruebas se construyen a partir de componentes y dispositivos de una misma tecnología y se subclasifican en:

- Banco de pruebas multirobot. – Son aquellos que están conformados por un conjunto de robots. Si la red de robots tiene las mismas características se

denominan homogéneos, en cambio si la red se está formada por robots de diferente tipo los bancos de pruebas se consideran heterogéneos, un ejemplo de este último se observa en la Figura 1.3.

Figura 1.3. Banco de pruebas multirobot heterogéneo (RAVEN). [9]

- Banco de pruebas con red de sensores inalámbricos. – También conocidos por sus siglas en inglés como Wireless Sensor Network (WSN). Además de ser bancos de pruebas homogéneos o heterogéneos se adaptan nodos WSN estáticos o transportados por el hombre.
- Banco de pruebas con red de cámaras. – Son usados generalmente para el procesamiento de imágenes en tareas específicas como: vigilancia, seguridad o métodos de localización.

Bancos de pruebas parcialmente integrados. – Permiten la interacción entre sistemas multirobot y redes de sensores, por tal motivo existen dos componentes tecnológicas diferentes, en donde una de ellas tiene el control sobre la otra; es decir, pueden ser bancos de pruebas multirobot con red de sensores en donde la comunicación entre las redes de sensores con los robots es unidireccional.

Bancos de pruebas altamente integrados. – Este tipo de banco de pruebas permite realizar experimentos usando redes de sensores y robots en donde la comunicación es bidireccional, su arquitectura es flexible y dispone de una interfaz de usuario. Por ejemplo, el banco de pruebas CONET desarrollado en la Universidad de Sevilla en el marco de la Unión Europea. El sistema está compuesto por cinco robots Pioneer 3AT, sistema de visualización mediante cámaras, redes de sensores y una interfaz de usuario para la ejecución de experimentos y visualización, con un área de trabajo de 500 m^2 (Figura 1.4).

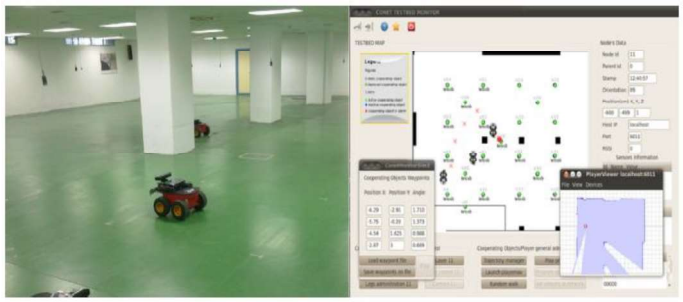


Figura 1.4. Banco de pruebas integrado CONET en la Universidad de Sevilla. [9]

De manera similar los bancos de pruebas pueden ser clasificados en base a otros criterios como: el rango de experimentos, la flexibilidad de su arquitectura y a quienes va dirigido (usuario objetivo).

Clasificación de bancos de pruebas según el rango de experimentos

Considerando el rango de experimentos o propósito de los bancos de pruebas se los clasifica en tres categorías:

- Aplicación. – Desarrollados con la finalidad de ser usados como herramientas experimentales en aplicaciones específicas, por ejemplo, aplicaciones de búsqueda y rescate, o monitoreo ambiental.
- Funcionalidad. – Pueden ser usados no únicamente para una sola aplicación, permiten mayor flexibilidad en la experimentación, por ejemplo, bancos de pruebas multirobot para la coordinación de agentes múltiples.
- Uso general. – Son aquellos bancos de pruebas diseñados para la realización de experimentos en diversas aplicaciones y funcionalidades, por ejemplo, los bancos de pruebas altamente integrados

Clasificación de bancos de pruebas según su arquitectura

La arquitectura de los bancos de pruebas es una de las consideraciones necesarias al momento de la experimentación. De acuerdo con su arquitectura pueden clasificarse en:

- Monolítica. – Por lo general tienen una arquitectura compacta para aplicaciones específicas, por ejemplo, los bancos de pruebas parcialmente integrados y no integrados.
- Modular. – Se encuentra presente en bancos de pruebas altamente integrados o de propósito general, en donde se puede realizar experimentos específicos utilizando un módulo central controlando a varios robots (centralizados),

utilizando varios módulos en experimentos de depuración y métodos sin necesidad de un solo módulo central (distribuido) o se puede combinar arquitectura centralizada y distribuida según la necesidad del usuario (flexible).

Clasificación de bancos de pruebas según el usuario objetivo

Pueden ser identificadas tres posibles categorías según los usuarios que realizarán la experimentación en los diferentes bancos de pruebas, así:

- Individual. – Los bancos de pruebas son usados exclusivamente por desarrolladores, por lo cual mucha de la información de estos es limitada para el público.
- Comunitario. – Los bancos de pruebas pueden ser usados por varios usuarios que no necesariamente tienen los mismos conocimientos, por lo cual es necesario la utilización de una interfaz, para facilitar la experimentación.
- Abierto. – Se incluyen todas las herramientas y mecanismos del banco de pruebas, así como documentación, interfaces estándar y posibles simulaciones para que sea usado por cualquier usuario.

A continuación, se muestran algunos ejemplos de bancos de pruebas que han sido desarrollados para aplicaciones específicas:

Duckietown

Se considera como un banco de pruebas altamente integrado. Mediante el uso de varios tipos de sensores y cámaras, se ejecutan cálculos que no pueden ser llevados a cabo por la naturaleza del robot y se enfoca en algoritmos de localización a través de técnicas de visión artificial. En la Figura 1.5 se muestra la mini ciudad creada por los desarrolladores de esta plataforma en el Instituto Tecnológico de Massachusetts [1].

Figura 1.5. Plataforma Duckietown. [1]

Este proyecto se enfoca en el diseño del sistema autónomo de Robo-Taxis dentro de la ciudad de Duckietown que a través de un procesador central provisto de una red de cámaras se puede acceder a su ubicación mediante cálculos realizados a través de marcadores de referencia matricial

Robotarium

Robotarium, además de ser un banco de pruebas altamente integrado, es de libre acceso es decir es un banco de pruebas abierto; sin embargo, se encuentra dirigido especialmente al ámbito de la educación e investigación. Robotarium posee una interfaz web pública que permite la comprobación de algoritmos de control para un grupo de robots móviles, un ejemplo se observa en la Figura 1.6. [2]

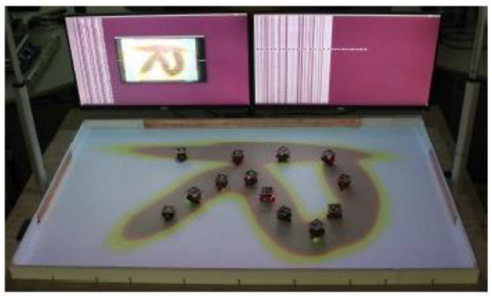


Figura 1.6. Representación de control aplicado en Robotarium. [2]

MICROMVP

Es una plataforma de pruebas destinada a la investigación y educación, se considera como un banco de pruebas altamente integrado, multirobot homogéneo y comunitario. Permite la experimentación con robots móviles en tareas de coordinación y planeación de rutas. Se encuentra conformado por: seis robots móviles (micro vehicles) de tamaño reducido con una masa de 100 gramos cada robot, sistema de monitoreo y sistema de control separados; esto quiere decir que se utilizan dos computadoras como se observa en la Figura 1.7. [10]

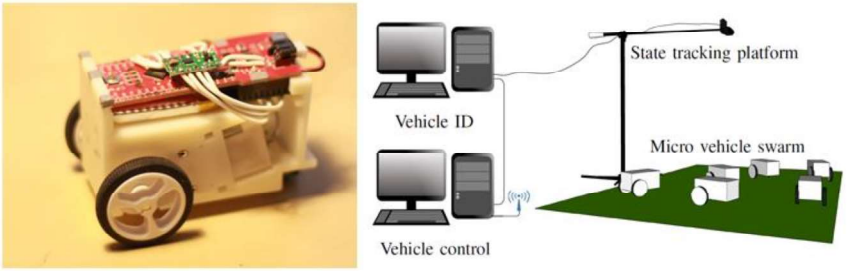


Figura 1.7. (Izquierda) Micro Vehicle ensamblado, (Derecha) arquitectura del banco de pruebas MICROMVP. [10]

Para el proyecto planteado en base a la clasificación mencionada en la sección de Banco de pruebas, se eligió un banco de pruebas multi-robot integrado con una cámara y comunicación inalámbrica entre los robots e interfaz gráfica. El banco de pruebas es de libre acceso al usuario y orientando a fines de educación e investigación.

De esta forma una vez seleccionado el banco de pruebas y el tipo de robots que lo conforman es necesario conocer los conceptos generales para poder controlar el funcionamiento del banco de pruebas implementado.

Para empezar, es necesario, entender el modelo matemático de los robots escogidos, posteriormente con esa información es posible desarrollar la técnica de control escogida, y por último la vinculación del sensor de visión y la comunicación inalámbrica. A continuación, se detalla el modelo matemático del tipo de robot seleccionado.

Modelo matemático del robot uniciclo

El modelo matemático del robot tipo uniciclo ha sido utilizado para describir el movimiento de los robots que tiene la capacidad de girar sobre su propio eje; sin embargo, en muchas ocasiones existe confusión al emplear los términos "robot diferencial" y "robot uniciclo", por lo cual se realiza la correspondiente explicación, la misma que se encuentra detallada en [11], y que nos servirá para conocer cómo se relaciona el modelo matemático del robot uniciclo con el robot diferencial.

El modelo del robot uniciclo, representado en la Figura 1.8, tiene tres estados que definen su postura. Estos son:

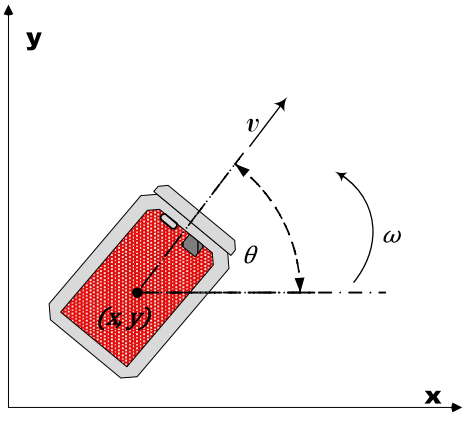


Figura 1.8. Robot tipo uniciclo.

- x: Posición en el eje x.
- y: Posición en el eje y.
- θ: Ángulo del robot en sentido antihorario respecto al eje x.

De igual manera el robot uniciclo presenta dos entradas que afectan a los estados definidos anteriormente, estas son:

- v: Velocidad lineal.
- ω: Velocidad angular.

Es así como el modelo cinemático del robot uniciclo queda representado por la siguiente ecuación.

$$\begin{cases} \dot{x} = v\,cos(\theta) \\ \dot{y} = v\,sin(\theta) \\ \dot{\theta} = \omega \end{cases}$$

Ecuación 1.1. Modelo matemático robot tipo uniciclo

Debido a la forma sencilla e intuitiva del modelo del robot uniciclo, este es usado para describir el movimiento de los robots móviles con ruedas, el robot diferencial no es la excepción. Para relacionar al diseño de robot diferencial seleccionado para nuestros prototipos se debe convertir las entradas del robot uniciclo en entradas de accionamiento diferencial como se muestra en la Figura 1.9. Se considera que las entradas del robot diferencial son:

- Ω_L**:** Velocidad angular de la rueda izquierda.
- Ω_R**:** Velocidad angular de la rueda derecha.

De la misma forma se debe conocer ciertos parámetros para que el modelo matemático del robot uniciclo sea aplicado al robot diferencial, así:

- R**:** Radio de las ruedas.
- L**:** Distancia entre las dos ruedas (eje).

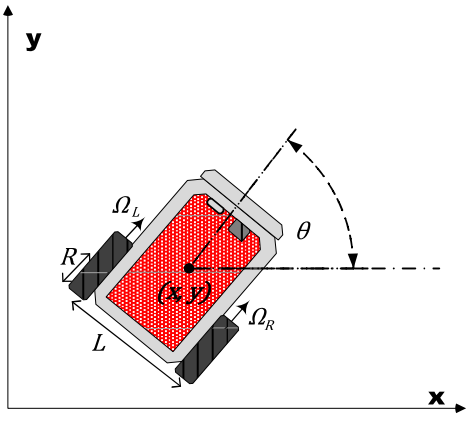


Figura 1.9. Robot móvil diferencial.

De esta manera se pueden obtener las Ecuaciones 1.2 y 1.3 que relacionan la velocidad lineal y angular del robot uniciclo con las velocidades angulares de las ruedas del robot diferencial. Las expresiones obtenidas son las siguientes:

$$v = \frac{R(\Omega_R + \Omega_L)}{2}$$

Ecuación 1.2. Velocidad lineal aplicada al robot diferencial.

$$\omega = \frac{R(\Omega_R - \Omega_L)}{L}$$

Ecuación 1.3. Velocidad angular aplicada al robot diferencial.

Una vez conocido el modelo matemático del robot uniciclo, es necesario identificar las características de movimiento. En robótica se usa la siguiente clasificación para denotar las características de la movilidad de los robots [12]:

- ***Robot holonómico:*** Son los robots que permiten controlar todos sus grados de libertad es decir cuando la cantidad de grados de libertad totales y controlables es la misma.
- ***Robot no holonómico:*** Son los robots que poseen una cantidad de grados de libertad controlables menor a la cantidad total de grados de libertad del robot.

Una restricción no holonómica ocurre cuando el robot tiene restricción en su movimiento. Basados en la mecánica del robot diferencial los movimientos permitidos son hacia adelante o hacia atrás, y no movimientos laterales directos como se indica en la Figura 1.10; por consiguiente, los robots implementados presentan una restricción no holonómica.

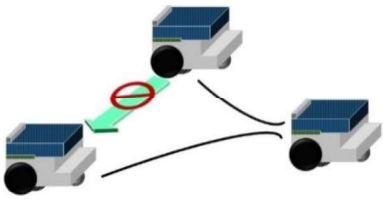


Figura 1.10. Representación de restricción no holonómica del robot uniciclo.

A partir del modelo matemático del robot uniciclo representado por la Ecuación 1.1 y su restricción de movimiento, se realizan algunas variantes en el modelo general para finalmente obtener el modelo cinemático que representa a los prototipos de los robots móviles que serán usados en nuestro banco de pruebas. A continuación, se indica a detalle el modelo cinemático aplicado a los prototipos de robots móviles del presente proyecto.

Modelo cinemático con restricción no holonómica mejorada

En nuestro proyecto se considera únicamente el modelo cinemático del robot uniciclo. Basados en [13] este modelo se lo puede aplicar cuando la carga del robot respecto a su estructura es despreciable y los movimientos de traslación ocurren a bajas velocidades. Por otro lado, si consideramos la restricción de movimiento lateral del robot ocurre cuando el punto de interés está ubicado en el centro del eje de las ruedas, por lo cual es necesario desplazarlo una distancia a, como se indica en la Figura 1.11.

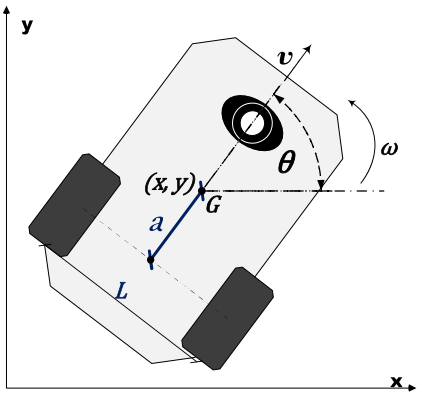


Figura 1.11. Representación del modelo cinemático del robot móvil.

El punto de interés (x, y) permite establecer la posición actual del robot asumiendo que se encuentra ubicado en el centro de gravedad G del robot. Esta consideración se conoce como restricción no holonómica mejorada [14] y la representación matemática del modelo cinemático viene dado por:

$$\begin{cases} \dot{x} = v\,cos(\theta) - a\omega\,sin(\theta) \\ \dot{y} = v\,sin(\theta) + a\omega\,cos(\theta) \\ \dot{\theta} = \omega \end{cases}$$

Ecuación 1.4. Modelo cinemático con restricción no holonómica mejorada.

Para obtener las ecuaciones que determinan el movimiento de los robots utilizados, se reemplazan las Ecuaciones 1.2 y 1.3 en la Ecuación 1.4, consiguiendo que las entradas de nuestro sistema sean las velocidades angulares de las ruedas derecha e izquierda, obteniendo:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \frac{R}{2}cos(\theta) - \frac{a\,R}{L}sen(\theta) & \frac{R}{2}cos(\theta) + \frac{a\,R}{L}sen(\theta) \\ \frac{R}{2}sen(\theta) + \frac{a\,R}{L}cos(\theta) & \frac{R}{2}sen(\theta) - \frac{a\,R}{L}cos(\theta) \\ \frac{R}{L} & \frac{-R}{L} \end{bmatrix} \begin{bmatrix} \Omega_R \\ \Omega_L \end{bmatrix}$$

Ecuación 1.5. Modelo cinemático con restricción no holonómica mejorada para el robot diferencial.

Luego de conocer el modelo cinemático de los robots, a continuación, se indican las principales problemáticas de control en robots móviles como: control de posición, seguimiento de caminos y seguimiento de trayectorias, de igual manera se resumen algunas técnicas de control con la finalidad de elegir una de ellas para implementarla en el presente proyecto.

Control de robots móviles

Si bien es cierto el objetivo principal del proyecto es el control de postura y seguimiento de trayectorias, es necesario establecer las principales problemáticas de control en robots móviles [15].

Control de posición

Consiste en ubicar al robot en una posición de referencia, como se muestra en la Figura 1.12. El objetivo del control de posición es lograr que el robot se mueva desde la posición inicial en el punto A hasta la posición final en el punto B, pero no necesariamente que alcance una orientación final deseada.

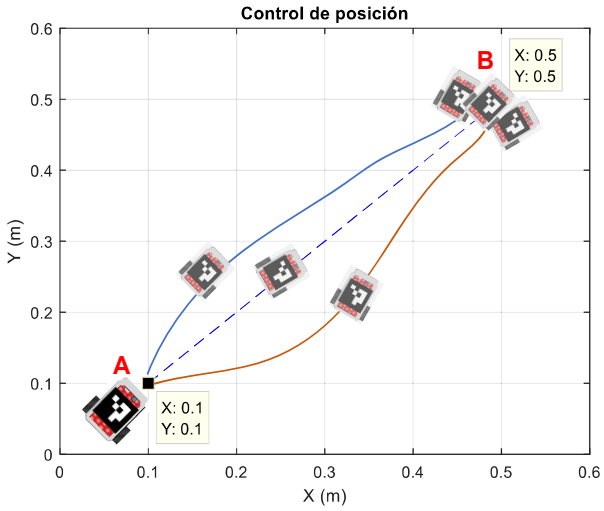


Figura 1.12. Control de posición, desde el punto A hasta el punto B.

Seguimiento de caminos

Se considera como camino a un conjunto de puntos localizados en el espacio. De la misma forma un camino puede tomarse a partir de la discretización de una curva continua en donde se consideran algunos puntos de esta. El objetivo del controlador consiste en que el robot móvil siga el camino o puntos ordenados independientemente del tiempo y de la curva que describe el robot para alcanzar estos puntos. Por ejemplo,

en la Figura 1.13 se muestran dos posibles casos del movimiento del robot al momento de seguir el camino propuesto y alcanzar la posición final.

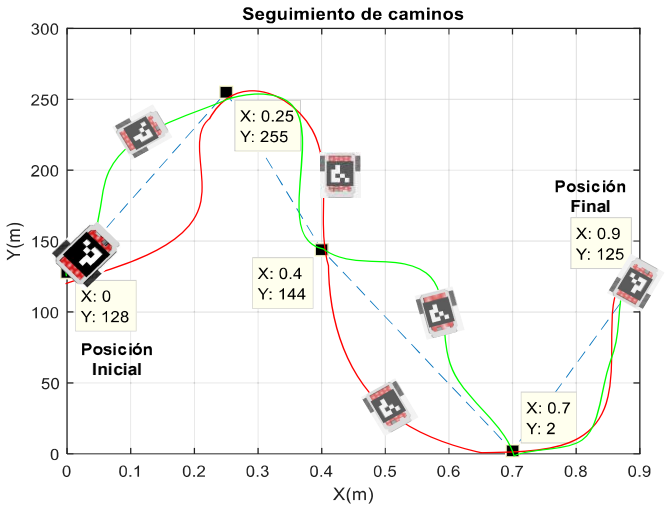


Figura 1.13. Seguimiento de caminos.

Seguimiento de trayectorias

La trayectoria es un camino parametrizado en el tiempo, el control de seguimiento de trayectoria consiste en que el robot móvil alcance y siga los estados deseados variantes con el tiempo que describen la trayectoria establecida. En la Figura 1.14. se muestra el seguimiento de una trayectoria lineal, se puede observar que a pesar de que el robot móvil no inicie en la posición inicial de la trayectoria, éste intenta alcanzar y seguir la trayectoria hasta llegar a la posición final.

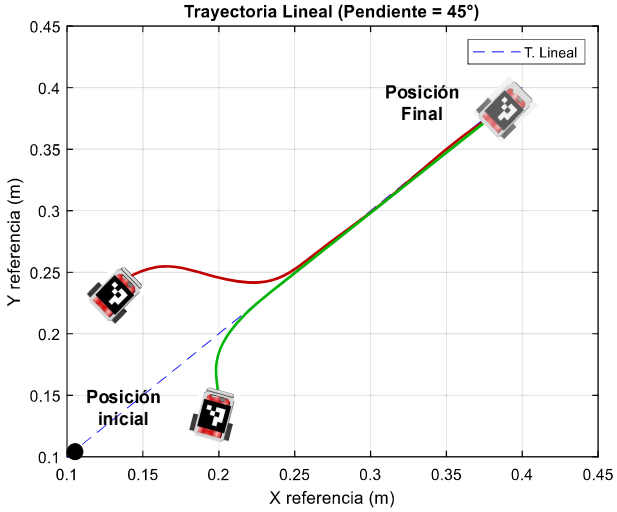


Figura 1.14. Seguimiento de trayectoria.

Métodos de control

Tanto para el control de posición y seguimiento de trayectoria es necesario aplicar la teoría de control para lograr que los robots móviles tengan un comportamiento deseado. Los métodos de control permiten alcanzar los valores deseados de las variables a controlar. Un lazo de realimentación del sistema es indispensable al momento de implementar un controlador, en la Figura 1.15 se muestran los elementos de un lazo de control aplicado a cualquier sistema a ser controlado [16].

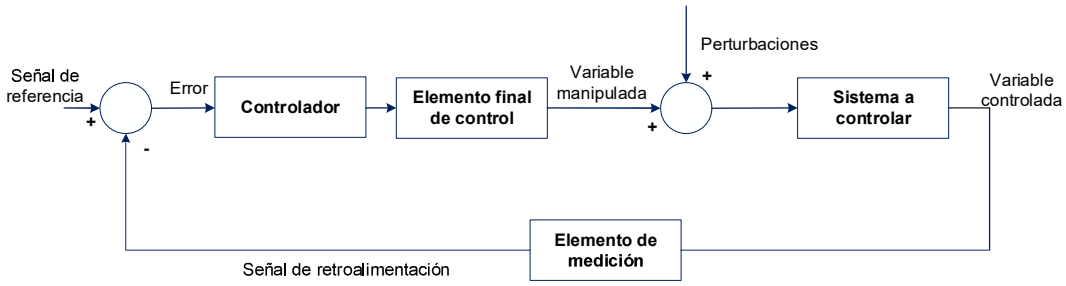


Figura 1.15. Diagrama de bloques de un sistema realimentado.

Las técnicas de control que se revisan a continuación tienen como objetivo el cálculo de las velocidades lineal y angular del robot móvil para seguir una trayectoria de referencia. Inicialmente el robot se encuentra en una posición actual (X_m, Y_m) con un ángulo de orientación θ_m, el móvil deberá pasar del estado actual al estado de referencia deseado (X_{ref_m}, Y_{ref_m}) en un instante t_m para posteriormente alcanzar la referencia en un estado ($X_{ref_{m+1}}, Y_{ref_{m+1}}$) para un tiempo t_{m+1} [14], como se muestra en la Figura 1.16.

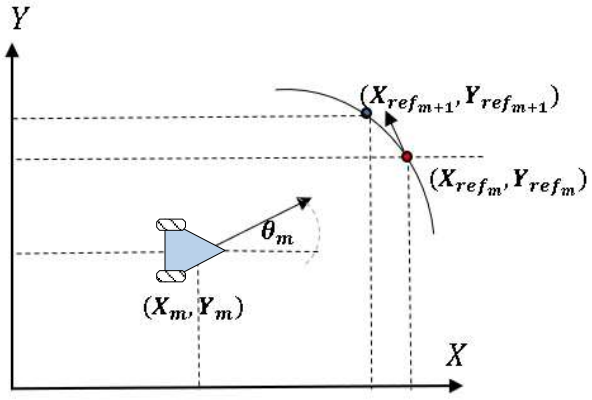


Figura 1.16. Representación del seguimiento de trayectoria del robot móvil.

Diferentes técnicas de control han sido implementadas para resolver las problemáticas anteriormente expuestas, algunas de estas técnicas se indican a continuación.

Controlador tipo PID

Este tipo de controlador es utilizado generalmente en procesos industriales, sin embargo, ha presentado buenos resultados en aplicaciones de robótica móvil como en [15] en donde se implementa el controlador PID utilizando la herramienta Simulink de Matlab para el control de posición y seguimiento de trayectorias para un manipulador móvil. El controlador PID puede ser representado por la siguiente ecuación [16]:

$$u_{PID}(t) = k\left(e(t) + \frac{1}{T_i}\int_0^t e(\tau)d_\tau + T_d\frac{de(t)}{dt}\right)$$

Ecuación 1.6. Controlador tipo PID.

Donde $u_{PID}(t)$ es la señal de control y $e(t)$ es el error. Los parámetros de calibración del controlador vienen dados por la ganancia proporcional k, integral T_i y derivativa T_d. Las acciones del controlador se obtienen variando los parámetros antes mencionados.

- Acción proporcional: Consiste en tomar el valor de error y multiplicarlo por una constante para obtener la salida del control, se usa cuando el sistema realimentado presenta error en estado estable.
- Acción integral: Se aplica cuando existe un error en estado estable, a diferencia de la acción proporcional logra eliminar el error, integrando el error en el tiempo para alcanzar la referencia.
- Acción diferencial: Es usada cuando los cambios en la referencia o en el error son lentas. No se puede implementar solo un controlador diferencial ya que los actuadores no pueden generar señales impulso, por lo cual la acción diferencial es acompañada de una acción proporcional.

Controlador SMC

El controlador por modo deslizante o SMC por sus siglas en inglés (Sliding Mode Control), puede aplicarse en procesos lineales y no lineales. En robótica móvil el controlador SMC ha sido implementado para el seguimiento de trayectorias de un robot móvil Pioneer 3DX mediante el uso de la plataforma robótica ROS como en [14]. Este tipo de controlador se caracteriza por ser una técnica robusta y para el desarrollo del controlador es necesario definir una superficie deslizante $S(t)$ por la cual el sistema pueda deslizarse hacia su valor deseado.

Cuando la variable controlada es igual a la referencia, $S(t)$ toma un valor constante por lo cual se debe cumplir la condición $\frac{dS(t)}{dt} = 0$. La señal de control $U_{SMC}(t)$ del controlador SMC representada por la Ecuación 1.7 se conforma de dos partes una continua $U_{C_{SMC}}(t)$

que mantiene a la variable controlada en la superficie deslizante, y una discontinua $U_{D_{SMC}}(t)$ que permite que la variable alcance la superficie deslizante [16].

$$U_{SMC}(t) = U_{C_{SMC}}(t) + U_{D_{SMC}}(t)$$

Ecuación 1.7. Controlador SMC.

La parte discontinua del controlador además emplea la función sigma (Ecuación 1.8) con la finalidad de suavizar la discontinuidad de esta parte.

$$U_{D_{SMC}}(t) = K_D \frac{S(t)}{|S(t)| + \delta}$$

Ecuación 1.8. Función sigma.

Donde K_D es la ganancia que permite el acercamiento de la variable controlada hacia la superficie deslizante y δ reduce las oscilaciones de alta frecuencia no deseadas.

Controlador basado en álgebra lineal

También conocido como controlador basado en métodos numéricos, es aplicable en sistemas no lineales que pueden ser representados mediante ecuaciones diferenciales, cuya solución no es exacta. Este tipo de control permite expresar a los sistemas de orden n en sistemas de ecuaciones de primer orden. Para el diseño del algoritmo de control es necesario aplicar algún método numérico que permita calcular el siguiente estado de las variables a ser controladas del sistema. En el presente proyecto se usó la aproximación de Euler; esta aproximación nos permite expresar un sistema continuo en tiempo discreto. Como se indica en la Figura 1.17 para obtener el siguiente estado de la variable $x(t)$ se debe integrar su derivada $\dot{x}(t)$ desde t_m hasta t_{m+1}. [14]

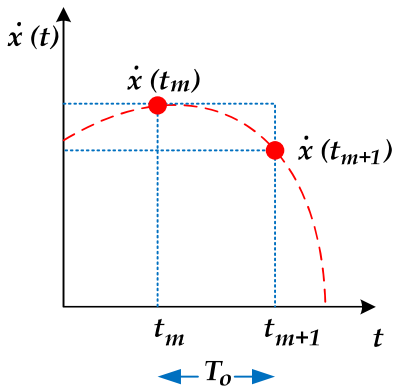


Figura 1.17. Representación de la aproximación de Euler.

Mediante el reemplazo de la integral por una diferencia finita hacia adelante, la aproximación de Euler permite obtener la siguiente expresión:

$$\Delta x(t) = \int_{t_m}^{t_{m+1}} \dot{x}(t)dx \approx \dot{x}(t_m)(t_{m+1} - t_m)$$

Ecuación 1.9. Aproximación de Euler en tiempo continuo.

De la consideración anterior se puede establecer la siguiente relación entre el estado actual y el siguiente estado:

$$X_{m+1} = X_m + T_0 X_m$$

Ecuación 1.10. Aproximación de Euler en tiempo discreto.

Esta aproximación sirve como base para el diseño del controlador (sección de Implementación de controlador) que se implementó en el presente proyecto.

Uno de los aspectos fundamentales al implementar un controlador como se indicó anteriormente es disponer de un lazo de realimentación, en nuestro caso se lo realiza mediante el sistema de visión. En las siguientes secciones se revisa los fundamentos que permitieron consolidar el sistema de monitoreo del banco de pruebas implementado.

Visión por computadora

Se podría interpretar que la tarea de visión es sumamente fácil, ya que como seres humanos nos es fácil encontrar un objeto dentro de un plano, pero en realidad el mecanismo cerebral divide la señal de visión en varios canales los cuales transmiten diferente tipo de información al cerebro. Éste identifica las partes más importantes de una imagen mientras que suprime lo menos importante. En la Figura 1.18 se muestra el mecanismo de visión humana [17]. Por otro lado, la visión por computadora además de la identificación de un objeto en el plano reconstruye las propiedades de una imagen, como por ejemplo forma, iluminación, distribución de colores, decodificación, entre otras.

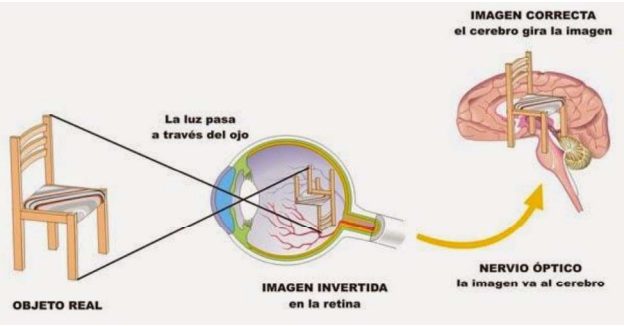


Figura 1.18. Representación de mecanismo de visión humana [17]

La visión por computadora es un campo de investigación en crecimiento dedicado al análisis, interpretación y modificación de alto nivel de las imágenes [18]. Existe un

amplio campo de aplicaciones, como áreas de vigilancia, tráfico vehicular, ventas, medicina, realidad aumentada, robótica, producción de películas, entre otros. La visión por computadora es una transformación de datos de una cámara estática o móvil. Este tipo de transformaciones se hacen para cumplir con un objetivo específico como conocer la posición de la cámara móvil, la ubicación de personas u objetos dentro de un marco, o convertir una imagen de color a escala de grises.Conforme este tipo de tecnología avanza la demanda de aplicaciones de visión por computadora también crece, la tecnología moderna de las cámaras permite el enfoque automático de las caras de las personas, reconocimiento de texto óptico, sistemas de parqueo inteligente, entre otras.

El proceso de visión por computadora envuelve un gran número de subprocesos para el análisis de la información. A continuación, se da un breve detalle de las técnicas de visión por computadora.

Técnicas de visión por computadora

El desarrollo más notable de la visión por computadora en la actualidad quizás es la interacción con los gráficos de computadora. El objetivo de manipular imágenes del mundo real para realizar su respectiva transformación en animaciones por computadora ha creado un alto interés entre los desarrolladores, pero para lograr este objetivo es necesario conocer el origen de esta transformación. Basados en [19,20], las técnicas de visión por computadora que existen son las siguientes.

- *Formación de imagen.* – Dado un conjunto de condiciones de iluminación, geometría de la escena y propiedades ópticas de la cámara es posible desarrollar la formación de la imagen. Esta incluye varios métodos como la formación de imágenes geométricas a través de puntos, líneas y planos. El análisis se realiza mediante el mapeo de las imágenes usando geometría proyectiva como se indica en la Figura 1.19.

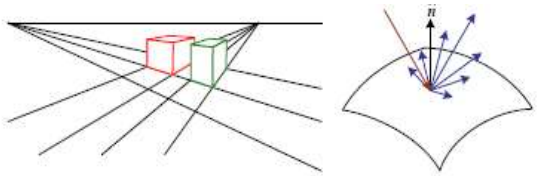


Figura 1.19. (a)Perspectiva de proyección (b) Dispersión de la luz sobre superficie [19]

- *Procesamiento de imagen.* – Esta técnica es fundamental en las aplicaciones de visión artificial, permite decodificar la imagen para su posterior análisis. Se incluyen temas de filtrado de imágenes, conversión de colores, entre otras. En

la Figura 1.20. se indica el procesamiento de la imagen a través de técnicas de conversión de mapa de bits.

Figura 1.20. Procesamiento de imagen [19]

- *Extracción y emparejamiento de características de imagen.* – En esta técnica se realiza la reconstrucción y reconocimiento de características de un entorno tridimensional. La Figura 1.21 indica un paisaje donde se extrae la información de bordes y puntos clave de la imagen.

Figura 1.21. Procesamiento de bordes de una imagen [19]

- *Segmentación.* – Realiza la división de una imagen en varios grupos de pixeles, intentando extraer la información más importante y la simplifica de manera que su procesamiento es más rápido y fácil de realizar. En la Figura 1.22. se observa la imagen original de una persona y a través de la segmentación se obtiene la imagen con la información más importante de la imagen.

Figura 1.22. Procesamiento de segmentación de una imagen [19]

- *Realidad Aumentada.* - Combina el mundo real con el mundo virtual, este tipo de tecnología está conformado por un dispositivo de visualización y un software de rastreo de movimiento [21]. En la figura 1.23 se observa una aplicación de realidad aumentada al enfocar un billete de dólar, y en el celular se observa una edificación.

Figura 1.23. Realidad aumentada en dispositivo inteligente [22]

- *Alineación.* – Esta técnica permite analizar las características de movimiento extraídas de patrones geométricos, con el fin de obtener características extrínsecas e intrínsecas de la cámara. Un claro ejemplo se observa en la Figura 1.24 donde el usuario enfoca el patrón de calibración en diferentes orientaciones.

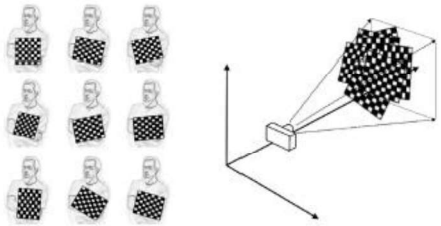


Figura 1.24. Procesamiento de calibración de cámara [18]

Las técnicas de visión por computadora mencionadas pueden ser desarrolladas en diferentes entornos de programación como OpenCV con la ayuda de Microsoft Visual Studio 2015 a continuación, una breve reseña de los programas mencionados.

Open CV

Open Source Computer Vision Library, es un conjunto de librerías diseñadas para aplicaciones de visión por computadora y con enfoque en aplicaciones de tiempo real. Es un software de libre acceso bajo la licencia de BSD además de estar disponible en varias interfaces como C++, Phyton, Mac OS, iOs y Android. Sus aplicaciones van desde los aspectos artísticos hasta aplicaciones en robótica avanzada como se observa en la Figura 1.25. [23]

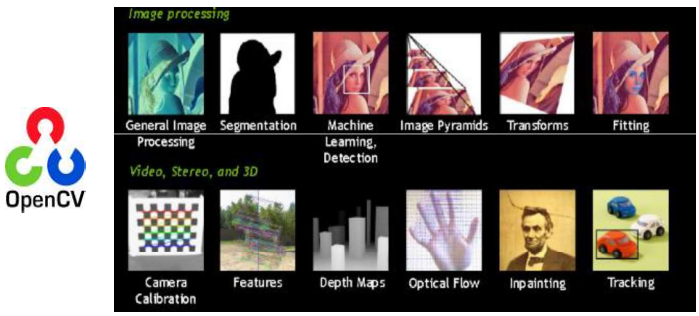


Figura 1.25. Logotipo de Open CV [23]; Aplicaciones de OpenCV [24].

La biblioteca consta de más de 2500 algoritmos optimizados, que permiten desarrollar aplicaciones de visión artificial. Estos algoritmos básicamente ayudan a detectar y reconocer rostros, objetos, movimientos, modelos 3D de objetos, etc.

Visual Studio 2015

Entorno de desarrollo integrado (IDE) es un conjunto de herramientas para desarrollo de software desde simples aplicaciones hasta sistemas complejos industriales [25]. Este entorno permite la depuración, verificación de errores, rendimiento de código y pruebas de diseños. Visual Studio 2015 es compatible con C#, C y C++, JavaScript, F # y Visual Basic. Para el proyecto planteado, se utilizó C++ como lenguaje de programación. Este lenguaje es de fácil utilización además de ser un lenguaje estandarizado ISO, permite compilar directamente en el código nativo de la máquina. [26]

A partir de la vinculación de OpenCV con Visual Studio 2015, es posible el manejo del entorno de OpenCV para generar los marcadores de referencia matricial. Como se indicó OpenCV permite realizar técnicas de visión avanzadas por lo que el usuario puede calibrar cámaras digitales, rastrear objetos determinados, realizar análisis de gráficos entre otras aplicaciones. A continuación, se especifica en que consiste el proceso de calibración de la cámara mediante OpenCV.

Calibración de cámara

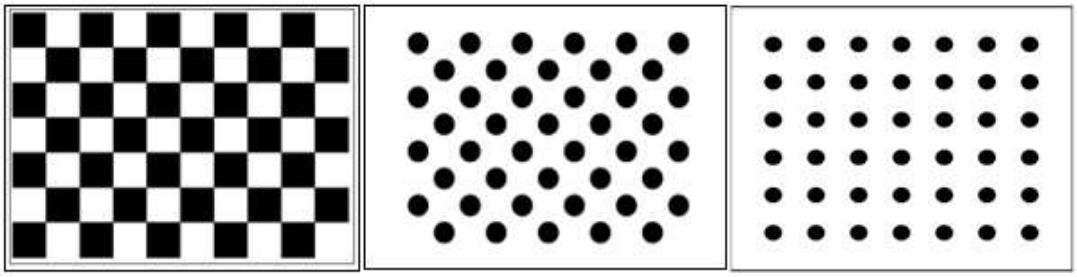
Mediante el proceso de calibración se obtiene parámetros fundamentales de la cámara. OpenCV permite realizar la calibración de la cámara mediante diferentes comandos basados en los denominados algoritmos de Zhang [27]. Estos se fundamentan en dos técnicas de calibración clásicas:

- *Calibración fotogramétrica.* - Se basa en la observación de un objeto de calibración en el espacio 3D. Este objeto consta de dos a tres planos ortogonales entre sí.
- *Auto – calibración.* – Está técnica no utiliza un objeto de calibración, utiliza la información proporcionada por el movimiento de la cámara en una zona estática. Al tomar varias imágenes sobre patrones permanentes del entorno, se puede realizar la correspondencia de parámetros internos y externos de la cámara.

A partir del uso de estas dos técnicas de calibración, el proceso de calibración de Zhang necesita que la cámara observe un patrón como los que se presentan en la Figura 1.26. Generalmente se tienen los siguientes patrones [28]:

- *Tablero de ajedrez.* - La matemática que utiliza la calibración basada en el tablero es mucho más simple que la del patrón circular. La detección de las uniones del patrón es más fácil al momento de la calibración.
- *Patrón circular asimétrico y simétrico.* - Este patrón puede ser más preciso que el tablero de ajedrez, pero su precisión depende de una selección correcta del tamaño de los círculos. Por esto a este tipo de patrón se utiliza en fotogrametría industrial.

Estos patrones deben estar sobre una superficie plana y rígida como una tabla, o una cubierta de un cuaderno. El usuario debe realizar varias instantáneas del patrón, el número mínimo son dos capturas, pero existe presencia de ruido en las imágenes por lo que es aconsejable que el número de muestras sea mínimo 10 instantáneas.



(a) (b) (c)

Figura 1.26. (a) Tablero de ajedrez (b) Patrón asimétrico circular (c) Patrón simétrico circular

El principal resultado de calibración es el modelo de la cámara el cual contiene sus parámetros internos y externos.

Modelo de la cámara

Basados en [29], sus parámetros son:

- *Internos.* - Son la descripción óptica interna y las características de la geometría de la cámara como lo son centro de la imagen (cx, cy), longitud focal (fx, fy) distorsión de los lentes $(k1, k2, k3, p1, p2)$.
- *Externos.* – Indican la posición de cámara en sistemas de coordenadas arbitrarios. Son la posición tridimensional y la orientación del sistema de coordenadas conformados por los parámetros de rotación (rx, ry, rz) y traslación

(tx, ty, tz) en comparación con el sistema de coordenadas tridimensional (X, Y, Z).

Un punto 3D (X, Y, Z) puede ser proyectado en pixeles considerando como punto de referencia la esquina superior izquierda de la imagen digital, como se indican en las Ecuaciones 1.11 y 1.12.

$$u = X\frac{fx}{Z} + cx$$

Ecuación 1.11. Coordenada en eje x expresada en pixeles

$$v = Y\frac{fy}{Z} + cy$$

Ecuación 1.12. Coordenada eje y expresada en pixeles

Con el objetivo de eliminar distorsión del lente y mejorar la precisión de las coordenadas es necesario obtener el modelo de la cámara ya que permite traducir las coordenadas del mundo exterior X, Y, Z al sistema de coordenadas fijo con respecto a la cámara, a estas coordenadas se las conocemos como coordenadas de homografía.

Para obtener el modelo de la cámara es necesario expresar los parámetros de rotación como una matriz de 3x3, esta matriz se forma a partir de la aproximación de Rodrigues [30]. Internamente OpenCV coloca los parámetros de rotación en un vector y utiliza el algoritmo de Rodrigues para obtener una matriz de 3x3 conocida como matriz de rotación (R). El modelo completo de la cámara incluye tanto la matriz de rotación como los parámetros de traslación.

El conjunto de estos parámetros puede ser representados como una matriz de 4x4 $[R|\mathrm{t}]$, como se indican en la Ecuación 1.13. La matriz de rotación-traslación $[R|\mathrm{t}]$ permitirá describir el movimiento de la cámara alrededor de algo estático, o el movimiento del objeto alrededor de una cámara fija.

$$[R|\mathrm{t}] = \begin{bmatrix} r_1 & r_2 & r_3 & tx \\ r_4 & r_5 & r_6 & ty \\ r_7 & r_8 & r_9 & tz \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Ecuación 1.13. Matriz de Rotación-Traslación.

Los parámetros $r_1, r_2, r_3, r_4, r_5, r_6, r_7, r_8, r_9$ son calculados internamente en OpenCV mediante la utilización del algoritmo de Rodrigues.

Al obtener estos parámetros es posible obtener el modelo completo de la cámara como se indica en la Ecuación 1.12, el cual permite obtener la relación que existe entre las unidades de la cámara (pixeles) y las unidades del mundo real [30].

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} fx & 0 & cx & 0 \\ 0 & fy & cy & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} [R|\mathrm{t}] \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Ecuación 1.14. Modelo de la cámara

Donde:

- $u, v.$ - Coordenadas del sistema de homografía de la cámara.

Parámetros internos:

- $fx, fy.$ - Longitud focal expresada en pixeles.
- $cx, cy.$ - Punto principal generalmente en el centro de imagen en pixeles.

Parámetros externos:

- $X, Y, Z.$ - Coordenadas 3D del espacio en milímetros.
- $[R|t].$ - Matriz de rotación-traslación.

Al conocer la relación de las coordenadas de la cámara y la imagen es posible determinar las coordenadas de homografía de la cámara u, v, la cuales son una transformación proyectiva de la imagen en el mundo real y la imagen capturada por la cámara [31].

La pose estimada del objeto viene dada en coordenadas 2D expresadas en unidades de pixeles. El límite de este plano está delimitado por la resolución de la cámara, es decir si se configura una cámara con una resolución de 640x480 el plano está establecido en el eje x por 640 pixeles y en el eje y por 480 pixeles.

Como se indica en la Figura 1.27, estas coordenadas permiten utilizar la cámara en cualquier posición ya que utilizan la relación de inclinación de la cámara para estimar la pose del objeto capturado, esta relación viene dada por los coeficientes de distorsión tangencial de la cámara.

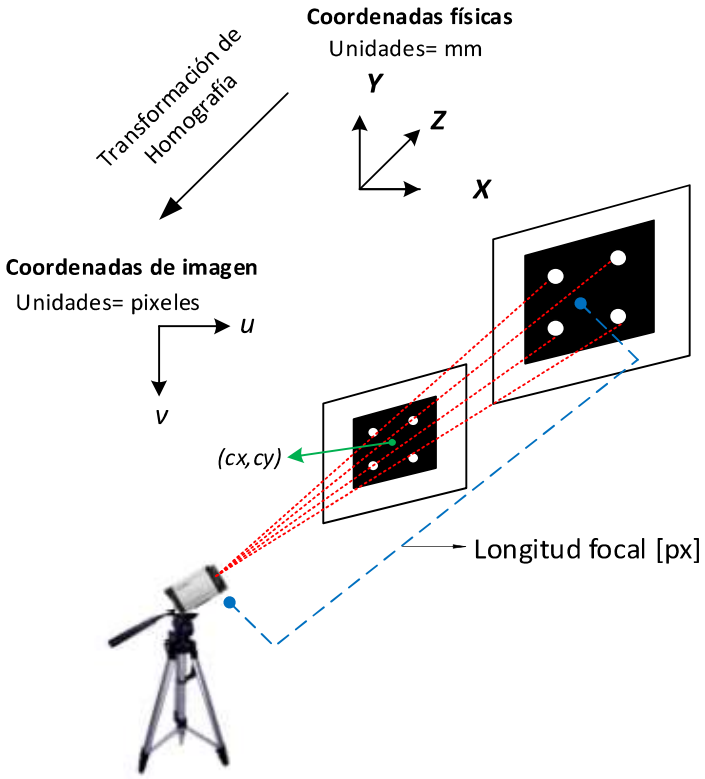


Figura 1.27. Estructura de sistema de coordenadas.

Los lentes de las cámaras digitales que no han sido calibradas poseen un grado de distorsión significativo. Existe distorsión tangencial y radial. En la Figura 1.28, se puede observar cómo afecta la captura de una imagen con distorsión y sin distorsión. La distorsión radial es la que provoca el ojo de pez en las cámaras, mientras que la distorsión tangencial se produce porque el lente de la cámara no es completamente paralelo a la imagen capturada.

Figura 1.28. Tipos de Distorsión en la imagen. [32]

Al calibrar la cámara se obtiene el vector de coeficientes de distorsión indicados en la Ecuación 1.15.

$$coef\ Distorción = [k1 \quad k2 \quad p1 \quad p2 \quad k3]$$

Ecuación 1.15. Elementos de vector de distorsión.

Donde:

$k1, k2, k3.$ - Coeficientes distorsión radial mejoran las curvas en los bordes de la imagen.

$p1, p2.$ - Coeficientes de distorsión tangencial indican la inclinación del sensor.

Estos coeficientes permiten mejorar la precisión de las coordenadas de homografía, por consiguiente, se puede establecer la mejora de las coordenadas de la siguiente manera [32].

- Mejora para distorsión radial:

$$u_{corregida} = u(1 + k1r^2 + k2r^4 + k3r^6)$$

Ecuación 1.16. Corrección de distorsión radial en componente u

$$v_{corregida} = v(1 + k1r^2 + k2r^4 + k3r^6)$$

Ecuación 1.17. Corrección de distorsión radial en componente v

- Mejora para distorsión tangencial:

$$u_{corregida} = u + ((2 * p1 * uv) + p2(r^2 + 2u^2)$$

Ecuación 1.18. Corrección de distorsión tangencial en componente u

$$v_{corregida} = v + (p1(r^2 + 2u^2) + (2 * p2 * uv))$$

Ecuación 1.19. Corrección de distorsión tangencial en componente v

Los coeficientes de distorsión son propios de la cámara independientemente de la resolución configurada.

Al realizar la calibración de la cámara a través de OpenCV se deben tomar en cuenta los siguientes pasos [33]:

1. Lectura de imágenes de patrón de calibración guardadas por el usuario.
2. Detección de uniones de las figuras del patrón escogido.

3. Aplicación del algoritmo para calcular parámetros internos y externos de la cámara

4. Retorno de matriz de calibración y vector de coeficientes de distorsión.

Este proceso se lo realiza a través de Visual Studio en conjunto con Open CV. Una vez calibrada la cámara es posible identificar figuras en el espacio exterior, en el caso del presente proyecto estos son marcadores de referencia. A continuación, se especifican los diferentes tipos de patrones que han sido desarrollados en aplicaciones de visión por computadora.

Tipos de marcadores de referencia

Se han desarrollado diferentes tipos de códigos utilizados tanto en la industria como en investigación. La tecnología de código de barras proporciona un control de gestión y actualización de información [34]. Este tipo de códigos son sistemas de barras impresas una tras otra como se observa en la Figura 1.29 (a). El espacio entre barras refleja la luz al escáner y son un comando de caracteres definidos. Una de las aplicaciones más comunes del código de barras es el control de inventarios de productos.

(a) (b)

Figura 1.29. (a) Código de Barras [35] (b) Código QR [36]

La tecnología de códigos QR (Quick Response) es un tipo de código bidimensional que está diseñado para ser analizado por teléfonos inteligentes [36].

El contenido que despliega al ser leído por la cámara es de rápido acceso. El código está compuesto por módulos de color negro dentro de un pequeño espacio cuadrado de fondo blanco como se observa en la Figura 1.29 (b). El enfoque de estos códigos redirecciona al usuario a un URL, textos o descarga de aplicaciones. La aplicación más destacada es el marketing de productos.

Actualmente el tipo de patrones desarrollados son los marcadores de referencia matricial utilizados, por ejemplo, en la tecnología de realidad aumentada (AR). Como se observa en la Figura 1.30, existen varios tipos de marcadores de referencia; a continuación, se describen las características de algunos de estos.

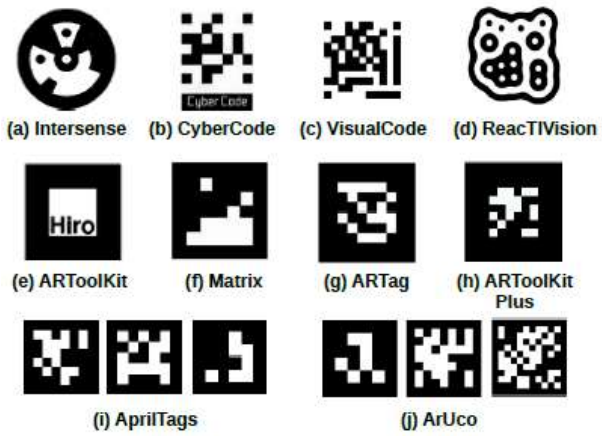


Figura 1.30. Representación de los tipos de marcadores de referencia. [5]

Marcadores de referencia circular

Este tipo de marcadores son un conjunto de círculos concéntricos alternados de color blanco y negro. Fueron diseñados para obtener la distancia entre los centros de los círculos concéntricos como lo hace por ejemplo el marcador circular Intersense [5]. A través del procesamiento de la imagen se extrae el centro del sector circular y se realiza el análisis del área de estos; a su vez, es posible hallar su localización en un plano establecido. Pero su principal inconveniente es que los resultados fueron insatisfactorios ya que mientras más precisión se necesitaba se debía agregar más anillos concéntricos y el marcador perdía la esencia de ser pequeño y portable.

Marcadores de referencia cuadrados sin borde

En esta categoría encontramos tres tipos importantes como:

- **CyberCode** Este tipo de marcadores 2D utilizan tecnología basada en códigos de barra utilizados en implementación de etiquetado de productos.
- **VisualCode** Este tipo de marcadores permiten detectar blobs (objetos binarios grandes) obteniendo el código binario de las 3 esquinas principales y una zona de decodificación de datos similar al de los códigos QR. Permite almacenar vínculos de internet específicos.
- **ReacTVision** Este tipo de marcador se encuentra desarrollado sobre una plataforma con un sistema de identificación de huella digital basados en algoritmos genéticos.

Marcadores de referencia cuadrados con borde

Uno de los tipos de marcadores de referencia más populares son los marcadores cuadrados de borde negro. Estos marcadores permiten una fácil detección y utilizan su codificación interna para obtener características propias del marcador, como posición dentro de un plano, identificador de marcador, ángulo de rotación, entre otros.

La ubicación de un marcador cuadrado dentro de un plano establecido es posible mediante la localización de los cuatro puntos principales; es decir, la ubicación de las esquinas dentro del plano. Dentro de estos tipos se encuentran:

- **ARToolKit** Incluyen un patrón en su región interna pero su principal inconveniente es que utiliza la coincidencia de plantillas para identificar los marcadores y son altamente sensibles a las condiciones de iluminación.
- **ARToolKit Plus** Es un paquete mejorado de ARToolKit, utiliza un método dinámico de codificación binaria para la detección incluye 4096 marcadores de 6x6 bits basados en codificación BCH.
- **AprilTags** El usuario proporciona la distancia entre marcadores, permitiendo la detección y corrección de errores. Se basan en un enfoque heurístico para el análisis de los marcadores y su complejidad geométrica es mínima para facilitar su detección, aunque su principal desventaja es la carga computacional para generar este tipo de marcadores.
- **ArUco** Esta clase de marcadores se ha vuelto popular en los últimos tiempos, ya que permite la generación de varios marcadores con una baja carga computacional. El marcador es cuadrado posee una codificación binaria interna que permite la posibilidad de desarrollar técnicas de detección. Esta codificación se almacena en forma de una matriz cuadrada la cual permite generar un código de identificación individual de cada marcador.

El presente proyecto utiliza marcadores de referencia ArUco que le permiten obtener correspondencias del entorno y puntos clave de imagen; por lo cual nos enfocamos a continuación en este marcador.

Marcador de referencia matricial ArUco

Para distintas aplicaciones se puede hacer uso de uno o varios marcadores los cuales se obtienen a través del manejo de librerías de programación determinadas.

Basados en [37] ArUco es una biblioteca proporcionada por OpenSource. Escrita en lenguaje C++.

Como se indicó anteriormente, los marcadores ArUco son marcadores cuadrados que contienen una matriz cuadrada binaria única para cada marcador. La matriz interna está constituida por unos y ceros la cual muestra el identificador único de cada marcador. Las dimensiones de las matrices que pueden generarse son 4x4, 5x5, 6x6 o 7x7. El borde de color negro alrededor de la matriz interna es definido por el usuario.

En la Figura 1.31 es posible observar la formación de un marcador de referencia, con un borde de 1 bit y una matriz interna de 6x6.

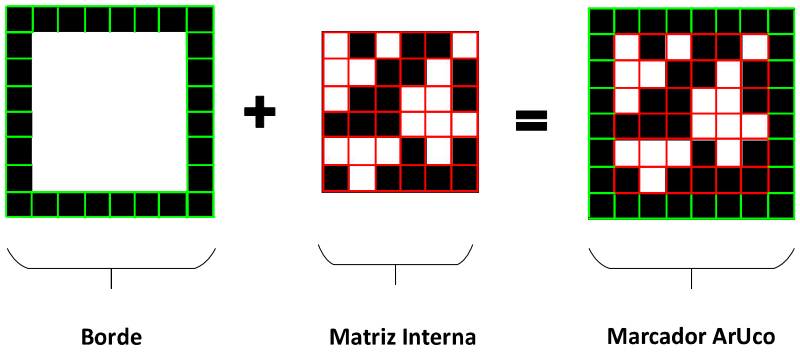


Figura 1.31. Formación del marcador ArUco.

El valor de bits total de un marcador ArUco es la suma del contenido de bits de la matriz interna y del borde. Al multiplicar la dimensión de la matriz; es decir, un marcador de 6x6, por ejemplo, tiene 36 bits en su matriz interna. El borde del marcador puede ser definido por el usuario. Por lo general el valor predeterminado del ancho del borde es de 1 bit, y su valor máximo es 5 bits. En la Figura 1.32, es posible observar el valor total del marcador de referencia con borde con un valor de 64 bits. El borde únicamente de 1 bit de ancho reúne 28 bits en total, pero la información que se necesita analizar es la matriz interna de 36 bits, ya que esta contiene el identificador propio del marcador.

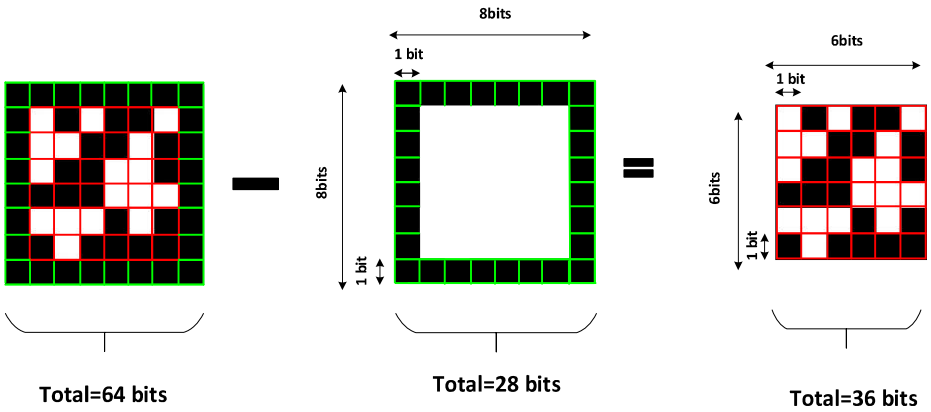


Figura 1.32. Formación del marcador ArUco.

La asignación de 1's y 0's de las matrices corresponden a sus colores, el valor 1 corresponde al color blanco y el valor 0 corresponde al valor negro como se visualiza en la Figura 1.33.

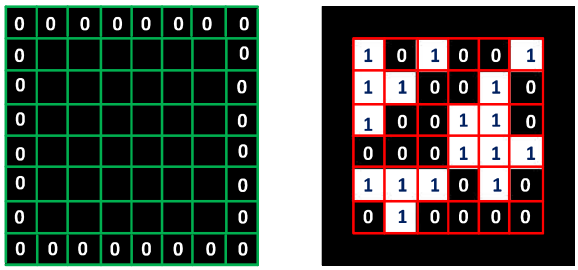


Figura 1.33. Valores de celdas de marcador ArUco.

La creación de marcadores ArUco se basa en el manejo de un conjunto de librerías de programación que son compatibles con el entorno de Open CV, Visual Studio 2015 y Windows 10.

Generación de marcadores de referencia matricial ArUco

Los diferentes tipos de marcadores se encuentran agrupados en diccionarios virtuales proporcionados por las librerías ArUco de OpenCV. La idea del manejo de estos diccionarios es agrupar a los marcadores según su dimensión de la matriz interna. Existen diccionarios compuestos por 50 hasta 1000 marcadores. En la Tabla 1.1. se indica algunos de los diccionarios que proporcionan las librerías de ArUco [38].

Los diccionarios deben ser escogidos tomando en cuenta dos características:

- *Tamaño de diccionario.* – Implica el número de marcadores que agrupa el directorio, existen diccionario de 50 a 1000 marcadores de referencia incluidos.
- *Tamaño de marcador.* – Se especifica el tamaño de marcador a través de la matriz interna del marcador.

Tabla 1.1. Diccionarios de Marcadores ArUco

Nombre de Diccionario	Número de marcadores	Dimensión de matriz interna
DICT_4X4_50	50	4X4
DICT_5X5_100	100	5X5
DICT_6X6_250	250	6X6
DICT_7X7_1000	1000	7X7

Una vez elegido el tipo de diccionario requerido, el usuario tiene a su disposición el número de marcadores generados para utilizarlos de manera individual, como se indica en la Figura 1.34. Para este ejemplo escogió el diccionario DICT_7X7_50, el cual permitió generar 50 marcadores diferentes constituidos por una matriz de 7x7.

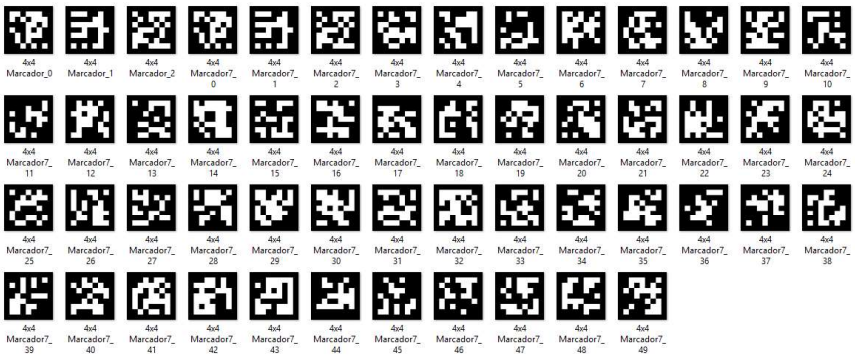


Figura 1.34. Marcadores ArUco generados por el diccionario DICT_7x7_50

Detección de marcadores de referencia matricial

Una vez creados los marcadores de referencia y calibrando la cámara los pasos a seguir para la detección de marcadores, basados en [40], son los que se indican en la Figura 1.35:

Figura 1.35. Pasos para detección demarcadores

Procesamiento de Imágenes

El procesamiento viene dado por varias transformaciones de la imagen original como se indica en la Figura 1.36. La imagen pasa por una conversión a escala de grises, luego una binarización de imagen y por último un análisis de contornos.

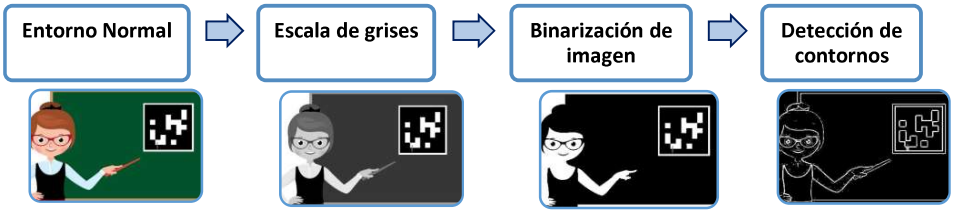


Figura 1.36. Procesamiento de la imagen

- *Conversión a escala de grises.* – Es un proceso donde la intensidad de la imagen a color se ve transformada por una escala de grises según la intensidad de la imagen. La escala de grises varía según la intensidad; la mayor intensidad se transforma en color blanco, la menor intensidad se transforma en color gris y la mínima intensidad se transforma en color negro.

- *Binarización de imagen.* – A partir de la transformación de escala de grises en este proceso se transforma cada píxel de la imagen en negro o blanco, según la intensidad que observe. El valor numérico de los colores corresponde 0 cuando existe presencia de color negro y 255 cuando el color es blanco. Este paso es esencial para encontrar contornos.

- *Detección de contornos.* – A partir de la binarización de imagen es posible detectar los contornos, como se observa en la Figura 1.36, todo lo que tenga un contorno definido se transforma en color blanco y el resto en fondo negro, pero esta técnica transforma hasta el mínimo contorno encontrado en el entorno.

Filtrado de contorno

A partir de la detección de contornos, es posible analizar todo el entorno. Como se indica en la Figura 1.37, todos los contornos analizados son mostrados, pero no todos pueden ser candidatos a ser marcadores. Esto provocaría una carga computacional significante, por lo que es necesario excluir los contornos que no son necesarios para el análisis de detección de marcadores.

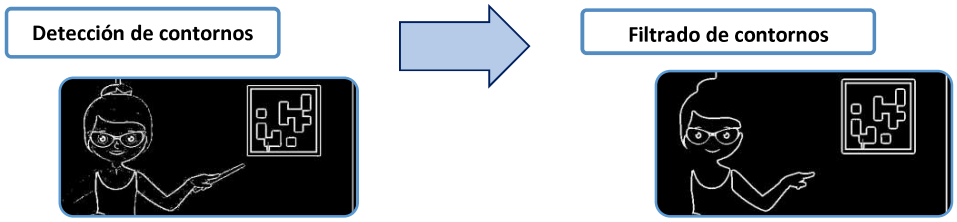


Figura 1.37. Filtrado de contornos en imagen

En este filtrado es posible resaltar lo que más se acerque a un polígono, además de eliminar puntos o partes donde la intensidad es demasiada baja.

Extracción de bits

Al extraer los posibles marcadores, las imágenes analizadas son divididas por medio de una cuadrícula (Figura 1.38). Esta viene dada por la dimensión de la matriz interna de los marcadores escogidos por el usuario; es decir, si el usuario determina marcadores de 6x6, la cuadrícula será de 6x6.

Esta cuadrícula se sobrepone sobre todas las imágenes que considere candidata de marcadores cuadrados con borde, a partir de esto, se compara cada cuadro con la codificación binaria de los marcadores del diccionario escogido.

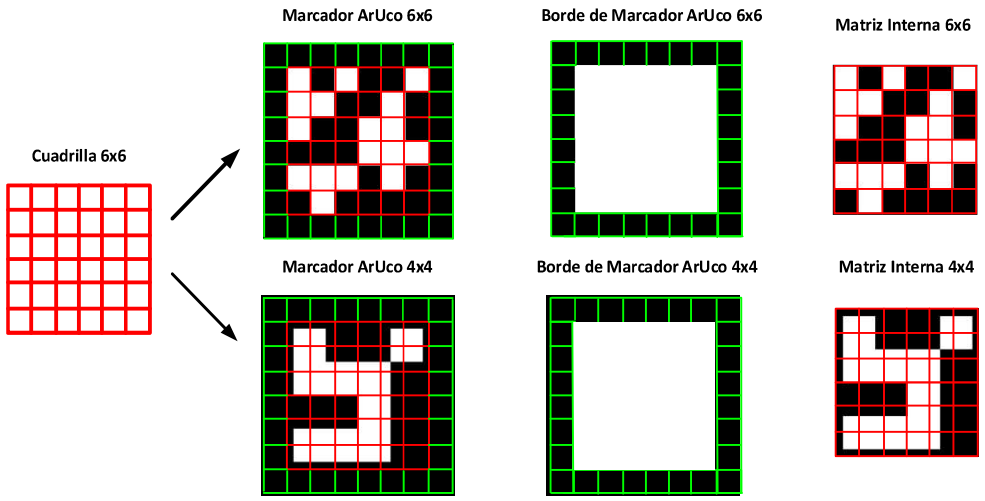


Figura 1.38. Análisis de cuadrilla 6x6 en marcador ArUco 6x6 y 4x4

El análisis de la cuadrilla sobre las candidatas, se indica en la Figura 1.38. Como se mencionó, si el usuario genera marcadores ArUco 6x6, la cuadrilla vendrá dada por la misma dimensión. Si en el entorno capturado existen candidatas de marcadores 6x6 estos coinciden con dicha cuadrilla.

Al realizar la coincidencia entre la imagen capturada y la cuadrilla, se extrae el contenido de bits del borde y se analiza la matriz interna de la imagen capturada. Posteriormente al análisis de la matriz, es posible extraer la codificación binaria de dicha matriz.

Esta codificación es comparada con la codificación establecida de cada marcador del diccionario escogido por el usuario. Para el caso de un marcador de 6x6 la codificación viene dada por 36 dígitos y si el usuario generó 50 marcadores se realizará 50 coincidencias de códigos para identificar el marcador observado.

Un ejemplo se muestra en la Tabla 1.2, donde se encuentran 3 de las codificaciones binarias del diccionario DICT_6X6_50, y la codificación de la imagen capturada. El análisis realiza la coincidencia de codificación, por lo que para el ejemplo la cámara capturó al marcador 8 del diccionario ya que coinciden en su codificación.

Tabla 1.2. Comparación de codificación binaria de marcador e imagen capturada

Codificación establecida de Marcadores de Diccionario DICT_6x6_50									
Marcador 8	1010	0111	0010	1001	1000	0111	1110	1001	0000
Marcador 21	1000	0011	0110	1101	1110	0001	1000	1101	1100
Marcador 36	0010	0101	1010	1000	1111	0111	1010	0001	1000
Codificación de Imagen Capturada									
Imagen Capturada	1010	0111	0010	1001	1000	0111	1110	1001	0000

Pero si en el entorno existen marcadores que el usuario no generó, como es el caso de la Figura 1.38, la cuadrícula de 6x6 automáticamente se sobrepone en la imagen detectada. Como es posible observar la cuadrilla no coincide con el tamaño del patrón ya que no se puede analizar una cuadrilla de 6x6 en un marcador de 4x4, entonces esta imagen será descartada en la detección.

El tamaño físico del marcador y la unión de cada celda son factores importantes en la detección, ya que si el marcador impreso es demasiado pequeño o un color se sobrepone al otro podrían existir errores de identificación. Para evitar problemas en el análisis de identificación se ignora pixeles en los márgenes de las celdas de la cuadrícula. Como se indica en la Figura 1.39, el cuadro verde es el que será analizado provocando una considerable separación entre celdas, y eliminando ruido entre las celdas vecinas.

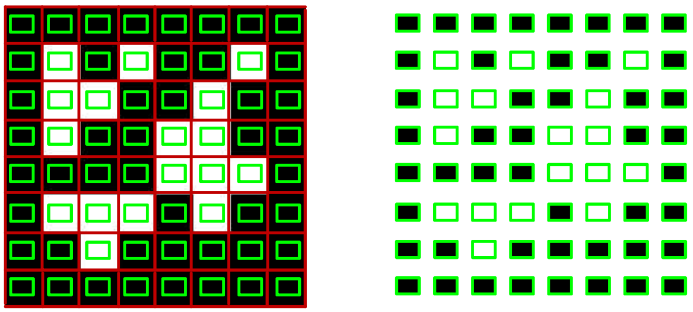


Figura 1.39. Análisis de pixeles en marcador ArUco

Protocolo de comunicación ZigBee

El término Xbee sirve para referirse a los módulos de comunicación inalámbrica mediante radiofrecuencia desarrollados por DIGI Internacional, mientras que la expresión ZigBee es utilizada para hacer referencia al protocolo de comunicación. Para establecer la comunicación entre los robots móviles y la computadora, en el presente proyecto se utiliza el protocolo ZigBee, el cual permite trabajar sobre redes inalámbricas

de baja de potencia. Una red de comunicación ZigBee se encuentra conformada por varios tipos módulos de comunicación Xbee que desempeñan funciones específicas, de acuerdo con las funciones que cumplen dentro de la red se los clasifica en [41]:

- ***Coordinador:*** Las redes ZigBee pueden tener únicamente un dispositivo de coordinación, cuya función es controlar la red, distribuir las direcciones hacia otros dispositivos y administrar otras funciones que definen a la red.
- ***Router:*** Es un nodo ZigBee y tiene la capacidad de unir redes, de igual manera permite enviar, recibir y enrutar información hacia otros dispositivos.
- ***End device:*** También conocido como dispositivo final, pueden unirse a redes ZigBee, pueden cumplir únicamente dos funciones: envío y recepción de información, por lo cual es necesario disponer de un coordinador o router que actúa como dispositivo principal.

Una vez conocidos los componentes de la red inalámbrica se detallan las topologías de comunicación que permite establecer el protocolo ZigBee, las mismas que se presentan en la Figura 1.40.

Topología ZigBee

Las redes ZigBee pueden presentar cuatro tipos de topologías:

- ***Par:*** Es la más simple, se forma por dos módulos o nodos.
- ***Estrella:*** Dispone de un módulo coordinador que se encuentra en el centro de la red y se conecta a dispositivos finales (End devices).
- ***Malla:*** Se conforma por un coordinador y varios routers o nodos que se comunican entre sí.
- ***Árbol:*** Los routers forman una especie de columna vertebral, con dispositivos finales conectados a estos y un coordinador en la raíz de la red.

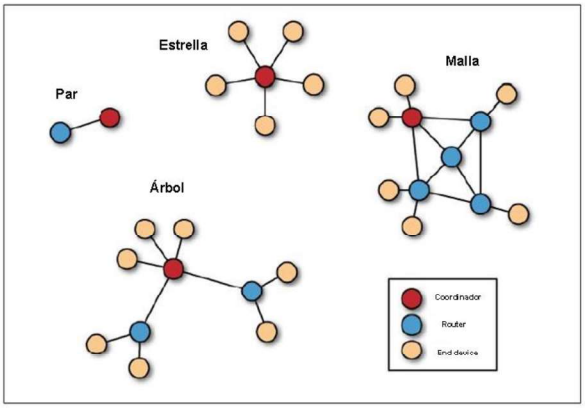


Figura 1.40. Topologías de red ZigBee. [41]

El sistema de comunicación del presente proyecto emplea la topología ZigBee estrella. Mediante el tipo de comunicación elegido, es posible vincular a los robots con el sistema de control.

Para culminar este capítulo, se analiza los trabajos similares que existen dentro de la Escuela Politécnica Nacional, con el fin de explicar las necesidades y ventajas del desarrollo del presente proyecto.

Trabajos similares

En vista de los diferentes proyectos desarrollados en el ámbito de la robótica móvil, es necesario establecer la comparación con algunas de las investigaciones realizadas en la carrera de Electrónica y Control de la Escuela Politécnica Nacional, las mismas que han sido aplicadas en robots móviles de configuración diferencial. Esto servirá para resaltar las contribuciones del proyecto desarrollado.

- ✓ R. Vizuete, Diseño y Simulación de algoritmos de control distribuido para la formación y seguimiento de trayectoria de robots móviles tipo uniciclo y para manipuladores móviles de 3 grados de libertad, Quito, 2017. [42]

En este trabajo se diseñan y simulan algoritmos de control distribuido en sistemas multiagentes para la formación y seguimiento de trayectoria de dos tipos de robots móviles: tipo uniciclo y manipuladores móviles de tres grados de libertad. Las diferentes trayectorias y formaciones son simuladas en el software Matlab.

El presente proyecto busca ensamblar un banco de pruebas físico inicialmente para dos prototipos de robots uniciclos, orientado al estudio y realización de aplicaciones prácticas de algoritmos de control de posición y seguimiento de trayectorias individuales, sin considerar la formación de los robots. De la misma forma se utilizará el software Matlab para generar las trayectorias preestablecidas y crear la interfaz de usuario.

- ✓ L. Capito; P. Proaño, Seguimiento de trayectorias mediante cuatro técnicas de control utilizando una plataforma robótica Pioneer 3DX y el sistema operativo robótico ROS, Quito, 2015 [14]

Para el desarrollo de los controladores en este trabajo se considera el modelo cinemático del robot diferencial no holonómico mejorado con la finalidad del diseño e implementación de algoritmos de control basados en álgebra lineal y PID aplicados únicamente a del robot Pioneer 3DX para el seguimiento de trayectorias a diferentes velocidades y perturbaciones; además, cuenta con una interfaz para monitoreo del robot.

La diferencia con el presente proyecto radica en la implementación de un banco de pruebas que consta de un sistema de monitoreo en base al posicionamiento del robot mediante visión artificial y un sistema de control desarrollado en el software Matlab basados en dos robots móviles de tamaño reducido. La técnica de control implementada también se basa en álgebra lineal y depende del modelo cinemático del robot diferencial no holonómico mejorado.

- ✓ B. Sarabia, Diseño y simulación de tres técnicas de control clásicas y robustas aplicadas al seguimiento de trayectorias ante la presencia de retardos fijos para la plataforma robótica Pioneer 3DX, Quito, 2017. [43]

El trabajo mencionado se basa en el modelo de una plataforma Pioneer 3DX con el propósito de estudiar el efecto que tienen los retardos en el seguimiento usando controladores: PD Like y Controlador por Modos Deslizantes (SMC), en un esquema de Predictor de Smith. Para la obtención de resultados se utiliza el simulador MobileSim.

En cambio, el trabajo planteado implementa un banco de pruebas para asignar y generar trayectorias individuales al grupo de robots móviles uniciclos ensamblados y de tamaño reducido, permitiendo realizar pruebas prácticas del control de posición y trayectorias individuales dentro de un área de trabajo a través de un sistema de posicionamiento basado en visión artificial.

.

2. METODOLOGÍA

En este trabajo se emplea una investigación de tipo aplicada, mediante la cual se implementó un banco de pruebas que contiene dos mini robots de tamaño reducido, sistema de monitoreo, sistema de control e interfaz gráfica con el fin de realizar control de posición y seguimiento de trayectoria. La recopilación de datos se realizó mediante una amplia consulta de fuentes bibliográficas como libros, artículos de periódicos científicos, artículos de conferencias y sitios web.

El presente trabajo consta de cuatro fases metodológicas: La fase teórica comprende el estudio de tipos de robots móviles, tipos de bancos de pruebas, tipos de algoritmos de control, además de técnicas de visión de computadora esta fase desarrolló en el capítulo uno. En el presente capitulo se desarrolla la fase de diseño e implementación del banco de pruebas con los diferentes sistemas mencionados. Finalmente, se presenta la fase de resultados del banco de pruebas ante el algoritmo de control implementado y las diferentes trayectorias establecidas a través de la interfaz gráfica.

La arquitectura de la plataforma se encuentra conformada por:

1. Dos prototipos de robots móviles de tamaño reducido.
2. Un sistema de monitoreo el cual permite obtener el posicionamiento de los robots móviles dentro del área de trabajo de 1 X 1.4 metros cuadrados.
3. Un sistema de control implementado basado en álgebra lineal.
4. Un sistema de comunicación entre los robots móviles y la computadora.
5. Un sistema de visualización conformado por una interfaz de usuario para el manejo del banco de pruebas.

En la Figura 2.1 se observa la representación gráfica del sistema implementado con las respectivas partes mencionadas anteriormente.

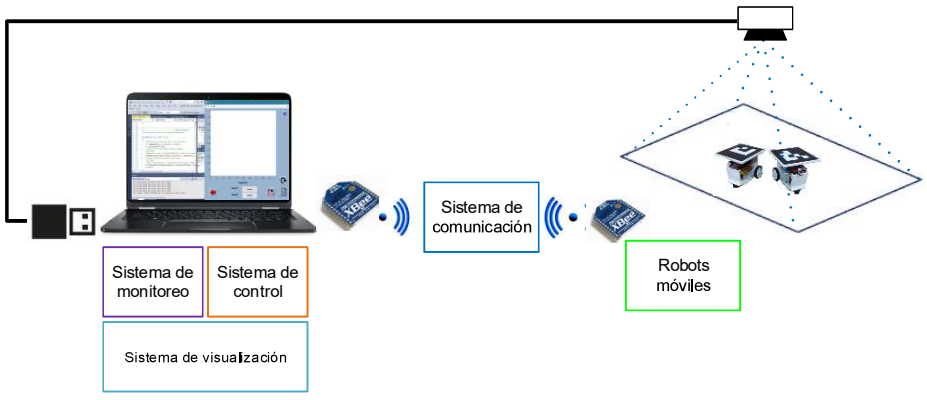


Figura 2.1. Representación de la arquitectura del banco de pruebas construido.

Conforme a la sección de Banco de Pruebas, a este banco de pruebas se lo puede clasificar como:

- Multirobot homogéneo ya que los dos robots son de las mismas características.
- Altamente integrado debido a que consta de un sensor (cámara web), una interfaz gráfica, y comunicación bidireccional que permite enviar las acciones de control a los robots móviles y a la vez conocer su estado.
- De aplicación específica para el seguimiento de trayectorias y el control de posición.
- Modular centralizado puesto que utiliza un módulo central de comunicación y dos esclavos.

El manual de usuario y los programas desarrollados para el presente proyecto pueden ser solicitados en la Facultad de Ingeniería Eléctrica y Electrónica. Por otro lado, los programas utilizados como Matlab y Visual Studio con sus respectivas licencias se encuentran disponibles para los estudiantes y docentes pertenecientes a la Escuela Politécnica Nacional.

La interconexión de las cinco partes del sistema se indica mediante el diagrama de bloques de la Figura 2.2. El sistema de monitoreo está relacionado con la localización de los robots; esta información es proporcionada a Matlab para realizar el control respectivo. Las acciones de control son enviadas hacia los robots mientras que el estado de la batería de cada uno se recibe mediante el sistema de comunicación. La interfaz permite realizar el monitoreo y tener el control de todo el sistema.

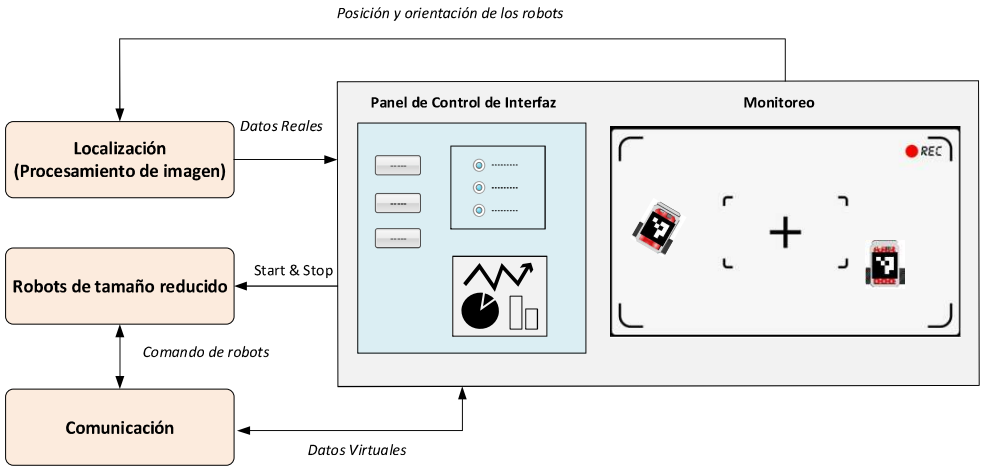


Figura 2.2. Funcionamiento del sistema implementado.

A continuación, se indican los elementos y el proceso de desarrollo de cada una de las partes de la arquitectura del banco de pruebas.

2.1 Robots móviles de tamaño reducido

Con la finalidad de cumplir los objetivos planteados, se construyeron dos prototipos de robots móviles cuyo volumen se encuentra dentro del rango de $1cm^3$ a $1dm^3$, y con una masa que no sobrepasa los 6 kg, justamente por estas características se denominan robots de tamaño reducido. Estos robots se basan en el diseño de los robots móviles de la plataforma MICROMVP [11]. A continuación, se indican cada uno de los componentes que se emplearon para la construcción de los mismos, así como su estructura en general. Los robots móviles constan tanto de una parte electrónica como una mecánica.

Componentes electrónicos

Los componentes electrónicos están constituidos por los siguientes elementos:

- Minimotores DC.
- Módulo DRV8835.
- XBee Serie 1.
- Fio V3 – Atmega 32U4.
- Batería de litio.

Las especificaciones dadas por los fabricantes de los componentes electrónicos se detallan en el ANEXO I.

Minimotores DC

Para el sistema de locomoción se utilizan minimotores de plástico con caja reductora como el que se presenta en la Figura 2.3. Estos minimotores de corriente continua son adecuados para el trabajo con pequeños robots, y están diseñados para trabajar con un voltaje nominal de 4.5 V. Además, pueden trabajar en un rango de voltaje alrededor de su voltaje nominal, operando entre 3 V a 6 V sin ningún problema; sin embargo, la rotación inicia a partir de una tensión de 0.5 V. El eje de salida tiene forma de "*D"*, con una longitud de 9.7 mm y un diámetro de 3mm.

Figura 2.3. Minimotor DC plástico. [44]

Módulo DRV8835

Este módulo se utiliza para manejar los dos minimotores de corriente continua. El circuito integrado DRV8835 es un pequeño puente H que se encargará del control bidireccional de ambos minimotores DC entregando hasta 1.2 A. Su tamaño es de 1 cm X 1.7 cm y dispone de 14 pines que facilitan su uso al momento de trabajar con placas de prueba, Figura 2.4.

Figura 2.4. Módulo DRV8835 vista superior. [45]

El módulo DRV8835 permite controlar a los minimotores DC mediante dos modos de operación: IN/ IN y PHASE/ENABLE. Si el pin de MODE se encuentra desconectado o en bajo se trabaja en modo IN/IN que consiste en seis maneras de operación de los minimotores: dos para freno, dos para reversa y dos para ir hacia adelante, este modo no se considera en el proyecto. Por otro lado, si el pin MODE se lo pone en alto se trabaja en modo PHASE / ENABLE, este modo de control es el elegido para manejar los minimotores de los robots móviles y su forma de operación se indica en la Tabla 2.1.

Tabla 2.1. Modo de operación PHASE/ENABLE del módulo DRV8835.

Módulo DRV8835 en modo: PHASE/ENABLE				
PHASE	**ENABLE**	**OUT1**	**OUT2**	**Funcionamiento**
0	PWM	PWM	Bajo	PWM % (Adelante)
1	PWM	Bajo	PWM	PWM % (Reversa)
X	0	Bajo	Bajo	Freno (Salidas cortocircuitadas a tierra)

Tanto las conexiones de potencia de los motores como las conexiones de control se encuentran separadas a los lados del módulo DRV8835, como se muestra en la Figura 2.5.

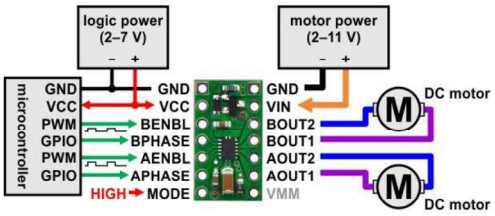


Figura 2.5. Conexiones del módulo DRV8835 en modo PHASE/ ENABLE. [45]

XBee Serie 1

El módulo de comunicación XBee Serie 1 también conocido como XBee 802.15.4, el cual se muestra en Figura 2.6, es fácil de usar y su topología de comunicación es punto a punto; sin embargo, es posible trabajar con estos módulos en topología estrella siempre que se trabaje con otros módulos XBee Serie 1.

Figura 2.6. Módulo Xbee Serie 1. [46]

Fio V3- Atmega 32U4

La tarjeta Fio V3 que se observa en la Figura 2.7 es la unión de dos tecnologías: Arduino Fio y Atmega 32U4. Permite conectar una batería LiPo como fuente de la tarjeta para que funcione de manera portátil, además ayuda a la comunicación con dispositivos inalámbricos XBee ya que posee un zócalo integrado.

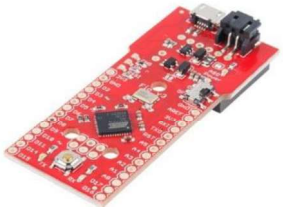


Figura 2.7. Tarjeta Fio V3 – Atmega 32U4. [47]

La distribución de pines en la tarjeta Fio V3 proporcionada por los fabricantes se muestra en la Figura 2.8. Las características más relevantes del hardware de la tarjeta son [47]:

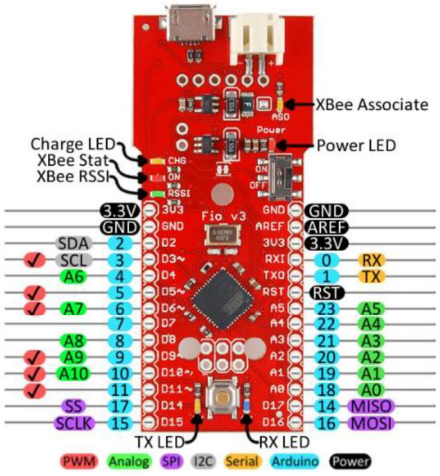


Figura 2.8. Hardware de la Tarjeta Fio V3. [47]

Pines de alimentación: Siempre que la placa se alimente a través del conector JST o USB, la tensión de alimentación se regula hasta 3.3V generando dicho voltaje por los pines etiquetados con '3V3'.

Pin RST: Permite reiniciar la tarjeta, internamente posee una resistencia de 10KΩ en pull up, por lo cual este pin se debe conectar a tierra para que se genere el reinicio de la tarjeta.

Pines de Entrada / Salida: Cada pin se puede usar como entrada o salida digital. Los pines pueden ser referenciados con el IDE de Arduino mediante su valor entero entre 0 y 23. Además, los pines A0 – A10 se pueden referenciar a través de su pin analógico o pin digital. Los pines 12 y 13 no se encuentran disponibles en la tarjeta Fio V3 debido a que se conectan internamente con el microcontrolador.

Hay once pines ADC que pueden ser usados con entradas analógicas, seis pines con salida analógica para modulación de ancho de pulso (PWM). También se dispone de clavijas para comunicación UART, I2C y SPI.

La Fio V3 tiene cinco interrupciones externas dispuesta de la siguiente manera: pin 3 – interrupción 0, pin 2 – interrupción 1, pin 0 – interrupción 2, pin 1 – interrupción 3, pin 7 – interrupción 4.

Leds integrados: Se tiene el led de encendido de la tarjeta etiquetado como 'Power' (rojo), los leds en la parte inferior de la tarjeta etiquetados como 'RX'(azul) y 'TX'(amarillo) indican cuando se transfieren los datos desde o hacia la Fio V3 mediante el USB. Hay tres leds vinculados con el módulo XBee que se puede conectar a la tarjeta Fio V3. El primer led etiquetado como 'ON' (rojo) indica si el XBee se encuentra conectado a la tarjeta, el segundo led 'RSSI' (verde) indica cuando la señal es recibida, el tercer led 'ASO' (amarillo) que parpadea cuando el XBee está asociado a la tarjeta.

Finalmente, el led etiquetado como 'CHG' (amarillo) indica si una batería LiPo está conectada o a su vez si esta descargada.

Cargador de baterías LiPo: La tarjeta Fio V3 tiene incorporado el circuito integrado MCP73831 el cual se encuentra configurado para cargar baterías hasta de 4.5V, es así como el indicador led 'CHG' se apagará una vez que la batería llegue a ese nivel de voltaje, la corriente de carga es de 500 mA.

Baterías de Litio

Una batería de Litio se usa como fuente de alimentación en cada uno de los robots móviles, gracias a que es liviana y potente, por lo que no agrega mayor masa a su

estructura. El voltaje de estas baterías varía entre 4.2 V y 3.7 V, y tienen una capacidad de 500 mA para trabajar a 1.9 Wh. Además, tienen un conector JST -PH para poder conectarse a la tarjeta Fio V3, Figura 2.9.

Figura 2.9. Batería de Litio de 3.7V – 500mA [48].

Componentes mecánicos

Los componentes mecánicos de los robots son:

- Ruedas.
- Bola de castor.
- Estructura mecánica de polímero.

Ruedas

Las ruedas empleadas para los prototipos se encuentran diseñadas para encajar a los ejes de salida D de los minimotores DC de 3mm. Las ruedas son hechas de plástico y tienen neumáticos de silicona como se muestra en la Figura 2.10. Su diámetro es de 32 mm, su ancho es de 6.5 mm y su masa de 3.2 gramos.

Figura 2.10. Par de ruedas para minimotores Pololu. [49].

Bola de castor

También conocida como bola loca que brinda mayor estabilidad en los robots móviles. La rueda de castor elegida es de la marca Pololu con una carcasa de Acrilonitrilo butadieno estireno (siglas en ingles ABS) que es un plástico muy resistente al impacto. La bola de metal tiene un diámetro de 3/8 de pulgada, además incluye espaciadores y tornillos para ser montada en el chasis, Figura 2.11. Los tornillos no son usados en los prototipos ya que las ruedas de castor serán fijadas a la estructura mecánica con pegamento.

Figura 2.11. Bola de castor o bola loca. [50].

Estructura mecánica de polímero

Como se indició anteriormente, la estructura de los robots móviles implementados se basa en el diseño de los robots pertenecientes a la plataforma MICROMVP que se menciona en la sección de Banco de pruebas. Las piezas que conforman la estructura original son de polímero y se diseñaron para la componente electrónica (piezas principales). Mediante el uso de SolidWorks se desarrollaron dos piezas adicionales (piezas secundarias) que permiten colocar los marcadores de referencia de manera más sencilla en los robots móviles. Cada una de las piezas se las puede encontrar en el archivo digital adjunto a este proyecto para que puedan ser impresas en 3D. Las dimensiones de las piezas que conforman la estructura de los robots se indican en el ANEXO II.

Ensamblaje de la estructura mecánica

El ensamblaje se realiza utilizando las piezas de polímero impresas y la bola de castor con un espaciador que conforman la componente mecánica, Figura 2.12. Para unir cada una de las piezas se usa pegamento.

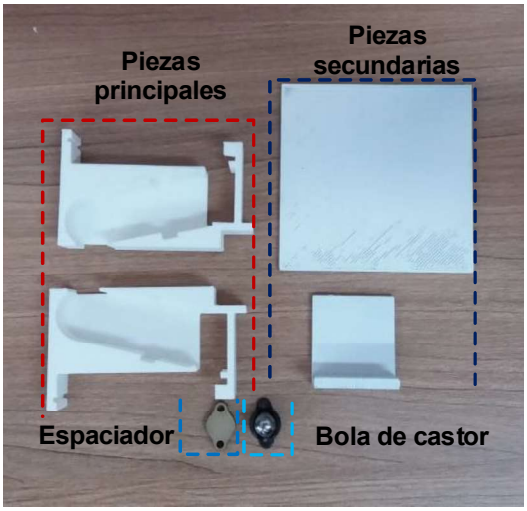
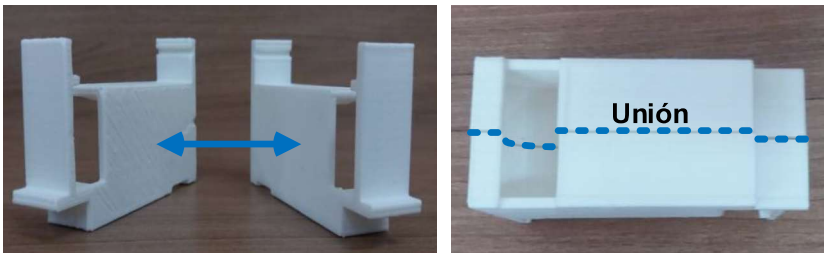


Figura 2.12. Piezas que componen la estructura mecánica.

Las dos piezas principales se unen como se indica en la Figura 2.13, para formar la base de los robots.



(a) (b)

Figura 2.13. (a) Dirección de unión de las dos piezas principales (b) Vista Inferior de la base del robot.

Una de las piezas secundarias se usa como soporte del marcador de referencia y se une a la base como se muestra en la Figura 2.14.

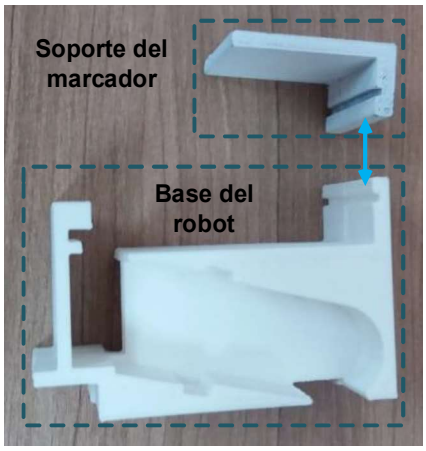


Figura 2.14. Ensamble del soporte del marcador con la base.

Luego en la parte inferior de la base se unen el espaciador y la bola de castor a la base de los robots como se muestra en la Figura 2.15.

Figura 2.15. Ensamble de la bola de castor con la base.

En la pieza secundaria que la denominaremos estructura para el marcador, se debe pegar en uno de sus lados el marcador de referencia impreso en papel y por el otro un pedazo de velcro para unir a la estructura final, como se muestra en la Figura 2.16.

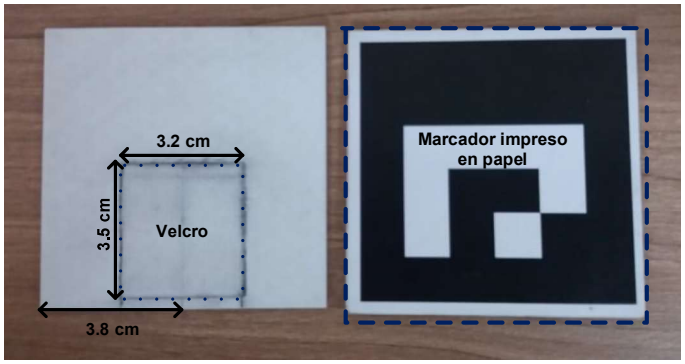


Figura 2.16. Estructura para marcador.

La estructura mecánica terminada se puede observar en la Figura 2.17, en esta se añade un pedazo de velcro con la finalidad de poner y quitar la estructura del marcador de manera rápida y sencilla.

Figura 2.17. Estructura mecánica final.

Implementación de los robots móviles

Utilizando los componentes detallados anteriormente: tarjeta Fio V3, minimotores DC, batería, XBee y módulo DRV8835 se realizan las conexiones que se muestran en el diagrama de la Figura 2.18 (a), con la finalidad de interconectar los elementos que conforman la parte electrónica de los robots móviles. Una vez realizada la conexión electrónica que se indica en la Figura 2.18 (b) se debe introducir cuidadosamente en la estructura mecánica final.

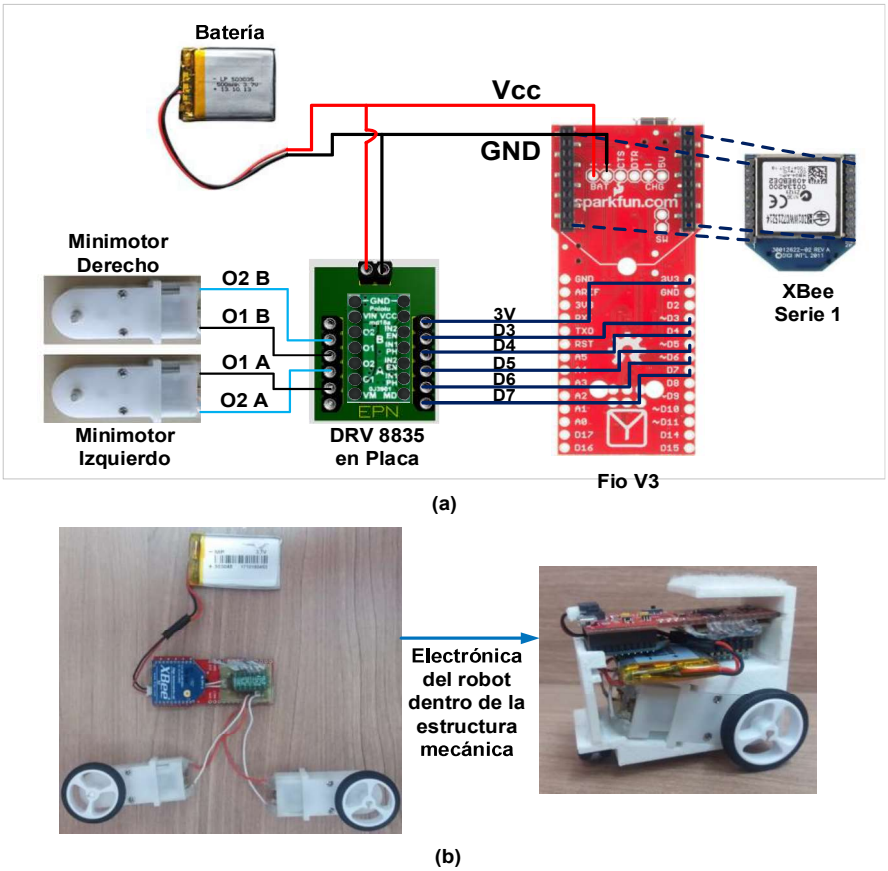


Figura 2.18. (a) Diagrama de conexiones (b) Conexión electrónica del robot.

Para conectar el módulo DRV8835 se diseñó una pequeña placa en Ares Proteus para cambiarlo fácilmente en caso de avería. Inicialmente se consideró colocar una resistencia de 10kΩ en paralelo para medir el estado de batería de los robots usando una entrada analógica de la tarjeta Fio V3. Esta última consideración no se la realizó debido a que la configuración interna de la tarjeta permite usar la entrada analógica A11 para verificar el estado de la batería.

En el ANEXO III se muestra la placa que se diseñó para el módulo DRV8835. Los prototipos implementados a partir de las componentes electrónica y mecánica se indican en la Figura 2.19. Cabe mencionar que como paso final se colocan las ruedas en los minimotores.

Los robots implementados tienen un volumen de 280 cm^3 es decir son Mini robots y una masa de 129 gramos por lo que se los considera como Small Sized robots, esto de acuerdo con la clasificación revisada en la sección de Clasificación de la robótica.

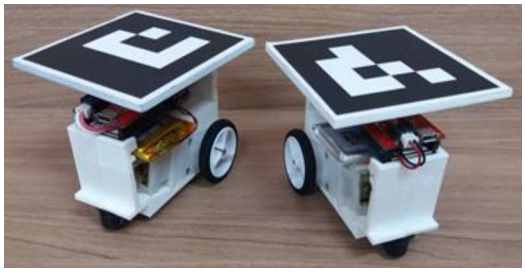


Figura 2.19. Robots móviles implementados.

Una vez construidos los prototipos de robots móviles se procede a implementar el sistema de monitoreo.

2.2 Sistema de monitoreo

El sistema de monitoreo está compuesto por un ordenador, cámara de video y marcadores de referencia como se observa en la Figura 2.20.

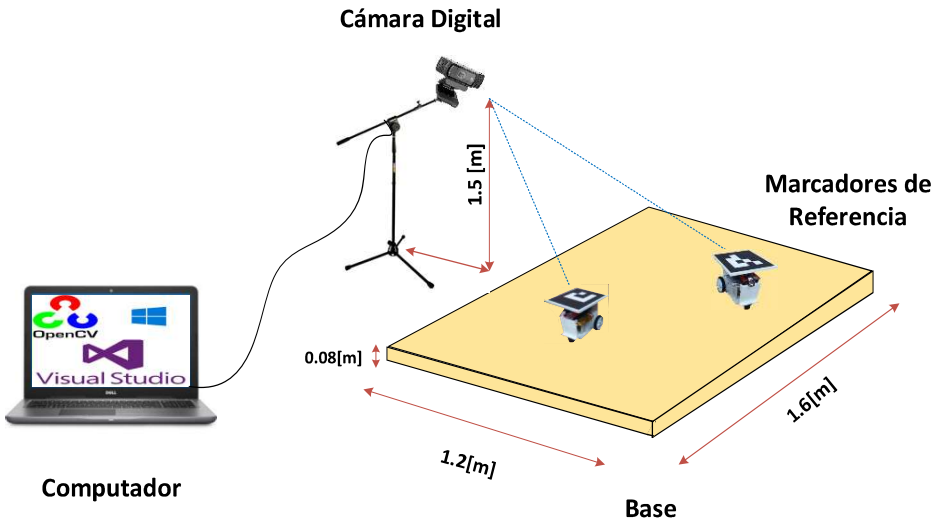


Figura 2.20. Arquitectura del sistema de monitoreo.

A continuación, se especifican las características de la cámara empleada para el presente proyecto.

Cámara de video

La cámara de video es la fuente de fotogramas para el sistema. A través de este dispositivo se puede observar lo que pasa en el entorno de trabajo. Es necesario colocar la cámara de video en un pedestal que permita mantenerla a una altura predeterminada como se indica en la Figura 2.20. El procesamiento de la imagen depende de la utilización de una cámara adecuada para la detección de los marcadores ArUco.

En el presente proyecto se realizaron pruebas con dos tipos de cámaras diferentes para escoger la de mejores características. En estas pruebas una persona se ubicó a una distancia de 1.5 metros frente al lente de la cámara para relacionar la misma altura del pedestal sobre la base (Ver Figura 2.20). La comparación de las características internas de estas dos cámaras se presenta en la Tabla 2.2.

Tabla 2.2. Comparación de cámaras de video Genius Eye 110 vs Logitech PRO-HD C920. [51,52]

Características	**Genius Eye 110**	**Logitech PRO-HD C920**
Formato de Video Digital	WMV	MP4
Máxima Resolución de Video [pixeles]	640x480	1920x1080 [HD]
Cuadros por segundo [fps]	15	30
Sensor de imagen	0.1 MP CMOS	3 MP CMOS
Formato de Imagen	JPEG	MJPEG
Memoria RAM Requerida del sistema	256 MB RAM	2 GB RAM
Procesador mínimo requerido del sistema	1Hz Pentium 4	2.4 GHz Intel Core 2 Duo processor

Como se observa en la Tabla 2.2 las características entre la cámara Genius Eye110 y la cámara Logitech PRO-HD C920 varían notoriamente. Se configuraron las cámaras con una resolución de 640x480 pixeles ya que permite reducir la carga computacional del sistema. Finalmente se procedió a capturar el entorno como se observa en la Figura 2.21 y Figura 2.22.

(a)

(b)

Figura 2.21. (a) Imagen Capturada con Cámara Genius; (b) Cámara Genius Eye 110 [51]

(a) (b)

Figura 2.22. (a) Imagen capturada con Cámara Logitech; (b) Cámara Logitech C920 [52].

La calidad de imagen capturada con la cámara Genius Eye 110, presenta gran distorsión y baja resolución, mientras que la imagen capturada con la cámara Logitech presenta una gran calidad de imagen donde es posible analizar cada detalle del entorno capturado. Como se especificó en la sección de Detección de marcadores de referencia matricial, la detección de los marcadores de referencia se basa en algunos factores, como el tamaño de marcador y calidad de imagen mostrada. Por lo tanto, es necesario una cámara que brinde el mayor número de características del entorno capturado. En base a los resultados obtenidos, la mejor imagen capturada fue con la cámara Logitech C920, por lo cual esta es la utilizada para el presente proyecto.

Figura 2.23. Montaje de Cámara y base del sistema de monitoreo.

Se colocó la cámara en un pedestal de micrófono para establecer su posición fija, como se puede observar en la Figura 2.23. La altura de la cámara se determinó a través de varias pruebas con el objetivo de poder capturar toda la base. La altura que permitió capturar el entorno necesario es de 1.5 metros. Esta altura permite la detección de manera efectiva de los marcadores de referencia dentro de su plano de trabajo establecido.

Para la realización del proceso de visión por computadora son necesarios tres aspectos importantes como se especificó en la sección de Técnicas de visión por computadora:

- *Generación de marcadores de referencia.* – Se eligió los marcadores ArUco como medio de identificación de los robots. En esta etapa se genera el número de marcadores necesarios y el tipo.
- *Calibración de la cámara.* – Se realizan las configuraciones necesarias de la cámara para la detección de marcadores
- *Detección de marcadores ArUco.* - La etapa de captura de video consiste en recibir fotogramas de la cámara instalada además involucra el procesamiento de las capturas de video recibidas, este procesamiento se lo realiza en OpenCV.

A continuación, se detallan cada uno de estos aspectos para el caso del presente proyecto.

Generación de Marcador ArUco

En la sección de Generación de marcadores de referencia matricial ArUCo, se especificó la manera de generar marcadores de referencia ArUco, a través del software computacional OpenCV y Microsoft Visual Studio 2015.

En la actualidad existen algunas versiones de OpenCV. Para el presente proyecto se utilizó la versión 3.4.1, que dispone de varios módulos, además de la compatibilidad con OpenCV Contrib, que es una extensión de nuevas librerías como las de ArUco Marker permitiendo la generación y rastreo de marcadores de referencia.

Basado en [5], los módulos de OpenCV utilizados en este proyecto, para generar los marcadores de referencia matricial ArUco son los siguientes:

- *Core.* – Este módulo contiene todos los tipos de objetos básicos y operaciones generales de OpenCV.
- *Imgproc.* – El procesamiento de imágenes está incluido en este módulo, procesos como decodificación de imágenes con filtros, operadores, y transformaciones básicas.

- *Imgcodecs.* – Este módulo permite la lectura y escritura de imágenes en el disco.
- *Highgui.* – Permite la creación de aspectos básicos de una interfaz gráfica, comenzando por el simple hecho de mostrar una imagen.
- *Aruco.* – Este módulo contiene toda la biblioteca ArUco, biblioteca diseñada por Rafael Muñoz y Sergio Garrido [30]
- *Calib3d.*- Este módulo incluye una gama de algoritmos que permitirá la calibración de cámaras digitales.

Las librerías de OpenCV han sido desarrolladas en el lenguaje C++ en Visual Studio 2015. Para la vinculación de Visual Studio 2015 y OpenCV es necesario realizar los siguientes pasos [18].

- *Configuración de las variables del sistema.* - Es necesario configurar el *path* del sistema agregando la ubicación de la carpeta donde se encuentran los archivos de OpenCV.
- *Incluir ubicación de directorios de OpenCV.* – En las propiedades generales del proyecto creado en Visual Studio 2015, es necesario agregar la ubicación de los todos los módulos instalados de OpenCV.
- *Incluir ubicación de librerías de OpenCV.* – En las propiedades de vinculación del proyecto creado en Visual Studio 2015, se agrega la ubicación de todas las librerías instaladas de OpenCV.
- *Establecer las librerías que se utilizarán en el proyecto.* – Al igual que el paso anterior, en esta sección se agregan las librerías necesarias que se utilizarán en el proyecto, en nuestro caso, se utilizan las siguientes librerías *opencv_core341d.lib, opencv_aruco341d.lib, opencv_imgproc341d.lib, opencv_highgui341d.lib, opencv_video341d.lib, opencv_imgcodecs341d.lib, opencv_videoio341d.lib, opencv_calib3d341d.lib, opencv_ccalib341d.lib, opencv_objdetect341.lib, opencv_ml341.lib.*

Luego de la vinculación de estos programas, para entender la generación de marcadores de referencia ArUco, es necesario especificar aspectos generales del uso de la biblioteca ArUco Markers. A continuación, se detallan estas características.

Como se indicó en la sección de Marcador de referencia matricial ArUco, los marcadores ArUco son marcadores cuadrados, que contienen una matriz binaria única para cada marcador. Estos marcadores se encuentran agrupados en diccionarios predefinidos.

Se estableció el diccionario DICT_4X4_50, este diccionario contiene 50 marcadores cada uno con una matriz interna de identificación de 4X4. Se eligió este tipo de marcadores ya que son los marcadores con menor carga computacional al momento de generarlos.

La generación de estos marcadores se basa en el proceso mostrado en la Figura 2.24, para esto se utilizaron ciertos comandos especiales.

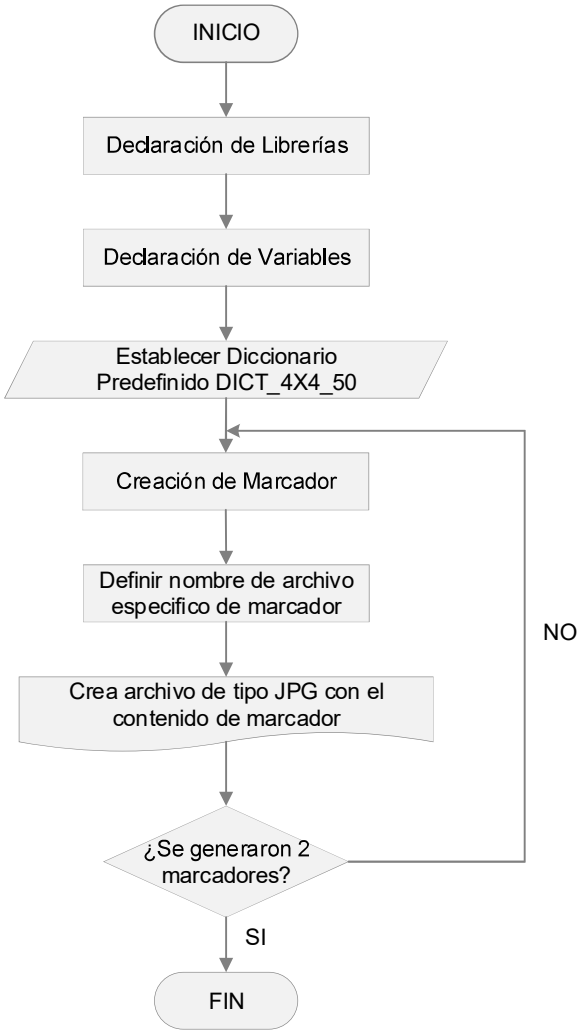


Figura 2.24. Generación de Marcadores de Referencia Matricial ArUco.

A continuación, se explican los comandos más importantes del proceso [53]:

- *getPredefinedDictionary.* – Mediante este comando se obtiene el contenido del diccionario elegido, se eligió DICT_4X4_50
- *drawMarker.* – Este comando permite dibujar los marcadores con su respectiva codificación binaria individual

- *imwrite.* – Permite crear el archivo de imagen tipo .JPG con el contenido de cada marcador de referencia matricial ArUco

El banco de pruebas planteado utiliza únicamente dos marcadores de este diccionario. Los marcadores utilizados se muestran en la Figura 2.25.

Figura 2.25. Marcador 0 y Marcador 1.

Como se indica en la Figura 2.26 (a) el marcador proporcionado tiene una matriz interna de 4x4 es decir 16bits, y en la Figura 2.26 (b) con un borde de 1 bit de ancho. La matriz interna, proporciona un código de 1's y 0's correspondiendo a los cuadros blancos y negros respectivamente.

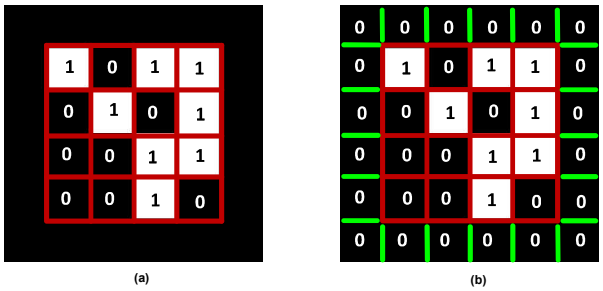


Figura 2.26. (a) Matriz Interna del Marcador 0 (b) Marcador con borde incluido

Para los marcadores escogidos, la codificación de las matrices internas se indica respectivamente en la Ecuación 2.1 y Ecuación 2.2.

$$Marcador\ 0 = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Ecuación 2.1. Matriz interna de Marcador 0

$$Marcador\ 1 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix}$$

Ecuación 2.2. Matriz interna de Marcador 1

Una vez generados los marcadores, antes del proceso de detección de los mismos es fundamental la calibración de la cámara. A continuación, se explica este proceso en base a la cámara Logitech C920 instalada en el sistema.

Calibración de cámara

En la sección de Calibración de cámara, se especificó que mediante el proceso de calibración se obtienen los parámetros necesarios para establecer las coordenadas 2D de la imagen capturada, además se indicó que el número de muestras necesarias para este proceso es de al menos 15. Para este fin, se eligió el patrón de tablero de ajedrez proporcionado por OpenCV. Este patrón tiene un tamaño predeterminado de 9x6 uniones en base a cada cuadro dentro del mismo.

Para la calibración de la cámara del proyecto, se imprimió el patrón mostrado en la Figura 2.27. Cada cuadro tiene una medida de 3.5cm, la cual es una medida adecuada para que la cámara detecte las uniones de las cuadrículas. El método de calibración de OpenCV busca las uniones de cada uno de los cuadros es por esto por lo que el tamaño del patrón es de 9x6 como se indica en la Figura 2.27. El comando *findChessboardCorners* [54], permite obtener las posiciones de cada una de estas uniones.

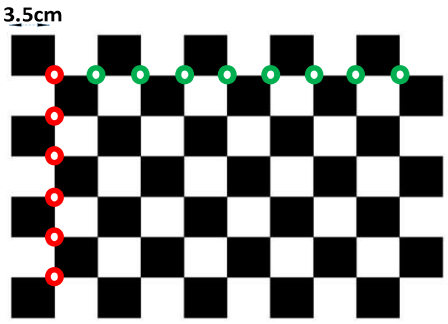


Figura 2.27. Patrón de Calibración provisto en OpenCV.

La calibración de la cámara se basa en tomar capturas del patrón en diferentes orientaciones. El número de muestras capturadas debe ser mayor a 15 y no superar las 50 muestras ya que, si bien se debe presentar un número considerable de muestras, también se debe evitar un exceso de carga computacional.

El proceso de calibración se resume en el diagrama de la Figura 2.28.

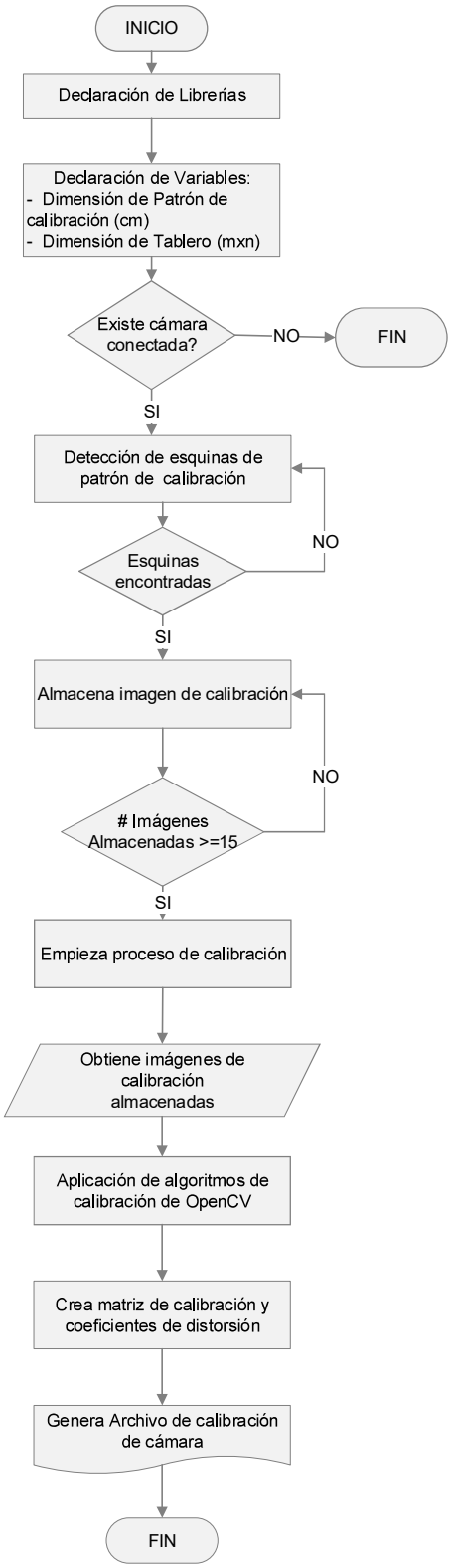


Figura 2.28. Proceso de calibración de cámara

El patrón puede ser colocado en distintas orientaciones, que desee el usuario como se indica en la Figura 2.29 (a). El comando *drawChessboardCorners* [54] permite dibujar las líneas entrelazadas entre cada unión como se indica en Figura 2.29 (b).

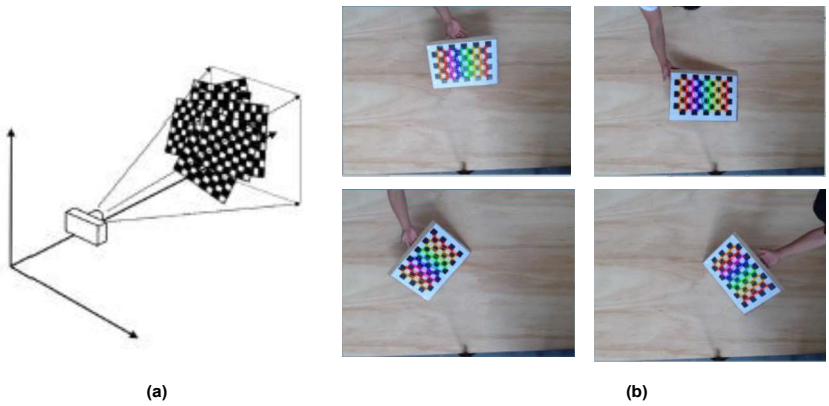


Figura 2.29. (a) Orientaciones del tablero de ajedrez (b) Líneas entrelazadas sobre patrón

El usuario puede acceder al programa de calibración de la cámara, el cual tiene el nombre de *calibracion_camara,* una vez puesto en marcha el programa, el usuario debe presionar la tecla *Espacio* para capturar las imágenes. Al acabar de capturar las imágenes del tablero en diferentes orientaciones, el usuario debe presionar la tecla *Enter* para guardar las capturas y el programa empieza automáticamente el proceso de extracción de parámetros de la cámara.

Modelo de la cámara

El modelo de la cámara es obtenido por medio del análisis de todas las capturas del patrón. Mediante el comando *calibratecamera* [32] es posible devolver al usuario los parámetros necesarios para formar la matriz de calibración y el vector de distorsión.

Los valores obtenidos para formar la matriz de calibración (Ecuación 2.3) y el vector de distorsión (Ecuación 2.4) son los siguientes:

$$Matriz\ de\ calibración = \begin{bmatrix} 509.977 & 0 & 3 \\ 0 & 511.092 & 4.5 \\ 0 & 0 & 1 \end{bmatrix}$$

Ecuación 2.3. Resultado de matriz de calibración.

$$coef\ Distorción = [-0.02405 \quad 9.75 * 10^{-5} \quad 0.005 \quad 0.005 \quad -9.84 * 10^{-5}]$$

Ecuación 2.4. Resultado de vector de coeficientes de distorsión.

Como fue explicado en la sección de Calibración de cámara, estos datos se requieren para ser manejados internamente en OpenCV. Con la ayuda de la matriz de calibración y el vector de coeficientes de distorsión, el momento que el usuario requiera las coordenadas de un objeto situado en el mundo real, podrá recibir las coordenadas 2D del objeto en el sistema de coordenadas propio de la cámara. Una vez terminada la calibración, es posible realizar el proceso de detección de los marcadores de referencia matricial.

Detección de marcadores de referencia ArUco

En la sección Detección de marcadores de referencia matricial, se especificó los pasos para la detección de los marcadores de referencia ArUco. Mediante el comando detectMarkers [39] se puede realizar este proceso especificando el cuadro capturado y el tipo de diccionario elegido. Este comando realiza la conversión de la imagen como se detalla a continuación. Para analizar la detección de los marcadores se realizaron pruebas sobre el entorno de la biblioteca de la Escuela Politécnica Nacional, con el fin de analizar la detección de un marcador ArUco dentro de un entorno con varios colores y formas que capturar. La persona se ubicó a una distancia de 70 centímetros del lente de la cámara, para observar de mejor manera las pruebas, sin embargo, como se detalla en el Capítulo 3, el sistema funciona de manera correcta a una distancia de 1.50 metros del lente de la cámara.

Conversión a escala de grises

El comando *detectMarkers* utiliza internamente el comando *CV_BGRA2GRAY* [55], que permite transformar la imagen de colores a una escala de grises, como se indica en la Figura 2.30.

Figura 2.30. Conversión a escala de grises en el entorno.

Binarización de imagen

El comando *DetectMarkers* permite realizar la binarización de la imagen a partir de la conversión de escala de grises mediante los siguientes comandos internos [56]:

- *threshold.* - Analiza la variación de intensidad del entorno capturado en base a un umbral de intensidad definido. *DetectMarkers* utiliza el valor de umbral por defecto de 125, si el valor de intensidad de un píxel es mayor a 125 se le asigna el color blanco, caso contrario si es menor a este valor se le asigna el color negro, como se indica en la Figura 2.31.

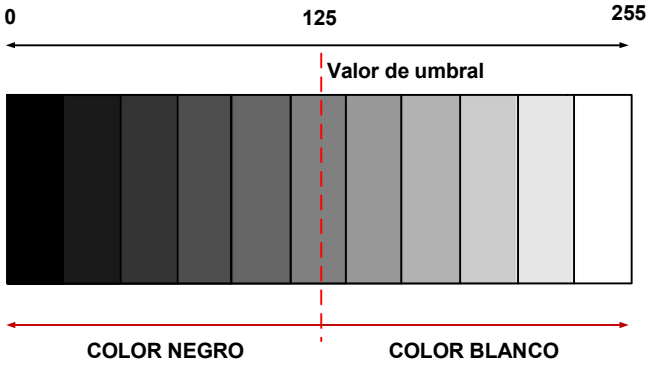


Figura 2.31. Conversión a Blanco y Negro

- *THRESH_BINARY.* - Este Código permite asignar los valores numéricos de 0 al color negro y de 1 al color blanco respectivamente.
- *THRESH_OTSU.* - Este comando permite corregir errores de análisis de intensidad. La binarización *otsu*, permite mejorar los valores de intensidad de los pixeles que se encuentran al margen del valor del umbral [57].

El proceso de la binarización realizado se indica en la Figura 2.32., donde es posible observar como la Figura 2.30 que se encontraba en escala de grises, se ha transformado en colores blanco y negro según su intensidad,

Figura 2.32. Binarización de imagen del entorno

Dibujo de contornos

A partir de la imagen binarizada, se analizan todos los contornos que existen en el entorno, pero las condiciones de iluminación en diferentes áreas generalmente son distintas, por lo que establecer un valor de umbral predeterminado no ayudaría en la

detección de contornos. Mediante el comando *AdaptiveThreshold* es posible analizar la imagen en distintas regiones, generando un valor de umbral de intensidad propio para cada región analizada.

Este valor de umbral de intensidad es calculado mediante el comando interno *ADAPTIVE_THRESH_GAUSSIAN_C* el cual realiza un ponderado de la intensidad alrededor de una ventana gaussiana en la región analizada [58]. Es necesario establecer el tamaño de la región donde se calcula el valor de la intensidad, el comando *AdaptiveThreshold* puede establecer valores de 3, 5 y 7 pixeles, *detectMarkers* utiliza el mayor valor de 7 pixeles para cada región analizada con el fin de analizar la mayor cantidad de pixeles. En la Figura 2.33 existen varios contornos ya sea los del marcador mostrado, los de la persona que sostiene el marcador, y los contornos del lugar. Estos son dibujados en color blanco.

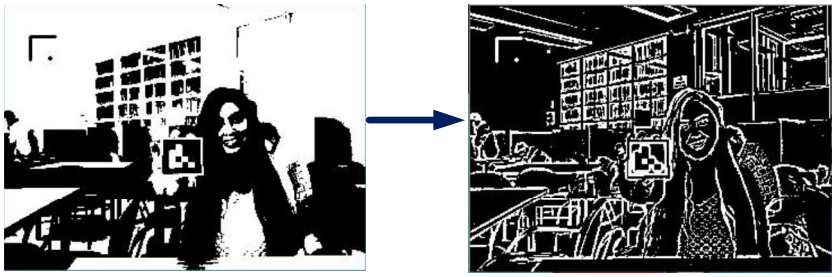


Figura 2.33. Detección de marcador de referencia mediante contornos en el entorno

Detección de Contornos

Una vez dibujados los contornos presentes, el comando interno que utiliza la librería para realizar el análisis de contornos es *findContours* [39]. Esta función retorna un vector de puntos que incluye cada contorno encontrado de la imagen binarizada. En base a la Figura 2.33, es posible analizar contornos como los del marcador de referencia, la persona que sostiene el marcador, la persona del fondo y el resto de carga en la imagen; el vector contiene la posición de todos los contornos encontrados expresado en pixeles.

Filtrado de Contornos

Para el filtrado de contornos es necesario analizar únicamente los contornos que formen un polígono, por lo que se realiza una aproximación poligonal a cada contorno encontrado. En la Figura 2.34 es posible identificar varios polígonos presentes en el entorno capturado, el presente análisis se enfoca únicamente en los polígonos que forman el marcador de referencia capturado, en base al cual, es posible analizar 5 polígonos encontrados.

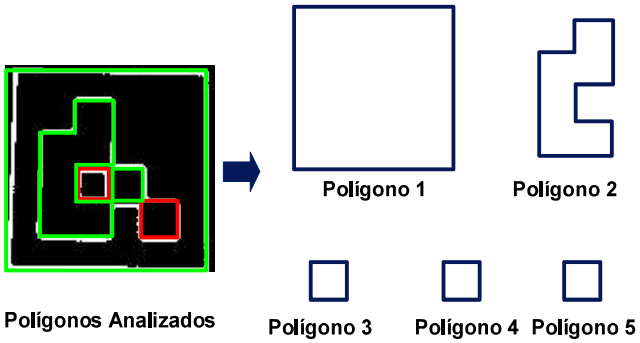


Figura 2.34. Polígonos analizados dentro de un marcador de referencia

Después de realizar el filtrado de contornos es necesario realizar una transformación de perspectiva con el objetivo de colocar los polígonos encontrados de manera paralela hacia el lente de la cámara, mediante el comando *getPerspectiveTransform* [39]. Al obtener la vista frontal de la imagen es posible visualizar de mejor manera los polígonos presentes, no es totalmente recto, pero forma la perspectiva de los polígonos presentes en una posición frontal para el análisis de matriz interna binaria. Posteriormente el comando *detectMarkers* realiza internamente el filtrado de contornos. Al recibir el vector de puntos de los contornos se realiza un análisis de perímetros y mediante el comando *minContourPointsAllowed* se excluye a los perímetros sumamente pequeños [59].

Después del análisis de perímetros es necesario considerar únicamente los polígonos que tengan 4 vértices mediante el comando interno *approxPolyDP* [37] es posible analizar únicamente los paralelepípedos encontrados, el resto serán descartados del análisis, por lo que en la Figura 2.34 únicamente serian analizados los polígonos 1,3,4 y 5 ya que el polígono 2 excede los 4 vértices. Una vez descartados los polígonos, es necesario analizar la codificación interna de cada uno para realizar una vez más la eliminación de posibles marcadores falsos.

Búsqueda de posibles marcadores

A partir del análisis de las candidatas a ser marcador como se indica en la Figura 2.34. *DetectMarkers* realiza los siguientes pasos:

1. *Valoración de imagen binaria.* - Si en la transformación de perspectiva se analiza un paralelepípedo, es posible realizar la codificación de la matriz interna. Gracias a la transformación de perspectiva es posible colocar la cuadrícula de análisis de codificación binaria sobre una imagen paralela al eje de la cámara como se indica en la Figura 2.35. Internamente se genera una cuadrícula de 4x4 predefinida que se sobrepone en los polígonos encontrados, es posible observar que la cuadrícula coincide únicamente en el polígono 1 excluyendo el borde

automáticamente por medio de programación, por lo que el resto será descartado automáticamente.

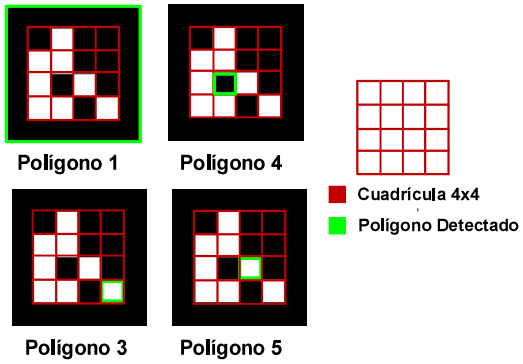


Figura 2.35. Análisis de candidatas de marcador de referencia

Una vez descartada las candidatas es posible analizar el contenido binario de la imagen mediante la binarización *otsu*. Esta binarización brinda los valores de la intensidad ya sean 0 o 1 cuando analiza el color negro o blanco respectivamente. A través del comando *perspectiveRemoveIgnoredMarginPerCell* [37] es posible analizar la cantidad de pixeles necesarios evitando chocar con márgenes o la mezcla de colores como se indica en la Figura 2.36, únicamente el contenido de los cuadros blancos será analizado.

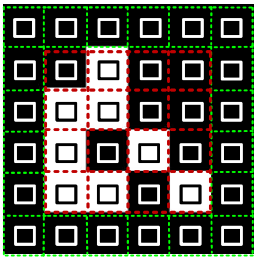


Figura 2.36. Eliminación de pixeles cercanos a bordes.

2. *Identificación de código de marcador.* – En este paso, el programa genera una cuadrícula de 4x4 como se indicó en la Figura 2.37 para analizar si el cuadrado mostrado es un marcador ArUco, y a su vez identificar si es el marcador 0 o 1. Si la cuadrícula coincide con la matriz interna se establece la presencia de un marcador ArUco, como en el caso del polígono 1 en la Figura 2.35.

Únicamente se analiza el contenido interno es decir la matriz binaria interna, sin tomar en cuenta el contenido de bits del borde, mediante este proceso es posible extraer la matriz de identificación de cada marcador. Tanto el marcador 0 y 1 tienen especificada su matriz interna la cual se debe leer de izquierda a derecha y de arriba

hacia abajo. En la generación de los marcadores, la librería ArUco establece la orientación base del marcador tomando como referencia la esquina derecha inferior como se indica en la Figura 2.37. Esta orientación da lugar a las Ecuaciones 2.1 y 2.2 para cada marcador respectivamente.

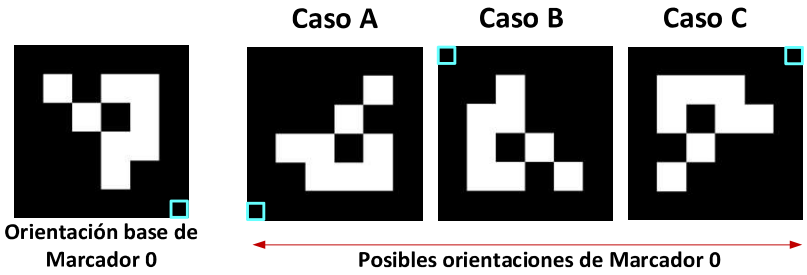


Figura 2.37. Orientaciones posibles de Marcador 0

Pero si el marcador no se encuentra en su orientación base, es posible que el programa observe tres codificaciones diferentes. Tomando como ejemplo el marcador 0 y en base la Figura 2.38 se tienen los siguientes tres casos.

$$Marcador\ 0 = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix}$$

Ecuación 2.5. Matriz de codificación Caso A

$$Marcador\ 0 = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{bmatrix}$$

Ecuación 2.6. Matriz de codificación Caso B

$$Marcador\ 0 = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

Ecuación 2.7. Matriz de codificación Caso C

Estas codificaciones son analizadas internamente, mediante el comando *hammDistMarker* [39] que permite analizar las posibles rotaciones de la matriz binaria base.

Finalmente, el marcador es identificado, por lo que se procede a analizar la posición del marcador sobre el plano. Para esto es necesario establecer la resolución de la cámara. Se estableció una resolución de 640 x 480 píxeles como se mencionó anteriormente. La

resolución determina el límite máximo de los ejes *X* y *Y* del sistema de coordenadas propio de la cámara; es decir, determina el límite máximo de captura de la cámara. Mediante los comandos [60]:

- `VideoCapture::set(`*CV_CAP_PROP_FRAME_WIDTH,640).*
- `VideoCapture::set(`*CV_CAP_PROP_FRAME_HEIGHT,480).*

Se establece los valores antes mencionados. La disposición de los ejes que brinda OpenCV son como se indica en la Figura 2.38.

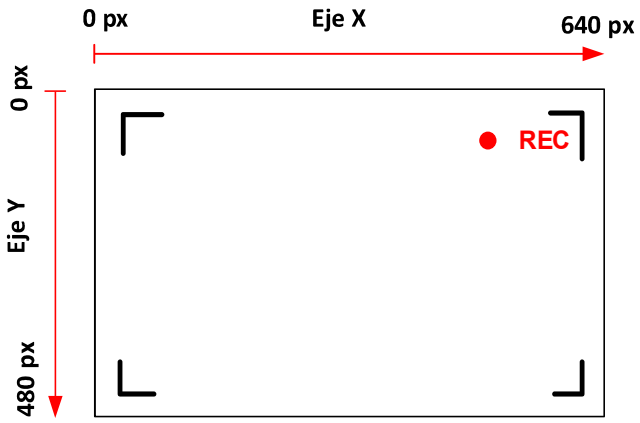


Figura 2.38. Límites de sistema de coordenadas de la cámara en pixeles.

Al detectar los marcadores 0 o 1, internamente las librerías ArUco puede ubicar la posición de las 4 esquinas en coordenadas 2D expresada en pixeles correspondientes al sistema de coordenadas de la cámara.

El presente proyecto, para mayor facilidad, realiza dos cambios importantes sobre la información de posición marcador de referencia capturado, estos son:

- Para entender el movimiento del objeto en el entorno capturado se invirtió el eje Y dispuesto por OpenCV como se indica en la Figura 2.39.
- Para el desarrollo del controlador se busca la posición del centro del marcador de referencia expresado en pixeles, la cual se asume que es el centro de masa del robot como se indica en la Figura 2.39.

La Figura 2.39 se encuentra el marcador 0 donde, sus cuatro esquinas son analizadas. La esquina inferior izquierda, por ejemplo, se encuentra ubicada en las coordenadas (462,364,0) correspondientes a los ejes *X, Y, Z* respectivamente. La última coordenada es 0 ya que no se trabaja con el eje *Z*. Por lo tanto, se realizó un análisis de la posición de las cuatro esquinas, y se calculó el centro del marcador, el cual será expresado en

pixeles y únicamente con coordenadas en *x* y *y*. Además, es posible observar el ángulo de orientación del marcador respecto al centro del marcador correspondientes a los ejes *X'* y *Y'*.

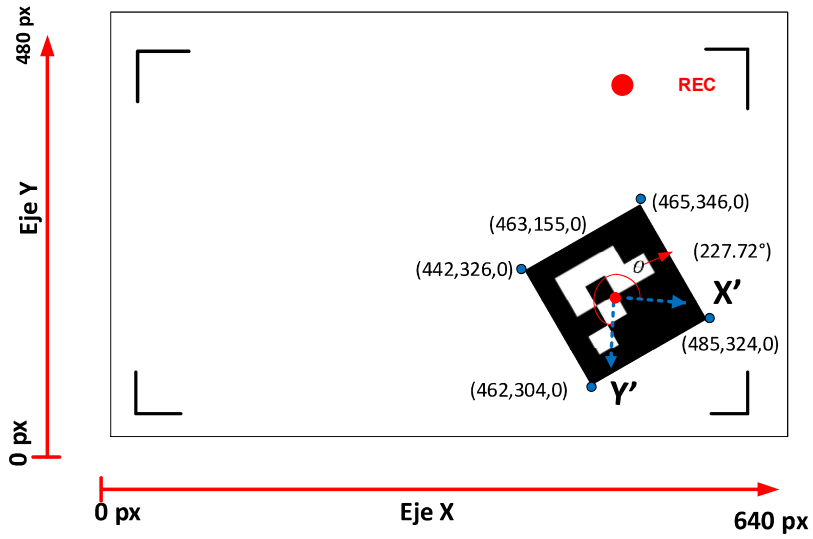


Figura 2.39. Ubicación de marcador de referencia en coordenadas 2D.

Para dibujar los números y ejes propios de cada marcador no basta con la ubicación 2D para eso se necesita conocer la posición del objeto en el mundo real respecto a la posición de la cámara.

Estimación de posición y orientación del marcador

Como se mencionó en la sección de Técnicas de visión por computadora, la realidad aumentada busca la unión del mundo virtual y mundo real. Al colocar el marcador en el mundo 3D es necesario conocer la ubicación del objeto respecto a la posición de la cámara. Mediante el comando *estimatePoseSingleMarkers* se obtiene la pose del objeto en 3D [37], la cual determina las coordenadas *x*, y *y* el ángulo de orientación del marcador respecto al centro del marcador (eje *X'* y eje *Y' ejes respecto al centro del marcador*).

Tomando como referencia la Figura 2.40, el punto C es el centro de la cámara, los puntos P1-P4 son puntos 3D y los puntos P1'-P4' son sus proyecciones en el plano de la cámara.

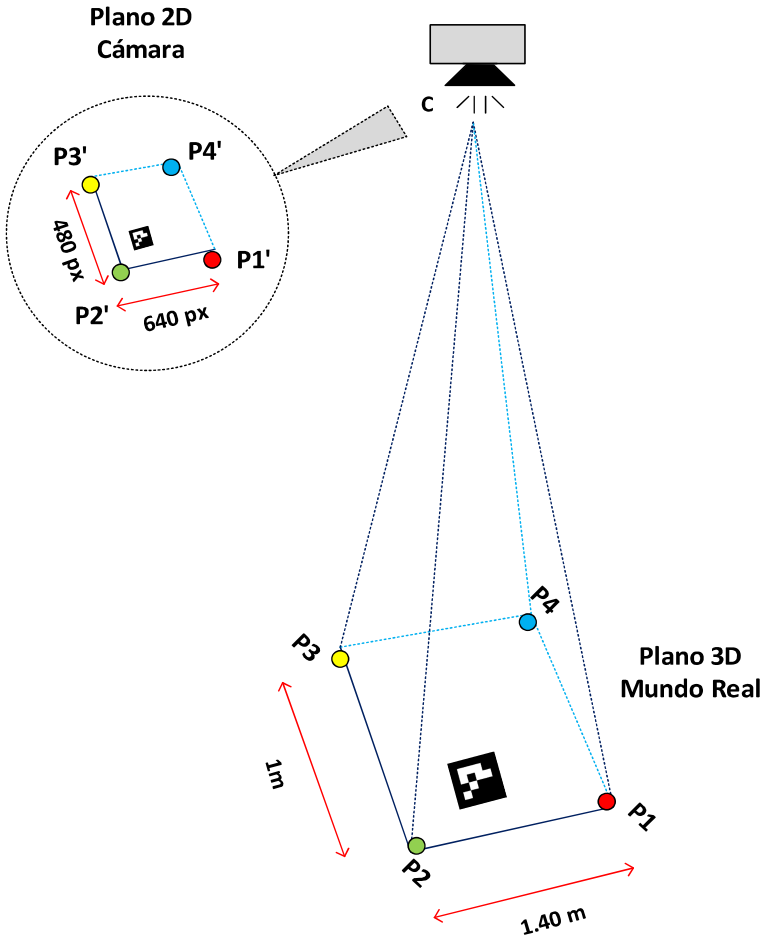


Figura 2.40. Proyección de cámara en mundo real.

Usando la matriz de calibración y vector de coeficientes de distorsión el comando permite transformar la ubicación del marcador presente sobre la base en las coordenadas 2D expresadas en pixeles, es decir las coordenadas de homografía y el ángulo de orientación.

Las coordenadas de homografía son expresadas en metros para ser utilizadas en el desarrollo del controlador, a continuación, se establece el procedimiento para obtener la equivalencia de pixeles a metros.

- Se colocó un elemento con dimensiones conocidas en el área de trabajo establecida, en nuestro caso se usó una regla de 30 cm.

- Mediante el uso de Visual Studio se obtuvo la posición del puntero mouse que generado en la ventana de visualización; esto nos permitió conocer la posición en X y Y de la regla colocada en el área de trabajo con respecto a las coordenadas de la cámara (pixeles).
- Usando el cursor mouse se tomó la posición de los extremos de la regla que se colocó en el área de trabajo, los valores en pixeles se visualizaron en la ventana generada por Visual Studio como se observa en la Figura 2.41.
- Con los valores obtenidos se procedió a sacar la medida de la regla en pixeles mediante la diferencia de las posiciones de los extremos de la regla.

Figura 2.41. Equivalencia de medida de la regla en pixeles.

- Una vez obtenido el valor en pixeles del elemento (regla) que se colocó en el área de trabajo; se realizó una regla de tres simple con la finalidad de conocer la equivalencia aproximada de un pixel en metros. Como se muestra en la Ecuación 2.8, X representa la equivalencia de $1\ [px]$ en metros.

$$X = \frac{0.3[m] * 1\ [px]}{132\ [px]} = 0.0023\ [m]$$

Ecuación 2.8. Conversión de pixeles a metros.

El ejemplo mostrado en la Figura 2.39 es analizado en pixeles, utilizando la Ecuación 2.8 se convierten las coordenadas expresadas en pixeles a metros, como se indica en la Figura 2.42 y estos datos son utilizados en el desarrollo del controlador.

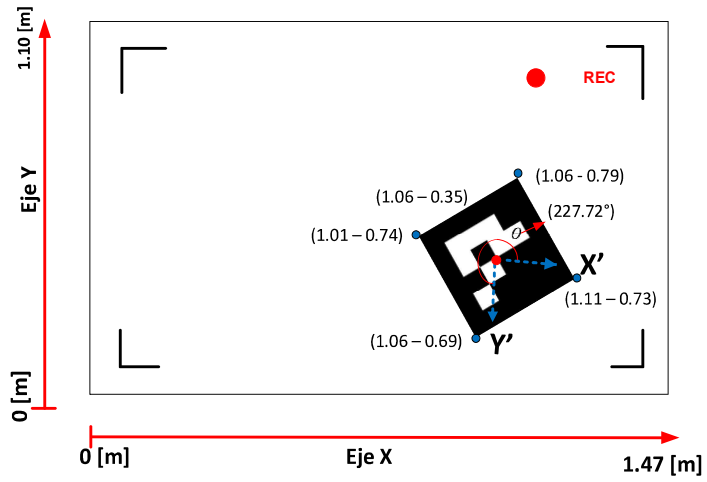
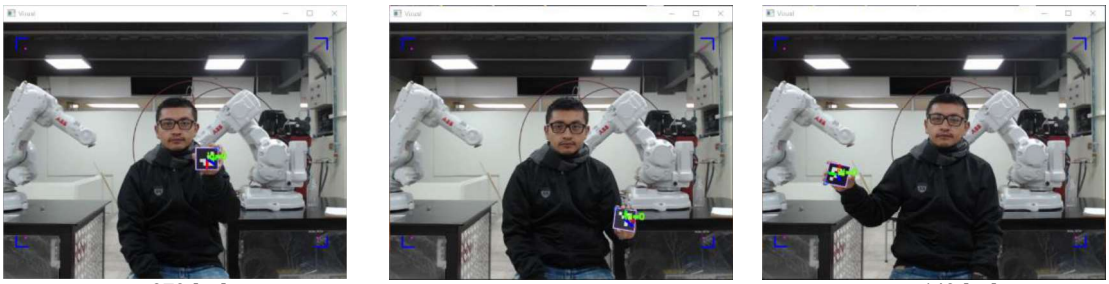


Figura 2.42. Aproximación de pixeles a metros en el espacio de trabajo.

Además, el comando *estimatePoseSingleMarkers* permite dibujar los ejes de color azul (eje *Z*), rojo (eje *X*) y verde (eje *Y*) con su respectivo identificador numérico sobre el marcador, incluso cuando este está en movimiento.

La Figura 2.43 permite visualizar el eje dibujado y el número del marcador correspondiente. En la Figura 2.43 (a) el marcador es ubicado paralelo al lente de la cámara, en la Figura 2.43 (b) el marcador es ubicado con un ángulo de orientación de 83.57°. Tanto en la Figura 2.43 (b) y (c) la persona coloca el marcador con distintos ángulos de inclinación frente al lente de la cámara, pese a que la persona lo ubica en cualquier inclinación, los ejes se reorientan y el identificador numérico se mantiene.



x: 379 [px]
y: 225 [px]
Ángulo= 0 [deg]
(a)

x: 438 [px]
y: 112 [px]
Ángulo= 83.57 [deg]
(b)

x: 140 [px]
y: 196 [px]
Ángulo= 168.23 [deg]
(c)

Figura 2.43. Marcador de Referencia matricial detectado en el entorno con ejes propios e identificador

Como es posible observar se sostiene el marcador de manera vertical, luego lo gira hacia la derecha 90 grados, y finalmente se lo ubica de manera horizontal mediante esto

se comprueba que la visualización no se ve afectada ante diferentes orientaciones del marcador. El proceso de detección de los marcadores explicado anteriormente se encuentra resumido en el diagrama de flujo en la Figura 2.44.

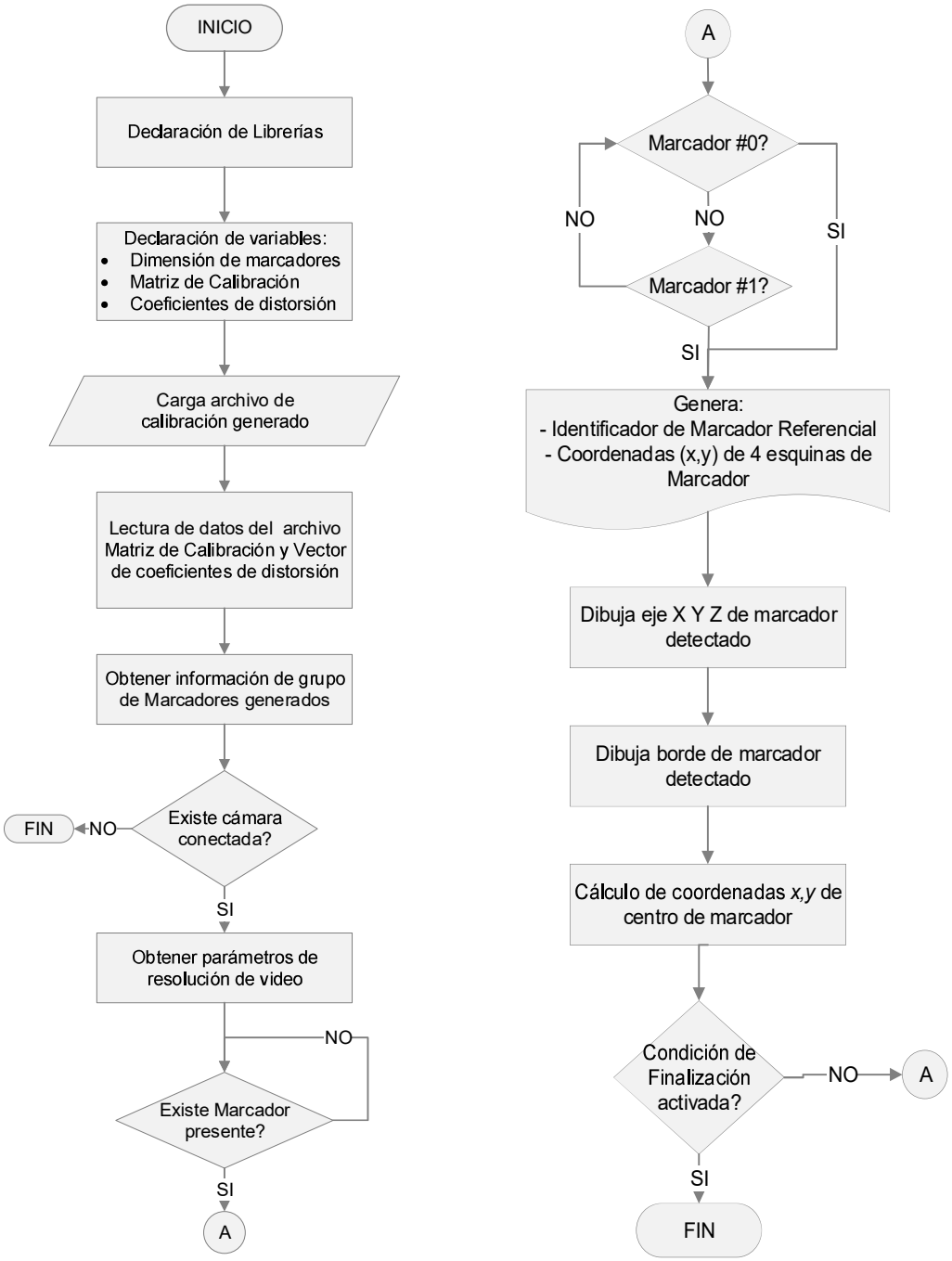


Figura 2.44. Proceso de detección de Marcadores 0 y 1

A partir de la obtención de la posición del centro del marcador, su identificador y ángulo es posible enviar esta información para el desarrollo del controlador, por lo que se utilizó las librerías de Matlab Engine compatibles con Visual Studio 2015.

Matlab Engine

La biblioteca de análisis de gráficos que proporciona Matlab está compuesta por muchas funciones altamente integradas, que permiten generar un resultado favorable para el usuario, por lo que es necesario enviar la información generada en el entorno de Visual Studio 2015 hacia Matlab.

Matlab Engine permite abrir la aplicación de escritorio de Matlab, ya que al inicializarlo permite visualizar una ventana que contiene funciones básicas como operaciones y gráficos sencillos. Al escribir la palabra *"desktop"*, se accede a la versión completa de Matlab R2018a. Como se indica en la Figura 2.45 Visual Studio 2015 y OpenCV únicamente envían los datos necesarios hacia Matlab R2018a. Para el presente proyecto, Matlab no necesita enviar ningún dato hacia Visual Studio 2015.

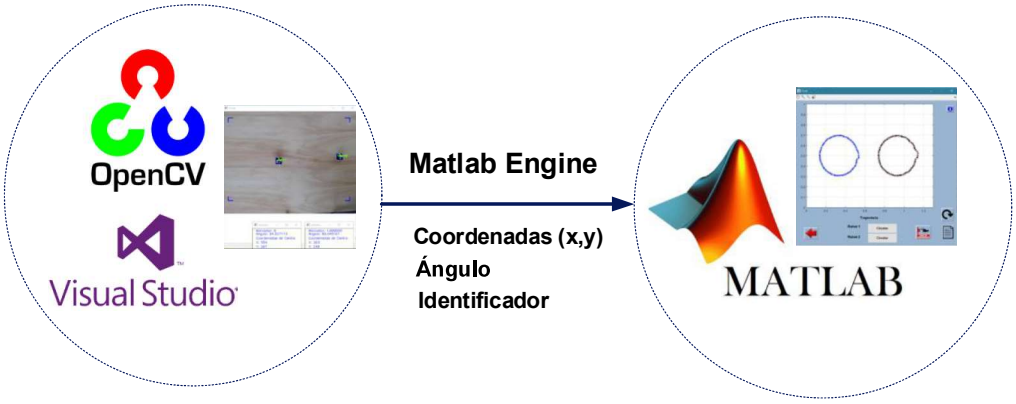


Figura 2.45. Esquema de envío de datos a través de Matlab Engine

Para este procedimiento se realizan los siguientes pasos [61]:

- Abrir diálogo de Matlab
- Enviar cadena de información
- Leer datos en el entorno de Matlab
- Escribir datos en el entorno de Matlab

A continuación, en la Figura 2.46, se encuentra detallado el proceso de envío de información a Matlab utilizado en el proyecto.

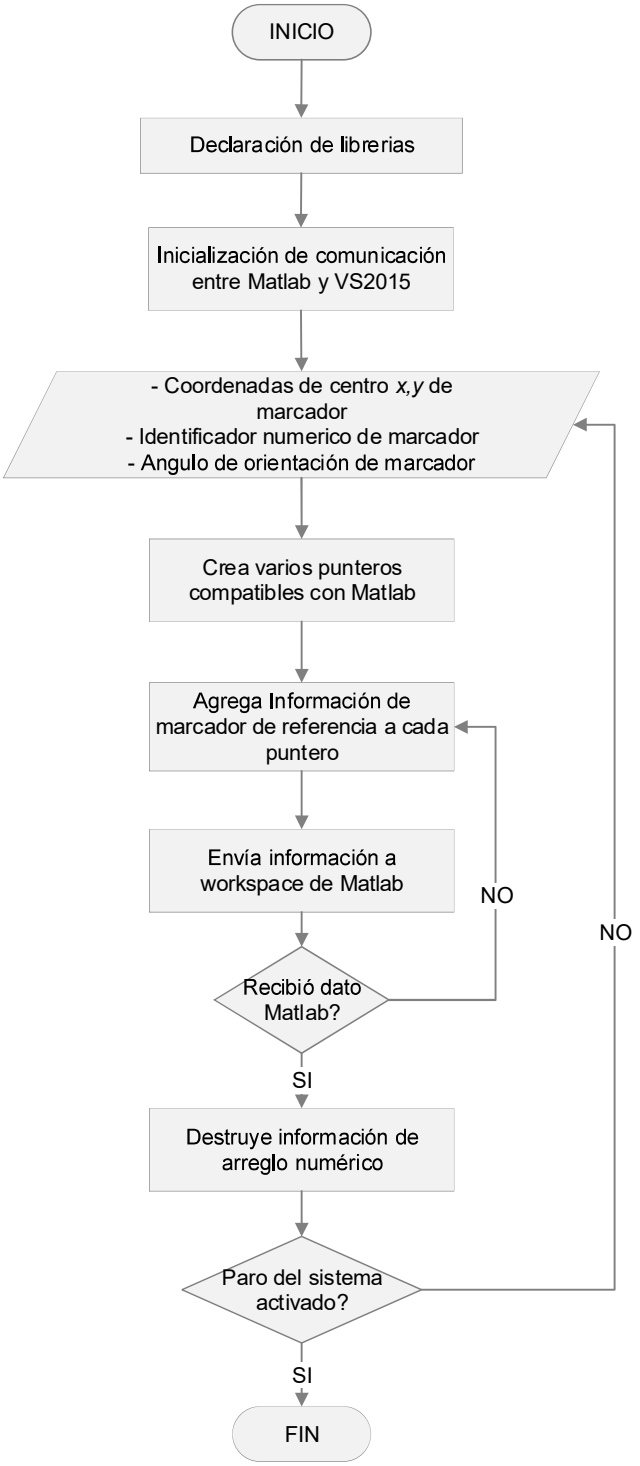


Figura 2.46. Proceso de envío de datos a Matlab.

Los comandos utilizados para el envío de la información necesario son los siguientes:

- *engOpen.* – Permite iniciar Matlab Engine, crea un puntero de apertura que, en el caso de falla, retorna vacío a C++ [62]

- *mxArray.* – Este tipo de objeto es una matriz que permite que Matlab realice el intercambio de datos entre Matlab y Visual Studio 2015 C++ [63].
- *mxCreateDoubleMatrix.* – Función que permite crear arreglos compatibles con Matlab especificando su tamaño y tipo [64].
- *memcpy y engPutVariable.* - Mediante este comando se puede copiar la variable al entorno de Matlab Engine con su respectivo nombre y contenido [65].
- *mxDestroyArray.* – El limite transferencia de datos es 2GB, por lo que, para evitar errores, es mejor eliminar el contenido en cada envió. Este comando permite borrar el contenido cuando ya ha sido utilizado [66].
- *mxGetPr.* – Obtiene los elementos de MxArray [67].

Cabe mencionar que la Escuela Politécnica Nacional permite a los estudiantes acceder a la licencia de activación de Matlab, la utilización de las librerías de Matlab Engine requiere dicha licencia para su correcto funcionamiento.

Terminada la explicación respecto al envío de información de los marcadores de referencia, se procede a detallar el diseño del controlador.

2.3 Sistema de control

El sistema de control está diseñado para comandar los dos robots móviles presentados en la sección de Implementación de los robots móviles por lo que, el algoritmo de control implementado para cada robot se basa en el mismo diseño que se indica a continuación.

Diseño del algoritmo de control

Antes de diseñar el controlador, es necesario conocer el comportamiento del robot que se espera obtener. Inicialmente el robot móvil se encontrará en una posición inicial con un ángulo de orientación, por lo que su estado es (x_m, y_m, θ_m).

Luego el robot debe pasar al estado deseado de referencia $(x_{ref_m}, y_{ref_m}, \theta_{ref_m})$ en un instante t_m, posteriormente alcanzará la referencia en $(x_{ref_{m+1}}, y_{ref_{m+1}}, \theta_{ref_{m+1}})$ para un tiempo t_{m+1}, como se muestra en la Figura 2.47.

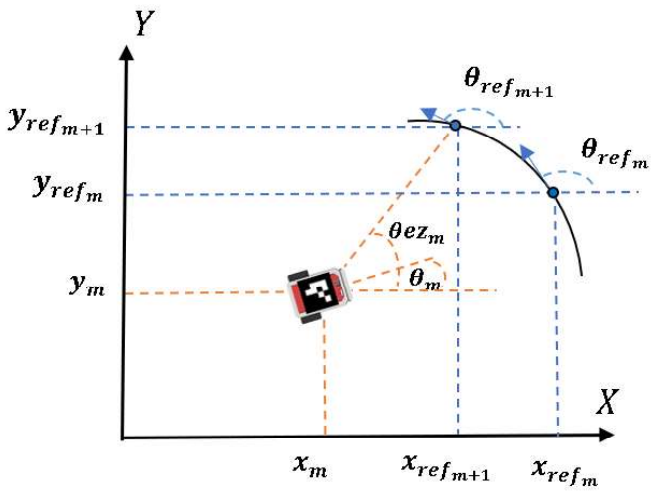


Figura 2.47. Representación del comportamiento del robot móvil.

Para el diseño del controlador se procede a trabajar con las variables descritas en la sección Modelo matemático del robot uniciclo. Aplicando la aproximación de Euler en el modelo del robot representado por la Ecuación 2.9, se obtiene el modelo matemático en tiempo discreto en la Ecuación 2.10.

$$\begin{cases} \dot{x} = v\,cos(\theta) - a\omega\,sin(\theta) \\ \dot{y} = v\,sin(\theta) + a\omega\,cos(\theta) \\ \dot{\theta} = \omega \end{cases}$$

Ecuación 2.9. Modelo matemático del robot móvil.

$$\frac{1}{T_O}\begin{bmatrix} x_{m+1} - x_m \\ y_{m+1} - y_m \\ \theta_{m+1} - \theta_m \end{bmatrix} = \frac{1}{T_O}\begin{bmatrix} \Delta x_m \\ \Delta y_m \\ \Delta\theta_m \end{bmatrix} = \begin{bmatrix} cos(\theta_m) & -a\,sin(\theta_m) \\ sin(\theta_m) & a\,cos(\theta_m) \\ 0 & 1 \end{bmatrix}\begin{bmatrix} v_m \\ \omega_m \end{bmatrix}$$

Ecuación 2.10. Modelo matemático en tiempo discreto.

Donde T_O es el tiempo de muestreo del sistema. El modelo matemático en tiempo discreto dado en la Ecuación 2.10 se lo puede condensar de la forma dada en la Ecuación 2.11.

$$\vec{b} = A\vec{u}$$

Ecuación 2.11. Representación del sistema lineal.

En donde:

$$\vec{b} = \frac{1}{T_O}\begin{bmatrix} \Delta x_m \\ \Delta y_m \\ \Delta\theta_m \end{bmatrix}$$

$$A = \begin{bmatrix} cos(\theta_m) & -a\ sin(\theta_m) \\ sin(\theta_m) & a\ cos(\theta_m) \\ 0 & 1 \end{bmatrix}$$

$$\vec{u} = \begin{bmatrix} v_m \\ \omega_m \end{bmatrix}$$

Posteriormente se procede a resolver la Ecuación 2.11 con la finalidad de dar solución al sistema y encontrar las acciones de control v_m (velocidad lineal) y ω_m (velocidad angular). El procedimiento matemático y los conceptos de álgebra aplicados para obtener estas acciones de control se detallan en [14], permitiendo obtener la Ecuación 2.12.

$$\begin{bmatrix} v_m \\ \omega_m \end{bmatrix} = \frac{1}{T_O} \begin{bmatrix} \Delta x_m\ cos(\theta_m) + \Delta y_m\ sin(\theta_m) \\ \dfrac{-a\Delta x_m\ sin(\theta_m) + a\Delta y_m\ cos(\theta_m) + \Delta\theta_m}{(a^2+1)} \end{bmatrix}$$

Ecuación 2.12. Acciones de control del robot móvil.

Con la finalidad de encontrar una solución al sistema de la Ecuación 2.11 se debe cumplir la restricción dada por la Ecuación 2.13. En donde θez_m representa la orientación necesaria al tiempo t_{m+1}, que hace que el robot móvil tienda hacia la trayectoria de referencia. (Figura 2.47).

$$\theta ez_m = \tan^{-1}\left(\frac{y_{ref_m} - y_m}{x_{ref_m} - x_m}\right)$$

Ecuación 2.13. Restricción del sistema (Ángulo deseado).

De esta manera el controlador propuesto viene dado por la Ecuación 2.14, en el cual se utiliza el valor θez_m en lugar de θ_m.

$$\begin{bmatrix} v_m \\ \omega_m \end{bmatrix} = \frac{1}{T_O} \begin{bmatrix} \Delta x_m\ cos(\theta ez_m) + \Delta y_m\ sin(\theta ez_m) \\ \dfrac{-a\Delta x_m\ sin(\theta ez_m) + a\Delta y_m\ cos(\theta ez_m) + \Delta\theta_m}{(a^2+1)} \end{bmatrix}$$

Ecuación 2.14. Acciones de control implementadas.

Para lograr que el margen de error disminuya suavemente los estados x_{m+1}, y_{m+1} y θ_{m+1} deben ser igual a las Ecuaciones 2.15, 2.16 y 2.17 respectivamente, validado en [14].

$$x_{m+1} = x_{ref_{m+1}} - k_x(x_{ref_m} - x_m)$$

Ecuación 2.15. Estado de x en t_{m+1}.

$$y_{m+1} = y_{ref_{m+1}} - k_y(y_{ref_m} - y_m)$$

Ecuación 2.16. Estado de y en t_{m+1}.

$$\theta_{m+1} \quad = \theta ez_m - k_\theta(\theta ez_m - \theta_m)$$

Ecuación 2.17. Estado de θ en t_{m+1}.

A partir de las Ecuaciones anteriormente indicadas, las variaciones Δx_m, Δy_m y $\Delta\theta_m$ se las representan mediante las Ecuaciones 2.18, 2.19 y 2.20.

$$\Delta x_m = x_{ref_{m+1}} - x_m - k_x(x_{ref_m} - x_m)$$

Ecuación 2.18. Variación en x.

$$\Delta y_m = y_{ref_{m+1}} - y_m - k_y(y_{ref_m} - y_m)$$

Ecuación 2.19. Variación en y.

$$\Delta\theta_m = \theta ez_m - \theta_m - k_\theta(\theta ez_m - \theta_m)$$

Ecuación 2.20. Variación en θ.

Las variables k_y, k_x y k_θ son las constantes de calibración del controlador de Euler. Los valores que pueden tomar las constantes se encuentran en un intervalo abierto entre 0 y 1. Cabe mencionar que al obtener el valor de θez_m, este puede ser positivo o negativo; de tal manera que mediante programación en Matlab se crea la función *fun_angle* que se adjunta en el archivo digital del proyecto.

Esta función permite hacer el análisis por cuadrantes y obtener el menor valor de ángulo de giro que debe realizar el robot; *fun_angle* utiliza la función *atan2* de Matlab que permite calcular el valor del ángulo en radianes considerando los cuatro cuadrantes. Si los ángulos se encuentran en los dos primeros cuadrantes la función atan*2* mide el ángulo en sentido antihorario, y si el ángulo se encuentra en los otros dos cuadrantes el ángulo se toma en sentido horario. A continuación, se indica las consideraciones necesarias para la implementación del controlador.

Implementación del controlador

En el presente proyecto la implementación del sistema de control se desarrolla en el entorno del software Matlab. Para relacionar las acciones de control representadas en la Ecuación 2.14 con las velocidades de las ruedas del robot móvil, se consideran las Ecuaciones 1.2 y 1.3 (sección de Modelo matemático del robot uniciclo) para obtener las acciones de control para las ruedas derecha e izquierda dadas por las Ecuaciones 2.21 y 2.22 respectivamente.

$$\Omega_{R_m} = \frac{(2v_m + \omega_m L)}{2R}$$

Ecuación 2.21. Acciones de control rueda derecha.

$$\Omega_{L_m} = \frac{(2v_m - \omega_m L)}{2R}$$

Ecuación 2.22. Acciones de control rueda izquierda.

Para realizar el control de los minimotores DC se utiliza la técnica de control PWM (Modulación de ancho de pulso) como se observa en la Figura 2.48.

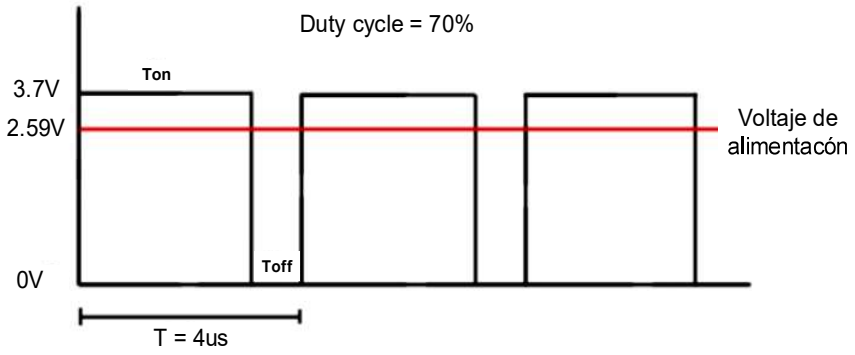


Figura 2.48. Representación del control PWM de los motores. (Duty cycle = 70%).

A pesar de no existir un control de velocidad, puede estimarse una velocidad máxima de las ruedas de acuerdo con las especificaciones de los motores (sección de Componentes electrónicos), de igual manera se midió la velocidad de las ruedas mediante el uso de un tacómetro para corroborar que las consideraciones que se exponen a continuación tienen concordancia.

Se conoce que a un voltaje de alimentación de 4.5 voltios los motores giran a 150 RPM (especificaciones dadas por el fabricante) equivalente a una velocidad angular de 15.7 rad/s. En base a esto determinó que para una alimentación de 3.7 voltios los motores giran aproximadamente a 123 RPM equivalente a 13 rad/s, esta velocidad angular se obtuvo de manera teórica y experimental (medición mediante tacómetro).

Ya que es necesario conocer la velocidad lineal máxima con la que se desplazará el robot móvil en el área de trabajo se utilizó la Ecuación 1.2 para calcular dicha velocidad, dando como resultado una velocidad lineal del robot de 0.21 m/s que se obtiene cuando los motores giran 123 RPM. Si bien 0.21 m/s es la velocidad máxima que puede alcanzar el robot móvil, se trabajó con velocidades inferiores a esta para que el desplazamiento del robot se dibuje de manera adecuada en el sistema de visualización y de esta forma también se consiguió que los robots trabajen a bajas velocidades acorde al modelo cinemático.

Los límites de velocidades con los cuales los robots recorrieron las trayectorias dentro del área de trabajo establecida se indicarán en base a pruebas realizadas detalladas en la sección de Resultados y Discusión.

En la Figura 2.49 se indica la implementación del algoritmo control de los robots móviles (Matlab) y en la Figura 2.50 se muestra la interpretación de las acciones de control de las ruedas en los robots uniciclos (Fio V3).

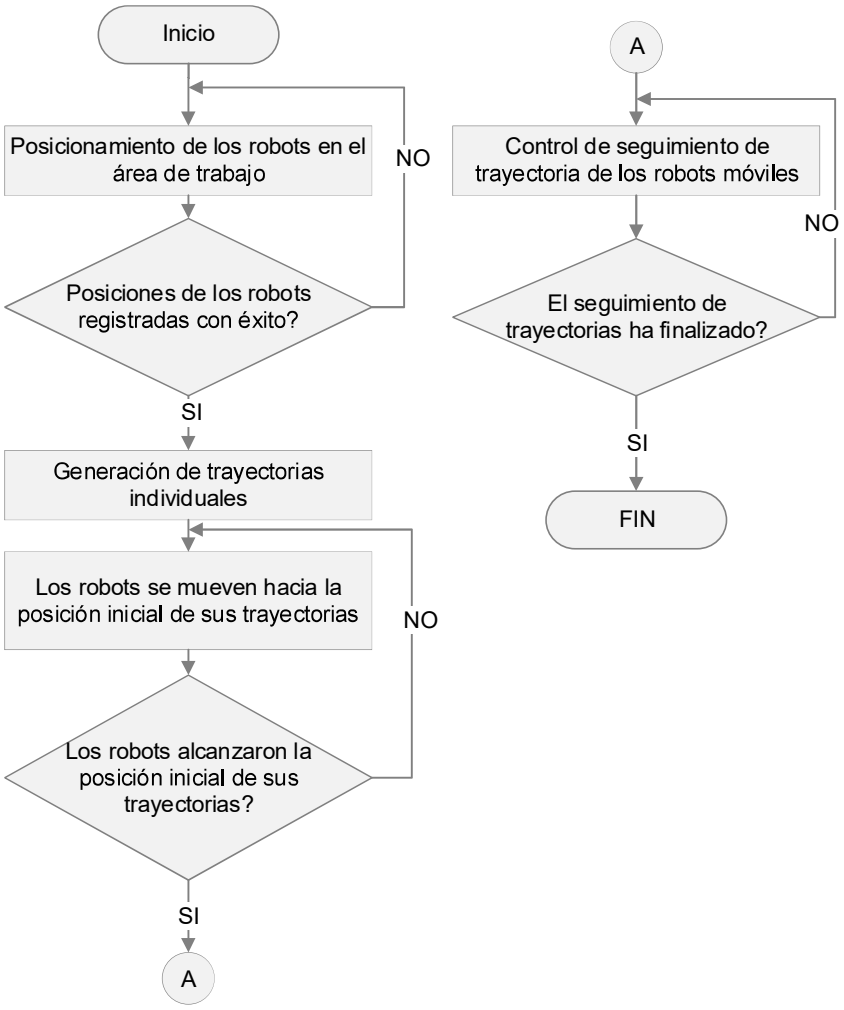


Figura 2.49. Algoritmo de control implementado en Matlab.

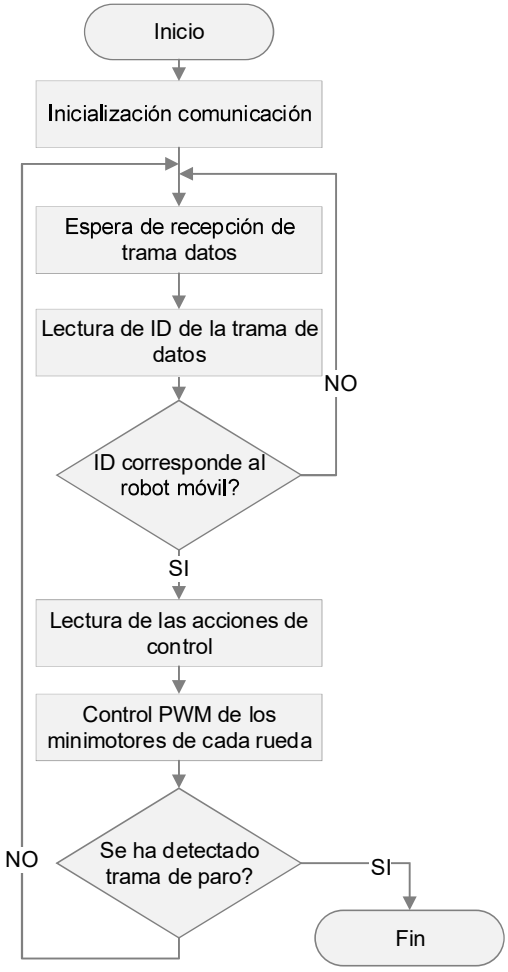


Figura 2.50. Acciones de control para las ruedas de los robots uniciclos.

Otro de los aspectos que se toman en cuenta al momento implementar el controlador es la generación de las trayectorias de referencia que debe seguir el robot móvil como se indica en la siguiente sección.

Generación de trayectorias

La generación de las trayectorias que realizarán los robots móviles en el banco de pruebas propuesto en este proyecto se basa en [14]. Cada una de las trayectorias depende del vector tiempo, a partir del cual se generan dos vectores adicionales de las componentes en los ejes X y Y del plano cartesiano. El número de elementos del vector tiempo se los obtiene mediante la Ecuación 2.23.

$$n = \frac{\tau}{T_0}$$

Ecuación 2.23. Número de elementos del vector tiempo.

En donde:

$\tau =$ Tiempo de duración del experimento.

$T_0 =$ Tiempo de muestreo.

Las ecuaciones paramétricas que nos permiten describir las trayectorias en el plano XY se indican en la Tabla 2.3.

Tabla 2.3. Ecuaciones Paramétricas de las trayectorias.

Trayectoria	Componente X	Componente Y
Lineal (pendiente = 0)	$\vec{x} = v\vec{t}$	$\vec{y} = \vec{0}$
Lineal (pendiente = 45°)	$\vec{x} = v\vec{t}$	$\vec{y} = v\vec{t}$
Circular	$\vec{x} = rcos(\frac{v}{r}\vec{t})$	$\vec{y} = rsin(\frac{v}{r}\vec{t})$
Doble Frecuencia (Ocho)	$\vec{x} = rcos(\frac{v}{r\sqrt{5}}\vec{t})$	$\vec{y} = rsin(\frac{2v}{r\sqrt{5}}\vec{t})$

Donde:

$v =$ Velocidad lineal.

$\vec{t} =$ Vector tiempo.

$r =$ Radio de la trayectoria.

Debido a que las acciones de control se envían desde la computadora hacia los dos robots móviles, se debe conocer la configuración de la red de comunicación y de los elementos que la conforman tomando en cuenta el protocolo de comunicación ZigBee revisado en la sección de Protocolo de comunicación ZigBee.

2.4 Sistema de comunicación

La topología ZigBee que se presenta en el presente proyecto es de tipo estrella. Para establecer esta topología se usan tres módulos de comunicación XBee Serie 1 donde uno de los módulos se conecta a la computadora donde está la interfaz de usuario para que trabaje en modo coordinador y otros en los robots los cuales son los dispositivos finales. En las siguientes secciones de describe el modo de configuración de los módulos para que el sistema de comunicación funcione sin problemas.

Configuración de los módulos XBee

La comunicación se la puede calificar como exitosa siempre y cuando a los módulos XBee se los configure de manera correcta. Para la configuración de cada módulo es necesario el uso del software XCTU que se lo pude descargar de manera gratuita de su página oficial [68]. En la Figura 2.51 se observa la primera ventana del software de programación de los módulos que se divide en dos paneles, uno de reconocimiento y otro de configuración.

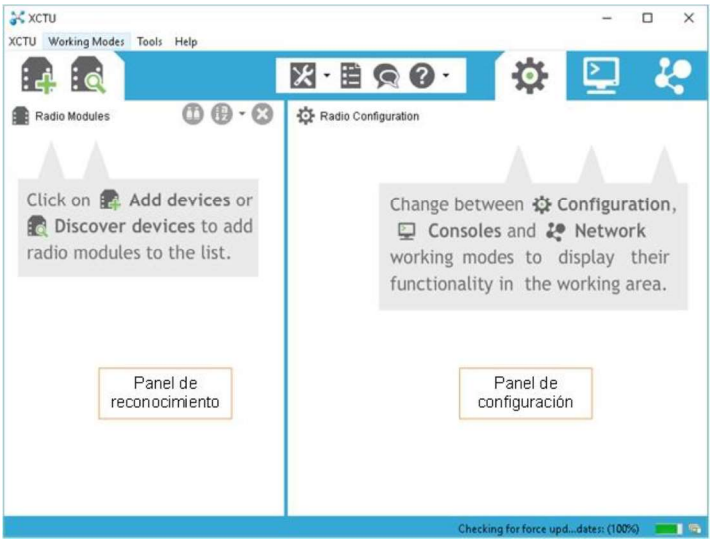


Figura 2.51. Ventana principal del software XCTU.

Para configurar a cada uno de los módulos XBee es necesario disponer de algún dispositivo que nos permita conectarlos a la computadora y acceder a las opciones de programación del software XCTU. Para el proyecto usaremos el módulo XBee Explorer que se muestra en la Figura 2.52, que permite conectar un XBee Serie 1 directamente al puerto USB de la computadora.

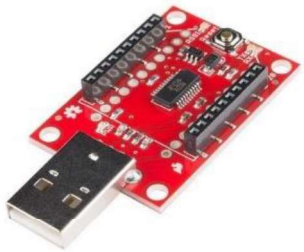


Figura 2.52. Módulo XBee Explorer. [69]

Una vez conectado el módulo Xbee a la computadora, dentro del software XCTU mediante el manejo de las pestañas del panel de reconocimiento se puede acceder a las configuraciones del puerto por el cual se realizará la comunicación. En nuestro sistema de

comunicación se usa el puerto serie número 7 (COM7). Los parámetros de configuración del puerto se presentan en la Figura 2.53 los mismos que son predeterminados por el software XCTU; considerando una velocidad de comunicación de 9600 baudios, 8 bits de datos, sin bit de paridad y 1 bit de parada. Los parámetros antes mencionados son utilizados para la configuración del puerto serie considerando a su vez las características técnicas de la tarjeta Fio V3 – Atmega 32U4 (sección de Componentes electrónicos). Una vez que se configure el puerto, se da clic en el botón finalizar para añadir el dispositivo.

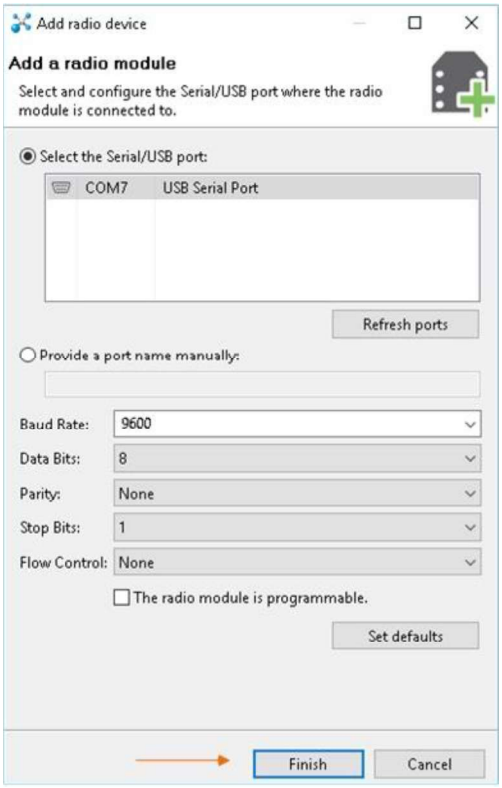


Figura 2.53. Configuración de puerto en XCTU.

Existen dos modos de operación dentro de una red ZigBee, modo transparente (AT) y API [70]. El primero es sencillo de configurar y nos permite trabajar en el envío y recepción de datos usando el puerto UART, en cambio el modo API (siglas en inglés Interface Programming Application) no se lo considera para el proyecto de titulación debido a su complejidad de configuración. Los módulos al trabajar en modo transparente deben tener:

- el mismo nombre de red (PAN ID) y canal (CH); y
- las direcciones de destino (DH y DL).

Cabe resaltar que se configurarán los parámetros antes mencionados y se mantendrán los predeterminados por el software XCTU para el modo AT.

En las siguientes secciones, se muestran las configuraciones basadas en [70] de los módulos XBee S1 que conforman la red de comunicación.

Configuración del coordinador

Cuando el módulo es reconocido en el software XCTU se ingresan los parámetros a ser modificados dando clic sobre el ícono del módulo conectado. Se eligió el tipo de modem XBee (XB24) y versión del firmware actualizada (10ef). Los parámetros se configuran de la siguiente manera:

- SC: 380.
- CH: 12.
- ID: 1993.
- DH: 0.
- DL: FFFF.

Donde SC (Scan Channel por sus siglas en inglés) al tomar el valor hexadecimal de 380 permite trabajar con tres canales, el canal (CH) seleccionado es el 12. ID es el nombre de la red o PAN ID. DH y DL es la dirección broadcast (0x0000FFFF) esto permite que el coordinador envíe los datos a los dispositivos finales. En la Figura 2.54 se indica la configuración antes mencionada.

Seleccionamos la manera de trabajo del módulo XBee (coordinador), cambiamos el valor del SC en 380 y desactivamos el modo API para trabajar en modo transparente como se indican en las Figuras 2.55 y 2.56 respectivamente.

Figura 2.54. Configuración de CH, ID y dirección del coordinador.

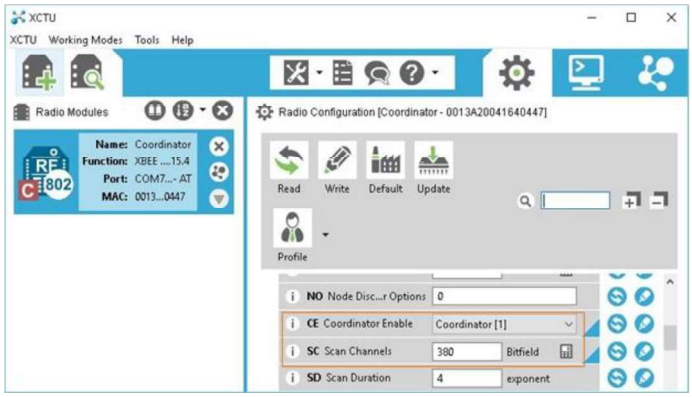


Figura 2.55. Configuración de SC y modo coordinador (CE).

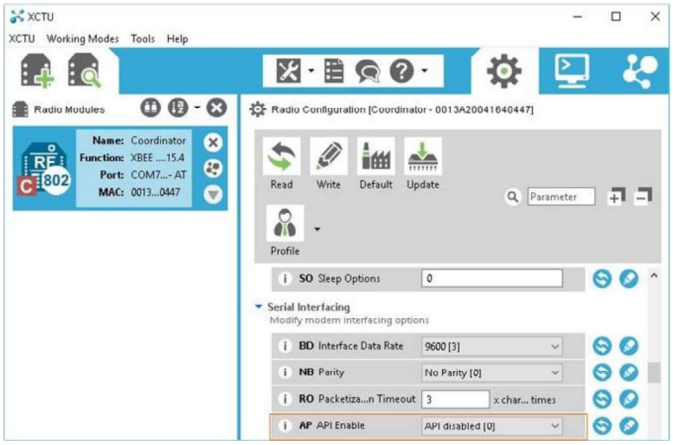


Figura 2.56. Configuración operación modo transparente.

Configuración de los dispositivos finales

Los parámetros que se configuran para el manejo de los dispositivos finales son similares a la configuración del coordinador, sólo se debe realizar una variación, seleccionar la manera de trabajo de cada módulo XBee como dispositivo final (end device).

Parámetros de los dispositivos finales.

- SC: 380.
- CH: 12.
- ID: 1993.
- DH: 0.
- DL: FFFF.
- API: disabled.

Finalmente, para que el banco de pruebas sea usado de manera sencilla se desarrolló un sistema de visualización o interfaz.

2.5 Sistema de visualización

Para el sistema de visualización se diseñó una interfaz gráfica desarrollada en Guide, el entorno gráfico de Matlab. A continuación, el proceso de manejo de interfaz se detalla en la Figura 2.57.

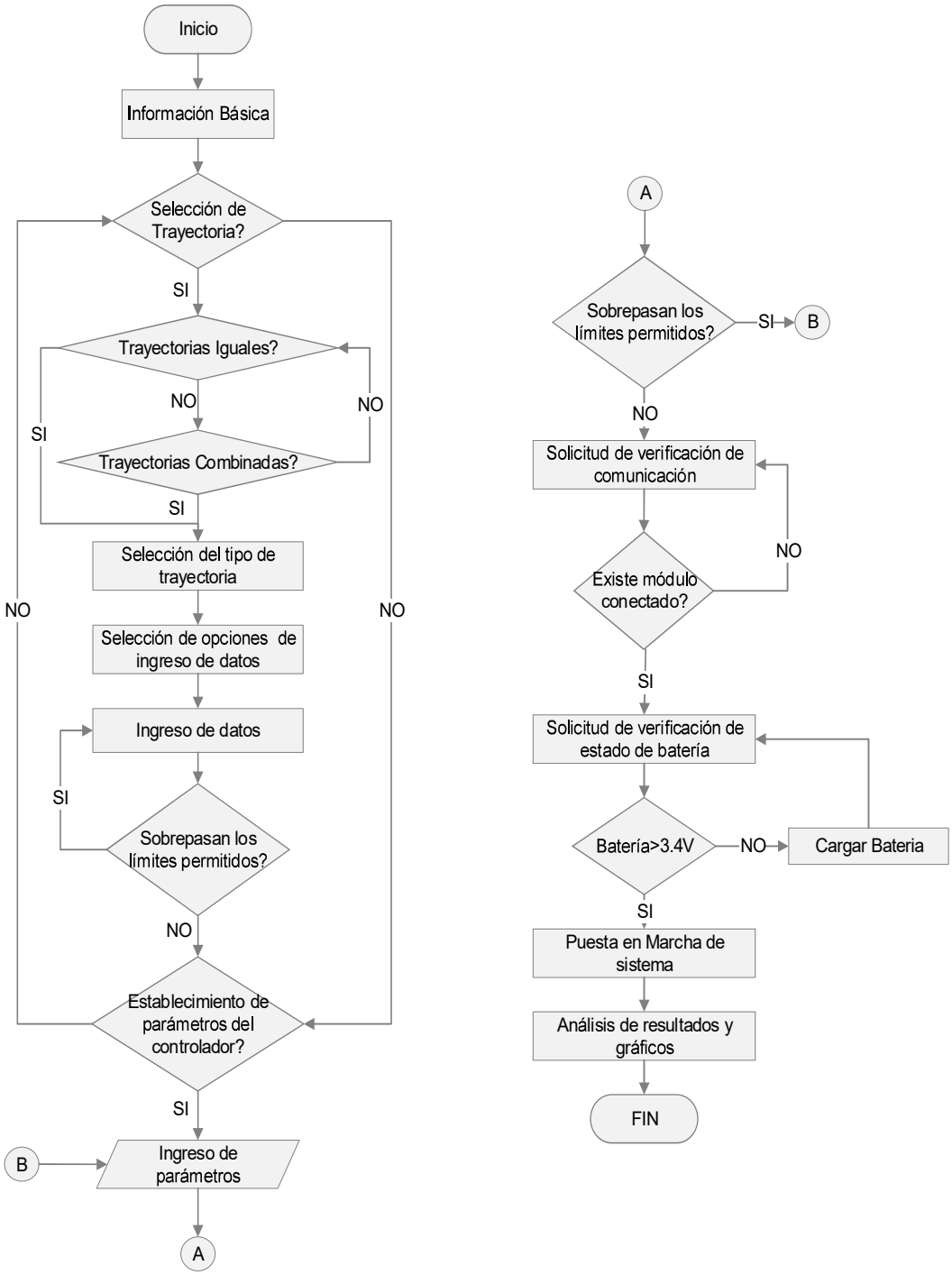


Figura 2.57. Funcionamiento de interfaz gráfica del proceso

A continuación, se especificarán los aspectos más importantes del contenido del sistema de visualización. El análisis detallado del funcionamiento total de la interfaz gráfica se encuentra en el ANEXO IV.

Basados en la norma ISA101 [71] se realizó una interfaz con diferentes características como:

- Posibilidad de ingreso de datos.
- Utilización de colores standard que no provoquen distracción al usuario.
- Alarmas visuales
- Análisis de gráficos
- Análisis de datos
- Ayuda de usuario

Ventanas Principales

Existen varias ventanas que permiten el funcionamiento de la interfaz gráfica. Para empezar, es necesario utilizar la ventana *presentación* que permite navegar hacia las demás ventanas; justamente, en la Figura 2.58, se observa la ventana de presentación.

Figura 2.58. Ventana de presentación de interfaz gráfica.

Es necesario guiar al usuario con una breve reseña de los temas tratados en el proyecto, para esto se implementó una ventana de información general como se indica en la Figura 2.59. El usuario puede navegar entre los temas que desee, ya sea robótica móvil, seguimiento de trayectoria, control implementado, o marcadores de referencia matricial. Al presionar en la flecha de color verde el usuario puede acceder a la ventana de selección de trayectoria y establecimiento de parámetros del controlador.

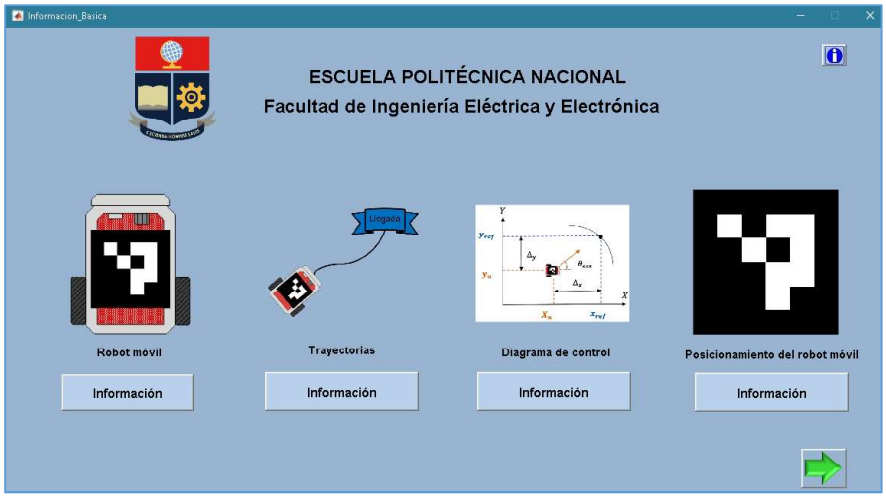


Figura 2.59. Ventana de información general del proyecto.

El mecanismo de la interfaz de usuario viene dado por una amplia gama de íconos que son fáciles de usar para navegar entre ventanas. A continuación, un breve detalle de los íconos utilizados y representados en la Figura 2.60:

Figura 2.60. Iconos de la interfaz gráfica.

- *Home.* – En las ventanas que se encuentre permite regresar al usuario a la ventana de información general.
- *Desplazamiento de ventanas.* – Para continuar y regresar entre las ventanas, existen dos íconos: flecha verde para continuar, y flecha roja para volver.
- *Abandonar el sistema.*- Si el usuario realizó todas las actividades dentro de la interfaz gráfica, puede abandonar el sistema pulsando el botón *Salir.*
- *Reiniciar el sistema.* – Si se necesita reiniciar el sistema, una vez terminado el proceso el usuario puede pulsar el ícono mostrado *Reiniciar.*

- *Análisis de resultados.* – Una vez terminado el proceso, el usuario puede acceder a los resultados generados, pulsando el icono *Resultados.*
- Paro de emergencia. – En el caso de necesitar detener por completo el sistema, el usuario puede pulsar el ícono *Paro de Emergencia.*

Ventana de selección de parámetros

El usuario tiene la facilidad de elegir en el menú de Directrices Generales la ventana de cada caso que se desee observar como lo son la selección de trayectoria o los parámetros del controlador como se indica en la Figura 2.61. Estas ventanas permiten al usuario elegir el tipo de trayectoria que desea establecer a los robots, o el ingreso de las constantes de calibración del controlador.

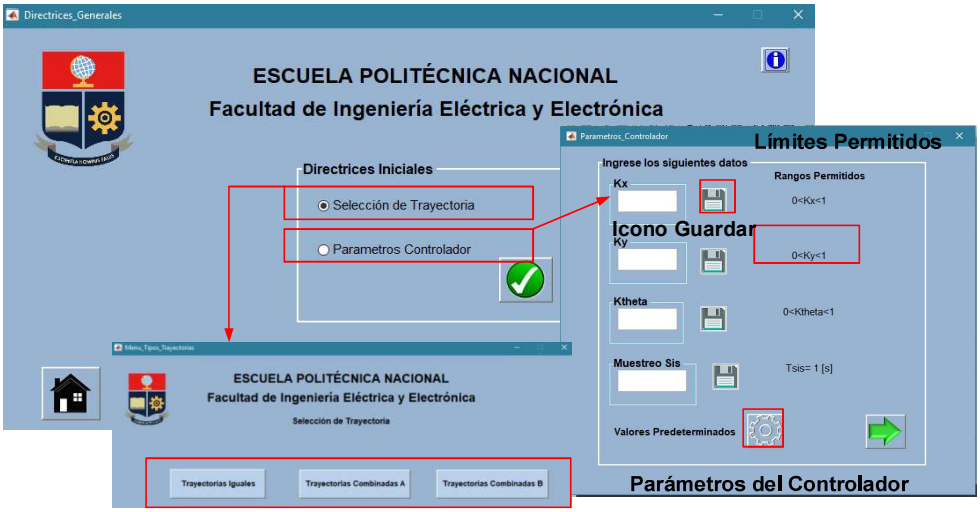


Figura 2.61. Selección de Directrices Generales.

El usuario puede elegir la forma de trabajo de los robots, es decir:

- *Trayectorias iguales.* – Los dos robots realizan la misma trayectoria elegida por el usuario.
- *Trayectorias combinadas.* – Los robos realizan diferentes trayectorias, según el caso elegido por el usuario.

Existen diferentes tipos de trayectorias que puede elegir el usuario. En la Figura 2.62, se presenta como ejemplo el caso de trayectorias iguales, donde se pueden seleccionar trayectorias lineales, lineales con pendiente de 45°, circular y doble frecuencia.

Figura 2.62. Selección de tipo de trayectorias.

En la Tabla 2.4 se observa las opciones de trayectorias disponibles en el proyecto.

Tabla 2.4. Tipos de funcionamiento de robots

Trayectorias Iguales		**Trayectorias Combinadas**	
Robot 1	**Robot 2**	**Robot 1**	**Robot 2**
Lineal	Lineal	Circular	Doble Frecuencia
Lineal m=45°	Lineal m=45°	Lineal	Lineal m=45°
Circular	Circular		
Doble Frecuencia	Doble Frecuencia		

Una vez elegida la trayectoria deseada, el sistema despliega un menú que permite elegir el caso de ingreso de datos. En la Figura 2.63 se muestra los casos para la trayectoria lineal, entre estos se encuentran los siguientes casos:

- *Distancia-Velocidad:* El usuario puede ingresar la distancia y velocidad lineal.
- *Distancia-Tiempo*: El usuario ingresa distancia y tiempo.
- *Velocidad-Tiempo:* El usuario puede ingresar velocidad y tiempo.

A continuación, se detalla un ejemplo de la selección *Distancia-Tiempo.*

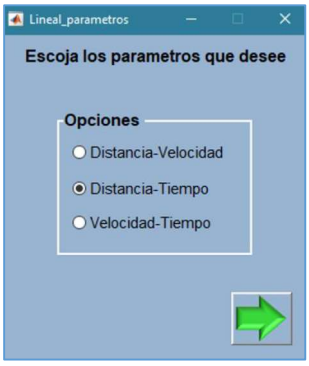


Figura 2.63. Menú de selección de ingreso de datos.

Ventana de Ingreso de Datos

En las ventanas de ingreso de datos existen varios parámetros a tomar en cuenta. El usuario tiene dos opciones de ingreso de datos.

1. La ventana proporciona al usuario el límite permitido de los valores a ingresar, como se observa en la Figura 2.64, el ingreso de datos es manual y mediante el ícono de disquete se guardan los mismos.
2. La otra forma de ingreso de datos es la carga de valores predefinidos por el sistema. Mediante la pulsación del ícono en forma de tuerca se cargan automáticamente los valores predeterminados.

Para el caso de la Figura 2.64 el caso elegido fue una trayectoria lineal, con el ingreso de distancia y tiempo. En el caso de pulsar el icono de tuerca se cargan los valores 0.9 metros y 25 segundos para distancia y tiempo respectivamente. En la parte superior es posible observar los límites permitidos por el sistema en el caso de realizar un ingreso manual de datos. Posteriormente se pulsa el icono de guardar para continuar con el proceso.

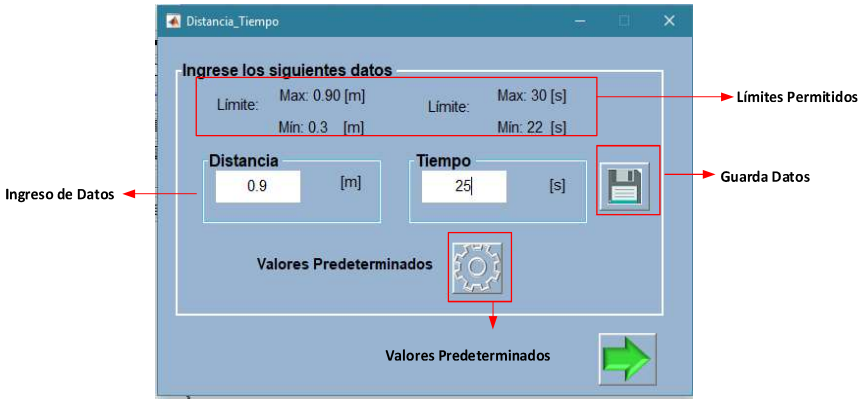


Figura 2.64. Ventana de Ingreso de Datos.

Ventanas emergentes

Es necesario que el usuario reciba alertas sobre el uso de la interfaz, por esto existen varias ventanas de alerta en el sistema:

- *Ventanas de Alerta.* – Si el usuario, maneja la interfaz de manera incorrecta recibirá una ventana de alerta indicando el error cometido. Las diferentes ventanas de error se indican en la Figura 2.65.

Un claro ejemplo de las ventanas de alerta es cuando el usuario olvida la conexión del módulo de comunicación, en este caso, la interfaz despliega una ventana de informe de comunicación fallida.

Figura 2.65. Ventanas de Alarma

- *Ventanas de éxito.* – Cuando el usuario ingrese los datos dentro de los valores permitidos, se despliega una ventana informando el ingreso exitoso de datos como se indica en la Figura 2.66.

Figura 2.66. Alerta de éxito.

- *Estado de Batería.* – El usuario antes de poner en marcha al sistema, puede analizar el estado de la batería. Como se muestra en la Figura 2.67, existen diferentes niveles de batería.
    - *Batería llena*. – Valor de batería mayor o igual 3.7 voltios.

- *Batería parcialmente llena.* – Valor de batería entre 3.5 y 3.69 voltios.
- *Batería media.* – Valor de batería entre 3.4 y 3.57 voltios.
- *Batería baja.* – Valor de batería menor a 3.4 voltios

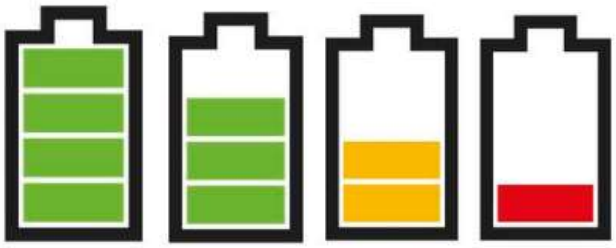


Figura 2.67. Estados de Batería.

Ventanas de ayuda

En cada ventana de usuario existe un ícono de información de cómo manejarla. El momento que el usuario presiona el botón de color azul se despliega una ayuda corta de cada opción presente en la ventana como se indica en la Figura 2.68.

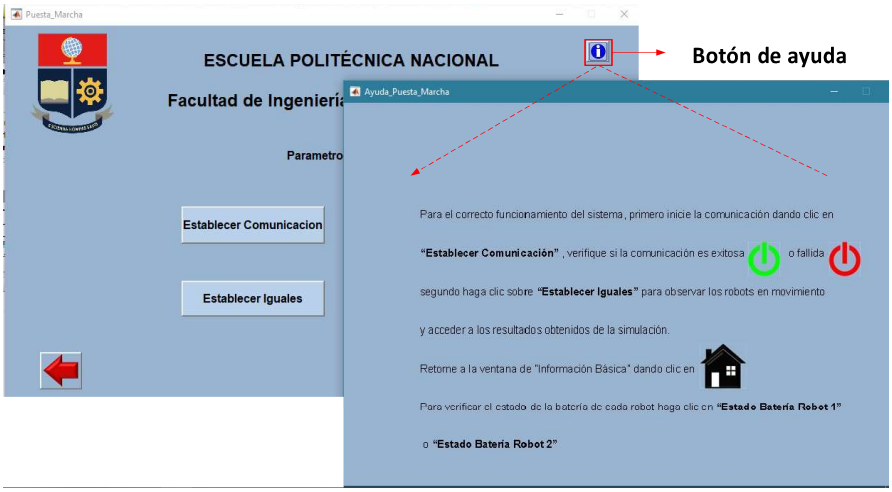


Figura 2.68. Ventana de ayuda.

Ventana de visualización y resultados

Como se indica en la Figura 2.69 la ventana de visualización permite graficar el movimiento de los dos robots al cumplir la trayectoria elegida por el usuario, en el caso que el usuario coloque únicamente un robot el sistema permite utilizar la misma ventana de visualización para un solo robot asumiendo que el otro robot se encuentra siempre en el mismo sitio.

En esta ventana se encuentran varios botones como paro de emergencia, reiniciar procedimiento, análisis de resultados entre otros.

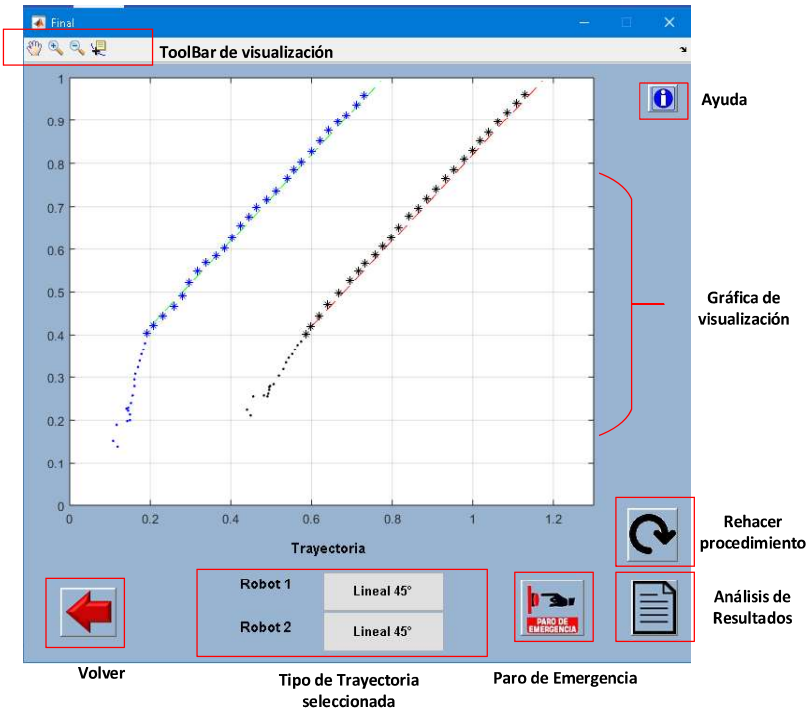


Figura 2.69. Ventana de visualización de sistema

Para el análisis de los resultados la ventana desplegada es la que se observa en la Figura 2.70, esta indica al usuario los datos ingresados, además de los datos calculados. Hay un menú de opciones de análisis de gráficos los cuales pueden ser accedidos por el usuario para su respectiva interpretación y análisis.

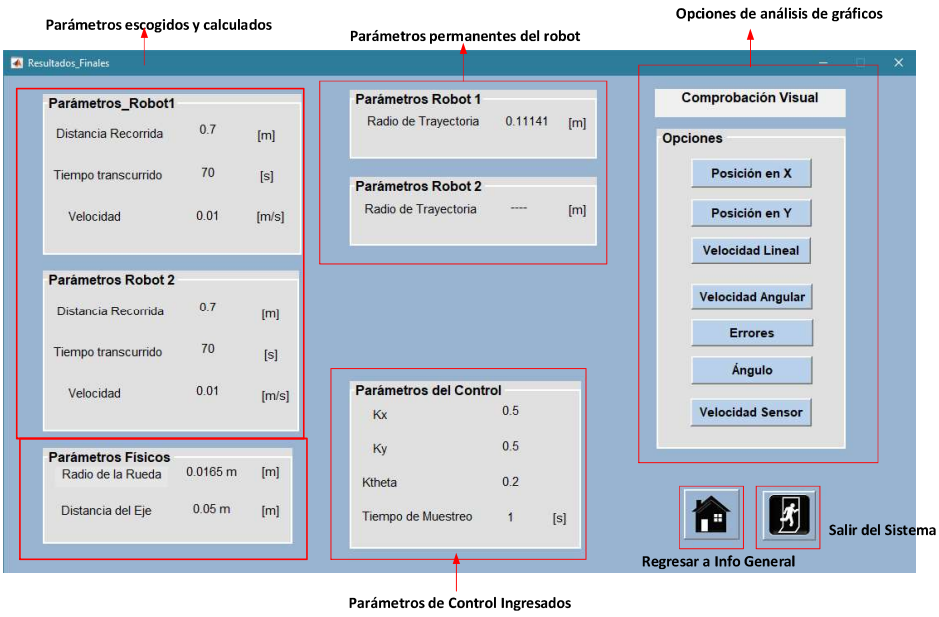


Figura 2.70. Ventana de análisis de resultados

Ventana de Análisis de gráficos

El caso mostrado en la Figura 2.71, es el de visualización de *posición en X* para los dos robots, en esta ventana se despliegan las dos gráficas con sus respectivas leyendas de valores actuales y de referencia. Además de una barra de manejo de gráficos (Toolbar de visualización) que contiene botones como zoom, manejo manual y puntero.

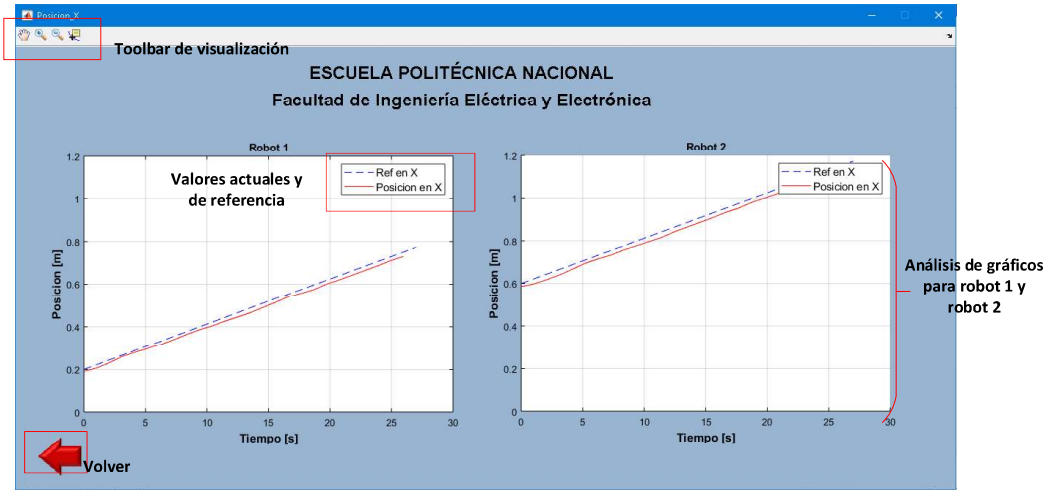


Figura 2.71. Análisis de gráficos

3. RESULTADOS Y DISCUSIÓN

En esta sección se presentan las pruebas realizadas en base al banco de pruebas implementado. Inicialmente se incluyen las pruebas de identificación de los marcadores de referencia mediante el sistema de monitoreo, con la finalidad de verificar su posición y orientación en el entorno. Posteriormente, se indican las pruebas de comunicación realizadas entre los robots y la computadora a través de los dispositivos XBee, a fin de enviar las respectivas ordenes de control y a su vez conocer el estado de la batería de cada robot. Finalmente, se indican los resultados obtenidos para el control de posición y seguimiento de trayectorias, comprobando el desempeño del sistema mediante el análisis de errores.

3.1 Pruebas de identificación de marcadores ArUco

Estas pruebas se realizaron para comprobar que la información obtenida corresponda únicamente a los dos marcadores de referencia ArUco, como se estableció en la sección de Generación de Marcador ArUco. Se considera que el marcador cero (id= 0) representa al robot 1 y el marcador uno (id=1) representa al robot 2.

En la Figura 3.1 se puede observar que la identificación y procesamiento de información se realiza exclusivamente para el marcador cero y uno. A pesar de que la cámara observa dos marcadores adicionales, estos son identificados, pero no se genera información alguna de su posición y orientación. Para cada marcador, su eje X' es de color rojo, el Y' de color verde y el Z' de color azul. Estos ejes se dibujan tomando como referencia el centro del marcador y su orientación.

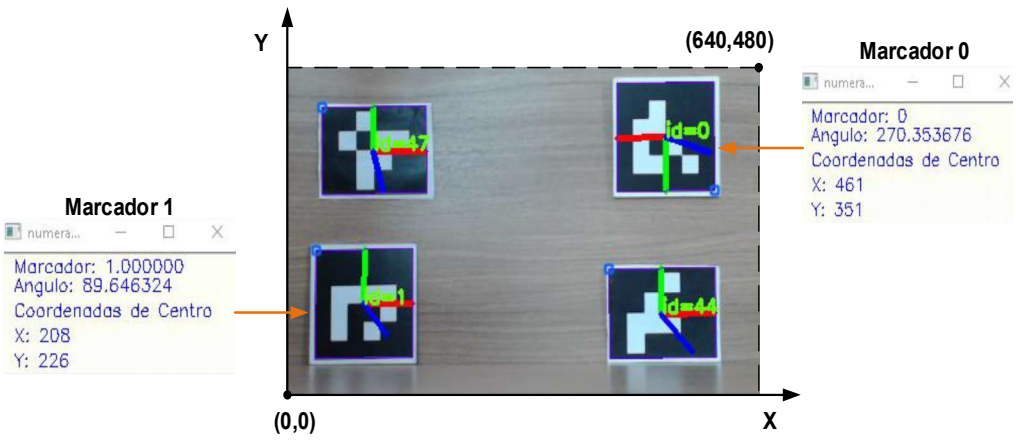


Figura 3.1. Identificación de marcadores.

Una vez identificados los marcadores cero y uno, se obtiene la orientación y posición en el plano correspondiente al sistema de coordenadas internas respecto a la cámara, el cual se

expresa en pixeles; es decir, el eje X empieza desde 0 hasta 640 pixeles y el eje Y desde 0 hasta 480 pixeles. Después de obtener la información del marcador cero y uno, es necesario enviar estos datos a Matlab mediante el uso de librerías de Matlab Engine. Esta información llega directamente al *Workspace*. Un ejemplo de los datos proporcionados por Visual Studio y que son mostrados en el *Workspace* se presenta en la Figura 3.2.

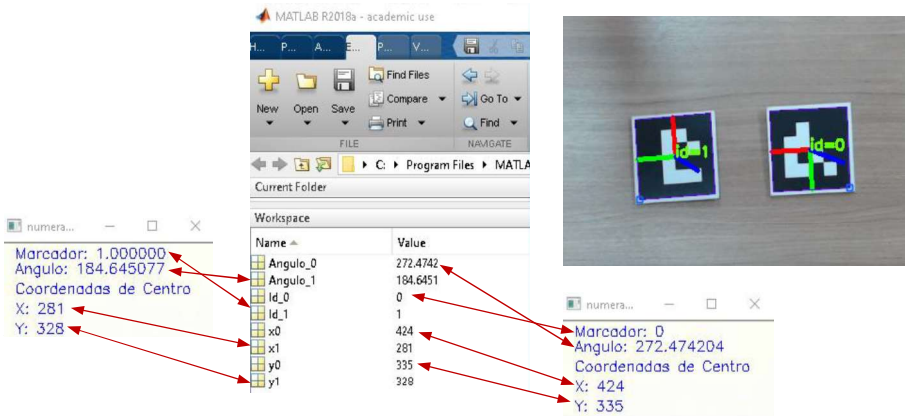


Figura 3.2. Verificación de envío de datos a Matlab.

Se debe considerar que, para la generación de trayectorias y control de los robots móviles dentro del área de trabajo, la información de los marcadores desplegada en el *Workspace* se convierte de pixeles a metros. En la Figura 3.3 se observa la posición en XY para los dos marcadores de referencia tanto en pixeles como en metros, y su orientación en grados. Como se indicó en la sección Estimación de posición y orientación del marcador, la posición que se obtiene en pixeles se transforma a metros gracias a la Ecuación 2.8.

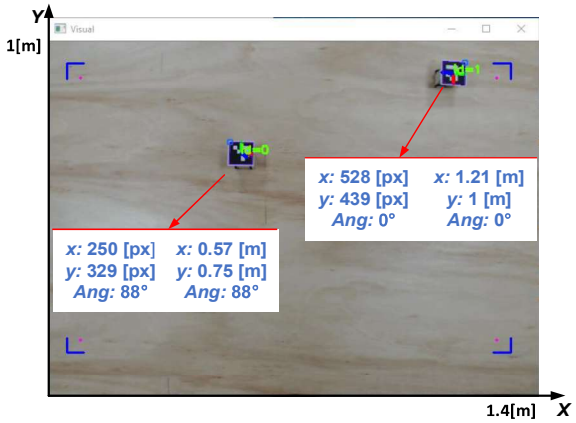


Figura 3.3. Orientación y posición de marcadores en el espacio de trabajo

Finalizadas las pruebas de identificación de los marcadores ArUco, se realizan las pruebas de comunicación entre los robots móviles y la computadora.

3.2 Pruebas de comunicación entre robots y ordenador

El objetivo de esta prueba es verificar el enlace de comunicación entre los robots y la computadora. La trama de datos empleada para la transmisión información presenta cuatro componentes que son: identificador del robot (Id), porcentaje de velocidad angular entre 0 y 100 de las ruedas derecha ($\%\omega_D$) e izquierda ($\%\omega_I$), y finalmente el estado de batería ($Estado\ Bat$) como se muestra en la Ecuación 3.2.

$$Trama\ de\ datos = [Id \quad \%\omega_D \quad \%\omega_I \quad Estado\ Bat]$$

Ecuación 3.2. Representación de Trama de datos.

Cuando el identificador del robot es cero ($Id = 0$) los comandos se envían al robot 1, caso contrario si el identificador del robot es uno ($Id = 1$) los comandos se envían al robot 2. Los porcentajes de velocidad angular enviados mediante la trama de datos representan el ciclo de trabajo de los motores (Control PWM) como se explicó en la sección de Implementación del controlador. Por otro lado, si estado de batería es igual a 1 ($Estado\ Bat = 1$) se verifica el voltaje de la batería y cuando estado de batería es igual a cero ($Estado\ Bat = 0$) no se realiza dicha verificación.

Para detener a los robots es necesario usar la trama de paro, que consta del identificador del robot, mientras que el porcentaje de las velocidades de las ruedas y el estado de batería toman el valor de cero. Una vez definida la trama datos, se realizaron algunas pruebas para verificar la conexión y el uso apropiado de la misma, así:

Prueba de envío de comandos a los robots

Mediante esta prueba se comprobó el envío y recepción de los porcentajes de velocidad angular de las ruedas. Como se indica en la Figura 3.4 se envió una trama de datos de acuerdo con la Ecuación 3.1, en donde: $Id = 0$, $\%\omega_D = 10$, $\%\omega_I = 0$ y $Estado\ Bat = 0$.

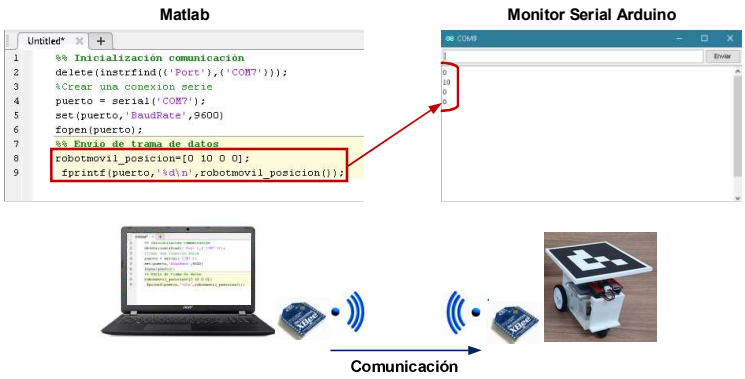


Figura 3.4. Verificación de envío de trama para el robot 1.

Prueba de lectura de estado de batería

En esta prueba se verificó el valor de voltaje de cada una de las baterías, cambiando el cuarto elemento de la trama por el valor de 1 y leyendo el dato recibido, como se muestra en la Figura 3.5; que es el estado de la batería del robot 1, obteniendo como respuesta (4.03 voltios en el ejemplo de la Figura 3.5).

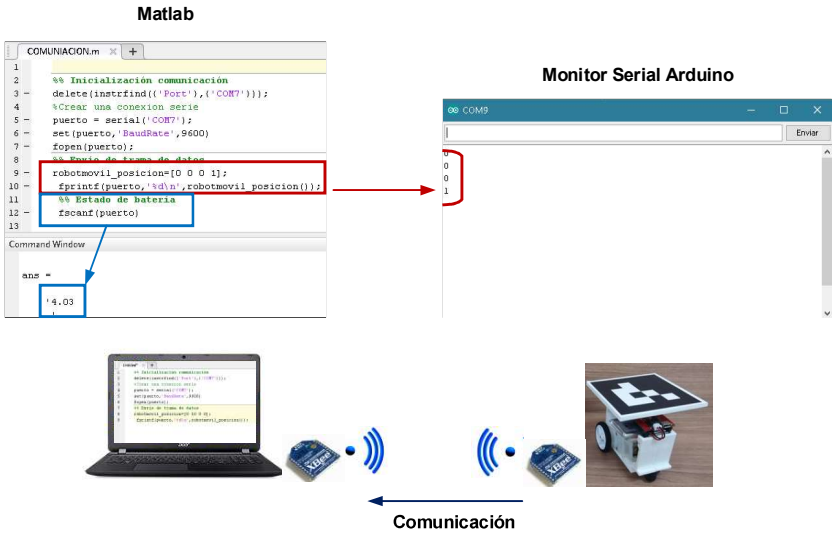


Figura 3.5. Estado de batería del robot 1.

3.3 Calibración del tiempo de adquisición de datos

Para la realización de las pruebas de calibración se considera el tiempo de retardo T_R que se genera en base al procesamiento de la imagen (monitoreo) y comunicación, además del tiempo de muestreo T_O correspondiente al algoritmo del controlador con el objetivo de establecer un tiempo total de muestreo para el sistema implementado.

Pruebas de tiempo de retardo del sistema

Esta prueba se realizó con la finalidad de obtener un promedio de retardo en el procesamiento de la información, para esto se efectuaron dos pruebas:

- ✓ Comprobación de tiempo de procesamiento en Visual Studio 2015.
- ✓ Comprobación de tiempo de procesamiento en Matlab R2018A

El desarrollo y las pruebas del banco de pruebas se realizaron en una computadora Acer Intel ® Core i5 con 8GB de memoria RAM. Estas pruebas se realizaron a través de los cronómetros provistos por los entornos de programación utilizados; es decir, se contabilizó el tiempo que demoran las líneas de programación en ejecutarse. Para el caso de Visual

Studio 2015 se mide el tiempo de procesamiento en la detección de los marcadores, cálculo de pose de los robots (coordenadas XY y posición angular), apertura de Matlab Engine, y envío de la información hacia Matlab. Como se indica en la Figura 3.6 a través de las pruebas realizadas se obtuvo un valor promedio de 0.9 segundos.

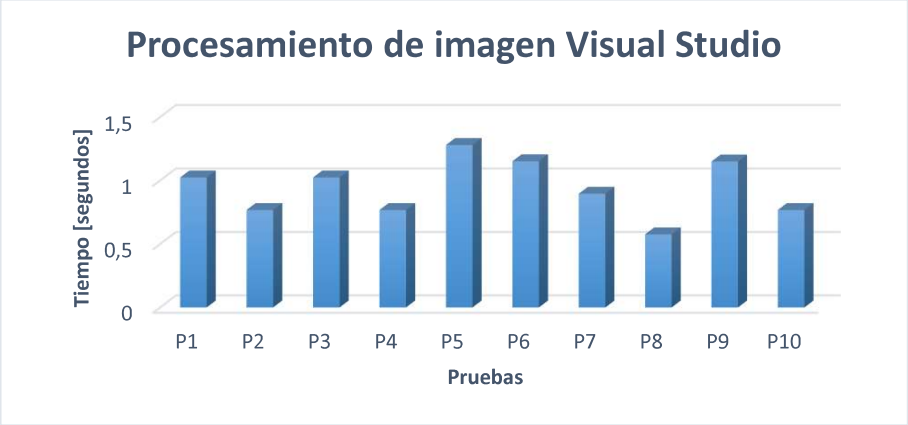


Figura 3.6. Estimación de tiempo de retardo en Visual Studio 2015

En el entorno de Matlab R2018a se realizaron mediciones del tiempo requerido para el envío de las acciones de control hacia los XBee de cada robot. En la Figura 3.7 se indican los valores de tiempo obtenidos. En base a estos resultados se obtiene un valor promedio de 0.002 segundos.

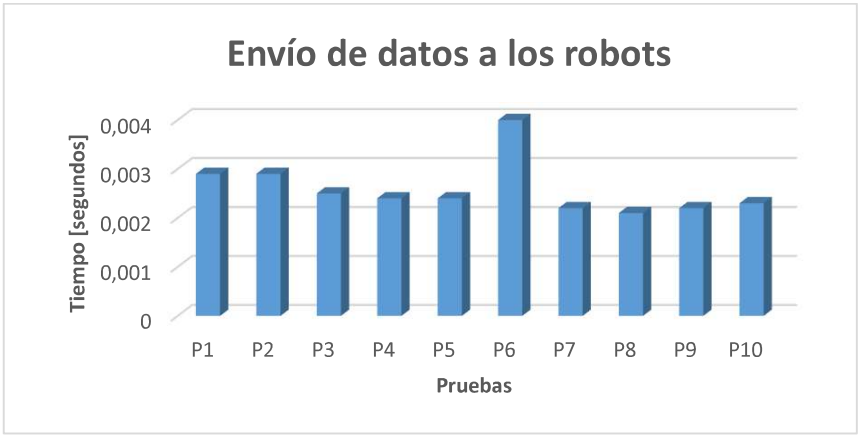


Figura 3.7. Estimación de tiempo del envío de las acciones de control.

Al final de estas pruebas se estimó un valor de tiempo de retardo total T_R de 0.902 segundos.

Prueba de tiempo de muestreo

El tiempo de muestreo T_O se lo calcula en base a la lógica de programación utilizada, es decir se considera el tiempo que se demora una iteración completa del controlador dentro del entorno de Matlab (tiempo del lazo del control). En la Figura 3.8 se observan los valores de tiempo obtenidos en cada iteración del lazo de control en Matlab, obteniendo un valor promedio de T_O igual a 0.17 segundos.

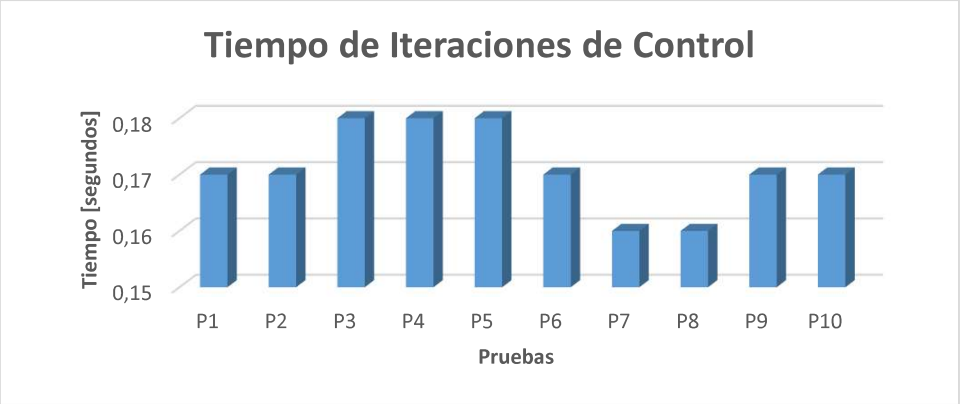


Figura 3.8. Estimación del tiempo de muestreo del controlador.

De esta manera el tiempo de latencia del sistema T_T es igual a 1.072 segundos. Para la generación de trayectorias y para la realización de pruebas en el testbed implementado el tiempo de latencia T_T se aproxima a 1 segundo. Para la sintonización de las constantes del controlador se considera el cálculo del error mediante la aplicación de la Norma 2 definida en la Ecuación 3.5. [42]

$$Error = \|Valor\ de\ Referencia - Valor\ Actual\| = \sqrt{\left(X_{ref} - X_{actual}\right)^2 + \left(Y_{ref} - Y_{actual}\right)^2}$$

Ecuación 3.5. Ecuación de la norma del error.

Luego de haber realizado las pruebas del algoritmo de control con diferentes valores de constantes de calibración, se observó que el sistema de control se vuelve inestable para ciertos valores como por ejemplo $k_x = 0.1$, $k_y = 0.1$ y $k_\theta = 0.2$. Las pruebas que se presentan a continuación se realizaron para los parámetros que se indican en la Tabla 3.1. Las pruebas se aplicaron a uno de los dos robots ya que se consideró que ambos presentan la misma naturaleza, se las realizó para una trayectoria circular con un radio de 0.25 metros y una velocidad lineal de 0.04 m/s.

Tabla 3.1. Valores de constantes de calibración del controlador

	k_x	k_y	k_θ
Constantes 1	0.6	0.5	0.3
Constantes 2	0.7	0.7	0.4
Constantes 3	0.86	0.86	0.38

3.4 Prueba de diferentes ganancias de control en trayectoria circular

Se realizó la prueba para un robot ante una trayectoria circular. Inicialmente el robot se ubicó en la posición (0.54, 0.23) m con una orientación de 23° (banderín de color verde en la Figura 3.9). Como primer objetivo el robot móvil debe alcanzar la posición (0.47, 0.58) m (banderín de color rojo en la Figura 3.9). Posteriormente el robot seguirá la trayectoria predeterminada. Esta trayectoria tiene un radio de 0.25 metros, con centro en (0.33, 0.5) m. Considerando la Tabla 3.1 las pruebas fueron realizadas para una velocidad lineal de 0.04 m/s.

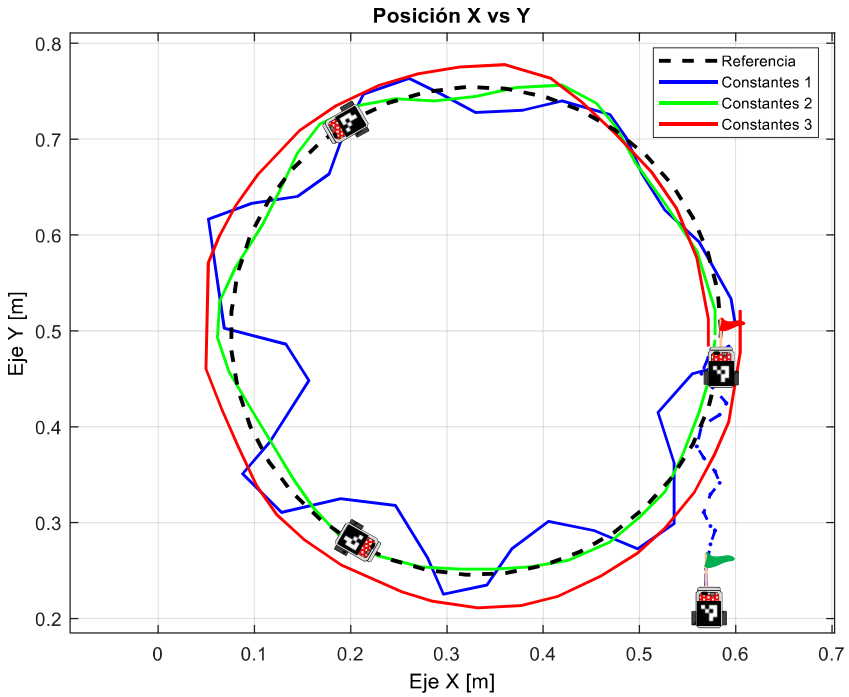


Figura 3.9. Seguimiento de trayectoria circular ante diferentes constantes de control.

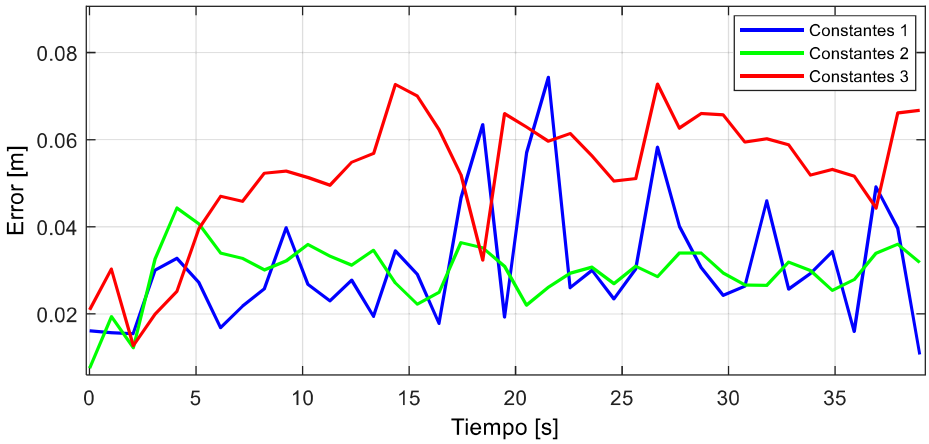


Figura 3.10. Norma del error de seguimiento de trayectoria circular para las diferentes constantes de control.

Ante las diferentes pruebas realizadas el robot llegó al punto objetivo como se indica en la Figura 3.9. En la Tabla 3.2 se puede observar el valor del error de posicionamiento inicial del robot, en donde el menor error obtenido fue con las Constantes 2. En la Figura 3.10 se muestra la norma del error de seguimiento de la trayectoria circular. Es posible observar que al emplear las Constantes 1 el error es notable a partir de los 15 segundos transcurridos, llegando a un valor de hasta 0.06 metros.

La prueba con las Constantes 2 presenta el mejor desempeño en el seguimiento de la trayectoria circular, con un error de hasta 0.04 metros. Mientras que el mayor error se presentó al utilizar las Constantes 3, las mismas que hacieron que el robot se aleje demasiado de la trayectoria de referecencia obteniendo un error máximo de 0.07 metros.

Tabla 3.2. Análisis de error de posicionamiento inicial ante constantes de control.

	Error de Posición [m]
Constantes 1	0.0138
Constantes 2	0.0093
Constantes 3	0.0244

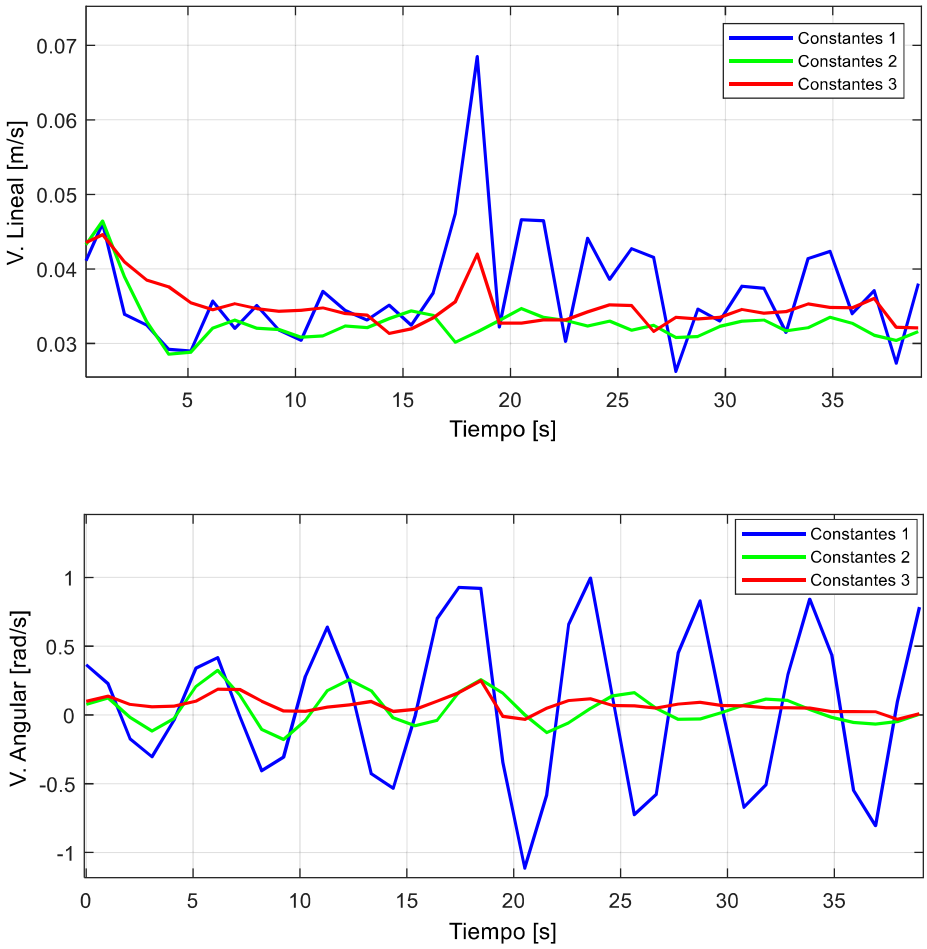


Figura 3.11. Señales de control de velocidad lineal y angular ante diferentes constantes de control.

Las señales de control de velocidad lineal y velocidad angular obtenidas en las pruebas realizadas se presentan en la Figura 3.11. Para las diferentes constantes establecidas, el controlador calculó señales de control necesarias para cumplir con la trayectoria designada. Con las Constantes 1, la señal de control de velocidad lineal fue mayor a la velocidad lineal establecida en la interfaz gráfica, llegando a valores de hasta 0.07 m/s. Para el caso de las Constantes 2 el controlador calculó valores de velocidad lineal menores a la velocidad de referencia (0.04 m/s), llegando a valores de velocidad lineal alrededor de 0.35 m/s. Finalmente, al utilizar las Constantes 3, los valores de velocidad calculados por el control presentaron valores mayores a la prueba anterior llegando hasta 0.042 m/s.

La señal de velocidad angular ante las diferentes constantes de control consideradas se ve seriamente afectada. En el caso de las Constantes 1; el controlador intenta manejar la orientación del robot obligándolo a volver a la trayectoria por lo que es posible observar un

mayor número de oscilaciones. Mientras que con las Constantes 2, el controlador calcula velocidades angulares de valores pequeños ya que no presenta gran dificultad al tomar la curva de la trayectoria, de la misma forma para las Constantes 3 se obtuvieron oscilaciones pequeñas de velocidad angular ya que como es posible observar en la Figura 3.9 no existe dificultad en la orientación del robot móvil al describir un círculo, pero no cumple con los puntos designados de la trayectoria.

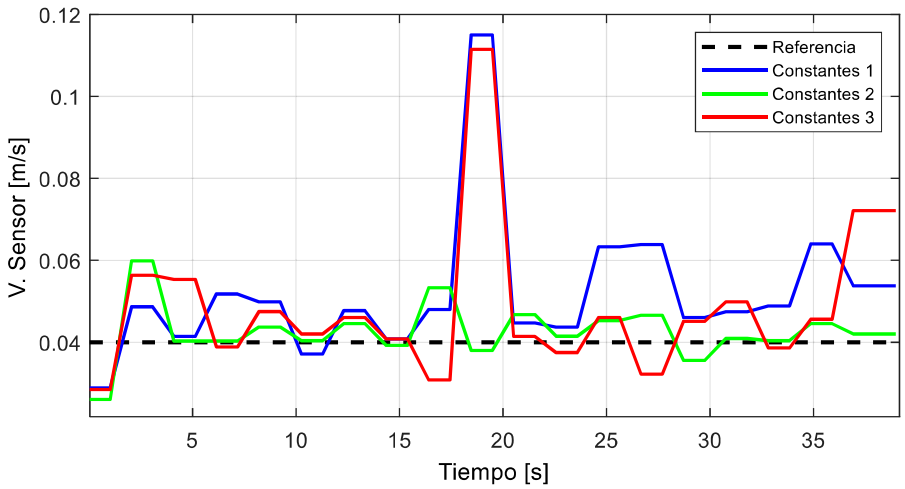


Figura 3.12. Velocidad lineal estimada por el sensor ante diferentes constantes de control.

Para verificar la velocidad lineal del robot móvil, en el presente proyecto se realizó una estimación de la velocidad lineal mediante el sensor de visualización, obteniendo una velocidad promedio calculada utilizando el desplazamiento del robot móvil registrado por sistema de monitoreo en cada instante de muestreo. Los resultados de esta estimación se indican en la Figura 3.12 donde es posible observar que la prueba de Constantes 2 presenta el mejor resultado cumpliendo en la mayor parte del tiempo con la velocidad lineal de 0.04 m/s establecida.

A través de las pruebas realizadas, fue posible determinar que los mejores resultados se presentaron con las Constantes 2. Por lo cual, las constantes de calibración del controlador que se eligieron para el control de posición y seguimiento de trayectorias fueron: $k_x = 0.7, k_y = 0.7$ y $k_\theta = 0.4$.

3.5 Prueba de seguimiento de trayectorias ante diferentes valores de velocidad lineal

Antes de realizar las pruebas para el seguimiento de trayectorias se procedió a determinar los rangos de velocidad lineal de referencia con las cuales, tanto el sistema de monitoreo como el sistema de control presenten resultados adecuados. De esta manera se determinó que para velocidades de referencia superiores a 0.05 m/s el sistema de control no actúa correctamente haciendo que el robot móvil se aleje demasiado de la trayectoria establecida debido a que el procesamiento del sistema de visión no funciona de manera correcta.

En cambio, para velocidades inferiores a 0.02 m/s el movimiento de los robots móviles puede ser oscilatorio alrededor de la trayectoria, esto se debe a que los robots móviles tienen mayor dificultad para vencer la inercia, además las distancias que debe recorrer el robot móvil son pequeñas en relación con las dimensiones del mismo. En la Figura 3.13 (a) se indica el sistema de visualización para una trayectoria circular con una velocidad lineal de 0.06 m/s y en la Figura 3.13 (b) se indica la visualización para la misma trayectoria con una velocidad de 0.01 m/s.

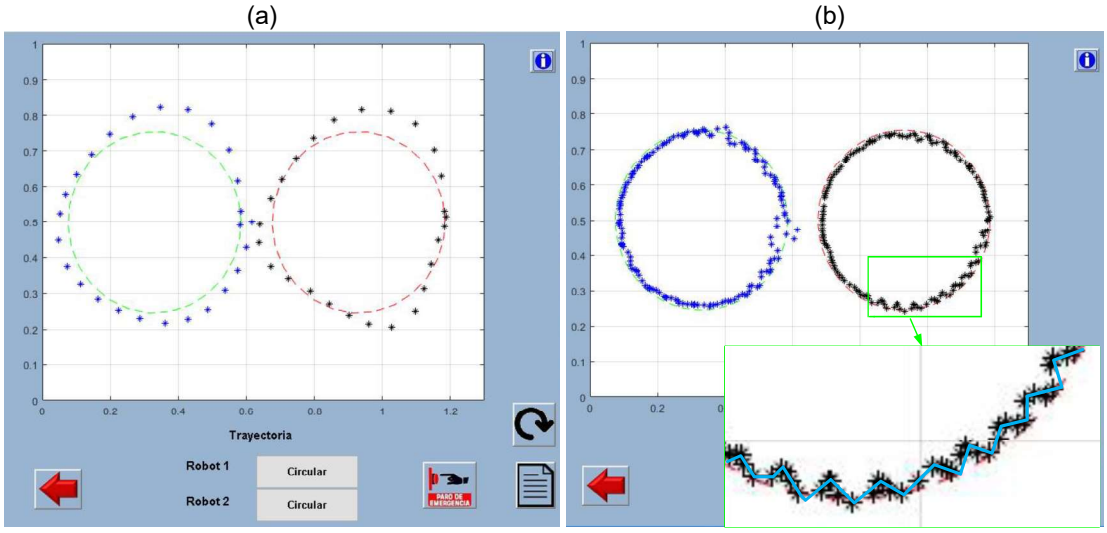


Figura 3.13. (a) Visualización para 0.06 m/s. (b) Visualización para 0.01 m/s.

De lo anteriormente expuesto se efectuaron las pruebas para diferentes trayectorias estableciendo los valores de velocidad lineal como se indican en los siguientes casos:

- Caso A: Seguimiento de trayectoria a 0.02 m/s.
- Caso B: Seguimiento de trayectoria a 0.04 m/s.
- Caso C: Seguimiento de trayectoria 0.05 m/s.

Antes de realizar el seguimiento de cada trayectoria los robots se encontrarán en una posición inicial denotada por las coordenadas $(x_0, y_0)\ m$ y una orientación inicial $\theta_0\ deg$.

Trayectoria Lineal con pendiente 0°

Se realizó la prueba para un solo robot ante una trayectoria lineal con pendiente de 0°. El robot se ubicó inicialmente en (0.0759, 0.6924) m con una orientación de 83.7°(banderín de color verde en la Figura 3.14). Como primer objetivo el robot móvil debe alcanzar la posición (0.3, 0.8) m (banderín de color rojo en la Figura 3.14) . Posteriormente el robot seguirá la trayectoria lineal de 0.9 m para las tres velocidades lineales establecidas.

El robot alcanza la posición inicial de la trayectoria (banderín de color rojo en la Figura 3.14) con un error 0.0162 m. Como se puede apreciar en la Figura 3.14 el robot móvil cumple con el seguimiento de trayectoria lineal para las tres diferentes velocidades lineales. Para las velocidades de 0.04 m/s y 0.05 m/s el robot sigue la trayectoria con dificultad, en cambio para una velocidad de 0.02 m/s la trayectoria que describe el robot es suave y se aproxima más a la trayectoria de referencia.

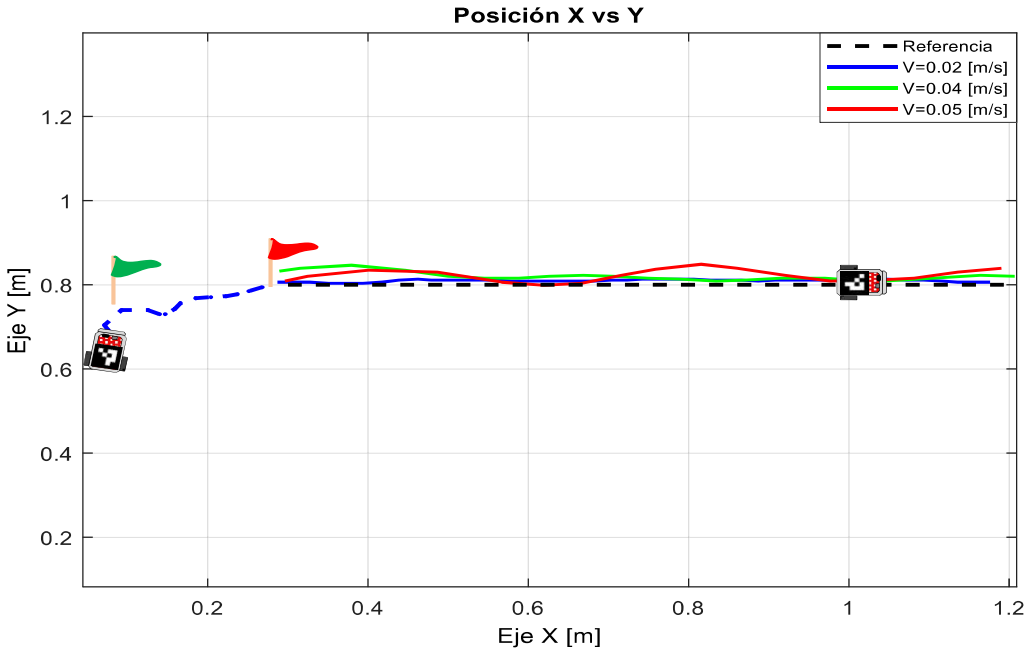


Figura 3.14. Seguimiento de trayectoria lineal ante diferentes velocidades lineales

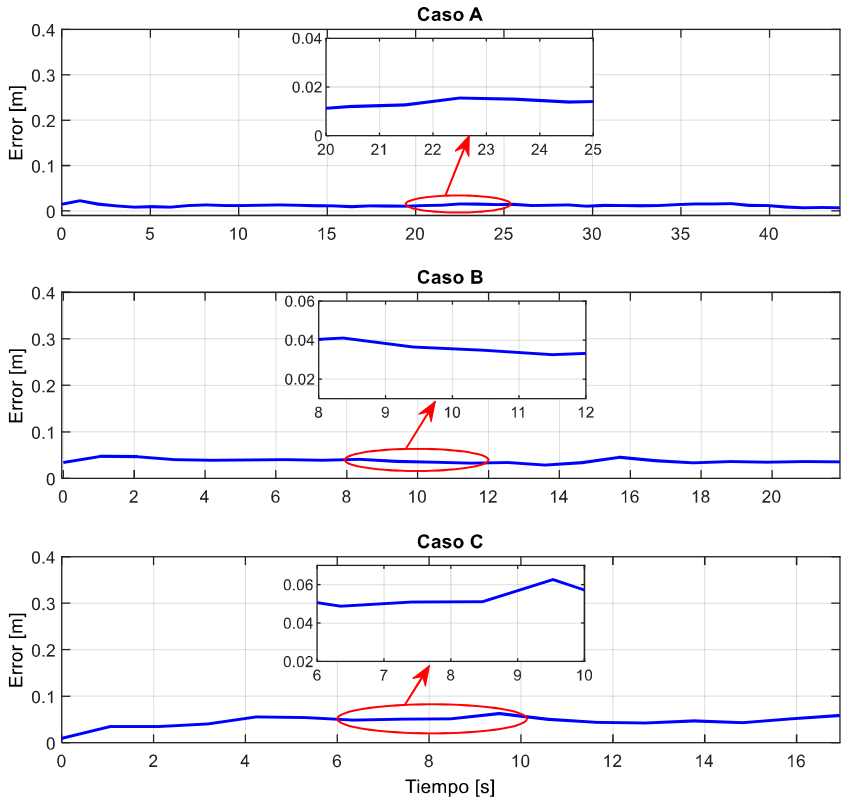


Figura 3.15. Norma del error de seguimiento de trayectoria lineal

En la Figura 3.15 se realiza el análisis de la norma del error del seguimiento de trayectoria ante los diferentes casos de velocidad consideradas. Para el Caso A el error es cercano a 0.01 m, en el Caso B se observa que el error puede llegar a 0.04 m. Finalmente, para el Caso C el error puede ascender hasta 0.06 m, es decir que existe un menor error cuando el robot móvil se desplaza a una velocidad de 0.02 m/s.

Las señales de control, velocidad lineal y angular, se indican en las Figuras 3.16 y 3.17, respectivamente. En la Figura 3.16 para el Caso A la señal de control de velocidad lineal es mayor a la a la velocidad de referencia aproximadamente 0.021 m/s, en los Casos B y C la velocidad lineal es menor a la referencia, aproximadamente 0.038 m/s y 0.042 m/s respectivamente. Las señales de control de velocidad angular que se muestran en la Figura 3.17 se puede apreciar que para los Casos A, B y C, esta oscila alrededor de 0 rad/s.

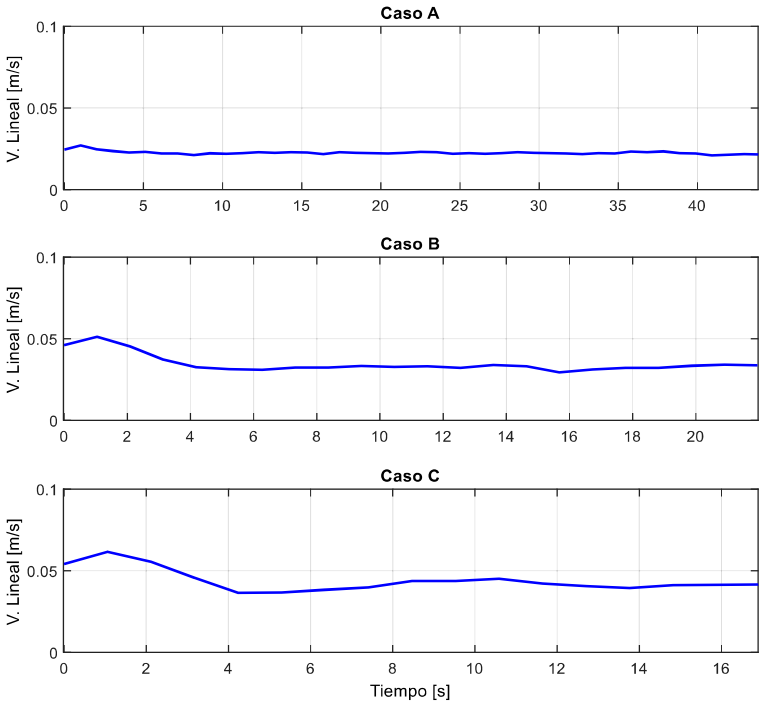


Figura 3.16. Señales de control de velocidad lineal

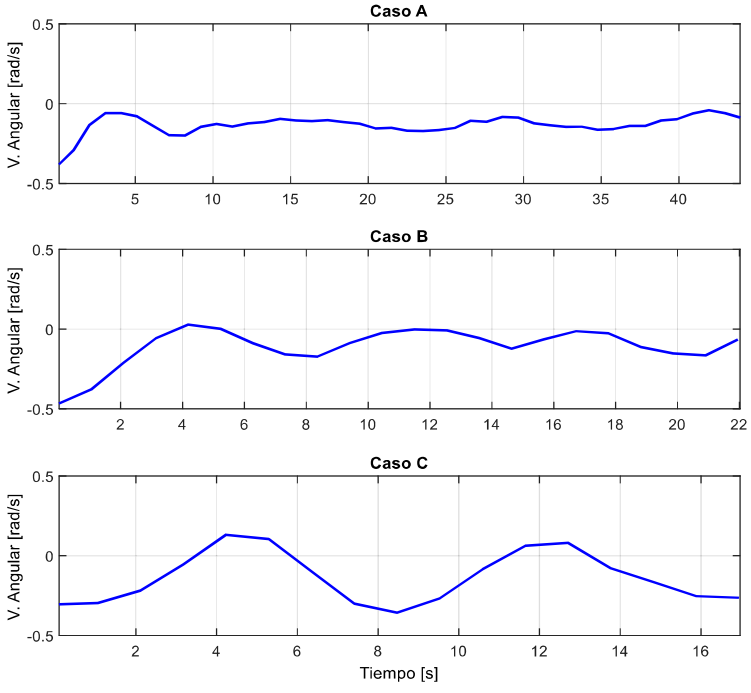


Figura 3.17. Señales de control de velocidad angular

A pesar de que no se cuenta con un sensor directo de la velocidad a la que se mueve el robot móvil, esta se puede estimar mediante los datos obtenidos por el sensor. En la Figura 3.18 se puede observar que para el Caso A la velocidad lineal con la que se desplaza el robot oscila alrededor de la velocidad de referencia, en cambio para los Casos B y C las velocidades que alcanza el robot móvil superan a las velocidades de referencia.

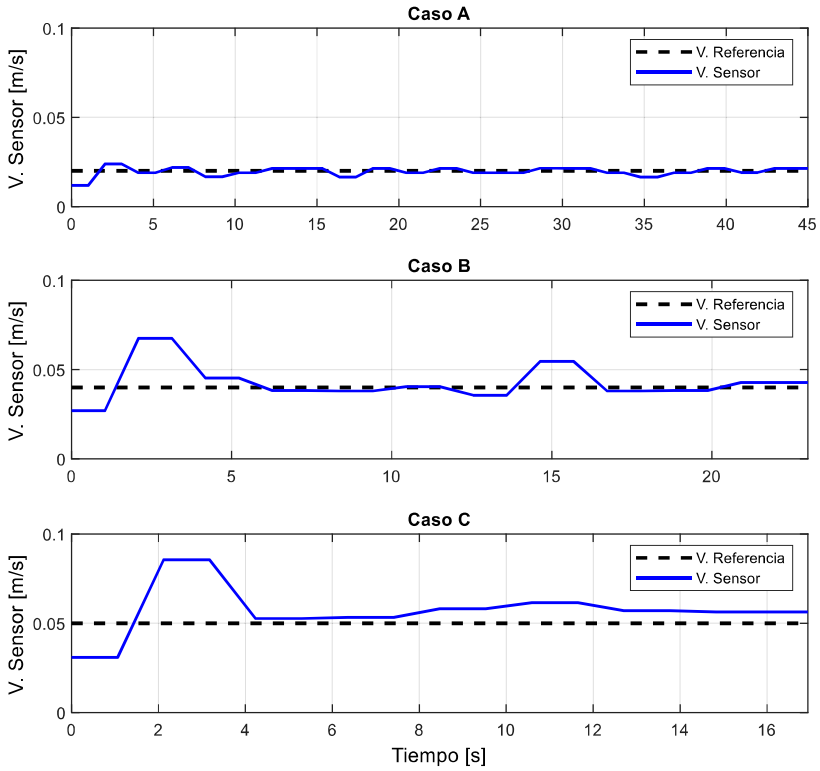


Figura 3.18. Velocidad lineal estimada por el sensor.

Trayectoria Lineal con pendiente de 45°

Se realizó la prueba para un robot ante una trayectoria lineal con pendiente de 45°. Inicialmente el robot se ubicó en la posición (0.11, 0.16) m con una orientación de 55.4° (banderín de color verde en la Figura 3.19).

Como primer objetivo el robot móvil debe alcanzar la posición (0.40, 0.18) m (banderín de color rojo en la Figura 3.19). Posteriormente el robot seguirá la trayectoria predeterminada. Esta trayectoria tiene una distancia de 0.8 metros. Se realizaron las pruebas ante los diferentes casos de velocidad lineal establecidas anteriormente.

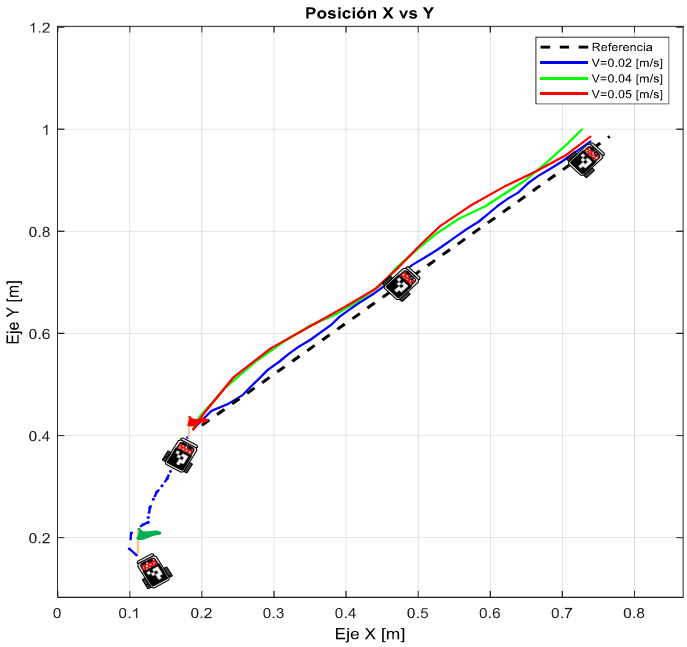


Figura 3.19. Seguimiento de trayectoria lineal con pendiente de 45° ante diferentes velocidades lineales.

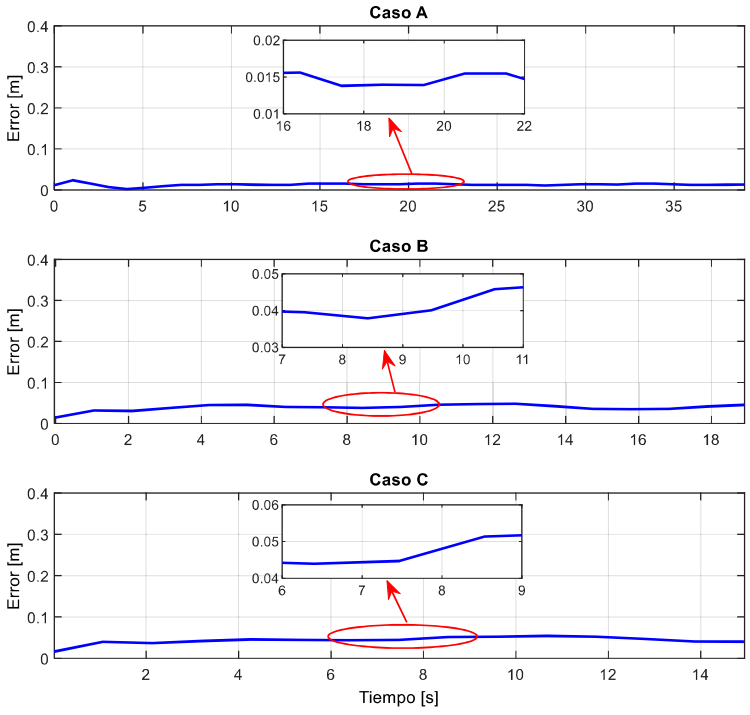


Figura 3.20. Norma del error de seguimiento de trayectoria lineal con pendiente de 45°

El error de posicionamiento inicial fue de 0.016 metros alejado del punto objetivo. En la Figura 3.20 se indica la variación de la norma de error con respecto al tiempo ante el seguimiento de la trayectoria lineal con pendiente de 45°. En el Caso A es posible observar que el menor error llega a un valor de 0.015 metros, esto se debe a que el robot cumplió la trayectoria en un mayor tiempo en comparación con los otros Casos. Los Casos B y C presentan valores de error de hasta 0.05 metros ya que las pruebas se realizaron con valores de velocidad lineal superiores a 0.02 m/s consecuentemente el robot cumplió la trayectoria en tan solo 19 y 15 segundos respectivamente.

Las señales de control se indican en la Figura 3.21 para cada caso. En el Caso A la señal de control es casi constante ya que el robot no tuvo dificultad en alcanzar la trayectoria, el valor de la señal de control es aproximadamente 0.021 m/s. En el Caso B inicialmente se observa un valor de velocidad lineal de 0.05 m/s, después de 8 segundos aproximadamente alcanza un valor de 0.035 m/s a lo largo del seguimiento de la trayectoria. En el Caso C al igual que el caso anterior el robot sufre una variación brusca al inicio ya que debe recorrer 0.8 metros en tan solo 15 segundos, por lo que la velocidad lineal alcanza 0.06 m/s para después de aproximadamente 8 segundos disminuir a un valor de 0.045 m/s durante el trayecto.

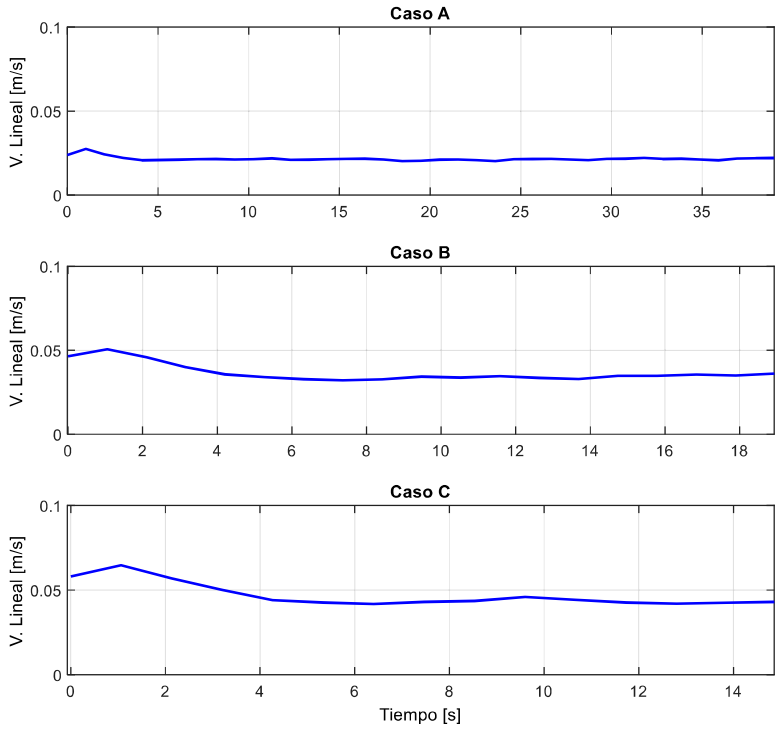


Figura 3.21. Señales de control de velocidad lineal

Al igual que en el caso de trayectoria lineal con pendiente de 0°, la velocidad angular debe alcanzar un valor de 0 rad/s en la Figura 3.22 se indican las señales de control de velocidad angular para cada caso establecido. En el Caso A el controlador intenta alcanzar una velocidad de 0 rad/s pero existen mayor número de picos ya que el valor de velocidad lineal del robot es bajo, provocando que las ruedas del robot ligeramente cambien de orientación.

En el Caso B y Caso C el controlador calcula señales de control de velocidad angular bajo los 0 rad/s ya que físicamente el robot va más rápido que en el Caso A, por lo que se aleja de la trayectoria constantemente. Sin embargo, el controlador permite al robot regresar de mejor manera hacia la trayectoria.

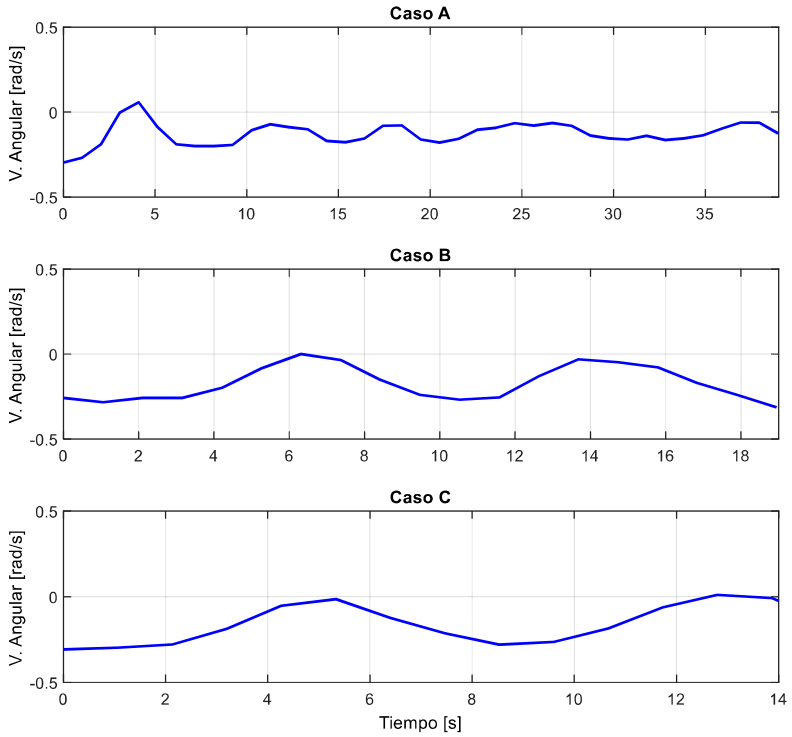


Figura 3.22. Señales de control de velocidad angular

Para la comprobación de la velocidad lineal del robot, se realizó una estimación de la misma, como se indicó anteriormente. En la Figura 3.23 se indica la velocidad estimada para cada caso establecido. En el Caso A el robot cumple satisfactoriamente con una velocidad lineal de 0.02 m/s. En el Caso B el robot alcanza los 0.04 m/s cumpliendo con lo establecido. En el Caso C el robot va a 0.05 m/s aproximadamente a lo largo de la trayectoria. Es notable en los 3 casos que al robot le toma alrededor de 8 segundos en establecer una velocidad constante.

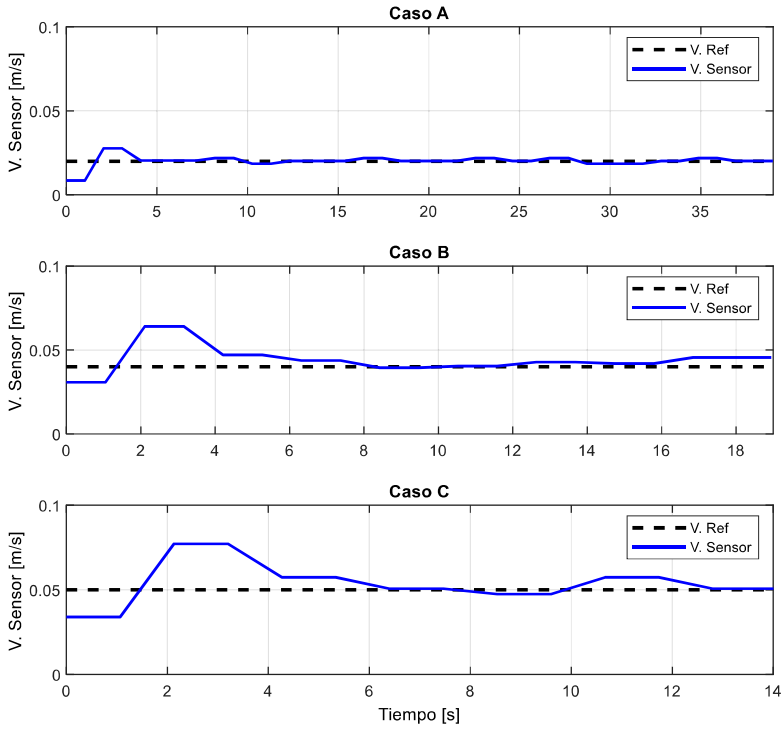


Figura 3.23. Velocidad lineal estimada por el sensor.

Trayectoria Doble frecuencia

Para el seguimiento de trayectoria de doble frecuencia se consideró un radio de 0.25 m, con una velocidad lineal de referencia única de 0.022 m/s. Inicialmente el robot se ubicó en la posición inicial (0.1162, 0.8465) m con una orientación de 262.1° (banderín de color verde en la Figura 3.24). El robot móvil debe alcanzar la posición (0.35, 0.75) m (banderín de color rojo en la Figura 3.24) para posteriormente realizar el seguimiento de trayectoria de doble frecuencia.

El error de posicionamiento inicial del robot móvil fue igual de 0.0142 m. Mientras que, para el seguimiento de trayectoria de doble frecuencia se observa que el error oscila alrededor de 0.02 metros como se muestra en la Figura 3.25. Cuando el robot cambia su orientación al describir la curva de la trayectoria el error puede ser superior a 0.03 m, este error generalmente se produce cuando el robot móvil disminuye su velocidad lineal para tomar la curva de la trayectoria.

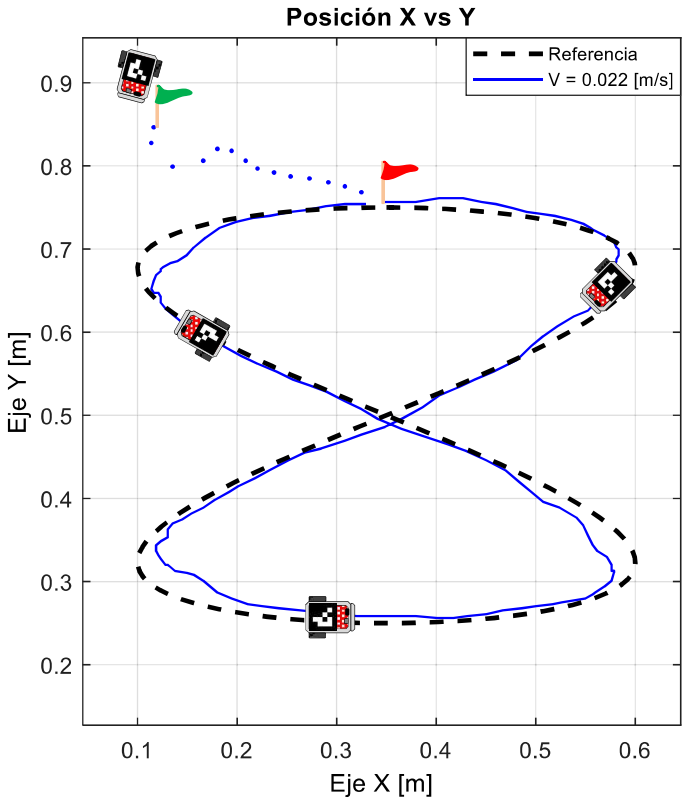


Figura 3.24. Seguimiento de trayectoria de doble frecuencia

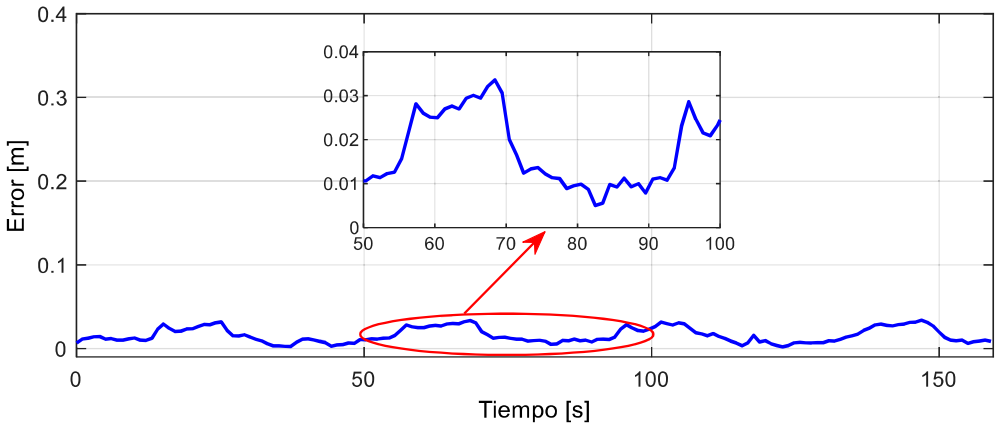


Figura 3.25. Norma del error de seguimiento de trayectoria de doble frecuencia

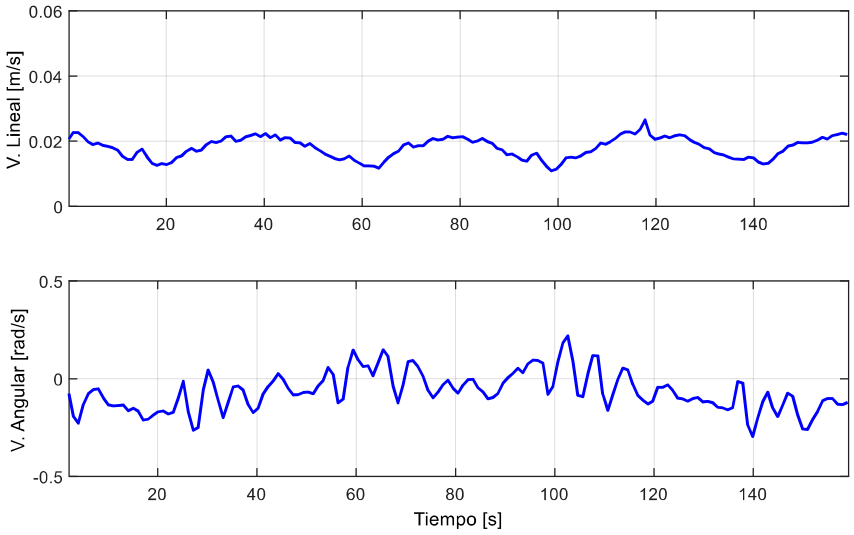


Figura 3.26. Señales de control de velocidad lineal y velocidad angular

Las señales de control de velocidad lineal y angular se presentan en la Figura 3.26, es notable la existencia de variaciones tanto en la velocidad lineal como angular. La velocidad lineal presentó variaciones entre 0.01 m/s y 0.025 m/s aproximadamente, la velocidad lineal disminuye cuando el robot toma la curva de la trayectoria como por ejemplo de 50 a 60 segundos. Para conocer a que velocidad lineal se desplazó el robot móvil, se puede observar en la Figura 3.27 la velocidad estimada a través de los datos del sensor (cámara web). De esta manera es posible apreciar que la velocidad lineal no es constante y presenta mayores oscilaciones, esto se debe a que el robot debe cambiar su orientación en cuatro ocasiones a lo largo de la trayectoria. En el intervalo de 50 a 65 segundos, la velocidad lineal disminuye ya que el robot cambió su orientación para tomar la curvatura de la trayectoria. Cuando el robot logra orientarse se desplaza con una velocidad lineal que se aproxima a 0.02 m/s.

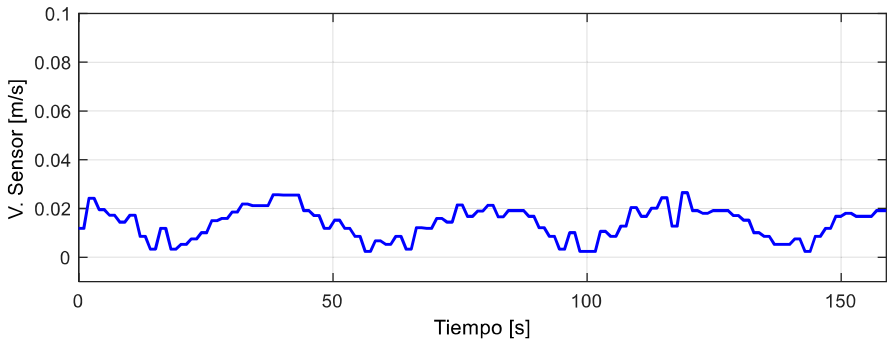


Figura 3.27. Velocidad lineal estimada por el sensor.

3.6 Resultados obtenidos del control implementado para dos robots

En esta sección se presentan los resultados más relevantes del control de posición y seguimiento de trayectorias cuando se trabaja con los dos robots. También se presentan las señales de control para las velocidades angulares de cada rueda de los robots móviles, obtenidas de la técnica de control escogida. Conforme a la sección 2.5, para la visualización del comportamiento de los robots móviles se utilizó la interfaz gráfica y la ventana de monitoreo como se muestra en la Figura 3.28.

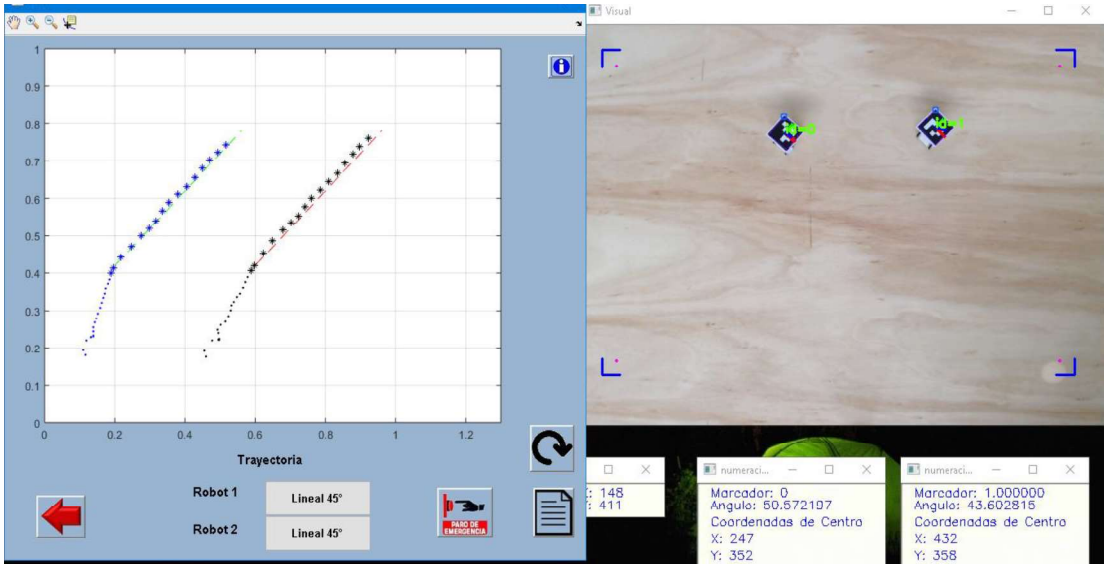


Figura 3.28. Pantalla compartida (Interfaz y monitoreo).

Prueba con dos robots trayectoria lineal con pendiente de 45°

Una vez ingresadas las constantes de control se utilizaron los dos robots con sus respectivos marcadores de referencia matricial. Para seguir una trayectoria lineal con pendiente de 45° y una distancia de 0.8 metros. Inicialmente el robot 1 se ubicó en la posición (0.11, 0.16) m con una orientación de 55.4° y el robot 2 se ubicó en la posición (0.46, 0.18) m con una orientación de 21.3° (banderín de color verde en la Figura 3.29). Como primer objetivo el robot 1 móvil debe alcanzar la posición (0.40, 0.18) m y el robot 2 debe alcanzar la posición (0.58, 0.40) m (banderín de color rojo en la Figura 3.29). Posteriormente, empieza el seguimiento de trayectoria lineal con pendiente de 45° y una distancia de 0.8 metros, ante los distintos valores de velocidad lineal establecidos.

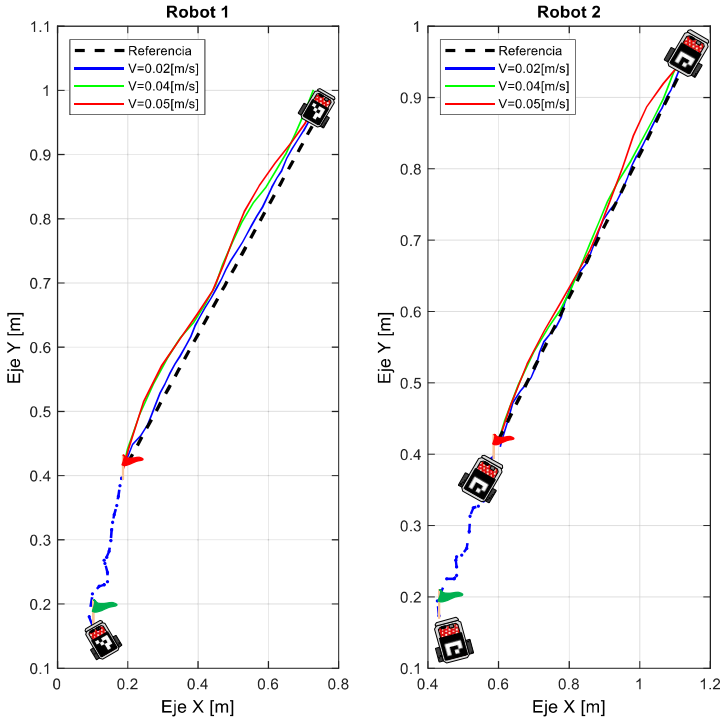


Figura 3.29. Seguimiento de trayectoria a distintas velocidades para dos robots

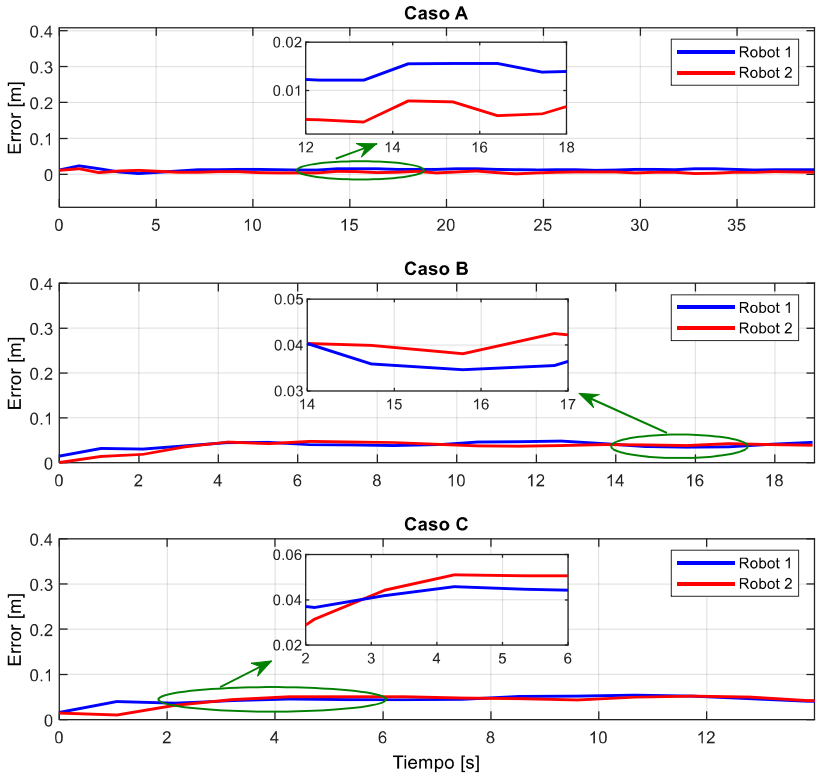


Figura 3.30. Norma del error de seguimiento de trayectoria a diferentes velocidades

El Robot 1 presentó un error de posicionamiento inicial de 0.16 metros y el Robot 2 presentó un error de 0.011 metros alejados del punto inicial. En la Figura 3.30 se indica la norma del error de seguimiento de trayectoria de los dos robots. En el Caso A es posible notar que el Robot 1 presenta mayor error que el Robot 2 ya que internamente el sistema presenta una alta carga computacional cuando existen dos robots presentes, es decir, el robot que mayor error presenta es el último en procesar. Además, en este caso existe una gran cantidad de datos ya que la trayectoria toma más tiempo para ser cumplida. En el Caso B el error llega a un valor de 0.04 metros al igual que en el Caso C el valor del error incrementó ya que la velocidad lineal establecida es mayor en los dos casos.

En la Figura 3.31. se indican las señales de control de velocidad lineal establecidas. En el Caso A se presentan valores entre 0.02 m/s hasta aproximadamente 0.025 m/s. En el Caso B el controlador calculó valores de velocidad lineal menores a 0.04 m/s cumpliendo con valores bajo la velocidad lineal establecida por el usuario (0.04 m/s). En el caso C los valores de velocidad lineal están bajo la referencia de 0.05 m/s ingresada por el usuario.

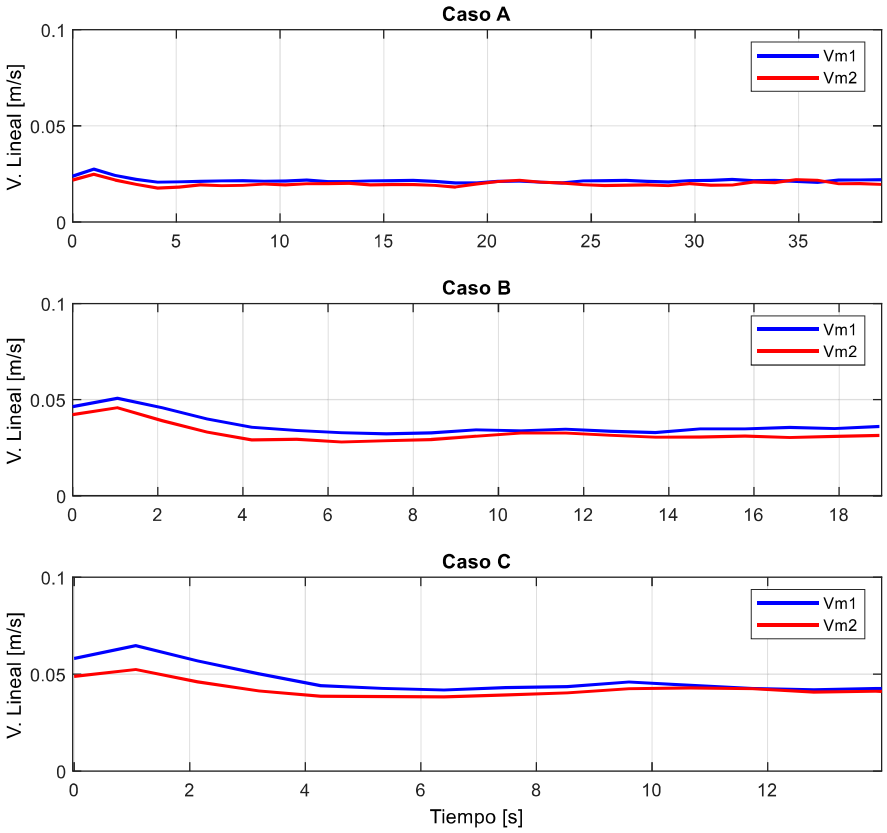


Figura 3.31. Señales de control de velocidad lineal para dos robots

En la Figura 3.32 se indican las señales de control de velocidad angular ante los diferentes casos establecidos. Teóricamente esta velocidad debería ser 0 rad/s ya que es una trayectoria lineal. En el Caso A se puede observar mayor número de oscilaciones ya que como se explicó anteriormente las ruedas del robot giran ligeramente debido a la velocidad de 0.01 m/s. En el Caso B y Caso C existen mayores valores de velocidad angular ya que la orientación del robot se ve afectada por la rapidez de los robots, intentando mantener una orientación de 45° además de acercarlo a la trayectoria en lo mejor posible.

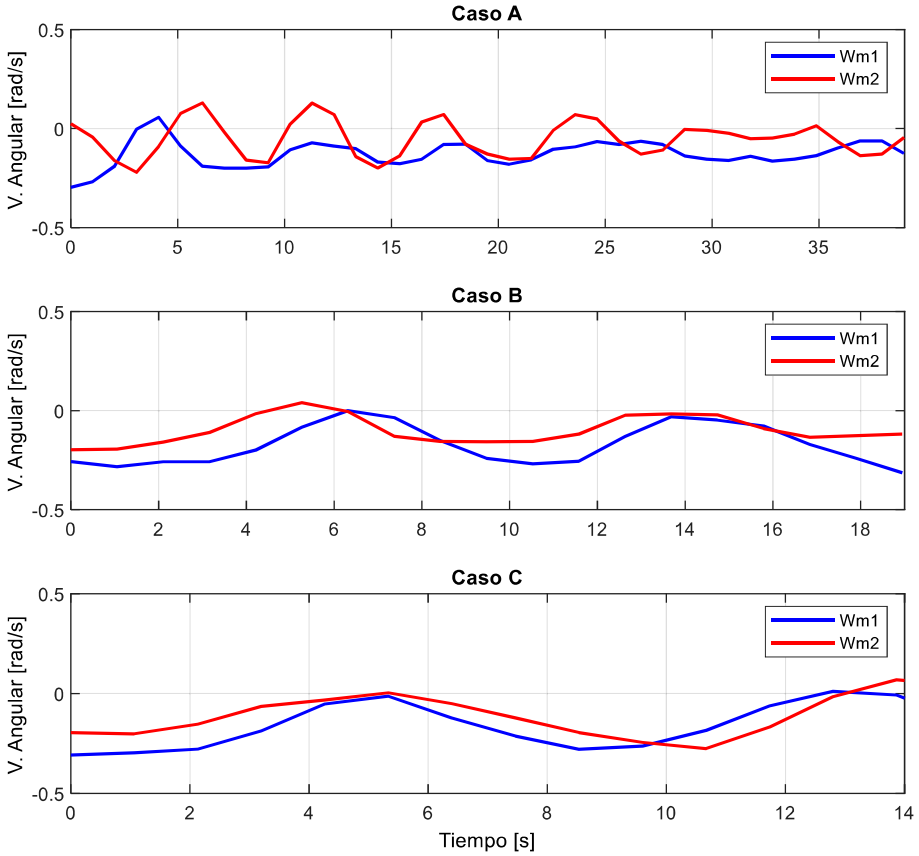


Figura 3.32. Señales de control de velocidad angular para dos robots

En la Figura 3.33 se indica la velocidad estimada por el sensor ante la trayectoria predefinida. En el Caso A los valores de velocidad estimada alcanzan los 0.02 m/s. En el Caso B la velocidad estimada alcanza un valor mayor a la de referencia llegando alrededor de 0.042 m/s. En el Caso C el robot mantiene una velocidad lineal estimada de 0.05 m/s. Para todos los casos es posible observar que el tiempo de establecimiento es de 8 segundos donde el sistema intenta alcanzar la referencia de velocidad lineal establecida.

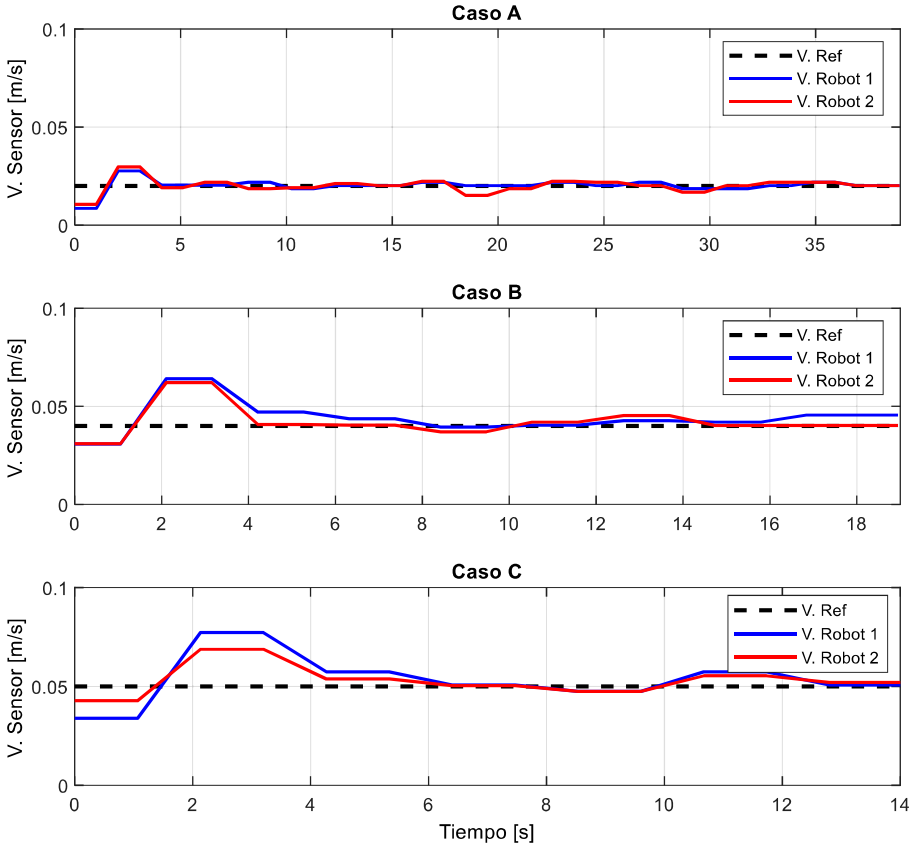


Figura 3.33. Velocidad lineal estimada por el sensor para dos robots.

3.7 Análisis de perturbaciones aplicadas al seguimiento de trayectoria de doble frecuencia

En la Figura 3.34 se indica el comportamiento del robot con la técnica de control implementada ante tres perturbaciones en 11, 51 y 81 segundos, estas perturbaciones consisten en alejar manualmente al robot de la trayectoria aproximadamente 0.15 m. Inicialmente el robot se encuentra en la posición (0.1114, 0.8276) m, con una orientación de 82.33°. El robot como primer objetivo debe moverse a la posición (0.35, 0.75) m, y luego describir la trayectoria de doble frecuencia cuyo radio es de 0.25 m.

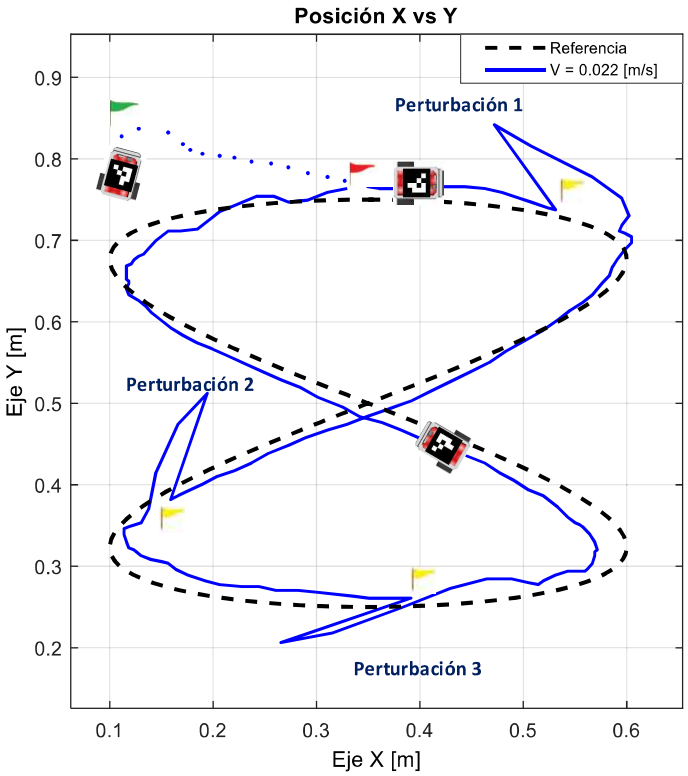


Figura 3.34. Seguimiento de trayectoria de doble frecuencia con perturbaciones.

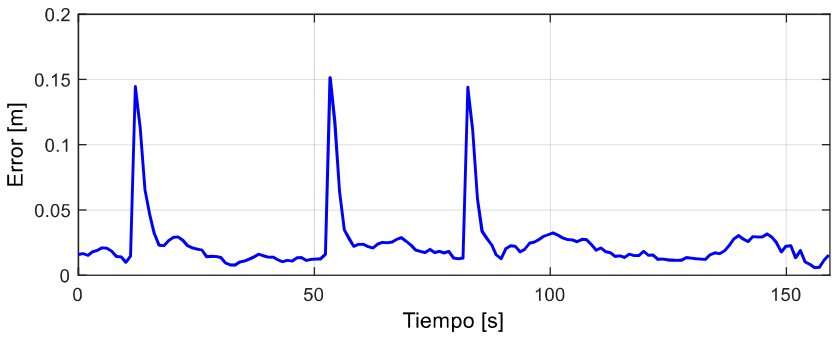


Figura 3.35. Norma del error de seguimiento de para trayectoria de doble frecuencia con perturbaciones.

El error de posicionamiento inicial del robot fue de 0.0184 m. En la Figura 3.35 es posible observar que existe un incremento del error desde 0.02 a 0.15 metros en cada perturbación realizada, y luego de haber transcurrido aproximadamente 8 segundos el error disminuye hasta 0.02 m.

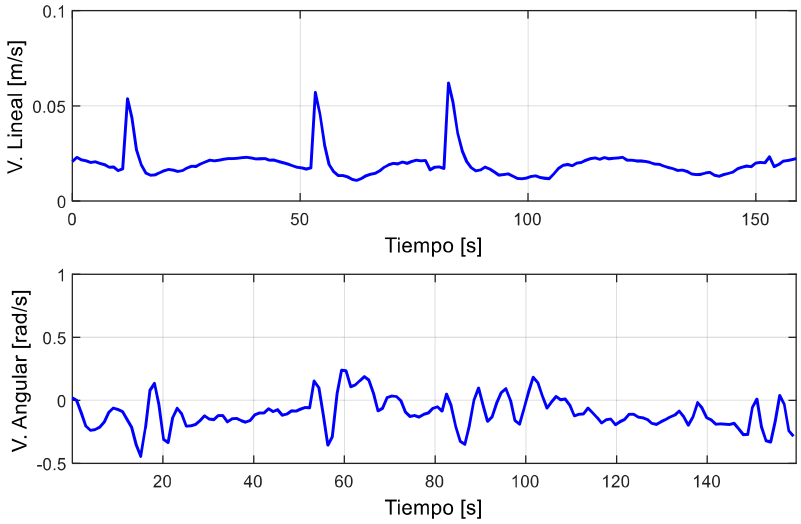


Figura 3.36. Señales de control de las velocidades lineal y angular del robot móvil.

Las señales de control de velocidad lineal y angular del robot móvil para el seguimiento de trayectoria de doble frecuencia con perturbaciones se indican en la Figura 3.36. El controlador calcula valores de velocidad lineal de hasta 0.06 m/s cuando las perturbaciones se presentan.

Es posible observar que la velocidad lineal tiende a 0.02 m/s en 8 segundos después de haber ocurrido la perturbación. Por otro lado, la velocidad angular que se observa en la Figura 3.36 presenta mayores oscilaciones durante y después de las perturbaciones ya que la orientación del robot se ve afectada.

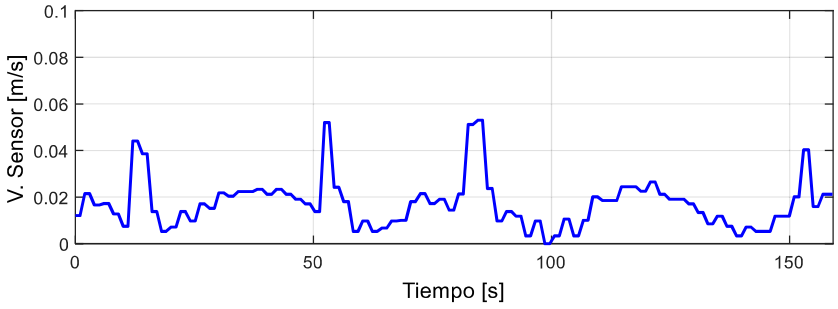


Figura 3.37. Velocidad lineal estimada por el sensor.

En la Figura 3.37 se puede observar la velocidad lineal estimada con la que se desplaza el robot móvil. La velocidad lineal aumenta hasta 0.05 m/s aproximadamente cada vez que se presentan las perturbaciones en los instantes de tiempo 11, 51 y 81 segundos.

4. CONCLUSIONES

- Una vez realizada la correspondiente revisión de la literatura acerca del modelo matemático del robot uniciclo se utilizó su modelo cinemático con restricción no holonómica mejorada, a fin de revisar diferentes técnicas de control de robots móviles para el control de posición y seguimiento de trayectorias.
- Debido a que el modelo matemático elegido puede ser aplicado en robots móviles con ruedas, cuya carga respecto a su estructura es despreciable y considerando a la vez que los movimientos de traslación ocurren a bajas velocidades, en el presente proyecto se procedió al ensamblaje de dos prototipos de robots diferenciales de dimensiones y masas establecidas para ser clasificados como robots de tamaño reducido.
- Se desarrolló un banco de pruebas multi-robot integrado con una cámara y comunicación inalámbrica entre los robots y una computadora central. El propósito de este banco de pruebas es permitir investigaciones futuras dentro de la Escuela Politécnica Nacional en el área de robótica móvil considerando robots de tamaño reducido.
- De acuerdo con desarrollo del sistema de monitoreo en el presente proyecto se adquirió una cámara web de alta definición junto con su correspondiente soporte. Se escogió una altura de la cámara de 1.50 metros permitiendo al sistema de monitoreo visualizar al robot mientras se encuentre dentro del espacio de trabajo de 1 x 1.4 metros cuadrados.
- Se desarrolló el sistema de monitoreo en base a la vinculación de Visual Studio 2015 con las librerías de Open CV. El desarrollo del procesamiento de imagen dependió de la calibración previa de la cámara, la cual permitió eliminar la distorsión y mejorar la precisión del sistema de coordenadas.
- La creación de marcadores de referencia determinó el método de localización del robot, este procesamiento constó de varias etapas como conversión a escala de grises, binarización de imagen y filtrado de contornos.
- Mediante el uso de Matlab Engine se envió la información propia del marcador como coordenadas de posición, ángulo de orientación e identificador numérico desde Visual Studio hacia el *Workspace* de Matlab. El software de desarrollo de interfaces gráficas GUIDE permitió diseñar una interfaz gráfica con ingreso y salida de datos, alarmas visuales, además de visualización compartida durante el funcionamiento del banco de pruebas.

- El control de posición y seguimiento de trayectoria de los robots móviles se realizó mediante del cálculo de las acciones de control correspondientes a las velocidades lineales y angulares de cada robot. Para el proyecto se realizó la conversión de estas acciones de control con la finalidad de conseguir las velocidades de accionamiento de las ruedas para cada uno, las mismas que permiten el desplazamiento de los robots a través del área de trabajo establecida.
- Se determinó el tiempo de muestreo del sistema a partir de los retardos generados en el software utilizado, como el tiempo de envío y recepción de datos. La vinculación de Visual Studio (procesamiento de imagen) con Matlab (controlador) provocó un aumento de la carga computacional al ordenador.
- Con la técnica de control basada en métodos numéricos que se implementó en el presente proyecto se obtuvo un buen desempeño del sistema de control al momento de conducir los robots móviles a través de diferentes tipos de trayectorias. Por otro lado, mediante la experimentación realizada se comprobó que el algoritmo de control propuesto presentaba un error bajo.
- Las pruebas obtenidas fueron satisfactorias para los dos robots con las constantes de controlador recomendadas. Se analizó conjuntamente el comportamiento de los robots ante diferentes velocidades lineales y trayectorias establecidas, por medio de los resultados se observó el grado en que la postura de los robots se vio afectada.
- A través de las pruebas realizadas se determinó los factores que afectan al sistema como exceso de iluminación y exceso de carga computacional. El banco de pruebas se ve afectado generando oscilaciones en la posición de los robots durante el seguimiento de trayectoria hasta volver a condiciones normales.
- Los futuros trabajos pueden considerar formación de robots móviles a partir del banco de pruebas implementado, donde se podría incluir evasión de obstáculos y mejoras en la estructura física del robot. Además, se podría vincular distintos tipos de robots creando un banco de pruebas heterogéneo.

RECOMENDACIONES

- El procesamiento de imagen además del envió de datos a los robots consume gran parte de recursos del ordenador, se concluye que la unión de diferentes entornos como Matlab 2018Ra y Visual Studio 2015 requiere un ordenador con mejores características de memoria RAM y gráfica, así como de procesador.
- Es necesario utilizar bases planas y rígidas para los marcadores de referencia matricial ya que en el caso de encontrarse arrugado o doblado el programa no

es capaz de detectarlo ya que se está eliminando contenido interno binario. Además, la superficie sobre la que se deslizan los robots debe ser lo más lisa posible para evitar que el robot desgaste sus ruedas además de posibles obstrucciones de la rueda de soporte.

- Antes de dar marcha al sistema, es necesario configurar el enfoque de la cámara de automático a manual evitando que la cámara busque condiciones de enfoque en cada instante provocando un efecto intermitente.
- Los robots implementados no disponen de un sensor de velocidad, para la medición de la velocidad el presente proyecto realizó una estimación mediante el sensor de visualización con el fin de comprobar la rapidez de los robots. En algunos casos esta velocidad superó el valor de referencia debido a la ausencia del control de velocidad por lo que se recomienda trabajar dentro de los límites de velocidad lineal cumpliendo con el rango de velocidades permitidos en la interfaz gráfica.
- La interfaz gráfica permite al usuario verificar el estado de la batería, es recomendable verificar que el valor de la batería no disminuya del valor recomendado de 3.4 voltios.
- Se recomienda antes de inicializar el sistema conectar el módulo Xbee Explorer al ordenador y verificar su estado mediante el análisis de comunicación provisto en la interfaz gráfica.
- La interfaz gráfica permite al usuario detener el sistema en el caso de presentar fallas. El botón de paro de emergencia se puede pulsar en cualquier momento que requiera el usuario.

5. REFERENCIAS BIBLIOGRÁFICAS

[1] E. Zürich, U. o. Montréal, N. C. T. University y T. T. I. a. Chicago, «Duckietown,» 2016. [En línea]. Available: duckietown.org/index.html

[2] D. Pickem, P. Glotfelter, L. Wang, M. Mote, A. Ames, E. Feron y M. Egerstedt, «The Robotarium: A remotely accessible swarm robotics research testbed,» IEEE International Conference on Robotics and automation (ICRA), vol. I, 2016.

[3] G. Caprari, in Autonomous micro-robots: applications and limitations, Lausanne: Escuela Politécnica Federal de Lausana, 2003, pp. 16-17.

[4] A. DOBRA, "General classification of robots. Size criteria," 23rd International Conference on Robotics in Alpe-Adria-Danube Region, 2014.

[5] S. Garrido Jurado, R. Medina Carnicer, F. J. Madrid Cuevas y R. Muñoz Salinas, «Generation of fiducial marker dictionaries using Mixed Integer Linear Programming,» *Pattern Recognition,* vol. 6, pp. 2280-2292, 2015.

[6] G. Bermúdez, «Robots móviles. Teoría, aplicaciones y experiencias.,» TECNURA, nº 10, pp. 1-12,Colombia: Universidad Distrital "Francisco José de Caldas", 2002.

[7] M. A. Molina Villa y E. L. Rodríguez Vásquez, Flotilla de robots para trabajos en robótica cooperativa., Bogotá: Universidad Militar Nueva Granada, 2014.

[8] J. L. Muñoz, Robótica móvil. Estudio y caracterización del robot móvil Kjunior desarrollo de aplicación de robot laberinto., Cartagena: Universidad Politécnica de Cartagena, 2011.

[9] A. Jiménez Gonzáles, J. R. Martinez DeDios y A. Ollero, «Testbeds for ubiquitous robotics: A survey,» ,España: University of Seville, 2013.

[10] J. Yu, S. Han y D. Rus, A portable, 3D-printing enabled multi-vehicle platform for robotics research and education., Cambridge:Rutgers University, 2016.

[11] C. C. Limited, «hackernoon.com,» [En línea]. Available: https://hackernoon.com/unicycle-to-differential-drive-courseras-control-of-mobile-robots-with-ros-and-rosbots-part-2-6d27d15f2010. [Último acceso: 26 Septiembre 2018].

[12] K. Hernández y F. Panchi, Diseño, simulación e implementación de un sistema de teleoperación para un robot móvil tipo carlike., Quito: Escuela Politécnica Nacional, 2017.

[13] G. Andaluz, V. Andaluz y A. Rosales, Modelación, identificación y control de robots móviles, Quito: Escuela Politécnica Nacional, 2012.

[14] L. J. Capito Ruiz y P. A. Proaño Chamorro, Seguimiento de trayectorias mediante cuatro técnicas de control utilizando una plataforma robótica Pioneer 3DX y el sistema operativo robótico ROS., Quito: Escuela Politécnica Nacional, 2015.

[15] R. Guamán y W. Lozada, Control de posición y seguimiento de trayectorias de un manipulador móvil de 3 grados de libertad., Quito: Escuela Politécnica Nacional, 2016.

[16] B. Arevalo, Diseño e implementación de un controlador SMC-Delay para el seguimiento de trayectoria de una formación de robots móviles con retardo en la entrada, Quito: Escuela Politécnica Nacional, 2018.

[17] «Educación Inclusiva,» [En línea]. Available: http://www.ite.educacion.es/formacion/materiales/129/cd/unidad_1/mo1_mecanismo_de_la_vision.htm. [Último acceso: 8 Septiembre 2018].

[18] A. Kaehler y G. Bradski, «What Is Computer Vision?,» de *Learning OpenCV 3*, vol. II, D. Schanafelt, Ed., Tokyo, O'Reilly Media. Inc, 2017, p. 3.

[19] R. Szeliski, "What is computer vision?," in *Computer Vision: Algorithms and Applications*, Seattle, 2010.

[20] K. Pulli, A. Baksheev, K. Kornyakov y V. Eruhimov, «Mobile computer-vision technology will soon become as ubiquitous as touch interfaces,» in *Proceedings Realtime Computer Vision,* 2012.

[21] X. Zhang, S. Fronz and N. Navab, "Visual marker detection and decoding in AR systems: a comparative study," in *Proceedings. International Symposium on Mixed and Augmented Reality*, Darmstadt, 2002.

[22] «cnet,» DigiCert Inc, [En línea]. Available: https://www.cnet.com/es/noticias/casa-blanca-realidad-aumentada/. [Último acceso: 4 Febrero 2019].

[23] Open CV Team, "About Open CV," 2018. [Online]. Available: https://opencv.org/about.html.

[24] A. R. Bazaga, "OpenCV: Librería de Visión por Computador," Universidad de La Laguna , 18 Agosto 2015. [Online]. Available: https://osl.ull.es/software-libre/opencv-libreria-vision-computador/. [Accessed 2018].

[25] © Microsoft , "Visual Studio IDE," Microsoft, 22 Agosto 2018. [Online]. Available: https://docs.microsoft.com/en-us/visualstudio/ide/visual-studio-ide?view=vs-2015. [Último acceso: Septiembre 2018].

[26] © Microsoft, "Visual Studio," Microsoft, 2018. [Online]. Available: https://visualstudio.microsoft.com/es/.

[27] Z. Zhang, "A Flexible New Technique for Camera Calibration," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 2009.

[28] J. Mallon and P. Whelan, "Which pattern? Biasing aspects of planar calibration patterns and detection methods," *Pattern Recognition Letters,* vol. 28, no. 8, pp. 921-930, June 2017.

[29] Y. Wang, Y. Li and J. Zheng, "A camera calibration technique based on OpenCV," *The 3rd International Conference on Information Sciences and Interaction Sciences,* 2010.

[30] R. Salinas Muñoz, ArUco: An efficient library for detection of planar markers and camera pose estimation, vol. I, Córdoba: Universidad de Córdoba, 2018.

[31] J. Lee, K.-C. Lee, S. Cho and S.-H. Sim, "Computer Vision-Based Structural Displacement," Korea: Ulsan National Institute of Science and Technology, 02 July 2017.

[32] Sphinx 1.3.6., «Calibration and 3D Reconstruction,» OpenCV 2.4.13.7 documentation , 2014. [En línea]. Available: https://docs.opencv.org/2.4/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html#zhang2000. [Último acceso: 2018].

[33] Y. Xin and S. Zhu, "A Calibration Method Based on OpenCV," in *2011 3rd International Workshop on Intelligent Systems and Applications*, Wuhan, 2011.

[34] T. Sriram, K. Vishwanatha Rao and S. . Biswas, “Applications of barcode technology in automated storage and retrieval systems,” in *Proceedings of the IEEE IECON. 22nd International Conference on Industrial Electronics, Control, and Instrumentation*, Taipei.

[35] Wikipedia®, “Código de barras,” Fundación Wikimedia, Inc, 2018. [Online]. Available: https://es.wikipedia.org/wiki/C%C3%B3digo_de_barras. [Accessed 2018].

[36] S. Tiwari, “An Introduction To QR Code Technology,” in *International Conference on Information Technology*, Jabalpur, 2016.

[37] Doxygen, «Detection of ArUco Markers,» 18 Diciembre 2015. [En línea]. Available: https://docs.opencv.org/3.1.0/d5/dae/tutorial_aruco_detection.html. [Último acceso: 2018].

[38] Doxygen, "dictionary.hpp File Reference," OpenCV, 2018. [Online]. Available: https://docs.opencv.org/trunk/dc/df7/dictionary_8hpp.html. [Accessed 2018].

[39] D. Baggio, S. Emani, D. Millan, K. Levgen, N. Mahmood, J. Saragih y R. Shilkrot, Mastering OpenCV with Practical Computer Vision Projects, BIRMINGHAM: PACKT, 2012, pp. 62-83.

[40] Doxygen, "Detection of ArUco Markers," Open CV, 2015. [Online]. Available: https://docs.opencv.org/3.1.0/d5/dae/tutorial_aruco_detection.html. [Accessed 2018].

[41] R. Faludi, Building wireless sensor networks, Cambridge: O’REILLY, 2011.

[42] R. Vizuete, Diseño y simulación de algoritmos de control distribuido para la formación y seguimiento de trayectoria de robots móviles tipo uniciclo y para manipuladores móviles de tres grados de libertad, Quito:Escuela Politécnica Nacional, 2017.

[43] B. Sarabia, Diseño y simulación de tres técnicas de control clásicas y robustas aplicadas al seguimiento de trayectorias ante la presencia de retardos fijos para

la plataforma robótica PIONEER 3DX, Quito: Escuela Politécnica Nacional, 2017.

[44] P. R. &. Electronics, «www.pololu.com,» [En línea]. Available: https://www.pololu.com/product/1511. [Último acceso: 29 Septiembre 2018].

[45] P. R. &. Electronics, «www.pololu.com,» [En línea]. Available: https://www.pololu.com/product/2135/specs. [Último acceso: 30 Septiembre 2018].

[46] S. Electronics, «www.sparkfun.com,» [En línea]. Available: https://www.sparkfun.com/products/11215. [Último acceso: 30 Septiembre 2018].

[47] Sparkfun, «learn.sparkfun.com,» [En línea]. Available: https://learn.sparkfun.com/tutorials/pro-micro--fio-v3-hookup-guide. [Último acceso: 30 Septiembre 2018].

[48] adafruit, «www.adafruit.com,» [En línea]. Available: https://www.adafruit.com/product/1578. [Último acceso: 1 Octubre 2018].

[49] www.pololu.com, «Pololu Robotics & Electronics,» [En línea]. Available: https://www.pololu.com/product/1088. [Último acceso: 2 Octubre 2018].

[50] www.sparkfun.com, «SparkFun Electronics,» [En línea]. Available: https://www.sparkfun.com/products/8909. [Último acceso: 2 Octubre 2018].

[51] KYE System Corp, "Eye 110," Genius, 2018. [Online]. Available: http://us.geniusnet.com/supports/eye-110.

[52] Logitech, "HD Webcam C920," 2011. [Online]. Available: https://www.arp.nl/webmedias/datasheet/575183859ce969320b1a2beb.pdf.

[53] Doxygen, «Drawing Functions,» OpenCV, Diciembre 2015. [En línea]. Available: https://docs.opencv.org/3.1.0/d6/d6e/group__imgproc__draw. [Último acceso: Septiembre 2018].

[54] opencv dev team., "Camera Calibration and 3D Reconstruction," OpenCV 2.4.13.7 documentation, 2014. [Online]. Available: https://docs.opencv.org/2.4/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html.

[55] opencv dev team, "Load, Modify, and Save an Image¶," OpenCV 2.4.13.7 documentation, 2014. [Online]. Available: https://docs.opencv.org/2.4/doc/tutorials/introduction/load_save_image/load_save_image.html. [Accessed 2018].

[56] Doxygen, "Miscellaneous Image Transformations," OpenCV, 2015. [Online]. Available: https://docs.opencv.org/3.1.0/d7/d1b/group__imgproc__misc.html#gae8a4a146d1ca78c626a53577199e9c57. [Accessed 2018].

[57] opencv dev team, "Basic Thresholding Operations" OpenCV 2.4.13.7 documentation, 2014. [Online]. Available: https://docs.opencv.org/2.4/doc/tutorials/imgproc/threshold/threshold.html. [Accessed 2018].

[58] doxygen 1.8.12, "Image Thresholding," OpenCV, 2018. [Online]. Available: https://docs.opencv.org/3.4/d7/d4d/tutorial_py_thresholding.html. [Accessed 2018].

[59] Doxygen, "Structural Analysis and Shape Descriptors," OpenCV, 2015. [Online]. Available: https://docs.opencv.org/3.1.0/d3/dc0/group__imgproc__shape.html#ga17ed9f5d79ae97bd4c7cf18403e1689a. [Accessed 2018].

[60] Sphinx, "Reading and Writing Images and Video," Opencv dev team, Nov 2018. [Online]. Available: https://docs.opencv.org/2.4/modules/highgui/doc/reading_and_writing_images_and_video.html. [Accessed 2018].

[61] H. Li and W. Ye, "Hybrid Programming Based on Matlab Engine Technology," in *2013 International Conference on Computer Sciences and Applications*, Wuhan, 2013.

[62] The MathWorks, Inc., "engOpen (C and Fortran)," 2006. [Online]. Available: https://la.mathworks.com/help/matlab/apiref/engopen.html. [Accessed 2018].

[63] The MathWorks, Inc., " mxArray ©," 2006. [Online]. Available: https://la.mathworks.com/help/matlab/apiref/mxarray.html. [Accessed 2018].

[64] The MathWorks, Inc., " mxCreateDoubleMatrix (C and Fortran)," 2006. [Online]. Available: https://la.mathworks.com/help/matlab/apiref/mxcreatedoublematrix.html. [Accessed 2018].

[65] The MathWorks, Inc., "engPutVariable (C and Fortran)," 2006. [Online]. Available: https://la.mathworks.com/help/matlab/apiref/engputvariable.html. [Accessed 2018].

[66] The MathWorks, Inc., "mxDestroyArray (C and Fortran)," 2006. [Online]. Available: https://la.mathworks.com/help/matlab/apiref/mxdestroyarray.html.[Accessed 2018].

[67] The MathWorks, Inc., "mxGetPr (C)," 2006. [Online]. Available: https://la.mathworks.com/help/matlab/apiref/mxgetpr.html.[Accessed 2018].

[68] www.digi.com, «DIGI,» [En línea]. Available: https://www.digi.com/products/xbee-rf-solutions/xctu-software/xctu. [Último acceso: 4 Octubre 2018].

[69] Sparkfun, «www.sparkfun.com,» [En línea]. Available: https://www.sparkfun.com/products/11697. [Último acceso: 30 Septiembre 2018].

[70] M. Rodríguez y S. Sandobalín, Diseño y construcción de tres mini robots exploradores cooperativos., Quito: Escuela Politécnica Nacional, 2013.

[71] The International Society of Automation, "ANSI/ISA-101.01-2015, Human Machine Interfaces for Process Automation Systems," 2015. [Online]. Available: https://www.isa.org/store/ansi/isa-10101-2015,-human-machine-interfaces-for-process-automation-systems/41921133. [Accessed 2018].

6. ANEXOS

ANEXO I

Especificaciones componentes electrónicos

Minimotores DC

Figura I.1. Minimotor DC Plástico (Pololu).

Tabla I.1. Características del Minimotor.

Voltaje nominal	4.5 V
Masa	19.5 gramos
Relación de la caja reductora	120:1
Velocidad angular	150 RPM a 4.5 V
Corriente en vacío	130 mA
Torque	1.8 Kg-cm
Máxima corriente con carga	1250 mA

DVR 8835

Figura I.2. Driver para motores DC 8835.

Tabla I.2. Características del DRV 8835.

Voltaje de operación	0 V a 11 V
Corriente máxima	1.5 A
Voltaje terminal lógico	2 V a 7 V
Masa	0.5 gramos
Frecuencia PWM	250 KHz

Módulo XBee S1

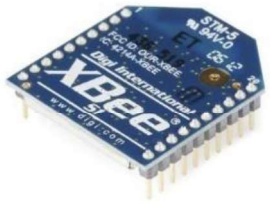


Figura I.3. XBee Serie 1.

Tabla I.3. Características del módulo XBee S1.

Banda	2.4 GHz
Interfaz de datos con velocidad máxima	250Kbps
Voltaje de alimentación	2.8 V a 3.4 V
Corriente de transmisión	45mA a 3.3V
Corriente de recepción	50mA a 3.3 V
Rango de temperatura de operación	-40 a 84 °C
Comunicación	UART
Distancia de comunicación en interiores	30 metros
Distancia de comunicación con línea de vista	100 metros
Pines digitales	8 I/O
Pines de entrada analógica	6 (10 bits)
Pines salida analógica	2 (PWM)
Costo	Bajo
Potencia	Baja
Topologías ZigBee	Estrella, punto a punto

Atmega 32U4

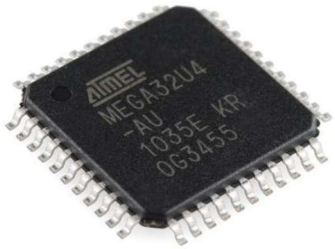


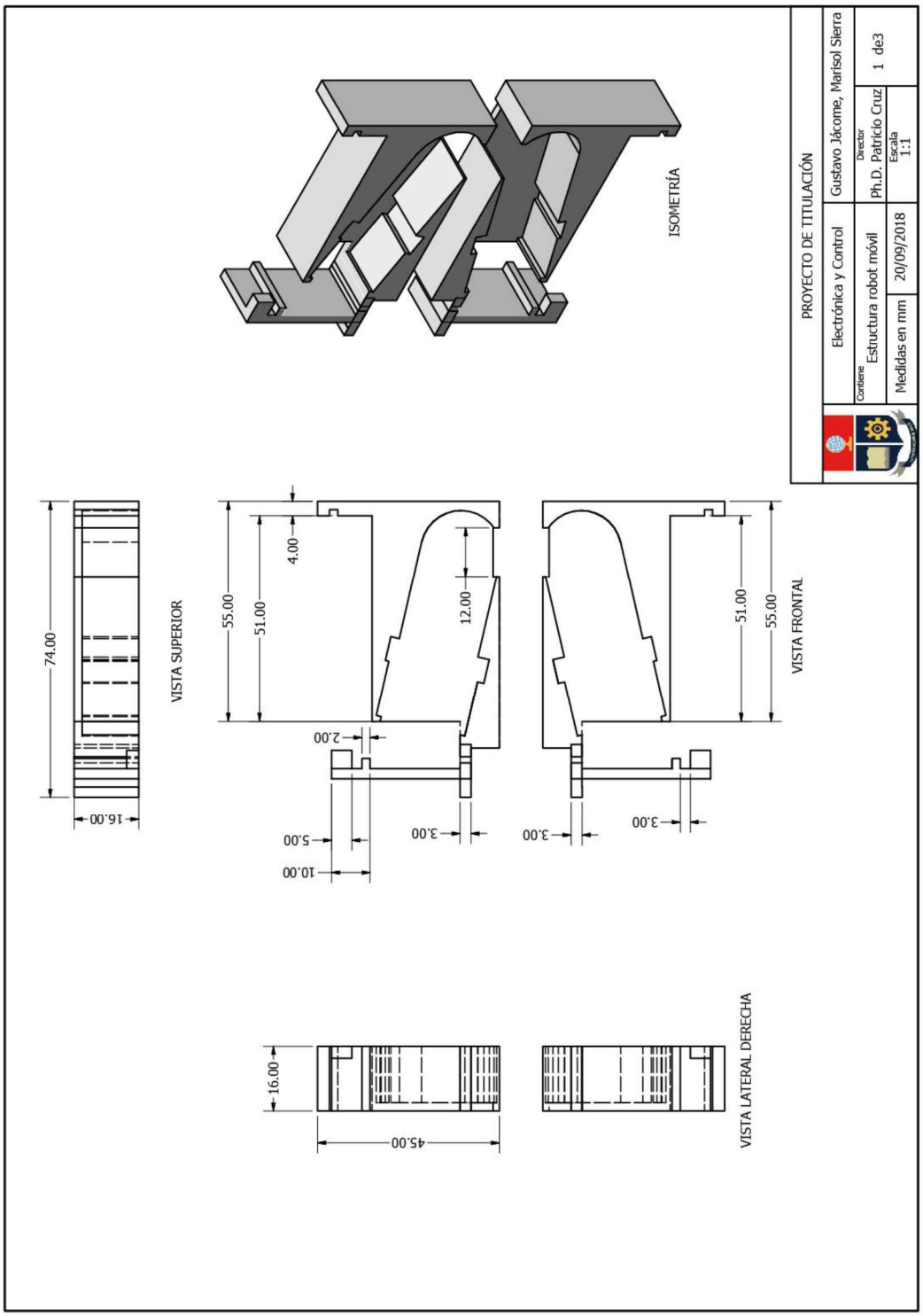
Figura I.4. Microcontrolador Atmega 32U4.

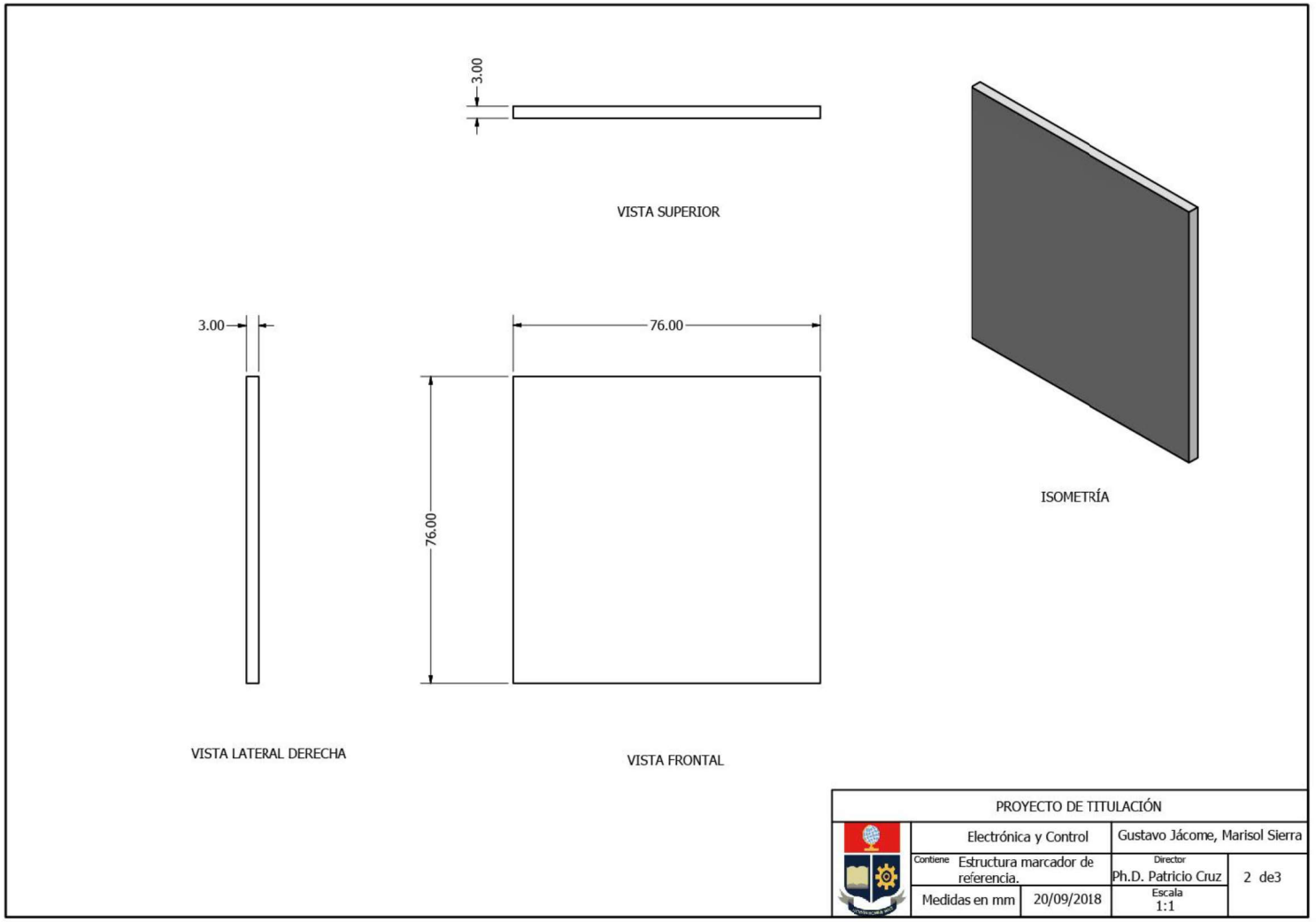
Tabla I.4. Características del microcontrolador Atmega 32U4.

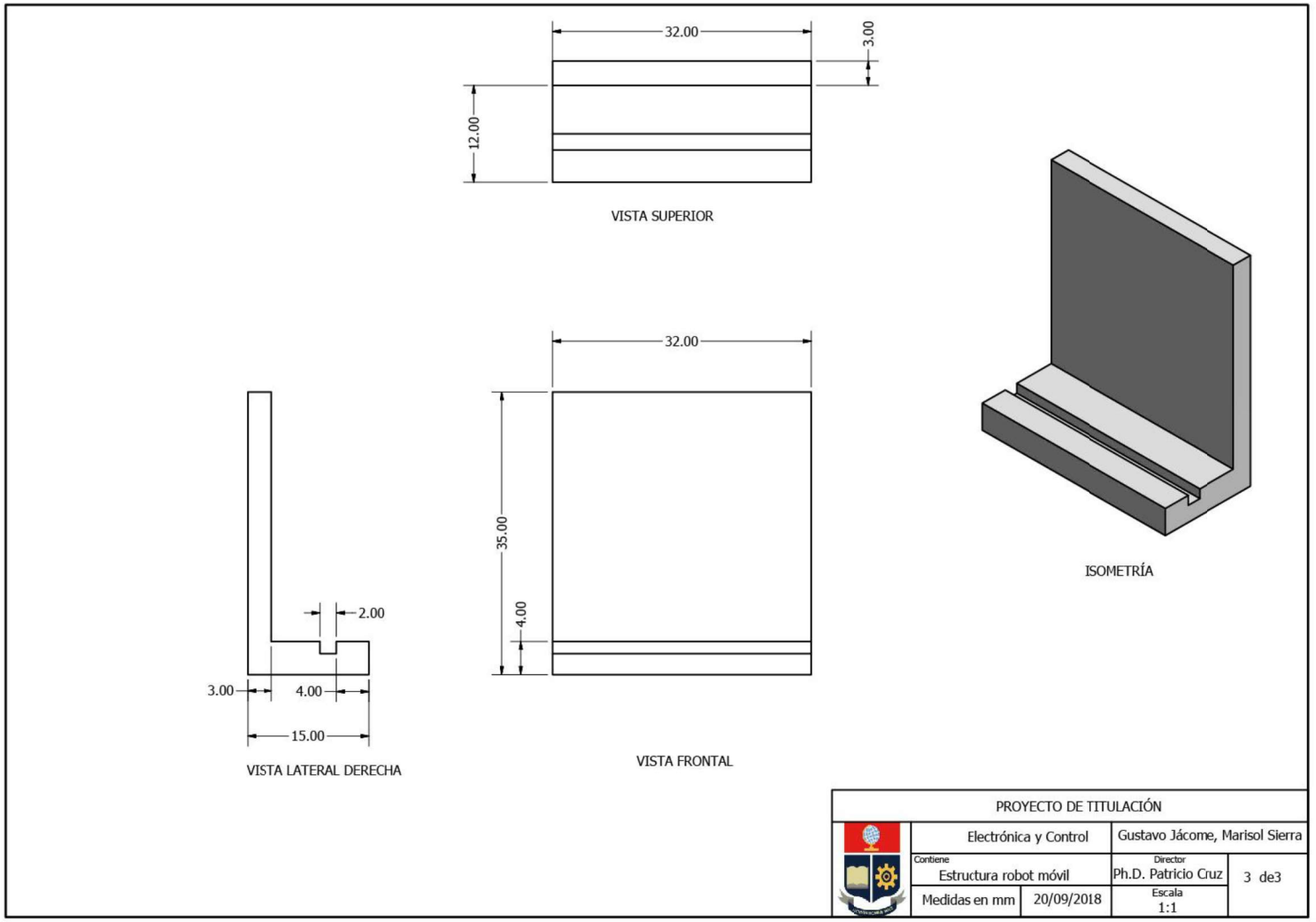
SRAM	2.5 KB
EEPROM	1 KB
Canales PWM (8 bits)	4
Canales PWM (2 a 16 bits)	4
Canales PWM (2 a 11 bits)	6
Canales ADC (10 bits)	12
Voltaje de funcionamiento	2.7 V a 5.5 V
Temperatura de funcionamiento	-40 a 85 °C
Frecuencia de operación	8Mhz a 2.1V / 16MHz a 4.5V
Salidas	CMOS
Entradas	LVTTL

ANEXO II

Piezas de polímero




3.00
VISTA SUPERIOR
ISOMETRÍA
3.00
76.00
76.00
VISTA LATERAL DERECHA
VISTA FRONTAL
PROYECTO DE TITULACIÓN
Electrónica y Control
Gustavo Jácome, Marisol Sierra
Contiene
Estructura marcador de referencia.
Director
Ph.D. Patricio Cruz
2 de3
Medidas en mm
20/09/2018
Escala
1:1


32.00
3.00
12.00
VISTA SUPERIOR
32.00
35.00
4.00
VISTA FRONTAL
2.00
3.00
4.00
15.00
VISTA LATERAL DERECHA
ISOMETRÍA
PROYECTO DE TITULACIÓN
Electrónica y Control
Gustavo Jácome, Marisol Sierra
Contiene
Estructura robot móvil
Director
Ph.D. Patricio Cruz
3 de3
Medidas en mm
20/09/2018
Escala
1:1

ANEXO III

Placa para el módulo DRV8835.

A continuación, se indica en la Figura III.5 y Figura III.6 el diseño de la placa para el módulo DRV 8835 realizado en Ares Proteus 8.7.

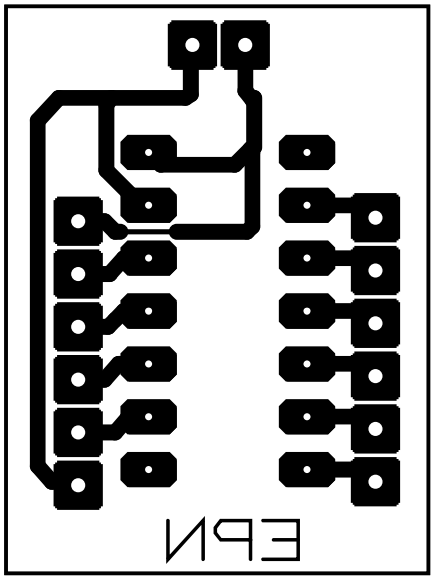


Figura III.5. Vista de las pistas de la placa diseñada.

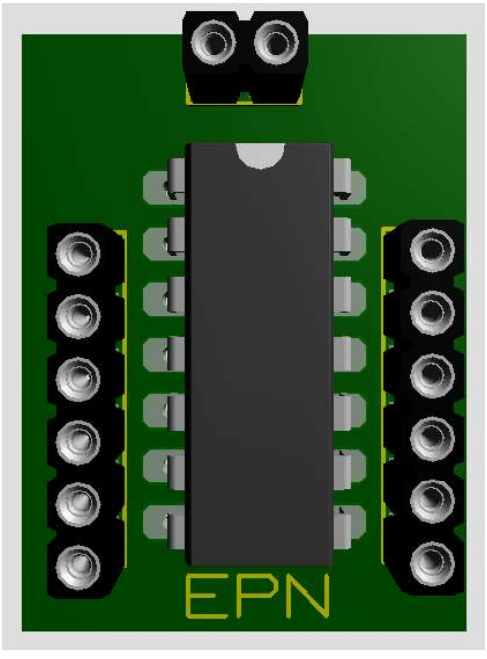


Figura III.6. Vista 3D de la placa diseñada.

ANEXO IV

Manual de Usuario

Es necesario leer toda la guía de usuario para evitar problemas en el funcionamiento del sistema. Los pasos por detallar están orientados para estudiantes y docentes en el caso de poner en marcha al sistema desde condiciones iniciales.

Descripción General del Equipo

El banco de pruebas está constituido por el sistema de monitoreo, sistema de control, e interfaz gráfica como se indica en la Figura IV.7. El proyecto se basa en el uso de un ordenador, cámara de video, prototipos de robot uniciclo, módulos de comunicación inalámbrica y software computacional.

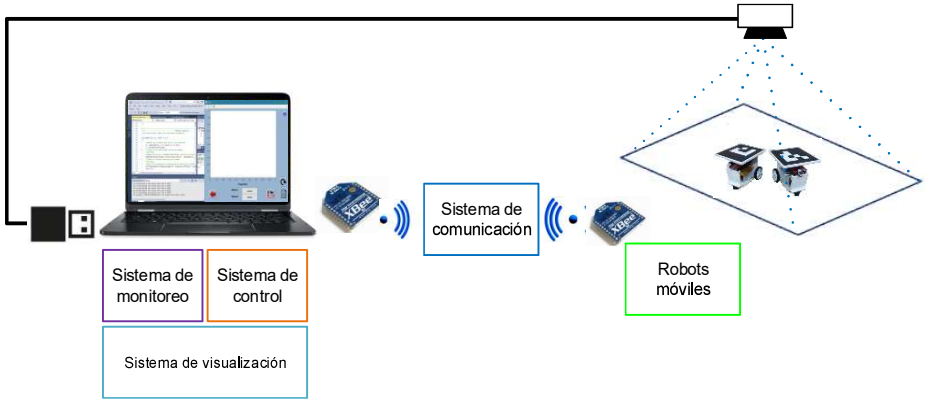


Figura IV.7 Arquitectura del Banco de pruebas

Ordenador Central

El computador central es el cerebro del banco de pruebas, ya que en éste se realizará el control individual de los robots, manejo de visión por computadora, comunicación e interfaz gráfica.

Requisitos que debe cumplir el ordenador:

- *Sistema operativo:*
  - Windows 10
- Software requerido:
  - Visual Studio 2015
  - OpenCV
  - XCTU
  - Maltab R2018a

Inicialización del sistema

Para empezar, es necesario ajustar el enfoque automático de la cámara. Para este procedimiento se ingresa la aplicación cámara del sistema. Al acceder a la cámara es necesario realizar el ajuste manual del enfoque de la cámara como se indica en la Figura IV.8.

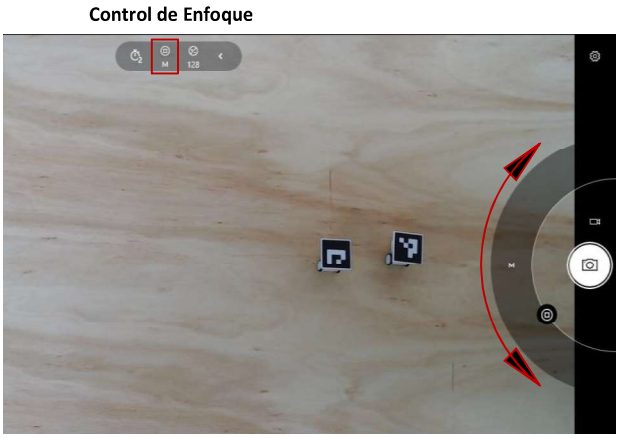


Figura IV.8 Ajuste manual de enfoque de cámara

Posteriormente, se activa el sistema de monitoreo desarrollado en Visual Studio 2015 dando clic en la aplicación *TrackingArUco* como se indica en la Figura IV.9 (a). El programa ejecutado, despliega la ventana de visualización del área de trabajo a través de la cámara como se muestra en la Figura IV.9 (b).

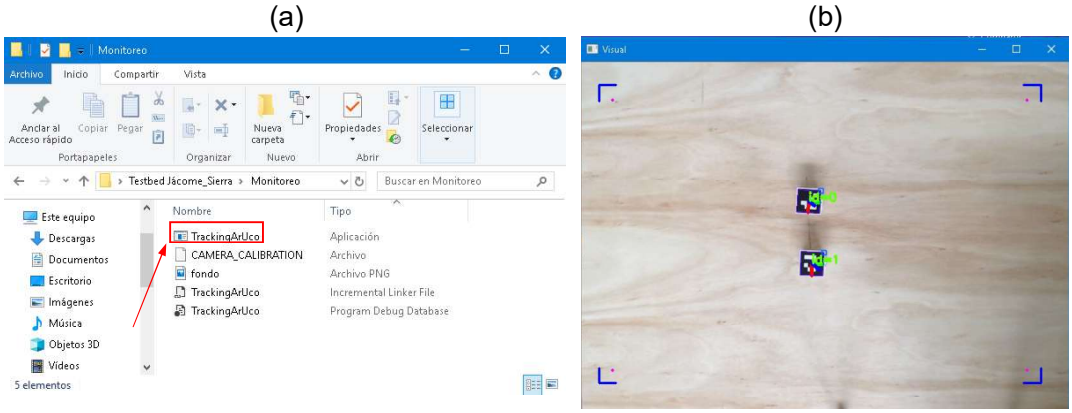


Figura IV.9 Sistema de monitoreo. (a) Aplicación para monitoreo. (b) Visualización de área de trabajo.

Manipulación de robots por parte de Usuario

El usuario debe encender el robot, a través del interruptor provisto por la tarjeta, como se muestra en la Figura IV.10.

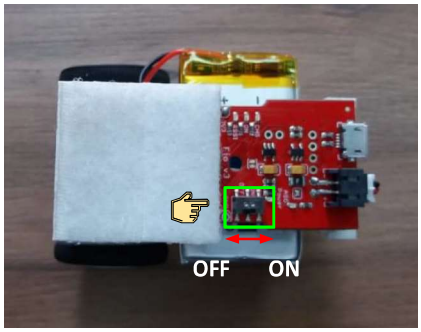


Figura IV.10. Encendido de robot

Se dispone de bases plásticas con el marcador de referencia escogido. Bajo cada robot existe su numeración, como se indica en la Figura IV.11, estos están provistos de velcro para poder colocarlos en la posición correcta.

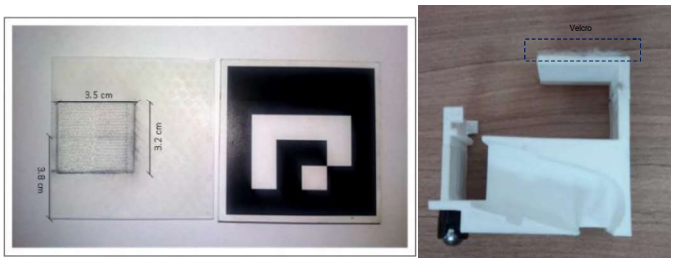


Figura IV.11. Colocación de marcador de referencia sobre robot

Nota: Es necesario colocar el robot sobre la base, antes de la trayectoria para un mejor funcionamiento.

Inicialización de Interfaz Gráfica

Una vez encendidos los prototipos y ejecutado el programa en VS2015, el usuario coloca los robots, sobre la base de madera. La cámara observa los marcadores de referencia, y automáticamente accede al *Command Window* de Matlab Engine como se observa en la Figura IV.12. Para acceder a la versión completa de Matlab R2018a, el usuario debe escribir la palabra *desktop.*

Figura IV.12. Command Window Matlab Engine

Al acceder a Matlab R2018a, el usuario debe ingresar a la carpeta provista por los estudiantes (Testbed Jácome-Sierra) y ejecutar la ventana *Principal*, como se indica en la Figura IV.13.

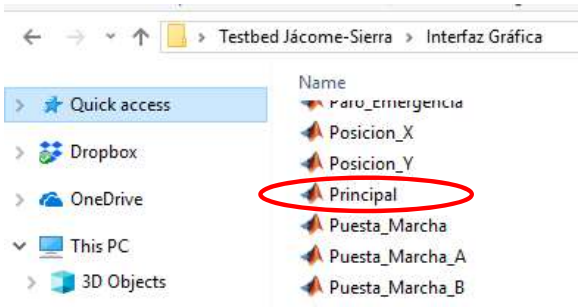


Figura IV.13. Puesta en Marcha de ventana Principal de interfaz gráfica

El programa despliega la portada del proyecto como se indica en la Figura IV.14,

Figura IV.14. Pantalla principal de Interfaz gráfica

Para más información acerca del proyecto, la interfaz gráfica provee al usuario una ventana de información básica del proyecto. Temas como robótica móvil, seguimiento de trayectoria, control, y visión artificial se despliegan en las ventanas de información como se indica en la Figura IV.15.

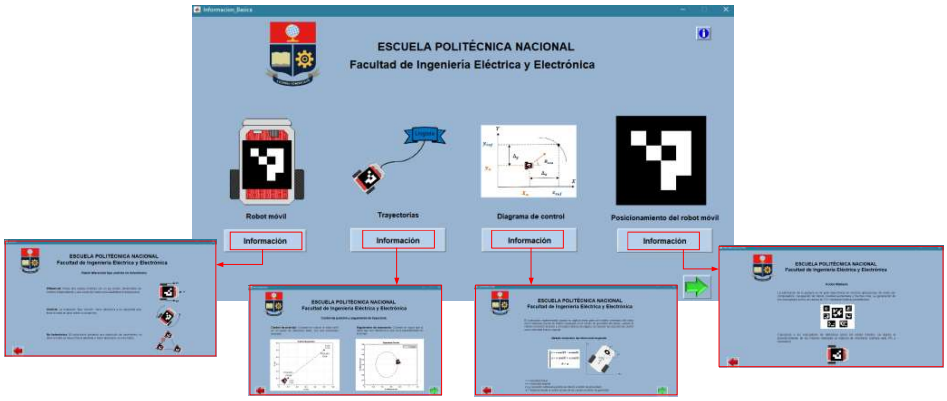


Figura IV.15. Pantalla de información básica

Continuando con el uso de la interfaz, la pantalla de directrices generales permite al usuario elegir el tipo de trayectorias que realizará el robot y el ingreso de parámetros de control, como se muestra en la Figura IV.16.

En la pantalla de parámetros de control, el usuario puede ingresarlos manualmente, y guardarlos pulsando el icono de disquete, o puede cargar los valores predeterminados pulsando el icono de tuerca.

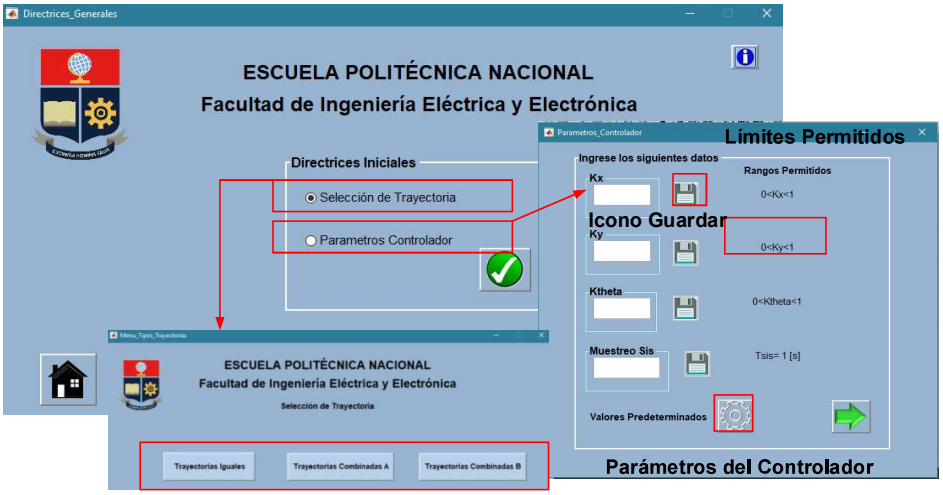


Figura IV.16. Pantalla de Directrices Generales

La opción de selección de trayectoria permite al usuario especificar la trayectoria individual de cada robot, para esto se despliegan tres casos:

- *Trayectorias iguales.* - Los dos robots realizan la misma trayectoria

- *Trayectorias combinadas A.*- Robot 1 realiza una trayectoria independiente a Robot 2.
- *Trayectorias combinadas B.*- Robot 1 realiza una trayectoria independiente a Robot 2.

Al pulsar en cualquier opción de las mencionadas, se despliegan las ventanas que se observa en la Figura IV.17, donde hay tipos de trayectorias lineal, lineal con pendiente de 45°, circular y de doble frecuencia.

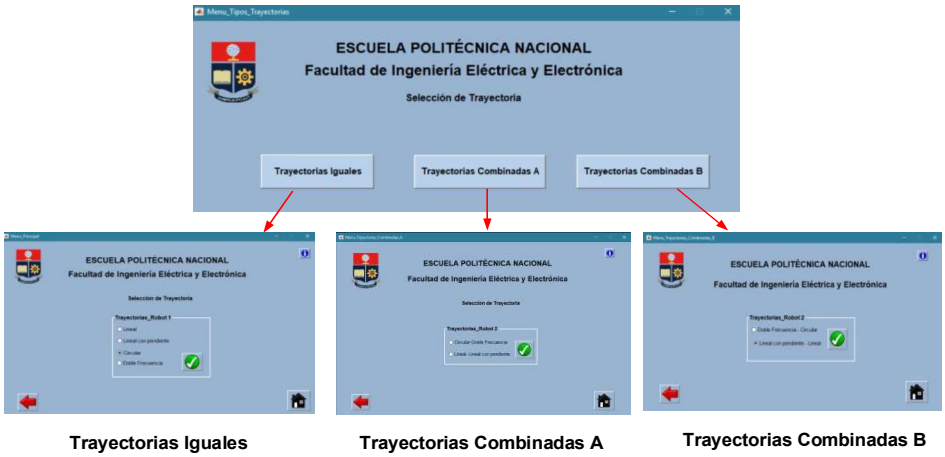


Figura IV.17. Opciones de tipos de trayectorias

En el caso de elegir un tipo de trayectoria, como en la Figura IV.18, se despliega el menú del tipo de ingreso de datos, para el ejemplo se elige el ingreso de *Distancia-Velocidad,* donde el programa de control calculará el tiempo de duración.

Figura IV.18. Pantalla de selección de tipo de trayectoria y tipo de ingreso de datos

Al elegir el tipo de ingreso de datos, se accede a la ventana de ingreso de datos como se indica en la Figura IV.19, donde el usuario podrá ingresar los datos manualmente y guardándolos, pulsando el icono de disquete, o cargar los datos predeterminados pulsando el icono de tuerca.

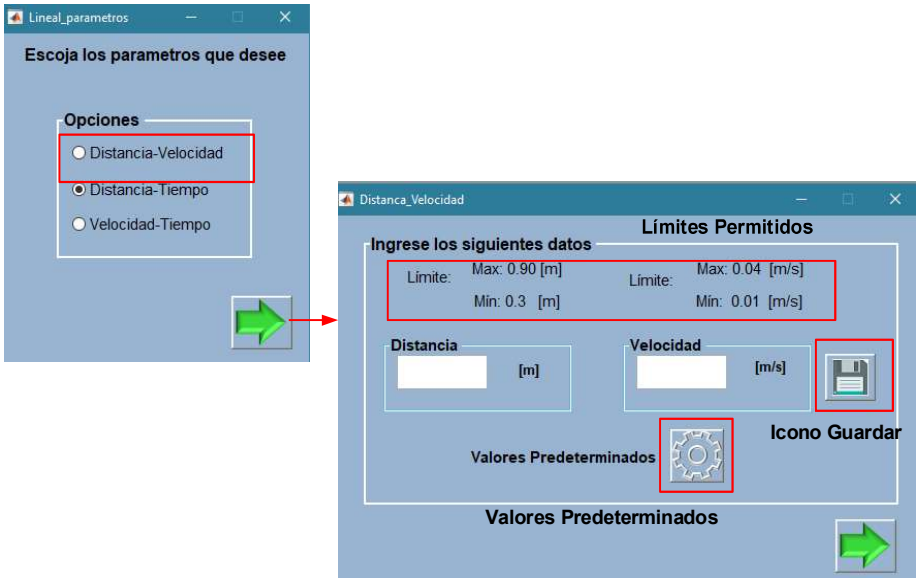


Figura IV.19. Pantalla de ingreso de datos

Para continuar es necesario conectar el módulo XBee a la computadora, como se indica en la Figura IV.20.

Figura IV.20. Conexión de módulo XBee a ordenador

Conectado el módulo de comunicación se procede a verificar la comunicación en la ventana de puesta en marcha, el sistema brindará un mensaje de comunicación exitosa o fallida según sea el caso. Además, en esta ventana el usuario puede verificar el estado de la batería, y cargarla en el caso de ser necesario, como se observa en la Figura IV.21.

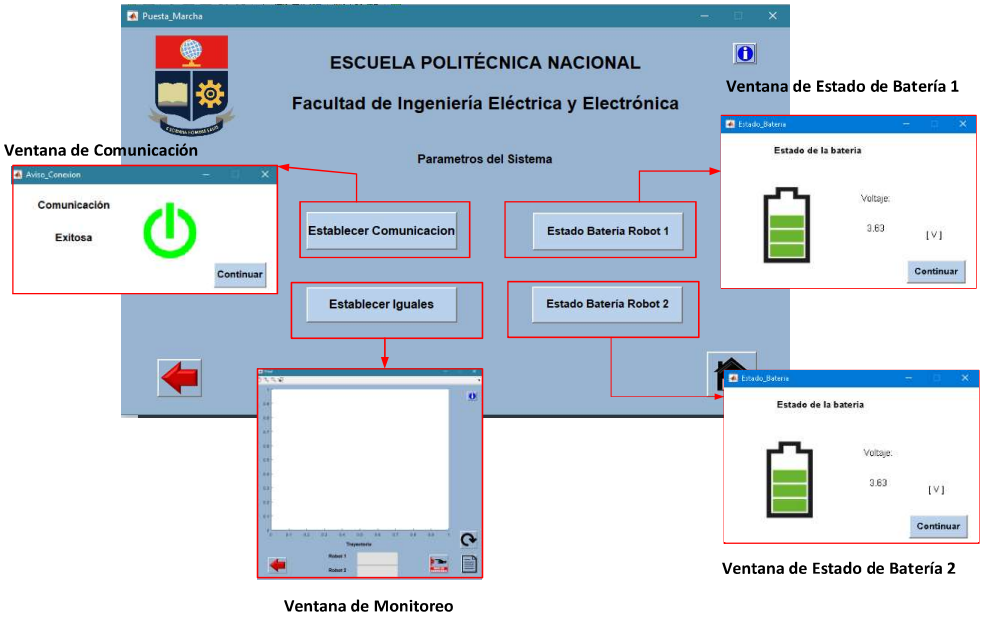


Figura IV.21. Pantalla de puesta en marcha.

Al presionar el botón puesta en marcha, se despliega la pantalla de visualización. El acceso a cada botón será posible siempre y cuando el sistema no esté en funcionamiento. En esta ventana se puede acceder al paro de emergencia, a la repetición del proceso, a directrices generales y a los resultados como se observa en la Figura IV.22.

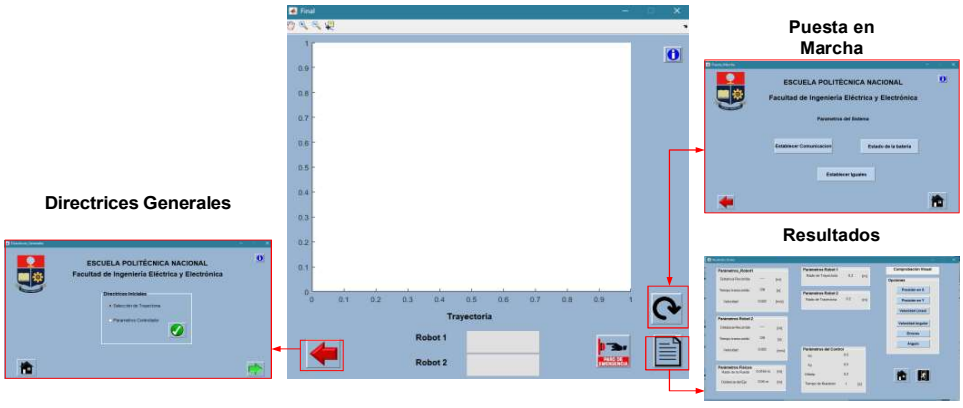


Figura IV.22. Pantalla de visualización.

El usuario tiene la oportunidad de analizar los resultados de proceso en la ventana de resultados como se indica en la Figura IV.23, en esta ventana se encuentra detallado los parámetros ingresados además de los calculados.

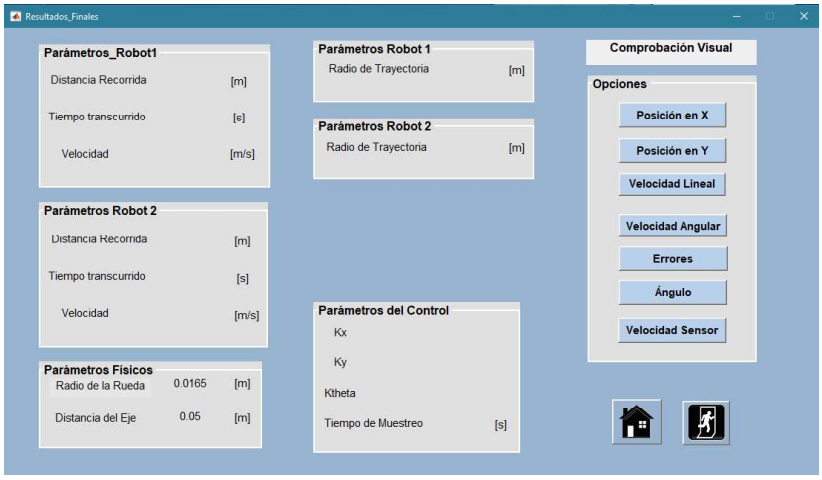


Figura IV.23. Pantalla de Resultados Finales

La ventana de resultados posee un menú de comprobación visual el cual permite acceder al análisis de los gráficos del proceso. En la Figura IV.24 se observa el tipo de gráficos que puede generar el usuario.

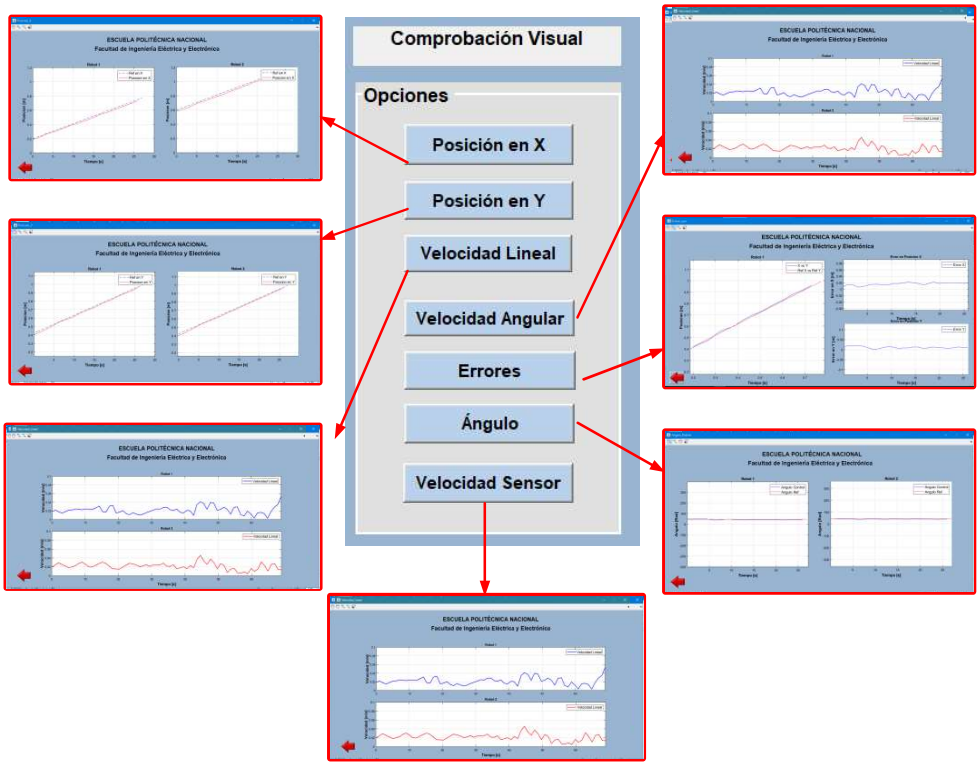


Figura IV.24. Menú de análisis de gráficos