ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

"DISEÑO DE UN SISTEMA DE CONTROL VISUAL PARA NAVEGACIÓN DE INTERIORES USANDO FUSIÓN SENSORIAL PARA UNA PLATAFORMA MÓVIL DIFERENCIAL"

JOSÉ LUIS TINAJERO LEÓN

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

MAGISTER EN SISTEMAS DE CONTROL Y AUTOMATIZACIÓN INDUSTRIAL

Riobamba - Ecuador

Enero - 2018

ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

CERTIFICACIÓN:

EL TRIBUNAL DE TRABAJO DE TITULACIÓN CARTIFICA QUE:

El Trabajo de titulación modalidad Proyectos de Investigación y Desarrollo, titulado: "DISEÑO DE UN SISTEMA DE CONTROL VISUAL PARA NAVEGACIÓN DE INTERIORES USANDO FUSIÓN SENSORIAL PARA UNA PLATAFORMA MÓVIL DIFERENCIAL", de responsabilidad del señor José Luis Tinajero León, ha sido prolijamente revisado y se autoriza su presentación.

Tribunal:

DR. JUAN VARGAS GUAMBO; M.Sc
PRESIDENTE _____

ING. PABLO LOZADA YÁNEZ; MgS.
DIRECTOR _____

ING. FAUSTO CABRERA AGUAYO; M.Sc.
MIEMBRO _____

ING. JAVIER GAVILANES CARRIÓN; M.Sc.
MIEMBRO _____

Riobamba, Enero de 2018

DERECHOS INTELECTUALES

Yo, **JOSÉ LUIS TINAJERO LEÓN** soy responsable de las ideas, doctrinas y resultados expuestos en el **Trabajo de Titulación modalidad Proyectos de Investigación y Desarrollo**, y que el patrimonio intelectual del mismo pertenece exclusivamente a la Escuela Superior Politécnica de Chimborazo.

_____

JOSÉ LUIS TINAJERO LEÓN

No. Cédula 060402667-4

DECLARACIÓN DE AUTENTICIDAD

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

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

Riobamba, Enero de 2018

_____

JOSÉ LUIS TINAJERO LEÓN

No. Cédula 060402667-4

DEDICATORIA

A ti mi pequeño tesoro José Agustín, porque me enseñas las cosas más hermosas de la vida con tu sincera sonrisa. A mi esposa Gisselle por ser mi mayor bendición. A mis padres por su lucha, entrega y sacrificio, José Antonio y Carmita. Y a todos los que me motivan a ser cada día mejor y diferente porque me demuestran que a pesar de todo, la vida es bella.

José Luis.

AGRADECIMIENTO

A Pablo Lozada quien desde el momento en que le comenté esta idea me apoyado y ayudado en el desarrollo, tomando la gran responsabilidad de ser mi tutor, a Narcisa Salazar y Alonso Alvarez por la ayuda brindada, además de ser quienes me han demostrado un verdadero ejemplo de personas y profesionales como ya muy pocos quedan.

José Luis.

ÍNDICE

CAPÍTULO IV

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN

El presente trabajo tuvo como finalidad diseñar un Sistema de control visual para navegación de interiores usando fusión sensorial sobre una Plataforma Móvil Diferencial. La investigación se desarrolló acerca de la tecnología Raspberry Pi y su utilización con el software Matlab, el ordenador controla la posición del servomotor ubicando la cámara de 0 a 180 grados, con avances de 45° para realizar una captura del entorno al instante, se realizó las operaciones de erosión y dilatación de las imágenes capturadas y de la base de datos, la finalidad de estas operaciones es obtener de forma clara los contornos de las capturas, para verificar si existe o no un obstáculo, en el sistema de control visual se emplea la correlación de imágenes. Para la verificación del sistema se empleó una tarjeta Arduino Nano con sensores ultrasónicos que permiten detectar si existe un objeto dentro del área de trabajo definida, si los dos sistemas concuerdan en la información sensada el robot toma una decisión en su desplazamiento, accionándose los motores desde el scrip. Para verificar la hipótesis se diseñó el experimento de manera que determine el nivel de eficiencia del sistema de control en diversas pistas que varían en la cantidad de obstáculos y forma de desplazamiento, se consideró el estadístico Anova obteniendo un resultado de 15,62 en el valor calculado de Fisher que comparado con el punto crítico del estadístico es de 9,55 se rechaza la hipótesis Nula, aceptando la hipótesis original de la investigación. Para finalizar se recomienda realizar una investigación para solucionar los errores encontrados en el sistema de control visual debido a los cambios que pueden existir en la luz ambiente, es decir, un sistema de iluminación que se adapte a las condiciones del sistema.

Palabras clave: <TECNOLOGÍA Y CIENCIAS DE LA INGENIERÍA>, <CONTROL AUTOMÁTICO>, <CONTROL VISUAL>, <FUSION SENSORIAL>, <RASPBERRY PI 3>, <ARDUINO NANO>, <PLATAFORMA MÓVIL DIFERENCIAL>, <CÁMARA MONOSCÓPICA>, <SERVOMOTOR>, <MATLAB (SOFTWARE)>.

SUMMARY

The aim of this work was to design a visual control system for interior navigation using sensorial fusion on a Differential Mobile Platform. The research was developed about the Raspberry Pi technology and its use with the Matlab software. The computer controls the position of the servomotor by locating the camera from 0 to 180 degrees with 45° advances to capture the environment instantly. The operations of erosion and dilatation of captured images and of the database were developed, the purpose of these operations is to clearly obtain the contours of the captures in order to verify whether or not there is an obstacle, in the visual system the correlation of images were used. To the verification of the system, an Arduino Nano card with ultrasonic sensors was used to detect if there is an object within the defined work area. If the two systems agree on the sensed information, the robot makes a decision on its displacement, activating the motors from the scrip. To verify the hypothesis, the experiment was designed in such a way that it determines the level of efficiency of the control system in various tracks that vary in the number of obstacles and the form of displacement. The ANOVA statistic was considered obtaining a result of 15.62 in the value calculated from Fisher that compared with the critical point of the statistic is 9.55 the Null hypothesis is rejected, accepting the original hypothesis of the investigation. Finally, it is recommended to carry out an investigation to solve the errors found in the visual control system due to the changes that may exist in the ambient light, that is, a lighting system that adapts to the conditions of the system.

Key words: <TECHNOLOGY AND ENGINEERING SCIENCES>, <AUTOMATIC CONTROL>, <VISUAL CONTROL> <SENSORY FUSION>, <RASPBERRY PI 3>, <ARDUINO NANO>, <DIFFERENTIAL MOBILE PLATFORM>, <MONOSCOPIC CAMERA>, <SERVOMOTOR>, <MATLAB (SOFTWARE)>.

CAPÍTULO I

1. INTRODUCCIÓN

El presente capítulo detalla de forma clara el problema, planteamiento y formulación de la investigación, además se incluye la justificación, los objetivos generales, específicos y la hipótesis de la investigación.

1.1 Planteamiento del problema

1.1.1 *Situación problemática*

En el contexto universal los sistemas de control visual son empleados en la mayoría de sistemas eléctricos y electrónicos siendo las principales aplicaciones los sistemas de seguridad en donde se destacan las características de reconocimiento facial o en pequeñas aplicaciones como detección y reconocimiento de objetos.

El desarrollo experimentado por América es bastante visible ya que existen dispositivos con sistemas de control visual empleados para la detección y rescate de víctimas después de un desastre ambiental.

En nuestro país son escasos los proyectos desarrollados sobre sistemas de control visual montados sobre plataformas móviles, pero se han desarrollado investigaciones sobre clasificación y detección de objetos sobre plataformas fijas.

El sistema de control visual se emplea generalmente para realizar la detección, ubicación y reconocimiento de objetos o características especiales de un determinado ambiente. Par ello emplea comparaciones por Correlación, Aprendizaje basado en redes neuronales o clasificadores estadísticamente óptimos.

El problema de un sistema de control visual se presenta en las condiciones de iluminación las mismas que afectan a la calidad de imagen y por ende el procesamiento de la misma lo que afecta de forma considerable la detección de los objetos o personas para lo cual fue diseñado el sistema.

El principio de funcionamiento está basado en la fusión sensorial, es decir, la corroboración de la información mediante dos sistemas independientes para obtener el mismo funcionamiento.

Los sistemas de control visual actuales deben cumplir altos niveles de exigencia puesto que son capaces de realizar detecciones de características con alto grado de complejidad.

1.2 Formulación del problema

¿Un robot móvil de configuración diferencial conformada por un sistema de control visual montado sobre una plataforma Raspberry Pi será capaz de desplazarse por interiores desconocidos?

1.3 Preguntas específicas

¿Un robot móvil diferencial será capaz de realizar desplazamientos por interiores evitando obstáculos?

¿Un sistema de control visual permitirá a un robot móvil de configuración diferencial navegar por interiores identificando objetos y obstáculos?

¿Es posible montar un sistema de control visual, almacenamiento de mapas y procesamiento para el guiado del robot móvil mediante el uso de una Raspberry Pi 3?

¿Una configuración diferencial permitiría el desplazamiento óptimo de un robot móvil para interiores?

1.4 Justificación de la investigación

En la actualidad se pretende el desarrollo de robots completamente autónomos los cuales son empleados con la finalidad de facilitar y mejorar las condiciones de vida de las personas sea cual sea el lugar de aplicación de dichos robots, de acuerdo con la aplicación algunos de ellos se encuentran diseñados con cámaras y varios sensores los cuales aseguran la integridad y correcto funcionamiento de los autómatas.

Mediante esta plataforma móvil se pretende tener una idea clara y precisa de cualquier ambiente sea industrial o residencial, de modo que se puedan evitar cualquier tipo de accidentes que puedan alterar la integridad física de las personas que circulen por un espacio totalmente desconocido.

El autómata estará en la capacidad de encontrar la mejor ruta de desplazamiento evitando cualquier obstáculo y obteniendo una reconstrucción del entorno, mediante capturas de imágenes consecutivas.

Antes de desplazarse por cualquier área, es necesario identificar muchos de los objetos que se encuentran en un determinado espacio físico, para este fin se debe realizar mediante la observación del área, la ventaja del sistema en estudio es de fácil observación, ya que todos los posibles obstáculos se encuentran a nivel de las personas.

En el presente proyecto se busca investigar sobre nuevas alternativas que tengan las principales características que son necesarias para realizar la tarea de desplazamiento por interiores mediante un robot diferencial, este sistema pretende implementar con un hardware relativamente nuevo a un precio bajo y que brinde todas las características innovadoras en el país, reduciendo de manera considerable el riesgo y el tiempo de desplazarse por ambientes desconocidos, peligrosos o inaccesibles para los humanos. Se pretende que mediante este robot logre tres aspectos, primero optimizar la forma de desplazamiento en tiempo y riesgo, se reduzca el riesgo para las personas al ingresar a un ambiente desconocido o peligroso, y finalmente obtener una imagen clara y estructurada del ambiente interno.

1.5 Objetivos de la investigación

1.5.1 *Objetivo general*

Diseñar un sistema de control visual para navegación de interiores usando fusión sensorial para una plataforma móvil diferencial.

1.5.2 *Objetivos específicos*

- Diseñar e implementar una estructura móvil diferencial con una cámara Monoscópica a bordo y una Raspberry Pi 3.
- Diseñar un sistema de visión artificial que permita realizar el mapeo completo del ambiente para tomar las decisiones correctas durante el desplazamiento.
- Implementar el algoritmo de planificación y trayectoria que mejor se adapte a la estructura y forma de desplazamiento.
- Verificar el correcto funcionamiento y detección de posibles errores del sistema completo.
- Reconstruir el ambiente interior en un modelo en dos dimensiones.

1.6 Hipótesis

El sistema de control visual utilizando fusión sensorial en una plataforma móvil diferencial permite el correcto desplazamiento y mapeo para la reconstrucción de interiores.

CAPÍTULO II

2. MARCO REFERENCIAL

2.1 Antecedentes del problema

Se puede realizar un desplazamiento de forma adecuada y autónoma por espacios totalmente desconocidos y que poseen muy pocas condiciones favorables, logrando que el robot se pueda guiar de forma adecuada y eficiente.

En la actualidad a nivel internacional se puede destacar varios trabajos desarrollados sobre Raspberry Pi entre los cuales se puede destacar Cámara estéreo inteligente sobre una plataforma móvil para robot de servicio de búsqueda en interiores (Jiang, 2016), Monitoreo y control de un robot móvil vía internet a través de un tarjeta Raspberry Pi presentado en ICONSTEM (Vanitha, Selvalakshmi, & Selvarasu, 2016), control de un robot de video vigilancia con Raspberry pi y un teléfono inteligente, el cual tiene como finalidad mediante una conexión Wi-Fi se le envían las ordenes de movimiento (Pi, Bokade, Ratnaparkhe, & Pi, 2016).

A nivel de Latinoamérica son escasos los trabajos realizados pero se puede destacar Ley de control de un servo visual usando un enfoque de visión 2D, para un sistema robótico de 3 DOF construido con LEGO EV3 y Raspberry Pi el cuál se enmarca en el área de robótica móvil y el uso de nuevas tecnologías (Milena, López, Enrique, & Uribe, 2016).

A nivel de país es nula la información que se puede recabar acerca de esta temática.

La situación creada por la dificultad en el desplazamiento de los individuos por los interiores de cualquier espacio físico desconocido con condiciones poco favorables para el reconocimiento del lugar, determina la necesidad de definir un nuevo método de desplazamiento empleando nuevos recursos tecnológicos, los cuales permitan facilitar y mejorar la forma de desplazamiento optimizando dicha tarea y de esa forma evitando al límite máximo cualquier accidente que pueda sufrir cualquier individuo ya sea en un ambiente residencial o industrial.

2.2 Robótica

Se debe indicar que la robótica posee una historia bastante antigua basada en el desarrollo cultural de la humanidad. Desde siempre el humano ha buscado dispositivos electromecánicos que sean capaces de sustituir los diversos comportamientos propios y característicos de los hombres en las diversas interacciones con el medio en el que habita, todo este avance tecnológico se lo ha generado por diversos aspectos en los cuales se puede destacar el filosófico, social, económico y científico (Siciliano, Sciavicco, Villani, & Oriolo, 2009).

El término robótica crea en nuestra mente dispositivos electromecánicos diseñados para la realización de rutinas repetitivas para la generación de trabajos productivos en áreas que pueden resultar de difícil acceso o peligrosas para los humanos, de forma adicional cabe indicar que son máquinas que imitan los movimientos y determinados casos el comportamiento de los seres humanos (Ollero, 2001).

La teoría de máquinas capaces de imitar movimientos y acciones se remonta a la misma existencia de la humanidad, cabe destacar dos eventos importantes en la aparición de esta tecnología , la primera se produce en el año de 1920 en la Obra Rossum´s Universal Robots escrita por el dramaturgo checo Karel Capek; el siguiente acontecimiento se da en los inicios de 1940 mediante el Ruso Isaac Asimov el cual genera la idea de un robot como un artefacto mecánico, el mismo que se desarrolló con la aparición de la ciencia ficción (Siciliano *et al*., 2009).

En la actualidad todo el desarrollo industrial y tecnológico se ha visto influenciado por dichos mecanismos, ya que sus inicios estas máquinas servían tan solo para magnificar la potencia muscular humana, es decir, sustituía a los hombres en realizar procesos que conllevaban gran esfuerzo físico. En la actualidad son máquinas capaces de adquirir datos, analizarlos, interpretarlos y tomar las decisiones en base a condiciones y parámetros previamente establecidos por los programadores, de modo, que dichos dispositivos han sido capaces de sustituir a los humanos de forma completa en determinadas actividades (Ollero, 2001).

Todo robot debe obedecer a las tres principales leyes:

1. Un robot no puede dañar a un ser humano, o por inacción, llegar a dañar a un humano.
2. Todo robot debe obedecer las órdenes dadas por los seres humanos, con excepción, cunado dichas ordenes entren en conflicto con la primera ley.
3. Un robot debe proteger su propia existencia, siempre y cuando, esta ley no entre en conflicto con las anteriores.

Estas leyes establecen reglas de comportamiento en las que se deben considerar el diseño de robot, desde este punto de vista se puede indicar que el robot es un producto industrial diseñado por ingenieros o técnicos especializados (Siciliano *et al.*, 2009).

2.2.1 *Estructura mecánica de un robot*

La característica clave de un robot es su estructura mecánica, es decir, el diseño sobre el cual se implementará dicha máquina. Existen dos tipos de estructuras mecánicas las de base fija y las de base móvil (Ollero, 2001).

En las estructuras de base fija se pueden encontrar los robots manipuladores, es decir, aquellos que están sujetos a un punto fijo y se desplazan de un lugar a otro mediante eslabones y articulaciones, mientras que en las estructuras de base móvil se consideran los robots móviles, sea mediante patas, ruedas o mixtos (Holland, 2004).

2.2.2 *Esquema general del sistema robot*

La principal característica de funcionamiento de un robot se seccionada en diferentes áreas, las cuales trabajan de forma conjunta y en las cuales se destaca:

- Sistema Mecánico
- Sensores
- Sistema de Control
- Actuadores

Cada uno de ellos se encarga de una condición específica de funcionamiento dentro de la máquina.

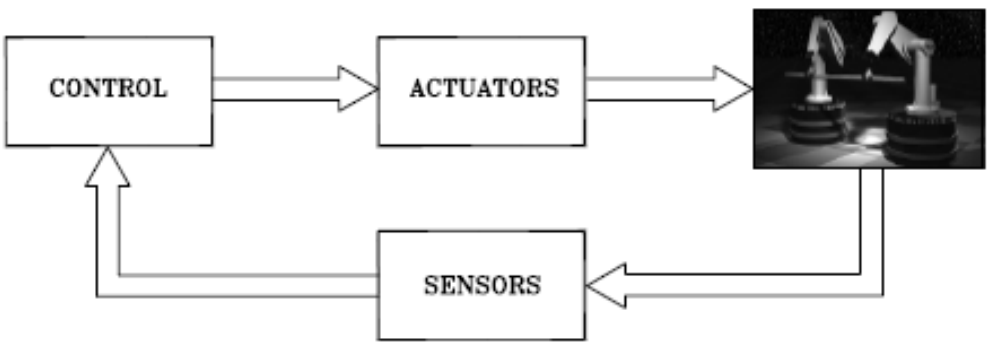


Figura 1-2: Componentes de un sistema robótico.
Fuente: Siciliano *et al.*, 2009.

El sistema mecánico está considerado todos los elementos que corresponden a la estructura de la máquina, es decir, un brazo articulado o un vehículo móvil, en el sector industrial la mayoría de los robots se encuentran sujetos a una base fija.

Los sensores se encargan de dotar de información sobre la parte interna y externa del robot. Se diferencia dos tipos de sensores el primero de ellos son los internos que se encargan de indicar el estado de la estructura mecánica, giros, desplazamientos, fuerzas y pares, mientras que los sensores externos se encargan de proporcionar toda la información que rodea a la máquina.

El sistema de control se encarga de realizar el procesamiento de la información obtenida de todos los sensores ubicados sobre la máquina.

Los actuadores responden a bucles o líneas de código que les permiten realizar acciones determinadas de acuerdo con las acciones tomadas por el sistema de control. Los actuadores en un robot pueden ser pinzas, motores, etc (Ollero, 2001).

2.2.3 *Robots manipuladores*

La estructura mecánica de un robot manipulador consiste en una secuencia de cuerpos rígidos que se encuentran interconectados mediante articulaciones que se denominan juntas. Un manipular se caracteriza por un brazo que permite la movilidad de la máquina, una muñeca que le surte destreza, y en su parte final un actuador que ejecuta la función para la cual está diseñada la máquina (Siciliano *et al.*, 2009).

Según Robot Institute Of América: "Un robot industrial es un manipulador programable multifuncional diseñado para mover materiales, piezas, herramientas o dispositivos especiales, mediante movimientos variados, programados para la ejecución de distintas tareas".

2.2.4 *Robots móviles*

Debido a la necesidad de extender el campo de la robótica, se desarrolló los robots móviles los mismos que conciben la idea de aumentar la autonomía de las máquinas reduciendo así la intervención de los humanos. La robótica móvil se enfoca en que el dispositivo tenga la suficiente inteligencia como para reaccionar y tomar decisiones acertadas basándose en los datos que puede tomar de su entorno, considerando que dicho medio no puede ser conocido completamente (Siciliano *et al.*, 2009).

Una de las características más relevantes de los robots móviles es que estos poseen una base móvil que los permite desplazarse de forma libre por un ambiente conocido o desconocido, su principal aplicación es en la de servicio, es decir, aplicaciones donde necesita movimiento de un lugar a otro. La base móvil denominada chasis es un cuerpo rígido que posee un sistema de locomoción el cual se encarga de realizar el movimiento de la máquina. Existen dos clases de robots móviles en las cuales se puede destacar:

- Los robots móviles con ruedas que están conformado por un chasis rígido y un sistema de ruedas que son las encargadas de proporcionar movimiento respecto al suelo.
- Robots móviles con patas se caracterizan por poseer uno o varios cuerpos rígidos que están conectados por juntas por articulaciones rotacionales. Las extremidades de la máquina son aquellas que entran en contacto de forma periódica con respecto al suelo para generar movimiento.
- Estos robots generalmente están basados en la naturaleza, pues se trata de imitar a organismos vivos como los mismos humanos o insectos (Ollero, 2001).

2.2.4.1 Vehículos con ruedas

Son una solución bastante eficiente y sencilla cuando el desplazamiento se lo debe realizar en terrenos duros que no presenta obstáculos, de acuerdo con la dureza del suelo se pueden presentar problemas en el desplazamiento debido a vibraciones o deslizamientos de la parte de locomoción, y esta se ve gravemente afectada cuando el terreno donde se desplaza es blando. Según Aníbal Ollero: “Los robots móviles emplean diferentes tipos de locomoción mediante ruedas que les confiere características y propiedades diferentes respecto a la eficiencia energética, dimensiones, cargas útiles y maniobrabilidad.”

Las configuraciones más importantes sobre robots con ruedas se pueden destacar:

a) Ackerman

La estructura mecánica de estas máquinas se encuentra formado por un chasis sobre el cual se encuentran montadas cuatro ruedas convencionales, generalmente se emplean vehículos tradicionales tales como automóviles o vehículos de mayor peso para aplicaciones para navegaciones autónomas en exteriores.

La mayor desventaja de la locomoción Ackerman es la limitada maniobrabilidad que existe (Ollero, 2001).

b) Triciclo Clásico

Este sistema de locomoción se encuentra formado por tres ruedas ubicadas una en la parte delantera del móvil y los dos restantes en la parte posterior posicionadas en la parte lateral del móvil. La rueda delantera posee una doble funcionalidad pues esta genera tracción y direccionamiento del robot, mientras que las ruedas posteriores se mueven de manera libre por lo que se considera un sistema pasivo. Respecto de la configuración Ackerman este tipo de locomoción presenta mayor movilidad, pero su estabilidad se ve afectada cuando el terreno posee variaciones en su morfología (Ollero, 2001).

c) Direccionamiento Diferencial

Su característica principal es que posee dos ruedas ubicadas en la los costados del chasis, dichas ruedas se encargan de la tracción y direccionamiento del móvil. Para lograr el direccionamiento se debe variar las velocidades de las ruedas laterales. De forma adicional existe una o varias ruedas las cuales se emplean sobre la estructura para dar soporte, pero no tienen función adicional. La aplicación más grande de esta configuración es para realizar desplazamientos por interiores (Ollero, 2001).

d) Skid Steer

Este tipo de configuración presentan varias ruedas a los costados de la plataforma las mimas que deberán actuar de manera simultánea para lograr movimiento, el movimiento es ideal y correcto cuando se logra combinar las velocidades de todas las ruedas ubicadas sobre el robot. Poseen aplicaciones para navegación en exteriores donde la dureza del libro juega un papel importante, de esta manera, tienen gran utilidad en la minería, o en inspección y obtención de mapas en lugares de difícil acceso (Ollero, 2001).

e) Pistas de deslizamiento

Son vehículos tipo oruga en las cuales el desplazamiento y direccionamiento se lo realiza por medio de pistas de deslizamiento, esta configuración es de especial aplicación donde los terrenos son completamente irregulares ya que en este tipo de suelos presentan un excelente rendimiento. Presente una gran similitud a la configuración Skid Steer (Siciliano *et al.*, 2009).

f) Síncronas

Es una configuración que presenta un grado de complejidad más alto, ya que se debe conseguir que todas las ruedas actúen de forma simultánea, es decir, que todas giren de forma síncrona. Para alcanzar este objetivo es necesario emplear engranajes o correas concéntricas (Ollero, 2001).

2.2.4.2 Locomoción mediante patas

Son configuraciones que permiten aislar el cuerpo del suelo o terreno, mediante la utilización de puntos discretos de soporte. Gracias a este tipo de configuraciones es posible desplazarse por ambientes que poseen obstáculos ya que posee mejores y mayores propiedades que los robots con ruedas. El deslizamiento es mucho menor en los mecanismos que poseen locomoción por patas, pero se tiene gran omnidireccionalidad.

Una desventaja es que este tipo de mecanismos tienen mayor complejidad en su desarrollo y el consumo energético es alto debido a la cantidad de elementos que deben ser alimentados para el correcto funcionamiento. Existen configuraciones de dos, cuatro, seis y ocho patas que son capaces de mantener la estabilidad de forma dinámica, de todas estas configuraciones la que más comúnmente se emplea es la de los hexápodos.

Cabe destacar que una modificación de los robots con patas son los robots trepadores, que poseen características similares, pero estos se desplazan de forma vertical, para lograr este tipo de desplazamiento los robots poseen en sus extremidades sistemas de succión o sistemas magnéticos que les permiten adherirse a la superficie.

2.2.4.3 Configuraciones articuladas

Son configuraciones aplicables para terrenos con características variantes en los que la máquina debe adoptar la condición del terreno, su estructura está conformada por varios eslabones los cuales están formados por dos o más módulos que tienen locomoción por ruedas. Cuando una configuración posee varias articulaciones son específicos para caminos estrechos, debido a la redundancia de esta estructura posee una alta seguridad en su funcionamiento.

2.2.4.4 Robots submarinos y aéreos

Este tipo de máquinas se las crea con la facilidad de inspeccionar, recolectar datos, mantenimiento de instalaciones donde el acceso del humano resulta complicado y en algunas situaciones hasta imposibles. De acuerdo a la aplicación se destaca dos tipos de robots: el primero los submarinos y los aéreos, cabe destacar que estos modelos surgen como una evolución de vehículos teleoperados o pilotados por el mismo hombre.

2.2.5 *Sensores*

Los sensores se encargan de proporcionar la información necesaria para realizar el control del robot. Los sensores se los puede emplear para la medición de diversos principios físicos y químicos.

En la actualidad la mayoría de sensores emplean un procesamiento electrónico, y para poder ser empleadas es necesario que estas sean señales eléctricas, teniendo en cuenta que en muchas o en casi todas las aplicaciones de estos dispositivos son varios datos que se requieren conocer, tales como, la magnitud y sus respectivas derivadas (Ollero, 2001).

2.2.5.1 Clasificaciones y características de sensores

A los diversos sensores se los pueden clasificar de varias formas entre las cuales se puede destacar: el uso de fuentes de energía externa o según la variación relativa o absoluta que poseen los sensores a la salida. Las características que se debe considerar en los sensores según Berney son:

- Linealidad
- Histéresis
- Repetibilidad
- Resolución
- Sensibilidad
- Ruido

En la mayoría de procesos automatizados actualmente se han empleado sensores de alta tecnología denominados como "Smart Sensors", estos dispositivos, incorporan características adicionales a las empleadas tradicionalmente:

- Comunicación Bidireccional.
- Autocalibración
- Filtros
- Fusión Multisensorial

2.2.5.2 Medidas de desplazamientos lineales y giros

a) Potenciómetros para medida de desplazamientos

Se los emplea bajo la configuración de divisores de potencial, se mide el desplazamiento mediante la variación de la resistencia del potenciómetro, se distinguen dos tipos de potenciómetros: lineales y rotacionales (Ollero, 2001).

b) Codificadores ópticos

Su principio de funcionamiento se caracteriza por convertir un desplazamiento rotacional en una señal digital sin la necesidad de emplear un Convertidor analógico-digital considerando las interrupciones de un haz de luz. Están formados por dos diodos generalmente, el primero de ellos puede ser una fuente incandescente o un diodo simple que se lo emplea como emisor, mientras que para el receptor se debe emplear un fotodiodo (Ollero, 2001).

Para completar el esquema de estos sensores es conveniente el empleo de ruedas perforadas que indican el cambio de nivel al pasar de una zona opaca a una transparente.

c) Medida de velocidad de ejes

El principio de funcionamiento de estos elementos consiste en obtener una salida eléctrica relacionada con la velocidad de entrada mecánica, consiste en un generador eléctrico de corriente continua que se encuentra acoplada de forma directa o acoplada al eje en el que se pretende obtener la velocidad. La tensión en los bordes de salida es directamente proporcional a la velocidad y la polaridad indica el sentido de rotación de los ejes (Ollero, 2001).

2.2.5.3 Sensores de presencia y proximidad

Entre los más importantes podemos destacar:

a) Finales de carrera:

Se basan en un funcionamiento mecánico, y son considerados interruptores mecánicos, cuando existe un contacto físico entre el objeto y el sensor, se cierra un contacto eléctrico que genera una señal eléctrica de carácter binario (Siciliano *et al.*, 2009).

b) Sensores inductivos

Son aquellos sensores que detectan presencia sin la necesidad de que exista un contacto mecánico, se emplean estos sensores cuando se desea detectar objetos metálicos que modifican el campo magnético pues induce corriente en la bobina del sensor que se detecta al medir la tensión en dicha bobina (Holland, 2004).

c) Sensores capacitivos

Permiten detectar cambio producidos en la capacidad del sensor inducidos por una superficie cercana, se emplean para la detección de materiales no metálicos, es decir, no ferromagnéticos.

d) Sensores ópticos

Según Ollero: "El principio de funcionamiento de interrupción del haz de luz, se emplean diodos emisores de luz y fotodetectores tales como foto resistores, fotodiodos o fototransistores". Los fotoresistores o también denominados fotocélulas son resistencias que varían su valor de acuerdo a la intensidad de luz percibida por el elemento, los fotodiodos poseen mayor sensibilidad y producen una señal lineal, pero su señal de salida de ser amplificada.

e) Sensores de ultrasonidos

Estos sensores emiten pulsos de sonido y calculan el tiempo de vuelo en el que tarda en regresar la señal hasta el sensor después de golpear con cualquier objeto. Para que estos elementos entreguen una distancia real se deben considerar aspectos de vital importancia como la velocidad del sonido, ubicación del sensor y rango de medida.

2.3 Visión artificial y procesamiento digital de imágenes

La visión artificial o también denominada Visión por computador es una de las áreas de la tecnología que actualmente ha ganado un gran campo de aplicación y desarrollo, para el desarrollo de estudios y prototipos que emplean visión artificial se basan en otras ramas y ciencias tales como matemáticas, física, ingeniería electrónica y eléctrica, computación y de todas las disciplinas que contribuyan al desarrollo de esta compleja área de estudio. La visión artificial pretende emular capacidades propias y únicas de algunos seres vivos para ver, interpretar y analizar la escena (García Santillán, 2008).

El Procesamiento Digital de Imágenes (PDI) para facilidad de estudio y análisis se los divide en seis grandes áreas, las mismas que consideran la complejidad y delicadeza que conlleva el PDI.

La captura y adquisición es la etapa inicial del PDI, se debe realizar el preprocesamiento de la imagen, se debe realizar la segmentación, descripción, reconocimiento e interpretación de las imágenes, se debe destacar que de acuerdo a la aplicación y funcionalidad del PDI se pueden o no aplicar todas las etapas anteriormente descritas (García Santillán, 2008).

2.3.1 *Imagen digital*

Una imagen es una función bidimensional que se puede expresar como: $f(x, y)$, de donde x, y corresponden a las respectivas coordenadas de un punto en el plano, la intensidad de dicho punto sea a color o en escala de grises se define como la amplitud f, si estos elementos son finitos, es decir, un número discreto de puntos denominados píxeles se dice que una imagen es digital, cada pixel posee un valor y posición particular. Cuando se capturan imágenes bidimensionales desaparece una importante cantidad de información de esta, pues el mundo que nos rodea posee escenas tridimensionales (García Santillán, 2008).

2.3.1.1 Clasificación de las imágenes digitales

Las imágenes se las puede clasificar en dos grandes grupos:

- Imágenes Vectoriales. - se consideran aquellas imágenes que conservan la nitidez de los bordes y que son capaces de mantener todos los detalles cuando se modifica o altera el tamaño de la imagen, pues estas imágenes son independientes de la resolución.
- Mapa de Bits. - también denominadas imágenes raster, y poseen un número determinado de píxeles, en base a esta característica estas dependen de la resolución, al depender de la resolución si se modifica el tamaño se puede perder detalles y perder información de la imagen (García Santillán, 2008).

2.3.1.2 Formatos de imágenes ráster

Los formatos más comunes con los que se puede operar son:

- TIFF

Son de poca utilidad en los servidores Web, pues al descomprimir dichas capturas pueden ser excesivamente grandes, pero tiene mucha aplicación pues al momento de comprimir las imágenes

éstas no pierden calidad, además de que se las puede leer con facilidad en varios sistemas operativos (García Santillán, 2008).

- BMP

Generalmente se lo emplea con propósitos generales tales como la edición de las imágenes, es un formato muy emplea para máquinas que poseen sistema operativo Windows, y presente problemas con computadoras que trabajan bajo el sistema operativo Macintosh pues los archivos tienden a ser considerablemente grandes (García Santillán, 2008).

- GIF

Según sus siglas en ingles se interpreta con formato de intercambio de gráficos, su limitación se ve representada en el uso de solo 256 colores, lo que puede afectar la calidad de la imagen (García Santillán, 2008).

- JPEG

Por sus siglas en inglés Grupo Unido de expertos en fotografía, es un archivo de comprimible que posee capacidad de escalamiento que permite la creación de archivos reducidos, trabaja con toda la paleta de colores a diferencia del formato anteriormente estudiado (García Santillán, 2008).

- PNG

Gráfico de red portable posee características muy similares al formato JPEG pues también trabaja con la gama completa de colores, pero su principal diferencia radica en que cuando se comprime una imagen con este formato no se reduce la calidad de la imagen (García Santillán, 2008).

2.3.1.3 Tipos de imágenes digitales

Para realizar un correcto Procesamiento Digital de Imágenes se las realiza a cuatro tipos de imágenes las mismas que se las explican de forma sencilla a continuación:

- Imágenes RGB(Reg-Green-Blue)

Las imágenes RGB utilizan tres canales de color para reproducir la captura en la pantalla, cada uno de los canales utiliza 8 bits, en total cada píxel posee 24 bits, tienen la capacidad de reproducir hasta 16,7 millones de colores y soporta diversos formatos de imágenes (García Santillán, 2008).

- Imágenes indexadas

El número máximo de colores está limitado a 256, reduce el tamaño del archivo pues elimina información correspondiente al color, lo formatos más comunes en imágenes indexadas son GIF y PNG-8 (García Santillán, 2008).

- Imágenes en Escala de Grises

Se caracterizan en representar la imagen en tonos de gris, cada píxel varía entre 0 y 255, donde el valor de 0 corresponde al color negro y 255 es considerado color blanco, si una imagen posee 8 bits ésta puede tener hasta 256 tono de color gris (García Santillán, 2008).

- Imágenes Binarias

Posee una profundidad de color de 1 bit, representa los píxeles de una imagen mediante dos valores de color, es decir, en blanco o negro (García Santillán, 2008).

2.3.1.4 Calidad de una imagen

La calidad de una imagen raster se determina por tres factores los mismos que afecta a la imagen en el proceso de captura, y estas son:

El tamaño del píxel, también denominada la resolución espacial es aquel rango determinado por el dispositivo de captura de la imagen muestrea la captura.

La profundidad del píxel, conocido como resolución del brillo está definido por un bit o un grupo de bits, la resolución del brillo es directamente proporcional al número de bits, es decir, mientras más bits se consideren mejor será la resolución de la imagen.

El ruido estará presente en casi todas las imágenes y este se presenta como píxeles aislados que poseen un color gris diferente a la de sus vecinos, esta perturbación se produce por el dispositivo de adquisición o el medio de transmisión.

La resolución de una imagen es el número de pixeles que se desea que contenga una captura, el número de píxeles depende de la aplicación para la cual está destinada la imagen.

El tamaño de una imagen es el producto entre la cantidad de píxeles horizontales, verticales y la profundidad de brillo de la captura de imágenes.

2.3.1.5 Captura de imágenes

El proceso de captura de una imagen se la realiza por medio de algún dispositivo que permite adquirir una imagen desde el mundo físico, existen dos métodos para lograr este fin, el primero de ellos y quizás ya fuera de uso es el que emplea elementos químicos, o en más empleado actualmente que es por medio de dispositivos electrónicos.

Entre los dispositivos electrónicos se puede destacar:

- Cámara digital

Las características que se debe considerar para la elección de una cámara digital son:

Resolución. - Es el número de píxeles que contendrá la imagen, por defecto la mayoría de las cámaras comerciales es de 640x480 píxeles, pero se puede modificar la resolución de un abanico de posibilidades con las que pueden trabajar la cámara digital.

Memoria. - Depende de la cantidad de información, es decir, imágenes se desean almacenar. Existen cámaras digitales que poseen memorias de almacenamiento internas y externas.

Compresión de Archivos. - la calidad de la imagen es inversamente proporcional al grado de compresión de las imágenes, es decir, a mayor compresión menor será la calidad de la imagen, pero se podrán almacenar una cantidad considerable de capturas.

Sistema de Transferencia. - Se debe considerar el medio de transmisión físico con el que se pretende realizar el traspaso de información hacia una PC (García Santillán, 2008).

- Escáner

Es un dispositivo electrónico que permite realizar una copia digital de una fotografía o cualquier tipo de documento. En un escáner se deben considerar dos aspectos importantes.

Resolución. - depende de la velocidad y precisión que posee el motor para realizar el desplazamiento por todo el documento.

Intensidad de color. - la mayoría de los escáner son capaces de digitalizar hasta 16,7 millones de colores (García Santillán, 2008).

2.3.1.6 Digitalización de imágenes

La digitalización se entiendo como el proceso por el cual se logra un cambio de una imagen analógica a una digital, la imagen digitalizada es una imagen bidimensional, la misma que está constituida por píxeles, estos elementos ofrecen información importante y única de la imagen como por ejemplo el brillo, el color o la posición. En la digitalización de imágenes se pueden diferenciar dos procesos:

El muestreo.-mediante este proceso se obtiene la imagen en una matriz de m filas y n columnas, donde se determina el tamaño del píxel y el valor del brillo (García Santillán, 2008).

La cuantización.- consiste en asignar valores a cada uno de los elementos de la matriz, los mismos que representan el valor de la variable física en un determinado punto.se considera que los valores más pequeños del rango de valores corresponde al color gris negro mientras que el valor más alto al gris blanco (García Santillán, 2008).

2.3.2 *Procesamiento digital de imágenes*

Hace referencia al procesamiento de las imágenes obtenidas del mundo que nos rodea de forma digital mediante un computador, el principal objetivo del PDI es el de mejorar el aspecto de las imágenes y hacer notar determinados detalles que se deseen en las distintas capturas.

Se consideran seis áreas para el procesamiento digital de imágenes, las mismas que pueden ser agrupadas en tres grandes grupos desde el punto de vista de la visión artificial.

- Nivel de Visión Bajo

Está conformada por las áreas de captura/adquisición y el preprocesamiento.

En la etapa de captura o adquisición de la imagen es el proceso mediante el cual se consigue una imagen digital por medio de un dispositivo electrónico tales como una cámara digital, un escáner, etc. En el preprocesamiento se emplean técnicas de reducción del ruido, realzar determinados detalles de interés en la imagen como por ejemplo bordes contraste, brillo, etc.

- Nivel de Visión Medio

Conformada por las etapas de Segmentación, descripción y reconocimiento. En la segmentación es el proceso por el cual se particiona a una imagen en objetos los mismos que son los sistemas de interés. En el área de la descripción se logra conseguir diferenciar un tipo de objeto de otro por

características propias y únicas de cada elemento como por ejemplo la forma, tamaño, área, etc. El reconocimiento consiste en identificar los diferentes objetos analizados en las etapas anteriores.

- Nivel de visión Alto

Se realiza la interpretación, que relaciona un determinado significado con un conjunto de objetos reconocidos. Resumiendo lo anteriormente expuesto se puede obtener la siguiente tabla:

Tabla 1-2: Niveles de visión y procesamiento digital de imágenes.

NIVEL DE VISIÓN	AREAS DEL PDI
Bajo	Captura o Adquisición
	Preprocesamiento
Medio	Segmentación
	Descripción
	Reconocimiento
Alto	Interpretación

Fuente: Visión artificial y procesamiento digital de imágenes usando Matlab.
Realizado por: Tinajero José, 2017.

2.3.2.1 Aplicaciones del procesamiento digital de imágenes

La aplicación para el Procesamiento Digital de Imágenes es muy extensa pues estas siempre se enfocan en la resolución de determinados problemas en las aplicaciones médicas, industriales, astronómicas, etc.

2.3.2.2 Componentes de un sistema PDI

En un sistema de PDI se encuentran sensores, digitalizadores, Hardware especializado en el PDI, Computadora, software, dispositivos de almacenamiento, monitores y acceso a la red (García Santillán, 2008).

2.3.3 *Ruido en imágenes*

Todas las imágenes digitales poseen una cantidad reducida de ruido, debido a muchos factores, entre los que podemos destacar el dispositivo electrónico de captura o el medio de transmisión. El ruido es visible en las capturas mediante píxeles aislados que posee un tono de color gris diferente a la de los píxeles vecinos, mediante algoritmos establecidos se puede eliminar o reducir el ruido presente en la captura. Se diferencian tres tipos de ruidos en una imagen:

2.3.3.1 Ruido gaussiano

Crea pequeñas variaciones en la imagen, este ruido generalmente se produce por diversas ganancias que pueden existir en la cámara, perturbaciones en el medio de transmisión o ruido en los digitalizadores.

2.3.3.2 Ruido Impulsional

O también llamado ruido sal y pimienta, y se considera cuando un píxel toma un valor que no guarda relación con un dato ideal, es decir, puede tomar valores muy altos o muy bajos que se desprenden casi siempre por la saturación de los sensores o por un valor mínimo captado.

2.3.3.3 Ruido multiplicativo

El resultado que se obtiene es el de la multiplicación de dos señales.

2.4 Matlab

2.4.1 *Introducción a Matlab*

El término Matlab proviene de la abreviación de las palabras Matriz Laboratory, el cual se caracteriza por ser un entorno computacional de gran aplicación y desarrollo en los que cabe destacar que realiza análisis numérico.

2.4.2 *Manejo de las imágenes en Matlab*

MATLAB posee la capacidad de almacenar imágenes en arreglos bidimensionales o tridimensionales, en los arreglos bidimensionales cada matriz corresponde a la intensidad de pixel dentro de la imagen analizada, si es un arreglo tridimensional se analizan tres matrices diferentes donde la primera de ellas corresponde a la intensidad de rojo en los pixeles, la segunda a la intensidad de verde y la tercera a la intensidad de azul existente en los píxeles de cada imagen,

generalmente esta característica es propia de las imágenes a color o definidas como RGB (García Santillán, 2008).

2.4.2.1 Tipos de imágenes

Para el procesamiento de imágenes en el entorno MATLAB se recomienda el empleo de cuatro tipos de imágenes:

- Imágenes Indexadas
- Imágenes en Escala de Grises.
- Imágenes Binarias
- Imágenes RGB

2.4.2.2 Lectura y visualización de imágenes

Para realizar la lectura de imágenes en el entorno MATLAB se la realiza mediante la utilización del comando imread(‘nombre. Formato’); donde nombre corresponde al nombre del archivo con el cual se guardó la imagen, y formato es el tipo imagen que se almacenó, se destacan los formatos JPG, PNG, TIFF, GIF, BMP, etc (García Santillán, 2008).

Este comando se lo puede emplear de dos maneras:

- Sin asignar a una variable

La imagen se almacena en el Workspace del entorno y se le asigna a la variable “ans”, si se desea leer más de una imagen éstas se reescriben sobre la misma variable.

El comando a ser empleado es: >> imread('foto1.png');

- Con asignación a una variable

Se la emplea cuando se necesita leer varias imágenes, pues a cada una se la asigna un nombre diferente, al igual que en el caso anterior se almacenan dichas variables en el Workspace del entorno, se utiliza la siguiente estructura: >> P=imread('foto1.jpg');

Siempre que se emplea este comando se lo asocia con imshow, que sirve para desplegar la imagen en la pantalla, su estructura es: imshow(Variable).

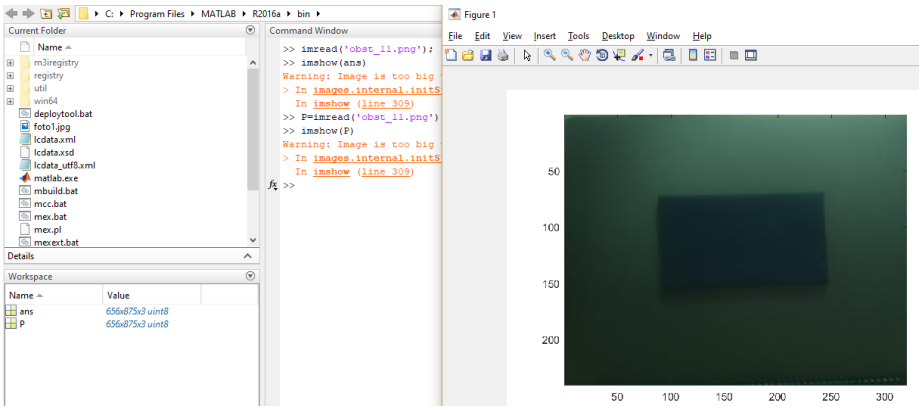


Figura 2-2: Lectura y visualización de imágenes en Matlab.
Realizado por: Tinajero José, 2017.

2.4.2.3 Conversión entre tipos de imágenes

Se emplea cuando se necesita modificar la imagen original de un formato dado a uno que mejore o facilite el procesamiento de la misma. La conversión de las imágenes se las puede realizar entre los formatos RGB, Binario, Escala de Grises e imágenes Indexadas.

Las principales funciones por destacar son:

Tabla 2-2: Funciones para conversiones de imágenes en Matlab.

FUNCIÓN	DESCRIPCIÓN	SINTAXIS
gray2ind	Convierte una imagen en escala de grises o binaria a una imagen indexada	[X, map] = gray2ind (I, n) [X, map] =gray2ind (BW, n)
ind2gray	Transforma una imagen indexada en una imagen en escala de grises	I=ind2gray (X, map)
mat2gray	Convierte una matriz en una imagen en escala de grises	I=mat2gray (A, [amin amax]) I=mat2gray(A)
rgb2gray	Convierte una imagen RGB o un mapa de colores en una imagen en escalas de grises	I=rgb2gray(RGB) newmap=rgb2gray(map)
ind2rgb	Convierte una imagen indexada en una imagen RGB	RGB=ind2rgb (X, map)
label2rgb	Convierte una matriz de título en una imagen RGB.	RGB=label2rgb(L) RGB=label2rgb (L, map) RGB=label2rgb (L, map, zerocolor)
Demosaic	Convierte una imagen codificada en un patrón Bayer en una imagen RGB.	RGB= (I, sensorAlignment)

Fuente: Matlab 2017a.
Realizado por: Tinajero José, 2017.

2.4.2.4 Escritura de imágenes

Para almacenar la imagen en un archivo se emplea el comando imwrite, el mismo que posee varias formas de escritura dependiendo de la forma en la que se desea almacenar dicha imagen.

- imwrite(A, filename). Se emplea para escribir lo datos que se encuentran en la variable A en el nombre de archivo especificado dentro de la carpeta actual. De acuerdo con la cantidad de

bits que posee la variable A se puede definir el formato de la imagen, es decir, si posee 8 bits es porque es del tipo uint8, pero si la imagen es del tipo uint16 es debido a que contiene 16 bits.

- imwrite (A, map, filename). Se emplea esta estructura cuando A contiene datos de una imagen indexada, por lo cual es necesario especificar el argumento de entrada map.
- imwrite (_, fmt). Escribe una imagen en el formato especificado por fmt, de forma independiente a la extensión que posee el nombre del archivo. Se puede especificar fmt después de los argumentos de entrada de cualquiera de las sintaxis anteriores.
- imwrite (_, Name, Value). Value especifica valores adicionales para diversos archivos como GIF, HDF, JPEG, PBM, PGM, PNG, PPM y TIFF de salida.

2.5 Filtrado y realzado de imágenes

2.5.1 *Operaciones básicas entre píxeles*

- Operaciones Aritméticas

Las operaciones Aritméticas más empleadas en el PDI son la suma, resta, multiplicación y división. Se debe considerar que para realizar cualquiera de estas operaciones básicas las imágenes a ser operadas deben tener el mismo tamaño en alto y ancho. Si se realiza dichas operaciones con valores constantes se puede aumentar o disminuir el brillo de una imagen.

En MATLAB para realizar a cabo dichas operaciones se emplean los siguientes comandos:

Tabla 3-2: Operaciones aritméticas aplicadas en Matlab.

FUNCIÓN ARITMETICA	DESCRIPCIÓN	SINTAXIS
Suma	Se suman dos imágenes de la misma medida y formato	K=imadd (I, J)
	Se suma una imagen con una constante para aumentar el brillo de la misma	K=imadd(I,60)
Resta	Se restan dos imágenes del mismo tamaño	K=imsubtract (I, J)
	Se resta una constante para disminuir el brillo de la imagen	K=imsubtract(I,60)
Multiplicación	Se multiplica dos imágenes del mismo tamaño, y se lo realiza multiplicando elemento a elemento de cada píxel.	K=immultiply (I, J)
	Se produce un escalonamiento de la imagen original, al ser menor que 1 se obscurece la imagen.	K=immultiply (I, a); a<1
	Se multiplica por una constante mayor a 1 para aumentar el brillo de la imagen	K=immultiply (I, a); a>1
División	Se realiza la división entre dos imágenes de las mismas dimensiones.	K=imdivide (I, J)

Fuente: Matlab 2017a.
Realizado por: Tinajero José, 2017.

- Operaciones lógicas

Las operaciones Lógicas se las aplica a imágenes binarias, las principales operaciones son AND, OR y NOT.

Tabla 4-2: Operaciones lógicas aplicables en Matlab.

FUNCIÓN LÓGICA	DESCRIPCIÓN	SINTAXIS
AND (Y)	Para emplear esta operación lógica A y B deben tener las mismas dimensiones, y a la salida entrega un arreglo que posee elementos de 0 y 1, 1 si es verdadero y 0 si es falso	C=and (A, B)
OR (O)	Retorna un arreglo que contiene elementos lógicos de 1 si es verdadero o 0 si es falso. A y B deben tener las mismas dimensiones donde ninguno de los elementos puede ser un valor escalar	C=or (A, B)
NOT (NO)	Niega los valores que posee a la entrada del arreglo, es decir contiene un elemento lógico 1 a la entrada su salida será 0 lógico y viceversa.	B=not(A)

Fuente: Matlab 2017a.
Realizado por: Tinajero José, 2017.

- Transformaciones geométricas

Operaciones Geométricas alteran las relaciones espaciales entre píxeles, las más comunes son la interpolación, la ampliación y reducción, rotación de imágenes y la correlación de matrices.

  - Interpolación de imágenes

    Es el proceso por el cual se estiman los valores de una imagen en una determinada sección de la imagen, en el entorno de desarrollo MATLAB existen tres métodos que pueden ser empleados y que funcionan de forma muy parecida, la diferencia entre ellos radica en el nivel de procesamiento pues mientras más píxeles se considera aumenta el nivel de complejidad de la interpolación (García Santillán, 2008).

Tabla 5-2: Métodos de interpolación de imágenes.

FUNCIÓN DE INTERPOLACIÓN	DESCRIPCIÓN
NEAREST	Vecino más próximo, es aquel pixel interpolado al cual se le asigna el valor que corresponde, es el método de interpolación por defecto en el entorno de desarrollo.
BILINEAR	Se realiza la interpolación bilineal, y es el promedio ponderado de los píxeles en una vecindad de 2x2
BICUBIC	Se emplea la interpolación bicúbica, y es el promedio ponderado de los píxeles en una vecindad de 4x4

Fuente: Visión Artificial y Procesamiento Digital de Imágenes usando Matlab.
Realizado por: Tinajero José, 2017.

  - Ampliación y reducción de imágenes

    Consiste en el cambio de tamaño de la imagen es decir se la puede ampliar o reducir, en el entorno de desarrollo MATLAB se puede especificar el tamaño de la imagen de salida, el método de interpolación a ser empleado, y el filtro para evitar la pérdida de información en cuando se comprime la imagen, el comando a ser empleado es: imresize (García Santillán, 2008).

La estructura a ser empleada puede tener los siguientes formatos:

B=imresize (a, SCALE), la operación devuelve una imagen que esta escalonada un número de veces determinado por SCALE, la imagen A puede ser de tipo RGB, indexada o en escala de grises.

B=imresize (A, [NUMROWS NUMCOLS]): cambia el tamaño de la imagen para que tenga un número definido de filas y columnas, el número de filas y columnas puede ser un valor indeterminado, es decir, NaN en este caso MATLAB calcula de forma automática el número de filas y columnas para mantener la relación de aspecto de la imagen de salida.

[Y, NEWMAP]=imresize (X, MAP, SCALE): cambia el tamaño de una imagen indexada.

[Y, NEWMAP]=imresize (X, MAP, [NUMROWS NUMCOLS]): cambia el tamaño de una imagen indexada, especificando el número de filas y columnas (MATLAB 2017a).

- Rotación de imágenes

  Consiste en rotar la imagen original en un determinado número de grados, también se puede especificar el método a ser empleado, el comando a ser empleado es imrotate.

  Se puede emplear varias sintaxis para rotar la imagen entre las cuales se destaca:

  B=imrotate (A, ANGLE): permite girar la imagen A un determinado número de grados en sentido antihorario, es decir, en contra de las manecillas del reloj; si se desea que la imagen gire en sentido horario se deberá especificar un valor negativo para el parámetro ANGLE. Cabe indicar que la imagen rota alrededor de su punto central y la imagen de salida es lo suficientemente grande para contener la imagen rotada (MATLAB 2017a).

  B=imrotate (A, ANGLE, METHOD): rota la imagen empleando la interpolación como un método, entre los cuales se puede emplear ‘nearest’, ‘bilinear’, ‘bicubic’.

  B=imrotate (A, ANGLE, METHOD, BBOX): gira la imagen A, especificando el tamaño de la imagen de salida, donde BBOX es una cadena de texto que puede tener los siguientes valores, ‘loose’ o ‘crop’

- Correlación de matrices

  Es una operación mediante la cual se calcula el valor de un píxel de salida como la suma ponderada de los píxeles vecinos, generalmente se la emplea para hallar la similitud o parecido entre los píxeles de una imagen, la correlación es un valor que varía entre -1 y 1, donde 1 y -1 representa alta correlación y 0 baja correlación entre los píxeles de las

imágenes, en MATLAB se emplea el comando corr2 y la condición predominante es que las imágenes deben tener la misma dimensión, es decir, que sus matrices sean iguales.

2.5.2 *Filtrado espacial*

El filtrado es una técnica que permite mejorar o modificar una determinada imagen, en la que se puede resaltar o atenuar algunas características de interés o análisis dentro de la captura. Se pueden destacar los filtros pasa bajo y filtros pasa alto.

2.5.2.1 Filtros espaciales pasa Bajo

Denominados también filtros suavizantes son usados con la finalidad de que la imagen sea un poco borrosa y reducir la cantidad de ruido existente en la imagen, se basa en el promedio de los píxeles adyacentes al píxel que se desea evaluar. El comando por usar es imfilter que realiza una operación del tipo lineal. Existe también el filtro de la mediana que tiene como objetivo sustituir el valor de un píxel por el de la mediana del conjunto formado por los 8 vecinos que rodean a dicho píxel, está a diferencia del filtro anterior no es lineal y se para reducir el ruido sal y pimienta, y se tiene como mira preservar los bordes de la imagen, su comando es medfilt2 (García Santillán, 2008).

2.5.2.2 Filtros especiales de paso Alto

Se los emplea para el realce de los bordes o detección de los contornos de la imagen a ser estudiada, el realce de los bordes comprende en resaltar los píxeles que contienen un valor diferente al de sus vecinos, mientras que la detección de los bordes corresponde a los límites de los objetos presentes en una imagen que son visibles mediante el cambio de la intensidad de los píxeles para la detección de los contornos se emplea el comando edge (García Santillán, 2008).

Edge encuentra los bordes en una imagen de intensidad o una imagen binaria, y devuelve una imagen binaria del mismo tamaño que la imagen de entrada, pero que tiene 1 donde la función encuentra los bordes y un 0 en el resto de la imagen (MATLAB2017a).

La sintaxis de la detección es:

B=edge (A, METHOD); donde A es la imagen de entrada, y METHOD es el algoritmo por el cual se pretende realizar la detección de los bordes, se destacan los métodos: ‘sobel’ y ‘canny’.

2.6 Operaciones morfológicas

Las operaciones morfológicas se encargan de suministrar información relevante sobre la forma o estructura de una imagen, basada en la geometría de los objetos, el análisis morfológico extrae los componentes más relevantes de los elementos existentes dentro de una imagen tales como la forma o tamaño de los objetos mientras que el procesado morfológico permite transformar la forma o tamaño de los objetos en una imagen (García Santillán, 2008).

2.6.1 *Elementos del proceso morfológico*

2.6.1.1 Conjuntos

En una imagen binaria son puntos que pertenecen a un espacio en dos dimensiones, los mismos que están conformados por una componente en el eje de las X y una en el eje de las Y, por lo que se define como un punto en un plano bidimensional (García Santillán, 2008).

2.6.1.2 Elementos estructurantes

Según Iván García en su libro Visión Artificial y Procesamiento Digital e imágenes usando Matlab expresa: “Examinar la estructura geométrica de una imagen usando como sonda un patrón de ajuste que se denomina elemento estructurante (SE). El SE puede tener cualquier tamaño y forma”.

En el entorno de desarrollo MATLAB se emplea el comando strel para poder emplear los elementos estructurantes, los elementos estructurantes que se pueden emplear son:

Tabla 6-2: Sintaxis de los elementos estructurantes en Matlab.

SINTAXIS	DESCRIPCIÓN
SE=strel (‘diamond’, R)	Crea una estructura plana en forma de diamante con el tamaño R que especifica la distancia desde el origen a la estructura del diamante. R no debe ser negativo
SE=strel (‘disk’, R, N)	Crea un elemento plano en forma de disco, donde R corresponde al radio del disco y N por defecto es 4.

SE=strel (‘line’, LEN, DEG)	Crea un elemento estructurador lineal plano, donde DEG especifica el ángulo de la línea con respecto del eje horizontal, y LEN es la distancia entre los centros de la estructuración.
SE=strel (‘octagon’, R)	Crea un octágono plano como elemento estructurador, donde R es la distancia desde el centro del elemento hacia los lados del octágono.
SE=strel (‘rectangle’, MN)	Crea una estructura rectangular con un tamaño específico definido en un vector con dos elementos no negativos que corresponden al número de fila y columna.
SE=strel (‘square’, W)	Crea un elemento estructurante cuadrado que posee un ancho W de píxeles
SE=strel (‘sphere’, R)	El elemento estructurante es una esfera cuyo radio está definido por R.
SE=strel (‘cube’, W)	Se crea un elemento estructurante Cubo que tiene por ancho W píxeles.
SE=strel (‘cuboid’, XYZ)	Crea un elemento estructurante cuboidal que posee un vector con tres elementos XYZ.

Fuente: Matlab 2017a.
Realizado por: Tinajero José, 2017.

2.6.1.3 Operadores morfológicos

Los operadores morfológicos más destacados son:

- Dilatación

Dilata la imagen en escala de grises, consiste en expandir los píxeles que posee una determinada imagen respecto del elemento estructurante sobre el que se opera, permite obtener los bordes de una imagen en análisis (García Santillán, 2008).

El comando que se emplea para el desarrollo en el entorno de programación es: imdilate, las sintaxis que se pueden emplear son:

Tabla 7-2: Sintaxis de la operación dilatación en Matlab.

SINTAXIS	DESCRIPCION
IM2=imdilate (IM, SE)	Dilata la imagen en escala de grises, binaria o indexada mediante un elemento estructurante con la función strel u offsetstrel.
IM2=imdilate (IM, NHOOD)	Dilata la imagen de entrada mediante una matriz por defecto NHOOD que está conformada por 0 y 1.
IM2=imdilate (IM, SE, PACKOPT)	Dilata una imagen cuando su imagen de entrada corresponde a una imagen binaria.
IM2=imdilate (……, SHAPE)	Dilata la imagen considerando el tamaño de la imagen de salida

Fuente: Matlab 2017a.
Realizado por: Tinajero José, 2017.

- Erosión

Erosiona la imagen, es decir, adelgaza la imagen sobre la cual se lo aplica, es la operación opuesta a la dilatación, se la emplea para mejorar la detección de los bordes de la imagen (García Santillán, 2008).

En MATLAB se emplea la palabra imerode:

Tabla 8-2: Sintaxis de la operación erosión en Matlab.

SINTAXIS	DESCRIPCION
IM2=imerode (IM, SE)	Erosiona una imagen en escala de grises, binaria o indexada mediante un elemento estructurante con la función strel u offsetstrel.
IM2=imerode (IM, NHOOD)	Erosiona la imagen de entrada mediante una matriz por defecto NHOOD que está conformada por 0 y 1.
IM2=imerode (IM, SE, PACKOPT)	Erosiona la imagen cuando su imagen de entrada corresponde a una imagen binaria.
IM2=imerode (……, SHAPE)	Erosiona la imagen considerando el tamaño de la imagen de salida

Fuente: Matlab 2017a.
Realizado por: Tinajero José, 2017.

- Apertura

Morfológicamente abre la imagen.

Consiste en realizar una erosión seguida de una dilatación de la imagen, eliminan los píxeles que quedan aisladas en la imagen analizada (García Santillán, 2008).

La palabra reservada con la que se relaciona esta operación es imopen.

Tabla 9-2: Sintaxis de la operación apertura en Matlab.

SINTAXIS	DESCRIPCIÓN
IM2=imopen (IM, SE)	Realiza la apertura morfológica de una imagen en la escala de grises o binaria con un elemento estructurador.
IM2=imopen (IM, NHOOD)	Realiza la apertura con el elemento estructurador STREL, que es una matriz de 0 y 1 que especifica la vecindad del elemento estructurador.

Fuente: Matlab 2017a.
Realizado por: Tinajero José, 2017.

- Cierre

Morfológicamente cierra la imagen.

Es la operación opuesta a la apertura, es decir, primero se realiza la dilatación de la imagen para posteriormente erosionarla con lo que se logra rellenar los pequeños agujeros que pueden existir en la imagen (García Santillán, 2008).

En el entorno de programación MATLAB se emplea la palabra imclose.

Tabla 10-2: Sintaxis de la operación cierre en Matlab.

SINTAXIS	DESCRIPCIÓN
IM2=imclose (IM, SE)	Realiza el cierre morfológico de una imagen en la escala de grises o binaria con un elemento estructurador.
IM2=imclose (IM, NHOOD)	Realiza el cierre con el elemento estructurador STREL, que es una matriz de 0 y 1 que especifica la vecindad del elemento estructurador.

Fuente: Matlab 2017a.
Realizado por: Tinajero José, 2017.

2.7 Segmentación

Es un proceso por el cual se divide a la imagen en regiones homogéneas o similares que posean una o más características en común con la finalidad de facilitar el procesamiento de las imágenes. Donde se puede destacar la segmentación basada en umbralización que es el proceso por el cual se convierte una imagen en escala de grises o una imagen a color en una imagen binaria, de forma que los objetos de interés se etiquetan con un valor diferente de los pixeles del resto de la imagen, esta técnica es muy empleada debido a su rapidez y bajo costo computacional, mientras que la segmentación por regiones que es otro método de segmentación consiste en separar los objetos de interés, para lo cual la imagen capturada se la particiona en diversas regiones las cuales contienen ciertas características y conjuntos de pixeles conectados.

2.8 Clasificación y reconocimiento

En un sistema de control visual el último objetivo corresponde a la etapa de clasificación, aquí es donde el sistema realiza el reconocimiento de una determinada letra, figura, rostro, etc. Existen varios métodos de los clasificadores de patrones entre los que se puede destacar:

- Adaptación

Denominado también como Pattern Matching, se emplean patrones prototipados para lograr un emparejamiento entre las imágenes, donde los métodos empleados dentro de este sistema de clasificación son:

Clasificador de mínima distancia y adaptación por correlación.

- Clasificadores estadísticamente óptimos

Este clasificador se basa en la teoría de la decisión estadística, para lo cual se emplea el clasificador Bayesiano de clases Gaussianas.

- Redes neuronales

Su principal fundamento se basa en la generación de soluciones flexibles y que se adaptan a cada problema, para que proporciones dichas soluciones las redes neuronales se basan en la teoría de aprendizaje estadístico.

2.9 Características de detección y extracción

Consiste en el registro de imágenes, detección de puntos de interés, extracción de descriptores de características y funciones de punto coincidente. Las características locales con sus respectivos descriptores son los componentes básicos de la mayoría de los algoritmos de visión por computadora. Las aplicaciones más notables consisten en el registro de imágenes, detección y clasificación de objetos, seguimiento y estimación de movimientos, destacando que dichos algoritmos utilizan características locales que permiten mejorar los cambios de escala, rotación u oclusión en la imagen (MATLAB2017a).

2.9.1 *Extracción de características locales*

Se emplea para la detección de los puntos de interés y la extracción de los descriptores de características, las principales funciones empleadas en Matlab para la detección de puntos de interés se detallan en la siguiente tabla:

Tabla 11-2: Descripción de las características de los puntos de interés.

FUNCIÓN	DESCRIPCIÓN	SINTAXIS
detectBRISKFeatures	Detecta las características BRISK y retorna el objeto BRISKPoints	points=detectBRISKFeatures(I) points=detectBRISKFeatures (I, Name, Value)
detectFastFeatures	Detecta las esquinas usando el algoritmo FAST y devuelve el objeto cornerPoints	points=detectFASTFeatures(I) points=detectFASTFeatures (I, Name, Value)
detectHarrisFeatures	Detecta esquinas usando el algoritmo Harris-Stephens y retorna un objeto cornerPoints	points=detectHarrisFeatures(I) points=detectHarrisFeatures (I, Name, Value)
detectMinEigenFeatures	Detecta las esquinas empleando el algoritmo del mínimo valor propio y devuelve el objeto cornerPoints	points=detectMinEigenFeatures(I) points=detectMinEigenFeatures (I, Name, Value)
detectMSERFeatures	Detecta las características MSER y retorna el objeto MSERRegions	regions=detectMSERFeatures(I) [regions, cc] =detectMSERFeatures(I) [__] =detectMSERFeatures (I, Name, Value)
detectSURFFeatures	Detecta las características SURF y devuelve el objeto SURFPoints	points=detectSURFFeatures(I) points=detectSURFFeatures (I, Name, Value)

Fuente: Matlab 2017a.
Realizado por: Tinajero José, 2017.

Para la extracción de los puntos de interés se emplean las siguientes funciones:

Tabla 12-2: Descripción de la extracción de los puntos de interés.

FUNCIÓN	DESCRIPCIÓN	SINTAXIS
extractFeatures	Extrae los descriptores de los puntos de interés.	[features, validPoints] =extractFeatures (I, points) [features, validPoints] =extractFeatures (I, points, Name, Value)
extractLBPFeatures	Extrae las características del patrón binario local (LBP)	features=extractLBPFeatures(I) features=extractLBPFeatures (I, Name, Value)
extractHOGFFeatures	Extrae las características del extracto de gradientes orientados (HOG)	features=extractHOGFFeatures(I) [features, validPoints] =extractHOGFFeatures (I, points)

Fuente: Matlab 2017a.
Realizado por: Tinajero José, 2017.

2.9.2 *Características de emparejamiento*

Consiste en el emparejamiento de las características de punto, se destacan dos funciones:

Tabla 13-2: Descripción del emparejamiento de los puntos de interés.

FUNCIÓN	DESCRIPCIÓN	SINTAXIS
matchFeatures	Encuentra las características de emparejamiento	indexPairs=matchFeatures (features1, features2) [indexPairs, matchmetric] = matchFeatures (features1, features2) [indexPairs, matchmetric] = matchFeatures (features1, features2, Name, Value)
showMatchedFeatures	Muestra las correspondientes características de los puntos	showMatchedFeatures (I1, I2, matchedPoints1, matchedPoints2) showMatchedFeatures (I1, I2, matchedPoints1, matchedPoints2, method)

Fuente: Matlab 2017a.
Realizado por: Tinajero José, 2017.

2.9.3 *Registro de imágenes*

Consiste en el registro automático de las imágenes o un marco de video, los algoritmos usados en la herramienta Computer Vision System Toolbox admiten el registro automático de imágenes, cabe destacar que dichos algoritmos emplean características para determinar las relaciones geométricas, las aplicaciones más comunes son mosaico de video, fusión de imágenes y visión 3D.

2.9.4 *Transformaciones geométricas*

Consiste en encontrar la similitud, afinidad y transformaciones proyectivas, para eliminar los valores atípicos que pueden presentarse en el desarrollo se puede emplear el método RANSAC

para calcular las transformaciones geométricas entre imágenes para finalmente aplicar la transformación geométrica para alinear las imágenes analizadas.

Tabla 14-2: Descripción de las principales transformaciones geométricas.

FUNCIÓN	DESCRIPCIÓN	SINTAXIS
estimateGeometricTransform	Estima la transformación geométrica de pares de puntos coincidentes.	Tform=estimateGeometricTransform (matchedPoints1, matchedPoints2, transformType) [Tform, inlierpoints1, inlierpoints2] = estimateGeometricTransform (matchedPoints1, matchedPoints2, transformType) [__, status] = estimateGeometricTransform (matchedPoints1, matchedPoints2, transform Type) [__] =estimate Geometric Transform (matchedPoints1, matchedPoints2, transformType, Name, Value)
Impar	Aplica una transformada geométrica a una determinada imagen	B=imwarp (A, tform) B=imwarp (A, D) [B, RB] =imwarp(Artform) B=imwarp (__, Interp) [B, RB] =imwarp (__, Name, Value)

Fuente: Matlab 2017a.
Realizado por: Tinajero José, 2017.

2.10 Controlador

2.10.1 *Raspberry PI 3*

La tarjeta Raspberry Pi son pequeñas computadoras que fueron desarrolladas con la finalidad de promover los procesos de aprendizaje sobre conceptos básicos de informática en países en vías de desarrollo. Se han desarrollado varias placas desde su creación, en las cuales se pueden destacar los modelos: Raspberry Pi 1 Modelo A y B, Raspberry Pi Zero, Raspberry Pi 2 y finalmente la Raspberry Pi3, el desarrollo de esta tecnología se remonta desde el año 2012, y su última tarjeta fue lanzada en febrero de 2016. La capacidad de la memoria varía de acuerdo con el modelo de la tarjeta, pero generalmente se encuentra entre 256MB y 1GB.

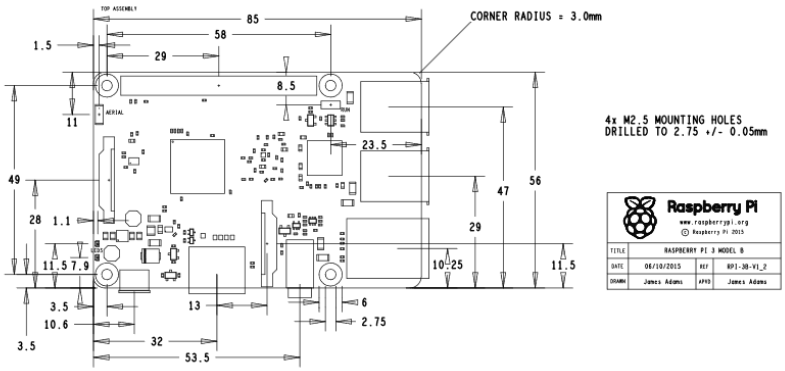


Figura 3-2: Arquitectura de la tarjeta Raspberry Pi 3.
Fuente: Raspberry Pi Foundation.

La Raspberry Pi 3 es la tercera generación de tarjetas y reemplaza a la Raspberry Pi 2 modelo B en el mes de febrero de 2016, comparada con la tarjeta anteriormente descrita posee las siguientes características (Raspberry Pi, 2017).

- CPU: Quad Core 1.2GHZ Broadcom BCM2837 de 64bits
- Memoria RAM: 1GB
- Wireless LAN BCM43438 y Bluetooth Baja Energía (BLE) en la tarjeta
- 40-pin extendidos GPIO
- 4 USB 2 puertos
- Salida estéreo de poste y puerto de video compuesto
- Full size HDMI
- CSI puerto de camera para conectar a la Raspberry PI Camera.
- DSI puerto para display para conectar el touchscreen display Raspberry Pi
- Puerto para MicroSD para cargar el sistema operativo y almacenamiento de datos
- Fuente de alimentación micro USB conmutada hasta 2,5ª

2.10.1.1 Cámara Monoscópica Raspberry Pi NoIR Camera V2

El módulo de cámara de infrarrojos V2 (Pi NoIR) reemplazó el módulo de cámara Pin oír original en el mes de abril del año 2016 (Raspberry Pi, 2017).

Es una cámara para visión nocturna desarrollada por la compañía Raspberry Pi, posee una calidad de 8 megapíxeles con sensor de imagen IMX219 de Sony diseñado para Raspberry, posee la capacidad de realizar capturas de imágenes de 3280x2464 píxeles y admite videos de hasta 1080p30. Se conecta a la placa base por medio de la interfaz CSI mediante su cable destinado para la conexión, es una cámara desarrollada con el objetivo de tomar foto fotografías infrarrojas y hacer fotografías en entornos con poca luz (Adafruit, 2017).

Los datos técnicos de la cámara Raspberry Pi NoIR son:

Tabla 15-2: Datos técnicos de la cámara Raspberry Pi NoIR.

PARÁMETRO	VALOR
Voltaje	5 Vcc
Corriente	1,8 m A

Fuente: Raspberry Pi Camera/NoIR Camera, Guía de Seguridad.
Realizado por: Tinajero José, 2017.

Figura 4-2: Cámara NoIR, Raspberry Pi.
Fuente: Raspberry Pi Foundation.

2.10.1.2 Fuente de alimentación

La tarjeta Raspberry Pi posee su propia fuente de alimentación la cual posee las siguientes características:

Tabla 16-2: Datos técnicos de la fuente para Raspberry Pi 3.

FUENTE CANAKIT	
Modelo(MODEL):	DCAR-052A5
Voltaje Entrada(INPUT):	100-240V AC
Frecuencia de Entrada	50-60Hz
Corriente de Entrada:	0.5 A
Voltaje de Salida(OUTPUT):	5V CC
Corriente de Salida:	2.5 A

Fuente: Datos de placa.
Realizado por: Tinajero José, 2017.

Figura 5-2: Fuente de alimentación Canakit.
Fuente: Raspberry Pi Foundation.

2.10.1.3 Controlador de motores

El dispositivo L293D es cuatro veces la mitad de los controladores H de alta corriente, está diseñado para proporcionar corrientes de accionamiento bidireccionales de hasta 600mA cunado trabaja en valores de voltaje de entre 4,5V a 36V. está diseñado para trabajar con cargas inductivas tales como relés, solenoides, motores paso a paso bipolares y motores de Corriente Continua, además de cargas que demandan alta corriente/alto voltaje en aplicaciones de suministro positivo. Cada salida es un circuito de accionamiento de tótem-pole completo, con un sumidero de transistor Darlington y una fuente pseudi-Darlington, los controladores se habilitante en pares (Texas Instruments Incorporated, 2016).

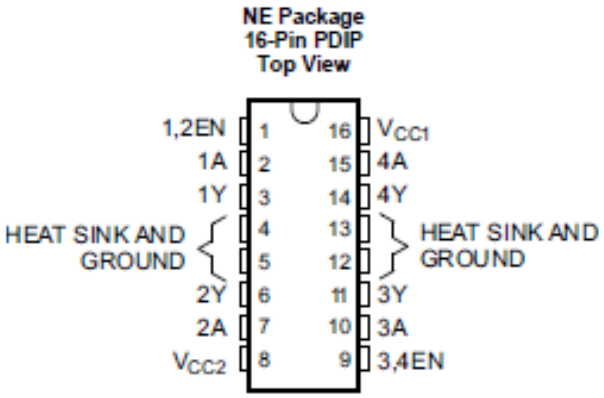
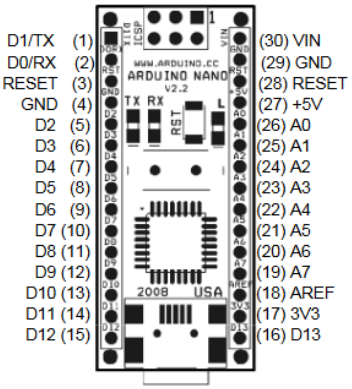


Figura 6-2: Integrado L293D.
Fuente: Datasheet L293D, Texas Instruments.

2.10.2 *Arduino Nano*

El Arduino Nano es una placa bastante pequeña, completo y fácil de usar basado en el microcontrolador Atmega328, ofrece la misma conectividad y especificaciones de la placa UNO en un formato más pequeño. Para programar un Arduino Nano se emplea el software de Arduino y al igual que el resto de placas puede funcionar en línea como fuera de línea (Arduino CC, 2017).



Pin No.	Name	Type	Description
1-2, 5-16	D0-D13	I/O	Digital input/output port 0 to 13
3, 28	RESET	Input	Reset (active low)
4, 29	GND	PWR	Supply ground
17	3V3	Output	+3.3V output (from FTDI)
18	AREF	Input	ADC reference
19-26	A7-A0	Input	Analog input channel 0 to 7
27	+5V	Output or Input	+5V output (from on-board regulator) or +5V (input from external power supply)
30	VIN	PWR	Supply voltage

Figura 7-2: Arquitectura Arduino Nano.
Fuente: Arduino CC.

2.10.2.1 Sensor ultrasónico

Es un sensor de distancia/proximidad cuya principal función es evitar obstáculos en diferentes proyectos de robótica, se emplean básicamente con las tarjetas Arduino y suministran al proyecto información del entorno. Las aplicaciones de estos dispositivos son muy amplias y eficaces además del bajo costo económico que representa adquirirlos (C, Scott, 2017).

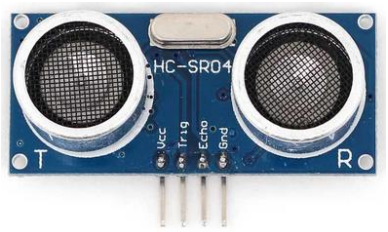


Figura 8-2: Sensor ultrasónico.
Fuente: Arduino CC.

Los datos más relevantes del dispositivo son:

Tabla 17-2: Detalle de los pines del sensor Ultrasónico HC-SR04.

PIN	DESCRIPCIÓN
Vcc	Voltaje de polarización +5V
GND	Pin GND, conexión a tierra
Echo	Recibe la señal del Trig
Trig	Emite una señal sonora

Fuente: Datos de Placa.
Realizado por: Tinajero José, 2017.

CAPÍTULO III

3. METODOLOGÍA DE LA INVESTIGACIÓN

3.1 Construcción de la plataforma diferencial

3.1.1 *Elementos que componen la estructura*

La estructura del robot diferencial está compuesto por elementos pasivos y activos, como por ejemplo, el chasis entre los pasivos y los motores entre los activos.

Tabla 1-3: Elementos de la plataforma diferencial.

ELEMENTO	CANTIDAD
Chasis 215mmx150mmx110mm	1
Chasis 150mmx100mm	1
Motores CC 3Va 6V	2
Ruedas Ⅽ65mm	2
Rueda Loca Ⅽ24mm	1
Tornillos y tuercas	Indeterminados

Realizado por: Tinajero José, 2017.

3.1.2 *Estructura finalizada*

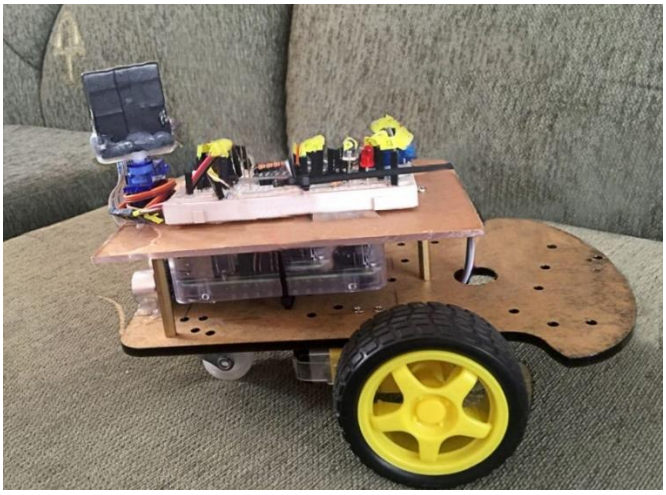


Figura 1-3: Plataforma diferencial.
Realizado por: Tinajero José, 2017.

3.2 Diseño del Software en Arduino

3.2.1 *Delimitación del área de trabajo*

El área de trabajo para el funcionamiento del robot móvil diferencial se basa en tres parámetros, los mismos que fueron seleccionados en base a la protección del móvil, a los desplazamientos a realizarse, y a la distancia que se deben realizar la captura de las imágenes.

Tabla 2-3: Parámetros delimitación del área del trabajo.

Parámetro	Valor
Distancia Máxima	300 cm
Distancia de Obstáculo	15 cm
Distancia e Bloqueo	8 cm

Realizado por: Tinajero José, 2017.

3.2.2 *Diagrama de flujo para la evasión de obstáculos*

El presente diagrama de flujo está compuesto de estructuras IF las mismas que se encuentran anidadas para determinar la mejor secuencia para la evasión de obstáculos, para la estructura IF se encuentra formada por operadores lógicos AND (&&) para que se cumpla la condición si y solo si las variables que se comparan sean verdaderas.

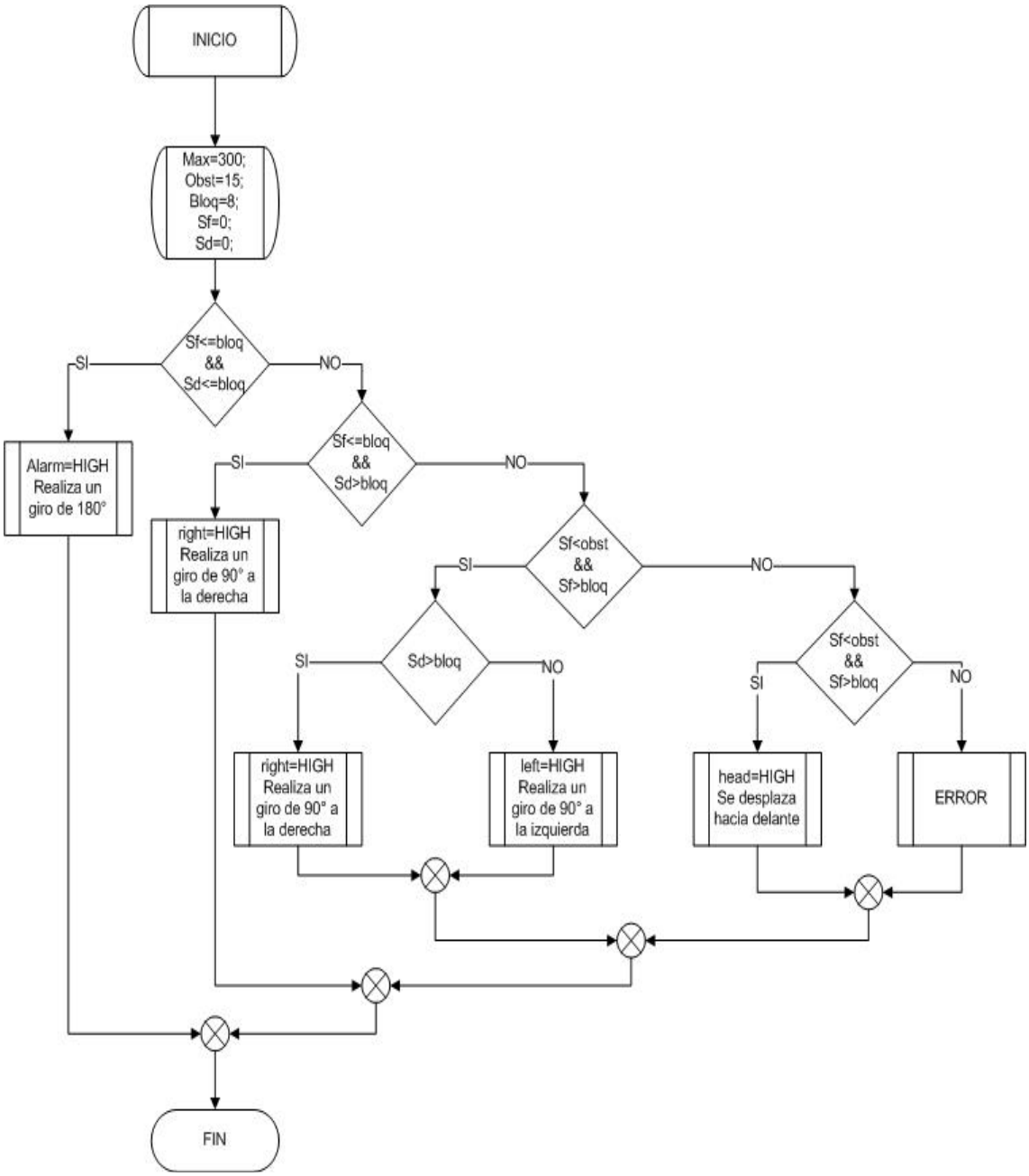


Figura 2-3: Diagrama de flujo, programación Arduino.
Realizado por: Tinajero José, 2017.

3.2.3 *Esquema de simulación*

Se presenta la estructura de conexión entre los sensores ultrasónicos y la tarjeta Arduino Nano especificando los pines de conexión, además de los pines que serán empleados para los actuadores que indicaran las acciones a ser tomadas por la placa.

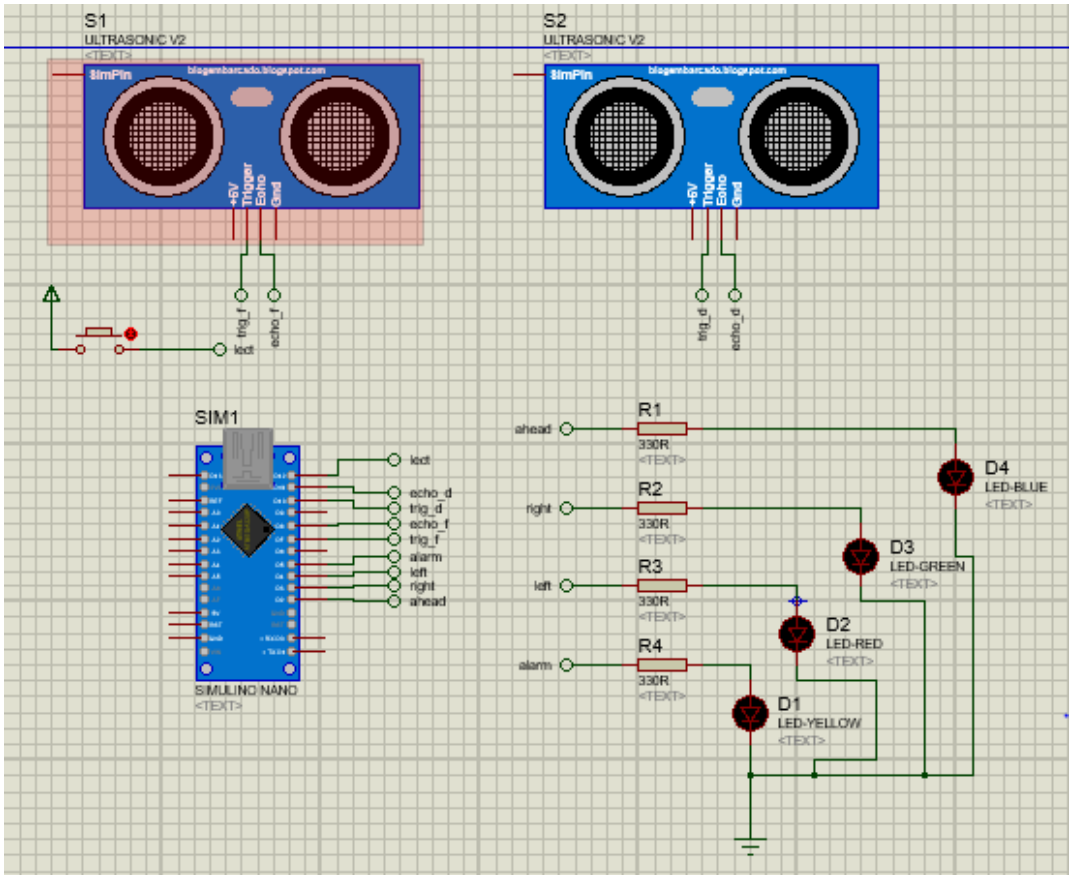


Figura 3-3: Esquema de simulación Proteus.
Realizado por: Tinajero José, 2017.

3.3 Diseño del Software en Raspberry

3.3.1 *Carga de programa de Linux específico para Matlab*

Para poder cargar el software en Matlab este se debe ejecutarse en modo administrador. De forma adicional se debe instalar el Package de Raspberry Pi en el entorno de desarrollo Matlab, eso se lo realiza dando click en Add-Ons y se descarga todos los paquetes correspondientes a la librería. En la ventana de comandos se debe ingresar: targetupdater, lo que cargará la siguiente captura:

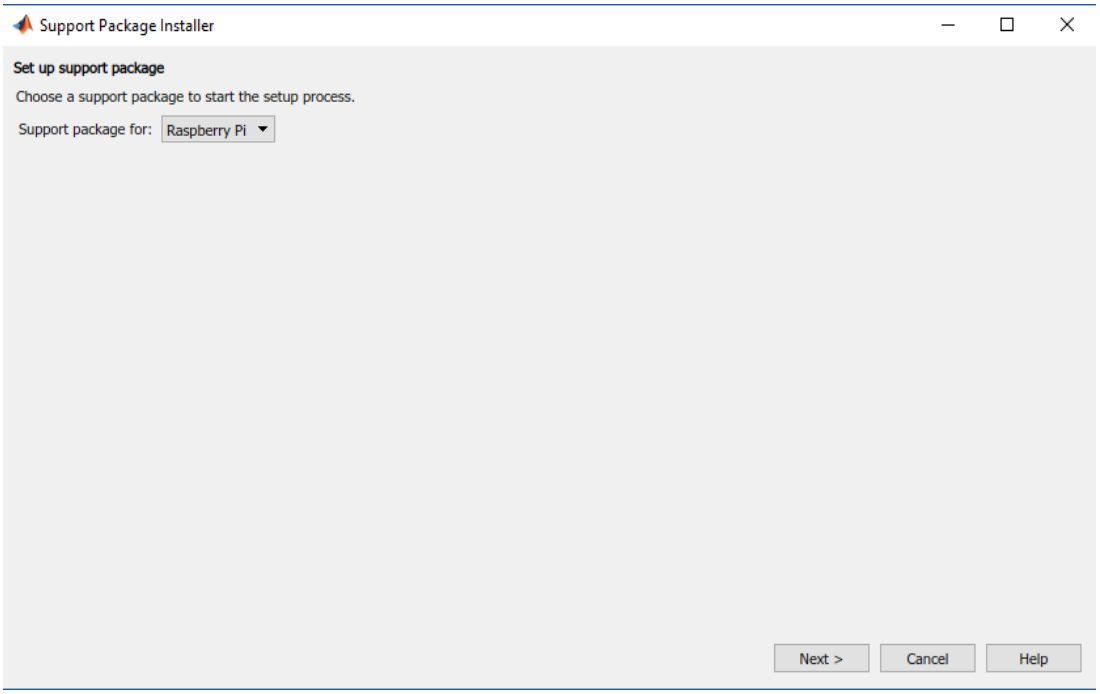


Figura 4-3: Selección de la librería Raspberry Pi.
Realizado por: Tinajero José, 2017.

En la siguiente ventana se deberá seleccionar el modelo de la tarjeta, es decir, Raspberry Pi 3 Model B.

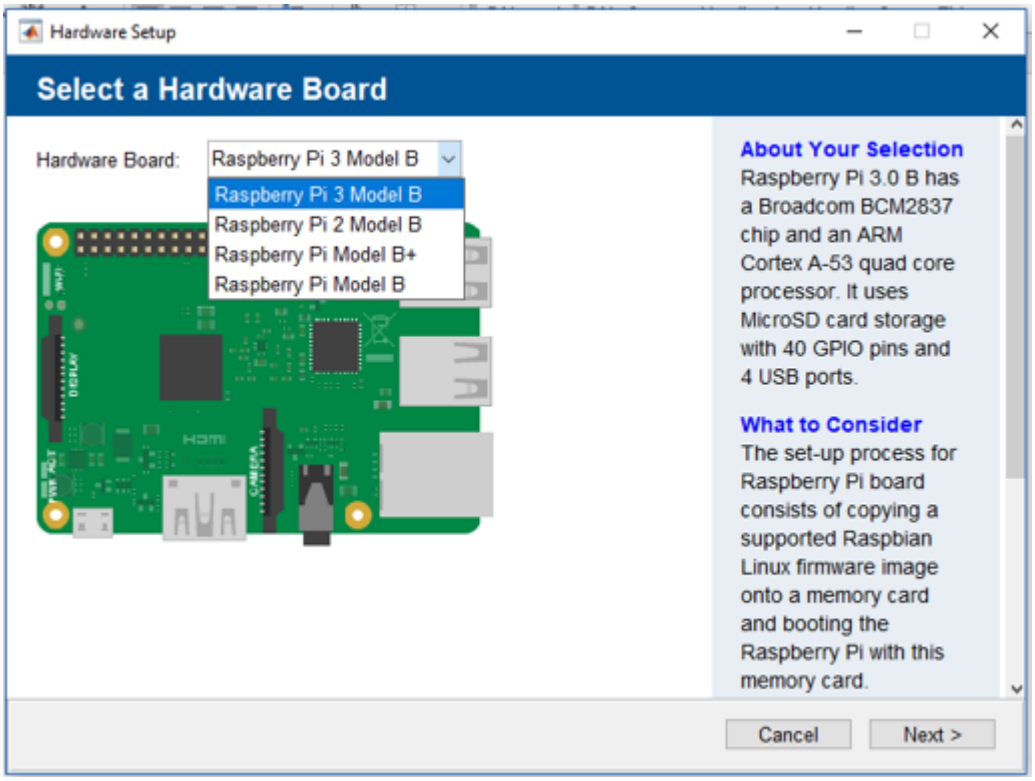


Figura 5-3: Selección de la tarjeta Raspberry Pi.
Realizado por: Tinajero José, 2017.

Una vez elegida la tarjeta, se debe seleccionar el método de comunicación entre Matlab y Raspberry Pi 3, en esta investigación se la realiza mediante un router inalámbrico.

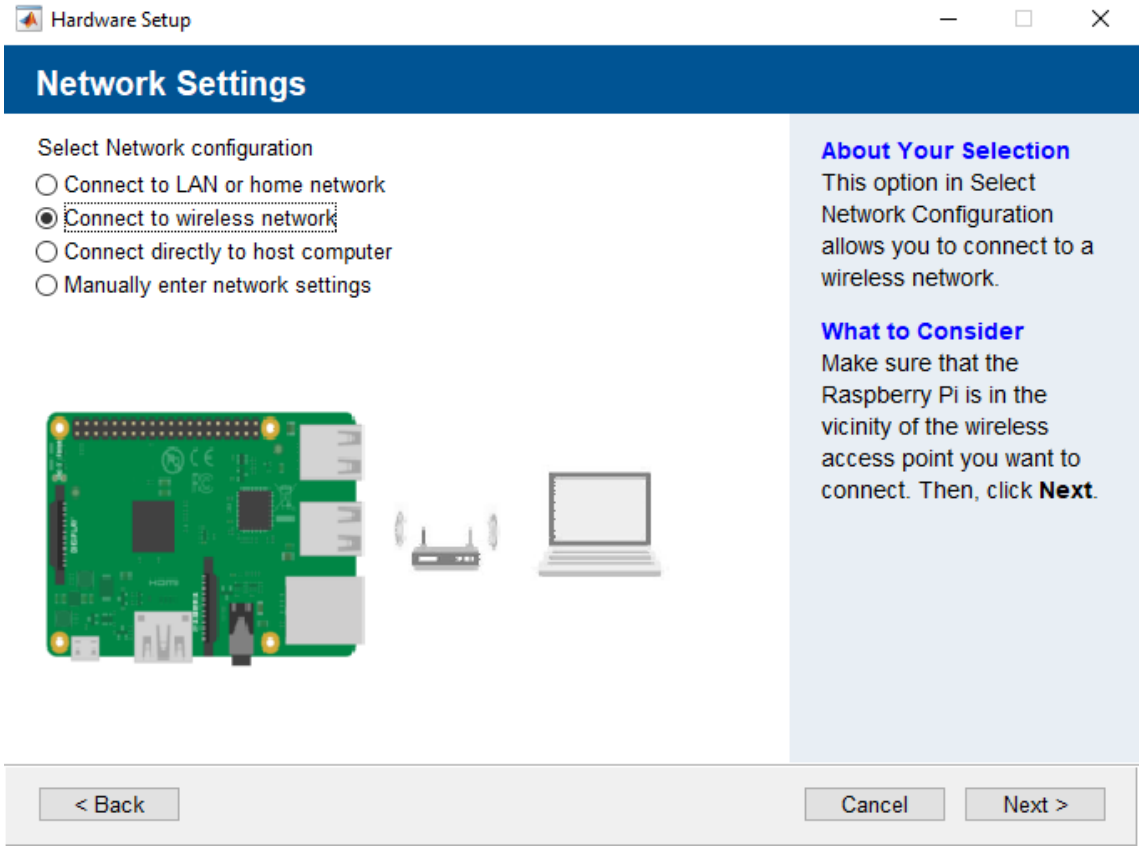


Figura 6-3: Selección del tipo de conexión.
Realizado por: Tinajero José, 2017.

A continuación, se debe ingresar el SSID que corresponde al nombre de la red inalámbrica y la correspondiente contraseña de dicha red, se puede configurar de dos maneras la dirección IP de la tarjeta, la primera consiste en que se asigne de forma automática una dirección, mientras que la segunda forma es ingresar de forma manual la dirección IP, la máscara de la red y la puerta de conexión. Se asigna una dirección estática para que la comprobación de la conexión sea exitosa. La forma de asignación se indica en las imágenes adjuntas.

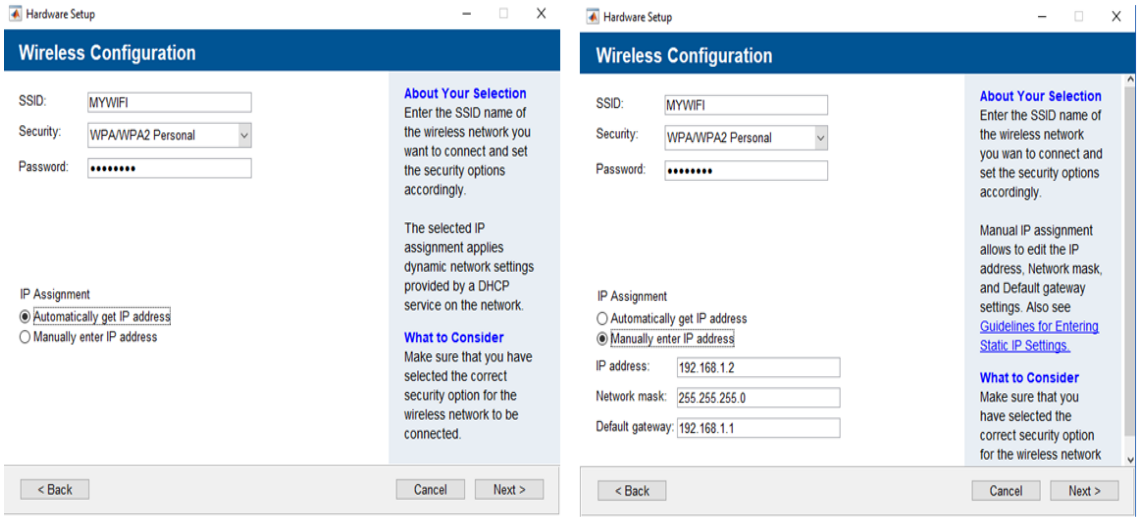


Figura 7-3: Asignación de la dirección IP.
Realizado por: Tinajero José, 2017.

Una vez ingresado estos datos el siguiente paso será ingresar la tarjeta de memoria que previamente deberá estar formateada, en la pantalla aparecerá cual es la letra asignada a la tarjeta SD.

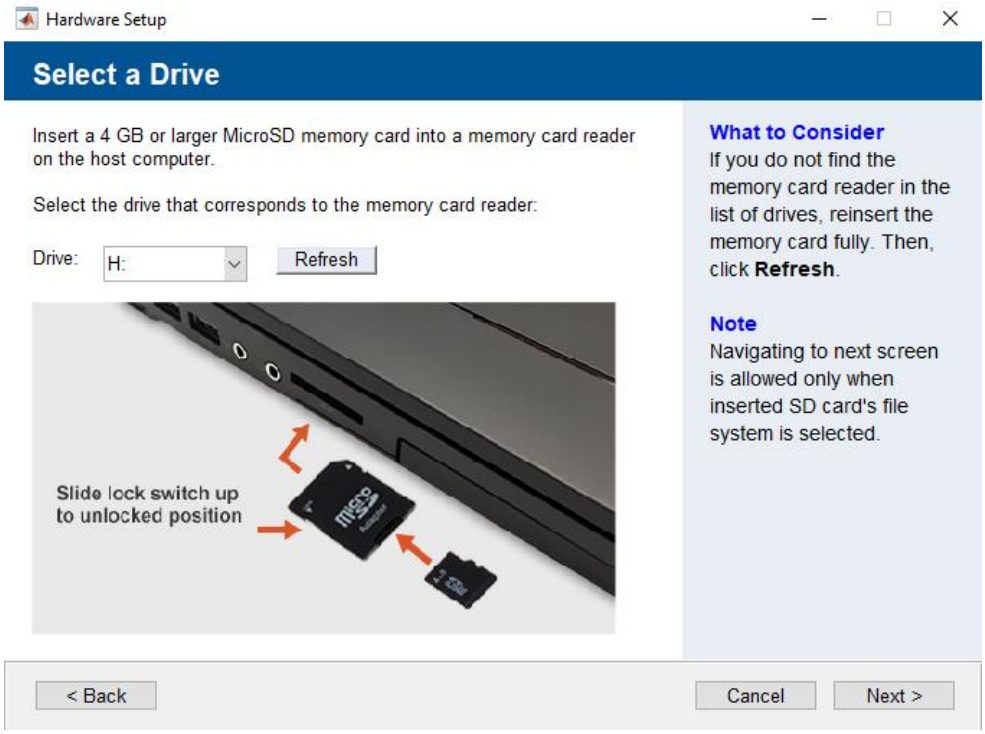


Figura 8-3: Identificación de la tarjeta MicroSD.
Realizado por: Tinajero José, 2017.

La siguiente ventana indica que se va a escribir sobre la tarjeta, a lo cual se debe dar click en el boton “write” y esperar algunos minutos hasta que finalice la carga del programa en la tarjeta. Una vez cargado el programa en la tarjeta el último paso corresponde hacer la conexión con la tarjeta Raspberry PI, para lo cual se encenderá el Led 0 en el mismo que se encuentra ubicado sobre la placa.

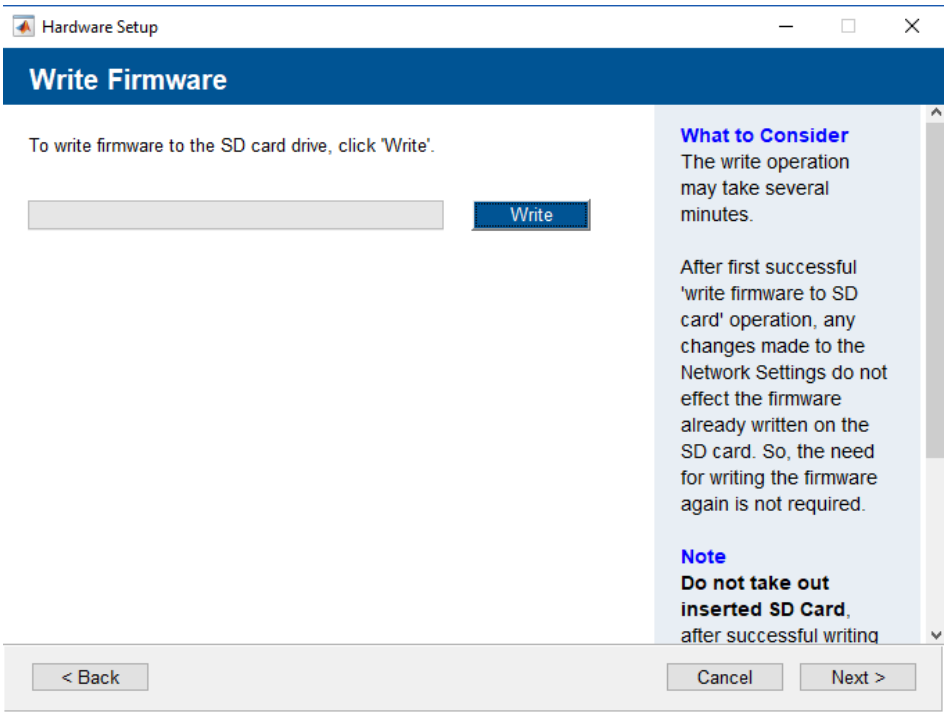


Figura 9-3: Escritura sobre la tarjeta MicroSD.
Realizado por: Tinajero José, 2017.

Antes de realizar la prueba es necesario verificar la configuración de la red inalámbrica ejecutando ipcongif en el CMD del computador. El último paso del proceso de configuración del hardware Raspberry, de forma adicional para que la conexión con la tarjeta sea exitosa se debe asignar una IP estática a la máquina donde se ejecuta Matlab.

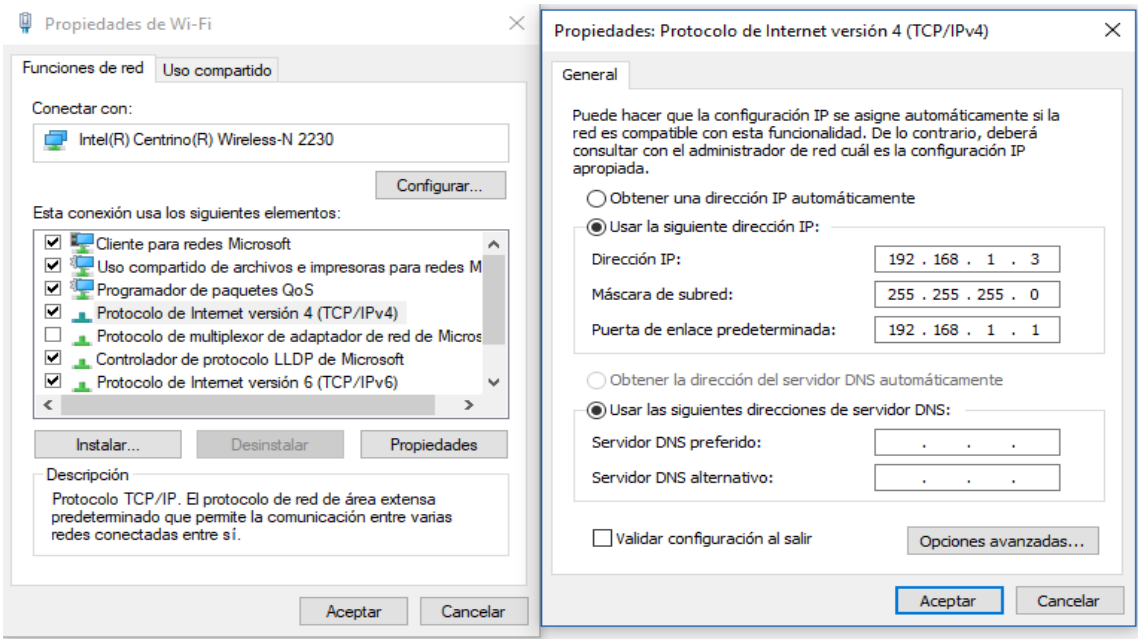


Figura 10-3: Asignación de la dirección estática del computador.
Realizado por: Tinajero José, 2017.

Realizado todas estas acciones de forma correcta el mensaje que se despliega debe ser igual al que se muestra en la figura adjunta:

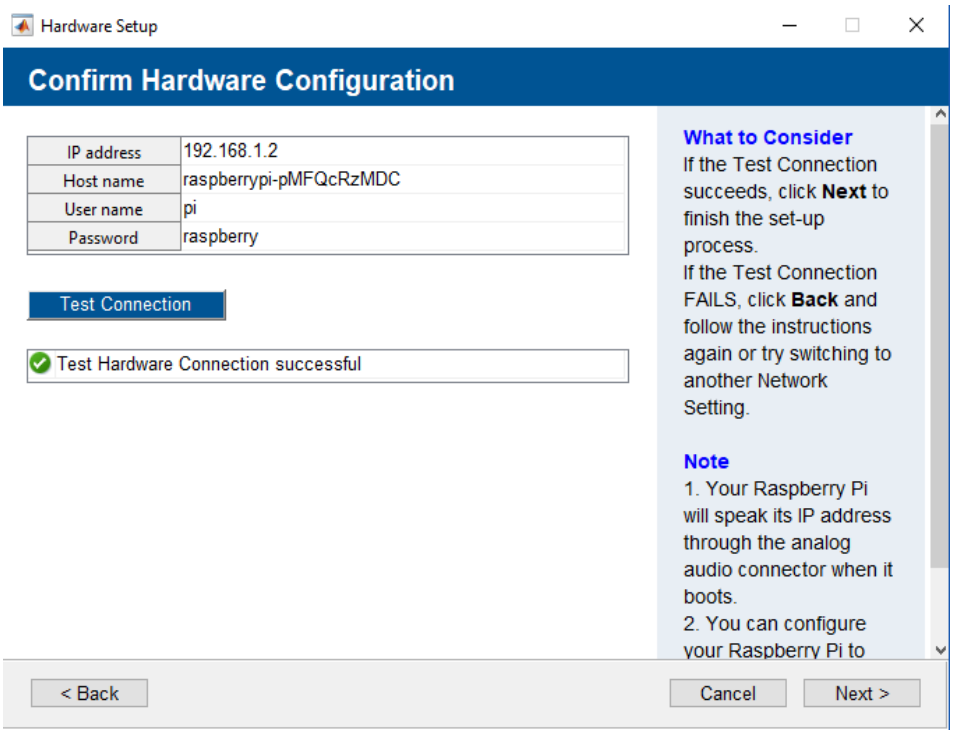


Figura 11-3: Conexión exitosa entre la Raspberry Pi y el computador.
Realizado por: Tinajero José, 2017.

3.3.2 *Comprobación de la conexión entre Matlab y Raspberry*

Una vez finalizada la carga del programa en la tarjeta MicroSD, se puede verificar fácilmente la conectividad entre Matlab y la tarjeta Raspberry Pi 3, en la ventana de comando se debe digitar el comando: mypi = raspi; la información que devuelve después de unos segundos es la que se muestra en la imagen, en la cual se puede apreciar claramente el modelo de placa, la dirección IP, pines disponibles, led disponible y el modelo de cámara que se encuentra conectada a la placa.

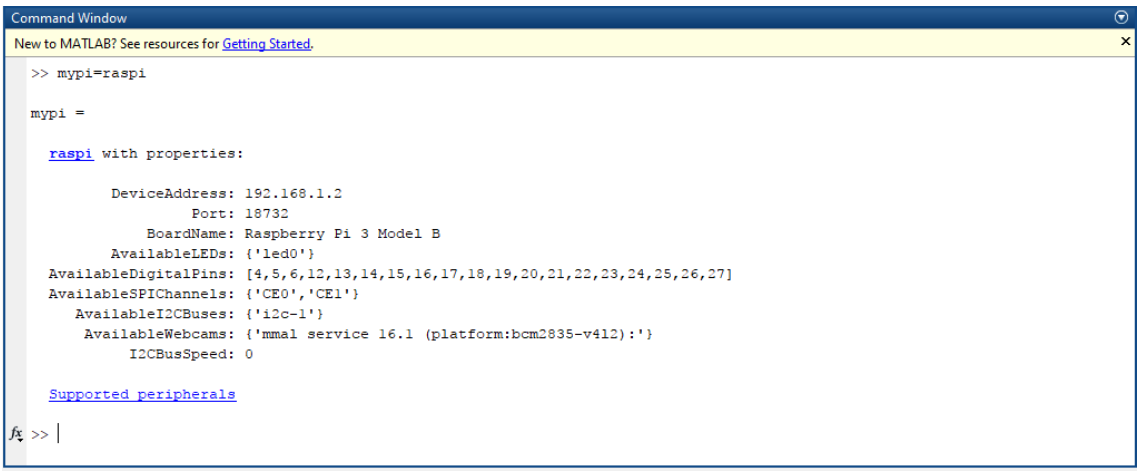


Figura 12-3: Datos de conexión entre Matlab y Raspberry Pi.
Realizado por: Tinajero José, 2017.

3.3.3 *Capturas de imágenes mediante la cámara a borde sobre la Raspberry*

Para realizar las capturas de imágenes se emplean los comandos: mycam=cameraboard(mypi, 'Resolution','320x240'); donde los parámetros empleados son la conexión con la tarjeta, la resolución de 320x240 píxeles, se tomó ese valor para que el procesamiento de las imágenes no sea excesivo para la computadora y tarjeta. La figura muestra las líneas de código empleadas para realizar la captura del ambiente.

```
clc; clear all;
mypi = raspi;
mycam = cameraboard(mypi,'Resolution','320x240');
for ii=0:2
    fot1=snapshot(mycam);
    image(fot1);
    figure();
end
```

Figura 13-3: Líneas de código para captura de imágenes.
Realizado por: Tinajero José, 2017.

Como resultado de la ejecución del scrip se puede visualizar la siguiente captura:

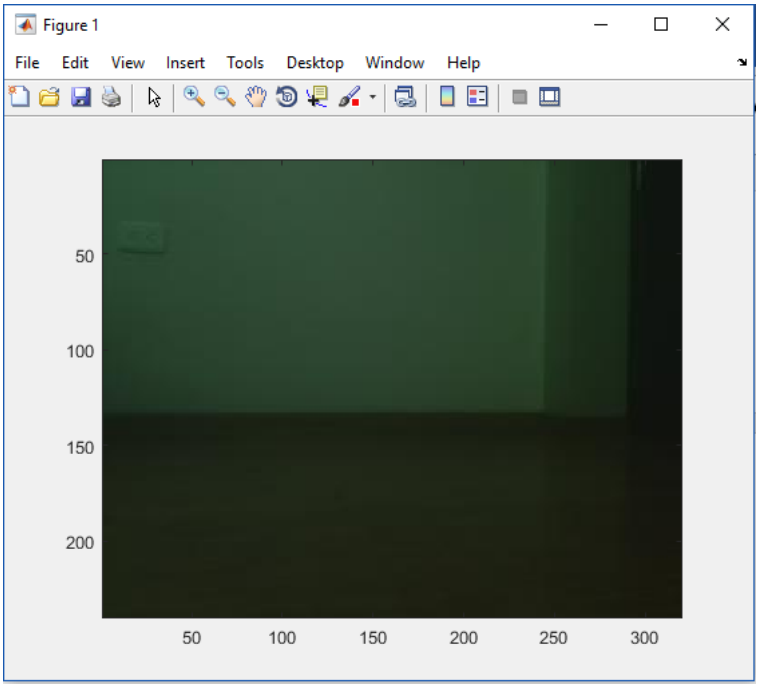


Figura 14-3: Captura realizada por la cámara de la Raspberry PI.
Realizado por: Tinajero José, 2017.

3.3.4 *Control del servomotor*

Para realizar el control de servomotor después de iniciar la respectiva conexión con la tarjeta se declara la variable s=servo(mypi,4); donde servo es el comando que indica que la tarjeta trabajará con un servomotor de 180° de rotación, mypi es la conexión que se estable entre el entorno de

programación y Raspberry Pi 3, y 4 corresponde al número del pin de la tarjeta al cual está conectada la señal de control del servomotor, la programación empleada permite realizar un giro de 45° cada 2 segundos, y se la presenta a continuación:

```
clc; clear all;
mypi = raspi;
s = servo(mypi,4);
pos=0;
for i=0:4
    writePosition(s,pos);
    pos=pos+45;
    pause(2);
end
```

Figura 15-3: Código fuente para control del Servomotor.
Realizado por: Tinajero José, 2017.

3.3.5 *Lectura y escritura en los puertos GPIO*

Una vez que se logra establecer la comunicación entre los elementos de análisis, para emplear los pines de la tarjeta se los debe declarar con la función que se desea que cumplan, ya que los 40 pines que posee la Raspberry son de propósito general, es decir, se los puede emplear como entradas o salidas.

Los comandos que se deben emplear para declarar a un pin como entrada es, `configurePin(mypi,13,'DigitalInput');` y si se desea que el pin sirva para escritura se debe configurar de la siguiente manera: `configurePin(mypi,21,'DigitalOutput').`

Dentro del proyecto desarrollado se ha empleado tanto pines de entrada como de salida, los pines de entrada sirven para corroborar si existe o no un obstáculo delante de la plataforma diferencial, mientras que los pines de salida son indicadores luminosos o control de los motores para el desplazamiento del robot.

3.3.6 *Procesamiento de imágenes*

3.3.6.1 Extracción de color

Las figuras a ser analizadas son de color azul, por lo cual de las imágenes capturadas se descomponen en sus tres matrices RGB, de las cuales solo se trabaja con la matriz azul de cada captura realizada. Se analizan los valores mayores de 10.5 para que las imágenes contengan las figuras a ser analizadas. El scrip desarrollado para la extracción del color azul de las imágenes es el siguiente:

```
% descomposicion en matrices individuales
o1r=o1(:,:,1); o1g=o1(:,:,2); o1b=o1(:,:,3);
o2r=o2(:,:,1); o2g=o2(:,:,2); o2b=o2(:,:,3);
o3r=o3(:,:,1); o3g=o3(:,:,2); o3b=o3(:,:,3);
% se realiza la extraccion del color azul de cada obstaculo
jb_o1=o1b-(o1r/2)-(o1g/2);
jb_o2=o2b-(o2r/2)-(o2g/2);
jb_o3=o3b-(o3r/2)-(o3g/2);
%se crea una nueva imagen en la que se define el color azul
b_o1=jb_o1>10.5; figure; imshow(b_o1);
b_o2=jb_o2>10.5; figure; imshow(b_o2);
b_o3=jb_o3>10.5; figure; imshow(b_o3);
```

Figura 16-3: Extracción del color azul de las capturas.
Realizado por: Tinajero José, 2017.

Las imágenes resultantes de este preprocesamiento son:

Figura 17-3: Imágenes con detección del color azul.
Realizado por: Tinajero José, 2017.

3.3.6.2 Modificación del tipo de imágenes

Una vez que se extrae el color azul de las imágenes se las debe transformar a imágenes binarias, se emplea el comando im2bw, en el cual se debe especificar el valor de contraste con el cual se transformará la imagen al formato deseado.

```
% se realiza la binarizacion de las imagenes que se extrajo el color azul
bi_o1=im2bw(b_o1,0.5); figure; imshow(bi_o1);
bi_o2=im2bw(b_o2,0.5); figure; imshow(bi_o2);
bi_o3=im2bw(b_o3,0.5); figure; imshow(bi_o3);
```

Figura 18-3: Código fuente de la conversión de imágenes.
Realizado por: Tinajero José, 2017.

El resultado de aplicar dicho comando son las imágenes capturas:

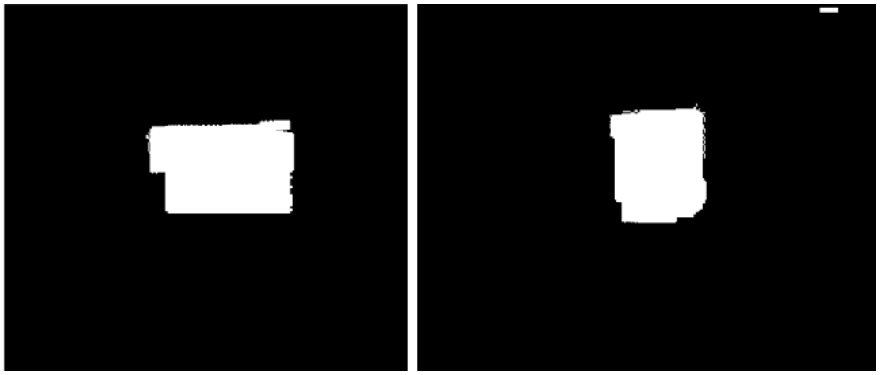


Figura 19-3: Imágenes binarias.
Realizado por: Tinajero José, 2017.

3.3.6.3 Creación de los elementos estructurantes

El siguiente proceso fue el desarrollo de los elementos estructurantes, entre los que cabe destacar que las figuras a ser empleados son: un rectángulo cuyas dimensiones están almacenadas en el vector con dos valores, el cuadrado que posee un valor de lado y una circunferencia con un R establecido de 90, las figuras se forman mediante el comando strel.

```
%% se aplica las operaciones para el rectangulo
FC=[150;150];
SER=strel('rectangle',FC);
ero_o1=imerode(bi_o1,SER); figure; imshow(ero_o1);
dil_o1=imdilate(ero_o1, SER); figure; imshow(dil_o1);
e_o1=edge(dil_o1, 'canny'); figure; imshow(e_o1); % se filtra todos los bordes de la imagen
% el algoritmo canny detecta todo el borde de la figura debera ser la implementada
SES=strel('square',150);
ero_o2=imerode(bi_o2,SES); figure; imshow(ero_o2);
dil_o2=imdilate(ero_o2,SES); figure; imshow(dil_o2);
e_o2=edge(dil_o2, 'canny'); figure; imshow(e_o2);
SED=strel('disk',90, 8);
ero_o3=imerode(bi_o3,SED); figure; imshow(ero_o3);
dil_o3=imdilate(ero_o3,SED); figure; imshow(dil_o3);
e_o3=edge(dil_o3, 'canny'); figure; imshow(e_o3);
```

Figura 20-3: Código fuente de los elementos estructurantes.
Realizado por: Tinajero José, 2017.

Al aplicar dichos comando con los parámetros necesarios para elemento estructurante se obtienen las siguientes figuras:

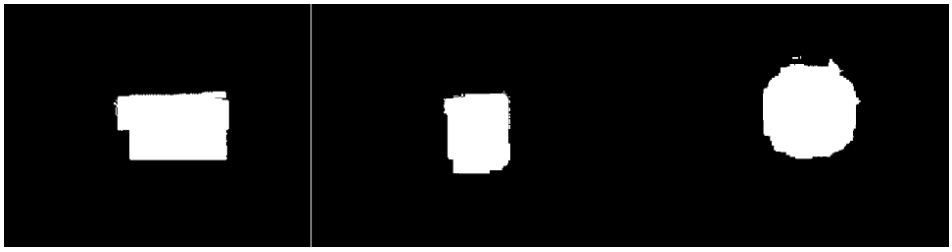


Figura 21-3: Imágenes binarias con elementos estructurantes.
Realizado por: Tinajero José, 2017.

3.3.6.4 Operación de erosión de las imágenes

Para realizar la erosión de los elemento estructurantes se emplea el comando imerode(imagen, SER); donde imagen corresponde a la respectiva imagen binaria, y SER el elemento estructurante definido anteriormente. Con esta operación se pretende eliminar cualquier bit que distorsione las capturas realizadas.

Las imágenes resultantes de esta operación son:

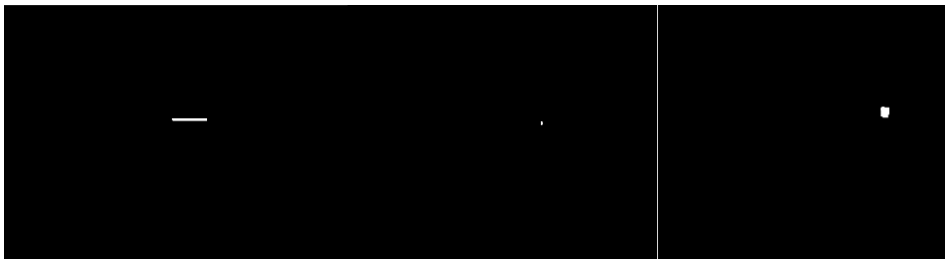


Figura 22-3: Imágenes binarias aplicadas erosión.
Realizado por: Tinajero José, 2017.

3.3.6.5 Operación de dilatación de las imágenes

Una vez erosionadas las imágenes, se las dilata para obtener una imagen completamente definida de los elementos en análisis, para llevar a cabo esta operación se recurre al comando imdilate(imagen, SER). Después de la ejecución del comando las imágenes obtenidas son:

Figura 23-3: Imágenes binarias aplicadas dilatación.
Realizado por: Tinajero José, 2017.

3.3.6.6 Detección de los bordes de las imágenes

El comando edge se emplea para la detección de los contornos de las imágenes anteriormente dilatadas, se emplea el método "canny" pues en pruebas realizadas se diferencia de forma clara que es el mejor método para realizar la extracción total del borde de las figures.

El resultado de este proceso son las figuras:

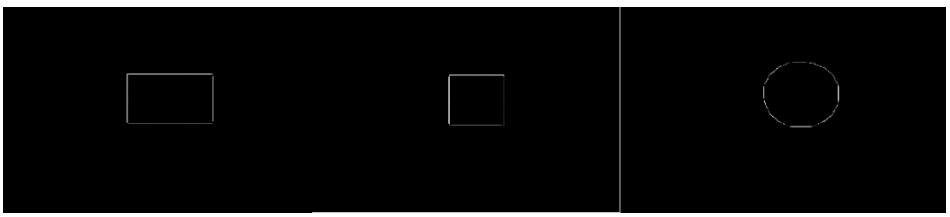


Figura 24-3: Imágenes binarias detectadas el borde del elemento.
Realizado por: Tinajero José, 2017.

3.3.6.7 Detección de los puntos más relevantes de las imágenes

Para la detección de los puntos más importantes de cada una de las imágenes se emplea el comando detectSURFFeatures debido a que es el algoritmo que mejores prestaciones entrega en

la detección de figuras con varios detalles, además que se acopla de forma sencilla a la aplicación, en el entorno de desarrollo se extraen diversas cantidades de puntos dependiendo de la figura a ser analizada.

```
42      % se realiza la extraccion de los puntos mas relevantes de cada obstaculo ,
43      % imagen preprocesada
44 -    reg1 = detectSURFFeatures(e_o1); figure(1); imshow(e_o1); hold on; plot(reg1.selectStrongest(100));
45 -    reg2 = detectSURFFeatures(e_o2); figure(2); imshow(e_o2); hold on; plot(reg2.selectStrongest(100));
46 -    reg3 = detectSURFFeatures(e_o3); figure(3); imshow(e_o3); hold on; plot(reg3.selectStrongest(100));
```

Figura 25-3: Código fuente de la detección de los puntos más relevantes.
Realizado por: Tinajero José, 2017.

Las imágenes resultantes de este proceso son:

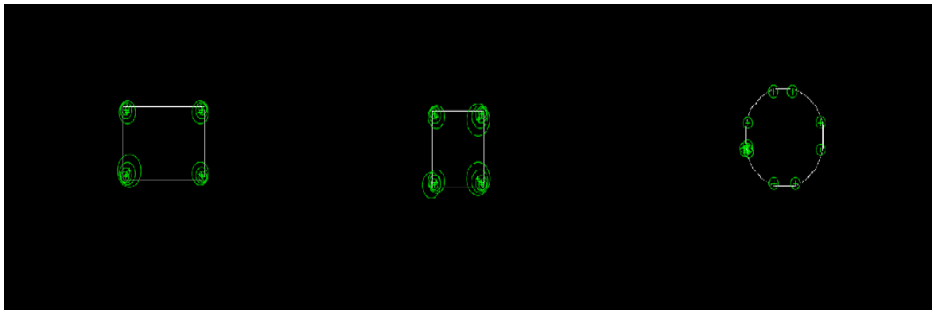


Figura 26-3: Imágenes binarias con los puntos más relevantes.
Realizado por: Tinajero José, 2017.

3.3.6.8 Extracción de los puntos más relevantes de las imágenes

Para la extracción de los puntos más relevantes de las imágenes se emplea el comando: extractFeatures, el mismo que compara las imágenes de la extracción de los bordes con la detección de puntos más importantes de la misma captura. La programación desarrollada se la muestra en la siguiente captura:

```
69      %% extraccion de las configuraciones cercanas de los obstaculos y camara central
70 -    [ob1,obpoints1]=extractFeatures(e_o1,reg1 );
71 -    [ob2,obpoints2]=extractFeatures(e_o2,reg2 );
72 -    [ob3,obpoints3]=extractFeatures(e_o3,reg3 );
73 -    [cam1,campoints1]=extractFeatures(e_cc,reg_cam1);
74 -    [cam2,campoints2]=extractFeatures(e_scc,reg_cam2);
75 -    [cam3,campoints3]=extractFeatures(e_ccc,reg_cam3);
```

Figura 27-3: Código fuente de la extracción de los puntos más relevantes.
Realizado por: Tinajero José, 2017.

3.3.6.9 Emparejamiento de las imágenes

El emparejamiento de las imágenes se las realiza en base a todos los procesos anteriormente descritos, las líneas de código empleadas se las muestra en la figura adjunta:

```
76      %% emparejamiento de configuraciones en las imagenes
77 -    indexPairs1=matchFeatures(ob1,cam1);
78 -    indexPairs2=matchFeatures(ob2,cam1);
79 -    indexPairs3=matchFeatures(ob3,cam1);
80 -    indexPairs4=matchFeatures(ob1,cam2);
81 -    indexPairs5=matchFeatures(ob2,cam2);
82 -    indexPairs6=matchFeatures(ob3,cam2);
83 -    indexPairs7=matchFeatures(ob1,cam1);
84 -    indexPairs8=matchFeatures(ob2,cam1);
85 -    indexPairs9=matchFeatures(ob3,cam1);
```

Figura 28-3: Código fuente del emparejamiento de imágenes.
Realizado por: Tinajero José, 2017.

Donde el comando emplea es matchFeatures que compara las extracciones de los puntos más relevantes.

3.3.6.10 Visualización de la correspondencia entre puntos

La visualización de la correspondencia de puntos entre las imágenes de la base de datos y la imagen que captura el robot en sus desplazamientos emplean el comando showMatchedFeatures, el método empleado con este comando es el "falsecolor" que tiene como característica sobreponer las imágenes que se analizan con diversos colores para que sean comparadas entre sí, uniendo los puntos que sean comunes a las dos imágenes. Las líneas de código empleadas son:

```
98      %% visualizacion de los correspondientes puntos
99 -    figure(12); showMatchedFeatures(e_o1,e_cc, matchedPoints1_1,matchedPoints1_2,'falsecolor');pause(2);
100 -   figure(13); showMatchedFeatures(e_o2,e_cc, matchedPoints5_1,matchedPoints5_2,'falsecolor'); pause(2);
101 -   figure(14); showMatchedFeatures(e_o3,e_cc, matchedPoints9_1,matchedPoints9_2,'falsecolor'); pause(2);
```

Figura 29-3: Código fuente de la visualización de la correspondencia de puntos.
Realizado por: Tinajero José, 2017.

Las imágenes resultantes son:

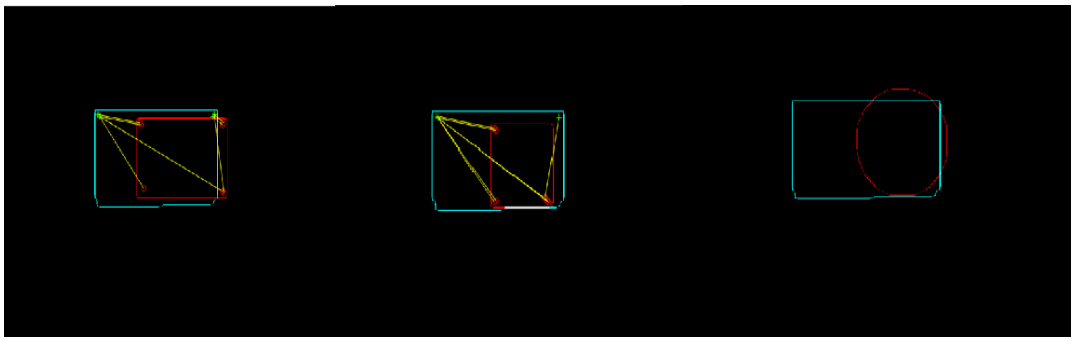


Figura 30-3: Visualización de la correspondencia entre puntos.
Realizado por: Tinajero José, 2017.

3.3.6.11 Correlación de imágenes

La correlación entre imágenes se la emplea para comparar los píxeles que poseen las dos capturas, e identificar si existe o no un obstáculo delante de la plataforma móvil diferencial, de lo cual

depende si el robot avance en línea recta o debe tomar una decisión de giro. El comando empleado para dicha operación es corr2.

```
103     % correlacio para verificar si hay un rectangulo
104 -   k1=corr2(e_o1, e_cc)
105     % correlacio para verificar si hay un cuadrado
106 -   k2=corr2(e_o2, e_scc)
107     % correlacio para verificar si hay una circunferencia
108 -   k3=corr2(e_o3, e_ccc)
109
110     % activo la senal si existe un obstaculo
111 -   if (((k1<=1)||(k1>-1)) && ((k2<=1)||(k2>-1)) && ((k3<=1)||(k3>-1)))
112 -       disp('EXISTE OBSTACULO')
113 -       writeDigitalPin(mypi,27,1)
114 -       pause(2)
115 -       writeDigitalPin(mypi,27,0)
116 -   else
117 -       disp('NO EXISTE OBSTACULO')
118 -       pause(3)
119 -   end
```

Figura 31-3: Código fuente de la correlación entre imágenes.
Realizado por: Tinajero José, 2017.

Los valores que devuelve dicha función están entre un rango de -1 y 1, donde los extremos son valores que indican alta correlación entre imágenes. De forma adicional cabe destacar que se despliega un mensaje indicando si el sistema de control visual encontró o no un obstáculo delante del robot, tal y como se muestra en la siguiente figura:

```
k1 =

    0.0017


k2 =

    0.1423


k3 =

    0.0038

EXISTE OBSTACULO
```

Figura 32-3: Valores de la correlación entre imágenes.
Realizado por: Tinajero José, 2017.

3.3.6.12 Reconstrucción del entorno

Para realizar la reconstrucción del entorno se considera la correspondencia de imágenes, ya que las capturas realizadas mediante la Raspberry Pi todas poseen las mismas dimensiones, pero a pesar de ello, se realiza un corte de las imágenes para asegurar que las dimensiones sean exactamente iguales. El código emplea es:

```
Editor - C:\Users\Usuario\Documents\MATLAB\2017a\recorte_recon\recprtes.m
programafinal2.m | motores_final.m | recon_1.m | recons.m
 3 -    I2=I1(50:584, 215:650,:);
 4 -    figure(2); imshow(I2); axis image
 5 -    imwrite(I2,'rec3.png')
 6 -    I3=imread('foto2.png');
 7 -    I4=I3(50:584, 210:480,:);
 8 -    figure(4); imshow(I4); axis image
 9 -    imwrite(I4,'rec2.png')
10 -    I5=imread('foto1.png');
11 -    I6=I5(50:584, 115:340,:);
12 -    figure(6); imshow(I6); axis image
13 -    imwrite(I6,'rec1.png')
14 -    I7=imread('foto4.png');
15 -    I8=I7(50:584, 360:640,:);
16 -    figure(8); imshow(I8); axis image
17 -    imwrite(I8,'rec4.png')
18 -    I9=imread('foto5.png');
19 -    I10=I9(50:584, 380:780,:);
20 -    figure(10); imshow(I10); axis image
21 -    imwrite(I10,'rec5.png')
22 -    z1=[I6 I4 I2 I8 I10];
23 -    figure(11); imshow(z1)
```

Figura 33-3: Valores de la correlación entre imágenes.
Realizado por: Tinajero José, 2017.

Se asignan las capturas a una nueva matriz, que da como resultado la figura siguiente:

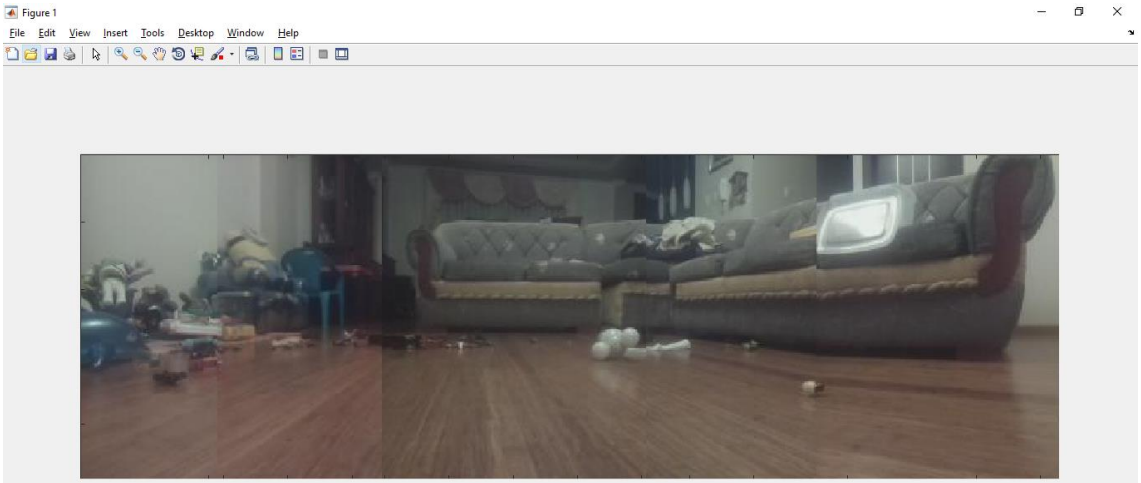


Figura 34-3: Valores de la correlación entre imágenes.
Realizado por: Tinajero José, 2017.

CAPÍTULO IV

4. RESULTADOS

Para la implementación del Sistema de Control Visual se empleó una tarjeta Raspberry Pi 3 con una cámara Raspberry y para la comprobación del sistema mediante fusión sensorial se utilizó la tarjeta Arduino Nano con sensores ultrasónicos. Para mayor detalle de la configuración y líneas de código empleadas referir al Apéndice A y B respectivamente.

Para lograr las características planteadas al inicio de la investigación fue necesario realizar varias operaciones de preprocesamiento de las imágenes, de forma que estas sean lo suficientemente idóneas para su análisis.

4.1 Validación de hipótesis

4.1.1 *Detalles del experimento*

Para el desarrollo del proyecto se platearon cuatro posibles escenarios, los mismos que se detallan a continuación:

Pista 1. Está conformada por dos obstáculos, y con predominancia de giro al lado derecho.

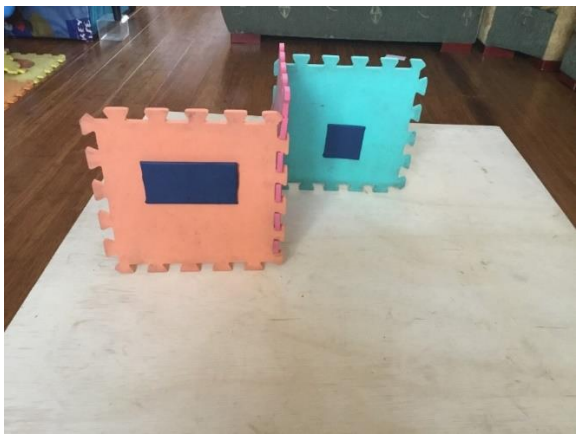


Figura 1-4: Pista de Pruebas N° 1
Realizado por: Tinajero José, 2017.

Tabla 1-4: Resultado de Pruebas de la Pista N°1.

PISTA 1					
N. Prueba	**# Obstáculos**	**# Obstáculos Evadidos**	**Eficiencia**	**Error**	**Observación**
1	2	2	100%	0%	-
2	2	2	100%	0%	-
3	2	2	100%	0%	-
4	2	1	50%	50%	Desborde de la matriz
5	2	2	100%	0%	-

Realizado por: Tinajero José, 2017.

Pista 2. Está conformada con dos obstáculos y con predominancia de giro hacia el lado izquierdo.

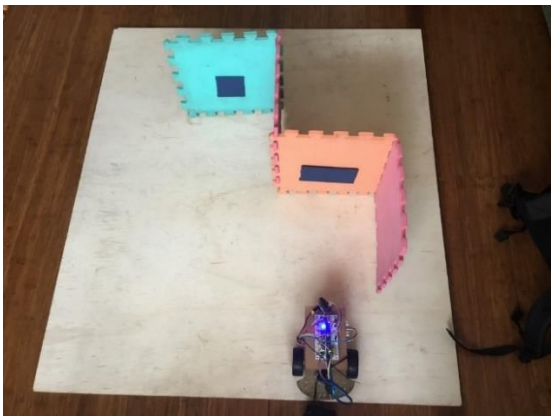


Figura 2-4: Pista de Pruebas N° 2.
Realizado por: Tinajero José, 2017.

Tabla 2-4: Resultado de Pruebas de la Pista N°2.

PISTA 2					
N. Prueba	**# Obstáculos**	**# Obstáculos Evadidos**	**Eficiencia**	**Error**	**Observación**
1	2	2	100%	0%	-
2	2	2	100%	0%	-
3	2	2	100%	0%	-
4	2	1	50%	50%	Desborde de la matriz
5	2	2	100%	0%	-

Realizado por: Tinajero José, 2017.

Pista 3. Es una pista hibrida, es decir combina giros en ambas direcciones, posee tres obstáculos la pista.

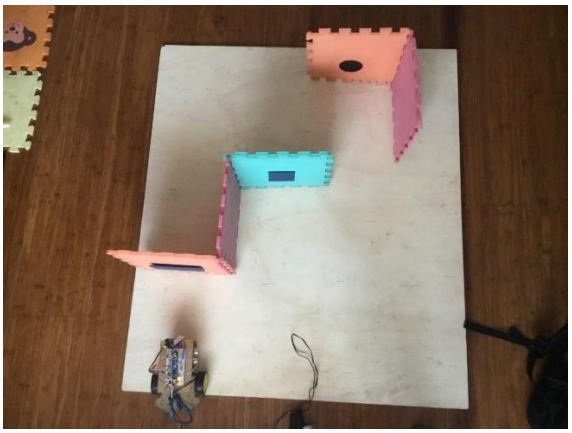


Figura 3-4: Pista de Pruebas N° 3.
Realizado por: Tinajero José, 2017.

Tabla 3-4: Resultado de Pruebas de la Pista N°3.

PISTA 3					
N. Prueba	**# Obstáculos**	**# Obstáculos Evadidos**	**Eficiencia**	**Error**	**Observación**
1	3	3	100%	0%	-
2	3	3	100%	0%	-
3	3	3	100%	0%	-
4	3	3	100%	0%	-
5	3	1	33%	67%	Desborde de la matriz

Realizado por: Tinajero José, 2017.

Pista 4. La pista posee 3 obstáculos, los mismos que están distribuidos de forma aleatoria para verificar el correcto desplazamiento de la plataforma móvil diferencial.

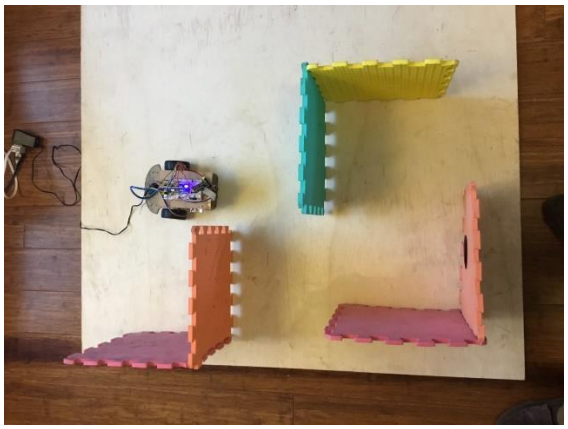


Figura 3-4: Pista de Pruebas N° 4.
Realizado por: Tinajero José, 2017.

Tabla 4-4: Resultado de Pruebas de la Pista N°4.

PISTA 4					
N. Prueba	**# Obstáculos**	**# Obstáculos Evadidos**	**Eficiencia**	**Error**	**Observación**
1	3	3	100%	0%	-
2	3	2	67%	33%	Desborde de la matriz
3	3	3	100%	0%	-
4	3	3	100%	0%	-
5	3	3	100%	0%	-

Realizado por: Tinajero José, 2017.

4.1.2 *Análisis de datos*

Pista 1.

Según los datos mostrados se evidencia un error en el sistema de control visual, ya que en la prueba 4 evade un solo obstáculo de los dos propuestos. El error se presenta debido al cambio de la luz ambiente del espacio físico donde se realizó las pruebas.

Pista 2.

De las 5 pruebas realizadas 4 de ellas se las desarrollo con total eficiencia del sistema, en una de ellas el sistema muestra un error debido a la calidad de iluminación que existe en el ambiente. El cambio de luz se produce por condiciones atmosféricas.

Pista 3.

El error hace que el sistema solo detecte uno de los tres obstáculos propuestos en la pista, es decir, el sistema de control visual no logró evadir todos los obstáculos propuestos.

Pista 4.

En la segunda prueba el sistema de control visual solo detecta dos de los tres obstáculos propuestos, el último obstáculo no pudo ser detectado debido al cambio en la iluminación ambiente lo que produce un error en la ejecución del programa.

4.1.3 *Prueba de hipótesis*

Para realizar la prueba de la Hipótesis se aplica la prueba estadística del Análisis de la Varianza a una cola(Montgomery, 2004), también denominada como ANOVA para lo cual es necesario plantear:

La hipótesis Nula: “El sistema de control visual utilizando fusión sensorial en una plataforma móvil diferencial no permite el correcto desplazamiento y mapeo para la reconstrucción de interiores”,

La hipótesis Alternativa: “El sistema de control visual utilizando fusión sensorial en una plataforma móvil diferencial permite el correcto desplazamiento y mapeo para la reconstrucción de interiores”

Para el desarrollo de la prueba estadística se considera que el nivel de significancia es del 5%, también se puede expresar que se ha tomado el 95% del nivel de confianza que son parámetros establecidos para dicha prueba.

La muestra considera para la prueba de la hipótesis son los datos obtenidos de 4 pistas diferentes con distinta cantidad de obstáculos y ubicación de los mismos, en las cuales se realizó 5 pruebas en cada una, obteniendo un total de 20 pruebas. Los datos obtenidos son:

Tabla 5-4: Datos para la prueba de hipótesis.

DATOS	Obstáculos Evadidos	Obstáculos No Evadidos	Eficiencia del sistema
2 obstáculos	18	2	8
3 obstáculos	27	3	8

Realizado por: Tinajero José, 2017.

Para la verificación de los datos se emplean una hoja de Cálculo Excel en donde se obtienen los siguientes valores:

Anova: Single Factor

SUMMARY

Groups	*Count*	*Sum*	*Average*	*Variance*
Column 1	2	45	22,5	40,5
Column 2	2	5	2,5	0,5
Column 3	2	16	8	0

ANOVA

urce of Variati	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Gro	427	2	213,5	15,62195	0,02593	9,552094
Within Group	41	3	13,66666667			
Total	468	5				

Figura 5-4: Aplicación del estadístico ANOVA en Excel.
Realizado por: Tinajero José, 2017.

Según los datos obtenidos basados a un nivel de significancia del 5%, en la aplicación del estadístico ANOVA se encuentra que el punto crítico tiene un valor de 9,55. El valor calculado de Fisher es de 15,62 mediante la comparación entre estos dos valores se evidencia que el valor de Fisher se encuentra fuera de la región de aceptación, es decir, el punto crítico es menor que el valor calculado, por lo tanto, se rechaza la hipótesis nula y se acepta de forma plena la hipótesis alternativa.

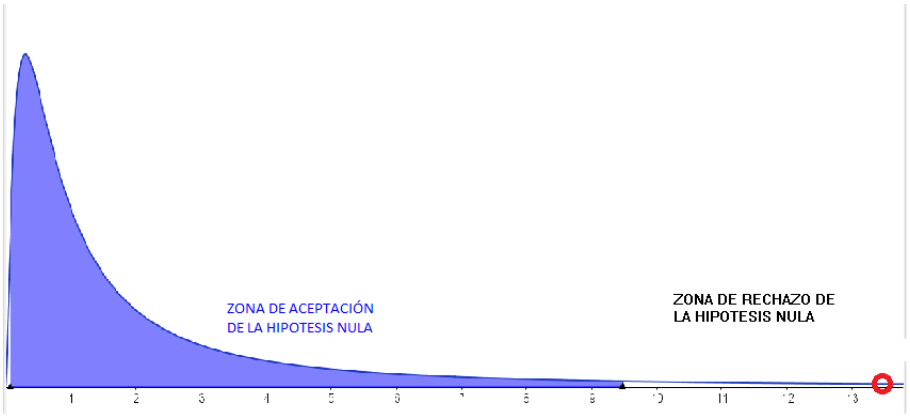


Figura 6-4: Distribución F. Análisis de varianza a una cola.
Realizado por: Tinajero José, 2017.

CONCLUSIONES

Se implementó una estructura móvil diferencial con una cámara Monoscópica Raspberry Pi NoIR Camera V2 conectada a la Raspberry Pi 3.

Se diseñó el sistema de control visual que permite adquirir una imagen panorámica del entorno sobre el cual se desplaza el robot, de forma adicional se realiza la detección de los obstáculos previamente cargados en la base de datos del sistema.

Se diseñó el algoritmo que permite la correcta toma de decisiones de acuerdo a distancia y presencia de objetos del cual se toma la mejor opción para el desplazamiento de la estructura móvil diferencial siempre considerando la protección de la misma, es decir, se evite al máximo los choques con los objetos existentes.

El sistema a pesar de tener un excelente desempeño en la detección y evasión de obstáculos genera un error debido a los cambios de luz producidos por condiciones externas al diseño y estructura.

Se logró obtener una secuencia de imágenes que dan una idea clara y precisa del entorno por el cual realizó el desplazamiento la estructura móvil diferencial sobre la cual se encuentra el sistema de control visual.

Según los datos obtenidos basados a un nivel de significancia del 5%, en la aplicación del estadístico ANOVA se encuentra que el punto crítico tiene un valor de 9,55. El valor calculado de Fisher es de 15,62 mediante la comparación entre estos dos valores se evidencia que el valor de Fisher se encuentra fuera de la región de aceptación, por lo tanto, se acepta de forma plena la hipótesis planteada.

RECOMENDACIONES

Se recomienda montar el sistema de control sobre otro tipo de estructuras móviles para analizar si esta modificación altera el funcionamiento del sistema de control visual para la detección de obstáculos.

Se debería dotar de inteligencia artificial al sistema para que el aprendizaje sea más amplio en la detección de objetos.

Resulta importante la búsqueda de una posibilidad que evite al máximo los errores en la ejecución del programa debido a cambios en la luz del entorno, para lo cual, se debería implementar un sistema de iluminación, que se ajuste de forma efectiva y eficiente a las características del sistema completo.

BIBLIOGRAFÍA

1. ADAFRUIT. (2017). Adafruit. [En línea]. [Consulta: 2017-05-07]. Disponible en: https://www.adafruit.com/product/3100

2. ARDUINO CC. (2017). Arduino. [En línea]. [Consulta: 2017-05-07]. Disponible en: https://www.arduino.cc/en/Guide/ArduinoNano

3. BASIC, A. (2017). Arduino Basic. [En línea]. [Consulta: 2017-05-07]. Disponible en: http://arduinobasics.blogspot.com/2012/11/hc-sr04-ultrasonic-sensor.html

4. C, S. (2017). Arduino Basic. [En línea]. [Consulta: 2017-05-07]. Disponible en: http://arduinobasics.blogspot.com/2012/11/hc-sr04-ultrasonic-sensor.html

5. GARCÍA SANTILLÁN, I. D. (2008). Visión Artifical y Procesamiento Digital de Imágenes usando Matlab, 103.

6. HOLLAND, J. (2004). Designing Autonomous Mobile Robots.

7. JIANG, N. (2016). Intelligent Stereo Camera Mobile Platform for Indoor Service Robot Research. Proceedings of the 2016 IEEE 20th International Conference on Computer Supported Cooperative Work in Desing, 5–9.

8. MILENA, A., LÓPEZ, L., ENRIQUE, J., & URIBE, A. (2016). Visual servo control law design using 2D vision approach , for a 3 DOF robotic system built with LEGO EV3 and a Raspberry Pi, (v).

9. MONTGOMERY, D. (2004). Diseño y Análisis de Experimentos. LIMUSA WILEY (Vol. 2). [En línea]. [Consulta: 2017-05-07]. Disponible en: http://doi.org/10.1016/S0141-3910(01)00005-2

10. OLLERO, A. (2001). Robotica Manipuladores y robots moviles. Marcombo S.A.

11. PI, R., BOKADE, A. U., RATNAPARKHE, V. R., & PI, A. R. (2016). Video Surveillance Robot Control using, 2094–2097.

12. RASPBERRY PI. (2017). Raspberry Pi. [En línea]. [Consulta: 2017-05-07]. Disponible en: https://www.raspberrypi.org/?s=raspberry+pi+3

13. SICILIANO, B., SCIAVICCO, L., VILLANI, L., & ORIOLO, G. (2009). Robotics Modelling, Planning and Control. [En línea]. [Consulta: 2017-05-07]. Disponible en: http://doi.org/10.1007/978-1-84628-642-1

14. TEXAS INSTRUMENTS INCORPORATED. (2016). L293x Quadruple Half-H Drivers. Texas Instruments Incorporated, 21. [En línea]. [Consulta: 2017-05-07] Disponible en: http://www.ti.com/lit/ds/symlink/l293.pdf

15. VANITHA, M., SELVALAKSHMI, M., & SELVARASU, R. (2016). Monitoring and controlling of mobile robot via internet through raspberry Pi board. 2016 2nd International Conference on Science Technology Engineering and Management, ICONSTEM 2016, 462–466. [En línea]. [Consulta: 2017-05-07]. Disponible en: http://doi.org/10.1109/ICONSTEM.2016.7560864

ANEXOS

ANEXO A

```
clear all; clc;
% recorte de las imagenes de la base de datos
re=imread('obst_11.png');
re_1=re(57:502, 240:660,:);
imwrite(re_1,'rectangulo.png')
cu=imread('obst_12.png');
cu_1=cu(57:502, 240:660,:);
imwrite(cu_1,'cuadrado.png')
ci=imread('obst_cir.png');
ci_1=ci(57:502, 240:660,:);
imwrite(ci_1,'circulo.png')
%% lectura de las imagenes desde la base de datos
o1=imread('rectangulo.png');
o2=imread('cuadrado.png');
o3=imread('circulo.png');
% descomposicion en matrices individuales
o1r=o1(:,:,1); o1g=o1(:,:,2); o1b=o1(:,:,3);
o2r=o2(:,:,1); o2g=o2(:,:,2); o2b=o2(:,:,3);
o3r=o3(:,:,1); o3g=o3(:,:,2); o3b=o3(:,:,3);
% se realiza la extraccion del color azul de cada obstaculo
jb_o1=o1b-(o1r/2)-(o1g/2);
jb_o2=o2b-(o2r/2)-(o2g/2);
jb_o3=o3b-(o3r/2)-(o3g/2);
%se crea una nueva imagen en la que se define el color azul
b_o1=jb_o1>10.5;
b_o2=jb_o2>10.5;
b_o3=jb_o3>10.5;
% se realiza la binarizacion de las imagenes que se extrajo el color azul
bi_o1=im2bw(b_o1,0.5); %figure; imshow(bi_o1);
bi_o2=im2bw(b_o2,0.5); %figure; imshow(bi_o2);
bi_o3=im2bw(b_o3,0.5); %figure; imshow(bi_o3);
%aplica operaciones sobre la imagen binarizada( erosion y dilatacion)
% se emplea strel con diferentes argumentos para recrear los obstaculos
%% se aplica las operaciones para el rectangulo
FC=[150;150];
SER=strel('rectangle',FC);
ero_o1=imerode(bi_o1,SER); %figure(1); imshow(ero_o1);
dil_o1=imdilate(ero_o1, SER); %figure(2); imshow(dil_o1);
e_o1=edge(dil_o1, 'canny'); %%figure(1); imshow(e_o1); % se filtra todos los bordes de la imagen
% el algoritmo canny detecta todo el borde de la figura debera ser la implementada

SES=strel('square',150);
ero_o2=imerode(bi_o2,SES); %figure(4); imshow(ero_o2);
dil_o2=imdilate(ero_o2,SES); %figure(5); imshow(dil_o2);
e_o2=edge(dil_o2, 'canny'); %%figure(2); imshow(e_o2);

SED=strel('disk',90, 8);
ero_o3=imerode(bi_o3,SED); %figure(7); imshow(ero_o3);
dil_o3=imdilate(ero_o3,SED);% figure(8); imshow(dil_o3);
e_o3=edge(dil_o3, 'canny'); %%figure(3); imshow(e_o3);

% se realiza la extraccion de los puntos mas relevantes de cada obstaculo ,
% imagen preprocesada
reg1 = detectSURFFeatures(e_o1); figure(1); imshow(e_o1); hold on; plot(reg1.selectStrongest(100));
reg2 = detectSURFFeatures(e_o2); figure(2); imshow(e_o2); hold on; plot(reg2.selectStrongest(100));
reg3 = detectSURFFeatures(e_o3); figure(3); imshow(e_o3); hold on; plot(reg3.selectStrongest(1000));
% hasta este punto el programa seejecuta una sola vez

%% se crea la conexion con la raspi
mypi = raspi;
% se crea el nombre de la camara y resolucion
mycam = cameraboard(mypi,'Resolution','320x240');
```

```
%se define el pin de la raspi a trabajar con el servo
s = servo(mypi,4);
%se configuran los los pines de la raspi para los motores y sensores
configurePin(mypi,13,'DigitalInput'); % activacion motores adelante
configurePin(mypi,5,'DigitalInput'); % activacion motores giro derecha
configurePin(mypi,23,'DigitalInput'); % activacion motores giro izquierda
configurePin(mypi,24,'DigitalInput'); % desactivacion de los motores por seguridad
configurePin(mypi,17,'DigitalInput'); % activacion ciclo repetitivo
configurePin(mypi,20,'DigitalOutput'); % rueda derecha
configurePin(mypi,21,'DigitalOutput'); % rueda derecha
configurePin(mypi,19,'DigitalOutput'); % rueda izquierda
configurePin(mypi,26,'DigitalOutput'); % rueda izquierda
configurePin(mypi,6,'DigitalOutput'); % señal desactivacion secuencia
configurePin(mypi,27,'DigitalOutput'); % señal indicativa de obstaculo

ini=readDigitalPin(mypi,17);
while ini
% foto posicion izquierda
writePosition(s,180);
pause(0.5)
 for ii=0:4
    fot1=snapshot(mycam);
    image(fot1);
    figure(4);
 end
print('-f4','foto1','-dpng')
pause(0.2)

%foto posicion rotada 135 grados
writePosition(s,135);
pause(.2)
for ii=0:4
    fot2=snapshot(mycam);
    image(fot2);
    figure(5);
 end
print('-f5','foto2','-dpng')
pause(0.2)

% foto frontal del movil
writePosition(s,90);
pause(.2)
for ii=0:4
    fot3=snapshot(mycam);
    image(fot3);
    figure(6);
end
print('-f6','foto3','-dpng')
pause(0.2)

% foto posicion der rotada 45
writePosition(s,45);
pause(.2)
for ii=0:4
    fot4=snapshot(mycam);
    image(fot4);
    figure(7);
 end
 print('-f7','foto4','-dpng')
 pause(.2)

% foto posicion izquierda completa
writePosition(s,0);
pause(.2)
for ii=0:4
   fot5=snapshot(mycam);
   image(fot5);
```

```
    figure(8)
end
print('-f8','foto5','-dpng')
pause(.2)
% se realiza el prooceso de carga y emparejamiento de las respectivas
% imagenes para ladeteccion de los obstaculos

%% se aplica los procedimientos a las imagenes capturadas por la raspberry
% se recorta la imagen para evitar el desborde de la matriz
f3=imread('foto3.png');
f3_1=f3(57:502, 240:660,:);
imwrite(f3_1,'central.png')
c_cam=imread('central.png'); %cargamos la imagen de la posicion central
cr=c_cam(:,:,1); cg=c_cam(:,:,2); cb=c_cam(:,:,3);
jb_cc=cb-(cr/2)-(cg/2);
b_cc=jb_cc>10.5; %figure; imshow(b_o1);
i_cc=im2single(b_cc);
bi_cc=imbinarize(i_cc,'adaptive','ForegroundPolarity','dark','Sensitivity',0.2);
%bi_cc=im2bw(b_cc,0.8); %figure(13); imshow(bi_cc);

see=strel('rectangle',FC);
ero_s=imerode(bi_cc,see); %figure(14); imshow(ero_s);
dil_s=imdilate(ero_s,see); %figure(15); imshow(dil_s);
e_cc=edge(dil_s,'canny');%figure(16); imshow(e_cc);

ero_scc=imerode(bi_cc,SES); %figure(17); imshow(ero_scc);
dil_scc=imdilate(ero_scc,SES); %figure(18); imshow(dil_scc);
e_scc=edge(dil_scc, 'canny'); %figure(19); imshow(e_scc);

ero_ccc=imerode(bi_cc,SED); %figure(20); imshow(ero_ccc);
dil_ccc=imdilate(ero_ccc,SED);% figure(21); imshow(dil_ccc);
e_ccc=edge(dil_ccc, 'canny'); %figure(22); imshow(e_ccc);
reg_cam1=detectSURFFeatures(e_cc); figure(9); imshow(e_cc); hold on; plot(reg_cam1.selectStrongest(10000));
reg_cam2=detectSURFFeatures(e_scc); figure(10); imshow(e_scc); hold on; plot(reg_cam2.selectStrongest(10000));
reg_cam3=detectSURFFeatures(e_ccc); figure(11); imshow(e_ccc); hold on;
plot(reg_cam3.selectStrongest(2000000000));

%% extraccion de las configuraciones cercanas de los obstaculos y camara central
[ob1,obpoints1]=extractFeatures(e_o1,reg1 );
[ob2,obpoints2]=extractFeatures(e_o2,reg2 );
[ob3,obpoints3]=extractFeatures(e_o3,reg3 );
[cam1,campoints1]=extractFeatures(e_cc,reg_cam1);
[cam2,campoints2]=extractFeatures(e_scc,reg_cam2);
[cam3,campoints3]=extractFeatures(e_ccc,reg_cam3);
%% emparejamiento de configuraciones en las imagenes
indexPairs1=matchFeatures(ob1,cam1);
indexPairs2=matchFeatures(ob2,cam1);
indexPairs3=matchFeatures(ob3,cam1);

indexPairs4=matchFeatures(ob1,cam2);
indexPairs5=matchFeatures(ob2,cam2);
indexPairs6=matchFeatures(ob3,cam2);

indexPairs7=matchFeatures(ob1,cam1);
indexPairs8=matchFeatures(ob2,cam1);
indexPairs9=matchFeatures(ob3,cam1);

%% recuperacion de los puntos correspondientes en cada imagen
%emparejamiento obstaculo 1 camara central
matchedPoints1_1=obpoints1(indexPairs1(:,1),:);
matchedPoints1_2=campoints1(indexPairs1(:,2),:);
%emparejamiento obstaculo 2 camara central
%%matchedPoints2_1=obpoints2(indexPairs2(:,1),:);
%%matchedPoints2_2=campoints1(indexPairs2(:,2),:);
%emparejamiento obstaculo 3 camara central
%%matchedPoints3_1=obpoints3(indexPairs3(:,1),:);
%%matchedPoints3_2=campoints1(indexPairs3(:,2),:);
```

```
%emparejamiento obstaculo 1 camara central
%%matchedPoints4_1=obpoints1(indexPairs1(:,1),:);
%%matchedPoints4_2=campoints2(indexPairs1(:,2),:);
%emparejamiento obstaculo 2 camara central
matchedPoints5_1=obpoints2(indexPairs2(:,1),:);
matchedPoints5_2=campoints2(indexPairs2(:,2),:);
%emparejamiento obstaculo 3 camara central
%%matchedPoints6_1=obpoints3(indexPairs3(:,1),:);
%%matchedPoints6_2=campoints2(indexPairs3(:,2),:);

%emparejamiento obstaculo 1 camara central
%%matchedPoints7_1=obpoints1(indexPairs1(:,1),:);
%%matchedPoints7_2=campoints3(indexPairs1(:,2),:);
%emparejamiento obstaculo 2 camara central
%%matchedPoints8_1=obpoints2(indexPairs2(:,1),:);
%%matchedPoints8_2=campoints3(indexPairs2(:,2),:);
%emparejamiento obstaculo 3 camara central
matchedPoints9_1=obpoints3(indexPairs3(:,1),:);
matchedPoints9_2=campoints3(indexPairs3(:,2),:);

%% visualizacion de los correspondientes puntos
figure(12); showMatchedFeatures(e_o1,e_cc, matchedPoints1_1,matchedPoints1_2,'falsecolor');pause(2);
%figure(11); showMatchedFeatures(e_o2,e_cc, matchedPoints2_1,matchedPoints2_2,'blend');
%figure(12); showMatchedFeatures(e_o3,e_cc, matchedPoints3_1,matchedPoints3_2,'blend');

%figure(29); showMatchedFeatures(e_o1,e_scc, matchedPoints4_1,matchedPoints4_2,'blend');
figure(13); showMatchedFeatures(e_o2,e_cc, matchedPoints5_1,matchedPoints5_2,'falsecolor'); pause(2);
%figure(31); showMatchedFeatures(e_o3,e_scc, matchedPoints6_1,matchedPoints6_2,'blend');

%figure(32); showMatchedFeatures(e_o1,e_cc, matchedPoints7_1,matchedPoints7_2,'blend');
%figure(33); showMatchedFeatures(e_o2,e_cc, matchedPoints8_1,matchedPoints8_2,'blend');
figure(14); showMatchedFeatures(e_o3,e_cc, matchedPoints9_1,matchedPoints9_2,'falsecolor'); pause(2);

% se encuentra la correlación entre las imageneshelp
% correlacio para verificar si hay un rectangulo
k1=corr2(e_o1, e_cc)
% correlacio para verificar si hay un cuadrado
k2=corr2(e_o2, e_scc)
% correlacio para verificar si hay una circunferencia
k3=corr2(e_o3, e_ccc)

% activo la senal si existe un obstaculo
if (((k1<=1)||(k1>-1)) && ((k2<=1)||(k2>-1)) && ((k3<=1)||(k3>-1)))
   disp('EXISTE OBSTACULO')
   writeDigitalPin(mypi,27,1)
   pause(2)
   writeDigitalPin(mypi,27,0)
else
   disp('NO EXISTE OBSTACULO')
   pause(3)
end
% se analiza las distancias de los ultrasonicos
front=readDigitalPin(mypi,13);
right=readDigitalPin(mypi,5);
left=readDigitalPin(mypi,23);
top=readDigitalPin(mypi,24);

   if front
      % avance rueda derecha configuracion pin 20-0 pin 21-1
      writeDigitalPin(mypi,20,0)
      writeDigitalPin(mypi,21,1)
      % avance rueda izquierda configuracion pin 19-1 pin 26-0
      writeDigitalPin(mypi,19,1)
      writeDigitalPin(mypi,26,0)
      pause(.75)
      writeDigitalPin(mypi,20,0)
```

```
    writeDigitalPin(mypi,21,0)
    writeDigitalPin(mypi,19,0)
    writeDigitalPin(mypi,26,0)
    pause(0.5)
    writeDigitalPin(mypi,6,1)
    pause(.5)
    writeDigitalPin(mypi,6,0)
end

    if right
    %LLANTA IZQUIERDA GIRA HACIA DELANTE
    writeDigitalPin(mypi,19,1)
    writeDigitalPin(mypi,26,0)
    % LLANTA DERCHA GIRA HACIA ATRAS
    writeDigitalPin(mypi,20,1)
    writeDigitalPin(mypi,21,0)
    pause(0.475)
    %se completa el giro y se detiene los motores
    writeDigitalPin(mypi,20,0)
    writeDigitalPin(mypi,21,0)
    writeDigitalPin(mypi,19,0)
    writeDigitalPin(mypi,26,0)
    pause(.15)
    %avanza hacia delante
    writeDigitalPin(mypi,20,0)
    writeDigitalPin(mypi,21,1)
    writeDigitalPin(mypi,19,1)
    writeDigitalPin(mypi,26,0)
    pause(1.)
    %se detienen las llantas
    writeDigitalPin(mypi,20,0)
    writeDigitalPin(mypi,21,0)
    writeDigitalPin(mypi,19,0)
    writeDigitalPin(mypi,26,0)
    pause(.2)
    %se realiza un nuevo giro para posicionar al robot de nuevo
    %LLANTA IZQUIERDA GIRA HACIA ATRAS
    writeDigitalPin(mypi,19,0)
    writeDigitalPin(mypi,26,1)
    % LLANTA DERCHA GIRA HACIA DELANTE
    writeDigitalPin(mypi,20,0)
    writeDigitalPin(mypi,21,1)
    pause(.475)
    writeDigitalPin(mypi,20,0)
    writeDigitalPin(mypi,21,0)
    writeDigitalPin(mypi,19,0)
    writeDigitalPin(mypi,26,0)
    pause(0.25)
    %indica que el giro termino
    writeDigitalPin(mypi,6,1)
    pause(.5)
    writeDigitalPin(mypi,6,0)
    end

    if left
    %se realiza el giro hacia el lado izquierdo
    %LLANTA IZQUIERDA GIRA HACIA ATRAS
    writeDigitalPin(mypi,19,0)
    writeDigitalPin(mypi,26,1)
    % LLANTA DERCHA GIRA HACIA DELANTE
    writeDigitalPin(mypi,20,0)
    writeDigitalPin(mypi,21,1)
    pause(.475)
    %se detiene los motores por un momento
    writeDigitalPin(mypi,20,0)
    writeDigitalPin(mypi,21,0)
    writeDigitalPin(mypi,19,0)
```

```
writeDigitalPin(mypi,26,0)
pause(0.15)
% avanza el robot hacia delante
writeDigitalPin(mypi,20,0)
writeDigitalPin(mypi,21,1)
writeDigitalPin(mypi,19,1)
writeDigitalPin(mypi,26,0)
pause(1.)
%se detienen los motores
writeDigitalPin(mypi,20,0)
writeDigitalPin(mypi,21,0)
writeDigitalPin(mypi,19,0)
writeDigitalPin(mypi,26,0)
pause(.2)
%realiza el giro hacia el lado derecho para posicionar la bot en
%posicion de partida
%LLANTA IZQUIERDA GIRA HACIA DELANTE
writeDigitalPin(mypi,19,1)
writeDigitalPin(mypi,26,0)
% LLANTA DERCHA GIRA HACIA ATRAS
writeDigitalPin(mypi,20,1)
writeDigitalPin(mypi,21,0)
pause(0.475)
%se completa el giro y se detiene el giro
writeDigitalPin(mypi,20,0)
writeDigitalPin(mypi,21,0)
writeDigitalPin(mypi,19,0)
writeDigitalPin(mypi,26,0)
pause(0.2)
% se indica que el giro se completo
writeDigitalPin(mypi,6,1)
pause(.5)
writeDigitalPin(mypi,6,0)
end

if top
% se detienen los motores alarma encendida
writeDigitalPin(mypi,20,0)
writeDigitalPin(mypi,21,0)
writeDigitalPin(mypi,19,0)
writeDigitalPin(mypi,26,0)
pause(.25)
%GIRA 180 GRADOS
writeDigitalPin(mypi,19,0)
writeDigitalPin(mypi,26,1)
writeDigitalPin(mypi,20,0)
writeDigitalPin(mypi,21,1)
pause(.950)
%se detienen los motores
writeDigitalPin(mypi,20,0)
writeDigitalPin(mypi,21,0)
writeDigitalPin(mypi,19,0)
writeDigitalPin(mypi,26,0)
pause(0.10)
%avanza hacia delante el bot
% avance rueda derecha configuracion pin 20-0 pin 21-1
writeDigitalPin(mypi,20,0)
writeDigitalPin(mypi,21,1)
% avance rueda izquierda configuracion pin 19-1 pin 26-0
writeDigitalPin(mypi,19,1)
writeDigitalPin(mypi,26,0)
pause(1.)
%se detiene los motores
writeDigitalPin(mypi,20,0)
writeDigitalPin(mypi,21,0)
writeDigitalPin(mypi,19,0)
writeDigitalPin(mypi,26,0)
```

```
        pause(.10)
        %se realiza el giro al lado derecho
        %GIRO A LA DERECHA
        writeDigitalPin(mypi,19,1)
        writeDigitalPin(mypi,26,0)
        % LLANTA DERCHA GIRA HACIA ATRAS
        writeDigitalPin(mypi,20,1)
        writeDigitalPin(mypi,21,0)
        pause(0.475)
        %se detienen los motores
        writeDigitalPin(mypi,20,0)
        writeDigitalPin(mypi,21,0)
        writeDigitalPin(mypi,19,0)
        writeDigitalPin(mypi,26,0)
        pause(.10)
        %AVANZA EL BICHO
        writeDigitalPin(mypi,19,1)
        writeDigitalPin(mypi,26,0)
        % LLANTA DERCHA GIRA HACIA ATRAS
        writeDigitalPin(mypi,20,0)
        writeDigitalPin(mypi,21,1)
        pause(1.0)
        %se detiene los motores antes de girar a la derecha
        writeDigitalPin(mypi,20,0)
        writeDigitalPin(mypi,21,0)
        writeDigitalPin(mypi,19,0)
        writeDigitalPin(mypi,26,0)
        pause(.10)
        %se realiza el giro hacia la derecha
        %GIRO A LA DERECHA
        writeDigitalPin(mypi,19,1)
        writeDigitalPin(mypi,26,0)
        % LLANTA DERCHA GIRA HACIA ATRAS
        writeDigitalPin(mypi,20,1)
        writeDigitalPin(mypi,21,0)
        pause(0.475)
        writeDigitalPin(mypi,20,0)
        writeDigitalPin(mypi,21,0)
        writeDigitalPin(mypi,19,0)
        writeDigitalPin(mypi,26,0)
        pause(.25)

        %indica que el giro se completo
        writeDigitalPin(mypi,6,1)
        pause(.5)
        writeDigitalPin(mypi,6,0)
      end

I1=imread('foto3.png');
%figure(1);image(I1)
I2=I1(50:584, 215:650,:);
figure(2); imshow(I2); axis image
imwrite(I2,'rec3.png')
I3=imread('foto2.png');
I4=I3(50:584, 210:480,:);
figure(4); imshow(I4); axis image
imwrite(I4,'rec2.png')
I5=imread('foto1.png');
I6=I5(50:584, 115:340,:);
figure(6); imshow(I6); axis image
imwrite(I6,'rec1.png')
I7=imread('foto4.png');
I8=I7(50:584, 360:640,:);
figure(8); imshow(I8); axis image
imwrite(I8,'rec4.png')
I9=imread('foto5.png');
I10=I9(50:584, 380:780,:);
```

```
figure(10); imshow(I10); axis image
imwrite(I10,'rec5.png')
z1=[I6 I4 I2 I8 I10];
figure(11); imshow(z1)


end
```

ANEXO B

```
# include <Ultrasonic.h>

  //declaracion de los pines de salida

  const int ahead=2; // avance

  const int right=3; //derecha

  const int left=4;  //izquierda

  const int alarm=5; //detiene

  const int lect=12; //senal para desactivar el mando

  int sign=0;


  //declaracion pines ultrasonico

  Ultrasonic head(7,8); //trig echo sensor frontal

  Ultrasonic side(10,11); // trig echo sensor lateral derecho


  // valores para comparacion

  const int bloq=8; //distancia minima del area de trabajo

  const int obst=25; // distancia a la que se encuentra el obstaculo

  const int mayor=300; //distancia maxima del area de trabajo



void setup() {

  // put your setup code here, to run once:

  pinMode(ahead,OUTPUT);

  pinMode(right,OUTPUT);

  pinMode(left, OUTPUT);

  pinMode(alarm,OUTPUT);

  pinMode(lect,INPUT);

  //Serial.begin(19200);

}


void loop() {

  // put your main code here, to run repeatedly:
```

```
int s1=(head.distanceRead());
int s2=(side.distanceRead());
delay(1000);

digitalWrite(ahead,LOW);
digitalWrite(right,LOW);
digitalWrite(left,LOW);
digitalWrite(alarm,LOW);

// se empieza la comparacion de las distancias y toma de decisiones
if((s1<=bloq)&&(s2<=bloq)){
 digitalWrite(ahead,LOW);
 digitalWrite(right,LOW);
 digitalWrite(left,LOW);
 digitalWrite(alarm,HIGH);
 //se desactiva la senal del left cuando cumple el giro
 sign=digitalRead(lect);
 if(sign==HIGH){
  digitalWrite(ahead,LOW);
  digitalWrite(right,LOW);
  digitalWrite(left,LOW);
  digitalWrite(alarm,LOW);
   }
   else{
    digitalWrite(ahead,LOW);
    digitalWrite(right,LOW);
    digitalWrite(left,LOW);
    digitalWrite(alarm,HIGH);
    }
 }
 else{
  if((s1<=bloq)&&(s2>bloq)){
   //la comparacion indica que debe girar a la derecha
    digitalWrite(ahead,LOW);
    digitalWrite(right,HIGH);
    digitalWrite(left,LOW);
```

```
digitalWrite(alarm,LOW);
//se desactiva la senal del right cuando cumple el giro
sign=digitalRead(lect);
if(sign==HIGH){
  digitalWrite(ahead,LOW);
  digitalWrite(right,LOW);
  digitalWrite(left,LOW);
  digitalWrite(alarm,LOW);
  }
  else{
    digitalWrite(ahead,LOW);
    digitalWrite(right,HIGH);
    digitalWrite(left,LOW);
    digitalWrite(alarm,LOW);
    }
}
else{
  if((s1<=obst)&&(s1>bloq)){
    if(s2>bloq){
      //la comparacion indica que debe girar a la derecha
      digitalWrite(ahead,LOW);
      digitalWrite(right,HIGH);
      digitalWrite(left,LOW);
      digitalWrite(alarm,LOW);
      //se desactiva la senal del right cuando cumple el giro
      sign=digitalRead(lect);
      if(sign==HIGH){
        digitalWrite(ahead,LOW);
        digitalWrite(right,LOW);
        digitalWrite(left,LOW);
        digitalWrite(alarm,LOW);
        }
        else{
          digitalWrite(ahead,LOW);
          digitalWrite(right,HIGH);
          digitalWrite(left,LOW);
```

```
      digitalWrite(alarm,LOW);
      }
   }
  else{
    //la comparacion indica que debe girar a la izquierda
    digitalWrite(ahead,LOW);
    digitalWrite(right,LOW);
    digitalWrite(left,HIGH);
    digitalWrite(alarm,LOW);
    //se desactiva la senal del left cuando cumple el giro
    sign=digitalRead(lect);
    if(sign==HIGH){
      digitalWrite(ahead,LOW);
      digitalWrite(right,LOW);
      digitalWrite(left,LOW);
      digitalWrite(alarm,LOW);
      }
      else{
        digitalWrite(ahead,LOW);
        digitalWrite(right,LOW);
        digitalWrite(left,HIGH);
        digitalWrite(alarm,LOW);
        }
    }
}
else{
  if((s1<=mayor)&&(s1>obst)){
    //la comparacion indica que debe avanzar
    digitalWrite(ahead,HIGH);
    digitalWrite(right,LOW);
    digitalWrite(left,LOW);
    digitalWrite(alarm,LOW);
    sign=digitalRead(lect);
    if(sign==HIGH){
      digitalWrite(ahead,LOW);
      digitalWrite(right,LOW);
```

```
            digitalWrite(left,LOW);
            digitalWrite(alarm,LOW);
            }
            else{
              digitalWrite(ahead,HIGH);
              digitalWrite(right,LOW);
              digitalWrite(left,LOW);
              digitalWrite(alarm,LOW);
              }
           }
           else{
            digitalWrite(ahead,LOW);
            digitalWrite(right,LOW);
            digitalWrite(left,LOW);
            digitalWrite(alarm,HIGH);
            sign=digitalRead(lect);
            if(sign=HIGH){
              digitalWrite(ahead,LOW);
              digitalWrite(right,LOW);
              digitalWrite(left,LOW);
              digitalWrite(alarm,LOW);
              }
              else{
                digitalWrite(ahead,LOW);
                digitalWrite(right,LOW);
                digitalWrite(left,LOW);
                digitalWrite(alarm,HIGH);
                }
              }
            }
          }
        }
    }
```