DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA LÁSER PARA DETECCIÓN DE OBSTÁCULOS EN APLICACIONES DE ROBÓTICA MÓVIL

PAULA ANDREA MUÑOZ LÓPEZ

UNIVERSIDAD DEL CAUCA
FACULTAD DE CIENCIAS NATURALES, EXACTAS Y DE LA EDUCACIÓN
DEPARTAMENTO DE FÍSICA
POPAYÁN
2011

DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA LÁSER PARA DETECCIÓN DE OBSTÁCULOS EN APLICACIONES DE ROBÓTICA MÓVIL

PAULA ANDREA MUÑOZ LÓPEZ

Trabajo de investigación para optar al título de
Ingeniera Física

Director
Mg. Juan Fernando Florez Marulanda

UNIVERSIDAD DEL CAUCA
FACULTAD DE CIENCIAS NATURALES, EXACTAS Y DE LA EDUCACIÓN
INGENIERÍA FÍSICA
POPAYÁN – CAUCA
2011

Nota de aceptación

_____
_____
_____
_____
_____

Director:_____
Mg. JUAN FERNANDO FLOREZ MARULANDA

Jurado:_____
Mg. CARLOS FELIPE ORDOÑEZ

Jurado:_____
Mg. LUIS FERNANDO ECHEVERRI

Fecha de sustentación: Popayán, 28 de Noviembre de 2011

DEDICATORIA

A Dios porque se ha manifestado en mi diario vivir, dándome grandes enseñanzas que me ayudan a crecer como persona y por colocar en mi camino personas tan maravillosas que llenan de sonrisas mis días.

A mi padre Agustin Muñoz Martinez y mi madre Martha Lucia López Toro porque me han enseñado grandes valores y han sido el impulso más grande que he tenido para llegar a donde estoy ahora y ser lo que soy.

A mis hermanos Gerardo Andres y Luis Fernando, a mi abuelita Blanca Ruby y mi tía Stella López, porque son personas que siempre han estado pendientes de mis aciertos y desaciertos, brindándome una mano amiga para salir adelante.

AGRADECIMIENTOS

A Dios porque día a día está presente en mi vida brindándome grandes enseñanzas y colocando en mi camino personas especiales que me ayudan a crecer como persona y como profesional. Por haberme regalado los padres tan maravillosos que me dio.

A mis padres porque sin el apoyo incondicional de ellos no habría sido posible estar donde estoy ahora y de los cuales me siento muy orgullosa.

A mi familia porque día a día están aportando en mi vida en especial a mis hermanos Gerardo y Fernando, mi abuela Blanca Ruby Toro y mi tía Luz Stella López.

A las personas que ya no se encuentran conmigo pero desde donde se encuentran sé que me envían muchas bendiciones y disfrutan de este logro, en especial a mi abuelito Alcibíades López y a mis tíos Diego Velasco y Juan Carlos García.

A mi director de tesis, el Ingeniero Juan Fernando Flórez Marulanda por su paciencia y aportes para el desarrollo de este trabajo de grado.

Al profesor Luis Fernando Echeverry gracias por ser más que un profesor un maestro en todo el sentido de la palabra, por su amistad y su apoyo incondicional en el transcurso de toda mi carrera.

A los evaluadores de mi trabajo de grado el profesor Luis Fernando Echeverri y al Ingeniero Físico Carlos Felipe Ordoñez por el tiempo que dedicaron a la evaluación de esta tesis.

A los docentes en general del departamento de Ingeniería Física por los conocimientos aportados en el transcurso de mi carrera.

A mis compañeros y amigos que estuvieron en el transcurso de mi carrera en especial a Claudia Bedoya, Claudia López, Karime Gómez, Jhon Jairo Flórez, Julian Agudelo, Angelo Reyes, Juan Carlos Molina, Cristian Villacrez, Oscar García, Paola Andrade, Andrés Bolaños, Sandra López, Freddy Villamizar, Dairo Alegría, Cristian Gálvez, Diana Chantre, Daniel Usama, Jorge Posada, Lina Jaller, David Revelo, Ximena Velasco y Nohora Lucia Urbano.

A Diana Rocío Chantre, Sandra Carolina Imbachí, Nohora Lucia Urbano, David Revelo, José Vivas, Jeison Vivas, Ximena Velasco y Cristian Gálvez porque en el desarrollo de este trabajo de grado fueron de gran ayuda con sus grandes aportes.

A Mariana Usama Imbachí por prestarme a su mamita para que me colaborara y por su compañía en los largos sábados de pruebas.

A mis amigos en general por su amistad y compañía incondicional, en especial a Paola Andrea Posada López que desde la infancia me ha acompañado en todos mis aciertos y desaciertos.

A la empresa UNISOFT COLOMBIA LTDA. y todo su equipo de trabajo porque me dio y me sigue dando la oportunidad de crecer como profesional.

RESUMEN

En las aplicaciones de navegación tanto autónoma como teleoperada de un robot móvil, se requiere que cuente con un sistema de detección de obstáculos. La característica principal de estos sistemas es generar información útil para el sistema de navegación autónoma del vehículo y también hacer de sistema de seguridad para preservar la integridad de la plataforma. Estos sistemas normalmente se apoyan en diferentes tecnologías de detección de objetos como ultrasonido, infrarrojo y láser. En la actualidad los sistemas de detección de obstáculos se apoyan en una fusión sensorial de todos los sensores disponibles, de los cuales el sistema radar láser es el más valioso.

En este trabajo de grado se diseñó, implementó y validó un sistema láser para detección de obstáculos en aplicaciones de robótica móvil, haciendo uso del *Safe Zone Scanner Laser* de la marca *Allen Bradley* de la empresa *Rockwell Automation*.

Inicialmente se realizó la deducción de las tramas del protocolo de comunicación entre el escáner láser y el computador, a través de la realización de una serie de experimentos que permitieron finalmente obtener dichas tramas de datos.

Se diseñó e implementó un sistema láser para detección de obstáculos teniendo en cuenta la información obtenida en la deducción de las tramas de datos y los requerimientos generales del sistema y finalmente se validó el sistema de detección de obstáculos, tanto en interiores como en exteriores, con el fin de comprobar el nivel de confiabilidad del mismo en comparación con el software propio del escáner láser.

TABLA DE CONTENIDO

LISTA DE FIGURAS

LISTA DE TABLAS

INTRODUCCIÓN

Nuevas e importantes exploraciones científicas se están llevando a cabo en ambientes adversos, tanto dentro del planeta como en el espacio exterior, con la asistencia de robots móviles. Algunos teleoperados manualmente y otros guiados autónomamente por plataformas software de robótica móvil apoyados en sistemas exterosensores. Los robots autónomos necesitan de sistemas exterosensores que les permitan detectar objetos por si solos, tanto para proteger su integridad como para poder crear trayectorias a seguir según sus sistemas de navegación.

Entre las tecnologías más utilizadas en la detección de objetos para aplicaciones de robótica móvil se encuentran sistemas mecánicos, infrarrojos, ultrasónicos y láser. Estos se integran para poder reunir información en diferentes anillos de proximidad alrededor del vehículo. Los sistemas mecánicos para detección de cuerpos en un anillo de proximidad cercano o en contacto, sistemas infrarrojos para detección en un anillo de proximidad medio de 1 a 3 metros, sistemas ultrasónicos y/o de radar láser para detección en un anillo de proximidad lejano de 3 hasta 50 metros. Dentro de las tecnologías mencionadas para detección de objetos, el sensor de rango láser, más conocido como LIDAR, es el sistema más empleado en este tipo de aplicaciones.

La implementación del sistema láser para la detección de obstáculos en aplicaciones de robótica móvil, permitirá crear mapas de cualquier lugar en el que se encuentre tanto en interiores como en exteriores sin importar las condiciones de luz del ambiente, permitiendo actualizar los datos en tiempos cortos del orden de los milisegundos (ms). Con esta información entregada por el sistema láser y con la ayuda de otros sensores el robot móvil estará en la capacidad de funcionar como un sistema autónomo.

El objetivo principal de este trabajo de grado es diseñar, implementar y validar un sistema láser para detección de obstáculos apoyado en el *Safe Zone Scanner Laser de Allen Bradley* y una plataforma software de robótica móvil.

En la primer capítulo se realiza un introducción teórica de la robótica móvil (definición, uso, autonomía, problemas, software y hardware), el problema de detección de obstáculos (definición, enfoques, sensores usados, limitaciones de los sensores y el escáner láser en la detección de obstáculos) y finalmente una descripción del escáner láser usado en este proyecto (definición, funcionalidad, escáner láser *Allen Bradley* y algunas aplicaciones).

En el segundo capítulo se realiza una recopilación de las principales plataformas software de robótica móvil de fuente abierta, se hace un análisis de cada una de ellas y finalmente se selecciona la plataforma software de robótica móvil que se usará para la implementación del sistema software de detección de obstáculos de este trabajo de grado.

En el tercer capítulo se realiza la deducción de las tramas de datos entre el escáner láser y el computador. Para esto se ilustra brevemente la forma como el escáner láser establece conexión con el computador y a través del uso del *Safety Configuration & Diagnostic* (SCD), se implementan una serie de experimentos que permiten la identificación de las tramas de datos entre el escáner láser y el computador, donde finalmente se realiza una validación de las tramas identificadas.

En el cuarto capítulo se realiza el diseño y la implementación del sistema software de detección de obstaculos, teniendo como referente los requerimientos generales del sistema de detección y haciendo uso de las tramas de datos obtenidas en el tercer capítulo. La implementación software se realiza con la plataforma software Player, creando un plugin driver que permite establecer la conexión entre el escáner láser y el computador para recibir los datos de las distancias y posteriormente entregar esta información a la aplicación software del sistema de detección de obstáculos para procesarla y cumplir con los requerimientos del sistema.

Finalmente en el quinto capítulo se realiza la validación del sistema de detección de obstáculos implementado, esta se realiza haciendo pruebas tanto en escenarios interiores como en exteriores.

Los Anexos se consultan en el CD adjunto a este documento.

CAPITULO 1. INTRODUCCIÓN TEÓRICA

La robótica móvil es un campo de constante de I+D y actualmente cuenta con un historial considerable de trabajos de investigación, desarrollos y propuestas, que en gran medida sustentan las bases para proyectos de mayor alcance e inclusive nuevas propuestas o líneas de investigación [1].

En este capítulo se presenta una breve introducción teórica sobre la robótica móvil (definición, uso, autonomía, problemas, software y hardware), el problema de detección de obstáculos (definición, enfoques, sensores usados, limitaciones de los sensores y el escáner láser en la detección de obstáculos) y finalmente una descripción del escáner láser usado en este proyecto (definición, funcionalidad, escáner láser *Allen Bradley* y algunas aplicaciones).

1.1 ROBÓTICA MÓVIL

Un robot es un dispositivo electro - mecánico multifuncional reprogramable de tres o más ejes, que puede estar fijo en un lugar o en movimiento para trasladar materiales, piezas, herramientas o dispositivos especiales, mediante movimientos programados y variables, que le permiten llevar a cabo diversas tareas [2]. Estos, se clasifican según su estructura mecánica en robots cartesianos, cilíndricos, esféricos, scara, articulado y paralelo; y según su tipo de control en robots de control de secuencia, de trayectoria operada, de adaptación y teleoperados [2].

La fabricación del primer robot móvil llamado Shakey fue iniciada desde 1966 hasta 1972, condujo a la aparición de nuevas tecnologías de planificación y razonamiento automático, este robot podía realizar tareas que requerían la planificación, búsqueda y reorganización de objetos simples. Shakey se desarrolló en dos etapas, la primera fue terminada en 1969, donde se implementó un sistema de robot móvil independiente equipada con una cámara de TV y otros sensores controlados por un computador *SDS-940*; y la segunda fue terminada en 1971, donde el sistema de robot era más robusto llevando a cabo grandes mejoras en el programa y sustituyendo el *SDS-940* por el equipo *Digital Equipment Corporation PDP-10/PDP-15* [3].

1.1.1 Uso de los robots móviles

Los robots móviles brindan la posibilidad de navegar en distintos terrenos y tienen aplicaciones como: exploración minera, exploración planetaria, misiones de búsqueda y rescate de personas, limpieza de desechos peligrosos, automatización de procesos, vigilancia, reconocimiento de terreno, y también son utilizados como plataformas móviles que incorporan un brazo manipulador [4].

1.1.2 Algunos problemas de la robótica móvil

La Robótica Móvil a pesar de los avances y múltiples aplicaciones, sigue presentando varios inconvenientes en procesos fundamentales para su optimo funcionamiento como: **Percepción:** proceso en el que se recibe, elabora e interpreta la información proveniente de su entorno, **Localización:** designa la orientación y posición de un objeto sobre la superficie de la tierra con la ayuda de un sistema de referencia, frecuentemente se utiliza las coordenadas geográficas, **Navegación:** busca que el robot móvil tenga la capacidad de evadir obstáculos, hacer un mapa de navegación interno y ser capaz de moverse en ambientes desconocidos. **Inteligencia:** capacidad de tomar decisiones al enfrentarse a situaciones desconocidas. **Autonomía:** indica cómo se deben hacer las cosas [5]. Sin embargo el problema de la **Detección de obstáculos** en la robótica móvil es clave, ya que es la encargada de dotar al robot de sensibilidad con el mundo que lo rodea, al atacar este problema se buscan dos objetivos primordiales; el primero, obtener la localización precisa de los posibles obstáculos y el segundo, la determinación de la distancia hasta dichos objetos [6].

1.1.3 Robótica – Software

En la robótica, el software se define como el código de comandos que le indican a un dispositivo mecánico (para este caso el robot) las tareas a realizar y cómo controlar sus acciones, para realizar y automatizar tareas. La programación de robots no es una tarea fácil, es por esto, que se han propuesto diferentes sistemas software para hacer que la programación de robots sea más fácil [7].

Algunas técnicas de programación de flujo de datos, son utilizadas por la mayoría de fabricantes de robots, y se basa en el concepto de que cuando el valor de una variable cambia, los valores de otras variables afectadas también deben cambiar. Un lenguaje de programación que incorpora los principios de flujo de datos se denomina un lenguaje de flujo de datos, donde además del procesamiento numérico, también incorpora conceptos funcionales. A diferencia de otros lenguajes de programación que utilizan la programación imperativa, la programación de flujo de datos se modela como una secuencia de funciones [8].

La robótica – software tiene el reto de resolver tareas de investigación o de aplicación que son cada vez más complejas. La integración de sistemas tiene que lidiar con diferentes componentes funcionales. En la actualidad, se usan las arquitecturas software de robótica para separar estos componentes entre sí y crear un software modular y reutilizable, con el fin de utilizar las características específicas de cada aplicación y mejorar la comunicación y eficiencia entre cada uno de los módulos software del robot. Al poner varios componentes en un solo hilo de procesamiento, se pueden evitar bloqueos o copias mientras los datos se

intercambian entre las partes que componen el robot [9].

1.1.4 Robótica – Hardware

Un robot se compone de tres secciones: estructura, sensores y actuadores. Cada módulo hardware que componen el robot tiene su propio identificador (ID), con el fin de controlar cada hardware [10].

En la robótica móvil los sensores se pueden clasificar de diferentes maneras dependiendo de su uso, a continuación se nombran las más comunes [11]:

1.1.4.1 Sensores proprioceptivos

Los sensores de este tipo perciben el estado interno de robot, brindándole información acerca de la posición y la orientación. También se conocen como sensores de posicionamiento y los más usados en robótica móvil son los sensores, GPS, giroscopios, *shaft encoders* incremental, *encoders* absolutos, sensores de velocidad, acelerómetros y detectores de marcas activas [11].

1.1.4.2 Sensores exteroceptivos

Son los que perciben los aspectos externos al robot. Los sensores de este tipo más usados en la robótica móvil son: sensores para medir distancias basados en ultrasonidos o láser, sensores para medir distancias que funcionan en el espectro infrarrojo, visión artificial, sensores de iluminación (fotorresistencias, fotodiodos y fototransistores), sensores de contacto, sensores táctiles, radar basado en microondas y sensores de proximidad (sensores inductivos, sensores de efecto Hall y sensores capacitivos) [11].

1.2 DETECCIÓN DE OBSTACULOS

Es importante definir qué es un obstáculo antes de discutir cómo un obstáculo puede ser detectado, por lo tanto se define como un obstáculo algo que va a provocar un comportamiento peligroso o indeseable en el robot si este lo golpea [12].

1.2.1 Detección de obstáculos en entornos interiores y exteriores

En la robótica móvil se encuentran dos tipos de entornos: los interiores y los exteriores; se pueden considerar como obstáculos en entornos interiores objetos tales como: mesas, sillas, cestos de basura, estantes, personas, paredes, entre otros y en entornos exteriores: edificios, arboles, vehículos, rocas, personas, entre otros.

El sistema de detección de obstáculos tanto en interiores como en exteriores debe trabajar en conjunto con un sistema de navegación, esto, con el fin de reducir el

número de falsas alarmas, provocadas por cosas tales como vehículos estacionados, estructuras laterales de carreteras, entre otros; la situación empeora cuando se tiene un ambiente subterráneo donde son más posibles los obstáculos a su alrededor [12].

La detección de obstáculos ayudará a prevenir que el robot choque con los objetos que lo rodean y pueda navegar libremente dentro del área de exploración, puesto que los obstáculos impiden que el robot pueda seguir una trayectoria libre de colisión. Por lo tanto es importante contar con algún mecanismo de detección de los mismos y saber su localización tan precisa como sea posible, esta precisión dependerá del tipo de dispositivo que se use para la detección [13].

1.2.2 Enfoques

Existen dos enfoques diferentes para el problema de la detección de obstáculos [12]: la detección de obstáculos directa y el terreno de mapeo y navegación [12], las cuales se detallan a continuación.

1. Detección de obstáculos directa: Los obstáculos se detectan ya sea por iluminación en la escena, por la reflexión de la luz o por percibir pasivamente energía a partir de los posibles obstáculos. Este enfoque solo detecta los obstáculos y pasa los datos a la navegación real del sistema para la toma de una decisión [12].

2. Terreno de mapeo y navegación: Los obstáculos no son detectados explícitamente, si no que detecta el espacio libre o las zonas navegables que encuentra el vehículo o robot. En este enfoque cualquier espacio que no sea navegable es considerado un obstáculo, se convierte más en un problema de asignación de terreno y la información del espacio libre es enviada directamente al navegador [12].

1.2.3 Sensores usados para la detección de obstáculos

Para la detección de obstáculos la robótica se basa en sensores de proximidad o de contacto y estos a su vez en diferentes tecnologías [14]:

- Sensores de proximidad: permiten saber qué tan cerca se encuentra un obstáculo, y las distancias de proximidad pueden variar dependiendo de las condiciones en las que se lleve a cabo la detección de los obstáculos. La detección puede variar con la iluminación, tipo de superficie del obstáculo o color del obstáculo [14].

- Sensores de contacto: estos sensores funcionan al entrar en contacto con un obstáculo. Su utilidad radica en hacerle conocer al robot que ha chocado con un objeto y por lo tanto es imposible seguir avanzando en esa dirección [14].

Al hacer uso de sensores, ya sea de contacto o de proximidad se busca que el robot navegue de manera segura aún cuando se presenten errores por parte de alguno de los sensores [13].

1.2.4 Problemas que pueden presentar algunos sensores

La mayoría de los problemas de detección de obstáculos se deriva a partir de las condiciones de detección, es decir, que si se tiene un sensor infrarrojo y una iluminación pobre, puede ser que la detección se realice cuando el robot esté muy cerca del objeto o en el peor de los casos no detecte nada y colisione con el objeto, lo cual se puede dar en un ambiente sin iluminación o con un objeto de color oscuro [8].

Otro problema que se puede presentar es el contrario de la situación anterior, cuando se tiene un ambiente con excesiva iluminación y se tiene un objeto cuya superficie refleja la luz. En este caso puede ser que el sensor detecte el obstáculo cuando el robot se encuentre todavía lo suficientemente lejos del objeto [12].

Para estos dos primeros casos en el momento de detectarse el obstáculo las distancias de detección varían y es imposible saber la distancia exacta al contar únicamente con un sensor infrarrojo [14].

En el caso de los sensores de ultrasonido ofrecen una medición de distancia más precisa, que depende del material del obstáculo, y con estos sensores error es mucho menor que en el caso de los sensores infrarrojos [13].

Para el caso de los sensores de contacto el problema radica en el desconocimiento de la ubicación del obstáculo ya que si el robot choca con una esquina del objeto se considera que el robot chocó de frente al objeto, puesto que no hay forma de ubicar el obstáculo [13].

1.2.5 El escáner láser y la detección de obstáculos

El escáner láser 2D proporciona un conjunto de datos en un plano escaneado, donde cada punto del láser se caracteriza por un ángulo de incidencia y una distancia que corresponde a la distancia del objeto más cercano en esa dirección. A partir de estos datos se deben construir un conjunto de grupos en el que cada uno corresponde a un objeto en la escena escaneada [12].

1.3 RADAR LÁSER

El Radar Láser es un equipo que por medio de una luz láser analiza la presencia de objetos y puede entregar datos ya sean en 2D o 3D dependiendo del radar láser usado. Estos datos pueden ser usados para construir modelos digitales que se usan en una amplia variedad de aplicaciones [15].

1.3.1 Definición de láser

Un láser es un oscilador que opera a frecuencias ópticas. Estas frecuencias de operación se encuentran dentro de una región del espectro que se extiende desde el infrarrojo lejano a la luz ultravioleta de vacío (VUV) o región blanda de rayos X [16].

1.3.2 Funcionalidad

El propósito de un radar o escáner láser ya sea 2D o 3D es crear una matriz o un vector de puntos a partir de muestras geométricas. Estos puntos pueden ser usados para extrapolar la forma del objeto [15].

El LIDAR (*Light Detection and Ranging*) o Radar Láser envía un pulso de luz por el medio ambiente el cual al encontrarse con una superficie es reflejado, dicho pulso reflejado es el que ayuda a determinar la distancia a la que se encuentra un objeto. En la mayoría de las situaciones no es suficiente un solo escaneo o barrido; generalmente se programan múltiples escaneos, desde diferentes direcciones para obtener información de todos los lados del objeto, los escaneos tienen que ser introducidos a un sistema común de referencia, este proceso se conoce como alineación y se usa generalmente para crear un modelo más completo [17].

1.3.3 Radar Láser Safe Zone de Allen Bradley

El *Safe Zone Scanner Laser Multizona* de *Allen-Bradley* observado en la Figura 1.1 es un dispositivo optoelectrónico de seguridad tipo 3, que utiliza la reflexión difusa de emisión de luz láser infrarrojo para determinar la intrusión de una persona o un objeto dentro de un área definida. Su principal aplicación es la protección de zonas en ambientes industriales.

Figura 1.1. Safe Zone Scanner Laser Multizona de Allen Bradley [18].

El sistema tiene internamente un espejo que rota periódicamente 360º desviando los pulsos de láser infrarrojo emitidos para crear un campo de detección bidimensional en una superficie de 190º, a una resolución de 0,5º o 0,25º dependiendo de la configuración del *scanner laser;* donde en los ángulos encontrados desde 185º hasta 355º el escáner láser no reporta medidas porque el espejo que desvía los pulsos se encuentra girando dentro del escáner. En la Figura 1.2 muestra el principio de operación de los pulsos emitidos por el *Safe Zone Scanner Laser* [18] [19].

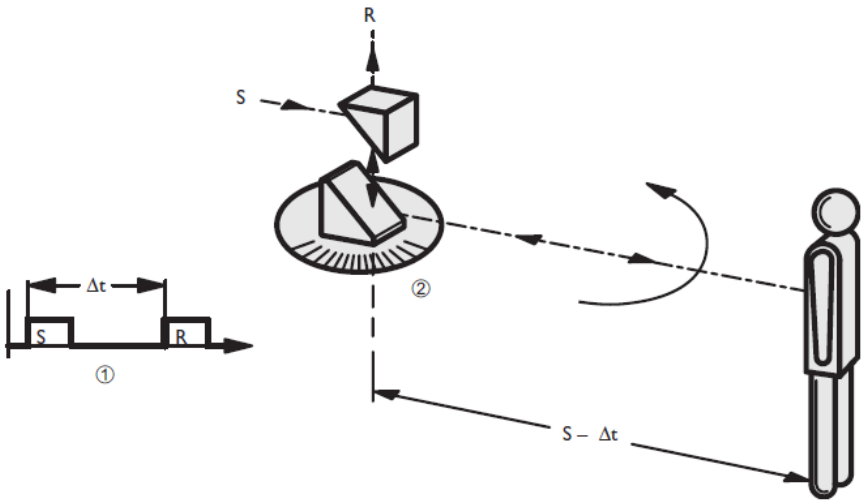


Figura 1.2. Principio de operación de los pulsos emitidos por el Safe Zone Scanner Laser [19].

Donde S es el pulso de luz infrarroja enviada por el escáner láser hacia el objeto y por medio de un cronometro electrónico registra el tiempo (Δt) que tarda en ir y volver el pulso de luz enviado y con esta información el escáner láser se encarga de hacer el cálculo de la distancia al objeto [19].

El escáner láser también cuenta con un espejo que gira a velocidad constante y que desvía los pulsos de luz de tal manera que cubren un campo de 190 °. Al fijar el ángulo de rotación del espejo, el escáner láser determina la dirección del objeto [19].

En la Figura 1.3 se muestra el campo de 190º descrito por las medidas realizadas por el escáner láser que van desde -5º hasta 185º.

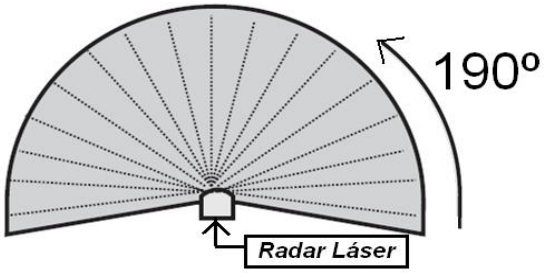


Figura 1.3. Campo descrito por las medidas tomadas por el escáner láser [19].

El escáner láser emite una luz que al encontrar un objeto es reflejada y procesada por el *Safe Zone Scanner Laser*, que envía una señal de parada al cambiar el estado de su OSSD (*Output signal switching device*), cuando se detecta la presencia de un objeto o persona dentro del cambio de detección. El *Safe Zone Scanner Laser* maneja dos campos o zonas, La zona de advertencia y la de protección como se observa en la Figura 1.4.

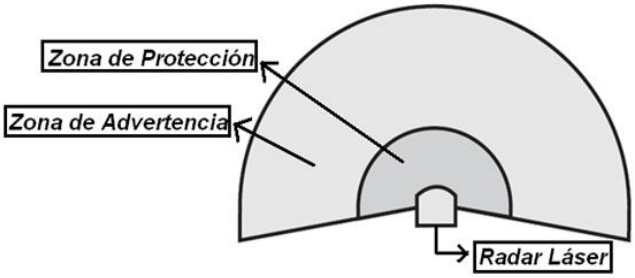


Figura 1.4. Campos de advertencia y de protección del Safe Zone Scanner Laser [19].

El *Safe Zone Scanner Laser* es un radar utilizado en la industria como sensor de seguridad; el dispositivo laser se puede configurar para alertar a una persona que está entrando a una zona de peligro, o para bloquear un proceso completamente si una persona u objeto han entrado a esta zona y el riesgo es inminente [19].

1.3.4 Uso del *Safe Zone Scanner Laser* en la industria

El *Safe Zone Scanner Laser* en la actualidad es usado en la industria para la protección de vehículos y zonas de protección existentes.

Una aplicación del *Safe Zone Scanner Laser* es el sistema: "Protección con el escáner de láser para carro transportador automatizado con relé de seguridad modular Serie MSR200". Esta aplicación describe un sistema de seguridad aplicado al movimiento de un carro transportador automatizado. Los escáneres láser se colocan en la parte frontal y trasera del carro. Los escáneres detectan objetos en la ruta del carro y emiten un comando de paro a los motores del variador si se detecta un objeto. Los escáneres están conectados a un relé de seguridad MSR200, el cual monitorea ambos escáneres y el botón pulsador de paro de emergencia [20].

1.3.5 Algunas Aplicaciones del Radar Láser en la Robótica Móvil

En general el Radar Láser en aplicaciones de robótica es usado para ayudar al posicionamiento del robot creando mapas que permiten al robot evitar los obstáculos.

Algunos trabajos en los que se ha usado el radar láser en la robótica son:

- *Cyclone*: *A Laser Scanner for Mobile Robot Navigation*.
  Es un trabajo realizado por "Carnegie Mellon Robotics Center", en el que usan un *scanner laser* capaz de realizar 7200 mediciones por segundo, con un espejo giratorio que puede generar una vista de 360 grados que es montado en un vehículo robot, este *scanner* ayuda al vehículo a detectar obstáculos en su trayectoria o a localizar el robot en un mapa [21].

- *Real-time Detection of Dynamic Obstacle Using Laser Radar*.
  La detección de obstáculos dinámicos en entornos desconocidos es un problema muy frecuente en los robots móviles, estos obstáculos afectan directamente la precisión en la construcción de mapas para estos robots, es por esto que en este trabajo se usa un radar láser 2D en tiempo real que ayuda a evitar los obstáculos dinámicos existentes en el entorno y al posicionamiento del robot mediante las coordenadas entregadas por el radar, este mapeo entregado en un sistema de coordenadas se superpone con un mapa cuadriculado de elementos previamente identificados y se realiza la comparación de estos 2 mapas obteniendo información de los objetos no identificados en un entorno inicial [22].

- *Adaptive Fuzzy Control for Inter-Vehicle Gap Keeping*.
  En este trabajo se trata el sistema de transporte inteligente (STI) que busca solucionar muchos problemas de transporte. Algunas de las principales características del STI es la adaptación de velocidad del vehículo realizada por un controlador de velocidad y el mantenimiento de una franja de seguridad entre el vehículo controlado y el vehículo precedente en el camino implementando en el vehículo un *scanner laser* que hace que el vehículo disminuya su velocidad cuando se detecte con el scanner la presencia de algún objeto o vehículo [23]. Una vez realizado este trabajo se continuó con el desarrollo de controladores que se usaron en el trabajo "*Cooperative Throttle and Brake Fuzzy Control for ACC+Stop&Go Maneuvers*", donde se sigue usando el *scanner laser* y se concentran en el control de la presión realizada sobre el freno y el acelerador del vehículo [24].

CAPITULO 2: PLATAFORMAS SOFTWARE DE ROBÓTICA MÓVIL

En este capítulo se hace un resumen de algunas plataformas software de uso libre, donde se analiza cada una y se hace la selección para poder cumplir con el objetivo principal; una vez escogida una de las plataformas se describe detalladamente su funcionamiento y el proceso que sigue la plataforma para su uso.

2.1 PLATAFORMAS SOFTWARE DE ROBÓTICA MÓVIL

Una plataforma software es un conjunto de componentes o subsistemas, que forman una estructura común a partir de la cual se pueden desarrollar o construir sistemas software de una forma eficiente [25].

Los sistemas derivados de una plataforma no sólo comparten código, sino requisitos y arquitectura. Por lo tanto, se realiza un proceso de reutilización natural de los artefactos de la plataforma en los diferentes sistemas [25].

2.2 ALGUNAS PLATAFORMAS SOFTWARE PARA ROBÓTICA MÓVIL DE FUENTE ABIERTA

Para este análisis se han seleccionado las plataformas que aparecen en la Tabla 2.1. La elección de cada una se ha realizado de acuerdo a la sonoridad que han tenido en varios estudios comparativos por ser fuente de estandarización y finalmente por ser el fruto de recientes e importantes proyectos de investigación [26].

Tabla 2.1. Listado de las principales Plataformas de Trabajo Robóticas de fuente abierta y libre [26].

Nombre de la Plataforma	Institución de Origen	Página Web
TCA – IPC/TDL	Carnegie Mellon University	www.cs.cmu.edu/~tdl/
TeamBots	Carnegie Mellon University	www.cs.cmu.edu/~trb/TeamBots/
Player / Stage / Gazebo	Universidad de California del Sur	http://playerstage.sourceforge.net/
GenoM	CNRS – LAAS	https://softs.laas.fr/openrobots/
SmartSoft	Universidad de Ulm	www.rz.fh-ulm.de/~cschlege/orocos/
Balt & Cast	Universidad de	www.cognitivesystems.org

	Birgminham	
RT – Middleware	METI, JARA, AIST en Japón	www.is.aist.go.jp/rt/OpenRTM-aist/ htmlen/index.html
MIRO	Universidad de Ulm	http://miro-middleware.berlios.de/
MARIE	Universidad de Sherbrooke	http://marie.sourceforge.net
ORCA2	Universidad de Sydney	http://orca-robotics.sourceforge.net
ADE	Universidad de NotreDame	www.nd.edu/ airolab/software

2.3 SELECCIÓN DE PLAYER / STAGE / GAZEBO

En la Tabla 2.2 se presentan algunas ventajas y desventajas de las plataformas anteriormente descritas:

Tabla 2.2. Ventajas y desventajas de las plataformas consideradas en la Tabla 2.1.

Nombre de la Plataforma	Ventajas	Desventajas
TCA – IPC/TDL [46]	• Se asemeja más a un sistema operativo para diseñar sistemas robóticos. • Comunicación entre módulos. • Planificación de tareas. • Asignación de recursos al robot. • Integración y coordinación de las principales actividades del robot (percepción, planificación y control en tiempo real).	• Es una plataforma que se encuentra en desuso. • Para poder trabajar con TCA se necesita conocer detalles de la API para así usar las librerías bases.
TeamBots [31]	• Soporta prototipos, simulación y ejecución de sistemas multirobot.	• Última versión de la plataforma fue en 2000 y no se volvió a actualizar.
GenoM [37]	• Producción automática de módulos. • Contexto de desarrollo estructurado. • Permite encapsular funciones operativas en módulos	

	independientes.	
SmartSoft [38]	• Maneja un enfoque basado en niveles. • Comunicación asíncrona en el nivel entre componentes, gracias al dominio en los patrones de comunicación.	• No tiene, ni se asocia a un módulo de simulación.
Balt & Cast [39]	• Usa memorias de trabajo para compartir datos.	• Su base se encuentra en otras plataformas tales como MARIE y CORBA.
RT – Middleware [47]	• Sistema basado en componentes. • Tienen un control distribuido.	• No posee una arquitectura funcional. • Manejo de arquitecturas de bajo nivel. • Presenta dificultades al trabajar en tiempo real.
MIRO [48]	• Interoperabilidad en procesos internos. • Herramienta flexible y expandible a nuevos dispositivos y servicios.	• La conexión entre MIRO y CORBA presenta muchas falencias ya que hereda la sobrecarga de CORBA (memoria y potencia de procesamiento) • Por ser tan flexible implica mucho tiempo para poder entender su funcionamiento.
MARIE [49]	• Adaptabilidad a diferentes protocolos de comunicación.	• Su principal interés es la integración de varias herramientas o plataformas de robótica.
ORCA2 [44]	• Manejo de comunicación síncrona y asíncrona.	• Dedicado a la integración de software para robótica. • Se actualizó hasta el 2009.
ADE [45]	• Combina características de los sistemas multiagentes.	• Problemas al trabajar en tiempo real o con

		controladores incorporados.
Player / Stage / Gazebo [50]	• Se encuentra conformado por desarrolladores de varias partes del mundo que se dedican a darle una continua mejora a la plataforma e incorporan nuevos *drivers* desarrollados por diferentes personas en el mundo. • Ofrece dos simuladores tanto para 2D como para 3D.	• Funciona muy bien para LINUX, pero su uso en Windows esta apenas en sus primeros pasos.

La información de la Tabla 2.2 se obtuvo de las referencias desde la [31] hasta la [50]; donde se analizan las ventajas y desventajas de cada una de las plataformas software de robótica móvil de fuente abierta de la Tabla 2.1, sin embargo en el mercado se encuentran varias plataformas tanto de fuente abierta como comerciales, además de las que aparecen en el mercado a diario.

Para el desarrollo de este trabajo se tienen únicamente en cuenta las plataformas de fuente abierta de la Tabla 2.1, ya que la idea es hacer uso de software libre. De las plataformas de la Tabla 2.1 se destaca PSG (Player/Stage/Gazebo), ya que es de gran uso entre los desarrolladores dedicados a la implementación de software para robótica, también por ser una plataforma que es manejada por un grupo de desarrolladores de varias partes del mundo, que se encuentran atentos a los aportes que realizan los usuarios de esta plataforma, con la implementación de nuevos *drivers* y correcciones a *bugs* que se puedan presentar en las versiones liberadas.

2.4 PLAYER / STAGE

Player / Stage / Gazebo son tres segmentos de software desarrollados originalmente en el laboratorio de investigación de robótica de la USC. En la actualidad se encuentra alojado en SourceForge.net.

2.4.1 Player

Es un proyecto que nace en los laboratorios de investigación robótica de la Universidad de California del Sur (USC) y es un servidor que permite controlar los dispositivos de un robot y obtener información de sus sensores. Player es una capa software que abstrae los detalles del hardware del robot, independizándolos. Los algoritmos de control del robot funcionan como clientes de Player (a través de sockets TCP/IP). Así se puede controlar el robot enviando mensajes que sigan el

protocolo de comunicación de Player o llamando a funciones de las librerías de Player, que abstraigan detalles de la comunicación. La distribución actual de Player incluye librerías en diversos lenguajes como C++, Java, Python o Lisp [33].

Player es una *capa de abstracción de hardware* (HAL) para dispositivos robóticos y proporciona un grupo de interfaces estándar, cada una de las cuales especifica la forma en que se interactúa con cierta clase de dispositivos. El trabajo de Player es dar apoyo a la interfaz estándar de cada uno de los dispositivos que componen el robot. De esta manera, el código de control de Player diseñado para un dispositivo podrá ser usado en la implementación de un robot compuesto por varios dispositivos [50].

Player también ofrece mecanismos de transporte que permite el intercambio de datos entre los *drivers* y programas de control que se ejecutan en máquinas diferentes. El transporte de datos más usado es el de cliente /servidor basado en *sockets* TCP. En esta configuración el servidor de Player se ejecuta en un archivo de configuración que define los *drivers* de cada dispositivo enlazándolos al hardware. Los *drivers* se ejecutan dentro del servidor de Player y el código de control se ejecuta como cliente. Las bibliotecas del cliente se encuentran disponibles en varios lenguajes para facilitar el desarrollo de los programas de control [50].

Aunque la mayoría de *drivers* de Player abarcan una gran cantidad de hardware, actualmente se siguen desarrollando *drivers* para soportar aquellos hardware que no se hayan incluido. De igual manera un *driver* a implementar puede hacer uso de los *drivers* existentes, para abstraer los datos de algún hardware si así se desea. El principal uso de los *drivers* abstractos es encapsular algoritmos útiles de manera que se pueden reutilizar fácilmente [50].
Inicialmente la plataforma Player se enfocó a implementar *drivers* para la familia de robots móviles *ActivMedia Pioneer 2*, pero se encontró que dichos *drivers* eran compatibles con otras marcas de sensores y actuadores.

2.4.1.1 Características de Player

Player está diseñado para ser una plataforma y lenguaje independiente. El programa del cliente puede ejecutarse en cualquier máquina que tenga una conexión de red al robot, y puede ser escrito en cualquier lenguaje que soporte sockets TCP. Actualmente se tiene utilidades para el cliente disponibles en C + +, Tcl, Java y Python [51].

Player permite trabajar con múltiples dispositivos que presenten la misma interfaz, tales como la interfaz encargada del control de movimiento del robot, así, el mismo código podría llevar a los dispositivos que trabajen bajo la misma interfaz. Esta característica es muy útil cuando se combina con una simulación de Stage; los programas de control escritos para simular en Stage, pueden ser usados sin cambio alguno en hardware real [51].

Player también está diseñado para apoyar cualquier número de clientes, pueden conectarse y leer los datos de un sensor en cualquiera de los casos de Player. Además de la detección distribuida para el control, también se puede utilizar Player para el seguimiento de los experimentos. Por ejemplo, mientras que los *drivers* de un cliente en C++ para un robot, puede ejecutar una herramienta de visualización gráfica, por medio de otras ventanas muestra los datos actuales del sensor y un programa de registro guarda los datos para un análisis posterior. En un recorrido de peticiones de los dispositivos permitirán a los clientes tener acceso a diferentes sensores y actuadores, según sea necesario para la tarea que se esté ejecutando. El comportamiento del servidor también puede ser configurado sobre el recorrido [51].

Player es software libre, publicado bajo licencia pública GNU [51].

2.4.1.2 Filosofía de la programación con Player

La librería de C++ se desarrolló siguiendo un modelo de programación basado en objetos locales (*proxies*). El cliente interactúa con *proxies*, que sirven como intermediarios de los dispositivos del robot, que serán remotos, ya que pueden estar en cualquier máquina de la red. El programador llama los métodos de los objetos *proxy* para controlar al robot u obtener información del mismo, olvidándose de los detalles de la comunicación por la red. Cada *proxy* es una clase de C++ y representa al propio robot o bien uno de los dispositivos. La documentación de Player incluye una referencia de todas las clases *proxy* existentes junto con los métodos [33].

Los programas realizados en Player, normalmente tienen en cuenta los siguientes pasos [33]:

1. Establecer conexión: la conexión con el robot se hace a través de un objeto de la clase *PlayerClient.*

2. Conexión con dispositivos: se buscan los dispositivos con los que se desea interactuar tales como: sonares, motores, cámaras, láser, entre otros. Se debe tener en cuenta que los *drivers* de dichos objetos incluyen una referencia en

PlayerClient creado en el paso 1, donde se debe indicar además el número de dispositivo al que se desea conectar y si el acceso será de lectura “r”, escritura “w” o lectura/escritura “a”.

3. Ingreso en al bucle: en este paso se leerá información de los sensores y se enviaran instrucciones a los actuadores. Player se comunica con el robot a intervalos regulares un número determinado de veces por segundo. Para esperar a que se produzca dicha comunicación se utiliza el método *Read ()* de la clase *PlayerClient*.

2.4.1.3 Ficheros de programación de Player

Los ficheros de configuración de Player cuya extensión es .cfg contienen información sobre los dispositivos del robot, en el caso de utilizar Stage, estos no serán reales, sino simulados. Para la configuración de los dispositivos del robot, Player distingue entre interfaz y driver. Una interfaz es una serie de funcionalidades que proporciona un determinado dispositivo, mientras que un driver es una implementación concreta de una interfaz. Por ejemplo, una cámara se sabe que puede moverse de izquierda-derecha, arriba-abajo y hacer zoom, para poder controlar un modelo de una cámara en particular se necesita de un driver que implemente esta interfaz. La documentación de Player incluye una referencia de todas las interfaces y los drivers que vienen implementados en la distribución actual del software, los cuales pueden ser usados, a demás de los drivers creados por el usuario [33].

En el fichero .cfg aparecerá cada uno de los *drivers* del robot definido con la palabra clave *driver* y seguido entre paréntesis de sus características, como su nombre, la/s interface/s que implementa, los otros *drivers* de los que depende, etc [33].

2.4.2 Stage

Stage simula robots móviles, sensores y objetos en un entorno de mapa de bits de dos dimensiones. Está diseñado para apoyar la investigación en sistemas multi-agente autónomos, proporcionando de manera sencilla, una gran cantidad de dispositivos [52].

El control del robot se compila y se carga en tiempo de ejecución, se puede conectar a cualquier modelo de dispositivo. Los controladores tienen acceso completo a la API de Stage [52].

Otro de sus usos destacados es emplearlo como una biblioteca C++ para proporcionar una simulación de un robot dentro de sus propios programas. Esto es

útil si Player no es adecuado para el objetivo que se busca, o si se quieren modelos personalizados de simulación basado en un motor de simulación conocido [52].

En la Figura 2.1 se muestra una simulación realizada con Stage.

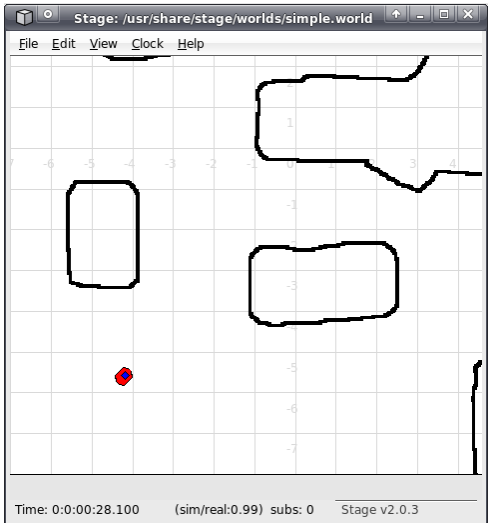


Figura 2.1. Simulación con Stage [52].

2.4.3 Player / Stage

Stage a menudo usa un *plugin* de Player, proporcionando dispositivos virtuales para Player. El usuario escribe el *driver* del dispositivo como "Cliente" para ser usado en el "Servidor" de Player. Normalmente, los clientes no distinguen la diferencia entre los dispositivos reales del robot y sus equivalentes en la simulación de Stage. Los clientes usados en Player para la simulación con Stage se han desarrollado utilizando una etapa a la que al trabajar con dispositivos reales no deberá hacérseles modificaciones y en el caso de haberlas serán muy pocas. Así Stage permite crear rápidamente prototipos de *drivers* destinados para robots reales, también permite simulación de dispositivos hardware interfazados con Player. Varios modelos de sensores y actuadores son proporcionados, incluyendo sonares, escáner, la visión (detección de color), la cámara de mapeo de profundidad 3D, odometría, entre otros. [52].

2.4.3.1. Cómo usar Player / Stage

Player/Stage es una herramienta de simulación de robots. Player, es una capa de abstracción de hardware que permite controlar el código del cliente por medio de las transferencias de datos entre Player y los dispositivos hardware del robot. Stage es un *plugin* para Player, que recibe los datos y simula la información recibida convirtiendo estas instrucciones en una simulación de su robot. También

simula los datos de los dispositivos y envía esto a Player haciendo que los datos simulados se encuentren a disposición del código del usuario [53].

Una simulación entonces se compone de tres partes:

1. Código del usuario: cumple con el papel de cliente al interactuar con Player

2. Player: es el servidor encargado de interactuar con el código del cliente y envía instrucciones a los dispositivos. Desde los dispositivos consigue datos y los envía a Stage para la simulación.

3. Stage: usa interfaces que interactúan con Player para realizar dicha simulación cuando el dispositivo real se encuentra conectado a Player, de no ser así, recibe las instrucciones de Player y pone a funcionar el dispositivo simulado sin necesidad de tener conectado el dispositivo real.

2.4.3.2 Tipos de archivos de Player / Stage

En Player/Stage hay 3 tipos de archivo que se deben entender para poder trabajar con Player/Stage: world, cfg (configuración) e inc (incluir) [53].

El archivo .world es el encargado de mostrar a Player/Stage los objetos que están disponibles para crear el espacio de simulación. En este archivo se puede describir el dispositivo, los elementos encontrados en el entorno y la disposición del entorno en sí.

El archivo .inc sigue la misma sintaxis que el formato de un archivo .world, con la diferencia de que los archivos .inc pueden ser incluidos. De esta manera si hay un objeto extraño en el entorno que puede ser usado en otros entornos, como un modelo de un robot, poniendo la descripción del robot en un archivo .inc no sólo hace que sea más fácil de copiar, sino que también significa que si se desea cambiar la descripción de su robot después, sólo tendrá que hacerlo en un lugar y las múltiples simulaciones también cambiaran.

El archivo .cfg es el encargado de la información de los dispositivos usados en Player .Este archivo indica a Player los *drivers* que se usaran para interactuar con los dispositivos del robot y como debe interpretar los datos recibidos del *driver*. Los objetos descritos en el archivo .world, deberán describirse en el archivo .cfg si se desea que el código del usuario interactué con estos objetos, si no quiere interactuar con ellos desde el código, entonces no es necesario. El archivo .cfg hace todas estas especificaciones usando interfaces y los drivers.

2.4.4 Stage / Gazebo

La plataforma Player ofrece dos simuladores multi-robots: Stage y Gazebo. Estos simuladores son compatibles con Player, usando programas escritos como clientes para cualquiera de los dos simuladores, sin necesidad de hacer modificaciones para ejecutarlos. La diferencia clave entre los dos simuladores es que Stage, está diseñado para simular una población de robots de gran tamaño con baja fidelidad en 2D, mientras que Gazebo, está diseñado para simular una pequeña población de robots con alta fidelidad en 3D. Ambos simuladores son gratuitos y los usuarios pueden alternar entre ellos de acuerdo a sus necesidades [34].

En la Figura 2.2 se puede ver una simulación realizada con Gazebo.

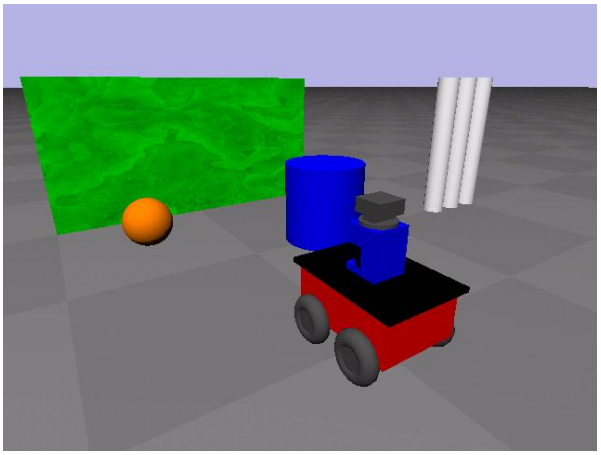


Figura 2.2. Simulación con Gazebo [34].

CAPITULO 3: DEDUCCIÓN DE LAS TRAMAS DE DATOS ENTRE EL ESCANER LÁSER Y EL COMPUTADOR

En este capítulo se ilustra cómo se establece conexión con el puerto serie, la configuración y la visualización de las distancias en el escáner láser *Safe Zone* usando la herramienta software *Safety Configuration & Diagnostic* (SCD).

Igualmente se diseñan una serie de experimentos para la deducción de las tramas del protocolo de comunicación con sus respectivos resultados y finalmente se presenta una solución para el envío de las tramas y la validación de los resultados obtenidos en los experimentos.

3.1 SOFTWARE SCD DE ALLEN *BRADLEY*

El software SCD es proporcionado por *Allen Bradley – Rockwell Automation* para establecer una conexión a nivel de software entre el computador y el *Safe Zone Scanner Laser Multizona*. Por medio del software SCD, se puede establecer la conexión por el puerto serie, configurar el escáner láser y visualizar las distancias en sus diferentes zonas tanto a nivel gráfico como de texto, para realizar dichas tareas el software SCD trabaja por medio de proyectos que son mostrados en forma de árbol.

Usando el software SCD, se pueden crear proyectos sin necesidad de establecer ninguna conexión con el dispositivo, configurarlo y guardar la configuración. Cuando se establezca una conexión con el dispositivo, se puede disponer de los proyectos guardados y transferir las configuraciones almacenadas.

Una vez iniciado el software SCD aparece una ventana como se muestra en la Figura 3.1 donde se deben tener en cuenta tres partes importantes en el uso de este software: 1) establecer conexión con el puerto serie, 2) realizar la configuración del escáner laser y 3) visualizar las distancias entregadas por el escáner láser.

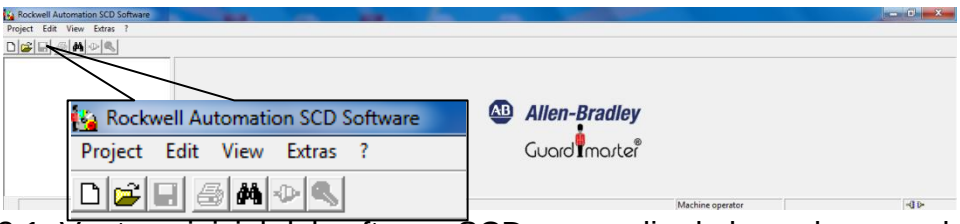


Figura 3.1. Ventana inicial del software SCD por medio de la cual se va a hacer la conexión, configuración y visualización de las distancias.

3.1.1 Conexión con el puerto serie

Para establecer una conexión con el puerto serie se debe:

- Escoger el puerto COM con el que se va a trabajar, ingresando a Extras / Enlace de comunicación *(Communication connection)* como se muestra en la Figura 3.2.

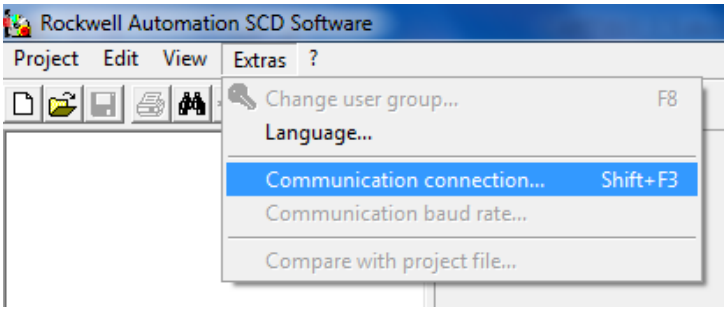


Figura 3.2. Ruta de acceso en el SCD para escoger el puerto serie de comunicación con el laser.

- Ingresando por la ruta mostrada en la Figura 3.2, se abre la ventana que se observa en la Figura 3.3 y se hace click en Conexión (*Connection)* para poder acceder a los parámetros de comunicación (*Communication parameters*), donde finalmente se escoge el puerto COM con el que se va a trabajar.

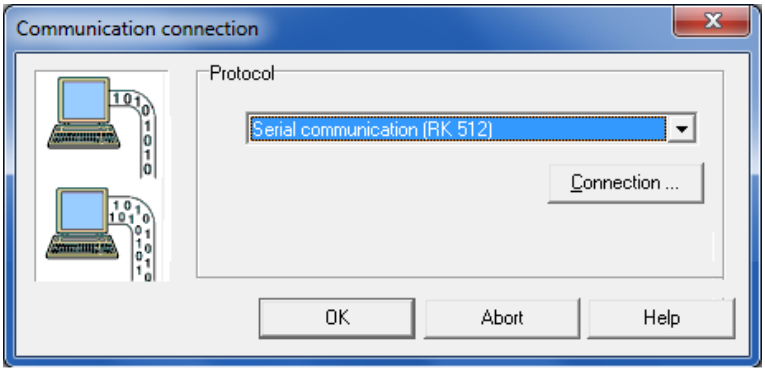


Figura 3.3. Ventana de enlace de conexión para acceder finalmente a modificar los parámetros de comunicación.

- En seguida se abre la ventana de parámetros de comunicación como aparece en la Figura 3.4, donde se puede escoger el puerto serie COM al que se conectó el escáner láser, dependiendo del computador usado, puede ofrecer más de un puerto COM para la conexión.

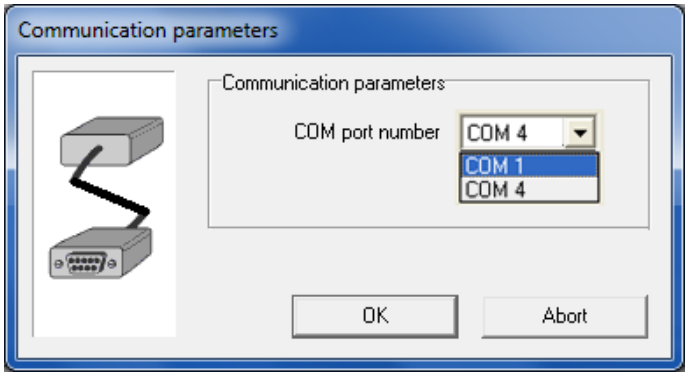


Figura 3.4. En la ventana de parámetros de comunicación se escoge el puerto COM al que se conectó el escáner láser.

Una vez se ha escogido el puerto serial al que se conecta el escáner láser se puede crear (*New*), abrir (*Open*) o identificar (*Identify*) un proyecto; para cualquiera de estas tres opciones se deben escoger de la lista que se despliega del menú proyecto (*Project*), tal como se observa en la Figura 3.5, en el caso de la identificación del proyecto para crear finalmente la conexión entre el escáner láser y el software SCD se debe:

- Ingresar a Proyecto y seleccionar Identificar, tal como se observa en la Figura 3.5.

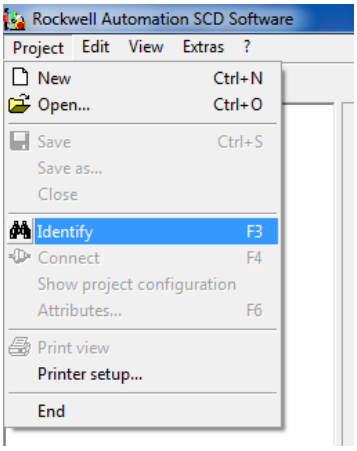


Figura 3.5. Ruta de acceso para realizar la identificación de la conexión.

Una vez identificada la conexión se procede a determinar el usuariocon el que se va a trabajar y se hacer por medio de la ventana mostrad en la Figura 3.6, donde se tienen tres opciones: 1) Operador de la máquina (*Machine Operator*), 2) Personal de mantenimiento (*Maintenance Personnel*) y 3) Cliente Autorizado (*Authorised client*). Cada uno de estos usuarios tiene permisos diferentes sobre el uso del software SCD, para el caso de este trabajo de grado se empleó el cliente autorizado.

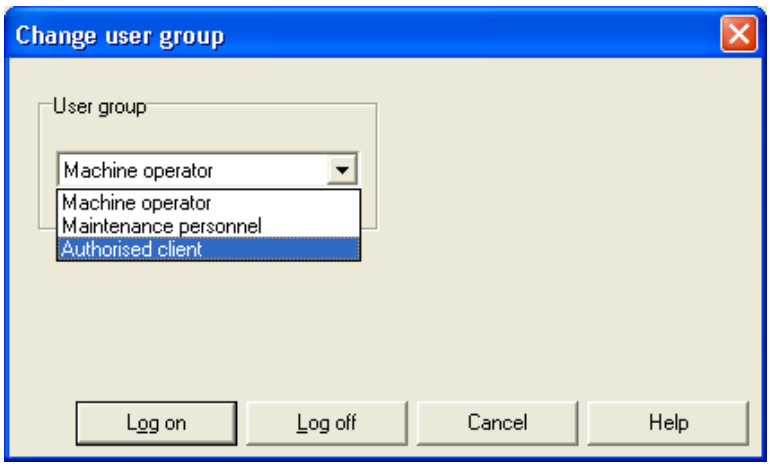


Figura 3.6. Ventana para el cambio de grupo de usuario.

Al escoger como usuario el cliente autorizado solicita una contraseña (P*assword*), la contraseña de fábrica es “ABGM”, La contraseña solo se usa en los usuarios: personal de mantenimiento y cliente autorizado.

Finalmente en el proceso de identificación se muestra el mensaje de la Figura 3.7, donde se confirma que el proyecto ha sido debidamente identificado.

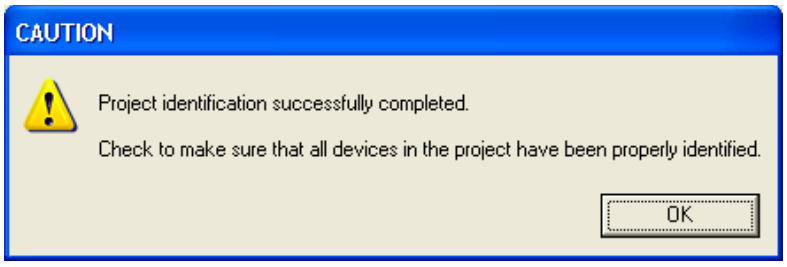


Figura 3.7. Confirmación de que la identificación del proyecto se ha completado.

3.1.2 Configuración del escáner láser

Para el caso de la configuración del escáner láser se cuenta con dos clases de configuración: 1) configuración del proyecto (*Configuration Draft*) y 2) editor del campo establecido (*Field set editor*). Cuando se establece conexión entre el computador y el escáner láser no es necesario realizar siempre la configuración, debido a que el escáner láser conserva la ultima configuración realizada.

A continuación se explica cada una de las dos clases de configuración.

3.1.2.1 Configuración del Proyecto (*Configuration Draft*)

Por medio de la configuración del proyecto se determina la resolución, tiempos de respuesta, distancias máximas del campo de protección, resolución angular y conjuntos de campos o zonas. Una vez identificado, se despliegan del proyecto

unas opciones, que varían de acuerdo a los dispositivos con los que se haya establecido comunicación, para los propósitos de la investigación será el dispositivo *Safe Zone.*

Al dar click derecho en el sistema *Safe Zone* (*Safe Zone system*) aparecen tres (3) opciones, entre ellas la configuración del proyecto que a su vez presenta opciones relacionadas con esta configuración: recibir, transmitir y editar. (Ver Figura 3.8).

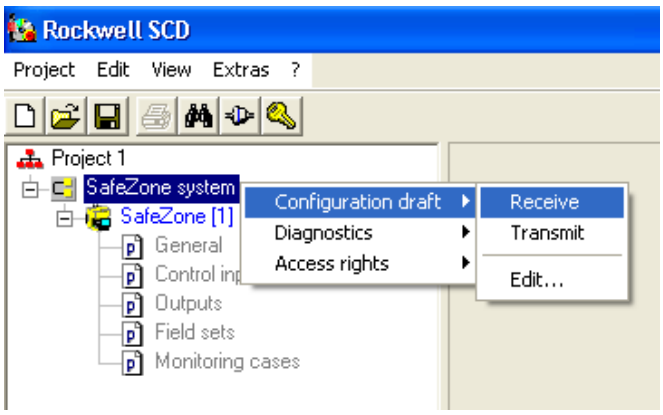


Figura 3.8. Opciones para la configuración del proyecto.

- **Recibir (*Receive*):** permite leer y cargar la última configuración guardada en el dispositivo.

- **Transmitir (*Transmit*):** seleccionando esta opción el software SCD se encarga de transmitir la configuración almacenada al dispositivo.

- **Editar (*Edit*):** por medio de esta opción se puede modificar la configuración del proyecto y se compone de 10 pasos guiados, donde se asigna a la configuración un nombre que la identifique, se determina la resolución, tiempos de respuesta, distancias máximas del campo de protección, resolución angular, conjuntos de campos o zonas y los casos; para avanzar de un paso a otro, se debe hacer click en el botón que aparece en la parte inferior derecha llamado “Continuar (*Continue*)”. (Ver Figura 3.9)

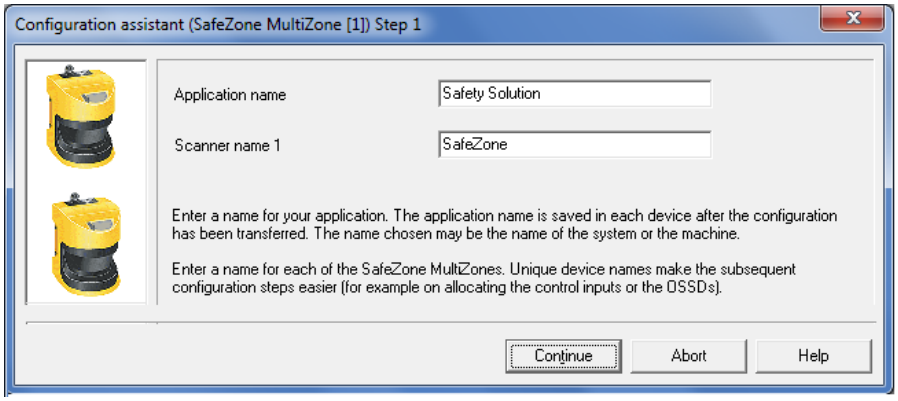


Figura 3.9. Paso 1 de la configuración del proyecto donde se asigna un nombre a la aplicación.

Para obtener mayor información acerca de cada uno de los pasos realizados para la configuración del proyecto, revisar el Anexo 1: "Instalación y manejo del *safety configuration & diagnostic software* (SCD)".

3.1.2.2 Editor del campo (*Field set editor*)

En el editor de campo se configuran las zonas de advertencia y peligro (o protección). Por medio de esta ventana se pueden crear tanto campos circulares como campos personalizados. En la Figura 3.10 se muestra la ventana del editor de campo, donde se tienen dos zonas creadas, la zona de color gris representa la zona de advertencia, la de color rojo la zona de peligro y de amarillo el escáner láser *Safe Zone Multizone*.

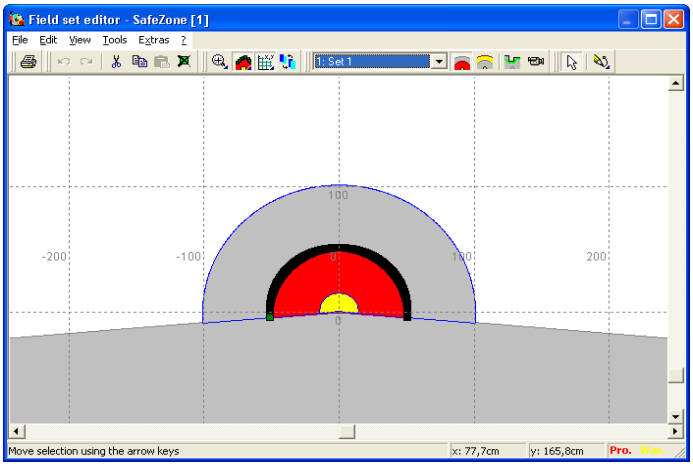


Figura 3.10. Ventana de configuración del editor de las zonas de advertencia y peligro.

3.1.3 Visualización de las distancias (*Data Recorder*)

Las distancias se pueden visualizar gráficamente o en forma de texto, donde los

datos en forma de texto solo se pueden visualizar y no se permite su exportación, para el caso de la visualización gráfica los datos van cambiando en tiempo real (ver Figura 3.11), las líneas azules y rojas representan los objetos que se encuentran alrededor del escáner láser, para este caso la zona amarilla será la de advertencia y de color rojo la zona de peligro o de protección.

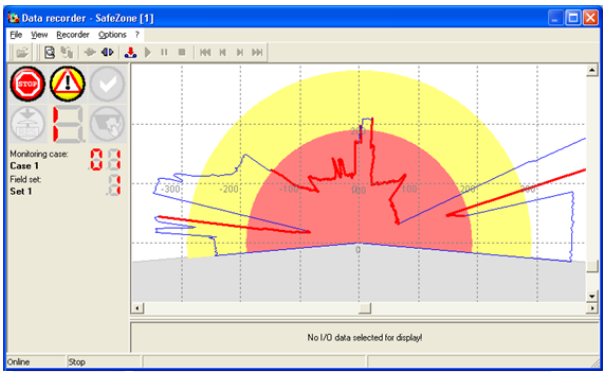
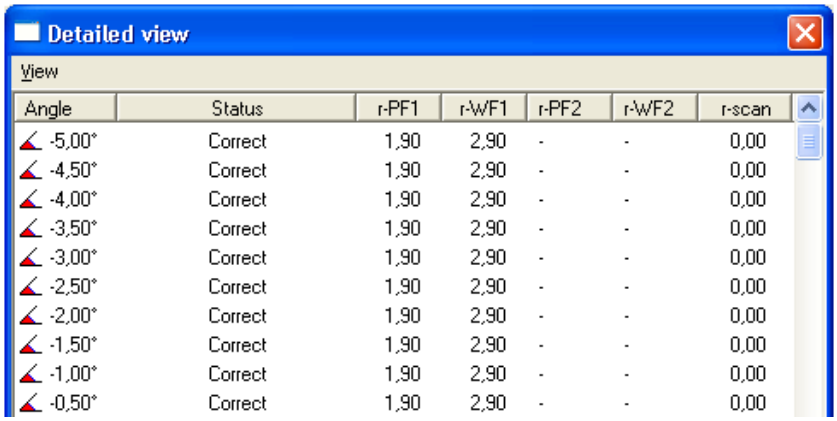


Figura 3.11. Representación gráfica de los datos obtenidos por el escáner láser.

Por otra parte se tienen los datos en forma de texto, cuya información solo se puede visualizar y no puede ser exportada a ningún formato, en la Figura 3.12 se muestran algunos de esos datos, cuando se abre esta ventana los datos que aparecen en ella son los datos medidos desde -5º hasta 185º hasta el momento de abrir la ventana, aunque los datos mostrados aquí son únicamente los de un muestreo no quiere decir que el escáner láser deje de enviar medidas. El incremento en los ángulos depende de la resolución angular (0.25º ó 0.5º) previamente configurada en el SCD.


Detailed view

View

Angle	Status	r-PF1	r-WF1	r-PF2	r-WF2	r-scan
-5,00°	Correct	1,90	2,90	-	-	0,00
-4,50°	Correct	1,90	2,90	-	-	0,00
-4,00°	Correct	1,90	2,90	-	-	0,00
-3,50°	Correct	1,90	2,90	-	-	0,00
-3,00°	Correct	1,90	2,90	-	-	0,00
-2,50°	Correct	1,90	2,90	-	-	0,00
-2,00°	Correct	1,90	2,90	-	-	0,00
-1,50°	Correct	1,90	2,90	-	-	0,00
-1,00°	Correct	1,90	2,90	-	-	0,00
-0,50°	Correct	1,90	2,90	-	-	0,00

Figura 3.12. Primeros 10 datos obtenidos por el escáner láser en forma de texto.

Para tener una mayor información relacionada con el software SCD, revisar el Anexo 1: “Instalación y manejo del *safety configuration & diagnostic software* (SCD)”.

En el proceso de conexión con el puerto serie, la configuración y la visualización de las distancias descritas en las secciones 3.1.1, 3.1.2 y 3.1.3, se llevaron a cabo varios eventos en cada una de las tres etapas, los cuales se listan en la Tabla 3.1, que son la base para los experimentos diseñados en la siguiente sección.

Tabla 3.1. Eventos para cada una de las etapas.

ETAPAS	EVENTOS
1) Establecer conexión	1) Escoger el puerto serial.
	2) Creación del proyecto e identificación de la conexión.
	3) Lectura de la última configuración almacenada.
2) Configuración	4) Selección del usuario.
	5) Iniciar configuración del proyecto.
	6) Iniciar la transferencia de la configuración.
	7) Reconocimiento y transferencia de la configuración.
	8) Cambios en las zonas de advertencia y de protección por medio del editor de campos.
3) Visualización de las distancias	9) Visualización de las distancias

3.2 DISEÑO DE EXPERIMENTOS

Con el diseño de una serie de experimentos se busca determinar la composición de las tramas de datos de la comunicación del puerto serie, tanto las que envía el escáner láser como las que recibe el computador por medio del SCD, por ello se han diseñado cuatro (4) experimentos que emplean elementos hardware y software para facilitar la obtención de las tramas.

En los elementos hardware se tienen: dos computadores, de los cuales uno establece la conexión con el escáner láser (PC1) y el otro se encarga de escuchar los datos que pasan por el puerto serie (PC2), dos conectores DB9 hembras que van conectados al puerto serie de los computadores usados para la prueba, un conector DB9 macho que va al conector hembra del puerto serie del escáner láser, esto con el fin de realizar la conexión de tal manera que permita trabajar con los dos computadores y el escáner láser, finalmente una *protoboard* donde se conectan los tres conectores DB9 para las pruebas que se van a realizar. Los conectores DB9 tienen un cable encargado para la transmisión (Tx) de datos, otro para la recepción (Rx) de los datos y tierra. En la Figura 3.13 se pueden observar los elementos hardware usados en los experimentos.

Figura 3.13. Elementos hardware usados en los experimentos y validaciones.

Al trabajar con el software SCD conectado al escáner no es posible determinar que está sucediendo internamente en el protocolo de comunicación, es por esto que se ve la necesidad de usar tres herramientas software como: el serial_port_monitoring [55] que monitorea los datos que se transmiten y se reciben por el puerto serie sin ocuparlo, la herramienta Terminal [56] que permite recibir los datos que se están transmitiendo por el puerto serie en sistema hexadecimal, decimal o binario, y según los requerimientos del proyecto se trabaja con el sistema hexadecimal; y el otro software usado es el Hercules [57] que permite enviar cadenas de datos en hexadecimal.

Con los elementos hardware y software nombrados se diseñan cuatro (4) experimentos; con el experimento 1 se quiere determinar, dentro de las tres etapas seguidas en el manejo del software SCD, en que eventos se transmiten datos, ya sea del escáner láser al computador o viceversa (ver Tabla 3.1); en el experimento 2 se busca determinar los datos enviados desde el computador al escáner láser, cuando el software SCD ejecuta las tres etapas con los eventos identificados en el experimento 1, con el experimento 3 se busca determinar los datos que da como respuesta el escáner láser al recibir los datos enviados desde el software SCD para los eventos identificados en el experimento 1 y con el experimento 4 deducir la organización de los datos en las tramas enviadas entre el escáner y el software SCD en los eventos identificados.

3.2.1 Experimento 1

Con este experimento de determinan los eventos en los que se transmiten datos a través del puerto serie para cada una de las tres etapas (ver Tabla 3.1), sin importar si son datos de transmisión o recepción.

Para la realización de este experimento se realiza montaje de equipos de la Figura 3.14 y a través del software SCD se lleva a cabo cada una de las etepas con sus respectivos eventos, y se emplea el software serial_port_monitoring para identificar los eventos en los cuales se manifiesta la transmisión de datos.

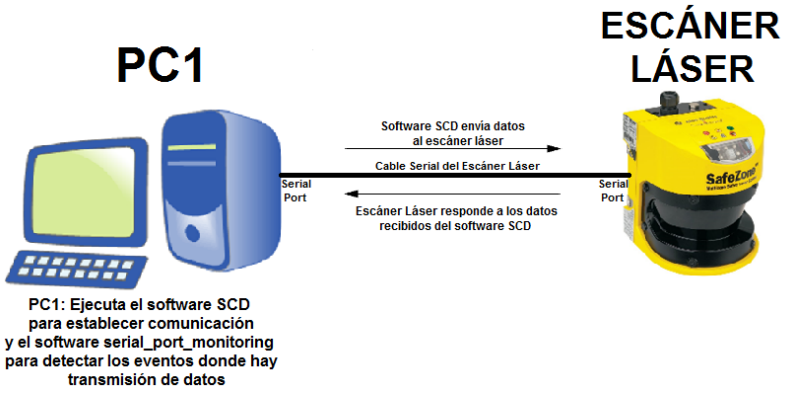


Figura 3.14. Montaje de los equipos usados en el experimento 1.

Para el experimento 1 se realizan los pasos de la Tabla 3.2, correspondientes a los nueve eventos observados en la Tabla 3.1 y se guardan los datos obtenidos del software serial_port_monitoring.

Tabla 3.2. Pasos realizados en el experimento 1 para cada uno de los nueve eventos.

PASOS	EVENTO REALIZADO EN EL SOFTWARE SCD
1	Escoger el puerto serial.
2	Creación del proyecto e identificación de la conexión.
3	Lectura de la última configuración almacenada.
4	Selección del usuario.
5	Iniciar configuración del proyecto.
6	Iniciar transferencia de la configuración.
7	Reconocimiento y transferencia de la configuración.
8	Cambios en las zonas de advertencia y de protección por medio del editor de campos.
9	Visualización de las distancias.

En los datos obtenidos del serial_port_monitoring se tienen unas funciones propias de la herramienta, para verlas y conocer la función de cada una de ellas, revisar la Tabla 3.3. En la descripción de algunas de las funciones se encuentran involucrados dos actores: el COM que en este caso representa al escáner láser y el CLIENTE que es el software SCD.

Tabla 3.3. Tabla de las funciones usadas por el software serial_port_monitoring con su respectiva descripción

FUNCIÓN	DESCRIPCIÓN
IRP_MJ_CREATE	Abre o crea conexión con el puerto.
IRP_MJ_CLOSE	Cierra el puerto.
IRP_MJ_WRITE	Solicita la transferencia de datos del CLIENTE al puerto COM.
IRP_MJ_READ	Transferencia de datos del puerto COM al CLIENTE.
IRP_MJ_DEVICE_CONTROL	Solicitud para la operación de un puerto serie.
IOCTL_SERIAL_PURGE	Solicita la cancelación de peticiones y borra los datos de los Buffers.
IOCTL_SERIAL_SET_BAUD_RATE	Solicita establecer la velocidad de transmisión que está configurada para el puerto COM (38.400 baudios)

Al realizar cada uno de los nueve (9) pasos descritos en la Tabla 3.2, se tiene que en los pasos 4, 5 y 8 no se registra ningún tipo de comunicación entre el escáner láser y el SCD; mientras que para los otros 6 pasos se obtienen datos de transmisión, algunos de estos datos se pueden ver en la Tabla 3.4. Las tablas completas se pueden ver en el Anexo 2: “Resultados de los experimentos”.

Tabla 3.4. Funciones y datos intercambiados entre el software SCD y el escáner láser en los eventos en los que se registro transmisión de datos.

EVENTO PASO 1: Se escoge el puerto	
FUNCIÓN	**DATOS HEXADECIMALES**
IRP_MJ_CREATE	
IRP_MJ_CLOSE	
EVENTO PASO 2: Creación del proyecto e identificación de la conexión	
FUNCIÓN	**DATOS HEXADECIMALES**
IRP_MJ_WRITE	00 00 45 44 03 00 00 05 ff 0f
IRP_MJ_READ	00 00 00 00
	03 00 00 05 ff 07 80 00 b7 0f
IRP_MJ_DEVICE_CONTROL	
IOCTL_SERIAL_PURGE	
IRP_MJ_CLOSE	
IRP_MJ_CREATE	
IRP_MJ_DEVICE_CONTROL	
IOCTL_SERIAL_SET_BAUD_RATE	

IOCTL_SERIAL_PURGE	
IRP_MJ_WRITE	00 00 45 44 03 00 00 05 ff 0e
IRP_MJ_READ	00 00 00 0a
IRP_MJ_DEVICE_CONTROL	
IOCTL_SERIAL_PURGE	
EVENTO PASO 3: Lectura de la ultima configuración almacenada	
FUNCIÓN	**DATOS HEXADECIMALES**
IRP_MJ_DEVICE_CONTROL	
IOCTL_SERIAL_PURGE	
IRP_MJ_WRITE	00 00 41 44 0e 00 00 09 ff 07 0e 00 00 09 ff 07
	03 00 0c b1 d8 a9 55 7b 54 0e f3 e2 00 09 ff 07
IRP_MJ_READ	00 00 00 00
IRP_MJ_DEVICE_CONTROL	
IOCTL_SERIAL_PURGE	
IRP_MJ_WRITE	00 00 41 44 0e 00 00 09 ff 07 0e 00 00 09 ff 07
	00 00 30 30 30 30 30 30 30 fb 07 00 09 ff 07
IRP_MJ_READ	00 00 00 00
IRP_MJ_DEVICE_CONTROL	
IOCTL_SERIAL_PURGE	
IRP_MJ_WRITE	00 00 45 44 19 00 00 05 ff 07
IRP_MJ_READ	00 00 00 00
	19 00 00 05 ff 07 07 0f 9f d0
IRP_MJ_DEVICE_CONTROL	
IOCTL_SERIAL_PURGE	
EVENTO PASO 6: Iniciar la transferencia de la configuración	
FUNCIÓN	**DATOS HEXADECIMALES**
IRP_MJ_DEVICE_CONTROL	
IOCTL_SERIAL_PURGE	
IRP_MJ_WRITE	00 00 41 44 19 00 00 05 ff 07 19 00 00 05 ff 07
	07 0f 9f d0 19 00 00 05 ff 07 19 00 00 05 ff 07
IRP_MJ_READ	00 00 00 00
IRP_MJ_DEVICE_CONTROL	
IOCTL_SERIAL_PURGE	

IRP_MJ_WRITE	00 00 45 44 0a 00 00 09 ff 07
IRP_MJ_READ	00 00 00 00
	0a 00 00 09 ff 07 09 33 54 09 b9 04 1f 00 00 00
	1c 87 00 09 ff 07 09 33 54 09 b9 04 1f 00 00 00
IRP_MJ_DEVICE_CONTROL	
IOCTL_SERIAL_PURGE	
IRP_MJ_WRITE	00 00 45 44 19 00 00 05 ff 07
IRP_MJ_READ	00 00 00 00
	19 00 00 05 ff 07 07 0f 9f d0
EVENTO PASO 7: Reconocimiento y transferencia de la configuración	
FUNCIÓN	**DATOS HEXADECIMALES**
IRP_MJ_DEVICE_CONTROL	
IOCTL_SERIAL_PURGE	
IRP_MJ_WRITE	00 00 41 44 04 00 00 05 ff 07 04 00 00 05 ff 07
	00 01 16 c3 04 00 00 05 ff 07 04 00 00 05 ff 07
IRP_MJ_READ	00 00 00 00
IRP_MJ_DEVICE_CONTROL	
IOCTL_SERIAL_PURGE	
IRP_MJ_WRITE	00 00 41 44 04 00 00 05 ff 07 04 00 00 05 ff 07
	00 00 37 d3 04 00 00 05 ff 07 04 00 00 05 ff 07
IRP_MJ_READ	00 00 00 00
IRP_MJ_DEVICE_CONTROL	
IOCTL_SERIAL_PURGE	
IRP_MJ_WRITE	00 00 45 44 0b 00 00 56 ff 07
IRP_MJ_READ	00 00 00 00
	0b 00 00 56 ff 07 08 3b fd fd 00 02 00 00 ff 7f
	00 00 00 08 26 00 ba 07 00 00 47 00 00 00 46 00
	47 00 00 00 46 00 b9 01 c6 01 42 00 04 00 00 00
EVENTO PASO 9: Visualización de las distancias	
FUNCIÓN	**DATOS HEXADECIMALES**
IRP_MJ_DEVICE_CONTROL	

IOCTL_SERIAL_PURGE	
IRP_MJ_WRITE	00 00 41 44 19 00 00 05 ff 07 19 00 00 05 ff 07
	07 0f 9f d0 19 00 00 05 ff 07 19 00 00 05 ff 07
IRP_MJ_READ	00 00 00 00
IRP_MJ_DEVICE_CONTROL	
IOCTL_SERIAL_PURGE	
IRP_MJ_WRITE	00 00 45 44 15 00 00 0e ff 07
IRP_MJ_READ	00 00 00 00
	15 00 00 0e ff 07 36 f1 5d 25 00 00 00 00 00 00
	00 00 00 00 00 00 00 00 00 00 56 88 00 00 00 00
IRP_MJ_DEVICE_CONTROL	
IOCTL_SERIAL_PURGE	
IRP_MJ_WRITE	00 00 45 44 19 00 00 05 ff 07
IRP_MJ_READ	00 00 00 00
	0b 00 00 56 ff 07 00 20 ff 00 00 02 00 00 ff 7f
	00 00 00 08 27 00 9d 01 00 00 48 00 00 00 47 00
	48 00 00 00 47 00 ba 01 c6 01 42 00 04 00 00 00
	cc 01 0d 00 07 00 03 00 9c 01 00 00 0c 00 07 00
	d4 01 00 00 01 00 08 00 9e 01 08 00 02 00 01 00
	c9 01 09 00 09 00 02 00 a5 01 02 00 07 00 09 00
	00 00 07 00 54 00 00 00 00 00 00 00 10 00 0b 49

Con base en los resultados obtenidos en este experimento se crea la Tabla 3.5, donde manteniendo como base las tres etapas en el manejo del software SCD: establecer conexión, configuración y visualización de las distancias; se registra en una tercera columna en cuales eventos hubo transmisión de datos al seguir los nueve pasos descritos en la Tabla 3.2 de este experimento.

Tabla 3.5. Etapas del software SCD e identificación de los eventos en los que se registra transmisión de datos.

ETAPAS	EVENTOS	TRANSFERENCIA DE DATOS
1. Establecer conexión	1. Escoger el puerto serial.	Sí
	2. Creación del proyecto e identificación de la conexión.	Sí
	3. Lectura de la última configuración almacenada.	Sí
2. Configuración	4. Selección del usuario.	No
	5. Iniciar configuración del proyecto.	No
	6. Iniciar la transferencia de la configuración.	Sí
	7. Reconocimiento y transferencia de la configuración.	Sí
	8. Cambios en las zonas de advertencia y de protección por medio del editor de campos.	No
3. Visualización de las distancias.		Sí

El software serial_port_monitoring permite ver los datos que se transmiten a través del puerto serial, pero no se sabe exactamente cuáles son las tramas enviadas por el software SCD, las respuestas del escáner láser y cuáles son las cadenas de datos que coloca el mismo serial_port_monitoring para comunicaciones internas del software.

Por lo tanto con el experimento 2, se busca deducir los datos enviados por el software SCD hacia el escáner y en el experimento 3 se busca deducir los datos de respuesta del escáner láser hacia el SCD. Cabe resaltar que en los experimentos 2 y 3 se utilizan los eventos donde hubo transmisión de datos, obtenidos en el experimento 1. Ver Tabla 3.5.

3.2.2 Experimento 2

Este experimeto se crea para determinar los datos transmitidos desde SCD al escáner láser; siguiendo los eventos de cada una de las etapas, para los que hubo transmisión de datos, ver Tabla 3.5 obtenida en el experimento 1. Para determinar los datos transmitido, se realiza el montaje de los equipos como se observa en la Figura 3.15, donde PC2 es el encargado por medio de la herramienta Terminal, de escuchar los datos transmitidos.

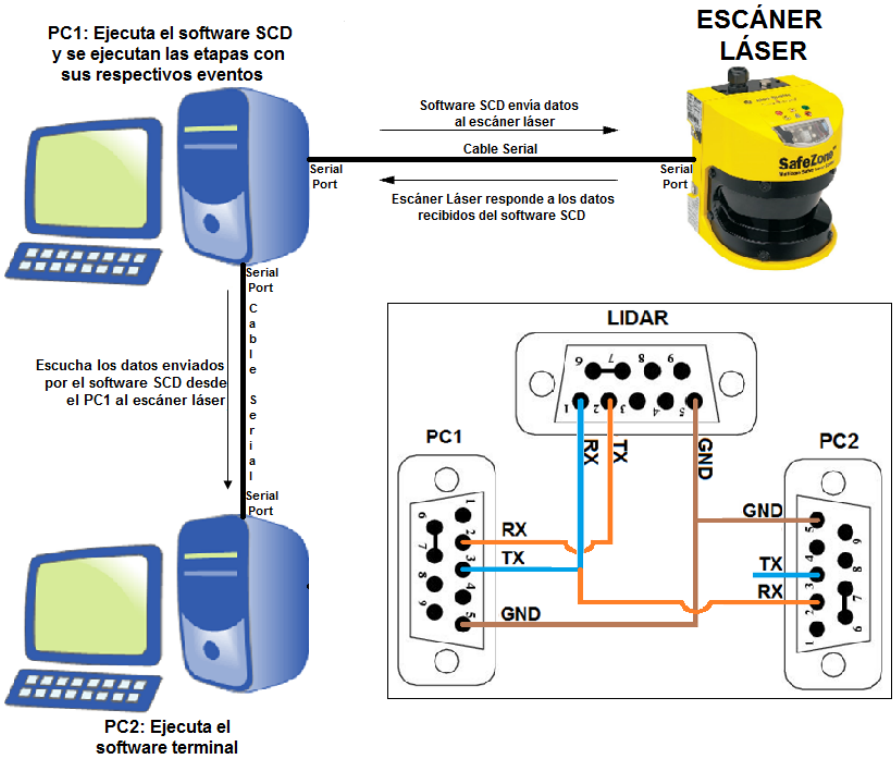


Figura 3.15. Montaje de los equipos usados para el experimento 2.

En este experimento la conexión de los conectores DB9 en la protoboard se realiza de la siguiente manera: Tx de PC1 se conecta a Rx del escáner láser, Rx de PC1 a Tx del escáner láser, Rx de PC2 a Tx de PC1 y Tx de PC2 queda desconectado porque este solo va a escuchar lo que se transmite sin dar respuesta alguna. Ver detalles de conexión en la Figura 3.15.

Una vez realizada la conexión a nivel de hardware y al ejecutar los software que se usan en cada uno de los computadores, en el PC1 con el software SCD se procede a realizar las tres etapas con sus respectivos eventos en los que hubo transmisión de datos en el experimento 1 (ver Tabla 3.5) y se van almacenando los datos registrados en la herramienta Terminal para cada evento por medio del PC2.

Para el experimento 2 se realizan los pasos de la Tabla 3.6, correspondientes a los eventos en los que hubo transmisión de datos; al ejecutar los pasos se guardan los datos obtenidos de la herramienta Terminal.

Tabla 3.6. Pasos realizados en el experimento 2 para cada uno de los eventos.

PASOS	EVENTO REALIZADO EN EL SOFTWARE SCD
1	Escoger el puerto serial.
2	Creación del proyecto e identificación de la conexión.
3	Lectura de la última configuración almacenada.
4	Iniciar transferencia de la configuración.
5	Reconocimiento y transferencia de la configuración.
6	Visualización de las distancias.

Al realizar cada uno de los pasos de la Tabla 3.6 se tiene como resultado que en los pasos 1 y 4 no hay transmisión de datos, a pesar de que en el experimento 1 hubo registro de datos, en los pasos 2, 3, 5 y 6 se tiene transmición de datos, los datos obtenidos se pueden consultar en el Anexo 2: “Resultados de los experimentos”.

Con este experimento se obtienen los datos enviados desde el SCD al escáner láser, estos datos se han logrado separar de todo el protocolo de comunicación usando la conexión mostrada en la Figura 3.15 y ejecutando la herramienta Terminal en el PC2. Para los pasos en los que no se obtuvo datos en este experimento se concluye que los datos que aparecían en el experimento 1 son los que usa internamente el software serial_port_monitoring.

En el experimento 2 solo se obtienen los datos enviados del SCD al escáner láser, para determinar la respuesta del escáner láser frente a cada uno de los pasos realizados en este experimento, se realiza el experimento 3.

3.2.3 Experimento 3

Este experimento busca determinar los datos de respuesta del escáner láser al SCD, siguiendo cada uno de los eventos donde hubo transmisión de datos, ver Tabla 3.5 obtenida en el experimento 1. Para determinar los datos de respuesta del escáner láser al SCD, se realiza el montaje de los equipos como se observa en la Figura 3.16, donde PC2 es el encargado de escuchar los datos de respuesta del escáner láser usando la herramienta Terminal.

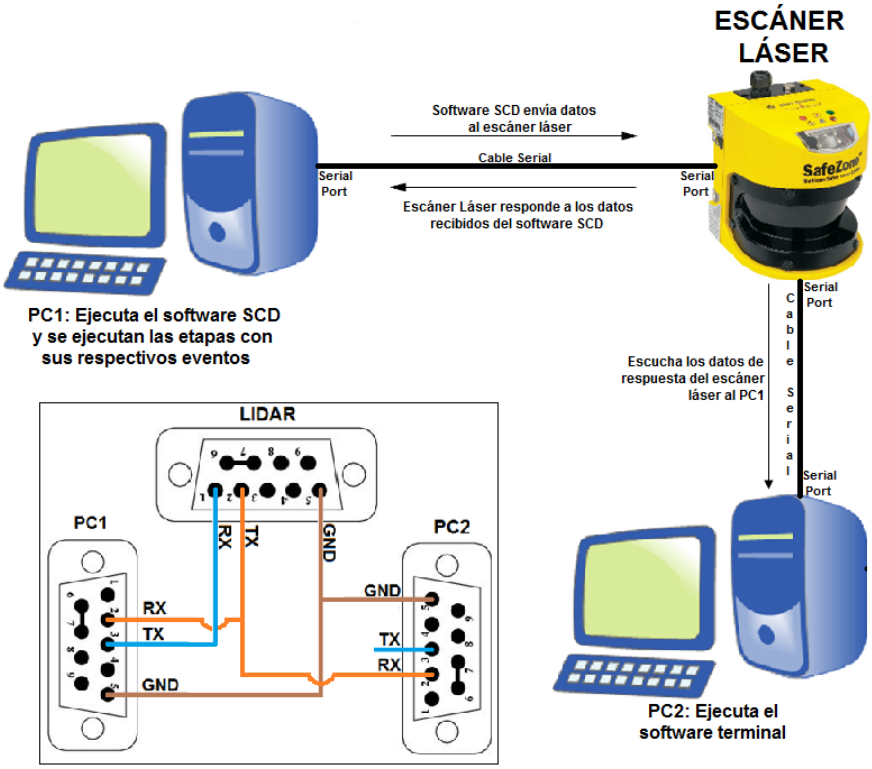


Figura 3.16. Montaje de los equipos usados para el experimento 3.

En esta prueba la conexión de los conectores DB9 en la protoboard se realiza de la siguiente manera: Tx de PC1 se conecta a Rx del escáner láser, Rx de PC1 a Tx del escáner láser, Rx de PC2 a Tx del escáner láser y Tx de PC2 queda desconectado porque este solo va a escuchar las respuestas del escáner láser. Ver detalle de conexiones en la Figura 3.16.

Una vez realizada la conexión a nivel de hardware y al ejecutar los software que se usan en cada uno de los computadores, en el PC1 con el software SCD se realizan las tres etapas con sus respectivos eventos en los que hubo transmisión de datos en el experimento 1 (ver Tabla 3.5), y se almacenan los datos registrados para cada evento por medio del PC2 que ejecuta la herramienta Terminal.
Para el experimento 3 se realizan los pasos de la Tabla 3.7, correspondientes a los eventos en los que hubo transmisión de datos; al ejecutar los pasos se guardan los datos obtenidos de la herramienta Terminal.

Tabla 3.7. Pasos realizados en el experimento 3 para cada uno de los eventos.

PASOS	EVENTO REALIZADO EN EL SOFTWARE SCD
1	Escoger el puerto serial.
2	Creación del proyecto e identificación de la conexión.

3	Lectura de la última configuración almacenada.
4	Iniciar transferencia de la configuración.
5	Reconocimiento y transferencia de la configuración.
6	Visualización de las distancias.

Al realizar cada uno de los pasos de la Tabla 3.7 se tiene como resultado que en los pasos 1 y 4 no hay ningún tipo de respuesta por parte del escáner láser, a pesar de que en el experimento 1 hubo registro de datos, en los pasos 2, 3, 5 y 6 se tiene respuesta del escáner láser, los datos obtenidos se pueden consultar en el Anexo 2: “Resultados de los experimentos”.

Con este experimento se obtienen los datos de respuesta del escáner láser al SCD, estos datos se han logrado separar de todo el protocolo de comunicación usando la conexión mostrada en la Figura 3.16 y ejecutando la herramienta Terminal en el PC2. Para los pasos en los que no se obtuvo datos en este experimento se concluye que los datos que aparecían en el experimento 1 son los que usa internamente el software serial_port_monitoring.

Finalmente basándose en los resultados obtenidos en este experimento se crea la Tabla 3.8, en la que se consignan los eventos en los que finalmente si hubo transmisión de datos, tanto en el experimento 2 como en el experimento 3.

Tabla 3.8. Eventos en los que hay transmisión de datos.

EVENTOS DONDE EXISTE TRANSMISIÓN DE DATOS
Creación del proyecto e identificación de la conexión.
Lectura de la última configuración almacenada.
Reconocimiento de la configuración y transferencia de la misma.
Visualización de las distancias.

3.2.4 Experimento 4

En este experimento se busca emular el software SCD por medio del software Hercules, usando los datos obtenidos en los experimentos 2 y 3, para cada uno de los eventos donde hubo transmisión de datos (ver Tabla 3.8), con el fin de determinar la composición de las tramas intercambiadas.

Para la emulación del software SCD, se envían los resultados obtenidos en el experimento 2 (generados por el software SCD) por medio del software Hércules hacia el escáner laser, con el fin de comparar las respuestas del escáner láser con los resultados del experimento 3 (generados por el escáner laser).

El montaje de los equipos para este experimento se observa en la Figura 3.17, donde el PC1 ejecuta el software Hercules que va a enviar los datos obtenidos del experimento 2 desde PC1 al escáner láser y en PC2 se ejecuta la herramienta Terminal para poder recibir la respuesta del escáner láser frente a lo enviado por PC1. De esta manera, con los datos obtenidos en este experimento y con los datos del experimento 3, se realiza una comparación y se buscan indicios para poder determinar la composición de las tramas de comunicación para poder independizar el escáner láser del software SCD.

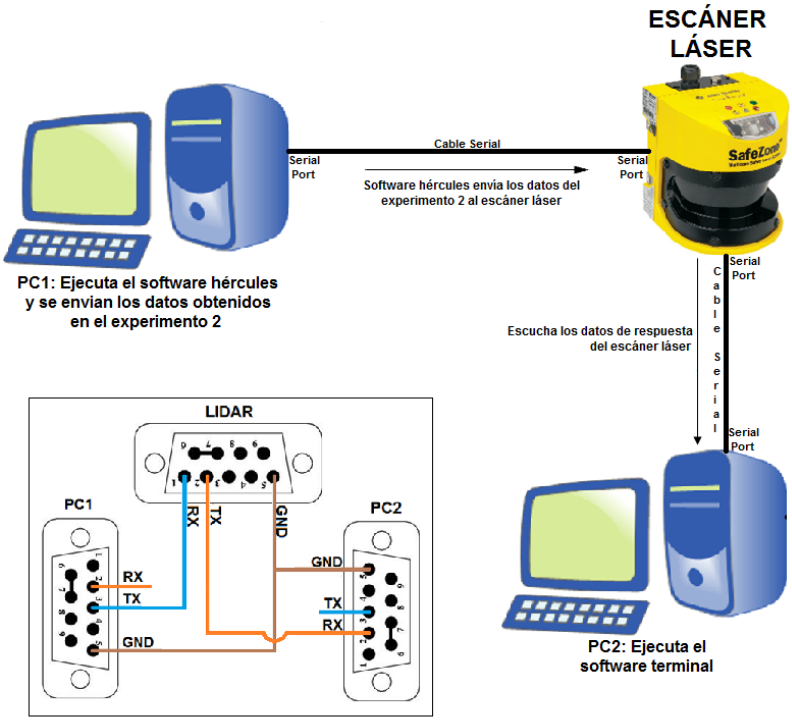


Figura 3.17. Montaje de los equipos usados para el experimento 4.

En este experimento la conexión de los conectores DB9 en la protoboard se realiza de la siguiente manera: Tx de PC1 se conecta a Rx del escáner, Rx de PC2 a Tx del escáner láser, Rx de PC1 queda desconectado porque solo se va a encargar de enviar las tramas y Tx de PC2 queda desconectado porque este solo va a escuchar las respuestas del escáner láser. Ver detalle de conexiones en la Figura 3.17.

Para el experimento 4 se realizan los pasos de la Tabla 3.9, donde se envían los datos obtenidos en el experimento 2 al escáner láser, para cada uno de los eventos en los que hubo registro de transmisión de datos en ese experimento.

Tabla 3.9. Pasos realizados en el experimento 4 para enviar los datos hexadecimales para cada uno de los eventos de la Tabla 3.8.

PASOS	DATOS ENVIADOS EN LA EMULACIÓN
1	Evento: Creación del proyecto e identificación de la conexión.
	Datos: 00 00 45 44 03 00 00 05 FF 0F 00 00 41 44 19 00 00 05 FF 07 19 00 00 05 FF 07 07 0F 9F D0 00 00 45 44 03 00 00 05 FF 0E 00 00 45 44 03 00 00 05 FF 01 00 00 45 44 03 00 00 05 FF 02 00 00 45 44 03 00 00 05 FF 03 00 00 45 44 03 00 00 05 FF 04 00 00 45 44 03 00 00 05 FF 05 00 00 45 44 03 00 00 05 FF 06 00 00 45 44 03 00 00 05 FF 07 00 00 45 44 03 00 00 05 FF 08 00 00 45 44 03 00 00 05 FF 09 00 00 45 44 03 00 00 05 FF 07 00 00 45 44 09 00 00 08 FF 07 00 00 45 44 0A 00 00 09 FF 07 00 00 45 44 0B 00 00 56 FF 07 00 00 45 44 74 00 00 66 FF 07 00 00 45 44 16 00 00 10 FF 07 00 00 45 44 03 00 00 05 FF 07 00 00 45 44 09 00 00 08 FF 07 00 00 45 44 0A 00 00 09 FF 07 00 00 45 44 0B 00 00 56 FF 07 00 00 45 44 74 00 00 66 FF 07 00 00 45 44 16 00 00 10 FF 07 00 00 45 44 0B 00 00 56 FF 07
2	Evento: Lectura de la última configuración almacenada.
	00 00 45 44 05 00 00 1F FF 07 00 00 45 44 1A 00 00 10 FF 07 00 00 45 44 33 00 00 0E FF 07 00 00 45 44 15 00 00 0E FF 07 00 00 45 44 14 00 00 07 FF 07 00 00 45 44 17 00 00 0F FF 07 00 00 45 44 1B 00 00 11 FF 07 00 00 45 44 2B 00 06 01 FF 07 00 00 45 44 15 00 00 0E FF 07
3	Reconocimiento y transferencia de la configuración.
	00 00 41 44 0E 00 00 09 FF 07 0E 00 00 09 FF 07 03 00 0C B1 D8 A9 55 7B 54 0E F3 E2 00 00 41 44 0E 00 00 09 FF 07 0E 00 00 09 FF 07 00 00 30 30 30 30 30 30 30 30 FB 07 00 00 45 44 19 00 00 05 FF 07 00 00 41 44 19 00 00 05 FF 07 19 00 00 05 FF 07 00 00 E7 B8
4	Visualización de las distancias.
	00 00 41 44 19 00 00 05 FF 07 19 00 00 05 FF 07 07 0F 9F D0 00 00 45 44 15 00 00 0E FF 07 00 00 45 44 19 00 00 05 FF 07 00 00 41 44 19 00 00 05 FF 07 19 00 00 05 FF 07 00 00 E7 B8 00 00 41 44 19 00 00 05 FF 07 19 00 00 05 FF 07 07 0F 9F D0 00 00 45 44 0B 00 00 56 FF 07 00 00 45 44 38 00 00 24 FF 07 00 00 45 44 0C 00 02 FE FF 07 00 00 45 44 0B 00 00 56 FF 07 00 00 45 44 38 00 00 24 FF 07 00 00 45 44 0C 00 02 FE FF 07 00 00 45 44 0B 00 00 56 FF 07 00 00 45 44 38 00 00 24 FF 07 00 00 45 44 0C 00 02 FE FF 07 00 00 45 44 0B 00 00 56 FF 07 00 00 45 44 38 00 00 24 FF 07 00 00 45 44 0C 00 02 FE FF 07 00 00 45 44 0B 00 00 56 FF 07 00 00 45 44 38 00 00 24 FF 07 00 00 45 44 0C 00 02 FE FF 07 00 00 45 44 0B 00 00 56 FF 07 00 00 45 44 38 00 00 24 FF 07 00 00 45 44 0C 00 02 FE FF 07 00 00 45 44 0B 00 00 56 FF 07 00 00 45 44 38 00 00 24 FF 07 00 00 45 44 0C 00 02 FE FF 07 00 00 45 44 0B 00 00 56 FF 07 00 00 45 44 38 00 00 24 FF 07 00 00 45 44 0C 00 02 FE FF 07 00 00 45 44 0B 00 00 56 FF 07 00 00 45 44 38 00 00 24

En cada uno de los pasos de la Tabla 3.9 se envían los datos hexadecimales emulando el software SCD, y se espera una respuesta de parte del escáner, con el fin de comparar estos resultados con el experimento 3 y de esta forma identificar las tramas de datos de la comunicación.

Para este experimento se obtienen los siguientes resultados:

- En el paso 1, al enviar los datos hexadecimales obtenidos en el experimento 2 para el evento: Creación del proyecto e identificación de la conexión, se obtiene como respuesta del escáner láser un byte de valor ‘16’.

- En el paso 2, al enviar los datos hexadecimales obtenidos en el experimento 2 para el evento: Lectura de la última configuración almacenada, se obtiene como respuesta del escáner láser un byte de valor ‘16’.

- En el paso 3, al enviar los datos hexadecimales obtenidos en el experimento 2 para el evento: Reconocimiento y transferencia de la configuración, se obtiene como respuesta del escáner láser un byte de valor ‘16’.

- En el paso 4, al enviar los datos hexadecimales obtenidos en el experimento 2 para el evento: Visualización de las distancias, se obtiene como respuesta del escáner láser un byte de valor ‘16’.

Interpretación y análisis de los resultados del experimento 4

Al revisar estos primeros resultados obtenidos se observa que la respuesta por parte del escáner no es la esperada según los resultados obtenidos en el experimento 3 para cada uno de los eventos.

Observando detenidamente los conjuntos de datos hexadecimales enviados en los cuatro eventos (ver Tabla 3.9), se observa que cada cierta cantidad de bytes se repiten los valores ’00 00’, seguidos de los valores ’41 44’ o ’45 44’. En el caso de los datos que inician con ’00 00 41 44’ le sigue un conjunto de diez y seis (16) bytes y para los datos que inician con ’00 00 45 44’ le siguen un conjunto de seis (6) bytes; de esta forma los datos hexadecimales de la Tabla 3.22, en cada uno de los eventos, se componen de una serie de cadenas a veces de 20 bytes otras de 10 bytes, iniciadas por uno de estos dos marcadores: ‘00 00 41 44’ o ‘00 00 45 44’, respectivamente.

Teniendo en cuenta lo anterior se realiza una segunda fase del experimento 4, pero esta vez solo con el conjunto de datos obtenidos del paso 4 en el evento: Visualización de las distancias, pero en lugar de enviar el conjunto de datos completos, se hace en forma parcial enviando cada una de las cadenas en forma independiente que conforman este conjunto de datos y analizando la respuesta del escáner, con el objetivo de constatar si estas cadenas identificadas determinan una secuencia de comunicación entre el SCD y el escáner laser.

Se observa una serie de cadenas que conforman el conjunto de datos hexadecimales mostrados en la Tabla 3.10, que corresponden al evento Visualización de las distancias, donde de color azul aparecen las cadenas de 10 bytes y en amarillo las cadenas de 20 bytes y, al detallar la organización de todas las cadenas se observa como aparecen tres (3) cadenas de 20 bytes y el resto de

10 bytes. Sin embargo las cadenas cambian de valores hasta la octava, desde la novena aparece una secuencia de tres cadenas de 10 bytes en continua repetición que corresponde con las cadenas 6, 7 y 8, como se observan las tonalidades de verde marcadas en la Tabla 3.10.

Tabla 3.10. Serie de cadenas identificadas del conjunto de datos del evento Visualización de las distancias.

PASO	DATOS ENVIADOS EN LA VISUALIZACION DE LAS DISTANCIAS
4	Evento: Visualización de las distancias.
	Datos: 00 00 41 44 19 00 00 05 FF 07 19 00 00 05 FF 07 07 0F 9F D0 00 00 45 44 15 00 00 0E FF 07 00 00 45 44 19 00 00 05 FF 07 00 00 41 44 19 00 00 05 FF 07 19 00 00 05 FF 07 00 00 E7 B8 00 00 41 44 19 00 00 05 FF 07 19 00 00 05 FF 07 07 0F 9F D0 00 00 45 44 0B 00 00 56 FF 07 00 00 45 44 38 00 00 24 FF 07 00 00 45 44 0C 00 02 FE FF 07 00 00 45 44 0B 00 00 56 FF 07 00 00 45 44 38 00 00 24 FF 07 00 00 45 44 0C 00 02 FE FF 07 00 00 45 44 0B 00 00 56 FF 07 00 00 45 44 38 00 00 24 FF 07 00 00 45 44 0C 00 02 FE FF 07 00 00 45 44 0B 00 00 56 FF 07 00 00 45 44 38 00 00 24 FF 07 00 00 45 44 0C 00 02 FE FF 07 00 00 45 44 0B 00 00 56 FF 07 00 00 45 44 38 00 00 24 FF 07 00 00 45 44 0C 00 02 FE FF 07 00 00 45 44 0B 00 00 56 FF 07 00 00 45 44 38 00 00 24 FF 07 00 00 45 44 0C 00 02 FE FF 07 00 00 45 44 0B 00 00 56 FF 07 00 00 45 44 38 00 00 24 FF 07 00 00 45 44 0C 00 02 FE FF 07 00 00 45 44 0B 00 00 56 FF 07 00 00 45 44 38 00 00 24 FF 07 00 00 45 44 0C 00 02 FE FF 07 00 00 45 44 0B 00 00 56 FF 07 00 00 45 44 38 00 00 24 FF 07 00 00 45 44 0C 00 02 FE FF 07 00 00 45 44 0B 00 00 56 FF 07 00 00 45 44 38 00 00 24 FF 07 00 00 45 44 0C 00 02 FE FF 07 00 00 45 44 0B 00 00 56 FF 07 00 00 45 44 38 00 00 24 FF 07 00 00 45 44 0C 00 02 FE FF 07 00 00 45 44 0B 00 00 56 FF 07 00 00 45 44 38 00 00 24 FF 07 00 00 45 44 0C 00 02 FE FF 07 00 00 45 44 0B 00 00 56 FF 07 00 00 45 44 38 00 00 24 FF 07 00 00 45 44 0C 00 02 FE FF 07 00 00 45 44 0B 00 00 56 FF 07 00 00 45 44 38 00 00 24

Para realizar la segunda fase del experimento 4 se siguen los siguientes pasos:

- En el paso 5, se envía la primera cadena de datos hexadecimales de 20 bytes por medio del software Hercules hacia el escáner, que se muestran en la Tabla 3.11, donde se ubica en un recuadro los cuatro bytes (00 00 41 44) que hacen de marcador de las cadenas de 20 bytes.

Tabla 3.11. Primer cadena de datos hexadecimales de 20 bytes enviada.

00 00 41 44 19 00 00 05 FF 07 19 00 00 05 FF 07 07 0F 9F D0

Enviando esta primera cadena de datos hexadecimales se tiene como respuesta del escáner láser en el PC2 los datos de la Tabla 3.12.

Tabla 3.12. Datos hexadecimales de respuesta del escáner láser a la primera cadena.

00 00 00 00

Comparando los datos hexadecimales de respuesta del escáner láser con los datos obtenidos en el experimento 3 en la etapa de Visualización de las distancias, se observa que la respuesta obtenida con la primera cadena de datos, se corresponde con los cuatro primeros bytes de los datos hexadecimales de respuesta del experimento 3. Esta comparación se muestra

en la Figura 3.18, donde en el lado izquierdo se encuentra la imagen de los datos recibidos al enviar la primera cadena de datos hexadecimales obtenida en la visualización de las distancias y al lado derecho resaltado con un cuadro rojo los resultados obtenidos en el experimento 3 para este mismo evento.

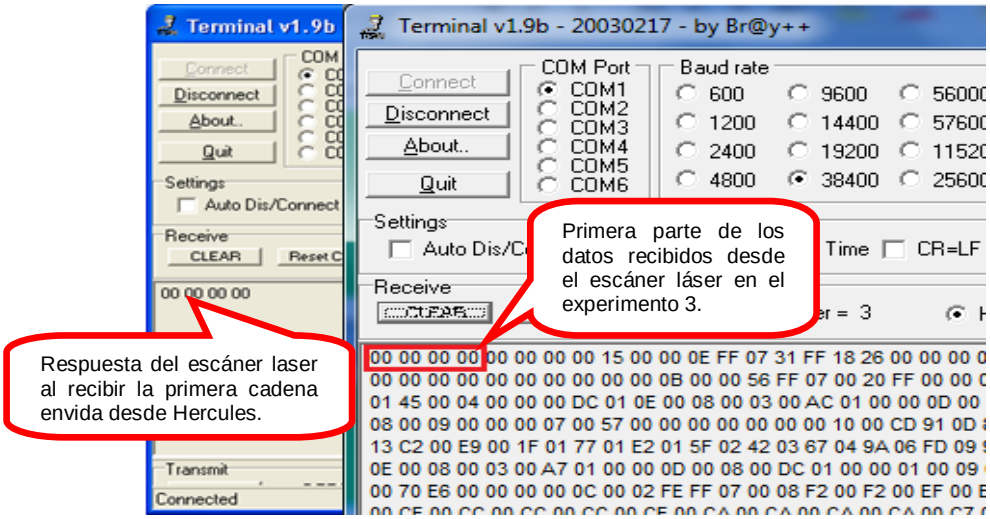


Figura 3.18. Comparación de los datos recibidos al enviar la primera cadena de datos hexadecimales de 20 bytes y los datos recibidos desde el escáner láser en el experimento 3 en el evento de visualización de las distancias.

- En el paso 6, se envía la segunda cadena de datos hexadecimales de 10 bytes desde la herramienta Hercules hacia el escáner, estos datos se muestran en la Tabla 3.13, donde se ubica en un recuadro los cuatro bytes (00 00 45 44) que hacen de marcador de las cadenas de 10 bytes.

Tabla 3.13. Segunda cadena de datos hexadecimales de 10 bytes enviada.

00 00 45 44 15 00 00 0E FF 07

Enviando esta segunda cadena de datos hexadecimales se tiene como respuesta del escáner láser en el PC2 los datos de la Tabla 3.14.

Tabla 3.14. Datos hexadecimales de respuesta del escáner láser a la segunda cadena.

00 00 00 00 15 00 00 0E FF 07 31 FF 18 26 00 00 00 00 **00 00 00 00 00 00 00 00 00 00 00 00 5E BE**

Comparando los datos hexadecimales de respuesta del escáner láser con los datos obtenidos en el experimento 3 en la etapa de visualización de las distancias, la respuesta obtenida con esta segunda cadena de datos se corresponden con los treinta y dos (32) bytes contados a partir de los cuatro primeros bytes de los datos

hexadecimales de respuesta en el experimento 3. Esta comparación se muestra en la Figura 3.19, donde en el lado izquierdo se encuentra la imagen de los 32 datos recibidos al enviar la segunda cadena de datos hexadecimales obtenida en la visualización de las distancias y al lado derecho resaltado con un cuadro rojo los resultados obtenidos en el experimento 3 para este mismo evento.

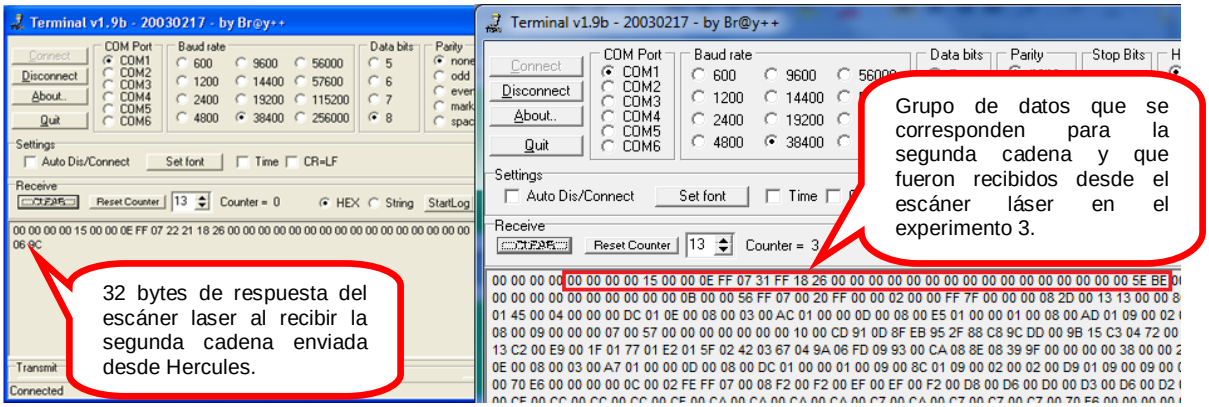


Figura 3.19. Comparación de los datos recibidos al enviar la segunda cadena de datos hexadecimales y los datos recibidos desde el escáner láser en el experimento 3: en la etapa visualización de las distancias.

- En el paso 7, se envía la tercera cadena de datos hexadecimales de la Tabla 3.15 por medio del software Hercules.

Tabla 3.15. Tercera cadena de datos hexadecimales de 10 bytes enviada.

00 00 45 44 19 00 00 05 FF 07

Enviando esta tercera cadena de datos hexadecimales se tiene como respuesta del escáner láser en el PC2 los datos de la Tabla 3.16.

Tabla 3.16. Datos hexadecimales de respuesta del escáner láser a la tercera cadena.

00 00 00 00 19 00 00 05 FF 07 07 0F 9F D0 00

Al comparar los datos hexadecimales de respuesta del escáner láser con los datos obtenidos en el experimento 3 en la etapa de visualización de las distancias, la respuesta obtenida con esta tercer cadena de datos se corresponde con los siguientes quince (15) bytes contados a partir del byte 36 de los datos hexadecimales de respuesta del experimento 3. La comparación se muestra en la Figura 3.20, donde en el lado izquierdo se encuentra la imagen de los datos recibidos al enviar la tercera cadena de datos hexadecimales obtenida en la visualización de las distancias y al lado derecho resaltado con un

cuadro rojo los resultados obtenidos en el experimento 3 para este mismo evento.

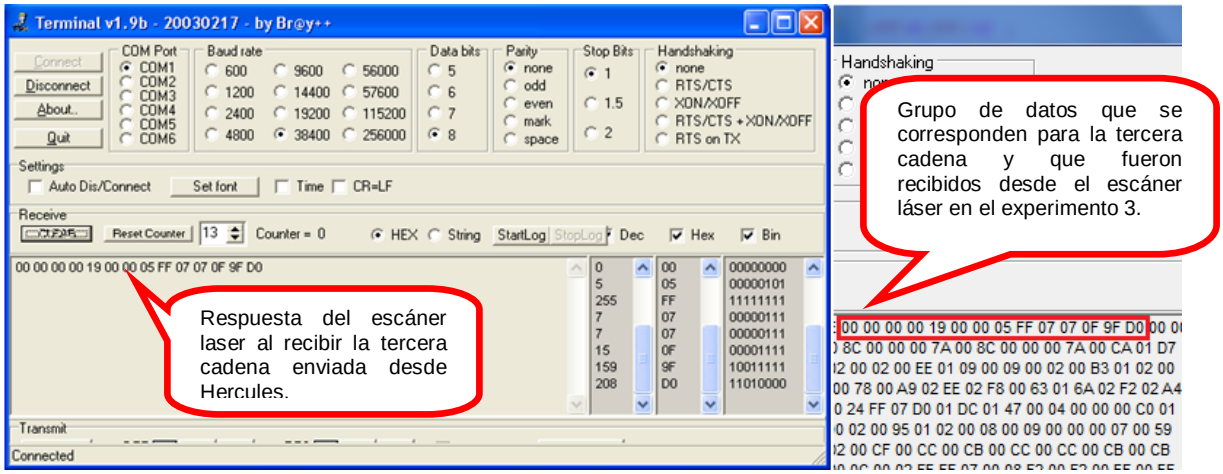


Figura 3.20. Comparación de los datos recibidos al enviar la tercera cadena de datos hexadecimales y los datos recibidos desde el escáner láser en el experimento 3, en la etapa visualización de las distancias.

En el experimento 4, fase 2, se realizó hasta el paso 15, correspondiente a la cadena número 11. Confirmando que para cada cadena enviada, corresponde un grupo de bytes en el conjunto de datos hexadecimales del experimento 3. Ver en la Tabla 3.17 la longitud de bytes correspondiente a cada una de las once cadenas enviadas. A partir de la cadena 9 se confirma que las cadenas 6, 7 y 8 son periódicas. (Ver Anexo 2: “Resultados de los experimentos.”)

Tabla 3.17. Cadenas enviadas por Hercules y longitud de las cadenas respuestas del escáner láser en el evento Visualización de las distancias.

CADENA	CADENA ENVIADA POR EL SOFTWARE HERCULES	LONG. CADENA RECIBIDA DEL ESCÁNER LÁSER (BYTES)
1	00 00 41 44 19 00 00 05 FF 07 19 00 00 05 FF 07 07 0F 9F D0	4
2	00 00 45 44 15 00 00 0E FF 07	32
3	00 00 45 44 19 00 00 05 FF 07	15
4	00 00 41 44 19 00 00 05 FF 07 19 00 00 05 FF 07 00 00 E7 B8	4
5	00 00 41 44 19 00 00 05 FF 07 19 00 00 05 FF 07 07 0F 9F D0	4
6	00 00 45 44 0B 00 00 56 FF 07	176

7	00 00 45 44 38 00 00 24 FF 07	76
8	00 00 45 44 0C 00 02 FE FF 07	1536
9	00 00 45 44 0B 00 00 56 FF 07	176
10	00 00 45 44 38 00 00 24 FF 07	76
11	00 00 45 44 0C 00 02 FE FF 07	1536

Al observar la Tabla 3.17 y los resultados de la fase 2, del experimento 4, se concluye que efectivamente las cadenas identificadas constituyen una secuencia de comunicación coordinada entre el SCD y el escáner laser, lo cual indica que para cada cadena enviada por el SCD le corresponde una cadena de respuesta del escáner. En particular se observa que cuando la cadena enviada por el SCD consta de veinte (20) bytes, la respuesta del escáner láser es de cuatro (4) bytes que corresponden a ’00 00 00 00’.

Finalmente se puede ver que existe una periodicidad en el tamaño de los datos enviados por el escáner cuando la secuencia de datos recibidos corresponde a las cadenas 6, 7 y 8, mientras que las cinco (5) primeras cadenas no presentan esta situación.

Se realiza una tercera fase del experimento 4, con el objetivo de deducir el orden o la secuencia mínima o exacta de las cadenas enviadas desde el SCD, para enganchar el escáner láser, garantizando una buena comunicación serial, y de esta manera identificar en que parte de la secuencia se reciben las distancias entregadas por el escáner.

Partiendo de los resultados de las observaciones de la segunda fase del experimento 4, se trabaja sólo con las primeras ocho (8) cadenas enviadas por el SCD, y no se incluyen las cadenas 9, 10 y 11 que son las mismas 6, 7, y 8. (Ver Tabla 3.17)

Para realizar la tercera fase del experimento 4, se realizan una serie de pruebas, que consisten en enviar cualquiera de las siete (7) primeras cadenas del SCD, excluyendo la cadena 1, con el fin de determinar si se obtiene o no la respuesta esperada del escáner laser. Después se envía la primera cadena y posteriormente cualquiera de las seis restantes, excluyendo la cadena dos, y así sucesivamente hasta determinar la secuencia correcta. Para ello se desarrollan los siguientes tres macro pasos:

- Macro paso 16, se envía por medio del software Hercules la cadena 6 de la Tabla 3.17, sin enviar las cadenas anteriores y se obtiene como respuesta del escáner láser los datos de la Tabla 3.18.

Tabla 3.18. Respuesta obtenida del escáner láser, al enviar la cadena 6 sin enviar las 5 cadenas anteriores, macro paso 16.

00 00 00 0a

Observando la respuesta obtenida del escáner láser y comparándola con la obtenida en la fase 2 se observa que son diferentes (ver Tabla 3.19).

Tabla 3.19. Comparación de los resultados Recibidos vs resultados obtenidos como respuesta del escáner láser macro paso 16.

DATOS RECIBIDOS	DATOS ESPERADOS
00 00 00 0a	00 00 00 00 0B 00 00 56 FF 07 00 20 FF 00 00 02 00 00 FF 7F 00 00 00 08 2D 00 13 13 00 00 8C 00 00 00 7A 00 8C 00 00 00 7A 00 CA 01 D7 01 45 00 04 00 00 00 DC 01 0E 00 08 00 03 00 AC 01 00 00 0D 00 08 00 E5 01 00 00 01 00 08 00 AD 01 09 00 02 00 02 00 EE 01 09 00 09 00 02 00 B3 01 02 00 08 00 09 00 00 00 07 00 57 00 00 00 00 00 00 00 10 00 CD 91 0D 8F EB 95 2F 88 C8 9C DD 00 9B 15 C3 04 72 00 78 00 A9 02 EE 02 F8 00 63 01 6A 02 F2 02 A4 13 C2 00 E9 00 1F 01 77 01 E2 01 5F 02 42 03 67 04 9A 06 FD 09 93 00 CA 08 8E 08 39 9F

Se hacen los seis intentos restantes (cadenas 2, 3, 4, 5, 7, y 8) tratando de obtener alguna respuesta por parte del escáner láser igual o parecida a las obtenidas en la fase 2, pero siempre se obtiene la misma respuesta ’00 00 00 0a’.

- En el macro paso 17, se envía por medio del software Hercules la cadena 6 de la Tabla 3.17, enviando previamente la cadena 1, pero sin enviar las cadenas anteriores a la seis y se obtiene como respuesta del escáner láser los datos de la Tabla 3.20.

Tabla 3.20. Respuesta obtenida del escáner láser, al enviar la cadena 6, previamente enviando la 1, pero sin enviar las 4 cadenas anteriores a la sexta, macro paso 17.

00 00 00 0a

Observando la respuesta obtenida del escáner láser y comparándola con la obtenida en la fase 2 se observa que son diferentes (ver Tabla 3.21).

Tabla 3.21. Comparación de los resultados Recibidos vs resultados obtenidos como respuesta del escáner láser, macro paso 17.

DATOS RECIBIDOS	DATOS ESPERADOS
00 00 00 0a	00 00 00 00 0B 00 00 56 FF 07 00 20 FF 00 00 02 00 00 FF 7F 00 00 00 08 2D 00 13 13 00 00 8C 00 00 00 7A 00 8C 00 00 00 7A 00 CA 01 D7 01 45 00 04 00 00 00 DC 01 0E 00 08 00 03 00 AC 01 00 00 0D 00 08 00 E5 01 00 00 01 00 08 00 AD 01 09 00 02 00 02 00 EE 01 09 00 09 00 02 00 B3 01 02 00 08 00 09 00 00 00 07 00 57 00 00 00 00 00 00 00 10 00 CD 91 0D 8F EB 95 2F 88 C8 9C DD 00 9B 15 C3 04 72 00 78 00 A9 02 EE 02 F8 00 63 01 6A 02 F2 02 A4 13 C2 00 E9 00 1F 01 77 01 E2 01 5F 02 42 03 67 04 9A 06 FD 09 93 00 CA 08 8E 08 39 9F

Se hacen los cinco intentos restantes (cadenas 3, 4, 5, 7, y 8) tratando de obtener alguna respuesta por parte del escáner láser igual o parecida a las obtenidas en la fase 2, pero siempre se obtiene la misma respuesta ’00 00 00 0a’.

Este procedimiento iterativo, de los macro pasos 16 y 17, se realiza sucesivamente hasta llegar al punto donde enviando las seis primeras cadenas en orden y enseguida la cadena 8, y no se logra obtener datos para esta cadena.

- Macro paso 18, se procede a enviar nuevamente las cadenas de la Tabla 3.17, pero esta vez se envían en orden, iniciando con la cadena 1 hasta la 7, y luego enviando la cadena 8.

  Observando las respuestas del escáner láser al recibir las cadenas de la Tabla 3.17 y comparándolas con los resultados obtenidos en la fase 2 se observa: que esta vez si hay datos para la cadena 8, que para las 5 primeras cadenas se obtienen los mismos resultados tanto en longitud como en el valor de los datos, pero algunos datos para las cadenas 6, 7 y 8 varían conservando como factor común la longitud de los datos recibidos. Ver Anexo 2: “Resultados de los experimentos”.

Al realizar el macro paso 18, se realiza nuevamente el macro paso 16 y se observa en las respuestas obtenidas por parte del escáner láser que, ante el envío de cualquiera de las 8 primeras cadenas, los resultados son los mismos que se obtuvieron en la fase 2, lo que permite concluir que para establecer comunicación con el escáner láser es necesario enviar una sola vez la secuencia de los siete 7 cadenas de datos. Igualmente se hizo la prueba de enviar en repetidas ocasiones la cadena 8, después de las 7 primeras cadenas, y se observa que la respuesta

por parte del escáner al recibir esta cadena siempre es un paquete de datos de 1536 bytes con algunas variaciones en su contenido. Lo que permite plantear como opción que al enviar la cadena 8 se obtiene como respuesta del escáner las distancias.

3.3 IDENTIFICACION DE LAS TRAMAS DE DATOS SCD - ESCANER

Haciendo uso de los resultados obtenidos en el experimento 4 en sus diferentes fases se procede a identificar las tramas de datos tanto las del SCD al escáner láser, como las del escáner láser al SCD.

3.3.1 Identificación trama de datos SCD > Escáner

Para lograr una buena comunicación entre el escáner láser y el computador se identificaron como necesarias las 7 primeras cadenas de datos de la Tabla 3.22.

Tabla 3.22. Cadenas necesarias para lograr una buena comunicación entre el escáner láser y el computador.

CADENAS ENVIADAS HACIA EL ESCÁNER LÁSER	
1	00 00 41 44 19 00 00 05 FF 07 19 00 00 05 FF 07 07 0F 9F D0
2	00 00 45 44 15 00 00 0E FF 07
3	00 00 45 44 19 00 00 05 FF 07
4	00 00 41 44 19 00 00 05 FF 07 19 00 00 05 FF 07 00 00 E7 B8
5	00 00 41 44 19 00 00 05 FF 07 19 00 00 05 FF 07 07 0F 9F D0
6	00 00 45 44 0B 00 00 56 FF 07
7	00 00 45 44 38 00 00 24 FF 07
8	00 00 45 44 0C 00 02 FE FF 07

Con el análisis de los resultados del experimento 4 fase 2 y 3 se concluye que las cadenas de datos se pueden denominar tramas de datos, y por lo tanto se van a tener tramas de datos enviadas y tramas de datos recibidas. Para este caso se muestra la estructura de las tramas de datos enviadas, donde se identificó que hay unas de 20 bytes y otras de 10 bytes.

Para las tramas de datos enviadas hacia el escáner láser de 20 bytes, la estructura se muestra en la Tabla 3.23.

Tabla 3.23. Estructura de las tramas enviadas hacia el escáner láser de 20 bytes.

Marcador	Datos a enviar
00 00 41 44	(16 bytes)

Para las tramas de datos enviadas hacia el escáner láser de 10 bytes, la estructura se muestra en la Tabla 3.24.

Tabla 3.24. Estructura de las tramas enviadas hacia el escáner láser de 10 bytes.

Marcador	Identificador
00 00 45 44	(6 bytes)

Al observar las respuestas obtenidas del escáner láser y comparándolas con la trama previamente enviada, se obtiene que los primeros 4 bytes de la respuesta recibida por parte del escáner láser van a ser '00 00 00 00', seguidos de 6 bytes que corresponde a los últimos bytes de la trama enviada, a estos 6 bytes se denominan: Identificador. Como ejemplo de esta observación en la Tabla 3.25 se usa la trama 2 y 3 con su respectiva respuesta, donde en un recuadro, se puede ver el conjunto de datos denominado identificador.

Tabla 3.25. Comparación entre la trama enviada al escáner láser y los datos recibidos.

Trama enviada	Datos recibidos del escáner láser
00 00 45 44 15 00 00 0E FF 07	00 00 00 00 15 00 00 0E FF 07 31 FF 18 26 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 5E BE
00 00 45 44 19 00 00 05 FF 07	00 00 00 00 19 00 00 05 FF 07 07 0F 9F D0 00

Finalmente en el macro paso 18, se obtiene como resultado que, para establecer conexión entre el escáner láser y el computador, se deben enviar obligatoriamente las primeras 7 tramas y como hipótesis se plantea que con la trama 8 el escáner láser va a entregar como respuesta un conjunto de datos, donde se van a encontrar las distancias medidas, esto basándose en la longitud de los datos recibidos. Teniendo en cuenta los resultados obtenidos y la hipótesis, se construye una trama general que de ahora en adelante se denominará Trama de Enganche del Escáner Láser (TEEL), cuya estructura está compuesta por 2 partes: 1) La cabecera, se compone de las 7 primeras tramas y 2) Solicitud de las distancias, se compone por la trama 8. En la Tabla 3.26, se tiene la TEEL y debe ser envíada en el orden de izquierda a derecha.

Tabla 3.26. Trama de Enganche del Escáner Láser (TEEL)

CABECERA (Bytes)							Solicitud Distancias (Bytes)		
1) 20	**2)10**	**3) 10**	**4) 20**	**5) 20**	**6) 10**	**7) 10**	**8) 10**	**. . .**	**8) 10**
00 00 41 44 19 00 00 05 FF 07 19 00 00 05 FF 07 07 0F 9F D0	00 00 45 44 15 00 00 0E FF 07	00 00 45 44 19 00 00 05 FF 07	00 00 41 44 19 00 00 05 FF 07 19 00 00 05 FF 07 00 00 E7 B8	00 00 41 44 19 00 00 05 FF 07 19 00 00 05 FF 07 07 0F 9F D0	00 00 45 44 0B 00 00 56 FF 07	00 00 45 44 38 00 00 24 FF 07	00 00 45 44 0C 00 02 FE FF 07		00 00 45 44 0C 00 02 FE FF 07

Teniendo en cuenta esta secuencia se obtiene como resultado final una comunicación entre el escáner láser y el SCD.

3.3.2 Identificación trama de datos Escáner > SCD

Una vez analizados las tramas de envío del SCD al escáner, se procede a analizar los datos que se reciben como respuesta del escáner láser, como se tiene la hipótesis que con la octava trama de datos se obtienen las distancias, se realiza una fase 4 del experimento 4 para confirmar dicha hipótesis, observando el comportamiento de los datos que se reciben, modificando la resolución angular del escáner láser y creando entornos con distancias conocidas.

Para la fase 4 del experimento 4, se realizan una serie de pruebas con el fin de confirmar la hipótesis de la trama que entrega las distancias y deducir el orden de los datos en esta trama, cuando recibe la trama encargada de solicitar las distancias. Para poder obtener distancias conocidas, se crea una señalización que permita tener distancias conocidas, las cuales se encuentran en un rango de aproximadamente entre 40 y 60 cm.

Para esta fase 4 se realizan los siguientes pasos:

- En el paso 19 y 20, se configura el escáner láser con una resolución angular de 0,25º y 0.5º respectivamente, y se crea un semicírculo alrededor del escáner láser, ver Figura 3.21., para poder obtener distancias conocidas y poder reconocer exactamente de qué forma está entregando las distancias.

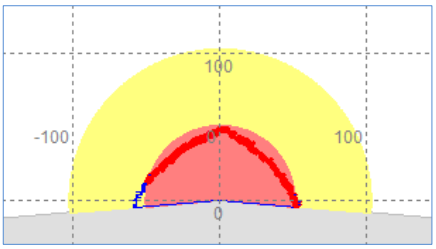


Figura 3.21. Semicírculo creado alrededor del escáner láser.

Este procedimiento se realiza varias veces con y sin el semicírculo, con el fin de observar la variación de los datos. Para ver los resultados obtenidos de estas pruebas consultar Anexo 2: “Resultados de los experimentos”.

Con los resultados obtenidos, se hace una comparación de los conjuntos de datos obtenidos en las pruebas de la fase 4, observando que la longitud de estos datos que envía como respuesta el escáner láser siempre es de 1536 bytes sin importar la resolución angular.

Teniendo en cuenta la Tabla 3.25, se espera que para la resolución angular de 0.25º se reciban 761 distancias y para la resolución angular de 0.5º se reciban 381 distancias, por lo tanto con esta información se descartan las otras respuestas recibidas del escáner láser y se confirma que la respuesta obtenida del escáner láser al recibir la trama 8, es la que entrega el conjunto de datos que contiene las distancias.

Comparando los resultados obtenidos en las diferentes pruebas de esta fase, se observa que los primeros 12 bytes son constantes (ver Tabla 3.27), en esta tabla solo se hace la comparación con los resultados de dos de las pruebas con diferentes objetos para ver el cambio en las distancias, los bytes que aparecen en amarillo es el inicio de los datos de respuesta, en azul el identificador que se relaciona con los 6 bytes final de la trama 8 y de color verde 2 bytes que son constantes en todos los paquetes de datos recibidos en estas pruebas que corresponden a los valores ’00 08’. Para mirar otros datos de respuesta obtenidos ir al Anexo 2: “Resultados de los experimentos”.

Tabla 3.27. Comparación de los primeros 12 bytes en dos de las pruebas realizadas en el experimento 4 fase 4.

Datos recibidos 1	Datos recibidos 2
00 00 00 00 0C 00 02 FE FF 07 00 08 7B 00 7E 00 8E 00 D4 00 D4 00 D6 00	00 00 00 00 0C 00 02 FE FF 07 00 08 36 00 36 00 36 00 36 00 33 40 33 40 36

D6 00 D3 00 D6 00 D4 00 D3 00 D0 00 D2 00 D2 00 D3 00 D3 00 D2 00 D3...	00 33 40 33 40 33 40 33 40 31 40 33 40 33 40 33 40 33 40 31 40 2F 40 32...

Una vez se identifica los 12 primeros bytes como aquellos que se repiten y conociendo las distancias tomadas para cada una de las pruebas, se procede a tratar de identificar cuáles son los datos que corresponden a las distancias. Lo anterior, se realiza pasando los datos hexadecimales a partir del byte 13 al sistema decimal, de esta manera se obtiene una relación con las distancias medias y las distancias de respuesta del escáner, confirmando que con la trama 8 se reciben los datos que contienen las distancias. Sin embargo se debe revisar de qué manera se reciben esas distancias porque la cantidad de bytes no coinciden con los esperados. Al revisar los resultados de varias de las pruebas donde se reciben los datos de las distancias, se observa que el byte 13 corresponde a la primer distancia medida a un ángulo de -5º y de ahí en adelante cada byte de por medio equivale a una distancia para el respectivo ángulo de acuerdo a la resolución angular configurada, en los datos denominados intermedios entre distancia y distancia se encuentran diferentes valores cuya explicación se explica más adelante en la Tabla 3.29.

En la Tabla 3.28, se tiene la relación del número de bytes para cada una de las configuraciones angulares.

Tabla 3.28. Divisiones de paquetes de bytes vs longitud de bytes para cada resolución angular.

DESCRIPCIÓN	Resolución 0.5º	Resolución 0.25º
Bytes constantes	12	12
Bytes de las distancias	381	761
Bytes intermedios	381	761
Bytes de corrección	2	2
Bytes que completan los datos	760	0
Bytes totales	**1536**	**1536**

En el caso de la resolución angular de 0.5º, los 760 bytes que hacen falta para completar los 1536 bytes de datos recibidos, se completan con bytes de valor ‘00’ y ‘05’.

Tanto los bytes de las distancias como los bytes intermedios son entregados por el escáner láser en hexadecimal. Al tomar los bytes de las distancias haciendo caso omiso a los bytes intermedios se obtuvo que para distancias superiores a 255 cm

la información entregada era errónea, ya que en el sistema hexadecimal el máximo valor para un byte es 'FF' (255). Observando el experimento 4 fase 4 del capítulo 3 se tiene que la longitud de la TDEL no varía independientemente del entorno o de la configuración de la resolución angular. Después de realizar un análisis de los datos obtenidos en dicho experimento y teniendo en cuenta que los datos intermedios en realidad si entregan información valiosa para poder establecer la distancias, se determina que los datos intermedios indican que, al byte de distancia que se encuentre a continuación del byte intermedio, se le debe sumar 'FF' (255) tantas veces como lo indique dicho byte. Para mayor claridad ver Tabla 3.29 donde se explica lo anterior por medio de ejemplos.

Tabla 3.29. Ejemplo de la interpretación del byte intermedio para determinar la distancia real.

Ejemplo	Descripción	Hexadecimal	Decimal	Interpretación
1	Byte intermedio	00	0	El byte intermedio indica que al valor de la distancia obtenido no se le debe sumar nada Distancia a la que se encuentra el objeto detectado para el ángulo correspondiente.
	Byte de la distancia	DE	222	
2	Byte intermedio	02	2	El byte intermedio indica que al valor de la distancia obtenido se le debe suma 'FF*2' (510) Para este caso como el valor de la distancia lo antecede un byte intermedio '02' se tiene que la distancia real a la que se encuentra el objeto es de 732cm (510+222).
	Byte de la distancia	DE	222	
3	Byte intermedio	03	3	El byte intermedio indica que al valor de la distancia obtenido se le debe suma 'FF*3' (765) Para este caso como el valor de la distancia lo antecede un byte intermedio '02' se tiene que la distancia real a la
	Byte de la distancia	A8	168	

	que se encuentra el objeto es de 933cm (765+168).

Composición trama de datos de respuesta del escáner láser.

La trama de datos de respuesta que entrega el escáner láser, se denominará de ahora en adelante Trama de Distancias del Escáner Láser (TDEL). A partir de la información obtenida en el experimento 4 fase 4, se plantean dos estructuras de la TDEL según su resolución angular de 0.25º o 0.5º, estas estructuras se muestran en la Tabla 3.30 y la Tabla 3.31 respectivamente.

Tabla 3.30. Estructura de la TDEL con resolución angular de 0.25º.

Bytes de inicio constantes	Datos de las distancias + datos intermedios	Bytes de corrección
00 00 00 00 0C 00 02 FE FF 07 00 08	(1522 bytes)	(2 bytes)

Tabla 3.31. Estructura de la TDEL con resolución angular de 0.5º.

Bytes de inicio constantes	Datos de las distancias + datos intermedios	Bytes que completan los datos	Bytes de corrección
00 00 00 00 0C 00 02 FE FF 07 00 08	(762 bytes)	(760 bytes)	(2 bytes)

El software SCD se sigue usando por medio de una máquina virtual para realizar la configuración de escáner láser, ya que éste guarda la última configuración y no sería necesario realizarla cada vez que se vaya a hacer uso del escáner a no ser que la configuración cambie.

Los resultados completos del experimento 4 en cada una de sus fases con las respectivas comparaciones, por ser tan extensos no se muestran en este capítulo, estos resultados se encuentran en el Anexo 2: “Resultados de los experimentos”.

Una vez obtenidas las tramas de datos TEEL y TDEL, se procede a implementar un programa en C++ para realizar la conexión escáner láser – computador haciendo uso de las tramas obtenidas en el experimento 4.

3.4 VALIDACIÓN DE LAS TRAMAS DE DATOS OBTENIDAS EN EL EXPERIMENTO 4 HACIENDO USO DEL PROGRAMA IMPLEMENTADO EN C++

En esta sección se detalla el diseño e implementación de un programa en C++ para el envío de la TEEL deducidas en el experimento 4 y se realiza un quinto

experimento para validar el programa implementado. Para ver el código del programa implementado ir al Anexo 3: “Códigos de programas implementados”

3.4.1 Experimento 5

Para este experimento se ejecuta el programa implementado en C++ y para la validación de los datos obtenidos de este programa se comparan con los resultados del experimento 4.

El hardware para el experimento 5 se observa en la Figura 3.22, donde el PC1 ejecuta el programa implementado en C++ enviando las tramas de datos encontradas en el experimento 4 desde PC1 al escáner láser y en pantalla se imprime la respuesta del escáner láser. De esta manera con los datos obtenidos y con la información del experimento 4 se realiza una comparación.

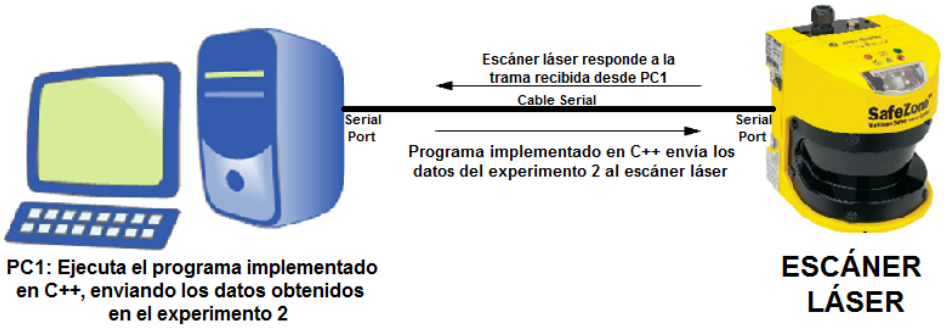


Figura 3.22. Montaje de los equipos usados para el experimento 5.

En el experimento 4 se obtuvo como resultado la identificación de 8 tramas de datos importantes para obtener como respuesta del escáner láser las distancias medidas. Con estos resultados se toman las tramas que se deben enviar según la secuencia de pasos de la Tabla 3.32, que son enviadas por medio del algoritmo implementado.

Tabla 3.32. Pasos realizados para enviar las 8 tramas de datos obtenidas en el experimento 4 por medio del algoritmo implementado

PASOS	TRAMAS
1	00 00 41 44 19 00 00 05 FF 07 19 00 00 05 FF 07 07 0F 9F D0
2	00 00 45 44 15 00 00 0E FF 07
3	00 00 45 44 19 00 00 05 FF 07
4	00 00 41 44 19 00 00 05 FF 07 19 00 00 05 FF 07 00 00 E7 B8
5	00 00 41 44 19 00 00 05 FF 07 19 00 00 05 FF 07 07 0F 9F D0
6	00 00 45 44 0B 00 00 56 FF 07
7	00 00 45 44 38 00 00 24 FF 07

8	00 00 45 44 0C 00 02 FE FF 07

El envío de los datos de la Tabla 3.32 se realiza con el fin de validar los resultados obtenidos con el algoritmo implementado. La validación de estos resultados se realiza con la respectiva comparación teniendo en cuenta los resultados obtenidos en el experimento 4.

A continuación se muestra un ejemplo del procedimiento realizado con cada trama enviada

- Primera trama envíada: 00 00 41 44 19 00 00 05 FF 07 19 00 00 05 FF 07 07 0F 9F D0, se recibe como respuesta la imagen mostrada en la Figura 3.23.

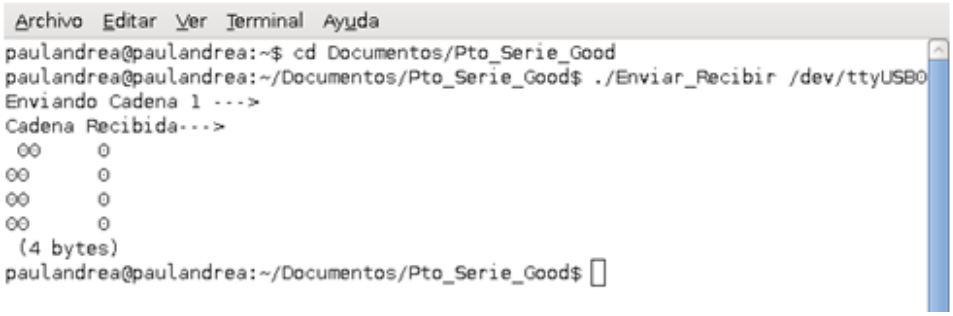


Figura 3.23. Respuesta del escáner láser a la primera trama enviada.

Al comparar los resultados obtenidos con el programa implementado en C++ y los obtenidos en Hercules, al enviar la trama 1, se observa que son los mismos, ver Figura 3.24.

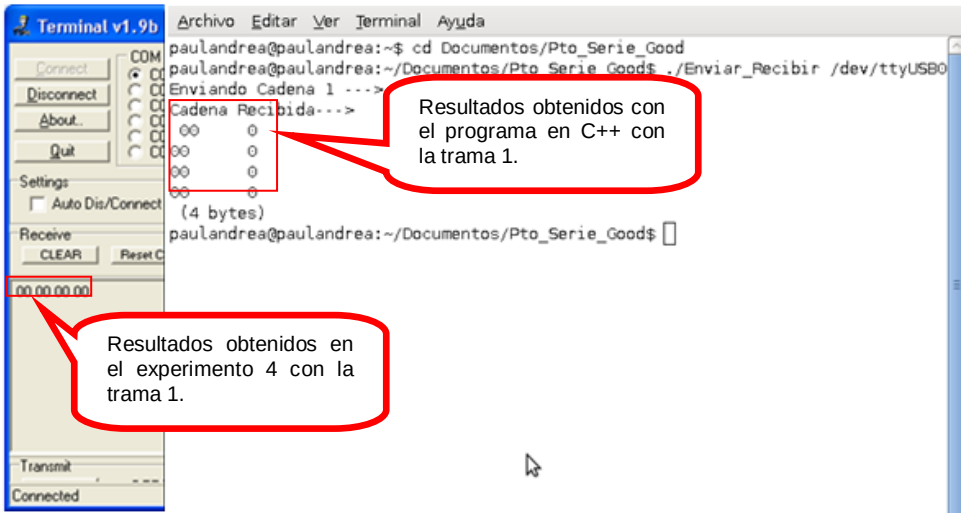


Figura 3.24. Comparación de los resultados obtenidos en el experimento 4 y con el programa implementado en C++, al enviar la trama 1.

Al realizar las respectivas comparaciones de los datos obtenidos en el anterior procedimiento y los del experimento 4 se observa que el algoritmo funciona con optimos resultados, lo que indica que se puede omitir el uso del SCD. El empleo en la implementación del Sistema de Detección de Obstáculos es importante.

Al estudiar las tramas de datos obtenidas en el experimento 4 e implementar el algoritmo, se llega a la conclusión que estas tramas de datos corresponden a las tramas de comunicación ya que se logra una comunicación exitosa haciendo uso del programa implementado y se recibe la respuesta del escáner láser con los datos de las distancias.

CAPITULO 4: DISEÑO E IMPLEMENTACIÓN DEL SISTEMA DE DETECCIÓN DE OBSTÁCULOS EN APLICACIONES DE ROBÓTICA MÓVIL

En este capítulo se detalla el diseño y la implementación modular del Sistema de Detección de Obstáculos, haciendo uso de las resultados obtenidos en el capitulo anterior.

4.1 DISEÑO DEL SISTEMA DE DETECCIÓN DE OBSTÁCULOS (SDO)

El LIDAR Safe Zone Scanner Laser Allen Bradley de la empresa Rockwell Automation requiere del diseño e implementación de un Sistema de Detección de Obstáculos (SDO), compuesto por una parte hardware denominada Sistema Láser de Detección de Obstáculos (SLDO) y una software denominada Sistema Software de Detección de Obstáculos (SSDO), el cual debe comunicarse con una plataforma software de robótica móvil, de tal forma que se permita procesar externamente la información entregada por el laser y habilitarlo para realizar aplicaciones de robótica móvil.

4.1.1 Requerimientos del SDO

Para el diseño del SDO se tienen requerimientos hardware y software:

- **Requerimientos hardware:**

  1. Un Láser Safe Zone Scanner Laser de la marca Allen Bradley de la empresa Rockwell Automation con su cable serial.

  2. Un Computador con puerto serie.

  3. Un Conversor USB – Serial, esto en el caso de que el computador no tenga puerto serie.

  4. Una Fuente de alimentación de voltaje de 24V para el láser.

- **Requerimientos software:**

  1. Hacer uso del sistema operativo Linux (Ubuntu 10.04) como soporte del SDO.

  2. Hacer uso de la plataforma software de robótica móvil Player como servidor para establecer conexión entre el escáner láser y un Aplicativo Software del

Sistema de Detección (ASSD).

3. Hacer uso del IDE Code::Blocks como entorno para desarrollo integrado libre en lenguaje C++ con el fin de implementar el ASSD.

4. Establecer conexión entre el escáner láser y el ASSD con la ayuda de la plataforma software de robótica móvil Player.

5. Diseñar e implementar un ASSD, que haga uso de la información entregada por Player y que cumpla con las siguientes funcionalidades:

5.1 Asociar las distancias con su respectivo ángulo, ya que la información obtenida del laser solo contiene las distancias.

5.2 Trazar un perfil de línea del entorno escaneado y permitir modificar el campo de visualización según se requiera.

5.3 Entregar información de las coordenadas (Distancia – ángulo) de uno de los puntos del objeto más cercano.

5.4 Entregar información de las coordenadas (Distancia – ángulo) de uno de los puntos del objeto más lejano.

5.5 Permitir exportar los datos de las distancias con sus respectivos ángulos a archivos de texto.

5.6 Realizar una interfaz gráfica que permita la integración de la información que se va a entregar al usuario.

4.1.2 Diseño del SDO

Para el diseño y la implementación del SDO se realiza el diagrama de la Figura 4.1, que muestra las componentes hardware y software del sistema teniendo en cuenta los requerimientos.

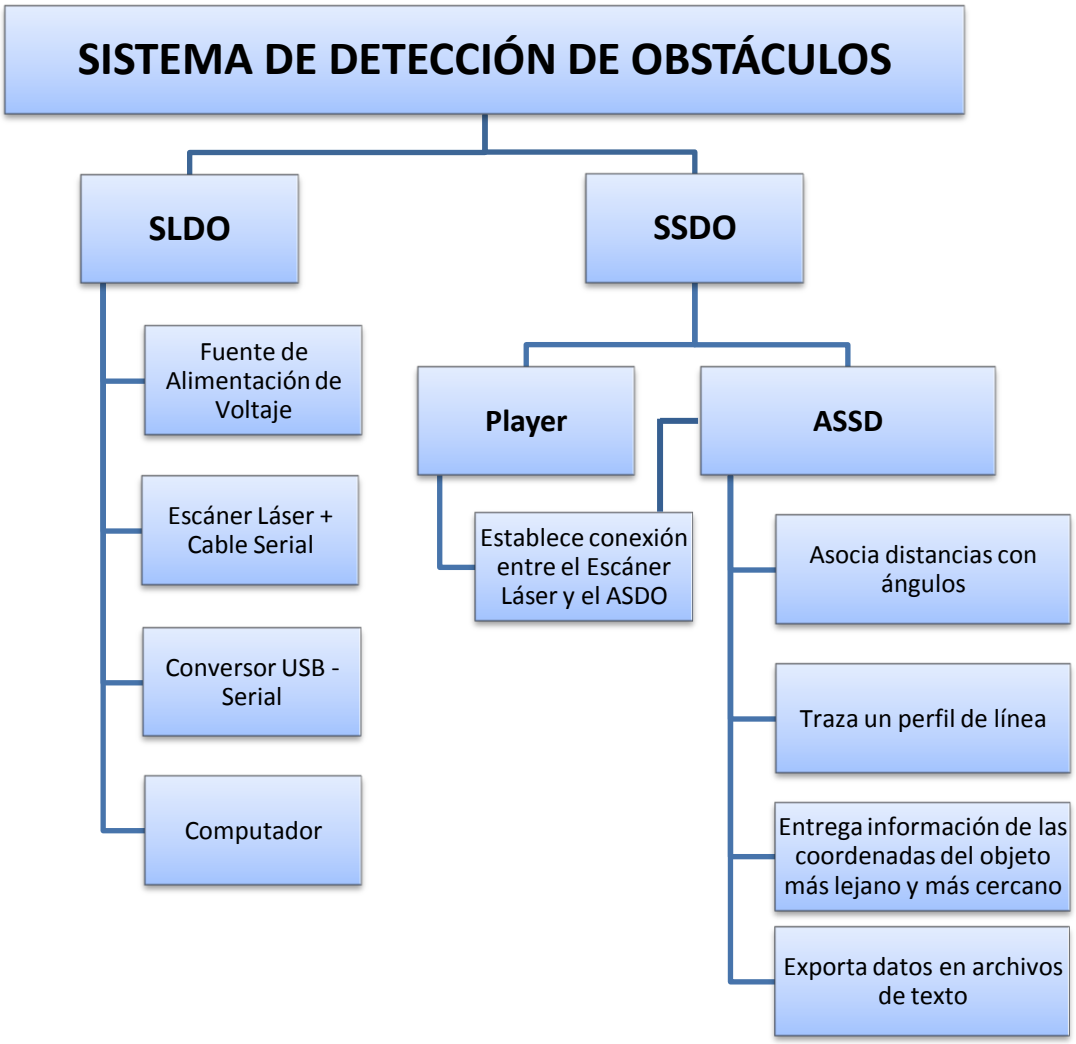


Figura 4.1. Diagrama del Sistema de Detección de Obstáculos.

En los requerimientos hardware se realiza la conexión entre un escáner láser y el computador por medio de un cable serial, en el caso de que el PC no posea puerto serie se hace uso de un conversor USB – Serial. Para la conexión del hardware se tiene en cuenta la Figura 4.2, donde el escáner láser tiene una fuente de alimentación de voltaje y este se conecta con el computador a través del puerto serie haciendo uso del cable serial del escáner.

Figura 4.2. Hardware usado en el Sistema Láser para Detección de Obstáculos

Para el diseño de la parte software SSDO se tienen dos módulos:
1. Módulo A: Conexión entre el escáner láser y el ASSD usando Player.
2. Módulo B: ASSD.

4.1.3 Diseño del Módulo A: Conexión entre el escáner láser y el ASSD

Este módulo es el encargado de establecer conexión entre el escáner láser y el ASSD haciendo uso de la plataforma software de robótica móvil Player. Para establecer esta conexión se debe tener en cuenta los siguientes pasos asociados con los requerimientos software:

1. Realizar la apertura del puerto serie y configurar la velocidad de transmisión de datos (baud rate).

2. Realizar el envío de la TEEL (Trama de Enganche del Escaner Laser) que se encuentra dividida en 2 partes:

2.1 Cabecera de la TEEL: se compone por 7 cadenas de datos que deben ser enviadas una a una, y para cada envío se debe leer la respuesta del escáner láser antes de enviar la siguiente. Las cadenas que se deben enviar y las longitudes de los datos recibis del escáner láser se encuentran en la Tabla 4.1. Es importante resaltar que la cabecera de la TEEL solo se envía al escáner láser una sola vez.

Tabla 4.1. Cadenas que componen la cabecera de la TEEL.

CADENA	CADENA ENVIADA AL ESCÁNER LÁSER	LONG. CADENA RECIBIDA DEL ESCÁNER LÁSER (BYTES)
1	00 00 41 44 19 00 00 05 FF 07 19 00 00 05 FF 07 07 0F 9F D0	4
2	00 00 45 44 15 00 00 0E FF 07	32
3	00 00 45 44 19 00 00 05 FF 07	15
4	00 00 41 44 19 00 00 05 FF 07 19 00 00 05 FF 07 00 00 E7 B8	4
5	00 00 41 44 19 00 00 05 FF 07 19 00 00 05 FF 07 07 0F 9F D0	4
6	00 00 45 44 0B 00 00 56 FF 07	176
7	00 00 45 44 38 00 00 24 FF 07	76

2.2 Cadena de solicitud de las distancias: es la encargada de solicitar al escáner láser la información de las distancias, una vez se ha enviado la cabecera de la TEEL, a través del envío repetitivo de la cadena que se tiene en la Tabla 4.2, se reciben datos constantemente de las distancias del escáner láser.

Tabla 4.2. Cadena de solicitud de distancias enviada al escáner láser

CADENA	CADENA DE SOLICITUD DE LAS DISTANCIAS ENVIADA AL ESCÁNER LÁSER	LONG. CADENA RECIBIDA DEL ESCÁNER LÁSER (BYTES)
8	00 00 45 44 0C 00 02 FE FF 07	1536

3. Recibir la información obtenida con la cadena de la solicitud de las distancias y dejarla disponible para hacer uso de ella en el ASSD.

4.1.4 Diseño del módulo B: ASSD

Este módulo es el encargado de recibir las distancias entregadas por la plataforma software de robótica móvil Player y procesar esta información con el fin de apoyar tareas de navegación autónoma. Dentro de este procesamiento de la información se debe realizar las siguientes seis (6) tareas asociadas a las mismas seis funcionalidades del requerimiento software 5:

1. Asociar las distancias con su respectivo ángulo.
El escáner láser solo entrega las distancias medidas, las cuales corresponden a

un ángulo que varía dependiendo de la resolución angular. Para asociar estas distancias con su respectivo ángulo se deben tener en cuenta las siguientes consideraciones:

- El escáner láser tiene un campo de escaneo que va desde -5º hasta 185º.

- La resolución angular del escáner láser depende de la configuración interna del mismo que se realiza con se hace con el SCD, solo hay dos opciones de configuración angular: 0.25º y 0.5º.

- La primera distancia entregada por el escáner láser corresponde al ángulo -5º.

2. Trazar un perfil de línea del entorno escaneado y permitir modificar el campo de visualización según se requiera.
Con la información de las distancias y el ángulo se debe trazar un perfil de línea que permita visualmente al usuario determinar los objetos presentes en el entorno escaneado. Para esto se debe:

- Tomar las distancias con sus respectivos ángulos y transformar de coordenadas polares a coordenadas rectangulares, con el fin de realizar el grafico del perfil de línea entregado por el SSDO. Las distancias se transforman a coordenadas rectangulares: con la ecuación 4.1 se obtiene la posición correspondiente al eje X y con la ecuación 4.2 se encuentra la posición correspondiente al eje Y.

$$X = r * \cos\left(\frac{\theta * \pi}{180}\right) \qquad (4.1)$$

$$Y = r * sen\left(\frac{\theta * \pi}{180}\right) \qquad (4.2)$$

  Donde r es la distancia entregada por el escáner láser para determinado ángulo Ɵ.

- Ubicar los datos obtenidos de la transformación a coordenadas rectangulares en una gráfica X – Y.

- Permitir modificar el X y Y máximo por efectos de visualización según lo determine el usuario.

3. Entregar información de las coordenadas (Distancia – ángulo) de uno de los puntos del objeto más cercano.
Los datos de las Distancias son almacenados en un vector [A] y se ubica el valor mínimo encontrado en el vector, se devuelve la posición en el que se encuentra

valor mínimo y con esta información poder mostrar al usuario el menor valor de la distancia con su respectivo ángulo y su equivalente en coordenadas rectangulares.

4. Entregar información de las coordenadas (Distancia – ángulo) de uno de los puntos del objeto más lejano.
Los datos de las Distancias son almacenados en un vector [A] y se ubica el valor máximo encontrado en el vector, se devuelve la posición en el que se encuentra valor máximo y con esta información poder mostrar al usuario el menor valor de la distancia con su respectivo ángulo y su equivalente en coordenadas rectangulares.

5. Permitir exportar los datos de las distancias con sus respectivos ángulos a archivos de texto.
Para hacer uso de la información del SSDO, es importante dejar exportar a archivos de texto la información de las distancias con sus respectivos ángulos, y de esta manera usarlos en el análisis de los datos recibidos del escáner láser, mejoramiento del SDO y trabajos futuros haciendo uso del SDO.

6. Realizar una interfaz gráfica que permita la presentación de la información que se va a entregar al usuario.
Para entregar toda la información del SSDO, es necesaria la implementación de una interfaz gráfica que le permita al usuario de manera amigable ver la información entregada, realizar las modificaciones de las zonas y campo de visualización y guardar la información en archivos de texto.

Las seis funcionalidades del módulo B: ASSD, se agruparon en 3 submódulos de la siguiente manera:

4.1.4.1 Submódulo B1: TD
Es el encargado de realizar el tratamiento a los datos de las distancias recibidas, hacer cambios de coordenadas, calcular máximos y mínimos, asociar distancias con sus respectivos ángulos, entre otros.

4.1.4.2 Submódulo B2: PLOT
Es el encargado de integrar las diferentes funciones para realizar la respectiva gráfica de los datos del entorno escaneado, para la implementación de este submódulo se hace uso de la librería GLUT.

4.1.4.3 Submódulo B3: HMI

Es el encargado de integrar las diferentes funciones que conforman la interfaz de usuario realizada con la librería GLUI.

4.2 IMPLEMENTACIÓN DEL SSDO

La implementación del SSDO se dividió en dos módulos de la misma manera como se estableció en el diseño del sistema: Módulo A y Módulo B.

4.2.1 Implementación del Módulo A: Conexión entre el escáner láser y el ASSD.

Para la implementación de este módulo se crea un *plugin driver* que es el encargado de establecer la conexión entre el escáner láser y el computador. Es importante resaltar que a pesar de que Player ya contiene una variedad de drivers, entre ellos, drivers para escáner láser, es necesario implementar un *plugin driver* en particular para el *Safe Zone Multizone Scanner Laser*, ya que como se describe en el diseño del módulo A, para que el escáner láser pueda establecer comunicación con el computador necesita enviar y recibir una serie de tramas de datos antes de comenzar a reportar las distancias.

El *plugin driver* es implementado en C++ y sigue un diseño de Programación Orientada a Objetos, como se describe a continuación:

- Declaración de la clase *driver plugin*:

```
class safezoneDriver : public ThreadedDriver
{
    public:
        safezoneDriver(ConfigFile* cf, int section);
        virtual int ProcessMessage(QueuePointer &resp_queue,
                                   player_msghdr * hdr,
                                   void * data);
    private:
        virtual void Main();
        virtual int MainSetup();
        virtual void MainQuit();
    int foop;
};
```

- La implementación de los métodos virtuales es necesaria para poder abrir el puerto serie, realizar el envío de las diferentes cadenas de datos, recibir las respuestas por parte del escáner láser cada vez que se envíe una cadena de datos y hacer el respectivo cierre del puerto.

```
int safezoneDriver::MainSetup()
void safezoneDriver::MainQuit()
void safezoneDriver::Main()
```

- Registro del plugin driver en el servidor e inicialización:
  Función safezoneDriver_Register, esta función es la encargada de la inicialización del *plugin driver*.

```
extern "C"
{
    int player_driver_init(DriverTable* table)
    {
        puts("safezone driver initializing");
        safezoneDriver_Register(table);
        puts("safezone driver done");
        return(0);
    }
}
```

  Función void safezoneDriver_Register, es la encargada de agregar el plugin driver a una tabla de drivers de Player.

```
void safezoneDriver_Register(DriverTable* table)
{
    table->AddDriver("safezonedriver", safezoneDriver_Init);
}
```

  Función safezoneDriver_Init, es la encargada de crear un nuevo objeto safezoneDriver y devuelve un puntero al driver.

```
Driver* safezoneDriver_Init(ConfigFile* cf, int section)
{
    return((Driver*)(new safezoneDriver(cf, section)));
}
```

- Constructor: en el constructor se lee el archivo de configuración (*.cfg) y se definen las interfaces utilizadas.

```
safezoneDriver::safezoneDriver(ConfigFile* cf, int section)
    ThreadedDriver(cf, section, false,
                        PLAYER_MSGQUEUE_DEFAULT_MAXLEN,
                        PLAYER_POSITION2D_CODE)
{
    return;}
```

El código del *plugin driver* se compila haciendo uso de CMake, para esto, se crea un archivo llamado “CMakeLists.txt” en el directorio donde este implementado el código del *plugin driver*. Una vez compilado el plugin driver, se crea el archivo de configuración que será el encargado de hacer uso de plugin driver implementado.
Para ver el código del *plugin driver* implementado ir al Anexo 3: “Códigos de programas implementados”.

La plataforma software para robótica móvil Player se compone de drivers y dos herramientas de simulación (Stage y Gazebo). En esta investigacion solo se hace uso de la plataforma para la creación del plugin driver que permite la conexión

entre el escáner láser y el computador; las herramientas de simulación, en especial Stage que sería la que permitiría la simulación en 2D, no se usan, ya que se cuenta con un dispositivo real que permite crear un entorno real y no se necesita de la simulación del dispositivo ni de un entorno.

4.2.2 Implementación del Módulo B: ASSD

Para la implementación de este módulo se toma la información entregada por el *plugin driver* implementado y se trabaja en el mismo orden del diseño (3 submódulos)

4.2.2.1 Implementación del Submódulo B1: TD

En este submódulo se realiza el tratamiento de los datos recibidos de Player por medio de las siguientes funciones y procedimientos:

- Para asociar las distancias con sus respectivos ángulos se crea un vector que contiene los ángulos que se llena teniendo en cuenta la resolución angular con la que se configuró el escáner láser y se asocia con el vector que contiene los datos de las distancias.

```
for(j=0;j<ndatos;j++)
{
    angulo[j]=pasogrados+angulo[j-1];
}
```

- Procedimiento void datospolar2cartesiano(), es el encargado de pasar los datos de coordenadas polares a rectangulares.

```
void datospolar2cartesiano()
{
    distancia[j]=abuffer[j];
    X[j]=distancia[j]*cos(angulo[j]*3.14159/180);
    Y[j]=distancia[j]*sin(angulo[j]*3.14159/180);
    x=(1.0/(Xmax-Xmin))*(2*X[j]-(Xmax+Xmin));
    y=(1.0/(Ymax-Ymin))*(2*Y[j]-(Ymax+Ymin));}
```

- Función int CalcMenor(int *dist), es la encargada de calcular la coordenada (Distancia – Ángulo) de uno de los puntos del objeto más cercano, devolviendo la posición en la que se encuentra esta distancia.

```
int CalcMenor(int *dist)
{
    int temp2=10000;
    int i_d,indice=0;
    for(i_d=0;i_d<ndatos;i_d++)
    {
        if(dist[i_d]<temp2){
            temp2=dist[i_d];
            indice=i_d;
        }
    }
    return indice;}
```

- Función int CalcMayor(int *dist), es la encargada de calcular la coordenada (Distancia – Ángulo) de uno de los puntos del objeto más lejano, devolviendo la posición en la que se encuentra esta distancia.

```
int CalcMayor(int *dist)
{
    int temp=0;
    int i_d,indice=0;
    for(i_d=0;i_d<ndatos;i_d++)
    {
        if(dist[i_d]>temp)
        {
            temp=dist[i_d];
            indice=i_d;
        }
    }
    return indice;
}
```

4.2.2.2 Implementación del Submódulo B2: PLOT

En este submódulo se integran las diferentes procedimientos para realizar la respectiva gráfica de los datos del entorno escaneado, tal como se explico en este submódulo y donde se hace uso de la librería GLUT.

- Procedimiento void Escribirtexto(float x, float y, void *font, int Num), se encarga de colocar el texto que se muestra en los ejes del perfil de línea.

```
void Escribirtexto(float x, float y, void *font, int Num)
{
    int cent=0, int mil=0, dec=0, unid=0;
    glRasterPos2f(x, y);
    if(Num>=1000)
    {
        mil=Num/1000;
        glutBitmapCharacter(font,mil+48);
    }
    if(Num>=100)
    {
        cent=(Num-mil*1000)/100;
        glutBitmapCharacter(font,cent+48);
    }
    if(Num>=10)
    {
        dec=(Num-cent*100-mil*1000)/10;
        glutBitmapCharacter(font,dec+48);
    }
    if(Num>=0)
    {
        unid=(((Num-mil*1000)-cent*100)-dec*10);
        glutBitmapCharacter(font,unid+48);
    }
}
```

- Procedimiento void pintarescala(), es el encargado de pintar las líneas que determinan la escala del perfil de línea.

```
void pintarescala()
{
    glPushMatrix();
    glColor3d(0.5,0.3,0.8);
    glRotated(girox,1,0,0);
    glRotated(giroy,0,1,0);
    glRotated(giroz,0,0,1);
    float cuad;
    int division=0;
    for(cuad=-1;cuad<1;cuad+=0.1)
    {
        division=(int)(Xmax-Xmin)*cuad/2;
        Escribirtexto(cuad, -0.85f, GLUT_BITMAP_HELVETICA_12, division);
        division=(int)(((Ymax-Ymin)/2)*cuad+(Ymax-(Ymax-Ymin)/2));
        Escribirtexto(-0.15, cuad, GLUT_BITMAP_HELVETICA_12, division);
    }
    glEnd();
    glPopMatrix();
}
```

- Procedimiento void pintarcuadricula(), se encarga de dibujar la cuadricula que usa como fondo el perfil de línea.

```
void pintarcuadricula()
{
    glPushMatrix();
    glColor3d(0.6,0.6,0.6);
    glRotated(girox,1,0,0);
    glRotated(giroy,0,1,0);
    glRotated(giroz,0,0,1);
    glBegin(GL_LINES);
    glLineWidth(3);
    float cuadx;
    for(cuadx=-1;cuadx<1;cuadx+=0.07)
    {
        glVertex3d(cuadx,-1,0);
        glVertex3d(cuadx,1,0);
        glVertex3d(-1,cuadx,0);
        glVertex3d(1,cuadx,0);
    }
    glEnd();
    glPopMatrix();
}
```

- Procedimiento void pintarejes(), es el encargado de dibujar los ejes X y Y.

```
void pintarejes()
{
    glBegin(GL_LINES);
    glLineWidth(1);
    glColor3d(0,0,1);
    glVertex3d(0.8,0.8,0);
    glVertex3d(0.85,0.75,0);
```

```
        glVertex3d(0.85,0.8,0);
        glVertex3d(0.8,0.75,0);
        glColor3d(1,0,0);
        glVertex3d(0.8,0.7,0);
        glVertex3d(0.825,0.65,0);
        glVertex3d(0.85,0.7,0);
        glVertex3d(0.8,0.6,0);
        glEnd();
        glPushMatrix();
        glColor3d(0,0,1);
        glRotated(girox,1,0,0);
        glRotated(giroy,0,1,0);
        glRotated(giroz,0,0,1);
        glLineWidth(1.0);
        glBegin(GL_LINES);
        glVertex3d(-0.8,-0.8,0);
        glVertex3d(0.8,-0.8,0);
        glColor3d(1,0,0);
        glVertex3d(0,-0.8,0);
        glVertex3d(0,0.8,0);
        glEnd();
        glPopMatrix();
}
```

- Procedimiento void pintarmapeo(): Dibuja el perfil de línea de los datos de las distancias entregadas Player.

```
void pintarmapeo()
{
    glPushMatrix();
    glLoadIdentity();
    glColor3d(0,0,0);
    glRotated(girox,1,0,0);
    glRotated(giroy,0,1,0);
    glRotated(giroz,0,0,1);
    glVertex3d((float)x,(float)y,0);
    glPopMatrix();
}
```

4.2.2.3 Implementación del Submódulo B3: HMI

Este submódulo integra las funciones que conforman la interfaz de usuario, de la siguiente manera:

- Procedimiento void guardardatos(int control), se encarga de guardar los datos en un archivo de texto al presionar el botón que aparece en la interfaz gráfica que se llama '*Save Data*'.

```
void guardardatos(int control)
{
    int nd;
    char campodato[10]="";
    char espacio[5]=" ";
    archivo=fopen("ReporteDistancia.txt","w");
    for(nd=-5;nd<ndatos;nd++)
```

```
        {
            sprintf(campodato,"Angulo=%.2f Distancia = %d\n", angulo[nd],
            distancia[nd]);
            fprintf(archivo,campodato);
        }
        fclose(archivo);
    }
```

- Procedimiento void girotox(int controlx), gira la gráfica en el sentido del eje X.

```
void girotox(int controlx)
{
    girox+=5;
}
```

- Procedimiento void girotoy(int controly), gira la gráfica en el sentido del eje Y.

```
void girotoy(int controly)
{
    giroy+=5;
}
```

- Procedimiento void girotoz(int controlz), gira la gráfica en el sentido del eje Z.

```
void girotoz(int controlz)
{
    giroz+=5;
}
```

- Las siguientes funciones agregan a la interfaz gráfica sus diferentes componentes tales como títulos, líneas de separación, checkbox, gráfica del perfil de línea, información de las distancias mayor y menor, entre otras componentes que hacen parte de la interfaz gráfica.

```
glui->add_checkbox( "Wireframe", &wireframe );
glui->add_separator();
GLUI_Panel *panel_datos=glui->add_panel("Info Datos");
glui->add_statictext_to_panel(panel_datos,"Objeto mas cercano");
Dminx->set_w(200);
Dminy=glui->add_edittext_to_panel(panel_datos,"MinY",GLUI_LIVE_INT);
Dminy->set_w(200);
Rmin->set_h(20);
Dmaxx->set_h(20);
Dmaxx->set_w(200);
Dmaxy=glui->add_edittext_to_panel(panel_datos,"MaxY",GLUI_LIVE_INT);
Dmaxy->set_h(20);
Dmaxy->set_w(200);
glui->add_separator_to_panel(panel_datos);
Rmax=glui->add_edittext_to_panel(panel_datos,"Rmax",GLUI_LIVE_INT);
Rmax->set_h(20);
Rmax->set_w(200);
Thetamax=glui->add_edittext_to_panel (panel_datos, "Angmax", GLUI_LIVE_INT);
glui2->add_column();
Zona2->set_int_limits( 10, 1000 );
Zona2->set_speed(5);
GLUI_Panel *backgr_panel=glui2->add_panel("Set XY Plane");
chkcuadric->set_int_val(1);
```

```
glui2->add_button("Save Data",0,guardardatos);
glui2->add_column();
GLUI_Panel *setline=glui2->add_panel("Set Line/Point");
chkline->set_int_val(1);
glui2->add_separator_to_panel(setline);
chkpoint=glui2->add_checkbox_to_panel (setline, "Puntos", &Puntos,1,
checkpunto);
GLUI_Master.set_glutIdleFunc( myGlutIdle );
```

Para ver el código implementado en detalle de ASSD ir al Anexo 3: “Códigos de programas implementados”.

Las distancias son entregadas en forma de texto y se tiene también una visualización de estas en tiempo de ejecución. En la visualización de los datos en tiempo de ejecución, se muestra una ventana con los datos, como se observa en la Figura 4.3, donde se tiene el ángulo, la distancia a la que se encuentra el objeto para ese ángulo y las distancias correspondientes en coordenadas rectangulares (X,Y).

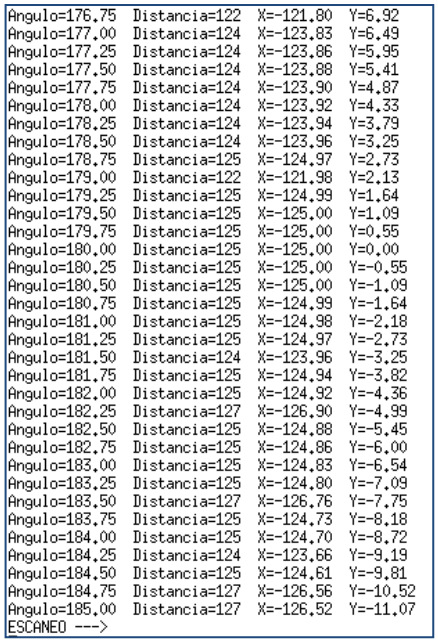


Figura 4.3. Visualización de los datos en tiempo de ejecución.

La interfaz gráfica donde se integra esta información, en donde además de la gráfica del perfil de línea entregado, de la información de las coordenadas de un punto del objeto más lejano y más cercano, también entrega opciones tales como giro de la gráfica en sus diferentes ejes, opción para determinar el máximo valor para X y Y, modificación visual de las zonas de advertencia y de protección, opción para guardar los datos en forma de texto al momento de dar click en ‘*save data’* y la opción de visualizar el perfil con línea o con puntos.

En la Figura 4.4 se muestran las diferentes partes que componen la interfaz gráfica del Sistema de Detección de Obstáculos distribuidos de la siguiente manera: en la parte de arriba, al lado izquierdo, se observa el perfil de línea para un entorno determinado, al lado derecho se encuentra la información del objeto más cercano y objeto más lejano en coordenadas polares y rectangulares, esta información se encuentra dada en centímetros para las distacias y en grados para los ángulos.

La modificación de las zonas de advertencia y de protección solo son visuales, ya que las zonas que toma el escáner láser internamente son las determinadas en la configuración interna realizada por medio del software SCD.

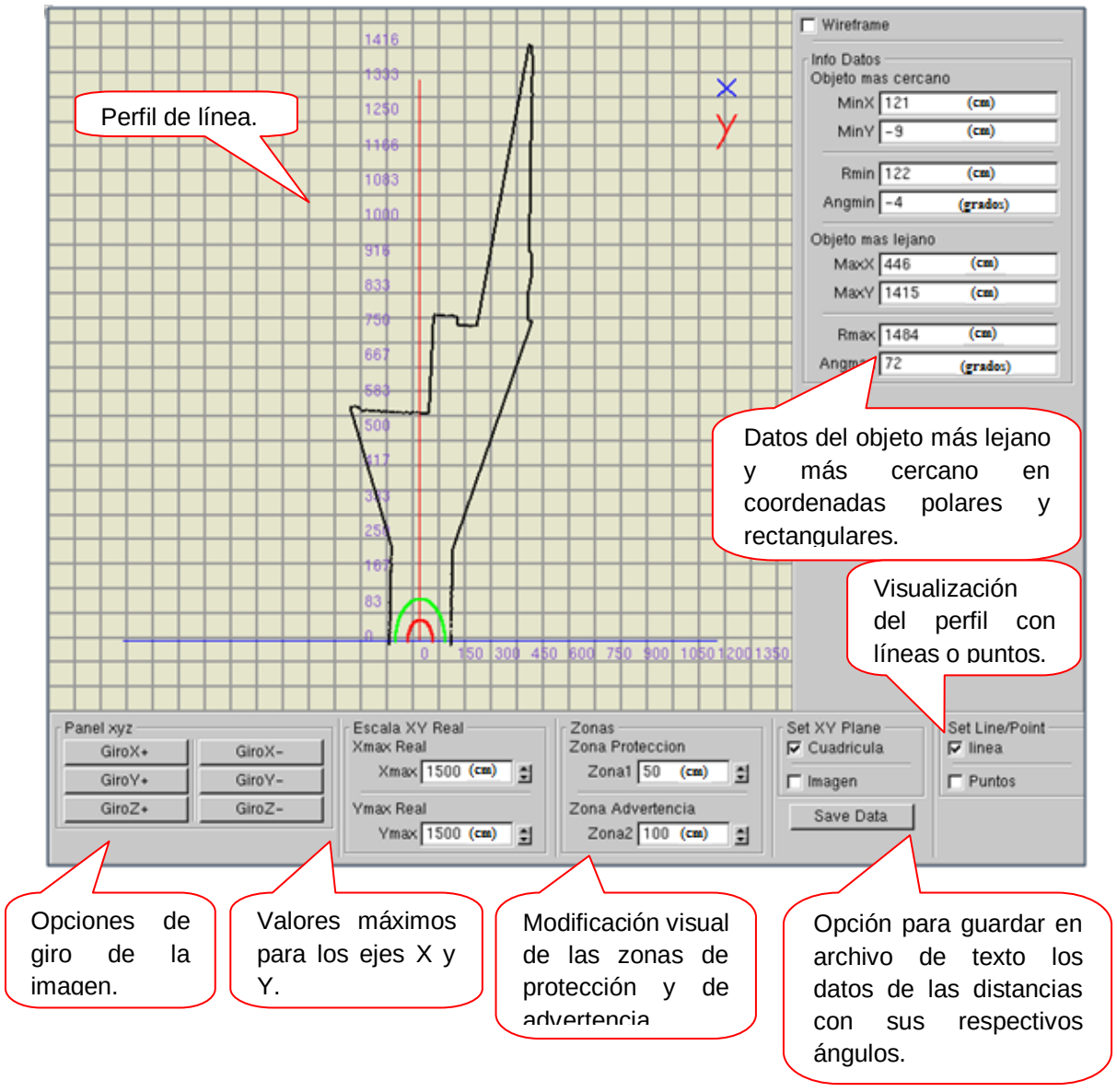


Figura 4.4. Descripción de las componentes de la interfaz gráfica del SSDO

CAPITULO 5: VALIDACIÓN DEL SISTEMA DE DETECCIÓN DE OBSTÁCULOS

En este capítulo se valida el SSDO (Sistema Software Detección Obstáculos) implementado para este trabajo de grado, para esta validación se diseñan una serie de pruebas experimentales que permiten calcular el error medio y la desviación estándar en cada una de las pruebas, y de esta manera realizar un análisis general del desempeño del sistema, teniendo como referencia el software propio (SCD) del escáner láser. La validación se hace tanto en escenarios interiores como en exteriores.

5.1. VALIDACIÓN DEL SSDO (Sistema Software Detección Obstáculos)

Con el diseño de una serie de experimentos se busca validar el SSDO tanto en escenarios interiores como en exteriores. Para los experimentos de validación se emplean elementos hardware y software que han hecho parte del proyecto, además del TopOCR [59]

Con el experimento 1 se busca determinar el error medio para el SCD y el SSDO en dos pruebas básicas de mediciones, en el experimento 2 se busca validar la operación del SSDO en interiores y con el experimento 3 se quiere validar el SSDO en exteriores.

Para todos los experimentos de validación se hace uso del montaje mostrado en la Figura 5.1

Figura 5.1 Montaje hardware usado para los experimentos de validación.

En cada una de las pruebas experimentales de este capítulo se toman 20 mediciones (20 barridos laser) para un mismo ángulo tanto con el SCD como con el SSDO. En todos los barridos el escáner láser se encuentra configurado a una resolución angular de 0.25º, obteniendo así 761 datos por barrido. Con el SSDO

se pueden tener los datos en archivos de texto por medio de la opción *Save Data* del software, no obstante con el SCD no es posible, por lo tanto se hace uso de la herramienta TopOCR para pasar los datos de imágenes a texto.

5.2 EXPERIMENTO 1: DETERMINACIÓN DEL ERROR MEDIO Y LA DESVIACIÓN ESTÁNDAR PARA EL SSDO Y EL SCD

Este experimento consta de dos pruebas, una de las pruebas se realiza con un entorno circular y la otra con un entorno cuadrado; para estas dos pruebas el escáner láser se coloca a nivel del suelo sobre una cartulina que tiene marcadas las líneas para los ángulos que tienen un incremento de 5º iniciando desde -5º hasta 185º. Al marcar las líneas sobre la cartulina, se tiene en cuenta que el punto de corte de ellas coincida con el centro donde se encuentra ubicado el espejo al interior del radar, que se encarga de direccionar el haz de luz infrarrojo hacia el objeto para el cálculo de las distancias en cada ángulo. La Figura 5.2 ilustra la ubicación del espejo al interior del radar, en la Figura 5.2.a el haz de luz infrarrojo se proyecta desde el centro de la pantalla frontal, pero a una profundidad de 67mm como se aprecia en la vista lateral de la Figura 5.2.b y en la vista superior de la Figura 5.2.c donde se observa el punto negro a partir del cual se realizan las medidas experimentales.

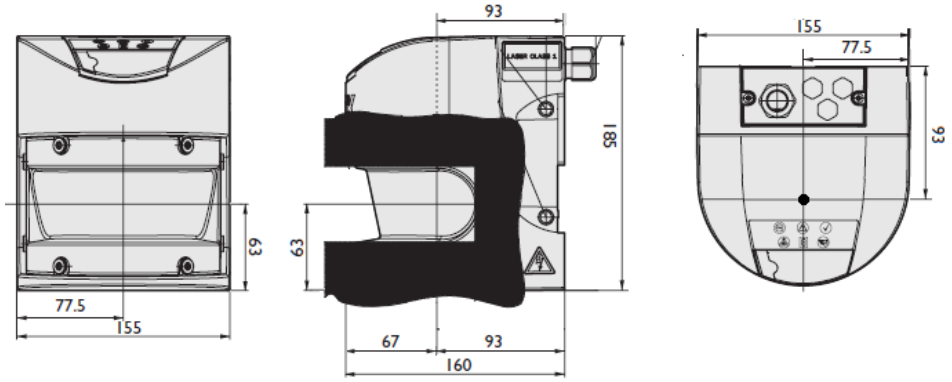


Figura 5.2. Ubicación del espejo en el interior del escáner láser [medidas en mm]: a. vista frontal, b. vista lateral, c. vista superior, adaptado de [19].

Prueba 1. Entorno circular: Se coloca el escáner láser a nivel del suelo sobre la cartulina y se crea un entorno circular de 50 cm de radio como se muestra en la Figura 5.3. Para tomar las distancias desde el escáner láser hasta el borde del cartón paja se marca sobre la cartulina el círculo creado.

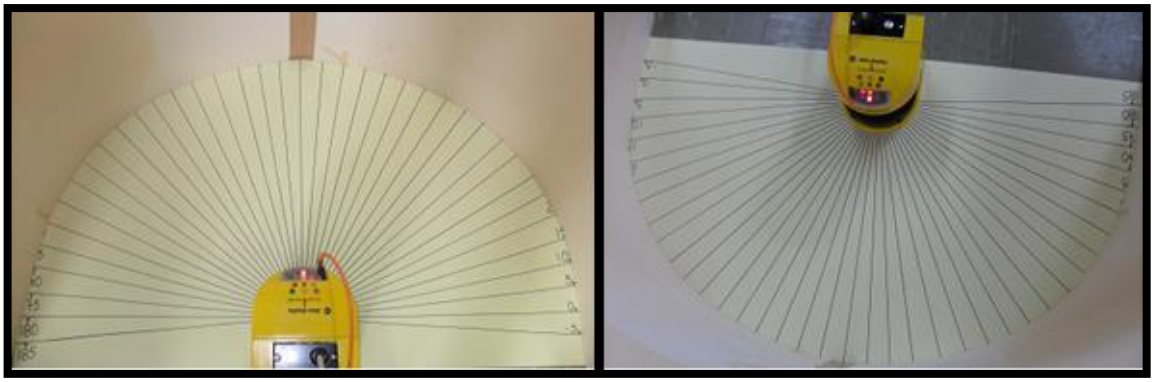


Figura 5.3 Experimento 1 prueba 1: entorno circular creado con el cartón paja.

Prueba 2. Entorno cuadrado: Se coloca el escáner láser a nivel del suelo sobre la cartulina y se crea un entorno cuadrado de 54 cm de lado como se muestra en la Figura 5.4. Para tomar las distancias desde el escáner láser hasta el borde de la caja se marca sobre la cartulina el cuadrado creado.

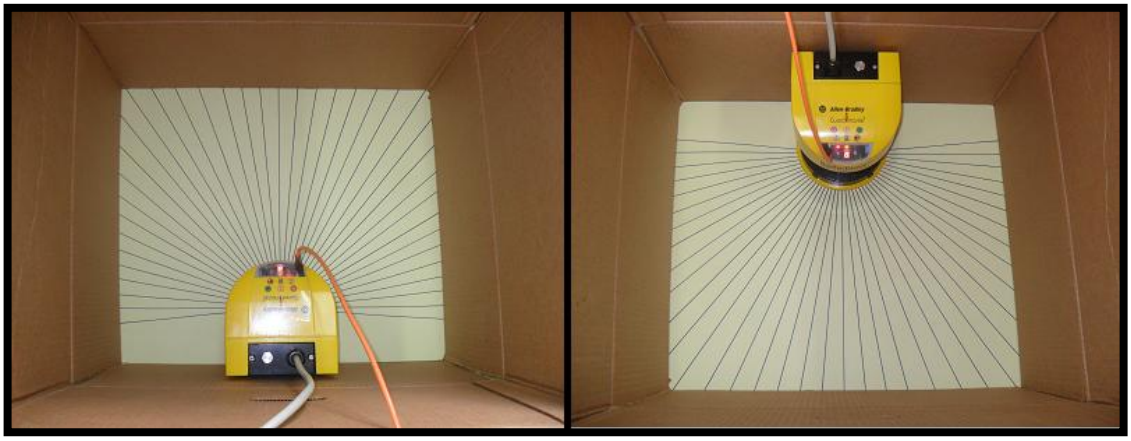


Figura 5.4 Experimento 1 prueba 2: entorno cuadrado creado con una caja.

En estas dos pruebas se procede a guardar las distancias entregadas por el SSDO y por el SCD. Para cada barrido tomado con el SCD y el SSDO se tienen 761 medidas usando una resolución angular de 0.25º, pero como en la cartulina solo se tienen 39 ángulos con un incremento de 5º iniciando desde -5º hasta 185º, se toman solo las medidas entregadas por el SSDO y el SCD correspondientes a estos 39 ángulos.

Los resultados que se tienen son:

El error y la desviación estándar para las pruebas 1 y 2 se calculan teniendo en cuenta las distancias reales (distancias medidas en la cartulina). En la Tabla 5.1, se muestran los datos del entorno circular obtenidos en la prueba 1, con los respectivos cálculos del error medio y la desviación estándar, tanto para el SCD

como para el SSDO.

Tabla 5.1 Datos del entorno circular con los resultados del error medio y la desviación estándar

A* (º)	DR* (cm)	PDO* (cm)		EM* (cm)		DE* (cm)	
		SCD	SSDO	SCD	SSDO	SCD	SSDO
-5	47.60	48.11	47.72	-0.55	-0.20	1.46	1.32
0	47.00	47.28	46.94	-0.25	0.05	0.72	0.76
5	46.60	46.83	46.22	-0.35	0.20	1.47	1.50
10	46.40	46.72	47.22	-0.35	-0.75	1.45	1.35
15	46.80	47.33	47.22	-0.45	-0.50	1.33	1.26
20	47.40	48.00	47.17	-0.75	0.30	1.18	1.41
25	47.80	48.11	48.00	-0.30	-0.30	0.45	0.45
30	48.40	48.33	48.33	0.10	0.10	0.73	0.73
35	49.20	49.11	49.28	0.10	-0.15	1.02	1.04
40	49.80	49.72	49.44	0.10	0.40	0.92	0.94
45	50.60	50.06	50.06	0.50	0.55	0.31	0.22
50	51.20	50.72	50.56	0.50	0.70	1.22	1.10
55	51.80	50.83	51.00	0.90	0.75	1.41	1.47
60	52.20	51.72	51.06	0.45	0.95	1.45	1.48
65	52.20	52.06	51.89	0.30	0.50	1.41	1.49
70	52.40	51.33	51.83	1.05	0.60	1.53	1.51
75	52.20	51.72	51.50	0.60	0.70	1.47	1.54
80	52.00	52.17	52.33	-0.10	-0.40	1.41	1.23
85	52.00	51.83	51.50	0.05	0.35	1.47	1.53
90	51.80	51.67	51.33	0.20	0.50	1.14	1.13
95	51.40	51.00	50.56	0.20	0.75	1.51	1.23
100	51.20	50.39	50.39	0.80	0.85	0.94	0.93
105	51.00	50.22	50.06	0.75	0.95	0.72	0.22
110	50.40	49.50	49.67	0.80	0.70	0.99	0.73
115	50.00	49.67	49.61	0.40	0.45	0.82	0.94
120	50.20	48.67	49.50	1.50	0.75	0.98	1.00
125	49.60	48.33	48.67	1.30	0.90	0.73	0.98
130	49.20	48.56	48.78	0.70	0.50	0.89	0.98
135	48.40	48.00	48.00	0.40	0.40	0.00	0.00
140	48.00	48.33	48.94	-0.30	-0.95	0.73	1.10
145	47.60	48.00	48.44	-0.40	-0.90	0.00	0.89
150	47.20	47.83	48.00	-0.65	-0.80	0.67	0.00
155	47.00	46.56	47.67	0.55	-0.70	1.47	0.92
160	46.60	46.72	46.50	-0.15	-0.05	1.45	1.53
165	46.40	46.56	46.50	-0.05	-0.10	1.47	1.54
170	46.00	46.17	45.89	-0.20	0.05	1.51	1.39
175	46.00	46.17	46.50	-0.20	-0.35	1.51	1.53
180	46.20	46.06	46.06	0.20	0.25	1.38	1.39

185	46.40	44.00	46.06	-0.40	0.15	1.51	1.48

* A: Ángulos, DR: Distancias Reales, PDO: Promedio Distancias Obtenidas, EM: Error Medio, DE: Desviación Estándar.

En la Tabla 5.2 se muestra el valor promedio y la desviación estándar del error medio tanto para el SCD como para el SSDO en la prueba 1. Estos valores son el resultado de los datos de la Tabla 5.1.

Tabla 5.2 Promedio y desviación estándar del error medio para el experimento 1 prueba 1.

	PEM** (cm)		DEEM** (cm)	
	SCD	SSDO	SCD	SSDO
Entorno circular	0.18	0.18	0.54	0.55

** PEM: Promedio del Error Medio, DEEM: Desviación Estándar del Error Medio

Con la información obtenida en la Tabla 5.1 y la Tabla 5.2 se realizan las graficas de resultados de la Figura 5.5 y la Figura 5.6, donde los puntos describen el comportamiento del error medio obtenido de las medidas procesadas.

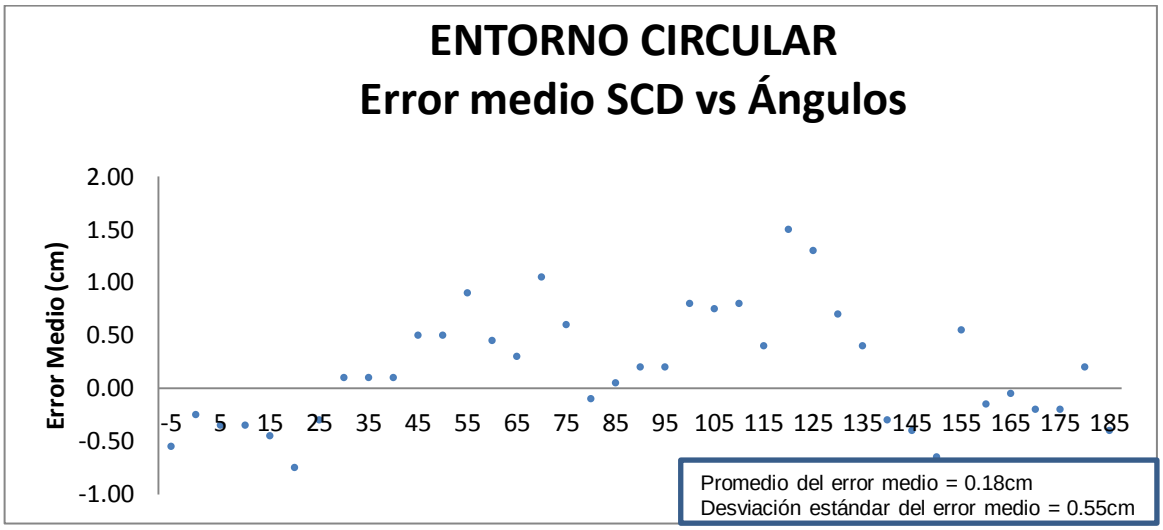


Figura 5.5. Gráfica del error medio SCD vs ángulos obtenidos en el experimento 1 prueba 1

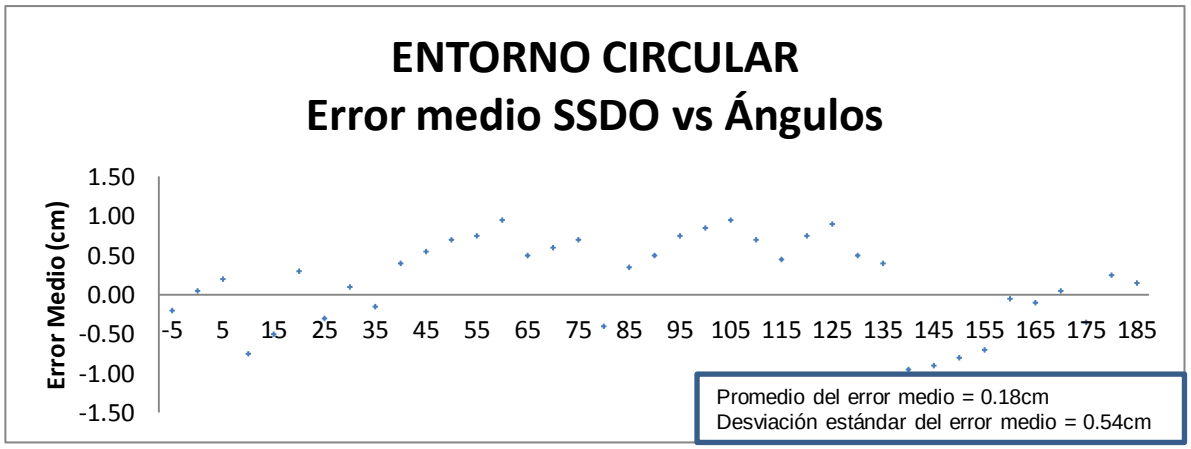


Figura 5.6. Gráfica del error medio SSDO vs ángulos obtenidos en el experimento 1 prueba 1

Con las distancias obtenidas en la prueba 1 y presentes en la Tabla 5.1 se realiza la gráfica mostrada en la Figura 5.7, donde se gráfica el perfil del entorno circular, la línea azul representa el perfil obtenido con el SCD, la línea roja representa el perfil obtenido con el SSDO y la línea verde representa el perfil obtenido de las mediciones reales.

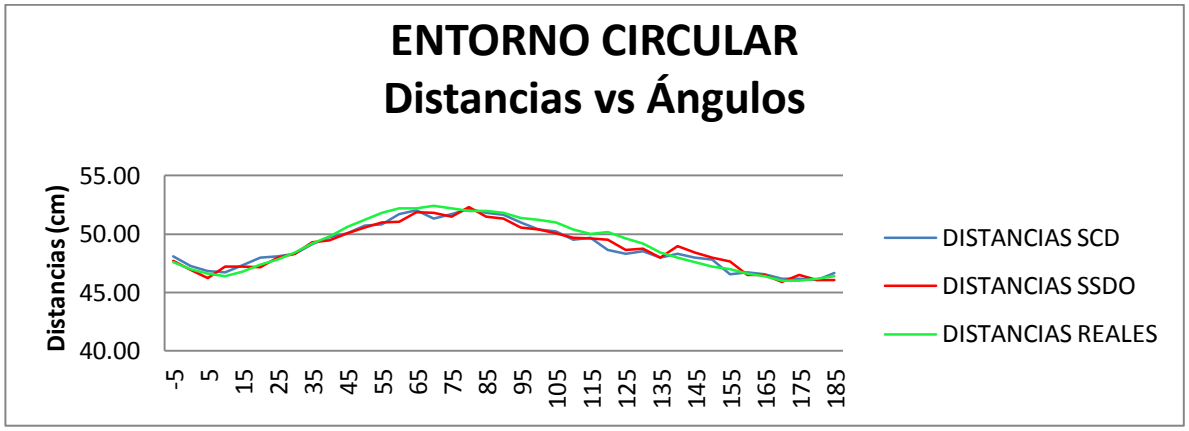


Figura 5.7 Gráfica de las Distancias vs Ángulos para las distancias del entorno circular.

En la Tabla 5.3, se muestran los datos del entorno cuadrado, con los respectivos cálculos del error medio y la desviación estándar, tanto para el SCD como para el SSDO.

Tabla 5.3 Datos del entorno cuadrado con los resultados del error medio y la desviación estándar

A* (º)	DR* (cm)	PDO* (cm)		EM* (cm)		DE* (cm)	
		SCD	SSDO	SCD	SSDO	SCD	SSDO
-5	26.20	26.60	28.67	-0.40	-2.50	1.54	0.92
0	26.20	26.27	28.53	-0.05	-2.30	1.16	0.51
5	26.40	26.20	28.53	0.25	-2.10	0.93	0.51
10	26.80	27.53	29.07	-0.90	-2.10	1.08	1.02
15	27.40	27.73	30.00	-0.35	-2.35	0.97	1.33
20	28.20	28.27	30.20	-0.20	-2.00	1.19	0.70
25	29.40	29.80	31.67	-0.50	-2.45	0.79	1.46
30	30.60	31.00	32.73	-0.65	-2.10	1.21	1.26
35	32.80	32.80	34.40	0.05	-1.65	1.21	0.94
40	35.00	36.07	36.73	-1.10	-1.75	0.31	0.91
45	38.20	39.40	39.53	-1.15	-1.35	1.31	1.15
50	42.00	42.27	43.27	-0.25	-1.30	0.91	0.47
55	47.20	48.33	46.67	-1.25	0.65	1.43	0.94
60	46.60	46.07	43.53	0.45	3.20	1.18	1.10
65	44.60	43.93	41.73	0.60	2.80	0.79	0.62
70	43.20	42.40	40.00	0.90	3.20	0.92	0.79
75	41.80	41.73	39.20	0.25	2.50	1.23	1.13
80	41.20	41.00	38.67	0.20	2.60	1.12	1.10
85	40.60	40.27	38.47	0.35	2.20	0.55	0.75
90	40.40	40.40	38.87	-0.10	1.35	0.51	1.10
95	40.60	40.33	38.13	0.30	2.40	0.73	0.62
100	41.00	40.93	39.60	-0.25	1.40	1.45	0.99
105	41.80	41.60	40.13	0.40	1.70	1.10	0.31
110	43.20	41.73	40.47	1.30	2.70	0.85	1.19
115	45.00	43.07	42.13	2.05	2.75	1.15	1.25
120	47.00	44.60	44.40	2.25	2.70	1.45	0.92
125	47.40	46.67	46.87	0.40	0.50	1.38	0.79
130	43.00	42.20	42.80	0.65	0.15	1.09	1.14
135	39.20	39.20	38.87	-0.15	0.30	1.35	1.29
140	36.00	35.73	36.53	0.25	-0.50	0.97	0.95
145	33.80	33.80	34.33	-0.10	-0.55	0.91	0.93
150	32.20	31.93	32.60	0.25	-0.25	0.60	1.10
155	30.60	30.40	30.73	0.25	-0.10	0.99	1.22
160	29.40	29.07	29.27	0.50	0.25	1.55	1.09
165	28.80	29.00	29.13	-0.10	-0.25	1.29	1.23
170	28.20	28.40	28.07	-0.45	0.05	1.14	1.04
175	27.80	28.07	27.60	-0.30	0.25	0.31	1.36
180	27.60	27.93	28.27	-0.40	-0.35	1.26	1.43
185	27.80	28.27	28.00	-0.45	-0.05	0.97	1.18

* A: Ángulos, DR: Distancias Reales, PDO: Promedio Distancias Obtenidas, EM: Error Medio, DE: Desviación Estándar.

En la Tabla 5.4 se muestra el valor promedio y la desviación estándar del error medio tanto para el SCD como para el SSDO en la prueba 2.

Tabla 5.4 Promedio y desviación estándar del error medio para el experimento 1 prueba 2.

	PEM** (cm)		DEEM** (cm)	
	SCD	SSDO	SCD	SSDO
Entorno cuadrado	0.07	0.20	0.73	1.85

** PEM: Promedio del Error Medio, DEEM: Desviación Estándar del Error Medio

Con la información obtenida en la Tabla 5.3 y Tabla 5.4 se realizan las graficas de resultados de la Figura 5.8 y Figura 5.9, donde los puntos describen el comportamiento del error medio obtenido de las medidas procesadas.

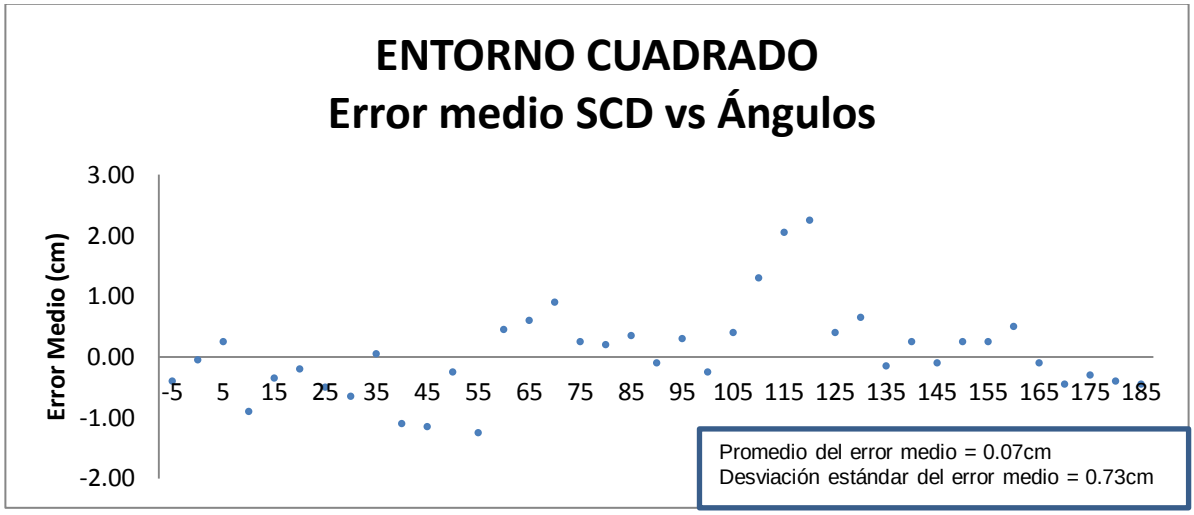


Figura 5.8. Gráfica del error medio SCD vs ángulos obtenidos en el experimento 1 prueba 2

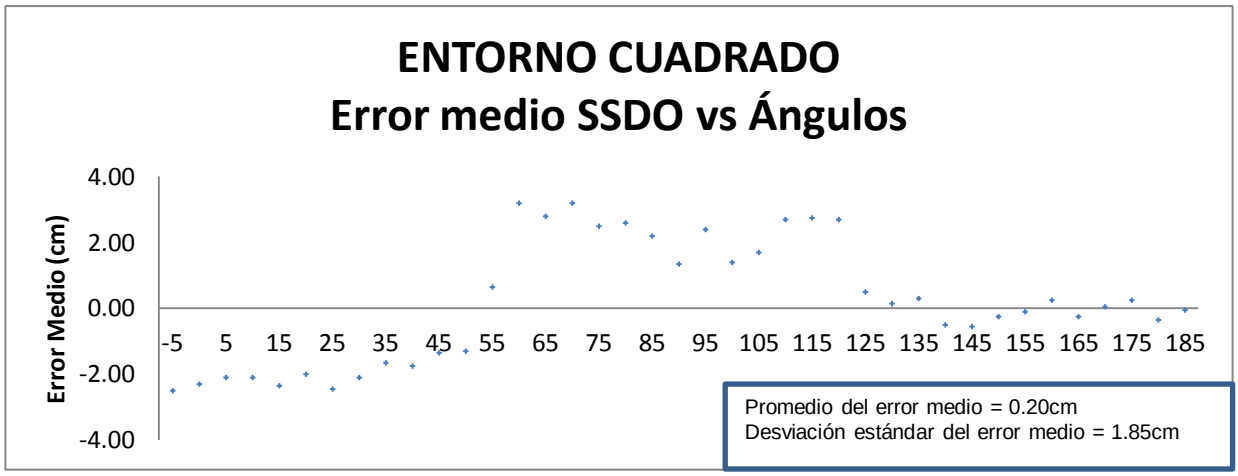


Figura 5.9. Gráfica del error medio SSDO vs ángulos obtenidos en el experimento 1 prueba 2

Con las distancias obtenidas en la prueba 2 y presentes en la Tabla 5.3 se realiza la gráfica mostrada en la Figura 5.10, donde se gráfica el perfil del entorno cuadrado, la línea azul representa el perfil obtenido con el SCD, la línea roja representa el perfil obtenido con el SSDO y la línea verde representa el perfil obtenido de las mediciones reales.

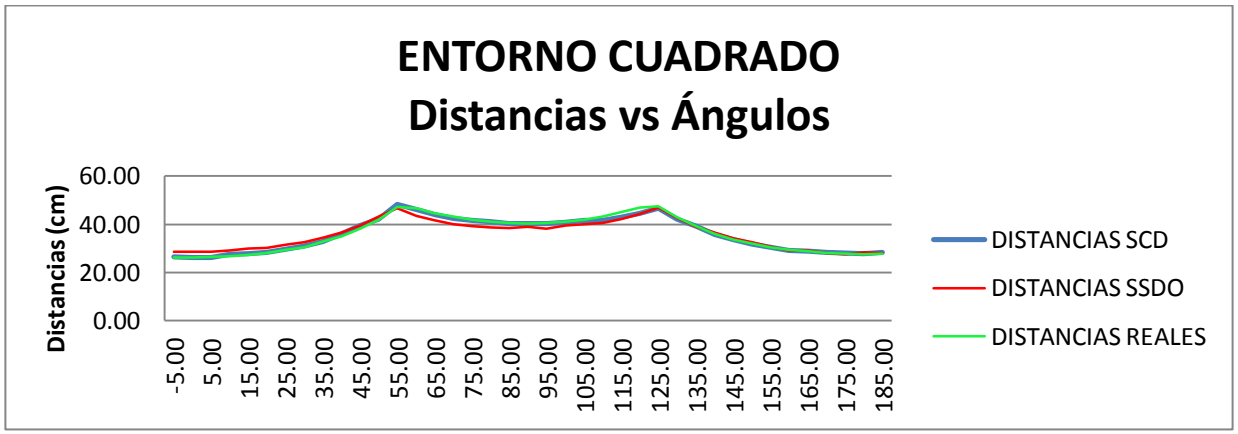


Figura 5.10 Gráfica de las Distancias vs Ángulos para las distancias del entorno cuadrado.

En la Tabla 5.5, se resumen los resultados de las dos pruebas, donde se tienen los promedios y las desviaciones estándar del error medio obtenido.

Tabla 5.5 Promedio y desviación estándar del error medio obtenido en el experimento 1 pruebas 1 y 2.

	PEM** (cm)		DEEM** (cm)	
	SCD	SSDO	SCD	SSDO
CIRCULO	0.18	0.18	0.54	0.55
CUADRADO	0.07	0.20	0.73	1.85

** PEM: Promedio del Error Medio, DEEM: Desviación Estándar del Error Medio

Se puede observar que las dos herramientas software se desempeñaron con el mismo comportamiento en la prueba del entorno circular, sin embargo en la prueba del entorno cuadrado hay una ventaja del SCD ante el SSDO.

Debido a que las dos herramientas software SCD y SSDO hacen uso del mismo sensor, y se realizo una óptima deducción de la trama de medición del radar laser, se debe esperar que las medidas entregadas por los dos sistemas sean idénticas, sin embargo el consolidado, del promedio y la desviación estándar de los errores medios, permite ver que no es así, sin embargo, esta situación tiene varias potenciales explicaciones:

1. Errores en el momento de la toma de la distancia real desde el escáner hasta la superficie del entorno. Vg. Cartón en movimiento, corrimiento del cartón, entre otros.

2. Errores de aproximación en las medidas reales debidas a la escala de medición del decámetro.

3. Error debido a la no exactitud en la coincidencia entre el eje coordenado del centro del radar laser y el eje coordenado dibujado en la cartulina.

4. Error en la coincidencia entre los 39 ángulos marcados en la cartulina y los 39 ángulos seleccionados del SCD y el SSDO.

5. Errores en la medida en línea debido a movimiento del entorno durante el momento de la toma de datos con cada una de las herramientas software.

6. Error inherente al instrumento radar laser.

7. Errores introducidos en la manipulación de los datos procesados del SCD, durante la conversión de formato de imagen a texto.

8. Errores en la metodología

5.3 EXPERIMENTO 2: VALIDACIÓN DEL SSDO EN INTERIORES

Este experimento consta de tres pruebas comparativas, una de las pruebas con un entorno circular, otra con un entorno cuadrado y la otra en un pasillo del segundo piso de la Facultad de Ingeniería Electrónica y Telecomunicaciones; para las pruebas del entorno circular y el entorno cuadrado el escáner láser se sigue colocando sobre el suelo y para la prueba del pasillo el escáner láser se coloca en una silla de aproximadamente 38cm de alto, ya que para distancias mayores a un metro el fabricante recomienda que el escáner láser se encuentre al menos a una altura de 30cm.

Prueba 1. Entorno circular: Esta prueba se realiza con el mismo entorno circular de 50 cm de radio creado anteriormente (ver Figura 5.11), pero se grafican los resultados con las 761 medidas de cada barrido, en cada una de las herramientas software.

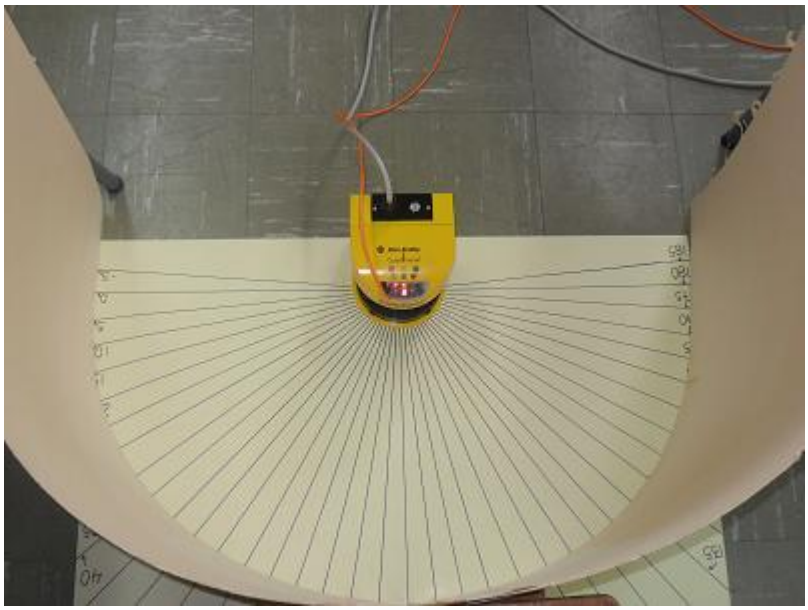


Figura 5.11 Entorno circular del experimento 2 prueba 1.

Prueba 2. Entorno cuadrado: Esta prueba se realiza con el mismo entorno cuadrado de 54 cm de lado utilizado en el experimento 1 prueba 2 (ver Figura 5.12) y se grafican las 761 medidas de cada barrido con cada una de las herramienta software.

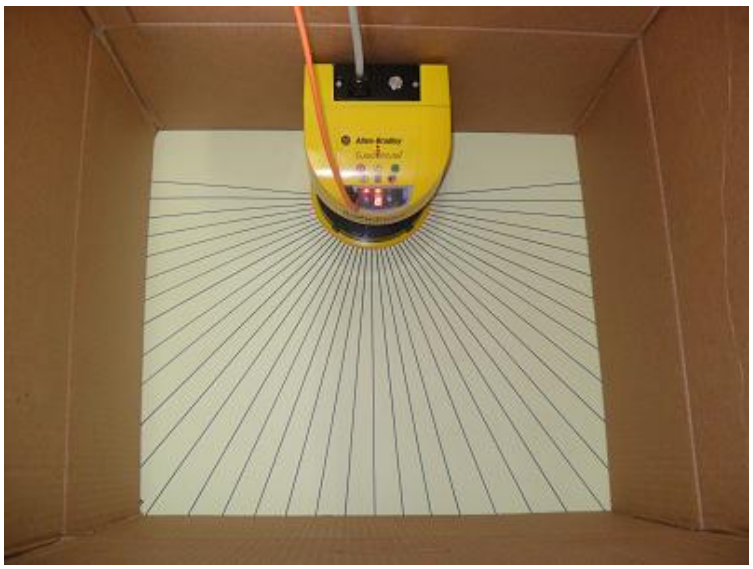


Figura 5.12 Entorno cuadrado del experimento 2 prueba 2.

Prueba 3. Pasillo: Se trabaja con el entorno que se muestra en la Figura 5.13, donde se procesa las 761 medidas de cada barrido en cada una de las herramientas software.

Figura 5.13 Pasillo de la Facultad de Ingeniería Electrónica y Telecomunicaciones.

En las tres pruebas se toma el promedio de las distancias en cada ángulo para veinte (20) barridos usando las distancias entregadas por el SCD como la medida de referencia. Para estas distancias obtenidas se debe hacer uso de la herramienta TopOCR para pasar los datos de imágenes a texto, ya que no es posible la exportación de los datos en forma de texto. Una vez se tienen los datos

en forma de texto tanto con el SCD como con el SSDO, se procede a realizar los promedios de las 20 medidas obtenidas para cada uno de las 761 distancias y con estos promedios poder realizar el cálculo de los errores medios y la desviación estándar. Las tablas de todas las distancias, promedios, errores medios y desviaciones estándar se pueden consultar en el Anexo 4: “Resultados de la validación del sistema”.

Los resultados obtenidos se muestran a continuación:

Para el entorno circulas se tiene en la Tabla 5.6 se muestra el promedio y la desviación estándar de los errores medios entre el SCD y el SSDO y su respectiva gráfica se observa en la Figura 5.14, donde se grafican los 761 errores medios entre el SCD y el SSDO para el experimento 2 prueba 1.

Tabla 5.6 Promedio y desviación estándar de los errores medios para el experimento 2 prueba 1.

	PEM** (cm)	DEEM** (cm)
Entorno circular	-0.08	0.41

** PEM: Promedio del Error Medio, DEEM: Desviación Estándar del Error Medio

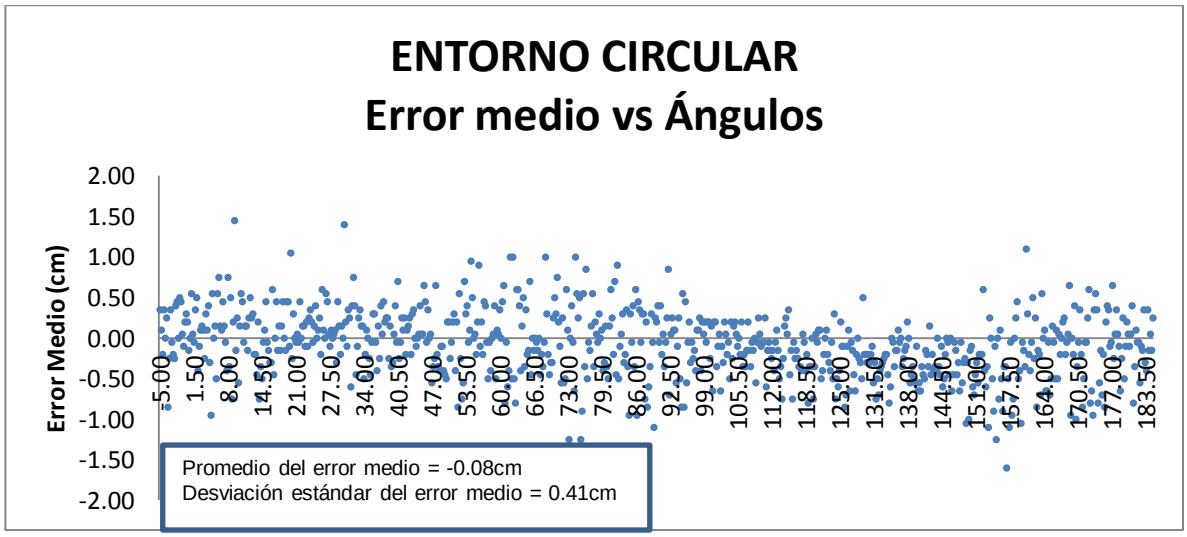


Figura 5.14. Gráfica del error medio vs ángulos obtenidos en el experimento 2 prueba 1

Se muestran en la Figura 5.15 las distancias obtenidas del SCD y el SSDO para la prueba 1, donde se gráfica el perfil del entorno circular, la línea azul representa el perfil obtenido con el SCD y la línea roja representa el perfil obtenido con el SSDO

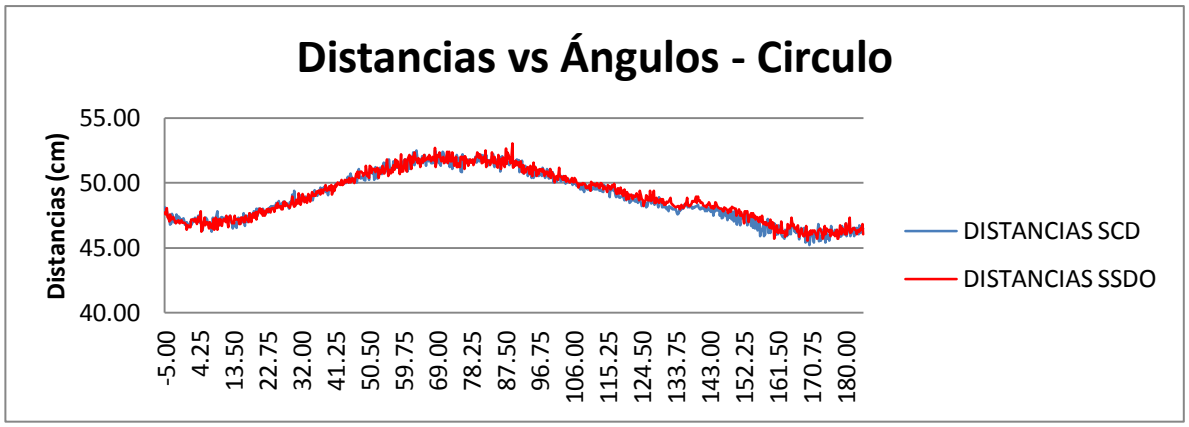


Figura 5.15 Distancias obtenidas del SCD Y SSDO en el experimento 2 prueba 1.

Se grafican los perfiles de línea obtenidos con cada herramienta software en el experimento 2 prueba 1 (ver Figura 5.16), al lado derecho se observa el perfil de línea obtenido con el SSDO y al lado izquierdo el perfil de línea obtenido con el SCD, para el entorno circular de la prueba 1.

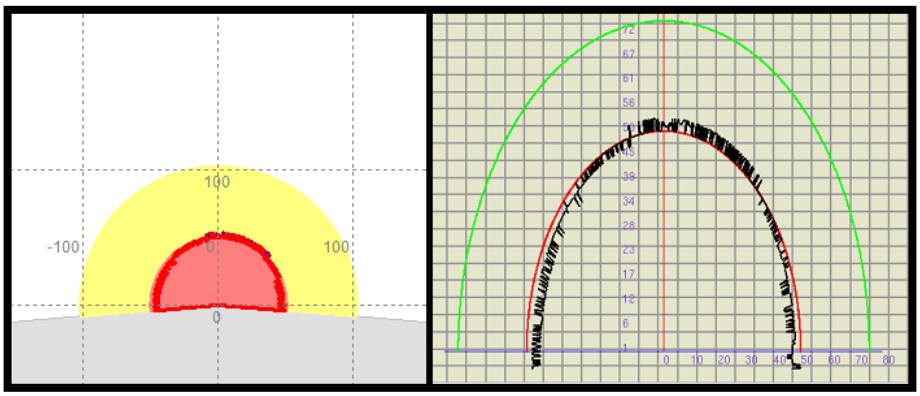


Figura 5.16 Perfil de línea obtenido con el SCD y el SSDO para el entorno circular del experimento 2 prueba 1.

Para el entorno cuadrado se tiene en la Tabla 5.7 se muestra el promedio y la desviación estándar de los errores medios entre el SCD y el SSDO y su respectiva gráfica mostrada en la Figura 5.17 se muestran los errores medios entre el SCD y el SSDO para el experimento 2 prueba 2.

Tabla 5.7 Promedio y desviación estándar de los errores medios para el experimento 2 prueba 2.

	PEM** (cm)	DEEM** (cm)
Entorno cuadrado	0.08	1.50

** PEM: Promedio del Error Medio, DEEM: Desviación Estándar del Error Medio

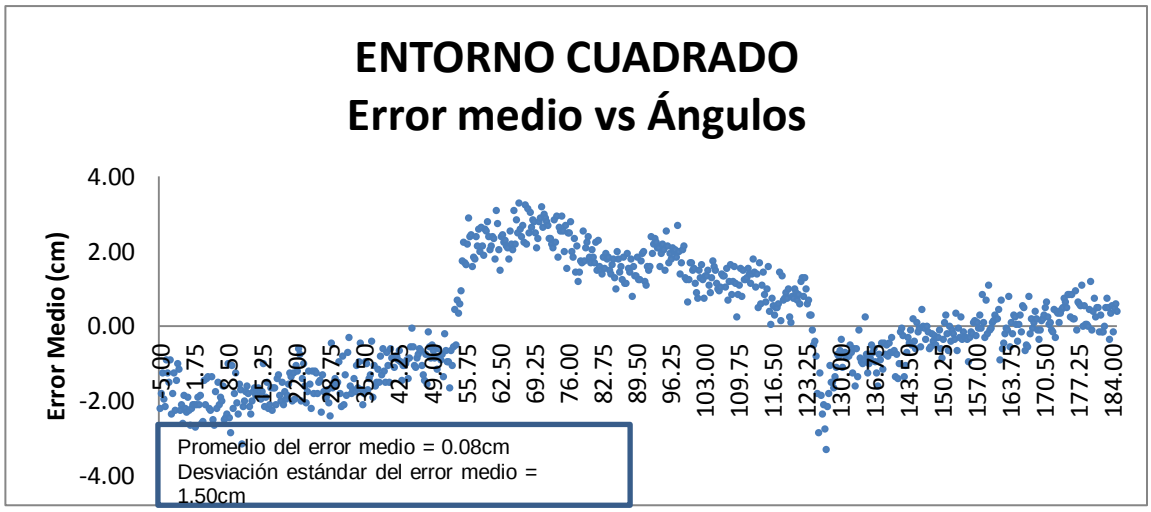


Figura 5.17. Gráfica del error medio vs ángulos obtenidos en el experimento 2 prueba 2

Se tiene en la Figura 5.18 las líneas obtenidas de las distancias del SCD y el SSDO para la prueba 2, donde se gráfica el perfil del entorno cuadrado, la línea azul representa el perfil obtenido con el SCD y la línea roja representa el perfil obtenido con el SSDO

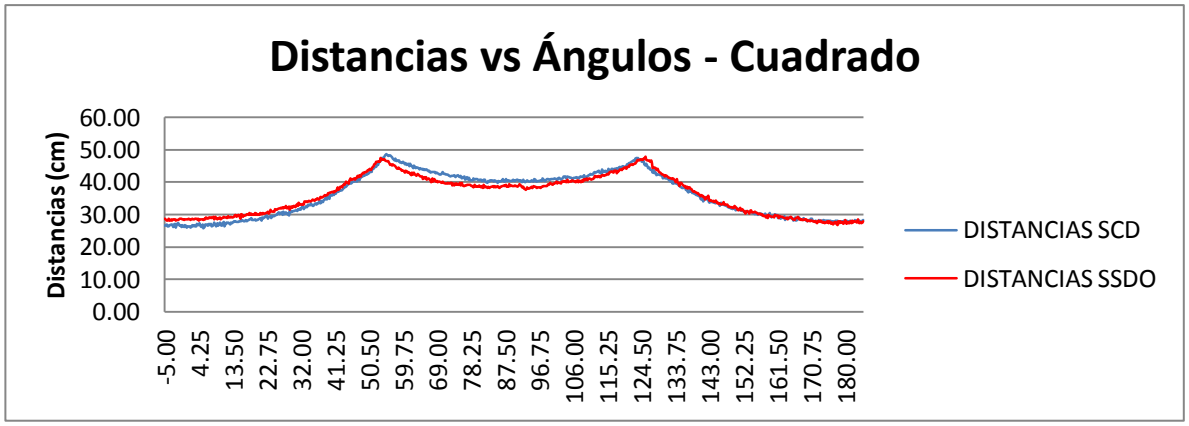


Figura 5.18 Distancias obtenidas del SCD Y SSDO en el experimento 2 prueba 2.

Se grafican los perfiles de línea obtenidos con cada herramienta software en el experimento 2 prueba 2 (ver Figura 5.19), al lado derecho se observa el perfil de línea obtenido con el SSDO y al lado izquierdo el perfil de línea obtenido con el SCD, para el entorno cuadrado de la prueba 2.

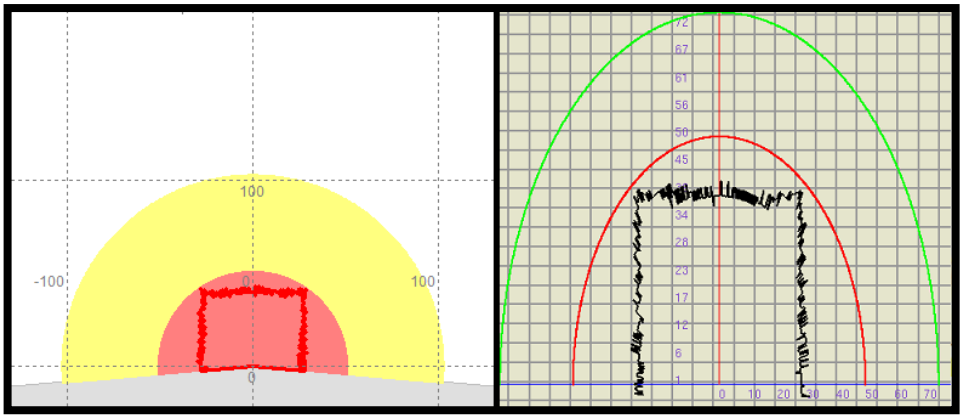


Figura 5.19 Perfil de línea obtenido con el SCD y el SSDO para el entorno cuadrado del experimento 2 prueba 2.

Se ilustra un comportamiento interesante en los datos obtenidos de la prueba 2 con cada herramienta software (ver Figuras 5.19 y 5.20), sin embargo la respuesta a tal disparidad se encuentra en un corrimiento del entorno cuadrado durante el cambio de software SCD y SSDO.

El promedio y la desviación estándar de los errores medios entre el SCD y el SSDO en el experimento 2 prueba 3 realizada en el pasillo se tiene en la Tabla 5.8 y se gráfican los errores medios entre el SCD y el SSDO (ver Figura 5.20).

Tabla 5.8 Promedio y desviación estándar de los errores medios para el experimento 2 prueba 3.

	PEM** (cm)	DEEM** (cm)
Entorno del pasillo	0.93	1.50

** PEM: Promedio del Error Medio, DEEM: Desviación Estándar del Error Medio

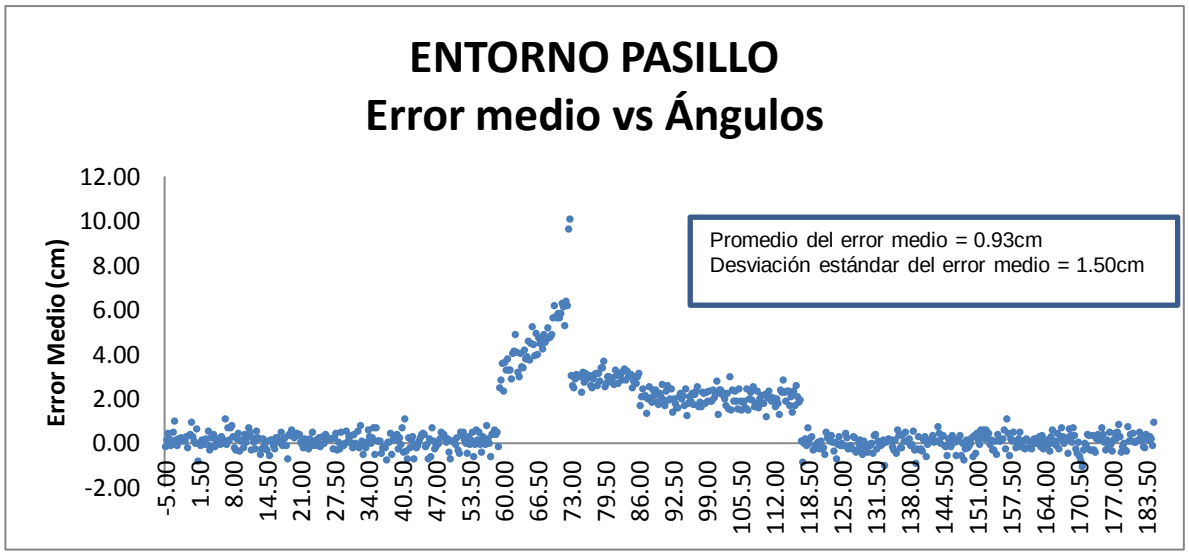


Figura 5.20. Gráfica del error medio vs ángulos obtenidos en el experimento 2 prueba 3

Las distancias obtenidas del SCD y el SSDO para el experimento 2 prueba 3 se muestran en la Figura 5.21, donde se gráfica el perfil del entorno pasillo, la línea azul representa el perfil obtenido con el SCD y la línea roja representa el perfil obtenido con el SSDO.

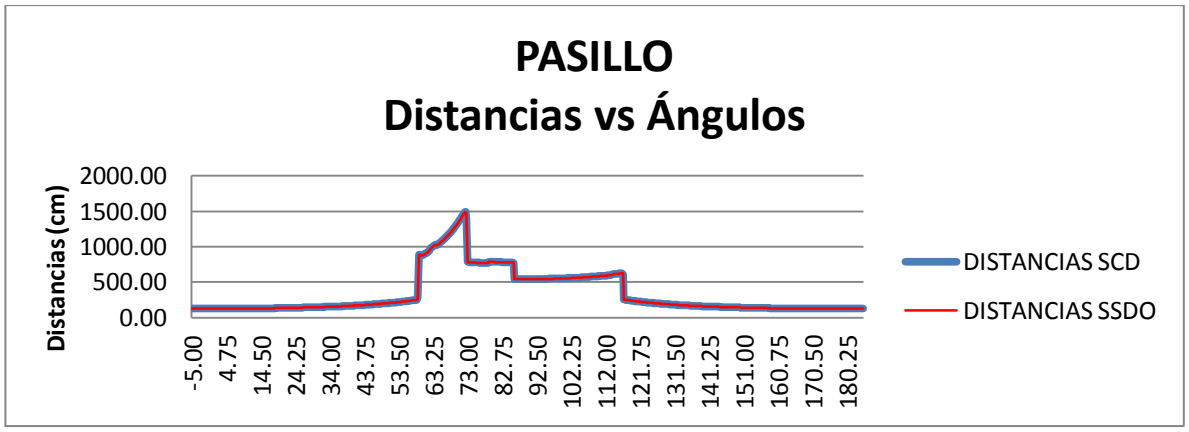


Figura 5.21 Distancias obtenidas del SCD Y SSDO en el experimento 2 prueba 3.

Los perfiles de línea obtenidos con cada herramienta software en el experimento 2 prueba 3 se ilustran en la Figura 5.16, al lado derecho se observa el perfil de línea obtenido con el SSDO y al lado izquierdo el perfil de línea obtenido con el SCD, para el pasillo de la prueba 3.

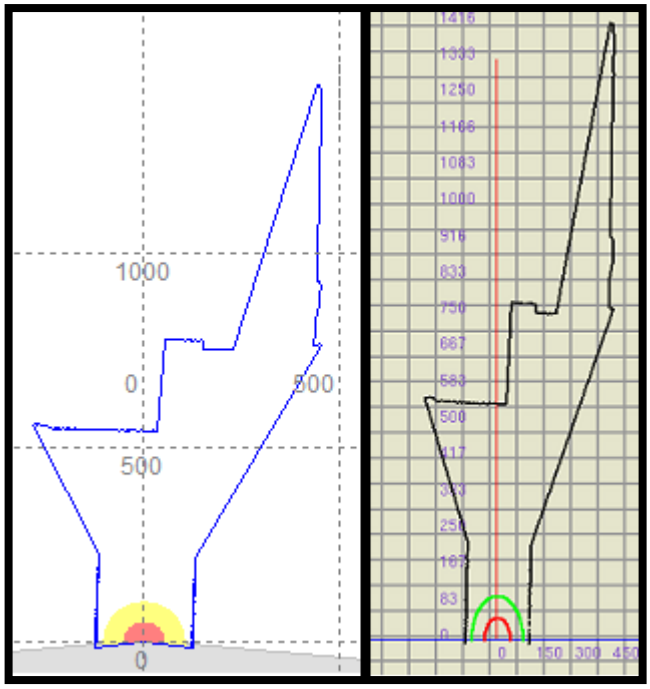


Figura 5.22 Perfil de línea obtenido con el SCD y el SSDO para el entorno del pasillo del experimento 2 prueba 3.

La Figura 5.22 presenta un comportamiento de interés en el error medio de los datos, donde se aprecia que en la zona central, alrededor de los ángulos 60 y 118 grados, el valor del error medio aumenta. Un análisis posterior permite explicar que este aumento del error está asociado a una mayor distancia de medición en la zona central del pasillo.

Se resumen los promedios y las desviaciones estándar del error medio obtenidos en las pruebas 1, 2 y 3 (ver Tabla 5.9).

Tabla 5.9 Promedio y desviación estándar del error medio obtenido en el experimento 2 pruebas 1, 2 y 3.

	PEM** (cm)	DEEM** (cm)
CIRCULO	-0.08	0.41
CUADRADO	0.08	1.50
PASILLO	0.93	1.50

** PEM: Promedio del Error Medio, DEEM: Desviación Estándar del Error Medio

Se puede observar que para la prueba del entorno circular, el entorno cuadrado y el pasillo en el promedio y la desviación estándar del error medio entre el SCD y el SSDO se tienen valores semejantes con los obtenidos en el experimento 1. Sin embargo los resultados obtenidos en la prueba 3 permiten afirmar que no se

puede manejar un único valor de error para todas las distancias, lo que permite inferir que para distancia pequeñas el error es bajo y a medida que las distancias aumentan el error también lo hace.

5.3 VALIDACIÓN DEL SSDO EN EXTERIORES

Este experimento consta de dos pruebas, ambas pruebas se realizan en el exterior de los laboratorios de Ingeniería Física. Una de estas pruebas se realiza en horas de la tarde y la otra en la noche; uno de los entornos se toma con distancias medias aproximadamente entre 60cm y 715cm y el otro entorno a grandes distancias aproximadamente entre 94cm y 5120cm. Para estas dos pruebas el escáner láser se coloca en una silla de aproximadamente 38cm de alto.

Prueba 1 distancias medias: Esta prueba se realiza para el entorno con distancias medias, en horas de la tarde (ver Figura 5.23) y se procesan las mediciones en cada una de las herramientas software.

Figura 5.23 Imágenes en forma de barrido del entorno del exterior para el experimento 3 prueba 1 distancias medias: orden de las imágenes 1, 2, 3 y 4.

Prueba 2: Esta prueba se realiza para el entorno para grandes distancias, en horas de la noche (ver Figura 5.24) y se procesan las mediciones en cada una de las herramientas software.

Figura 5.24 Entorno del exterior del experimento 3 prueba 2 para grandes distancias.

En las dos pruebas se toma el promedio de las distancias en cada ángulo para veinte (20) barridos obtenidas por el SCD como la medida de referencia. Para estas distancia obtenidas se debe hacer uso de la herramienta TopOCR para pasar los datos de imágenes a texto, ya que no es posible la exportación de los datos en forma de texto, una vez se tienen los datos en forma de texto tanto con el SCD como con el SSDO, se procede a realizar los promedios de las 20 medidas obtenidas para cada uno de las 761 distancias y con estos promedios poder realizar el cálculo de los errores y desviación estándar. Las tablas de todas las distancias, promedios, errores y desviaciones estándar se pueden consultar en el Anexo 4: “Resultados de la validación del sistema”.

Los resultados obtenidos en este experimento se muestran a continuación:

El promedio y la desviación estándar de los errores medios entre el SCD y el SSDO en experimento 3 prueba 1se tienen en la Tabla 5.610 y se grafican los errores medios entre el SCD y el SSDO (ver Figura 5.14).

Tabla 5.10 Promedio y desviación estándar de los errores medios para el experimento 3 prueba 1.

	PEM** (cm)	DEEM** (cm)
Exteriores distancias medias	0.84	0.95

** PEM: Promedio del Error Medio, DEEM: Desviación Estándar del Error Medio

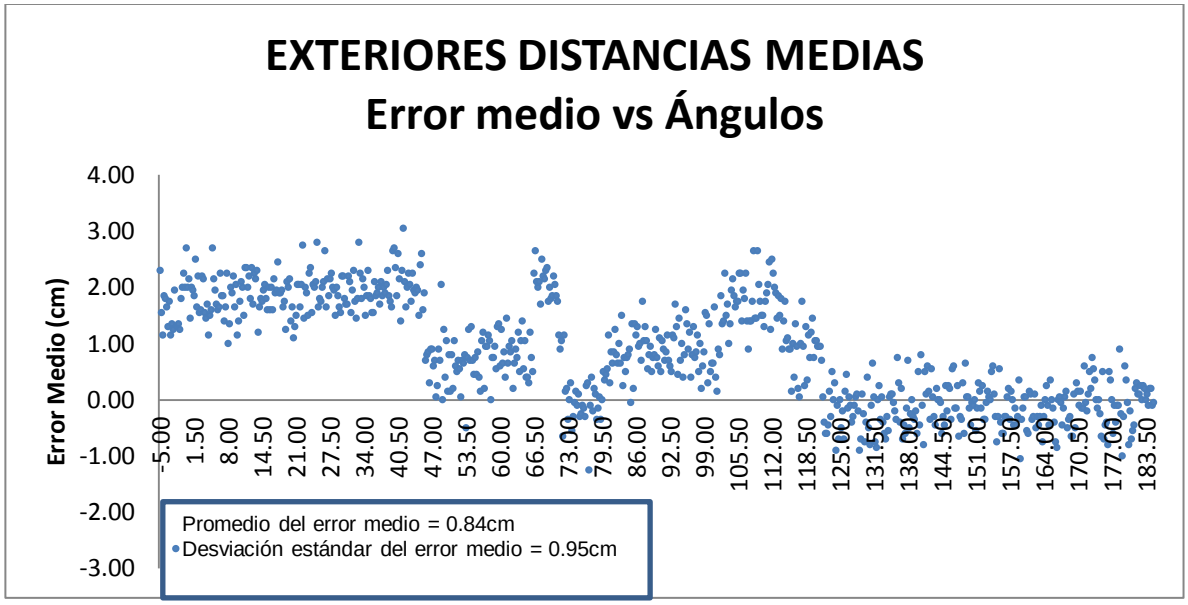


Figura 5.25. Gráfica del error medio vs ángulos obtenidos en el experimento 3 prueba 1

Las distancias obtenidas del SCD y el SSDO para el experimento 3 prueba 1 se gráfican y se obtiene el perfil de exteriores para distancias medias (ver Figura 5.15), la línea azul representa el perfil obtenido con el SCD y la línea roja representa el perfil obtenido con el SSDO.

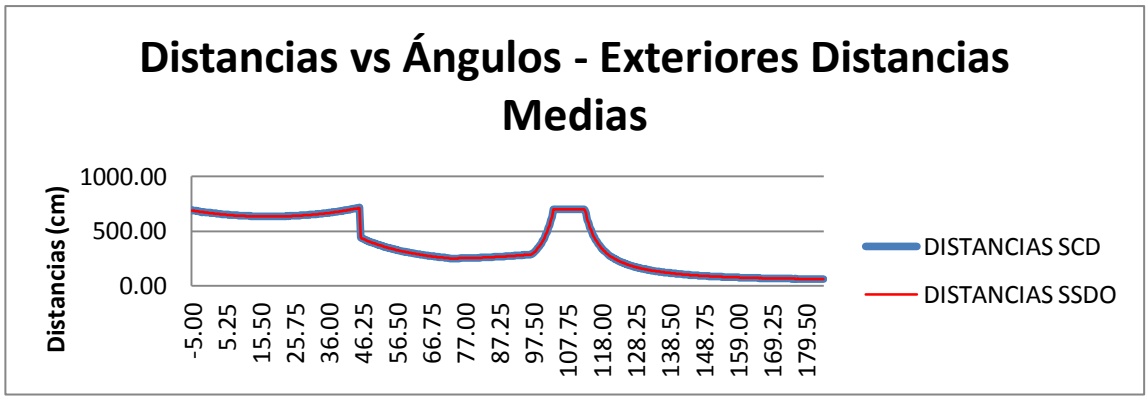


Figura 5.26 Distancias obtenidas del SCD Y SSDO en el experimento 3 prueba 1.

Los perfiles de línea obtenidos con cada herramienta software en el experimento 3 prueba 1 se muestran en la Figura 5.16, donde al lado derecho se observa el perfil de línea obtenido con el SSDO y al lado izquierdo el perfil de línea obtenido con el SCD, en exteriores para distancias medias de la prueba 1.

Figura 5.27 Perfil de línea obtenido con el SCD y el SSDO para el entorno circular del experimento 2 prueba 1.

El promedio y la desviación estándar de los errores medios entre el SCD y el SSDO en experimento 3 prueba 2 se tiene en la Tabla 5.611 y se grafican los errores medios entre el SCD y el SSDO (ver Figura 5.14).

Tabla 5.11 Promedio y desviación estándar de los errores medios para el experimento 3 prueba 2.

	PEM** (cm)	DEEM** (cm)
Exteriores distancias grandes	3.32	4.03

** PEM: Promedio del Error Medio, DEEM: Desviación Estándar del Error Medio

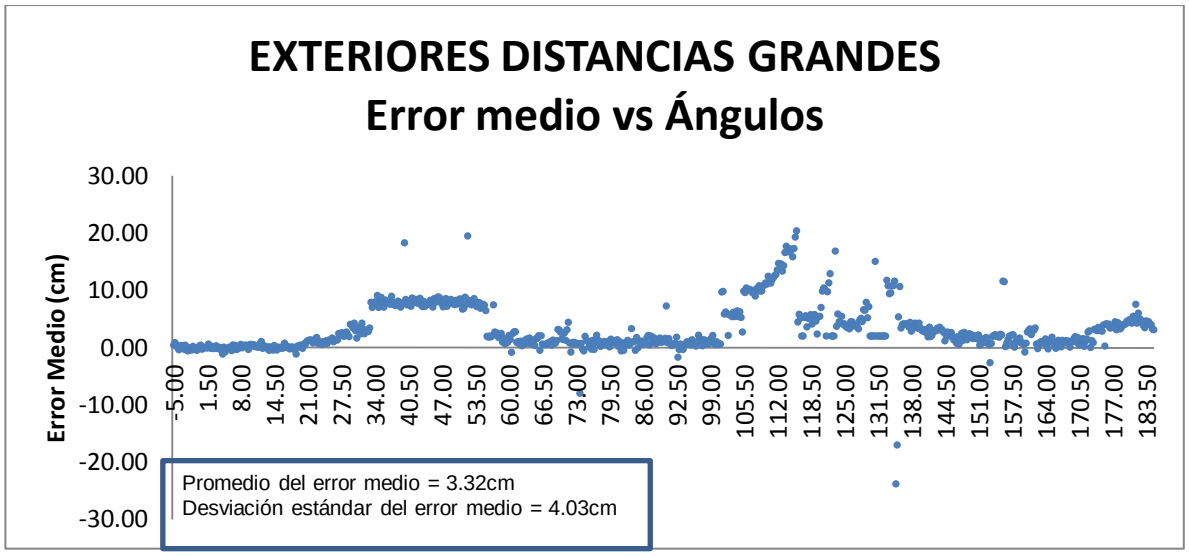


Figura 5.28. Gráfica del error medio vs ángulos obtenidos en el experimento 3 prueba 1

Las distancias obtenidas del SCD y el SSDO para el experimento 3 prueba 2 se gráfican obteniendo como resultado el perfil de exteriores para distancias grandes (ver Figura 5.15), la línea azul representa el perfil obtenido con el SCD y la línea roja representa el perfil obtenido con el SSDO.

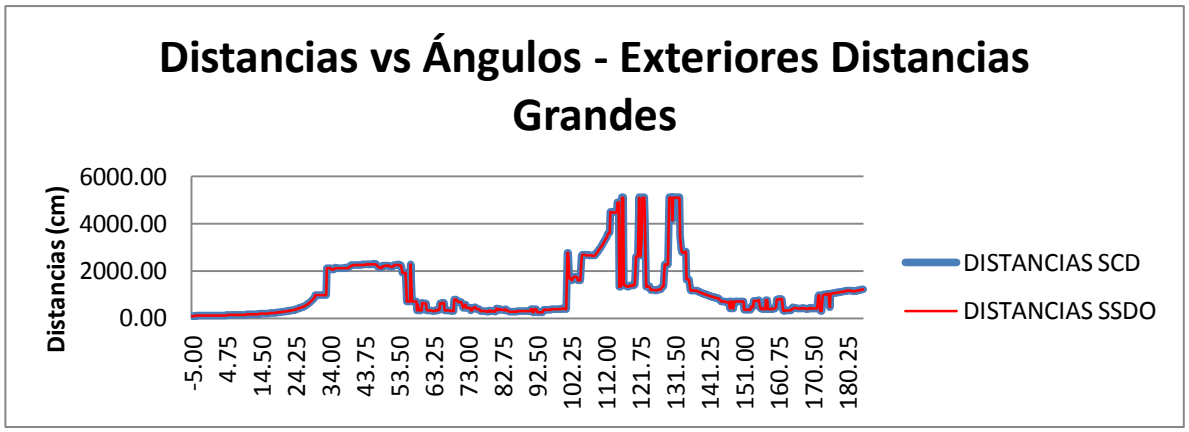


Figura 5.29 Distancias obtenidas del SCD Y SSDO en el experimento 3 prueba 2.

En la prueba 2 se obtiene un perfil de línea con cada uno de los software, en la Figura 5.16, al lado derecho se observa el perfil de línea obtenido con el SSDO y al lado izquierdo el perfil de línea obtenido con el SCD, en exteriores para distancias grandes de la prueba 2.

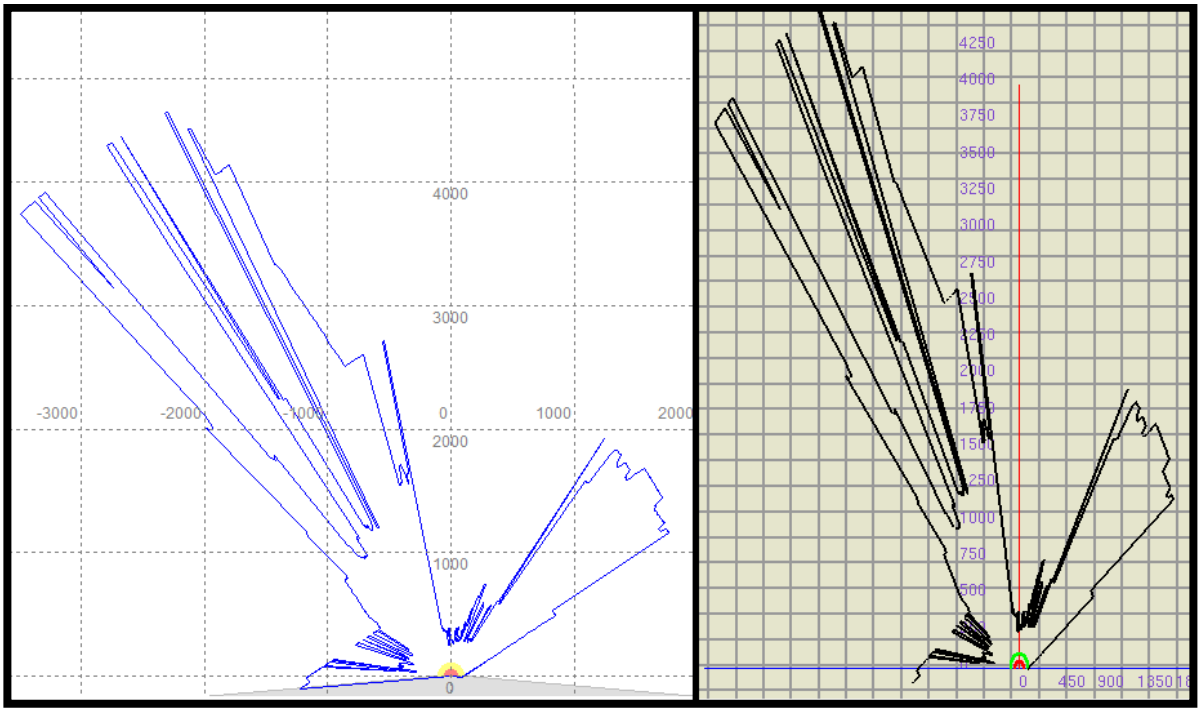


Figura 5.30 Perfil de línea obtenido con el SCD y el SSDO para el entorno circular del experimento 2 prueba 1.

Interpretación de los resultados de las pruebas de validación en interiores y exteriores

En la Tabla 5.9, se resumen los promedios y las desviaciones estándar del error medio obtenido en las pruebas 1 y 2.

Tabla 5.12 Promedio y desviación estándar de los errores medios obtenidos en el experimento 3 pruebas 1 y 2.

EXTERIORES	PEM** (cm)	DEEM** (cm)
Distancias Medias	0.84	0.95
Distancias Grandes	3.32	4.03

** PEM: Promedio del Error Medio, DEEM: Desviación Estándar del Error Medio

De la tabla anterior se puede observar que en la prueba de exteriores – distancias medias el promedio y la desviación estándar del error medio entre el SCD y el SSDO es similar en comparación con los datos obtenidos en el experimento 2, pero para los exteriores de grandes distancias estos valores incrementan significativamente. Esto ultimo va en concordancia con el análisis realizado en la interpretación de los resultados del experimento 2,donde se determina que el error medio aumenta con la mayor longitud de la distancia medida por el radar laser.

Las tablas de los resultados obtenidos en cada una de estas pruebas se pueden revisar en el Anexo 4: “Resultados de la validación del sistema”.

CONCLUSIONES

- Se dedujeron las tramas de datos entre el escáner láser y el software SCD de Rockwell Automation que se ejecutan en un computador, donde se realizaron una serie de experimentos que permitieron finalmente la identificación de la trama de datos SCD>Escáner y la trama de datos de las distancias Escáner > SCD.

- Se diseñó e implemento un sistema de detección de obstáculos en el que se tuvo en cuenta los requerimientos generales para un potencial sistema de conducciòn de robots moviles a partir de las tramas de datos detectadas para establecer una conexión entre el escáner láser *Safe Zone Multizone* de *Allen Bradley* y un computador.

- Se implementó en la plataforma software de robótica móvil Player el *plugin driver* necesario para establecer conexión entre el escáner láser *Safe Zone Multizone* de *Allen Bradley* y un computador y entregar esta información a la Apliaciòn Software de Deteccion Obstaculos (ASSD) para poder procesar los datos entregados por el *plugin*.

- Se validó el sistema láser para detección de obstáculos implementado. Esta validación se realizó tanto en interiores como en exteriores, teniendo como resultado que el sistema implementado es altamente confiable tomando como punto de comparación la herramienta software SCD que viene con el radar laser *Safe Zone Scanner Laser* de *Allen Bradley.*

- Se logró independizar el escáner láser *Safe Zone Scanner Laser* de *Allen Bradley* del software SCD de *Rockwell Automation*, lo que permite establecer conexión entre el escáner y un computador tanto en Linux como en Windows, permitiendo extraer la información entregada por parte del dispositivo, ya que el software SCD no permite la exportación de esta información.

- Los resultados obtenidos en la validacion del sistema son un gran aporte para la robotica movil, ya que los errores en distancia son considerablemente pequeños (<3cm para distancias de 30m a 50m y <1cm para distancias menores a 10m) si se tiene en cuenta que el robot toma decisiones en aproximadamente 50 cm para cuidar su integridad.

TRABAJOS FUTUROS

- Diseñar e implementar un módulo para realizar la configuración con el fin de independizar completamente el escáner láser del SCD.

- Crear una plataforma que permita que el SDO no trabaje datos solamente en 2D, si no que entregue datos y gráficos en 3D.

- Integrar el SDO con un sistema que haciendo uso de la información entregada y de las funcionalidades del escáner láser sea capaz de tomar decisiones.

- Realizar un sistema de fusión sensorial que integre el sistema láser para detección de obstáculos y el sistema de visión estereoscópica para aplicaciones robóticas.

- Integrar el SDO con un GPS para poder entregar información exacta de la posición de los objetos identificados en los diferentes entornos en los que se encuentre.

REFERENCIAS

[1] DILLANES LUNA, Alejandro. (2002). Sistema de Exploración y Visualización de Obstáculos en un Ambiente con Robot Móvil Khepera. Universidad de las Américas Puebla. Cholula, Puebla, México.

[2] Asociación Internacional de Estándares (ISO). (1994). Ésta en su norma ISO 8373 (en España corresponde a la UNE EN ISO 8373:1998. «Robots Manipuladores Industriales. Vocabulario»).

[3] NILSSON, Nils J. (1984). Shakey the robot. Technical Report 323. Artificial Intlligence Center, SRI international.

[4] R. Silva Ortigoza; J. R. García Sánchez; V. R. Barrientos Sotelo; M. A. Molina Vilchis; V. M. Hernández Guzmán; G. Silva Ortigoza. (2007). "Una panorámica de los robots móviles", Revista electrónica de estudios telemáticos, ISSN: 1856-4194. Volumen 6 Edición No 3.

[5] DUEÑAS RODRIGUEZ, Francisco Armando. (2008). La Robótica. Universidad La Salle. Cancún, Quintana Roo México.

[6] Gonzalez J. Javier, Ollero B. Anibal. (1996). Estimación de la Posición de un Robot Móvil. Universidad de Málaga, Universidad de Sevilla.

[7] Mallet. Anthony, Fleury, Sara, Bruyninckx, Herman. (2003). A specification of generic robotics software components: future evolutions of GenoM in the Orocos context. LAAS, CNRS, KU-Leuven.

[8] Panos J. Antsaklis, Kevin M. Passino. (1993). An Introduction to Intelligent and Autonomous Control. ISBN: 0-7923-9267-1.

[9] Dirk Thomas and Oskar von Stryk. (2010) Efficient Communication in Autonomous Robot Software. In Proceedings of 2010 IEEE/RJ - International Conference on Intelligent Robots and Systems (IROS). Taipei, Taiwan, Oct. 18 – 22.

[10] Página web. <http://www2.parc.com/spl/projects/modrobots/chain /polybot/parc/doc/tutorial/lesson1.html>. Consulta en Marzo de 2011.

[11] SÁNCHEZ MILLARES, Álvaro. (2006). Sensores y actuadores.pdf <http://www.iit.upcomillas.es/~alvaro/teaching/Clases/Robots/teoria/Sensores%20y%20actuadores.pdf >. Universidad Pontificia Comillas (ICAI).

[12] Jonathan Roberts & Peter Corke. Obstacle Detection for a Mining Vehicle using a 2D Laser. CSIRO Manufacturing Science & Technology. PO Box 883, Kenmore, Qld 4069, Australia

[13] SOLORZANO, Reynolds Bonifaz. (2003, Enero). Robots de Exploración. Universidad de las Américas Puebla. Cholula, Puebla, México.

[14] HELLSTRÖM, Thomas. (2002). Autonomous Navigation for Forest Machines. Umea University. Sweden.

[15] Página web. <http://es.wikipedia.org/wiki/LIDAR>. Consultada en marzo 2011.
[16] Edited by WEBB, Colin E., JONES, Julian. (2004). Handbook of Laser Technology and Applications. Vol I.
[17] Edited by BASS, Michael. Hanbook of Optics. (1995). Sponsored by the Optical Society of America. Vol I.
[18] Página web. < http://www.rockwellautomation.es/applications/gs/emea/gses.nsf/pages/update_2009-14_12>. Consultada en Marzo de 2011
[19] Allen Bradley. (2005). User Manual Safe Zone Multizone Safety Laser Scanner. Rockwell Automation.
[20] Allen Bradley. (2007). Protección de escáner láser para carro transportador automatizado con relé de seguridad modular Serie MSR200. Rockwell Automation.
[21] SINGH, Sanjiv y WEST, Jay. (1991, Septiembre). *Cyclone*: *A Laser Scanner for Mobile Robot Navigation*.
[22] Baifan Chen, Zixing Cai, Zheng Xiao, Jinxia Yu, Limei Liu. (2008, Noviembre). "*Real-time Detection of Dynamic Obstacle Using Laser Radar*" *IEEE* 18-21.
[23] J. E. Naranjo, C. González, J. Reviejo, R. García, and T. de Pedro. (2003, Septiembre) "Adaptive fuzzy control for inter-vehicle gap keeping," *IEEE Trans. Intell. Transp. Syst.*, vol. 4, no. 3, pp. 132–142. Special Issue on ACC.
[24] J. E. Naranjo, C. González, J. Reviejo, R. García, and T. de Pedro.(2007, Julio) "Adaptive fuzzy control for inter-vehicle gap keeping," *IEEE* Transactions on Vehicular Technology, vol. 56, no. 4, pp. 1623 – 1630.
[25] Página web. <http://www.moisesdaniel.com/bloc /archives /2007/12/entry_17.html>. Consultada en Febrero de 2011
[26] MARULANDA, Juan Fernando. (2009). "Arquitectura Genérica para Sistemas de Robótica Móvil" Tesis de Maestría. Universidad del Cauca. Popayán, Colombia.
[27] David Wettergreen, Hans Thomas y Chuck Thorpe. (1990). Planning Strategies for the Ambler Walking Robot. The Robotics Institute, Carnegie Mellon University.
[28] SIMMONS, Reid. (2001). A Brief History of Our Research in Task-Level Control Architectures.
[29] SIMMONS, Reid. (1994). Structure Control for Autonomous Robots. IEEE Transcriptions on Robotics and Automation. Vol. 10.
[30] SIMMONS, Reid. (1994). Task Control Architecture: Structure Control for Autonomous Robots.
[31] Página web. <http://www.teambots.org/>. Consultada en Marzo de 2011.

[32] Página web. < http://www.teambots.org/src/EDU/gatech/cc/is/clay/index.html>. Consultada en marzo de 2011.
[33] Otto Colomina y Miguel Ángel Cazorla. (2007). Introducción a Player / Stage.
[34] Página web. <http://playerstage.sourceforge.net/index.php?src=gazebo>. Consultada en marzo de 2011.
[35] Página web. <http://playerstage.sourceforge.net/index.php?src=index>. Consultada en Marzo de 2011.
[36] Página web. <http://www.openrobots.org/wiki/genom>. Consultada en Marzo de 2011.
[37] Sara Fleury, Matthieu Herrb y Raja Chatila. GeNom: A Tool for the Specification and the Implementation of Operating Modules in a Distributed Robot Architecture. LAAS – CNRS.
[38] Página web. <http://smart-robotics.sourceforge.net/overview.php>. Consultada en Marzo de 2011.
[39] Nick Hawes, Michael Zillich y Jeremy Wyatt. (2007). BALT & CAST: Middleware for Cognitive Robotics. University of Birmingham.
[40] H. Chishiro, Y. Fujita, A. Takeda, Y. Kojima, K. Funaoka, S. Kato y N. Yamasaki. Extended RT – Component Framework for RT – Middleware. School of Science for Open and Environment Systems Keio University. Japan.
[41] D. Krüger, I. Lil, N. Sünderhauf, R. Baumgartl y P. Protzel. Using and Extending the Miro Middleware for Autonomous Mobile Robots. University of Technology.
[42] N. Mohamed, J. Al-Jaroodi y I. Jawhar. (2008) Middleware for Robotics: A Survey. United Arab Emirates University.
[43] N. Mohamed, J. Al-Jaroodi y I. Jawhar. (2009). A Review of Middleware for Networked Robots. UAE University.
[44] A. Brooks, T. Kaupp, A. Makarenko, S. Williams y A. Orebä. (2006). ORCA: A Component Model and Repository.
[45] V. Andronache y M. Scheutz. ADE – An Architecture Development Environment for Virtual and Robotic Agents. University of Notre Dame.
[46] CÁCERES, Diego Alonso. (2008) Desarrollo de software para robots de servicio: Un enfoque dirigido por modelos y orientado a componentes. Universidad Politécnica de Cartagena. Tesis Doctoral.
[47] BRUYNINCKX, Herman. (2008). Simulation, Modeling and Programming for Autonomous Robots: The Open Source Perspective.
[48] KRÜGER, Daniel. (2005). Porting the MIRO Middleware to a Mobile Robot Platform. Chemnitz University of Technology.
[49] C. Côté; Y. Brosseau, D. Létourneau, C. Raïevsky, F. Michaud. Robotic Software Integration Using MARIE. Université de Sherbrooke.

[50] Página web. <http://playerstage.sourceforge.net/doc/Player-2.1.0/player/index.html>. Consultada en Marzo de 2011.
[51] Página web. <http://playerstage.sourceforge.net/index.php?src=player>. Consultada en Marzo de 2011.
[52] Página web. <http://playerstage.sourceforge.net/index.php?src=stage>. Consultada en Marzo de 2011.
[53] OWEN, Jennifer. (2010). Cómo usar Player/Stage.
[54] Mark Kilgard. (1994, Nov/Dic). “An OpenGL Toolkit,” The X Journal, SIGS Publications.
[55] Página web, <http://www.hhdsoftware.com/Downloads/serial-port-monitoring-control>. Consultada en marzo de 2011.
[56] Página web, <http://hw-server.com/software/termv19b.html>. Consultada en marzo de 2011.
[57] Página web, <http://www.hw-group.com/products/Hercules/index_en.html>. Consultada en marzo de 2011.
[59] Pagina web, < http://www.topocr.com/>. Consultada en octubre de 2011.