UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

CONSTRUCCIÓN Y PROGRAMACIÓN DE UN ROBOT MÓVIL AUTÓNOMO CON CONEXIÓN A UNA APLICACIÓN ANDROID

TESIS

Que para obtener el título de

Ingeniero en Computación

P R E S E N T A

EDGAR ROBERTO SILVA GUZMÁN

DIRECTOR DE TESIS

Dr. JESÚS SAVAGE CARMONA

Ciudad Universitaria, Cd. Mx., 2017

Agradecimientos

A mis padres Francisco y Azucena por motivarme día con día y hacerme entender que uno puede lograr todo lo que se propone.

A mi familia por apoyarme en los momentos más difíciles.

A Ana por inspirarme a finalizar todas mis metas.

Al Dr. Jesús Savage por sus consejos y por darme la oportunidad de pertenecer a su equipo de trabajo.

A los compañeros del laboratorio de Bio-robótica por compartirme sus conocimientos y experiencias.

Al proyecto PAPIIT IG100915 "Desarrollo de técnicas de la robótica aplicadas a las artes escénicas y visuales"por el apoyo recibido durante esta investigación.

Índice general

CAPÍTULO 1

Introducción

1.1. Justificación

En la actualidad la robótica está viviendo un cambio significativo impulsado por un gran avance tecnológico en la computación, tanto en el software como en el hardware. Esto se hace evidente a diario en diversos artículos y noticias relacionadas con el buen desempeño de un robot, ya sea en el ámbito de seguridad (como en una planta nuclear manipulando elementos altamente nocivos para la salud), en la exploración espacial (navegando autónomamente en la superficie lunar), en las plantas industriales (ensamblando piezas de automóviles) o en el hogar (donde un robot trata de asistir en las actividades diarias a los humanos).

La seguridad en nuestro hogar es un tema que a todos nos preocupa, no solamente por el miedo a sufrir un robo, también por una posible fuga de gas, algún corto circuito que genere un incendio, o simplemente por dejar abiertas puertas, ventanas o alguna lámpara encendida. De lo anterior nació la idea de aprovechar los grandes avances en la robótica y combinarlos con un ambiente controlado en el hogar. Esta idea se centra en la construcción de un pequeño robot móvil, capaz de navegar a través del hogar de manera autónoma o siendo controlado por un usuario remotamente. El robot estaría integrado por varios sensores que recopilarían la información de su entorno, una cámara que transmitiría video en tiempo real y por último, una aplicación Android que se encargaría de elegir el modo de operación y mostrar los datos obtenidos.

La facilidad de tener toda esta información en un aplicación Android haría la vida más tranquila en el hogar.

1.2. Objetivo

Diseñar un robot móvil autónomo y una aplicación en el sistema operativo Android que permita establecer una conexión con el robot móvil para elegir su modo de operación. El robot móvil tendrá varios modos de operación, podrá navegar en un hogar, tanto en interiores como en exteriores. Incorporará una pequeña cámara que transmitirá video en tiempo real a

la aplicación Android. Tendrá integrados sensores para recopilar información de su entorno, como un sensor láser, una fotoresistencia, un sensor de temperatura y un sensor de gas. Estos valores se reportarán en la interfaz de la aplicación Android. Utilizará una red local Wi-Fi para el intercambio de información entre el robot móvil y la aplicación de Android.

1.3. Estructura general de la tesis

Este proyecto de tesis abarca muchos campos: el diseño y construcción de circuitos eléctricos, el armado de una estructura resistente para el robot móvil, la programación de módulos sensoriales y actuadores, por último el diseño y la programación de una aplicación en Android.

La tesis está desglosada de la siguiente forma:

- En el primer capítulo se da breve introducción de qué es un robot y cómo ha influido a lo largo de la vida humana. También se abordan aspectos importantes del sistema operativo Android.
- En el segundo capítulo se describe la historia de la robótica y la investigación que se llevó a cabo para elegir el tipo de robot móvil que será implementado.
- En el tercer capítulo se describe detalladamente la tarjeta de desarrollo elegida y todos los componentes (tanto sensores como actuadores) que fueron integrados en el robot móvil.
- En el cuarto capítulo se describe el desarrollo del prototipo. Se explican los algoritmos utilizados para la adquisición de datos, el desarrollo de los circuitos eléctricos, el intercambio de información, los cuatro modos de operación del robot móvil.
- En el quinto capítulo se describe el desarrollo de la aplicación Android para mostrar los datos provenientes del robot móvil y controlar sus actividades.
- En el sexto capítulo se muestran las pruebas realizadas sobre el funcionamiento del robot móvil y de la aplicación Android, los errores que surgieron y cómo se corrigieron. Además se describen aspectos que no se contemplaron al momento de diseñar el robot móvil.
- Para finalizar se exponen las conclusiones y el trabajo futuro.

CAPÍTULO 2

Antecedentes y marco teórico

Los robots han logrado captar la atención del mundo por el hecho de que realizaron hazañas imposibles para los seres humanos, por ejemplo, entrar en los sitios donde han ocurrido accidentes nucleares (donde el entorno es altamente nocivo para la salud), navegar en las zonas de difícil acceso para el ser humano (en cuevas, minas y sobre todo en los viajes espaciales), en las industrias (donde se manipulan objetos muy pesados o que requieren gran precisión para ser colocados) o en el hogar (donde un robot ayuda con las tareas del día a día)[1].

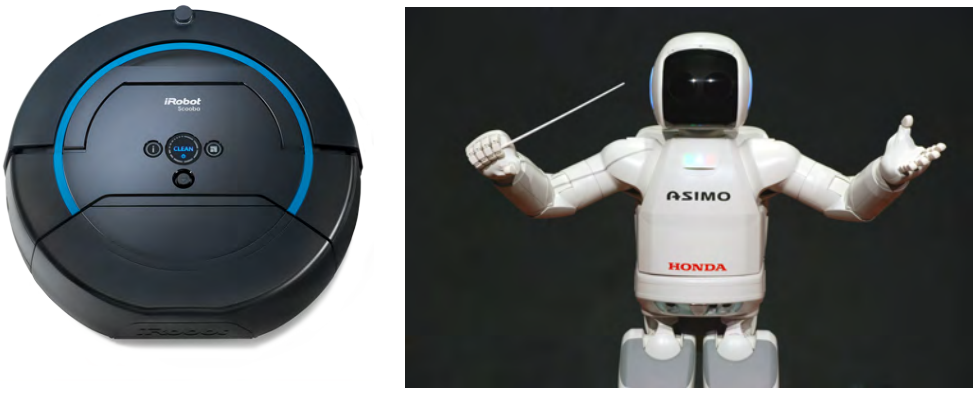
Los robots de servicio se dedican a hacer la vida más cómoda para el ser humano en el hogar, es decir, se diseñan robots específicamente para ayudar en las tareas domésticas. Estos robots pueden hacer tareas sencillas como barrer y trapear *(Robot Scooba*[1]*, figura 2.1 (a))*, o llegar a interactuar en lenguaje natural con un humano para hacer una determinada acción, como caminar de una habitación a la cocina y regresar con el desayuno *(Robot Asimo*[2]*, figura 2.1 (b))*. Este último, un robot humanoide, puede estar conectado a internet con la ventaja de descargar información pedida por el usuario (como el clima o las noticias) y de la misma manera, tener todos los componentes de una casa sincronizados (como la cafetera, la estufa, las lámparas, las puertas y ventanas, etc.), con el motivo de saber de manera remota en qué condiciones se encuentra el hogar. Por ejemplo, si el robot detecta que una puerta está abierta cuando no hay nadie en casa, lo reportaría de inmediato al usuario. De esta manera, el robot podría interactuar con los elementos de la casa, cerrando puertas, apagando lámparas, prendiendo la cafetera, detectando un intruso, etc. En este momento se hace evidente las ventajas de tener un robot dentro de una casa inteligente.

Cada año se organiza una competencia de robótica a nivel mundial, la *RoboCup*[3], donde los robots compiten en varias categorías: futbol soccer, robots de rescate, ligas de simulación

[1]Figura recuperada de: http://www.irobotweb.com/media/Images/iRobot/Robots/HRD/Scooba/Product20Page/scooba-overview.png

[2]Figura recuperada de: https://upload.wikimedia.org/wikipedia/commons2/28/ASIMO-Conducting-Pose-on-4.14.2008.jpg

[3]Véase http://www.robocup.org/



(a) Robot Scooba (b) Robot Asimo

Figura 2.1: Robots de servicio

(robots en ambientes virtuales), robots de servicio, entre otras. Los robots de servicio navegan por un departamento reconociendo objetos,rostros y obstáculos, interactúan con un operador desconocido en lenguaje natural, manipulan objetos y llevan el desayuno o el periódico a un lugar en específico. Estos robots tratan de actuar lo más parecido a un ser humano. El objetivo de esta competencia es que en el año 2050 un equipo de robots pueda vencer al mejor equipo de futbol soccer.

En la última década se ha dado el boom de los dispositivos móviles, cada año salen equipos con mejores características y más funcionalidades, además estos dispositivos ya son accesibles a la mayoría de las personas. Por esta razón, gran cantidad de desarrolladores se ha dedicado a programar aplicaciones móviles para entrar en este creciente mercado.

Android es un sistema operativo que ha liderado en el mercado de los teléfonos inteligentes, en gran medida porque su código es libre, lo cual hace que gran cantidad de desarrolladores opten por este sistema para realizar sus aplicaciones. En resultado a esto hay un sin fin de aplicaciones en su tienda virtual *(Google Play)*[4], tanto de paga como gratuitas que ya son indispensables en el teléfono inteligente de todas las personas[2].

Por estas razones, se optó por desarrollar una aplicación en Android que se encargará de elegir el modo de operación del robot móvil (autónomo o controlado por un usuario) y mostrar los datos importantes que transmite (como los datos de los sensores y el video en tiempo real).

2.1. ¿Qué es un robot?

La palabra *robot* proviene del vocablo checo *robota*, que significa *trabajo forzado* o *esclavitud*[3]. El término fue utilizado por primera vez por Karel Čapek en su historia R.U.R. (*Rossum's Universal Robots*), interpretada por primera vez en 1921[3]. Aunque los robots de Čapek eran humanos artificiales orgánicos, la palabra robot es casi siempre utilizada para referirse a humanos mecánicos.

[4]Véase https://play.google.com/store?hl=es

Actualmente se le llama robot a un dispositivo mecánico o electrónico que está diseñado y programado para automatizar un proceso, con o sin supervisión humana, en el cual éste es controlado por un procesador y no necesariamente tiene una forma humanoide[1]. Un robot está integrado por componentes mecánicos, electrónicos, eléctricos, de comunicación y un sistema de control en tiempo real, percepción del entorno y toma de decisiones.

La eficiencia que han tenido los robots realizando dichas tareas está cambiando el mundo, ahora todos los procesos que lleva a cabo el ser humano tratan de ser realizados en un dispositivo mecánico con o sin autonomía. Esto empezó con pequeños prototipos y al ver las ventajas que se obtenían de ellos, se han hecho robots cada vez más complejos que van desde la fabricación detallada de objetos hasta la interacción total con los humanos.

2.1.1. Esquema general del robot

Los robots están compuestos básicamente por un sistema mecánico, sensores, actuadores y el sistema de control como un elemento indispensable para la toma de decisiones. El sistema de control está basado en ciclos de realimentación de la información obtenida tanto por los sensores internos como por los sensores externos[1].

Los sensores internos se encargan de medir el estado del robot, como giros o desplazamientos, el movimiento entre articulaciones, velocidades, fuerzas, etc., los sensores externos dotan de sentidos al robot. Estos sistemas sensoriales proporcionan las capacidades necesarias para que un robot pueda adaptar automáticamente su comportamiento dependiendo de las variaciones que se producen en su medio ambiente, haciendo frente a situaciones imprevistas como la evasión de obstáculos[4]. La percepción no solo trata de la captura de información sensorial, sino también su tratamiento e interpretación.

La figura 2.1 muestra robots industriales automatizados para la producción de vehículos[5].

La rama de la robótica aún es joven y hay un gran campo de desarrollo que explorar, ya sea automatizando procesos o creando máquinas que interactúen con los humanos en la vida diaria. De esto nace la inspiración de crear un prototipo de robot móvil de servicio que pueda interactuar con un usuario, navegar a través del hogar tanto en interiores como en exteriores, capaz de otorgar seguridad y comodidad en el hogar, un robot capaz de ser operado remotamente de distintas formas, junto con una aplicación Android que muestre al usuario la información sensorial (como el video en tiempo real o el nivel de gas en el hogar) en el momento que lo desee.

2.1.2. Historia de la Robótica

Aunque el término robot viene de los años 20, la robótica industrial nace hasta los años 50 y en los años 60 se comienzan a impartir cursos en las universidades. En la robótica industrial se trata de otorgar flexibilidad a los procesos productivos manteniendo la productividad, reduciendo los riesgos para los trabajadores, disminuyendo tiempos de operación, producción en serie, etc[1]. Entre los años 80 y 90 se han diseñado un gran numero de máquinas cuyo objetivo ya no es sustituir la productividad humana, sino que se trata de realizar tareas en lugares difícilmente accesibles a los humanos o que involucran algún riesgo de accidentes (por trabajos en donde se utilizan componentes que por su tamaño son muy

[5]Figura recuperada de: https://singularidadpara7000millones.files.wordpress.com/2013/04/robots-industriales.jpg

Figura 2.2: Robots Industriales

difíciles de manipular). Así mismo se desarrollaron actividades relacionadas con la exploración espacial, actividades subacuáticas, manipulación y transporte de materiales peligrosos, minería, agricultura, construcción y hasta pequeñas incisiones quirúrgicas. La mayoría de estas actividades son fundamentalmente teleoperadas pero con el paso del tiempo se trata de dotarlas de una mayor autonomía y finalmente, construir robots[4].

Otro sector de importante crecimiento en la robótica es el de los robots de servicio, en los cuales se incluyen los robots domésticos, robots de ayuda a los discapacitados y robots asistentes generales. En este sector ya no se trata sobre producción o investigación, ahora la principal actividad es la interacción con un humano de manera natural, es decir, interacción mediante el habla, señas o gestos[1].

2.1.3. Clasificación

De acuerdo con su grado de autonomía, los robots pueden clasificarse en teleoperados, de funcionamiento repetitivo y autónomos o inteligentes.

En los robots teleoperados las tareas de percepción del entorno, planificación de acciones y manipulación son realizadas por un operador, éste actúa en tiempo real sustituyendo varios ciclos de control de alto nivel. Estos robots son utilizados para actividades donde el control se realiza de manera remota (en lugares de difícil acceso, materiales contaminados o peligrosos), en tareas difíciles de automatizar y en entornos no estructurados, tales como en una construcción o en el mantenimiento de líneas eléctricas[1].

Los robots de funcionamiento repetitivo son aquellos que se usan en las industrias, trabajan en tareas predecibles e invariantes. Son precisos y relativamente rápidos, incrementan la productividad y ahorran al hombre trabajos repetitivos e incluso peligrosos[1].

Los robots autónomos o inteligentes son los mas avanzados desde el punto de vista del

procesamiento de la información. Estos robots son capaces de percibir, modelar el entorno, planificar acciones y actuar para alcanzar todas las metas propuestas sin la necesidad de la intervención de un operador, o en algunas ocasiones con una intervención mínima[1]. Las dificultades surgen por la elevada capacidad de procesamiento requerida para ejecutar acciones en tiempo real, necesarias para resolver problemas suficientemente significativos, como hacer frente a situaciones imprevistas (evadir obstáculos, tareas complejas, etc.), y sobre todo por la propia incertidumbre de la información del entorno, es decir, por la poca precisión de algunos sensores.

Este proyecto de tesis trata de desarrollar un robot móvil con varios modos de operación, tanto autónomo como teleoperado. En el modo autónomo el robot móvil navegará por el hogar de un punto inicial a un punto final, durante su trayecto irá evadiendo obstáculos, como por ejemplo, cajas, muebles o botes que se encuentren en su trayectoria. En el modo teleoperado, el usuario dará comandos de movimiento a través de una aplicación Android y el robot móvil los ejecutará.

2.2. Robots móviles

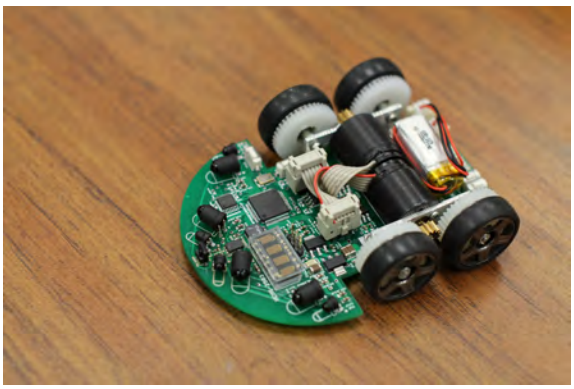
Desde sus orígenes, la robótica industrial estuvo relacionada con las tareas de manipulación, los robots no necesitaban desplazarse de un lugar a otro para realizar sus tareas, por esta razón siempre estaban colocados en un lugar en específico. Los robots móviles se diferencian de los robots manipuladores convencionales en que no están anclados, si no por el contrario pueden desplazarse por el terreno, agua o incluso volar libremente[3].

Desde el punto de vista de la autonomía, los robots móviles tienen como precedente los dispositivos electromecánicos, tales como los denominados *"micromouse*[6]*" (fig 2.2 a)* creados desde los años 30 para desarrollar funciones inteligentes como descifrar laberintos, como el que se muestra en la figura 2.2 b. Cabe destacar la tortuga *Walter* presentada en 1948, que podía reaccionar ante la presencia de obstáculos, subir pendientes y cuando la alimentación comenzaba a ser insuficiente podía dirigirse a una toma de corriente[1].

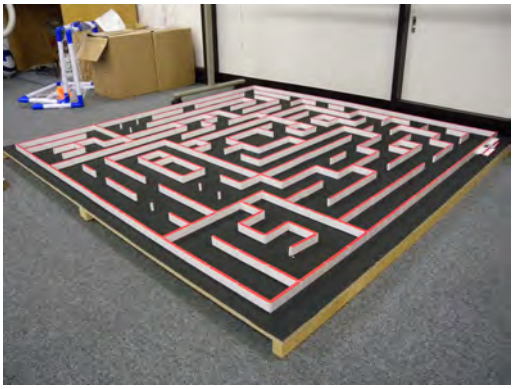
En los años 60 se vuelve a trabajar en el desarrollo de robots móviles dotados de una mayor autonomía. La mayor parte de los prototipos se desarrollan empleando plataformas que soportan sistemas de visión. Sin embargo, estos prototipos aún no podían navegar autónomamente de manera eficiente. En los años 80 viene un incremento en la capacidad de procesamiento computacional y un mejor desarrollo de sensores, mecanismos y sistemas de control que dotan de una mayor autonomía a los nuevos prototipos de robots móviles[1].

La autonomía de un robot móvil se basa en un sistema de navegación automático, en estos sistemas se incluyen las tareas de planificación de acciones, percepción del entorno y control de acciones. En los robots móviles, el problema de navegación abarca la planificación global de la tarea, es decir, elegir una ruta adecuada, recorrer una trayectoria planeada y finalmente evitar obstáculos no esperados[4]. En un robot móvil diseñado para interiores (como una sala o un departamento), la tarea consiste en determinar a qué habitación o lugar en específico hay

[6]Figura recuperada de: https://upload.wikimedia.org/wikipedia/commons/ 2/25/Micromouse



(a) Robot Micromouse



(b) Laberinto del robot Micromouse

Figura 2.3: El robot micromouse era capaz de descifrar laberintos

que desplazarse, mientras que la ruta establecería el camino desde la posición inicial hasta una posición final, definiendo puntos intermedios de paso, y durante la trayectoria, evadir obstáculos no esperados (como una caja o una persona). Durante la ejecución de la tarea surgen algunos inconvenientes, el robot móvil puede desviarse de la trayectoria establecida debido a la acumulación de imprecisiones mecánicas y de control, esto hace que sea más difícil llegar a la posición final. La corrección de la acumulación de errores hace necesario el empleo de otros sensores (como los encoders en los motores), que nos informan qué tanto se ha desplazado el robot de acuerdo a la distancia que se le indicó recorrer, y si hay fallas, se compensan en ese momento.

2.2.1. Tipos de configuración para robots móviles con ruedas

Los robots móviles con ruedas son la solución mas simple para conseguir un desplazamiento eficiente en terrenos planos y libres de obstáculos, permitiendo conseguir velocidades constantes y relativamente altas. Los robots móviles emplean diferentes tipos de configuraciones para las ruedas, ya que les confieren características y propiedades diferentes respecto a la eficiencia energética, dimensiones, cargas útiles y sobre todo para una mejor maniobrabilidad.

Configuración Ackerman

Esta configuración es la utilizada en vehículos de cuatro ruedas convencionales (figura 2.3)[7]. La rueda delantera interior gira un ángulo ligeramente superior a la exterior $(\theta 1 > \theta 0)$ para eliminar el deslizamiento. Las prolongaciones de los ejes de las dos ruedas delanteras (directrices) intersectan en un punto *P1* llamado CIR (Centro instantáneo rotación) sobre la prolongación de los ejes de las ruedas traseras (motrices). Uno de los problemas de la configuración Ackerman es que tiene una maniobrabilidad limitada[1].

[7]Figura recuperada del libro "Robótica: Manipuladores y robots móviles" cuyo autor es Aníbal Ollero Baturone

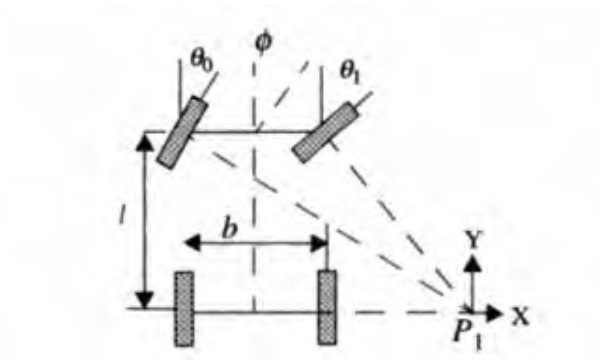


Figura 2.4: Configuración Ackerman

Triciclo clásico

En esta configuración hay una sola rueda delantera, que sirve tanto para la tracción como para el direccionamiento (figura 2.4)[8]. El eje trasero tiene 2 ruedas laterales que se mueven libremente. La maniobrabilidad es mayor que en la configuración Ackerman pero puede presentar problemas de estabilidad en terrenos difíciles. Otro inconveniente es que el centro de gravedad tiende a desplazarse cuando el robot se mueve por una pendiente[1].

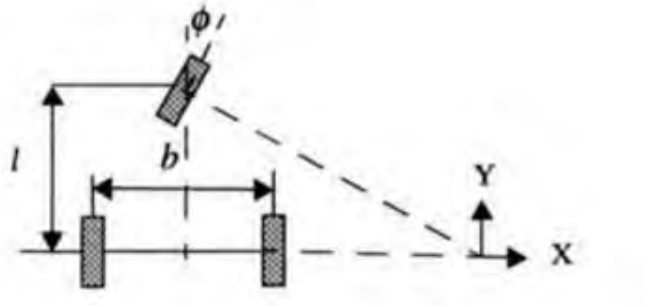


Figura 2.5: Configuración Triciclo clásico

Direccionamiento diferencial

En esta configuración, la velocidad del robot móvil viene dada por las velocidades de las ruedas laterales, de igual manera, el direccionamiento viene dado por la diferencia de velocidades de ambas ruedas (figura 2.5)[9]. Para una mayor estabilidad pueden existir una o mas ruedas de soporte[1].

[8]Figura recuperada del libro "Robótica: Manipuladores y robots móviles" cuyo autor es Aníbal Ollero Baturone

[9]Figura recuperada del libro "Robótica: Manipuladores y robots móviles" cuyo autor es Aníbal Ollero Baturone

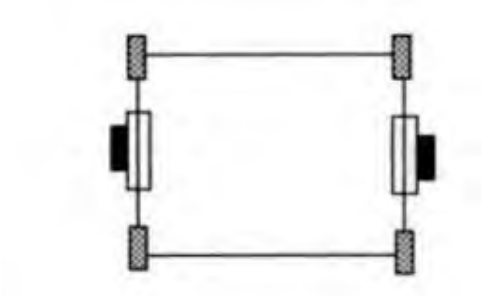


Figura 2.6: Configuración Diferencial

Skid Steer

En esta configuración se disponen de varias ruedas en cada lado del robot que actúan de forma simultanea (figura 2.6)[10]. La velocidad del robot móvil es el resultado de combinar las velocidades de las ruedas de izquierda con las de la derecha. Los giros se dan por la diferencia de velocidades de las ruedas de cada lado. Esta configuración es similar a la configuración diferencial, pero al agregarle mas ruedas se produce un deslizamiento transversal[1].

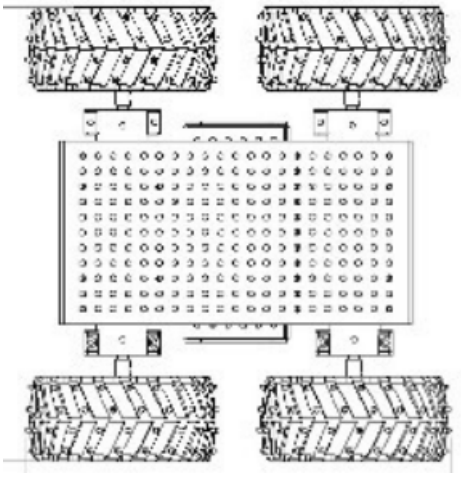


Figura 2.7: Configuración Skid Steer

Pistas de deslizamiento

Estos robots son catalogados como tipo oruga, en donde tanto las velocidades como los giros se consiguen mediante pistas de deslizamiento (figura 2.7)[11]. Al igual que en la configuración Skid Steer, el desplazamiento se consigue mediante la diferencia de velocidades de cada pista. Esta configuración es útil en terrenos irregulares en los cuales presenta un mejor rendimiento que otras configuraciones[1].

[10]Figura recuperada del libro "Robótica: Manipuladores y robots móviles" cuyo autor es Aníbal Ollero Baturone

[11]Figura recuperada de: http://g03.a.alicdn.com/DAGU-RS035-All-metal-T-rex-Robot-Tank-Gearbox-Synchronizing-Wheel-Robot-Chassis-DIY-RC-Car.jpg

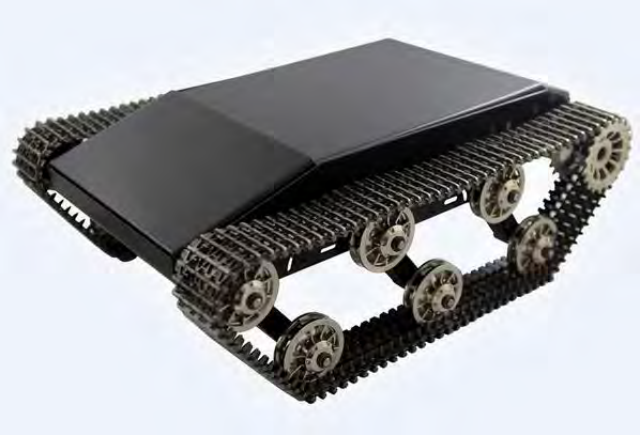


Figura 2.8: Pistas de deslizamiento

Configuración Omnidireccional

Esta configuración le permite al robot desplazarse en cualquiera de las componentes de un plano, es decir, hacia adelante, en reversa o de forma lateral, sin necesidad de hacer algún giro (figura 2.8)[12]. Por esta razón este tipo de configuración tiene un mayor grado de complejidad para su control. Esta configuración nos otorga mayor maniobrabilidad que otras configuraciones[1].

Figura 2.9: Configuración Omnidireccional

Locomoción mediante patas

Esta configuración permite aislar al cuerpo del terreno empleando únicamente los puntos de soporte (patas), es posible adaptar estos puntos para dar una mayor estabilidad en terrenos difíciles y, además, pasar sobre los obstáculos (figura 2.9)[13] Así mismo, mediante patas el

[12]Figura recuperada de: http://www.electan.com/images/Seeedstudio/2276.jpg

[13]Figura recuperada de: http://www.smh.com.au/content/dam/images/2/z/g/5/s/image.related. article-Leadwide.gnmiy7.png

desplazamiento es mejor y es posible conseguir un movimiento omnidireccional. Por esta razón se considera que tiene mejores propiedades que las ruedas, por otro lado, la complejidad de los mecanismos necesarios es mayor, así como su control y su consumo de energía[1].

Figura 2.10: WildCat, robot capaz de galopar a grandes velocidades (Boston Dynamics, EU)

Configuraciones Articuladas

Los robots con este tipo de configuración consisten en 2 o más módulos articulados, cada uno con una configuración con ruedas (figura 2.10)[14]. Esta configuración otorga mayores ventajas en los terrenos difíciles a los que se debe adaptar el cuerpo del robot. Los robots con muchos eslabones son apropiados para caminos estrechos[1].

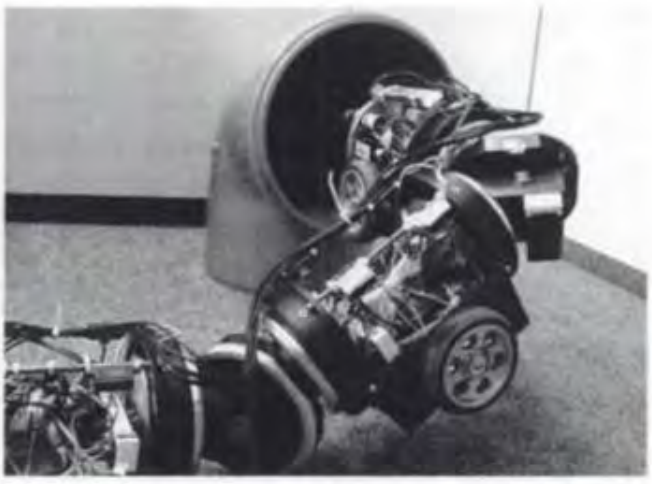


Figura 2.11: MAKRO, robot móvil articulado de GMD (Alemania)

[14]Figura recuperada del libro "Robótica: Manipuladores y robots móviles" cuyo autor es Aníbal Ollero Baturone

2.3. Android

Android es un sistema operativo para teléfonos móviles que está basado en una versión modificada de Linux. Fue creado por un negocio del mismo nombre, Android inc. En el año de 2005 Google lo adquiere como parte de una estrategia para entrar en el mercado de la telefonía móvil, posteriormente se hizo cargo del desarrollo del sistema[5].

La idea principal de Google era que Android fuera libre y de código abierto, por lo que la mayor parte de código Android se puso disponible bajo la licencia Apache de código abierto, lo que hacía que cualquiera que quisiera utilizar Android pudiera descargar su código fuente completo e implementarlo para sus propios proyectos. Con el paso del tiempo, este modelo de desarrollo hizo que Android fuera muy atractivo, y empezó a llamar la atención de muchos fabricantes.

En concreto Android es una plataforma de código abierto, es decir, ni los desarrolladores ni los fabricantes de dispositivos móviles pagan derechos de autor o gastos de licencia para desarrollar en la misma. Otro punto importante a favor de esta plataforma es que se cuenta con varios tipos de emuladores dentro de su IDE (sigla en inglés de Integrated Development Environment) de desarrollo (Android Studio), por lo cual se puede elegir un emulador con las características del teléfono móvil (tanto de hardware como de software) para las cuales va a estar dedicada la aplicación[2].

2.3.1. Características del S.O. Android

Una de las principales ventajas de desarrollar aplicaciones en Android es que ofrece un enfoque unificado, es decir, los desarrolladores sólo necesitan programar para Android y sus aplicaciones podrán ejecutarse en diversos dispositivos con este sistema operativo, independientemente de la marca o modelo del mismo.

Para llevar a cabo un proyecto en esta plataforma, Android cuenta con un kit de desarrollo de software gratuito llamado Android Studio, que incluye Android SDK (sigla en inglés de software development kit) y sus herramientas, este software se encuentra disponible desde el sitio web de Android [15].

Arquitectura de Android

El sistema operativo Android está diseñado con una arquitectura de cuatro capas o niveles que están relacionadas entre sí[2].

- Kernel de Linux: Android está diseñado a partir del kernel de Linux, éste ha sido modificado para ser adaptado a dispositivos móviles. En esta capa se encuentran todos los controladores del dispositivo de bajo nivel para la los diversos componentes de hardware, como el controlador de la cámara, el controlador Wi-Fi, el controlador de audio, entre otros.
- Capa de librerías: En esta capa se encuentran librerías escritas en lenguaje c/c++ compiladas para la arquitectura específica del teléfono móvil. Su función es propor-

[15]Véase https://www.android.com/

cionar las herramientas necesarias para la ejecución adecuada de las aplicaciones. Por ejemplo:

  - Hardware Abstraction Layer (**HAL**). Es un componente que permite la independencia del hardware, es decir, Android está diseñado para ejecutarse en cualquier dispositivo independientemente de su arquitectura física.
  - Las librerías nativas. Son interfaces de código abierto que le proporcionan las características más importantes al sistema Android: **OpenGL** proporciona herramientas para los gráficos en 3D, **SQLite** proporciona soporte de base de datos, **WebKit** proporciona funcionalidades para la navegación Web, entre muchas otras.
  - Demonios (Daemons). Son códigos que se ejecutan para ayudar a un servicio del sistema, por ejemplo, cuando se va a instalar una aplicación se ejecuta el demonio **installd**, el cual se encarga de administrar todo el proceso.
  - Consola. Al igual que otros sistemas operativos, Android permite que se empleen comandos de línea para ejecutar procesos del sistema.

- Tiempo de ejecución. Se encuentra en el mismo nivel que las librerías. Las aplicaciones Android se encuentran escritas en lenguaje Java y son traducidas a bytecodes, pero no son interpretadas por la máquina virtual de Java debido a que Android tiene su propia máquina virtual llamada **ART (Android Runtime)**. ART lleva a cabo la transformación de una aplicación móvil en instrucciones máquina, que luego son ejecutadas por el entorno de ejecución nativo del dispositivo. Esto otorga flexibilidad ante el diseño de hardware del teléfono móvil.

- Arquitectura de software de aplicación. En esta capa se encuentran todas las librerías Java necesarias para desarrollar las aplicaciones. Los paquetes mas importantes son los *android.**, donde se encuentran las características más importantes de la programación en Android.

- Aplicaciones: Esta capa superior se centra en la ejecución, comunicación y estabilidad de las aplicaciones que se distribuyen con el dispositivo Android (tales como teléfono, contactos, navegador, cámara, etc.), al igual que aplicaciones que se descargan y se instalan desde Google Play.

La figura 2.11 muestra la arquitectura del sistema operativo Android en sus cuatro capas[16].

[16]Figura recuperada de: http://www.hermosaprogramacion.com/wp-content/ uploads/2014/08/android-arquitectura.jpg

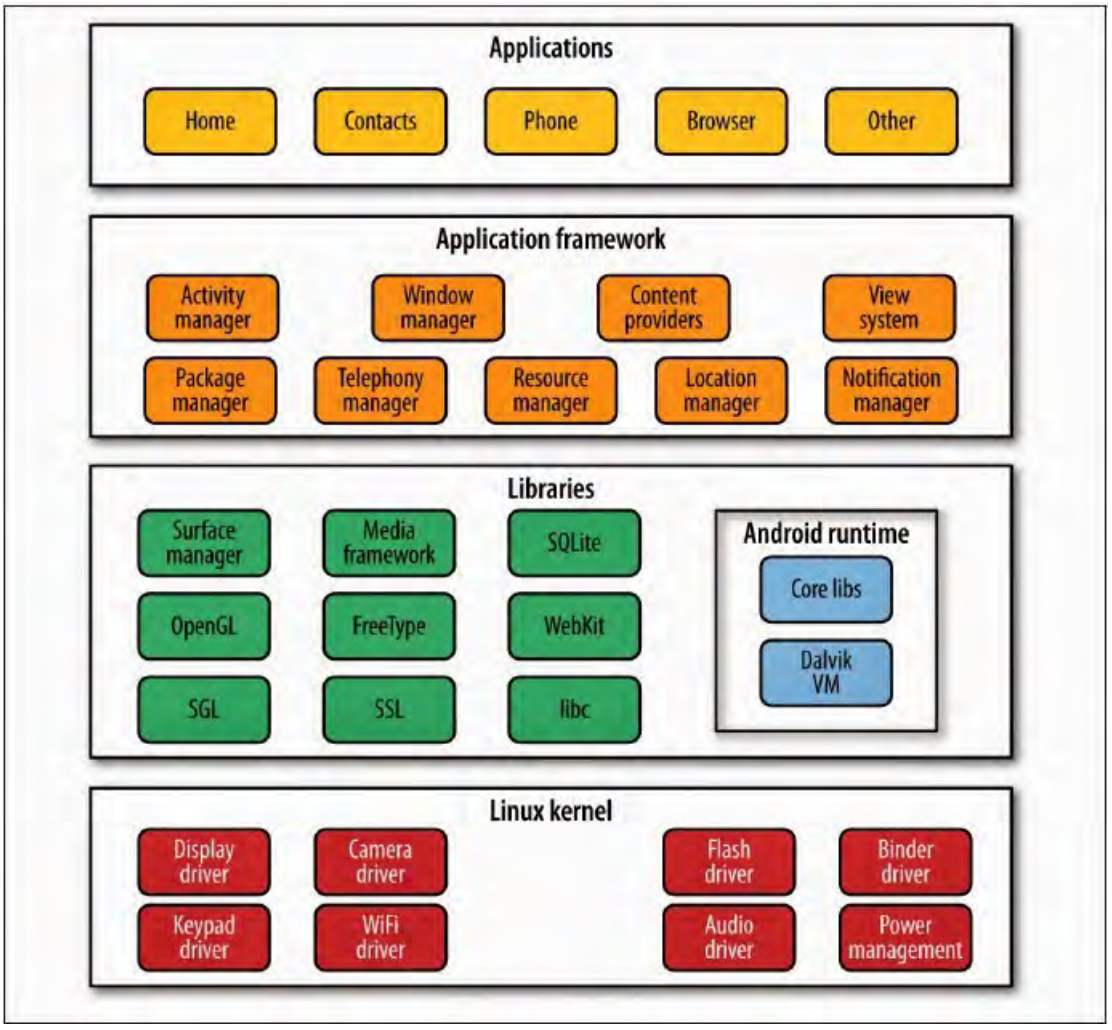


Figura 2.12: Arquitectura del sistema operativo Android

Google Play

Uno de los principales factores que determina el éxito de un sistema operativo móvil es la gran cantidad de aplicaciones que se pueden instalar y ejecutar. Por ello, en 2008, Google anunció Android Market (ahora Google Play), como una tienda en línea de aplicaciones para dispositivos Android, que almacena tanto aplicaciones de paga como gratuitas. En esta tienda se pueden encontrar aplicaciones de entretenimiento, videojuegos, reproductores de audio y video, fotografía, comunicación, noticias, sociales y además se pueden comprar películas, libros y discos de música. Cualquier desarrollador de Android puede subir sus aplicaciones a Google Play después de adquirir una licencia de desarrollador, lo que hace que cualquier usuario pueda descargar la aplicación e instalarla en su dispositivo móvil[2].

Gracias a las facilidades que ofrece la plataforma de desarrollo de Android y la compatibilidad con varios dispositivos independientemente de la marca o modelo, se decidió programar la aplicación móvil en esta plataforma utilizando el lenguaje de programación Java.

CAPÍTULO 3

Componentes del robot móvil

Se eligieron varios dispositivos necesarios para llevar a cabo el proyecto de tesis:

- Raspberry Pi 2. Es el módulo central donde se procesa la información obtenida por los sensores, se encarga de controlar los motores del robot móvil y establece una conexión con la aplicación Android.
- Raspberry Pi Cam. Es una pequeña cámara que captura el video en tiempo real para ser transmitido a la aplicación Android.
- Sensores. Se eligieron varios sensores para adquirir la información del ambiente. Un sensor de temperatura para saber la temperatura actual del hogar. Una fotoresistencia que se encargará de automatizar el encendido y apagado de luces (en un trabajo a futuro). Un sensor de gas que notifica la presencia anormal de gas en el ambiente. Y un láser de alta precisión, que tiene como propósito detectar obstáculos en la trayectoria del robot móvil, y como un trabajo futuro, se encargará de crear un mapa del entorno. Como la Raspberry Pi 2 carece de un convertidor analógico-digital se decidió integrar un pequeño circuito ADC de 8 canales con conexión SPI.
- Componentes de alimentación eléctrica. La mayor parte de componentes necesitan una alimentación eléctrica de 5 V, por lo cual se optó por un pequeño regulador de voltaje. Para los motores se eligió un controlador de motores adecuado para su funcionamiento y, como fuente de alimentación, una batería LiPo. Como se mencionará más adelante en este capítulo, los puertos USB 2.0 no pueden suplir de corriente a todos los componentes del robot móvil, por lo cual se decidió incluir un hub USB con alimentación externa.
- Kit de robótica Wild Thumper. Este kit tiene los elementos necesarios para armar un robot móvil adecuado para el proyecto de tesis.

3.1. Raspberry Pi 2

La Raspberry Pi 2[6] es una placa-computadora económica desarrollada en Reino Unido por la Fundación Raspberry Pi, con el objetivo de impulsar la enseñanza de ciencias de la computación en las escuelas. Se trata de una placa de 85 x 54 milímetros aproximadamente (el tamaño de una tarjeta de crédito) integrada con un chip Broadcom BCM2835 con procesador ARM de 1 GHz de velocidad, GPU VideoCore IV y 1GB de memoria RAM (figura 3.1)[17].

La Raspberry Pi 2 cuenta con un sistema de entrada/salida llamado GPIO (siglas del inglés General Porpuse Input/Output), éste está compuesto por una serie de pines que pueden utilizarse como entradas (para la adquisición de datos) o como salidas (para el control de actuadores). Estos pines proporcionan una gran ventaja en cuanto a la comunicación con otros dispositivos, para la adquisición de datos provenientes de los sensores, el control de la velocidad y dirección de los motores, entre otras cosas.

Su funcionamiento depende de un sistema operativo, el cual va a ser instalado en un medio de almacenamiento (en este caso una microsd). Como fuente de alimentación puede utilizarse un cargador micro-USB de 2000mah o una batería externa de 5 V a 2 A. El sistema operativo dispone de 3 modos de uso: modo interfaz gráfica, modo línea de comandos y modo scratch (este último es una herramienta de programación visual para crear animaciones y video-juegos). Para utilizar la interfaz gráfica, la Raspberry Pi 2 tiene un puerto HDMI para la transmisión de video y 4 puertos USB 2.0 para controlar el sistema, donde se pueden conectar el ratón y el teclado. Para la conexión de red, se dispone de un puerto ethernet, en el caso de que se desee tener una conexión WiFi se puede utilizar un adaptador inalámbrico Wi-Fi compatible. Podría decirse que la Raspberry Pi 2 equivale a una computadora con capacidades gráficas similares a la XBOX de Microsoft, con la posibilidad de reproducir vídeo en alta definición (1080p). Se optó por conectarle un dispositivo USB WiFi, éste proveerá de una conexión a Internet sin necesidad de utilizar el puerto ethernet.

Debido a su pequeño tamaño y a las grandes ventajas que presenta, la Raspberry pi 2 se utilizará como modulo central para la adquisición y el procesamiento de la información.

3.1.1. Software

La Raspberry Pi 2 no cuenta con un sistema operativo de fábrica, para obtenerlo se debe descargar de la página oficial una distribución de acuerdo a las necesidades de uso; éste se debe instalar en tarjeta micro-SD. Hay varias distribuciones oficiales como Raspbian, Arch Linux, RaspBMC, Windows, Android, Pidora u OpenELEC. Raspbian OS es la distribución recomendada para desarrollar proyectos de este tipo, está basada en la distribución Debian Wheezy (Debian 7.0) optimizando el código de ésta para la Raspberry Pi 2. La distribución es ligera y se puede navegar fácilmente en el hardware de la tarjeta, tiene un ligero entorno de escritorio LXDE y Midori como navegador web predeterminado. Además incluye muchas herramientas para el desarrollo de proyectos de programación[18].

[17]Figura recuperada de: http://domisan.sakura.ne.jp/article/rp-toppers/ raspberrypi000.jpg
[18]Véase https://www.raspberrypi.org/

Figura 3.1: Raspberry Pi 2 Modelo B

3.1.2. Procesador ARM

La Raspberry Pi 2 tiene un procesador ARM a 1 GHz. Estos procesadores fueron creados principalmente para dispositivos portátiles, ya que son pequeños, tienen un bajo consumo de energía y no generan mucho calor, por lo que no se necesita de ventiladores o ningún otro sistema de refrigeración forzada.

La relativa simplicidad de estos procesadores los hace ideales para aplicaciones de baja potencia, como resultado, han dominado en el mercado de la electrónica móvil, integrados en microprocesadores y microcontroladores pequeños, de bajo consumo y relativamente bajo costo.

3.1.3. ¿Por qué usar la Raspberry Pi?

Debido a sus características, tamaño y precio, la Raspberry Pi 2 es una herramienta indispensable para llevar a acabo proyectos de electrónica, ya sean de aprendizaje o de automatización. Con ella se pueden crear diversos proyectos, como un centro multimedia para reproducir películas, videos y navegar en internet. También se puede utilizar para automatizar determinadas actividades en el hogar, como el control remoto del aire acondicionado o la cafetera, el control de la iluminación, apertura o cierre de puertas y el monitoreo de video a través de una pequeña cámara.

A continuación se describen las ventajas y desventajas de utilizar la Raspberry Pi 2 como un módulo central para un proyecto de robótica.

Ventajas Raspberry Pi 2:

- Consumo eléctrico 5 volts. Tiene gran compatibilidad con otros dispositivos que funcionan al mismo voltaje, además se puede utilizar una fuente de alimentación (como cargador móvil) o una batería externa de una manera sencilla.

- No necesita gran capacidad de almacenamiento, con una tarjeta micro-SD de 2GB se puede instalar el sistema operativo y las herramientas para aprovechar todas sus características.
- Su sistema GPIO de entrada/salida. Estos pines nos otorgan la ventaja de poder conectar varios sensores y actuadores a la Raspberry Pi. Pudiendo crear grandes circuitos que se comuniquen entre sí.
- Pequeño tamaño y buena relación características/precio.

Desventajas Raspberry Pi 2:

- No cuenta con un convertidor analógico-digital en los pines de entrada.
- Poca memoria RAM que se comparte con la GPU y con la tarjeta gráfica.
- Los puertos USB tienen una limitación de corriente, por lo cual no se podrán conectar dispositivos que demanden mucha energía y habrá que hacerlo a través de un hub USB con alimentación.

Gracias al pequeño tamaño y las ventajas que proporcionan cada una de sus características tanto de software como de hardware, hace que muchos proyectos de robótica puedan realizarse a un bajo costo. La posibilidad de programar en diversos lenguajes de programación, la característica de tener un módulo dedicado a una cámara de gran resolución, las herramientas para el tratado de imágenes como OpenCv, entre otras características, hacen que la Raspberry Pi 2 sea la tarjeta de desarrollo elegida para realizar este proyecto de tesis.

3.1.4. Raspberry Pi Cam

Existen gran cantidad de accesorios para la Raspberry Pi 2, entre ellos destaca la Raspberry Pi Cam, lanzada por la misma empresa. La cámara cuenta con un sensor de 5 megapíxeles que además graba de vídeo en alta definición (1080p) a 30 fotogramas por segundo y es capaz de hacer fotos con una resolución de 2 592 X 1 944 píxeles (figura 3.2)[19]. Para poder utilizarla se conecta mediante un cable plano al conector CSI (sigla del inglés Camera Serial Interface) de la Raspberry Pi 2 [7].

[19]https://www.raspberrypi.org/wp-content/uploads/ 2014/03/raspberry-pi-camera-module.jpg

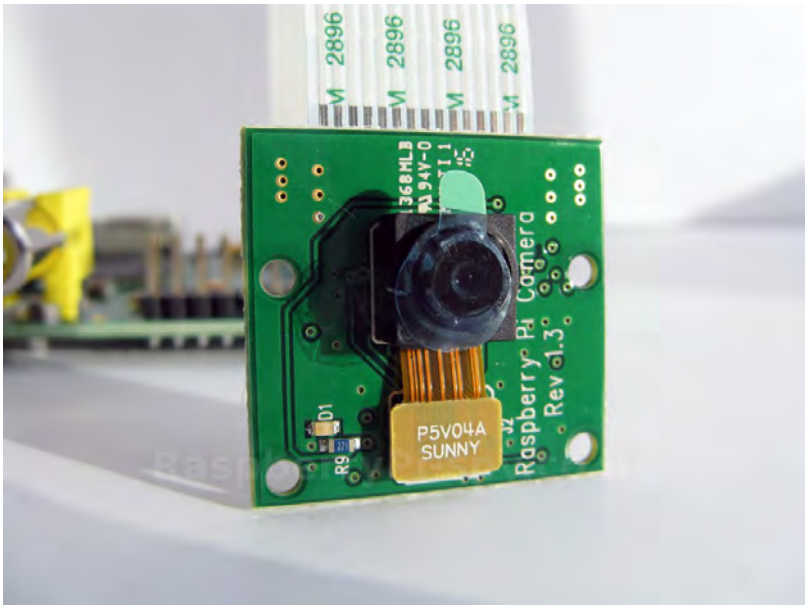


Figura 3.2: Raspberry Pi 2 Modelo B

3.2. Sensores

En la actualidad, las personas pasan la mayor parte del día fuera de casa, ya sea trabajando o paseando por la ciudad, así que todo lo ocurrido en casa pasa desapercibido por el dueño. El tema de la seguridad es muy importante a tratar, ya que no solo se centra en los robos o asaltos, si no también en un aspecto importante que es monitorear cualquier elemento que ponga en riesgo nuestra salud (como una fuga de gas). Por estas razones, un sistema de seguridad que mantenga al tanto al usuario sobre lo que ocurre en su casa, ya sea ante un robo o una fuga de gas, otorgaría una mayor tranquilidad para él y su familia.

Por estas razones se decidió crear un sistema integrado de varios sensores que sean capaces de obtener la información del ambiente y notificar al usuario los datos mas relevantes de manera remota. En este prototipo de robot móvil se integraron los siguientes sensores: una fotoresistencia que nos va a decir la intensidad luminosa dentro del hogar, un sensor de gas que detecta si hay humo o alguna fuga de gas y por último un sensor de temperatura que mide la temperatura a la cual se encuentra el hogar.

3.2.1. Sensor de temperatura

Un sensor de temperatura es un dispositivo electrónico que convierte los cambios de temperatura en señales eléctricas, para después ser tratadas en un circuito eléctrico o procesadas en un microcontrolador (figura 3.3)[20]. Se decidió utilizar el componente TMP36 [8], que es un sensor de temperatura de precisión en grados centígrados y bajo voltaje. La salida de voltaje es linealmente proporcional a la temperatura en grados centígrados del ambiente. No

[20]Figura recuperada de: https://cdn.shopify.com/s/files/1/1040/8806/products/temperatura03.jpeg?v=1454364999

requiere ninguna calibración externa y tiene una precisión de ±1°a 25°y ±2°por encima del rango de -40°a 125°.

Este sensor tiene un funcionamiento sencillo, cuenta con 3 pines de los cuales dos son de alimentación (GND y V) y el tercer pin da una salida de voltaje proporcional a la temperatura ambiente. La salida de voltaje del sensor será procesa por la Raspberry Pi 2.

Figura 3.3: Sensor de Temperatura TMP36

3.2.2. Fotoresistencia

La resistencia dependiente de la luz (LDR sigla del inglés Light Dependent Resistor), también conocida como fotoresistencia, es un dispositivo electrónico sensible a los cambios de intensidad de la luz, es decir, es un dispositivo que varía su resistencia en función de la intensidad de luz que incide sobre su superficie (figura 3.4)[21]. Entre más sea la intensidad de luz, menor será su resistencia y en el caso contrario, cuando la intensidad de luz sea menor su resistencia será mayor.

Las fotoresistencias tienen muchas aplicaciones en la vida cotidiana, se utilizan en el encendido y apagado de lámparas, en cámaras fotográficas, en los medidores de luz y en algunas alarmas. Una fotoresistencia nos otorga ventajas al querer crear un sistema automatizado en el hogar, ésta nos indica la intensidad de luz en el ambiente y, dependiendo de eso, se pueden encender o apagar luces o cerrar y abrir cortinas. La salida de voltaje será procesada por la Raspberry Pi 2.

[21]Figura recuperada de: http://cdn.shopify.com/s/files/1/0409/9041/products/ foto2-grande.jpg

Figura 3.4: Fotoresistencia

3.2.3. Sensor de Gas

El sensor de gas analógico **MQ2** [9] es un dispositivo electrónico que detecta la presencia de gas o humo en el ambiente, es decir, varía su resistencia cuando se expone a determinados gases (figura 3.5)[22]. Se utiliza principalmente para detectar fugas de gas en las industrias. Este sensor detecta varios tipos de gas, como el gas LP, i-butano, propano, metano e hidrógeno, además tiene una alta sensibilidad (que puede ser ajustada por medio de un potenciómetro) y un tiempo de respuesta rápido. Este pequeño dispositivo es de bajo consumo y detecta la presencia de gas combustible y humo en concentraciones de 100 a 10.000 ppm[9].

Para su funcionamiento, el circuito que integra este dispositivo incorpora una sencilla interfaz de 4 pines, dos de ellos son de alimentación (GND, 5 V), un pin de salida analógica y un pin de salida digital. La salida analógica será procesada por la Raspberry Pi 2 dependiendo del gas a detectar, es decir, para cada tipo de gas hay una curva de sensibilidad diferente [10].

Figura 3.5: Sensor de Gas MQ2

[22]Figura recuperada de: https://rydepier.files.wordpress.com/2015/ 07/img-2006.png

3.2.4. Convertidor analógico-digital

El dispositivo MCP3008 [14] es un circuito integrado encargado de convertir señales analógicas a una señal digital. Tiene 8 canales de entrada para diferentes dispositivos analógicos, cada uno con una resolución de 10 bits (figura 3.10)[23], después transmite estos datos a un microcontrolador mediante una conexión SPI (sigla del inglés Serial Peripheral Interface).

SPI es un protocolo de comunicación serial utilizado para la transferencia de información entre microcontroladores y pequeños periféricos, trabaja con una interfaz maestro-esclavo. El bus SPI utiliza 4 canales de comunicación, Master Out Slave In (MOSI), Master In Slave Out (MISO), el reloj (SCK) y Slave Select (SS). La Raspberry Pi 2 es compatible con este protocolo y para poder comunicarse con el circuito MCP3008 es necesario activar la comunicación SPI a través del sistema operativo. Después se podrán conectar los 4 canales de comunicación a los pines GPIO dedicados a este protocolo.

La Raspberry Pi 2 no tiene entradas analógicas, por esta razón se optó por utilizar el dispositivo MCP3008. Este circuito integrado tendrá la tarea de obtener los valores que dan los sensores: el sensor de temperatura, el sensor de gas, la fotoresistencia y el divisor de voltaje (éste para el monitoreo de la batería), y así poder transmitir estos datos a la Raspberry Pi 2 para ser procesados.

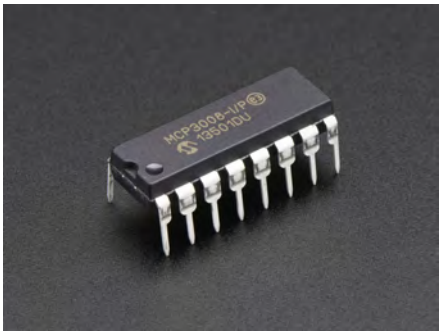


Figura 3.6: Convertidor Analógico-Digital MCP3008

3.2.5. Láser Hokuyo

El láser Hokuyo(URG-04LX) es un pequeño láser de escaneo muy preciso, utilizado principalmente por estudiantes e investigadores en aplicaciones robóticas (figura 3.6)[24]. Este láser es capaz de reportar distancias de 40 mm a 5600 mm con 1 mm de resolución en un arco de 240°con una resolución angular de 0.36°[11]. Para su funcionamiento se necesita alimentar a través del puerto USB, con 5 V a 500 mA, lo cual lo hace adecuado para ser utilizado en un pequeño robot integrado con una batería, después necesita recibir comandos de operación para empezar a escanear y transmitir datos. Estos datos serán procesados por la Raspberry Pi 2 para detectar obstáculos en un modo de navegación autónomo.

[23]Figura recuperada de: https://www.adafruit.com/product/856
[24]Figura recuperada de: https://www.hokuyo-aut.jp/02sensor/07scanner/img/ urg-04lx-ug01-top.jpg

Figura 3.7: Láser Hokuyo

3.3. Componentes de alimentación eléctrica

3.3.1. Batería LiPo

Las baterías LiPo (litio y polímero) son un tipo de batería recargable que se utiliza comúnmente en los sistemas eléctricos de radiocontrol debido a su buena relación entre capacidad, voltaje, corriente y peso. Estas baterías son denominadas con un número que indica las celdas de que está compuesta y una letra que indica el tipo de conexión (S para serie y P para paralelo). Cada celda tiene un voltaje nominal de 3.7V.

En comparación con otras baterías, las baterías LiPo tienen algunas ventajas que las hacen adecuadas para llevar acabo un proyecto de control remoto o de robótica:

- Son ligeras y se pueden encontrar de muchas formas y tamaños.
- Tienen gran capacidad de almacenamiento en un tamaño reducido.
- Tienen una tasa alta de descarga para alimentar componentes electrónicos que demanden mucha potencia.

En cuanto a las desventajas existen algunos inconvenientes:

- Las baterías LiPo son inseguras si no se les trata correctamente debido al electrolito volátil del que están compuestas, un mal trato puede causar que se incendien o exploten.
- La carga, la descarga y el medio donde se almacenan requiere un cuidado adecuado para que tengan una vida útil de uso.

Para el prototipo de robot móvil propuesto se eligió una batería LiPo de 2 celdas, es decir, de un voltaje nominal de 7.4 V a 2200 mAh, que es adecuado para controlar los motores y alimentar todos los componentes eléctricos con un tiempo útil de 30 minutos aproximadamente.

3.3.2. Regulador de voltaje Step-Down

El regulador de voltaje Polou D15V35F5S3 [12] es un pequeño circuito que regula un voltaje de entrada de 4.5 V - 24 V a 5 V o 3.3 V dependiendo del uso que le dé. Este regulador puede dotar de 3.5 A en uso continuo para los proyectos electrónicos exigentes (figura 3.7)[25].

Debido a su pequeño tamaño y que cumple con las necesidades energéticas del prototipo de robot móvil, este dispositivo regulará el voltaje de la batería LiPo de 7.4 V a 5 V para alimentar el módulo central del robot móvil, la Raspberry Pi 2, además de otros componentes.

Figura 3.8: Regulador de Voltajee 5 V

3.3.3. Controlador para motores Pololu

El controlador Pololu VNH5019 [13] es un pequeño circuito diseñado para controlar motores de alta potencia. Tiene una voltaje de operación de 5.5 V a 24 V y puede entregar 12 A (30 A pico) por motor o 24 A (60 A pico) por un motor conectado a ambos canales (figura 3.8)[26]. Este controlador cumple las capacidades necesarias para otorgar el voltaje y la corriente de operación a cada uno de los 6 motores que estarán integrados en el prototipo de robot móvil, además tiene un modo se operación sencillo y puede ser fácilmente controlado por la Raspberry Pi 2.

[25]Figura recuperada de: https://www.pololu.com/product/2110
[26]Figura recuperada de: https://www.pololu.com/product/2507

Figura 3.9: Pololu Driver Shield

3.3.4. Hub USB

Después de haber creado el prototipo robot móvil se hicieron pruebas de rendimiento, los resultados arrojaron que la Raspberry Pi 2 no podía suplir de energía eléctrica a todos los componentes que tenía interconectados (el USB WiFi, el controlador de motores Pololu, la Raspberry Pi Cam, el láser Hokuyo, el arduino mega y el circuito de Sensores), ya que esto provocaba un mal funcionamiento en alguno de estos dispositivos, por lo cual se optó por incluir un nuevo componente en el prototipo, un hub USB alimentado de manera externa (figura 3.9)[27]. Este pequeño hub será energizado por el regulador de voltaje y se encargará de alimentar todos los dispositivos USB de la Raspberry Pi 2 (USB Wi-Fi, láser Hokuyo y el arduino) con la finalidad de que tenga un mejor rendimiento. Después de incluir este dispositivo mejoró el rendimiento del sistema, corrigiendo las fallas y el mal funcionamiento de los componentes.

Figura 3.10: USB Hub

[27]Figura recuperada de: http://i.ebayimg.com/images/a/28KGrHqZHJDYFCD!HlJp ZBQrw3BKp28g/s-l300.jpg

3.4. Wild Thumper (kit)

Wild Thumper 6WD [15] es un kit de robótica que incluye los componentes necesarios para armar la estructura de robot móvil de 6 ruedas con una configuración Skid Steer. Este kit está diseñado para que, tanto estudiantes como investigadores, desarrollen su propio circuito de control y lo prueben en el robot móvil (figura 3.11)[28].

El kit de desarrollo incluye los siguientes componentes:

- Una suspensión llamada "Super Twist", la cual mantiene todas las llantas en el suelo aún en terrenos difíciles como la tierra.
- Ruedas con púas de 120 mm para una mejor estabilidad en la navegación.
- 6 Motores de corriente contínua de 7.2 V con una reducción de 34:1.
- Agujeros de montaje de 10 mm, adecuados para colocar los circuitos electrónicos.
- Está hecho de una placa resistente de aluminio de 2 mm de espesor.

Figura 3.11: Wild Thumper 6WD

Este kit de robótica es adecuado para llevar a cabo el prototipo de robot móvil, ya que es resistente y adaptable a muchos tipos de terrenos, sus motores se adaptan a las necesidades energéticas de los demás componentes y su chasis nos da la ventaja de colocar cualquier circuito en el espacio que lo deseemos.

[28]Figura recuperada de: https://www.sparkfun.com/products/11056

CAPÍTULO 4

Diseño y programación del robot móvil

En este capítulo se tratarán todos los aspectos tomados en cuenta para la construcción, el diseño y la programación del robot móvil. En el capítulo 5 se describe el proceso de diseño y la programación de la aplicación en el sistema operativo Android, que actuará como un módulo de control de acciones del robot móvil.

El desarrollo de este proyecto de tesis está dividido en 3 etapas importantes:

- Construcción del robot móvil. Es esta etapa se describen los tipos de componentes utilizados en el proyecto y cuál será su tarea dentro del sistema. Además se describe el diseño de algunos circuitos y su implementación.
- Programación del módulo central. La Raspberry Pi 2 fungirá como un módulo central, en éste se llevarán acabo tareas de sensado, control de actuadores, transmisión de video y la comunicación con un dispositivo Android. En esta etapa se describe la programación de estos procesos.
- Aplicación Android. En esta etapa se describe el diseño y la programación de una aplicación en el sistema operativo Android. Esta aplicación se encargará de interactuar con el usuario mediante una interfaz amigable para qué, dependiendo de su elección, se determine la acción que ejecutará el robot móvil.

4.1. Construcción del robot móvil

Para llevar a cabo el desarrollo de este proyecto de tesis, se pensó en las características necesarias que debía tener la estructura del robot móvil: un armazón resistente a golpes; llantas grandes para pasar por encima de pequeños obstáculos y que fueran estables en terrenos difíciles (ya que el robot móvil podrá navegar tanto en el interior como al exterior del hogar); por último que esta estructura le fueran adaptables los circuitos que controlan al robot móvil. El kit Wild Thumper 6WD cumple con todas estas características, además de que su precio es accesible ($4000 pesos aproximadamente).

4.1.1. Diseño

El ensamblado de la estructura se hizo de manera sencilla, el armazón de aluminio estaba atornillado y ya incluía los motores con su suspensión, cada una de las llantas necesitó de un tornillo para que estuviera fija a la estructura.

La estructura de aluminio tiene dos cajones, uno entre cada par de llantas, los cuales sirven para colocar las tarjetas de desarrollo o los circuitos de control. En un cajón está la tarjeta controladora de motores Pololu, ésta se encarga de suministrar la el voltaje y la corriente de control de los seis motores, y el regulador de voltaje, el cual se encarga de suministrar 5 V a la tarjeta Raspberry Pi 2 y el hub USB. En el otro cajón está la tarjeta Raspberry Pi 2, ésta es el módulo central del robot móvil y se encarga de adquirir la información de los sensores y otorgar el control de los actuadores.

Circuito de sensores

Como se mencionó en el capítulo 4, una de las desventajas de la Raspberry Pi 2 es que carece de entradas analógicas para el tratamiento de la información proveniente de los sensores, por esta razón se optó por diseñar un pequeño circuito compuesto por estos sensores y un convertidor analógico-digital (ADC) que se comunica con la Raspberry Pi 2 por medio del protocolo SPI. La ventaja de utilizar el ADC es que cuenta con ocho canales de información, es decir, puede recibir datos de hasta 8 sensores y transmitirlos al mismo tiempo por medio de la conexión SPI. La Raspberry Pi 2 recibe estos datos y posteriormente los procesa.

El circuito de los sensores se compone del integrado MCP3008 (ADC), una foto-resistencia que mide la intensidad de luz en el ambiente, un sensor de temperatura para notificar la temperatura del hogar en todo momento y un sensor de gas que detecta humo o algún tipo de gas dentro del hogar. Además se toma la cantidad de voltaje de la batería para notificar a la aplicación qué porcentaje de carga tiene el robot móvil. Este circuito está colocado sobre la carcasa de aluminio del robot móvil, teniendo así una mejor adquisición en los datos de los sensores.

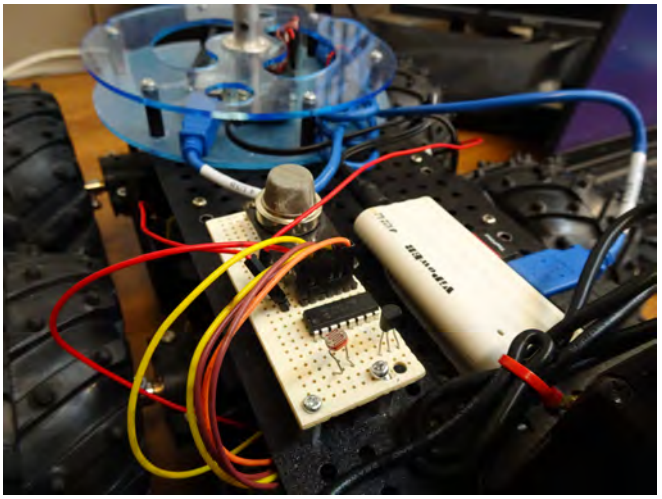


Figura 4.1: Circuito de sensores colocado en el robot móvil

Láser Hokuyo

El láser Hokuyo URG-04LX es un dispositivo de poco consumo de energía, adecuado para la detección de obstáculos durante la navegación autónoma del robot móvil. Este láser puede detectar obstáculos que estén entre 4 cm y 5.4 m de distancia en un arco de 240°. Debido a estas características y a su pequeño tamaño, éste es colocado sobre el armazón de aluminio en la parte frontal del robot móvil.

Raspberry Pi Cam

La Raspberry Pi Cam es una pequeña cámara diseñada específicamente para trabajar con la Raspberry Pi, puede tomar fotos con una resolución de 5 Mpx y grabar video en alta definición. Esta cámara se conecta a la Raspberry Pi 2 través de un cable plano, y por medio del sistema operativo, se activa el módulo que la controla. Su principal acción es transmitir video en tiempo real a la aplicación Android mientras el robot navega, debido a esto, está colocada en la parte frontal del robot móvil.

Figura 4.2: Raspberry Pi Cam y el láser Hokuyo

Torre de sensores de luz

La torre de sensores de luz es un dispositivo hecho en el laboratorio de Bio-Robótica de la Facultad de Ingeniería, que tiene como función medir la intensidad de la luz en cualquier dirección y notificar de dónde proviene. Este dispositivo está integrado por un poste largo, un anillo de 8 sensores de luz en su parte superior y una plataforma circular a su base que integra un Arduino Mega.

Figura 4.3: Torre de sensores de luz

Este dispositivo está colocado sobre la carcasa de aluminio del robot móvil en la parte trasera. Desde esta posición se conecta por medio de un cable USB al hub USB. Este último está conectado a la Raspberry Pi 2.

Raspberry Pi 2

La Raspberry Pi 2 actúa como un módulo central en el cual se procesa toda la información recibida por los sensores y la Raspberry Pi Cam. Está colocada en un cajón de la carcasa de aluminio del robot móvil, desde ahí se conecta por medio de 8 canales (VCC, GND, M1INA, M1INB, M1PWM, M2INA, M2INB, M2PWM) al controlador de motores Pololu. La Raspberry Pi 2 controla la dirección y la velocidad de los motores a través de estas conexiones.

La Raspberry Pi Cam está conectada por medio de un cable plano al conector CSI de la Raspberry Pi 2 para la transmisión de video. También están conectados los cuatro canales de la conexión SPI del circuito de los sensores. Por último, el hub USB se conecta a un puerto USB 2.0 de la Raspberry Pi 2 donde está conectado el Arduino Mega (de la torre de sensores de luz), el USB Wi-Fi y el láser Hokuyo.

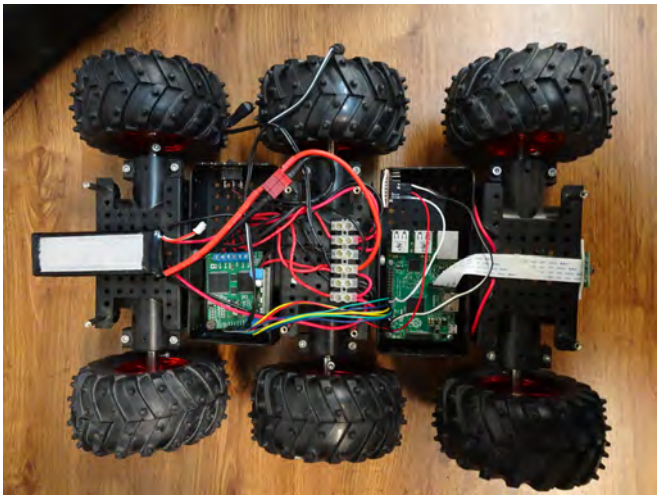


Figura 4.4: Raspberry Pi 2 y el controlador de motores Pololu colocados en el robot móvil

4.2. Programación de la Raspberry Pi 2

Como se mencionó en el capítulo 3, la Raspberry Pi 2 es el módulo central del robot móvil, en esta placa se adquieren los datos provenientes de los sensores para después procesarlos y, si es necesario, dar una tarea a algún actuador.

Se decidió programar en el lenguaje Python[16] por que es compatible con muchas herramientas de programación ya incluidas en el sistema operativo de la Raspberry Pi 2.

A continuación de describen todos los aspectos tomados en cuenta para la programación del módulo central del robot móvil.

4.2.1. Programación del circuito de los sensores

La programación del módulo que controla el circuito de los sensores se hizo en el lenguaje Python, y la comunicación por medio del protocolo SPI. Este protocolo debe habilitarse dentro del sistema operativo de la Raspberry Pi 2 de la siguiente manera:

- Teclear en una terminal **sudo raspi-config** para entrar a configuraciones.
- Navegar dentro del menú e ir a **avanced** > **options** > **SPI** y habilitar el módulo.
- Reiniciar el sistema operativo.

Una vez realizada esta configuración se activa el protocolo SPI, posteriormente se deben conectar los pines GPIO dedicados a esta conexión.

Conexiones

El circuito de los sensores está diseñado para la adquisición de datos provenientes de los sensores. Este circuito tiene integrado un convertidor analógico-digital que utiliza el protocolo de comunicación SPI para transmitir los datos a un microcontrolador, en este caso a la Raspberry Pi 2. En este dispositivo están conectados los sensores que serán utilizados en el proyecto de tesis.

A continuación se muestra la tabla 4.1 que representa las conexiones realizadas entre el ADC MCP3008 y el circuito de los sensores:

Tabla 4.1: Pines de conexión entre la Raspberry Pi 2 y el ADC MCP3008.

ADC MCP3008	**Sensores**
CH0	Fotoresistencia
CH1	Sensor de temperatura
CH2	Sensor de gas
CH3	Voltaje de la batería
CH4	Disponible
CH5	Disponible
CH6	Disponible
CH7	Disponible

La fotoresistencia tiene 2 pines de conexión, un pin está conectado a tierra (GND) y el otro pin está conectado al canal CH0 del convertidor analógico-digital MCP3008 y a una resistencia de 330 Ω, que a su vez está conectada a voltaje (5 V).

El sensor de temperatura LM35 tiene 3 pines de conexión, VCC está conectado a 5 V, GND está conectado a tierra (GND) y la salida analógica está conectada al canal CH1 del convertidor analógico-digital MCP3008.

El circuito del sensor de gas MQ-2 tiene 4 pines de conexión, de los cuales se usan 3, VCC está a 5 V, GND está conectado a tierra (GND) y el pin A0 al canal CH2 del convertidor analógico-digital.

Ya que el voltaje de operación del MCP3008 es de 5 V, es necesario dividir el voltaje de la batería de 7.4 V. Por esta razón se crea un divisor de voltaje con 2 resistencias de 10 kΩ, el punto entre las 2 resistencias está conectado al canal CH3 del MCP3008, en el cual se mide la mitad del voltaje actual de la batería.

Para poder comunicar la Raspberry Pi 2 con el circuito de los sensores es necesario utilizar 6 pines GPIO. A continuación se muestra la tabla 4.2 que representa las conexiones realizadas.

Tabla 4.2: Pines de conexión entre la Raspberry Pi 2 y el ADC MCP3008.

Raspberry Pi 2	**ADC MCP3008**
Pin 04 (5 V)	VDD
Pin 09 (GND)	GND
Pin 19 (GPIO_10 SPI_MOSI)	MOSI
Pin 21 (GPIO 09 SPI_MISO)	MISO
Pin 23 (GPIO 10 SPI_CLK)	SCK
Pin 24 (GPIO 08 SPI_CE0)	SS

La figura 4.5 muestra el diagrama de las conexiones realizadas:

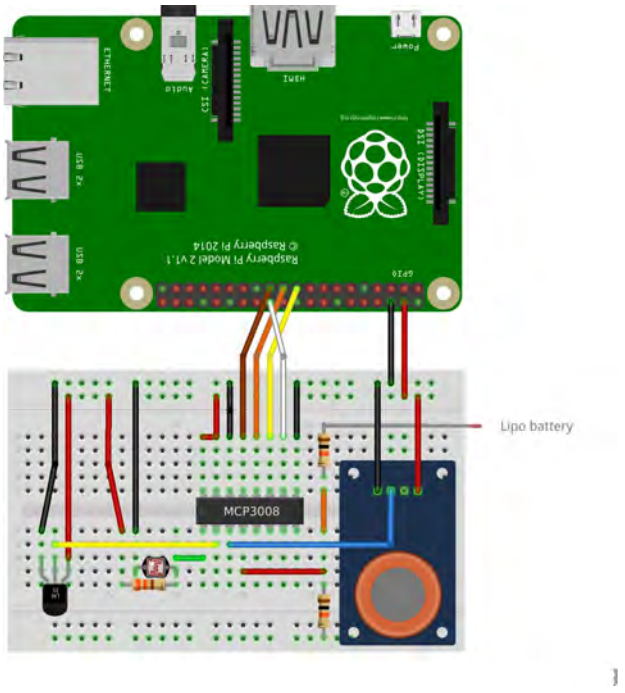


Figura 4.5: Diseño del circuito de sensores

Adquisición de datos

Para la adquisición de datos se utilizó el lenguaje de programación Python. Este módulo se encarga de adquirir la información transmitida mediante la comunicación SPI, es decir, se adquieren los datos de los 4 sensores y se guardan en una variable. Cada valor se procesa para convertirlo de un valor de voltaje a sus unidades correspondientes. Los valores recibidos del canal CH0 corresponden a la fotoresistencia, estos valores de voltaje son convertidos a lux(lx), la unidad de iluminancia. En el canal CH1 son recibidos los valores del sensor de temperatura, estos valores de voltaje son convertidos a grados centígrados. Los valores del sensor de gas son recibidos por el canal CH2, estos son convertidos a ppm (partes por millón) dependiendo del gas a detectar. Por último, en el canal CH3 se lee el voltaje de la batería y después es convertido a un rango de 0 a 100 % dependiendo del estado de carga de la batería.

A continuación se muestra el algoritmo utilizado para la toma de datos:

Algoritmo 1 Algoritmo de adquisición de datos del circuito de los sensores

Entradas:

$ReadChannel[]$ = Datos recibidos de los canales del ADC.

Salidas:

$light\_l$ = Luminosidad ambiente.
$temp\_l$ = Temperatura ambiente.
gas_l = Nivel de gas en el ambiente.
$battery$ = Nivel de batería.

1: **while** True **do**
2: Leer la información proveniente del ADC $ReadChannel[]$.
3: Convertir el valor de $ReadChannel[0]$ en lux y almacenarlo en $light\_l$.
4: Convertir el valor de $ReadChannel[1]$ en °C y almacenarlo en $temp\_l$.
5: Convertir el valor de $ReadChannel[2]$ en ppm y almacenarlo en gas_l.
6: Convertir el valor de $ReadChannel[3]$ en el porcentaje de carga de la batería y almacenarlo en $battery$.
7: Enviar gas_l al modulo de sensado de gas.
8: Concatenar todas las variables en una cadena de caracteres y enviarla a la aplicación Android.
9: **end while**

En resumen, este algoritmo recibe los valores provenientes del ADC y convierte todos sus datos a sus unidades correspondientes. Todos los valores son enviados a la aplicación en Android. El nivel de gas es enviado al módulo de sensado de gas para reportar, si es el caso, una posible fuga de gas o algún tipo de humo anormal[29].

4.2.2. Programación del controlador del láser Hokuyo

Se eligió el lenguaje Python para programar este módulo. En un principio se leyó la documentación sobre el láser Hokuyo, los códigos de arranque y toma de valores. Posteriormente se programó una función que almacena el bloque de información obtenido de la lectura del láser, lo decodifica y después almacena en un arreglo los valores de la distancia en milímetros en un rango de 240°. Con esta información se puede detectar si hay un obstáculo en la dirección en que navegue el robot móvil.

Adquisición de datos

Se describe el algoritmo desarrollado para la adquisición, decodificación y el almacenamiento de los datos de distancia en milímetros:

[29] Véase Apéndice A

Algoritmo 2 Controlador del Láser Hokuyo

Entradas:

$values_a$ = Bloque de datos codificados recibidos de la lectura del láser.

Salidas:

$values_r$ = Arreglo de 650 datos de distancia en milímetros.

Variables:

X=Variable de datos codificados.

$data_bin$=Variable de datos en binario.

1: **while** $i \leq 650$ **do**
2: **for** $j = 0$ to $j < 3$ **do**
3: Leer un caracter de $values_a$ y almacenarlo en X
4: Restar $30H$ a X
5: Convertir X a Binario
6: Concatenar X con $data_bin$
7: **end for**
8: Convertir $data_bin$ a decimal y almacenarlo en $values_r$
9: Eliminar el contenido de $data_bin$
10: **end while**

En resumen, el láser Hokuyo manda un bloque de información codificado(B) con los 650 datos de distancia. La documentación del láser dice que un valor de distancia está compuesto por 3 caracteres de este bloque de información, a cada uno de estos 3 valores le es restado 30 en hexadecimal y después es convertido a binario y almacenado en X. Estos 3 valores en binario son concatenados en una variable(C) que después se convierte a su equivalente en decimal. Este dato en decimal es la distancia en milímetros en una lectura del láser. Este procedimiento se hace 650 veces almacenándose en el arreglo $D[]$ [17] [30].

Rango de detección

El objetivo de este módulo es saber si hay un obstáculo y de qué dirección proviene, para esto necesitamos saber los rangos de muestreo del láser *(figura 4.6)*

De acuerdo a la figura, el frente está a 120°, con base en esto definiremos los rangos para detectar los obstáculos, para esto tendremos 3 casos:

- Hay un obstáculo en la parte izquierda del robot móvil. El robot móvil detectará si hay un obstáculo a 25 cm en un rango de 140°a 170°.
- Hay un obstáculo en la parte derecha del robot móvil. El robot móvil detectará si hay un obstáculo a 25 cm en un rango de 70°a 100°.
- Hay un obstáculo al frente del robot móvil. El robot móvil detectará se hay un obstáculo a 50 cm en un rango de 100°a 140°.

[30] Véase Apéndice B

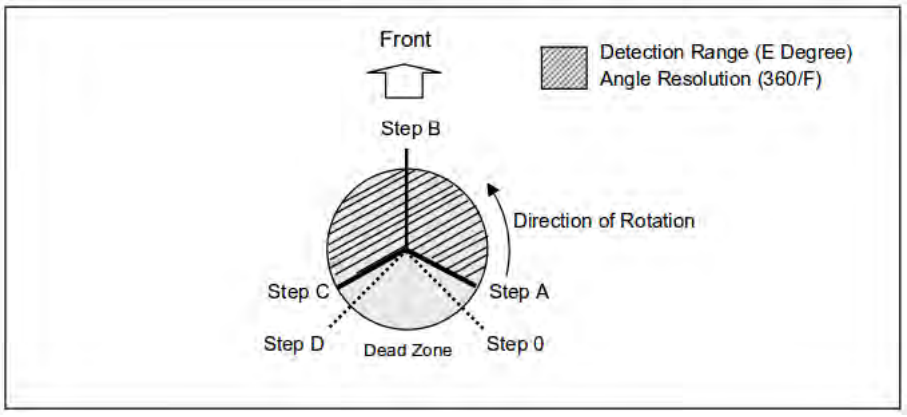


Figura 4.6: Rangos de muestreo de Láser Hokuyo

Después de realizar 25 pruebas se considera que 25 cm de distancia son adecuados para que el robot avance sin tener alguna colisión con un obstáculo. Por la parte frontal, se considera que 50 cm son adecuados para que el robot pueda detectar el obstáculo y ejecutar la acción correspondiente para evadirlo. Dependiendo dónde se encuentre el obstáculo (izquierda, derecha o al frente) se ejecutará un algoritmo de evasión de obstáculos por medio de una máquina de estados.

4.2.3. Programación de la torre de sensores de Luz

El robot móvil no tiene la capacidad de crear un mapa del entorno donde va a navegar, debido a esto no hay un punto inicial o un punto de partida establecido, y no puede recibir una instrucción para llegar a un punto final. Por esta razón se integra la Torre de sensores de luz, ésta podrá darnos un punto destino al cual deba llegar el robot móvil, es decir, este dispositivo indica dónde se encuentra la fuente de luz y en qué momento se llega a ésta. La finalidad de este dispositivo es que se puedan ejecutar algoritmos de navegación en máquinas de estados que nos lleven a un punto final sin chocar con los obstáculos.

La idea principal es detectar de dónde proviene la mayor fuente de luz, navegar hacia ella evadiendo obstáculos y notificar al sistema cuando se haya alcanzado el punto de interés, en este caso será una lámpara.

Adquisición de datos

La programación del circuito que controla la adquisición y el envío de información de los sensores de luz es realizado en el entorno de programación Arduino IDE, en un lenguaje de programación pseudo-C.

A continuación se describe el algoritmo utilizado para la adquisición de datos:

Algoritmo 3 Algoritmo de toma de datos de la Torre de Sensores de luz

Entradas:

I = Comando de toma de datos

Intensidad de luz proveniente de los sensores de luz.

Salidas:

C = Comando de datos para la Raspberry Pi 2.

```
1: while True do
2:   if I == "string" then
3:     for i = 0 to i < 4 do
4:       Leer el valor de 2 resistencias adjuntas, hacer el promedio y almacenarlo en C[i]
5:     end for
6:     Convertir a C[] en una cadena de datos y enviarla por el puerto serial
7:   end if
8: end while
```

Se programó un módulo en la Raspberry Pi 2 que solicita la lectura de valores de la Torre de sensores de luz mediante un comando de inicio, en este caso la cadena de caracteres *"string"*. El Arduino Mega se encarga de recibir esta cadena y la compara con el comando de inicio, si es correcta se inicia la lectura y se hace un promedio cada 2 valores de las 8 resistencias, resultado así 4 valores que nos indican la intensidad luminosa en 4 direcciones: adelante, atrás, derecha e izquierda. Estos valores son transformados a una cadena de caracteres y se envían a la Raspberry Pi 2 por medio del puerto serial. Dependiendo en qué dirección se encuentre la mayor intensidad luminosa se ejecutará un algoritmo en una máquina de estados que tendrá como objetivo llegar a ese punto [31].

4.2.4. Programación del módulo de la tarjeta Pololu

La tarjeta controladora de motores Pololu es controlada por la Raspberry Pi 2 con una conexión de 8 canales. Ésta le indicará la velocidad y la dirección que deberán tener los motores de cada lado del robot móvil, en este caso serán 3 motores por cada lado. La tarjeta Pololu se encarga de otorgar el voltaje necesario para dar la velocidad y dirección solicitadas por la Raspberry Pi 2.

Conexiones

Para controlar la tarjeta Pololu se utilizan 8 canales de la Raspberry Pi 2, 2 de alimentación y 6 de datos (3 por cada lado). La tabla 4.3 muestra las conexiones de cada pin de la Raspberry Pi 2 con la tarjeta Pololu.

[31] Véase Apéndice D

Tabla 4.3: Pines de conexión entre la Raspberry Pi 2 y la tarjeta Pololu.

Raspberry Pi 2	**Tarjeta Pololu**
Pin 02 (5 V)	VDD
Pin 14 (GND)	GND
Pin 12 (GPIO_18)	M1PWM
Pin 16 (GPIO_23)	M2PWM
Pin 18 (GPIO_24)	M1INA
Pin 18 (GPIO_25)	M1INB
Pin 11 (GPIO_17)	M2INA
Pin 13 (GPIO_27)	M2INB

La figura 4.7 muestra el diagrama de conexiones realizado:

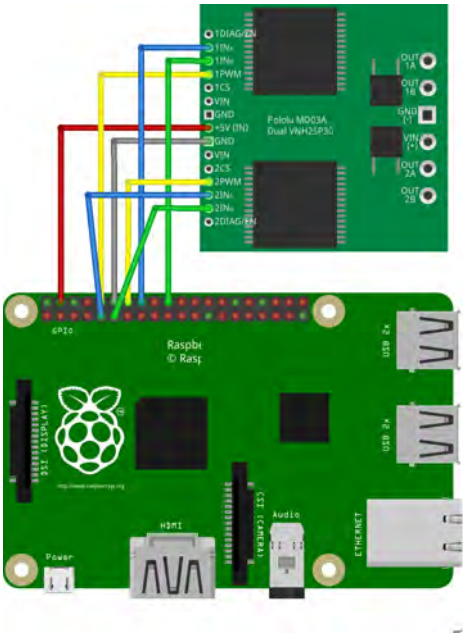


Figura 4.7: Conexiones de la Raspberry Pi 2 con el controlador de motores Pololu

Control

La tarjeta controladora de motores Pololu controla los 6 motores, 3 por lado. Cada lado necesita de 3 canales de información:

- M1INA y M1INB tienen como propósito dar el sentido al cual girarán los motores del lado izquierdo. Reciben un dato de tipo booleano, es decir *verdadero o falso* (*TRUE O FLASE*).
- M1PWM recibirá una señal PWM (modulación por ancho de pulso) que se encargará de dar la velocidad a los motores del lado izquierdo. La Raspberry Pi 2 maneja el PWM

como un porcentaje, siendo el 100 % todo el voltaje disponible de la fuente de voltaje. Los motores comenzarán a girar con un 5 % de PWM.

- De igual manera funcionan M2INA y M2INB para dar el sentido del giro y M2PWM para la velocidad de los motores del lado derecho.

A continuación se muestra la tabla 4.4 que representa el control de los motores:

Tabla 4.4: Control de los motores.

M1INA ó M2INA	**M1INB ó M2INB**	**M1PWM ó M2PWM**	**Movimiento**
TRUE	FLASE	5 % - 100 %	Giro negativo
FALSE	TRUE	5 % - 100 %	Giro positivo
FALSE	FALSE	*	Sin giro
TRUE	TRUE	*	Sin giro

Con base en estos valores se muestra la tabla de movimiento del robot móvil *(tabla 4.5)*:

Tabla 4.5: Control del movimiento del robot móvil.

Motores izquierdos	**Motores derechos**	**Movimiento**
Giro positivo	Giro negativo	Adelante
Giro negativo	Giro positivo	Atrás
Giro positivo	Giro positivo	Giro a la derecha (negativo)
Giro negativo	Giro negativo	Giro a la izquierda (positivo)
Sin giro	Sin giro	Sin movimiento

A partir de esta información se programaron 5 funciones que establecerán la dirección en la que navegará el robot móvil:

- forward(). El movimiento del robot será hacia adelante. Se establece la velocidad en un 20 %, adecuada para navegar en el hogar.
- backward(). El movimiento del robot será hacia atrás. Se establece la velocidad en un 20 %.
- left(). El movimiento del robot será un giro hacia la izquierda. La velocidad de los PWM se estableció en 30 %.
- right(). El movimiento del robot será un giro a la derecha, la velocidad de los PWM se estableció en 30 %.
- stop(). El robot se queda sin movimiento.

A partir de estas funciones básicas se tendrán los tipos de movimiento que puede realizar el robot móvil, posteriormente se describirán los comportamientos que tendrá dependiendo del modo de navegación (autónomo o controlado)[32].

[32]Véase Apéndice C

4.2.5. Herramientas de software

MJPG-Streamer.

MJPG-Streamer es una herramienta que se encarga de capturar imágenes en formato JPG de cámaras web en Linux, con ellas transmite un video en tiempo real a través del protocolo http en un navegador web. Esta herramienta tiene como propósito transmitir video en tiempo real tomado por la Raspberry Pi Cam a la aplicación Android y a un navegador web [18].

Para activar MJPG-Streamer se tendrán que seguir estos pasos:

- Descargar el paquete: *wget http://lilnetwork.com/download/raspberrypi/mjpg-streamer.tar.gz*
- Descomprimirlo: *tar xvzf mjpg-streamer.tar.gz*
- Descargar las librerías necesarias: *sudo apt-get install libjpeg8-dev —— sudo apt-get install imagemagick*
- Instalar y ejecutar MJPG-Streamer.

Esta herramienta transmite el video tomado por la Raspberry Pi Cam en la dirección IP a la que se ha conectado, desde ahí será mostrado en la aplicación Android y, como alternativa, podrá ser visto en cualquier computadora conectada a la misma red.

4.2.6. Programación del módulo central

El módulo central tiene como función ejecutar 1 de los 4 modos de operación del robot móvil, además se encarga de ejecutar las herramientas y los controladores de los dispositivos integrados al robot móvil, como el Arduino Mega, la tarjeta controladora Pololu, el circuito de sensores, el láser Hokuyo, la conexión WiFi, entre otras.

Conexión Wi-Fi

La Raspberry Pi 2 no tiene integrado un módulo de red inalámbrica, por esta razón se optó por usar un adaptador USB Wi-Fi.

Se configuró una conexión de red con una IP estática, con los siguientes propósitos:

- Hacer la conexión con la aplicación Android.
- Transmitir el video tanto a la aplicación Android como a otra computadora conectada a la misma red.
- Enviar correos electrónicos.
- Actualizar el sistema.

El adaptador USB WiFi irá conectado al hub USB y al iniciar el sistema se establece la conexión automáticamente.

Script de inicio

La herramienta MJPG-Streamer no tiene como opción empezar a transmitir video de manera automática, por esta razón se programó un script que se ejecuta al iniciar el sistema operativo de la Raspberry Pi 2. Este script configura las características del video que será transmitido (como la resolución y los cuadros por segundo), después ejecuta el programa para empezar a transmitir el video el tiempo real. Además se encarga de ejecutar el módulo principal para que se pueda establecer una conexión con la aplicación Android [19].

Modos de operación

El robot móvil cuenta con 4 modos de operación, éstos son elegibles desde la aplicación móvil después de que se haya establecido una conexión:

- *Manual Control*. En este modo de operación se controla al robot móvil desde la aplicación Android. Cuenta con 5 comandos de movimiento: adelante, atrás, izquierda, derecha y alto.
- *Sensors Value*. En este modo de operación se recopila la información proveniente del circuito de los sensores, se procesa y se envía a la aplicación Android para poder ser vista.
- *Obstacle Avoidance*. En este modo de operación el robot móvil navega de manera autónoma evadiendo obstáculos sin llegar a un punto de destino.
- *Search of a light source*. En este modo de operación el robot navega de manera autónoma evadiendo obstáculos en busca de una fuente intensa de luz, en este caso una lámpara, la cual simula un punto destino en su navegación.

A continuación se describen los modos de operación del robot móvil.

Manual Control

Este modo de operación consta de 5 comandos de movimiento que controlan directamente al robot móvil desde la aplicación en Android.

Los comandos de navegación son los siguientes:

- Comando *Forward*. Este comando de operación ejecuta la función *forward()* para que el robot móvil avance hasta que el usuario presione la tecla de *alto*.
- Comando *Backward*. Este comando de operación ejecuta la función *backward()* para que el robot móvil retroceda hasta que el usuario presione la tecla de *alto*.
- Comando *Left*. Este comando de operación ejecuta la función *left()* para que el robot móvil gire a la izquierda, es decir, que dé un giro positivo hasta que usuario presione la tecla de *alto*.
- Comando *Right*. Este comando de operación ejecuta la función *right()* para que el robot móvil gire a la derecha, es decir, que dé un giro negativo hasta que el usuario presione la tecla de *alto*.

- Comando *Stop*. Este comando de operación ejecuta la función *stop()* para que el robot móvil quede sin movimiento, en la espera de un nuevo comando de navegación.

En este modo de operación el usuario controla totalmente la dirección y al distancia que recorrerá el robot móvil. Además se estará transmitiendo el video en tiempo real a la aplicación móvil, por lo que se verá la ruta que recorrerá el robot.

Este modo de operación estará siendo ejecutado hasta que en la aplicación móvil Android se elija otro modo de operación.

Lectura de sensores

Este modo de operación se encargará de adquirir los datos provenientes del circuito de los sensores y procesarlos para convertirlos de un valor de voltaje a las unidades correspondientes de cada sensor.

Se estarán sensando 4 canales provenientes del circuito de los sensores:

- Fotoresistencia. Se encargará de medir la iluminancia o nivel de iluminación en el ambiente, su unidad es el lux(lx). Para convertir los valores que transmite el ADC y transmitirlos a la aplicación móvil se siguió el siguiente algoritmo:

Algoritmo 4 Función que convierte los datos adquiridos de la fotoresistencia en **lx**

Entradas:

f = Valor de la fotoresistencia en **V**.
A = 1000. Resistencia en la luz KΩ
B = 15. Resistencia en la oscuridad (10 Lux) KΩ
RC = 10. Resistencia en la calibración KΩ

Salidas:

R = Iluminacia en **lx**

1: $R = f*A*10/B*RC*(1024-f)$
2: Se convierte R a una cadena de caracteres.
3: Se concatena R con la cadena de caracteres *"comando"* que será enviada a la aplicación Android.

- Sensor de temperatura. Se encargará de medir la temperatura del ambiente en grados centígrados. El fabricante del sensor de temperatura TMP36 especifica que la salida de voltaje será 10 mV por cada grado de temperatura.

  Para convertir los valores que transmite el ADC se siguió el siguiente algoritmo:

Algoritmo 5 Función que convierte los datos adquiridos del sensor de temperatura en °**C**

Entradas:

t = Valor del sensor de tempertura en **V**.
v = 5. Voltaje máximo sensado.
m = 1024. Valor máximo de la entrada digital.

Salidas:

T = Temperatura en °C

1: $T = (v/m*t*100)$-50
2: Se convierte T a una cadena de caracteres.
3: Se concatena T con la cadena de caracteres *"comando"* que será enviada a la aplicación Android.

- Sensor de Gas. Este sensor se encarga de detectar concentraciones anormales de diferentes tipos de gases. En este proyecto alertará sobre altas concentraciones de gas butano, que comúnmente es el combustible utilizado en las estufas del hogar. Para obtener el valor específico de cada gas se tiene una curva de configuración diferente.

  Para convertir los valores provenientes del ADC en partes por millón(ppm) se siguió el siguiente algoritmo:

Algoritmo 6 Función que convierte los datos adquiridos del sensor de gas en *ppm*

Entradas:

l = Valor del sensor de gas en **V**.
$G[]$ = 2.3,0.21,-0.47. Curva de sensibilidad del sensor de gas para el gas butano.
r = 10. Resistencia a la calibración.

Salidas:

p = Concentración de gas.

1: $p = potencia(10,((log(l/RC)\text{-}G[1])/G[2])+G[0])$
2: Se envía p al módulo de sensado de gas.
3: Se convierte p a una cadena de caracteres.
4: Se concatena p con la cadena de caracteres *"comando"* que será enviada a la aplicación Android.

Una vez procesados los datos de los sensores, éstos son concatenados en una cadena que será enviada a la aplicación móvil Android con el siguiente formato:

comando = "_modo de operación_iluminancia_temperatura_nivel de gas_nivel de batería_"

por ejemplo:

comando = "_2_128_25_30_80_"

Donde 2 es el modo de operación, 128 es la iluminancia en lux, 25 es la temperatura, 30 es la concentración de gas en ppm y 80 es el porcentaje de batería actual.

Después de que la cadena es recibida por la aplicación móvil, es procesada para ser mostrada en la interfaz del dispositivo. El modo de operación *lectura de sensores* estará siendo ejecutado hasta que el usuario cambie por otro modo de operación.

4.2.7. Máquinas de estado

El módulo central de la Raspberry Pi 2 está programado para ejecutar 4 modos de operación, dos de ellos son *Evasión de Obstáculos* y *Búsqueda de una fuente de Luz* en donde se utilizarán algoritmos en máquinas de estado para navegar autónomamente y llegar a un punto final. Estas máquinas de estado tomarán las decisiones sobre la navegación autónoma del robot móvil. A continuación se describe la programación de las máquina de estado.

Máquina de estados de *Evasión de obstáculos*

En este modo de operación el robot navegará autónomamente esquivando obstáculos, sin un punto fijo al cual llegar. Esto nos sirve en un escenario donde el robot no tenga un destino y solo esté navegado a través del hogar recopilando información en los sensores y transmitiendo el video en tiempo real.

Se programó una función que recibe los datos del láser hokuyo e indica si hay un obstáculo y en qué dirección se encuentra: al frente, a la izquierda o a la derecha del robot móvil. Para cada una de estas posibilidades hay una acción correspondiente en el algoritmo de la máquina de estados.

En la figura 4.8 se muestra la máquina de estados programada.

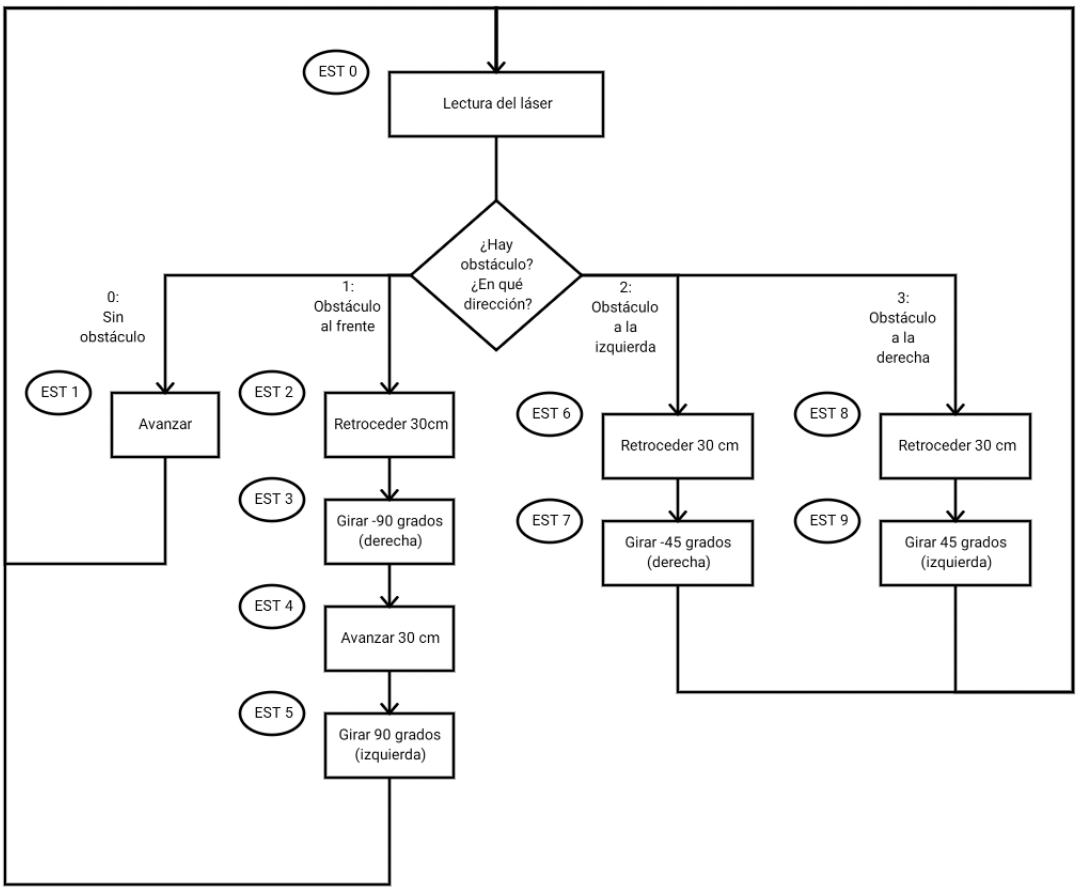


Figura 4.8: Máquina de estados del modo de operación Evasión de obstáculos

Esta máquina de estados está compuesta por 9 estados:

- Estado Cero. En este estado se ejecuta la función *lectura_laser()*, que indica si hay un obstáculo al frente, a la izquierda o a la derecha del robot móvil. En caso de que no haya obstáculo se pasará al *estado uno*. Si hay un obstáculo al frente se pasará al *estado dos*. Si hay un obstáculo a la izquierda se pasará al *estado seis*. Por último si el obstáculo se encuentra del lado derecho se pasará al *estado ocho*.
- Estado Uno. En caso de que no encuentre un obstáculo el robot avanzará libremente ejecutando la función *forward()*. Después se regresa al *estado cero*.
- Estado Dos. En caso de haber encontrado un obstáculo al frente el robot móvil retrocederá 30 cm ejecutando la función *backward()* y pasará al *estado tres*.
- Estado Tres. El robot móvil dará un giro de 90°a la derecha ejecutando la función *right()* y pasará al *estado cuatro*.

- Estado Cuatro. El robot móvil avanzará 30 cm ejecutando la función *forward()* y pasará al *estado cinco*.
- Estado Cinco. El robot móvil dará un giro de 90°a la izquierda ejecutando la función *left()* y pasará al *estado cero* donde se volverá a evaluar si hay un obstáculo.
- Estado Seis. En caso de haber encontrado un obstáculo a la izquierda el robot móvil retrocederá 30 cm ejecutando la función *backward()* y pasará al *estado siete*.
- Estado Siete. El robot móvil dará un giro de 45°a la derecha ejecutando la función *right()* y pasará al *estado cero* donde se volverá a evaluar si hay un obstáculo.
- Estado Ocho. En caso de haber encontrado un obstáculo a la derecha el robot móvil retrocederá 30 cm ejecutando la función *backward()* y pasará al *estado nueve*.
- Estado Nueve. El robot móvil dará un giro de 45°a la izquierda ejecutando la función *left()* y pasará al *estado cero* donde se volverá a evaluar si hay un obstáculo.

De esta manera el robot estará navegando y evadiendo obstáculos hasta que el usuario cambie el modo de operación. Todos los estados de la máquina de estados serán reportados en la interfaz de la aplicación móvil.

Máquina de estado de *Búsqueda de una fuente de luz*

En este modo de operación el robot móvil navegará autónomamente en la búsqueda de una fuente de luz, en este caso será una lámpara que simulará un punto destino al cual deba llegar. Como el robot no tiene la capacidad de llegar a un punto específico se simulará éste con la fin de probar algoritmos de navegación con máquinas de estado. Por ejemplo, ir de la sala a una habitación.

Al igual que la máquina de estados anterior, el robot móvil navegará esquivando obstáculos ejecutando la función *lectura_laser()* que indicará si hay un obstáculo y en qué dirección se encuentra.

En la figura 4.9 se muestra la máquina de estados programada.

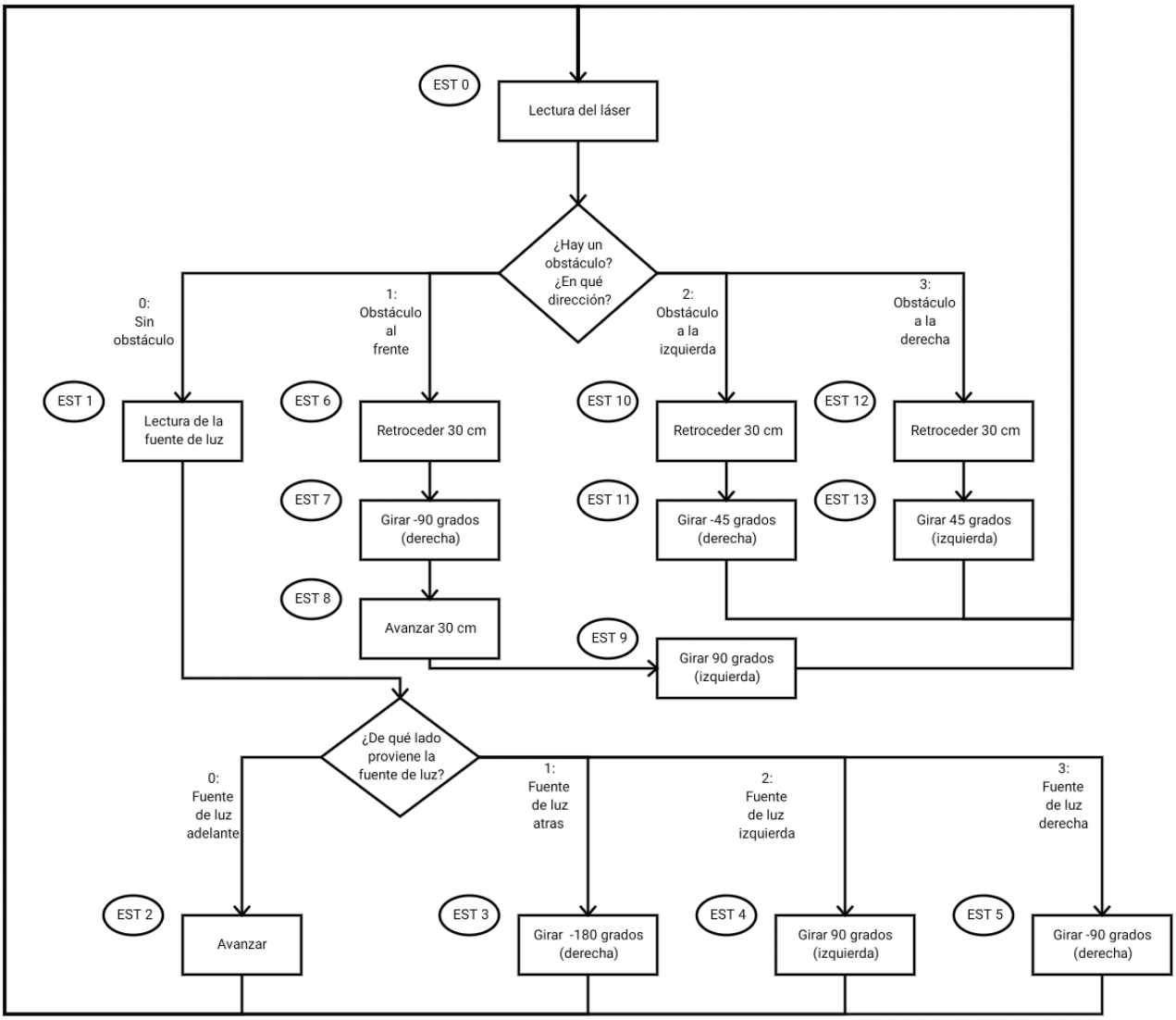


Figura 4.9: Máquina de estados del modo de operación Búsqueda de una fuente de luz

Esta máquina de estados está compuesta por 13 estados:

- Estado Cero. En este estado se ejecuta la función *lectura_laser()*, que indica si hay un obstáculo al frente, a la izquierda o a la derecha del robot móvil. En caso de que no haya obstáculo se pasará al *estado uno*. Si hay un obstáculo al frente se pasará al *estado seis*. Si hay un obstáculo a la izquierda se pasará al estado *estado diez*. Por último si el obstáculo se encuentra del lado derecho se pasará al estado *estado doce*.
- Estado Uno. En caso de que no encuentre un obstáculo se ejecutará la función *lectura_luz()* que indicará de que lado proviene la mayor intensidad de luz. En caso de que la luz se encuentre al frente se pasará al *estado dos*. Si se encuentra atrás se pasará al

estado tres. Si se encuentra del lado izquierdo se pasará al *estado cuatro*. Y por último si se encuentra del lado derecho se pasará al *estado cinco*.

- Estado Dos. En caso de haber encontrado la mayor intensidad de luz al frente, el robot móvil avanzará libremente ejecutando la función *forward()* y pasará al *estado cero*, donde se volverá a evaluar si hay un obstáculo.
- Estado Tres. En caso de haber encontrado la mayor intensidad de luz atrás, el robot móvil girará 180°a la derecha ejecutando la función *right()* y pasará al *estado cero*, donde se volverá a evaluar si hay un obstáculo.
- Estado Cuatro. En caso de haber encontrado la mayor intensidad de luz a la izquierda, el robot móvil girará 90°a la izquierda ejecutando la función *left()* y pasará al *estado cero*, donde se volverá a evaluar si hay un obstáculo.
- Estado Cinco. En caso de haber encontrado la mayor intensidad de luz a la derecha, el robot móvil girará 90°a la derecha ejecutando la función *right()* y pasará al *estado cero*, donde se volverá a evaluar si hay un obstáculo.
- Estado Seis. En caso de haber encontrado un obstáculo al frente el robot móvil retrocederá 30 cm ejecutando la función *backward()* y pasará al *estado siete*.
- Estado Siete. El robot móvil dará un giro de 90°a la derecha ejecutando la función *right()* y pasará al *estado ocho*.
- Estado Ocho. El robot móvil avanzará 30 cm ejecutando la función *forward()* y pasará al *estado nueve*.
- Estado Nueve. El robot móvil dará un giro de 90°a la izquierda ejecutando la función *left()* y pasará al *estado cero* donde se volverá a evaluar si hay un obstáculo.
- Estado Diez. En caso de haber encontrado un obstáculo a la izquierda el robot móvil retrocederá 30 cm ejecutando la función *backward()* y pasará al *estado once*.
- Estado Once. El robot móvil dará un giro de 45°a la derecha ejecutando la función *right()* y pasará al *estado cero* donde se volverá a evaluar si hay un obstáculo.
- Estado Doce. En caso de haber encontrado un obstáculo a la derecha el robot móvil retrocederá 30 cm ejecutando la función *backward()* y pasará al *estado trece*.
- Estado Trece. El robot móvil dará un giro de 45°a la izquierda ejecutando la función *left()* y pasará al *estado cero* donde se volverá a evaluar si hay un obstáculo.

De esta manera el robot estará navegando y evadiendo obstáculos hasta que se llegue a la fuente de luz o el usuario cambie el modo de operación. Todos los estados de la máquina de estados serán reportados en la interfaz de la aplicación móvil.

4.2.8. Correo electrónico

El robot móvil estará conectado a la red de internet, esta característica se aprovechará para que se puedan enviar notificaciones vía correo electrónico a uno o varios usuarios previamente configurados en el sistema. Se creó una cuenta en G-mail llamada *raspirobot7@gmail.com* y se programó un módulo que se encargará de enviar correos electrónicos automáticamente cuando el sistema reporte una alerta.

A continuación se describe el funcionamiento del módulo:

- Una función recibe como parámetros el destinatario, el asunto y el mensaje en el correo electrónico.
- Se crea una conexión con el servidor SMTP de gmail.
- Se configura el mensaje.
- Se envía el mensaje a los destinatarios programados.

A continuación se describen las situaciones en las que se enviará un correo electrónico.

Correo electrónico de inicio

Al encender la Raspberry Pi 2 ésta se conecta a una red local previamente configurada y se le asigna una dirección IP estática, estos datos son necesarios para establecer una conexión con la aplicación Android.

El usuario recibirá un correo electrónico que le indicará que el robot móvil ha sido encendido, dentro se este mensaje se encuentra la dirección IP a la cual se conectó y el puerto al que se debe hacer la conexión con la aplicación en Android. Esta información no cambiará a menos que la dirección IP ya esté asignada, en este caso es necesario revisar el correo electrónico para saber a que dirección IP se ha conectado. Si la dirección IP y el puerto no son correctos no se podrá establecer una conexión con el robot móvil.

Correo electrónico de alerta de gas

El módulo encargado de monitorear la presencia de humo o algún tipo gas en el ambiente enviará un correo electrónico cuando detecte niveles anormales. Dentro de este correo se especificará que se trata de una alerta y enseguida se mostrarán los niveles de gas en ppm. Este módulo estará sensando cada medio segundo [33].

[33] Véase Apéndice E

CAPÍTULO 5

Aplicación Android

Android Studio es un entorno de desarrollo integrado (IDE) para la programación de aplicaciones que se ejecutan dentro del sistema operativo Android. Tiene grandes ventajas de programación, así como diversos emuladores que se adaptan a las características físicas de los teléfonos móviles para los cuales se desarrollará la aplicación. Se eligió el lenguaje de programación Java y XML para la interfaz gráfica [5].

La aplicación Android se encargará de dar instrucciones al robot móvil, es decir, indicar su modo de operación, solicitar la información de los sensores, y principalmente para ver lo que ocurre a través de la Raspberry Pi Cam.

A continuación se describe el desarrollo de la aplicación Android.

5.1. Comunicación con el servidor

La Raspberry Pi 2 y la aplicación Android establecerán comunicación por sockets de internet [20]. Utilizarán un sistema cliente-servidor, donde el cliente será la aplicación Android y el servidor será la Raspberry Pi 2. La función del servidor se programó en Python, ésta esperará la conexión con un cliente y posteriormente entrará a "modos de operación" dentro del módulo central. El cliente se programó en Java, éste establecerá una conexión con el servidor insertando los campos "IP y Port", de ser correctos se establecerá una conexión [21].

La comunicación entre el módulo central y la aplicación Android será por medio de cadenas de caracteres (comandos). La aplicación Android enviará estos comandos con el propósito de indicar una acción o de hacer la petición de algún dato de los sensores. La sintaxis del comando es la siguiente:

Comando: "*_modoDeOperación_datosNecesarios*"

5.2. Actividades de la aplicación Android

En la aplicación Android se programaron 6 actividades (ventanas)[22], que cumplen con distintas tareas dependiendo de la elección del usuario, tratan sobre el control del robot móvil por parte del usuario, la navegación autónoma del robot móvil, el despliegue de información proveniente de los sensores y la transmisión de video en tiempo real desde la Raspberry Pi Cam.

Las actividades programadas fueron las siguientes:

- Actividad *Connect*.
- Actividad *Operation Mode*.
- Actividad *Sensors*.
- Actividad *Manual Control*.
- Actividad *Obstacle Avoidance*.
- Actividad *Search of a light Source*.

En la figura 5.1 se muestra el funcionamiento de la aplicación.

A continuación se describen a detalle el diseño y la programación de cada actividad.

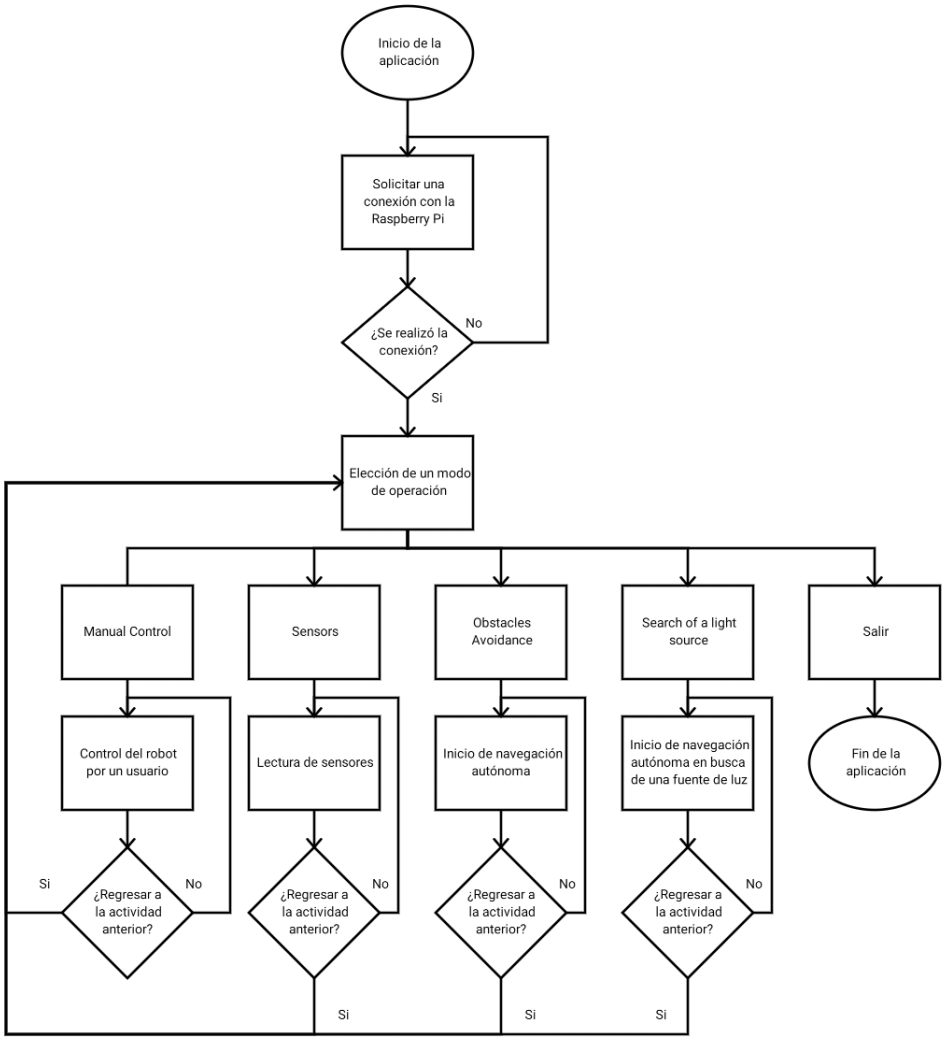
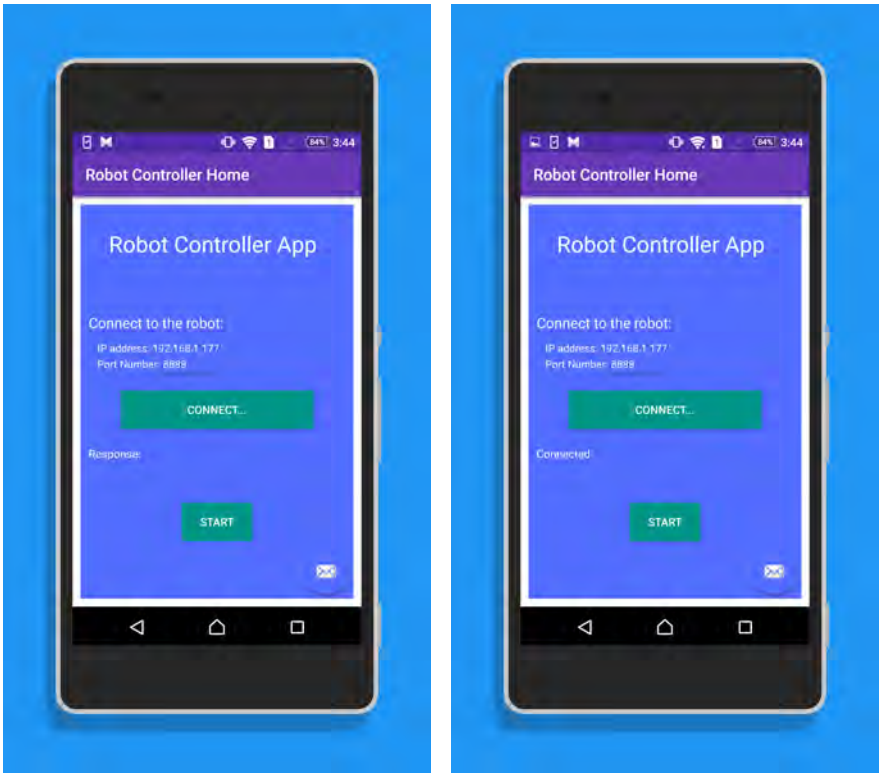


Figura 5.1: Funcionamiento de la aplicación Android

5.2.1. Actividad *Connect*

Esta es la pantalla principal de la aplicación Android. Tiene como propósito realizar la conexión con el robot móvil a través de Sockets, para que esta conexión se lleve a cabo se tendrán que introducir 2 parámetros: *IP* y *Port*. El primer campo pertenece a la misma dirección IP a la cual se ha conectado el robot móvil, esta IP es estática y se puede ver el el correo electrónico de inicio, *Port* será el puerto previamente programado en el servidor, de igual manera se puede apreciar en el correo electrónico de inicio. Este puerto actuará como una contraseña para que otra aplicación no pueda conectarse sin consentimiento.



(a) Antes de realizar la conexión (b) Después de realizar la conexión

Figura 5.2: Actividad Connect

Una vez introducidos estos campos el usuario oprimirá el botón *Connect*, dependiendo si se realizó la conexión o no se desplegará un mensaje. En el caso que la conexión sea exitosa el mensaje será *"Connected" (figura 4.10 (b))*, que indica que se ha realizado una conexión con el servidor y el botón de *Start* se activará. Si la conexión falló se indicará en el mensaje sobre el error ocurrido y el botón de *Start* permanecerá desactivado hasta que se hayan introducido los campos de manera correcta.

Después de haber establecido la conexión, el servidor enviará al cliente el comando de confirmación *"_conectado_"*, éste lo recibirá como parámetro para desbloquear el botón *Start*. Este botón cambia a la actividad "Operation Mode", donde se comenzará a dar instrucciones de control al robot móvil [34].

5.2.2. Actividad *Operation Mode*

Esta actividad tiene como propósito dar a elegir al usuario el modo de operación que será ejecutado en el robot móvil.

En la barra superior se muestra el nombre de la actividad, en este caso es *"Operation Mode"*, del lado derecho se encuentra el primer dato que recibimos del módulo central, el porcentaje de batería actual. En la parte superior de la ventana se puede apreciar el video transmitido en tiempo real desde la Raspberry Pi Cam. Debajo se encuentran los botones que dan a elegir el modo de operación que será ejecutado en el módulo central de la Raspberry

[34] Véase Apéndice F

Pi 2:

- *Manual Control*. En este modo de operación se controla directamente al robot móvil a través de comandos de movimiento.
- *Sensors Value*. En este modo de operación se muestran los datos obtenidos por los sensores en sus unidades correspondientes.
- *Obstacle Avoidance*. En este modo de operación el robot navegará sin un destino evadiendo obstáculos.
- *Search of a light source*. En este modo de operación el robot móvil navegará evadiendo obstáculos hasta encontrar una fuente intensa de luz (una lámpara).

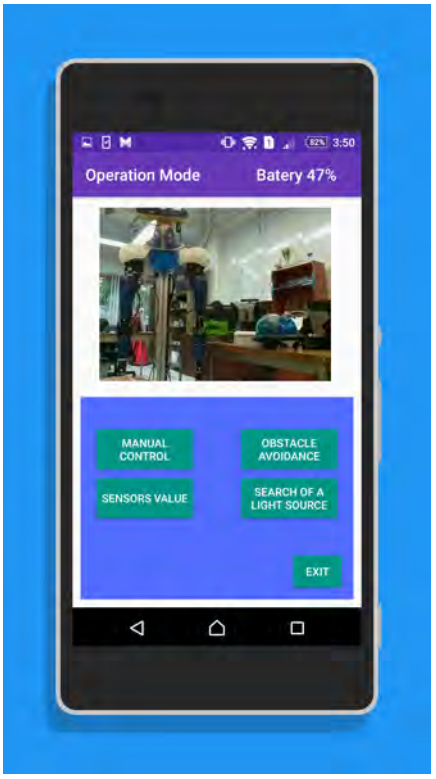


Figura 5.3: Actividad Operation Mode

Dependiendo del modo de operación elegido por el usuario, la aplicación Android enviará un comando específico para ejecutar ese modo de operación en el robot móvil. El módulo central contestará con otro comando para confirmar que se ha ingresado al modo de operación elegido [35].
La sintaxis es la siguiente:

Comando: *"_modoDeOperación_"*

Por ejemplo, si se elige el modo de operación 3:

Comando: *"_3_"*

5.2.3. Actividad *Manual Control*

Esta actividad tiene como propósito que el usuario ejecute comandos de movimiento en el robot móvil. Su diseño es similar a la actividad anterior.

Se mostrará el nombre de la actividad *"Manual Control"* en la barra superior, del lado derecho se presenta el porcentaje de batería actual. En la parte superior de la ventana se muestra la transmisión de video en tiempo real, debajo encontramos los botones que ejecutarán un comando de movimiento en el robot móvil. La sintaxis del comando es de la siguiente forma:

Comando: *"_modoDeOperacion_movimiento_"*

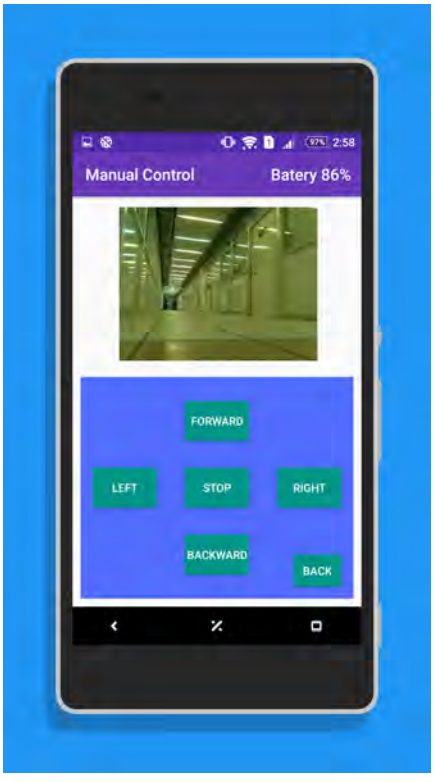


Figura 5.4: Actividad Manual Control

[35] Véase Apéndice G

Los botones que se presentan en la actividad son los siguientes:

- *Forward*. Este botón manda un comando de movimiento hacia adelante al módulo central de la siguiente forma:

  Comando: *"_1_forward_"*

- *Left*. Este botón manda un comando de giro hacia la izquierda al módulo central de la siguiente forma:

  Comando: *"_1_left_"*

- *Stop*. Este botón manda un comando de alto al módulo central de la siguiente forma:

  Comando: *"_1_stop_"*

- *Right*. Este botón manda un comando de giro hacia la derecha al módulo central de la siguiente forma:

  Comando: *"_1_right_"*

- *Backward*. Este botón manda un comando de movimiento hacia atrás al módulo central de la siguiente forma:

  Comando: *"_1_backward_"*

Cada comando es procesado por el módulo central de la Raspberry Pi 2, después se ejecuta la función correspondiente para llevar la tarea a cabo.

Para volver a la actividad *"Operation Mode"* se presiona el botón *Back* en la parte inferior derecha de la pantalla.

5.2.4. Actividad *Sensors*

La finalidad de esta actividad es mostrar los datos de los sensores ya procesados en la interfaz gráfica.

En la barra superior se muestra el nombre de la actividad *"Sensors"* seguido del porcentaje de batería actual. En la parte superior de la ventana se muestra el video en tiempo real proveniente de la Raspberry Pi Cam. Debajo se ubican etiquetas con los nombres de cada uno de los sensores seguidos de su valor actual. A continuación se describen estos sensores:

- Temperature. Se muestra el valor de la temperatura actual dada en grados centígrados (°C).

- Gas sensor. Se muestra la cantidad de gas dada en partes por millón (ppm).

- LDR. Se muestra la intensidad de luz en el ambiente dada en lux (lx)

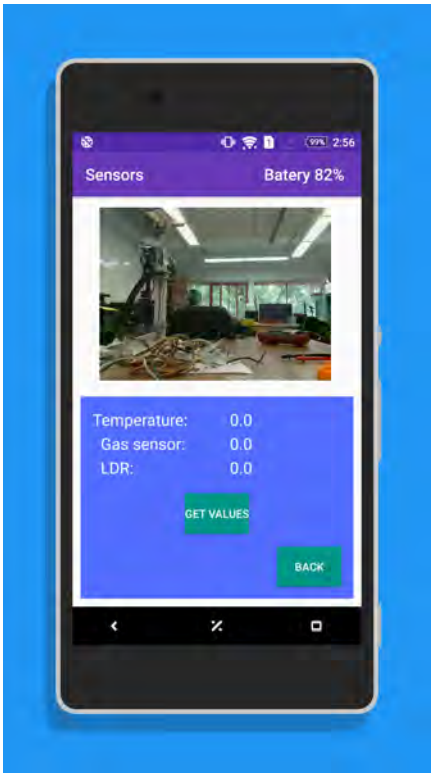


Figura 5.5: Actividad Sensors

Se deberá presionar el botón *Get Values* para que la aplicación Android haga la petición de información al módulo de sensores. Después se procesa y se muestra en la interfaz gráfica. Los datos serán actualizados cada medio segundo.

Al oprimir el botón *Get Values* la aplicación enviará un comando con la siguiente sintaxis:

Comando: *"_modoDeOperación_sensors_"*

Por ejemplo: *"_2_sensors_"*

Una vez recibido el comando por el módulo central, éste enviará la información con la siguiente sintaxis:

Comando: *"_modoDeOperación_sensor1_sensor2_sensor3_sensor4_"*

Por ejemplo: *"_2_182_24_39_47_"*

Esta cadena será particionada en la aplicación Android y posteriormente se mostrará cada valor en el campo correspondiente.

Para volver a la actividad *"Operation Mode"* se presiona el botón *Back* en la parte inferior derecha de la pantalla.

5.2.5. Actividad *Obstacle Avoidance*

El propósito de esta actividad es mostrar la navegación autónoma del robot móvil evadiendo obstáculos. La trayectoria recorrida por el robot móvil será mostrada en la interfaz gráfica en el video en tiempo real, al igual que el estado actual de la máquina de estados.

En la barra superior se muestra el nombre de la actividad *"Obstacle Avoidance"* seguido del porcentaje de batería actual. Debajo se ubica el video en tiempo real que transmite la Raspberry Pi Cam. Después se muestra la etiqueta *State:* y un cuadro de dialogo debajo, el cual indicará en que estado de la máquina de estados del algoritmo de "evasión de obstáculos" se encuentra el robot móvil.

Para comenzar con este modo de operación se debe oprimir el botón *Start* y automáticamente el robot comenzará a navegar evadiendo obstáculos. El comando que envía la aplicación móvil tendrá la siguiente sintaxis:

Comando: *"_modoDeOperación_obstacle_"*

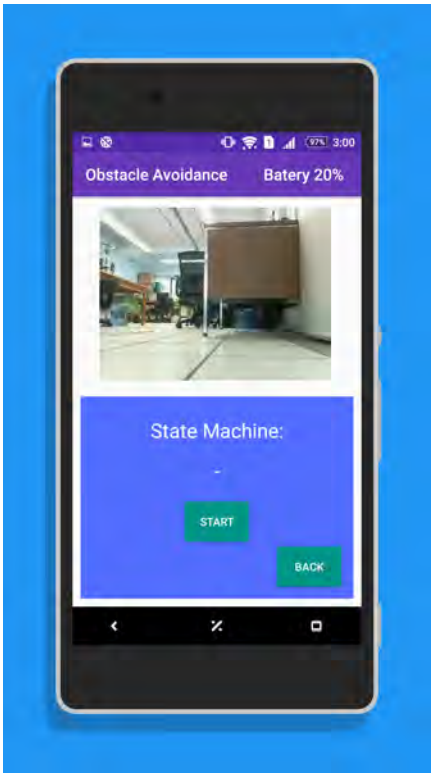


Figura 5.6: Actividad Obstacle Aviodance

El módulo central de la Raspberry Pi 2 recibe este comando y empezará a ejecutar la máquina de estados de "evasión de obstáculos". En cada estado, el módulo central enviará un comando a la aplicación Android notificando el estado actual de la siguiente manera:

Comando: *"_modoDeoperación_estadoActual_"*

Por ejemplo: *"_3_state3_"*

La aplicación Android recibe esta cadena, la particiona y coloca el dato en el campo correspondiente. El robot móvil estará navegando autónomamente hasta que el usuario presione el botón *Back*, donde se regresará a la actividad anterior *"Operation Mode"*.

5.2.6. Actividad *Search of a light source*

El propósito de esta actividad es mostrar la navegación autónoma del robot móvil evadiendo obstáculos y llegando a un destino final. La trayectoria recorrida por el robot móvil será mostrada en la interfaz gráfica en el video en tiempo real, al igual que el estado actual de la máquina de estados.

En la barra superior se muestra el nombre de la actividad *Light Source* seguido del porcentaje de batería actual. Debajo encontraremos el video en tiempo real que transmite la Raspberry Pi Cam. Al igual que en la actividad *Obstacle Avoidance* se muestra una etiqueta con la palabra *State:* y debajo un cuadro de dialogo que indicará el estado de la máquina de estados del algoritmo "Búsqueda de una fuente de luz" en que se encuentra. Una vez que el robot ha llegado a su destino se mostrará en el cuadro de diálogo la palabra *Goal*.

Para comenzar con el modo de operación *Light Source* se debe oprimir el botón *Start* y automáticamente el robot comenzará a navegar evadiendo obstáculos y buscando una fuente de luz. El comando que envía la aplicación Android tendrá la siguiente sintaxis:

Comando: *_modoDeOperación_light_*

El módulo central de la Raspberry Pi 2 recibe este comando y ejecutará la máquina de estados *"Búsqueda de una fuente de luz"*. En cada estado, el módulo central enviará un comando a la aplicación Android notificando el estado actual de la siguiente manera:

Comando: *_modoDeOperación_estadoActual_*

Por ejemplo: *"_4_state9_"*

La aplicación Android recibe esta cadena, la particiona y coloca el dato en el campo correspondiente. El robot estará navegando autónomamente hasta que encuentre una fuente de luz intensa, o hasta que el usuario presione el botón *Back*, donde se regresará a la actividad anterior *Operation Mode*.

5.3. Término de la aplicación

Para finalizar la sesión de uso del robot móvil se tendrá que regresar a la actividad *Operation mode*, en la parte inferior derecha se encontrará el botón *Exit*, éste tiene como propósito terminar todas las actividades y cerrar la aplicación.

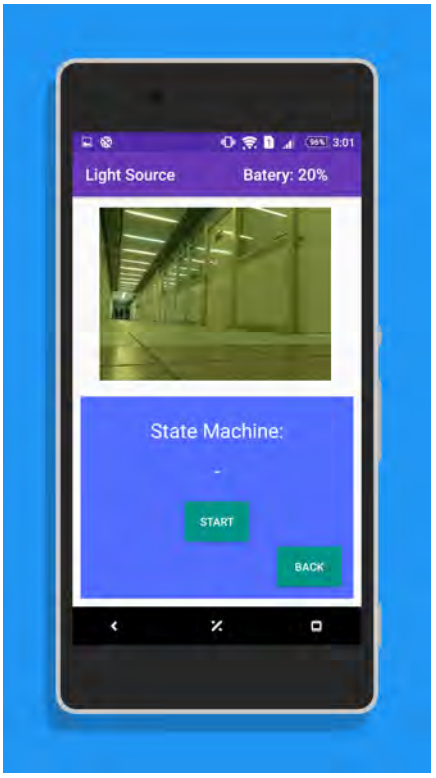


Figura 5.7: Actividad Light Source

Además de terminar la ejecución de todas las actividades, el botón *Exit* mandará un último comando al módulo central de la Raspberry Pi 2 que le indicará apagar el sistema. El comando tiene la siguiente sintaxis:

Comando: *_modoDeOperación_exit_*

Por ejemplo: *_5_exit_*

Después de oprimir este botón la aplicación terminará y se podrá apagar el robot móvil presionado el interruptor de energía.

CAPÍTULO 6

Pruebas y resultados

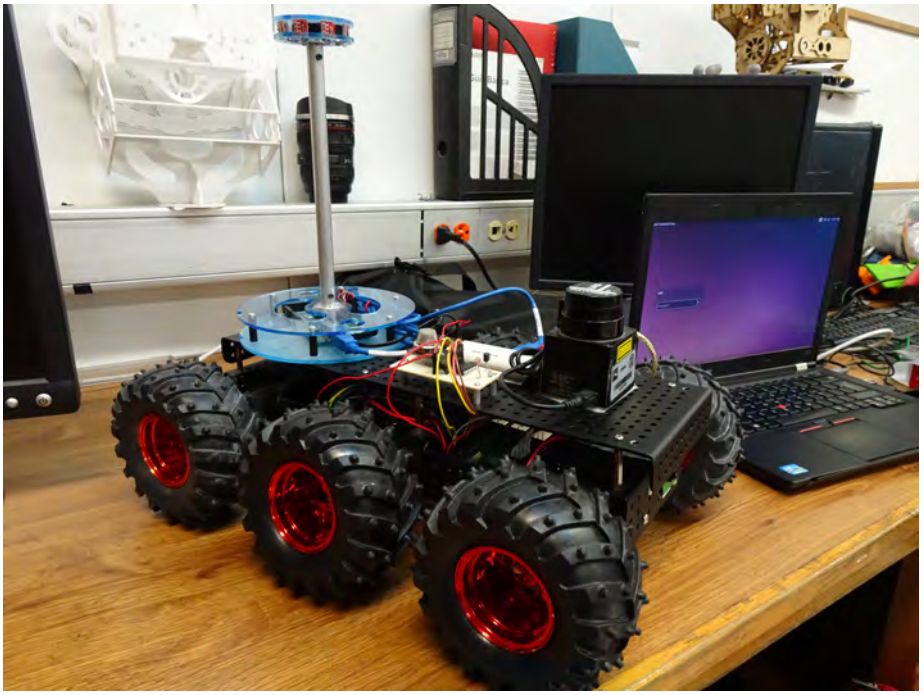


Figura 6.1: Prototipo final del robot móvil

En este capítulo se describen todas las pruebas realizadas en el prototipo final del robot móvil y en la última versión de la aplicación Android.

- Se comenzó con las pruebas de velocidad y ángulo de giro que tomó el robot móvil al navegar. Se estimó por tiempo de ejecución para la distancia recorrida y el ángulo de giro y se hicieron pruebas hasta llegar a un tiempo adecuado dadas las características del movimiento.
- Lectura de sensores. Se describen las pruebas que se hicieron sobre la toma de valores y el procesamiento de esta información.
- Detección de obstáculos. Se hicieron pruebas sobre la detección de obstáculos en las 3 direcciones de detección: al frente, a la izquierda y a la derecha.
- La transmisión de video. Se describen las pruebas realizadas sobre la transmisión de video en tiempo real de la Raspberry Pi Cam a la aplicación Android y a un navegador Web.
- Torre de sensores de luz. Se hicieron pruebas sobre la detección de luz en las 4 direcciones: al frente, a la izquierda, a la derecha y detrás del robot móvil.
- La conexión entre la Raspberry Pi 2 y la aplicación Android. Se hicieron pruebas sobre el funcionamiento de la conexión, es decir, si se perdía la conexión o si había mensajes que no llegaban al destinatario.
- Correo electrónico. Se describen las pruebas sobre el envío de correos electrónicos, tanto al iniciar el sistema como en la alerta de gas.
- Modos de operación. Se describen las pruebas sobre el funcionamiento de cada modo de operación, si se cumple el objetivo y qué errores se mostraron.

A continuación se describen a detalle las pruebas y los resultados obtenidos.

6.1. Distancia y ángulo de giro del robot móvil

Como se mencionó en el capítulo 5, la Raspberry Pi 2 actúa como el módulo central que controla diversos dispositivos integrados al robot móvil. La tarjeta Pololu es controlada directamente por la Raspberry Pi 2, ésta dará la dirección y velocidad de los motores en la navegación del robot móvil.

Los motores no tienen integrados *encoders* (codificadores rotatorios), estos son componentes utilizados para adquirir la posición, velocidad y aceleración de un motor, por esta razón no se puede saber con exactitud si la distancia recorrida por el robot móvil es la distancia que se le indicó recorrer en su trayectoria.

Como alternativa se optó por ejecutar comandos de movimiento por un cierto tiempo para alcanzar la distancia deseada. Se estimó este tiempo para varios valores de distancia y varios ángulos de giro. Este tiempo será multiplicado por la distancia a recorrer dada en metros y el ángulo dado en grados centígrados. Teniendo un aproximado de este tiempo se hicieron 20 pruebas para cada distancia y se obtuvo un promedio de la distancia recorrida.

Para medir la distancia recorrida se desarrolló un programa en Python que toma la información del láser hokuyo para saber la posición inicial y la posición final del robot móvil respecto a un obstáculo. El programa toma la distancia inicial que hay entre el robot móvil y un obstáculo, posteriormente se le da un comando de movimiento por un cierto tiempo, después se toma la distancia final que hay entre el robot móvil y el obstáculo, por último se hace la resta entre la distancia final y la distancia inicial quedando como resultado la distancia recorrida por el robot móvil.

En la tabla 6.1 se muestran las distancias recorridas.

Tabla 6.1: Pruebas de distancia recorrida en el robot móvil.

Distancia (cm)	**10**	**-10**	**30**	**-30**	**50**	**-50**	**100**	**-100**
Tiempo estimado (s)	**2.55**	**2.55**	**1.62**	**1.62**	**1.4**	**1.4**	**1.17**	**1.17**
Prueba 1	9.8	-9.9	28.5	-30.2	52.9	-51.1	97.6	-98.5
Prueba 2	9.5	-11	31	-29.6	50.4	-49	98.8	-103.1
Prueba 3	10.1	-9.3	31.3	-30.4	53.5	-50.1	101.1	-98.8
Prueba 4	10.7	-10.2	29	-32.1	49.3	-51.2	100.4	-101.4
Prueba 5	10.3	-11.1	28.8	-31.1	47.9	-49.5	101.8	-101
Prueba 6	9.4	-9.8	30.5	-30	52.5	-50.2	99.1	-97.6
Prueba 7	11.1	-10.2	30.2	-28.7	52.7	-48.8	102.2	-99.4
Prueba 8	10.4	-9.9	31.3	-28.9	48.4	-49	102.5	-99.1
Prueba 9	9.5	-10.2	28.6	-29.6	54.8	-50.3	99.1	-100.8
Prueba 10	9.8	-10.9	31.6	-31.1	51.1	-51.8	96.5	-99.9
Prueba 11	9.9	-10.2	30	-30.3	49.8	-46.7	99.7	-99.6
Prueba 12	9.4	-10.2	29.6	-29.2	48.6	-49.6	99.7	-99.6
Prueba 13	9.8	-10.1	31.8	-27.8	50.7	-49.1	101.5	-98
Prueba 14	10.9	-9.6	29.7	-28	49.6	-47.4	100.4	-100
Prueba 15	9.9	-10.3	29.1	-29.9	49.5	-49	102.6	-102.1
Prueba 16	11.1	-10.4	28.2	-28.5	50.3	-50.5	102.8	-97.2
Prueba 17	10.8	-9.5	32.1	-30.3	50.8	-52.3	101.5	-98.1
Prueba 18	9.5	-10	28.6	-29.1	51.2	-49.2	104.2	-99.3
Prueba 19	11.1	-10.3	28.8	-30.9	49.8	-51.1	99.1	-103.9
Prueba 20	9.8	-10.6	31.1	-29.5	50.3	-51.7	98.5	-98.9
Promedio	**10.09**	**-10.13**	**29.99**	**-29.76**	**50.7**	**-49.88**	**100.63**	**-99.81**

Como se aprecia en la tabla 6.1, el tiempo elegido para la ejecución de cada función de distancia fue el apropiado dados los resultados.

En la tabla 5.2 se muestran los ángulos de giro.

Tabla 6.2: Pruebas de ángulo de giro en el robot móvil.

Ángulo °C	**45**	**-45**	**90**	**-90**	**180**	**-180**
Tiempo estimado (s)	**2.55**	**2.55**	**1.62**	**1.62**	**1.4**	**1.4**
Prueba 1	48.1	-40.7	98.6	-88	190.7	174.3
Prueba 2	40.7	-46.4	88.2	-93.8	172.5	-188.2
Prueba 3	47.4	-53.8	99.6	-97.3	170.2	-185.2
Prueba 4	39.3	-46.1	83.7	-88.4	178.4	-177.7
Prueba 5	43.3	-47.9	84.3	-87.8	177.9	-190.3
Prueba 6	46.1	-46.9	92.1	-95.5	190.7	-174.6
Prueba 7	44.1	-42.3	90.4	-98.3	188.3	-192.2
Prueba 8	51.1	-43.4	80.7	-87.7	176.9	-184.8
Prueba 9	40.6	-46.7	91.5	-86.4	178.9	-186.5
Prueba 10	42.9	-49.6	93.7	-96.4	198.4	-176.5
Prueba 11	51.9	-47.1	96.3	-87.9	186.5	-191.5
Prueba 12	46.6	-44.7	92.4	-83.2	174.8	-188.4
Prueba 13	43.9	-47.9	92.7	-85.4	171.6	-176.5
Prueba 14	50.4	-41.7	83.4	-93.7	183.2	-169.7
Prueba 15	49	-51.4	93.2	-84.4	184.7	-179.3
Prueba 16	47.1	-42.9	91	-95.6	182.5	-182.1
Prueba 17	48.3	-39.9	85.7	-86.8	169.9	-176.1
Prueba 18	47.1	-42.8	82.5	-93.5	174	-169.3
Prueba 19	40.1	-47.9	91.6	-91.2	180.2	-178.4
Prueba 20	42.3	-44.2	93.5	-96.4	173.9	-166.9
Promedio	**45.51**	**-45.72**	**90.25**	**-90.88**	**180.21**	**-180.42**

Como se aprecia en la tabla 6.2, el tiempo elegido para la ejecución de cada función de giro fue el apropiado dados los resultados.

6.1.1. Resultados

En la asignatura de "Robots Móviles", impartida por el Doctor Jesús Savage Carmona en la Facultad de Ingeniería de la UNAM, se utiliza un simulador de un robot móvil, éste se encarga de representar el comportamiento de un robot en un ambiente virtual. El robot móvil trata de llegar a un punto destino a través de comandos de movimiento (ángulo de giro y distancia).

El objetivo será representar la trayectoria obtenida en el simulador en un robot real y comparar los resultados.

Se creó un programa en Python que se comunicará con el robot móvil a través de *sockets*, este programa actuará como un cliente que se encargará de leer un archivo de texto creado por el simulador. En este archivo están los comandos de movimiento ejecutados por el robot

móvil virtual. Después se enviará cada comando al servidor para que sean ejecutados en el módulo central del robot móvil.

La sintaxis del comando es la siguiente:
mv ángulo_de_giro distancia

También se creó un programa en Python en la Raspberry Pi 2 que actuará como un servidor, este programa recibe la información del cliente, es decir, los comandos de movimiento y ejecuta una función determinada para cada caso.

Los alumnos de la materia de "Robots Móviles" utilizaron el simulador en una práctica y generaron el archivo de comandos de movimiento. Este archivo fue leído por el programa cliente en una laptop y el robot móvil ejecutó adecuadamente todos los comandos de movimiento.

Como resultado se obtuvo un buen movimiento en cada comando, el error en la distancia y ángulo de giro fue mínimo, causado en ocasiones por el terreno en el cual navegó el robot móvil, sin embargo, este error se fue sumando en cada ejecución y al finalizar el robot móvil no llegó a su punto de destino.

6.2. Detección de obstáculos

El láser fue instalado en la parte frontal del robot móvil y su función es detectar si hay obstáculos en tres direcciones: adelante, a la izquierda y a la derecha. Para esto se segmentan los valores del láser dependiendo del ángulo de muestreo.

6.2.1. Resultados

En las primeras pruebas usando el láser Hokuyo, el robot permaneció en reposo y solo se colocaron frente al él distintos objetos para que detectara en qué dirección estaban. En todas las pruebas los resultados fueron exitosos, se detectaron obstáculos(cuando los había) en las 3 direcciones programadas.

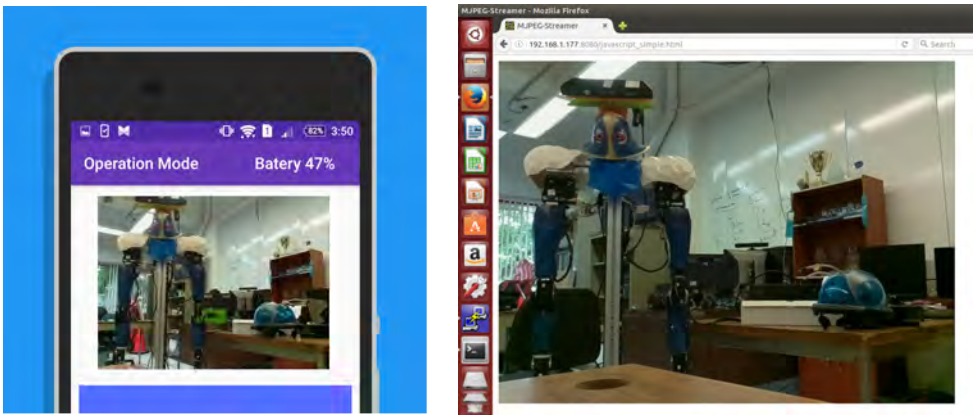
6.3. Raspberry Pi Cam y la transmisión de video

Se colocó la Raspberry Pi Cam en la parte frontal inferior del robot móvil, protegida por parte de la estructura metálica del chasís, para así evitar golpes o rayones sufridos por una mala navegación el robot móvil.

La herramienta MJPG-Streamer se instaló con éxito y se ejecutaron funciones para la transmisión de video y captura de imágenes, dando resultados satisfactorios. Posteriormente se conectó la Raspberry Pi 2 a la red local y se ejecutó la herramienta MJPG-Streamer para transmitir el video a un navegador web. En las primeras pruebas, el video fue transmitido con éxito a cualquier navegador web (ya sea en una laptop o en un Smartphone), pero había ocasiones en las que había latencia en la recepción de fotogramas. Se corrigió este error cambiando el Router por mal funcionamiento. Las nuevas pruebas se realizaron con éxito, el

video fue transmitido en tiempo real sin retrasos. Después se verificó la transmisión de video en la interfaz de la aplicación Android dando resultados correctos.

En la figura 5.2 se muestran las imágenes de la transmisión de video tanto en el navegador web como en la aplicación Android.



(a) Aplicación Android (b) Navegador Web

Figura 6.2: Transmisión de video

6.4. Torre de sensores de luz

La torre de sensores de luz tiene como dispositivo controlador un Arduino Mega, éste se encarga de recopilar los datos que transmite cada sensor, procesarlos y obtener 4 valores que nos indicarán la dirección en que se encuentra la fuente de luz. Las pruebas se realizaron con una lámpara.

6.4.1. Resultados

La información es transmitida a la Raspberry Pi 2 por medio de una cadena de caracteres que, después de ser procesada, indica en qué dirección proviene la mayor intensidad de luz en el ambiente.

Para probar esta tarea se programó una función que hace girar al robot dependiendo de dónde se coloca la fuente luz. Las pruebas se realizaron moviendo una lámpara 15 veces en cada dirección alrededor del robot móvil, donde el robot debía girar hacia la lámpara.

Después de realizar todas las pruebas se concluye que en todos los casos (al frente, a la izquierda, a la derecha o por detrás), la lámpara fue detectada y el robot móvil giró hacia ésta. La recopilación y el tratamiento de datos se hizo de manera exitosa.

6.5. Aplicación Android y conexión con la Raspberry Pi Cam

La aplicación Android fue programada en en lenguaje de programación Java, su interfaz fue diseñada con colores **"Material Design"** para darle una apariencia llamativa y presentable. Todas las actividades fueron diseñadas para comunicarse con la Raspberry Pi 2 a través cadenas de caracteres: la aplicación envía comandos de control y elección de acciones al robot móvil, y recibe cadenas de caracteres de información que serán mostradas en la interfaz de la aplicación. En cinco de las seis actividades se muestra el video que transmite la Raspberry Pi Cam en tiempo real, de dimensiones agradables a la vista del usuario.

6.5.1. Resultados

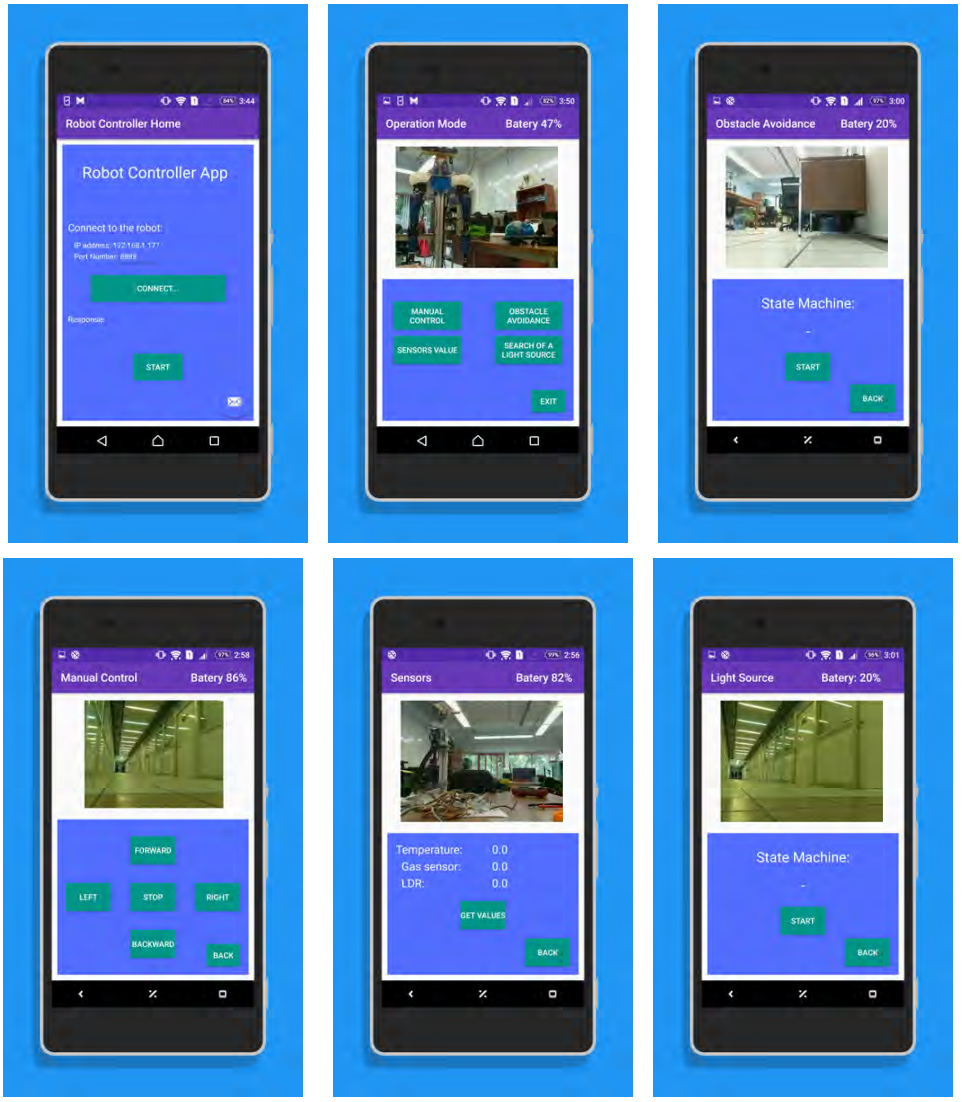


Figura 6.3: Actividades de la aplicación Android

En las primeras pruebas de conexión con la Raspberry Pi 2 hubo problemas en la transmisión de datos, en ocasiones alguno de los dos módulos (Raspberry Pi 2 o la aplicación Android) no procesaba bien los datos o no recibía los comandos esperados, por lo cuál se quedaban en espera. Estos errores se corrigieron mediante programación.

Otro problema que surgió al ir integrando más componentes al prototipo de robot móvil fue la alimentación eléctrica, en ocasiones dejó de funcionar el láser Hokuyo o la transmisión de video.

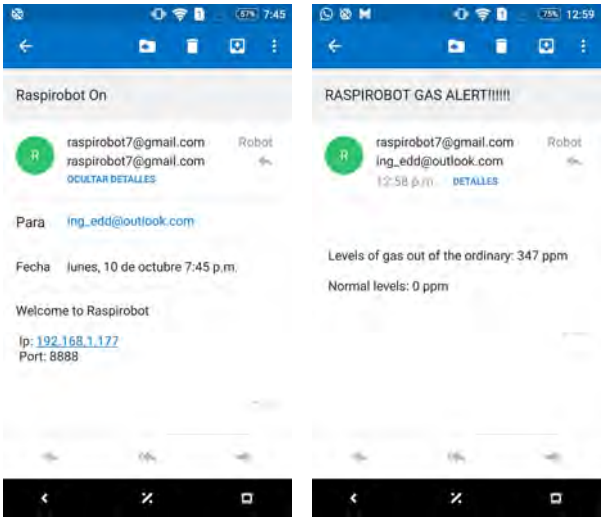
El arduino mega, los sensores, el láser hokuyo, el adaptador Wi-Fi y la Raspberry Pi Cam consumen mucha energía eléctrica en conjunto, por esta razón la Raspberry Pi 2 no podía alimentar eléctricamente a todos estos dispositivos. La solución fue colocar un *Hub USB* alimentado directamente por la batería. De esta manera todos los dispositivos funcionaron de manera correcta.

En las últimas pruebas, todos los intentos de conexión entre la Raspberry Pi 2 y la aplicación Android se hicieron con éxito. La transmisión de video se hizo sin problemas.

6.6. Correo electrónico y Scripts de Inicio

El módulo central de la Raspberry Pi 2 tiene como objetivo recibir las instrucciones de la aplicación Android y ejecutar la función que realice dicha acción. Sin este módulo no se podría iniciar una conexión. Por esta razón, cada vez que se inicie el sistema operativo de la Raspberry Pi 2 es necesario que se ejecute el módulo principal y las herramientas necesarias (como la herramienta MJPG-Streamer), de una manera automática. Esto se logra creando un *Script* que se ejecute junto con al arranque del sistema operativo de la Raspberry Pi 2.

Este *Script* ejecuta el módulo central de la Raspberry Pi 2, la herramienta MJPG-Streamer y envía un correo electrónico de inicio a un usuario *(figura 5.4 (a))*. Al iniciar el módulo central se empieza a sensar la cantidad de gas en el ambiente enviando un correo electrónico (de ser el caso) donde se advierte sobre niveles anormales de gas *(figura 5.4 (b))*.



(a) email inicio (b) email alerta de gas

Figura 6.4: Correos electrónicos

6.6.1. Resultados

Durante las pruebas, cada vez que se encendió el robot móvil se ejecutó correctamente el *Script*. Se inició correctamente el módulo central del robot móvil, al igual que la transmisión de video por la herramienta MJPG-Streamer y se enviarán correctamente tanto el correo electrónico de inicio como el correo electrónico de alerta de gas.

6.7. Modos de Operación

El robot móvil está programado para ejecutar cuatro modos de operación a través de la interfaz gráfica de la aplicación Android: Control Manual, Lectura de sensores, Evasión de obstáculos y Búsqueda de una fuente de luz.

A continuación se presentan las pruebas realizadas para cada modo de operación.

6.7.1. Modo Control Manual

Se colocó al robot móvil en el piso de un cuarto normal, estando ahí se encendió y se estableció la conexión con la aplicación Android, posteriormente se eligió el modo de operación *Manual Control* y se comenzó a controlar por toda el área libre.

Resultados

La transmisión de video en tiempo real se hizo sin retrasos y de una manera muy fluida. Al presionar el botón de cada acción se enviaron los comandos de movimiento y se ejecutaron en el robot móvil sin tener algún retraso significativo. El control de la navegación se hizo de manera sencilla.

En la figura 6.5 se presentan las imágenes que describen las pruebas del modo de operación *Manual Control*

(a) Aplicación Android (b) Pruebas

Figura 6.5: Modo de operación Manual Control

6.7.2. Lectura de sensores

Se diseñó el circuito de los sensores con la posibilidad de agregar más sensores para que completen la información adquirida del medio ambiente, por el momento se tienen 4 dispositivos que se están sensando continuamente: fotoresistencia, sensor de temperatura, sensor de gas y la batería. Este circuito se puede modificar para agregar 4 sensores más.

Todos los sensores reportan los datos al módulo central de la Raspberry Pi 2 y ésta se encarga de procesarlos y transmitirlos a la aplicación Android.

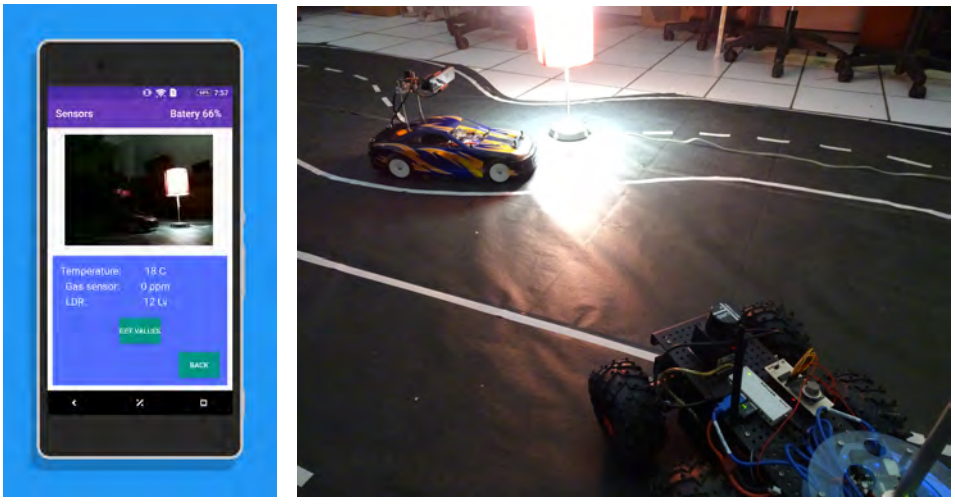
Resultados

El robot se encendió, después se estableció la conexión con la aplicación Android y se eligió el modo de operación *Sensors*. Para iniciar la lectura de información se presionó el botón de *Start*.

Se hicieron pruebas en los 3 sensores integrados al circuito de los sensores y en la batería:

- Fotoresistencia. Se varió la intensidad de luz que incidía en la superficie de la resistencia. Se apagó la luz y luego se expuso su superficie a una lámpara, posteriormente se tomaron los resultados.

  En la figura 6.6 se presentan las imágenes que describen estas pruebas.

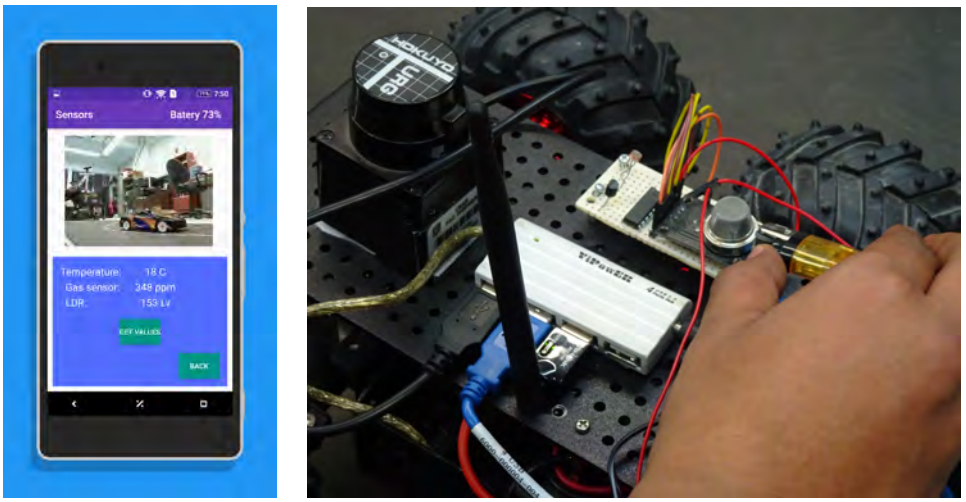


(a) Aplicación Android (b) Pruebas

Figura 6.6: Modo de operación Sensors con la luz apagada

- Sensor de temperatura. Se tomó la medición de la temperatura ambiente y después se aumentó colocando dos dedos alrededor del sensor, posteriormente se tomaron los resultados.
- Sensor de gas. Se hizo una medición de gas en el ambiente y después se varió utilizando el gas de un encendedor, después se tomaron los resultados.

En la figura 6.7 se presentan las imágenes que describen estas pruebas.



(a) Aplicación Android (b) Pruebas

Figura 6.7: Modo de operación Sensors aplicando gas de un encendedor

- Batería. Se tomaron los valores del porcentaje de batería actual que se veía en la interfaz de la aplicación durante la ejecución de ésta.

A continuación se muestran los datos de los sensores en la aplicación Android:
Dados estos resultados, el valor de los sensores se transmite de forma correcta y sin retardos, todos los sensores funcionaron correctamente.

6.7.3. Modo Evasión de obstáculos

Se colocó al robot móvil en el piso de un cuarto normal, rodeado de cajas y otros objetos que representaron obstáculos para su navegación. El robot se encendió, después se estableció la conexión con la aplicación móvil y se eligió el modo de operación *Obstacle Avoidance*. Para iniciar la máquina de estados de este modo de operación se oprimió el botón de *Start* y el robot comenzó a navegar.

Resultados

Se hicieron 10 pruebas con un límite de tiempo de un minuto. En la figura 6.8 se muestra la gráfica que representa los resultados de la navegación autónoma del robot móvil en este modo de operación.

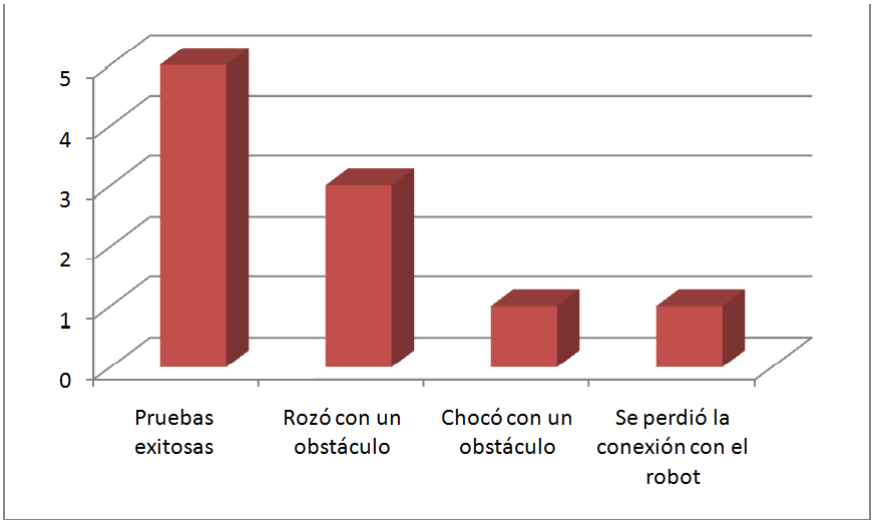


Figura 6.8: Resultados del modo de Evasión de obstáculos

De las 10 pruebas realizadas, 5 fueron exitosas. El robot móvil avanzó en linea recta cuando su camino estaba libre de obstáculos. Los obstáculos a la izquierda y a la derecha fueron detectados de manera correcta entre 10 y 20 centímetros. Los obstáculos al frente fueron detectados entre 20 y 35 cm del robot móvil. En 3 ocasiones el robot móvil pasó rosando un obstáculo o al retroceder terminó por empujarlo. En una ocasión chocó de frente con un obstáculo que no pudo detectar y terminó su funcionamiento. Y por último en una ocasión se perdió la conexión con el robot móvil, por lo cual no pudo terminar su tarea.

En aspectos generales el robot navegó de forma correcta, no hubo colisiones constantes y la transmisión de la información de la máquina de estados a la aplicación Android se hizo sin retrasos. De igual forma que la actividad anterior, la transmisión de video en tiempo real se hizo sin mucho retraso.

6.7.4. Modo Búsqueda de una fuente de luz

Se colocó al robot móvil en el piso de un cuarto normal, rodeado de cajas que representaron obstáculos y una lámpara que representó el punto destino de la navegación. El robot móvil se encendió, se estableció comunicación con la aplicación Android y se eligió el modo de operación *Search of a Light Source*. Después se presionó el botón de *Start* para iniciar la máquina de estados y el robot comenzó a navegar.

Resultados

Se hicieron 20 pruebas sin un límite de tiempo con la finalidad de que el robot llegara al punto de destino (una lámpara). En la figura 6.9 se muestra la gráfica que representa los resultados de la navegación autónoma del robot móvil en este modo de operación.

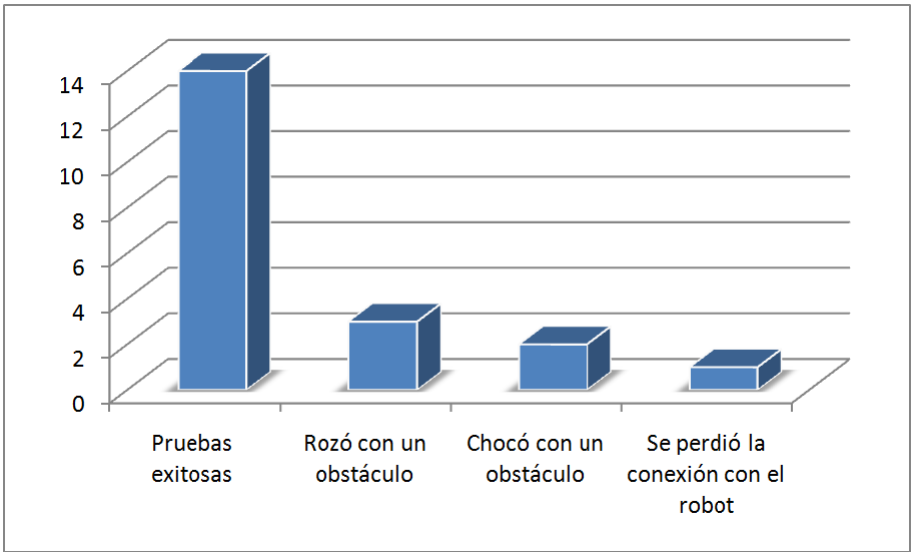


Figura 6.9: Resultados del modo Búsqueda de una fuente de luz

Catorce de las 20 pruebas se realizaron con éxito. Se detectó de manera correcta en qué dirección estaba la fuente de luz (lámpara) y se giró hacia ella. Al igual que la máquina de estados anterior, el robot avanzó en línea recta hacia la luz cuando no hubo obstáculos en su camino y pudo detectar éstos en las tres direcciones: a la izquierda, a la derecha y al frente. En 3 ocasiones rozó con un obstáculo al pasar por un lado o al retroceder. En 2 ocasiones chocó con un obstáculo y terminó su navegación. Por último en una ocasión se perdió la comunicación con la aplicación.

La transmisión de información a la aplicación Android se hizo de manera correcta y se llegó a la fuente de luz entre 10 a 20 cm del robot móvil. En todos los casos se indicó en la aplicación Android cuando se alcanzó el punto destino, la lámpara.

CAPÍTULO 7

Conclusiones y trabajo futuro

7.1. Conclusiones

Se construyó un prototipo de robot móvil controlado por una tarjeta Raspberry Pi 2, resistente a golpes, que puede navegar en espacios difíciles y pasar por encima de pequeños obstáculos. Esto con la finalidad de navegar por un hogar tanto en interiores como en exteriores. El módulo central (programado en la Raspberry Pi 2), se comunica a través de la red local con una aplicación Android donde se puede elegir el modo de operación del robot móvil y ver los datos más relevantes de los sensores integrados a éste.

Se integró la Raspberry Pi Cam ya que tiene un tamaño reducido y captura video en alta definición. La transmisión de video se hace en buena calidad y sin retrasos significativos tanto a la aplicación Android como al navegador web. De esta manera se cumplen los objetivos propuestos al inicio de la tesis.

Los modos de operación fueron ejecutados sin ningún problema. En el modo *Control Manual* se puede controlar al robot a través de comandos de movimiento de una manera sencilla. En el modo *Lectura de Sensores* se aprecia la información proveniente de los sensores integrados al robot móvil. En el modo de *Evasión de obstáculos* el robot navegó sin un destino evadiendo obstáculos de manera correcta. Y por último en el modo *Búsqueda de una fuente de luz* el robot navegó evadiendo obstáculos y llegando a un destino (una lámpara).

Por último, el robot móvil envía un correo electrónico a un usuario con la información necesaria para establecer la conexión y otro correo electrónico cuando el módulo de sensado de gas detecte un nivel anormal.

7.2. Trabajo futuro

Con el fin de mejorar el funcionamiento del robot móvil se puede realizar el siguiente trabajo futuro:

- Utilizar *ROS* (sigla en inglés de Robot Operating System) para controlar todos los módulos descritos anteriormente (tanto de la tarjeta Raspberry Pi 2 como de la aplicación Android), para proporcionar una mejor funcionamiento del sistema. Además se tiene la ventaja de poder utilizar varias herramientas desarrolladas para robots de este tipo.
- La navegación. En el siguiente prototipo se utilizarán *encoders* para obtener la distancia que exacta que se desplazó el robot móvil y poder corregir errores.
- Mapa del entorno. Se agregará un módulo de *ROS* capaz de crear un mapa del entorno a través del láser hokuyo, ésto con la finalidad de poder hacer una trayectoria óptima al momento de navegar de un punto a otro en una habitación.
- Sensores y actuadores. Se agregarán más módulos de sensado, como detectores de movimiento dentro del hogar, sensores en puertas y ventanas para conocer el estado éstas (abiertas o cerradas). También actuadores como los relevadores para prender lámparas, cerrar o abrir cortinas y para prender otros aparatos, como la cafetera. Todos estos serán controlados desde el módulo principal del robot móvil.
- Un servidor web. Se programará un servidor web para poder comunicar al robot móvil sin necesidad de estar conectado a una red local, es decir, para poder comunicarlo con cualquier conexión a internet.
- Estación de carga. Se agregará la funcionalidad de que el robot vaya a una estación de carga automáticamente cuando la batería esté baja.

Índice de figuras

Apéndices

APÉNDICE A

Código: Controlador del circuito de los sensores

Este código que tiene como función leer la información proveniente del convertidor analógico-digital (sensores) y transformarlos a sus unidades correspondientes. Posteriormente se envía esta información a la aplicación Android.

```
1  #Function to read ADC data channel
2  def ReadChannel(channel):
3          adc  = spi.xfer2([1,(8+channel)<<4,0])
4          data = ((adc[1]&3)<<8) + adc[2]
5          return data
6
7  A = 1000
8  B = 15
9  RC = 10
10 light_l=(1024-ReadChannel(0))*A*10/B*RC*(1024-ReadChannel(0)) #
       conversion to lux
11 temp_l=((ReadChannel(1)*330)/860) #conversion to celsius
12 gas_l=math.pow(10,((math.log(ReadChannel(2)/RC)-0.21)/-0.47)+2.3)
       #Conversion to ppm
13 comando = "_2_"+str(light_l)+"_Lv"+"_"+str(temp_l)+"_C"+"_"+str(
       gas_l)+"_ppm"+"_"+str(battery)+"\n"
14 conn.send(comando) #send data to android application
```

APÉNDICE B

Código: Controlador del láser Hokuyo

Este código tiene como propósito leer el bloque de información proporcionado por el láser hokuyo, decodificarlo y transformarlo a las unidades correspondientes. Esta función regresa el valor en milímetros de las distancias que leyó el láser.

```
1  def distance():
2          values_a = []  #Array that keeps values ASCCI
3          values_r = []   #Array that keeps values in millimeters
4          com = "MD0044072500001\n"  #Command requests values of
               laser
5          ser.write(com)
6
7          # Flags to partition the information into blocks of 3 bits
8          flag_1= flag_2 = 0
9          i=j=k=0  #Sum, Index bits
10         aux=distance=""
11
12         #Cycle storage blocks in values_a
13         while k<651:
14                 if flag_2==0:
15                         data = ser.readline()
16                         if data == "99b\n":
17                                 flag_1 = 1
18                         if flag_1 == 1:
19                                 i+=1
20                         if i==2:
21                                 flag_2=1
22                 elif flag_2==1:
23                         data = ser.read(3)
```

```
24                              j+=1
25                              if j==21:
26                                      data = ser.read()
27                                      aux=data
28                                      data = ser.read(4)
29                                      aux+=data
30                                      values_a.append(aux)
31                                      k+=1
32                              elif j==42:
33                                      data = ser.read(2)
34                              elif j==63:
35                                      data = ser.read(2)
36                                      j=0
37                              else:
38                                      values_a.append(data)
39                                      k+=1
40          #End data collection
41          # Conversion to millimeters"
42
43          for z in range (651):
44                  for y in range (3):
45                          data_bin=bin(int(hex(ord(values_a[z][y]))
                                ,16)-48)
46                          data_c = 6 - (len(data_bin)-2)
47                          data_l=2
48                          for x in range(6):
49                                  if x < data_c:
50                                          distance+='0'
51                                  else:
52                                          distance+=data_bin[data_l]
53                                          data_l+=1
54                  values_r.append(int(distance,2))
55                  distance=""
56
57          return values_r #Returns array in millimeters
```

APÉNDICE C

Código: Controlador de los motores

Este código tiene como propósito establecer la velocidad y la dirección a los motores del robot móvil para cada dirección (adelante, atrás, izquierda o derecha).

```
1  def forward():
2          vel=20
3          global pwm_l
4          global pwm_r
5          pwm_l.ChangeDutyCycle(0)
6          pwm_r.ChangeDutyCycle(0)
7          time.sleep(0.1)
8          GPIO.output(24,True)
9          GPIO.output(25,False)
10         GPIO.output(17,False)
11         GPIO.output(27,True)
12         pwm_l.ChangeDutyCycle(vel)
13         pwm_r.ChangeDutyCycle(vel)
14
15
16 def backward():
17         vel=20
18         global pwm_l
19         global pwm_r
20         pwm_l.ChangeDutyCycle(0)
21         pwm_r.ChangeDutyCycle(0)
22         time.sleep(0.1)
23         GPIO.output(24,False)
24         GPIO.output(25,True)
25         GPIO.output(17,True)
26         GPIO.output(27,False)
27         pwm_l.ChangeDutyCycle(vel)
28         pwm_r.ChangeDutyCycle(vel)
29
30
31 def left():
```

```
32          vel=30
33          global pwm_l
34          global pwm_r
35          pwm_l.ChangeDutyCycle(0)
36          pwm_r.ChangeDutyCycle(0)
37          time.sleep(0.1)
38          GPIO.output(25,True)
39          GPIO.output(24,False)
40          GPIO.output(27,True)
41          GPIO.output(17,False)
42          pwm_l.ChangeDutyCycle(vel)
43          pwm_r.ChangeDutyCycle(vel)
44
45  def right():
46          vel=30
47          global pwm_l
48          global pwm_r
49          pwm_l.ChangeDutyCycle(0)
50          pwm_r.ChangeDutyCycle(0)
51          time.sleep(0.1)
52          GPIO.output(25,False)
53          GPIO.output(24,True)
54          GPIO.output(27,False)
55          GPIO.output(17,True)
56          pwm_l.ChangeDutyCycle(vel)
57          pwm_r.ChangeDutyCycle(vel)
58
59  def stop():
60          global pwm_l
61          global pwm_r
62          pwm_l.ChangeDutyCycle(0)
63          pwm_r.ChangeDutyCycle(0)
```

APÉNDICE D

Código: Controlador de la torre de sensores de luz

El objetivo de este código es leer la cadena de datos proveniente del Arduino, procesarla y determinar en qué dirección existe mayor intensidad de luz (al frente, atrás, izquierda o derecha).

```
1  def read_Light():
2          light_s.write('s\n')
3          aux = True
4          while (aux):
5
6                  if (light_s.read() == 'l'):
7                          a=light_s.readline().split()
8                          b = []
9                          b.append(int(a[0]))
10                         b.append(int(a[1]))
11                         b.append(int(a[2]))
12                         b.append(int(a[3]))
13                         d = []
14                         if (b[2] >= b[0] and b[2] >= b[1] and b[2] >= b[3]):
15                                 d.append(b[2])
16                                 return d
17                         elif (b[1] >= b[0] and b[1] >= b[2] and b[1] >= b[3]):
18                                 d.append(b[1])
19                                 return d
20                         elif (b[3] >= b[0] and b[3] >= b[1] and b[3] >= b[2]):
21                                 d.append(b[3])
22                                 return d
23                         elif (b[0] >= b[1] and b[0] >= b[2] and b[0] >= b[3]):
24                                 d.append(b[0])
25                                 return d
```

APÉNDICE E

Código: Correos electrónicos

Este código tiene como función enviar correos electrónicos a un usuario determinado. Al iniciar el robot móvil enviará un correo con la información de la conexión y, de igual manera, al detectar niveles anormales de gas en el ambiente.

```
1
2  def send_email(message):
3          fromaddr = 'raspirobot7@gmail.com'
4          toaddrs = 'ing_edd@outlook.com'
5          msg = "\r\n".join([
6                    "From: raspirobot7@gmail.com",
7                    "To: ing_edd@outlook.com",
8                    "Subject: RASPIROBOT GAS ALERT!!!!!!",
9                    "",
10                   message
11                   ])
12
13         username = 'raspirobot7@gmail.com'
14         password = 'raspi-robot'
15         server = smtplib.SMTP('smtp.gmail.com:587')
16         server.starttls()
17         server.login(username,password)
18         server.sendmail(fromaddr, toaddrs, msg)
19         server.quit()
20
21 if (gas_l > 100):
22                   message="\n\n Levels of gas out of the ordinary: "+ str(
                         gas_level)+" ppm\n\n Normal levels: 0 ppm"
23                   send_email(message)
```

APÉNDICE F

Código: Actividad principal de la aplicación Android

Este código representa la programación de la actividad principal de la aplicación Android. Ésta realiza la conexión con la Raspberry Pi 2, para posteriormente, elegir el modo de operación a ejecutar.

```
1  protected void onCreate(Bundle savedInstanceState) {
2
3  super.onCreate(savedInstanceState);
4  setContentView(R.layout.activity_main);
5  mToolbar = (Toolbar) findViewById(R.id.toolbar);
6  TextAddress = (EditText) findViewById(R.id.textAddress);
7  TextPort = (EditText) findViewById(R.id.textPort);
8  b_Connect = (Button) findViewById(R.id.bConnect);
9  b_Start = (Button) findViewById(R.id.bStart);
10 response = (TextView) findViewById(R.id.responseTextView);
11
12 setSupportActionBar(mToolbar);
13 getSupportActionBar().setTitle("Robot Controller Home");
14 getSupportActionBar().setDisplayShowHomeEnabled(true);
15 b_Start.setEnabled(false);
16 b_Connect.setOnClickListener(new View.OnClickListener() {
17 @Override
18 public void onClick(View arg0) {
19 Robot_Connection myRobot = new Robot_Connection(TextAddress.getText()
20       .toString(), Integer.parseInt(TextPort
21       .getText().toString()), response);
22       myRobot.execute();
23       }
24 });}
25 public void start(View view) {
26     Intent i = new Intent(this, operation_mode.class);
27     i.putExtra("ip_address", TextAddress.getText().toString());
28     i.putExtra("port", Integer.parseInt(TextPort.getText().toString()));
29     i.putExtra("response", response.getText().toString());
30     i.putExtra("battery", battery);
```

```
31      startActivity(i);
32      finish();
33      }
34
35  public class Robot_Connection extends AsyncTask<Void, Void, Void> {
36      String dstAddress, response = "", data="";
37      int dstPort;
38      TextView textResponse;
39      Socket rasp_robot;
40      DataOutputStream robot_write;
41      Robot_Connection(String addr, int port, TextView textResponse) {
42      dstAddress = addr;
43      dstPort = port;
44      this.textResponse = textResponse;
45          }
46      @Override
47      protected Void doInBackground(Void... arg0) {
48      try {
49           rasp_robot = new Socket(dstAddress, dstPort);
50           DataInputStream robot_read = new DataInputStream(rasp_robot.
                 getInputStream());
51      if (rasp_robot.isConnected()){
52           robot_write = new DataOutputStream(rasp_robot.getOutputStream());
53           response = "Connected" ;
54           activate_start = true;
55           robot_write.writeUTF("_Socket creado");
56           data = robot_read.readLine();
57           battery=data;
58          }
59       } catch (UnknownHostException e) {
60         e.printStackTrace();
61         response = "UnknownHostException: " + e.toString();
62       } catch (IOException e) {
63         e.printStackTrace();
64         response = "IOException: " + e.toString();
65       } finally {
66       try {
67          if(rasp_robot != null) {
68          Log.i("AsyncTask", "Cerrando sockets");
69          rasp_robot.close();
70          }
71      } catch (IOException e) {
72         e.printStackTrace();
73        }}
74      return null;
75          }
76  @Override
77  protected void onPostExecute(Void result) {
78  textResponse.setText(response);
79  if (activate_start == true) {
80     b_Start.setEnabled(true);
81     }
82  super.onPostExecute(result);
83     }}}
```

APÉNDICE G

Código: Actividad Modo de operación

Este código tiene como función dar a elegir a un usuario el modo de operación a ejecutar. De igual manera se da una acción para cada botón en la actividad.

```
1  public class operation_mode extends ActionBarActivity {
2      TextView response;
3      private Toolbar mToolbar;
4      boolean exit = false;
5      String battery="20";
6      @Override
7      protected void onCreate(Bundle savedInstanceState) {
8          String url;
9          final Bundle bundle = getIntent().getExtras();
10         super.onCreate(savedInstanceState);
11         setContentView(R.layout.operation_mode);
12         url= "http://"+(bundle.getString("ip_address"))+":8080/
               javascript_simple.html";
13         mToolbar = (Toolbar) findViewById(R.id.toolbar);
14         setSupportActionBar(mToolbar);
15         getSupportActionBar().setTitle("Operation Mode Batery "+bundle.
               getString("batery")+"%");
16         getSupportActionBar().setDisplayShowHomeEnabled(true);
17         final WebView webView = (WebView) findViewById(R.id.webView);
18         int default_zoom_level = 135;
19         webView.setInitialScale(default_zoom_level);
20         webView.getSettings().setJavaScriptEnabled(true);
21         webView.loadUrl(url);
22         Robot_Connection myRobot = new Robot_Connection(bundle.getString("
               ip_address"), bundle.getInt("port"), response);
23         myRobot.execute();
24     }
25
26     public void exit(View view) {
27         exit = true;
28         final Bundle bundle = getIntent().getExtras();
```

```
29          Robot_Connection myRobot = new Robot_Connection(bundle.getString("
                ip_address"), bundle.getInt("port"), response);
30          myRobot.execute();
31          startActivity(new Intent(getBaseContext(), operation_mode.class)
32                  .addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP | Intent.
                        FLAG_ACTIVITY_SINGLE_TOP));
33          finish();
34      }
35
36      public void manually_c(View view) {
37          final Bundle bundle = getIntent().getExtras();
38          Intent i = new Intent(this, robot_control.class);
39          i.putExtra("ip_address", bundle.getString("ip_address"));
40          i.putExtra("port", bundle.getInt("port"));
41          i.putExtra("battery", battery);
42          startActivity(i);
43          finish();
44      }
45
46      public void obstacle_mode(View view) {
47          final Bundle bundle = getIntent().getExtras();
48          Intent i = new Intent(this, Obstacle_Ad.class);
49          i.putExtra("ip_address", bundle.getString("ip_address"));
50          i.putExtra("port", bundle.getInt("port"));
51          i.putExtra("battery", battery);
52          startActivity(i);
53          finish();
54      }
55
56      public void sensors_op(View view) {
57          final Bundle bundle = getIntent().getExtras();
58          Intent i = new Intent(this, sensors.class);
59          i.putExtra("ip_address",bundle.getString("ip_address"));
60          i.putExtra("port", bundle.getInt("port"));
61          i.putExtra("battery", battery);
62          startActivity(i);
63          finish();
64      }
65
66      public void light_op(View view) {
67          final Bundle bundle = getIntent().getExtras();
68          Intent i = new Intent(this, light.class);
69          i.putExtra("ip_address",bundle.getString("ip_address"));
70          i.putExtra("port", bundle.getInt("port"));
71          i.putExtra("battery", battery);
72          startActivity(i);
73          finish();
74      }
75
76      public class Robot_Connection extends AsyncTask<Void, Void, Void> {
77          String dstAddress, response = "",data="";
78          int dstPort;
79          TextView textResponse;
80          Socket rasp_robot;
81          DataOutputStream robot_write;
```

```
82          Robot_Connection(String addr, int port, TextView textResponse) {
83              dstAddress = addr;
84              dstPort = port;
85              this.textResponse = textResponse;
86          }
87          @Override
88          protected Void doInBackground(Void... arg0) {
89             try {
90                  rasp_robot = new Socket(dstAddress, dstPort);
91                  robot_write = new DataOutputStream(rasp_robot.getOutputStream
                        ());
92                  DataInputStream robot_read = new DataInputStream(rasp_robot.
                        getInputStream());
93                  if (rasp_robot.isConnected()) {
94                      robot_write = new DataOutputStream(rasp_robot.
                            getOutputStream());
95                      if(exit == true){
96                          robot_write.writeUTF("_5_Exit");
97                          Log.i("AsyncTask", "Exit");
98                      }
99                      else {
100                         robot_write.writeUTF("_0_Stop");
101                         data = robot_read.readLine();
102                         batery=data;
103                         Log.i("AsyncTask", "Alto");
104                     }}
105             } catch (UnknownHostException e) {
106                 e.printStackTrace();
107                 response = "UnknownHostException: " + e.toString();
108             } catch (IOException e) {
109                 e.printStackTrace();
110                 response = "IOException: " + e.toString();
111             } finally {
112                 try {
113                     if (rasp_robot != null) {
114                         Log.i("AsyncTask", "Cerrando sockets");
115                         rasp_robot.close();
116                         robot_write.close();
117                     }
118                 } catch (IOException e) {
119                     e.printStackTrace();
120                 }}
121             return null;
122         }}}
```

Bibliografía

[1] Aníbal Ollero Baturone. *Robótica: Manipuladores y robots móviles*. John Wiley and Sons, 1a edition, 2001.

[2] Marko Gargenta and Masumi Nakamura. *Learning Android*. O'Reilly Media, 2a edition, 2014.

[3] Spyros G. Tzafestas. *Introduction to mobile robot control*. Elsevier, 1a edition, 2013.

[4] Roland Siegwart and Illah R. Nourbakhsh. *Introduction to Autonomous Mobile Robots*. The MIT Press, 1a edition, 2004.

[5] Lee Wei-Meng. *Android 4: Desarrollo de Aplicaciones*. Anaya Multimedia, 1a edition, 2012.

[6] Raspberry Pi. Raspberry pi documentation. `https://www.raspberrypi.org/documentation/`, 2016. (Specifications).

[7] Raspberry Pi. Camera module. `https://www.raspberrypi.org/products/camera-module/`, 2016. (Specifications).

[8] Analog Devices. Low voltage temperature sensors (tmp36). `http://pdf.datasheetcatalog.com/datasheet/analogdevices/32847740TMP3567c.pdf`, 2002. (Datasheet).

[9] Hector Torres. Sensor de gas (mq2). `http://hetpro-store.com/TUTORIALES/sensor-de-gas-mq2/`, 2014. Visitada: 11-9-2016.

[10] Hanwei Electronics. Mq-2 gas sensor. `http://www.platan.ru/pdf/datasheets/winsensor/mq-2.pdf`, 2003. (Datasheet).

[11] Hokuyo. Scanning laser range finder urg-04lx-ug01. `http://www.hokuyoaut.jp/02sensor/07scanner/download/pdf/URG-04LXUG01specen.pdf`, 2009. (Specifications).

[12] Pololu. Step-down voltage regulator d15v35f5s3. https://www.pololu.com/product/2110, 2014. (Specifications).

[13] Pololu. Dual vnh5019 motor driver shield for arduino. https://www.pololu.com/product/2502, 2014. (Specifications).

[14] Microchip. 2.7v 4-channel/8-channel 10-bit a/d converters with spi serial interface. https://cdn-shop.adafruit.com/datasheets/MCP3008.pdf, 2008. (Datasheet).

[15] Pololu. Dagu wild thumper 6wd all-terrain chassis, silver, 75:1. https://www.pololu.com/product/1561, 2014. (Specifications).

[16] Magnus Lie Hetland. *Beginning Python*. Apress, 2a edition, 2008.

[17] Hokuyo. Communication protocol specification for scip2.0 standard. http://www.hokuyo-aut.jp/02sensor/07scanner/download/pdf/URGSCIP20.pdf, 2006. (Specifications).

[18] Tom Stöveken. Mjpg-streamer. https://github.com/jacksonliam/mjpg-streamer, 2016. Visitada: 1-10-2016.

[19] Adam. Scripts. http://www.raspberry-projects.com/pi/pi-operating-systems/raspbian/scripts, 2012. Visitada: 12-9-2016.

[20] Juan David Mesa González. Usando sockets en java. https://www.programarya.com/Cursos/Java-Avanzado/Sockets, 2015. Visitada: 20-8-2016.

[21] Douglas Bell Mike Parr. *Java para estudiantes*. Pearson Education México, 1a edition, 2013.

[22] Ken Arnold, James Gosling, and David Holmes. *El lenguaje de programación Java*. Pearson Education Madrid, 1a edition, 2001.