INSTITUTO POLITÉCNICO NACIONAL

Escuela Superior de Ingeniería Mecánica y Eléctrica
Unidad Profesional Adolfo López Mateos Zacatenco

Departamento de Ingeniería en Control y Automatización

Desarrollo de un prototipo automático de un estacionamiento radial

T E S I S

Para obtener el título de

INGENIERO EN CONTROL Y AUTOMATIZACIÓN

PRESENTAN

José Manuel Juárez Arredondo
Miriam Citlali Sánchez Rodríguez

ASESORES

M. en C. Antonio Obregón Tenorio
M. en C. Mauricio Aarón Pérez Romero

MÉXICO, D. F. Diciembre 2013

INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE INGENIERÍA MECÁNICA Y ELÉCTRICA

UNIDAD PROFESIONAL "ADOLFO LÓPEZ MATEOS"

T E M A D E T E S I S

QUE PARA OBTENER EL TITULO DE INGENIERO EN CONTROL Y AUTOMATIZACIÓN

POR LA OPCIÓN DE TITULACIÓN TESIS COLECTIVA Y EXAMEN ORAL INDIVIDUAL

DEBERA(N) DESARROLLAR C. MIRIAM CITLALI SÁNCHEZ RODRÍGUEZ

C. JOSÉ MANUEL JUÁREZ ARREDONDO

"DESARROLLO DE UN PROTOTIPO AUTOMÁTICO DE UN ESTACIONAMIENTO RADIAL"

DESARROLLAR EL ACCIONAMIENTO AUTOMÁTICO DE UN ESTACIONAMIENTO RADIAL A TRAVÉS DE UN MICROCONTROLADOR ARDUINO, IMPLEMENTADO EN UN PROTOTIPO.

- **ESTADO DEL ARTE.**
- **MARCO TEÓRICO.**
- **DESARROLLO DE INGENIERÍA.**
- **DISEÑO DE LA LÓGICA DE CONTROL.**
- **PRUEBAS Y RESULTADOS**

MÉXICO D. F., A 23 DE ENERO DE 2015.

A S E S O R E S

M. EN C. MAURICIO AARÓN PÉREZ ROMERO

M. EN C. ANTONIO OBREGÓN TENORIO

M. EN C. MIRIAM GÓMEZ ÁL
JEFE DEL DEPARTAMENTO ACADÉMICO DE
INGENIERÍA EN CONTROL Y AUTOMATIZACIÓN

Miriam Citlali Sánchez Rodríguez

A Dios
Agradezco infinitamente a Dios por haberme dado la oportunidad de alcanzar mis metas, de cuidarme siempre en el camino, llenarme de bendiciones, darme fuerzas para superar los obstáculos que se me han presentado y haberme rodeado siempre de personas tan buenas, en especial mis padres, hermano y abuelas.

A mis padres
Por apoyarme incondicionalmente en este camino, tanto en el aspecto moral como en el económico. Por ser mi ejemplo de trabajo duro, honradez, superación y amor. Por apoyar incondicionalmente sin ninguna responsabilidad ni esperando algo a cambio a Manuel y permitirle lograr este sueño también. Los amo y sin ustedes nada hubiera sido posible.

A mi hermano
Por ser una motivación en mi vida, y estar presente en mi mente en cada reto que se me presenta haciendo de cada obstáculo algo más sencillo de superar, siendo mi responsabilidad demostrarte que todo lo que uno se proponga con dedicación es posible. Te amo hermanito

A mi familia
Por apoyarme, estar presente siempre y ser felices con mi éxito, siendo todos, una motivación de superación. Todos son y serán importantes en mi vida, tíos, primos, sobrinos, abuelos, familia de Manuel Juárez, pero muy especialmente mi abuelita Marina y Carmen.

José Manuel Juárez Arredondo

A Dios
Por colmar mi vida de bendiciones en cada momento y poner buenas personas en mí camino.

A mis padres
Por guiarme, cuidarme y enseñarme los valores esenciales para afrontar la vida; porque sin su amor, apoyo y esfuerzo, este gran logro no hubiera sido posible. Los amo

A mis hermanas
Por ser alimento puro para mi alma y colmar mi vida de dicha y felicidad, al estar a mi lado durante mi formación profesional y personal.

A mi familia
Agradezco infinitamente a tíos, primos, sobrinos y muy en especial a mis abuelos, por ser todos, la motivación principal en mi vida, los cuales me han enseñado que nada es imposible.

A la familia Sánchez Rodríguez
Por haber creído en mí y de manera incondicional brindarme todo su apoyo moral y económico, les agradezco por formar parte primordial tanto de mi vida profesional como personal, haciendo estos caminos de lo más agradables y felices. Con este trabajo, expreso mi gratitud, respeto y amor que les tengo.

Miriam Sánchez y Manuel Juárez

Al Instituto Politécnico Nacional
Por ser nuestra segunda casa y brindarnos todas las herramientas intelectuales necesarias para lograr este sueño. Gracias por formarnos como profesionistas y personas de bien que contribuirán a la sociedad de México. Ser parte de esta honorable institución es y será un orgullo.

A nuestros asesores
Por guiarnos y ayudarnos incondicionalmente no solamente en la realización de este trabajo, sino durante la carrera también, siendo ustedes de los mejores profesores que tuvimos el honor de tener durante todo este camino, dignos de admiración y respeto.

A nuestros amigos
Por estar presentes siempre que los necesitamos, por trabajar siempre juntos y hacer de este un camino más sencillo, ameno y divertido de lo que hubiese sido sin ellos. En especial queremos dedicar este agradecimiento a Mariano Cervantes, Alejandrina Rodríguez y Fernando Delgado. Igualmente queremos agradecer al Ing. Eduardo Castillo por haber formado parte de este trabajo que inició como proyecto en Electrónica, pero especialmente por enseñarnos con su ejemplo el amor y respeto hacia México y por ser una persona ejemplar a seguir que lucha por sus ideales.

Índice

Resumen

La presente tesis muestra la propuesta de un estacionamiento radial automático que propone dar las bases para el desarrollo de un estacionamiento de este tipo en lugares concurridos en México.

En este trabajo se detalla el diseño y construcción de un prototipo a escala para demostrar la funcionalidad del proyecto. Así mismo se implementa una comunicación inalámbrica que propone dar solución a los problemas generados por cableado en estructuras móviles.

Como primer punto se mencionan los estacionamientos robotizados actuales en el mundo, así como la tecnología utilizada para la automatización de los mismos. Después se hace referencia a la situación en México referente a los estacionamientos tradicionales, así como características y problemas generados por los mismos, y tomando como base esa información se desarrolla una propuesta de solución con un sistema para el accionamiento de los motores con comunicación inalámbrica.

Considerados los problemas a los que se requiere dar solución se diseña un estacionamiento a escala, y en este trabajo se muestra y desarrolla la idea conceptual, el diseño en una herramienta de CAD-CAE y posteriormente la construcción de dicho prototipo. Después se explica el funcionamiento y características requeridas por el estacionamiento propuesto, para que tomando como base esto se desarrolle la programación de los microcontroladores. Posteriormente se explican los criterios de selección del equipo a utilizar en el proyecto, tales como los microcontroladores, sensores, motores y módulos de comunicación.

Finalmente se muestran las pruebas y resultados obtenidos del prototipo, las conclusiones generadas con el desarrollo de este trabajo y los anexos donde además de contener las hojas de datos de los dispositivos del proyecto se muestra el código de programación de los microcontroladores.

Introducción

Los estacionamientos automáticos han estado presentes en el mundo desde mediados del siglo XX, y al menos en México no se les ha dado la importancia que estos representan.

El propósito de este trabajo es dar las bases para el desarrollo de un estacionamiento radial automático en el país, proponiendo la automatización y estructura móvil. El punto central de este trabajo de investigación es que el estacionamiento propuesto pueda ser implementado en lugares concurridos, teniendo la principal ventaja de ahorrar tiempo en el proceso de estacionarse y espacio terrestre. Por lo cual este trabajo se estructura de la siguiente manera.

En el Capítulo I se presenta el estado del arte correspondiente a estacionamientos automáticos, señalando los avances tecnológicos a lo largo de la historia y enunciando a las empresas líderes en el desarrollo de estas edificaciones. Con respecto al Capítulo II, se hace hincapié en la situación en México, dando antecedentes históricos que datan el desarrollo de estacionamientos en nuestro país. Posteriormente se muestran las bases de operación de los aparcamientos. Por último se comparan las actividades a realizar en el proceso de aparcamiento de un estacionamiento tradicional contra uno automático.

Durante el desarrollo de este trabajo nos referiremos a estacionamiento tradicional a un lote o edificación de aparcamiento no automático. Por lo que se refiere al Capítulo III se mencionan las características del equipo y software utilizados en el desarrollo de este trabajo, así como los aspectos que se consideraron para la selección del equipo implementado en el prototipo. De igual manera en este capítulo se especifica cómo se desarrolló el modelo propuesto.

En cuanto al capítulo IV se explica la forma en que se programaron los microcontroladores Leonardo y Mega 2560, el diagrama de flujo del proceso y la configuración de los módulos *XBee* para crear la comunicación inalámbrica entre ambos microcontroladores. En el capítulo V se muestran las pruebas y resultados experimentales obtenidos durante el desarrollo del prototipo. Finalmente se muestran las conclusiones obtenidas del trabajo, además de la bibliografía donde se enlistan las fuentes que sustentan la investigación, y anexos que están conformados por hojas de datos de equipo y normatividad de estacionamientos en México.

Relación de Figuras

Relación de Tablas

Objetivo

Desarrollar el accionamiento automático de un estacionamiento radial automático a través de un microcontrolador Arduino, implementado en un prototipo. Para lo anterior, es indispensable considerar los siguientes objetivos específicos:

- Crear una representación del estacionamiento propuesto en CAD-CAE.
- Seleccionar el equipo con el cual se desarrollará el prototipo.
- Desarrollar el prototipo de estacionamiento radial.
- Diseñar la lógica del programa para el accionamiento del prototipo.
- Establecer la comunicación inalámbrica entre dos microcontroladores Arduino, empleando el protocolo de comunicación *ZigBee*.
- Desarrollar la etapa de potencia para el accionamiento de motores.
- Realizar las pruebas de funcionamiento a los mecanismos y el tiempo empleado para cada acción.

Justificación

Con la realización de este trabajo se busca proponer un estacionamiento semi-automático para lugares concurridos, así como disminuir el espacio comúnmente utilizado por un aparcamiento tradicional, puesto que no se requieren elevadores, escaleras, pasillos ni rampas que comuniquen un nivel con otro del edificio. A causa de lo mencionado anteriormente se da a entender que ninguna persona ajena al sistema tendrá acceso al lugar de resguardo de los vehículos, razón por la cual el automóvil estará seguro y no sufrirá algún percance como robo, colisión o mal uso de la unidad.

Actualmente en México no se ha desarrollado algún estacionamiento radial automático, por consiguiente en este trabajo se dan las bases para el desarrollo de un proyecto de este tipo con ingeniería mexicana, para lo cual se desarrolla un prototipo en el que se demuestra el funcionamiento, mecanismos y tiempos aproximados para recibir un automóvil o devolver éste a su propietario.

Planteamiento del problema

La situación en el mundo es cada vez más complicada, ya que aproximadamente existen 7 125 000 000 habitantes en el planeta Tierra y el área habitable del mismo, es de aproximadamente 11 000 000 000 de hectáreas. Si se divide esta cifra se obtiene que a cada persona le corresponde cerca de 1.5 hectáreas donde poder vivir, guardar pertenencias y colocar la basura producida (INEGI, 2013).

Se puede determinar con esta aproximación de espacio disponible por ser humano, que cada vez se cuenta con menor espacio terrestre para vivir, sin mencionar que en algunos lugares hay mayor densidad de población que en otros. Por ejemplo el Distrito Federal tenía hasta el año 2010 una densidad poblacional de 5920.45 habitantes por kilómetro cuadrado, y comparando esta cifra con el estado de Baja California Sur que tiene aproximadamente 9 habitantes por kilómetro cuadrado, se nota que la diferencia es enorme (INEGI, 2013). Casos como el de la Ciudad de México hay en otras partes del mundo y es importante contribuir o dar una solución para tal problema.

Al hablar de espacio terrestre disponible no sólo se hace referencia al lugar para habitar, sino también para transportarse, y esto también es un grave problema para los países o ciudades con sobrepoblación. Tomando nuevamente como ejemplo a la Ciudad de México, hasta el año 2011 se tenían cerca de 4 252 089 automóviles registrados en circulación, esto quiere decir que aproximadamente por cada 100 habitantes había 41 automóviles (INEGI, 2013).

Como se ha mencionado, el espacio terrestre que queda es mínimo, y los aspectos preocupantes no solamente deben ser tener un lugar dónde habitar o espacio dónde transportarse, sino también la preservación del medio ambiente. Esto lleva a tomar conciencia y a actuar para dar soluciones prontas ante tal situación.

Debido a los problemas anteriormente mencionados, y a la importancia de los mismos, se propone el desarrollo de un prototipo de estacionamiento radial automático, para tomarse éste como base para el desarrollo de un proyecto a futuro.

Algunas de las principales problemáticas que presentan los estacionamientos tradicionales se muestran a continuación:

1. Gran espacio desperdiciado, ya que hay que considerar los pasillos y rampas que comunican un nivel con otro del edificio.
2. El personal que acomoda los vehículos no es capacitado para llevar a cabo esa tarea y ocasiona daños a los automóviles sin responsabilizarse de los mismos.
3. El establecimiento no se encuentra en condiciones de higiene, seguridad y comodidad para el usuario.
4. Cuando se encuentran ocupados todos los lugares autorizados de estacionamiento, no hay un anuncio que así lo indique en la entrada del establecimiento.
5. Se permite una entrada mayor de vehículos al número de cajones autorizados.
6. No se tiene conocimiento del estado físico en el que se encuentra el trabajador.
7. Utilizan los vehículos sin la autorización del propietario o poseedor.

Considerando lo anteriormente mencionado se desarrollará una propuesta de solución ante tales problemas.

Hipótesis

El estacionamiento propuesto tendrá una HMI, será automático y contará con señalización. Por consiguiente se obtendrán los siguientes resultados:

- Cuidado del medio ambiente, ya que ahorrará espacio terrestre al almacenar vehículos, energía eléctrica y se reducirá la contaminación generada por los automóviles.
- El tiempo de recepción y entrega de la unidad al propietario se reducirá.
- Seguridad.
- Se evitará el congestionamiento vial en la entrada/salida.

Nomenclatura

m	metro	m
s	segundo	s
A	ampere	A
mA	miliampere	mA
W	watt	W
V	volt	V
R	resistencia	Ω
m	masa	Kg
SV		Semáforo Verde
SR		Semáforo Rojo
DV		Cajón D Verde
DR		Cajón D Rojo
CV		Cajón C Verde
CR		Cajón C Rojo
BV		Cajón B Verde
BR		Cajón B Rojo
AV		Cajón A Verde
AR		Cajón A Rojo
N		Navegar
E		Ejecutar
CA		Cuarto de acceso
Seg		Seguro
Sen C		Sensor de cajón C
Sen D		Sensor de cajón D
Sen A		Sensor de cajón A
Sen B		Sensor de cajón B
RX2		Cajón X __
RX1		Cajón X ¬
RA2		Cajón A __
RA1		Cajón A ¬

RB2 .....Cajón B __
RB1 .....Cajón B ¬
RC2 .....Cajón C __
RC1 .....Cajón C ¬
RD2.....Cajón D __
RD1.....Cajón D ¬
LO .....Switch límite afuera
LI.....Switch límite dentro
CO_2.....Dióxido de carbono
NOx .....Óxidos de nitrógeno

Capítulo I Estado del Arte

I.1 Introducción

La finalidad de este capítulo es ofrecer un panorama general acerca de la temática esencial del presente trabajo, como lo son los estacionamientos automáticos. Se comenzará por definir las palabras estacionamiento y automático. Estacionamiento se define como "lugar o recinto reservado para estacionar vehículos" y automático como dicho de un mecanismo, "que funciona en todo o en parte por sí sólo" (Real Academia Española, 2001).

Partiendo de las definiciones anteriores, un estacionamiento automático puede describirse como "un sistema mecánico de almacenamiento de vehículos que funciona por sí sólo. El usuario deposita su vehículo en un punto de recepción / entrega y se va" (IPS, 2013).
El sistema resuelve automáticamente el aparcamiento y posteriormente la entrega del vehículo sin intervención del usuario (IPS, 2013).

I.2 Evolución de los estacionamientos automáticos

- 1850. Se dieron avances técnicos para el desarrollo de elevadores seguros de personas y cosas. Estos avances tecnológicos hicieron posible la construcción en altura por lo cual también hubo un aumento drástico de la densidad de ocupación de la ciudad.

  En consecuencia de lo anteriormente mencionado y al desencadenamiento de la revolución industrial el automóvil se convierte en el protagonista de nuestras ciudades, de ahí que exista la necesitad de aparcar los mismos (IPS, 2013).

- 1920. En este año se desarrollaron los primeros sistemas de estacionamiento mecanizado y estos se basaban en el paternóster[1], o noria[2] vertical, observe la Figura I.1, con esto se consigue que el apilamiento de los vehículos sea eficaz. Este tipo de sistemas ha sido utilizado en Asia y Europa desde entonces hasta nuestros días.

  Inicialmente este sistema se trató de máquinas desmontables y temporales, que simplemente servían para resolver la demanda de un lugar de una manera eficaz (IPS, 2013).

Figura I. 1 *Estacionamiento con elevador. Chicago 1925* ***(IPS, 2013).***

- 1930. En los años 30 se desarrollaron sistemas más sofisticados, ya que contaban con los elementos básicos de los estacionamientos automáticos de hoy en día, los cuales eran: cabinas de recepción, elevadores, robots y almacenes. La diferencia era prácticamente la operación del sistema, ya que el estacionamiento de esa época se manipulaba manualmente (IPS, 2013). La Figura I.2 muestra una maqueta que ejemplifica un aparcamiento de esa época.

[1] Ascensor que consiste en una serie de compartimentos enlazados sin puertas moviéndose continuamente en una cinta sin fin.
[2] Consistente en una gran rueda con asientos que gira verticalmente.

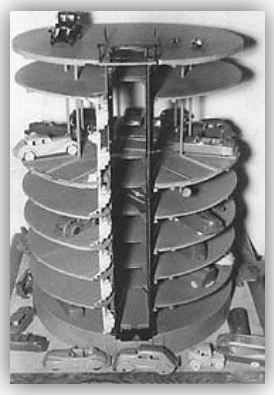


Figura I. 2 *Fernández. C. Estacionamiento Radial* ***(IPS, 2013)***

- 1940. A partir de los años 40 el aparcamiento automático se presentó como una innovación revolucionaria, ya que se desarrollaron las primeras unidades autónomas, de número variable de plantas, completamente mecanizadas y operadas por personal de servicio y mantenimiento (IPS, 2013). La Figura I.3 muestra una maqueta de un estacionamiento de esa época.

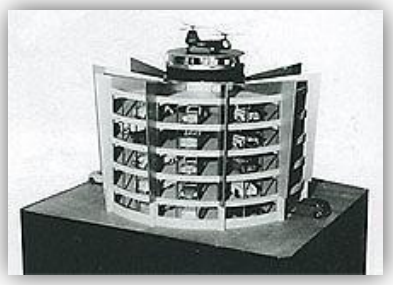


Figura I. 3 *Fernández. C. Estacionamiento Radial Automático* ***(IPS, 2013).***

- 1950. Los años 50 vieron surgir grandes unidades modulares de aparcamientos automáticos, estos fueron desarrollados principalmente en Estados Unidos y Europa. Se trataban de sistemas con capacidad de almacenar grandes volúmenes de vehículos, y estos aparcamientos eran orientados hacia el uso público en zonas comerciales de alta densidad de ocupación (IPS, 2013).

- 1960. En esta década los sistemas se desarrollaron según las necesidades y el entorno donde se implementaría la idea del aparcamiento. La Figura I.4 ilustra un estacionamiento desarrollado en esta década en Nueva York. Regularmente la automatización se vincula o se anuncia como sinónimo de innovación, velocidad y modernidad (IPS, 2013). Se observa un estacionamiento automático matricial.

Figura I. 4 *Perspectiva seccionada de la estación Speed-Park. Nueva York* ***(IPS, 2013).***

- 1970. Fue a partir de este año cuando se desarrollaron estacionamientos diversos, con distintas geometrías y principios mecánicos. Se empezó a desarrollar una nueva filosofía de implantación de sistemas de aparcamiento automático con diversidad de soluciones, y esto dependía de las necesidades a cubrir y de la imaginación en la aplicación. En esta década fue cuando la mecánica tradicional se complementó con sistemas electrónicos de gestión. La Figura I.5 muestra un tipo de estacionamiento desarrollado en los setentas. Se observa como la noria ya no solo era vertical como en un inicio (IPS, 2013).

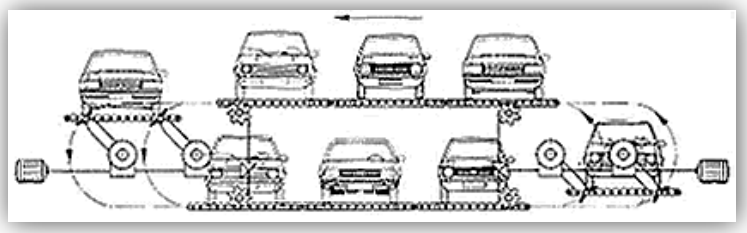


Figura I. 5 *Noria Horizontal* ***(IPS, 2013).***

La Figura I.6 muestra un estacionamiento basado en noria vertical, pero con doble plataforma.

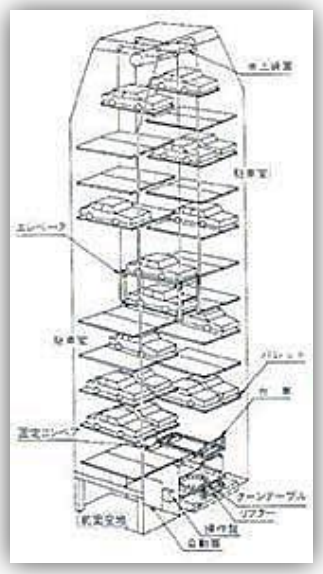


Figura I. 6 *Noria con plataformas dobles* ***(IPS, 2013).***

- 1980-1990. A partir de 1980 se mostró un avance conceptual muy considerable que apuntaba a los sistemas de aparcamiento de hoy en día, ya que como se muestra en la Figura I.7, el estacionamiento cuenta con una plataforma central que de manera automática recibe o entrega vehículos a los usuarios de estacionamiento (IPS, 2013).

Figura I. 7 *Planta de silo circular* ***(IPS, 2013).***

- 1990-2006. En estos años se vio la incorporación plena de la informática y la electrónica a la gestión totalmente automática de los sistemas.

  "Los más modernos sistemas de aparcamiento robotizado son completamente autónomos, fáciles y cómodos para el usuario, seguros y fiables para el operador" (IPS, 2013).

I.3 Estacionamientos automáticos actuales

I.3.1 Autostadt (Volkswagen)

El Autostadt es una atracción en el área alrededor de la fábrica de Volkswagen en Wolfsburg, Alemania. El sistema de aparcamiento Autostadt no es de uso público, ya que prácticamente se resguardan vehículos que esperan ser recogidos por sus compradores, y mientras tanto visitantes pueden admirar las unidades que se almacenan (Autostadt, 2013). El Autostadt tiene capacidad para almacenar 400 vehículos, cuenta con dos torres de almacenamiento y una de ellas se muestra en la Figura I.8. El mecanismo encargado de llevar y traer autos se encuentra en la parte central, se desplaza y gira sobre su propio eje y cuenta con 2 plataformas.

Figura I. 8** Autostadt **(Autostadt, 2013).

I.3.2 Perfect Park (Automated Parking Garage System)

El sistema de estacionamiento automatizado PerfectPark fue desarrollado por la ingeniería civil de renombre mundial y el grupo de la industria pesada, TREVI-Finanziaria Industriale (Trevi Group) Estados Unidos. Cada sistema de aparcamiento Perfect Park está instalado en un silo de hormigón y se basa en el método de apilamiento. Cada nivel tiene plazas de aparcamiento para 12 coches. Un máximo de 9 niveles con capacidad 108 vehículos (Perfect Park, 2013).

Los estacionamientos de cuentan con las siguientes características:

- Diseño cilíndrico.
- Hasta 108 coches por cada sistema.
- Hasta 9 niveles de 12 coches por nivel.
- Esencialmente un elevador de coches.
- Completamente automática y controlada por ordenador.
- Por encima o por debajo del suelo del diseño.
- Estructura circular soporta la actividad sísmica.
- Utiliza poca electricidad, puede ser alimentado por energías alternativas.

Como se visualiza en la Figura I.9 este sistema consta de un mecanismo que se ubica en la parte central del aparcamiento y solamente cuenta con una plataforma que se encarga de entregar y recibir vehículos.

Figura I. 9 *Corte transversal que muestra un estacionamiento de 6 niveles* ***(Perfect Park, 2013).***

I.3.3 Gesellschatfür Innovative Verkehrs Technologien mbH (GIVT) Sociedad para las tecnologías innovadoras de transporte mbH

Esta empresa alemana diseña y planifica sistemas de estacionamientos automáticos. En la Figura I.10 se muestra un ejemplo de estacionamiento automático subterráneo, el cual como se observa es matricial y tiene distintos puntos de acceso.

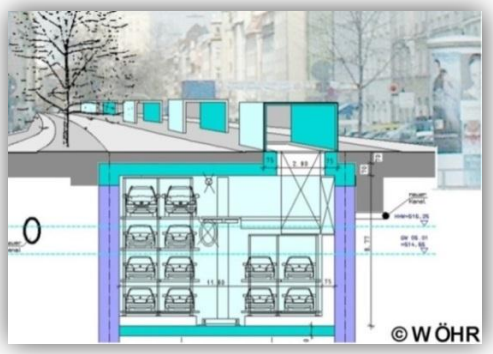


Figura I. 10 *Estacionamiento subterráneo automatizado* ***(GIVT, 2013).***

La Figura I.11 muestra las diferentes geometrías de aparcamiento disponibles, adaptándose al lugar donde se desea implementar el estacionamiento. De igual forma se aprecia en la imagen que se cuenta con una plataforma por aparcamiento encargada de llevar y entregar automóviles.

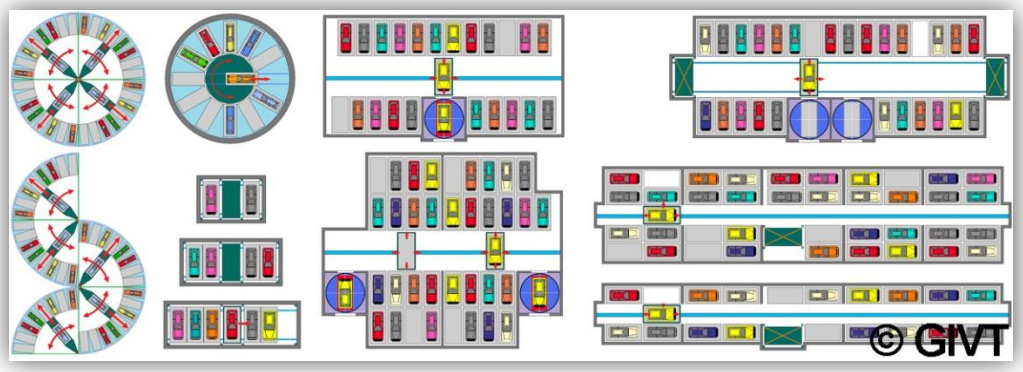


Figura I. 11 *Estacionamientos de GIVT* ***(GIVT, 2013).***

I.3.4 Robotic Parking Systems, Inc.

Esta empresa fue fundada en 1994 en EE.UU. y fue pionera en el desarrollo del estacionamiento automático de alta capacidad. Los aparcamientos desarrollados por Robotic Parking Systems son matriciales y cuentan con plataformas móviles en cada uno de sus cajones de resguardo, esto para con base en una combinación de posición encontrar un lugar para almacenar un vehículo o para llevarlo hasta la cabina donde el cliente podrá recoger su unidad.

La Figura I.12 muestra el interior de un estacionamiento automático matricial, y como se aprecia, cada nivel cuenta con su plataforma de distribución de automóviles, al igual que cada cajón tiene plataformas móviles, ya que como anteriormente se mencionó se debe encontrar la combinación de posición de los vehículos para almacenarlos o devolverlos a los clientes.

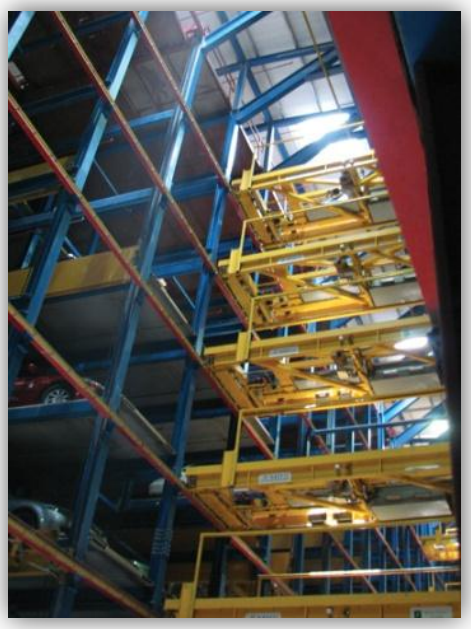


Figura I. 12 *Vista interior de estacionamiento* ***(Robotic Parking System, 2013).***

I.3.5 ALSE Mexicana. Estacionamientos Automatizados. Sistema de plataformas deslizables con elevadores

Empresa mexicana dedicada al desarrollo de estacionamientos matriciales automatizados. Estos estacionamientos permiten estacionar de 5 a 50 autos, en módulos que utilizan la mínima cantidad de espacio (ALSE Mexicana).

Las características de los aparcamientos desarrollados por esta empresa son las siguientes:

- De muy fácil operación.
- No se requiere un asistente en el estacionamiento.
- Rápida devolución del vehículo, generalmente en 60 segundos.
- Extremadamente seguro y confiable, con sensores de seguridad y barreras automatizadas opcionales.

En la Figura I.13 se muestra un ejemplo de estacionamiento diseñado por ALSE en Guadalajara, como anteriormente se mencionó es matricial, y al igual que los estacionamientos de Robotic Parking Systems se tiene que encontrar una combinación para que los automóviles de plantas superiores puedan llegar a planta baja.

Figura I. 13 *Estacionamiento de ALSE Mexicana* ***(ALSE Mexicana)***

I.3.6 Integral Park Systems (IPS)

IPS es una empresa Española en el sector de los aparcamientos robotizados y semiautomáticos, con más de 400.000 plazas instaladas en casi todo el mundo (IPS, 2013). En la Figura I.14 se ilustran aplicaciones que han tenido los estacionamientos de IPS.

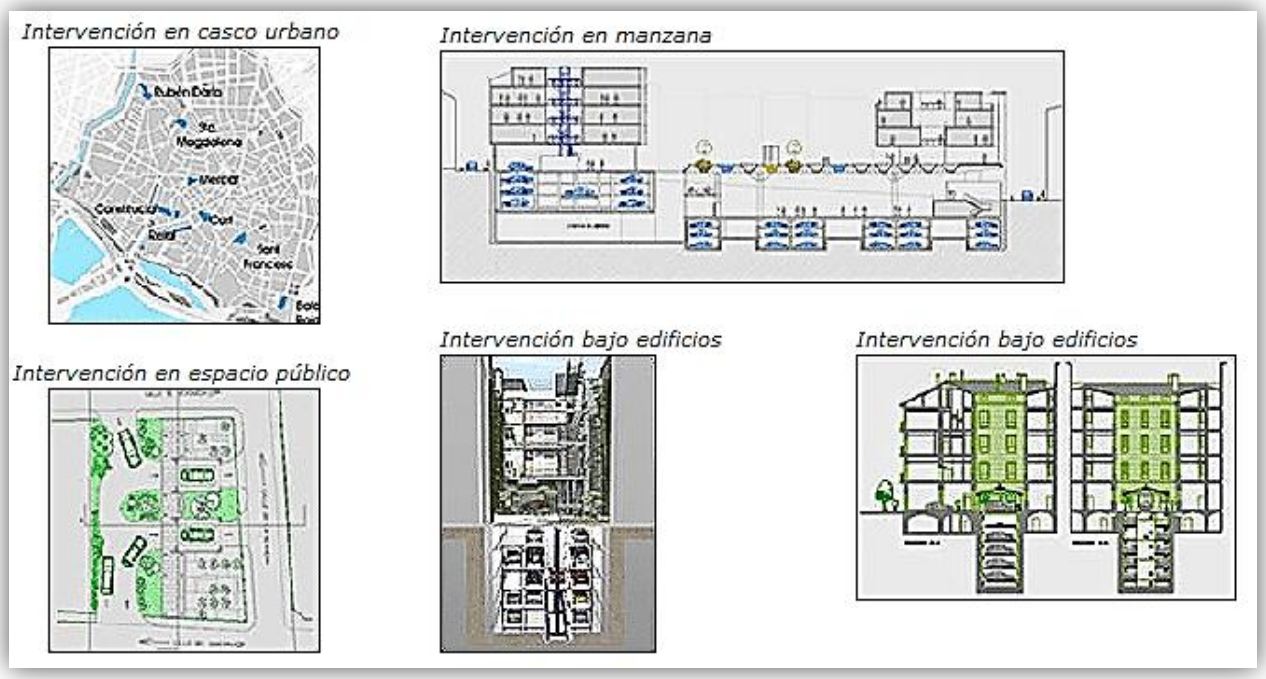


Figura I. 14 *Aplicaciones de estacionamiento robotizado IPS* ***(IPS, 2013).***

La Figura I.15 se muestran los países donde han sido instalados aparcamientos de IPS, y como se observa en México no se ha adquirido un estacionamiento de esta empresa, los factores que han intervenido no se mencionan, pero probablemente sea por el costo tan elevado de los mismos.

Figura I. 15 *Países donde IPS está presente* ***(IPS, 2013).***

Tomando como base la información anteriormente recopilada de las empresas líderes en el mundo en el desarrollo de este tipo de proyectos, se propone un diseño de estacionamiento radial, así como su accionamiento, superando a los estacionamientos actuales en México. En los capítulos siguientes se describirá detalladamente el diseño y funcionamiento del estacionamiento radial propuesto en este documento.

Capítulo II Marco teórico

II.1 Introducción

"Uno de los aspectos sobresalientes del siglo XX fue el fenómeno de la motorización de la vida del hombre en sus diversas actividades, especialmente en su transporte". El principal y más notable cambio en nuestras vidas fue la invasión de millones de vehículos de motor. Los automóviles se han convertido parte de nuestra vida diaria para diferentes actividades, como para ir a la escuela, ir a trabajar, ir de compras, transportar cosas, pasear, etc. "Los vehículos de motor son, quizás, el factor más importante en la conformación del trazo de nuestras ciudades, en la comunicación entre los hombres, en el comercio, en las actividades productivas y en nuestras costumbres" (Cal y Mayor, Estacionamiento, 1986).

Debido a que el ser humano adicionó el automóvil a su vida diaria, tuvo que construir caminos y modernizar la vialidad. Tiempo después se determinó que una buena parte del tiempo el vehículo requería de un lugar para ser guardado mientras su usuario desempeñaba sus actividades habituales, pero en un inicio no se prestó atención en algún lugar que sirviera para guardar los vehículos después de cada viaje. Al principio se empezó usando para este propósito la misma vía pública. Tiempo después se comenzó a notar que en el centro de la ciudad y a lo largo de calles importantes esto se tornaba conflictivo porque se necesitaba todo el espacio de la vía pública para la circulación de vehículos. Fue hasta entonces cuando se determinó que el ser humano debía proporcionar al vehículo de motor, un lugar de estacionamiento, también fuera de la vía pública, y así fue como nacieron los lotes y los edificios de estacionamiento (Cal y Mayor, Estacionamiento, 1986).

II.2 Delimitación geográfica del proyecto

El prototipo propuesto puede tomarse como base para el desarrollo de un proyecto de estacionamiento en cualquier parte del mundo, sin embargo este trabajo se enfocará en México, y específicamente al Distrito Federal. La razón principal por la cual se eligió a la Ciudad de México para desarrollo de un estacionamiento de estas características fue debido a que actualmente no existe una empresa mexicana dedicada al desarrollo de estacionamientos automáticos radiales capaz de almacenar más de 200 vehículos, y resultaría muy costoso si se contrata alguna empresa extranjera líder en el desarrollo de los mismos. Otra razón para tomar como base al Distrito Federal es debido a que es la Ciudad con mayor densidad poblacional en el país, y por lo tanto también es el lugar donde existe mayor concentración vehicular, así pues, con este trabajo se pretende dar solución a los problemas presentes en los estacionamientos que existen actualmente en la Ciudad de México.

La Figura II.1 muestra la delimitación geográfica de este trabajo de investigación.

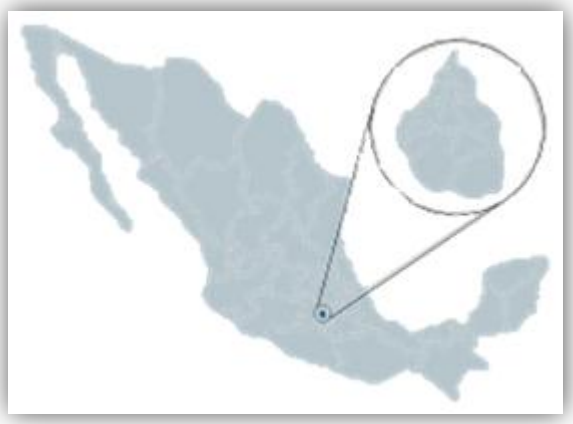


Figura II. 1 México, Distrito Federal

II.3 Situación en México

Hasta 1947 había registrados 63,368 vehículos de motor en la Ciudad de México. Seguramente esta cifra parecerá comparativamente insignificante ya que para 1983 el número de vehículos registrados casi llegó a los dos millones (Cal y Mayor, Estacionamiento, 1986).

En los años 40's la ciudad tenía muy pocos vehículos pero ya resultaba muy difícil transitar en ella debido al congestionamiento por la falta de buena vialidad y de espacios de estacionamientos fuera de la vía pública. Fue hasta el año de 1946 cuando se construyó el primer edificio en la Ciudad de México destinado a aparcar automóviles, ya que tan sólo algunos grandes edificios tenían sótanos destinados a ese fin, aunque siempre insuficientes. La reducción de la anchura de las calles debido al estacionamiento de vehículos en ellas produce bajas velocidades, congestión, pérdidas de tiempo, molestias incontables y principalmente, fuertes pérdidas económicas (Cal y Mayor, Estacionamiento, 1986).

El acelerado crecimiento poblacional y vehicular tomó por sorpresa y desprevenidos a quienes en algún momento pudieron haber tomados decisiones técnicas, económicas, legales y políticas. Estos problemas no solo están presentes actualmente en la Ciudad de México, sino en muchas partes del mundo. Como anteriormente se mencionó, el primer edificio de estacionamiento se construyó en 1946, pero el primer lote de servicio público destinado a aparcar automóviles en la Ciudad de México se habilitó en 1940. A mayor nivel económico corresponde a una menor relación de habitantes por vehículo. A mayor grado de motorización, habrá mayor demanda de estacionamientos. En la Ciudad de México se tenía una relación aproximada de 20 habitantes por vehículo, en 1960. De 10 habitantes por vehículo en 1970 y de 5 habitantes por vehículo en 1982. Esto último equivale a un vehículo de motor por cada familia (Cal y Mayor, Estacionamiento, 1986).

Un estudio realizado en el año 2011 por el Instituto de Estadística y Geografía determinó que los automóviles registrados en circulación en el Distrito Federal aproximadamente eran 4,252,089, por lo tanto esto corresponde a un promedio de 41 automóviles por cada 100 habitantes. En México hay aproximadamente 10,730,575 automóviles en circulación, esto quiere decir que en el Distrito Federal se encuentra el 39.62% del total de los vehículos del país (INEGI, 2013).

Esto no significa más que la demanda de espacios para estacionamientos se seguirá incrementando. Probablemente no en la misma proporción, pero sí habrá más demanda de espacios para guardar vehículos de todo tipo (Cal y Mayor, Estacionamiento, 1986).

Actualmente ya no se puede concebir la construcción de casas, departamentos, oficinas, comercios, fábricas, edificios de esparcimiento y otros que no dispongan de un lugar donde resguardar sus automóviles (Cal y Mayor, Estacionamiento, 1986). En 1949 se promulgó un Decreto en el Distrito Federal que disponía que "cada nuevo edificio que se construyera en el primer cuadro, con más de 5 pisos, debería proporcionar estacionamiento en relación con la superficie rentable adicional a los primeros cinco pisos en edificios de comercios, despachos y departamentos" (Cal y Mayor, Estacionamiento, 1986).

En la Ley sobre Estacionamientos de Vehículos en edificios y Construcciones Destinadas a Centros de Reunión, de 1953, se declaró de interés público el establecimiento de locales para estacionamiento. Se confirmó la obligatoriedad de construir estacionamientos en los edificios de más de 5 pisos, pero se estableció que podría salvarse dicha obligación con un impuesto "sustitutivo". Esto no evitó que la ciudad se llenara de edificaciones sin estacionamiento, siendo la mayoría de menos de 5 pisos. Por eso, de noche y de día se ven innumerables calles invadidas por vehículos que se estacionan frente a edificios de oficinas y de departamentos (Cal y Mayor, Estacionamiento, 1986).

"Para 1960 cuando la Ciudad de México registraba 248,000 vehículos de motor, contaba ya con 233 estacionamientos de servicio público, principalmente lotes, que correspondían a una creciente demanda de espacios para estacionar automóviles de gente que estaba dispuesta a pagar por ello. Sin embargo, el diseño de la mayor parte de los edificios era deficiente. Esta cifra subió a 376 estacionamientos de servicio público en 1970 y a 1219 en 1984. En este último año la ciudad ya registraba, 1,997,000 vehículos de motor. De estos estacionamientos 251 son edificios y 968 son lotes, con cupo conjunto de 78,119 y 140,622 espacios, respectivamente, lo que daba un total de 218,741 cajones disponibles, fuera de la calle" (Cal y Mayor, Estacionamiento, 1986).

Aunque el Reglamento de Construcciones para el Distrito Federal, del 9 de febrero de 1966, marcaba algunas normas básicas del proyecto para estacionamientos, se han seguido observando serias deficiencias en el diseño de nuevos edificios para este servicio. Cal y Mayor menciona en su libro que "revisando también los planes de estudio de algunas escuelas de Arquitectura e Ingeniería se vio que no se proporciona a los futuros profesionistas la necesaria enseñanza para los estudios y proyectos de estacionamientos en el país" (Cal y Mayor, Estacionamiento, 1986).

El 9 de marzo de 1973 fue promulgada una nueva Ley sobre Estacionamientos de Vehículos en el Distrito Federal, la cual exigía que cada nueva edificación proporcionara el estacionamiento que requiriese. Así mismo, el Departamento del Distrito Federal estableció una tabla de requisitos de espacios de estacionamiento para diferentes tipos de edificaciones en función del uso del suelo (Cal y Mayor, Estacionamiento, 1986).

En el año de 1977 el Departamento del D.F. creó un organismo descentralizado denominado Servicios Metropolitanos, S.A. de C.V. La finalidad que tiene este organismo es la de construir estacionamientos de servicio público para buscar un mejor equilibrio en la oferta y demanda de estacionamientos (Cal y Mayor, Estacionamiento, 1986).

En México se tiene el antecedente de un Decreto Presidencial, del 30 de agosto de 1949, que "requería que los nuevos edificios con más de cinco pisos deberían proveer estacionamiento con superficie equivalente al 30% del área rentable, en caso de comercios y despachos de los pisos arriba de altura indicada. En caso de viviendas se exigía tan sólo el 20%" (Cal y Mayor, Estacionamiento, 1986).

Por lo que respecta a edificaciones nuevas "reglamentaba el requisito de proveer estacionamiento en centro de reunión con cupo mayor de 500 personas. Además, establecía el posible pago de un impuesto sustitutivo para los casos en que no se proveyeran los estacionamientos requeridos". Hasta 1973 la ley sobre Estacionamientos de vehículos en el D.F. exigía que "toda edificación nueva o modificación en la ciudad tuviese el estacionamiento necesario según sus necesidades, independientemente de la altura o destino del edificio".

Dichas necesidades quedaron establecidas en la Tabla de Requisitos de Estacionamiento, para diferentes tipos de edificaciones y que el Departamento del D.F. publicó junto con la citada Ley. La tabla anteriormente mencionada fue modificada varias veces al año. Aunque en el año de 1982 la ley de Estacionamientos fue rechazada la tabla subsiste como requisito para los permisos de construcción (Cal y Mayor, Estacionamiento, 1986).

Investigaciones llevadas a cabo en la Ciudad de México, permitieron establecer que el automóvil particular solamente permanece en movimiento unas 3 horas al día, como máximo y que el resto del tiempo permanece estacionado cerca de donde su conductor desarrolla las actividades correspondientes (Cal y Mayor, Estacionamiento, 1986). En una ciudad los motivos de los viajes diarios son debidos, principalmente, a fines de trabajo, educación y comercio, y esto puede observarse en la Tabla II.1.

Tabla II- 1 *Motivos de viaje en automóvil en la Ciudad de México* ***(Cal y Mayor, Estacionamiento, 1986).***

MOTIVO	%
Trabajo	47.0
Educación	24.7
Compras	18.4
Recreación	1.9
No especificado	8.0
SUMA	100.0

Lo anterior significa dos cosas: 1°) que los automóviles están estacionados la mayor parte del tiempo. 2°) El estacionamiento principalmente se hace por motivos de trabajo (Cal y Mayor, Estacionamiento, 1986).

El autor Otto Sill menciona lo siguiente: "Las experiencias llevadas a cabo en todo el mundo indican que las zonas comerciales siempre faltan lugares de estacionamiento y que nunca hay de sobra…" (Cal y Mayor, Estacionamiento, 1986).

La situación que actualmente se presenta en el mundo ya no se presta para pensar en la vialidad que debe alojar a ese vehículo en su desplazamiento habitual, sin pensar en los lugares donde debe parar después de cada viaje (Cal y Mayor, Estacionamiento, 1986).

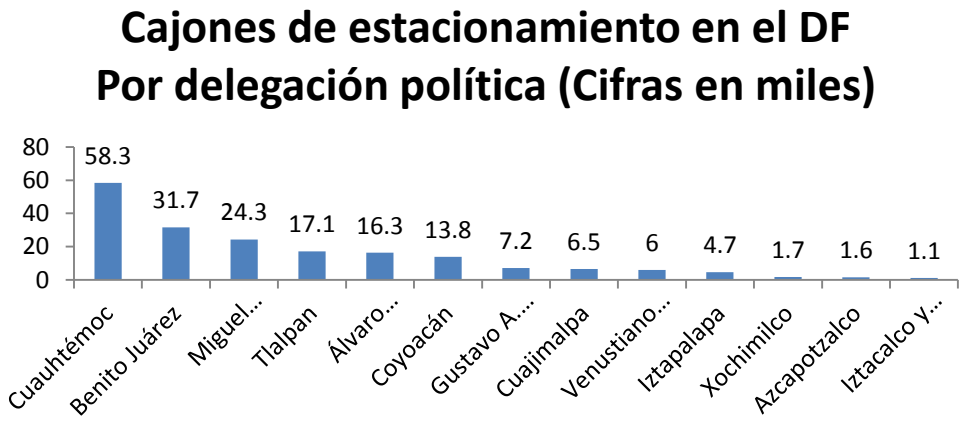
El propósito de la mayor parte de los viajes, como hemos visto, es el trabajo. Le sigue, en México, la educación y el comercio. Lógicamente esto influye en el tiempo que estará estacionado cada automóvil dependiendo la actividad a desarrollar de su conductor. También influye en el cumplimiento de las restricciones de estacionamiento en la vía pública. Tanto en trabajo como en educación el usuario viajará a su destino con o sin tener un lugar fijo donde estacionar su vehículo, ya que va hasta esos lugares por obligatoriedad o necesidad. Caso contrario al comercio, ya que el conductor para ese destino puede elegir el de su preferencia, y muchas veces su decisión se afectada por las alternativas de ubicación y el conductor tiene preferencia por comercios con facilidades de estacionamiento (Cal y Mayor, Estacionamiento, 1986).

Las consecuencias que se presentan a falta de estacionamientos son las siguientes:

- Limitaciones en el desarrollo económico
- Calles congestionadas
- Viajes innecesarios
- Clientes y negocios abandonan sitios de trabajo y comercio
- Deterioro de los valores de la propiedad y rentas

Los anteriores son sólo algunos problemas. Cabe una advertencia aconsejada por la práctica observada en el Distrito Federal. "Cada cajón de estacionamiento debe ser accesible individualmente, sin que sea necesario mover otro vehículo. Esto evitará muchas fricciones personales, molestias y pérdidas de tiempo, especialmente tratándose de estacionamientos para espectáculos, centros de reunión, departamentos, etcétera" (Cal y Mayor, Estacionamiento, 1986).

La Secretaría del Medio Ambiente del Distrito Federal, hablando específicamente de la ciudad de México menciona que "la flota vehicular registrada en la zona metropolitana del Valle de México se estima en más de 4.2 millones de vehículos, de los cuales 62% corresponde a unidades registradas en el Distrito Federal y 38% a unidades registradas en el Estado de México" (PROFECO, 2009). Las cantidades anteriormente mencionadas representan un gran problema, ya que la Ciudad de México cuenta con un número de cajones inferior contra el número de vehículos en el área Metropolitana, ya que en total se tienen 190 000 cajones de estacionamiento. La gráfica siguiente muestra la distribución de los cajones d estacionamiento en el Valle de México.



II.4 Contaminación generada por un vehículo

Los automotores representan una fuente importante de contaminación del aire. El parque automotor incluye un numeroso y activo conjunto de vehículos propulsados por la combustión de hidrocarburos (ciclomotores, automóviles y camiones). Las emisiones procedentes de los escapes de estos vehículos contienen CO_2 o dióxido de carbono; SO_2 que es dióxido de azufre; plomo; partículas sobrantes de la combustión del diesel; NO_2 u óxido de nitrógeno; COV que son compuestos orgánicos volátiles y O_3, que es el ozono. Por esta razón, las zonas urbanas más pobladas son las que sufren la mayor contaminación de este tipo.

La contaminación vehicular del aire produce efectos nocivos para la salud humana. Los estudios epidemiológicos demuestran que el aumento de los casos de enfermedades respiratorias está relacionado con el uso del vehículo (Salamanca, 1968). Los modelos de vehículos más comunes en México son Aveo, Versa, Tsuru, Vento, Spark, Jetta, March, Clásico, Sentra y Tiida. El Instituto Nacional de Ecología, del gobierno Mexicano, tiene entre las opciones sobre eficiencia energética, un portal para que puedas hacer el análisis de algún vehículo. Las categorías van por calificación de emisiones, por marca y modelo, clase de automóvil y gasto anual de combustible, como opciones principales (Mexicano, 2010). En la tabla II-3 se detalla la contaminación al aire generada por estos vehículos en México.

Tabla II- 2 *Cantidad de contaminantes generados por automotores* ***(Mexicano, 2010)***

Marca	Submarca	Combustible	CO_2 (gr/Km)	NO_x (gr/1000Km)
Chevrolet	Aveo	Gasolina	184	44
Nissan	Versa	Gasolina	170	24
Nissan	Tsuru	Gasolina	208	55
Volkswagen	Vento	Gasolina	192	42
Chevrolet	Spark	Gasolina	156	91
Volkswagen	Jetta	Gasolina	212	12
Nissan	March	Gasolina	168	17
Volkswagen	Clásico	Gasolina	196	211
Nissan	Sentra	Gasolina	197	90
Nissan	Tiida	Gasolina	211	25
		PROMEDIO	189	61

II.5 Bases de operación

El sueño de un usuario es poder llegar a la fuente misma de su destino y poder estacionar su automóvil en forma rápida y cómoda, dejándolo cerrado y en resguardo seguro, bajo techo. También deseará un mínimo de tiempo para sacarlo del estacionamiento. Estos objetivos deben ser tomados siempre en cuenta por el proyectista. Cuando más se acerque el proyecto del edificio a estas finalidades, mayor será el éxito del estacionamiento (Cal y Mayor, Estacionamiento, 1986).

Otro factor que es de suma importancia y que le sumará puntos a un estacionamiento es la seguridad que este ofrezca, ya que el conductor desea estacionar su vehículo libre de riesgos como el robo, los incendios, el maltrato de los acomodadores y colisiones con otros vehículos o la estructura misma del lugar. Finalmente no debe despreciarse el hecho de que el usuario tiende a buscar siempre la solución más económica, pagando lo menos posible por el estacionamiento, ya que muchas veces preferirá correr los riesgos mencionados estacionándose en la vía pública, sin pagar o pagando bastante menos (Cal y Mayor, Estacionamiento, 1986).

Con base a lo anteriormente mencionado se puede deducir que el proyecto de estacionamiento debe cuidar aspectos como costo mínimo con una máxima seguridad y comodidad (Cal y Mayor, Estacionamiento, 1986). Una vez definidas las bases de operación, se muestra un esquema en la Figura II.2 en el cual se engloban los principales sistemas que conforman un estacionamiento robotizado.

Figura II. 2 Esquema de los elementos principales en un estacionamiento robotizado

Máquinas eléctricas.- La máquina eléctrica implementada en un estacionamiento robotizado es el motor de corriente alterna, ya que fundamentalmente consigue un buen rendimiento, bajo mantenimiento y sencillez en su construcción. Esta máquina transformará la energía eléctrica en mecánica para que el sistema ejecute los movimientos requeridos, y dependiendo de las características del sistema se seleccionará el motor correspondiente.

Sistemas de control.- Básicamente es un conjunto de componentes que pueden regular la conducta de otro sistema con el objetivo de lograr el funcionamiento predeterminado para obtener los resultados deseados.
Para el control del funcionamiento de los motores existen diferentes sistemas de control aplicables, como: tecnología TTL, microcontroladores, PLC y PAC. A continuación se dará una breve descripción de cada uno de estos elementos.

Tecnología TTL.- "Acrónimo inglés de Transistor-Transistor Logic o Lógica Transistor a Transistor". Tecnología de construcción de circuitos electrónicos digitales, en los que los elementos de entrada de la red lógica son transistores, así como los elementos de salida del dispositivo" (UNICROM, 2013).

Microcontroladores.- "Es un circuito integrado o chip que incluye en su interior las tres unidades funcionales de una computadora: CPU, Memoria y Unidades de E/S, es decir, se trata de un computador completo en un solo circuito integrado.
Son diseñados para disminuir el coste económico y el consumo de energía de un sistema en particular. Por eso el tamaño de la CPU, la cantidad de memoria y los periféricos incluidos dependerán de la aplicación" (Galeón, 2013).

PLC.- "Es una computadora utilizada en la ingeniería automática o automatización industrial, para automatizar procesos electromecánicos, tales como el control de la maquinaria de la fábrica en líneas de montaje o atracciones mecánicas" (National Instruments, 2013).

PAC.- "El controlador de automatización programable es una tecnología orientada al control automático, combina tanto la funcionalidad de una PC y la confiabilidad de un PLC" (National Instruments, 2013).

Sistemas de comunicación.- "Un sistema de comunicación permite que la información sea transferida, a través del espacio y el tiempo, desde un punto llamado fuente hasta otro punto de destino, mediante un cable o por ondas electromagnéticas" (WordPress,

2013). Debido a que en este proyecto se implementará un sistema de comunicación inalámbrica se dará una breve definición de algunos de los protocolos existentes.

Bluetooth.- "La tecnología Bluetooth es un sistema de comunicaciones inalámbrico destinado a reemplazar los cables de conexión de muchos tipos diferentes de dispositivos, desde teléfonos móviles y auriculares para monitores cardíacos y equipos médicos. Está basado en la especificación IEEE 802.15.1" (Bluetooth, 2013).

ZigBee.- "Protocolo diseñado para satisfacer las necesidades inalámbricas de baja potencia, de bajo costo y redes de control en casi cualquier mercado. ZigBee puede utilizarse en casi cualquier lugar, es fácil de implementar y requiere poca energía para funcionar. Está basado en la especificación IEEE 802.15.4" (ZigBee Alliance, 2013).

RFID.- "La identificación por radiofrecuencia o RFID por sus siglas en inglés (radio frequency identification), es una tecnología de identificación remota e inalámbrica en la cual un dispositivo lector o "reader" vinculado a un equipo de cómputo, se comunica a través de una antena mediante ondas de radio" (egomexico, 2013).

II.6 Tiempos para estacionar o sacar un vehículo de un estacionamiento

"En general los estudios realizados en edificios de estacionamiento establecen que los dueños de automóviles manejan a menor velocidad que los acomodadores" (Cal y Mayor, Estacionamiento, 1986). Se observa en la Tabla II.3 que los acomodadores se tardan un promedio de 185 segundos desde recibir un automóvil, estacionarlo y regresar a la entrada. La Tabla II.4 muestra que para sacar un automóvil, desde tomar o recibir el boleto, ir por el vehículo, sacarlo del cajón y entregarlo, el acomodador tarda un promedio de 175 segundos. Cabe mencionar que estos tiempos son un promedio considerando solo 2 pisos y estos promedios varían con el tamaño del edificio, el proyecto del mismo y el movimiento que haya en el estacionamiento (Cal y Mayor, Estacionamiento, 1986).

Tabla II- 3 *Tiempos para estacionar un automóvil (por medio de acomodadores)* ***(Cal y Mayor, Estacionamiento, 1986)***

Entrar al automóvil y arrancar		8 Segundos
Circular en la planta baja		4 Segundos
Circular en la rampa (con promedio de 2 pisos, a 12 segundos por piso)		24 Segundos
Demora promedio en rampa		15 Segundos
Circular en el piso a estacionarse		6 Segundos
Estacionarse		18 Segundos
Apagar motor y salir del automóvil		6 Segundos
Anotar ubicación del vehículo en el boleto que queda en la oficina		31 Segundos
Caminar al elevador de servicio (promedio de 2 pisos)		45 Segundos
Bajar por el elevador		20 Segundos
Caminar a la entrada		8 Segundos
	SUMA	185 Segundos

Tabla II- 4 *Tiempos para sacar un automóvil (por medio de acomodadores)* ***(Cal y Mayor, Estacionamiento, 1986)***

Obtener el boleto		5 Segundos
Caminar al elevador de servicio		8 Segundos
Subir por el elevador		20 Segundos
Caminar hacia el cajón		45 Segundos
Verificar ubicación del automóvil		20 Segundos
Entrar al automóvil y arrancar		8 Segundos
Circular en el piso del estacionamiento		8 Segundos
Circular por la rampa		24 Segundos
Demora promedio en rampa		15 Segundos
Circular por la planta baja		6 Segundos
Pararse y salir del automóvil		6 Segundos
Entrega al usuario		10 Segundos
	SUMA	175 Segundos

Las tablas anteriores muestran las actividades que regularmente se llevan a cabo en un estacionamiento tradicional de la Ciudad de México. Con la implementación de un estacionamiento como el propuesto en este trabajo, se omitirán algunas de esas actividades, ya que se será automático. A continuación se muestran de forma muy general las actividades a realizar por el cliente para estacionar o sacar su vehículo, y esto se observa en las Tablas II.5 y II.6. Cabe mencionar que no se consideran tiempos, ya que no es un sistema real, y por lo tanto no puede ser comparado con los tiempos de las tablas anteriores.

Tabla II- 5 *Actividades para estacionar un automóvil (Estacionamiento automático)*

Estacionarse
Apagar motor y salir del automóvil
Obtener el boleto

Tabla II- 6 *Actividades para sacar un automóvil (Estacionamiento automático)*

Pagar cuota
Entrega de unidad al cliente
Entrar al automóvil y arrancar

Como puede apreciarse, el estacionamiento automático propuesto disminuirá el tiempo de recepción y entrega de vehículos al reducir las actividades a realizar para dicho propósito. Ahora bien, considerando la información mostrada en la Tabla II-2 y comparándola con los tiempos obtenidos para estacionar y sacar un vehículo mostrados en las Tablas II-3 y II-4 respectivamente, de un aparcamiento tradicional se obtiene la información siguiente.

Si se considera el tiempo que el automotor permanece encendido dentro del estacionamiento, tanto a la entrada como a la salida, obtenemos un tiempo total de 2 minutos, y si la velocidad permitida para circular en los aparcamientos es de 10 km/hr, se obtiene una distancia recorrida de 330 m ó 0.33 km, lo que resulta en 62.4 gr de CO_2 y 0.02 gr de NOx por vehículo. Ahora bien, eso es por un solo cajón de estacionamiento, pero si se toman en cuenta el total de cajones de aparcamiento en la ciudad, tenemos los resultados siguientes:

- 11 856 000 gr de CO_2
- 3 824.7 gr de NOx

Es importante aclarar que este resultado es por día, y que con base a la experiencia personal se sabe que no todos los estacionamientos son de dos pisos, que los autos tardan más tiempo en salir y en entrar y que los cajones de estacionamiento no sólo se ocupan una vez al día. El estacionamiento propuesto en este trabajo disminuirá ese bióxido de carbono generado por los vehículos, ya que en cuanto entrar al estacionamiento, los automotores permanecen apagados hasta que son devueltos en el mismo punto a sus dueños.

Capítulo III Desarrollo de ingeniería

III.1 Introducción

Haciendo un análisis de los aspectos y argumentos expuestos en los capítulos anteriores, se puede concluir que existe una creciente necesidad de generar más espacios públicos o privados destinados al aparcamiento de automóviles en la Ciudad de México, pero para lograr esto no se deben descuidar aspectos importantes en materia de medio ambiente.

A partir de este capítulo se desarrolla la idea que pretende solucionar las problemáticas presentes en los estacionamientos tradicionales. Esta idea consta de una propuesta que en esencia, generará un lugar de aparcamiento, el cual en comparación con un estacionamiento tradicional lo superará en diversos aspectos, ya que se propone un estacionamiento automático y seguro.

La idea propuesta se verá reflejada en el diseño de un prototipo, que estará apoyado de una herramienta CAD-CAE, en el cual se dibujarán las piezas, que posteriormente se llevarán a la realidad con diversos materiales, y de igual manera, se hará el estudio de movimiento correspondiente para confirmar su funcionalidad. Vale la pena decir que la finalidad del prototipo es mostrar el accionamiento de motores, así como la gestión del mismo para conseguir autonomía en su funcionamiento.

En este capítulo se desarrollan los criterios y aspectos considerados para la selección del equipo para el prototipo.

Tomando en cuenta las problemáticas que se han mencionado en el desarrollo de este trabajo, se pensó en un diseño arquitectónico que cumpla con la demanda de cajones de estacionamiento del lugar donde se vaya a construir, pero con las grandes ventajas de hacerlo en menor espacio terrestre que un estacionamiento tradicional, y de ofrecer al cliente la comodidad que necesita; también se contemplan aspectos como el diseño del mecanismo, con la finalidad de que cuente con la movilidad y versatilidad necesaria de acomodar y devolver automóviles.

III.2 Diseño del prototipo

III.2.1 Diseño conceptual

Para llevar a cabo el diseño del prototipo se consideraron aspectos como las necesidades que se desean satisfacer y las características de los estacionamientos automáticos actuales. El estacionamiento automático radial propuesto en esencia, es un sistema compuesto de múltiples elementos que en conjunto serán capaces de ejecutar las tareas de almacenar y devolver automóviles cuando se solicite.

La finalidad principal de este sistema, es que pueda funcionar de manera autónoma, puesto que el estacionamiento sólo espera recibir la instrucción del conductor, ya sea asignar un lugar de estacionamiento a su vehículo, o que éste le sea devuelto.

Para lograr lo anterior, se necesita de un mecanismo que gobernado por un algoritmo de control sea capaz de ejecutar esas tareas, un mecanismo que pueda acceder sin ningún inconveniente al lugar donde el usuario va a dejar su automóvil y de igual manera, pueda acceder a los diferentes lugares de resguardo de los mismos. Es por esto que la forma geométrica del proyecto propuesto en este trabajo se determinó tal que cumpliera con la funcionalidad deseada del mecanismo y a la vez fuera vanguardista, es así como se decide que una forma cilíndrica se adecua a lo planteado.

El estacionamiento propuesto contará con un lugar dónde el cliente podrá dejar su automóvil, que será la entrada al establecimiento, posteriormente el conductor se dirigirá a un panel de control dónde le indicará al sistema que puede asignarle un lugar de aparcamiento a su vehículo, esto con el objetivo de asegurar que en el automóvil no se encuentre alguna persona o que haya olvido alguna cosa. Acto seguido el mecanismo tomará el automóvil del punto de

acceso y le asignará de forma automática el lugar de estacionamiento más próximo.

El problema inicial que se presenta, es tomar el vehículo de la entrada, que es el punto donde lo deja su conductor. Para esto se necesita de un elemento que tome al automóvil sin ocasionarle algún daño y la forma de hacerlo es cuidando que el apoyo del vehículo sea siempre sobre sus llantas. La Figura III.1 muestra un boceto de la forma en que debe ser tomado el vehículo. Las piezas de color verde son elementos estáticos llamadas plataformas fijas y son las encargadas de recibir al conductor con su automóvil. Estos elementos estarán ubicados en un llamado cuarto de acceso, la función de éste, es recibir los automóviles que serán estacionados o entregados a sus conductores, además es el único elemento visible y accesible del sistema para el usuario. La pieza mostrada en color rojo es la encargada de tomar al automóvil, su función es recoger el vehículo que se encuentre en el cuarto de acceso y llevarlo a un lugar de estacionamiento, esto en el caso de acomodar el automóvil. Pero también puede ejecutar la función contraria, como lo es recoger un auto del lugar donde está aparcado para posteriormente llevarlo al cuarto de acceso, esto en el caso de entregar el vehículo. Esta pieza será nombrada plataforma móvil.

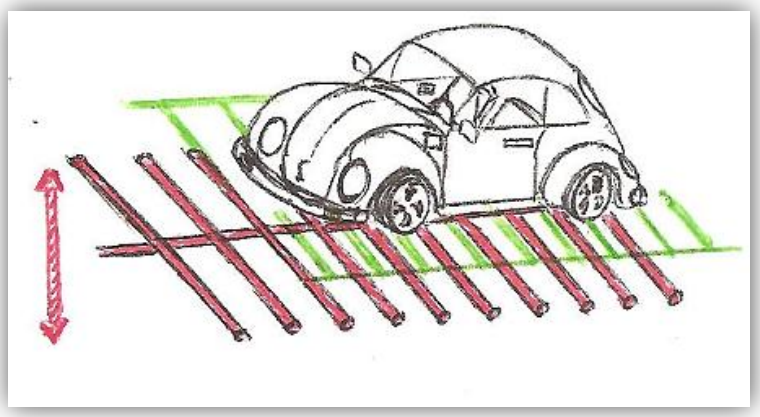


Figura III. 1 Diseño conceptual de plataformas fijas y plataforma móvil

Después de que un automóvil llega al cuarto de acceso, se posiciona en las plataformas fijas y el usuario desde el panel de control indica al sistema que debe llevarse su vehículo, la plataforma móvil se entrelazará con las plataformas fijas para conseguir que el vehículo ahora descanse sobre la plataforma móvil.

Una vez logrado lo anterior, el mecanismo debe ubicar al vehículo en un cuarto llamado cajón, que será dónde el automóvil permanecerá estacionado. El número de cajones del estacionamiento estará determinado por la demanda del lugar, estos estarán distribuidos en el perímetro del estacionamiento, como se muestra en la Figura III.2. En el caso del prototipo se tendrán únicamente 4 cajones que servirán perfectamente para demostrar el funcionamiento del mecanismo. Los cajones están conformados de igual forma por plataformas fijas y el principio de funcionamiento de la plataforma móvil será el mismo que en el cuarto de acceso.

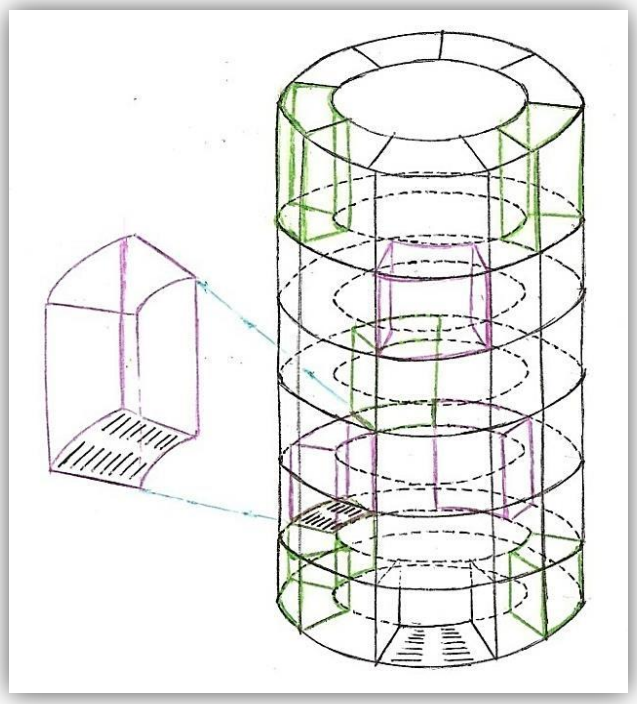


Figura III. 2 Diseño conceptual de cajones de estacionamiento

Como se observa en la imagen anterior, se requiere que el mecanismo se ubique en la parte central del estacionamiento. Los estacionamientos automáticos radiales actuales cuentan con un mecanismo ubicado en el centro del edificio, éste realiza un movimiento rotatorio para ubicarse en cualquier cajón del aparcamiento, pero dichos estacionamientos sólo cuentan con un mecanismo capaz de transportar los vehículos, como se observa en la Figura I.9. El estacionamiento propuesto en este trabajo podrá tener más de un mecanismo encargado de estacionar o entregar vehículos, dichos mecanismos funcionarán sin depender uno de otro, por lo que se requiere una estructura que permita el movimiento de traslación de trayectoria circular. La Figura III.3 muestra de color rojo unos ejes llamados rieles, que se ubican en la parte central del prototipo, estos permitirán que los mecanismos se trasladen por el edificio. Las piezas de color rojo achuradas representan las bases de los mecanismos que cumplirán la función de ubicar en cualquier columna de cajones a la plataforma móvil, estas piezas serán nombradas bases móviles.

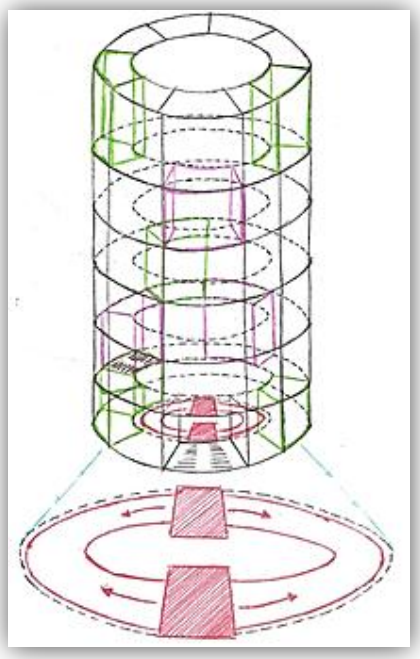


Figura III. 3 Diseño conceptual de rieles y bases móviles

Para poder desplazarse por cualquier nivel del estacionamiento el mecanismo contará con una cabina, la cual contendrá la plataforma móvil y realizará un movimiento de traslación de trayectoria rectilínea sobre el eje z. La Figura III.4 muestra de color morado la cabina, la cual se deslizará por los ejes que se muestran de color azul, de esta forma la cabina podrá ubicarse en cualquier nivel del estacionamiento.

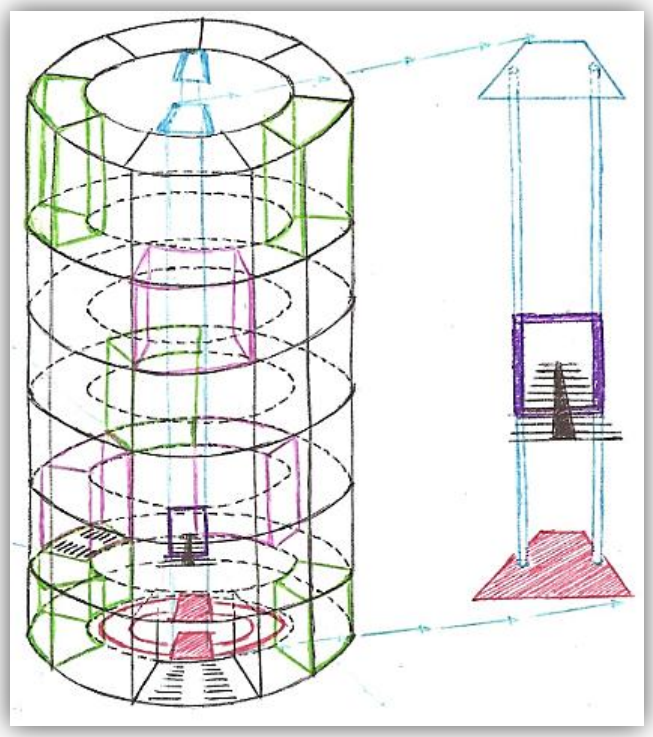


Figura III. 4 Diseño conceptual de cabina y ejes

En la Figura III.5 se engloban los elementos que conforman el estacionamiento, los cuales son: plataforma móvil, plataforma fija, cajón, cabina, base móvil, rieles, cuarto de acceso y ejes.

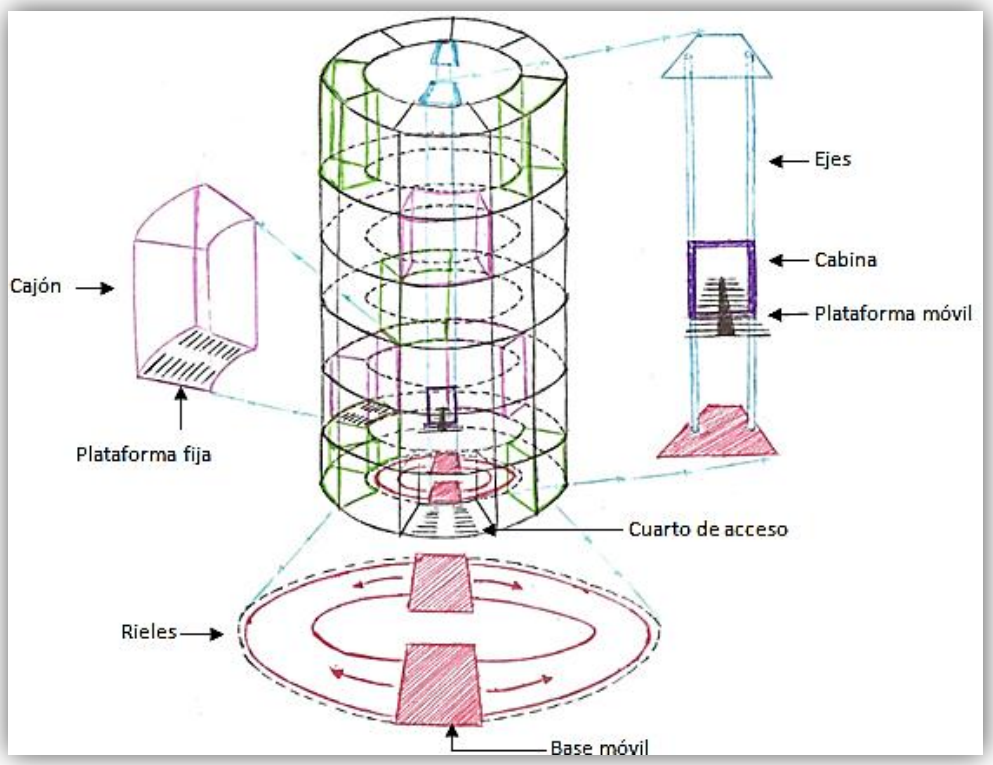


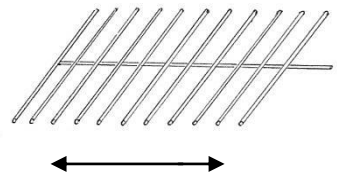
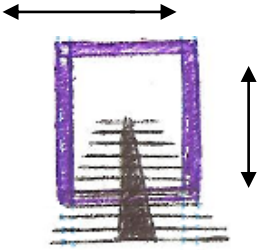
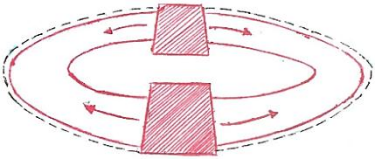
Figura III. 5 Diseño conceptual de elementos del estacionamiento

Para facilitar el estudio del prototipo se dividirá en dos partes: estructura móvil y estructura fija.

Estructura móvil

Sistema compuesto por los elementos siguientes: base móvil, cabina y plataforma móvil, que en conjunto generan 3 movimientos básicos necesarios con la capacidad de acomodar y entregar un automóvil. En la Tabla III.1 se describen estos movimientos:

Tabla III- 1 Descripción de movimientos de los mecanismos del sistema

Movimiento	Descripción	Figura
1. Traslación de trayectoria rectilínea	Cambio de posición de la plataforma móvil en el eje X.	
2. Traslación de trayectoria rectilínea	Cambio de posición de la cabina en el eje z.	
3. Traslación de trayectoria curvilínea	Cambio de posición de la base móvil a través de los rieles.	

Estructura fija

Sistema compuesto por los elementos siguientes: cuarto de acceso, cajones y rieles. Como se puede observar hasta este momento sólo se tienen los diseños conceptuales de los elementos que conforman el prototipo, sin embargo no se ha desarrollado la forma en que el mecanismo realizará sus tareas, por lo que en las siguientes líneas se dará solución a esta problemática.

Mecanismo de plataforma móvil

Como se observa en la Figura III.6 el mecanismo que permitirá el movimiento de traslación de la plataforma móvil estará compuesto por diversos elementos, como lo son: un riel, dado, tornillo sin fin y motor eléctrico. En conjunto estos elementos logran que la plataforma móvil realice su función, ya que convierten el movimiento rotatorio del motor eléctrico a uno lineal.

La plataforma móvil debe cumplir con ese movimiento debido a que el cuarto de acceso y los cajones no están en el mismo perímetro que la estructura móvil, ya que estos se encuentran en un perímetro de radio superior.

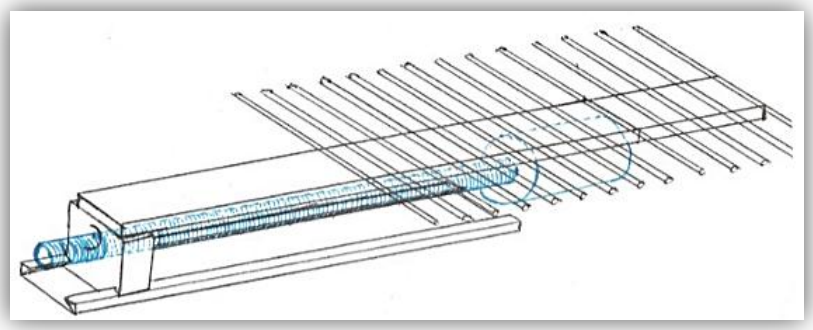


Figura III. 6 Diseño conceptual del mecanismo de plataforma móvil

Mecanismo de cabina

En la Figura III.7 se muestra a la cabina montada en los ejes ubicados en los extremos, que servirán para guiar a esta pieza en su movimiento ascendente/descendente debido a la acción de un motor eléctrico que estará fijo en la parte superior de la estructura móvil (techo).

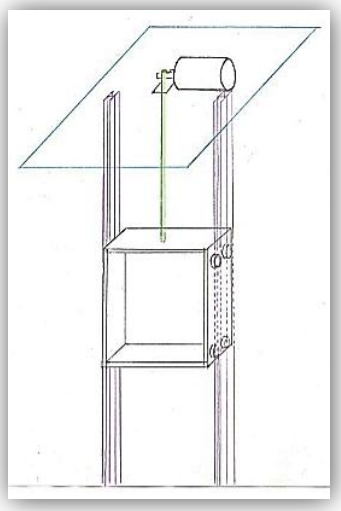


Figura III. 7 Diseño conceptual del mecanismo de cabina

Mecanismo de la base móvil

La acción que se desea realizar en este mecanismo es la de mover a toda la estructura móvil en el perímetro de los rieles, esto inicialmente se planteó como se observa en la Figura III.8 en la cual se planeaba colocar un motor en la base móvil acoplando la flecha para que transmitiera el movimiento rotatorio del motor a un riel de la estructura fija.

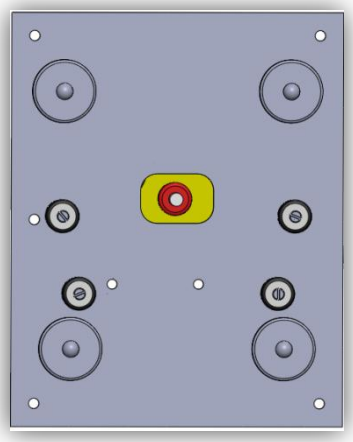


Figura III. 8 Mecanismo de la base móvil

Cabe mencionar que las ideas planteadas en esta sección son solamente ideas conceptuales, que dan las bases para la construcción del prototipo.

III.2.2 Selección del Hardware

En este apartado se describirán los criterios para la selección del equipo implementado en el prototipo, y como se mencionó en el capítulo 2 el estacionamiento puede dividirse en tres áreas principales, máquinas eléctricas, sistemas de control y sistemas de comunicación.

Máquina eléctrica

Para la selección de esta, se deben tener en consideración aspectos como las medidas del prototipo y el peso de los mecanismos a los cuales les darán movimiento, ya que sabiendo estas características se elegirá el motor dependiendo del par y la corriente nominal del mismo. Los motor reductores de corriente directa se vuelven los motores indicados para realizar el tipo de tareas que demandan los mecanismos del proyecto ya que cuentan con un juego de engranes que reducen la velocidad de la flecha, pudiendo así ejecutar los movimientos de la estructura sin cambios de posición bruscos como sucedería con motores de corriente directa. Además de que los motor reductores tienen un torque mayor que un motor de CD a causa de la caja de engranes con la que cuentan, y pueden desplazar la cabina, la base móvil y la rejilla sin que su peso se vuelva un inconveniente.

Tomando en cuenta la escala del prototipo y el peso del mecanismo se decidió implementar un motor reductor de corriente continua de 5-12 V, de 100 rpm y un par de 5 kg, para el accionamiento de los tres mecanismos del sistema. La Figura III.9 muestra el motor reductor anteriormente mencionado.

Figura III. 9 Motor reductor de corriente continua

Sistema de control

Para llevar a cabo el accionamiento de los motores de este proyecto, se puede elegir entre una amplia variedad de controles como pueden ser compuertas lógicas de tecnología TTL, PLC, PAC, microcontroladores, por mencionar algunos. Pero para este proyecto se torna sencilla la elección de uno de estos, ya que como es un prototipo a escala en donde el espacio con el que se cuenta es reducido queda descartado el uso de un PLC o un PAC, que son equipos de gran tamaño que se volverían difíciles de adecuar, además de ser un equipo de un costo superior a un microcontrolador.

El uso de compuertas lógicas no es viable debido a la gran complejidad de los circuitos, refiriéndose al cableado que se necesita para interconectarlas, al gran espacio que ocupan y a la carencia de dispositivos para lograr comunicación inalámbrica. Los microcontroladores son dispositivos en los cuales se pueden desarrollar las mismas funciones que en un circuito diseñado con compuertas lógicas solo que en un menor espacio, su reducido tamaño vuelven a estos dispositivos los ideales para este proyecto además de ser equipos de un bajo costo comparado con un PLC o PAC y casi todos los modelos de microcontroladores sin importar la marca, cuentan con puertos de comunicación.
Los microcontroladores utilizados en este trabajo son:

- Tarjeta Arduino Mega 2560
- Tarjeta Arduino Leonardo

Estas tarjetas Arduino satisfacen las necesidades a cubrir, ya que cada una de ellas tiene las entradas/salidas necesarias que requiere el sistema.

La tarjeta Arduino Mega 2560 estará en la estructura fija del prototipo, y en ella llegarán las señales de los sensores ubicados en los cajones, los cuales indicarán si hay o no automóvil, y la tarjeta Arduino Leonardo estará ubicada en la estructura móvil, y estará monitoreando la acción de los motores, así entre los dos microcontroladores se estará efectuando una comunicación inalámbrica dúplex (más adelante se explica por qué comunicación inalámbrica), debido a que el sistema es capaz de mantener una comunicación bidireccional enviando y recibiendo mensajes. A continuación se muestran las especificaciones de cada microcontrolador.

Arduino Mega 2560

El Arduino Mega 2560 es un microcontrolador basado en el ATmeg1280. Tiene 54 entradas/salidas digitales (de las cuales 14 proporcionan salida PWM), 16 entradas analógicas, 4 puertos serie por hardware, un cristal oscilador de 16MHz, conexión USB, entrada de alimentación y botón de reset. En la Figura III.10 se observa la tarjeta Arduino Mega 2560 y la Tabla III-2 engloba las especificaciones generales de la misma (Arduino, 2013).

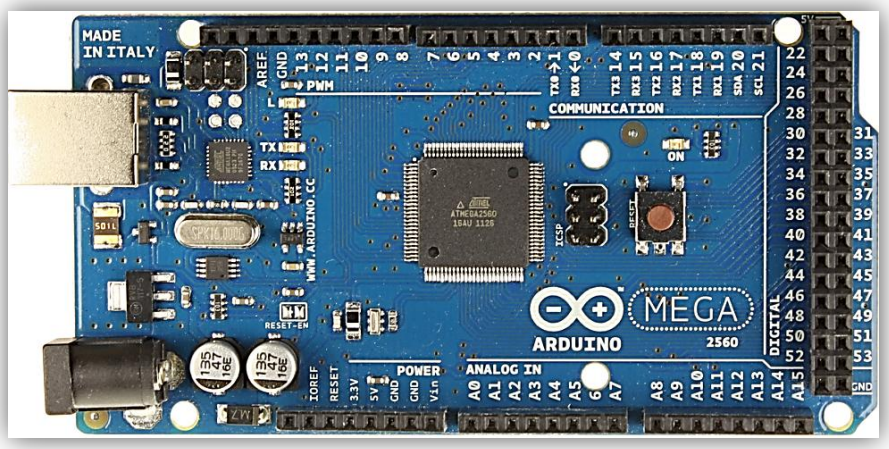


Figura III. 10 Tarjeta Arduino Mega 2560 (Arduino, 2013)

Tabla III- 2 Especificaciones Arduino Mega 2560 (Arduino, 2013)

Microcontrolador	ATmega1280
Voltaje de funcionamiento	5V
Voltaje de entrada (recomendado)	7-12V
Voltaje de entrada (limite)	6-20V
Pines E/S digitales	54 (14 proporcionan salida PWM)
Pines de entrada analógica	16
Intensidad por pin	40 Ma
Intensidad en pin 3.3V	50 Ma
Memoria Flash	128 KB de las cuales 4 KB las usa el gestor de arranque (bootloader)
SRAM	8 KB
EEPROM	4 KB
Velocidad de reloj	16 MHz

Cada uno de los 54 pines digitales en el microcontrolador Mega operan a 5 volts, pueden proporcionar o recibir una intensidad máxima de 40 mA y pueden utilizarse como entradas o como salidas usando las funciones pinMode(), digitalWrite(), y digitalRead(). Algunos pines tienen funciones especializadas, y a continuación se describen cada uno de ellos:

- Serie: 0 (RX) y 1 (TX), Serie 1: 19 (RX) y 18 (TX); Serie 2: 17 (RX) y 16 (TX); Serie 3: 15 (RX) y 14 (TX). Usado para recibir (RX) transmitir (TX) datos a través de puerto serie TTL.
- PWM: de 0 a 13. Proporciona una salida PWM (Pulse Wave Modulation, modulación de onda por pulsos) de 8 bits de resolución a través de la función analogWrite().

- LED: 13. Hay un LED integrado en la placa conectado al pin digital 13, cuando este pin tiene un valor HIGH(5V) el LED se enciende y cuando este tiene un valor LOW(0V) este se apaga.

El Mega tiene 16 entradas analógicas, y cada una de ellas proporciona una resolución de 10bits (1024 valores). Por defecto se mide de referencia a 5 volts, aunque es posible cambiar la cota superior de este rango usando el pin AREF y la función analogReference(). Además algunos pines tienen funciones especializadas, como los siguientes:

- I²C: 20 (SDA) y 21 (SCL). Soporte del protocolo de comunicaciones I²C (TWI) usando la librería Wire.
- AREF. Voltaje de referencia para las entradas analógicas. Usado por analogReference().
- Reset. Suministrar un valor LOW (0V) para reiniciar el microcontrolador.

Arduino Leonardo

El Arduino Leonardo es una tarjeta basada en el ATMEGA32U4. Dispone de 20 pines de entrada /salida digital (de los cuales 7 se puede utilizar como salidas PWM y 12 como entradas analógicas), un oscilador de 16MHz, una conexión micro USB, un conector de alimentación y un botón de reinicio. La Figura III.11 muestra la tarjeta Arduino Leonardo y la Tabla III-3 sus especificaciones.

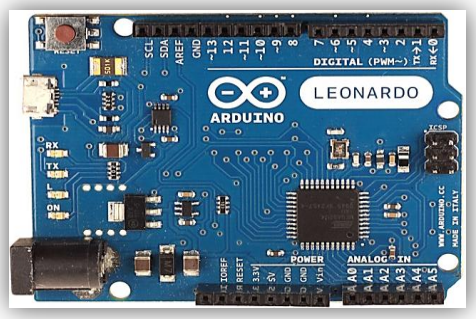


Figura III. 11 Tarjeta Arduino Leonardo (Arduino, 2013)

Tabla III- 3 Especificaciones Arduino Leonardo (Arduino, 2013)

Microcontrolador	ATmega32u4
Voltaje de funcionamiento	5V
Voltaje de entrada(recomendado)	7-12V
Voltaje de entrada (límite)	6-20V
Pines E/S digitales	20 (7 proporcionan salida PWM)
Pines de entrada analógica	12
Intensidad por pin	40 mA
Intensidad en pin 3.3 V	50 mA
Memoria Flash	32 KB (ATmega32u4) de las cuales 4 KB las el gestor de arranque (bootloader)
SRAM	2.5 KB
EEPROM	1 KB
Velocidad de reloj	16 MHz

Cada una de los 20 pines digitales en el Arduino Leonardo se puede utilizar como una entrada o salida, utilizando funciones digitalRead (), pinMode () y digitalWrite (). Cada pin puede proporcionar o recibir un máximo de 40 mA y tiene una resistencia pull-up interna (desconectada por defecto) de 20 a 50 kΩ. Además, algunos pines tienen funciones especializadas, y estos se describen a continuación:

- Serie: 0 (RX) y 1 (TX). Se utiliza para recibir (RX) y transmitir (TX) datos en serie TTL.
- Interrupciones externas: 3 (interrupción 0), 2 (alarma 1), 0 (alarma 2), 1 (interrupción 3) y 7 (interrupción 4). Estos pines pueden ser configurados para activar una interrupción en un valor bajo, un flanco ascendente o descendente, o un cambio en el valor.

- PWM: 3, 5, 6, 9, 10, 11, y 13. Proporcionar salida PWM de 8 bits con la función analogWrite().
- LED: 13. Hay un LED conectado al pin digital 13. Cuando el pin tiene un valor alto, el LED está encendido, cuando el valor es bajo, está apagado.
- Entradas analógicas: A0- A5, A6 - A11 (en los pines digitales de 4, 6, 8, 9, 10 y 12). El Leonardo tiene 12 entradas analógicas, con la etiqueta A0 a A11, estos pines también puede ser utilizado como E / S digitales.
- AREF. Voltaje de referencia para las entradas analógicas. Se utiliza con analogReference().
- Restablecer. Lleve esta línea BAJO para reajustar el microcontrolador.

Sistema de comunicación

Cabe mencionar que una parte esencial del prototipo es la comunicación inalámbrica que se necesita para el control de los motores, ya que los motores estarán expuestos a un constante movimiento de translación debido a que están montados en la estructura móvil, y para ejecutar las secuencias de estos se necesita de una interfaz para el usuario accesible, eso quiere decir, que se encuentre estática, un lugar donde este fuera del ir y venir de la estructura móvil, como lo es la estructura fija. Una conexión por medio de cables entre el tablero de control para el usuario y los motores, se volvería un problema para la parte mecánica del proyecto. Es por eso que con justa razón se plantea una comunicación inalámbrica entre los elementos descritos anteriormente.

El protocolo de comunicación empleado en este proyecto es ZigBee, el cual se describe a continuación:

ZigBee es una tecnología inalámbrica diseñada para satisfacer las necesidades únicas de baja potencia, de bajo costo y redes de control en casi

cualquier mercado. ZigBee se puede implementar en diferentes proyectos, es fácil de aplicar y requiere poca energía para funcionar, esto da oportunidad de crecimiento en nuevos mercados, así como la innovación en los mercados existentes (ZigBee Alliance, 2013).

Este protocolo está basado en el estándar IEEE 802.15.4 de redes inalámbricas de área personal. Su objetivo son las aplicaciones que requieren comunicaciones seguras con baja tasa de envío de datos y maximización de la vida útil de sus baterías (ZigBee Alliance, 2013).

Se decidió implementar el protocolo de comunicación ZigBee en lugar de otro inalámbrico considerando lo anteriormente planteado de este protocolo y a que tiene grandes ventajas. A continuación se muestra la Tabla III-4 que engloba las ventajas de ZigBee contra Bluetooth.

Tabla III- 4 Características principales ZigBee y Bluetooth

Información	ZigBee	Bluetooth
Distancia de comunicación	10m a 1.5 km	10m a 100m
Consumo de energía	60mW	120mW
Tiempo de vida	Meses a años	Semanas
Número de nodos	65,535	7

Arduino cuenta con un shield que permite comunicar tarjetas Arduino de forma inalámbrica usando ZigBee. La Xbee Shield es compatible con el microcontrolador Mega 2560 y Leonardo, así que esto lo hace el equipo indicado. Este Shield permite conectar un módulo XBee con los microcontroladores Arduino Leonardo y Mega 2560, estos módulos son los que hacen posible la comunicación inalámbrica. La Figura III.12 muestra el XBee Shield, el cual se coloca sobre los cabezales de los microcontroladores, y sobre él se conecta el módulo XBee.

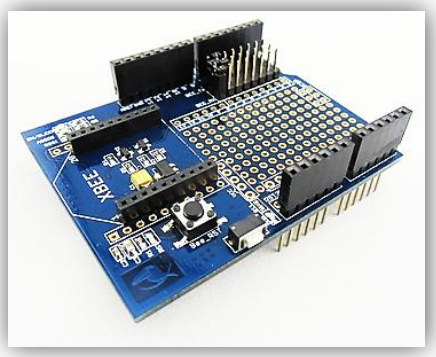


Figura III. 12 XBee Shield

La Figura III.13 muestra el módulo XBee el cual puede comunicarse hasta 100ft (30 metros) en interior o 300ft (90 metros) al aire libre (en visión directa). Cuando el módulo posee antena puede tener un alcance de 1.5 km (Arduino, 2013).

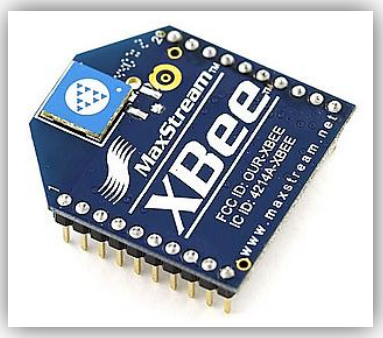


Figura III. 13 Módulo XBee

A continuación se describirán los comandos y direccionamiento de los módulos XBee

Comandos

La Tabla III-5 muestra los comandos a configurar del módulo XBee dependiendo de las características del sistema, así como su descripción y valores.

Tabla III- 5 Comandos módulo Xbee (Arduino, 2013)

Comando	*Descripción*	*Valores válidos*	*Valor por defecto*
ID	El ID de la red del módulo XBee.	0 - 0xFFFF	3332
CH	El canal del módulo XBee.	0x0B - 0x1A	0X0C
SH y SL	El número serie del módulo XBee (SH devuelve los 32 bits superiores, SL los 32 inferiores). De solo-lectura	0 - 0xFFFFFFFF (para ambos SH y SL)	diferente para cada módulo
MY	La dirección de 16-bit del módulo.	0 - 0xFFFF	0
DH y DL	La dirección de destino para las comunicaciones inalámbricas (DH son los 32 bits superiores, DL son los 32 inferiores).	0 - 0xFFFFFFFF (para ambos DH y DL)	0 (para ambos DH y DL)
BD	La velocidad de transmisión usada para las comunicaciones con el Arduino o el ordenador.	0 (1200 bps) 1 (2400 bps) 2 (4800 bps) 3 (9600 bps) 4 (19200 bps) 5 (38400 bps) 6 (57600 bps) 7 (115200 bps)	3 (9600 bps)
RE	Restaura los valores por defecto	No aplica	No aplica
WR	Escribe un nuevo valor para un parámetro configurado	No aplica	No aplica
CN	Sale del modo de comandos.	No aplica	No aplica

Direccionamiento

Hay múltiples parámetros que necesitan ser configurados correctamente para que dos módulos puedan comunicarse entre ellos. Necesitan estar en la misma red, definida por el parámetro ID. Los módulos necesitan estar en el mismo canal, definido por el parámetro CH. Finalmente, la dirección de destino de un módulo (parámetros DH y DL) determina que módulo en esa red y canal recibirá los datos transmitidos (Arduino, 2013).

- Si el DH de un módulo es 0 y su DL es menor de 0xFFF, los datos transmitidos por ese módulo serán recibidos por cualquier módulo cuyos 16 bits de dirección del parámetro MY sea igual al DL.
- Si el DH es 0 y el DL es igual a 0xFFFF, las transmisiones del módulo serán recibidas por todos los módulos.
- Si el DH no es cero o el DL es mayor de 0xFFFF, la transmisión solo será recibida por el módulo cuyo número de serie sea igual a la dirección de destino del módulo transmisor.

En el capítulo IV se describirá la configuración de los módulos XBee implementados en este prototipo.

Sensores

En el proyecto desarrollado en este trabajo se implementaron sensores infrarrojos con nomenclatura QRD1114, los cuales son sensores de corto alcance, basados en un emisor de luz y un receptor, ambos apuntando en la misma dirección, y cuyo funcionamiento se basa en la capacidad de reflexión del objeto, y la detección del rayo reflectado por el receptor. El detector consiste en un fototransistor, y estos nos ayudarán a detectar la posición del mecanismo, así como a gestionar si existe automóvil o no en el cuarto de acceso o cajones. La Figura III.14 muestra un sensor QRD1114.

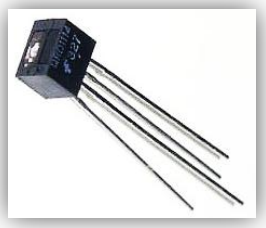


Figura III. 14 Sensor QRD1114

Para detectar la posición de la plataforma móvil se utilizan dos interruptores de límite, los cuales estarán ubicados en cada extremo o límite de posición de dicha plataforma. La Figura III.15 muestra el interruptor utilizado en el prototipo

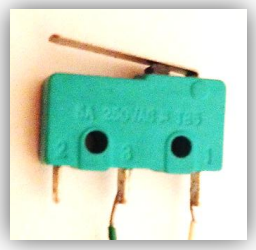


Figura III. 15 Interruptor de límite

Panel de control (HMI)

El panel de control forma parte del hardware del proyecto desarrollado en este trabajo, y a continuación se define todo lo referente al mismo. El panel de control es un elemento necesario en los sistemas automáticos para la comunicación del hombre con la máquina. En el caso de este proyecto es indispensable la implementación de un tablero de control, en donde el usuario pueda indicarle a la máquina instrucciones como asignarle un lugar de estacionamiento a su automóvil, o caso contrario, solicitarle a esta que le sea devuelto. El tablero de control diseñado para este prototipo cuenta con los elementos indicados en la Figura III.16. Estos elementos son:

Indicadores led: estos indicadores corresponden a los lugares de estacionamiento habilitados en este prototipo A, B, C, D respectivamente. Son los encargados de indicarle al usuario el lugar de estacionamiento que le fue asignado a su automóvil como puede ser A, B, C o D.

Display de 7 segmentos: Muestra las letras A, B, C o D. Funciona simultáneamente con los indicadores led.

Micropulsadores: Son los elementos que permiten al usuario del sistema interactuar con la máquina. A continuación se describe la función de cada botón pulsador en el proyecto: el botón "Llevar" lo oprime el usuario inmediatamente después de descender de su auto, con la finalidad de indicarle al sistema que puede llevarse su coche a un lugar de aparcamiento. Con el botón "Navegar" el usuario es capaz de buscar en el panel de control por medio de los indicadores led y el display, el cajón de estacionamiento en donde fue guardado su automóvil. Y el botón "Traer" lo oprime el usuario cuando encontró el lugar que le fue asignado a su vehículo con anterioridad por el sistema y quiere que le sea devuelto.

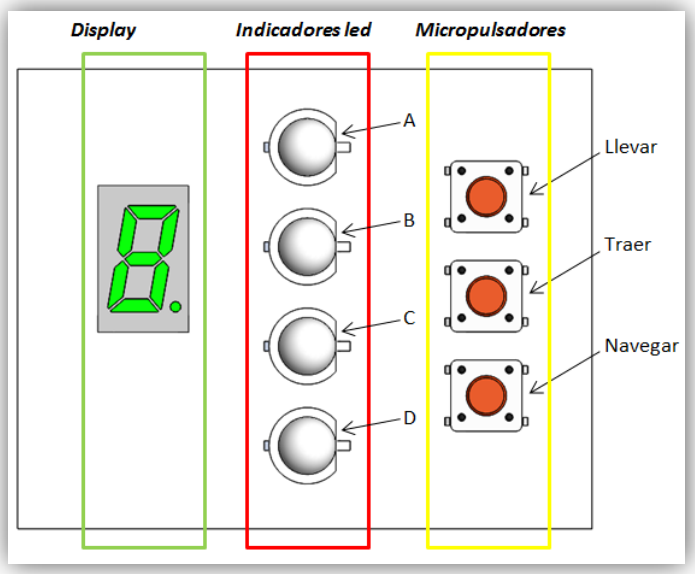


Figura III. 16 Tablero de control.

El conjunto de los indicadores led y el display de 7 segmentos son la ayuda visual con la que cuenta el usuario cuando esta frente al tablero, es por eso que se implementaron led bicolor (verde, rojo) como indicadores, con la finalidad de que cada color indique al cliente las acciones que está ejecutando la máquina:

- Verde: Cada vez que se presiona el botón de Navegar consecutivamente los led A, B, C, D encienden uno por uno con el color verde como se muestra en la Figura III.17, indicando al usuario los cajones de estacionamiento que se encuentran ocupados por automóviles, así si un led no enciende y la secuencia es saltada significa que no hay vehículo en ese cajón. De igual forma cada led enciende color verde cuando el sistema está trabajando para traer un automóvil de algún cajón del estacionamiento.

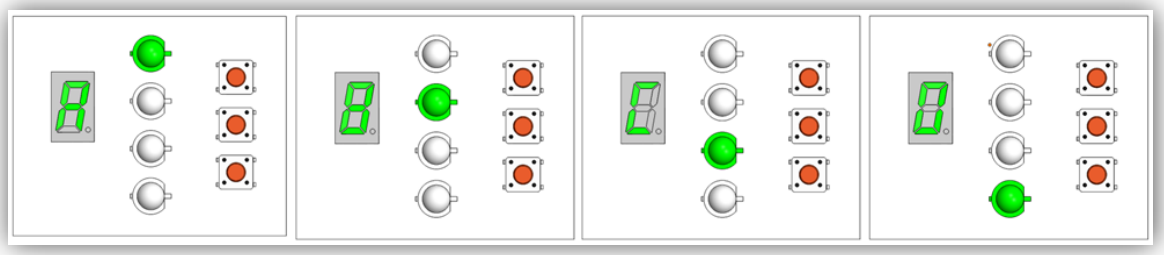


Figura III. 17 Función 1 panel de control

- Naranja: Los led encienden de este color después de que el usuario presiona el botón "Traer" como se muestra en la Figura III.18, indicando a la persona que el sistema está realizando un escaneo en el cuarto de acceso con el objetivo de encontrar un vehículo; si en el escaneo no se encuentra automóvil, el sistema ejecutará la acción de traer el vehículo que le fue solicitado y el led cambiará de color naranja a color verde, pero si se encuentra automóvil, el led comenzará a parpadear con el color naranja y el sistema esperará a que el dueño del auto del cuarto de acceso presione el botón "Llevar", una vez que presione el botón, el led cambiará de color naranja a color rojo indicando al usuario que el sistema está llevando el

automóvil del cuarto de acceso a un lugar de aparcamiento, cuando esta acción concluya el led cambiará de color rojo a color verde indicando así, que está trayendo el auto que le fue solicitado. Así con esta lógica el sistema ahorrará energía eléctrica debido a que primero estacionará el vehículo del cuarto de acceso y después traerá el automóvil que le fue solicitado.

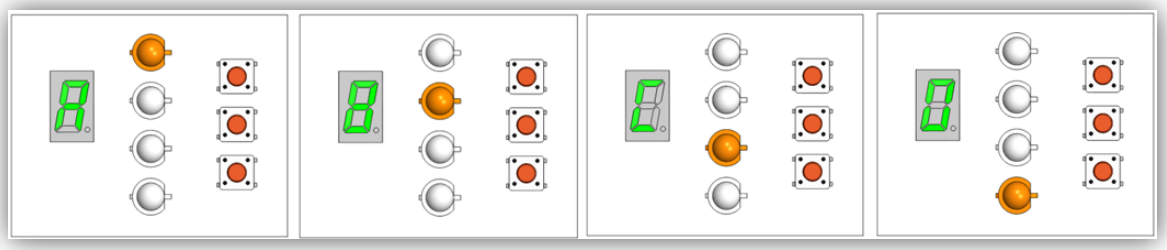


Figura III. 18 Función 2 panel de control

- Rojo: Este color en cada uno de los led se hace presente cuando es presionado el botón "Llevar". Dependiendo del led que encienda (A, B, C, D) es el cajón de estacionamiento que se le asigna al vehículo como se muestra en la Figura III.19. Cuando los cuatro led parpadean simultáneamente 3 veces con el color rojo como en se ilustra en la Figura III.20 significa que se está cometiendo un error en el sistema como puede ser que alguien en el panel de control está presionando el botón "Llevar" sin que exista carro en el cuarto de acceso o que alguien presione el botón de "Traer" sin que haya automóviles en el estacionamiento.

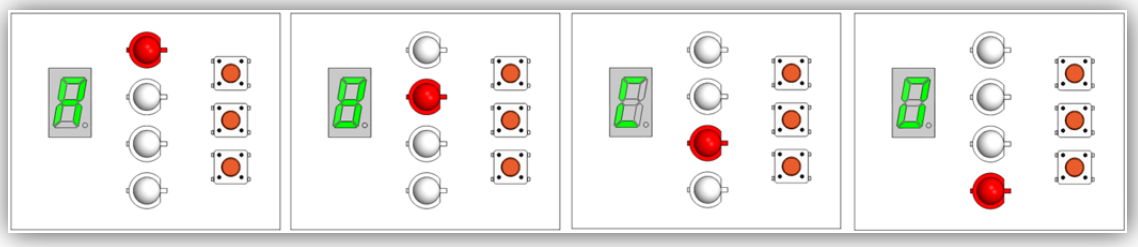


Figura III. 19 Función 3 panel de control

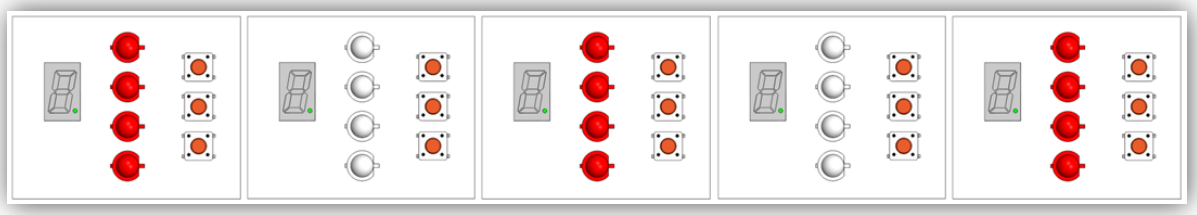


Figura III. 20 Función 4 panel de control

III.2.3 Desarrollo en herramienta de CAD- CAE

Para llevar a cabo la construcción del prototipo, el primer aspecto a considerar fue la escala a implementar. Se determinó que la escala indicada es 1:43 debido a que con ésta se puede demostrar claramente el accionamiento del prototipo, además de que las dimensiones de los diferentes materiales existentes en el mercado hacen de ésta escala la ideal para fabricar las piezas del sistema.

Haciendo un análisis de los materiales existentes en el mercado se concluyó que el correcto era Aluminio, ya que es un metal ligero y resistente para las acciones que realizará el prototipo. El Aluminio es un material del cual existe gran variedad de perfiles y medidas, además de ser fácil de manipular para la creación de piezas. Tomando como base las medidas y perfiles existentes de Aluminio, se diseñó el sistema propuesto en *una herramienta de CAD-CAE*, iniciando con la plataforma móvil.

Plataforma móvil

Como se mencionó anteriormente la escala empleada es 1:43, por lo tanto para un vehículo a esa escala, la plataforma móvil se diseñó en con las medidas mostradas en la Figura III.21 a), con el perfil de Aluminio que se observa en la Figura III.21 b)

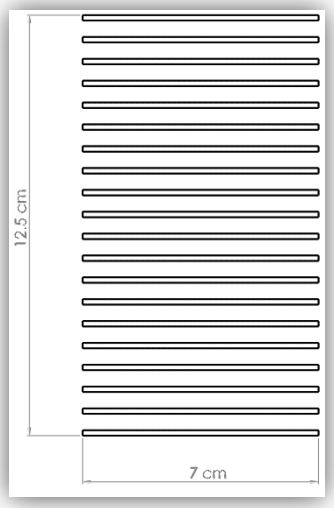


Figura III. 21 a) Dimensiones plataforma móvil b) Perfil de Aluminio

En la Figura III.22 a) se observan las dimensiones del perfil que soporta las tiras de Aluminio y en la Figura III.22 b) el perfil utilizado.

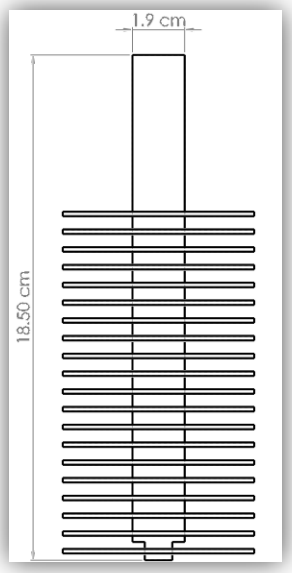


Figura III. 22 a) Soporte rejillas b) Perfil de Aluminio

El diseño conceptual de la Figura III.6 plantea que la forma de conseguir que la plataforma móvil realice el movimiento de translación es a partir de tres piezas, una de éstas es fija (riel), otra pieza ensamblada a la plataforma móvil que se deslizará sobre el riel (dado), y la última, un tornillo sinfín que transmitirá el movimiento rotatorio del motor al dado. La Figura III.23 a) muestra los perfiles ocupados para generar las piezas y la Figura III.23 b) muestra un ensamblaje de las piezas para transformar el movimiento rotatorio del motor en un movimiento lineal.

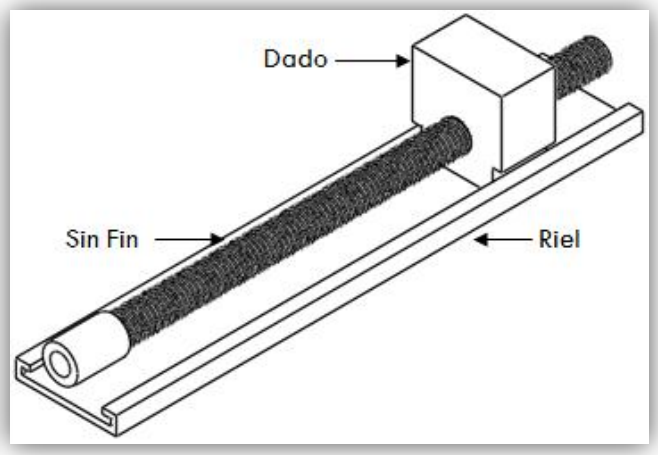


Figura III. 23 a) Perfiles de Aluminio b) Ensamblaje dado, riel y sin fin

Como se observa en la imagen anterior, el cubo se modificó de tal forma que se ajustara a la medida del perfil que fungirá como riel, de igual forma se realizó un orificio con cuerda para que por él circule el tornillo sin fin.

La Figura III.24 muestra el diseño realizado en una herramienta CAD-CAE de la plataforma móvil ya finalizada y con todas sus piezas ensambladas. Como se observa en un extremo del sin fin hay un conector que será el encargado de acoplar la flecha del motor con dicho tornillo.

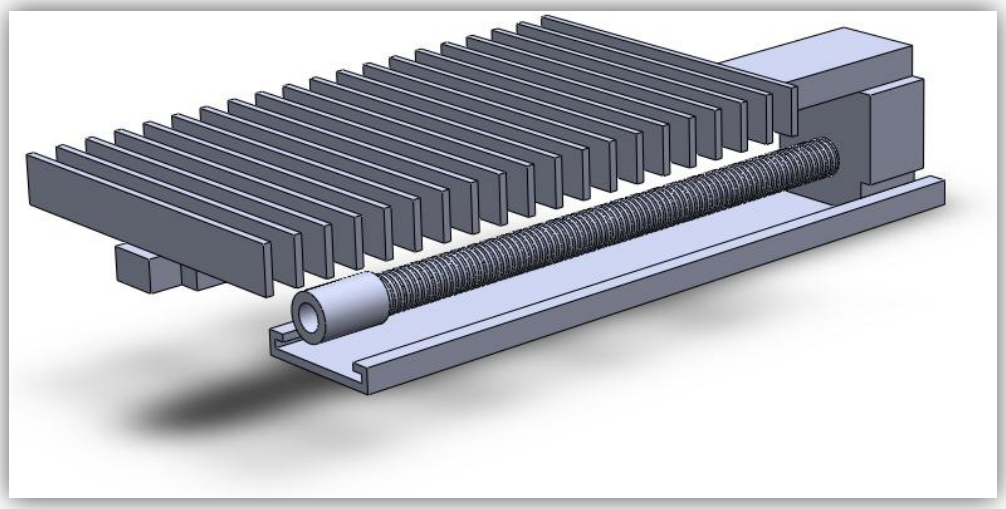


Figura III. 24 Mecanismo de plataforma móvil en herramienta CAD-CAE

El motor que realizará la tarea de mover la plataforma móvil será nombrado a lo largo de este trabajo como Motor 1.

Cabina

Continuando con el diseño de las piezas en la herramienta CAD-CAE, el siguiente mecanismo a tratar es el mecanismo de la cabina, la cual como se mencionó anteriormente contendrá a la plataforma móvil, y con base a esto y a los perfiles existentes se determinan las dimensiones de tal pieza. La Figura III.25 a) muestra los perfiles ocupados para la construcción de la cabina, mientras que la Figura III.25 b) muestra el CAD-CAE y dimensiones de la misma.

El motor encargado de generar el movimiento ascendente/descendente de la cabina será el Motor 2.

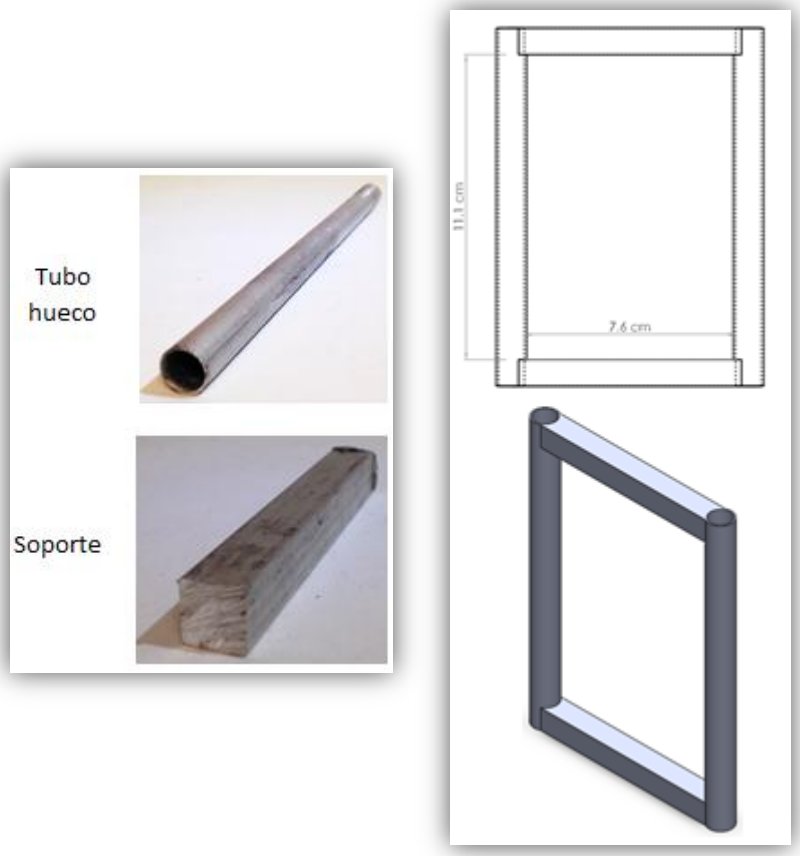


Figura III. 25 a) Perfiles de Aluminio ***b) Cabina***

El perfil cilíndrico de la cabina es el que circulará por los ejes ubicados en su interior, y con el perfil cuadrado se dará soporte a la pieza.

Base móvil

La siguiente pieza que se generó en la herramienta de CAD-CAE fue la base móvil. El planteamiento que se realizó en la idea conceptual cambió, ya que para el mecanismo implementado en el prototipo se realizó lo a continuación descrito.

Se colocaron cuatro llantas guías ubicadas en la base móvil, las cuales están distribuidas tangencialmente al riel central de la estructura fija. Con lo anteriormente descrito se logra guiar a la estructura móvil en la periferia de los

rieles, pero para conseguir que ésta se desplace, se necesita de un motor eléctrico colocado en algún punto de la base móvil que en la flecha tenga acoplada una llanta de plástico posicionada tangencialmente al riel central para que transmita el movimiento rotatorio y se logre el desplazamiento de la estructura móvil.

La Figura III.26 a) muestra la posición de las llantas, de color rojo la llanta acoplada a la flecha del motor y en color negro las llantas guías. La base móvil consiste en una lámina de Aluminio calibre 18, y en las esquinas de la misma se observan unas esferas pequeñas, las cuales se apoyarán en los rieles externos y facilitarán el giro de la estructura móvil. La Figura III.26 b) muestra la vista superior de la base móvil, pudiéndose observar de igual forma el motor fijo a la lámina de aluminio, el cual durante el trabajo tendrá el nombre Motor 3.

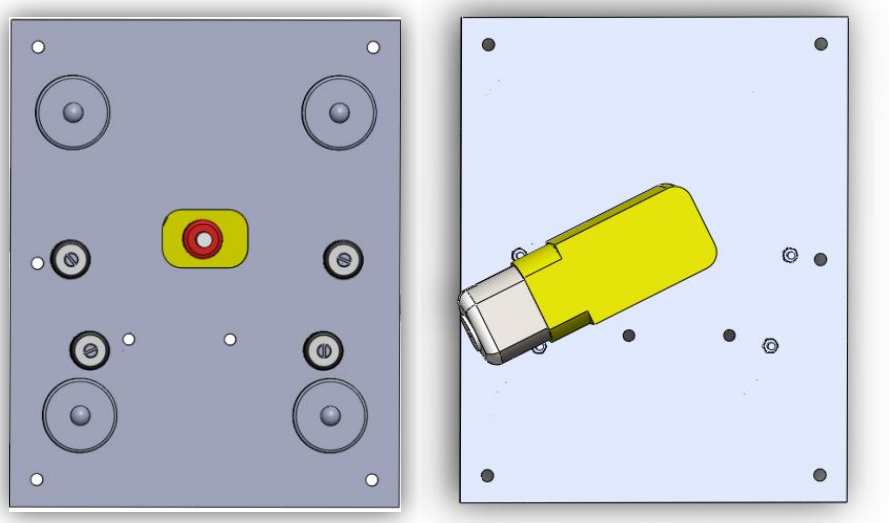


Figura III. 26 a) Vista inferior base móvil b) Vista superior base móvil

En la Figura III.27 se observa el mecanismo descrito anteriormente ya ensamblado en los rieles.

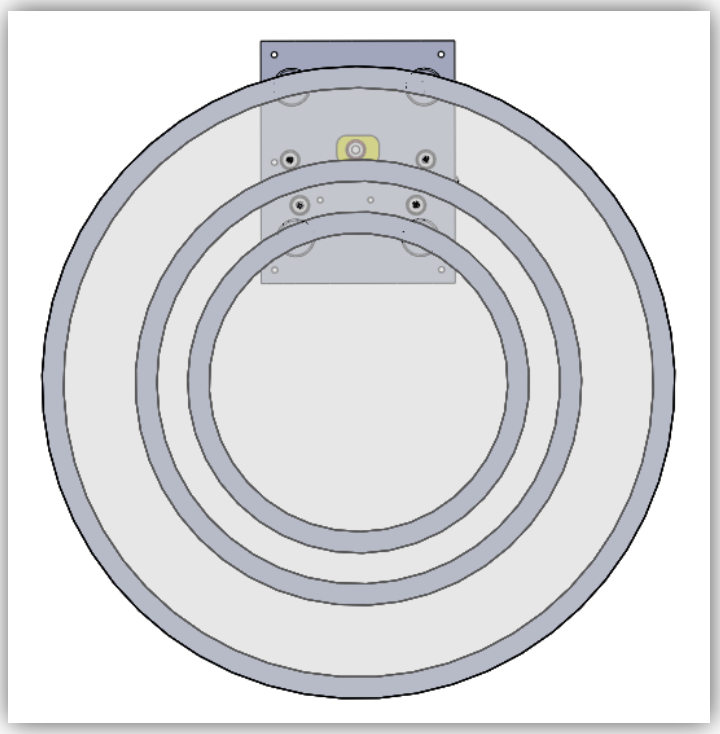


Figura III. 27 Base móvil y rieles

Durante el desarrollo de este trabajo el motor ensamblado en la base móvil se llamará Motor 3.

La Tabla III.6 básicamente resume la acción y funcionamiento que desempañará cada motor en el prototipo propuesto. Se muestra en la primera columna el número motor, en la siguiente columna la descripción del mismo, y finalmente la acción del mecanismo que se realiza.

Tabla III- 6 Descripción de motores

Motor	Descripción	Acción del mecanismo
1	Motor encargado de darle movimiento al mecanismo de la plataforma móvil.	Plataforma sale de la estructura móvil Plataforma entra a la estructura móvil
2	Motor encargado de darle movimiento al mecanismo de la cabina.	Cabina asciende Cabina desciende
3	Motor encargado de darle movimiento al mecanismo de la base móvil.	Estructura móvil gira levógiramente Estructura móvil gira dextrógiramente

Rieles

Los rieles dónde circulará el mecanismo del prototipo se diseñaron en una herramienta de CAD-CAE tomando como referencia la forma y dimensiones del perfil de aluminio que se observa en la Figura III.28, el cual es un perfil cuadrado, y sus paredes planas lo hacen ideal para la transmisión de movimiento de la llanta del Motor 3 al riel central.

Figura III. 28 Perfil de Aluminio para rieles

Como se muestra en la Figura III.29 existen tres rieles, de los cuales los dos externos servirán como soporte de la estructura móvil, y el central como guía. Al riel central se transmitirá el movimiento rotario del Motor 3 por medio de la llanta de plástico que tendrá acoplada a la flecha, y así se logrará que la estructura móvil cumpla con un movimiento de traslación curvilínea. El radio de los rieles se determinó tomando como referencia el tamaño de la estructura móvil, para que pudiera moverse con libertad por toda la estructura fija del prototipo.

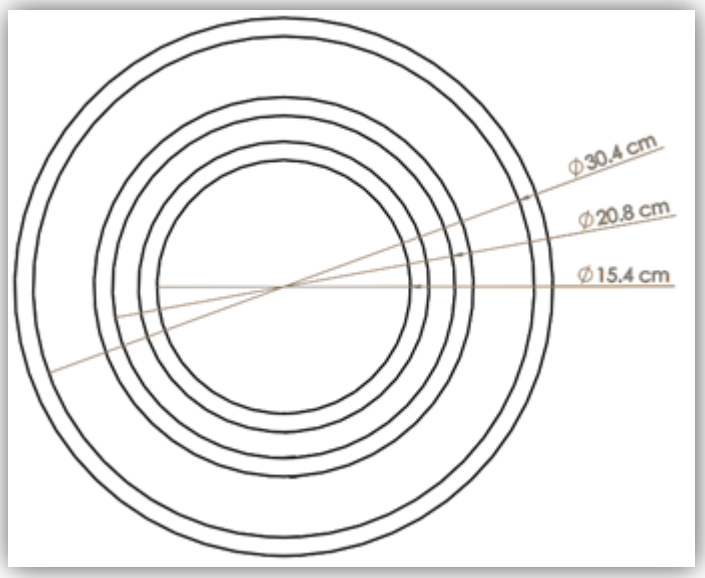


Figura III. 29 Diámetro de los rieles

En la Figura III.30 se observa el diseño final en la herramienta de CAD-CAE que se tomó como base para el desarrollo del prototipo, y como puede apreciarse, los rieles están montados sobre una lámina de Aluminio, la cual es calibre 18.

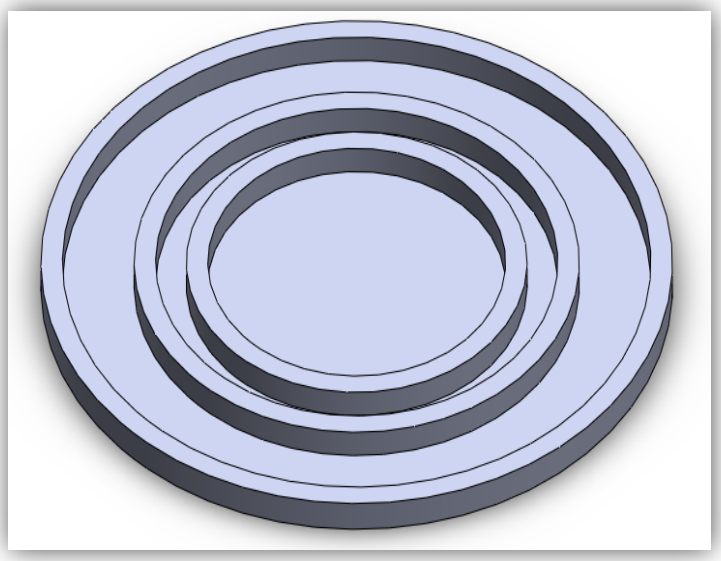


Figura III. 30 Rieles

Cuarto de acceso

Las dimensiones del cuarto de acceso de igual forma se definieron con base a la plataforma móvil, la Figura III.31 muestra la base fija del cuarto de acceso.

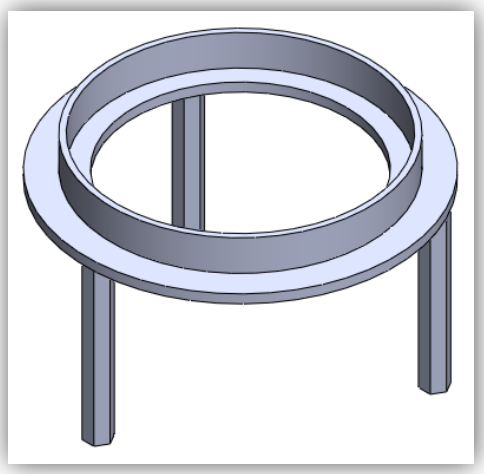


Figura III. 31 Base fija

La base giratoria del cuarto de acceso se puede observar en la Figura III.32, esta base es la encargada de cambiar el sentido del vehículo

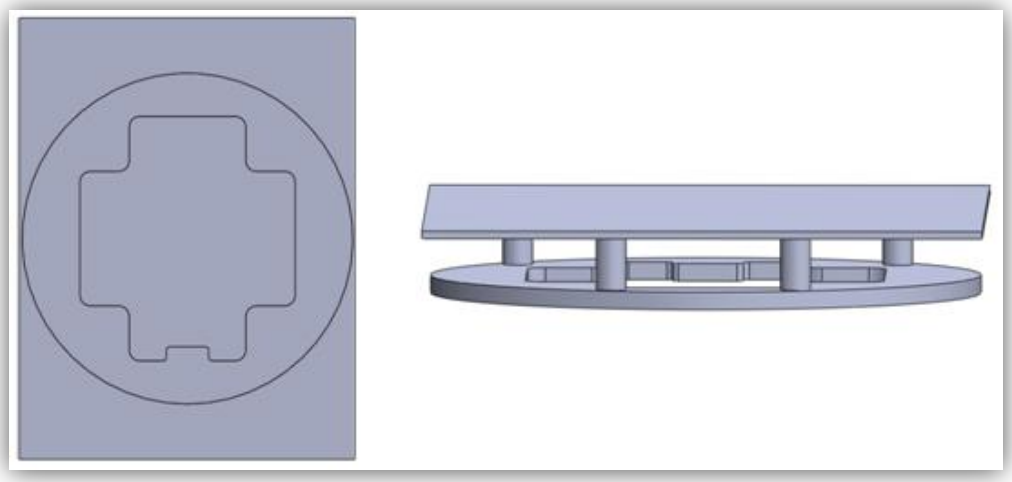


Figura III. 32 Base giratoria (Vista inferior Vista lateral respectivamente)

El cuarto de acceso en CAD-CAE se muestra en la Figura III. 33

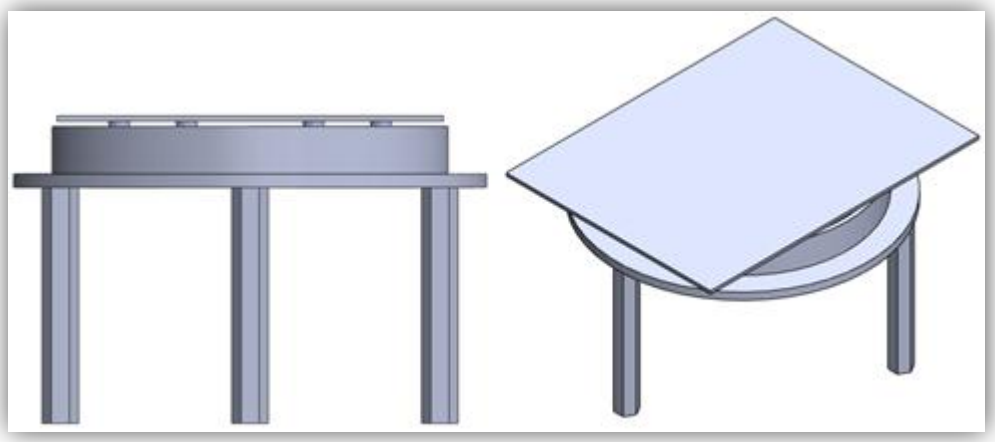


Figura III. 33 Vista lateral e isométrica del cuarto de acceso

Columnas

Estas piezas fueron diseñadas tomando como referencia el perfil de madera que se muestra en la Figura III.34 y el diseño de las columnas en herramienta de CAD-CAE se observa en la Figura III. 35

Figura III. 34 Perfil de madera

Estas columnas serán las que soportarán las rejillas de los cajones, y estarán distribuidas a lo largo del perímetro de los rieles.

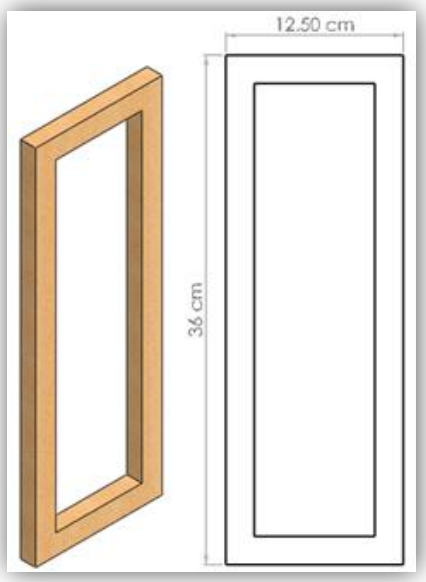


Figura III. 35 Dimensiones de columna

Cajones

Para demostrar el funcionamiento del sistema se habilitarán 4 cajones en el prototipo, y básicamente las rejillas que conforman los cajones emplean el mismo perfil de Aluminio de la plataforma móvil. La Figura III.36 muestra el diseño de las rejillas del cajón en la herramienta de CAD-CAE

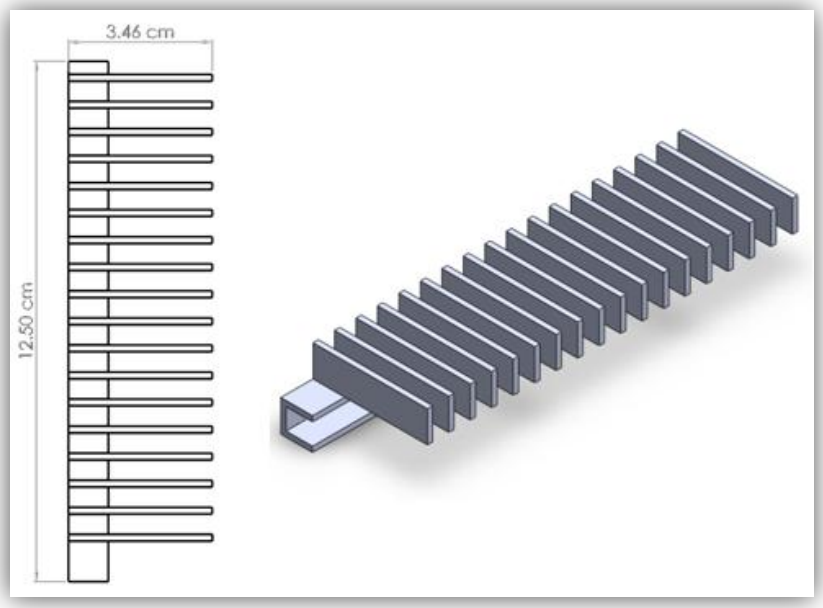


Figura III. 36 Rejillas de cajones (Dimensiones y vista isométrica)

Diseño de prototipo en herramienta de CAD-CAE

La estructura móvil y fija, descansan sobre una tabla de madera de 80 cm X 80 cm, y la Figura III.32 muestra el diseño en la herramienta de CAD-CAE del prototipo de estacionamiento automático propuesto.

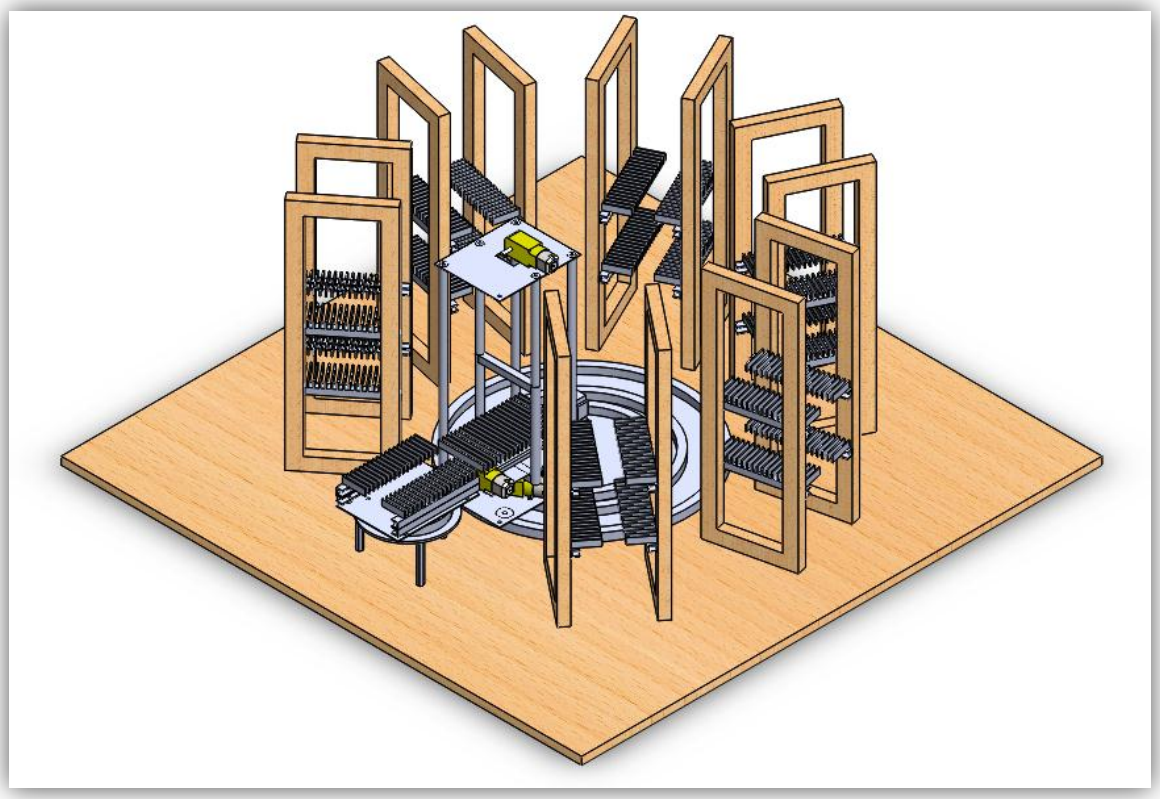


Figura III. 37 CAD-CAE de prototipo de estacionamiento completo

Tomando como base el diseño realizado y desarrollado en este apartado se dará inicio a la construcción del prototipo, y se describirá en las siguientes líneas.

III.3 Construcción del prototipo

A continuación se muestra el desarrollo de la construcción del prototipo tomando como referencia el diseño con la herramienta de CAD-CAE. Para cada pieza se describirá de una forma general el procedimiento de construcción.

Plataforma móvil

La Figura III.38 muestra de forma general el desarrollo de la construcción de la plataforma móvil.

Figura III. 38 Proceso de construcción plataforma móvil

En la Figura III.39 se pueden observar 3 vistas de la plataforma móvil, a) es la vita superior, b) la vista frontal y c) muestra una vista lateral. Y la Figura III.40 muestra fotografías de la plataforma móvil finalizada.

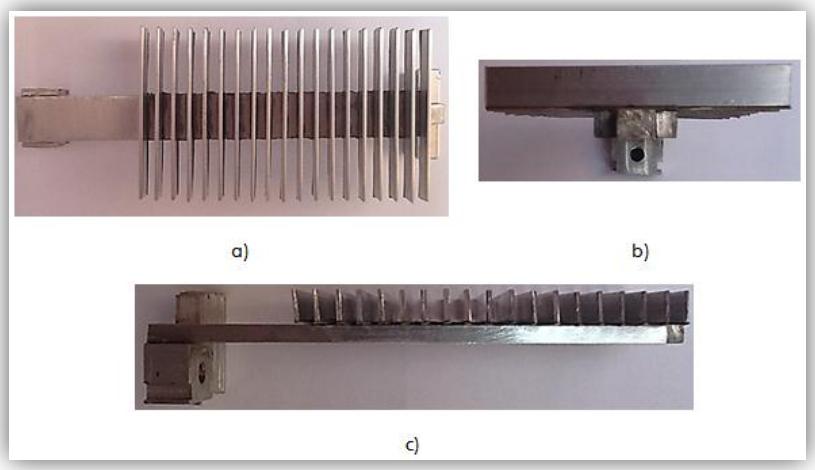


Figura III. 39 Vistas de la plataforma móvil

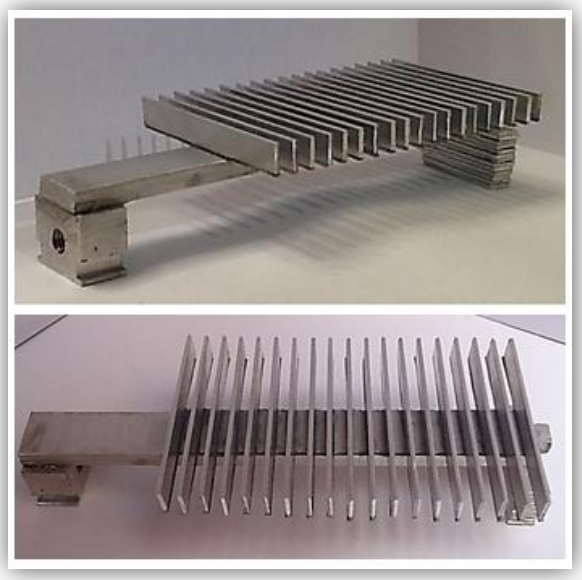


Figura III. 40 Plataforma móvil

Una vez terminada la construcción de la plataforma móvil se fijan el riel y el motor a una lámina de Aluminio calibre 18, la cual reposará en la cabina. En la Figura III. 41 se observa el riel y el motor fijos a la lámina y el tornillo sin fin acoplado a la flecha del motor por medio del conector.

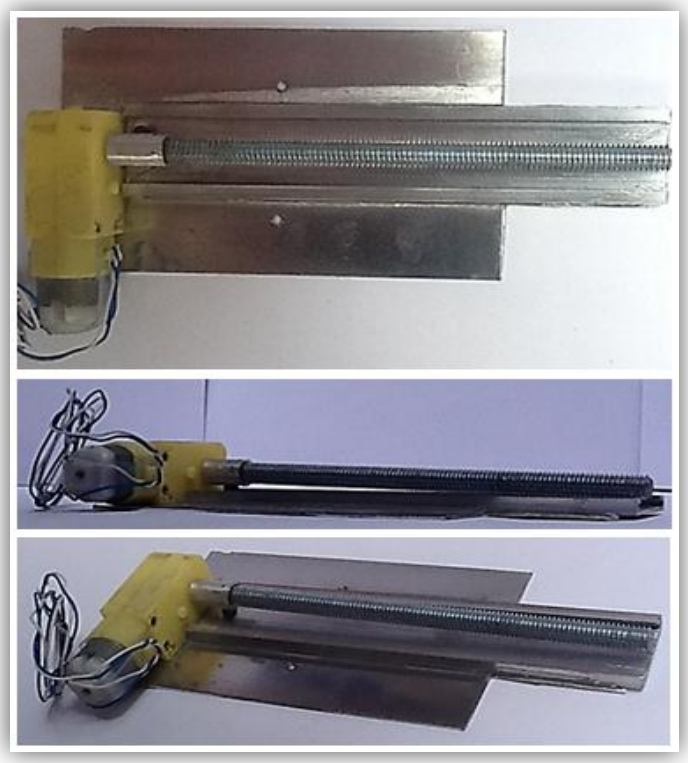


Figura III. 411 Motor 2, riel y sin fin

La Figura III.42 muestra el sin fin acoplado a la flecha del motor y al dado, además se observa la idea del vehículo reposando en la plataforma móvil.

Figura III. 42 Mecanismo de plataforma móvil

Cabina

La Figura III.43 muestra el proceso de construcción de la cabina.

Figura III. 43 proceso de construcción de cabina

Ya soldadas las cuatro piezas, en la Figura III.44 se muestra la imagen de la cabina finalizada.

Figura III. 44 Cabina

Posteriormente se cortan los ejes que soportarán dicha cabina, y en la Figura III.45 se muestran dichas piezas. Y como se observa, los ejes tienen un orificio, lo cuales permiten que los ejes se acoplen por medio de tornillos a la base móvil y a la parte superior (techo) del mecanismo.

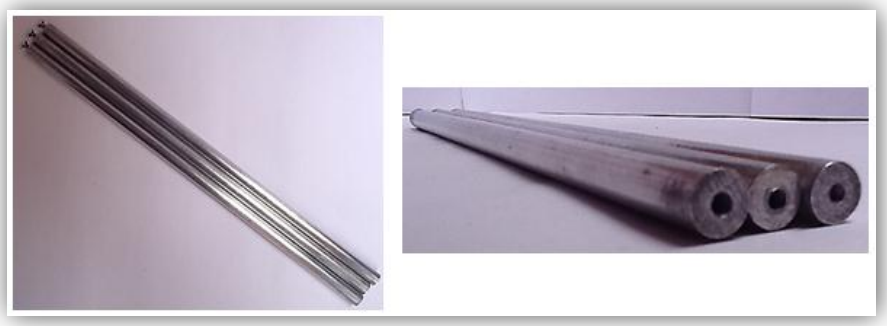


Figura III. 45 Ejes

Como se mencionó anteriormente la cabina será desplazada por medio del Motor 2, y esto se logrará mediante una banda que estará acoplada de un extremo a la flecha del motor y del otro a la cabina. Para acoplarla a la cabina se utilizó una armella que se colocó en la parte central de la cabina, para que posteriormente la banda pudiera soportarse de ese punto. La Figura III.46 muestra la cabina con la armella y banda colocadas.

Figura III. 46 Armella de cabina

En la Figura III.47 se puede observar en la parte inferior a la cabina, ya acoplada a los ejes ya fijos en la parte superior del mecanismo. El Motor 2 se fijó en una lámina de Aluminio calibre 18 y se observa la banda acoplada a la flecha y a la cabina.

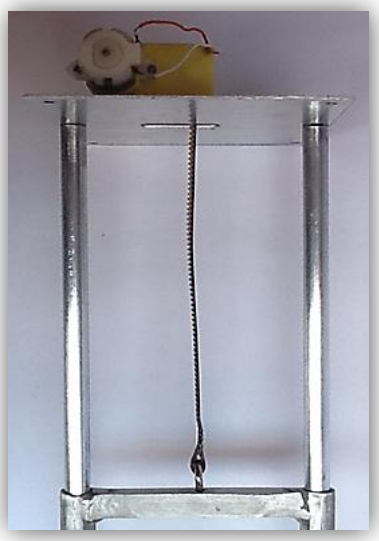


Figura III. 47 Mecanismo de la cabina

La Figura III.48 muestra el Motor 2 acoplado a la parte superior del mecanismo con la banda ya fija.

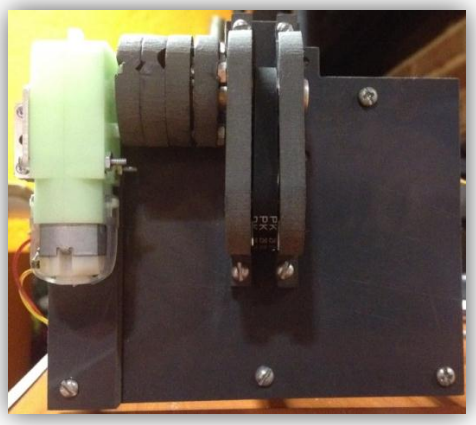


Figura III. 48 Motor 2 en parte superior del mecanismo

Rieles

En la Figura III.49 se observa el desarrollo de la construcción de los rieles

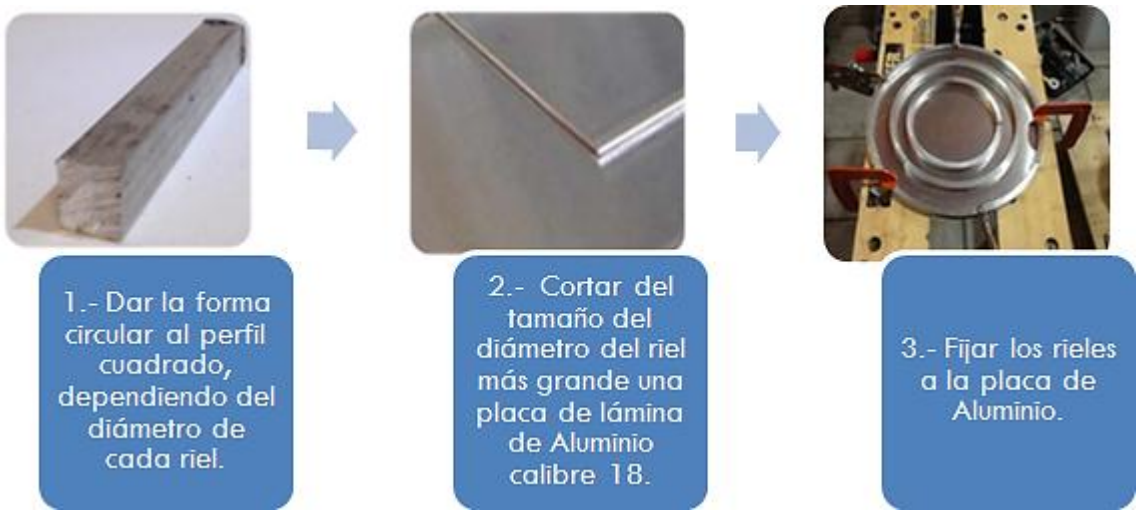


Figura III. 49 Proceso de construcción de los rieles

La Figura III.50 muestra los rieles ya fijos en la placa de Aluminio.

Figura III. 50 Rieles

Base móvil

En la Figura III.51 muestra el desarrollo en la construcción de la base móvil.

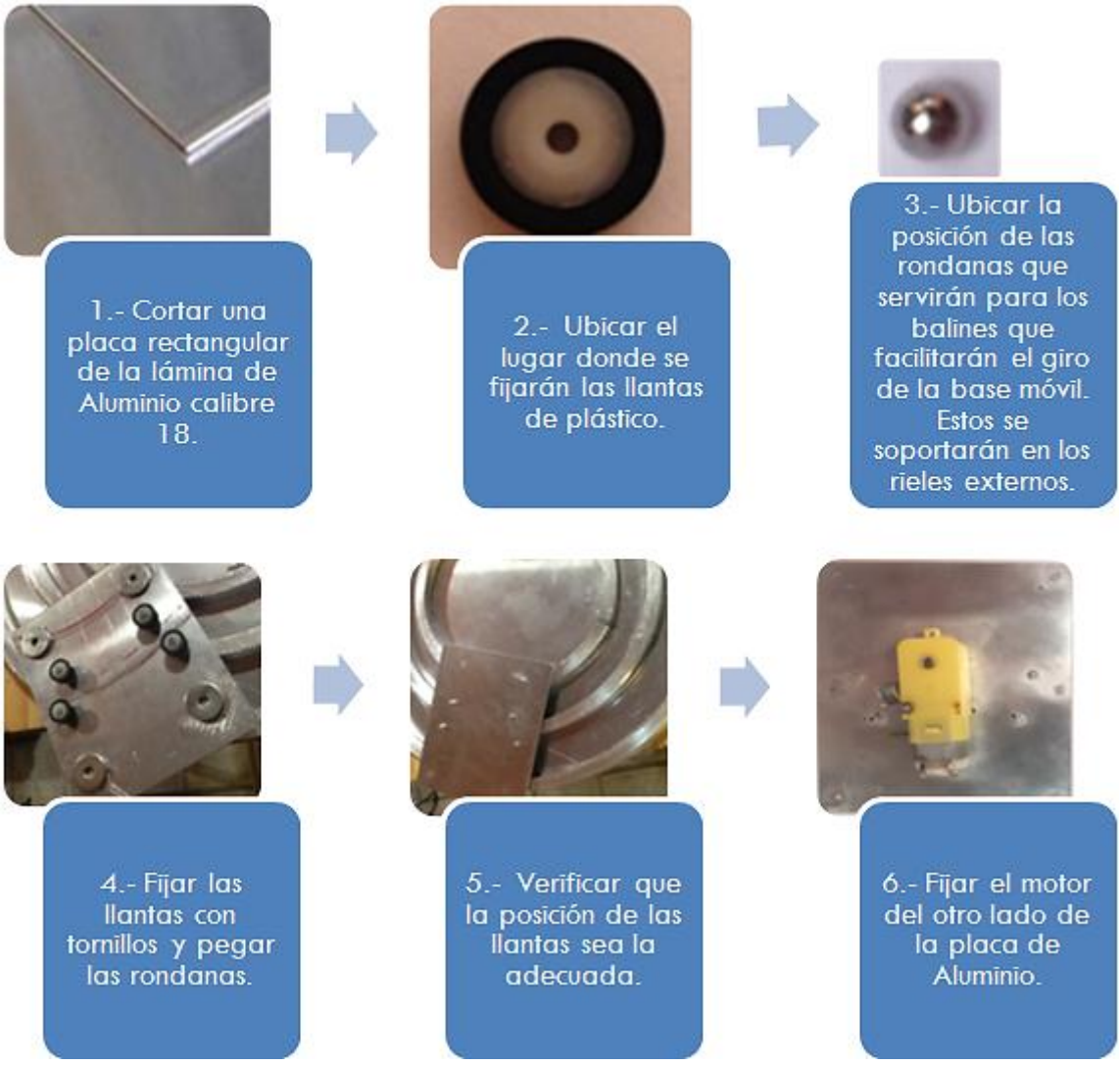


Figura III. 51 Proceso de construcción de base móvil

En la Figura III.52 se observa la vista superior e inferior de la base móvil, así como unos orificios en cada esquina de la placa de Aluminio, que servirán para fijar los ejes.

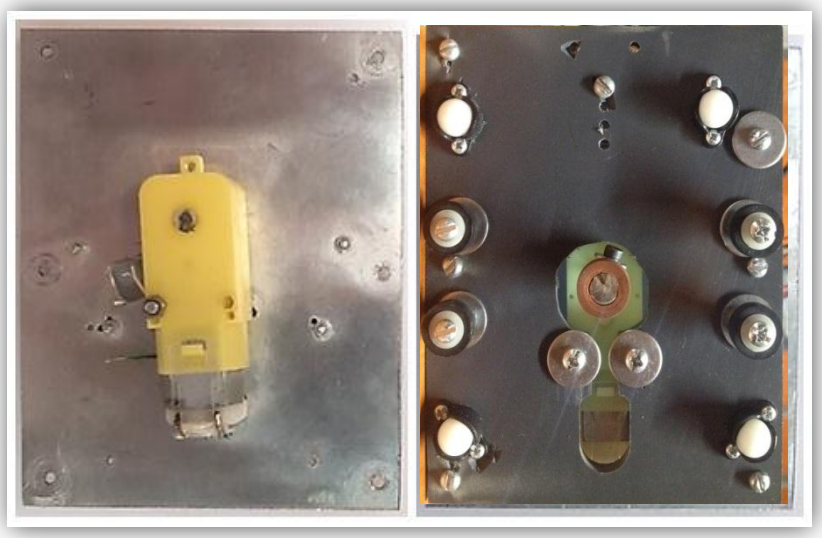


Figura III. 52 Vista superior e inferior base móvil

Cuarto de acceso

Para construir el cuarto de acceso se ocupó una pieza de una videocámara la cual lo único que servía era el mecanismo que funcionaba a base de un motor a pasos con la flecha acoplada a la base del mecanismo y básicamente el principio de funcionamiento de ese mecanismo es el mismo que el de la base móvil. La Figura III.53 muestra el proceso de construcción del cuarto de acceso.

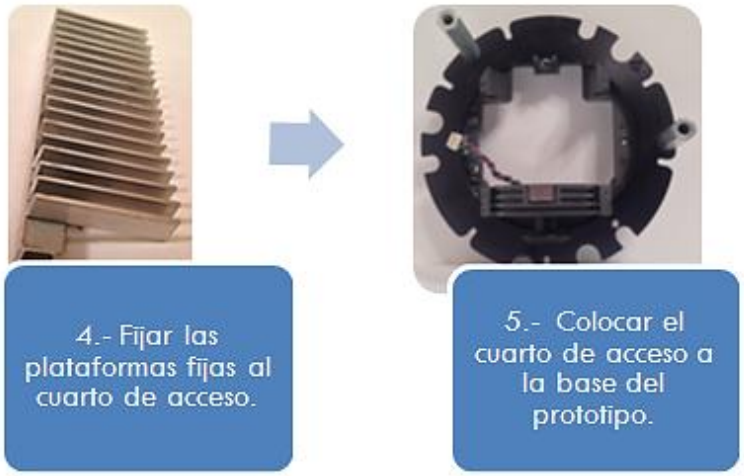


Figura III. 53 Proceso de construcción del cuarto de acceso

En la Figura III.54 se muestra la vista inferior y superior del mecanismo del cuarto de acceso.

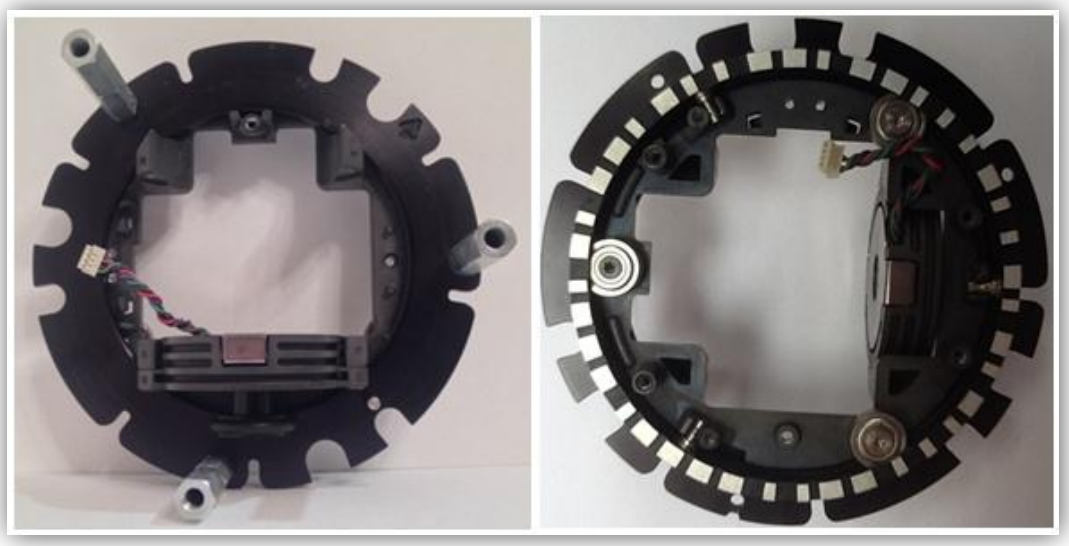


Figura III. 54 Vista inferior y superior mecanismo cuarto de acceso

Columnas

La Figura III.55 muestra el desarrollo de la construcción de las columnas

Figura III. 55 Proceso de construcción de las columnas

La Figura III.56 muestra a) la vista frontal de la columna b) la vista superior e inferior y c) la vista lateral.

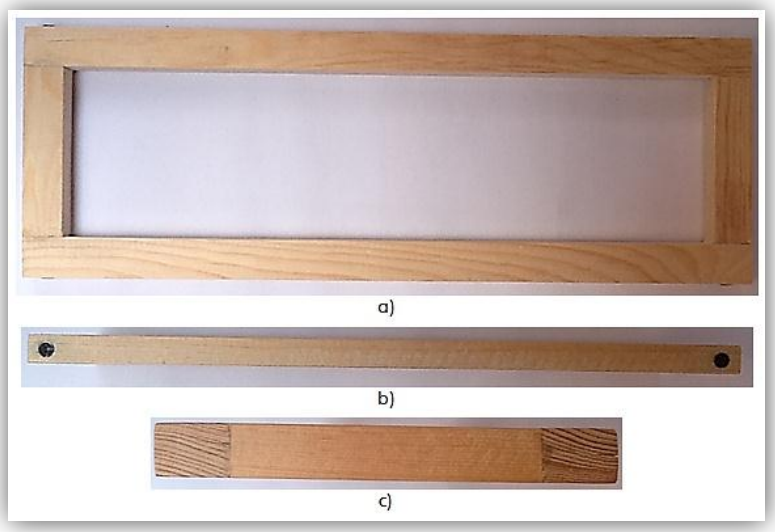


Figura III. 56 Vistas de columna

Una vez ensambladas las piezas de madera se procede a fijarlas en la base del prototipo, y esto se observa en la Figura III.57, en la cual se puede apreciar cómo se colocan en todo el perímetro donde estarán colocados los rieles.

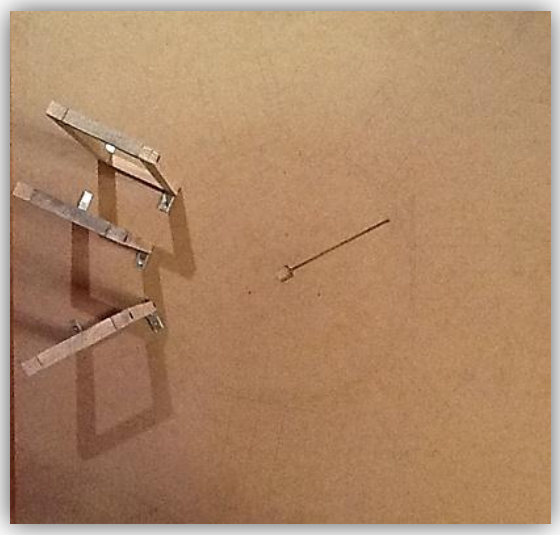


Figura III. 57 Columnas en base de prototipo

Cajones

En la Figura III.58 se muestra el desarrollo de la construcción de los cajones.

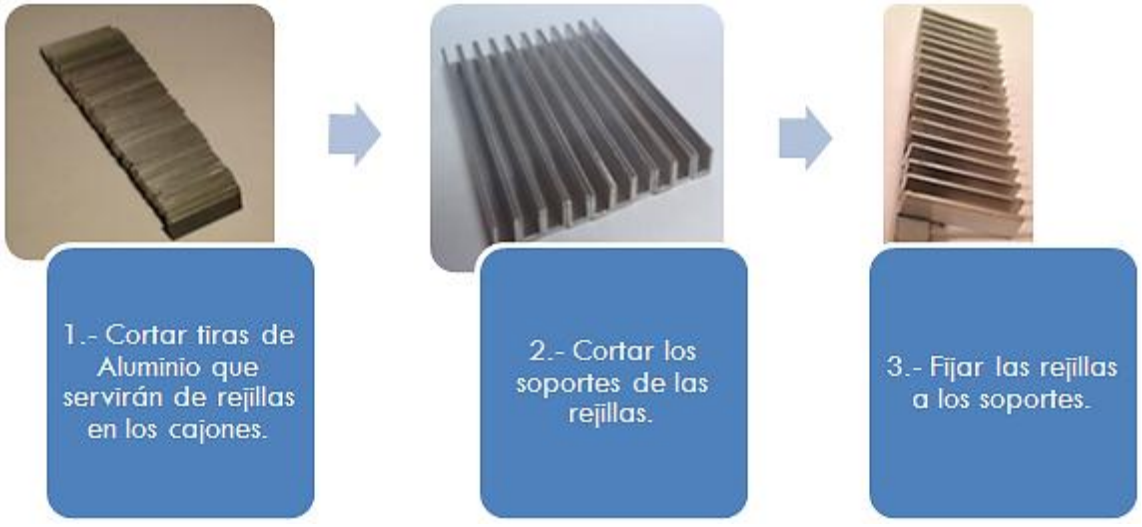


Figura III. 58 Proceso de construcción de los cajones

La Figura III.59 muestra en a) la vista superior de la plataforma fija y en b) la vista lateral.

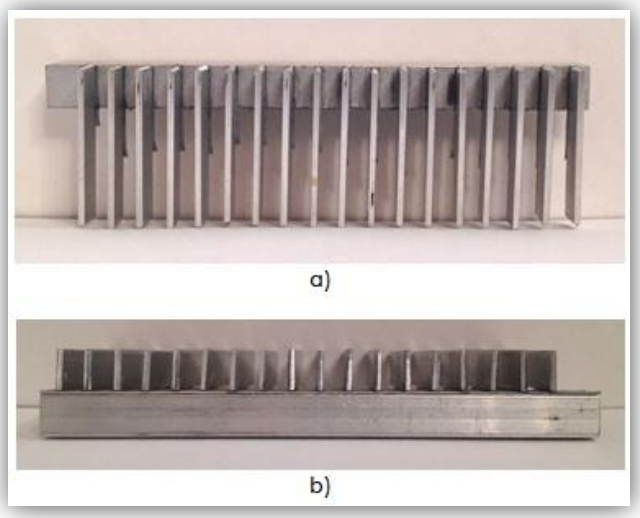


Figura III. 59 Vista superior y lateral de plataformas fijas

En la Figura III.60 se observan las plataformas fijas que se colocarán en las columnas para así tener los cuatro cajones con los que cuenta el prototipo.

Figura III. 60 Plataformas fijas de cajones y cuarto de acceso

Capítulo IV Diseño de la lógica de control

IV.1 Introducción

Este capítulo está destinado a plantear la conexión del hardware, cuya finalidad es exponer los elementos o dispositivos empleados en la electrónica de control y de potencia de los microcontroladores, para posteriormente describir sus códigos de programación, pero no sin antes explicar los entornos de programación utilizados tanto para las tarjetas Arduino como para los módulos Xbee.

En el diagrama de bloques que se puede apreciar en la Figura IV.1 se explica el funcionamiento del sistema. En tal diagrama, se encuentran representados los microcontroladores Arduino. Por un lado se observa el microcontrolador Arduino Mega 2560 con sus respectivas entradas y salidas, así como su módulo Xbee; por el otro lado está el microcontrolador Leonardo, al igual que el anterior con sus respectivas entradas y salidas, y su módulo Xbee. Para su análisis, es posible estructurar en dos bloques generales este diagrama, donde uno está representado por el Arduino Mega y el otro por el Arduino Leonardo.

El primero va a ser el encargado de gestionar todos los elementos o dispositivos que se encuentran en la estructura fija del prototipo, como el panel de control, el semáforo y los sensores IR; el segundo, está ubicado en la estructura móvil para gestionar los accionamientos de los motores y el monitoreo de los interruptores de límite. Estos bloques están en constante comunicación gracias a los módulos Xbee.

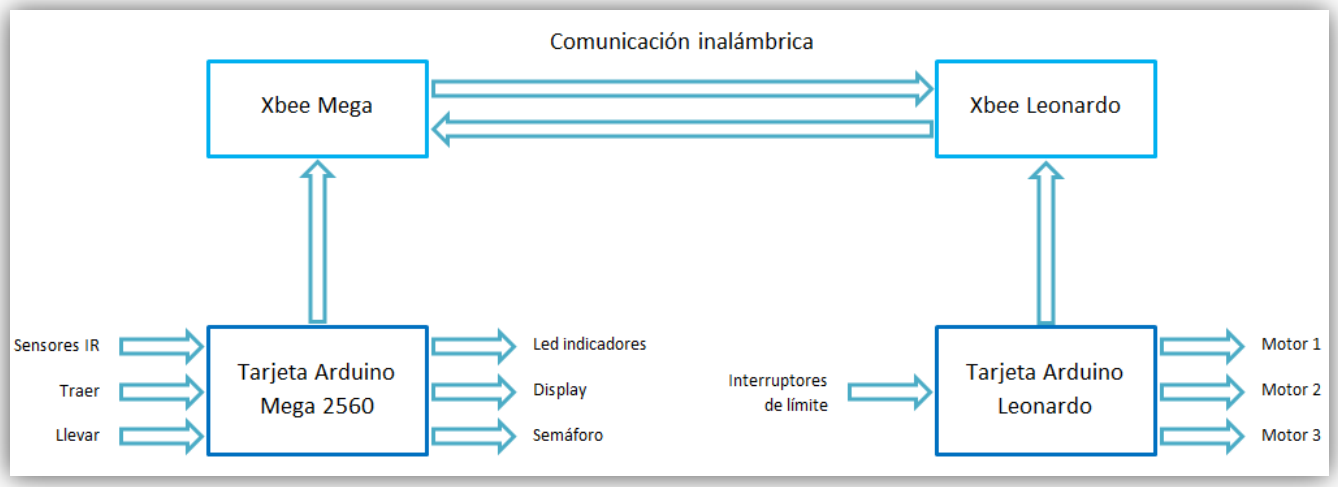


Figura IV. 1 Diagrama de bloques general esquemático

En la Figura IV.2 se observa el mismo contenido que en el diagrama de bloques anterior pero ahora con los dispositivos reales del prototipo.

IV.2 Electrónica de control y de potencia

Como se observó en el diagrama de bloques de la Figura IV.1 existen dos bloques principales en el sistema, estos bloques están compuestos por dispositivos físicos como sensores IR, leds, leds IR, micropulsadores y motores, los cuales pueden estar conectados de forma directa o indirecta a las tarjetas Arduino. Para poder hacer esas conexiones se necesita de la etapa de control y de potencia para cada microcontrolador, que es lo que se desarrolla en las líneas siguientes.

IV.2.1 Arduino Mega 2560

En el diagrama elaborado con imágenes que se muestra en la Figura IV.2 se pueden apreciar todos los elementos físicos conectados a la tarjeta Arduino Mega 2560. Estos elementos son:

– Receptores infrarrojos. Son fototransistores que al momento de que un haz de luz infrarroja incide sobre su base, este se satura actuando como un interruptor.

– Leds infrarrojos. Diodos emisores de luz infrarroja.

– Smith trigger. Circuito integrado CMOS (74HCT14) que usa la histéresis para prevenir el ruido que podría interferir en la señal original y que causaría falsos cambios de estado si los niveles de referencia y entrada son parecidos. La hoja de datos de este circuito integrado se encuentra en el Anexo A.

– Micropulsadores. Interruptores electrónicos momentáneos N/A.

– Shield de XBee. Permite a una placa Arduino comunicarse de forma inalámbrica usando el protocolo de comunicación ZigBee.

– XBee. Modulo para comunicación inalámbrica que utiliza el protocolo ZigBee. La hoja de datos de este dispositivo se encuentra en el Anexo B.

– Leds RG. Encapsulado en donde se contienen dos diodos emisores de luz, rojo y verde. Y que al iluminarse los dos leds simultáneamente, se puede observar en el encapsulado translúcido el color naranja.

– Semáforo. Dispositivos de señales que se sitúan en intersecciones viales, pasos de peatones y otros lugares para regular el tráfico y el tránsito de peatones.

Cabe mencionar que el esquema de la Figura siguiente es únicamente una representación simbólica para denotar los elementos que están conectados al microcontrolador Arduino Mega. Se considera que la imagen de la tarjeta Mega que se muestra en la Figura IV.2 es un bloque, y las flechas en color negro determinan si los elementos conectados son entradas o salidas.

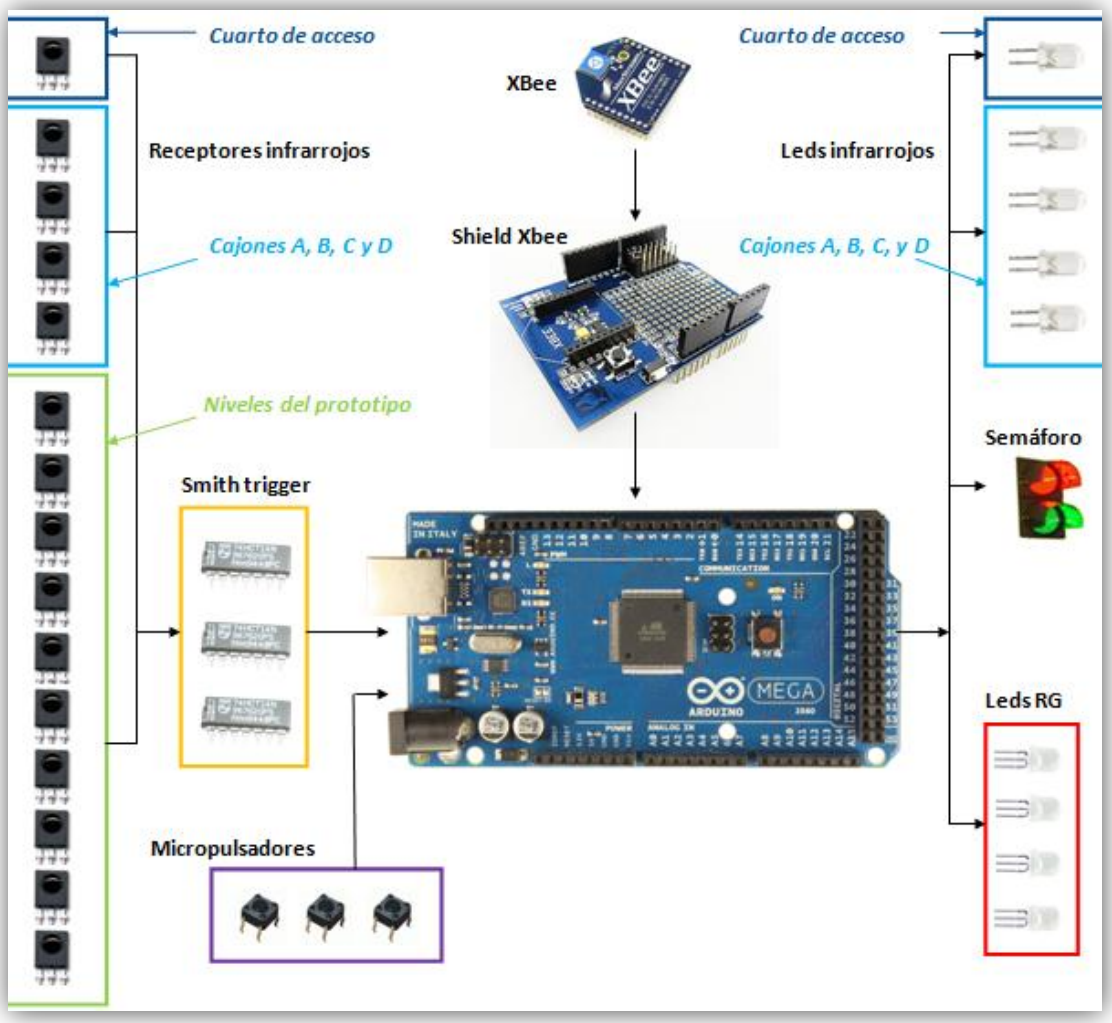


Figura IV. 2 Diagrama de bloques con fotografías del hardware (Arduino Mega 2560)

Diagrama físico de conexión

En el diagrama de conexión del microcontrolador Arduino Mega que se presenta en la Figura IV.3 se observa que los leds RG, no se encuentran en un encapsulado ya que el software con el que se desarrolló no cuenta con esos dispositivos.

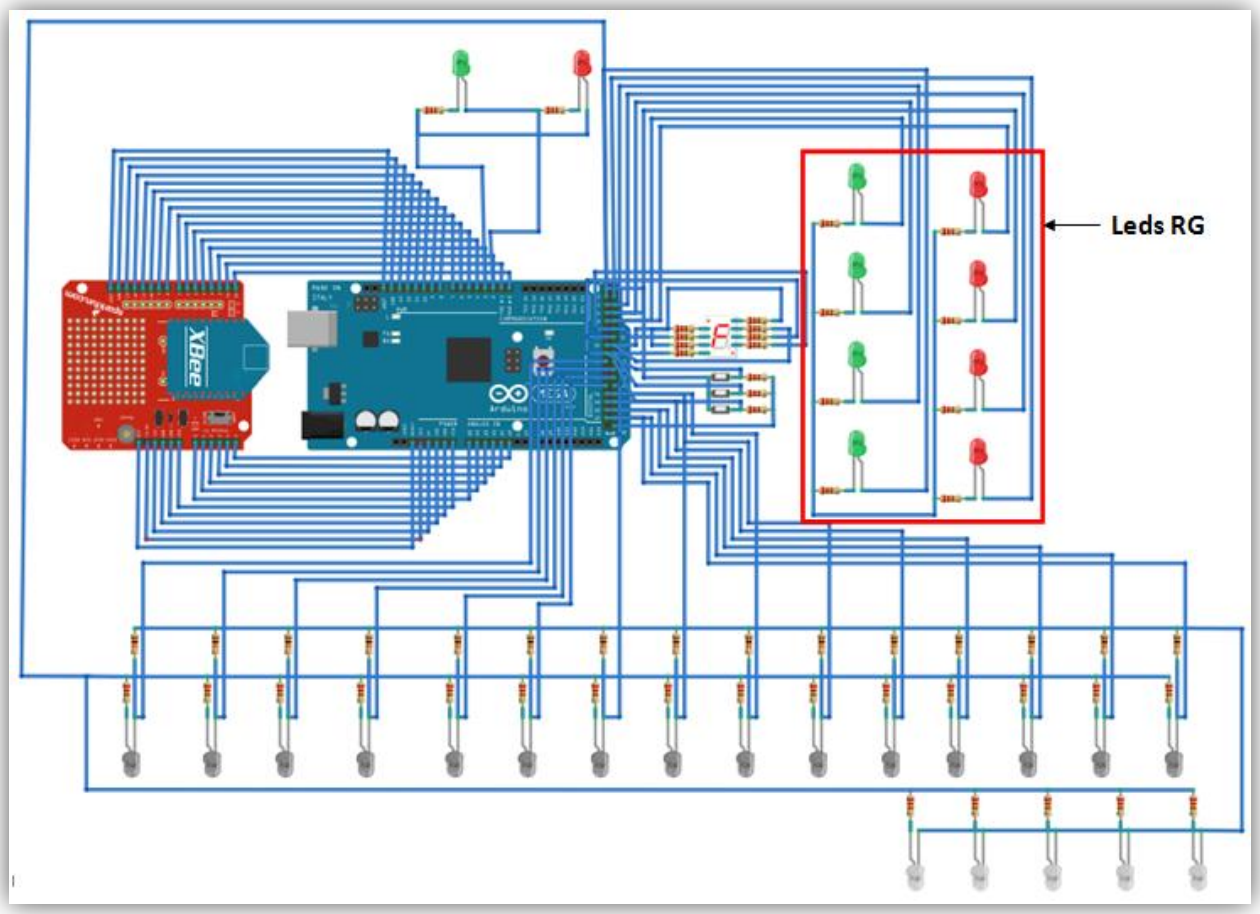


Figura IV. 3 Diagrama físico de conexión del Arduino Mega 2560

Diagrama electrónico de conexión

El diagrama electrónico del microcontrolador Arduino Mega se presenta en la Figura IV.4, el cual fue desarrollado en el software ISIS Professional, y en la Figura IV.5 se observa el diagrama de conexión de la PCB.

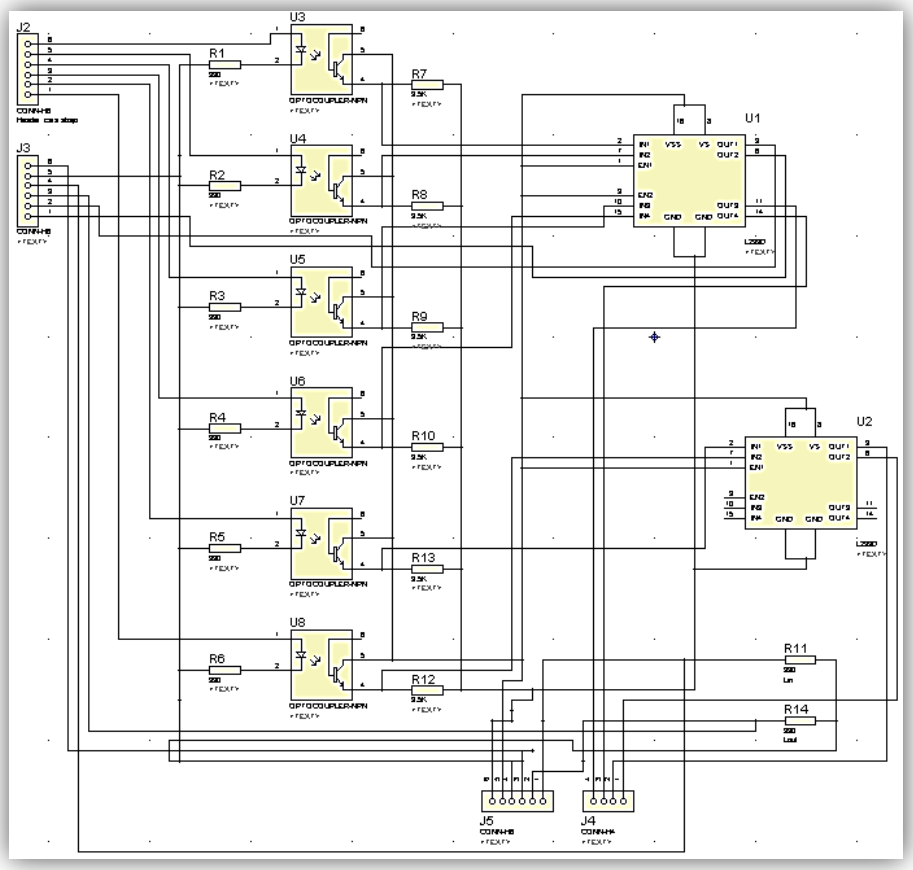


Figura IV. 4 Diagrama electrónico de conexión Arduino Mega 2560

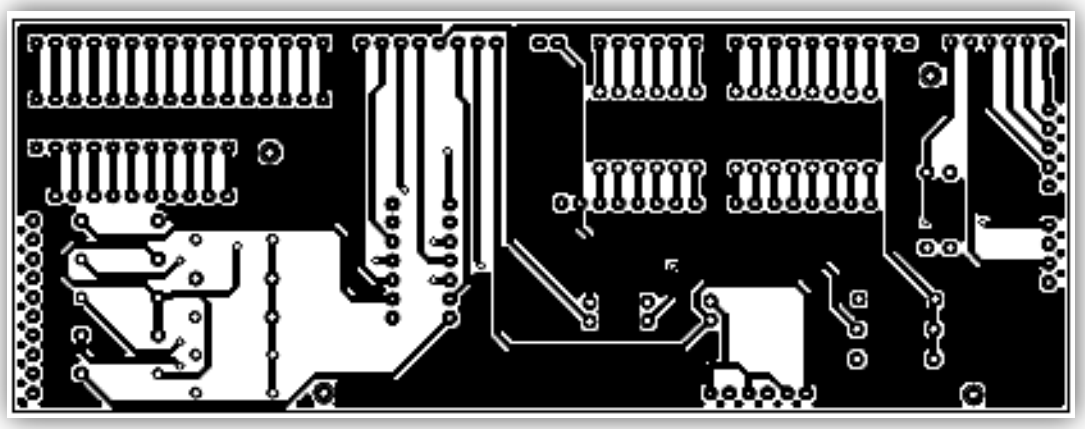


Figura IV. 5 Diagrama de PCB Arduino Mega 2560

IV.2.2 Arduino Leonardo

En el diagrama elaborado con imágenes que se muestra en la Figura IV.6 se pueden apreciar todos los elementos físicos conectados a la tarjeta Arduino Leonardo. Estos elementos son:

- Led infrarrojo. Diodo emisor de luz infrarroja.
- Shield de XBee. Permite a una placa Arduino comunicarse de forma inalámbrica usando el protocolo de comunicación ZigBee.
- XBee. Modulo para comunicación inalámbrica que utiliza el protocolo ZigBee.
- Interruptores de límite. También conocidos como final de carrera o sensores de contacto, son dispositivos eléctricos situados al final del recorrido de un elemento móvil, con el objetivo de enviar señales que puedan modificar el estado del circuito. Internamente contienen interruptores normalmente abiertos (NA) o normalmente cerrados (NC).
- Optoacopladores (4N25). Son dispositivos que aíslan circuitos electrónicos; en su interior contienen un led que emite un haz de luz, que satura un componente optoelectrónico como un fototransistor o fototriac. La hoja de datos de este circuito integrado se encuentra en el Anexo C.
- Puentes H (L293B). Son circuitos integrados que incorporan dos drivers en su interior. Estos drivers son puentes H, los cuales sirven para controlar el sentido de giro de motores de CD. Estos drivers tienen la ventaja de poder ser alimentados con dos fuentes diferentes, una para la lógica de control y la otra para la etapa de potencia. La hoja de datos de este circuito integrado se encuentra en el Anexo D.
- Motor reductor. Motores de CD que en la flecha tienen acoplados cierto número de engranes con la finalidad de reducir el número de vueltas del motor, además de aumentar el par o torque del mismo.

Cabe mencionar que el esquema de la Figura siguiente es únicamente una representación simbólica para denotar los elementos que están conectados al microcontrolador Arduino Leonardo. Se considera que la imagen de la tarjeta Leonardo que se muestra en la Figura IV.6 es un bloque, y las flechas en color negro determinan si los elementos conectados son entradas o salidas.

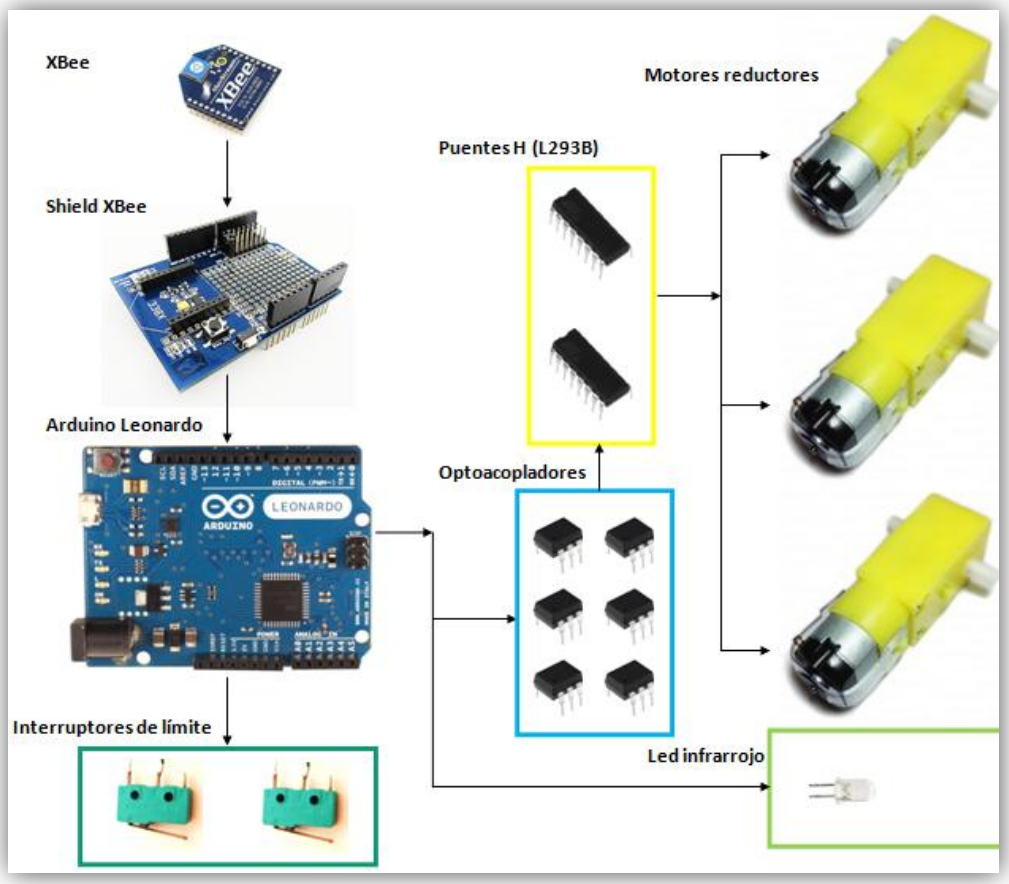


Figura IV. 6 Diagrama de bloques con fotografías del hardware (Arduino Leonardo)

Diagrama físico de conexión

La batería de 9V que se aprecia en la Figura IV.5 es una representación de la alimentación de la etapa de potencia. La alimentación de la etapa de control de este circuito es suministrada por la tarjeta Arduino Leonardo.

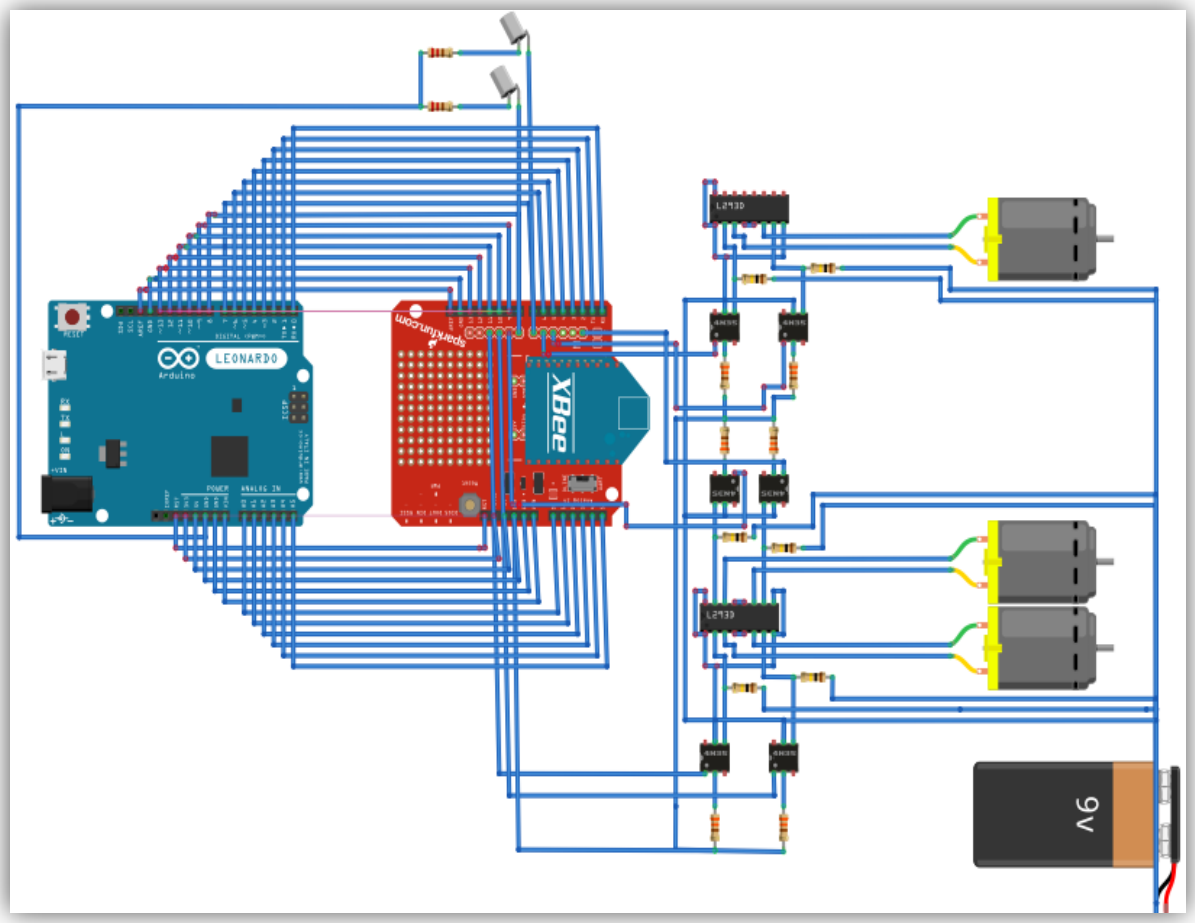


Figura IV. 7 Diagrama físico de conexión Arduino Leonardo

Diagrama electrónico de conexión

El diagrama electrónico del microcontrolador Arduino Leonardo se muestra en la Figura IV.8, el cual fue desarrollado en el software ISIS Professional, y en la Figura IV.9 se observa el diagrama de conexión de la PCB.

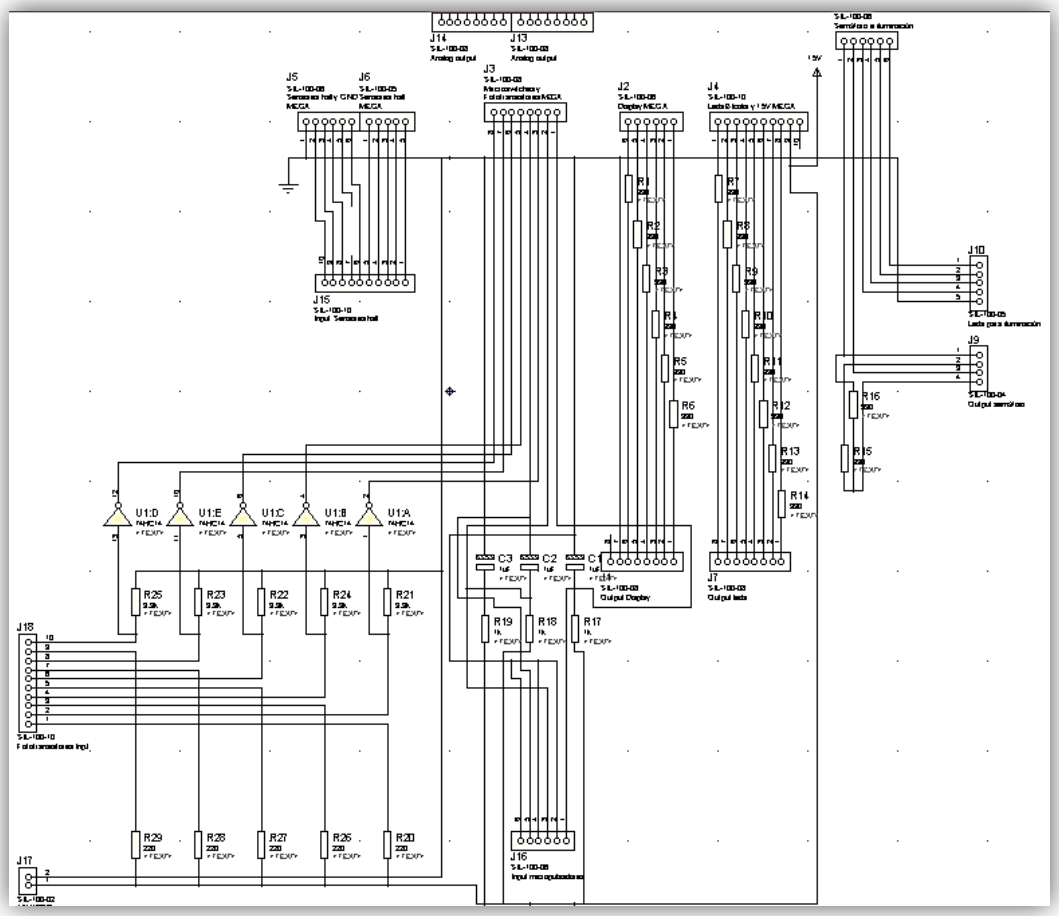


Figura IV. 8 Diagrama electrónico de conexión Arduino Leonardo

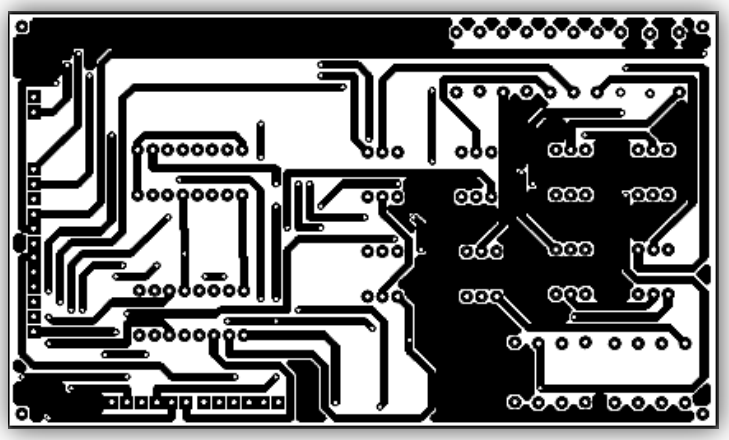


Figura IV. 9 Diagrama de PCB Arduino Leonardo

IV.3 Descripción del entorno de programación Arduino

El entorno de Desarrollo Arduino está constituido por un editor de texto para escribir el código, un área de mensajes, una consola de texto, una barra de herramientas con botones para las funciones comunes, y una serie de menús, como se muestra en la Figura IV.10. El entorno permite la conexión con el hardware de Arduino para cargar los programas y comunicarse con ellos.

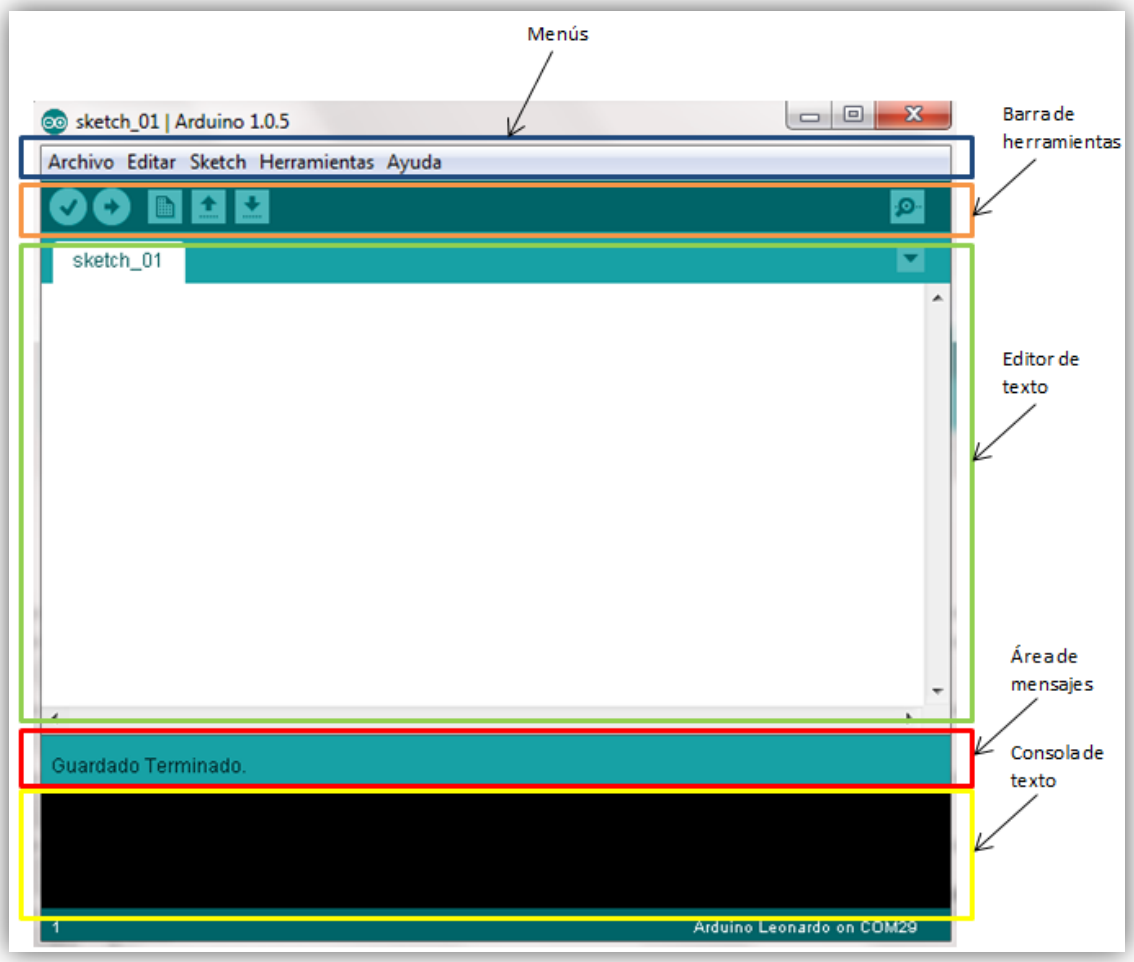


Figura IV. 10 Entorno de desarrollo arduino

Arduino utiliza para escribir un software que denomina "sketch" (programa). Estos programas son escritos en el editor de texto. En el área de mensajes se muestra información mientras se cargan los programas y también muestra errores. La consola muestra el texto de salida para el entorno de Arduino incluyendo los mensajes de error completos y otras informaciones. La barra de herramientas permite verificar el proceso de carga, creación, apertura y guardado de programas, y la monitorización serie. En la Tabla IV.1 se describen los botones de la barra de herramientas.

Tabla IV. 1 Barra de herramientas

	*Verificar* Chequea el código en busca de errores.
	*Nuevo* Crea un nuevo sketch.
	*Abrir* Presenta un menú de todos los programas sketch de su "sketchbook", (librería de sketch).
	*Guardar* Guarda el sketch (programa).
	*Cargar* Compila el código y lo carga en la placa E/S de Arduino.
	*Serial Monitor* Inicia la monitorización serie.

El entorno cuenta con otros cinco menús: *Archivo*, *Editar*, *Sketch*, *Herramientas*, *Ayuda*. Los menús son sensibles al contexto, lo que significa que estarán disponibles sólo los elementos relevantes para la tarea que esté realizando en ese momento. Solo se describirán los menús *Sketch* Figura IV.11 *y Herramientas* Figura IV.12, que son los más relevantes:

Sketch

- *Verificar/Compilar*
  Verifica los errores de su programa (sketch)
- *Importar librería…*

Añade una librería al programa (sketch) insertando la sentencia #include en el código.

- *Mostrar la carpeta de Sketch*

  Abre la carpeta de programas (sketch) en el escritorio.

- *Agregar archivo...*

  Añade un fichero fuente al programa (se incluirá desde su ubicación actual). El fichero aparece en una nueva pestaña en la ventana del programa. Los ficheros pueden ser quitados del programa (sketch) utilizando el menú "tab".

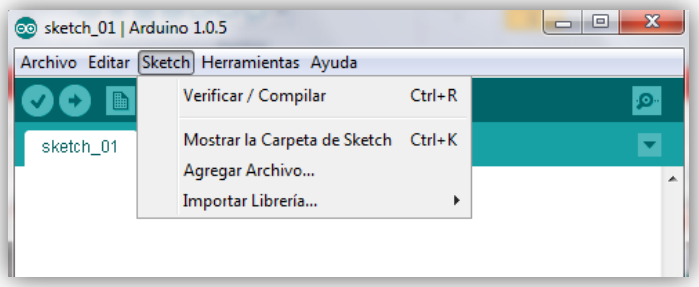


Figura IV. 11 Menú Sketch

Herramientas

- *Formato automático*

  Da formato al código proporcionando (estética): por ejemplo realiza tabulaciones entre la apertura y cierre de llaves, y da tabulación a las sentencias que tengan que ser tabuladas.

- *Tarjeta*

  Este menú es en donde se selecciona la placa arduino que se está usando.

- *Puerto serial*

  Este menú contiene todos los dispositivos serie (reales o virtuales) del equipo. Se refrescará automáticamente cada vez que se abra el menú herramientas.

- *Monitor serial*

  Muestra los datos enviados desde la placa Arduino (placa USB o serie).

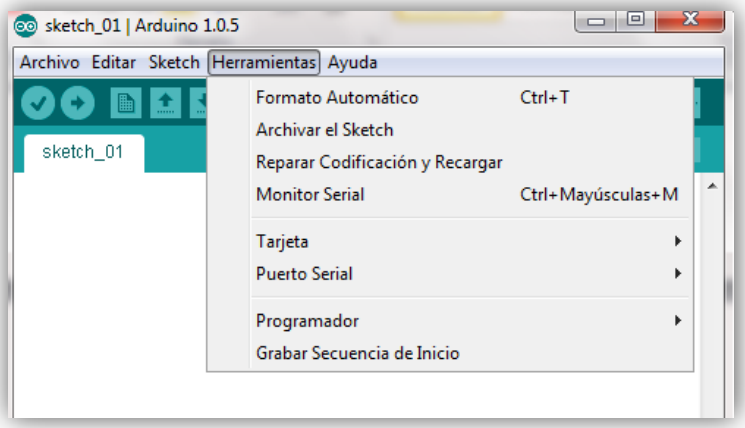


Figura IV. 12 Menú Herramientas

Para escribir un programa en el entorno de desarrollo Arduino se ocupa un lenguaje de programación exclusivo de Arduino, el cual está basado en el lenguaje de programación de alto nivel como C/C++. Las instrucciones de este lenguaje son diversas, pero para este proyecto fueron empleadas solo algunas, las cuales se describirán a continuación.

Descarga

Antes de descargar un "sketch" a la placa Arduino, se necesitan configurar los elementos correspondientes desde los menús *Herramientas>Tarjeta>Puerto serial*. En Windows el puerto serie, probablemente sea COM1 o COM2 (para una placa serie) o COM4, COM5, COM7, o superior (para una placa USB).

Una vez que se ha seleccionado el puerto serie y la placa, se tiene que presionar el botón de carga en la barra de herramientas o seleccionar desde el menú File. Las actuales placas de Arduino se resetearán automáticamente y comenzará la descarga del programa a la tarjeta. Como las placas antiguas carecen de auto-reset, se necesita presionar el botón de reset en la placa, justo antes de iniciar la descarga. En muchas placas el led RX y TX parpadea cuando el "sketch" está actualizándose. El entorno de Arduino mostrará un mensaje cuando la descarga esté completa, o mostrará un error de no ser así.

IV.3.1 Descripción de las instrucciones de Arduino

A continuación se describen las principales instrucciones empleadas para la programación de las tarjetas Arduino Mega 2560 y Leonardo

setup()

La función setup() se establece cuando se inicia un programa. Se emplea para iniciar variables, establecer el estado de los pins, inicializar librerías, etc. Esta función se ejecutará una única vez después de que se conecte la placa Arduino a la fuente de alimentación, o cuando se pulse el botón de reinicio de la placa.

loop()

Luego de crear la función setup(), la cual inicializa y prepara los valores iniciales, la función loop() hace justamente lo que su nombre sugiere, por lo tanto se ejecuta consecutivamente, permitiéndole al programa variar y responder.

If

Puede ser usado en conjunto con uno o más operadores de comparación, comprueba si cierta condición se cumple, por ejemplo, si un input posee un valor mayor a cierto número.

switch / case

Controla el flujo de programas permitiendo a los programadores especificar diferentes códigos que deberían ser ejecutados en función de varias condiciones. En particular, una sentencia switch compara el valor de una variable con el valor especificado en las sentencias case. Cuando se encuentra una sentencia case cuyo valor coincide con dicha variable, el código de esa sentencia se ejecuta.

while

Los bucles while se ejecutan continuamente, hasta que la expresión de dentro del paréntesis, (), pasa a ser falsa. Algo debe modificar la variable comprobada, el bucle while nunca terminará. Lo que modifique la variable puede estar en el código, como una variable que se incrementa, o ser una condición externa, como el valor que da un sensor.

break

break es usado para salir de los bucles do, for, o while, pasando por alto la condición normal del bucle. Es usado también para salir de una estructura de control switch.

return

Termina una función y devuelve un valor a la función que la llama. Puede no devolver nada.

HIGH

El significado de HIGH (en referencia a un pin) depende de si el pin está configurado como entrada (INPUT) o como salida (OUTPUT). Cuando un pin se configura como entrada (INPUT) usando pinMode, y se lee con digitalRead, el microcontrolador nos retornará HIGH si en el pin hay 3 voltios o más. Cuando un pin se configura como salida (OUTPUT) con pinMode, y se establece a HIGH con digitalWrite, el pin tiene 5V. En este estado puede usarse como fuente de corriente.

LOW

El significado de LOW difiere también según esté configurado como entrada (INPUT) o como salida (OUTPUT). Cuando un pin está configurado como entrada (INPUT) con pinMode, y se lee con digitalRead, el microcontrolador retornará LOW si el voltaje presente en el pin es de 2V o menor.

Cuando un pin es configurado como salida (OUTPUT) con pinMode, y establecido LOW con digitalWrite, el pin tiene 0 voltios. En este estado puede meter corriente.

char

Es un tipo de dato que ocupa un byte de memoria y almacena un valor de carácter. Los caracteres literales se escriben con comillas simples: 'A' (para varios caracteres -strings- utiliza dobles comillas "ABC").

byte

Un byte almacena un número sin signo de 8-bit, desde 0 hasta 255.

int

Integers (Números enteros) son el principal tipo de datos para almacenar números, y guardan valores de 2 bytes. Esto produce un rango entre -32,768 hasta 32,767.

long

Las variables de tipo Long son variables de tamaño extendido para almacenamiento de números, y 32 bits (4 bytes), desde -2,147,483,648 hasta 2,147,483,647.

void

La palabra reservada void se usa sólo en la declaración de funciones. Indica que se espera que no devuelva información a la función donde fue llamada.

pinMode()

Configura el pin especificado para comportarse como una entrada o una salida.

digitalWrite()

Escribe un valor HIGH o LOW hacia un pin digital.

Si el pin ha sido configurado como OUTPUT con pinMode(), su voltaje será establecido al correspondiente valor: 5V (o 3.3V en tarjetas de 3.3V) para HIGH, 0V (referencia) para LOW.

Si el pin es configurado como INPUT, escribir un valor de HIGH con digitalWrite() habilitará una resistencia interna de 20K conectada en pull-up. Escribir LOW invalidará la resistencia.

digitalRead()

Lee el valor de un pin digital especificado, HIGH o LOW.

analogRead()

Lee el valor de tensión en el pin analógico especificado. Esto proporciona una resolución en la lectura de: 5 voltios / 1024 unidades, es decir, 0.0049 voltios (4.9 mV) por unidad. El rango de entrada puede ser cambiado usando la función analogReference().

El conversor tarda aproximadamente 100 microsegundos (0.0001 segundos) en leer una entrada analógica por lo que se puede llevar una tasa de lectura máxima aproximada de 10.000 lecturas por segundo.

analogWrite()

Escribe un valor analógico (PWM) en un pin. La frecuencia de la señal PWM sera de aproximadamente 490 Hz. En la placa Arduino Mega, se puede llevar a cabo con los pines desde el 2 hasta el pin 13. La función analogWrite no tienen ninguna relación con los pines de entrada analógicos ni con la función analogRead.

delay()

Pausa el programa por un tiempo determinado (en milisegundos) especificado por un parámetro.

Serial

Se utiliza para la comunicación entre la placa Arduino y un ordenador u otros dispositivos. Todas las placas Arduino tienen al menos un puerto serie (también conocido como UART o USART): Serial. Se comunica a través de los pines digitales 0 (RX) y 1 (TX), así como con el ordenador mediante USB.

La placa Arduino Mega tiene tres puertos adicionales de serie: Serial1 en los pines 19 (RX) y 18 (TX), Serial2 en los pines 17 (RX) y 16 (TX), Serial3 en los pines 15 (RX) y 14 (TX).

begin()

Establece la velocidad de datos en bits por segundo (baudios) para la transmisión de datos en serie.

print()

Imprime los datos al puerto serie como texto ASCII. Este comando puede tomar muchas formas. Los números son impresos mediante un juego de caracteres ASCII para cada dígito.

IV.4 Descripción del entorno de programación XCTU

Digi, esta empresa facilita la conexión de dispositivos a la red al fabricar productos y tecnologías rentables. Digi fabrica los módulos Xbee, y para la configuración de los mismos proporciona el software X-CTU, el cual se describe a continuación.

Este programa se diseñó para interactuar con los archivos de firmware que se encuentran en productos de radiofrecuencia de Digi, y para proporcionar una interfaz gráfica fácil de utilizar para el usuario.

El ícono de X-CTU se puede observar en la Figura IV.13

Figura IV. 13 Ícono X-CTU

Al iniciar el software se ven cuatro pestañas en la parte superior del programa, como se observa en la Figura IV.14. A continuación se describe cada una de ellas:

PC Settings: Permite al usuario seleccionar el puerto COM.

Range Test: Permite al usuario realizar una prueba de rango entre dos radios.

Terminal: Permite el acceso al puerto COM de la computadora con un programa de emulación. También permite la posibilidad de acceder al firmware de los radios utilizando el comando AT.

Modem Configuration: Permite programar la configuración del firmware de las radios a través de una interfaz gráfica de usuario. Permite a los usuarios cambiar las versiones de firmware.

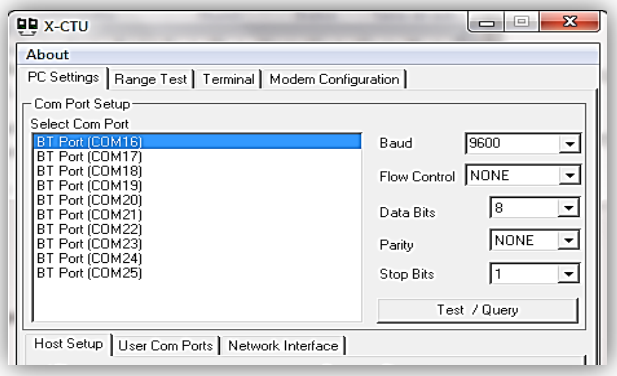


Figura IV. 14 Software X-CTU

IV.4.1 Configuración módulos Xbee

Básicamente para configurar los módulos Xbee en la pestaña PC Settings se elige el puerto en el cual se está estableciendo la comunicación entre el módulo Xbee y la computadora. Posteriormente en la pestaña Terminal se establece la comunicación entre el software y el módulo Xbee escribiendo tres veces el signo + el software contesta con un ok si la comunicación es exitosa. Finalmente el la pestaña Modem Configuration se configuran los comandos de los cuales se hicieron mención en el Capítulo III. La Figura IV.15 muestra el panel de configuración del software.

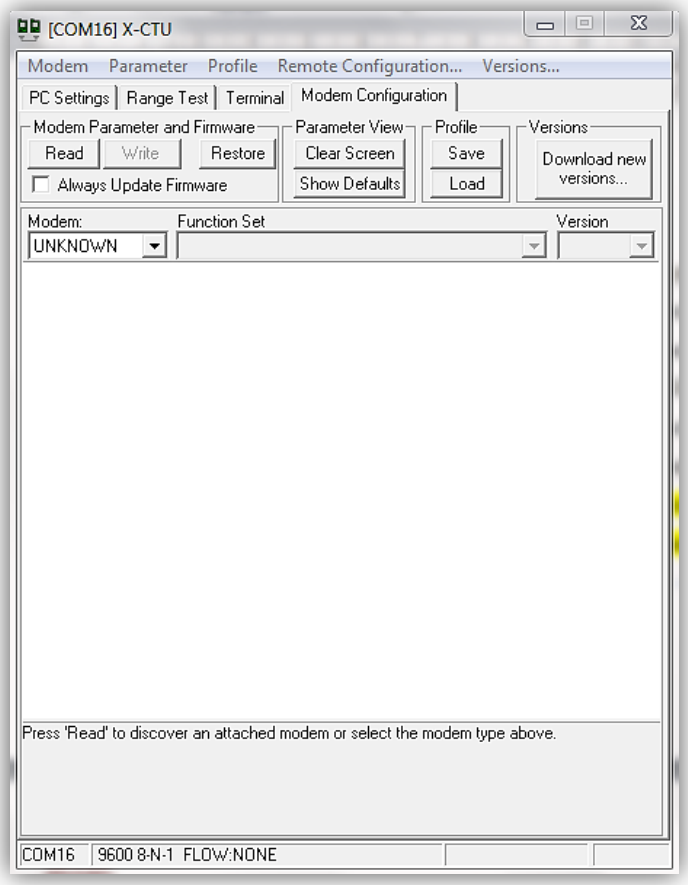


Figura IV. 15 Panel de Configuración

Para la configuración de los módulos Xbee utilizados en este proyecto después de confirmar la comunicación en Terminal se configuraron los siguientes parámetros en Modem Configuration mostrados en la Tabla IV.2:

Tabla IV. 2 Comandos configurados módulos XBee

Comando	Arduino Leonardo	Arduino Mega
ID	2009	2009
CH	1	1
DH	0	0
DL	0	0
MY	0	0

Como se puede apreciar los valores de los comandos DH, DL y MY se dejaron tal cual sus valores de fábrica, ya que no es una red que cuente con más de dos elementos, por lo tanto es suficiente con configurar el canal y la red. En el Capítulo III se definieron cada uno de los parámetros a configurar.

IV.5 Programación de los microcontroladores Arduino

En el diagrama de flujo de la Figura IV.16 se explica de manera sintetizada, la lógica de control considerada para el desarrollo de los programas en el entorno Arduino, ya que considera dos datos de entrada (traer y llevar), que son entradas físicas correspondientes a los micropulsadores "Traer" y "Llevar" respectivamente, ubicados en el tablero de control. Básicamente el diagrama de flujo inicia con la lectura de esos dos datos, enseguida se verifica su estado, el cual puede ser 1 lógico o 0 lógico, si el dato "llevar" es 1, la acción siguiente, es buscar el cajón de estacionamiento desocupado más próximo, una vez encontrado el cajón, se ejecuta la secuencia correspondiente para aparcar el automóvil en el lugar asignado. Cuando la secuencia termina, se finaliza el programa y se regresa al inicio. Si el dato "llevar" es 0, entonces se procede a verificar el estado del dato "Traer" en donde sucede un caso muy similar al descrito anteriormente,

pero ahora se ejecuta directamente la secuencia que el usuario ordenó en el tablero de control, como por ejemplo: si el usuario solicitó su automóvil que está aparcado en el cajón C, el sistema ejecuta la secuencia correspondiente para traer ese vehículo. Cuando la secuencia termina, se finaliza el programa y puede ser ejecutado de nueva cuenta. *En el anexo IV.F están los diagramas de flujo que contemplan todos los factores del proyecto, entradas, salidas, sensores y motores.*

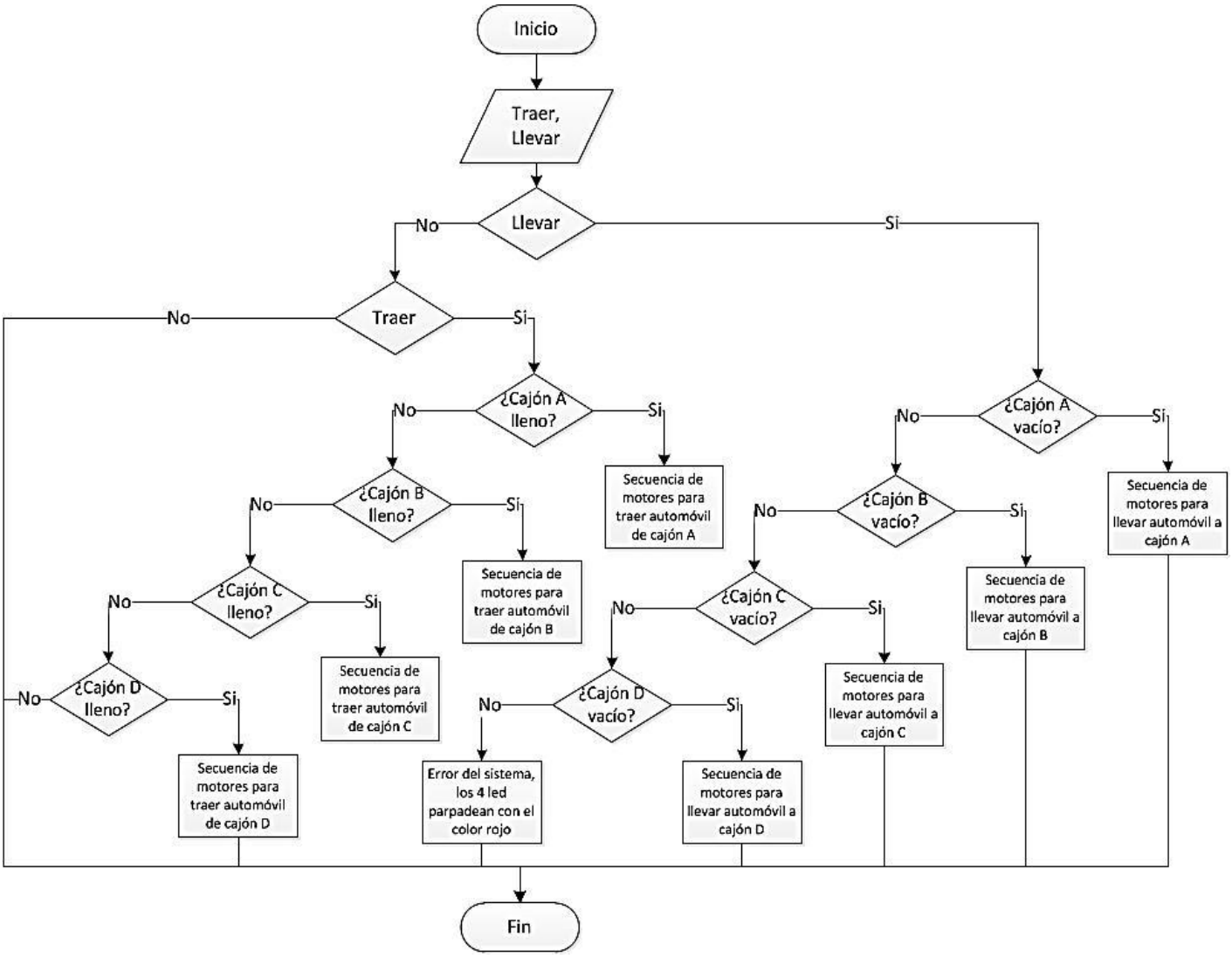


Figura IV. 16 Diagrama de flujo general

IV.5.1 Arduino Mega 2560

Para realizar el código de programación en el entorno de desarrollo Arduino para la tarjeta Mega, se realizó un diagrama de flujo de forma detallada que se encuentran en Anexo IV.F, pero para explicar las líneas de programación, se simplificó y se agrupó toda esa información en un diagrama de flujo que explica de manera general el código, y éste se muestra en la Figura IV.17.

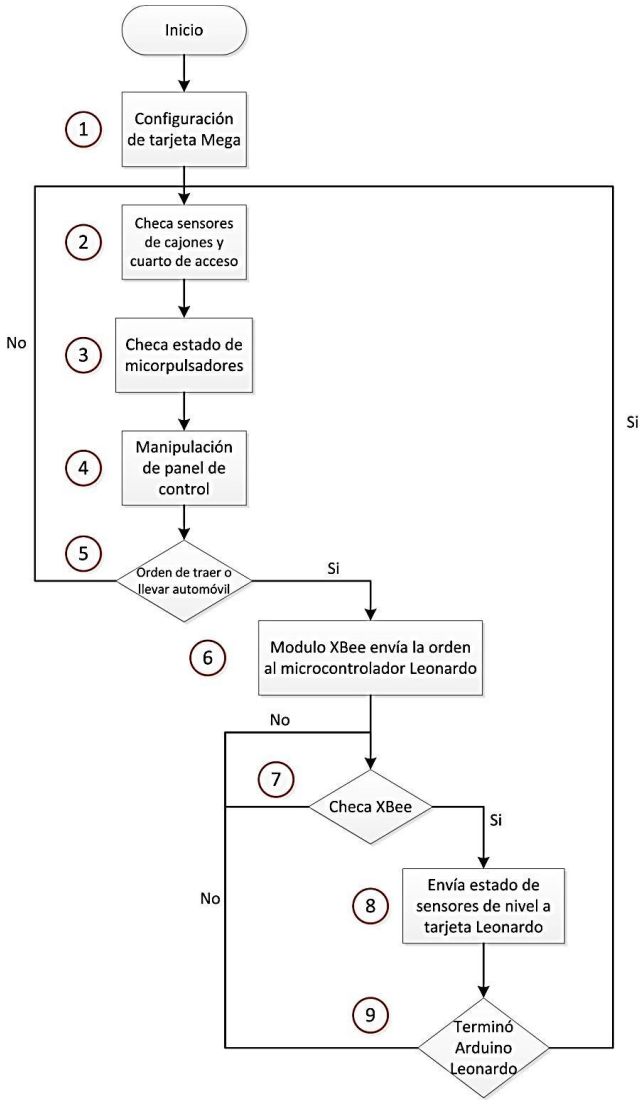


Figura IV. 17 Diagrama de flujo Arduino Mega 2560

En la Tabla IV.3 se explican cada uno de los bloques del diagrama de la Figura IV.17 y se anexa las líneas de código de programación correspondientes a ese bloque.

Tabla IV. 3 Código de programación Arduino Mega 2560

No. de bloque	Descripción y líneas de programación

1	Se configuran las entradas, salidas y el puerto de comunicación que se va a utilizar en el microcontrolador.
	
	`//Configuración inicial del microcontrolador Mega`
	`void setup(){`
	`  //Se inicia la comunicación serie a 9600 baudios.`
	`  Serial.begin(9600);`
	
	`  //Salidas  Configuración de algunas variables del programa como salidas.`
	`  pinMode (SV,OUTPUT);`
	`  pinMode (SR,OUTPUT);`
	`  pinMode (DV,OUTPUT);`
	`  pinMode (DR,OUTPUT);`
	`  pinMode (CV,OUTPUT);`
	`  pinMode (CR,OUTPUT);`
	`  pinMode (BV,OUTPUT);`
	`  pinMode (BR,OUTPUT);`
	`  pinMode (AV,OUTPUT);`
	`  pinMode (AR,OUTPUT);`
	`  pinMode (g,OUTPUT);`
	`  pinMode (punto,OUTPUT);`
	`  pinMode (c,OUTPUT);`
	`  pinMode (d,OUTPUT);`
	`  pinMode (x,OUTPUT);`
	`  pinMode (b,OUTPUT);`
	
	`  //Entradas  Configuración de algunas variables del programa como entradas.`
	`  pinMode (N,INPUT);`
	`  pinMode (E,INPUT);`
	`  pinMode (CA,INPUT);`
	`  pinMode (Seg,INPUT);`
	`  pinMode (Sen_C,INPUT);`
	`  pinMode (Sen_D,INPUT);`
	`  pinMode (Sen_A,INPUT);`
	`  pinMode (Sen_B,INPUT);`
	`  pinMode (RD1,INPUT);`
	`  pinMode (RD2,INPUT);`
	`  pinMode (RC1,INPUT);`
	`  pinMode (RC2,INPUT);`

2	Verifica el estado que tienen los sensores de presencia de los cajones de aparcamiento A, B, C, D, y el cuarto de acceso.

```
//Funciones que ejecutan el control del estacionamiento.

void Checa_cajones(){
  if(digitalRead(Sen_A)==1){
    if(digitalRead(Sen_B)==1){
      if(digitalRead(Sen_C)==1){
        if(digitalRead(Sen_D)==1){
          Semaforo_rojo();
        }
        else{
          Semaforo_verde();
        }
      }                                    //Checa los sensores de prescencia de los
      else{                                //cajones y controla las luces del semáforo.
        Semaforo_verde();
      }
    }
    else{
      Semaforo_verde();
    }
  }
  else{
    Semaforo_verde();
  }
}

void Checa_CA(){
  if(digitalRead(CA)==1){
    Semaforo_rojo();
    if(digitalRead(Seg)==1){
      char n='n';
      Asignar_cajon(n);
    }                                  //Verifica el estado del sensor del cuarto de
    else{                              //acceso para así manipular las luces del semáforo
      Navegador_principal();           //y losindicadores del tablero de control.
    }
  }
  else{
    Navegador_principal();
  }
}
```

3	Verifica el estado que tienen los micropulsadores del panel de control.
	
	`void Navegador_principal(){`
	`  if(digitalRead(N)==1){`
	`  Navegador_secundario();`
	`  }`
	`  else{`
	`  return;`
	`  }`
	`}`

4	Se ilumina el display y los led indicadores del panel de control dependiendo del estado de los sensores de presencia y de los micropulsadores leídos anteriormente.
	
	`void Navegador_aux(int x){`
	`  int i=0;`
	`  switch (x){`
	`  case 1:`
	`  if(digitalRead(Sen_A)==1){`
	`  char a='a';`
	`  A();`
	`  Encender_verde(a);`
	`  delay(1);`
	`  Navegador_aux1(x,i,a);`
	`  }`
	`  else{`
	`  x=2;`
	`  Navegador_aux(x);`
	`  }`
	`  break;`
	`  case 2:`
	`  if(digitalRead(Sen_B)==1){`
	`  char b='b';`
	`  B();`
	`  Encender_verde(b);`
	`  delay(1);`
	`  Navegador_aux1(x,i,b);`
	`  }`
	`  else{`
	`  x=3;`
	`  Navegador_aux(x);`
	`  }`
	`  break;`

```
  case 3:
    if(digitalRead(Sen_C)==1){
      char c='c';
      C();
      Encender_verde(c);
      delay(1);
      Navegador_aux1(x,i,c);
    }
    else{
      x=4;
      Navegador_aux(x);                    //Bloques encargados de verificar el estado
    }                                      //del botón Navegar y mostrar en los
    break;                                 //indicadores led la letra que se solicita.
  case 4:
    if(digitalRead(Sen_D)==1){
      char d='d';
      D();
      Encender_verde(d);
      delay(1);
      Navegador_aux1(x,i,d);
    }
    else{
      x=1;
      Navegador_aux(x);
    }
    break;
  }
}
```

5	Este bloque verifica si el usuario realizó alguna petición como acomodar o pedir que le sea devuelto su automóvil.
6	Se envía un carácter por medio de los módulos XBee a la tarjeta Leonardo, que le ordena realizar alguna secuencia en los motores, ya sea acomodar o devolver un vehículo.

```
void Traer_auto(char m){
  Leds_bajo();
  Encender_verde(m);
  switch (m){
  case 'a':
    Semaforo_rojo();
    A();
    Posicion_inicial3(m);
    break;
  case 'b':
    Semaforo_rojo();
    B();
    Posicion_inicial3(m);
    break;
```

```
  case 'c':
    Semaforo_rojo();
    C();
    Posicion_inicial3(m);
    break;
  case 'd':
    Semaforo_rojo();
    D();
    Posicion_inicial3(m);
    break;
  case 'n':
    return;
  }
}
void Llevar_auto(char x,char y){
  Leds_bajo();
  Encender_rojo(x);
  switch (x){
  case 'a':
    A();
    Posicion_inicial(x,y);
    break;
  case 'b':
    B();                                  //Lineas de código cuya finalidad es la de
    Posicion_inicial(x,y);                //manipular los led indicadores y el display,
    break;                                //cuando se está ejecutando la sesuencia de
  case 'c':                               //llevar automóvil a algún cajón X.
    C();
    Posicion_inicial(x,y);
    break;
  case 'd':
    D();
    Posicion_inicial(x,y);
    break;
  }
}
```

7

Este bloque verifica las peticiones del microcontrolador Arduino Leonardo.

```
void Actualizar_estado(long hl){
  Estado_de_los_sensores();
  if(Serial.available()>0){
    v=Serial.read();
    if(v=='T'){
      return;
    }                                     //Refresca la lectura del estado de los
  }                                       //sensores  del Arduino Mega mientras se
  if(hl==h){                              //está ejecutando una secuencia, como
    Actualizar_estado(hl);                //puede ser traer o llevar automóvil.
  }
  else{
    Codificador(h);
```

8	Envía el estado de los sensores de nivel a la tarjeta Leonardo para que esta realice adecuadamente la secuencia de los motores.
	`void Codificador(int h){`
	`  save=bitRead(h,0);`
	`  Serial.print(save,BIN);`
	`  save=bitRead(h,1);`
	`  Serial.print(save,BIN);`
	`  save=bitRead(h,2);`
	`  Serial.print(save,BIN);`
	`  save=bitRead(h,3);`
	`  Serial.print(save,BIN);  //Envía el estado de cada uno de los sensores`
	`  save=bitRead(h,4);  //al Arduino Leonardo mediante los modulos Xbee.`
	`  Serial.print(save,BIN);`
	`  save=bitRead(h,5);`
	`  Serial.print(save,BIN);`
	`  save=bitRead(h,6);`
	`  Serial.print(save,BIN);`
	`  save=bitRead(h,7);`
	`  Serial.print(save,BIN);`
	`  save=bitRead(h,8);`
	`  Serial.print(save,BIN);`
	`  save=bitRead(h,9);`
	`  Serial.print(save,BIN);`
	`}`

9	Verifica si en el puerto de comunicación serial hay un carácter enviado por el microcontrolador Leonardo que indique el término de la secuencia ejecutada.
	`void Actualizar_estado(long hl){`
	`  Estado_de_los_sensores();`
	`  if(Serial.available()>0){`
	`  v=Serial.read();`
	`  if(v=='T'){`
	`  return;`
	`  }  //Refresca la lectura del estado de los`
	`  }  //sensores  del Arduino Mega mientras se`
	`  if(hl==h){  //está ejecutando una secuencia, como`
	`  Actualizar_estado(hl);  //puede ser traer o llevar automóvil.`
	`  }`
	`  else{`
	`  Codificador(h);`
	`  Actualizar_estado(h);`
	`  }`
	`}`

El código de programación completo se encuentra en el Anexo E.

IV.5.2 Arduino Leonardo

Para realizar el código de programación en el entorno de Arduino para la tarjeta Leonardo, se realizó un diagrama de flujo de forma detallada que se encuentran en Anexo IV.F, pero para explicar las líneas de programación, se simplifico ese diagrama y se agrupo toda esa información en un diagrama de flujo que explica de manera general el código, y éste se observa en la Figura IV.18.

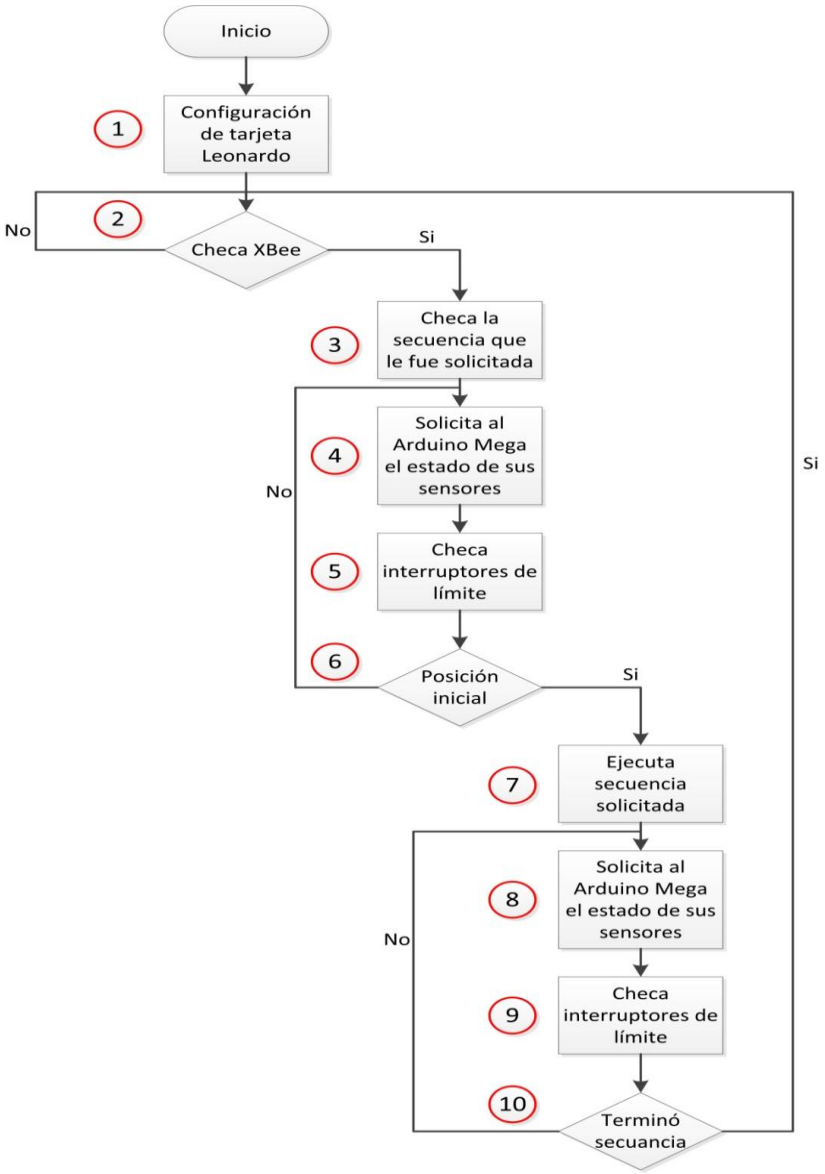


Figura IV. 18 Diagrama de flujo de Arduino Leonardo

En la Tabla IV.4 se explican cada uno de los bloques del diagrama de la Figura IV.18 y se anexa las líneas de código de programación correspondientes a ese bloque.

Tabla IV. 4 Código de programación Arduino Leonardo

No. de bloque	Descripción y líneas de programación

1	Se configuran las entradas, salidas y el puerto de comunicación que se va a utilizar en el microcontrolador.
	`//Configuración inicial del microcontrolador Leonardo`
	`void setup() {`
	`  //Se inicia la comunicación serie a 9600 baudios.`
	`  Serial.begin(9600);`
	`  Serial1.begin(9600);`
	`  //Entradas  Configuración de variables del programa como entradas.`
	`  pinMode(LI,INPUT);`
	`  pinMode(LO,INPUT);`
	`  //Salidas  Configuración de algunas variables del programa como salidas.`
	`  pinMode(M1Out,OUTPUT);`
	`  pinMode(M1In,OUTPUT);`
	`  pinMode(M2Up,OUTPUT);`
	`  pinMode(M2Down,OUTPUT);`
	`  pinMode(M3Left,OUTPUT);`
	`  pinMode(M3Right,OUTPUT);`
	`}`

2	Verifica si no ha recibido señales del XBee del Arduino Mega para realizar una secuencia.
	`void loop(){`
	`  if(Serial1.available()>0){`
	`  m=Serial1.read();  //Ciclo que se ejecuta de forma infinita despues de`
	`  Serial.println(m);  //energizar la tarjeta Leonardo.`
	`  int x=0;`
	`  Checa_secuencia(m,x);`
	`  }`
	`}`

3	Codifica el carácter que le fue enviado por el Arduino Mega para saber la secuencia que tiene que ejecutar.
	`void Checa_secuencia(char m,int x){`
	`  switch(m){`
	`  //Funciones para llevar auto.`
	`  case 'P':`
	`  Llevar_auto_A(x);`
	`  break;`
	`  case 'O':`
	`  Llevar_auto_B(x);`
	`  break;`

```
    case 'I':
      Llevar_auto_C(x);
      break;
    case 'U':
      Llevar_auto_D(x);
      break;
  //Funciones para traer auto.          Codifica el caracter que le envía el Arduino
    case 'Q':                          //Mega con el fin de que el Arduino Leonardo
      Traer_auto_A(x);                 //determine la secuencia que van a ejecutar los
      break;                           //motores.
    case 'W':
      Traer_auto_B(x);
      break;
    case 'E':
      Traer_auto_C(x);
      break;
    case 'R':
      Traer_auto_D(x);
      break;
  }
}
```

4

Solicita al Arduino Mega que le envíe el estado de los sensores de nivel.

```
int Checa_sensores_de_mega(int x){
  while(x<10){
    if(Serial1.available()>0){
      i=Serial1.read();
      if(i==48){
        bitWrite(j,x,0);                  //Codifica los caracteres que fueron
        x++;                              //enviados por el Arduino Mega para saber
      }                                   //el estado de los sensores IR.
      if(i==49){
        bitWrite(j,x,1);
        x++;
      }
    }
  }
  //Serial.println(j,BIN);
  return j;
}
```

5

Checa el estado de los interruptores de límite de la plataforma móvil.

```
int Estado_de_sensores_de_limite(){
  if(digitalRead(LI)==1){
    bitWrite(k,1,1);
  }else{
    bitWrite(k,1,0);
  }
  if(digitalRead(LO)==1){                 //Verifica el estado de los sensores de
    bitWrite(k,0,1);                      //límite.
  }else{
    bitWrite(k,0,0);
  }
```

6,7,8,10	Determina si la posición en la que está ubicada la estructura móvil, es la posición inicial. De no ser así, no inicia con la secuencia de los motores solicitada (Bloque 6). Caso contrario Inicia la secuencia correspondiente en los motores, como la de acomodar o devolver un vehículo (Bloque 7). En las líneas de programación siguientes, se da un ejemplo con la secuencia llevar auto al cajón "A". También, en estas líneas de programación están presentes los bloques número 8 y 9, ya que se está solicitando checar los sensores del Arduino Mega y los sensores de límite conectados al Arduino Leonardo con el objetivo saber la ubicación de la estructura móvil y realizar la secuencia correctamente. Una vez concluida la operación, se envía el carácter 'T' a través de los módulos XBee al microcontrolador Mega para confirmar el final de la secuencia.

```
void Llevar_auto_A(int x){
  while(j!=B10){
    Checa_sensores_de_mega(x);
  }
  while((k==B10)==0){
    Estado_de_sensores_de_limite();
  }
  digitalWrite(M1Out,HIGH);
  while((k==B1)==0){
    Estado_de_sensores_de_limite();
  }
  digitalWrite(M1Out,LOW);
  delay(1000);
  digitalWrite(M2Up,HIGH);
  while(j!=B1){
    Checa_sensores_de_mega(x);
  }
  digitalWrite(M2Up,LOW);
  delay(1000);
  digitalWrite(M1In,HIGH);
  while((k==B10)==0){
    Estado_de_sensores_de_limite();
  }
  digitalWrite(M1In,LOW);                    //Ejecuta la secuencia en los motores
  delay(1000);                               //con el fin de llevar un automóvil al
  digitalWrite(M3Right,HIGH);                //cajón A.
  while(j!=B100){
    Checa_sensores_de_mega(x);
  }
  digitalWrite(M3Right,LOW);
  delay(1000);
  digitalWrite(M1Out,HIGH);
  while((k==B1)==0){
    Estado_de_sensores_de_limite();
  }
```

```
  digitalWrite(M1Out,LOW);
  delay(1000);
  digitalWrite(M2Down,HIGH);
  while(j!=B1000){
    Checa_sensores_de_mega(x);
  }
  digitalWrite(M2Down,LOW);
  delay(1000);
  digitalWrite(M1In,HIGH);
  while((k==B10)==0){
    Estado_de_sensores_de_limite();
  }
  digitalWrite(M1In,LOW);
  delay(1000);
  digitalWrite(M3Left,HIGH);
  while(j!=B10){
    Checa_sensores_de_mega(x);
  }
  digitalWrite(M3Left,LOW);
  Serial1.print('T');
}
```

El código de programación completo se encuentra en el Anexo F.

Capítulo V Pruebas y resultados

Introducción

En este capítulo se desarrollan las pruebas realizadas al prototipo, en donde se describen los resultados obtenidos del mecanismo de la plataforma móvil, cabina y estructura móvil, alimentados a diferentes tensiones. Así mismo se realizaron pruebas al panel de control, a los módulos XBee, a la etapa de potencia y a los sensores implementados en este proyecto. A continuación se describen las pruebas y resultados obtenidos.

Prueba de accionamiento de los motores

Los tres mecanismos de los que consta el prototipo deben cumplir con los movimientos para los que fueron diseñados, a la mayor velocidad posible para realizar un aparcamiento o devolución del vehículo en el menor tiempo, pero sin dejar de lado la seguridad de la unidad. Es por ello, que las pruebas mostradas en esta sección, están enfocadas a la verificación del funcionamiento mecánico de la plataforma móvil, cabina y estructura móvil. Para realizar estas pruebas, se requiere del accionamiento de los motores correspondientes, este accionamiento se lleva a cabo manualmente y a diferentes tensiones de alimentación, lo único con lo que se cuenta eléctricamente en estas pruebas es con el motor de cada mecanismo y con un interruptor para evitar el paso de la corriente. Las tensiones de prueba son 5V, 9V y 12V, con la finalidad de determinar la tensión óptima para cada mecanismo, considerando funcionamiento aceptable y tiempo. Con estas pruebas se esperan obtener movimientos satisfactorios en el menor tiempo posible para cada uno de los mecanismos.

Plataforma móvil

En la Figura V.1 se muestra el mecanismo de la plataforma móvil con su respectivo motor eléctrico, la función de la plataforma es deslizarse sobre el riel con la ayuda del tornillo sin fin acoplado a la flecha del motor reductor. La Tabla V-1 plantea los resultados obtenidos a diferentes tensiones de alimentación.

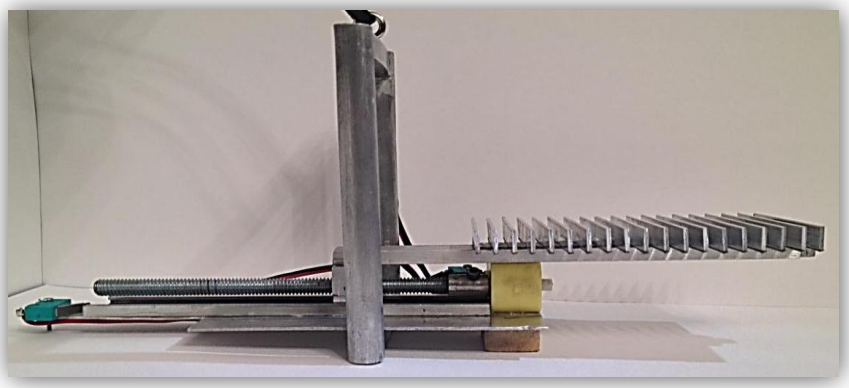


Figura V. 1 Plataforma móvil

Tabla V- 1 Resultados experimentales de plataforma móvil

Tensión de alimentación (V)	Resultado del deslizamiento	Tiempo que tarda en llegar al final de carrera (s)
5	El mecanismo de la plataforma móvil se comportó de la manera esperada con las tres diferentes tensiones de alimentación. Llegó al final de carrera en todos los casos y no realizó ningún movimiento irregular a la hora de deslizarse sobre el riel. La única diferencia evidente y lógica fue la velocidad con la que realizó el desplazamiento, siendo 5V la tensión de alimentación que provocó que se desplazara de forma lenta y 12V la que logró que se desplazara con mayor rapidez.	40
9		25
12		15

En la Figura V.2 a) y V.2 b) se aprecia el deslizamiento que se obtiene al ser energizado el motor con cualquier tensión de alimentación, se observa la plataforma en su posición inicial (Figura V.2 a)) y en la Figura V.2 b) se puede apreciar que la plataforma llega al final de carrera.

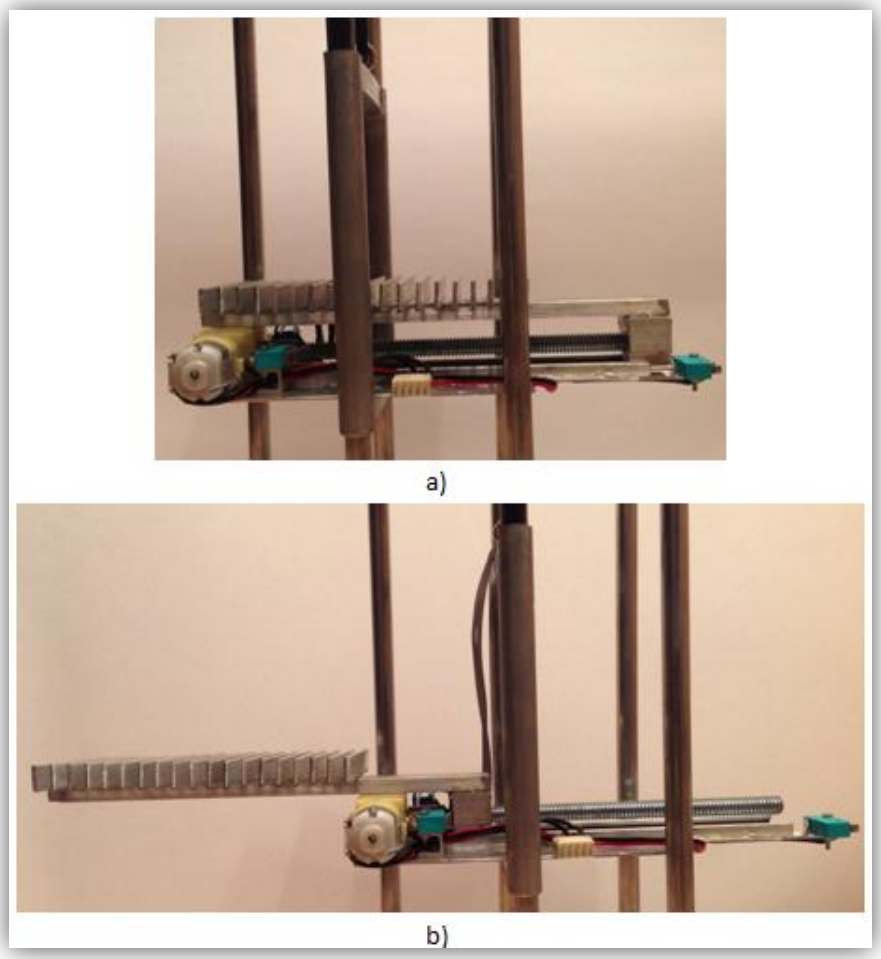


Figura V. 2 Desplazamiento plataforma móvil

Observando los tiempos que se obtuvieron en la Tabla V-1 es claro que 12V fue la tensión de alimentación que logró que la plataforma móvil llegara al final de carrera más rápido. Entonces se concluye que 12V será la tensión a la que se alimentará este motor reductor.

Cabina

La Figura V.3 muestra el mecanismo de la cabina con su respectivo motor eléctrico, El movimiento que realiza es vertical y la distancia que corre dicho mecanismo es de 10 cm, por lo tanto se requiere una velocidad que no ponga en peligro el vehículo ni la funcionalidad del mismo mecanismo. En la Tabla V-2 se muestran los resultados obtenidos a diferentes tensiones de alimentación.

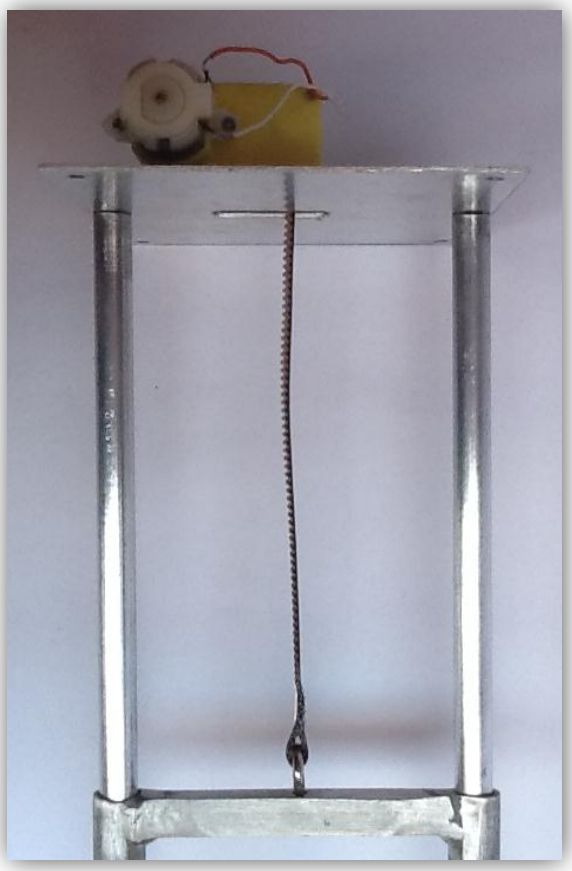


Figura V. 3 Cabina

Tabla V- 2 Resultados experimentales de la cabina

Tensión de alimentación (V)	Resultado del deslizamiento	Tiempo que tarda en llegar al final de carrera (s)
5	El mecanismo de la cabina alimentado a una tensión de 12v cumplió con su función, pero el automóvil a escala que se estaba transportando se movió de su posición original, y corrió peligro de caerse de la cabina. Con 9v ocurrió lo mismo que con 12v debido a que el vehículo no se mantuvo en la posición original. Finalmente se realizó la prueba con 5v, la cabina cumplió el desplazamiento requerido y el vehículo permaneció estático.	2
9		1.5
12		1

En la Figura V.4 se observa el desplazamiento que se obtiene al ser energizado el motor con cualquier tensión de alimentación, la Figura V.4 a) muestra la cabina en su posición inicial y en la Figura V.4 b) se aprecia la altura máxima que alcanza dicho mecanismo.

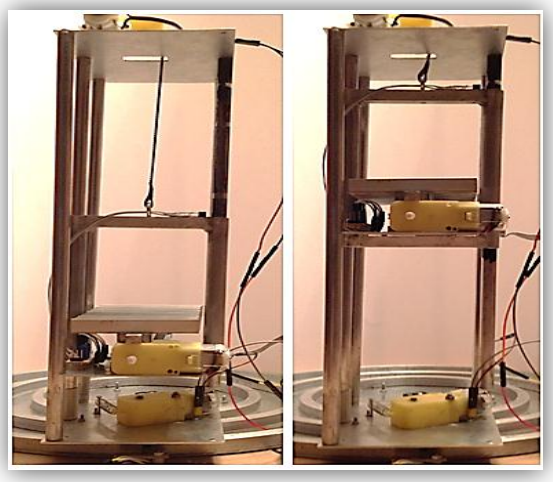


Figura V. 4 Desplazamiento de la cabina

Tomando en cuenta las consideraciones y datos obtenidos experimentalmente se determina que la tensión ideal para el buen funcionamiento del mecanismo de la cabina será de 5v.

Estructura móvil

La Figura V.5 muestra la estructura móvil, la cual realiza un movimiento de traslación curvilínea. Como se puede apreciar el motor que está acoplado en la base móvil es el que permite que dicha estructura se desplace hasta encontrar la columna del cajón requerido. Así como en los mecanismos de plataforma móvil y cabina, el movimiento que debe realizar debe ser suave y constante, para que la unidad no corra peligro al ser aparcado o devuelto a su respectivo dueño. En la Tabla V.3 se muestran los resultados obtenidos a diferentes tensiones de alimentación.

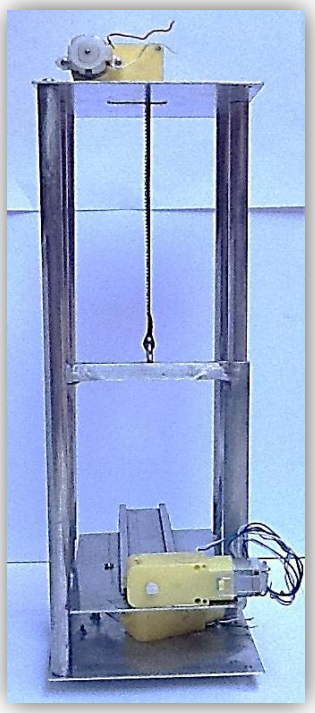


Figura V. 5 Estructura móvil

Tabla V- 3 Resultados experimentales de la estructura móvil

Tensión de alimentación (V)	Resultado del deslizamiento	Tiempo que tarda en llegar a la columna de los cajones A y C o B y D según sea el caso (s)
5	La estructura móvil con una tensión de alimentación de 5v no cumplió con el movimiento requerido, ya que esta tensión no fue suficiente para mover todo el mecanismo debido a la presión ejercida de la llanta de la flecha del motor al riel.	6
9	Con 9v el desplazamiento fue uniforme y el vehículo permaneció en la posición original.	3
12	Con una tensión de alimentación de 12v el desplazamiento de la estructura móvil fue rápido y provocó que el vehículo saltara durante el recorrido.	1.5

Finalmente se concluye que la alimentación del motor de la base móvil será de 9v. Para alimentar los motores de los mecanismos anteriormente mencionados se utilizó una fuente conmutada, empleando los 12 v de la misma, y se implementó un PWM para obtener los 5 v y 9 v deseados.

Panel de control

El panel de control fue alimentado con 5v de la fuente del Arduino Mega 2560 y las pruebas realizadas constaron en que lo mostrado por el display del panel fuese lo requerido por programa.

La primer prueba fue presionando el botón de seguro, para indicarle al sistema que podía llevarse el vehículo, y conforme a programa se mostró la letra A en el display, que es el lugar más próximo, mientras el mecanismo ejecutaba las tareas correspondientes, el primer led (de arriba a abajo) encendió rojo, hasta que se concluyó la tarea de resguardo del vehículo.

Posteriormente se realizaron las mismas pruebas cajón por cajón, para llevar y traer vehículos, y el panel de control funcionó correctamente en cada una de estas. La Figura V.6 muestra el panel de control implementado en el prototipo, e indica la tarea de cada botón.

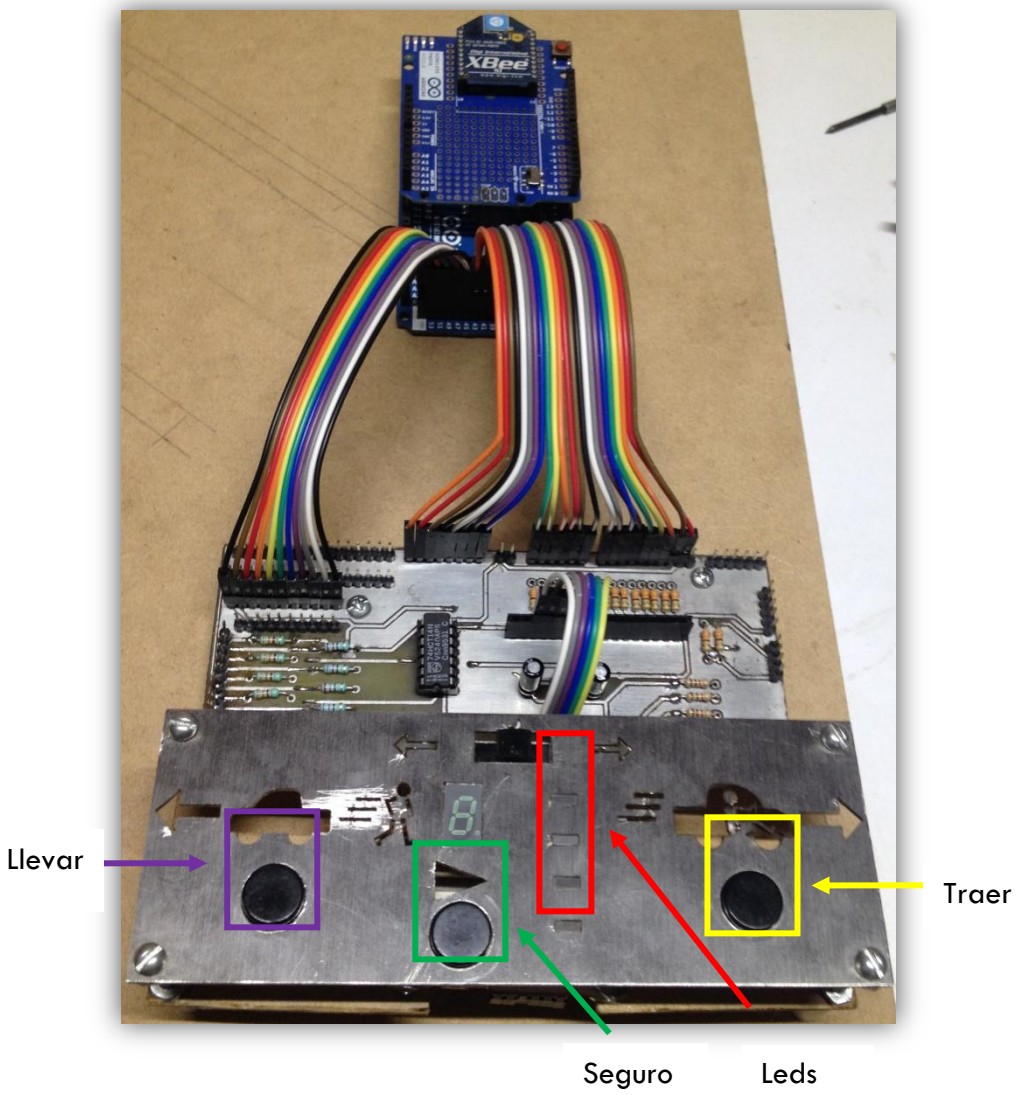


Figura V. 6 Panel de control

Etapa de potencia

La primera prueba fue medir la corriente de los motores ya acoplados al mecanismo, ya que así se considera la carga que soportarán los mismos. La Tabla V.4 muestra la corriente demandada por cada motor.

Tabla V- 4 Corriente de los motores

Motor	Tensión de alimentación (V)	Corriente (A)
1	12	0.2
2	12	0.3
3	12	0.35

Estos motores fueron alimentados con una fuente conmutada, y se conectaron sus terminales a un puente H L293B a los cuales preceden los optoacopladores para separar la etapa de control de la de potencia. Una vez conectados y alimentados los motores a la tensión requerida por cada uno, se realizó una prueba para llevar automóvil al cajón A. Esta prueba se realizó alámbricamente. Los resultados fueron favorables, ya que los motores realizaron las tareas programadas.

Módulos Xbee

La primera prueba realizada a estos módulos fue encender y apagar un led inalámbricamente. En el Arduino Leonardo se encontraba el interruptor que encendía y apagaba el led, y en el Arduino Mega 2560 se encontraba dicha salida. El resultado de esta prueba fue favorable, ya que el led respondió satisfactoriamente a la entrada. Este resultado indicó que los módulos estaban correctamente configurados y que podíamos continuar con pruebas al prototipo.

Una vez obtenidos los resultados esperados en pruebas a etapa de potencia y sabiendo que los módulos XBee estaban correctamente configurados, se continuó con pruebas de la comunicación inalámbrica al prototipo. La prueba fue llevar el vehículo al cajón A y el resultado fue el esperado, ya que el mecanismo cumplió con las tareas programadas.

Sensores

La prueba se realizó con dos DIP switches que se muestran en la Figura V.7, los cuales se describen a continuación. El DIP switch de la izquierda representa los sensores que indican la posición de la estructura móvil y de la cabina. Los interruptores 1 y 2 del DIP switch de la derecha representan la ubicación de la cabina en el cuarto de acceso, los interruptores 3, 4, 5 y 6 representan el estado de los cajones, ya sea libre u ocupado, finalmente el interruptor 7 indica el estado del cuarto de acceso, que puede ser libre u ocupado.

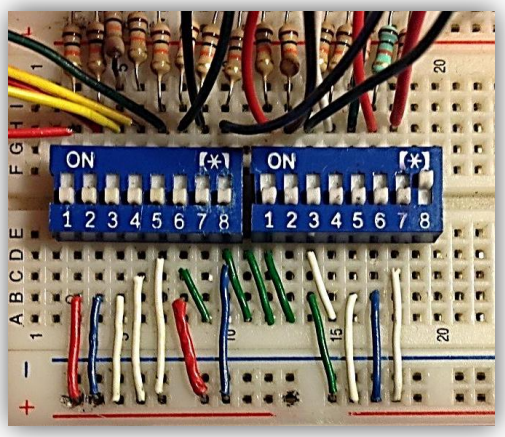


Figura V. 7 DIP switches

Al ejecutar correctamente las tareas el mecanismo del prototipo se concluye que los interruptores funcionan correctamente.

Tiempos para llevar y traer un vehículo

Realizando secuencias para llevar un vehículo al cajón A del estacionamiento, se obtuvo un promedio de tiempo, el cual es de 89 segundos equivalente a 1.4 minutos, contra los 6 minutos que se tarda un estacionamiento tradicional en entregar un vehículo, claro que la realidad es muy diferente y este tiempo puede variar dependiendo de las condiciones y el equipo implementado.

Conclusiones

Con el desarrollo de este trabajo se puede concluir que el proyecto propuesto demuestra la funcionalidad de un estacionamiento radial automático, ya que la cantidad de automóviles almacenados en el espacio de un estacionamiento de este tipo será superior al de cualquier estacionamiento tradicional, debido a que el incremento de lugares representará un crecimiento vertical del estacionamiento, ya sea un estacionamiento subterráneo o sobre el nivel del suelo. Así mismo se demuestra la seguridad que ofrece un estacionamiento como el propuesto, debido a su autonomía y a que al único lugar donde accede el conductor es el cuarto de acceso.

Se concluye que el estacionamiento radial automático propuesto en este trabajo reducirá el tiempo de recepción y entrega a usuarios, ya que asignará el lugar más próximo de resguardo a los vehículos. También se habla de reducción de tiempo debido a que no intervendrá alguna persona para llevar del cuarto de acceso al cajón de aparcamiento el vehículo o viceversa, esto gracias a su autonomía.

El código de programación de los microcontroladores es óptimo para la aplicación en el sistema propuesto, ya que aunque se agreguen o quiten cajones de estacionamiento el código se repetirá para todos, ya que es la misma lógica en cada uno de los cajones para llevar o traer un vehículo.

Se habla de un sistema eficiente debido a que aunque el sistema esté ejecutando alguna acción, ya sea llevar o traer un vehículo, si se le solicita ejecutar otra acción el sistema termina la actividad previa y posteriormente inicia con la última solicitada.

Finalmente se concluye que los objetivos planteados se cumplieron, ya que se demostró el accionamiento de los motores con la comunicación inalámbrica y se desarrolló un sistema autónomo que hace de este un estacionamiento eficiente y seguro.

Trabajo a Futuro

A continuación se mencionan los posibles trabajos a futuro que pueden complementar este proyecto desarrollado durante el presente trabajo.

- Implementar un sistema de seguridad en el cuarto de acceso, el cual conste de un sensor para detectar la presencia de personas o animales dentro del vehículo.
- Desarrollar un sistema que proporcione un boleto automáticamente después de dejar el automóvil en el cuarto de acceso, para que sólo el poseedor de éste pueda solicitar el vehículo.
- Este proyecto da las bases para el desarrollo a futuro de un estacionamiento de este tipo en lugares concurridos en México, pero en este trabajo no se consideran datos civiles, así que, puede hacerse un estudio de la estructura para un estacionamiento real y así complementar lo propuesto.
- Hacer un estudio del equipo mecánico a implementar en un estacionamiento real de este tipo.
- Hacer un estudio del equipo eléctrico a implementar en un estacionamiento real tomando como base el propuesto es este trabajo.
- Probar otros métodos para controlar los motores y para saber la posición del mecanismo.
- Diseñar una HMI en algún entorno de desarrollo de sistemas, como LabVIEW.
- Realizar el análisis de costos correspondiente si se quiere potencializar a un sistema real un estacionamiento como el propuesto.
- Implementar un sistema de ventilación y aire acondicionado (HVAC por sus siglas en inglés) para proporcionar un enfriamiento adecuado al aparcamiento, debido al calentamiento de los motores vehiculares.

Bibliografía

ALSE Mexicana. (s.f.). *ALSE Mexicana*. Recuperado el 20 de Agosto de 2013, de http://www.alsemexicana.com

Arduino. (2013). Recuperado el 15 de Agosto de 2013, de http://arduino.cc/

Autostadt. (2013). *Autostadt*. Recuperado el 15 de Agosto de 2013, de http://www.autostadt.de

Bluetooth. (2013). Recuperado el 15 de Noviembre de 2013, de http://www.bluetooth.com/

Cal y Mayor, R. (1947). *Tesis Profesional*. México, D.F.: Escuela Nacional de Ingenieros. UNAM.

Cal y Mayor, R. (1986). *Estacionamiento*. México D.F.: Asociación Mexicana de Caminos.

egomexico. (2013). Recuperado el 15 de Noviembre de 2013, de http://www.egomexico.com/

Galeón. (2013). Recuperado el 15 de Noviembre de 2013, de http://microcontroladores-e.galeon.com/

GIVT. (2013). *GIVT*. Recuperado el 15 de Agosto de 2013, de http://www.givt.de

INEGI. (2013). *INEGI*. Recuperado el 13 de Agosto de 2013, de http://www3.inegi.org.mx

IPS. (2013). *Integral Park Systems*. Recuperado el 15 de Septiembre de 2013, de http://www.integralparksystems.com

Mexicano, I. N. (2010). *Eco vehículos*. Recuperado el 17 de Junio de 2014, de http://www.ecovehiculos.gob.mx/

National Instruments. (2013). Recuperado el 15 de Noviembre de 2013, de http://www.ni.com/

Perfect Park. (2013). *Perfect Park*. Recuperado el 20 de Agosto de 2013, de http://www.perfectparkusa.com

PROFECO. (20 de Noviembre de 2009). *Brújula de compra de PROFECO*. Recuperado el 20 de Enero de 2014, de http://www.profeco.gob.mx/

Real Academia Española. (2001). *Real Academia Española*. Recuperado el Septiembre de 2013, de http://rae.es

Robotic Parking System. (2013). *Robotic Parking System*. Recuperado el 16 de Agosto de 2013, de http://www.roboticparking.com

Salamanca, J. G. (1968). La Contaminación Atmosférica. Barcelona: Espasa.

UNICROM. (2013). Recuperado el 15 de Noviembre de 2013, de http://www.unicrom.com/

WordPress. (2013). Recuperado el 15 de Noviembre de 2013, de http://sistemascomunic.wordpress.com/

ZigBee Alliance. (2013). Recuperado el 15 de Noviembre de 2013, de http://www.zigbee.org/

ZigBee Alliance. (2013). Recuperado el 15 de Agosto de 2013, de http://www.zigbee.org/

Anexo A Datasheet 74HCT14

74HCT14

Hex Schmitt-Trigger Inverter with LSTTL Compatible Inputs

High–Performance Silicon–Gate CMOS

The 74HCT14 may be used as a level converter for interfacing TTL or NMOS outputs to high–speed CMOS inputs.

The HCT14 is useful to "square up" slow input rise and fall times. Due to the hysteresis voltage of the Schmitt trigger, the HCT14 finds applications in noisy environments.

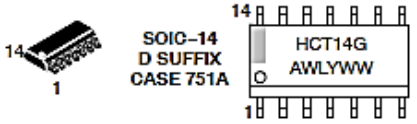
Features

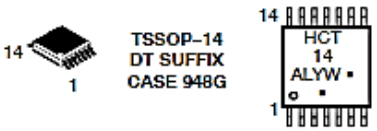
- Output Drive Capability: 10 LSTTL Loads
- TTL/NMOS–Compatible Input Levels
- Outputs Directly Interface to CMOS, NMOS, and TTL
- Operating Voltage Range: 4.5 to 5.5 V
- Low Input Current: 1.0 μA
- In Compliance With the JEDEC Standard No. 7A Requirements
- ESD Performance: HBM > 2000 V; Machine Model > 200 V
- Chip Complexity: 72 FETs or 18 Equivalent Gates
- These are Pb–Free Devices

ON Semiconductor®

http://onsemi.com

MARKING DIAGRAMS



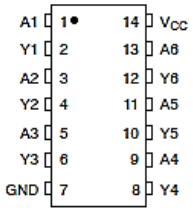


HCT14	= Device Code
A	= Assembly Location
L, WL	= Wafer Lot
Y	= Year
W, WW	= Work Week
G or ▪	= Pb–Free Package

(Note: Microdot may be in either location)

74HCT14

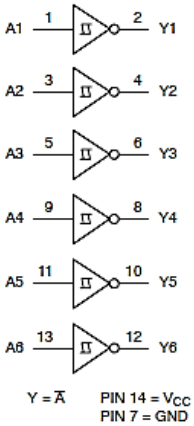
PIN ASSIGNMENT



FUNCTION TABLE

Input A	Output Y
L	H
H	L

LOGIC DIAGRAM



$Y = \overline{A}$ PIN 14 = V_{CC}
PIN 7 = GND

MAXIMUM RATINGS

Symbol	Parameter		Value	Unit
V_{CC}	DC Supply Voltage	(Referenced to GND)	−0.5 to +7.0	V
V_I	DC Input Voltage	(Referenced to GND)	−0.5 to V_{CC} +0.5	V
V_O	DC Output Voltage	(Referenced to GND)	−0.5 to V_{CC} +0.5	V
I_{IK}	DC Input Diode Current		±20	mA
I_{OK}	DC Output Diode Current		±25	mA
I_O	DC Output Sink Current		±25	mA
I_{CC}	DC Supply Current per Supply Pin		±50	mA
I_{GND}	DC Ground Current per Ground Pin		±50	mA
T_{STG}	Storage Temperature Range		−65 to +150	°C
T_L	Lead Temperature, 1 mm from Case for 10 Seconds		260	°C
T_J	Junction Temperature under Bias		+150	°C
θ_{JA}	Thermal Resistance	SOIC TSSOP	125 170	°C/W
P_D	Power Dissipation in Still Air at 85°C	SOIC TSSOP	500 450	mW
MSL	Moisture Sensitivity		Level 1	
F_R	Flammability Rating	Oxygen Index: 30% – 35%	UL 94 V–0 @ 0.125 in	
V_{ESD}	ESD Withstand Voltage	Human Body Model (Note 1) Machine Model (Note 2)	>2000 >200	V
$I_{Latchup}$	Latchup Performance	Above V_{CC} and Below GND at 85°C (Note 3)	±300	mA

Stresses exceeding Maximum Ratings may damage the device. Maximum Ratings are stress ratings only. Functional operation above the Recommended Operating Conditions is not implied. Extended exposure to stresses above the Recommended Operating Conditions may affect device reliability.

1. Tested to EIA/JESD22–A114–A.
2. Tested to EIA/JESD22–A115–A.
3. Tested to EIA/JESD78.
4. For high frequency or heavy load considerations, see the ON Semiconductor High–Speed CMOS Data Book (DL129/D).

RECOMMENDED OPERATING CONDITIONS

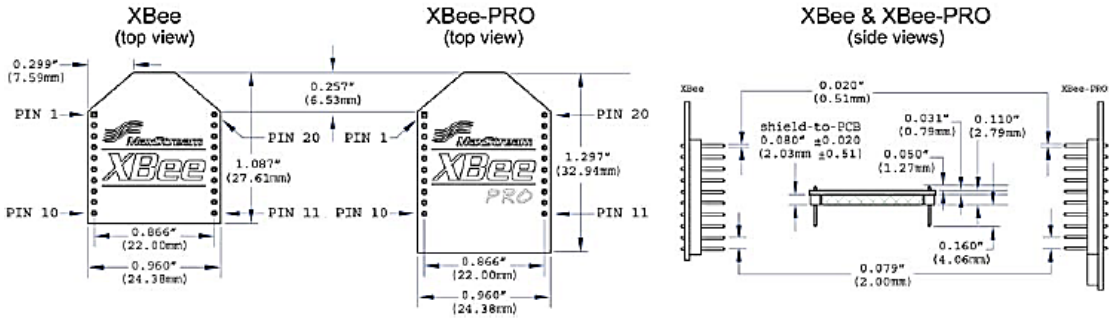
Symbol	Parameter		Min	Max	Unit
V_{CC}	DC Supply Voltage	(Referenced to GND)	4.5	5.5	V
V_I, V_O	DC Input Voltage, Output Voltage	(Referenced to GND)	0	V_{CC}	V
T_A	Operating Temperature, All Package Types		−55	+125	°C
t_r, t_f	Input Rise and Fall Time (Figure 1)		–	(Note 5)	ns

5. No Limit when $V_I \approx 50\%$ V_{CC}, $I_{CC} > 1$ mA.
6. Unused inputs may not be left open. All inputs must be tied to a high–logic voltage level or a low–logic input voltage level.

Anexo B Datasheet Módulo XBee

Mechanical Drawings

Figure 1-01. Mechanical drawings of the XBee/XBee-PRO OEM RF Modules (antenna options not shown)
The XBee and XBee-PRO RF Modules are pin-for-pin compatible.



Specifications of the XBee/XBee-PRO OEM RF Modules

Specification	XBee	XBee-PRO
Performance		
Indoor/Urban Range	up to 100 ft. (30 m)	Up to 300' (100 m)
Outdoor RF line-of-sight Range	up to 300 ft. (100 m)	Up to 1 mile (1500 m)
Transmit Power Output (software selectable)	1mW (0 dBm)	60 mW (18 dBm) conducted, 100 mW (20 dBm) EIRP*
RF Data Rate	250,000 bps	250,000 bps
Serial Interface Data Rate (software selectable)	1200 - 115200 bps (non-standard baud rates also supported)	1200 - 115200 bps (non-standard baud rates also supported)
Receiver Sensitivity	-92 dBm (1% packet error rate)	-100 dBm (1% packet error rate)
Power Requirements		
Supply Voltage	2.8 – 3.4 V	2.8 – 3.4 V
Transmit Current (typical)	45mA (@ 3.3 V)	If PL=0 (10dBm): 137mA(@3.3V), 139mA(@3.0V) PL=1 (12dBm): 155mA (@3.3V), 153mA(@3.0V) PL=2 (14dBm): 170mA (@3.3V), 171mA(@3.0V) PL=3 (16dBm): 188mA (@3.3V), 195mA(@3.0V) PL=4 (18dBm): 215mA (@3.3V), 227mA(@3.0V)
Idle / Receive Current (typical)	50mA (@ 3.3 V)	55mA (@ 3.3 V)
Power-down Current	< 10 µA	< 10 µA
General		
Operating Frequency	ISM 2.4 GHz	ISM 2.4 GHz
Dimensions	0.960" x 1.087" (2.438cm x 2.761cm)	0.960" x 1.297" (2.438cm x 3.294cm)
Operating Temperature	-40 to 85° C (industrial)	-40 to 85° C (industrial)
Antenna Options	Integrated Whip, Chip or U.FL Connector	Integrated Whip, Chip or U.FL Connector
Networking & Security		
Supported Network Topologies	Point-to-point, Point-to-multipoint & Peer-to-peer	
Number of Channels (software selectable)	16 Direct Sequence Channels	12 Direct Sequence Channels
Addressing Options	PAN ID, Channel and Addresses	PAN ID, Channel and Addresses

Anexo C Datasheet 4N25

MOTOROLA
SEMICONDUCTOR TECHNICAL DATA

Order this document by 4N25/D

GlobalOptoisolator™

VDE | UL | CSA | SETI | SEMKO | DEMKO | NEMKO | BABT

6-Pin DIP Optoisolators Transistor Output

4N25*
4N25A*
4N26* [CTR = 20% Min]
4N27
4N28 [CTR = 10% Min]
*Motorola Preferred Devices

The 4N25/A, 4N26, 4N27 and 4N28 devices consist of a gallium arsenide infrared emitting diode optically coupled to a monolithic silicon phototransistor detector.

- Most Economical Optoisolator Choice for Medium Speed, Switching Applications
- Meets or Exceeds All JEDEC Registered Specifications
- *To order devices that are tested and marked per VDE 0884 requirements, the suffix "V" must be included at end of part number. VDE 0884 is a test option.*

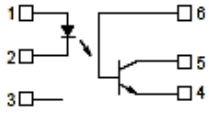
Applications

- General Purpose Switching Circuits
- Interfacing and coupling systems of different potentials and impedances
- I/O Interfacing
- Solid State Relays

STYLE 1 PLASTIC

STANDARD THRU HOLE
CASE 730A–04

SCHEMATIC



PIN 1. LED ANODE
2. LED CATHODE
3. N.C.
4. EMITTER
5. COLLECTOR
6. BASE

MAXIMUM RATINGS (T_A = 25°C unless otherwise noted)

Rating	Symbol	Value	Unit
INPUT LED			
Reverse Voltage	V_R	3	Volts
Forward Current — Continuous	I_F	60	mA
LED Power Dissipation @ T_A = 25°C with Negligible Power in Output Detector	P_D	120	mW
Derate above 25°C		1.41	mW/°C
OUTPUT TRANSISTOR			
Collector–Emitter Voltage	V_{CEO}	30	Volts
Emitter–Collector Voltage	V_{ECO}	7	Volts
Collector–Base Voltage	V_{CBO}	70	Volts
Collector Current — Continuous	I_C	150	mA
Detector Power Dissipation @ T_A = 25°C with Negligible Power in Input LED	P_D	150	mW
Derate above 25°C		1.76	mW/°C

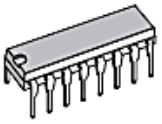
Anexo D Datasheet L293B

SGS-THOMSON MICROELECTRONICS

L293B
L293E

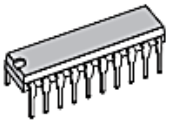
PUSH-PULL FOUR CHANNEL DRIVERS

- OUTPUT CURRENT 1A PER CHANNEL
- PEAK OUTPUT CURRENT 2A PER CHANNEL (non repetitive)
- INHIBIT FACILITY
- HIGH NOISE IMMUNITY
- SEPARATE LOGIC SUPPLY
- OVERTEMPERATURE PROTECTION



DIP16

ORDERING NUMBER : L293B



POWERDIP (16 + 2+ 2)

ORDERING NUMBER : L293E

DESCRIPTION

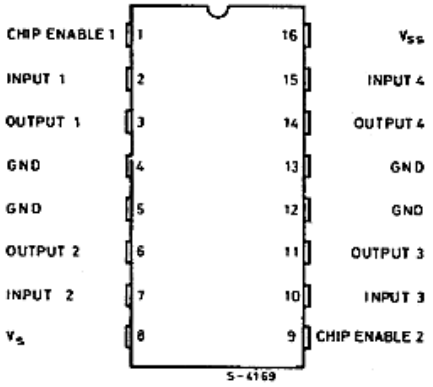
The L293B and L293E are quad push-pull drivers capable of delivering output currents to 1A per channel. Each channel is controlled by a TTL-compatible logic input and each pair of drivers (a full bridge) is equipped with an inhibit input which turns off all four transistors. A separate supply input is provided for the logic so that it may be run off a lower voltage to reduce dissipation.

Additionally, the L293E has external connection of sensing resistors, for switchmode control.

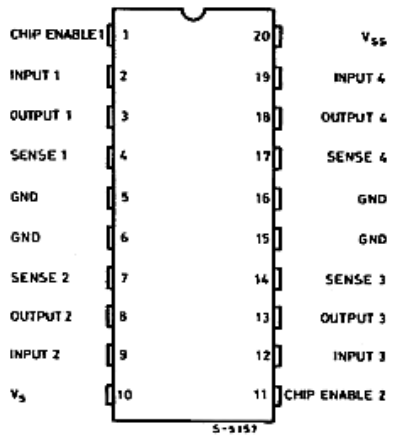
The L293B and L293E are package in 16 and 20-pin plastic DIPs respectively ; both use the four center pins to conduct heat to the printed circuit board.

PIN CONNECTIONS

DIP16 - L293B



POWERDIP (16+2+2) - L293E



ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter	Value	Unit
V_s	Supply Voltage	36	V
V_{ss}	Logic Supply Voltage	36	V
V_i	Input Voltage	7	V
V_{inh}	Inhibit Voltage	7	V
I_{out}	Peak Output Current (non repetitive t = 5ms)	2	A
P_{tot}	Total Power Dissipation at $T_{ground\text{-}pins}$ = 80°C	5	W
T_{stg}, T_j	Storage and Junction Temperature	–40 to +150	°C

THERMAL DATA

Symbol	Parameter		Value	Unit
$R_{th\ j\text{-}case}$	Thermal Resistance Junction-case	Max.	14	°C/W
$R_{th\ j\text{-}amb}$	Thermal Resistance Junction-ambient	Max.	80	°C/W

ELECTRICAL CHARACTERISTICS

For each channel, V_S = 24V, V_{SS} = 5V, T_{amb} = 25°C, unless otherwise specified

Symbol	Parameter	Test Conditions	Min.	TYp.	Max.	Unit
V_s	Supply Voltage		V_{ss}		36	V
V_{ss}	Logic Supply Voltage		4.5		36	V
I_s	Total Quiescent Supply Current	V_i = L I_o = 0 V_{inh} = H V_i = H I_o = 0 V_{inh} = H V_{inh} = L		2 16	6 24 4	mA
I_{ss}	Total Quiescent Logic Supply Current	V_i = L I_o = 0 V_{inh} = H V_i = H I_o = 0 V_{inh} = H V_{inh} = L		44 16 16	60 22 24	mA
V_{IL}	Input Low Voltage		-03.		1.5	V
V_{IH}	Input High Voltage	$V_{SS} \leq 7V$ $V_{ss} > 7V$	2.3 2.3		V_{ss} 7	V
I_{IL}	Low Voltage Input Current	V_{il} = 1.5V			-10	µA
I_{IH}	High Voltage Input Current	$2.3V \leq V_{IH} \leq V_{ss} - 0.6V$		30	100	µA
V_{InhL}	Inhibit Low Voltage		-0.3		1.5	V
V_{InhH}	Inhibit High Voltage	$V_{SS} \leq 7V$ $V_{ss} > 7V$	2.3 2.3		V_{ss} 7	V
I_{InhL}	Low Voltage Inhibit Current	V_{InhL} = 1.5V		-30	-100	µA
I_{InhH}	High Voltage Inhibit Current	$2.3V \leq V_{InhH} \leq V_{ss} - 0.6V$			±10	µA
V_{CEsatH}	Source Output Saturation Voltage	I_o = -1A		1.4	1.8	V
V_{CEsatL}	Sink Output Saturation Voltage	I_o = 1A		1.2	1.8	V
V_{SENS}	Sensing Voltage (pins 4, 7, 14, 17) (**)				2	V
t_r	Rise Time	0.1 to 0.9 V_o (*)		250		ns
t_f	Fall Time	0.9 to 0.1 V_o (*)		250		ns
t_{on}	Turn-on Delay	0.5 V_i to 0.5 V_o (*)		750		ns
t_{off}	Turn-off Delay	0.5 V_i to 0.5 V_o (*)		200		ns

* See figure 1
** Referred to L293E

Anexo E Código de programación Arduino Mega 2560

```
//Entradas              Asignación de los pines de la tarjeta Mega a las variables del
//                      programa que son entradas
int N = 36;       //    Botón Navegar
int E = 37;       //    Botón Traer
int CA = 38;      //    Sensor de prescencia de cuarto de acceso
int Seg = 39;     //    Botón Llevar
int Sen_C = 40;   //
int Sen_D = 41;   //    Sensores de presencia de los cajones A,B,C y D respectivamente
int Sen_A = 42;   //
int Sen_B = 43;   //
int RD1 = 53;     //
int RD2 = 52;     //
int RC1 = 51;     //
int RC2 = 50;     //
int RB1 = 49;     //    Sensores de nivel. Se ubican 2 en cada cajón para determinar
int RB2 = 48;     //    el nivel al que se encuentra la plataforma móvil.
int RA1 = 47;     //
int RA2 = 46;     //
int RX1 = 45;     //
int RX2 = 44;     //
//Salidas               Asignación de los pines de la tarjeta Mega a las variables del
//                      programa que son salidas
int SV = 2;       //    Semáforo
int SR = 3;       //
int DV = 22;      //    D
int DR = 23;      //
int CV = 24;      //    C
int CR = 25;      //
int BV = 26;      //    B          Indicadores led RG
int BR = 27;      //
int AV = 28;      //    A
int AR = 29;      //
int g = 30;
int punto = 31;
int c = 32;       //  Display
int d = 33;
int x = 34;
int b = 35;
```

```
//Variables locales del programa
int luminosidad = 0;
int incremento = 5;

int h;
int hl=0;
char v;
byte save;
//Configuración inicial del microcontrolador Mega
void setup(){
  //Se inicia la comunicación serie a 9600 baudios.
  Serial.begin(9600);

  //Salidas              Configuración de algunas variables del programa como salidas.
  pinMode (SV,OUTPUT);
  pinMode (SR,OUTPUT);
  pinMode (DV,OUTPUT);
  pinMode (DR,OUTPUT);
  pinMode (CV,OUTPUT);
  pinMode (CR,OUTPUT);
  pinMode (BV,OUTPUT);
  pinMode (BR,OUTPUT);
  pinMode (AV,OUTPUT);
  pinMode (AR,OUTPUT);
  pinMode (g,OUTPUT);
  pinMode (punto,OUTPUT);
  pinMode (c,OUTPUT);
  pinMode (d,OUTPUT);
  pinMode (x,OUTPUT);
  pinMode (b,OUTPUT);

  //Entradas            Configuración de algunas variables del programa como entradas.
  pinMode (N,INPUT);
  pinMode (E,INPUT);
  pinMode (CA,INPUT);
  pinMode (Seg,INPUT);
  pinMode (Sen_C,INPUT);
  pinMode (Sen_D,INPUT);
  pinMode (Sen_A,INPUT);
  pinMode (Sen_B,INPUT);
  pinMode (RD1,INPUT);
  pinMode (RD2,INPUT);
  pinMode (RC1,INPUT);
  pinMode (RC2,INPUT);
  pinMode (RB1,INPUT);
  pinMode (RB2,INPUT);
  pinMode (RA1,INPUT);
  pinMode (RA2,INPUT);
  pinMode (RX1,INPUT);
  pinMode (RX2,INPUT);
}
```

```
void loop(){                          //Ciclo que se ejecuta de forma infinita despues de
  P();                                //energizar la tarjeta Mega
  Checa_cajones();
  Checa_CA();
}
//Funciones que ejecutan el control del estacionamiento.

void Checa_cajones(){
  if(digitalRead(Sen_A)==1){
    if(digitalRead(Sen_B)==1){
      if(digitalRead(Sen_C)==1){
        if(digitalRead(Sen_D)==1){
          Semaforo_rojo();
        }
        else{
          Semaforo_verde();
        }
      }                                   //Checa los sensores de prescencia de los
      else{                               //cajones y controla las luces del semáforo.
        Semaforo_verde();
      }
    }
    else{
      Semaforo_verde();
    }
  }
  else{
    Semaforo_verde();
  }
}

void Checa_CA(){
  if(digitalRead(CA)==1){
    Semaforo_rojo();
    if(digitalRead(Seg)==1){
      char n='n';
      Asignar_cajon(n);
    }                                  //Verifica el estado del sensor del cuarto de
    else{                              //acceso para así manipular las luces del semáforo
      Navegador_principal();           //y losindicadores del tablero de control.
    }
  }
  else{
    Navegador_principal();
  }
}
```

```
void Asignar_cajon(char m){
  if(digitalRead(Sen_A)==1){
    if(digitalRead(Sen_B)==1){
      if(digitalRead(Sen_C)==1){
        if(digitalRead(Sen_D)==1){
          Apaga_todo();
          Parpadeo_alerta();
          return;
        }
        else{                              //Busca un cajón de estacionamiento vacio y
          char d='d';                      //asigna uno.De no encontrar alguno, lanza
          Llevar_auto(d,m);                //una advertencia haciendo parpadear 3 veces
          return;                          //los 4 led RG, con el color rojo.
        }
      }
      else{
        char c='c';
        Llevar_auto(c,m);
        return;
      }
    }
    else{
      char b='b';
      Llevar_auto(b,m);
      return;
    }
  }
  else{
    char a='a';
    Llevar_auto(a,m);
    return;
  }
}

void Navegador_principal(){
  if(digitalRead(N)==1){
    Navegador_secundario();
  }
  else{
    return;
  }
}
```

```
void Navegador_secundario(){
  if(digitalRead(Sen_A)==0){
    if(digitalRead(Sen_B)==0){
      if(digitalRead(Sen_C)==0){
        if(digitalRead(Sen_D)==0){
          Apaga_todo();
          Parpadeo_alerta();
          return;
        }
        else{
          int x=4;
          Navegador_aux(x);
        }
      }
      else{
        int x=3;
        Navegador_aux(x);
      }
    }
    else{
      int x=2;
      Navegador_aux(x);
    }
  }
  else{
    int x=1;
    Navegador_aux(x);
  }
}
void Navegador_aux(int x){
  int i=0;
  switch (x){
  case 1:
    if(digitalRead(Sen_A)==1){
      char a='a';
      A();
      Encender_verde(a);
      delay(1);
      Navegador_aux1(x,i,a);
    }
    else{
      x=2;
      Navegador_aux(x);
    }
    break;
```

```
case 2:
  if(digitalRead(Sen_B)==1){
    char b='b';
    B();
    Encender_verde(b);
    delay(1);
    Navegador_aux1(x,i,b);
  }
  else{
    x=3;
    Navegador_aux(x);
  }
  break;
 case 3:
   if(digitalRead(Sen_C)==1){
     char c='c';
     C();
     Encender_verde(c);
     delay(1);
     Navegador_aux1(x,i,c);
   }
   else{
     x=4;
     Navegador_aux(x);                    //Bloques encargados de verificar el estado
   }                                      //del botón Navegar y mostrar en los
   break;                                 //indicadores led la letra que se solicita.
 case 4:
   if(digitalRead(Sen_D)==1){
     char d='d';
     D();
     Encender_verde(d);
     delay(1);
     Navegador_aux1(x,i,d);
   }
   else{
     x=1;
     Navegador_aux(x);
   }
   break;
  }
}
```

```
void Navegador_aux1(int x,int i,char z){
  if(digitalRead(E)==1){
    int j=0;
    Leds_bajo();
    Encender_naranja(z);
    Traer_seguro(z,j);
  }
  else{
    delay(130);
    if(i>100){
      Apaga_todo();
      return;
    }
    else{
      i++;
    }
    if(digitalRead(N)==1){
      if(x<4){
        x++;
        Navegador_aux(x);
      }
      else{
        Navegador_secundario();
      }
    }
    else{
      Navegador_aux1(x,i,z);
    }
  }
```

```
void Traer_seguro(char x,int j){
  if(digitalRead(CA)==1){
    Semaforo_rojo();
    if(digitalRead(Seg)==1){
      Asignar_cajon(x);
    }
    else{
      delay(1);
      if(j<500){
        j++;
        Encender_naranja(x);
      }
      else{
        if(j<1000){
          Apagar_naranja(x);
          j++;
        }
        else{
          j=0;
        }
      }
      Traer_seguro(x,j);
    }
  }
  else{
    delay(10);                          //Este delay y este if regulan el tiempo para
    if(j>1000){                         //verificar que no llegue un usuario al CA.
      Leds_bajo();
      Traer_auto(x);
    }
    else{
      j++;
      Traer_seguro(x,j);                 //Lineas de programa cuya función es la de
    }                                    //manipular los led indicadores y el display,
  }                                      //cuando se está ejecutando la secuancia de
}                                        //traer automóvil de algún cajón.
```

```
void Traer_auto(char m){
  Leds_bajo();
  Encender_verde(m);
  switch (m){
  case 'a':
    Semaforo_rojo();
    A();
    Posicion_inicial3(m);
    break;
  case 'b':
    Semaforo_rojo();
    B();
    Posicion_inicial3(m);
    break;
  case 'c':
    Semaforo_rojo();
    C();
    Posicion_inicial3(m);
    break;
  case 'd':
    Semaforo_rojo();
    D();
    Posicion_inicial3(m);
    break;
  case 'n':
    return;
  }
}
void Posicion_inicial3(char z){
  Imprime_caracterl(z);
  Estado_de_los_sensores();
  Codificador(h);                    //Verifica por medio de los sensores, que
  Actualizar_estado(h);              //la estructura móvil se encuentre en la
  Apaga_todo();                      //posición inicial para traer un vehículo.
  return;
}
```

```
void Imprime_caracterl(char x){
  switch (x){
  case 'a':
    Serial.println('Q');
    delay(3000);
    break;
  case 'b':                                   //Envía un caracter al Arduino Leonardo
    Serial.println('W');                      //por medio de los modulos Xbee para que
    delay(3000);                              //este ejecute secuancias en los motores
    break;                                    //para traer automóviles.
  case 'c':
    Serial.println('E');
    delay(3000);
    break;
  case 'd':
    Serial.println('R');
    delay(3000);
    break;
  }
}
void Llevar_auto(char x,char y){
  Leds_bajo();
  Encender_rojo(x);
  switch (x){
  case 'a':
    A();
    Posicion_inicial(x,y);
    break;
  case 'b':
    B();                                  //Lineas de código cuya finalidad es la de
    Posicion_inicial(x,y);                //manipular los led indicadores y el display,
    break;                                //cuando se está ejecutando la sesuencia de
  case 'c':                               //llevar automóvil a algún cajón X.
    C();
    Posicion_inicial(x,y);
    break;
  case 'd':
    D();
    Posicion_inicial(x,y);
    break;
  }
}
```

```
void Posicion_inicial(char z,char y){
  Imprime_caracter(z);
  Estado_de_los_sensores();
  Codificador(h);                          //Verifica que la estructura móvil se
  Actualizar_estado(h);                    //encuentre en la posición inicial para
  Apaga_todo();                            //llevar un automóvil.
  delay(1000);
  Traer_auto(y);
}
void Imprime_caracter(char x){
  switch (x){
  case 'a':
    Serial.print('P');
    delay(3000);
    break;
  case 'b':
    Serial.println('O');                   //Envía un caracter al Arduino Leonardo por
    delay(3000);                           //medio de los modulos Xbee para que este
    break;                                 //ejecute cierta secuencia en los motores
  case 'c':                                //para llevar un automóvil a un cajón de
    Serial.println('I');                   //estacionamiento. La secuencia ejecutada
    delay(3000);                           //por la tarjeta Leonardo va a depender del
    break;                                 //caracter enviado en estas lineas de programa.
  case 'd':
    Serial.println('U');
    delay(3000);
    break;
  }
}
 int Estado_de_los_sensores(){
   if(digitalRead(RX1)==1){
     bitWrite(h,0,1);
   }
   else{
     bitWrite(h,0,0);
   }
   if(digitalRead(RX2)==1){
     bitWrite(h,1,1);
   }
   else{
     bitWrite(h,1,0);
   }
```

```
if(digitalRead(RA1)==1){
  bitWrite(h,2,1);
}
else{                                   //Lee el estado de los sensores de nivel de
  bitWrite(h,2,0);                      //cada cajón y los datos obtenidos los almacena
}                                       //en un byte.
if(digitalRead(RA2)==1){
  bitWrite(h,3,1);
}
else{
  bitWrite(h,3,0);
}
if(digitalRead(RB1)==1){
  bitWrite(h,4,1);
}
  else{
    bitWrite(h,4,0);
  }
  if(digitalRead(RB2)==1){
    bitWrite(h,5,1);
  }
  else{
    bitWrite(h,5,0);
  }
  if(digitalRead(RC1)==1){
    bitWrite(h,6,1);
  }
  else{
    bitWrite(h,6,0);
  }
  if(digitalRead(RC2)==1){
    bitWrite(h,7,1);
  }
  else{
    bitWrite(h,7,0);
  }
  if(digitalRead(RD1)==1){
    bitWrite(h,8,1);
  }
  else{
    bitWrite(h,8,0);
  }
```

```
  if(digitalRead(RD2)==1){
    bitWrite(h,9,1);
  }
  else{
    bitWrite(h,9,0);
  }
  return h;
}
void Codificador(int h){
  save=bitRead(h,0);
  Serial.print(save,BIN);
  save=bitRead(h,1);
  Serial.print(save,BIN);
  save=bitRead(h,2);
  Serial.print(save,BIN);
  save=bitRead(h,3);
  Serial.print(save,BIN);              //Envía el estado de cada uno de los sensores
  save=bitRead(h,4);                   //al Arduino Leonardo mediante los modulos Xbee.
  Serial.print(save,BIN);
  save=bitRead(h,5);
  Serial.print(save,BIN);
  save=bitRead(h,6);
  Serial.print(save,BIN);
  save=bitRead(h,7);
  Serial.print(save,BIN);
  save=bitRead(h,8);
  Serial.print(save,BIN);
  save=bitRead(h,9);
  Serial.print(save,BIN);
}
void Actualizar_estado(long hl){
  Estado_de_los_sensores();
  if(Serial.available()>0){
    v=Serial.read();
    if(v=='T'){
      return;
    }                                  //Refresca la lectura del estado de los
  }                                    //sensores  del Arduino Mega mientras se
  if(hl==h){                           //está ejecutando una secuencia, como
    Actualizar_estado(hl);             //puede ser traer o llevar automóvil.
  }
  else{
    Codificador(h);
    Actualizar_estado(h);
  }
}
```

```
//A continuación se muestran las funciones encargadas de gestionar el color de los led
//del panel de control.
void Semaforo_verde(){
  digitalWrite(SR,LOW);                    //Ilumina el semáforo con el color verde.
  digitalWrite(SV,HIGH);
}

void Semaforo_rojo(){
  digitalWrite(SR,HIGH);                   //Ilumina el semáforo con el color verde.
  digitalWrite(SV,LOW);
}
void Parpadeo_alerta(){
  Leds_alto();
  delay(200);
  Leds_bajo();
  delay(200);
  Leds_alto();
  delay(200);
  Leds_bajo();                             //Hace parpadear los indicadores led en
  delay(200);                              //color rojo.
  Leds_alto();
  delay(200);
  Leds_bajo();
  delay(200);
  Leds_alto();
  delay(200);
  Leds_bajo();
  delay(200);
}
void Leds_alto(){
  digitalWrite(AR,HIGH);
  digitalWrite(BR,HIGH);                   //Enciende todos los indicadores led del
  digitalWrite(CR,HIGH);                   //panel de control.
  digitalWrite(DR,HIGH);
}

void Leds_bajo(){
  digitalWrite(AR,LOW);
  digitalWrite(AV,LOW);
  digitalWrite(BR,LOW);
  digitalWrite(BV,LOW);                  //Apaga todos los indicadores led del panel
  digitalWrite(CR,LOW);                  //de control.
  digitalWrite(CV,LOW);
  digitalWrite(DR,LOW);
  digitalWrite(DV,LOW);
}
```

```
void Encender_verde(char y){
  switch (y){
  case 'a':
    digitalWrite(AV,HIGH);
    digitalWrite(AR,LOW);
    digitalWrite(BV,LOW);
    digitalWrite(BR,LOW);
    digitalWrite(CV,LOW);
    digitalWrite(CR,LOW);
    digitalWrite(DV,LOW);
    digitalWrite(DR,LOW);
    break;
  case 'b':
    digitalWrite(AV,LOW);
    digitalWrite(AR,LOW);
    digitalWrite(BV,HIGH);
    digitalWrite(BR,LOW);
    digitalWrite(CV,LOW);
    digitalWrite(CR,LOW);
    digitalWrite(DV,LOW);
    digitalWrite(DR,LOW);                    //Enciende en color verde el indicador
    break;                                   //led requerido A,B,C o D.
  case 'c':
    digitalWrite(AV,LOW);
    digitalWrite(AR,LOW);
    digitalWrite(BV,LOW);
    digitalWrite(BR,LOW);
    digitalWrite(CV,HIGH);
    digitalWrite(CR,LOW);
    digitalWrite(DV,LOW);
    digitalWrite(DR,LOW);
    break;
  case 'd':
    digitalWrite(AV,LOW);
    digitalWrite(AR,LOW);
    digitalWrite(BV,LOW);
    digitalWrite(BR,LOW);
    digitalWrite(CV,LOW);
    digitalWrite(CR,LOW);
    digitalWrite(DV,HIGH);
    digitalWrite(DR,LOW);
    break;
  }
}
```

```
void Encender_naranja(char y){
  switch (y){
  case 'a':
    digitalWrite(AV,HIGH);
    digitalWrite(AR,HIGH);
    break;
  case 'b':
    digitalWrite(BV,HIGH);
    digitalWrite(BR,HIGH);                    //Enciende en color naranja el indicador
    break;                                    //led requerido A,B,C o D.
  case 'c':
    digitalWrite(CV,HIGH);
    digitalWrite(CR,HIGH);
    break;
  case 'd':
    digitalWrite(DV,HIGH);
    digitalWrite(DR,HIGH);
    break;
  }
}
void Apagar_naranja(char y){
  switch (y){
  case 'a':
    digitalWrite(AV,LOW);
    digitalWrite(AR,LOW);
    break;
  case 'b':
    digitalWrite(BV,LOW);
    digitalWrite(BR,LOW);
    break;                                     //Apaga el color naranja del indicador
  case 'c':                                    //led requerido A,B,C o D.
    digitalWrite(CV,LOW);
    digitalWrite(CR,LOW);
    break;
  case 'd':
    digitalWrite(DV,LOW);
    digitalWrite(DR,LOW);
    break;
  }
}
```

```
void Encender_rojo(char y){
  switch (y){
  case 'a':
    digitalWrite(AR,HIGH);
    digitalWrite(AV,LOW);
    digitalWrite(BV,LOW);
    digitalWrite(CV,LOW);
    digitalWrite(DV,LOW);
    break;
  case 'b':
    digitalWrite(BR,HIGH);
    digitalWrite(BV,LOW);
    digitalWrite(AV,LOW);
    digitalWrite(CV,LOW);
    digitalWrite(DV,LOW);
    break;                                  //Enciende en color rojo el indicador
  case 'c':                                 //led requerido A,B,C o D.
    digitalWrite(CR,HIGH);
    digitalWrite(CV,LOW);
    digitalWrite(AV,LOW);
    digitalWrite(BV,LOW);
    digitalWrite(DV,LOW);
    break;
  case 'd':
    digitalWrite(DR,HIGH);
    digitalWrite(DV,LOW);
    digitalWrite(AV,LOW);
    digitalWrite(BV,LOW);
    digitalWrite(CV,LOW);
    break;
  }
}
 void A(){
   digitalWrite(x, HIGH);
   digitalWrite(b, HIGH);
   digitalWrite(c, HIGH);                   //Muestra en el display la letra "A"
   digitalWrite(d, LOW);
   digitalWrite(g, HIGH);
   digitalWrite(punto, LOW);
 }
```

```
void B(){
  digitalWrite(x, HIGH);
  digitalWrite(b, HIGH);
  digitalWrite(c, HIGH);                          //Muestra en el display la letra "B"
  digitalWrite(d, HIGH);
  digitalWrite(g, HIGH);
  digitalWrite(punto, LOW);
}
void C(){
  digitalWrite(x, HIGH);
  digitalWrite(b, LOW);
  digitalWrite(c, LOW);                           //Muestra en el display la letra "C"
  digitalWrite(d, HIGH);
  digitalWrite(g, LOW);
  digitalWrite(punto, LOW);
}
void D(){
  digitalWrite(x, HIGH);
  digitalWrite(b, HIGH);
  digitalWrite(c, HIGH);                           //Muestra en el display la letra "D"
  digitalWrite(d, HIGH);
  digitalWrite(g, LOW);
  digitalWrite(punto, LOW);
}

void P(){
  Apaga_todo();
  analogWrite(punto, luminosidad);                  //Muestra una intermitencia en el
  luminosidad = luminosidad + incremento;           //punto del display del panel de
  if (luminosidad == 0 || luminosidad == 255) {     //control cuando el sistema se
    incremento = -incremento ;                      //encuentra en reposo.
  }
  delay(30);
}
void Apaga_todo(){
  digitalWrite(x, LOW);
  digitalWrite(b, LOW);
  digitalWrite(c, LOW);
  digitalWrite(d, LOW);
  digitalWrite(g, LOW);
  digitalWrite(punto, LOW);
  digitalWrite(AV, LOW);                            //Apaga todo el tablero de control,
  digitalWrite(AR, LOW);                            //led indicadores y display.
  digitalWrite(BV, LOW);
  digitalWrite(BR, LOW);
  digitalWrite(CV, LOW);
  digitalWrite(CR, LOW);
  digitalWrite(DV, LOW);
  digitalWrite(DR, LOW);
}
```

Anexo F Código de programación Arduino Leonardo

```
//Entradas              Asignación de los pines de la tarjeta Leonardo a las variables
//                      del programa que son entradas.
int LI=2;           //  Interruptor de límite que sirve para detectar que
                    //  la plataforma móvil está adentro de la estructura móvil.
int LO=3;           //  Interruptor de límite que sirve para detectar que
                    //  la plataforma móvil está afuera de la estructura móvil.
//Salidas               Asignación de los pines de la tarjeta Leonardo a las variables
//                      del programa que son salidas.
int M1Out=9;        //  M1: Encargado de darle movimiento a la plataforma móvil.
int M1In=10;
int M2Up=7;         //  M2: Encargado de darle movimiento a la cabina.
int M2Down=8;
int M3Left=5;       //  M3: Encargado de hacer girar en sentido levógiro y dextrógiro
int M3Right=6;      //  a la estructura móvil.
//Variables locales del programa
int i;
char m;
int j=0;
byte k=0;
byte k1;
//Configuración inicial del microcontrolador Leonardo
void setup() {
  //Se inicia la comunicación serie a 9600 baudios.
  Serial.begin(9600);
  Serial1.begin(9600);
  //Entradas            Configuración de variables del programa como entradas.
  pinMode(LI,INPUT);
  pinMode(LO,INPUT);
  //Salidas             Configuración de algunas variables del programa como salidas.
  pinMode(M1Out,OUTPUT);
  pinMode(M1In,OUTPUT);
  pinMode(M2Up,OUTPUT);
  pinMode(M2Down,OUTPUT);
  pinMode(M3Left,OUTPUT);
  pinMode(M3Right,OUTPUT);
}
void loop(){
  if(Serial1.available()>0){
    m=Serial1.read();             //Ciclo que se ejecuta de forma infinita despues de
    Serial.println(m);            //energizar la tarjeta Leonardo.
    int x=0;
    Checa_secuencia(m,x);
  }
}
```

```
void Checa_secuencia(char m,int x){
  switch(m){
  //Funciones para llevar auto.
    case 'P':
      Llevar_auto_A(x);
      break;
    case 'O':
      Llevar_auto_B(x);
      break;
    case 'I':
      Llevar_auto_C(x);
      break;
    case 'U':
      Llevar_auto_D(x);
      break;
  //Funciones para traer auto.          Codifica el caracter que le envía el Arduino
    case 'Q':                          //Mega con el fin de que el Arduino Leonardo
      Traer_auto_A(x);                 //determine la secuencia que van a ejecutar los
      break;                           //motores.
    case 'W':
      Traer_auto_B(x);
      break;
    case 'E':
      Traer_auto_C(x);
      break;
    case 'R':
      Traer_auto_D(x);
      break;
  }
}
int Checa_sensores_de_mega(int x){
  while(x<10){
    if(Serial1.available()>0){
      i=Serial1.read();
      if(i==48){
        bitWrite(j,x,0);                    //Codifica los caracteres que fueron
        x++;                                //enviados por el Arduino Mega para saber
      }                                     //el estado de los sensores IR.
      if(i==49){
        bitWrite(j,x,1);
        x++;
      }
    }
  }
  //Serial.println(j,BIN);
  return j;
}
```

```
int Estado_de_sensores_de_limite(){
  if(digitalRead(LI)==1){
    bitWrite(k,1,1);
  }else{
    bitWrite(k,1,0);
  }
  if(digitalRead(LO)==1){                    //Verifica el estado de los sensores de
    bitWrite(k,0,1);                         //límite.
  }else{
    bitWrite(k,0,0);
  }
  return k;
}
void Llevar_auto_A(int x){
  while(j!=Bl0){
    Checa_sensores_de_mega(x);
  }
  while((k==Bl0)==0){
    Estado_de_sensores_de_limite();
  }
  digitalWrite(M1Out,HIGH);
  while((k==Bl)==0){
    Estado_de_sensores_de_limite();
  }
  digitalWrite(M1Out,LOW);
  delay(1000);
  digitalWrite(M2Up,HIGH);
  while(j!=Bl){
    Checa_sensores_de_mega(x);
  }
  digitalWrite(M2Up,LOW);
  delay(1000);
  digitalWrite(M1In,HIGH);
  while((k==Bl0)==0){
    Estado_de_sensores_de_limite();
  }
  digitalWrite(M1In,LOW);                    //Ejecuta la secuencia en los motores
  delay(1000);                               //con el fin de llevar un automóvil al
  digitalWrite(M3Right,HIGH);                //cajón A.
  while(j!=Bl00){
    Checa_sensores_de_mega(x);
```

```
  }
  digitalWrite(M3Right,LOW);
  delay(1000);
  digitalWrite(M1Out,HIGH);
  while((k==B1)==0){
    Estado_de_sensores_de_limite();
  }
  digitalWrite(M1Out,LOW);
  delay(1000);
  digitalWrite(M2Down,HIGH);
  while(j!=B1000){
    Checa_sensores_de_mega(x);
  }
  digitalWrite(M2Down,LOW);
  delay(1000);
  digitalWrite(M1In,HIGH);
  while((k==B10)==0){
    Estado_de_sensores_de_limite();
  }
  digitalWrite(M1In,LOW);
  delay(1000);
  digitalWrite(M3Left,HIGH);
  while(j!=B10){
    Checa_sensores_de_mega(x);
  }
  digitalWrite(M3Left,LOW);
  Serial1.print('T');
}
void Llevar_auto_B(int x){
  while(j!=B10){
    Checa_sensores_de_mega(x);
  }
  while((k==B10)==0){
    Estado_de_sensores_de_limite();
  }
  digitalWrite(M1Out,HIGH);
  while((k==B1)==0){
    Estado_de_sensores_de_limite();
  }
  digitalWrite(M1Out,LOW);
  delay(1000);
  digitalWrite(M2Up,HIGH);
  while(j!=B1){
    Checa_sensores_de_mega(x);
  }
```

```
digitalWrite(M2Up,LOW);
delay(1000);
digitalWrite(M1In,HIGH);
while((k==B10)==0){
  Estado_de_sensores_de_limite();
}
digitalWrite(M1In,LOW);
delay(1000);
digitalWrite(M3Left,HIGH);
while(j!=B10000){
  Checa_sensores_de_mega(x);
}
  digitalWrite(M3Left,LOW);                    //Ejecuta la secuencia en los motores
  delay(1000);                                 //con el fin de llevar un automóvil al
  digitalWrite(M1Out,HIGH);                    //cajón B.
  while((k==B1)==0){
    Estado_de_sensores_de_limite();
  }
  digitalWrite(M1Out,LOW);
  delay(1000);
  digitalWrite(M2Down,HIGH);
  while(j!=B100000){
    Checa_sensores_de_mega(x);
  }
  digitalWrite(M2Down,LOW);
  delay(1000);
  digitalWrite(M1In,HIGH);
  while((k==B10)==0){
    Estado_de_sensores_de_limite();
  }
  digitalWrite(M1In,LOW);
  delay(1000);
  digitalWrite(M3Right,HIGH);
  while(j!=B10){
    Checa_sensores_de_mega(x);
  }
  digitalWrite(M3Right,LOW);
  Serial1.print('T');
}
```

```
void Llevar_auto_C(int x){
  while(j!=B10){
    Checa_sensores_de_mega(x);
  }
  while((k==B10)==0){
    Estado_de_sensores_de_limite();
  }
  digitalWrite(M1Out,HIGH);
  while((k==B1)==0){
    Estado_de_sensores_de_limite();
  }
  digitalWrite(M1Out,LOW);
  delay(1000);
  digitalWrite(M2Up,HIGH);
  while(j!=B1){
    Checa_sensores_de_mega(x);
  }
  digitalWrite(M2Up,LOW);
  delay(1000);
  digitalWrite(M1In,HIGH);
  while((k==B10)==0){
    Estado_de_sensores_de_limite();
  }
  digitalWrite(M1In,LOW);                    //Ejecuta la secuencia en los motores
  delay(1000);                               //con el fin de llevar un automóvil al
  digitalWrite(M3Right,HIGH);                //cajón C.
  while(j!=B100){
    Checa_sensores_de_mega(x);
  }
  digitalWrite(M3Right,LOW);
  delay(1000);
  digitalWrite(M2Up,HIGH);
  while(j!=B1000000){
    Checa_sensores_de_mega(x);
  }
  digitalWrite(M2Up,LOW);
  delay(1000);
  digitalWrite(M1Out,HIGH);
  while((k==B1)==0){
    Estado_de_sensores_de_limite();
  }
  digitalWrite(M1Out,LOW);
  delay(1000);
  digitalWrite(M2Down,HIGH);
  while(j!=B10000000){
    Checa_sensores_de_mega(x);
  }
```

```
  digitalWrite(M2Down,LOW);
  delay(1000);
  digitalWrite(M1In,HIGH);
  while((k==B10)==0){
    Estado_de_sensores_de_limite();
  }
  digitalWrite(M1In,LOW);
  delay(1000);
  digitalWrite(M2Down,HIGH);
  while(j!=B1000){
    Checa_sensores_de_mega(x);
  }
  digitalWrite(M2Down,LOW);
  delay(1000);
  digitalWrite(M3Left,HIGH);
  while(j!=B10){
    Checa_sensores_de_mega(x);
  }
  digitalWrite(M3Left,LOW);
  Serial1.print('T');
}
void Llevar_auto_D(int x){
  while(j!=B10){
    Checa_sensores_de_mega(x);
  }
  while((k==B10)==0){
    Estado_de_sensores_de_limite();
  }
  digitalWrite(M1Out,HIGH);
  while((k==B1)==0){
    Estado_de_sensores_de_limite();
  }
  digitalWrite(M1Out,LOW);
  delay(1000);
  digitalWrite(M2Up,HIGH);
  while(j!=B1){
    Checa_sensores_de_mega(x);
  }
  digitalWrite(M2Up,LOW);
  delay(1000);
  digitalWrite(M1In,HIGH);
  while((k==B10)==0){
    Estado_de_sensores_de_limite();
  }
```

```
  digitalWrite(M1In,LOW);
  delay(1000);
  digitalWrite(M3Left,HIGH);
  while(j!=B10000){
    Checa_sensores_de_mega(x);
  }
  digitalWrite(M3Left,LOW);                    //Ejecuta la secuencia en los motores
  delay(1000);                                 //con el fin de llevar un automóvil al
  digitalWrite(M2Up,HIGH);                     //cajón D.
  while(j!=256){
    Checa_sensores_de_mega(x);
  }
  digitalWrite(M2Up,LOW);
  delay(1000);
  digitalWrite(M1Out,HIGH);
  while((k==B1)==0){
    Estado_de_sensores_de_limite();
  }
  digitalWrite(M1Out,LOW);
  delay(1000);
  digitalWrite(M2Down,HIGH);
  while(j!=512){
    Checa_sensores_de_mega(x);
  }
  digitalWrite(M2Down,LOW);
  delay(1000);
  digitalWrite(M1In,HIGH);
  while((k==B10)==0){
    Estado_de_sensores_de_limite();
  }
  digitalWrite(M1In,LOW);
  delay(1000);
  digitalWrite(M2Down,HIGH);
  while(j!=B100000){
    Checa_sensores_de_mega(x);
  }
  digitalWrite(M2Down,LOW);
  delay(1000);
  digitalWrite(M3Right,HIGH);
  while(j!=B10){
    Checa_sensores_de_mega(x);
  }
  digitalWrite(M3Right,LOW);
  Serial1.print('T');
}
```

```
void Traer_auto_A(int x){
  while(j!=Bl0){
    Checa_sensores_de_mega(x);
  }
  while(k!=Bl0){
    Estado_de_sensores_de_limite();
  }
  digitalWrite(M3Right,HIGH);
  while(j!=Bl000){
    Checa_sensores_de_mega(x);
  }
  digitalWrite(M3Right,LOW);
  delay(1000);
  digitalWrite(Ml0ut,HIGH);
  while(k!=Bl){
    Estado_de_sensores_de_limite();
  }
  digitalWrite(Ml0ut,LOW);
  delay(1000);
  digitalWrite(M2Up,HIGH);
  while(j!=Bl00){
    Checa_sensores_de_mega(x);
  }
  digitalWrite(M2Up,LOW);
  delay(1000);
  digitalWrite(MlIn,HIGH);
  while(k!=Bl0){
    Estado_de_sensores_de_limite();
  }
  digitalWrite(MlIn,LOW);                    //Ejecuta la secuencia en los motores
  delay(1000);                               //con el fin de traer un automóvil del
  digitalWrite(M3Left,HIGH);                 //cajón A.
  while(j!=Bl){
    Checa_sensores_de_mega(x);
  }
  digitalWrite(M3Left,LOW);
  delay(1000);
  digitalWrite(Ml0ut,HIGH);
  while(k!=Bl){
    Estado_de_sensores_de_limite();
  }
  digitalWrite(Ml0ut,LOW);
  delay(1000);
  digitalWrite(M2Down,HIGH);
  while(j!=Bl0){
    Checa_sensores_de_mega(x);
  }
```

```
  digitalWrite(M2Down,LOW);
  delay(1000);
  digitalWrite(M1In,HIGH);
  while(k!=B10){
    Estado_de_sensores_de_limite();
  }
  digitalWrite(M1In,LOW);
  Serial1.print('T');
}
void Traer_auto_B(int x){
  while(j!=B10){
    Checa_sensores_de_mega(x);
  }
  while(k!=B10){
    Estado_de_sensores_de_limite();
  }
  digitalWrite(M3Left,HIGH);
  while(j!=B100000){
    Checa_sensores_de_mega(x);
  }
  digitalWrite(M3Left,LOW);
  delay(1000);
  digitalWrite(M1Out,HIGH);
  while(k!=B1){
    Estado_de_sensores_de_limite();
  }
  digitalWrite(M1Out,LOW);
  delay(1000);
  digitalWrite(M2Up,HIGH);
  while(j!=B10000){
    Checa_sensores_de_mega(x);
  }
  digitalWrite(M2Up,LOW);
  delay(1000);
  digitalWrite(M1In,HIGH);
  while(k!=B10){
    Estado_de_sensores_de_limite();
  }
```

```
  digitalWrite(M1In,LOW);                    //Ejecuta la secuencia en los motores
  delay(1000);                               //con el fin de traer un automóvil del
  digitalWrite(M3Right,HIGH);                //cajón B.
  while(j!=B1){
    Checa_sensores_de_mega(x);
  }
  digitalWrite(M3Right,LOW);
  delay(1000);
  digitalWrite(M1Out,HIGH);
  while(k!=B1){
    Estado_de_sensores_de_limite();
  }
  digitalWrite(M1Out,LOW);
  delay(1000);
  digitalWrite(M2Down,HIGH);
  while(j!=B10){
    Checa_sensores_de_mega(x);
  }
  digitalWrite(M2Down,LOW);
  delay(1000);
  digitalWrite(M1In,HIGH);
  while(k!=B10){
    Estado_de_sensores_de_limite();
  }
  digitalWrite(M1In,LOW);
  Serial1.print('T');
}
   void Traer_auto_C(int x){
     while(j!=B10){
       Checa_sensores_de_mega(x);
     }
     while(k!=B10){
       Estado_de_sensores_de_limite();
     }
     digitalWrite(M3Right,HIGH);
     while(j!=B1000){
       Checa_sensores_de_mega(x);
     }
     digitalWrite(M3Right,LOW);
     delay(1000);
     digitalWrite(M2Up,HIGH);
     while(j!=B10000000){
       Checa_sensores_de_mega(x);
     }
```

```
digitalWrite(M2Up,LOW);
delay(1000);
digitalWrite(M1Out,HIGH);
while(k!=B1){
  Estado_de_sensores_de_limite();
}
digitalWrite(M1Out,LOW);
delay(1000);
digitalWrite(M2Up,HIGH);
while(j!=B1000000){
  Checa_sensores_de_mega(x);
}
digitalWrite(M2Up,LOW);                    //Ejecuta la secuencia en los motores
delay(1000);                               //con el fin de traer un automóvil del
digitalWrite(M1In,HIGH);                   //cajón C.
while(k!=B10){
  Estado_de_sensores_de_limite();
}
digitalWrite(M1In,LOW);
delay(1000);
digitalWrite(M2Down,HIGH);
while(j!=B100){
  Checa_sensores_de_mega(x);
}
digitalWrite(M2Down,LOW);
delay(1000);
digitalWrite(M3Left,HIGH);
while(j!=B1){
  Checa_sensores_de_mega(x);
}
digitalWrite(M3Left,LOW);
delay(1000);
digitalWrite(M1Out,HIGH);
while(k!=B1){
  Estado_de_sensores_de_limite();
}
```

```
  digitalWrite(M1Out,LOW);
  delay(1000);
  digitalWrite(M2Down,HIGH);
  while(j!=B10){
    Checa_sensores_de_mega(x);
  }
  digitalWrite(M2Down,LOW);
  delay(1000);
  digitalWrite(M1In,HIGH);
  while(k!=B10){
    Estado_de_sensores_de_limite();
  }
  digitalWrite(M1In,LOW);
  Serial1.print('T');
}
void Traer_auto_D(int x){
  while(j!=B10){
    Checa_sensores_de_mega(x);
  }
  while(k!=B10){
    Estado_de_sensores_de_limite();
  }
  digitalWrite(M3Left,HIGH);
  while(j!=B100000){
    Checa_sensores_de_mega(x);
  }
  digitalWrite(M3Left,LOW);
  delay(1000);
  digitalWrite(M2Up,HIGH);
  while(j!=512){
    Checa_sensores_de_mega(x);
  }
  digitalWrite(M2Up,LOW);
  delay(1000);
  digitalWrite(M1Out,HIGH);
  while(k!=B1){
    Estado_de_sensores_de_limite();
  }
  digitalWrite(M1Out,LOW);
  delay(1000);
  digitalWrite(M2Up,HIGH);
  while(j!=256){
    Checa_sensores_de_mega(x);
  }
```

```
  digitalWrite(M2Up,LOW);                    //Ejecuta la secuencia en los motores
  delay(1000);                               //con el fin de traer un automóvil del
  digitalWrite(M1In,HIGH);                   //cajón D.
  while(k!=B10){
    Estado_de_sensores_de_limite();
  }
  digitalWrite(M1In,LOW);
  delay(1000);
  digitalWrite(M2Down,HIGH);
  while(j!=B10000){
    Checa_sensores_de_mega(x);
  }
  digitalWrite(M2Down,LOW);
  delay(1000);
  digitalWrite(M3Right,HIGH);
  while(j!=B1){
    Checa_sensores_de_mega(x);
  }
  digitalWrite(M3Right,LOW);
  delay(1000);
  digitalWrite(M1Out,HIGH);
  while(k!=B1){
    Estado_de_sensores_de_limite();
  }
  digitalWrite(M2Down,LOW);
  delay(1000);
  digitalWrite(M1In,HIGH);
  while(k!=B10){
    Estado_de_sensores_de_limite();
  }
  digitalWrite(M1In,LOW);
  Serial1.print('T');
}
```