Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería Mecánica Eléctrica

DISEÑO E IMPLEMENTACIÓN DE UN ROBOT MÓVIL TIPO EXPLORADOR PARA FINES ACADÉMICOS

Rodrigo Otoniel Méndez Canú
Asesorado por el Ing. Carlos Eduardo Guzmán Salazar

Guatemala, septiembre de 2019

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

DISEÑO E IMPLEMENTACIÓN DE UN ROBOT MÓVIL TIPO EXPLORADOR PARA FINES ACADÉMICOS

TRABAJO DE GRADUACIÓN

PRESENTADO A LA JUNTA DIRECTIVA DE LA
FACULTAD DE INGENIERÍA
POR

RODRIGO OTONIEL MÉNDEZ CANÚ
ASESORADO POR EL ING. CARLOS EDUARDO GUZMAN SALAZAR

AL CONFERÍRSELE EL TÍTULO DE

INGENIERO EN ELECTRÓNICA

GUATEMALA, SEPTIEMBRE DE 2019

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

NÓMINA DE JUNTA DIRECTIVA

DECANA	Inga. Aurelia Anabela Cordova Estrada
VOCAL I	Ing. José Francisco Gómez Rivera
VOCAL II	Ing. Mario Renato Escobedo Martínez
VOCAL III	Ing. José Milton de León Bran
VOCAL IV	Br. Luis Diego Aguilar Ralón
VOCAL V	Br. Christian Daniel Estrada Santizo
SECRETARIO	Ing. Hugo Humberto Rivera Pérez

TRIBUNAL QUE PRACTICÓ EL EXAMEN GENERAL PRIVADO

DECANO	Ing. Pedro Antonio Aguilar Polanco
EXAMINADOR	Ing. Julio Rolando Barrios Archila
EXAMINADOR	Ing. Armando Alonso Rivera Carrillo
EXAMINADOR	Ing. Julio César Solares Peñate
SECRETARIA	Inga. Lesbia Magalí Herrera López

HONORABLE TRIBUNAL EXAMINADOR

En cumplimiento con los preceptos que establece la ley de la Universidad de San Carlos de Guatemala, presento a su consideración mi trabajo de graduación titulado:

DISEÑO E IMPLEMENTACIÓN DE UN ROBOT MÓVIL TIPO EXPLORADOR PARA FINES ACADÉMICOS

Tema que me fuera asignado por la Dirección de la Escuela de Ingeniería Mecánica Eléctrica, con fecha 7 de noviembre de 2018.

Rodrigo Otoniel Méndez Canú

Guatemala, 16 de mayo de 2019

Ingeniero
Julio Solares Peñate
Coordinador Área de Electrónica
Escuela de Ingeniería Mecánica Eléctrica
Facultad de Ingeniería
Universidad de San Carlos de Guatemala

Estimado ingeniero Solares:

Hago de su conocimiento por este medio que he concluido la revisión del trabajo de graduación del estudiante **Rodrigo Otoniel Méndez Canú**, titulado **DISEÑO E IMPLEMENTACIÓN DE UN ROBOT MÓVIL TIPO EXPLORADOR PARA FINES ACADÉMICOS.** Considero que el mismo ha cumplido con los objetivos que se propusieron para su ejecución. Por lo que, doy mi APROBACIÓN al mismo como ASESOR nombrado por la Escuela de Ingeniería Mecánica Eléctrica.

Así mismo, índico que tanto el estudiante Méndez Canú como el suscrito, somos responsables del contenido del trabajo de graduación referido.

Reciba un cordial saludo,

Carlos Guzmán Salazar
ASESOR

CARLOS GUZMAN SALAZAR
Ingeniero Electricista
Col. No. 2762

UNIVERSIDAD DE SAN CARLOS
DE GUATEMALA

FACULTAD DE INGENIERIA

Guatemala, 27 de mayo de 2019

Señor Director
Ing. Otto Fernando Andrino González
Escuela de Ingeniería Mecánica Eléctrica
Facultad de Ingeniería, USAC.

Señor Director:

Por este medio me permito dar aprobación al Trabajo de Graduación titulado **DISEÑO E IMPLEMENTACIÓN DE UN ROBOT MÓVL TIPO EXPLORADOR PARA FINES ACADÉMICOS,** desarrollado por el estudiante **Rodrigo Otoniel Méndez Canú**, ya que considero que cumple con los requisitos establecidos.

Sin otro particular, aprovecho la oportunidad para saludarlo.

Atentamente,

ID Y ENSEÑAD A TODOS

Ing. Julio César Solares Peñate
Coordinador de Electrónica

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
DIRECCION ESCUELA DE INGENIERIA MECANICA ELECTRICA
FACULTAD DE INGENIERIA

UNIVERSIDAD DE SAN CARLOS
DE GUATEMALA

FACULTAD DE INGENIERÍA

REF. EIME 32. 2019.

El Director de la Escuela de Ingeniería Mecánica Eléctrica, después de conocer el dictamen del Asesor, con el Visto bueno del Coordinador de Área, al trabajo de Graduación de el estudiante: RODRIGO OTONIEL MÉNDEZ CANÚ titulado: DISEÑO E IMPLEMENTACIÓN DE UN ROBOT MÓVIL TIPO EXPLORADOR PARA FINES ACADÉMICOS, procede a la autorización del mismo.

Ing. Otto Fernando Andrino González

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
DIRECCION ESCUELA DE INGENIERIA MECANICA ELECTRICA
FACULTAD DE INGENIERIA

GUATEMALA, 7 DE JUNIO 2019.

Universidad de San Carlos
De Guatemala

Facultad de Ingeniería
Decanato

Ref. DTG.400-2019

La Decana de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, luego de conocer la aprobación por parte del Director de la Escuela de Ingeniería Mecánica Eléctrica, al trabajo de graduación titulado: **DISEÑO E IMPLEMENTACIÓN DE UN ROBOT MÓVIL TIPO EXPLORADOR PARA FINES ACADÉMICOS,** presentado por el estudiante universitario: **Rodrigo Otoniel Méndez Canú,** y después de haber culminado las revisiones previas bajo la responsabilidad de las instancias correspondientes, se autoriza la impresión del mismo.

IMPRÍMASE.

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
DECANA
FACULTAD DE INGENIERÍA

Inga. Aurelia Anabela Cordova Estrada
Decana

Guatemala, Septiembre de 2019

/cc

ACTO QUE DEDICO A:

Dios	Por sus planes perfectos que me trajeron hasta aquí.
Mis padres	Andrés Méndez y Walda Canú, por su apoyo incondicional. Por la formación y el impulso para cumplir mis metas y por el ánimo en seguir hacia adelante.
Mi hermano	Pablo Méndez. Siempre presente y dispuesto a apoyar, por el ánimo alegre y la amistad.

AGRADECIMIENTOS A:

Mis tíos y primos	De la familia Gatica Canú. Por su apoyo siempre puntual y desinteresado.
Los amigos en la carrera	Son muchos para nombrarlos a cada uno. Gracias por darle a esta etapa alegrías, experiencias y emociones; y por su amistad y apoyo.
Los amigos de trivias y quinielas	Su experiencia ha sido de gran ayuda, así como su invaluable amistad.
Alberto Márquez	Por la amistad constante y la presión para terminar la tesis.
Andrea López	Por el ánimo y la presión para terminar la tesis.
Universidad de San Carlos de Guatemala	Por la oportunidad de acceder a una educación superior, y poder aportar al desarrollo de nuestro país.

ÍNDICE GENERAL

ÍNDICE DE ILUSTRACIONES

FIGURAS

TABLAS

LISTA DE SÍMBOLOS

Símbolo	Significado
cm	Centímetro
Gbps	Gigabits por segundo
kg/cm	Kilogramo por centímetro
km	Kilómetro
Mbps	Megabits por segundo
ms	Milisegundo
p	Píxel

GLOSARIO

Acorn Computers Compañía inglesa dedicada al diseño y construcción de computadoras en los años ochenta. Diseñaron el primer procesador RISC.

Host Un host en una red es una computadora o equipo conectado a la red.

Machine learning Una rama de la inteligencia artificial que estudia métodos para que las computadoras aprendan.

PWM Pulse Width Modulation, o modulación de ancho de pulso, es un tipo de señal de voltaje utilizada para enviar información o para modificar la cantidad de energía que se envía a una carga.

Relay Un interruptor mecánico controlado magnéticamente a través de una bobina que genera un campo magnético cuando una corriente pasa a través de ella.

slalon Competencia en la que la pista está dispuesta en forma de zigzag.

sprint En carreras, es un esfuerzo de aceleración.

Streaming	Retransmisión de datos en un orden secuencial, para ser accedido por partes sin necesidad de obtener la totalidad de un archivo para acceder a él.
Timeout	Cese de operaciones automático cuando un período de tiempo predefinido ha terminado.
WiFi	Es una tecnología basada en radio utilizada para crear redes de área local de forma inalámbrica utilizando los protocolos IEEE 802.11.
World Wide Web	Un sistema de información en internet que permite la conexión de documentos con otros documentos a través de hipervínculos.

RESUMEN

La Corporación Unificada Nacional de Educación Superior (CUN) invitó a la Universidad de San Carlos de Guatemala a participar a la competencia Mercury Robotics Challenge Latinoamérica en Bogotá, Colombia. El primer paso en el diseño del robot es conocer los requerimientos de la competencia, para realizar una planificación acertada.

Una vez se conocen todos los elementos de la competencia, se analizan algunas configuraciones de robots móviles, así como los elementos que los conforman y los tipos de actuadores y sensores que existen, cuáles se utilizaron y como se implementaron los elementos en la construcción del robot. Una de las partes más importantes de un robot es su controlador, que es como su cerebro. Tras un breve repaso de los procesadores y sus tecnologías, se selecciona el sistema de cómputo en base a la construcción del robot y las funciones que este debe realizar. Se analizan los requerimientos de software para el control remoto y la optimización necesaria para lograr un control en tiempo real.

Para poder unificar cada una de las partes en un sistema robótico coherente, se seleccionan los actuadores del robot y se analizan las necesidades de potencia de cada uno, seleccionando los componentes adecuados para manejarlos correctamente e interconectarlos con el sistema de cómputo a través de una placa de circuito impreso. Por último, se estudia la forma en que las instrucciones se transmiten a través de internet, cómo se transmite información de video de forma eficiente, el manejo adecuado las comunicaciones usando el modelo cliente-servidor y el proceso a seguir si ocurre una pérdida de conexión.

OBJETIVOS

General

Desarrollo e implementación de un controlador para un robot manejado a distancia, controlado vía internet, capaz de comunicarse con una computadora y realizar tareas específicas, para su uso en un concurso internacional de robótica.

Específicos

1. Diseñar e implementar de la placa de circuito impreso que controlará el robot.

2. Implementar algoritmos de control que sean rápidos para tener un control en tiempo real.

3. Diseñar e implementar un control para un brazo robótico de al menos dos grados de libertad.

4. Lograr una transmisión de video en tiempo real para tener un control adecuado en la ubicación remota.

INTRODUCCIÓN

La robótica es una de las ciencias con más auge en nuestros días. La podemos encontrar en todas partes, por ejemplo, en la industria, en líneas de producción, en facilidades médicas, en aplicaciones militares y de rescate, incluso en nuestras casas, como en impresoras 3D y dispositivos de limpieza.

Siendo un campo tan desarrollado, es importante que los ingenieros en formación tengan un contacto directo con la robótica y la implementación de sus conceptos teóricos, así como la comprensión de la relación entre las distintas ramas de la ingeniería, y como estas se complementan para lograr un producto final de alta calidad y de fácil uso.

En este proyecto se describen los criterios de diseño de las partes robóticas, así como el desarrollo y programación del controlador lógico del robot, y el diseño e implementación de un canal de comunicación a través de internet, con alta eficiencia, utilizando el protocolo TCP/IP, para transmisión de datos y de video en tiempo real.

Cabe destacar que el desarrollo de este proyecto es con fines académicos, basado en un robot que participó en el Mercury Robotics Challenge 2018 en Bogotá, Colombia, representando a la Universidad de San Carlos de Guatemala, pero los principios y aplicaciones pueden extenderse más allá de los usos académicos.

1. DESCRIPCIÓN GENERAL

La Corporación Unificada de Educación Superior (CUN) con sede en Bogotá, Colombia, realiza una competición anual de robótica a nivel Latinoamericano. Esta competencia consiste en construir un robot que pueda ser controlado de forma remota vía internet, que sea capaz de recorrer una pista con obstáculos, y que logre realizar una tarea específica que cambia cada año para la competencia.

1.1. Planteamiento del problema

La Universidad de San Carlos fue invitada a participar a la competencia Mercury Robotic Challenge Latinoamérica, por lo que se debe desarrollar un robot de tamaño reducido, con autonomía para funcionar por lo menos 15 minutos. Debe ser capaz de cargar con su propio peso y que sus elementos de potencia no se sobrecalienten, logrando realizar todas las acciones que se requieren para la realización exitosa del circuito.

Además, debe ser fácil de controlar remotamente, por lo que debe realizarse un *streaming* de video en tiempo real y de alta calidad. También, facilidad de conducción ya sea utilizando un control o algún otro medio para obtener las instrucciones, ejecutándolas a una velocidad aceptable para que el robot pueda ser controlado de forma adecuada.

Por último, debe ofrecer una gestión de conexión automática. Si existe algún inconveniente con la red, debe volver a realizar la conexión de manera automática, manejando de forma independiente las reconexiones y la

interrupción de cualquier instrucción en ejecución para evitar daños al robot y a sus alrededores, con el fin de evitar cualquier accidente.

1.2. Justificación

Para poder cumplir con el recorrido y el objetivo planteado por la competencia Mercury Robotic Challenge es necesario diseñar un robot de máximo 46 cms de ancho y 31 cms de altura, sin restricción en la longitud, una vez pueda maniobrar sin problemas en la pista. En el robot debe integrar los dispositivos de comunicación, control y potencia, así como la fuente de alimentación, brazo robótico y efectores finales.

Se debe crear entonces una solución viable para realizar el cableado de cada uno de los elementos que componen el robot, así como para las conexiones de potencia en cada uno de los motores. Esto se puede realizar en una placa de circuito impreso (PCB) que unifique todos los elementos necesarios.

En el apartado del software, se debe crear un programa que se encargue de las comunicaciones a través del protocolo TCP/IP de internet, que pueda gestionar las conexiones de forma manual y automática, que pueda detectar cuando ocurra una desconexión. También se debe generar un *streaming* de video en tiempo real. Por último, se debe diseñar e implementar la programación del robot, que permitirá realizar acciones y movimientos para que este cumpla con su misión.

La misión es recoger un objeto, transportarlo y colocarlo en una estructura diseñada para este fin.

Para esto, es necesario construir un brazo robótico de al menos dos grados de libertad, que cuente con la precisión suficiente para lograr colocar la carga en el espacio especificado.

1.3. Descripción de la competencia

El Desafío Mercury Latino Robot Challenge es una competición internacional, interuniversitaria, que consiste en el diseño e implementación de un robot capaz de realizar una variedad de tareas, mientras es controlado por un operador que se encuentra mínimo a 80 kms del lugar de la competencia. Cualquier comunicación entre el robot y el operador debe llevarse a cabo a través del canal de comunicaciones en el sitio.

Además, el operador solo puede recibir la información proporcionada por el robot, esto significa que cualquier fuente de información, tal como el video en vivo, que se origina a partir de una fuente que no sea el robot y no utiliza el canal de comunicaciones en el sitio no puede ser utilizado como referencia por el operador.

Cada robot comienza con un tiempo de configuración de cinco minutos seguido de diez minutos en pista, en los que el robot puede intentar un máximo de tres carreras; el robot debe seguir la trayectoria de la pista predefinida desde el “Inicio” hasta la “Meta” y llevar a cabo la recolección, transporte y entrega de una carga útil en el tiempo asignado, tratando de evitar diferentes obstáculos.

“El golpear o derribar los obstáculos generará sanciones. El robot no puede dejar nada en el campo, cualquier robot que pueda causar daños a personas o propiedades se considerará no apto para competir.

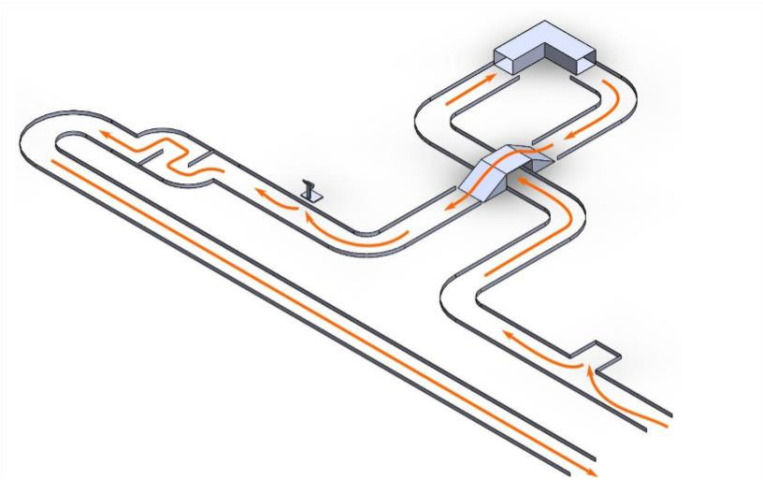
Se entiende que pueden producirse daños menores debido a que los robots golpean las paredes de la pista”[1].

A continuación, se detallan los elementos principales para el desarrollo de la competencia desarrollada en el año 2018.

- Pista

La pista se define como un camino de 61 cms de ancho, la cual se encuentra delimitado en ambos lados por paredes elevadas de 8 cms. A lo largo de la pista se encuentran los demás componentes de la competencia, que son la zona de carga y la zona de entrega, los obstáculos y el sprint.

Figura 1. **Vista general de la pista**



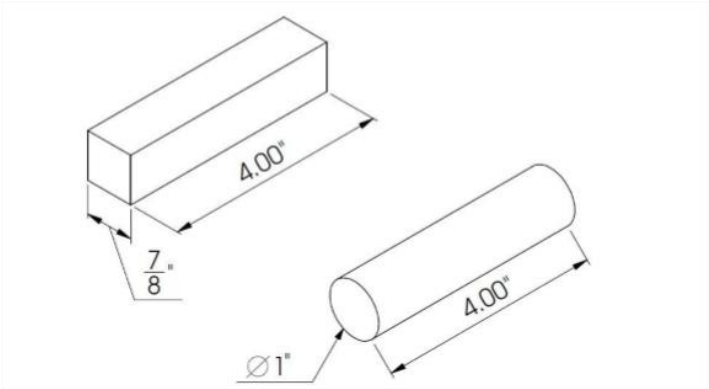
Fuente: CUN. *Reglamento competencia MERCURY 2018*. p. 5

[1] Corporación Unificada Nacional de Educación Superior. *Reglamento competencia Mercury 2018*. p. 9

- Área de carga

Esta zona se define como un cuadrado de 61 centímetros, es un extremo muerto de la pista, en el cual se colocará la carga para ser recogida por el robot. El equipo decide la ubicación y orientación de la carga, y no puede ser modificado al empezar la carrera. La figura 2 muestra los dos objetos disponibles para el transporte, se debe elegir solamente uno para usarlo en la competencia.

Figura 2. **Carga**

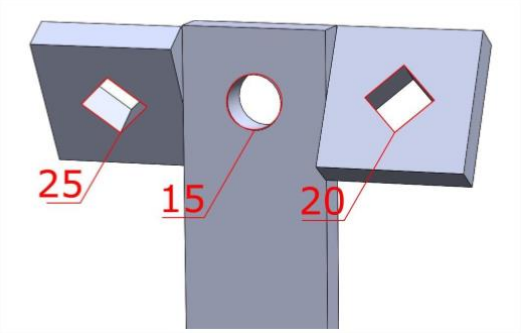


Fuente: CUN. *Reglamento competencia* Mercury *2018*. p.7

- Área de entrega

En la zona de entrega se encuentra un objeto independiente puesto fuera de la pista. La zona de entrega tiene tres orificios de dos tipos para acomodar cualquiera de las dos cargas, la cuadrada y la circular. Se entregarán puntos al equipo según el orificio en el que se entregue la carga, según se muestra en la figura 3.

Figura 3. **Zona de entrega y puntajes por agujero**



Fuente: CUN. *Reglamento competencia MERCURY 2018*. p.9

- Obstáculos

La pista tiene una variedad de obstáculos que retan la construcción del robot y la habilidad del conductor.

- Túnel: es una estructura de madera en forma de L de 30 centímetros de alto por 46 centímetros de ancho. Este limita las dimensiones del robot y es oscuro por dentro.
- Puente: es de 61 centímetros de ancho. Tiene una inclinación de 30 grados en la subida y la bajada, ambas con longitud de 30 centímetros. La parte plana del puente es de 61 centímetros de longitud.
- *Slalon*: un camino de zigzag, que pone a prueba la maniobrabilidad del robot.
- *Sprint*: La sección final de la pista es una carrera recta de 10 metros de largo, sincronizada con cables de disparo infrarrojos en ambos extremos. Esta parte del campo pondrá a prueba el control de velocidad y el control en línea recta de los robots.

- Tiempo

Cada equipo se le permitirá un máximo de 15 minutos de tiempo de operación durante la competencia. Los 15 minutos se dividen en dos secciones:

Los primeros 5 minutos son para la configuración y puesta en marcha del robot, que permite que el piloto obtenga el control remoto del vehículo. El tiempo de configuración termina cuando el robot comienza a funcionar. Los últimos 10 minutos son para recorrer la pista. Si el equipo utiliza más de 5 minutos para la configuración, se descontarán de los 10 minutos de tiempo de ejecución para recorrer la pista.

- Pérdida de conexión

El equipo debe pasar una prueba de “Pérdida de Señal” *(Loss Of Signal*) para poder ser elegido el campeón de la competencia. El equipo debe demostrar claramente que el conductor puede controlar el robot. El técnico oficial encargado del *router* lo apagará (desconexión red eléctrica), y el robot debe ser capaz de indicar claramente que está experimentando una situación de pérdida de señal y detenerse.

Después de que se reinicie el *router* oficial, el equipo debe ser capaz de demostrar que el conductor pudo restablecer la conexión nuevamente con el robot sin que ningún integrante del equipo manipule el robot. El robot debe mostrar que la conexión se restableció apagando el indicador de Pérdida de Señal.

1.4. Elementos del robot

Para poder participar en la competencia, se realizó la planificación del robot. Primero definiendo los elementos necesarios para poder cumplir todas las tareas requeridas en la competencia. También se definió la disposición de todos los elementos en la estructura para que el robot fuera eficiente en distintos aspectos. Esta sección refleja la planificación previa a la construcción del robot y, de manera general, los elementos que lo componen. En los capítulos posteriores se detallará como se implementó cada elemento.

1.4.1. Estructura

Para la estructura principal del robot, se eligió un vehículo de 4 ruedas, con forma rectangular. Se eligió esta disposición para poder tener suficiente espacio para colocar los elementos necesarios para que el robot cumpla su misión. Se eligieron ruedas en vez de orugas ya que los vehículos con ruedas se desplazan a mayor velocidad. Si se hace una buena selección de llantas, estas pueden tener suficiente fricción para desplazarse sobre cualquier superficie.

Se utilizó un motor para cada llanta. La elección de los motores es crucial para el desempeño adecuado, ya que se debe elegir un balance entre torque y velocidad. Es necesario tener en cuenta el peso combinado de todos los actuadores, elementos de control y de potencia que componen el robot. Pensando en eso, se utilizó una estructura metálica, que pudiera sostener el peso de todos los elementos incluyendo la batería. El problema que surge es que la estructura de metal puede funcionar como jaula de Faraday, por lo que la antena de wifi no puede estar dentro de la misma.

Para cubrir todo el cableado externo, y como seguro para transporte de la carga, se colocaron dos compartimientos que se desplazan horizontalmente de forma lineal para abrir y cerrar un espacio que permite al brazo de la garra, colocado al frente, moverse libremente cuando está abierto, y asegurar la carga cuando está cerrado.

1.4.2. Brazos robóticos

El robot se define como un robot móvil, pero para poder cumplir las tareas de la competencia se construyeron dos brazos robóticos. Uno para poder recoger la carga, sostenerla durante su transporte y depositarla en el área de entrega. Este brazo tiene dos grados de libertad, y tiene como efector final una pinza. Su colocación es cerca del suelo para que sea más fácil recoger la carga desde el suelo, y facilitar la alineación de la estructura del carro con la carga. Esto permite que el proceso sea rápido y se pueda automatizar para mejorar los tiempos de ejecución.

El otro brazo fue utilizado para la cámara. Este brazo necesitaba estar en la parte trasera, en una posición alta, para tener una referencia clara de la ubicación de la estructura del carro para tener mejor control al manejar. Cuenta también con dos grados de libertad, y su efector final es la cámara.

1.4.3. Placa de circuito impreso

Se utilizó una placa de circuito impreso para facilitar el cableado dentro del robot. En total, se utilizaron 6 servomotores y 5 motores DC, así como luces, puentes H, conexiones de comunicación y conexiones de alimentación. Para reducir los costos, se utilizó una tarjeta de desarrollo como controlador de

hardware, y se diseñó una PCB para centralizar las conexiones de las terminales que proveía cada elemento conectado.

Se implementaron en ella elementos de potencia y elementos de control, para los motores, las luces y los cambios de velocidad. También se centralizaron las conexiones provenientes de la batería, permitiendo que todos los elementos pudieran ser alimentados desde un mismo lugar.

1.4.4. Controlador

El controlador de hardware fue implementado en la tarjeta de desarrollo Launchpad Tiva C, diseñada y construida por Texas Instruments con el microcontrolador TM4C123GH6PM y programado en C. Consta de un proceso de selección de acciones que el robot puede ejecutar, buscando una alta velocidad de ejecución para poder estar al ritmo de la transmisión de instrucciones y tener un control en tiempo real. Para lograr esto, la programación se hizo en varios niveles y se realizó lo más sencilla posible.

Para mejorar el control del hardware, se utilizaron muchos periféricos integrados en el microcontrolador. Se eligió esta tarjeta de desarrollo debido a su flexibilidad y a su capacidad de utilizar los pines en varias formas según las necesidades de la construcción sin necesidad de rehacer la placa de forma completa.

1.4.5. Comunicación cliente servidor

Para manejar las comunicaciones se implementó un modelo cliente servidor. El cliente se definió como la computadora que el piloto del robot utiliza. A esta computadora se conectó un control de videojuegos. El cliente establece

una conexión con el servidor y lee los datos del control; los procesa para generar una cadena de caracteres que es transmitida al servidor. También realiza una solicitud de conexión hacia el *streaming* de video que genera el servidor.

El servidor es la computadora que se encuentra en el robot. Se seleccionó una Raspberry debido al bajo costo, al bajo consumo de energía y la conexión wifi que ofrece. Su función es manejar todas las conexiones de red a través de los puertos. En el servidor se genera un *streaming* de video que se puede acceder desde cualquier host en la red, y provee conexión al cliente por medio de un *socket* para la transmisión de los datos del control. Estos datos son decodificados por el servidor y transmitidos al controlador de hardware para su ejecución en tiempo real. El servidor fue programado en Python, y para la transmisión multimedia se usó Netcat.

2. ROBÓTICA

"La robótica es un campo relativamente joven de desarrollo tecnológico que sobrepasa los límites tradicionales de la ingeniería. Comprender la complejidad que envuelve un sistema robótico, así como sus aplicaciones, requiere conocimientos de ingeniería eléctrica y electrónica, ingeniería mecánica, ingeniería de sistemas e industrial, de ciencias de la computación, de economía y matemáticas"[2]. La naturaleza multidisciplinaria de la robótica ha permitido integrar conocimientos en diversas áreas para investigar y desarrollar sistemas mecánicos muy particulares, entre los que destacan los robots manipuladores. Estos sistemas están diseñados para una amplia variedad de aplicaciones, entre las que se encuentran actividades científicas, industriales, comerciales y domésticas[3].

En este capítulo se discute la fundamentación teórica de los elementos robóticos seleccionados en la construcción del robot usado en la competencia, así como las consideraciones de diseño y de implementación. También se incluye la clasificación de los robots por su construcción y su aplicación, haciendo énfasis en su aplicación como manipuladores industriales y como robots móviles.

2.1. Qué es un robot

Al hablar de robots, se puede pensar en un robot como una máquina humanoide, que tiene cierto grado de inteligencia y es capaz de realizar ciertas

[2] SPONG, Mark W., HUTCHINSON, Seth and VIDYASAGAR, M. *Robot Modeling and Control*. p. 1

[3] REYES, Fernando. *Robótica. Control de robots manipuladores*. p. 3

tareas en las que pueden sustituir a los humanos. El término robot se ha aplicado a una gran variedad de dispositivos mecánicos como *rovers* autónomos, drones, entre otros. Existen muchos usos y aplicaciones, formas y tamaños, tipos de construcción y diseños para lo que puede denominarse un robot. Actualmente, casi cualquier cosa que opera con cierto nivel de autonomía, usualmente bajo control de una computadora, ha sido llamado robot en algún momento. Esta diversidad implica que no es fácil determinar una definición exacta de lo que es un robot.

Sin embargo, la *Robotics Industries Association (RIA)* ofrece una definición oficial, que dice:

"Manipulador funcional reprogramable, capaz de mover material, piezas, herramientas o dispositivos especializados mediante movimientos variables programados, con el fin de realizar tareas diversas"[4].

En su artículo *¿Qué es un robot?*, Erico Guizzo nos ofrece otra definición, que no es ni muy general ni muy específica:

"Un robot es una máquina autónoma capaz de detectar su entorno, realizar cálculos para tomar decisiones y realizar acciones en el mundo real"[5].

Las definiciones anteriores, por sí mismas, no permiten distinguir con certeza en que se diferencia un robot de una máquina automatizada. La RIA en su definición resalta un elemento muy importante, y es que los robots son

[4] GONZÁLEZ, Victor, LÓPEZ CRUZADO, Antonio y CABERO, José. *Control y Robótica. ¿Qué es un robot?* http://platea.pntic.mec.es/vgonzale/cyr_0708/archivos/_15/Tema_5.1.htm. Consulta: diciembre de 2018

[5] GUIZZO, Erico. *Learn. Everything you need to get started in robotics*. https://robots.ieee.org/learn/. Consulta: diciembre de 2018

reprogramables. Esta característica les da a los robots una gran adaptabilidad y un alto grado de utilidad en diversas situaciones.

La programabilidad, que es la capacidad de modificar la tarea que realiza un robot al cambiarle el software de control, permite que se adapten rápidamente y de forma más económica a diferentes aplicaciones. Los robots también poseen versatilidad para llevar a cabo distintas tareas, incluso algunas para las que no fueron diseñadas. Por eso se dice que tienen gran adaptabilidad. Esta adaptabilidad sostiene la segunda definición, dado que los robots pueden detectar su entorno con sensores y adaptarse para funcionar de manera óptima según las condiciones de su alrededor.

2.2. Clasificaciones de los robots

De la misma forma que es difícil dar una definición clara de qué es un robot, la forma de clasificación es también muy grande y diversa. Hoy en día existe una gran variedad de robots, que varían en estructura, geometría, función y aplicación. La siguiente tabla muestra una clasificación general de los robots.

Tabla I. **Tipos de robots**

Clasificación de los robots		
Móviles	Terrestres: ruedas, patas	
	Submarinos, aéreo-espaciales	
Humanoides	Diseño complejo	
Industriales	Brazos mecánicos	Robots manipuladores

Fuente: REYES, Fernando. *Robótica. Control de robots manipuladores.* p.9

Los robots móviles son todos aquellos que no están fijos en algún lugar y pueden moverse por sí solos. Entre ellos se encuentran los *rovers*, los drones,

y robots de exploración submarina. Los robots humanoides son los que imitan la forma del cuerpo humano, y muchas veces buscar realizar tareas realizadas por humanos. Los robots industriales son probablemente los más conocidos, y por lo general son brazos robóticos que realizan tareas determinadas, aunque pueden ser reprogramados para cambiar su uso o mejorar su rendimiento.

En el caso de los robots manipuladores, estos pueden ser clasificados según diversos criterios. Entre ellos encontramos su fuente de energía, su geometría, su área de aplicación o su método de control. La principal razón de esta clasificación es poder determinar qué tipo de robot es el más adecuado para realizar una tarea específica.

- Fuente de energía: los robots pueden ser eléctricos, hidráulicos o neumáticos. Los robots eléctricos pueden usar motores con mucha precisión, y son bastante silenciosos en su operación, por lo que están cobrando popularidad. Los robots hidráulicos son de operación rápida y producen mucho torque, pero el mantenimiento es complicado. Los robots neumáticos tienen la desventaja de que no se pueden controlar con precisión, por lo que sus aplicaciones son limitadas.
- Área de aplicación: usualmente se clasifican en dos áreas, de ensamblaje y de propósito general. Los de ensamblaje son robots pequeños y de mucha precisión, mientras que los de propósito general pueden tener cualquier tamaño y su precisión es limitada, y sus usos son pintura, soldadura, y carga y descarga de materiales.
- Método de control: los robots se clasifican también por su lazo de control. Pueden ser de lazo abierto. A estos robots se les programan una serie de puntos en el espacio que fijan su trayectoria. Los robots de lazo cerrado utilizan control por computadora para determinar sus movimientos y se vuelven realmente multifuncionales y reprogramables.

- Geometría: Los robots pueden se clasificados como articulados (RRR), esféricos (RRP), SCARA, entre otros. Esta clasificación se hace en base a las primeras tres articulaciones del manipulador.[6]

Para el desarrollo de este trabajo, se hará una clasificación de los robots en robots móviles y estáticos. Los robots móviles se clasifican como los robots que pueden moverse por sí mismos, y los estáticos como aquellos que están colocados en un lugar fijo, y solo alcanzan posiciones hasta donde su construcción lo permiten.

2.3. Cinemática

La cinemática es una de las áreas de desarrollo más importantes de la robótica. Mark Spong define (el problema de) la cinemática como “describir el movimiento del manipulador sin considerar las fuerzas y torques que lo producen”[7]. Esto implica que la descripción cinemática de un movimiento sea geométrica. La cinemática directa busca determinar la posición y dirección del efector final, dados los valores de las articulaciones variables del robot. La cinemática inversa busca los valores de las articulaciones del robot al conocer la orientación y la dirección del manipulador final.

Aunque el análisis geométrico de un robot, así como el desarrollo de las matrices que definen su movimiento son herramientas muy útiles en el desarrollo de un robot, en este trabajo no se profundiza en esta área. La principal razón es que el robot está diseñado para ser operado manualmente de forma remota. Como no es un robot automatizado, el controlador del robot no debe conocer la

[6] SPONG, Mark W., HUTCHINSON, Seth and VIDYASAGAR, M. *Robot Modeling and Control*. p. 6

[7] Ibíd. p.65

posición y orientación del robot para realizar los movimientos, sino que todo se realiza a consideración del operador del robot, que a través de una cámara puede conocer estos datos y tomar decisiones en base a ello.

Otra razón es que el diseño de los brazos en el robot posee pocas articulaciones. Debido a eso, el operario del robot puede controlar cada una de las articulaciones manualmente. Está fuera del alcance de este proyecto proponer una solución de un brazo más complejo, con distintas funciones, que utilice los conceptos de cinemática directa e inversa para obtener resultados diferentes.

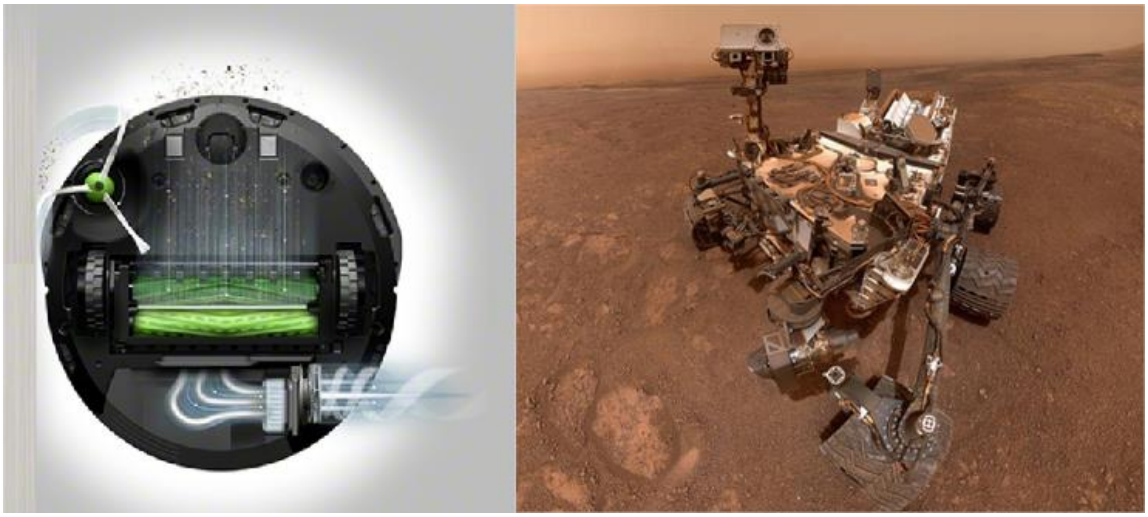
2.4. Robots móviles

Los robots móviles son los robots que no están confinados a un espacio de trabajo limitado, como en el caso de un brazo robótico instalado en una fábrica. Su principal característica es que poseen un espacio de trabajo prácticamente ilimitado, ya que pueden cambiar su posición en un momento dado. Estos robots cuentan con un sistema locomotor que les permite desplazarse en su entorno. Su capacidad de movilidad y las distintas configuraciones de construcción permite que sean muy adaptables a diferentes tipos de terrenos y usos. Los robots móviles pueden ser autónomos o controlados remotamente. Como parte del reglamento de la competencia el robot construido debe ser operado de forma remota.

Los robots móviles pueden ser construidos para su uso en agua, tierra y aire. Sus aplicaciones son muy variadas, entre las que se encuentran robots de investigación, de seguridad, y de entretenimiento. Algunos ejemplos de robots móviles son los drones, o vehículos aéreos no tripulados, con aplicaciones en investigación científica, fotografía y video. Sin embargo, por su facilidad de

construcción y gran cantidad de usos, los robots con ruedas son de los más extendidos en la actualidad. Sus aplicaciones incluyen desde robots de limpieza como la Roomba, comercializada por iRobots, hasta las exploraciones del espacio, como el *rover* Curiosity, que explora la superficie de Marte.

Figura 4. **Aplicaciones de los robots con llantas. La Roomba (izquierda) y el Curiosity(derecha)**



Fuente: iRobot shop Roomba e Series. https://bit.ly/2yjUSi1; NASA Mars Science Laboratory – Curiosity Rover. https://go.nasa.gov/2GqeZiU. Consulta: diciembre de 2018

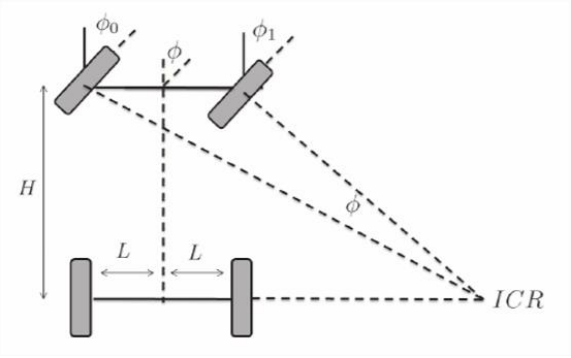
Los robots terrestres se pueden clasificar según el tipo de sistema de locomoción que poseen. Entre estos se incluyen las orugas, las patas y las ruedas. En este documento se presentará únicamente el diseño con llantas, ya que, al analizar distintas opciones entre orugas y llantas, y tomando en cuenta que la competencia es una carrera con tiempo límite, se optó por un sistema de cuatro llantas convencionales. Las llantas proveen suficiente fricción con la superficie, estabilidad, y permiten un desplazamiento a mayor velocidad que las orugas.

2.4.1. Consideraciones de diseño para un robot con llantas

Los robots móviles pueden tener distintos tipos de llantas, y de esto depende cómo se mueven. La respuesta a las instrucciones puede variar dependiendo a ciertas condiciones en las llantas. Cada una de las llantas contribuye al movimiento del robot y le impone restricciones. El chasis del robot provee la base para la unión de todas las llantas. El robot diseñado para este proyecto cuenta con cuatro ruedas convencionales y fijas, que no cuentan con partes móviles en su unión con el chasis del robot.

Cuando un robot móvil se construye con llantas, existen diversas configuraciones que permiten distintos grados de control sobre el movimiento, estabilidad y tracción. Algunas de estas configuraciones son la configuración *Ackerman*, enfocada en estabilidad, que utiliza dos ruedas con movimiento para la dirección y dos con movimiento hacia adelante y hacia atrás para la trayectoria. Existen otras que utilizan tres llantas como la configuración triciclo o la configuración síncrona[8].

Figura 5. **Robot en configuración *Ackerman***

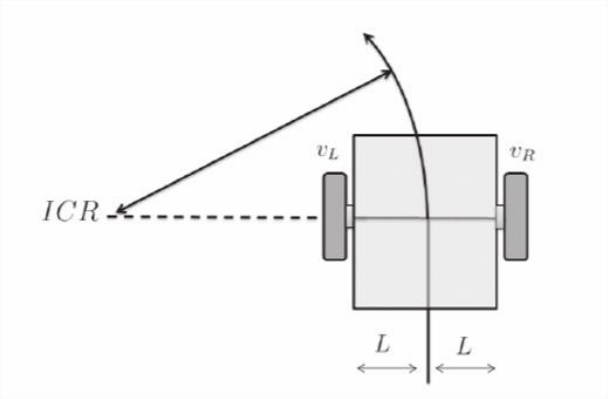


Fuente: REYES, Fernando. *Robótica. Control de robots manipuladores.* p. 520

[8] REYES, Fernando. *Robótica. control de robots manipuladores*. p.519-523

Tras el análisis de varias configuraciones, se utilizó una configuración de cuatro ruedas similar a la configuración *Ackerman*, para asegurar la estabilidad sin importar la distribución de peso en el interior del robot generada por los componentes electrónicos y de potencia. Aunque la distribución de las llantas es como en la configuración *Ackerman*, se utilizó el direccionamiento diferencial para dirigir al robot. El Centro Instantáneo de Rotación, o IRC por sus siglas en inglés, es el punto sobre el cual el robot gira. Este punto está sobre una línea perpendicular que atraviesa el centro de las dos ruedas.

Figura 6. **Robot móvil en configuración de direccionamiento diferencial**



Fuente: REYES, Fernando. *Robótica. Control de robots manipuladores.* p.522

Utilizando el direccionamiento diferencial, el robot puede moverse en línea recta, puede realizar curvas de distinta intensidad y girar sobre sí mismo. El esquema del direccionamiento diferencial se basa en un chasis con dos llantas. Cada una de las ruedas es controlada por un motor. Cuando ambas llantas se desplazan a la misma velocidad, se tiene una estabilidad en la posición perpendicular de las llantas, y el robot se mueve hacia adelante. Al generar una diferencia de velocidad entre las llantas, una se moverá más rápido que la otra,

provocando un giro. Al variar la diferencia en la velocidad de cada lado, se pueden generar cambios de dirección hacia donde se desea.

El diseño utilizado para el robot aplica el direccionamiento diferencial en un diseño con cuatro ruedas, moviendo ambas llantas en cada lado para gobernar la dirección del robot. Para mover adecuadamente el peso de la batería, los motores y los demás componentes electrónicos, se utilizó un motor para cada llanta. Para que el robot se mueva hacia delante, sin desviarse, es necesario que todos los motores se muevan a la misma velocidad. Sin embargo, no todos los motores se mueven igual.

Hay dos formas de solucionar este problema. Una es monitoreando constantemente la velocidad de cada motor y compensar electrónicamente la velocidad para una conducción estable. La otra forma es probar que los motores sean lo más parecidos posibles, y colocarlos en la posición en la que impacten menos la conducción. Debido a que no es un robot autónomo sino operado manualmente de forma remota, se optó por la segunda forma, ya que, al seleccionar la posición óptima de cada motor, se logra una estabilidad adecuada y el conductor del robot puede reajustar la trayectoria en tiempo real en el momento que sea necesario, simplificando el desarrollo del software del controlador.

2.5. Robots estacionarios

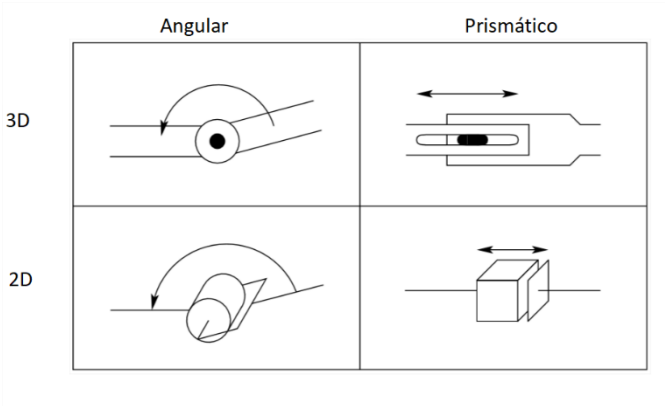
Los robots estacionarios no pueden desplazarse por sí mismos a otra posición. Están restringidos a realizar tareas dentro de un espacio de trabajo limitado a sus dimensiones físicas. Estos robots son conocidos como robots manipuladores o robots industriales, ya que su mayor aplicación se encuentra en la industria.

Aunque el espacio de trabajo de estos robots es limitado, sus aplicaciones son innumerables, entre las que se encuentran trabajos de soldadura de precisión, inspección y control de calidad, movimiento de piezas pesadas en líneas de producción, entre otras. Una ventaja muy importante que presentan estos robots es que pueden combinarse con algún sistema móvil para lograr robots mucho más útiles y que son adaptables a muchísimas situaciones. Los robots manipuladores se componen de tres elementos básicos:

- Articulaciones: un brazo robótico está formado por eslabones y articulaciones. Las articulaciones son los elementos que permiten el movimiento entre dos eslabones continuos en el robot. Según su movimiento, se clasifican en angulares y prismáticas.
- Actuadores: Los actuadores proveen las señales para realizar los movimientos en el robot. Pueden ser servomotores, sistemas hidráulicos o neumáticos.
- Sensores: los sensores no son exclusivos a los robots manipuladores o estáticos, pero son muy utilizados en esta área para poder determinar la velocidad y posición del robot en un momento determinado, y realizar cambios según sea necesario de forma autónoma o asistida, según la aplicación del robot[9].

[9] REYES, Fernando. *Robótica. control de robots manipuladores*. p.14

Figura 7. **Representación simbólica de las articulaciones**



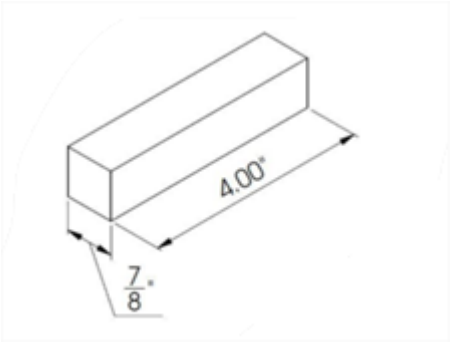
Fuente: SPONG, Mark W. *Robot Modeling and Control*. p.4

Aparte de los tres elementos mencionados anteriormente, es importante mencionar los efectores finales. Los efectores finales son las herramientas que se colocan al final de un robot manipulador para que realicen las tareas para las que fueron diseñadas. Algunos ejemplos de efectores finales son garras, puntas de soldadura, herramientas de pintura y herramientas de desgaste. Los efectores finales varían según las necesidades de la aplicación. En los últimos años, la investigación y el desarrollo de los efectores finales se ha centrado en las manos antropomórficas, para su uso en la industria de manufactura y en el área de la salud.

2.6. Mecanismos y sensores

El reglamento de la competencia Mercury Robotic Challenge explica que el objetivo de la carrera es recoger, transportar y entregar uno de los dos objetos provistos por la organización de la competencia. Uno es un objeto cuadrado de 2,23 cms * 10 cms y otro es cilíndrico, de 2,54 cms * 10 cms. Ambos están representados en la figura 2.

Figura 8. **Objeto seleccionado para la competencia**



Fuente: CUN. Reglamento competencia Mercury 2018. p.7

Se seleccionó el objeto cuadrado para el uso en la competencia, y en base a esto se decidieron los mecanismos para recogerlo, transportarlo y entregarlo de una forma segura y rápida. Se optó por construir un brazo robótico de dos grados de libertad, que será explicado más adelante. Como efector final se escogió una pinza de dos dedos, con bases planas, para poder recoger la pieza y mantenerla segura. La pinza es actuada por medio de un tornillo sin fin conectado a un motor DC, que actúa como un tornillo de potencia, transformando el movimiento giratorio en un movimiento lineal que permite abrir y cerrar las pinzas. Las pinzas son construidas por Makeblock, y se denomina Makeblock Gripper.

Figura 9. **Makeblock Gripper**

Fuente: Robotshop, MakeBlock Robot Gripper. https://www.robotshop.com/media/catalog/product/cache/image/900x900/9df78eab33525d08d6e5fb8d27136e95/m/a/makeblock-robot-gripper-1.jpg. Consulta: marzo 2019

Siguiendo también las instrucciones de la competencia, que indican que los robots deben ser operados por una persona a una distancia no menor de 80 kms, y que deben tener un canal de comunicación bidireccional capaz de transmitir información. Para obtener información relevante que permita al conductor la toma de decisiones en la competencia se eligió como sensor una cámara con un ángulo de visión de 120 grados. Esto permite al conductor obtener información bastante precisa de la orientación y posición del robot respecto a los límites de las pistas, pudiendo realizar ajustes en la trayectoria del robot para recoger la carga, entregarla y esquivar los obstáculos en el trayecto.

La cámara funciona como efector final de otro brazo robótico de dos grados de libertad, que provee al piloto la posibilidad de ampliar el ángulo de visión para poder guiar el robot de una forma más adecuada. Para proveer un efecto de profundidad y una aproximación de la distancia a la que se encuentra el robot de

objetos frente a él se le colocó un led láser que apunta a una distancia fija frente al robot.

Para la construcción de los brazos robóticos se decidió utilizar servomotores de 180 grados de rotación, para obtener un control preciso y sencillo, fácil de implementar en el controlador del robot, y que no proporcione fallas que provoquen sanciones en el desarrollo de la competencia.

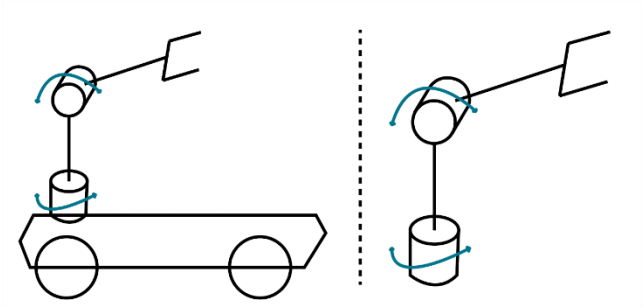
2.7. Diseño de los brazos robóticos

Los brazos robóticos implementados en el robot son implementados con servomotores, y cada uno cuenta con dos grados de libertad. Sin embargo, la aplicación que tienen es diferente y cada uno tiene una construcción distinta. El rango de movimiento de cada uno es limitado, pero combinado con la capacidad de movilidad de la parte móvil del robot, su área de trabajo se extiende notablemente.

2.7.1. Brazo de la cámara

El brazo de la cámara está diseñado para dar al conductor del robot un campo de visión extendido del terreno donde está. Su función es proveer una imagen clara del lugar donde está ubicado el robot. Ofrece un campo de visión de más de 180 grados horizontalmente, y aproximadamente 90 grados verticalmente. La idea de este brazo robótico es poder dar movilidad adicional a la cámara, para que el robot completo pueda estar en una posición estática, y se pueda realizar un reconocimiento del terreno con bastante precisión. El alcance de la cámara es ampliado por el movimiento del robot.

Figura 10. **Modelo del brazo de carga y colocación en el carro**

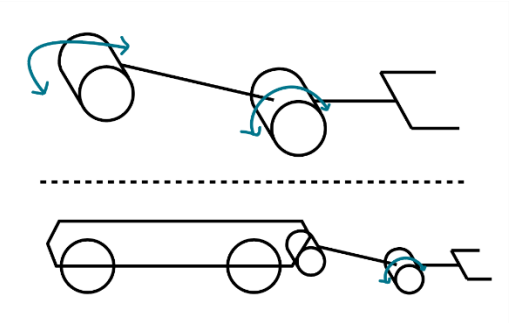


Fuente: elaboración propia

2.7.2. Brazo para la carga

El brazo para la carga está diseñado con la capacidad de recoger y depositar la carga, y está diseñado para poder descansar en el cuerpo del robot. Al descansar sobre el cuerpo del robot el tamaño total del robot es más pequeño, permitiendo mejor maniobrabilidad, y mayor estabilidad al transportar la carga. Es un brazo de dos grados de libertad, que solamente provee movimiento en una dirección, para poder extender el brazo, dándole mayor longitud para poder recoger y depositar la carga. El movimiento en las demás direcciones es provisto por la parte móvil del robot, al cambiar su dirección completa. La dirección diferencial con la que cuenta el robot es bastante precisa para poder realizar ajustes pequeños de dirección, facilitando orientar el brazo hacia la carga que debe transportar y el lugar de depósito.

Figura 11. **Modelo del brazo de carga y colocación en el carro**



Fuente: elaboración propia

El efector final, como se explicó anteriormente, es una pinza manipulada con un motor DC, con dedos planos, para poder tomar el objeto cuadrado. El brazo fue calibrado cuidadosamente para poder extenderse en su totalidad sin tocar el suelo, y poder regresar a descansar al cuerpo del robot sin afectar ninguno de los componentes que están sobre él.

2.7.3. Caja de resguardo de la carga

En el robot se implementó un mecanismo de seguridad para el transporte de la carga. Este consiste en dos piezas de plástico que por medio de un movimiento prismático se abren y cierran sobre el cuerpo del robot, encerrando entre ellas el efector final del brazo de la carga. Esto permite asegurar que la carga que se está transportando no caerá del robot, evitando la pérdida de puntos. Otra de las ventajas que ofrece es que el piloto puede conducir el robot por cualquier tipo de terreno sin temor a perder la carga, lo que produce mayor seguridad en la conducción, mejorando los tiempos de recorrido de la pista.

El robot construido y en funcionamiento puede ser visto en los siguientes enlaces:

- https://www.youtube.com/watch?v=GtgPC0SubtY
- https://www.youtube.com/watch?v=6v-xZL8wSts

Figura 12. **SentinelGT**

Fuente: elaboración propia

3. COMPUTADORAS Y CONTROLADOR DEL ROBOT

Actualmente, los servicios digitales que utilizamos son, casi en su totalidad, controlados por computadora. Las computadoras impactan en todas las áreas de nuestra vida. Se utilizan para manejar datos, hacer cálculos, en la industria y se encuentran también en dispositivos de entretenimiento personal como reproductores de música, consolas de videojuegos, reproductores de CD, DVD y Blu-ray.

En la robótica, las computadoras leen las entradas de los sensores, procesan los datos y toman decisiones de acuerdo con la información que reciben. Son el cerebro que da a los robots su capacidad de tomar decisiones y actuar de una forma u otra, según las necesidades del momento.

Para la competencia Mercury Robotics Challenge una computadora se encargaba de las comunicaciones y otra del control de hardware. Al unir las capacidades específicas de cada una se logró un robot completo y funcional, de acuerdo con las necesidades de la competencia.

- Computadora

Una computadora es “una máquina electrónica que, mediante determinados programas, puede procesar información y resolver problemas”[10]. La arquitectura básica de una computadora se compone de cinco elementos: la unidad de entrada, las unidades aritméticas y de control, la unidad de memoria, y la unidad

[10] Real Academia Española. *Diccionario de la Lengua Española*. https://dle.rae.es/?w=computadora. Consulta: enero de 2019

de salida. La computadora puede tomar datos en su unidad de entrada, procesarlos, realizar operaciones con ellos y generar información y datos útiles, que son enviados a la unidad de salida para presentarlos de una forma conveniente al usuario.

Las computadoras no son máquinas inteligentes. Pueden almacenar mucha información, pero solo realizan las instrucciones que han sido programadas en su memoria. Las partes físicas de una computadora se conocen como *hardware*, e incluye los circuitos electrónicos, cables, entre otros. Para que esta parte sea útil, la memoria de programa debe decir a la unidad central de procesamiento (CPU) qué tiene que hacer. La lista de instrucciones que la CPU debe ejecutar constituye un programa. Estos programas son conocidos generalmente como *software*.

Específicamente, las computadoras se componen de un procesador, memoria de acceso volátil (RAM), memoria de solo lectura (ROM) y puertos de entrada y salida (E/S)[11]. El hecho de que las computadoras sean programables les permite realizar una amplia variedad de tareas, convirtiéndose en dispositivos de uso general.

3.1. Procesador

Un procesador es un circuito integrado digital de propósito general, controlado por medio de un reloj, basado en registros, que recibe datos binarios como entrada, realiza operaciones según las instrucciones guardadas en su memoria, y coloca los resultados en sus salidas. Debido a la mejora de la tecnología de la fabricación de circuitos integrados, se ha reducido el tamaño de

[11] VALVANO, Jonathan. *Embedded Systems: Introduction to ARM Cortex M Microcontrollers*. p. 52

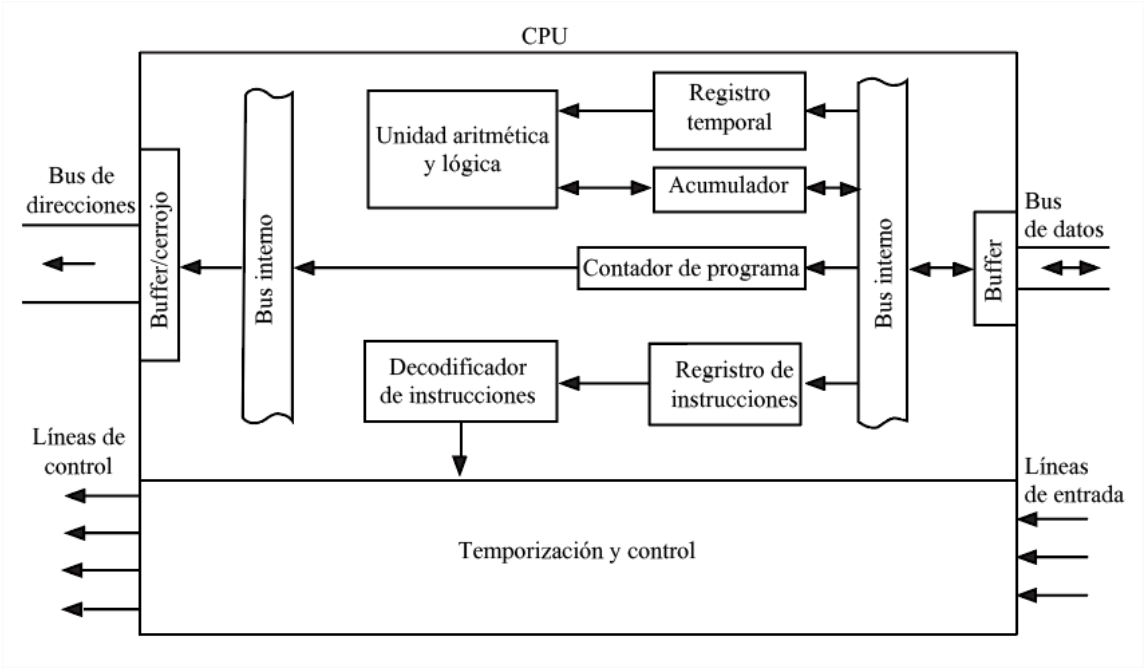
estos, permitiendo la construcción de elementos muy pequeños de mucha capacidad. A partir de esto surge el término microprocesador. Un microprocesador es un pequeño procesador, donde pequeño se refiere a tamaño físico y no a la capacidad de cómputo[12]. Un microprocesador puede definirse entonces como una pastilla de muy alta escala de integración (VLSI), que realiza las tareas de la unidad central de tratamiento de una microcomputadora u otro sistema de control automático.

Los microprocesadores están estrechamente relacionados con las computadoras. El microprocesador, llamado también la unidad central de procesamiento (C*entral Processing Unit*, CPU), es el elemento principal de control en las computadoras. En general, los microprocesadores poseen elementos de memoria llamados registros, y la parte de proceso de instrucciones llamada Unidad Aritmética Lógica (*Arithmetic Logic Unit,* ALU). La ALU contiene circuitería de decodificación de instrucciones, una sección de temporización y control, además de las conexiones de entrada y salida necesarias. La mayoría de CPUs contienen además una serie de registros especiales para usos específicos[13].

[12] VALVANO, Jonathan. *Embedded Systems: Introduction to ARM Cortex M Microcontrollers*. p. 52

[13] TOKHEIM, Roger L. *Fundamentos de los microprocesadores*. p. 85

Figura 13. **Arquitectura simplificada de una CPU**



Fuente: TOKHEIM, Roger. *Fundamentos de los microprocesadores*. p. 85

El microprocesador controla la memoria y las conexiones de entrada/salida (E/S) a través de una serie de conexiones llamadas buses. Los buses seleccionan una conexión E/S o un dispositivo de memoria, intercambian datos entre la conexión E/S o la memoria y el microprocesador, y controlan las entradas y salidas, así como la memoria.

El microprocesador realiza tres tareas principales para un sistema computarizado: transfiere datos entre sí mismo y los sistemas E/S, realiza operaciones aritméticas y lógicas simples y dirige el flujo del programa a través

de decisiones simples. Estas tareas pueden parecer sencillas, pero a través de ellas el procesador realiza cualquier tipo de operaciones[14].

El verdadero poder de los procesadores está en su capacidad de ejecutar millones de instrucciones cada segundo de los programas, o el software, guardadas en los sistemas de memoria. El software es una serie ordenada de instrucciones específicas que son guardadas en la memoria, y que definen exactamente cómo y cuándo se deben ejecutar ciertas tareas. El procesador ejecuta el software obteniendo e interpretando las instrucciones una por una.

3.2. Tipos de instrucciones

A lo largo de la historia, se han diseñado y construido distintos tipos de procesadores. Cada uno con un set de instrucciones, para poder realizar las instrucciones programadas en ellos. El grupo de instrucciones que un procesador puede ejecutar se denomina repertorio de instrucciones. Los repertorios de instrucciones, o *Instruction Set Architecture (ISA)*, como se conoce en inglés, definen todos los componentes y las operaciones de la computadora que son visibles o accesibles al programador o compilador. Esto significa que el ISA provee toda la información necesaria para alguien que desea escribir un programa en lenguaje de máquina o traducirlo de algún lenguaje de alto nivel, como C++ o Java.

Los repertorios de instrucciones no están estandarizados, debido al individualismo de cada fabricante y al uso de los microprocesadores. La arquitectura y construcción de un procesador son dependientes del ISA, ya que se puede incluir hardware que facilite la ejecución de instrucciones complejas que

[14] BREY, Barry B. *Intel Microprocessors. Architecture, Programming and Interfacing*. p. 25

se realizan continuamente[15]. Precisamente de este punto se generaron dos clasificaciones generales de los procesadores: la CISC o *Complex Instruction Set Computing* (instrucciones complejas), y la RISC o *Reduced Instruction Set Computing* (instrucciones reducidas). En la actualidad existe un amplio rango de arquitecturas, que pueden ser clasificadas entre totalmente CISC o totalmente RISC. En los años 90, existía una gran discusión sobre cual arquitectura era mejor. En la actualidad, aunque la mayoría de los procesadores en el mercado se promocionan como CISC, internamente funcionan como RISC, y el software que se crea para casi todo tipo de procesadores genera instrucciones CISC[16].

3.2.1. CISC

Una arquitectura CISC (*Complex Instruction-set Computing*) tiene como objetivo completar una tarea en la menor cantidad de código de ensamblador posible[17]. Para esto, es necesario construir un procesador capaz de interpretar y ejecutar una serie particular de instrucciones. Si una instrucción es muy utilizada en un programa, esta se implementa en hardware para que su ejecución sea más eficiente[18]. Por ejemplo, si una aplicación recupera datos de una dirección de memoria, añade uno a la dirección de memoria y luego repite, se puede implementar la lógica que realice ambas acciones en una sola instrucción.

Esto resulta en un hardware más complejo, que ejecuta un código más corto y, por tanto, más rápido. En un modelo CISC, se pueden implementar instrucciones como crear un JPG, o comprimir *strings* en memoria, o comprimir

[15] TOKHEIM, Roger L. *Fundamentos de los microprocesadores*. p. 79

[16] GLASKOWSKY, Peter. *AMD's SSE5 ends the old RISC vs. CISC debate*. CNET News. https://www.cnet.com/news/amds-sse5-ends-the-old-risc-vs-ciscdebate/. Consulta: enero de 2019

[17] CHEN, Crystal, NOVICK, Greg and SHIMANO, Kirk. *risc vs. cisc. Risc Architecture* https://cs.stanford.edu/people/eroberts/courses/soco/projects/risc/risccisc/index.html. Consulta: enero de 2019

[18] Ibíd a 16

un mp3. Una de las principales ventajas de esta arquitectura es que los compiladores deben realizar poco trabajo para traducir de un lenguaje de alto nivel a lenguaje ensamblador. Como la longitud del código es relativamente corta, se necesita poco espacio en la memoria RAM para almacenar las instrucciones. El énfasis es construir instrucciones complejas directamente en el hardware.

Las principales características que definen las instrucciones de la arquitectura CISC son:

- Muchas instrucciones.
- Instrucciones con longitud variable.
- Distintas instrucciones pueden acceder a la memoria.
- En una sola instrucción, el procesador puede leer y escribir a la memoria.
- El procesador posee menos registros, pero son más especializados.
- Muchos tipos diferentes de modos de direccionamiento[19].

3.2.2. RISC

RISC o *Reduced Instruction Set Computing* es un tipo de arquitectura de procesador que utiliza pequeñas instrucciones altamente optimizadas para realizar las operaciones en lugar de instrucciones especializadas, que pueden encontrarse en otro tipo de arquitecturas, como CISC[20]. En las arquitecturas RISC, la complejidad reside en el código ensamblador que escribe el programador o que genera el compilador. Esto le da una ventaja clara sobre CISC, ya que requiere una arquitectura de hardware más sencilla, pues necesita

[19] VALVANO, Jonathan. *Embedded Systems: Introduction to ARM Cortex M Microcontrollers.* p. 166

[20] CHEN, Crystal, NOVICK, Greg and SHIMANO, Kirk. *risc vs. cisc. Risc Architecture* https://cs.stanford.edu/people/eroberts/courses/soco/projects/risc/risccisc/index.html. Consulta: enero de 2019

interpretar menos instrucciones, y se pueden diseñar microprocesadores de bajo consumo de energía. En aplicaciones de dispositivos móviles que funcionan con baterías, como celulares, tablets, drones, la alta capacidad de computación con bajo consumo de energía es indispensable.

La arquitectura RISC está basada en la velocidad de ejecución de las instrucciones. Por esto, una de las características que definen esta arquitectura es que se ejecuta una instrucción por ciclo de reloj (c*lock per instruction*, CPI). Esto se debe a una optimización de las instrucciones en el microprocesador, y a una técnica conocida como *pipelining*. El *pipelining* es una técnica que permite la ejecución simultánea de partes, etapas o instrucciones para procesar las instrucciones más eficientemente. Por último, los procesadores RISC cuentan con un gran número de registros de propósito general idénticos. Estos previenen grandes cantidades de interacciones con la memoria, acelerando la ejecución de las instrucciones[21].

Otras características que permiten definir una arquitectura como RISC son:

- Un repertorio de instrucciones limitado.
- Instrucciones de longitud fija (32 bits generalmente).
- Solo algunas instrucciones pueden acceder a la memoria.
- Ninguna instrucción puede leer y escribir a la memoria en la misma instrucción.
- Modos de direccionamiento limitados[22].

[21] CHEN, Crystal, NOVICK, Greg and SHIMANO, Kirk. *risc vs. cisc. Risc Architecture*. https://cs.stanford.edu/people/eroberts/courses/soco/projects/risc/ri sccisc/index.html. Consulta: enero de 2019.

[22] VALVANO, Jonathan. *Embedded Systems: Introduction to ARM Cortex M Microcontrollers*. p. 166

3.3. Sistemas embebidos

Usualmente se concibe un sistema de cómputo como una computadora de escritorio, una computadora portátil, o tal vez un servidor. Sin embargo, existen millones de dispositivos que se construyen con fines específicos, que también son sistemas de cómputo. Estos dispositivos se encuentran dentro de otros dispositivos, realizando tareas repetitivas pero muy importantes, y que muchas veces no se pueden reconocer. Estos son los que se conocen como sistemas embebidos.

Mohit Arora define un sistema embebido como “la combinación de hardware de computadora con un software que está diseñado para un propósito específico”[23]. Para entenderlo mejor, se debe considerar lo que es un sistema. “Un sistema es un grupo de componentes que trabajan en conjunto para lograr un fin específico de una forma coordinada”[24]. Estos componentes pueden ser partes mecánicas, elementos químicos, componentes electrónicos sensores y actuadores. La estructura de un sistema se define por sus componentes y sus interconexiones. En el caso de un sistema embebido, el comportamiento se refleja en la respuesta en las salidas a los cambios en las entradas.

Según el diccionario de la Real Academia Española, embebido significa que “existe una cosa que está contenida dentro de otra[25]”. En el día a día podemos encontrar miles de dispositivos que tienen sistemas embebidos dentro de ellos y no nos percatamos. Algunos ejemplos son los televisores, las refrigeradoras, los vehículos, los sistemas de aire acondicionado y los vehículos.

---

[23] ARORA, Mohit. *Embedded System Design*. p.1

[24] VALVANO, Jonathan. *Embedded Systems: Introduction to ARM Cortex M Microcontrollers*. p. 69

[25] Real Academia Española. *Diccionario de la Lengua Española*. https://dle.rae.es/?w=embebido. Consulta: febrero de 2019

Usualmente, los sistemas embebidos son sistemas computacionales completos, con los mismos elementos que tiene una computadora personal (PC), pero con un fin completamente diferente. Un sistema embebido normalmente está basado en un sistema basado en un microcontrolador, que está construido para controlar una función o una serie de funciones y que no está diseñado para que el usuario final la programe o modifique de la manera que una PC lo puede hacer. Por ejemplo, en una PC el usuario puede usar un procesador de texto, después ver una película, usar software de diseño, y usarla para una variedad de tareas extensa.

En un sistema embebido, el usuario final probablemente pueda tomar decisiones respecto al funcionamiento del sistema, pero no puede agregar funciones ni usos diferentes agregando software o programas personalizados[26].

Las principales características de un sistema embebido son:

- Cuenta con hardware y funcionalidad limitada de software: esto limita su capacidad de ejecución, pero reduce su consumo de energía; pueden tener o no un sistema operativo.
- Están diseñados para una función específica: muchos sistemas embebidos cumplen solo una o varias tareas específicas. Raramente son sistemas generales que pueden usarse para ejecutar cualquier programa de elección del usuario.
- Pueden ser de alta calidad y muy confiables: algunos dispositivos embebidos pueden operar largos periodos de tiempo sin fallas, y son muy confiables.

[26] HEATH, Steve. *Embedded Systems Design*. p. 2-12

- Operan con un programa en tiempo real: por la naturaleza de sus aplicaciones, los sistemas embebidos funcionan por medio de interrupciones, donde sus acciones son definidas por eventos, son disparados por eventos externos o internos al sistema, proveyendo baja latencia en la respuesta. Estos sistemas funcionam normalmente con sistemas operativos en tiempo real (*Real Time Operating Systems, RTOS*) para proveer determinismo en los tiempos de ejecución de tareas específicas[27].

Existen dos formas de implementar sistemas embebidos. La primera utiliza microcontroladores, y es la más usual. Por tanto, se suele restringir el término embebidos a aquellos sistemas que no se ven ni se comportan como una computadora personal. Al implementar los sistemas con microcontroladores, normalmente no se utilizan sistemas operativos, por lo que el sistema completo de software es desarrollado. Estos dispositivos son adecuados para aplicaciones de bajo consumo de energía y baja complejidad. La segunda forma se utiliza para desarrollar sistemas embebidos de alta capacidad, basados en microprocesadores más capaces. Estos sistemas típicamente incluyen un sistema operativo, y se desarrollan las aplicaciones con base a este. Ambas formas se implementaron en este proyecto para el desarrollo satisfactorio del mismo.

3.3.1. Arquitectura ARM

La arquitectura ARM ha dado forma a la historia reciente de la computación. Con el alza de la computación móvil en teléfonos y *tablets*, ARM se ha vuelto muy importante. La arquitectura ARM tiene un gran impacto en el mundo actual y en los sistemas embebidos.

[27] ARORA, Mohit. *Embedded System Design*. p. 3-5

Sus aplicaciones se extienden desde los microcontroladores hasta las computadoras personales, así como los celulares, las *tablets*, servidores, inteligencia artificial y *machine learning*, y aplicaciones de seguridad por hardware. Los procesadores ARM son muy versátiles, de bajo costo y de bajo consumo de energía.

En los años 80, por la empresa británica Acorn Computers, ahora ARM Holdings, buscaba un procesador para su línea de computadoras personales. Steve Furber y Sophie Wilson desarrollaron una arquitectura propia de procesadores denominada Acorn RISC Machine (ARM). Como su nombre lo indica, está basada en la arquitectura RISC, y ahora solamente es conocida como ARM. El modelo de negocios de ARM Holdings es por licenciamiento de los diseños de los procesadores, para que cada fabricante integre los núcleos a sus propios diseños.

Las versiones de la arquitectura se designan como ARMv*X*, y se extienden desde ARMv1 hasta la actual, ARMv8. Desde la implementación de la ARMv7 existen tres tipos de núcleos Cortex:

- Cortex-R (real time): para aplicaciones en tiempo real, con baja latencia, alta predictabilidad y memoria protegida.
- Cortex-M (microcontroller): bajo consumo de energía, bajo número de transistores y predictibilidad.
- Cortex-A (application processors): alto desempeño con bajo consumo de energía, optimizados para la multitarea.

Dependiendo del uso, los núcleos se implementan de manera diferente. Desde núcleos aislados, sin *pipelining* hasta multinúcleos con ejecución fuera

de orden, predicción de saltos y ejecución especulativa, comparable a los procesadores de computadoras personales más usados en el mercado.

La arquitectura interna de los procesadores ARM hasta la versión 7 estuvo basada en registros de 32 bits, convirtiéndose en una arquitectura de 64 bits en la versión 8. Esto significa que hasta la versión 7, manejaba un máximo de 4GB de memoria física, mientras que a partir de la versión 8, este límite ya no existe. Al igual que otros microprocesadores, tiene diversos modos de procesador. Los programas ordinarios corren en el modo de usuario, el código privilegiado del sistema operativo funciona en modo de sistema. El procesador tiene registros específicos para cada modo, que el sistema remapea durante los cambios, manejando las interrupciones de manera muy eficiente. Muchas de sus implementaciones poseen además una Unidad de Administración de la Memoria (*Memory Management Unit*) para virtualización de almacenamiento y protección de la memoria.

Como es un procesador RISC, tiene un set de instrucciones reducido, con un hardware optimizado para la lectura y escritura de la memoria en los registros de propósito general, que son muchos comparados con un procesador CISC como el x86 de Intel. Como tiene una pequeña cantidad de instrucciones, y pocos modos de direccionamiento de memoria, todas las instrucciones se pueden codificar en 32 bits. Esto aprovecha la arquitectura de los buses y registros de 32 bits, y simplifica los circuitos de decodificación de instrucciones. Todo lo que debe hacer el decodificador es cargar una instrucción de la memoria y decodificarla.

Cada procesador ARM tiene alrededor de 40 registros, pero solo 16 son accesibles al usuario. El resto es para uso interno del procesador en los distintos modos de ejecución. De los registros R0 a R12, el programador puede hacer

cualquier uso de ellos, mientras que el R13 es el registro de pila, R14 es el registro de enlace para guardar las direcciones de retorno en llamadas de procedimientos especiales, y el R15 es el contador de programa. Una de las características principales de la arquitectura es que casi todas las instrucciones pueden ser condicionales. Esto se debe al uso de una máscara especial de 4 bits que especifican qué condiciones se deben cumplir para ejecutar una instrucción (negativo, cero, acarreo, desbordamiento), permitiendo que el código sea compacto y evite saltos innecesarios.

La arquitectura ARM ofrece soporte para coprocesadores, como unidades de punto flotante. También posee la capacidad de ejecutar otros tipos de instrucciones, conocidos como *Thumb*, para aumentar la densidad de código por medio de instrucciones de 16 bits; *Neon*, con soporte para gráficos por medio de aceleración de hardware; y *Jazelle*, que ofrece aceleración de hardware para la interpretación de *bytecode* de Java. En las últimas versiones de la arquitectura se ha provisto la capacidad de ejecutar instrucciones paralelamente con la introducción del procesamiento multinúcleo. Esto permite mejor rendimiento con menor consumo de energía[28].

3.4. Microcontrolador

Se explicó previamente que muchos sistemas embebidos están basados en microcontroladores. Esta sección pretende diferenciar los microprocesadores de los microcontroladores y dar una idea de por qué se adaptan tan bien a las tareas que se realizan en los sistemas embebidos.

[28] RICHLING, Jan and BUSSE, Anselm. *The ARM architecture – yesterday, today, and tomorrow*. http://www.linux-magazine.com/Issues/2013/156/ARM-Architecture. Consulta: enero de 2019

Un microprocesador es la unidad central de procesamiento de una computadora digital. Para construir una microcomputadora se necesitan otros componentes externos, como memoria adicional, interfaces, y puertos de E/S. Estos se agregan de forma externa, conectados por medio de circuitos especializados. El microprocesador es prácticamente el corazón de un sistema de cómputo. Un microcontrolador, por su parte, es un sistema autónomo, que tiene un procesador, memoria y periféricos capaces de entrada y salida, por lo que muchas veces solo se necesita agregar código para poder utilizarlos dentro de un sistema embebido. En resumen, un microcontrolador es un sistema de microprocesador con todos los dispositivos de soporte integrados en un solo circuito integrado. No necesitan componentes externos, ya que todos los circuitos necesarios se encuentran en los periféricos construidos en el integrado.

3.5. Periféricos del microcontrolador

Uno de los criterios más importantes a considerar en la selección del microcontrolador más adecuado para un sistema embebido es el de los periféricos que incluye. El sistema embebido se debe comunicar con el mundo exterior y eso se realiza a través de los periféricos. Los periféricos de entrada pueden ser sensores o dispositivos seriales, que obtienen información externa y así pueden controlar efectivamente las operaciones de salida realizadas por el sistema. La operación de los microcontroladores se puede obtener en tres etapas: primero el microcontrolador obtiene la información externa, después la procesa, y por último refleja la información por sus salidas.

Es muy importante definir el uso que tendrá el microcontrolador. Esto permite seleccionar uno que tenga los periféricos más apropiados para la aplicación del sistema embebido.

Los periféricos más comunes en un microcontrolador pueden incluir *timers*, relojes de tiempo real, o contadores de interrupción periódica. Es importante determinar la capacidad de los contadores y *timers*, para que estén dentro de los lineamientos necesarios. También se incluyen las líneas de entrada y salida, como puertos de entrada/salida de uso general, puertos de comunicaciones seriales como UART, SPI o I2C, o incluso comunicaciones paralelas. Otros periféricos importantes pueden ser los convertidores analógico-digital (ADC), los convertidores digital-analógico (DAC), y los moduladores de ancho de pulso (PWM) para controlar motores. Es importante también conocer las capacidades propias de cada periférico, para poder utilizar los recursos que tiene el microcontrolador de una manera efectiva.

3.6. Máquinas de estado finitas

Una máquina de estados finitos, o *Finite State Machine (FSM)*, por su nombre en inglés, es un modelo de abstracción para la manipulación de símbolos. Estas se utilizan para realizar procesos bien definidos en un tiempo discreto. Es un modelo abstracto de una computadora que puede estar en un estado de una serie de estados finitos en un momento dado. La máquina de estados finitos, o el autómata finito, como también se le conoce, puede cambiar de un estado a otro en respuesta a algún impulso externo o interno. Este cambio se conoce como transición. Los principios de abstracción en el diseño de una FSM son sus entradas, salidas, estados y transiciones de estados.

Para plantear una FSM es necesario encontrar una serie finita de pasos o estados necesarios para resolver el problema que tenemos. Esto se puede realizar con la ayuda de los algoritmos, que presentan una serie finita de pasos para realizar un proceso. Si podemos tomar un problema difícil y plantearlo como una máquina de estados, podemos resolverlo con las herramientas de software

para escribir máquinas de estado finitas. Una implementación bien hecha será fácil de implementar, de depurar y de cambiar.

Para diseñar una máquina de estados finitos el primer paso es definir qué constituye un estado. Por ejemplo, en un sistema como un semáforo puede ser un patrón específico de luces. El siguiente paso es realizar un listado de todos los pasos en los que el sistema pueda estar. Se le agregan salidas para que modifique su entorno, así como entradas para que pueda recolectar información de su ambiente o recibir comandos.

Existen dos tipos de máquinas de estados finitas. Las FSM tipo Moore, y las FMS tipo Mealy. En una FSM tipo Moore el valor de salida depende solamente del estado, y las entradas provocan las transiciones de estado. Su secuencia de ejecución es la siguiente:

- Escribir la salida, depende del estado actual.
- Esperar un tiempo determinado (opcional).
- Recibir señal de entrada.
- Pasar al siguiente estado, que depende del estado actual y la entrada.

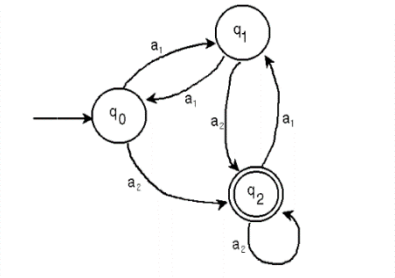
En las FSM tipo Mealy las salidas y las transiciones dependen tanto de las entradas como del estado actual. Su secuencia de ejecución es la siguiente:

- Esperar un tiempo predeterminado (opcional).
- Recibir señal de entrada.
- Escribir la salida, que depende del estado actual y de la entrada.
- Ir al siguiente estado, que depende del estado actual y de la entrada.[29]

[29] VALVANO, Jonathan. *Embedded Systems: Introduction to ARM Cortex M Microcontrollers*. p. 303-306

Una máquina de estados finita puede ser representada por un diagrama de transiciones, que ilustra todos los estados en los que puede estar la máquina, así como la dirección de transición de los estados respecto a cada entrada. Cada estado se representa con un círculo, y cada transición de estado con su dirección por una flecha que indica las conexiones. Debe haber un nodo por cada posible combinación de estados con los símbolos de entrada, que deben ser finitos. El estado inicial q_0 es apuntado por una flecha que no proviene de ningún otro estado[30].

Figura 14. **Diagrama de transición de una FSM**



Fuente: GUTIERREZ, Alejandro. *Máquinas de estados finitos. Breve introducción*, p.5, http://uncomp.uwe.ac.uk/genaro/Papers/Veranos_McIntosh_files/alejandroFinal2008.pdf. Consulta: febrero 2019

3.7. Diseño del controlador del robot

El diseño del controlador del robot se realizó en 4 etapas que se explicarán a detalle. Primero se seleccionó el hardware adecuado para realizar la tarea. Esto incluye los dispositivos electrónicos y actuadores. Con estos elementos se puede buscar el hardware de control. Se realizó el diseño del controlador que el piloto utilizó para enviar instrucciones al robot. Se diseñó el controlador del robot, que

[30] GUTIÉRREZ OROZCO, Jorge Alejandro. *Máquinas de estado finitos*. Complex Celular Automata Repository. p.12-13

identifica las instrucciones generadas por el piloto y las ejecuta de la forma más eficiente posible. Por último, se implementó en software y se depuró para que no tuviera problemas que afectaran el funcionamiento general del robot.

- Selección del hardware

La primera etapa para el diseño del controlador es escoger el hardware que compone el robot. Esto es importante hacerlo, ya que, según las necesidades de control del hardware como motores, actuadores y otros componentes funcionales, podemos determinar los mejores elementos para controlarlos. También se deben considerar los aspectos de comunicaciones y facilidad de desarrollo y programación. Como se explicó previamente, el robot cuenta con dos brazos mecánicos conformados por 4 servomotores. Existen también 5 motores DC, que son controlados por puentes H, y cuya lógica de control debe ser transmitida individualmente para tener un control total del robot. Como el control del robot es realizado remotamente, el ente organizador de la competencia ofrece un enlace punto a punto mediante una red wifi. La implementación a detalle de este punto se realizará en el capítulo 5. La última consideración, es que el controlador tiene que ser de bajo consumo de energía, ya que es un sistema que está alimentado por medio de una batería.

Priorizando la conectividad y la facilidad de administrar conexiones por medio de wifi, así como las posibilidades flexibles de programación, se optó como controlador principal la computadora Raspberry Pi 3 Modelo B. Esta computadora cuenta con un procesador ARM-A57 construido por Broadcom. El Broadcom BCM2837 es un procesador *quad-core* de octava generación, de 64 bits. Cuenta con wifi y bluetooth LE (*Low Energy*), puerto Ethernet, 4 puertos USB, entre otros. Cuenta con 1GB de RAM, y un puerto microSD para agregar memoria para instalar el sistema operativo y guardar los datos de almacenamiento.

Utiliza una fuente de 5 voltios y 2,5 amperios, que equivalen a 12,5 Watts. Ofreciendo todas las prestaciones de una computadora personal, con un consumo tan bajo de energía y un costo de $35 USD, la convierten en la opción perfecta para sistemas móviles con requerimientos medios de capacidad de computación. Cuenta además con un puerto CSI para un módulo de cámara, perfecto para el control remoto de un sistema.

La selección de la Raspberry Pi como controlador simplifica las conexiones de redes y el manejo de la cámara. Pero para manejar el hardware no es suficiente. Para controlar cada servomotor con precisión es necesario usar PWM (*Pulse Width Modulation*), pero la Raspberry Pi solo cuenta con dos módulos de hardware dedicados a esta tarea. Se pueden utilizar otras salidas como PWM, configurándolas por medio de software. Pero como se explicó anteriormente, los sistemas embebidos, especialmente los de control, deben funcionar en tiempo real. La programación de PWMs y el uso de pines de propósito general (GPIO, *General Purpose Input Output*) requieren el uso de *delays*, y eso retrasa la respuesta en tiempo real del sistema.

Por lo tanto, se optó por buscar un microcontrolador secundario, con periféricos especializados para controlar el robot. Para esta tarea se seleccionó el microcontrolador TM4C123GH6PM, de Texas Instruments. Este microcontrolador está basado en la arquitectura ARM Cortex-M4F, que está contenido en la tarjeta de desarrollo Launchpad Tiva C, diseñada por Texas Instruments. Es una tarjeta de desarrollo de bajo costo, del tamaño de una tarjeta de crédito. Se optó por utilizar el microcontrolador en la placa de desarrollo en lugar de desarrollar una placa dedicada, principalmente por el tiempo necesario para diseñar una PCB de esa dificultad.

El microcontrolador TM4C123GH6PM fue seleccionado porque cuenta con una cantidad de periféricos que posee, ideal para manejar una gran cantidad de hardware simultáneamente.

Sus principales características son:

- Reloj de operación a 80 MHz.
- 8 UARTS.
- 43 GPIO reprogramables a los periféricos.
- 16 salidas PWM.
- Instrucciones Thumb-2 de 16 y 32 bits.
- Manejo determinístico de alto rendimiento de las interrupciones para sistemas de tiempo crítico.
- Sistema de depurado integrado (ICDI).
- Bajo consumo de energía.
- *Nested Vectored Interrupt Controller* (NVIC), para el manejo de las interrupciones.

Una de las principales razones para la selección de esta tarjeta de desarrollo es la gran cantidad de pines de propósito general, casi todos con posibilidad de manejo de interrupciones. Esta característica es ideal para el manejo de un sistema en tiempo real. Su alto número de salidas de PWM provee una plataforma ideal para controlar un gran número de motores y servomotores. Por último, la arquitectura ARM con instrucciones *Thumb-2* asegura que las instrucciones se ejecutarán lo más rápido posible.

La comunicación entre las computadoras se realizó por medio del protocolo serial UART. La Raspberry Pi cuenta con un módulo UART por hardware y uno por software. El módulo de hardware es de alto rendimiento, y está conectado

al bluetooth. Para tener una comunicación confiable, se cambió el controlador de hardware para atender a las comunicaciones a través de UART en vez de las comunicaciones bluetooth. Esto aseguraba una comunicación rápida y sin errores, que permitía controlar el hardware de manera ideal.

- Diseño del controlador que usaba el piloto

Con el hardware seleccionado, se procedió al diseño de dos programas que controlan el robot. Uno para el piloto, para obtener los datos que el generaba para dar instrucciones al robot, y uno para controlar el hardware. Para el manejo del robot se optó por usar un control genérico de videojuegos. Con el fin de evitar problemas de *drivers* y asegurar la estabilidad del programa, se utilizó Ubuntu 16.04 como sistema operativo para el desarrollo de todos los programas, así como Python 2.7 como lenguaje de programación para las computadoras y el lenguaje C para el microcontrolador.

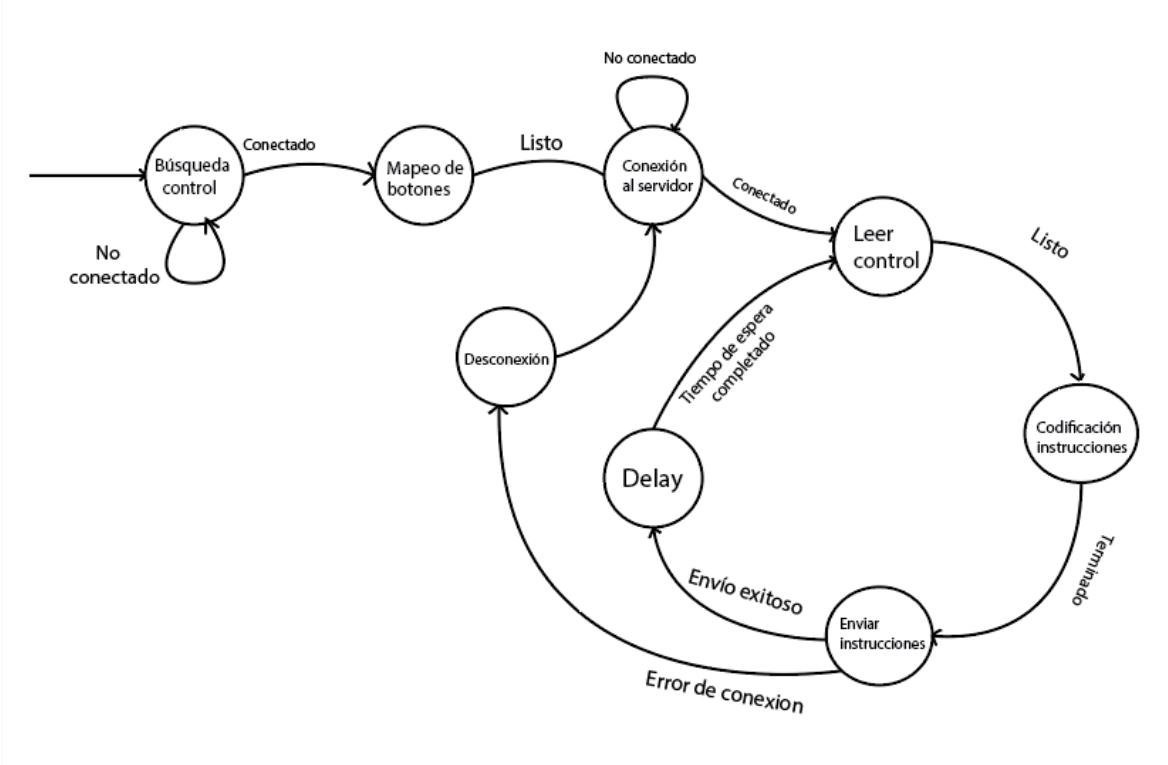
El primer paso para realizar el programa de control es mapear el control. Esto se realiza para identificar las instrucciones que se generan al presionar los botones y así poder utilizarlas a voluntad. Python ofrece la librería *pygame* para interactuar con controles de videojuegos de distintos fabricantes, con posibilidades de mapear controles genéricos. Para obtener el mejor rendimiento en el manejo del robot, es muy importante que el piloto esté cómodo con los controles. En base a lo que el piloto prefiera se deben generar los controles. Con el mapeo realizado eficientemente, se puede asignar cualquier botón a cualquier instrucción a ejecutar.

Lo primero que hace el programa es revisar los periféricos conectados al dispositivo. Si el control no está conectado, espera que la conexión se realice antes de empezar. Casi todos los controles tienen botones digitales y *joysticks* y

botones analógicos. Los botones digitales generan los valores 1 y 0, mientras que los analógicos generan un valor entre 0 y 1. Se debe considerar que no se generan valores negativos, por lo que la programación debe solventar este problema por medio de software, para poder indicar las direcciones de los movimientos y diferenciar que botón es presionado.

La librería *pygame* tiene una documentación extensa que permite conocer el valor de cada botón y *joystick* del control, y funciones para poder obtener los valores de los botones en un determinado momento. Con esta información, se puede hacer una máquina de estados finitos para leer los botones, codificar la información para generar las instrucciones y enviarla al robot, cuyo diagrama es:

Figura 15. **Diagrama de estados controlador para el piloto**



Fuente: Elaboración propia

Los estados son:

- Búsqueda de control: verifica que un control compatible esté conectado para poder obtener instrucciones. Si no hay un control conectado, espera a que se conecte uno.
- Mapeo de botones: Obtiene los identificadores de cada uno de los botones y *joysticks* del control, tanto analógicos como digitales.
- Conexión al servidor: Crea una conexión TCP/IP con el servidor ubicado en el robot a través de un *socket*. Permanece en este estado hasta que se establece la conexión.
- Leer el control: Revisa uno a uno los botones seleccionados para el uso, y obtiene su valor un momento dado.
- Codificación de instrucciones: En base al valor del botón o *joystick* accionado, define que instrucción se ha generado, le asigna un carácter ASCII, que se une a una cadena de caracteres de 10 bits. Las posibles instrucciones son: avanzar hacia delante, hacia atrás, hacia ambos lados. Mover el brazo robótico de la carga, el brazo robótico de la cámara en sus respectivos grados de libertad, encender o apagar la luz, abrir o cerrar la garra, abrir o cerrar el compartimiento de la carga y cambio de velocidad.
- Envío de instrucciones: Envía la cadena de caracteres a través del *socket* TCP/IP al servidor. La duración del siguiente estado determina el *data rate* de la conexión.
- *Delay*: La transmisión de datos continua generaría demasiados datos para procesar, generando un cuello de botella en el receptor, que debe decodificar y ejecutar todas las instrucciones. Para asegurar que las instrucciones son recibidas correctamente y ejecutadas a cabalidad se debe dejar un tiempo de espera.

- Desconexión: Si el servidor se desconecta, se reinician los parámetros de conexión para volver a realizar la petición de conexión.

- Diseño de los controladores del robot

La decisión de colocar dos computadoras altamente eficientes para el control del robot con fines diferentes resulta en dos programas de control. El desarrollo del software que controla el robot se complica, pero las ganancias en rendimiento son justificables, principalmente porque el sistema se vuelve determinista, ya que los datos siempre son del mismo tamaño y llegan a intervalos regulares. Esto permite diseñar un controlador eficiente y rápido, necesario para que el robot realice los movimientos en tiempo real.

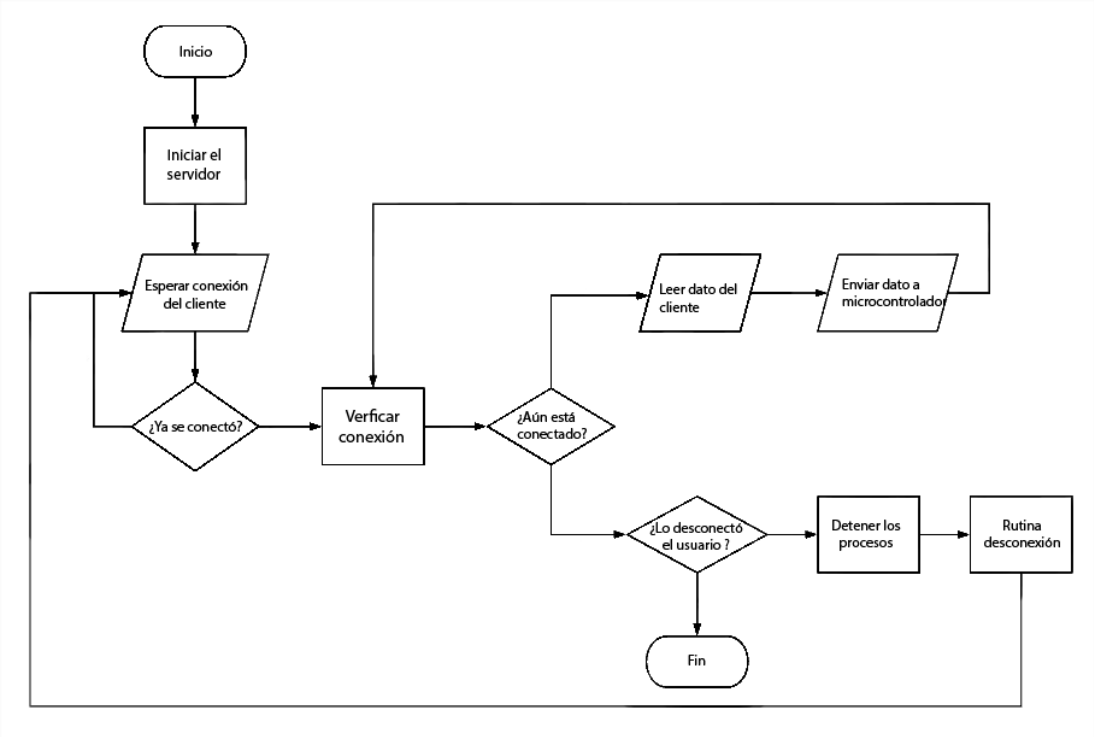
El primer sistema es el de la Raspberry Pi, que maneja las comunicaciones. En ella está implementado un servidor TCP/IP, por medio del cual se reciben las instrucciones. El controlador es sencillo en su naturaleza. Crea el servidor y espera por una conexión, hasta que un cliente se conecta y empieza a procesar los datos. El cliente envía datos de una manera determinística, ya que en un intervalo de tiempo se manda una cantidad específica de datos. Esta característica permite detectar problemas en la transmisión y tomar las decisiones adecuadas para combatir el problema. Si la transmisión fue exitosa, el programa decodifica la instrucción de 10 bits, y la interpreta para convertirla en una cadena de dos caracteres.

Para la ejecución de las instrucciones, se dividieron en dos tipos: instrucciones instantáneas e instrucciones fijas. Las instrucciones instantáneas afectan las acciones del robot. Deben ser provistas continuamente para que el robot se mueva, y siga la dirección establecida. Las instrucciones fijas cambian

estados del robot. Por ejemplo, la instrucción instantánea le indica al robot a dónde debe girar. Se le envía la dirección durante el tiempo necesario para que el robot alcance el ángulo deseado. La dirección fija, por su parte, enciende y apaga la luz. El robot debe ser capaz de encender la luz mientras se mueve, el estado de la luz es irrelevante al movimiento del robot. Para poder realizar estas acciones de manera simultánea, se establecieron estos dos tipos de instrucciones.

El programa recibe la cadena de 10 bits e identifica la instrucción fija y la instrucción instantánea, y asigna un carácter para cada una. La nueva instrucción de dos caracteres se envía a través de UART a la Tiva. Esto reduce el tiempo de transmisión hacia el microcontrolador, y reduce la carga computacional para el mismo, disminuyendo el número de combinaciones que el microcontrolador debe descifrar, dejando esta tarea a la Raspberry Pi, que tiene un procesador más capaz con un reloj más rápido. Esto permite que la ejecución de las acciones en el hardware ocurra en tiempo real. El diagrama de flujo se presenta a continuación.

Figura 16. **Diagrama de flujo del servidor receptor de instrucciones**



Fuente: elaboración propia

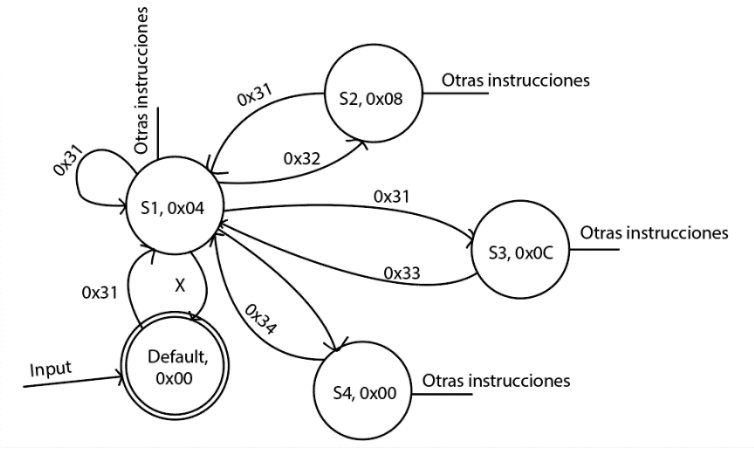
Una de las características más importantes del servidor receptor es la rutina de desconexión. Esta detecta cuanto se ha perdido la conexión con el cliente. Esta debe ser muy rápida para actuar, porque si la conexión se pierde y el servidor no detecta la desconexión, puede seguir ejecutando la última instrucción hasta que se reinicie el servidor. Esto puede provocar que el robot choque o pierda puntos, siga avanzando indefinidamente en una dirección o provoque algún accidente. La rutina de desconexión consiste en enviar un carácter especial a la Tiva para que cancele todas las acciones actuales, y reiniciar todos los procesos del servidor para esperar por una nueva conexión. El proceso preciso para detectar la desconexión y gestionar las nuevas conexiones serán discutidos más extensamente en el capítulo 5.

El controlador de hardware embebido en el microcontrolador Launchpad Tiva C, no se basa en un modelo de máquina de estados finita, pero su comportamiento se puede analizar como una máquina de estados finita, del tipo Moore. La instrucción que llega dice qué acción realizar y pasa del estado actual al siguiente, donde obtiene el nuevo valor de la salida. Cada estado puede pasar a cualquier otro, sin ninguna limitación.

No se puede considerar una máquina de estados porque la ejecución de estados no es continua, y cambia el orden de ejecución según las necesidades actuales del programa, ramificando el código[31]. La siguiente ilustración da una idea simplificada de cómo puede cambiar de un estado a otro, y de todas las posibles transiciones dentro de un solo estado, contando con 10 estados diferentes, y 9 entradas válidas que producen cambios a un estado de ejecución de instrucciones. Cualquier otra entrada no válida es descartada y se dirige al estado *default*, que detiene cualquier proceso en ejecución.

[31] KELLER, Robert M. *Computer Science: Abstraction to Implementation*. *12. Finite-State Machines.* https://www.cs.hmc.edu/~keller/cs60book/%20%20%20All.pdf. p.496-501. Consulta: marzo de 2019

Figura 17. **Diagrama simplificado de las conexiones posibles para el estado S1**



Fuente: elaboración propia

La máquina de estados está implementada en C, con ayuda de la librería TivaWare, creada por Texas Instruments para facilitar el desarrollo de aplicaciones con la serie de microcontroladores Tiva C. Al estar implementado en C, las máquinas de estado están implementadas con sentencias *switch-case*. Estas son más eficientes ya que pueden ser optimizadas en el proceso de compilación. La ejecución de esta función es revisar una a una las diferentes opciones hasta encontrar una coincidencia, y si existe la instrucción *break*, detenerse en ese punto.

La implementación de la rutina de desconexión se basa en el orden de ejecución de las sentencias *switch-case*. Como primera instrucción en ambas máquinas de estado se coloca la interrupción. La rutina detiene todos los motores, aunque mantiene la posición de los servomotores, y reinicia el control de velocidad.

Después, alterna el estado de la luz entre encendido y apagado en intervalos de tiempo regulares, con un código basado en contadores. De esta manera se puede asegurar que la rutina de desconexión tiene prioridad sobre el resto de las instrucciones.

Las instrucciones instantáneas dependen de una máquina de estado, y su salida depende únicamente del estado en el que se encuentra. Por ejemplo, para mover el brazo de la cámara, se utiliza un módulo de PWM para controlar el servomotor del grado de libertad correspondiente. Al llegar la instrucción al controlador se realiza la acción correspondiente. Si era girar el brazo a la derecha, no importa la orientación del brazo, se intentará girar a la derecha. Por medio de software se limita el movimiento del brazo para no dañar los motores. Cada una de las acciones no depende del estado anterior para que el robot pueda reaccionar, independientemente de la acción actual, de forma inmediata a las instrucciones generadas por el piloto.

Para que las acciones se realicen en tiempo real, se optó por una ejecución rápida del código, que no depende de *delays*. Para cualquier acción de espera se utilizaron contadores, por lo que cada instrucción que llegaba podía ser interrumpida por medio de las ISR (*Interrupt Service Routine*), y que al regresar de la rutina de interrupción el procesador estuviera en total disponibilidad de trabajar.

En la aplicación de un robot controlado de forma remota esto es muy importante, ya que la respuesta a las instrucciones es de vital importancia. Las instrucciones generadas por el piloto en el control se leen a intervalos de tiempo regulares.

Ya que se usaron tramas de bits cortas, la transmisión es muy rápida. Incluyendo el tiempo de decodificación en Python en la Raspberry Pi, y la transmisión por medio de UART, la ejecución de la instrucción debe realizarse antes que se lea nuevamente el control que utiliza el piloto.

Si el control de tiempo en el microcontrolador no se realiza tomando en cuenta el tiempo de ejecución de las instrucciones, se perderá la sincronización con el microcontrolador y este no estará listo antes que la siguiente instrucción llegue. Esto provoca instrucciones retrasadas (debido a la memoria FIFO del receptor UART) e inestabilidad en la ejecución de las instrucciones. En una competencia donde el control que se tiene sobre el robot es muy importante es un aspecto que no puede descuidarse. Si se utilizan *delays* en un sistema de tiempo real, las probabilidades de que el sistema no esté en sincronía son muy altas, y se generarán saltos en la ejecución de instrucciones, reduciendo el control del robot por parte del piloto.

Todo el código, debidamente comentado, se puede acceder en el repositorio SentinelGT en Github, o a través del siguiente enlace.

- https://github.com/mendezrod57/SentinelGT

4. SISTEMAS ELECTRÓNICOS EN UN ROBOT

Los robots son sistemas complejos que involucran distintas áreas del desarrollo científico y la ingeniería. Uno de los principales retos es utilizar los elementos de hardware y enlazarlos con el sistema de control para poder tener un robot funcional, que responda a las necesidades para las cuales es creado, y pueda ser eficiente en su operación. Este capítulo presenta los elementos de hardware seleccionados para construir el robot, que son los actuadores y que proveen de movimiento al robot, y como son controlados por medio del software para dar la funcionalidad requerida para participar en la competencia.

4.1. Interfaz con el mundo real

Un robot, ya sea móvil, estacionario, o una combinación de las dos, es mucho más que los elementos mecánicos que los componen. Se puede considerar que, en general, es un sistema robótico. Un sistema robótico está integrado por los actuadores que lo mueven, así como los eslabones y las articulaciones que permiten y dirigen el movimiento. También posee una interfaz de computadora en la computadora de control, sensores internos y externos, una fuente de energía externa y efectores finales. El software que controla el robot se puede considerar también parte del sistema, porque puede alterar el funcionamiento y el rendimiento del robot[32].

El software y la computadora que controla el robot, presentados en el capítulo 3, dirigen las acciones del robot según las instrucciones que el conductor

[32] SPONG, Mark W., HUTCHINSON, Seth and VIDYASAGAR, M. *Robot Modeling and Control*. p. 7

ingresa en la computadora. El conductor tiene un control, en el que, al presionar botones y mover los *joysticks* ingresa información, que es interpretada por la computadora que él utiliza, y la envía al robot. El robot recoge información por medio de la cámara y la envía hasta el conductor.

El conductor, por medio del control, puede controlar las acciones del robot. El software puede interpretar las instrucciones, y definir si el conductor indicó “hacia adelante”, o “girar hacia la derecha”. El software entonces genera señales de control, que permiten que el robot interactúe con su alrededor a través del hardware que lo compone. Uno de los programas del robot controla con precisión los actuadores que componen al robot y le permite realizar acciones como girar, recoger la carga y asegurarla.

4.2. Actuadores y efectores

Los actuadores y efectores fueron discutidos brevemente en el capítulo dos, con algunas características de los actuadores y el efector final elegido para los brazos robóticos. En esta sección se consideran nuevamente estos y los actuadores y efectores utilizados para el sistema de locomoción del robot completo.

4.2.1. Motores DC con tren de engranajes

Los motores DC son los actuadores que se eligieron para mover el robot. Ya que el diseño del robot es de cuatro llantas, se optó por poner un motor en cada llanta. Los motores DC son un tipo de actuador eléctrico, que convierte la energía eléctrica en movimiento rotacional, con el que giran las llantas que mueven al motor. En general, los motores presentan una velocidad rotacional alta, llegando a los miles de revoluciones por minuto.

Esto presenta una desventaja grande para el control del movimiento, ya que es difícil de regular por su alta velocidad, e incluso pueden presentar peligros por la misma razón.

Otra característica general de los motores DC es que no ofrecen un par o un torque elevado, que sea capaz de mover cargas pesadas. El robot diseñado y utilizado en la competencia posee un peso aproximado de 6 kilogramos. Si los motores no pueden entregar el par necesario para mover una carga de ese peso se vuelven inservibles. Sin embargo, existe un dispositivo mecánico que resuelve ambos problemas, y es el sistema de engranajes. El sistema de engranajes es un conjunto de engranajes diseñado para reducir la velocidad de rotación del motor, al mismo tiempo que se aumenta el par ofrecido por el mismo. Un motor acoplado a un sistema de engranajes bien diseñado provee una velocidad constante y regulada, junto con un par que puede mover cargas pesadas.

La competencia Mercury Robotic Challenge es una competencia cronometrada, en la que realizar el recorrido en poco tiempo es muy importante. Es importante seleccionar motores adecuados que ofrezcan la fuerza y la velocidad adecuada para mover el peso completo del robot en un tiempo competitivo. Ya que los sistemas de engranajes permiten aumentar el par ofrecido por el motor a cambio de reducir la velocidad, son una parte crítica del sistema. Para el robot utilizado se seleccionaron motores DC con un sistema de engranajes, con un torque de 1,5 kg/cm y 200 revoluciones por minuto. Con cuatro motores, el robot podía ser desplazado a una velocidad media de 1 metro por segundo.

El robot cuenta con dos velocidades diferentes, una para manejar lento con mayor control y maniobrabilidad, y una para manejar a máxima velocidad, con un control más reducido. El cambio de velocidad se realiza cambiando el voltaje de

entrada de los motores, ya que a mayor voltaje mayor velocidad de rotación. Para cambiar el voltaje que llega a los motores se realiza la conmutación de un *relay*, que cambia la alimentación de los motores entre 9 voltios y 12 voltios. No se utiliza PWM para variar la velocidad de los motores.

4.2.2. Servomotores

Los servomotores son sistemas de posicionamiento que se componen de un motor DC, un sistema de engranajes y un sensor de posición. El servomotor tiene la capacidad de colocarse en cualquier posición de su rango de operación, y permite un control preciso de la posición en la que se encuentra. El sensor de posición compara los valores actuales con los valores deseados, y realiza ajustes para obtener la salida deseada.

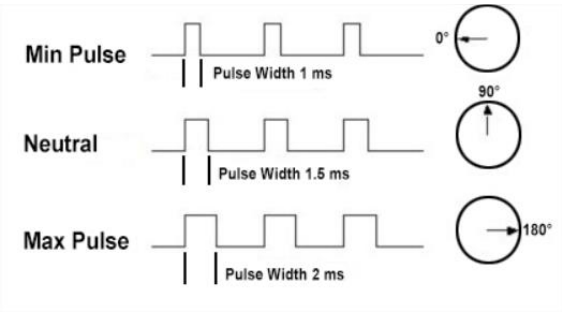
Los servomotores por lo general tienen una libertad de movimiento de 180 grados, aunque también existen de movimiento lineal o prismático. Su capacidad de ajustarse a una posición en su rango de movimiento los hace perfectos para aplicaciones de robótica. Si se implementan correctamente las posiciones mecánicas y un software de control adecuado, se pueden obtener resultados muy útiles.

Para el robot, se utilizaron dos tipos de servomotores diferentes. Se utilizaron servomotores angulares y servomotores prismáticos. Cada uno de los brazos robóticos construidos está compuesto de servomotores. El brazo de la carga utiliza servomotores de 20 kg/cm para garantizar la estabilidad del brazo en toda su extensión. El brazo de la cámara, mucho más pequeño y sin mucho peso, utiliza servomotores de 1 kg/cm para realizar los movimientos. La caja de resguardo de la carga utiliza servomotores lineales para abrirse y cerrarse.

Los servomotores se controlan con una señal eléctrica de pulso variable. Estos refrescan la información de posición cada 20 milisegundos y dependiendo de la duración del ancho de pulso el motor varía entre su posición máxima y su posición mínima. El ancho del pulso debe ser entre 1 milisegundo para su posición mínima y 2 milisegundos para su posición máxima.

A través del hardware dedicado de la Tiva C, se utiliza un generador de PWM para generar la señal de control. Cada uno de los servomotores está limitado por software en su movimiento para evitar daños y dificultades en la conducción del robot.

Figura 18. **Pulsos de control de la posición del servomotor**



Fuente: https://www.jameco.com/Jameco/workshop/howitworks/how-servo-motors-work.html. Consulta: marzo de 2019.

4.2.3. Efectores finales

Los efectores finales son las herramientas con las que el robot realiza las tareas para las que fueron diseñados. El robot tiene tres funciones principales para las que fue diseñado. La primera es poder desplazarse sobre superficies planas o con una inclinación de hasta 30 grados. Para que pueda realizar esto, se seleccionaron 4 llantas de caucho de 8 centímetros de diámetro, que proveen

suficiente estabilidad y agarre al suelo para conducir al robot en los diferentes terrenos en los que se pueda encontrar.

La segunda es poder recoger, transportar y entregar una carga provista por la organización de la competencia. Para esta tarea se escogió una pinza con dedos planos para poder transportar una carga de lados planos. Esto facilita el diseño del controlador. Esta pinza es accionada por un motor DC, que es controlado por un puente H para girar en ambas direcciones.

La tercera función que tiene el robot es recoger información visual y transmitirla al piloto para que pueda conducirlo a través de la pista. Para esta parte se utiliza el módulo de cámara para la Raspberry Pi, provista con un lente de gran angular de 120 grados. Esto permite al piloto obtener suficiente información visual de sus alrededores para dirigir el robot sin temor a chocar objetos frente a él o a sus alrededores inmediatos. La Raspberry Pi proporciona la fuente de poder y las señales de control necesarias para poder comunicar la cámara con la computadora.

4.3. Elementos de potencia

Para que un sistema electrónico funcione correctamente necesita una fuente de energía adecuada, y elementos que puedan aprovecharla bien y entreguen la potencia necesaria a las partes que lo componen. Los motores DC necesitan suficiente corriente para funcionar correctamente. Los sistemas de cómputo necesitan una fuente de voltaje constante y limpia, sin mucho ruido.

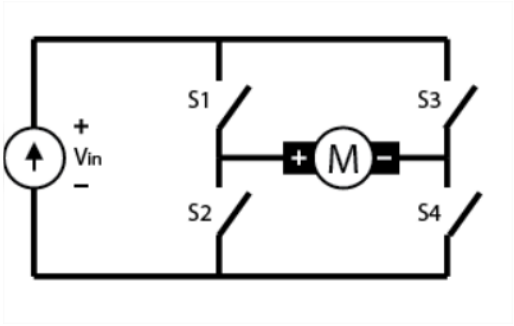
Los motores, tanto los servomotores como los motores convencionales necesitan mucha más corriente que los circuitos integrados en los microcontroladores. Por tanto, se debe utilizar una fuente de corriente capaz

de entregar distintos voltajes y corrientes según las necesidades de cada elemento. También se debe tomar en consideración la mejor forma de suministrar la potencia a todos los elementos, para que funcionen como es debido.

Con todas las consideraciones anteriores, se seleccionó una batería de la marca Talentcell, con salidas de voltaje reguladas de 12V, 9V y, a través de un puerto USB, 5V. La Raspberry Pi y la Tiva C utilizan 5 voltios, los servomotores 9 voltios y los motores 9 y 12 voltios para las velocidades altas y bajas. Al tener una batería con múltiples voltajes se tiene una sola referencia a 0 voltios, lo que evitan problemas de referencia y todos los sistemas funcionan sin importar el voltaje que utilicen en sus salidas y entradas.

Los motores DC giran en distinto sentido según la forma de la polarización. Esto permite utilizar los mismos motores para mover el robot en cualquier dirección. Para poder utilizar una sola batería y cambiar la forma de polarización a través de un microcontrolador se necesita un mecanismo especial que permita realizar una conmutación de la polarización. Uno de los circuitos más extendido para controlar la polarización de un motor es el puente H.

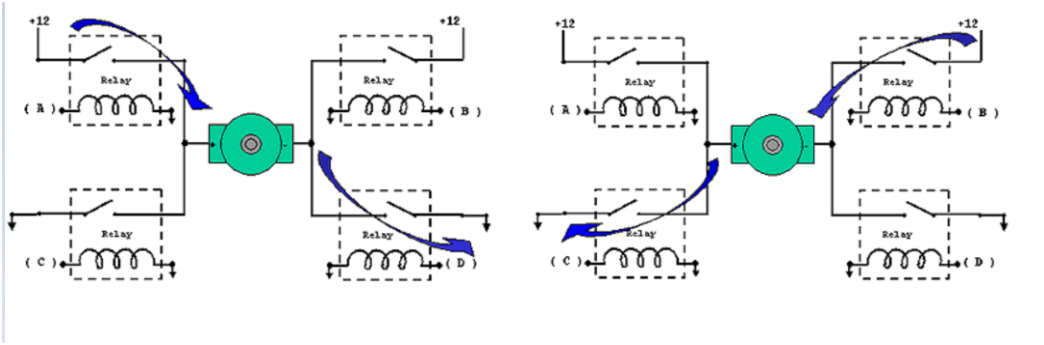
Figura 19. **Puente H**



Fuente: Digilent Blog. https://blog.digilentinc.com/what-is-an-h-bridge/. Consulta: marzo de 2019

El puente H consta con cuatro interruptores, de estado sólido o mecánicos, que controlan el flujo de corriente en una carga. Posee dos interruptores que proveen voltaje positivo a la carga, llamados fuente (*source*), y dos interruptores que proveen tierra, que permiten el drenaje de la corriente (*sink*).

Figura 20. **Flujo de corriente en un puente H**



Fuente: Dallas Personal Robotics Group. http://stan.cropsci.uiuc.edu/people/LSY_teaching/Fall2008/bioengin2008/Top/Manuals/Motors/DC_MotorDrivers/DPRGBriefH-BridgeTheoryofOperation.html. Consulta: marzo de 2019

Siguiendo el formato de la imagen anterior, si los interruptores A y D se activan al mismo tiempo el motor gira en una dirección, y si se activan los interruptores B y C se puede girar el motor en la otra dirección. Se debe tener cuidado de no activar los interruptores A y C o B y D a la vez para no generar un corto circuito que pueda destruir el puente H. Las señales de control para un motor son 4, y se muestran en la siguiente tabla[33].

[33] BROWN, Jim. *Brief H-Bridge Theory of Operation*. Dallas Personal Robotics Group. http://stan.cropsci.uiuc.edu/people/LSY_teaching/Fall2008/bioengin2008/Top/Manuals/Motors/DC_MotorDrivers/DPRGBriefH-BridgeTheoryofOperation.html. Consulta: abril 2019

Tabla II. **Señales de control para un puente H**

A	B	C	D	Función
1	0	0	1	Adelante
0	1	1	0	Atrás
1	1	0	0	Freno
0	0	1	1	Freno
1	0	1	0	Cortocircuito
0	1	0	1	Cortocircuito

Fuente: Dallas Personal Robotics Group
http://stan.cropsci.uiuc.edu/people/LSY_teaching/Fall2008/bioengin2008/Top/Manuals/Motors/DC_MotorDrivers/DPRGBriefH-BridgeTheoryofOperation.html. Consulta: marzo de 2019

Los puentes H se pueden construir utilizando transistores o *relays*, y también existen como circuitos integrados. Uno de los circuitos integrados más utilizados para construir puentes H, es el integrado LN298, que cuenta con dos puentes H, capaces de entregar hasta 4 amperios en total, y con entradas lógicas para las señales con voltajes de entrada TTL o CMOS. Existen en el mercado muchos módulos que implementan este circuito integrado en pequeñas placas de circuito impreso, que incluyen las protecciones necesarias para mover dos motores por circuito integrado.

Debido a la facilidad de integración con cualquier microcontrolador, se utilizó un módulo de cuatro puentes H que cuenta con dos LN298 y disipadores de calor, con capacidad de manejar hasta 4 motores. La decisión de controlar cada motor independientemente se hizo para tener más flexibilidad en el control del robot, ya que se pueden cambiar el giro y la velocidad de cada llanta de forma independiente. Para la pinza que posee el robot, que funciona con un motor para abrir y cerrar la pinza, se utilizó un puente H construido con componentes discretos. Cuando el motor gira en una dirección abre la pinza, y la cierra al girar en la otra.

Para mover este motor no se utilizó un módulo con el integrado LN298, ya que el motor es de menor tamaño y la corriente necesaria para moverlo no es tan grande. Esto permite utilizar componentes más pequeños y optimizar el espacio dentro del robot.

4.4. Placas de circuito impreso

En la construcción de un sistema electrónico, todos los componentes deben conectarse uno con otro para que funcione correctamente. Una de las herramientas más utilizadas para esta tarea son las placas de circuito impreso. Las placas de circuito impreso (PCB, por sus siglas en inglés) son utilizadas en la electrónica para sostener mecánicamente y conectar eléctricamente los componentes de un sistema electrónico[34]. Una PCB consiste en una superficie no conductiva que ha sido laminada con cobre. Sobre las láminas de cobre se graban los caminos, pistas o buses que interconectan los diferentes componentes en un circuito[35].

4.4.1. Diseño del PCB

Después de seleccionar cada uno de los actuadores, los efectores finales, las computadoras y los elementos de potencia y control se deben conectar eléctricamente para que el sistema electrónico funcione correctamente. Para facilitar el proceso de diseño y reducir los costos, se diseñó una PCB donde se puede conectar el microcontrolador Launchpad Tiva C directamente a través de sus pines a los diferentes componentes del sistema.

---

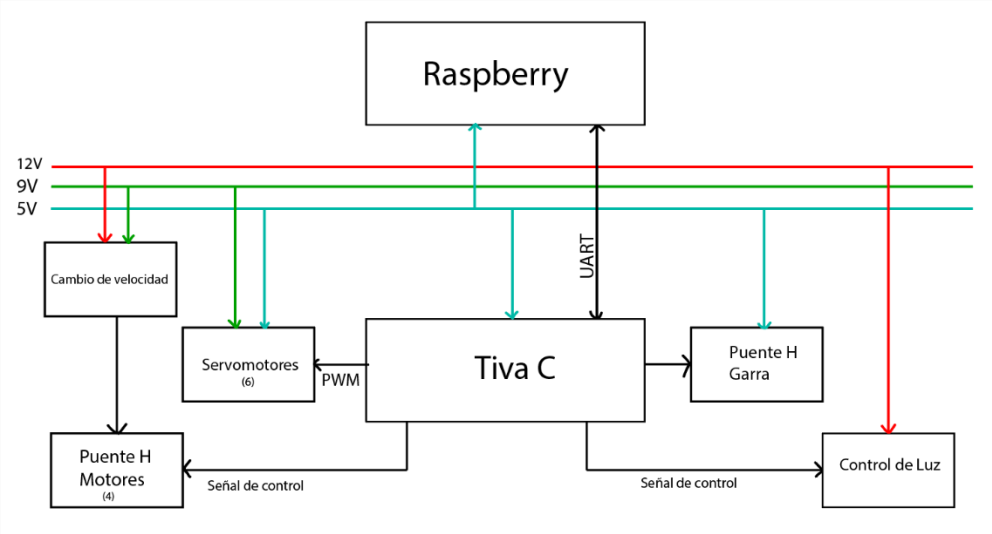
[34] COOMBS, Clyde. *Printed Circuits Handbook*. p. 3

[35] New World Encyclopedia contributors. *Printed circuit boards*. http://www.newworldencyclopedia.org/p/index.php?title=Printed_circuit_board&oldid=988465. Consulta: marzo de 2019

La decisión de conectar todos los componentes a través de una placa de circuito impreso en vez de los pines que ofrece la Tiva se debe a la forma en la que están ruteados los pines en la Tiva. Los pines en la tarjeta de desarrollo no están colocados en forma ordenada según los puertos GPIO, ni ofrece facilidades para conectar periféricos como dispositivos seriales, o los conectores estándar para los servomotores.

El robot cuenta con 6 servomotores, 5 puentes H para los 5 motores DC, un *relay* para el cambio de velocidad, una serie de luces led blancos de alta intensidad para la iluminación en ambientes oscuros y un led láser para referencia. Todos los actuadores son controlados por el microcontrolador en la Tiva, mientras que la Rapsberry Pi se encarga de las comunicaciones y la decodificación de instrucciones. Todos estos elementos deben estar conectados a la batería. La PCB fue diseñada para proveer una forma fácil de conectar todos los actuadores que utiliza el robot y facilitar el cableado, reduciéndolo en tamaño y complejidad para poder colocarlo dentro del robot. La siguiente figura presenta el diagrama de conexiones.

Figura 21. **Diagrama de conexiones eléctricas**



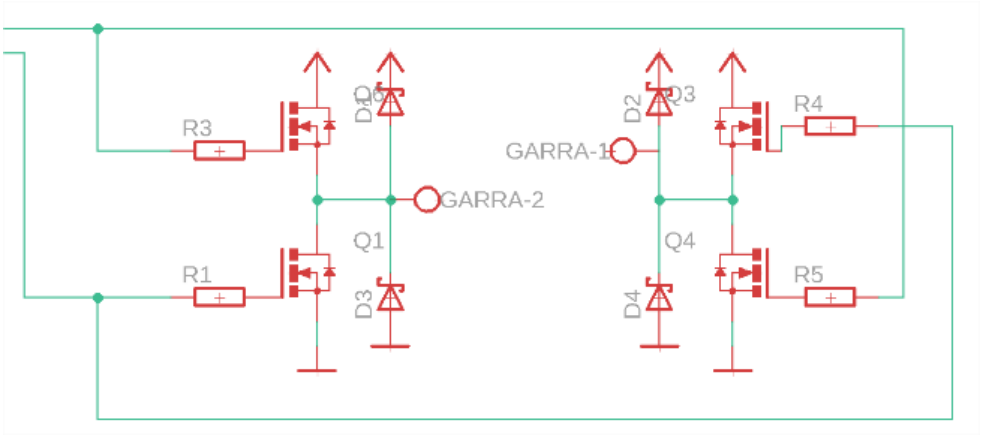
Fuente: elaboración propia

En la placa se integraron el circuito de control de las luces led, el circuito del cambio de velocidad y el puente H para la pinza. Esto eliminó el cableado externo para estos elementos. Se incluyeron *sockets* para conectar 9 servomotores con la conexión estándar de 3 pines, uno para la señal de control, uno para voltaje y uno para tierra. Como se utilizan 4 motores para mover el robot, se necesitan 8 señales de control. Para reducir la cantidad de pines utilizados se duplicaron en los pines de salida para controlar dos motores por lado simultáneamente. Por último, se integraron conexiones de 5 voltios, de 9 voltios y de 12 voltios para alimentar todos los elementos integrados en la placa.

- Puente H para la garra

El puente H para la garra está implementado con 4 transistores TIP120 y sus diodos de protección. Soporta hasta 1 amperio de corriente, pero consume mucho menos. El bajo consumo de corriente hace innecesario el uso de disipadores de calor.

Figura 22. **Diagrama del puente H para la garra**

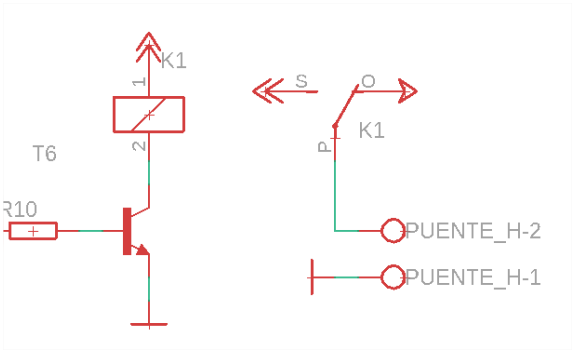


Fuente: elaboración propia

- *Relay* de cambio de velocidad

Un *relay* que conmuta entre dos entradas de voltaje que alimentan el módulo del puente H. El cambio de voltaje en la alimentación provoca una caída en la potencia total entregada a los motores que resulta en el aumento o reducción de la velocidad.

Figura 23. **Diagrama de cambio de velocidad**

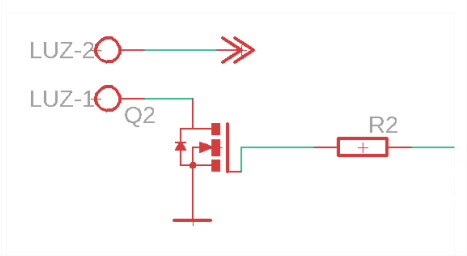


Fuente: elaboración propia

- Apagado y encendido de luces

Implementado con un transistor TIP120, el transistor provee suficiente corriente para encender las luces, y al funcionar como un conmutador enciende o apaga las luces según se necesita

Figura 24. **Diagrama del encendido y apagado de luces**

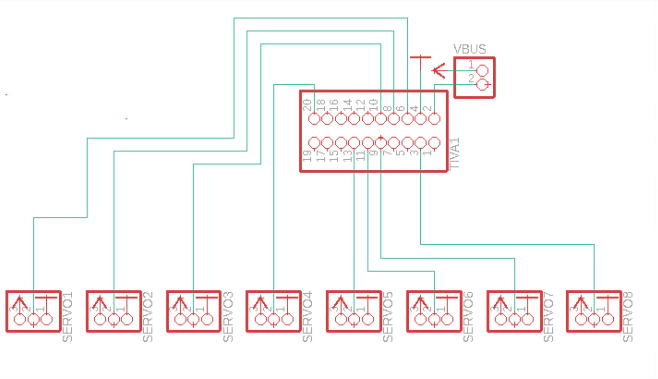


Fuente: elaboración propia

- Conexiones de control y voltaje servomotores y puentes H de las llantas

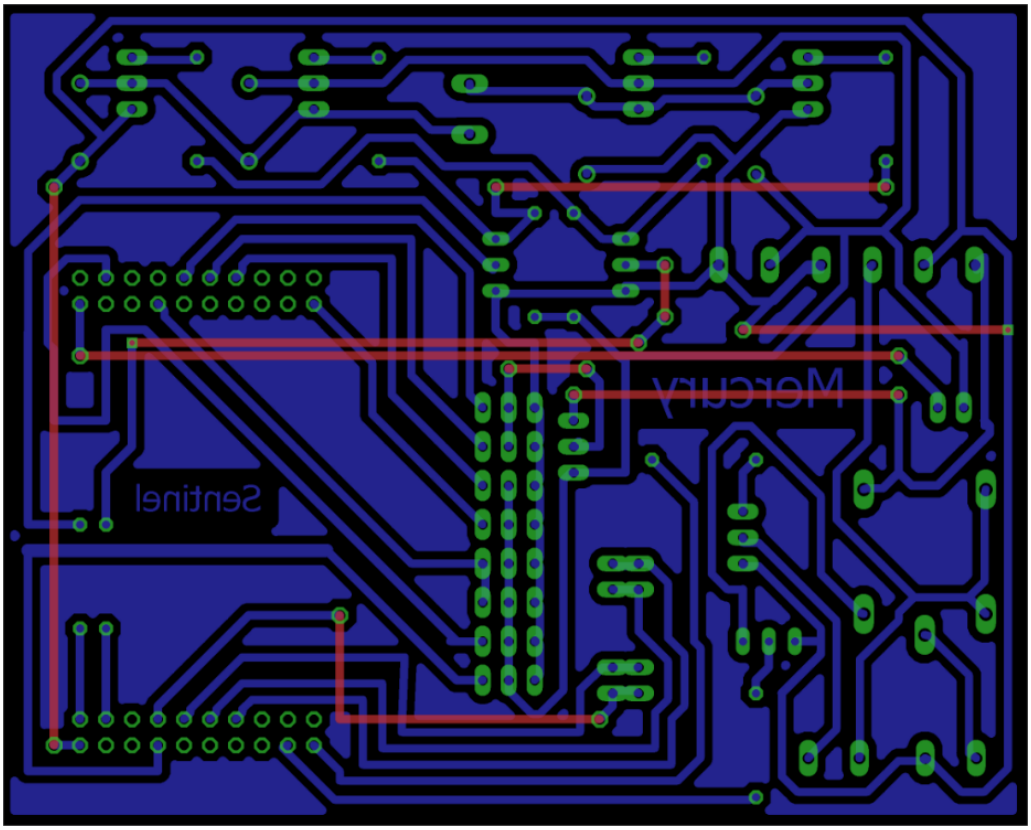
Conecta los pines del microcontrolador con los pines de control del puente H y los servomotores, al mismo tiempo que provee el voltaje según la disposición de los conectores de cada elemento.

Figura 25. **Diagrama de conexión de servomotores**



Fuente: Elaboración propia

Figura 26. **PCB Completa**



Fuente: Elaboración propia

5. TELECOMUNICACIONES

La necesidad de comunicarse instantáneamente desde largas distancias propició a desarrollar las telecomunicaciones. Actualmente, estas juegan un papel importante en el acceso a la información y el conocimiento; tienen además un papel muy importante en las áreas de la economía, las finanzas y la industria. El mundo actual gira alrededor de las telecomunicaciones. Poder manejarlas correctamente y aprovecharlas para el desarrollo de la humanidad es el deber de los Ingenieros.

Una de las áreas más importantes de las telecomunicaciones son las redes de computadoras, que pueden utilizar transmisión por cable, por fibra óptica y transmisión inalámbrica. Su variedad de usos las ha colocado prácticamente en todos lados, y la vida no se concibe sin conexión a la red mundial, el Internet.

La competencia Mercury Robotic Challenge incluye entre su reglamento que el robot debe ser controlado a una distancia no menor de 80 kilómetros, por medio de un enlace punto a punto, a través de internet. La conexión provista fue utilizada para transmitir la información de control, que son las instrucciones del piloto, hacia el robot. De la misma forma, se realizó una transmisión de vídeo en tiempo real. Todo esto se hizo a través de los protocolos de internet TCP/IP.

A continuación, se presentará una breve explicación de las redes, su funcionamiento, y cómo fueron aprovechadas para poder realizar varias conexiones simultáneas para la transmisión de los datos de control y de video, y cómo las características propias de las conexiones pueden ser utilizadas para

encontrar información sobre el estado de la red para generar una interrupción en el programa del robot para que lo detuviera en caso de una desconexión.

5.1. Conceptos de una red de computadoras

Escuchar la palabra red aplicada a telecomunicaciones nos hace pensar en Internet. Una red de computadoras conectadas a través de todo el mundo, que a través de la World Wide Web nos permite ver cientos de páginas de contenido en todo el mundo, como las redes sociales, el correo electrónico, el chat, entre otros. Aunque las redes de computadora se concibieron para conectar computadoras entre sí, actualmente se conectan una gran cantidad de dispositivos como celulares, laptops, tablets, teléfonos, dispositivos de seguridad como el circuito cerrado de televisión y consolas de videojuegos.

Actualmente, los usos de las redes de telecomunicaciones se han ampliado más allá del ámbito empresarial e industrial, ingresando a los hogares por medio de la domótica y la automatización. Las propuestas del Internet de las Cosas han propiciado que termostatos, luces, electrodomésticos y otros aparatos de uso en casa se conecten a la red para el control remoto.

La gran diversidad de aplicaciones ha convertido al Internet en un recurso al que se puede conectar desde cualquier parte del mundo, y sus aplicaciones son incontables. Por tanto, se puede utilizar eficientemente para el control remoto de máquinas como un robot. Utilizando los protocolos adecuados, se puede establecer la conexión del robot utilizado en la competencia, desde cualquier parte del mundo, superando de manera efectiva el límite de al menos 80 kilómetros entre el piloto y el robot.

Se explicarán brevemente algunos de los conceptos básicos de las redes para poder explicar claramente el desarrollo de las técnicas utilizadas para el control remoto del robot, así como la transmisión multimedia a través de internet.

- Red

En esencia, el propósito de una red es hacer conexiones entre equipos, aplicaciones y usuarios[36]. Es como una carretera que conecta ciudades. El estudio de las redes comprende el diseño y construcción de estos caminos, para que el tráfico circule fluidamente por ellos, y en qué puntos deben haber puestos de control para regular el tráfico de la información. Las conexiones que pueden existir pueden ser entre una computadora y una impresora, o una computadora y el internet. Sin embargo, el valor de una red está basado en el tráfico que pasa por ella.

La red también se puede definir como un grupo de computadoras autónomas interconectadas por una misma tecnología. Si dos computadoras pueden intercambiar información entre sí se considera que están conectadas. Las conexiones no deben ser de cable de cobre, pueden ser por fibra óptica, microondas, rayos infrarrojos y comunicaciones satelitales[37].

- Clasificación geográfica de las redes

Existen diversas clasificaciones de las redes. Para el propósito de este documento se clasificarán solamente por la geografía, aunque las redes también pueden clasificarse por su topología y la localización de sus recursos. El criterio de clasificación geográfico para las redes es, qué tan dispersas están en un

[36] BARKER, Keith and WALLACE, Kevin. *CompTIA® Network+ N10-006 Cert Guide*. p. 4
[37] TANENBAUM, Andrew S. and WETHERALL, David J. *Computer Networks*. p. 89

territorio. Una red puede existir en una sola oficina conectando diversos equipos, o puede conectar una base de datos en la oficina central a sus afiliadas en otras partes del mundo.

Basándose en la dispersión geográfica de la red, se puede clasificar en las siguientes categorías:

- *Local Area Network* (LAN): Red de Área Local que interconecta distintos equipos en una misma ubicación geográfica, por ejemplo, un edificio o una casa.
- *Metropolitan Area Network* (MAN): "Una red que conecta varias subredes dentro de una misma ciudad o área metropolitana. Un ejemplo son los edificios de gobierno en una ciudad"[38].
- *Wide Area Network* (WAN): Una red que se extiende a través de un país o un continente.
- *Personal Area Network* (PAN): "Red pequeña de uso personal, con cobertura de unos pocos metros y proporcionada por computadoras portátiles o smartphones, entre otros. Una de sus aplicaciones más importantes son las conexiones bluetooth"[39].

- Componentes de las redes

Todos los dispositivos deben poder conectarse y comunicarse entre sí. Para poder realizarlo de forma efectiva existen ciertos dispositivos especializados y diferentes métodos de transmisión.

[38] BARKER, Keith and WALLACE, Kevin. *CompTIA® Network+ N10-006 Cert Guide*. p. 7-9

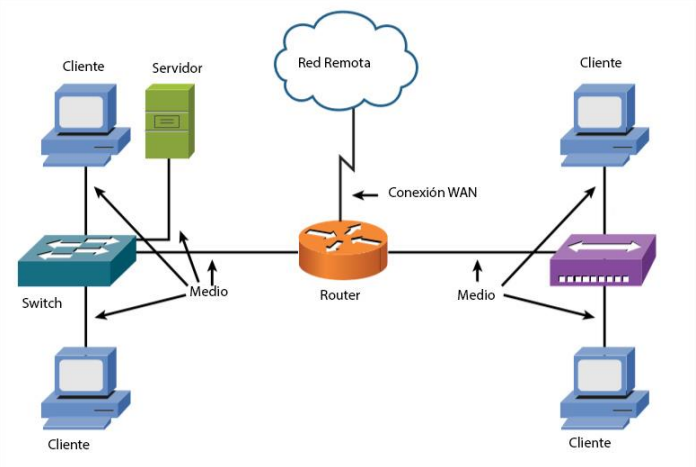
[39] ESCOBAR, Danilo. *Introducción Práctica a las Redes de Computadoras*. p. 4

Cada dispositivo, conocido como cliente, debe conectarse a la red. Para esto necesita una tarjeta de red, o una NIC (*Network Interface Card*). Esta incluye una dirección embebida, única a cada dispositivo, que lo identifica en la red para enviar y recibir paquetes. Para realizar la conexión y la transmisión se necesita de un medio, ya sea un cable de cobre, ondas electromagnéticas, o fibra óptica.

Por el medio de transmisión los equipos se pueden conectar a un concentrador, que permite la comunicación entre dispositivos pertenecientes a la misma red local. El dispositivo más usado para esta tarea actualmente se conoce como *switch*, el cual utiliza las direcciones únicas de los dispositivos, conocidas como direcciones MAC, para redireccionar los paquetes a su destino.

Finalmente, se necesita un dispositivo que nos permita conectarnos a redes externas, como el internet. Este dispositivo debe ser capaz de encontrar una ruta por la que se pueda alcanzar la red externa. El *router* es un dispositivo que se usa como una puerta de enlace entre distintas redes, y todos los dispositivos se pueden comunicar a través de él a redes externas, que se considera como una conexión a una WAN externa.

Figura 27. **Ejemplo de una red**



Fuente: BARKER, Keith. *CompTIA® Network+ N10-006 Cert Guide*. p. 5

5.2. Modelo OSI

En un esfuerzo de estandarizar las comunicaciones entre dispositivos de diferentes tipos, la Organización Internacional para la Estandarización desarrolló un conjunto de protocolos conocido como OSI (*Open Systems Interconections*) o Sistema Abierto de Interconexiones. Su enfoque es la seguridad y escalabilidad de las comunicaciones.

El modelo OSI es un modelo de referencia, basado en siete capas, que busca separar lógicamente las operaciones necesarias para el funcionamiento de una red. Estas capas ayudan a categorizar la función de una tecnología de red en el proceso de comunicación. Basados en cómo funciona una tecnología de red que realiza cierta función en cierta capa del modelo OSI, se puede determinar si un dispositivo puede comunicarse con otro dispositivo que utiliza o no una tecnología similar en esa capa del modelo OSI. Es importante notar que el modelo OSI es un modelo de referencia, que categoriza los distintos protocolos en capas, que no necesariamente se deben categorizar en alguna de las capas existentes, ya que las implementaciones reales pueden variar.

El modelo OSI consta de siete capas, cada una con una función definida y cada una aportando su propia información a la transmisión (encapsulamiento), siendo las mismas presentadas a continuación.

- Capa 7 - Capa de aplicación: Es la capa más familiar al usuario final, provee acceso a la red a las aplicaciones. También anuncia los servicios existentes en la red. Ejemplo: Un navegador web.
- Capa 6 - Capa de presentación: Esta capa convierte la información a manera que pueda ser usada por la aplicación a la que está destinada, dándole formato, y provee servicios de encriptación y desencriptación.

- Capa 5 - Capa de sesión: Maneja las conexiones entre dispositivos y ayuda mantener separados los distintos flujos de información.
- Capa 4 - Capa de transporte: Establece el tipo de comunicación: confiable o no confiable. Y provee una manera de distinguir entre varios procesos utilizando direcciones de transporte conocidas como puertos. Funciona como la unión entre las capas superiores e inferiores del modelo OSI.
- Capa 3 - Capa de red: Provee direccionamiento lógico a través de las direcciones del *Internet Protocol* o direcciones IP.
- Capa 2 - Capa de enlace: Se encarga de empaquetar los datos en *frames* y transmitiéndolas. Realiza detección y corrección de errores, provee direccionamiento físico a través de las direcciones MAC y maneja el flujo de datos.
- Capa 1 - Capa física: Se encarga de la transmisión de datos en bits en la red, así como de las características físicas y eléctricas de la red[40].

5.3. Modelo TCP/IP

El modelo OSI fue diseñado como un modelo de referencia genérico para las comunicaciones, especialmente para las redes, pero puede ser usado para definir otros tipos de comunicaciones digitales. En realidad, la mayoría del tráfico en las redes de computadoras está basada en los protocolos TCP/IP.

El modelo TCP/IP fue desarrollado por Vinton Cerf y Robert Kahn en 1974 para el Departamento de Defensa de los Estados Unidos. El objetivo del gobierno estadounidense era interconectar por medio de una red descentralizada varios sistemas y departamentos de gobierno y universidades, y esa red evolucionó más adelante en lo que hoy se conoce como Internet. Uno de los objetivos principales

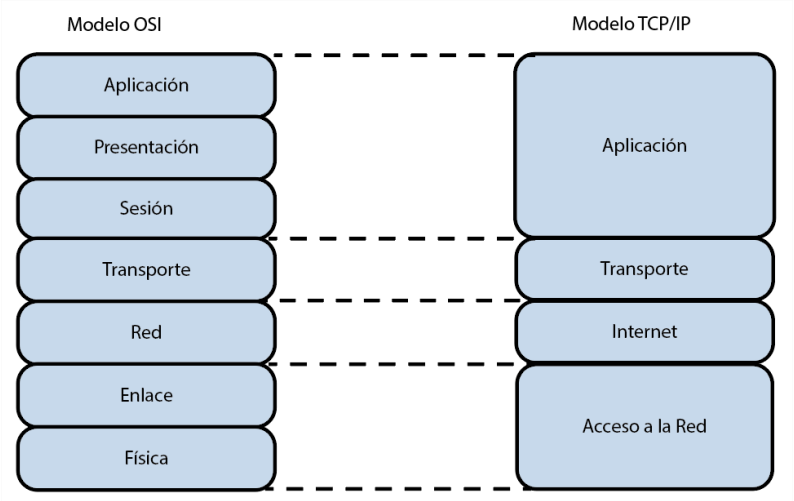
[40] ESCOBAR, Danilo. *Introducción Práctica a las Redes de Computadoras*. p. 7-9

de la red era interconectar redes existentes de forma perfecta y que se pudieran comunicar entre sí. Este modelo toma el nombre de sus dos protocolos más importantes, el *Transmission Control Protocol* (TCP) y el *Internet Protocol* (IP).

El Departamento de Defensa de los Estados Unidos temía que la Unión Soviética pudiera destruir parte de su red en algún ataque. Por tanto, otro de sus objetivos de desarrollo era que la red perdiera parte de su hardware sin que la comunicación se perdiera. Otro de los requerimientos de esta red era un uso muy variado de la red. Desde transferir archivos de texto, a tener transmisión de voz en tiempo real hasta lograr transmisión de video en vivo. La arquitectura flexible detrás de este modelo ha permitido que este se desarrolle para cumplir las grandes demandas de transporte de tráfico de red necesarias en la actualidad.

Distinto al modelo OSI, el modelo TCP/IP se compone solamente de cuatro capas, que engloban todos los protocolos necesarios para cumplir las comunicaciones dentro de una red de computadoras.

Figura 28. **Comparación entre el modelo OSI y el modelo TCP/IP**



Fuente: BARKER, Keith. *CompTIA® Network+ N10-006 Cert Guide*. p. 48

Las capas que componen el modelo TCP/IP son:

- Capa 4 – Capa de aplicación: La capa se encarga de las conexiones, servicios, el formato de los paquetes para que las aplicaciones puedan entenderlos y usarlos y mostrarlo al usuario final para que pueda utilizarlo. Equivale a la capa 5, 6 y 7 del modelo OSI.
- Capa 3 – Capa de transporte: Transforma los mensajes de las capas inferiores a la capa de aplicación. Usualmente utiliza el protocolo TCP para el transporte de paquetes, permitiendo la conexión entre servicios y las aplicaciones disponibles. Equivale a la capa 4 en el modelo OSI.
- Capa 2 – Capa de internet: Se encarga del direccionamiento lógico de los paquetes a través del protocolo IP. Equivalente a la capa 3 en el modelo OSI.
- Capa 1 – Capa de acceso a la red: Provee direccionamiento físico y define las conexiones físicas de la red. Equivalente a la capa 1 y capa 2 del modelo OSI.

El modelo TCP/IP utiliza esencialmente dos protocolos para enrutar todo el tráfico a través de una red. El protocolo TCP y el protocolo IP. Estos protocolos permiten que dispositivos como computadoras, *smartphones* y dispositivos embebidos de cualquier tamaño, de distintos vendedores, con hardware y software muy diferentes, puedan comunicarse unos con otros. Por medio de estos protocolos es posible utilizar un control para manejar un robot en otra ciudad, por medio de la red WAN. En la competencia Mercury Robotic Challenge, se utilizó la red WAN de Colombia para transmitir datos y video, interconectando una computadora personal (que leía el control) usando Ubuntu 16.04 LTS a una Raspberry Pi 3B con Raspbian Stretch, el sistema operativo basado en Debian que utilizan estas computadoras.

Para identificar a cada miembro o *host* en una red, se le asigna una dirección lógica, o dirección IP. Esta dirección es un número de 32 bits en la versión 4 y un número de 128 bits en su versión 6. Cada elemento en la red tiene una dirección diferente, y así los dispositivos encargados de entregar los datagramas de los protocolos pueden encontrar una ruta a través de la red. La dirección IP se compone de dos partes, una que identifica la red y otra que identifica al *host*. El componente de la dirección IP que identifica la parte de red y del *host* se conoce como máscara de subred. Una dirección IP normalmente se escribe en una notación decimal punteada, donde se divide es dividida por puntos en cuatro partes, y cada parte representa ocho bits de los 32 que posee la dirección. Las direcciones IP pueden escribirse como 10.1.0.124, 192.168.125.123, por ejemplo. El libro Introducción práctica a las redes de computadoras, por el Ing. Danilo Escobar, presenta una explicación detallada de las direcciones IP si se desea profundizar más en el tema.

El protocolo TCP se utiliza para transportar la información en la red. TCP es un protocolo orientado a la conexión. La transmisión a través de este protocolo es confiable, porque establece una conexión entre las partes que se comunicarán, utiliza secuencias numeradas para la entrega de los datos, pudiendo detectar pérdidas en la transmisión. Esto lo logra enviando información de sincronización (SYN). También utiliza acuses de recibido (ACK) para asegurar que la transmisión se haya realizado.

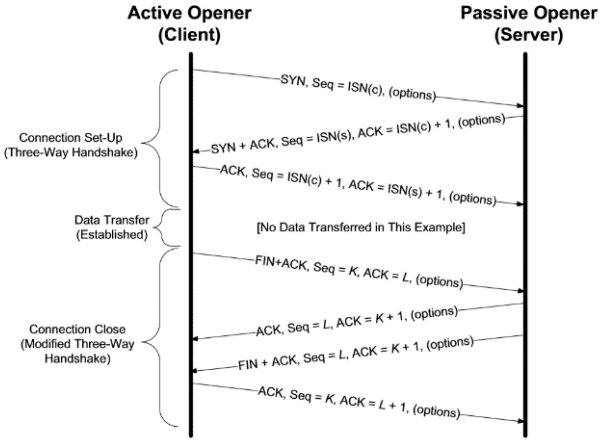
Para establecer una conexión de transmisión confiable TCP utiliza un mecanismo conocido como intercambio de tres vías o 3-*Way handshake*. En un modelo cliente servidor, el cliente envía una solicitud de sincronización (SYN). El servidor responde con su propia información de sincronización (SYN) y un acuse de recibido (ACK). La respuesta del servidor incluye ambos datos en una misma transmisión (SYN+ACK).

Por último, el cliente confirma que recibió la información de sincronización del servidor con un acuse de recibido (ACK). De esta forma se establece una transmisión confiable, ya que ambos lados saben la secuencia de numeración actual de los paquetes transmitidos y pueden dar un acuse de recibido para confirmar la transmisión.

Cuando la conexión está establecida, tanto el cliente como el servidor pueden enviar y recibir datos entre sí, utilizando las instrucciones SEND y RECEIVE del protocolo TCP en una conexión *full duplex*. Ambos lados deben ejecutar la secuencia de terminación para cerrar la conexión.

Para terminar la conexión, se realiza un intercambio de 4 vías, donde cualquiera de los dos lados inicia la secuencia de cierre con un acuse de recibido del último paquete (ACK) y una petición de cierre (FIN). El otro lado contesta con un acuse de recibido (ACK) de la petición de cierre, e inicia su propio proceso de cierre de la conexión. Si el cliente inicia el proceso de cierre, enviará un FIN+ACK y el servidor dará el acuse de recibido (ACK). Después, el servidor enviará su propio FIN + ACK y esperará el acuse de recibido (ACK) del cliente.

Figura 29. **Establecimiento de una conexión TCP**



Fuente: STEVENS, Richard. *TCP/IP Illustrated, Volume 1.* p. 596

5.3.1. Sockets

Uno de los principales objetivos de diseño de la suite de protocolos TCP/IP mencionados anteriormente era interconectar diversas redes, y este también engloba el uso efectivo de las redes por medio de la multiplexación, para que varios servicios puedan ser utilizados al mismo tiempo en la red interconectada.

Un puerto es una conexión lógica que puede ser usada entre programas para intercambiar información directamente[41]. Los procesos de aplicación son identificados por número de puerto. Estos son números enteros positivos de 16 bits. Estos números no se relacionan con ningún elemento físico de la red. En cambio, cada dirección IP tiene 65,536 números de puerto asociados para cada protocolo de transporte, y se utilizan para determinar la aplicación correcta que recibe o envía datos[42]. Existe un puerto fuente, que identifica el proceso que envía los datos; y existe un puerto de destino, que identifica el proceso que espera recibir los datos.

Cuando los datos transitan por las capas TCP/IP, se necesita un mecanismo que los lleve al protocolo correcto en la capa correspondiente. El sistema debe ser capaz de combinar datos de diferentes fuentes en unos pocos protocolos de transporte, y de estos protocolos al protocolo de internet (IP). Tomar datos de diferentes fuentes en una sola fuente de datos se conoce como multiplexación. Pero los datos deben ser demultiplexados para que puedan ser leídos por los diferentes procesos que esperan recibirlos. El protocolo TCP/IP utiliza números de protocolo para identificar los protocolos de transporte y los números de puerto para identificar las aplicaciones[43].

[41] ESCOBAR, Danilo. *Introducción Práctica a las Redes de Computadoras*. p. 25
[42] STEVENS, Richard. *TCP/IP Illustrated, Volume 1*. p. 17-19
[43] Hunt, Craig. *TCP/IP Network Administration*. p. 45

Como se explicó anteriormente, cada *host* dentro de una red debe tener un identificador único. Por medio de las direcciones IP se puede direccionar un paquete de datos del transmisor al destinatario en cualquier parte de la red. Para que los datos puedan ser recibidos por la aplicación correcta, se identifica esta con un número de puerto. La combinación de una dirección IP con un número de puerto es conocida como *socket*. Esta combinación permita la comunicación simultánea de varios procesos en la misma red. Una pareja de *sockets*, uno para el *host* que envía datos y uno para el *host* que los recibe, es el fundamento para la conexión para protocolos de transporte que están orientados a la conexión, como el TCP.

5.3.2. NAT

Para que una red funcione correctamente, cada *host* o dispositivo conectado a ella debe tener un identificador único para que los paquetes puedan ser direccionados de manera adecuada y que lleguen a su destino. Este identificador es la dirección IP. En los años 90, el internet creció de una forma explosiva. Una de las principales preocupaciones en ese entonces era que el número de direcciones disponibles se agotaran. Para paliar este problema se crearon una serie de soluciones que permitió que se administrara de mejor manera la cantidad de direcciones disponibles.

Una de las soluciones más importantes fue que cada organización que contaba con una red interna pudiera reutilizar un grupo de direcciones reservadas para identificar *hosts* internos. Esto ayudó a ralentizar el agotamiento de las direcciones. Sin embargo, como cualquier organización podría utilizar estas direcciones, ya no son únicas, y no pueden enrutarse en una red más grande. Este grupo de direcciones se conocieron como direcciones IP privadas. Se hizo

necesario un método para que las redes privadas pudieran comunicarse con la red pública, compuesta por las direcciones IP no reservadas.

Con este fin se creó la traducción de direcciones de red, o *Network Address Translation (NAT)*. El NAT permite que las redes con direcciones IP privadas puedan comunicarse con la red pública. Una de sus grandes ventajas es que puede conectar muchos equipos utilizando pocas direcciones públicas.

Normalmente se definen 3 tipos de traducción de direcciones, la traducción estática, la traducción dinámica y la sobrecargada. La traducción sobrecargada es la más utilizada. Este tipo de traducción aprovecha los puertos disponibles en cada dirección IP para traducir las distintas direcciones privadas a una sola dirección pública. Esto permite que varios dispositivos puedan compartir una sola dirección pública, y se conoce como *Port Address Translation* (PAT)[44].

El reglamento de la competencia Mercury Robotic Challenge especifica que se proveerá una conexión wifi que asignará una dirección IP al dispositivo, en el caso de esta implementación, a la Raspeberry Pi. Se tendrán también tres puertos disponibles para los distintos servicios que sean útiles para el control remoto del robot. Estos serán traducidos a la red pública por medio de NAT por medio de un enlace dedicado de 10 *Mbps*. Por tanto, se debe desarrollar el sistema de control utilizando no más de 3 puertos, y con un protocolo de transporte que no sea afectado por el NAT.

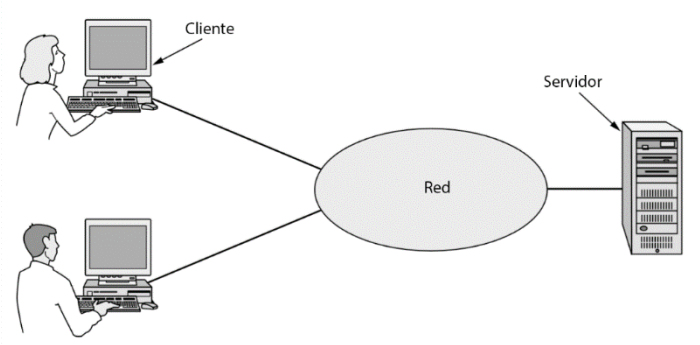
5.4. Modelo cliente servidor

En una red, se da el caso que existe un recurso que está en una sola computadora, y se necesita acceder a él desde otro equipo. Puede ser un servicio

[44] ESCOBAR, Danilo. *Introducción Práctica a las Redes de Computadoras*. p. 197

o acceso a archivos de una base de datos en una empresa, o incluso otro equipo como una impresora. La mayoría de las aplicaciones de red están creadas en un modelo cliente servidor. En este modelo, el servidor provee un tipo de servicio a los clientes. Usualmente, los servidores están alojados en potentes computadoras, pero en realidad el modelo es independiente de los equipos que lo conforman. Este modelo se puede aplicar tanto si los equipos están en el mismo lugar geográfico como si están en distintos lugares con grandes distancias de por medio.

Figura 30. **Modelo cliente servidor**



Fuente: TANENBAUM, Andrew S. and WETHERALL, David J. *Computer Networks.* p. 4

Si se analiza el modelo cliente servidor a detalle, se puede determinar fácilmente que se compone de dos programas que funcionan separadamente y que se ven envueltos en el proceso. Uno es del cliente, y otro es del servidor. Para establecer la comunicación, el cliente realiza una petición al servidor a través de la red y queda en espera de una respuesta del servidor. Cuando el servidor recibe la petición, la procesa y envía una respuesta[45].

[45] TANENBAUM, Andrew S. and WETHERALL, David J. *Computer Networks*. p. 4

Para el caso específico del robot explorador tratado en este trabajo, el robot es controlado de forma remota a través de la red. Tiene sentido entonces que posea una forma de transmisión de imágenes o video para que el piloto pueda controlarlo. Aprovechando el hardware integrado de la Raspberry Pi se le conectó una cámara en el puerto dedicado que esta posee para periféricos de video.

La cámara y la comunicación serial por hardware que tiene la Raspberry con el microcontrolador que controla el hardware del robot son dos elementos muy importantes para el manejo del robot. Ya que el reglamento de la competencia exige que el robot no sea manipulado más allá del encendido y la configuración inicial, se puede elegir a la Raspberry dentro del robot como el servidor del modelo, que provee los servicios de transmisión de video en tiempo real y la capacidad de comunicación para recibir las instrucciones del piloto y transmitirlas a la Tiva C, que controla el hardware del robot. El cliente entonces es la computadora que utiliza el piloto para leer el control y recibir la transmisión de video. Por eso se dice que en el modelo cliente servidor estos nombres se refieren a las aplicaciones y funciones que tiene cada equipo y no a los sistemas de computadoras en los que funcionan.

Otra de las reglas establecidas en la competencia es que el robot, una vez configurado y conectado a la red, y con una conexión establecida con el cliente sea capaz de reestablecer automáticamente la conexión en caso de una falla en la red que ocasione una desconexión. Para poder entender la implementación, es necesario conocer cómo se establecen las conexiones TCP, y cuáles son los mecanismos para poder reestablecer la comunicación en caso de pérdida.

Una conexión TCP/IP está definida por dos direcciones IP y dos números de puerto, es decir, dos *sockets*. Para la primera conexión, el servidor crea un *socket* a través de un puerto y crea un punto de comunicación. Este *socket* no

tiene ninguna dirección asociada, para no generar errores con otro servicio que ya tenga una dirección dedicada. Después, enlaza una dirección IP al puerto y se pone en modo de escucha. Esto se conoce como Abierto Pasivo o *Passive Open*. En este modo el servidor esperará una solicitud de conexión por parte del cliente. Después, el servidor puede aceptar o denegar las solicitudes, y dependiendo de cómo esté programado, puede generar nuevas instancias de su proceso o servicio para atender las solicitudes del cliente[46].

El cliente, por su parte, también crea un *socket* en un puerto que no tiene asociado ninguna dirección IP. En el proceso del cliente, no es necesario enlazar una dirección al *socket*, porque al servidor no le interesa conocer la dirección del cliente para prestar el servicio. La petición del cliente proveerá los datos de direccionamiento para que el ruteo de información sea exitoso y la información se transmita a donde corresponda. Por último, el cliente inicia el proceso de conexión activamente. Esto se conoce como *Active Open* o Abierto Activo[47].

5.4.1. Implementación del modelo cliente servidor

Como se explicó anteriormente en el capítulo 3, el software del robot fue desarrollado en Python 2.7. Mientras que el capítulo 2 se dedicó al programa que controla las funciones de control del robot, este se centra en el área de telecomunicaciones y cómo se estableció el modelo cliente servidor.

Python 2.7 cuenta con un módulo de interfaz de red de bajo nivel. Provee acceso a la interfaz de *sockets* BSD, implementada en los sistemas operativos de más grande alcance, como sistemas UNIX, LINUX, Windows y MacOS. La función *socket()* de Python devuelve un objeto cuyos métodos implementan las

[46] STEVENS, Richard. *TCP/IP Illustrated, Volume 1*. p. 59-60
[47] Ibíd. p. 595

diversas llamadas de sistema de los *sockets*. En los ejemplos de código, la numeración de las líneas se corresponde con el código fuente. El código para implementar el servidor es el siguiente:

```
131     HOST = ' '
132     PORT = 5000
133
134     SocketServer = socket.socket()
135     SocketServer.setsockopt(socket.SOL_SOCKET,
        ______socket.SO_REUSEADDR, 1)
136     SocketServer.bind((HOST, PORT))
137     SocketServer.listen(1)
138     connection, address = SocketServer.accept()
```

El primer paso es crear el *socket*; esto se hace en la línea 134. La línea 135 indica algunas opciones para el *socket*. En el protocolo TCP, cuando se cierra una conexión, el puerto entra en un estado conocido como TIME_WAIT[48]. En este estado el *socket* espera un tiempo antes que el puerto pueda ser reutilizado. Sin embargo, para la competencia, es necesario que el robot pueda reestablecer la conexión de la forma más rápida posible, ya que solo se cuenta con 3 de ellos, que se están utilizando simultáneamente en otras tareas.

La interfaz de *sockets* BSD define la bandera SO_REUSEADDR como una instrucción que indica al sistema operativo que el puerto puede ser reutilizado sin esperar el tiempo de espera establecido. Con esta opción se puede anular el contador y realizar la reconexión. En las aplicaciones normales de red, los clientes usan puertos aleatorios para generar conexiones con el servidor, por lo que el tiempo de espera en el estado TIME_WAIT no es un problema, pero las reglas de la competencia establecen que solamente se pueden utilizar tres

[48] STEVENS, Richard. *TCP/IP Illustrated, Volume 1*. p. 620

puertos por equipo, por lo que es necesaria una forma para poder reutilizar el *socket* inmediatamente, que se realiza por medio de esta bandera[49].

Como se explicó anteriormente, al crear un *socket*, se define un puerto y después se le asocia una dirección IP. En la línea 131 y 132, se crean dos variables con los valores de la IP y el puerto para el *socket* del servidor. La instrucción HOST = ‘ ‘ indica que se usa la dirección IP que existe en la interfaz de red del sistema operativo. Esto se realiza así porque la asignación de direcciones IP es por medio de un servidor DHCP, y no se conoce la dirección asignada. Si se asigna una en el programa y existe otra en la interfaz de red el programa no funcionará.

La línea 136 asocia el puerto establecido anteriormente con la dirección IP en la interfaz de red, y la línea 137 pone el servidor en estado de escucha. Cuando llega una solicitud del cliente, el servidor procesará la petición y la aceptará.

Del lado del cliente, se establece una dirección IP y un puerto para crear un *socket*. La dirección IP que se utiliza es la dirección del servidor hacia el cual se quiere dirigir la solicitud. Esto se realiza en las líneas 8 y 9 del siguiente código. Los números de línea son los números en el código fuente del programa, y se han eliminado pasos intermedios para colocar solo las líneas de código referentes al establecimiento del modelo cliente servidor.

[49] Python Software Foundation. *17.2. socket — Low-level networking interface*. The Python Standard Library: Documentation. https://docs.python.org/2/library/socket.html. Consulta: marzo de 2019

```
8       HOST = '192.168.0.100'
9       PORT = 5000
227
228     socketClient = socket.socket()
229     socketClient.connect((HOST, PORT))
242     socketClient.settimeout(0.5)
243     try:
244         socketClient.send(InstructionChain)
245     except socket.timeout:
246         socketClient.close()
```

En la línea 228 se crea el *socket* para la conexión del cliente. En la línea 229 se asocia el *socket* del cliente con el *socket* del servidor para realizar la solicitud de conexión. La línea 242 establece un *timeout*. Al finalizar el tiempo establecido en esta instrucción (medido en segundos), el *socket* levantará una excepción, indicando que la instrucción siguiente no se pudo realizar. Si el programa no logra enviar datos, cerrará su conexión e iniciará una nueva solicitud de conexión al servidor. El programa del cliente está en la computadora del piloto, y lee las instrucciones del control y las envía al servidor a intervalos regulares de tiempo para que este los procese.

La instrucción *send* en la línea 244 envía la cadena de caracteres de 10 bits generadas por el proceso del programa que lee el control. Si en 0.5 segundos del envío de los datos no se recibe el acuse de recibido de parte del servidor, se asume que la conexión se perdió, y se cierra el *socket*. Esto se realiza las líneas 245 y 246. Si el envío es exitoso, se lee el control nuevamente y se continúa el flujo del programa.

Para detectar la pérdida de conexión, se debe entender los tipos de estados de conexión de un *socket* en el protocolo TCP. El primer estado es el abierto. En este estado tanto el cliente como el servidor están disponibles para realizar una conexión. El segundo estado es el cerrado, cuando alguna de las partes indica a la otra que terminará la transmisión de datos. Estos estados se habían

comentado anteriormente. Existe un tercer estado que se conoce como medio cerrado, o *half-closed*. En este estado, una de las partes indica a la otra que ya no enviará datos, pero que quiere recibirlos. Entonces una de las partes en la conexión enviará datos y la otra ya no, y la comunicación activa será unidireccional[50].

El último estado que puede existir se conoce como medio abierto, o *half-open*. Este estado ocurre cuando una de las partes se desconecta de forma inesperada, ya sea porque perdió la energía o algún elemento sin redundancia en la red se perdió. En este caso, una de las partes no sabe que la conexión se ha perdido, he intentará transmitir o recibir datos en una conexión que ya no existe[51].

Para encontrar las desconexiones, en el lado del cliente se estableció un *timeout*. Este cierra el *socket* al terminar el tiempo establecido y no recibir el acuse de recibido por parte del servidor. Inmediatamente, crea el *socket* de nuevo e intenta crear una nueva conexión.

En el servidor, se utiliza el mismo mecanismo. Como el cliente envía datos en intervalos regulares de tiempo, se establece un *timeout* 4 veces más grande del tiempo que espera el cliente antes de enviar nuevos datos. De esta manera se compensan retrasos en la red y se asegura la transmisión si la red está congestionada. Como se indicó anteriormente, se estableció la opción SO_REUSEADDR para que el sistema operativo pudiera reestablecer la conexión lo más pronto posible.

[50] STEVENS, Richard. *TCP/IP Illustrated, Volume 1*. p. 598
[51] Ibídem. p. 628

5.5. Transmisión de datos

Después que se estableció la conexión entre el cliente y el servidor, se pueden enviar los datos. En la sección anterior se explica brevemente el uso de las instrucciones *send* y *recv* para el envío de información de datos.

El programa que lee las instrucciones del control las codifica como una cadena de caracteres de 10 *bytes*. Con 10 letras se representa el estado del control cuando el programa leyó las instrucciones. Estos datos se almacenan en una variable que el cliente envía al servidor para su proceso. A través del método *socket.send*(), Python envía la instrucción al otro lado del *socket*. Como Python es un lenguaje de alto nivel, este envía de forma implícita la longitud del paquete de envío.

Al enviar datos, el protocolo TCP intentará enviar la mayor cantidad de información en una transmisión. Utilizando los acuses de recibido, determinará la cantidad de paquetes que pueden enviarse al mismo tiempo sin generar errores. Esto se conoce como ventana deslizante, y es un parámetro que el protocolo ajusta automáticamente según las condiciones de red. La posibilidad de enviar paquetes sin errores reduce las posibilidades de retransmisión, mejorando el aprovechamiento de la red[52].

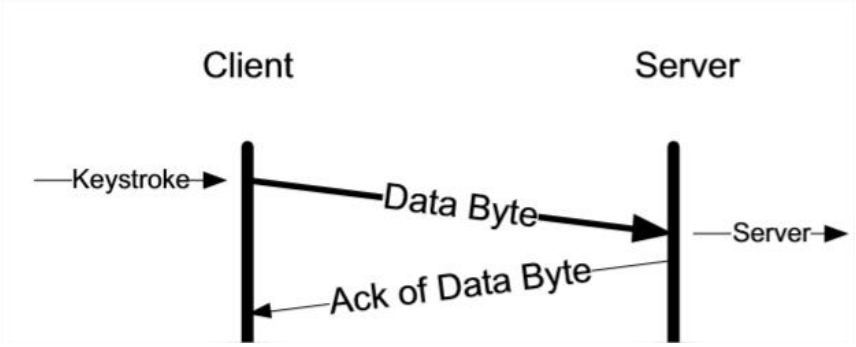
TCP tiene mecanismos de control que posibilitan la transmisión de grandes cantidades de datos de forma eficiente. Sin embargo, también puede transmitir pequeñas cantidades de datos. Una conexión TCP interactiva es una en la que se envían pequeños mensajes, tales como pulsaciones del teclado, movimientos del mouse o lecturas de un control. A las conexiones interactivas se les puede añadir control de flujo y procesos de administración asociados a la ventana

[52] ESCOBAR, Danilo. *Introducción Práctica a las Redes de Computadoras*. p. 24

deslizante en la transmisión, que permiten la transmisión de paquetes de gran tamaño. Al utilizar mensajes cortos, la información asociada al protocolo de transporte puede generar retrasos debido a la cantidad de datos asociados a un paquete. Por otra parte, si se espera a crear una cantidad de información más grande para aprovechar la transmisión, los paquetes pueden retrasarse, lo que en aplicaciones dependientes del tiempo se pueden generar retrasos.

Dado que en la aplicación para el control del robot se utilizan únicamente 10 *bytes* para el control, se puede decir que solo es necesaria una conexión interactiva, que intercambia datos de manera rápida. El control en tiempo real de un robot necesita actualizaciones frecuentes, por lo que el envío de paquetes con poca información es más efectivo. El proceso de envío de datos es decidido automáticamente por el protocolo TCP, por lo que si se envían los 10 *bytes* en un solo mensaje o se envían uno por uno depende del protocolo, y la implementación en Python no permite control sobre esto.

Figura 31. **Proceso de envío de paquetes por TCP**



Fuente: STEVENS, Richard. *TCP/IP Illustrated, Volume 1.* p. 693

La instrucción *socket.recv*() realiza la reservación de memoria para el *buffer* de entrada de forma automática. El protocolo TCP puede enviar uno o varios *bytes* a la vez. Por esta razón, se limitó el tamaño del buffer de entrada para que

fuera únicamente de 10 *bytes*. Al no realizar esta limitación, se obtenían instrucciones de más de 10 *bytes*, que podrían generar errores, o la pérdida de estas al ser procesadas en otro paquete de datos. Al realizar la limitación del buffer por un tamaño preestablecido, se logró una recepción sin errores en la cantidad de datos, sin importar el estado de la red, y se comprobó en distintas redes, con diferentes cantidades de tráfico y usuarios conectados, y el sistema funcionó sin errores.

5.6. Transmisión multimedia

Las redes fueron creadas inicialmente para la transferencia de datos. Se utilizaban para control remoto de equipos con terminales virtuales como *switches* o *routers*, y la administración de servidores. También para acceder a archivos en servidores FTP, o para la comunicación entre trabajadores en las empresas, con servicios de correo electrónico o servicios de mensajería instantánea[53].

Las redes que anteriormente utilizaban solo texto para transmitir toda la información ahora cuentan con la posibilidad de transmitir audio y video, entre otras cosas. Algunas de las aplicaciones más grandes que se derivaron de este avance en las redes es la telefonía sobre IP (*Voice-over-IP*, VoIP), y las video conferencias. Esta clase de servicios multimedia, y aplicaciones de tiempo crítico necesitan una red que esté disponible siempre, que tenga suficiente ancho de banda y que administre de la forma más eficiente el ancho de banda disponible en un momento dado. Al contrario, algunas aplicaciones de red, como la *World Wide Web* puede tener pequeñas interrupciones, y las páginas tener un pequeño retraso en cargar, sin superar los límites admisibles, sin que sea un problema mayor.

[53] TANENBAUM, Andrew S. and WETHERALL, David J. *Computer Networks*. p. 4

La competencia Mercury Robotic Challenge exige que el control del robot sea realizado a una distancia mínima de 80 kilómetros. Para poder hacer un control eficiente se necesita una imagen visual, provista por una cámara, y una forma de transmisión de datos. La transmisión de datos fue discutida en la sección anterior. Para la transmisión de video, no es necesario tener una transmisión de datos. En el capítulo 3, dedicado al diseño e implementación del controlador del robot se explica que se escogió la cámara de la Raspberry Pi para realizar el *streaming* en tiempo real de la imagen para uso del conductor.

La transmisión de video, o información multimedia en general, es distinta al tráfico normal de datos en una red. Si el tráfico multimedia es en tiempo real, como en este caso, la situación se complica, porque los datos tienen que ser transmitidos a un ritmo predeterminado para que sea útil. Existen utilidades para la optimización de la red que permiten la optimización de la transmisión de paquetes de alta prioridad como QoS (*Quality of Service* o Calidad de Servicio). Estas utilidades están implementadas en *routers* y *switches* en la red[54]. Sin embargo, en la competencia no se tiene acceso a la configuración de la red, por lo que utilizar herramientas como estas queda descartado.

Se debe tener en cuenta también que la transmisión de datos para el control del robot también es en tiempo real, entonces se debe considerar el ancho de banda del canal para adecuar las especificaciones de la transmisión de video y de datos para que ambos paquetes de información lleguen a su destino sin causar interferencias uno con el otro.

[54] BARKER, Keith and WALLACE, Kevin. *CompTIA® Network+ N10-006 Cert Guide*. p. 304

5.7. Streaming de video

El video, al contrario de otro tipo de datos, requiere una gran cantidad de ancho de banda. El ancho de banda se define como la capacidad de transmitir cierta cantidad de bits en un segundo. Por ejemplo, si una red puede transmitir 10 millones de bits en un segundo, se dice que su ancho de banda es 10 *Mbps* (megabits por segundo).

La representación digital de un video puede describirse como una secuencia de cuadros, o *frames*. Cada una está compuesta por una cuadrícula rectangular de elementos de imagen conocidos como pixeles. Para representar los colores, se utilizan componentes de color rojo, azul y verde, conocidos como RGB. Esta representación es posible porque a partir de la superposición de la luz de estos colores se puede generar cualquier color si se varía la adecuadamente la intensidad de cada uno de ellos. La mayoría de los sistemas usan 8 bits para representar cada uno de los componentes RGB, resultando en un sistema con 24 bits por píxel, con posibilidad de generar más de 16 millones de colores, más de los que el ojo humano puede detectar.

Si los cuadros varían a una velocidad adecuada, el ojo deja de percibir que está viendo imágenes fijas, y tiene la impresión de movimiento. Algunas velocidades comunes de cuadros por segundo son 24, 30 y 60 cuadros por segundo. La cantidad de bits por píxel, la cantidad de cuadros por segundo y la cantidad de pixeles en una imagen determinan el ancho de banda que necesita una señal de video para ser transmitida. Por ejemplo, un video con un cuadro de 640 por 480 pixeles, con 24 bits de información por color y 30 cuadros por segundo consume más de 200 *Mbps* de ancho de banda. Esto excede la capacidad de la mayoría de las conexiones en oficinas y casas.

Para enfrentar este problema se han producido técnicas de compresión que permiten que la transmisión de video a través de la red sea factible[55].

La infraestructura de las redes actuales, especialmente las redes WAN, permite un ancho de banda de varios *Gigabits* por segundo (*Gbps*). Los equipos que se ofrecen para los usuarios finales también han aumentado su capacidad de transmisión. Este aumento en capacidad, sumado a las técnicas de compresión hacen posible que la transmisión de video sea factible. Las redes tienen suficiente ancho de banda para transmitir video comprimido, por lo que las transmisiones de video pueden verse afectadas más por retrasos en la transmisión que por falta de ancho de banda.

La Raspberry Pi cuenta con un módulo de cámara que tiene capacidad de grabar video de alta definición en una resolución hasta 1080p a 30 cuadros por segundo. La Raspberry Pi tiene el procesador de gráficos VideoCore IV, que le permite codificar con hardware el video. La herramienta para capturar video desde la línea de comandos es *raspivid*. Esta herramienta captura video como un *raw video stream* en h.264, un *codec* de video con alta compresión y calidad. Esto permite que la calidad no se pierda en la transmisión y que el video no ocupe todo el ancho de banda disponible en el enlace.

Además de las consideraciones sobre la codificación y la compresión del video, se debe tomar en cuenta el formato, ya que no todos los dispositivos son capaces de decodificar h.264 sin un contenedor de formato adecuado como MP4. Buscando una solución a este problema, se optó por utilizar Mplayer, un reproductor multimedia *open-source*, disponible para distribuciones Linux, macOS y Windows, que puede leer el formato h.264 sin necesidad de un contenedor de formato.

[55] TANENBAUM, Andrew S. and WETHERALL, David J. *Computer Networks*. p. 697-698

El reproductor Mplayer se instaló en la computadora cliente, y el servidor de video utilizó *raspivid* para utilizar la cámara en la Raspberry y Netcat para escribir el contenido a un puerto TCP.

Netcat es una utilidad de red que permite leer y escribir datos de un puerto TCP. Permite establecer conexiones en un modelo cliente servidor, utilizando *sockets*. Permite también otros protocolos de transporte como UDP. La herramienta *raspivid* obtiene la imagen de la cámara y procede a escribirla al *socket* establecido. Para obtener una transmisión de video en tiempo real, se utilizó una resolución baja, de 640 por 360 pixeles, con una velocidad de 20 cuadros por segundo.

Esta resolución, acoplada a la codificación h.264, permitió un video en tiempo real con poca latencia y sin generar interrupción a la transmisión de datos de control. La transmisión de video y datos con poca latencia permitieron un control casi en tiempo real, a pesar de que el día de la competencia el robot se encontraba en Bogotá, y el piloto en Medellín. Se debe tomar en cuenta que la computadora que utilizó el piloto no tenía una pantalla de alta resolución, lo que permitió utilizar una baja resolución de video en la transmisión sin tener un efecto adverso.

Para crear un *streaming* a través de Netcat se debe tomar el *feed* de video de la cámara utilizando *raspivid*, utilizando los parámetros encontrados en la documentación de *raspivid* por la *Raspberry Foundation*[56]. Como parámetro de la dirección de escritura se coloca el puerto al que se quiere escribir. Al unir todos los parámetros se obtiene la siguiente instrucción:

[56] Raspberry Pi Foundation. *Raspberry Pi Camera Module*. Raspberry Pi Documentation. https://www.raspberrypi.org/documentation/raspbian/applications/camera.md. Consulta: marzo de 2019

/opt/vc/bin/raspivid -t 0 -w 640 -h 360 -br 60 -fps 20 -o - | nc -l 8080

Tabla III. **Parámetros para toma de video**

Parámetro	Función
raspivid	API para tomar video
-t 0	Tiempo de toma de video (0 es sin límite)
-w 640	Ancho de video 640 pixeles
-h 360	Alto de video 360 pixeles
-br 60	Brillo al 60%
-fps 20	Tomar video a 20 frames por segundo
-o - | nc -l 8080	Salida al puerto 8080 de Netcat

Fuente: elaboración propia

Para ver el *streaming*, se utiliza Mplayer, poniendo como parámetro de reproducción el puerto de transmisión usando Netcat, indicando como parámetros la dirección IP del servidor y el puerto de transmisión, así como el formato del video, en este caso, h264. La instrucción queda de la siguiente manera:

nc 192.168.0.100 -p 8080 | mplayer -fps 200 -demuxer h264es-

Tabla IV. **Parámetros para reproducción de video**

nc 192.168.0.100	Conectar a Netcat en esa IP
-p	Conectar a puerto Netcat 8080
| mplayer	Salida del puerto abrirla con Mplayer
-fps 200	Reproducir a 200 fps
-demuxer h264es-	Enviar a decodificador de video con h.264

Fuente: elaboración propia

CONCLUSIONES

1. Se diseñó e implementó un robot que contaba con varios servidores para la transmisión de datos por internet, permitiéndole recibir instrucciones enviadas desde una computadora, para usarse en la competencia Mercury Robotic Challenge Latinoamérica, en Bogotá, Colombia.

2. El cableado del robot se facilitó considerablemente al diseñar y construir una placa de circuito impreso que contaba con conectores estándar para algunos tipos de motores e integraba elementos de potencia y control para manejar los actuadores del robot.

3. Al dividir la tarea de decodificación y ejecución entre dos computadoras diferentes se logró que el tiempo entre la llegada de una instrucción al robot y la ejecución de esta fuera mínima, permitiendo un control casi en tiempo real.

4. Se construyeron dos brazos robóticos de dos grados de libertad, uno para una cámara y otro para recoger, transportar y entregar la carga provista en la competencia. Ambos brazos podían ser fácilmente controlados por el conductor del robot independientemente del movimiento del resto del cuerpo del robot.

5. El día de la competencia y los días de prueba previos a la competencia se consiguió transmitir datos de video de 640x480 pixeles de resolución, codificados en h.264, con una latencia de entre 3 ms y 5 ms, suficiente para conducir el robot en tiempo real.

RECOMENDACIONES

1. Para la transmisión de video, se pueden lograr mejoras en la transmisión en malas condiciones de conexión utilizando el protocolo UDP, que descarta los *frames* que no han sido transmitidos con éxito, evitando problemas de video retrasado y retransmitido por el protocolo TCP.

2. Integrar todos los elementos de potencia y control de los actuadores del robot en una sola placa de circuito impreso, utilizando la menor cantidad de módulos externos a utilizar. Si el diseño se realiza correctamente, se elimina parte del cableado necesario, se tienen conexiones más confiables y de mejor calidad y más facilidad de corrección de errores en las conexiones cableadas.

3. Al utilizar el esquema cliente-servidor para servicios de comunicación por internet para control del robot y recepción de información deben ser alojados en el robot utilizándolo como servidor. Ya que estos servicios pueden automatizarse para activarse en el encendido del robot, las solicitudes de conexión se realizan desde el cliente, en control del piloto, lo que reduce el tiempo de preparación previo a una competencia.

BIBLIOGRAFÍA

1. ARORA, Mohit. *Embedded system desing.* 1a ed. USA : Learning Bytes Publishing, 2016. 213 p.

2. BARKER, Keith y WALLACE, Kevin. *CompTIA® Network+ N10-006 Cert Guide.* 1a ed. USA : Pearson Certification, 2015. 668 p.

3. BREY, Barry B. *Intel Microprocessors. Architecture, programming and interfacing.* 8va ed. USA : Pearson Prentice Hall, 2009. 944 p.

4. BROWN, Jim. *Brief H-bridge theory of operation*. Dallas Personal Robotics Group*.* [en línea] <http://stan.cropsci.uiuc.edu/people/LSY_teaching/Fall2008/bioengin2008/Top/Manuals/Motors/DC_MotorDrivers/DPRGBriefH-BridgeTheoryofOperation.html> [Consulta: marzo de 2019.]

5. CHEN, Crystal; NOVICK, Greg y SHIMANO, Kirk. *RISC VS. CISC*. *RISC architecture.* [en línea] <https://cs.stanford.edu/people/eroberts/courses/soco/projects/risc/risccisc/index.html> [Consulta: enero de 2019.]

6. COOMBS, Clyde. *Printed Circuits Handbook.* 7a ed. USA : McGraw-Hill, 2016. 1606 p.

7. Corporación Unificada Nacional de Eduación Superior. *Reglamento competencia Mercury 2018 .* Bogotá : CUN, 2018.

8. ESCOBAR, Danilo. *Introducción práctica a las redes de computadoras. Guatemala* : s.n. 216 p.

9. GLASKOWSKY, Peter. *AMD's SSE5 ends the old RISC vs. CISC debate.* CNET News. [en línea] <https://www.cnet.com/news/amds-sse5-ends-the-old-risc-vs-cisc-debate/> [Consulta: enero de 2019.]

10. GONZÁLEZ, Victor, LÓPEZ CRUZADO, Antonio y CABERO, José Ángel. Control y Robótica. *5.1 ¿Qué es un robot?* [en línea] <http://platea.pntic.mec.es/vgonzale/cyr_0708/archivos/_15/Tema_5.1.htm> [Consulta: enero de 2019.]

11. GROOVER, Mikel P., y otros. *Robótica industrial.* 1a ed. México D.F. : McGraw-Hill, 1989. 546 p.

12. GUIZZO, Erico. *Learn. Everything you need to get started in robotics*. [en línea] <https://robots.ieee.org/learn/> [Consulta: diciembre de 2018.]

13. GUTIÉRREZ OROZCO, Jorge Alejandro. *Máquinas de estado finitos*. [en línea] <http://uncomp.uwe.ac.uk/genaro/Papers/Veranos_McIntosh_files/alejandroFinal2008.pdf> [Consulta: febrero de 2019.]

14. HALL, Jon. *The continuing drama of RISC vs. CISC*. [en línea] <http://www.linux-magazine.com/Issues/2016/190/Doghouse-RISC-vs.-CISC [Consulta: enero de 2019.]

15. HEATH, Steve. *Embedded systems design.* 2a ed. Burlington : Elsiever Science, 2003. 451 p.

16. HUNT, Craig. *TCP/IP Network administration.* 3a ed. USA : O'Reilly Media, Inc, 2002. 748 p.

17. KELLER, Robert M. *Computer science: abstraction to implementation*. *12. Finite-State Machines.* [en línea] <https://www.cs.hmc.edu/~keller/cs60book/%20%20%20Title.pdf>. [Consulta: marzo de 2019.]

18. New World Encyclopedia contributors. *Printed circuit board*. New World Encyclopedia*.* [en línea] <http://www.newworldencyclopedia.org/p/index.php?title=Printed_circuit_board&oldid=988465> [Consulta: marzo de 2019.]

19. Python Software Foundation. *17.2. socket — Low-level networking interface*. The Python Standard Library: Documentation*.* [en línea] <https://docs.python.org/2/library/socket.html> [Consulta: marzo de 2019.]

20. Raspberry Pi Foundation*. Raspberry Pi Camera Module*. *Raspberry Pi Documentation.* [en línea] <https://www.raspberrypi.org/documentation/raspbian/applications/camera.md> [Consulta: marzo de 2019.]

21. Real Academia Española. *Diccionario de la Lengua Española.* [en línea] <http://dle.rae.es/?id=A4hIGQC> [Consulta: enero de 2019.]

22. REYES, Fernando. *Robótica. Control de robots manipuladores.* 1a ed. México : Alfaomega Grupo Editor, S.A. de C.V., 2011. 590 p.

23. RICHLING, Jan y BUSSE, Anselm. *The ARM architecture – yesterday, today, and tomorrow*. Linux Pro Magazine. [En línea] <http://www.linux-magazine.com/Issues/2013/156/ARM-Architecture> [Consulta: enero de 2019.]

24. SPONG, Mark W., HUTCHINSON, Seth y VIDYASAGAR, M. *Robot modeling and control.* 1a ed. USA : John Wiley & Sons, Inc., 2005. 419 p.

25. STEVENS, Richard. *TCP/IP illustrated, Volume 1.* 2a ed. USA : Pearson Education, 2012. 1059 p.

26. TANENBAUM, Andrew S. y WETHERALL, David J. *Computer networks.* 5a ed. USA : Prentice Hall, 2011. 962 p.

27. TOKHEIM, Roger L. *Fundamentos de los microprocesadores.* 2a ed. USA : McGraw-Hill, 1992. 417 p.

28. VALVANO, Jonathan. *Embedded systems: introduction to ARM Cortex-M microcontrollers*. 5a ed. USA : s.n., 2014. 594 p.