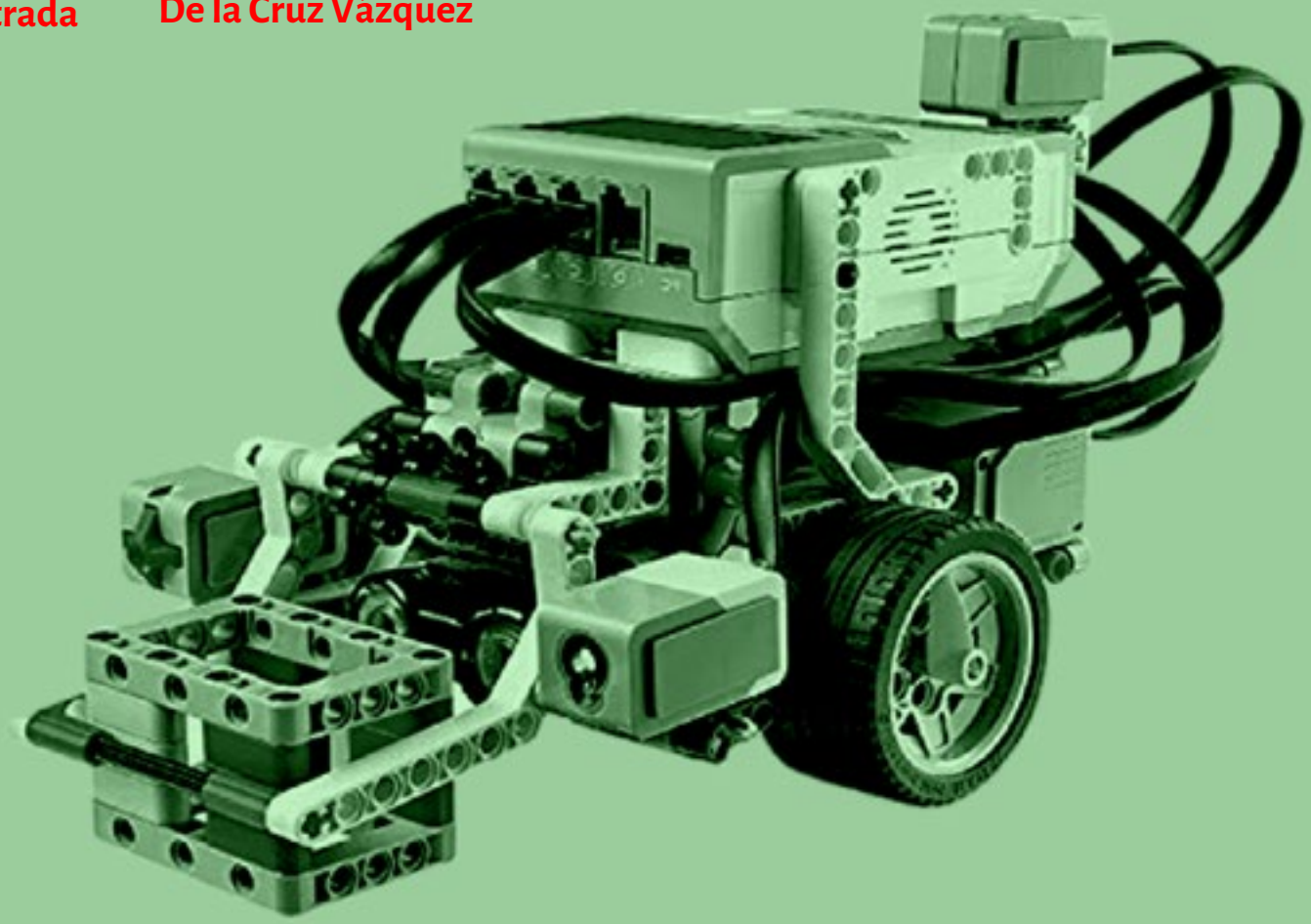
INTRODUCCIÓN A LA ROBÓTICA

Luis Antonio **Álvarez Oval**

Christian Mauricio **Castillo Estrada**

Aron **De la Cruz Vázquez**



CONTENIDO GENERAL

INTRODUCCIÓN A LA ROBÓTICA

Luis Antonio **Álvarez Oval**

Christian Mauricio **Castillo Estrada**

Aron **De la Cruz Vázquez**

UNACH
FACULTAD DE NEGOCIOS, Campus IV,
Grupo de investigación:
Tecnologías en Sistemas Computacionales

LUIS ANTONIO ÁLVAREZ OVAL
Profesor de tiempo completo, investigador y especialista en Robótica educativa, Cómputo en la nube, Diseño y administración de base de datos y Programación de procedimientos almacenados, Big data y Ciencia de datos
loval@unach.mx

CHRISTIAN MAURICIO CASTILLO ESTRADA
Profesor de tiempo completo, investigador y especialista en Ingeniería de software (Arquitectura REST y microservicios), Ingeniería web para el desarrollo de aplicaciones empresariales de tipo web y móviles; Big data y Ciencia de datos.
cmce@unach.mx

ARON DE LA CRUZ VÁZQUEZ
Profesor de tiempo completo, investigador y especialista en Teoría matemática para la computación, Algebra lineal y Estadística para el diseño de modelos de análisis de datos.
aron.cruz@unach.mx

Palabras clave: iniciación a la robótica, programación básica de robots: tareas, parametros, comportamientos, funciones, variables, bucles; sensores infrorrojo y ultrasónico.

Keywords: introduction to robotics, basic robot programming: tasks, parameters, behaviors, functions, variables, loop; infrared and ultrasonic sensors.

RESUMEN

ABSTRACT

Introducción a la robótica, de Luis Antonio Álvarez Oval, Christian Mauricio Castillo Estrada y Aron De la Cruz Vázquez es una obra editada por la Universidad Autónoma de Chiapas, que fue dictaminada por el método de par ciego externo.

ISBN Colección Sin fronteras: 978-607-561-118-1
ISBN Volumen: 978-607-561-130-3

Dirección editorial: Luis Adrián Maza Trujillo
Corrección de estilo, diseño y edición: José Urióstegui

1ª. edición 2022.

Boulevard Belisario Domínguez km 1081, sin número, Terán, C. P. 29050, Tuxtla Gutiérrez, Chiapas.

La Universidad Autónoma de Chiapas es miembro de la Cámara Nacional de la Industria Editorial con el número de afiliación 3932.

Editada en México / *Edited in Mexico*

IMPORTANTE:

ROBOTC® es una marca registrada de Robomatter, Inc.
LEGO® y MINDSTORMS® son marcas registradas de LEGO Juris A/S Corporation.

LEGO Educación y el grupo LEGO no patrocinan, respaldan ni apoyan este trabajo, su edición o difusión.

CONTENIDO GENERAL

Luis Antonio Álvarez Oval, Christian Mauricio Castillo Estrada y Aron de la Cruz Vázquez, *Introducción a la robótica*, Sin fronteras, núm. 3, UNACH, México, 2022, 180 pp. ISBN: 978-607-561-130-3

www.unach.mx

Disponible en: www.editorial.unach.mx

Luis Antonio Álvarez Oval, Christian Mauricio Castillo Estrada y Aron de la Cruz Vázquez, *Introducción a la robótica*, Sin fronteras, núm. 3, UNACH, México, 2022, 180 pp. ISBN: 978-607-561-130-3

www.unach.mx

Disponible en: www.editorial.unach.mx

Luis Antonio Álvarez Oval, Christian Mauricio Castillo Estrada y Aron de la Cruz Vázquez, *Introducción a la robótica*, Sin fronteras, núm. 3, UNACH, México, 2022, 180 pp. ISBN: 978-607-561-130-3

www.unach.mx

Disponible en: www.editorial.unach.mx

RESUMEN

Definir el comportamiento de un robot educativo a través del diseño de algoritmos y su codificación con el lenguaje de programación denominado ROBOTC es el ámbito de esta obra. Desde el control del movimiento y detección de variables del medioambiente usando los sensores que nos proporciona el robot se le presentarán al lector de manera amena. Manos a la obra.

ABSTRACT

Defining the behavior of an educational robot through the design of algorithms and their coding with the programming language called ROBOTC is the scope of this work. From the control of movement and detection of environmental variables using the sensors provided by the robot will be presented to the reader in an entertaining way. Let's do it.

Luis Antonio Álvarez Oval, Christian Mauricio Castillo Estrada y Aron de la Cruz Vázquez, *Introducción a la robótica*, Sin fronteras, núm. 3, UNACH, México, 2022, 180 pp. ISBN: 978-607-561-130-3

www.unach.mx

Disponible en: www.editorial.unach.mx

Breves currículos de los autores

M. C. LUIS ANTONIO ÁLVAREZ OVAL. Ingeniero en Computación por la Universidad Autónoma de Guadalajara; posee el grado de Maestro en Ciencias de la Computación por la Universidad de Houston Campus Clear Lake. Actualmente colabora en la Universidad Autónoma de Chiapas como profesor de tiempo completo en la Facultad de Negocios, Campus IV. Sus áreas de especialidad e investigación son: Robótica educativa, Cómputo en la nube, Diseño y administración de bases de datos y Programación de procedimientos almacenados, Big data y Ciencia de datos. Es integrante del grupo de investigación denominado Tecnologías en Sistemas Computacionales. Posee el reconocimiento como profesor de tiempo completo con Perfil Deseable vigente otorgado de la Secretaría de Educación Pública.

DR. CHRISTAN MAURICIO CASTILLO ESTRADA. Es Licenciado en Sistemas Computacionales por la Universidad Autónoma de Chiapas; Maestro en Comercio Electrónico con especialidad en Tecnologías de Información por el Instituto Tecnológico y de Estudios Superiores de Monterrey, y Doctor en Gestión para el Desarrollo por la Universidad Autónoma de Chiapas. Actualmente se desempeña como profesor de tiempo completo en la Facultad de Negocios, Campus IV, de la Universidad Autónoma de Chiapas. Es reconocido como profesor de tiempo completo con Perfil Deseable vigente otorgado por el PRODEP Tipo Superior de la SEP, y como integrante del Sistema Estatal de Investigadores (SEI) que otorga el Instituto de Ciencia, Tecnología e Innovación del Estado de Chiapas. Sus áreas de especialización e investigación son: Ingeniería de software (Arquitectura REST y microservicios), Ingeniería web para el desarrollo de aplicaciones empresariales de tipo web y móviles; Big data y Ciencia de datos. Integra el grupo de investigación Tecnologías en Sistemas Computacionales.

DR. ARON DE LA CRUZ VÁZQUEZ. Se desempeña como profesor de tiempo completo en la Facultad de Negocios, Campus IV, de la Universidad Autónoma de Chiapas; posee los grados de Maestro en Ciencias de la Computación con especialidad en Redes, Maestro en Administración con Formación en Organizaciones, y el Doctorado en Gestión para el Desarrollo por la Universidad Autónoma de Chiapas. También es reconocido como integrante del Sistema Estatal de Investigadores (SEI) del Instituto de Ciencia, Tecnología e Innovación del Estado de Chiapas. Sus áreas de especialización e investigación son: Teoría matemática para la computación, Algebra lineal y Estadística para el diseño de modelos de análisis de datos. Es integrante del Grupo de Investigación denominado Tecnologías en Sistemas Computacionales.

PRÓLOGO

En la actualidad los robots representan un avance tecnológico relevante; en la sociedad han logrado captar la atención cuando se han aplicado para facilitar tareas de la vida cotidiana, como localización de personas, exploración de volcanes, vigilancia y telemedicina, entre muchas otras, convirtiéndose en herramientas complementarias para las personas. Siguiendo lo anterior, en el ámbito de la educación los robots también juegan un papel importante para desarrollar en los estudiantes competencias, desde genéricas hasta profesionales, a lo largo de sus trayectorias escolares, despertándose en ellos la creatividad, razonamiento y lógica; en otras palabras, la robótica coadyuva al desarrollo cognitivo de los jóvenes.

Es evidente el crecimiento acelerado de la robótica en los últimos años, ampliando su campo de aplicación, derivado de los avances significativos en computación, electrónica, internet de las cosas y software especializado; el *internet de las cosas* ha permitido ampliar el campo de aplicación de los robots; estamos en la antesala de una era en la cual

Luis Antonio Álvarez Oval, Christian Mauricio Castillo Estrada y Aron de la Cruz Vázquez, *Introducción a la robótica*, Sin fronteras, núm. 3, UNACH, México, 2022, 180 pp. ISBN: 978-607-561-130-3

www.unach.mx

Disponible en: www.editorial.unach.mx

las personas convivirán todos los días con robots en sus labores o actividades, esto les permitirá elevar su productividad y facilitar tareas. Por esa razón resulta importante aprender su funcionamiento y programación de los mismos.

La división temática de esta obra comprende la aplicación de conceptos básicos de programación para manipular el comportamiento de un robot educativo denominado LEGO® Mindstorms EV3 a través del diseño de algoritmos y codificación de los mismos mediante el lenguaje de programación (ROBOTC®) iniciando con el control de movimientos básicos del robot y toma de decisiones hasta desarrollar rutinas avanzadas de monitoreo usando una webcam móvil.

Por lo anterior, se ha incluido en cada tema una sección de ejercicios enfocada a resolver un reto distinto porque cambia su grado de complejidad conforme el lector avanza. Cabe señalar que esta publicación digital ha sido elaborado con la finalidad de ser un recurso educativo para fortalecer el proceso de enseñanza de las unidades de competencia de metodología de la programación y programación estructurada. Los diversos desafíos que se presentan en cada uno de los laboratorios, son el resultado de experiencias educativas generadas en ciclos escolares anteriores, y toman como referencia los recursos de capacitación y manuales de usuario que proporcionan la empresa Robomatter Inc. y la Universidad Carnegie Mellon, respetando en todo momento sus derechos de autor.

Finalmente, manifestamos que este compendio de ejercicios tiene el objetivo es comprobar la correcta adquisición de conocimientos de robótica y su aplicación a través del diseño de un algoritmo propio por parte de los estudiantes.

INTRODUCCIÓN GENERAL

Desde los inicios del programa educativo de Licenciatura en Sistemas Computacionales, que se ofertaba en la Facultad de Negocios del Campus IV de la Universidad Autónoma de Chiapas, se identificaba en el proceso de enseñanza-aprendizaje la necesidad de utilizar herramientas educativas complementarias, para coadyuvar al reto que representa para los estudiantes el aprendizaje y aplicación de los principios de la programación estructura y metodología de programación, muchas generaciones alertaban de diferentes problemas, pero una constante aparece siempre: el razonamiento de pasar de las nociones abstractas de la computadora a la implementación práctica. Limitar esta brecha entre teoría y práctica resulta una decisión vital para controlar el arte de programar computadoras.

Derivado de la incorporación de un nuevo programa educativo a la oferta académica de la Facultad de Negocios, nuevos retos surgen y la necesidad de dar solución a cada uno de ellos es prioritario. Los laboratorios de prácticas académicas que se han elaborado para la unidad de competencia denominada Metodología de la Programación, la cual forma parte de la Licenciatura en Ingeniería en Desarrollo y Tecnologías de Software tratan de resolver la problemática mencionada; estos laboratorios de prácticas consideran como elemento educativo innovador, el uso de robots LEGO EV3 programables, permitiendo la aplicación del Razonamiento Computacional como una técnica que se requiere para resolver problemas en las diferentes etapas de formación de los estudiantes. Cada

Luis Antonio Álvarez Oval, Christian Mauricio Castillo Estrada y Aron de la Cruz Vázquez, *Introducción a la robótica*, Sin fronteras, núm. 3, UNACH, México, 2022, 180 pp. ISBN: 978-607-561-130-3

www.unach.mx

Disponible en: www.editorial.unach.mx

una de las prácticas plasmadas en el laboratorio, inician con la descripción de un problema o reto, posteriormente se representa de manera gráfica la solución a través de Diagramas de Flujo, y finalmente se escribe un programa en un lenguaje de programación denominado ROBOTC®, el cual es cargado o transferido al robot LEGO® para su correspondiente ejecución. Lo anterior permite una sinergia de la teoría y la práctica como una herramienta para desarrollar proyectos de programación; el estudiante no es un simple observador, sino todo lo contrario, es el responsable de diseñar el algoritmo correcto y codificarlo para que el robot pueda cumplir con el reto propuesto en cada una de las prácticas que conforman al laboratorio.

Por todo lo anterior, en este libro didáctico se contempla el uso del lenguaje de programación ROBOTC®, en virtud de que provee un entorno de desarrollo fácil para fortalecer el aprendizaje del estudiante, y que pueda aplicar los fundamentos de programación, asociándolos con el lenguaje de programación C, el cual es usado en los semestres subsecuentes en otras unidades competencias tales como: Estructura de Datos, Sistemas Operativos, Bases de Datos, Diseño de interfaces, etcétera. En la actualidad, existe un número reducido de libros relacionados con el lenguaje ROBOTC® en idioma español; no obstante, en inglés existe una vasta documentación generada por la Universidad Carnegie Mellon. Estos laboratorios se basan en los materiales educativos y manuales de usuario, sin ser su copia, a los que se puede acceder mediante los sitios web oficiales señalados en las secciones de bibliografía de este libro. Todos los programas aquí presentados han sido debidamente probados para asegurar la calidad del trabajo. Debido a que no existe bibliografía en idioma español, hemos buscado cómo estos materiales pueden formar la base de este libro.

En estos laboratorios algunos desafíos de programación se han tomado de los publicados en el sitio web de ROBOTC® y LEGO®, otros son parte de la experiencia de programación de computadoras de los autores y la necesidad de explicar a los estudiantes el plan de estudios de la asignatura y todos se han resuelto con programas desarrollados por los autores.

También es necesario comentar que este material ha sido probado con estudiantes de la Escuela Secundaria General "16 de Septiembre", en el ejido La Libertad, municipio de Ciudad

Hidalgo, Chiapas. También se ha usado para capacitar estudiantes del CONALEP plantel Tuxtla Chico. En ambos casos tuvimos excelentes resultados. Es nuestra intención reproducir este curso fuera de las aulas universitarias para la mayor cantidad de educandos posible.

Además se explotará la amplia infraestructura robótica que posee la facultad. En su inventario cuenta con robots de las marcas VEX®, LEGO® y ARDUINO®; todos se pueden programar con el mismo lenguaje de programación ROBOTC®. También contamos con licencias de ROBOTC® para VEX® y LEGO®, y licencias para ARDUINO®, que son gratuitas. Esta estrategia permite preparar a nuestros estudiantes para que programen una amplia variedad de robots, participen en las competencias que organizan los fabricantes de los dispositivos y los prepara para enfrentar los retos de programación que presenta el programa académico que se oferta.

Finalmente, esperamos que los materiales preparados para los estudiantes universitarios puedan permear a estudiantes de nivel medio superior, y que nuevos retos se implementen para enriquecer la asignatura, como la programación de otros dispositivos robotizados aéreos, mejor conocidos como drones. ¡Manos a la obra!

ROBOT VELADOR, LABORATORIO 1

COMPORTAMIENTO BÁSICO: MARCHA

- **RECORRIDO RECTILÍNEO**
- **RECORRIDO CON VUELTAS Y GIROS**

OBJETIVOS DEL DESAFÍO QUE SE TE PLANTEA:

→ Desarrollar el pensamiento lógico necesario para analizar el comportamiento básico del robot en cuanto a velocidad y dirección.

→ Diseñar un algoritmo y construir un programa de computadora apropiado con lenguaje de programación ROBOTC®.

→ Implementar ese algoritmo en un robot Lego® y probar que el robot ejecuta las instrucciones adecuadas.

Luis Antonio Álvarez Oval, Christian Mauricio Castillo Estrada y Aron de la Cruz Vázquez, *Introducción a la robótica*, Sin fronteras, núm. 3, UNACH, México, 2022, 180 pp. ISBN: 978-607-561-130-3

www.unach.mx

Disponible en: www.editorial.unach.mx

INTRODUCCIÓN

De acuerdo al sitio de documentación oficial de Robomatter Inc. La tecnología ROBOTC® es considerada como un lenguaje de programación eficaz con base en los principios y sintaxis del lenguaje de programación C. Esta tecnología ofrece un entorno de programación integrado (IDE, por sus siglas en inglés) orientado al sistema operativo Windows utilizado para escribir y depurar códigos.

Así también, es importante señalar que ROBOTC es un lenguaje de programación que expone un depurador con búsqueda paso a paso en tiempo real. Es una solución multiplataforma para que los estudiantes tengan acceso al aprendizaje de robótica básica aplicando los principios de la programación estructurada (Robomatter 2016).

Cabe señalar que el estudiante debe consultar la lista de comandos que se visualizan en el IDE, escribirlos en pantalla, posteriormente serán procesados por el compilador traduciéndose a un lenguaje máquina. Finalmente, estos comandos ya traducidos serán cargados en el robot para ser ejecutados.

ROBOTC® y el robot nos permiten trasladar la programación del mundo abstracto dentro de la computadora al mundo real, en donde cada orden que se escribe puede ser percibida como la actividad que desarrolla el robot. El estudiante debe comprender la programación de computadoras como una actividad lógica sencilla de entender y aplicar, esa comprensión le permitirá desarrollar conceptos mucho más abstractos tales como la comunicación entre procesos remotos, programación orientada a objetos, programación web, etcétera.

El robot que usaremos a lo largo de este curso tiene una configuración de dos ruedas, cada una con su propio motor, que serán los actuadores, y una rueda de arrastre omnidireccional; en el tema de sensores, para percibir su ambiente, el robot dispone de un sensor táctil, ultrasónico, y otro de luz/color.

DESAFÍO POR RESOLVER

La compañía ACME Robotics ha ganado un proyecto para diseñar un robot programable que patrulle los alrededores de un edificio rectangular que almacena autos, por lo que ACME te ha contratado para desarrollar el programa para que ese robot prototipo pueda, de forma autónoma, "vigilar" los alrededores del edificio. Otro equipo de trabajo ya ha desarrollado el prototipo mecánico-electrónico del robot y está disponible para su uso. El proceso de ingeniería implica que la solución se vaya construyendo paso a paso, así que la rutina de vigilancia constante será desarrollada en laboratorios posteriores.

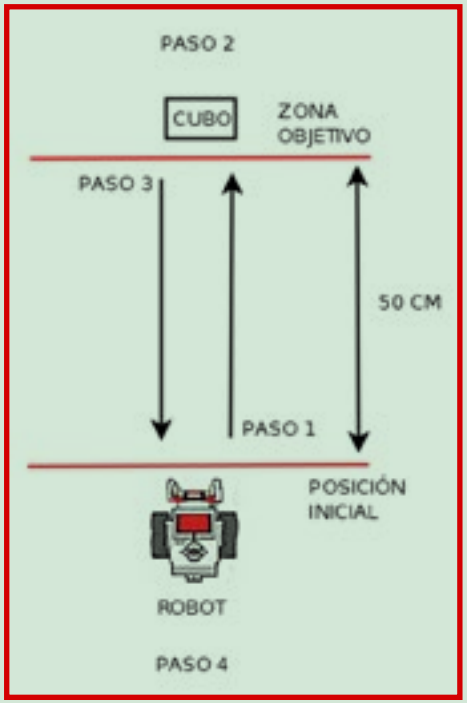
Es necesario que imprimas este documento ya que debes responder las preguntas de las secciones "Solución del desafío".

El administrador de este proyecto ha dividido en seis etapas el proceso de ingeniería para el desarrollo del sistema que va a controlar el robot y que resolverá la problemática planteada por el cliente.

La problemática que se debe resolver en esta sección es la siguiente: el robot debe arrancar de su posición inicial hasta la zona objetivo, ahí debe recoger, con la mandíbula frontal, el cubo que se te proporciona, y debe regresar a su posición inicial en reversa. Finalmente, ya en la posición inicial, debe depositar el cubo en el piso. Esto nos permitirá ganar comprensión del comportamiento de la mecánica del robot.

Figura 1.

RECORRIDO DEL ROBOT VELADOR.



Para este desafío el estudiante necesita los siguientes materiales:

- Cinta aislante de color negro
- Flexómetro
- Transportador
- Lenguaje de programación ROBOTC®
- Robot LEGO® armado.
- Editor de diagramas DIA. Liga de descarga: http://dia-installer.de/

CONCEPTOS BÁSICOS

Aquí encontrarás material de apoyo para entender el objetivo primario de este documento. En general, los apuntes provienen de los documentos que el Laboratorio Nacional de Ingeniería Robótica de la Universidad Carnegie Mellon ha proporcionado en los cursos de robótica que imparte y en la página web del lenguaje ROBOTC® (Carnegie Mellon University 2020).

Diagramas de flujo

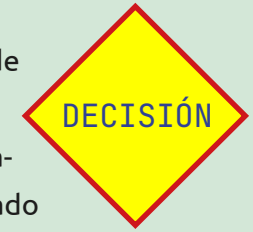
Los diagramas de flujo son un conjunto de símbolos gráficos que representan un algoritmo, por tal motivo, representan de manera gráfica a una secuencia de pasos para solucionar un problema. Estos diagramas comúnmente están integrados por una combinación de símbolos estandarizados (bloques y flechas). Los bloques típicamente exponen procesos, es decir, tareas específicas; y el orden o secuencia de las acciones se representan utilizando las flechas. En este apartado es necesario manifestar la posibilidad que de un mismo bloque se deriven diferentes flechas de dirección donde cada una de ellas representa una opción o ruta. La notación que se usa para construir un diagrama de flujo se detalla a continuación (Tabla 1).

Comportamientos

Un comportamiento es cualquier cosa que un robot hace; tan solo encender un motor es un comportamiento, moverse hacia delante, seguir una línea y navegar un laberinto son, cada uno, comportamientos. Hay tres principales tipos de comportamientos: básicos, simples y complejos. La figura 2 muestra los tres tipos de comportamiento.

Tabla 1.

PRINCIPALES SÍMBOLOS DE UN DIAGRAMA DE FLUJO.

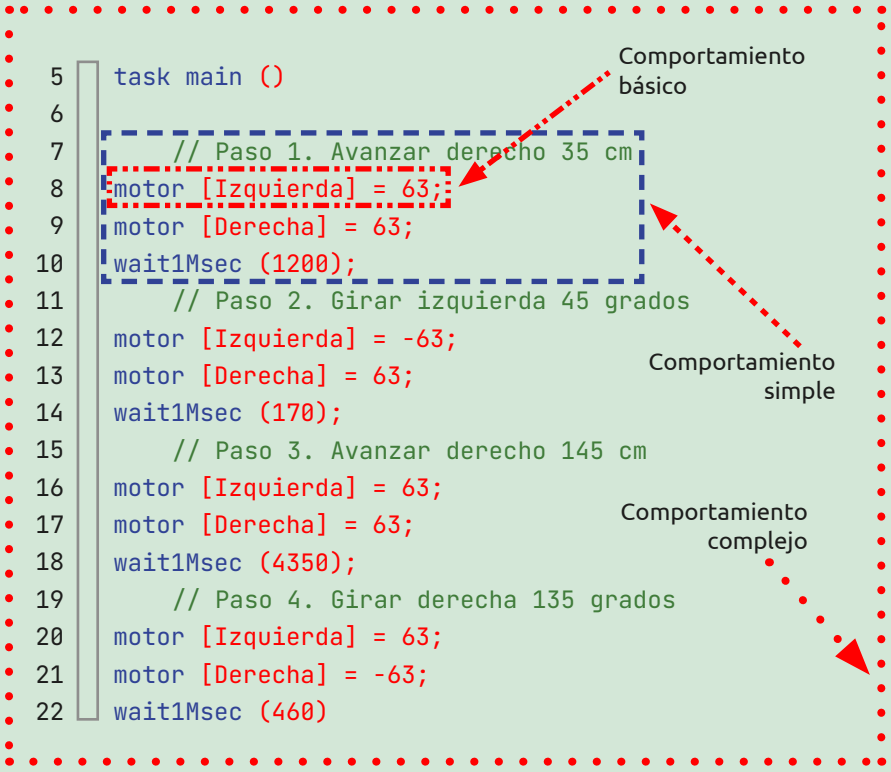
Inicio y fin. El primer y último paso de cada algoritmo, suelen indicarse visualmente a través de rectángulos redondeados, que contienen las palabras `inicio` o `fin`.	
Acciones. Las acciones son representadas con rectángulos y actúan como comandos básicos: `“wait1Msec (1000)”`; `“increment line count by 1”` o `“motor full ahead”`.	
Decisiones. Los bloques de decisiones son representados mediante la figura de rombos; estos bloques suelen ser una estructura selectiva doble, es decir, evalúan el valor de una variable y el resultado se traduce en una respuesta `verdadero/falso` o `sí/no`; con base a esa respuesta se toma la decisión de elegir una u otra ruta en el flujo de un programa.	

Comportamientos básicos

En el nivel más básico, todo en un programa tiene que ser dividido en pequeños comportamientos que tu robot pueda comprender y realizar directamente. En ROBOTC®, estos son los comportamientos de un solo enunciado, como encender un solo motor o reajustar un cronómetro.

Figura 2.

LOS TRES TIPOS DE COMPORTAMIENTO DE UN ROBOT.



Comportamientos simples

Estos comportamientos son pequeños, su tamaño aproximado es un bit, y permiten a tu robot realizar una simple pero significativa tarea, como moverse hacia delante por un tiempo. Son, quizá, los comportamientos más útiles en los que debes pensar, porque son suficientemente grandes para describir acciones útiles, y suficientemente pequeños para programarlos fácilmente con órdenes básicas de ROBOTC®.

Comportamientos complejos

Estos son comportamientos en los niveles más altos, como navegar todo un laberinto. Pueden parecer complejos, pero una buena propiedad de los comportamientos complejos es que están compuestos de otros más pequeños. Si los observas con detalle, verás que puedes dividirlos en comportamientos más y más pequeños y, con práctica, llegarás a reconocer todos.

Cuerpo de un programa en ROBOTC®

La ejecución de un programa en ROBOTC® inicia en la línea identificada por el comando "`task main`", por omisión el compilador lo buscará y si no lo encuentra marcará `error`. Después ejecuta el siguiente comando hacia abajo y así sucesivamente. En el programa que se muestra en la figura 3 puedes ver dos comandos: `motor` y `wait1msec`.

Comando motor

El comando `motor` indica al robot encender un motor a un determinado nivel de potencia. El ejemplo del programa 1 configura los puertos B y C, a los que están conectados los motores izquierdo y derecho, respectivamente, para que corran a toda potencia hacia delante. Recuerda que cada comando en ROBOTC® termina con punto y coma (`;`).

Comando wait1sec()

El comando `wait1sec()` le indica al robot cuánto esperar antes de que ejecute el siguiente comando, por un tiempo dado en milisegundos. El número entre paréntesis representa los milisegundos que el robot tendrá que esperar para ejecutar el siguiente comando.

Figura 3.

CUERPO DE UN PROGRAMA EN ROBOTC®.

`Task main`. Esta línea crea una tarea llamada `task main`, la cual contiene los comportamientos que queremos que el robot realice. `Task main` marca el inicio de una estructura.

```
4
5   task main()
6   {
7
8   // Paso 2. Girar izquierda
9   motor[Izquierda] = 0;
10  motor[Derecha]   = 63;
11  wait1Msec (170);
12
13  }
14
```

{Cuerpo} los corchetes [ ] encierran el cuerpo de la estructura. Entre los corchetes se indica al programa qué comandos seguir como parte de la `task main`.

A continuación se muestra un resumen del comportamiento que se puede lograr variando la potencia que se aplica a cada motor. En cada caso es necesario agregar el tiempo que dedicará el robot a ejecutar dicho comportamiento (Figura 4).

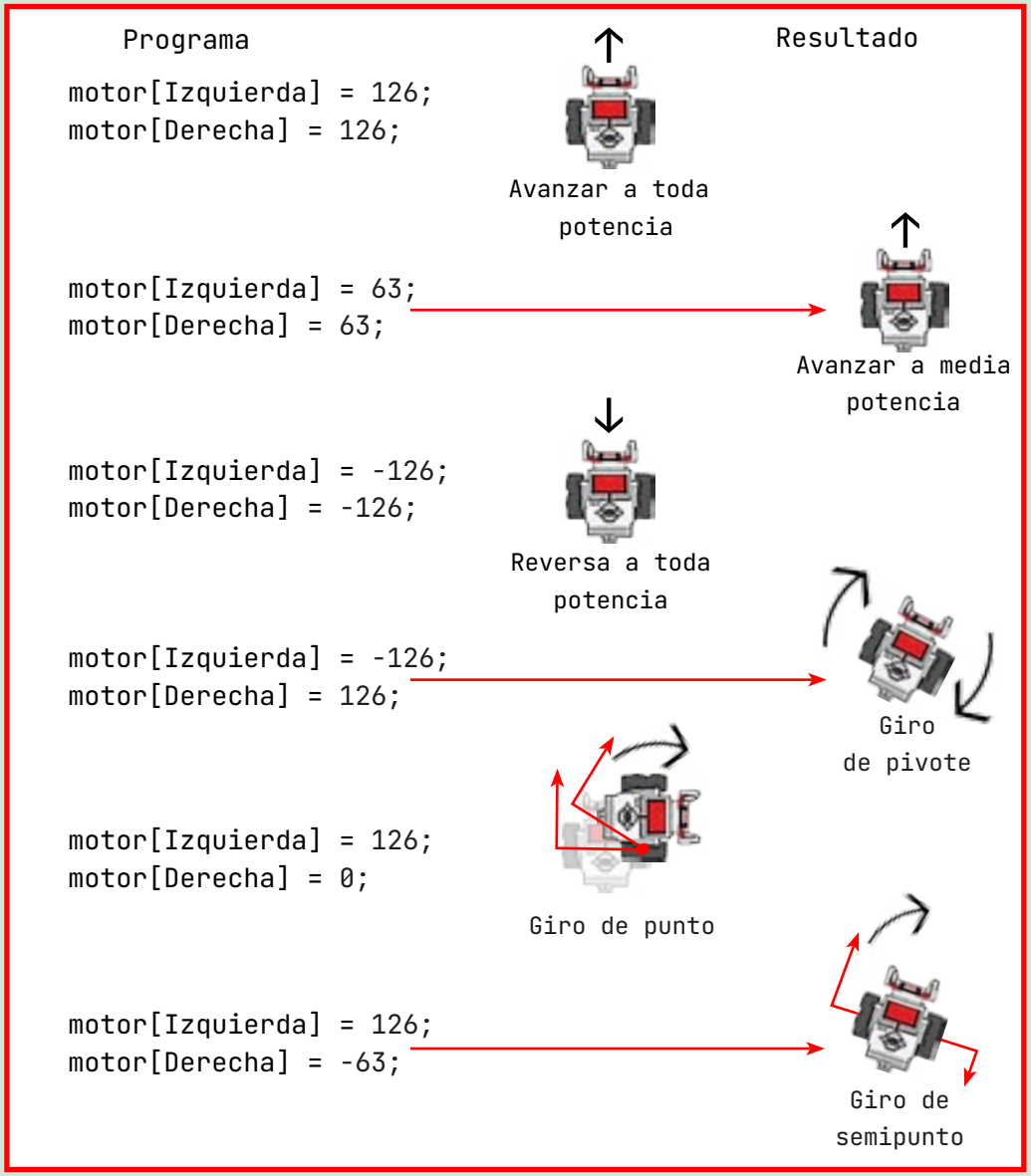


Figura 4.
COMPORTAMIENTO VARIANDO POTENCIA DE LOS MOTORES.

SOLUCIÓN DEL DESAFÍO

Puesto que nunca haz enfrentado un proyecto de ingeniería de este tipo es necesario que apliques el concepto de "divide y vencerás": dividiremos el desafío en varias etapas para que vayas construyendo el conocimiento necesario y resuelvas la problemática de esta sección.

Velocidad y dirección

Lograr que el robot vaya derecho no es simple. Hay muchas variables que entran en juego:

- Los motores pueden generar diferentes empujes,
- Puede haber más fricción en un eje que en otro,
- El peso del robot podría no estar equilibrado o parejo,
- Las llantas pueden estar fuera de balance, etcétera.

Eventualmente recibirás respuesta de los sensores que te pueden ayudar a enderezar el movimiento de tu robot, pero mientras tanto tendrás que programarlo manualmente. Para lograrlo hay que variar ligeramente la potencia que se aplica a los motores. Primero tenemos que conocer el comportamiento del robot, y para eso tienes que hacer otras investigaciones.

Investigación de la distancia recorrida vs. nivel de potencia para recorrido rectilíneo

En esta investigación tienes que encontrar la relación entre el nivel de potencia del motor y la distancia recorrida. Determina la relación programando tu robot para que corra a 25, 50, 75, y a 100% de potencia por 1, 3, y 5 segundos. Considera que la potencia máxima del robot es de 127, si es necesario, aplica redondeo al número inmediato inferior. Marca la zona de arranque para que obtengas resultados consistentes en todas las pruebas. El valor numérico asignado al comando motor representa la fuerza con la que el motor funciona, 127 indica a toda potencia mientras que un valor de 63 indica aproximadamente media potencia. Llena la tabla 2 con los valores de potencia faltantes (usa la regla de tres). Usa el Programa 1 (véase) para llenar la tabla 3 con los valores resultantes al momento de ejecutar las pruebas.

Programa 1.

PROGRAMA 1 PARA LLENAR LA TABLA 2.

```
1  #pragma config(Motor,  motorB,      Derecha.     tmotorNormal)
2  #pragma config(Motor,  motorC,      Izquierda,   tmotorNormal1)
3
4
5  task main (){
6  // Activación de los motores
7  motor[motorB] = 126;
8  motor[motorC] = 126;
9  // Tiempo de espera  de
10 // concluir el programa
11 // Igual al tiempo de recorrido
12 wait1Msec (1000);
13 }
```

Nivel de potencia.
Modificar valores para: 100, 75, 50 y 25%

Tiempo de recorrido.
Modificar para 1, 3 y 5 segundos

Tabla 2.

VALORES DE POTENCIA POR CADA PORCENTAJE.

Porcentaje	Potencia
100%	127
75%	
50%	63
25%	

Nivel de potencia	Tiempo recorrido milisegundos)	Distancia recorrida
25%	1000	
50%	1000	
75%	1000	
100%	1000	

25%	3000	
50%	3000	
75%	3000	
100%	3000	

25%	5000	
50%	5000	
75%	5000	
100%	5000	

Tabla 3.

RESULTADOS DE NIVEL DE POTENCIA

VS. DISTANCIA RECORRIDA.

Para que el robot mueva el brazo, aplica los comandos que ya conoces (se mueve con un motor en puerto D), aplica una potencia baja (25%). Documenta el movimiento circular del brazo en la tabla 4.

Tabla 4.

TIEMPO DE RECORRIDO

VS. GRADOS DE GIRO DEL BRAZO.

Tiempo de recorrido	Grados de giro
100	
200	
300	

Ahora se te proporcionan tres herramientas que permiten resolver el problema.

1. Solución del problema con una serie de pasos (Tabla 5) explicados en idioma español, como una receta de cocina.
2. Solución con pseudocódigo (Tabla 6), expresado también en idioma español, concreto.
3. Solución con algoritmo usando la notación de diagramas de flujo (Figura 5).

Asegúrate de estudiar completamente cada solución, ya que el siguiente paso es que generes el programa que resuelva el desafío.

Relaciona estos conceptos con el proceso de ingeniería que se te ha presentado en la sección "Conceptos básicos".

Todas las soluciones se explican usando la técnica del comportamiento simple. Tu tarea es convertirla en comportamiento básico (observa la sección "Conceptos básicos-Comportamientos".

A continuación, las tres herramientas mencionadas.

Tabla 5.

SOLUCIÓN PARECIDA

A UNA RECETA DE COCINA.

1. Inicio del programa
2. Esperar 2 segundos
3. Avanzar *x* segundos
4. Levantar el brazo *y* grados
5. Retroceder *x* segundos
6. Bajar el brazo *y* grados
7. Final del programa

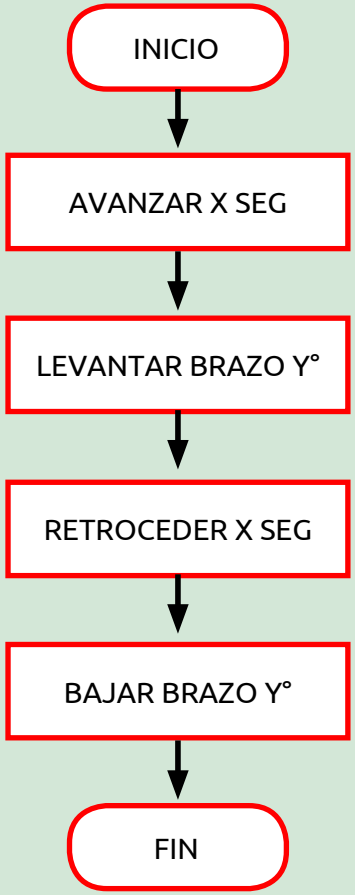
Tabla 6.

PSEUDOCÓDIGO PARA EL DESAFÍO.

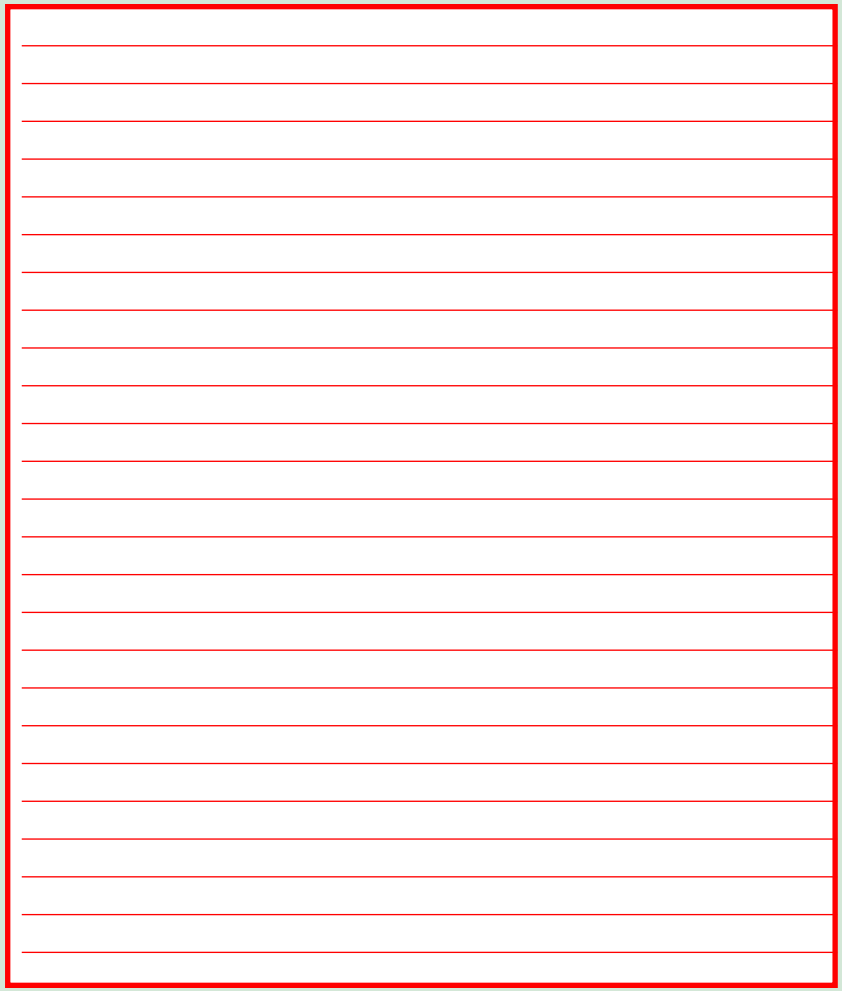
```
 8 task main()
 9 {
10   Esperar 2 segundos
11   Avanzar x segundos
12   Levantar brazo x segundos
13   Retroceder x segundos
14   Bajar brazo y grados
15 }
```

Figura 5.

ALGORITMO PARA EL DESAFÍO.



Anota en este recuadro el programa resultante con el que diste solución a los pasos de este desafío:



Construyendo el conocimiento del recorrido rectilíneo

1. Crea tres gráficas separadas (una para cada tiempo de recorrido) colocando el nivel de potencia *vs.* distancia recorrida.
2. ¿Existen relaciones entre el nivel de potencia, distancia recorrida y el tiempo de recorrido? Explícalas, o explica porqué no hay relaciones.
3. Selecciona un nivel de potencia y calcula el desnivel de su línea en cada una de las tres gráficas. ¿Los desniveles son los mismos? Si no es así, especula porqué existen estas diferencias.
4. ¿Cómo se corrige la desviación del robot en recorrido rectilíneo cuando avanza a toda potencia?

TRABAJO ADICIONAL

Modifica tu programa para que ejecute el siguiente desafío.

El robot debe arrancar de su posición inicial hasta la zona objetivo, ahí debe recoger con la mandíbula frontal el cubo que se le proporciona; dará un giro de 180° y regresará a su posición

inicial. Finalmente, debe dar otro giro de 180° grados y depositar el cubo en el piso. Esto te permitirá comprender el comportamiento de la mecánica del robot.

Explora los comportamientos que obtienes al variar la potencia de los motores, documentados en la sección "Conceptos básicos".

Referencias

RobotC for LEGO Mindstorms Web Help. (s.f.). ROBOTC Downloads. Consultado 2 de enero de 2018: https://www.robotc.net/WebHelpMindstorms/index.htm#Resources/topics/ROBOTC_Interface/Overview.htm

Curriculum for Tetrix and Lego Mindstorm Product Overview. (s.f.). Consultado el 2 de enero de 2018: http:// www.education.rec.ri.cmu.edu/products/teaching_ROBOTC_tetrix/home/ROBOTC_TETRIX_Curriculum.pdf

Cairó O. Metodología de la programación. *Algoritmos, diagramas de flujo y programas*. Alfaomega. 3ª. edición.

ROBOT VELADOR, LABORATORIO 2

COMPORTAMIENTO BÁSICO: RECORRIDO DE GIRO

OBJETIVOS DEL DESAFÍO QUE SE TE PLANTEA:

→ Analizar el comportamiento básico del robot en cuanto a velocidad y dirección para desarrollar pensamiento lógico

→ Conseguido ese pensamiento lógico, con él diseñar un algoritmo y construir un programa de computadora apropiado con lenguaje de programación ROBOTC®. Diseñar un algoritmo y construir un programa de computadora apropiado con lenguaje de programación ROBOTC®.

→ Implementar ese algoritmo en un robot Lego® y probar que el robot ejecuta las instrucciones adecuadas al desafío.

Luis Antonio Álvarez Oval, Christian Mauricio Castillo Estrada y Aron de la Cruz Vázquez, *Introducción a la robótica*, Sin fronteras, núm. 3, UNACH, México, 2022, 180 pp. ISBN: 978-607-561-130-3

www.unach.mx

Disponible en: www.editorial.unach.mx

INTRODUCCIÓN

ROBOTC® y el robot nos permiten trasladar la programación del mundo abstracto dentro de la computadora al mundo real, en donde cada orden que se escribe puede ser percibida como la actividad que desarrolla el robot. El estudiante debe comprender la programación de computadoras como una actividad lógica, sencilla de entender y aplicar; esa comprensión le permitirá desarrollar conceptos mucho más abstractos tales como la comunicación entre procesos remotos, programación orientada a objetos, programación web, etcétera.

El robot que estaremos usando a lo largo de este curso tiene una configuración de dos ruedas, cada una con su propio motor, que serán los actuadores, y una rueda de arrastre omnidireccional; en el tema de sensores, para percibir el ambiente el robot dispone de un sensor táctil, ultrasónico, y otro de luz/color.

DESAFÍO POR RESOLVER

La compañía ACME Robotics ha ganado un proyecto para diseñar un robot programable que patrulle los alrededores de un edificio rectangular que almacena autos, por lo que ACME te ha contratado para que desarrolles el programa para que ese robot prototipo pueda, de forma autónoma, "vigilar" los alrededores del edificio. Otro equipo de trabajo ya ha desarrollado el prototipo mecánico-electrónico del robot y está disponible para su uso. El proceso de ingeniería implica que la solución se vaya construyendo paso a paso, así que la rutina de vigilancia constante será desarrollada en laboratorios posteriores. Es necesario que imprima este documento ya que debes responder las preguntas que se te hacen en la sección "Solución del desafío". El administrador de este proyecto ha planteado dividir en seis etapas el proceso de ingeniería para el desarrollo del sistema que va a controlar el robot y que resolverá la problemática planteada por el cliente.

La problemática que se debe resolver en esta sección es la siguiente: el robot debe arrancar de su posición inicial hasta la posición final, el recorrido lo obliga a evadir algunos obstáculos. El recorrido adopta una forma de una equis.

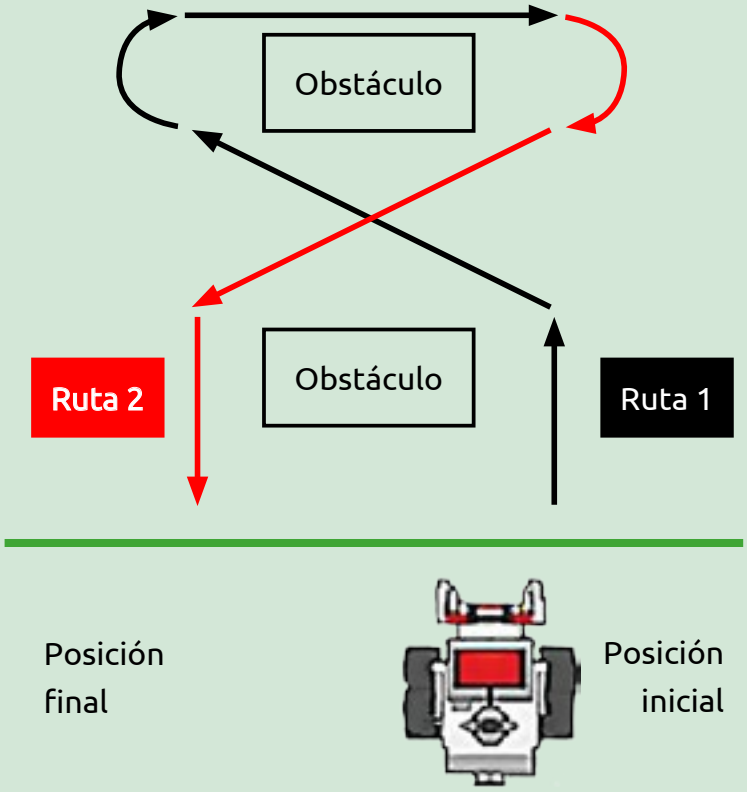
Solucionar el desafío incrementará tu comprensión del comportamiento de la mecánica del robot durante los giros.

Para este desafío necesitas los siguientes materiales:

- Cinta aislante de color negro
- Flexómetro
- Transportador
- Lenguaje de programación ROBOTC®
- Robot LEGO® armado
- Editor de diagramas DIA. Liga de descarga: http://dia-installer.de/

Figura 1.

RECORRIDO DEL ROBOT VELADOR



Es interesante abordar el concepto de diagramas de flujo nuevamente en este laboratorio, el estudiante debe conocer cada símbolo para representar de manera visual cada uno de los pasos de su algoritmo; para esto es necesario combinar un conjunto de bloques y flechas para representar las acciones. Así también, debe cuidar el orden en el cual las acciones ocurren, y para ello hacer uso de flechas de dirección que apuntan de una acción a otra. Hay que considerar que de un mismo bloque pueden derivar varias flechas de dirección, y cada una de ellas indicará la trayectoria o camino por seguir.

- **Inicio y fin.** Son representados mediante las figuras o símbolo de rectángulos redondeados; y dentro del mismo deberá contener las palabras "`inicio` o `fin`".
- **Acciones.** Se debe usar la figura de rectángulo para representarlas, mismas que} actúan como acciones básicas: "`wait1Msec (1000)`"; "`increment line count by 1`" o "`motor full ahead`".
- **Decisiones.** Los bloques de decisiones son representados como rombos; los cuales indican la toma de decisión de una u otra ruta en el flujo de un programa,

indicando los posibles caminos a seguir. Las flechas deben ser siempre etiquetadas de adecuadamente.

SOLUCIÓN DEL DESAFÍO

Puesto que nunca nos hemos enfrentado a un proyecto de ingeniería de este tipo, es necesario que apliquemos el concepto de "divide y vencerás"; dividiremos el desafío en varias etapas para ir construyendo el conocimiento necesario para resolver la problemática de esta sección.

Recorrido de giro

Hay dos tipos de giros que nuestro robot puede ejecutar: giro de pivote y giro de punto. El primero ocurre cuando una rueda gira mientras la otra permanece estacionaria. El segundo ocurre cuando a los dos motores se les asignan valores opuestos, a uno se asigna un valor positivo mientras que al opuesto uno negativo.

Determina los tiempos de recorrido para diferentes potencias de motor para lograr una vuelta a la izquierda de 90, 180 y 360 grados. Modifica el programa 1 para llenar las tablas 1 y 2 con tus resultados. Recorta la plantilla para resolver esta tarea.

Programa 1.

```
5   task main(){
6   // activación de los motores. Giro de Punto
7   motor [motorB] = -126;
8   motor [motorC] = 126;
9   // Tiempo de espera antes de
10  // Concluir el programa
11  // Igual al tiempo de recorrido
12  wait1msec(1000);
13  }
14
```

Nivel de potencia. Modificar valores para: 100, 75, 50, y 25% de potencia.

Tiempo de recorrido. Modificar para 1, 3 y 5 segundos.

Tabla 1.
RESULTADOS DE NIVEL DE POTENCIA
VS. GRADOS CON GIRO DE PIVOTE.

Nivel de potencia	Grados de giro	Tiempo de recorrido
25%	90°	
50%	90°	
75%	90°	
100%	90°	
25%	180°	
50%	180°	
75%	180°	
100%	180°	
25%	360°	
50%	360°	
75%	360°	
100%	360°	

Tabla 2.
RESULTADOS DE NIVEL DE POTENCIA
VS. GRADOS CON GIRO DE PUNTO.

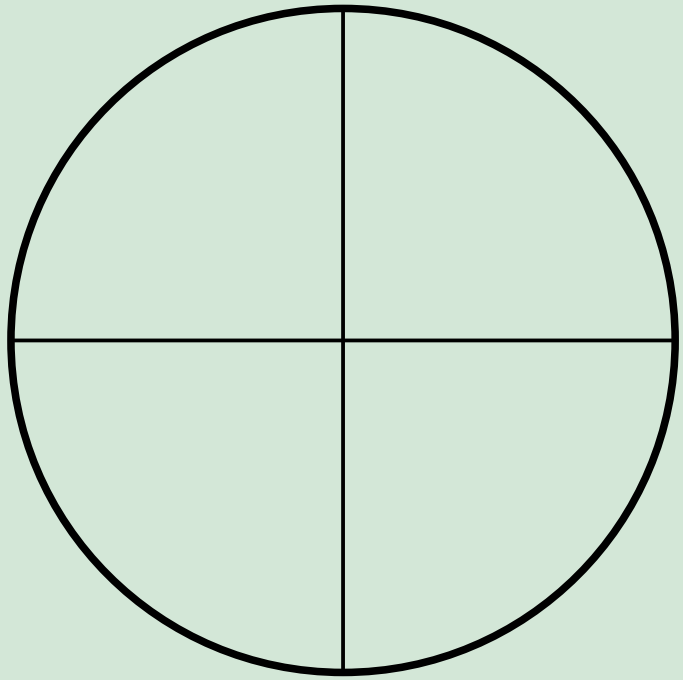
Nivel de potencia	Grados de giro	Tiempo de recorrido
25%	90°	
50%	90°	
75%	90°	
100%	90°	
25%	180°	
50%	180°	
75%	180°	
100%	180°	
25%	360°	
50%	360°	
75%	360°	
100%	360°	

Construyendo tu conocimiento sobre los giros

1. ¿Cuál de los dos tipos de giro es más preciso?
2. ¿Cómo determinaste en la práctica los grados recorridos por el robot durante el giro?
3. ¿Cómo determinarías la distancia lineal que recorrió el robot en el giro?
4. ¿Qué deberías hacer para que tu robot ejecute un giro hacia atrás? Construye un programa.
5. ¿Qué le pasaría a tu programa si le cambiaras el diámetro de las ruedas a tu robot?
6. ¿Qué pasa con la potencia cuando se carga la batería del robot?

Figura 2.

PLANTILLA PARA LA INVESTIGACIÓN DE LOS GIROS.



Ahora se te proporcionan tres herramientas que permiten resolver el problema.

1. Solución del problema con una serie de pasos (Tabla 3) explicados en idioma español, como una receta de cocina.
2. Solución con pseudocódigo (Tabla 4), expresado también en idioma español concreto.
3. Solución con algoritmo usando la notación de Diagramas de flujo (Figura 3). Asegúrate de estudiar completamente cada solución, ya que el siguiente paso es que generes el programa que resuelva el desafío. Relaciona estos conceptos con el proceso de ingeniería que se te ha presentado en la sección "Conceptos básicos". Todas las soluciones se explican usando la técnica del comportamiento simple, su tarea es convertir esta en comportamiento básico (ver sección "Conceptos básicos-comportamientos"). Con cinta aislante marca la ruta que debe seguir el robot y con el transportador determina los grados que debe girar el robot. A continuación, las tres herramientas ya mencionadas.

Tabla 3.

SOLUCIÓN PARECIDA

A UNA RECETA DE COCINA.

1.	Inicio del programa
2.	Esperar 2 segundos
3.	Avanzar *x* segundos
4.	Girar a la izquierda *y* grados
5.	Avanzar *x* segundos
6.	Girar a la derecha Y grados.
7.	Avanzar *x* segundos
8.	Girar a la derecha *y* grados.
9.	Avanzar *x* segundos
10.	Girar a la izquierda *y* grados
11.	Avanzar *x* segundos
12.	Fin del programa

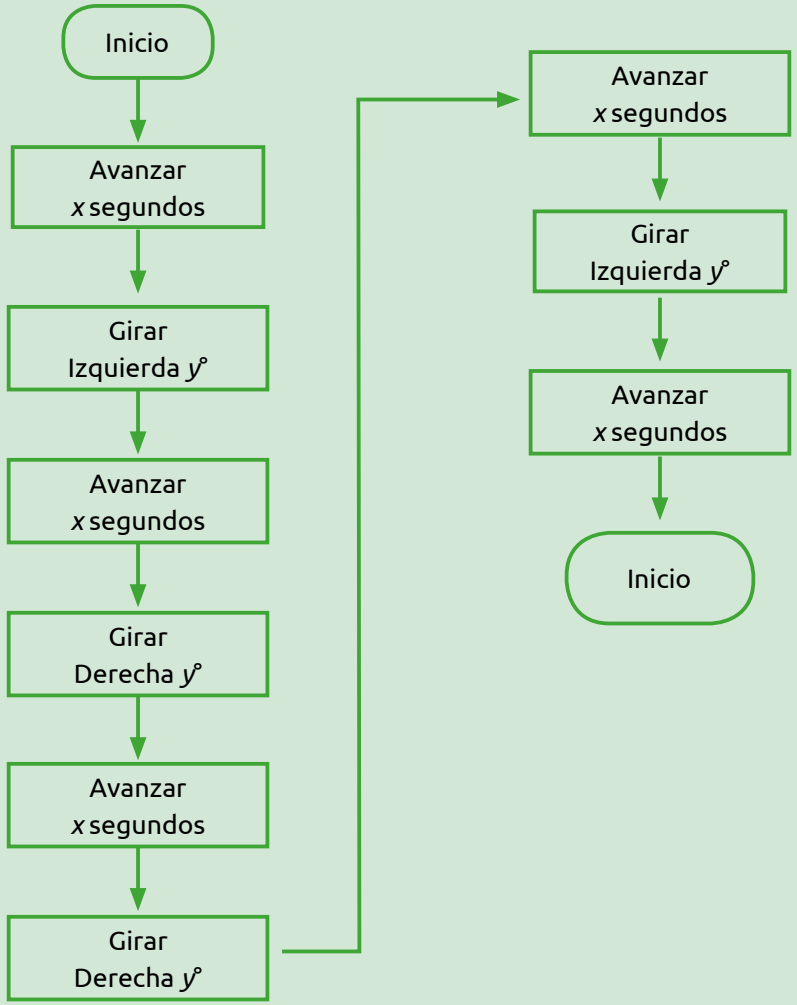
Tabla 4.

PSEUDOCÓDIGO PARA EL DESAFÍO.

```
 8     task main ()
 9     {
10     Esperar 2 segundos
11     Avanzar x segundos
12     Girar a la izquierda y grados
13     Avanzar x segundos
14     Girar a la derecha y grados
15     Avanzar x segundos
16     Girar a la derecha y grados
17     Avanzar x segundos
18     Girar a la izquierda y grados
19     Avanzar x segundos
20     }
21
```

Figura 3.

ALGORITMO PARA EL DESAFÍO.



Anota en el recuadro el programa resultante con el que solucionaste este desafío.

TRABAJO ADICIONAL

Modifica tu programa para resolver una nueva situación. El robot ejecuta las mismas funciones que el desarrollado previamente, sólo que después de recolectar el cubo, el robot debe dar un giro de 180° y viajar en reversa hasta su posición final (las líneas rojas indican el recorrido en reversa). Tal como se aprecia en la figura 4.

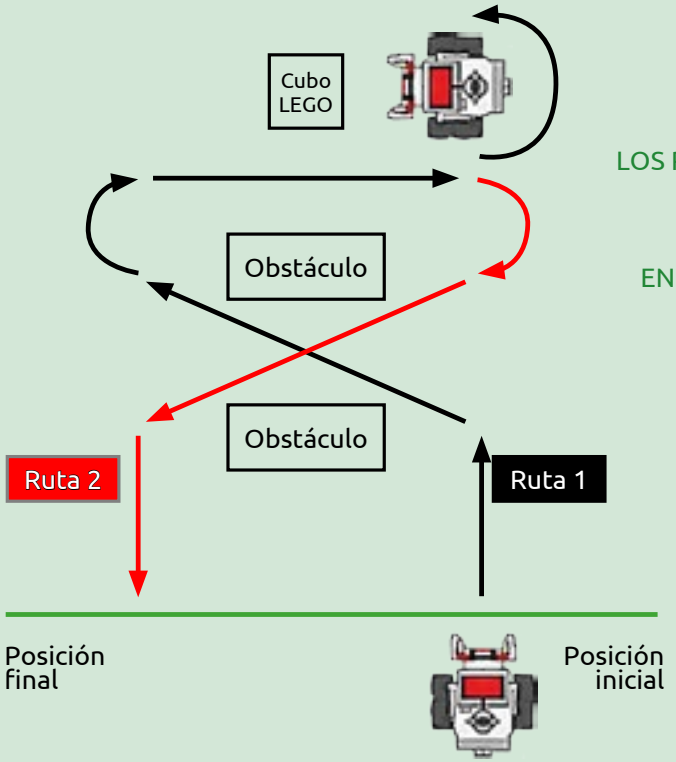


Figura 4.
LOS RECORRIDOS DEL ROBOT EN EL TRABAJO ADICIONAL.

Referencias

Robomatter Inc. (2016). ROBOTC for LEGO Mindstorms 4.X - Users Manual. Disponible para consulta en el sitio web: https://www.robotc.net/WebHelpMindstorms/index.htm#Resources/topics/ROBOTC_Interface/Overview.htm

Carnegie Mellon University (2020). ROBOTC for TETRIX & LEGO® MINDSTORMS.https://www.cmu.edu/roboticsacademy/roboticscurriculum/Lego%20Curriculum/ROBOTC-Tetrix_Mindstorms.html

Cairó, O (2005). Metodología de la programación. Algoritmos, diagramas de flujo y programas. Alfaomega. 3ª. edición.

Joyanes, L. (2008). Fundamentos de Programación. Algoritmos y estructura de datos, Mc Graw Hill.

ROBOT VELADOR, LABORATORIO 3

UNA VUELTA COMPLETA AL EDIFICIO

OBJETIVOS DEL DESAFÍO QUE SE TE PLANTEA:

→ Desarrollar el pensamiento lógico necesario para diseñar un algoritmo, y construir un programa de computadora apropiado con ROBOTC

→ Implementar ese algoritmo en un robot Lego y probar que el robot ejecuta las instrucciones adecuadamente para que el robot ejecute una vuelta completa a la maqueta de un edificio de forma rectangular.

Luis Antonio Álvarez Oval, Christian Mauricio Castillo Estrada y Aron de la Cruz Vázquez, *Introducción a la robótica*, Sin fronteras, núm. 3, UNACH, México, 2022, 180 pp. ISBN: 978-607-561-130-3

www.unach.mx

Disponible en: www.editorial.unach.mx

INTRODUCCIÓN

En este tercer laboratorio, el estudiante deberá utilizar el lenguaje de programación ROBOTC® para diseñar el algoritmo y codificarlo; así también, hará uso de ciertos materiales complementarios: bloque de madera, flexómetro y cronómetro con la finalidad de desarrollar una solución para el reto planteado, y finalmente, realizar las pruebas del correcto funcionamiento del algoritmo, con el robot en ejecución.

Así también, el estudiante debe hacer uso del IDE que ofrece ROBOTC®, consultando el apartado de ayuda, mismo que despliega una lista de comandos disponibles y su correspondiente descripción; cada uno de estos comandos se traduce a lenguaje máquina con la finalidad que el robot los pueda interpretar y ejecutar.

ROBOTC® es un poderoso lenguaje de programación con base en C, en ambiente Windows® para escribir y depurar programas, y es el único que depura exhaustivamente en tiempo real. Es una solución multiplataforma que permite que los estudiantes aprendan el tipo de programación con base en C, usada en educación avanzada y aplicaciones profesionales.

Los comandos para el robot primero se escriben en la pantalla, después son procesados por el compilador de ROBOTC® en un archivo de lenguaje de máquinas que el robot comprende. Finalmente, son cargados en el robot en donde pueden ejecutarse.

ROBOTC® y el robot nos permiten trasladar la programación del mundo abstracto de la computadora al mundo real, en donde cada orden escrita puede percibirse como actividad del robot. El estudiante debe comprender la programación de computadoras como una actividad lógica sencilla de entender y aplicar; esa comprensión le permitirá desarrollar conceptos mucho más abstractos.

El robot que estaremos usando a lo largo de este curso tiene una configuración de dos ruedas, cada una con su propio motor, que serán los actuadores, y una rueda de arrastre omnidireccional; en el tema de sensores, percibe el ambiente que lo rodea con un sensor táctil, ultrasónico, y otro de luz/color.

Para este desafío necesitas los siguientes materiales:

- Marco de madera (que representará el edificio).
- Flexómetro.
- Teléfono celular con sistema operativo Android y con la aplicación StopWatch (cronómetro) instalada. Enlace para descargar la aplicación móvil: https://play.google.com/store/apps/details?id=com.hybrid.stopwatch&hl=en_US&gl=US
- https://play.google.com/store/apps/details?id=com.LogicalACTS.Stopwatch&hl=es
- Lenguaje de programación ROBOTC®.
- Robot Lego® armado.
- Editor de diagramas DIA. Liga de descarga: http://dia-installer.de/

DESAFÍO POR RESOLVER

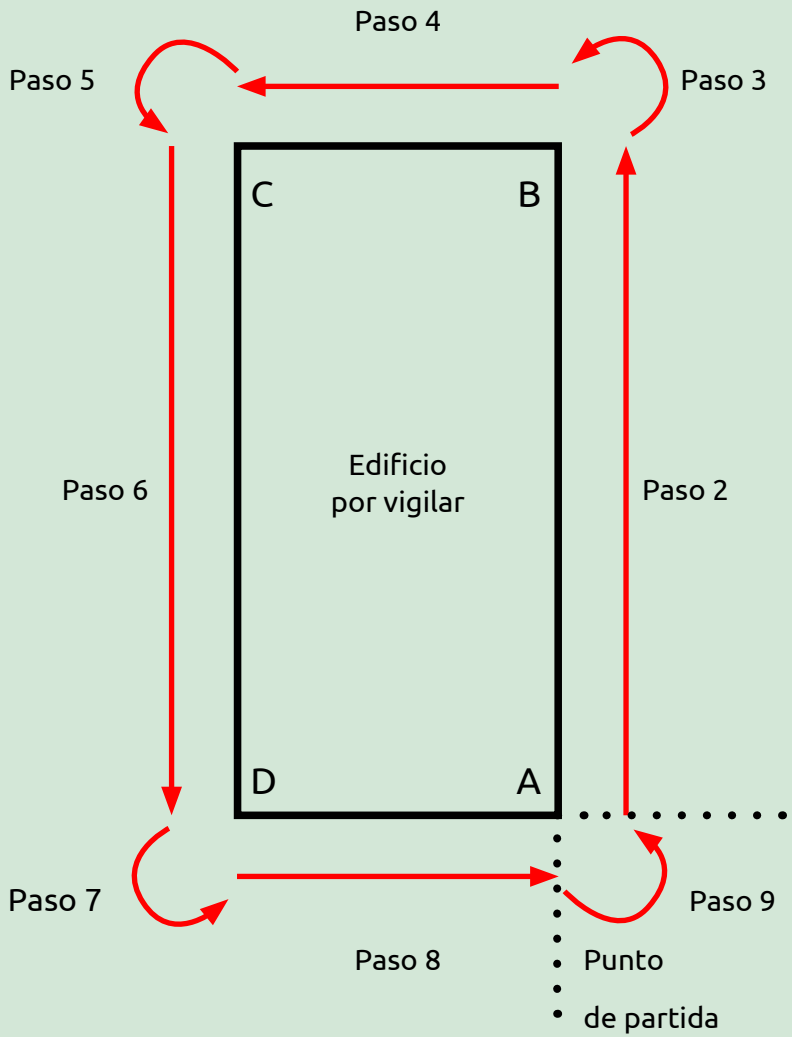
La compañía ACME Robotics ha ganado un proyecto para diseñar un robot programable que patrulle los alrededores de un edificio rectangular que almacena autos, por lo que ACME te ha contratado para desarrollar el programa para que ese robot prototipo pueda, de forma autónoma, "vigilar" los alrededores del edificio. La figura 1, explica paso a paso la solución del problema. Otro equipo de trabajo ya ha desarrollado el prototipo mecánico-electrónico del robot y está disponible para su uso.

El programa que debes construir en ROBOTC®, debe contener las instrucciones para dar una vuelta completa al edificio; como se trata de un prototipo no es necesario que ejecute una rutina de vigilancia constante. El proceso de ingeniería implica que la solución se vaya construyendo paso a paso, así que la rutina de vigilancia constante será desarrollada en laboratorios posteriores.

Es necesario que imprimas este documento ya que deberás responder las preguntas de la sección "Solución del desafío".

Figura 1.

RECORRIDO DEL ROBOT VELADOR.



CONCEPTOS BÁSICOS

Aquí encontrarás material de apoyo para entender el objetivo primario de este documento. En general los apuntes provienen de los documentos que el Laboratorio Nacional de Ingeniería Robótica de la Universidad Carnegie Mellon ha proporcionado en los cursos de robótica que imparte.

Pseudocódigo

El pseudocódigo es una técnica para expresar la serie de pasos o acciones que integran un algoritmo, y ocupa diversas estructuras de control, selección y repetitivas usando expresiones verbales informales sin depender de un lenguaje de programación en particular. La importancia radica en expresar rápidamente el comportamiento o resultado de cada uno de los pasos o acciones del algoritmo a través del uso de palabras comunes; en vez de invertir gran cantidad de tiempo en aprender y codificar con una correcta sintaxis de cierto lenguaje de programación.

En el ámbito de la programación, el pseudocódigo resulta ser una técnica para delimitar un programa antes de codificarlo usando el conjunto de palabras reservadas y sintaxis propia. Esto

representa un beneficio durante la etapa de análisis y planeación. Otro beneficio adicional del pseudocódigo es la facilidad que ofrece para poder traducirlo a diferentes lenguajes de programación; en otras palabras, es posible señalar que el pseudocódigo se enfoca en capturar la lógica de una solución propuesta.

En los laboratorios anteriores se ha mencionado que los diagramas de flujo suelen ser representaciones visuales de los pasos o acciones de un programa; en ese sentido, estará integrado por diversos símbolos o figuras, bloques y flechas de dirección para indicar la secuencia de cada paso o acción. Por otra parte, en ocasiones de un mismo bloque es posible derivar diversas flechas de dirección para representar cada una de las rutas o trayectorias por seguir.

- **Inicio y fin.** Estos términos son representados mediante símbolos de figuras geométricas, de manera particular un rectángulo redondeado, usualmente contienen las palabras "`inicio` o `fin`"; sin embargo, no es limitativo para incluir

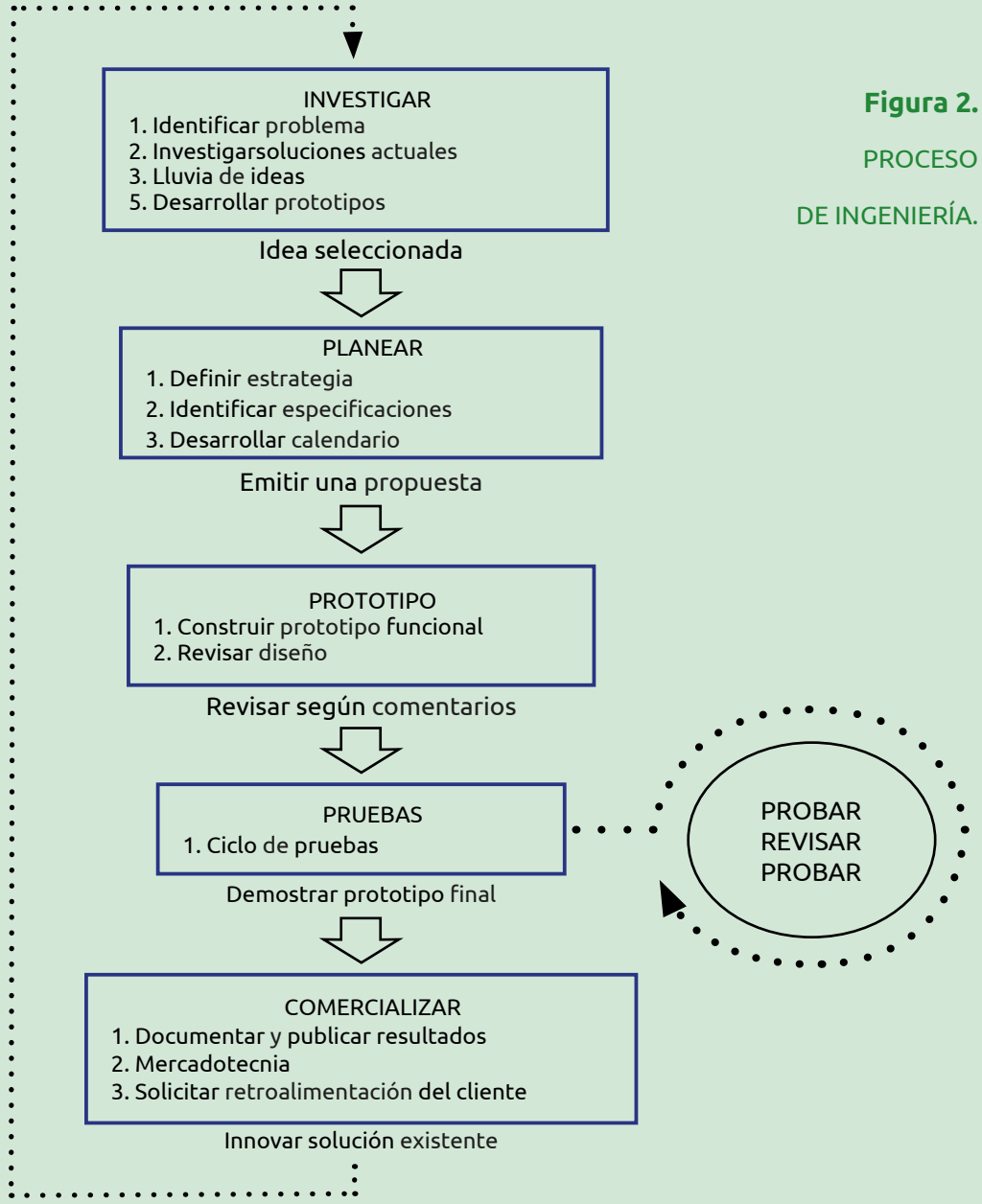


Figura 2.
PROCESO
DE INGENIERÍA.

otras palabras como por ejemplo “potencia del robot apagada” o “detener todos los motores”.

- **Cada tarea o acción** es simbolizada a través de figuras de rectángulos y funcionan como comandos elementales.
- **Decisiones.** Los bloques de decisiones son representados por el símbolo o figura de rombo; evaluando un valor mediante el uso de expresiones booleanas que ayudarán a tomar la decisión de elegir el camino correcto de acuerdo las flechas de dirección.

Proceso de ingeniería

Con base en lo manifestado en el diccionario de la Real Academia de Lengua Española, el término *proceso* se define como “conjunto de las fases sucesivas de un fenómeno natural o de una operación artificial”. Observa la figura 2, que te describe todos los pasos que debes llevar a cabo para proyectos de ingeniería.

SOLUCIÓN DEL DESAFÍO

Ahora se te proporcionan las tres herramientas que permiten resolver el problema.

1. Solución del problema con una serie de pasos (Tabla 1) explicados en idioma español, como una receta de cocina.
2. Solución con pseudocódigo (Tabla 3), expresado también en idioma español concreto.
3. Solución con algoritmo usando la notación de diagramas de flujo (Figura 3).

Asegúrate de estudiar completamente cada solución, ya que el siguiente paso es que generes el programa que resuelva el desafío. Relaciona estos conceptos con el proceso de ingeniería que se te ha presentado en la sección “Conceptos básicos”.

A continuación, las tres herramientas ya mencionadas.

Tabla 1.

SOLUCIÓN DEL PROBLEMA COMO UNA RECETA DE COCINA

(sigue en la página siguiente)

1. Mide el extremo más largo del rectángulo. Anota cuánto mide: _____
2. Haz que el robot viaje del punto A al B a ½ potencia. Usa el cronómetro del celular para determinar el tiempo que le toma al robot. Apaga el motor al llegar al extremo contrario del marco. Considera que al llegar al extremo contrario del punto de partida el robot deberá dar vuelta a la izquierda, por lo que debes darle suficiente espacio para que el dispositivo no choque con el edificio. Describe tus acciones: _____
3. Ahora, haz que el robot gire a la izquierda (aplica ¼ de potencia del motor del robot). ¿Cuánto tiempo debes aplicar fuerza a las ruedas para dar un giro de 90°? Anota tu respuesta: _____.
4. Haga que el robot viaje del punto B al C a ½ potencia. Usa los conceptos de matemáticas que ya conoces (regla de tres) y determina

Tabla 2.

SOLUCIÓN DEL PROBLEMA COMO UNA RECETA DE COCINA

(finaliza)

cuánto tiempo necesita el robot para alcanzar la siguiente esquina del edificio. Haz los ajustes necesarios a su programa. Anota tu primera respuesta: _____
5. Ahora, haz que el robot gire a la izquierda (aplica ¼ de potencia del motor del robot). Usa los datos que obtuviste en el punto 3.
6. Haz que el robot viaje del punto C al D a ½ potencia. Usa los datos que obtuviste en el paso 2.
7. Ahora, que el robot gire a la izquierda (aplique ¼ potencia al motor del robot). Usa los datos del punto 3.
8. Haz que el robot viaje del punto D al A a ½ potencia. Usa los datos obtenidos en el paso 4.
9. Ahora, haz que el robot gire a la izquierda (aplique ¼ potencia del motor del robot). Use los datos que obtuviste en el punto 3.

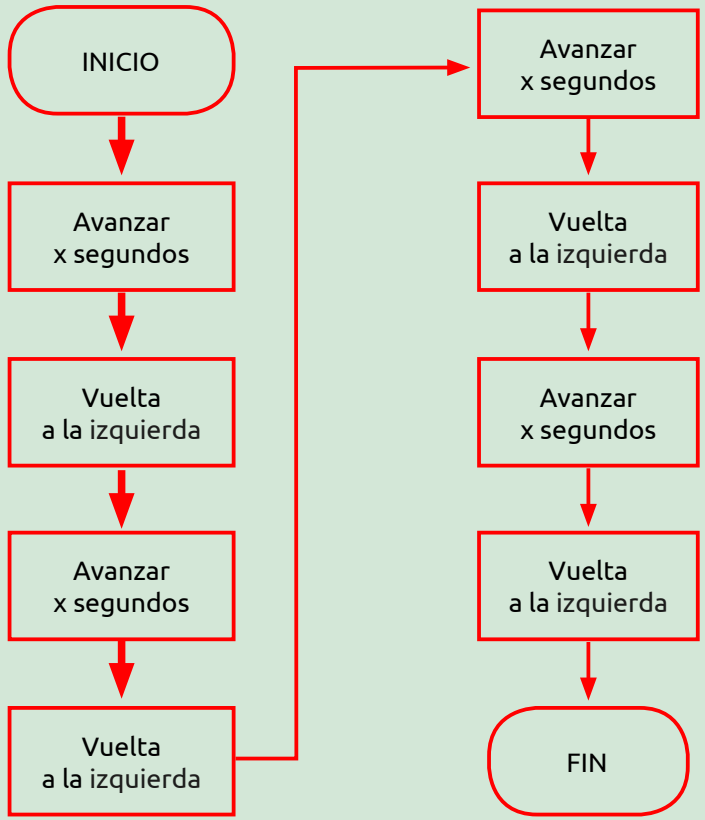
Tabla 3.

PSEUDOCÓDIGO PARA EL DESAFÍO.

```
 8    // Desafío Robot Velador
 9    // Mueve el robot intentando seguir un rectángulo
10    // por lo que el tiempo de recorrido puede variar
11    // En cada esquina gira 90 grados
12    task main()
13    {
14    // Movimiento del punto A al punto B
15    avanzar el robot x segundos
16    // Giro en la esquina
17    girar a la Izquierda
18    // Movimiento del punto B al punto C
19    avanzar el robot x segundos
20    // Giro en la izquierda
21    girar a la Izquierda
22    // Movimiento del punto C al punto D
23    avanzar el robot x segundos
24    // Giro en la esquina
25    girar a la izquierda
26    // Movimiento del punto D al punto A
27    avanzar el robot x segundos
28    // Giro en la esquina
29    girar a la Izquierda
30    }
31
```

Figura 3.

ALGORITMO PARA EL DESAFÍO.



Anota en el recuadro el programa resultante con el que diste solución a los pasos de este desafío.

TRABAJO ADICIONAL

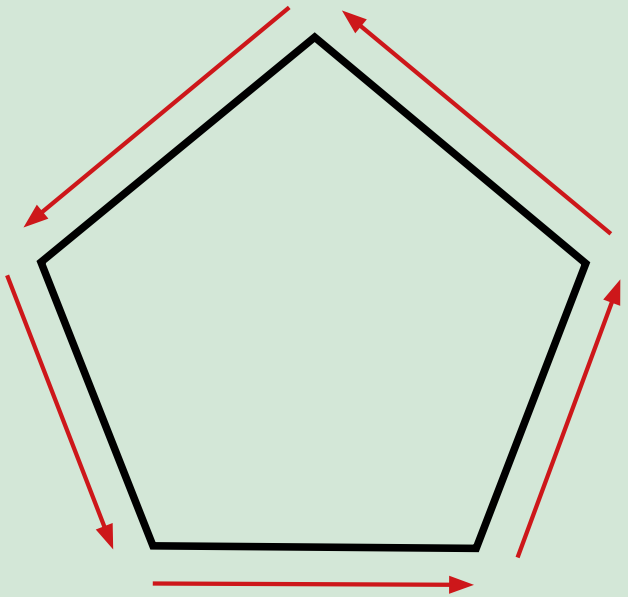
Los desafíos que se encuentran en esta sección deben ser resueltos por el estudiante, es necesario crear un algoritmo y el programa que da solución a cada desafío. La finalidad de este ejercicio es la de ir construyendo programas cada vez más complejos y verificar el avance que el estudiante tiene.

El Ejército de los Estados Unidos posee un edificio conocido como el "Pentágono" ahí se concentra el poder militar de dicho país. Si ACME Robotics tuviera que vigilar dicho complejo con el mismo robot, ¿qué cambios tendrías que hacer a tu programa para que el robot dé una vuelta completa al edificio? Construye tu algoritmo y el programa en ROBOTC® para hacer el recorrido. Dibuja en un cartón un pentágono con lados de 40 centímetros para probar el recorrido del robot, construye paredes de 10 cm de altura para que el pentágono sea tridimensional. Observa la figura 4, para que tengas una idea del recorrido del robot. Aunque no se muestran las vueltas en la figura, considera que el robot debe girar en las aristas.

Figura 4.

RECORRIDO DEL ROBOT

PARA EL EDIFICIO DEL “PENTÁGONO”.



1) Un proyecto adicional ha sido asignado a ACME Robotics, el edificio ahora tiene la forma de una cruz.

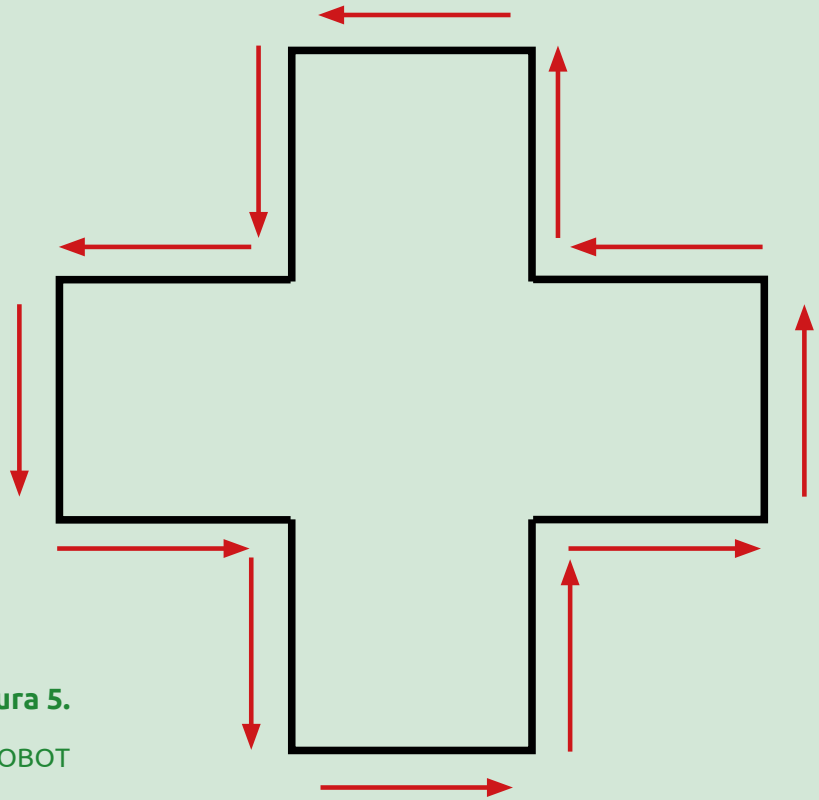


Figura 5.

RECORRIDO DEL ROBOT

PARA EL EDIFICIO CON FORMA DE CRUZ.

¿Qué cambios tendrías que hacer a tu programa para recorrer el perímetro completo del edificio? Construye tu algoritmo y el programa en ROBOTC® para hacer el recorrido. Recorta una cruz con lados de 30 centímetros en un cartón para probar el recorrido del robot, construye paredes de 10 cm de altura para que el modelo sea tridimensional. Observa la figura 5, para que tengas una idea del recorrido del robot. Aunque no se muestran las vueltas en la figura, considera que el robot debe girar al dar vuelta en las aristas.

Referencias

Robomatter Inc. (2016). ROBOTC for LEGO Mindstorms 4.X - Users Manual. Disponible para consulta en el sitio web: https://www.robotc.net/WebHelpMindstorms/index.htm#Resources/topics/ROBOTC_Interface/Overview.htm

Carnegie Mellon University (2020). ROBOTC for TETRIX & LEGO® MINDSTORMS.https://www.cmu.edu/roboticsacademy/roboticscurriculum/Lego%20Curriculum/ROBOTC-Tetrix_Mindstorms.html

ROBOT VELADOR, LABORATORIO 4

TRES VUELTAS COMPLETAS AL EDIFICIO

OBJETIVOS DEL DESAFÍO QUE SE TE PLANTEA:

→ Desarrollar el pensamiento lógico necesario para diseñar un algoritmo usando CICLOS®.

→ Construir un programa de computadora apropiado con ROBOTC® para implementar dicho algoritmo en un robot LEGO®.

→ Probar que el robot ejecuta las instrucciones adecuadamente para que el robot ejecute una vuelta completa a un edificio de tipo rectangular.

Luis Antonio Álvarez Oval, Christian Mauricio Castillo Estrada y Aron de la Cruz Vázquez, *Introducción a la robótica*, Sin fronteras, núm. 3, UNACH, México, 2022, 180 pp. ISBN: 978-607-561-130-3

www.unach.mx

Disponible en: www.editorial.unach.mx

INTRODUCCIÓN

El desafío por resolver en este cuarto laboratorio consiste en diseñar y codificar un algoritmo que permita al robot efectuar una vuelta completa a la representación de un edificio de tipo rectangular, que haremos con un bloque de madera; el estudiante debe aplicar los principios teóricos de las estructuras repetitivas e identificar los valores de potencia idóneos para ser asignados a los motores, que permitan realizar los giros con precisión en cada esquina del edificio.

De igual manera que en los laboratorios anteriores, el estudiante debe seleccionar el conjunto de comandos necesarios para codificar el algoritmo, posteriormente debe efectuar el proceso de compilación para traducir los comandos al lenguaje-máquina; y finalmente, realizar las pruebas evaluando el comportamiento del robot al ejecutar el programa

ROBOTC® es un poderoso lenguaje de programación basado en C con un ambiente Windows para escribir y depurar programas, y es el único que ofrece un depurador exhaustivo en tiempo real. Es una solución multiplataforma que permite a los estudiantes aprender el tipo de programación basada en C usada en educación avanzada y aplicaciones profesionales.

Los comandos para el robot los escribes primero en la pantalla, después los procesas en el compilador de ROBOTC®, que es un archivo de lenguaje de máquinas que el robot comprende. Finalmente, los cargas en el robot para que pueda ejecutarlos.

El robot que estaremos usando a lo largo de este curso tiene una configuración de dos ruedas, cada una con su propio motor, que serán los actuadores, y una rueda de arrastre omnidireccional; en el tema de sensores, para percibir el ambiente

que lo rodea, el robot dispone de un sensor táctil, ultrasónico, y de otro de luz/color.

Para este desafío necesitas los siguientes materiales:

- Marco de madera (para simular el edificio).
- Lenguaje de programación ROBOTC®.
- Robot Lego® armado.
- Editor de diagramas DIA. Liga de descarga: http://dia-installer.de/

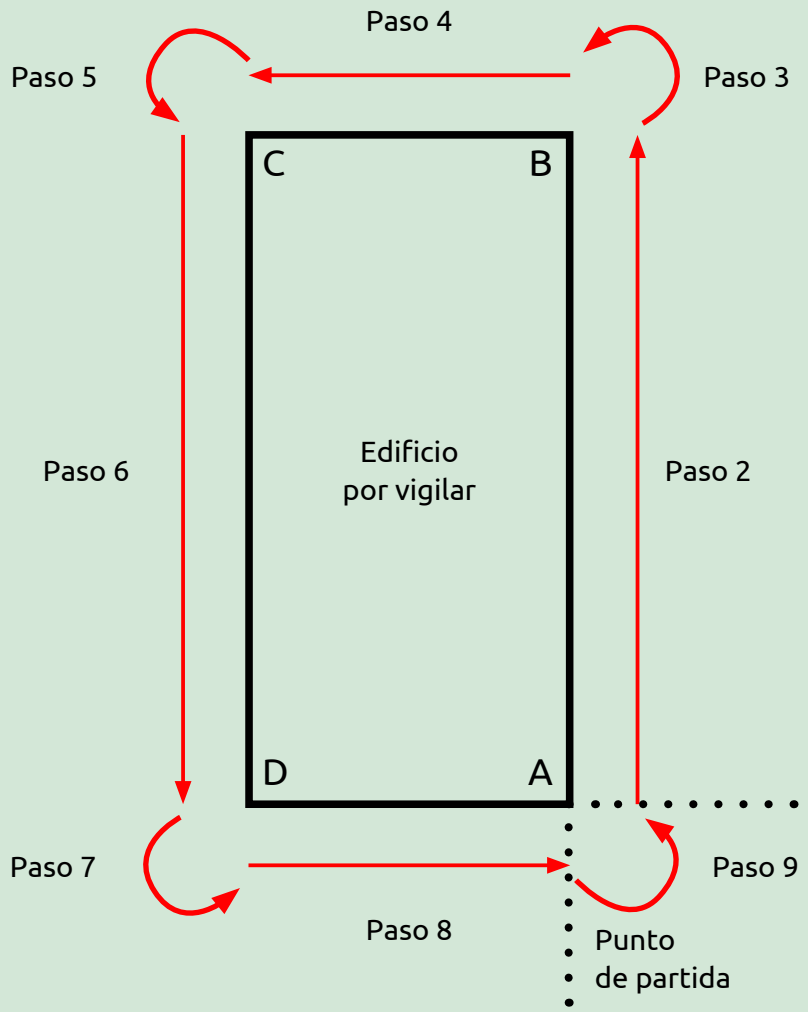
DESAFÍO POR RESOLVER

La compañía ACME Robotics ha ganado un proyecto para diseñar un robot programable que patrulle los alrededores de un edificio rectangular que almacena autos, por lo que ACME te ha contratado para que desarrolles el programa y que ese robot prototipo pueda de forma autónoma "vigilar" los alrededores del edificio.

La figura 1 explica paso a paso la solución del problema. Otro equipo de trabajo ya ha desarrollado el prototipo mecánico-electrónico del robot y está disponible para tu uso.

Figura 1.

RECORRIDO PARA SOLUCIONAR EL PROBLEMA



El programa que debes construir en ROBOTC® debe contener las instrucciones para dar tres vueltas completas al edificio. Como se trata de un prototipo no es necesario que ejecute una rutina de vigilancia constante. El proceso de ingeniería implica que vayas construyendo la solución paso a paso, así que la rutina de vigilancia constante será desarrollada en laboratorios posteriores.

Es necesario que imprimas este documento ya que debes responder las preguntas de la sección "Solución del desafío".

CONCEPTOS BÁSICOS

Aquí vas a encontrar material de apoyo para entender el objetivo primario de este documento. En general los conceptos básicos provienen de los documentos que el Laboratorio Nacional de Ingeniería Robótica de la Universidad Carnegie Mellon ha proporcionado en los cursos de robótica que imparte.

Variables globales

Cuando se crea una variable, ésta puede ser usada únicamente dentro de la función o tarea en la que fue declarada. Esto puede ser un problema cuando necesitas usar la misma variable en muchos lugares diferentes, por ejemplo, una función que creaste, y `task main`.

Alcance

Las variables existen únicamente entre ciertos límites, por ejemplo, sólo dentro de las funciones en las que fueron declaradas. El "alcance" es la "definición" de estos límites: que tan ampliamente aplicable (o visible) es un valor o variable. El alcance existe para prevenir los conflictos entre las funciones con variables de nombres parecidos, y más importante, para evitar que las variables de diferentes funciones intervengan accidentalmente unas con las otras, o incluso con ellas mismas si la misma función es ejecutada más de una vez.

En general, la regla es que una variable puede ser usada solamente dentro de la tarea o función en la que es declarada, incluyendo la tarea principal (`task main`). Si tratas de usar una variable en una locación fuera de su alcance, ROBOTC® te dará un error y declara que tal variable no existe, porque el programa en ese punto no es capaz de "verla".

Declarar una variable global. La variable global es declarada fuera de cualquier tarea o función, permitiendo que todas las “vean”.

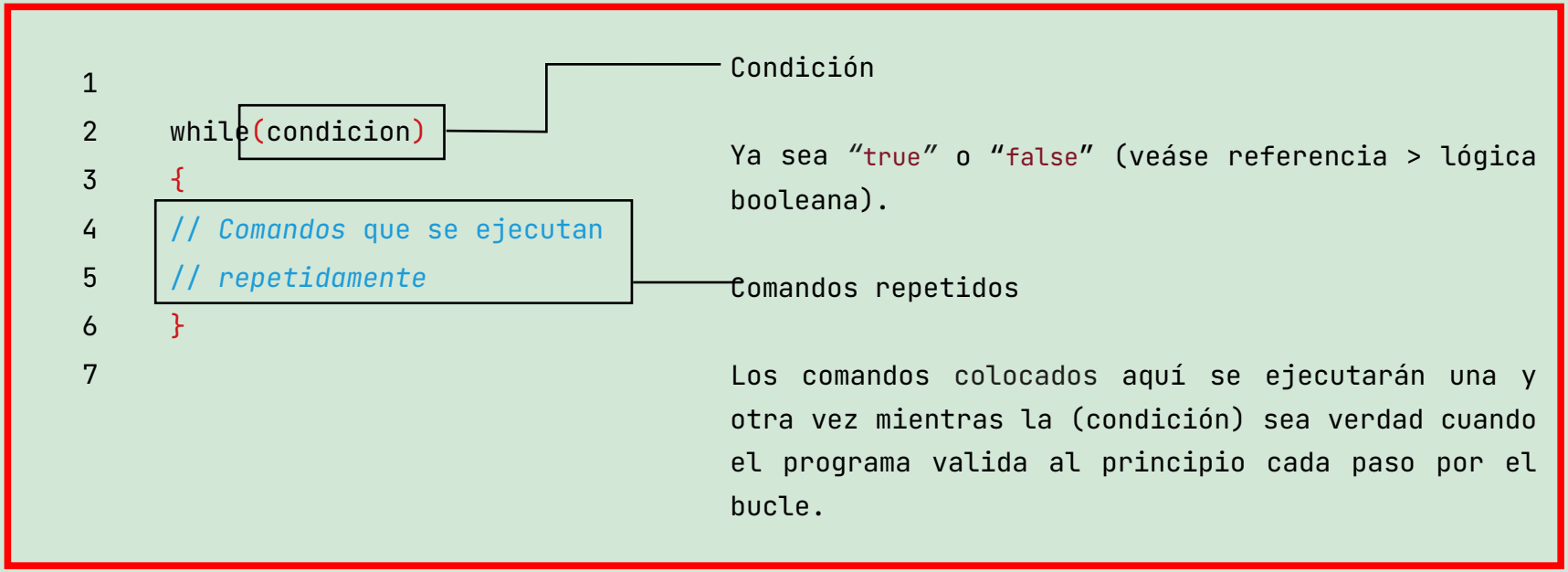
La variable es visible dentro de las funciones. La variable global puede ser usada y modificada dentro de las funciones

La variable es visible dentro de la tarea principal. La variable global puede ser usada y modificada dentro de las tareas, por ejemplo, task main.

```
2   int variable;
3   void cambiar valor(){
4       variable = 2000;
5   {
6
7   task main(){
8       variable = 1000;
9       cambiarValor();
10      wait1msec(variable);
11  }
```

Ciclo while

Un bucle `while` es una estructura dentro de ROBOTC® la cual permite que una porción de código se ejecute una y otra vez mientras una condición permanezca verdadera.



SOLUCIÓN DEL DESAFÍO

Ahora se te proporcionan tres herramientas que permiten resolver el problema.

1) Solución del problema con una serie de pasos (Tabla 1) explicados en idioma español, como en una receta de cocina.

2) Solución con pseudocódigo (Tabla 2), expresado también en idioma español, concreto.

3) Solución con algoritmo usando la notación de diagramas de flujo (Figura 2).

Asegúrate de estudiar completamente cada solución, ya que en el siguiente paso generarás el programa que resuelva el desafío. Relaciona estos conceptos con el proceso de ingeniería que se te ha presentado en la sección "Conceptos básicos" del Laboratorio 3 (véase página 43).

A continuación, las tres herramientas ya mencionadas.

Tabla 1.

SOLUCIÓN PARECIDA A UNA RECETA DE COCINA

(continúa en la pagina siguiente)

1. Inicio del programa.
2. Numero de vuelta es igual a 0.
3. Medir el lado más largo del rectángulo. Anótalo:

_____

4. Si ya va en la cuarta vuelta ir al paso 15.
5. Haz que el robot viaje del punto A al B a ½ potencia. Usa el cronómetro del celular y determina cuánto tiempo le toma al robot. Apaga el motor al llegar al lado contrario del marco. Considere que al llegar ahí, desde el punto de partida, el robot deberá dar vuelta a la izquierda, por lo que debe haber suficiente espacio para que el dispositivo no choque con el edificio. Anota tus observaciones y acciones: _____
_____.
6. Ahora haz que el robot gire a la izquierda (aplica ¼ de potencia del motor). ¿Cuánto tiempo debe aplicar fuerza a las ruedas para que dé un giro de 90°? Anota tu respuesta: _____.

Tabla 1.

SOLUCIÓN PARECIDA A UNA RECETA DE COCINA

(finaliza)

7. Haz que el robot viaje del punto B al C a ½ potencia. Usando los conceptos de matemáticas que ya conocese (regla de tres) determina cuánto tiempo necesitará el robot para alcanzar la siguiente esquina del edificio. Haz los ajustes necesarios a su programa. Anota tu primera respuesta: _____.
8. Haz que el robot gire a la izquierda (aplica ¼ de potencia del motor).
9. Haz que el robot viaje del punto C al D a ½ potencia.
10. Ahora haz que gire a la izquierda (aplica ¼ de potencia del motor).
11. Haz que el robot viaje del punto D al A a ½ potencia.
12. Ahora que gire el robot a la izquierda (aplica ¼ de potencia al motor).
13. Número de vuelta = Número de vuelta + 1.
14. Vé al paso 4.
15. Fin del programa.

Tabla 2.

PSEUDOCÓDIGO

PARA EL DESAFÍO.

```
// Desafío Robot Velador
// El robot da 3 vueltas alrededor del edificio rectangular
// por lo que el tiempo de recorrido en cada recta puede variar
// En cada esquina gira 90 grados
task main()
{
it NoVuelta = 0;
// Se darán 3 vueltas
While(NoVuelta > 4)
{
// Movimiento del punto A al punto B
avanzar el robot x segundos
// Giro en la esquina
girar a la Izquierda
// Movimiento del punto B al punto C
avanzar el robot x segundos
// Giro en la esquina
girar a la Izquierda
// Movimiento del punto C al punto D
avanzar el robot x segundos
// Giro en la esquina
girar a la Izquierda
// Movimiento del punto D al punto A
avanzar el robot x segundos
// Giro en la esquina
girar a la Izquierda
//Contador de vueltas
NoVuelta = NoVuelta + 1
```

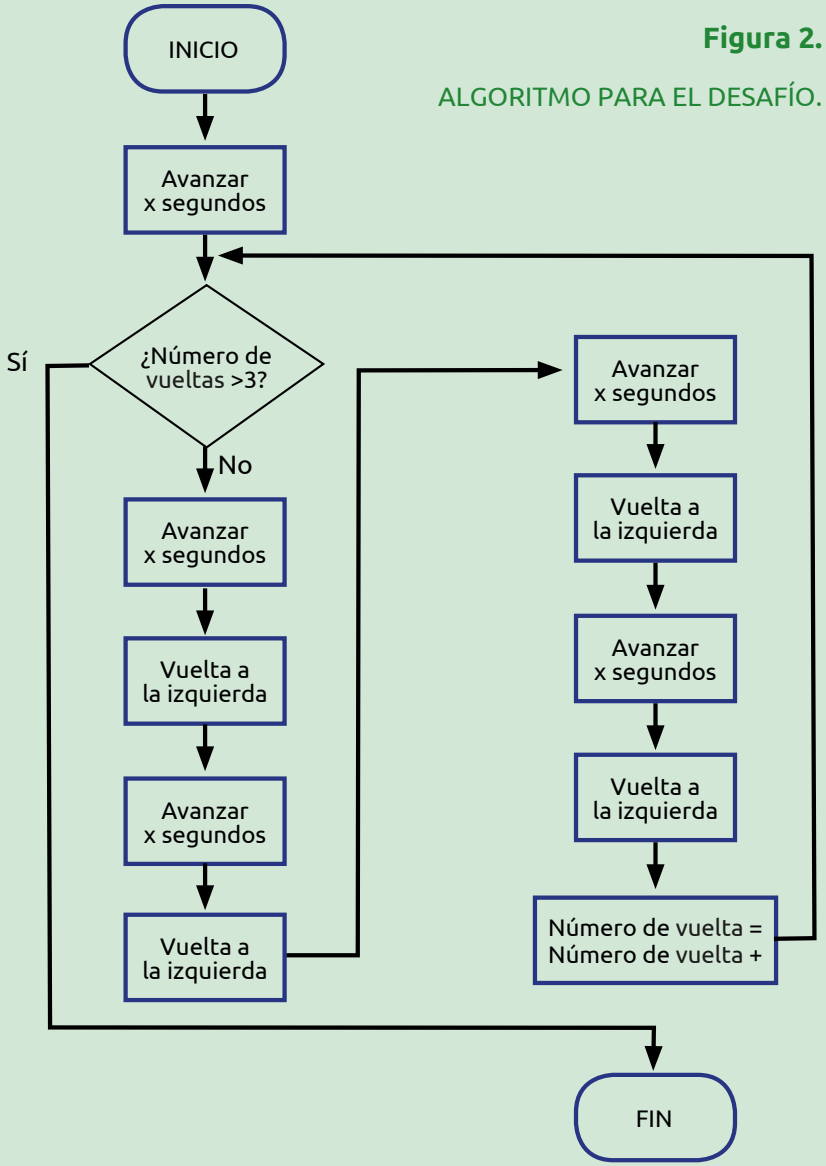


Figura 2.

ALGORITMO PARA EL DESAFÍO.

Anota en el recuadro el programa resultante con el que diste solución a cada paso de este desafío.

TRABAJO ADICIONAL

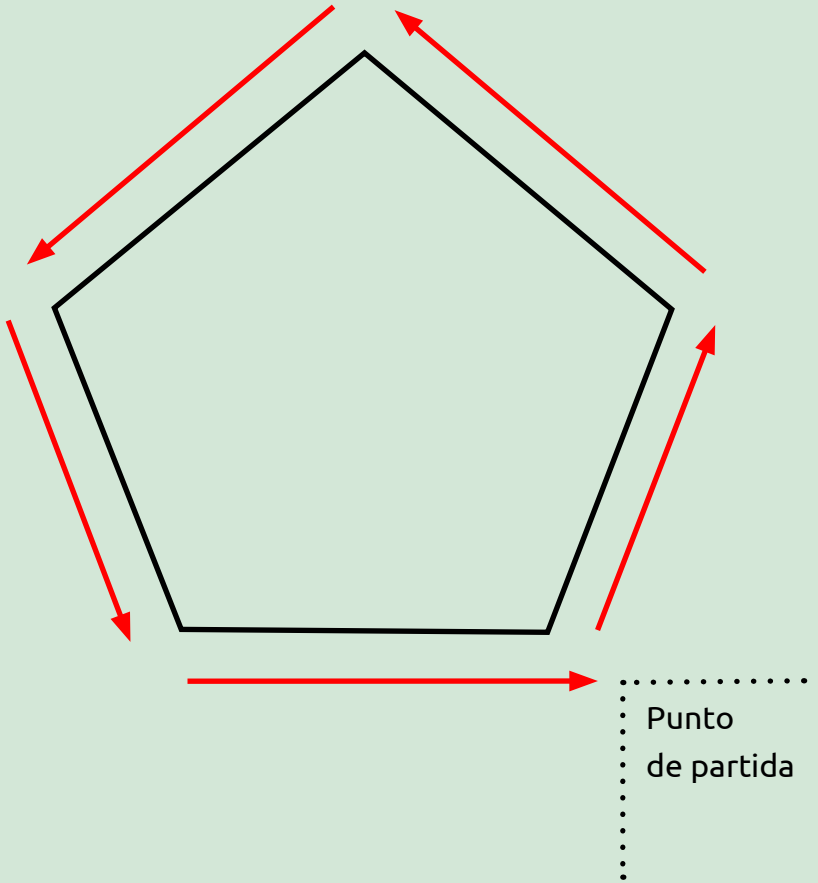
Los desafíos de esta sección debe resolverlos el estudiante. Es necesario crear un algoritmo y que el programa dé solución a cada desafío. La finalidad de este ejercicio es que construya programas cada vez más complejos y verificar sus avances.

1. El Ejército de los Estados Unidos posee un edificio conocido como el "Pentágono"; ahí se concentra el poder militar de dicho país. Si ACME Robotics tuviera que vigilar dicho complejo con el mismo robot, ¿qué cambios tendrías que hacer a su programa para que dé tres vueltas completas al edificio? Construye tu algoritmo y el programa en ROBOTC® para hacer el recorrido. Usa la maqueta desarrollada en el Laboratorio 3 para probar que el robot está funcionando adecuadamente. Observa la figura 5, para que tengas idea del recorrido del robot, y aunque no muestra las vueltas en la figura, tú considera que el robot girará al dar vuelta en las aristas.
2. Un proyecto adicional ha sido asignado a ACME Robotics. El edificio ahora tiene la forma de una cruz. ¿Qué

Figura 3.

RECORRIDO DEL ROBOT

PARA EL EDIFICIO DEL "PENTÁGONO".

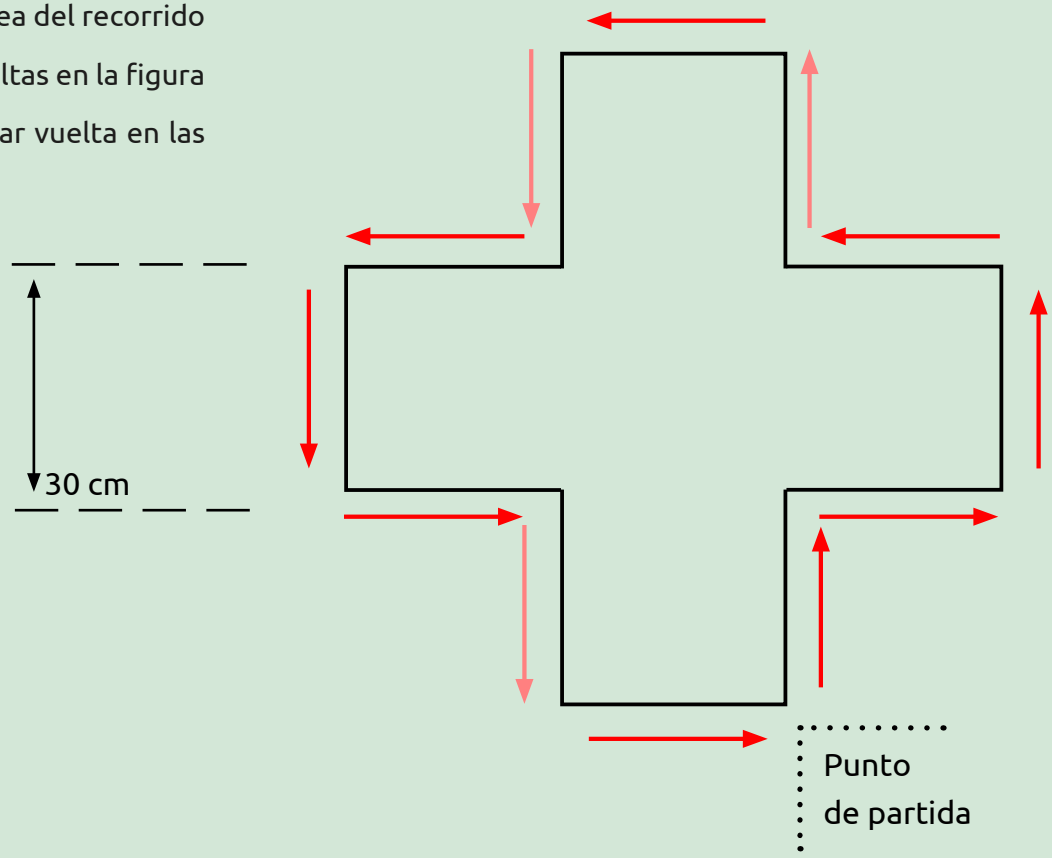


cambios tendrías que hacer a su programa para que dé tres vueltas completas al edificio? Construye tu algoritmo y el programa en ROBOTC® para hacer el recorrido. Usa la maqueta desarrollada en el Laboratorio 3 para probar que el robot funciona adecuadamente. Observa la figura 6, para que tengas idea del recorrido del robot, y aunque no muestra las vueltas en la figura considera que el robot debe girar al dar vuelta en las aristas.

Figura 4.

RECORRIDO DEL ROBOT

PARA EL EDIFICIO CON FORMA DE CRUZ.



Referencias

Robomatter Inc. (2016). ROBOTC for LEGO Mindstorms 4.X - Users Manual. Disponible para consulta en el sitio web: https://www.robotc.net/WebHelpMindstorms/index.htm#Resources/topics/ROBOTC_Interface/Overview.htm

Cairó, O (2005). Metodología de la programación. Algoritmos, diagramas de flujo y programas. Alfaomega. 3ª. edición.

Joyanes, L. (2008). Fundamentos de Programación. Algoritmos y estructura de datos, Mc Graw Hill.

ROBOT VELADOR, LABORATORIO 5

TRES VUELTAS COMPLETAS AL EDIFICIO, CON OBSTÁCULOS Y ARRANQUE MANUAL

OBJETIVOS DEL DESAFÍO QUE SE TE PLANTEA:

→ Desarrollar el pensamiento lógico necesario para diseñar un algoritmo usando SENSORES.

→ Construir un programa de computadora apropiado con ROBOTC® para implementar dicho algoritmo en un robot Lego.

→ Probar que el robot ejecuta las instrucciones adecuadamente y que da una vuelta completa a un edificio de tipo rectangular.

Luis Antonio Álvarez Oval, Christian Mauricio Castillo Estrada y Aron de la Cruz Vázquez, *Introducción a la robótica*, Sin fronteras, núm. 3, UNACH, México, 2022, 180 pp. ISBN: 978-607-561-130-3

www.unach.mx

Disponible en: www.editorial.unach.mx

INTRODUCCIÓN

El desafío que corresponde a este quinto laboratorio de prácticas, resulta ser una continuación al anterior; considerando como principal reto la evasión de obstáculos por parte del robot durante el recorrido de vigilancia que realiza. Por lo anterior, el estudiante deberá incorporar a la estructura del robot, un sensor de contacto que le permita identificar obstáculos localizados en la parte frontal en relación a la trayectoria que debe seguir.

En virtud de lo anterior, el estudiante debe seleccionar la lista de comandos que ofrece ROBOTC para la manipulación de sensores de contactos, los cuales emitirán valores que deben ser evaluados en tiempo de ejecución, y con base a ellos, tomar decisiones que le permitan al robot evadir el obstáculo a través de la implementación de estructuras de decisión simple.

ROBOTC® es un poderoso lenguaje de programación basado en C con un ambiente Windows para escribir y depurar programas, y es el único que ofrece un depurador exhaustivo en tiempo real. Es una solución multiplataforma que permite que los estudiantes aprendan el tipo de programación basada en C usada en educación avanzada y aplicaciones profesionales.

El robot que estaremos usando a lo largo de este curso tiene una configuración de dos ruedas, cada una con su propio motor, que serán los actuadores, y una rueda de arrastre omnidireccional; en el tema de sensores, para percibir el ambiente

que lo rodea, el robot dispone de sensor táctil, ultrasónico, y de luz/color.

Para este desafío necesitas los siguientes materiales:

- Marco de madera (para simular el edificio).
- Lenguaje de programación ROBOTC®.
- Robot LEGO® armado.
- Editor de diagramas DIA. Liga de descarga: http://dia-installer.de/
- Cinta aislante de color negro.

DESAFÍO POR RESOLVER

La compañía ACME Robotics ha ganado un proyecto para diseñar un robot programable que patrulle los alrededores de un edificio rectangular que almacena autos, por lo que ACME te ha contratado para desarrollar el programa que permita que ese robot prototipo pueda, de forma autónoma, "vigilar" los alrededores del edificio, desde las 20:00 horas de un día hasta las 6:00 horas del siguiente. Para este desafío hemos agregado "marcas" en las esquinas del edificio. También "sabemos" que algunos trabajadores, que instalaron equipos de aire acondicionado, han olvidado los "pallets" en donde se transportaban dichos equipos (se desconoce la ubicación exacta de los obstáculos). Sabemos que no hay obstáculo cuando el robot está girando, pero puede estar en cualquier otro lugar. El robot debe estar preparado para evadir dicho obstáculo.

También ha surgido un nuevo requerimiento, el robot debe estar "encendido" al inicio de la jornada, pero debe empezar a moverse hasta que un interruptor del robot sea presionado. La figura 1, explica paso a paso el problema, las líneas de color verde muestran en dónde se deben ubicar las marcas en las esquinas del edificio. Otro equipo de trabajo ya ha desarrollado el prototipo mecánico-electrónico del robot y está disponible para su uso.

El programa que debes construir en ROBOTC® debe contener las instrucciones para que el robot inicie el recorrido hasta que el usuario presione un interruptor, dé tres vueltas completas al edificio, eludiendo obstáculos. Puesto que se trata de un prototipo no es necesario que ejecute una rutina de vigilancia constante. El proceso de ingeniería implica que vayas construyendo la solución paso a paso, así que la rutina de vigilancia constante será desarrollada en laboratorios posteriores.

Es necesario que imprimas este documento ya que se deben responder las preguntas que se te hacen en la sección “Solución del desafío”.

CONCEPTOS BÁSICOS

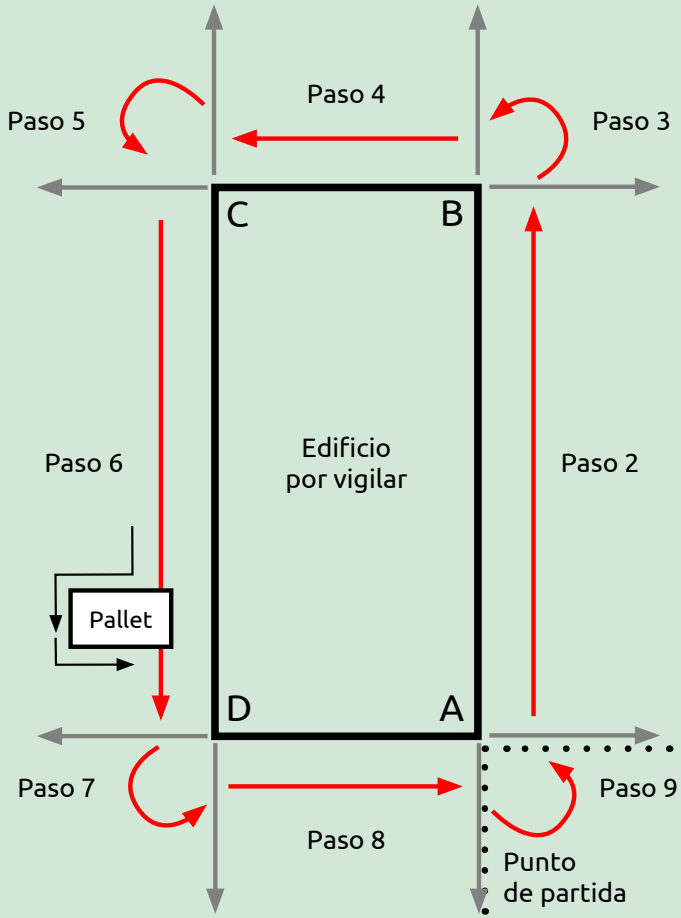
Aquí vas a encontrar material de apoyo para entender el objetivo primario de este documento. En general estos conceptos provienen de los documentos que el Laboratorio Nacional de Ingeniería Robótica de la Universidad Carnegie Mellon ha proporcionado en los cursos de robótica que imparte.

Percibir, planear y actuar (PPA)

Percibir, planear y actuar es el proceso más importante de los robots abreviado como PPA. Este concepto tiene la finalidad de resaltar las tres capacidades difíciles o críticas que deben ser consideras para que un robot pueda operar con eficacia y eficiencia:

Figura 1.

RECORRIDO DEL ROBOT VELADOR.



- **Percibe.** En esta actividad el robot requiere de la habilidad para percibir objetos a su alrededor, es decir, percibir la existencia de obstáculos o indicadores de trayectoria; esto nos lleva a plantear las siguientes preguntas: ¿qué información necesita conocer el robot en relación a su entorno? y ¿qué estrategia se debe implementar para recolectar la información?
- **Planea.** Es necesario generar una estrategia para ser implementada en el robot, y que le permita responder apropiadamente de acuerdo a los sucesos del entorno en donde se encuentre. Por lo anterior, se generan las siguientes preguntas: ¿tienes alguna estrategia?, ¿el robot determina la respuesta idónea con base a esa esa estrategia?
- **Actúa.** En esta última actividad el robot debe realizar acciones establecidas dentro el plan o estrategia; y surge el siguiente cuestionamiento: ¿el robot se ha construido para que haga lo que necesita hacer de acuerdo al algoritmo diseñado?

¿Dónde están S, P y A en este programa?

Tabla 1.

CÓDIGO PERCIBIR, PLANEAR Y ACTUAR.

```
1 task main(){
2     while (true) {
3           if(SensorValue(Toque) = = 0){
4                 // Mientras no se enciende el sensor
5                 motor[Izquierda] = 33;
6                 motor[Derecha]  = 33;
7           }
8           else{
9                 // Cuando choca, el sensor se activa, se activa reversa
10                motor[Izquierda] = -33;
11                motor[Derecha]   = -33;
12                wait1msec(1500);
13                // Gira a la izquierda, el sensor se desactiva
14                motor[Izquierda] = -33; motor[Derecha]  = 33;
15          }
16    }
17 } /* Fin del amin */
```

Percibe. El robot usa el sensor de contacto para descubrir o identificar que ha topado con algún objeto.

Planea. La estrategia para el robot es ir hacia delante, a menos que haya algo en su camino, y lo detectará usando el sensor de contacto. Si el sensor de contacto no es presionado, los

motores marcharán hacia delante; si ese sensor es presionado, el robot se mueve en reversa y luego girará para alejarse del obstáculo. Esto está capturado dentro del programa que ejecuta el robot, al tiempo que lee los datos del sensor y emplea los comandos de motor apropiados.

Actúa. El robot actúa moviendo sus motores en respuesta de los comandos dados, los motores hacen combinaciones para producir los movimientos hacia delante y giros apropiados.

Enunciados if-else

Un enunciado "`if-else`" es una manera de permitirle a tu computadora que tome una decisión. Con este comando el programa checará la `(condición)` y después ejecutará una de las dos partes del código, dependiendo si la condición es `true` o `false`.

Abajo está el pseudocódigo que da una idea general de un enunciado `IF-ELSE`

```
1    if(condicion){
2         // Si la condición es VERDADERA
3         //poner los comandos a ejecutar
4    }
5    else{
6         // Si la condición es FALSA
7         // poner los comandos a ejecutar
8    }
9
```

```
(condición) ya sea VERDADERA o FALSA
(véase Referencia > lógica booleana).

Comandos (true).

Los comandos que se encuentran aquí
serán ejecutados si la condición es
verdadera.

Comandos (false).

Los comandos que se encuentran aquí
serán ejecutados si la condición es
falsa.
```

Abajo, un ejemplo de programa que contiene un enunciado IF-ELSE

```
1 task main(){
2    while (true) {
3         if(SensorValue(Sonar) > 25){
4              motor[Izquierda] = 33;
5              motor[Derecha]   = 33;
6         }
7         else{
8              motor[Izquierda] = 0;
9              motor[Derecha]   = 0;
10        }
11   } /*Fin del while */
12 } /* Fin del main */
```

(Condición) True si el sensor lee más de 25. False si es lo contrario

Comandos (True). Estos comandos se ejecutan si la condición es verdadera. Se activan los motores.

Comandos (False). Estos comandos se ejecutan si la condición es falsa. Se apagan los motores.

Este enunciado `if-else` indica al robot que encienda ambos motores a toda potencia si el objeto más cercano al sensor ultrasónico está a más de 25 pulgadas de distancia. Si el sensor ultrasónico detecta un objeto a menos de 25 pulgadas, entonces la parte "`else`" del código será ejecutada y el robot dejará de moverse. El bucle `while` (`true`), que está más afuera, hace que el enunciado se ejecute una y otra vez para siempre.

Sensores

Los sensores suelen ser uno de los componentes más importantes de cualquier máquina a la que se considere o denomine *robot*; los cuales le proporcionan información relacionada al ambiente en el cual interactua el procesador central del robot. Estos componentes son una extensión del robot y en ocasiones simulan los sentidos del cuerpo humano, y resultan de vital importancia para que el robot decida la siguiente acción a ejecutar con

base a la información emitida por los sensores. En el kit LEGO EV3 Existen diferente sensores, a continuación, se describen algunos de ellos.

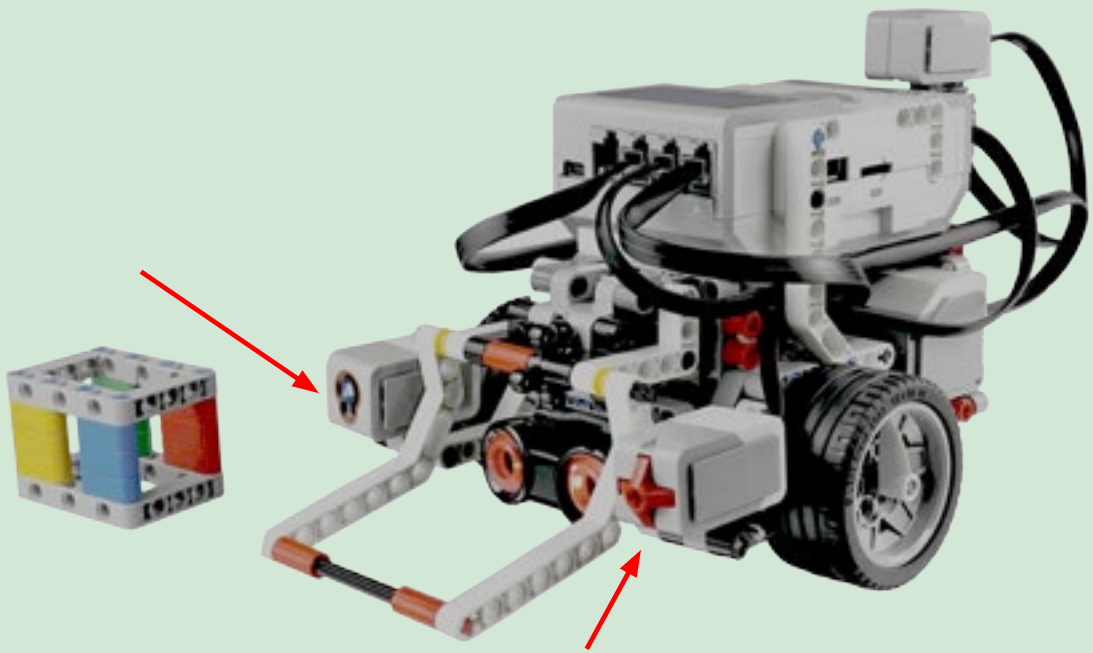
Sensor ultrasónico

El sensor ultrasónico ofrece exactamente esta capacidad. Usando el mismo principio físico de un murciélago o del sonar de un submarino, el sensor ultrasónico mide las distancias usando sonido. Manda una señal de sonido, después espera a escuchar el eco del sonido sobre un objeto sólido alrededor. Al medir en cuánto tiempo el sonido rebota, el sensor puede calcular la distancia que debe ser recorrida, y por lo tanto, qué tan lejos estaba el objeto cuando fue reflejado. El programa de la página previa muestra el uso del sensor ultrasónico en un programa. Mientras que la figura 2 identifica cada sensor instalado en el robot para que el programador los conozca.

Este sensor digital genera ondas de sonido de alta frecuencia y lee el tiempo en que regresa el eco para

Figura 2.

LOS SENSORES INSTALADOS EN EL ROBOT LEGO.



detectar y medir la distancia de objetos, de la misma manera en que lo hacen los murciélagos. Es capaz de enviar ondas de sonido individuales lo que permite que funcione como un sonar, también puede funcionar como un micrófono esperando un sonido para iniciar la ejecución de un programa.

Con la ayuda del sensor ultrasónico, los estudiantes tienen los elementos necesarios para diseñar un modelo de simulación de un sistema de monitorización de tráfico, el cual permite calcular distancias entre autos. Así también, los estudiantes descubrirán el impacto significativo que genera el uso de este tipo de sensores aplicados a la vida cotidiana.

Aspectos por considerar:

- Precisión de +/- 1 cm.
- Rango de medida de distancia (1 - 250 cm).
- Iluminación frontal es constante mientras se realiza la lectura (escucha).
- Retorna un valor de verdadero o true al identificar otro sonido ultrasónico.

Sensor de tacto

El sensor analógico de contacto que posee el Robot LEGO Mindstorms EV3, resulta ser un elemento simple, pero al mismo tiempo excepcional para detectar las acciones de pulsar o soltar un botón frontal; así también, proporciona información acerca del número de presiones simples y múltiples que ha detectado. Con este sensor, el robot adquiere una nueva funcionalidad que le permitirán a los estudiantes construir modelos de robots con capacidad de dar solución a desafíos de laberintos o crear interfaces de interacción con diversos dispositivos tales como: instrumentos musicales, teclados de computadoras, entre otros.

```
1     task main(){
2           int velocidad = 50;
3           while (true) {
4           if(SensorValue(Toque) = = 1){
5                 velocidad = velocidad * -1;
6           else
7                 velocidad = 50;
8           } /* Fin del IF */
9           motor[Izquierda] = velocidad;
10          motor[Derecha] = velocidad;
11    }
12    } /* Fin del main */
```

Variable para la velocidad inicial de los motores.

Si el sensor está activo(presionado) se invierte la velociad del motor (reversa).

Cuando se deja de presionar el sensor, el motor regresa a velocidad normal (hacia adelante).

Los motores avanzan de acuerdo al valor almacenado en la variable velocidad.

Sensor de luz

Este sensor digital de color es capaz de distinguir hasta ocho colores. Otra de sus funciones es ocupar el rol de sensor de luz, es decir, permite la identificación de diversas intensidades de la misma. Por lo anterior, existe la posibilidad que los estudiantes puedan incorporar este tipo de sensor al robot, con la finalidad de construir modelos de clasificación de objetos con base al color o robots seguidores de líneas. Así también, usar estos sensores ayuda al proceso de aprendizaje, permitiendo al estudiante experimentar con el reflejo derivado de la captura de luz probando con diferentes colores, y adquirir experiencia con una tecnología que se utiliza ampliamente en industrias.

El sensor de luz tiene un rango de estados entre 0 y 100. Mientras más bajo sea el número, el color que "ve" es más oscuro; mientras más alto el número, el color que "ve" es más claro. Por esto debe ser calibrado para la cantidad de luz en donde esté operando, usando como referencia el color que se quiere detectar.

- Capaz de detectar ocho colores.
- Frecuencia de muestreo de 1 kHz

```
2   task main(){
3         while (true) {
4               while(SensorValue(SensorLuz) < 45){
5                     motor[Izquierda] = 65;
6                     motor[Derecha]   = 0;
7         }
8         while(SensorValue(SensorLuz) > = 45){
9                     motor[Izquierda] = 0;
10                    motor[Derecha]   = 65;
11        } /* Fin del IF */
12  }
13  } /* Fin del main */
```

Si la lectura del sensor de luz es inferior a 45,lo que corresponde a un color que tiende al negro, el robot da vueltas hacia la derecha.

Si la lectura del sensor de luz es superior a 45, lo que corresponde a un color que tiende al blanco, el robot da vuelta hacia la izquierda.

SOLUCIÓN DEL DESAFÍO

Con la cinta aislante agrega las marcas en las esquinas del edificio, tal y como puedes verlo en el diagrama de la figura 1. Ahora el robot no se moverá con tiempo, lo hará hasta que el robot detecte el cambio de color del fondo blanco al color negro de la cinta aislante. Incluso las vueltas pueden ser controladas hasta que el sensor detecte el cambio de color.

Ahora se te proporcionan tres herramientas que te permitirán resolver el problema.

A) Solución del problema, como una receta de cocina (serie de pasos, observa la tabla 2) explicada en idioma español.

B) Solución con pseudocódigo (Tabla 3), expresado también en idioma español concreto.

C) Solución con algoritmo usando la notación de diagramas de flujo (Figura 3).

Asegúrate de estudiar completamente cada solución, ya el siguiente paso es que generes un programa que resuelva el desafío y relaciones estos conceptos con el proceso de ingeniería que se te ha presentado en la sección "Conceptos básicos" del Laboratorio 3.

A continuación, las tres herramientas ya mencionadas.

Tabla 2.

SOLUCIÓN PARECIDA

A UNA RECETA DE COCINA.

(continúa en la pagina siguiente)

```
1. Inicio del programa
2. Leer el estado del interruptor
3. Si el interruptor está apagado vaya al paso 2
4. Numero de vuelta es igual a 0
5. Si ya va en la cuarta vuelta ir al paso 64.
6. -- Viaje del punto A al B.
7. Ciclar mientras el sensor no “vea” color negro
8.    Avanzar a media potencia
9.    Si el robot se acerca a un obstáculo al menos a 10 cm
10.         Girar a la derecha
11.         Avanzar
12.         Girar a la Izquierda
13.         Avanzar
14.         Girar a la Izquierda
15.         Avanzar
16.         Girar a la Derecha
17.      Fin Si
18. Fin de Ciclo. Vaya al paso 7
19. Ahora gire el robot a la izquierda (aplique media potencia al motor del robot).
20. -- Viaje del punto B al C.
21. Ciclar mientras el sensor no vea color negro
22.     Avanzar a media potencia
23.     Si el robot se acerca a un obstáculo al menos a 10 cm
24.         Girar a la derecha
25.         Avanzar
26.         Girar a la Izquierda
27.         Avanzar
28.         Girar a la Izquierda
29.         Avanzar
30.         Girar a la Derecha
31.     Fin Si
32. Fin de Ciclo. Vaya al paso 21
```

```
33. Ahora gire el robot a la izquierda (aplique media potencia al motor del robot).
34. -- Viaje del punto C al D
35. Ciclar mientras el sensor no vea color negro
36.     Avanzar a media potencia
37.     Si el robot se acerca a un obstáculo al menos a 10 cm
38.         Girar a la derecha
39.         Avanzar
40.         Girar a la Izquierda
41.         Avanzar
42.         Girar a la Izquierda
43.         Avanzar
44.         Girar a la Derecha
45.     Fin Si
46. Fin de Ciclo. Vaya al paso 35
47. Ahora gire el robot a la izquierda (aplique media potencia al motor del robot)
48. -- Viaje del punto D al A
49. Ciclar mientras el sensor no vea color negro
50.     Avanzar a media potencia
51.     Si el robot se acerca a un obstáculo al menos a 10 cm
52.         Girar a la derecha
53.         Avanzar
54.         Girar a la Izquierda
55.         Avanzar
56.         Girar a la Izquierda
57.         Avanzar
58.         Girar a la Derecha
59.     Fin Si
60. Fin de Ciclo. Vaya al paso 49
61. Ahora gire el robot a la izquierda (aplique media potencia al motor del robot).
62. Numero de vuelta = Numero de vuelta + 1.
63. Vaya al paso 5.
64. Fin del Programa.
```

Tabla 2.

SOLUCIÓN PARECIDA

A UNA RECETA DE COCINA.

(finaliza)

Tabla 3.

PSEUDOCÓDIGO

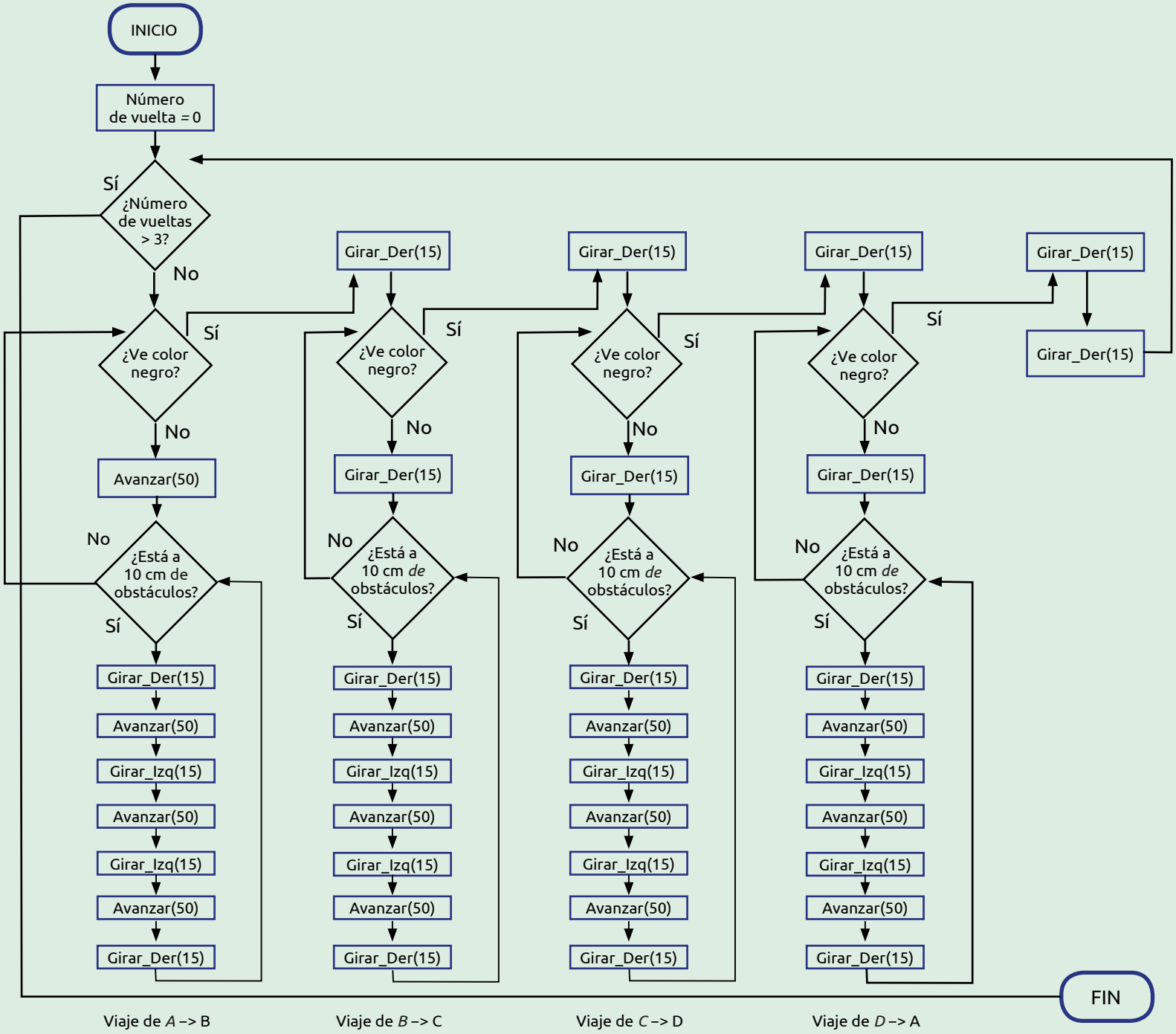
PARA SOLUCIONAR EL DESAFÍO.

```
// Desafío Robot Velador
// El robot da 3 vueltas alrededora el edificio retangular
// por lo que el tiempo de recorrido en cada recta puede variar
// En cada esquina 90 grados
task main ()(
       int NoVuelta = 0;
       int valorInt = 0
       // Evaluando el interruptor
       while (valorInt = )(
            if SensorValue( interruptor ) = 1{
            valorInt = 1;
            }
  }
       // Solo debe dar tres vueltas
       while(NoVuelta < 4)
       {
            // Movimiento del punto A al punto B
            While (sensor no vea color Nero)(
                 Avanzar(50);
                 // Librando el pallet
                 IF SensorValue(sonarSensor) < 10 {
                   GirarDerecha(15);
                   Avanzar();
                   GirarIzq()
                   Avanzar();
                   GirarDerecha();
                   Avanzar(50);
                   GirarDerecha();
                 }
            }
            //Giro en la esquina
            GirarIzq(15);
            // Movimiento del punto B al punto C
            While (sensor no ea color Negro){
                 Avanzar(50);
                 // Librando el pallet
                 IF SensorValue(sonarSensor) < 10 {
                   GirarDerecha (15);
                   Avanzar();
                   GirarIzq();
                   Avanzar(50);
                   GirarIzq();
                   Avanzar(50);
                   GirarDerecha();
                 }
            }
            // Giro en la esquina
            GirarIzq(15);
            // Movimiento del punto C al punto D
            While (sensor no vea color Negro){
                 Avanzar(50);
                 // Librando al pallet
                 IF SensorValue(sonarSensor) < 10 {
                   GirarDerecha(15);
                   Avanzar();
                   GirarIzq();
                   Avanzar();
                   GirarIzq();
                   Avanzar(50);
                   GirarDerecha();
                 }
            }
            // Giro en la esquina
            GirarIzq(15);
            // Movimiento del punto D al punto A
            While (sensor no vea color Negro)(
                 Avanzar(50);
                 // Librando el pallet
                 IF SensorValue(sonarSensor) < 10 {
                   GirarDerecha(15);
                   Avanzar();
                   Girar()
                   Avanzar()
                   GirarIzq();
                   GirarDerecha();
                   GirarDerecha();
                 }
            }
            // Giro en la esquina
            GirarIzq(15);
            // Contador de vueltas
       NoVuelta = NoVuelta + 1
  }
}
```

Figura 3.

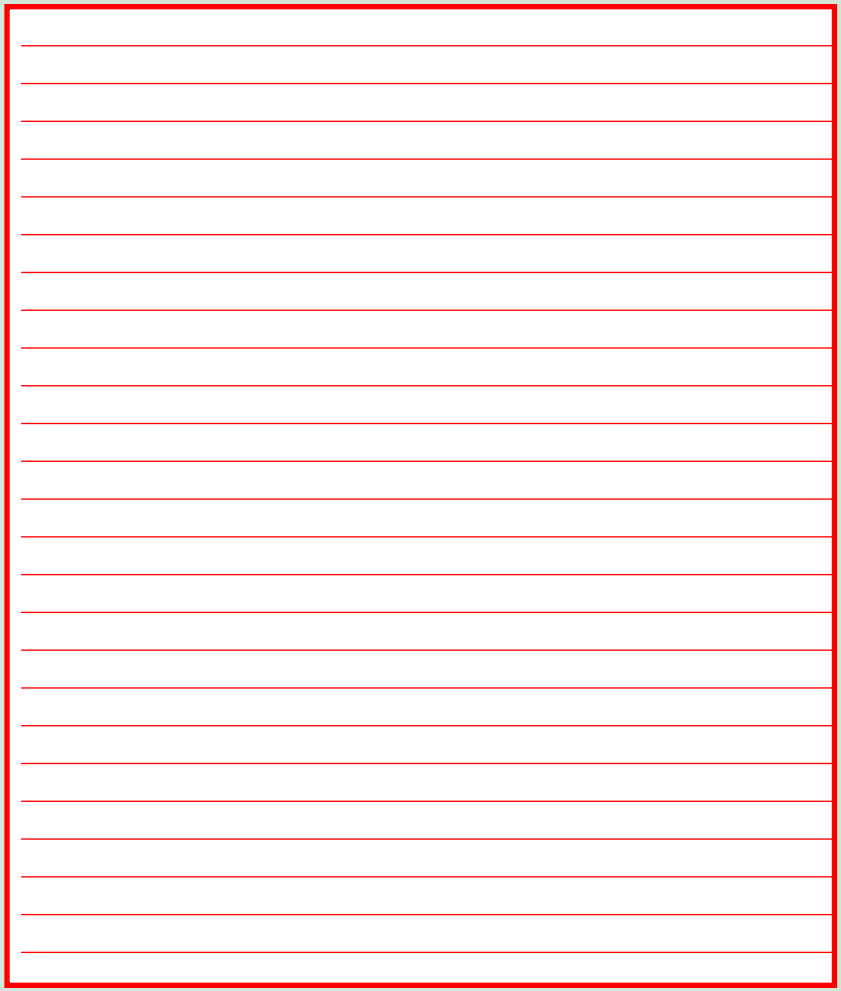
DIAGRAMA DE FLUJO

PARA EL DESAFÍO.



CONTENIDO GENERAL

Anota en el recuadro el programa resultante con el que diste solución a los pasos de este desafío.



TRABAJO ADICIONAL

Desarrolla el programa para que el giro del robot se ejecute al detectar que ha llegado a la esquina del edificio y hasta que vuelva a detectar la segunda marca que se encuentra al doblar la esquina. Integra ese programa al código desarrollado en este laboratorio.

Referencias

Robomatter Inc. (2016). ROBOTC for LEGO Mindstorms 4.X - Users Manual. Disponible para consulta en el sitio web: https://www.robotc.net/WebHelpMindstorms/index.htm#Resources/topics/ROBOTC_Interface/Overview.htm

Carnegie Mellon University (2020). ROBOTC for TETRIX & LEGO® MINDSTORMS.https://www.cmu.edu/roboticsacademy/roboticscurriculum/Lego%20Curriculum/ROBOTC-Tetrix_Mindstorms.html

ROBOT VELADOR, LABORATORIO 6

TRES VUELTAS COMPLETAS AL EDIFICIO, CON OBSTÁCULOS Y FUNCIONES

OBJETIVOS DEL DESAFÍO QUE SE TE PLANTEA:

→ Desarrollar el pensamiento lógico necesario para diseñar un algoritmo usando FUNCIONES.

→ Construir un programa de computadora apropiado con ROBOTC® para implementar dicho algoritmo en un robot Lego.

→ Probar que el robot ejecuta las instrucciones adecuadamente y que da tres vueltas completas al modelo de un edificio de forma rectangular.

Luis Antonio Álvarez Oval, Christian Mauricio Castillo Estrada y Aron de la Cruz Vázquez, *Introducción a la robótica*, Sin fronteras, núm. 3, UNACH, México, 2022, 180 pp. ISBN: 978-607-561-130-3

www.unach.mx

Disponible en: www.editorial.unach.mx

INTRODUCCIÓN

Con la finalidad de que el estudiante aprenda el diseño e implementación de funciones, o procedimientos dentro del programa, usando el lenguaje de programación robotc; en este sexto laboratorio el desafío consiste en optimizar el código a través del uso de funciones que ejecuten las tareas que permitan dar cumplimiento al objetivo principal.

Para dar solución al desafío planteado, es importante identificar las tareas más relevantes y transformarlas en funciones; posteriormente, el estudiante debe usar los comandos necesarios para mandar llamar a las funciones, reflexionando acerca del ahorro y optimización del código. Finalmente, es necesario realizar las pruebas de funcionamiento.

ROBOTC® y el robot nos permiten trasladar la programación del mundo abstracto dentro de la computadora al mundo real, en donde cada orden que se escribe puede ser percibida en la actividad que desarrolla el robot. El estudiante debe comprender la programación de computadoras como una actividad lógica sencilla de entender y aplicar; esa comprensión le permitirá desarrollar conceptos mucho más abstractos, como la Comunicación entre procesos remotos, Programación orientada a objetos, Programación web, etcétera.

El robot que estaremos usando a lo largo de este curso tiene una configuración de dos ruedas, cada una con su propio motor, que serán los actuadores, y una rueda de arrastre omnidireccional; en el tema de sensores, para percibir el ambiente que lo rodea, el robot dispone de sensores táctil, ultrasónico, y de luz/color.

Para este desafío necesita los siguientes materiales:

- Marco de madera (para simular el edificio).
- Lenguaje de programación ROBOTC®.
- Robot LEGO® armado.
- Editor de diagramas DIA. Liga de descarga: http://dia-installer.de/
- Cinta aislante de color negro.

DESAFÍO POR RESOLVER

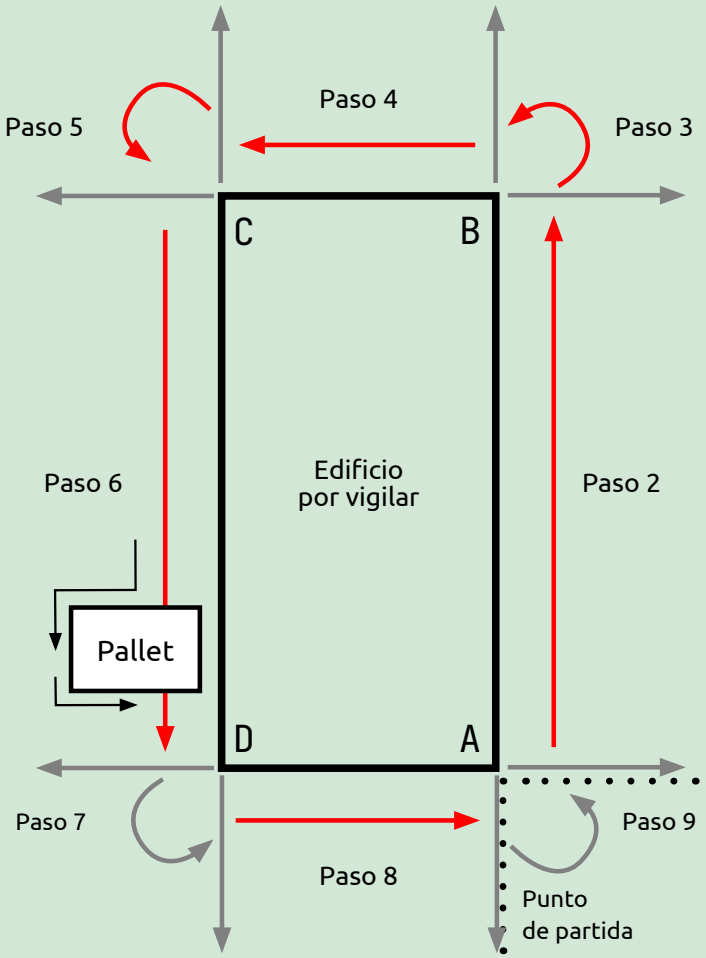
La compañía ACME Robotics ha ganado un proyecto para diseñar un robot programable que patrulle los alrededores de un edificio rectangular que almacena autos. ACME te ha contratado para desarrollar el programa para que ese robot prototipo pueda, de forma autónoma, "vigilar" los alrededores del edificio desde las 20:00 horas de un día hasta las 6:00 horas del siguiente. Para este desafío hemos agregado "marcas" en las esquinas del edificio. También debemos suponer que algunos trabajadores que instalaron equipos de aire acondicionado han olvidado los "pallets" en donde se transportaban dichos equipos (se desconoce la ubicación exacta del obstáculo). Una condición con la que se cuenta es que el obstáculo no estorbará los giros del robot, y que sí puede estorbarlo en cualquier otro lugar. El robot debe estar preparado para evadir dicho obstáculo. También ha surgido un nuevo requerimiento, el robot debe estar "encendido" al inicio de la jornada, pero debe empezar a moverse hasta que un interruptor del robot sea presionado. La figura 1 explica paso a paso el problema, las líneas de color verde muestran en dónde se deben ubicar las marcas en las esquinas del edificio. Otro equipo de trabajo ya ha desarrollado el prototipo mecánico-electrónico del robot y está disponible para su uso.

El programa que debes construir en ROBOTC® contendrá las instrucciones para que se mueva hasta que el usuario presione un interruptor e inicie el recorrido, que será de tres vueltas completas al edificio, evadiendo obstáculos. Puesto que se trata de un prototipo no es necesario que ejecute una rutina de vigilancia constante. Se aplicará el concepto "crear funciones" para completar el desafío. El proceso de ingeniería implica que vayas construyendo la solución paso a paso, así que la rutina de vigilancia constante será desarrollada en laboratorios posteriores.

Es necesario que imprimas este documento ya que debes responder las preguntas que se te hacen en la sección "Solución del desafío"

Figura 1.

RECORRIDO DEL ROBOT VELADOR.



CONCEPTOS BÁSICOS

Aquí encontrarás material de apoyo para entender el objetivo primario de este documento. Cabe señalar que ciertas ideas plasmadas en este libro se relacionan con los apuntes en los manuales de usuario y guías de aprendizaje redactadas por el Laboratorio Nacional de Ingeniería Robótica de la Universidad Carnegie Mellon, que se encuentran para libre consulta de los usuarios y utilizados como referencia en los diversos cursos de robótica que imparten (Carnegie Mellon University 2020).

Funciones

Es importante comprender el concepto de función o procedimiento, expresándose como el conjunto de enunciados o instrucciones que son ejecutadas en una sola unidad, por ejemplo, task main(). Por otra parte, es necesario considerar que cada función representa una tarea o comportamiento específico; y en la mayoría de las ocasiones retornará un valor como resultado.Las funciones ofrecen varias ventajas sobre codificación básica paso por paso.

Las funciones ofrecen diversas ventajas, por mencionar algunas:

- Ahorro de tiempo permitiendo que tareas comunes sean escritas como funciones, y después sean ejecutadas juntas como un solo enunciado.
- La separación de tareas o comportamientos en diferentes funciones permite el código cumpla con los objetivos del plan establecido.
- Diferentes tareas relacionadas pueden ser controladas con una sola función.
- Las funciones puede recibir parámetros, los cuales permiten controlar y modificar el resultado de una tarea.

Utilizando las funciones

Para poder usar las funciones dentro de un programa codificado con el lenguaje ROBOTC, deben ser declaradas, desarrolladas y finalmente ejecutadas de forma independiente. Se considera que una función es creada al momento de declararla, y ejecutada al momento de mandar llamarla.

1. Declara *la* función con el tipo de dato de retorno void: En el código de ejemplo se escribe el tipo de dato, y después el nombre que tendrá la función. Se recomienda elegir un nombre adecuado para la función, mismo que represente el comportamiento o la tarea que ejecutará. Dentro de los [corchetes] de la función escribe los comandos de la misma manera en que lo harías normalmente. Cuando la función es invocada, ejecutará las líneas entre los corchetes en orden, tal y como task main lo hace con el código entre sus propios corchetes.

2. Invoca *tu* función: Acto posterior a la declaración de una función, esta es identificada o representada como un nuevo comando en el lenguaje de ROBOTC. Por tal motivo, para invocar o llamar a la función, es necesario escribir el nombre incluyendo paréntesis con sus respectivos valores de los parámetros y finalizando por punto y coma.

```
22    void girarIzquierda90()[
23          nMotorEncoder[Derecha] = 0;
24          while [nMotorEncoder[Derecha] < = 170)[
25                motor[Izquierda] = -31;
26                motor[derecha] = 31;
27    ]
28    motor[Izquierda] = 0;
29    motor[derecha] = ;
30    ]
31
32    task main()[
33          // Avanzar de punta A → B
34    avanzarConGrados(60);
35    girarIzquierda90();
```

1. Declara un parámetro.
Se recomienda realizar la declaración de un parámetro del mismo modo que una variable, es decir, en primera instancia especificar el tipo de dato, después del nombre del parámetro, todo ello dentro de los paréntesis de la función.

2. Uso de parámetros.
El valor del parámetro se comporta como un "guarda valores". Cualquier valor que se asigne al parámetro cuando la función es invocada aparece aquí.

3. Invocar la función con parámetros.
Cuando es invocada, se debe proveer un valor dentro de los paréntesis para que tome el lugar del parámetro dentro de la función.

```
12   void GirarIzq(int ptiempo)[
13        motor[motorB] = -50;
14        motor[motorC] = 50;
15        wait1msec[ptiempo];
16   ]
17
18   task main()[
19        int NoVuelta = 0;
20        // Se darán 3 vueltas
21        While(NoVuelta < 4)
22        [
23        Avanzar(50);
24        GirarIzq(15);
25        Avanzar(50);
```

Funciones avanzadas

Parámetros

Los parámetros son un medio para el paso de valores o datos a una función; son representados por variables locales, es decir, su ámbito de validez es el cuerpo de la función, y permiten que una función se ejecute de distintas maneras. Cabe señalar que los valores se asignan a cada uno de los parámetros que posee la función, en el momento en que es invocada, respetando el orden y el tipo de dato de cada uno.

Sustitución

En el código ejemplificado, se observan dos flechas que indican el camino que sigue el valor asignado al parámetro; partiendo desde la posición donde fue asignado, hasta donde su valor es sustituido en la función.

```
12   void Girar(Izq int ptiempo)
13        motor[motorB] = -50
14        motor[motorC[ = 50;
15   wait1msec[ptiempo[;
16   ]
17
18   task main()[
19        int NoVuelta = 0;
20   // Se darán 3 vueltas
21   while(NoVuelta < 4)
22   [
23        Avanzar(50);
24        GirarIzq(15);
25        Avanzar(50);
```

La función será ejecutada de acuerdo al orden en que fueron invocadas en el código.

```
1
2     motor[motorB] = -50;
3     motor[motorC] = 50;
4     wait1msec[15];
5
```

Valores de retorno

En relación a los tipos de datos de retorno utilizados al momento de declarar una función, es importante señalar que el tipo void es usualmente elegido; sin embargo, el estudiante debe analizar el tipo de información o valor que desea obtener al ejecutar la función; por ejemplo, si la función efectúa un cálculo matemático entonces se obtendrá como valor de salida un dato de tipo entero, en esa situación se debe utilizar la palabra int.

1. Declara el tipo de retorno.

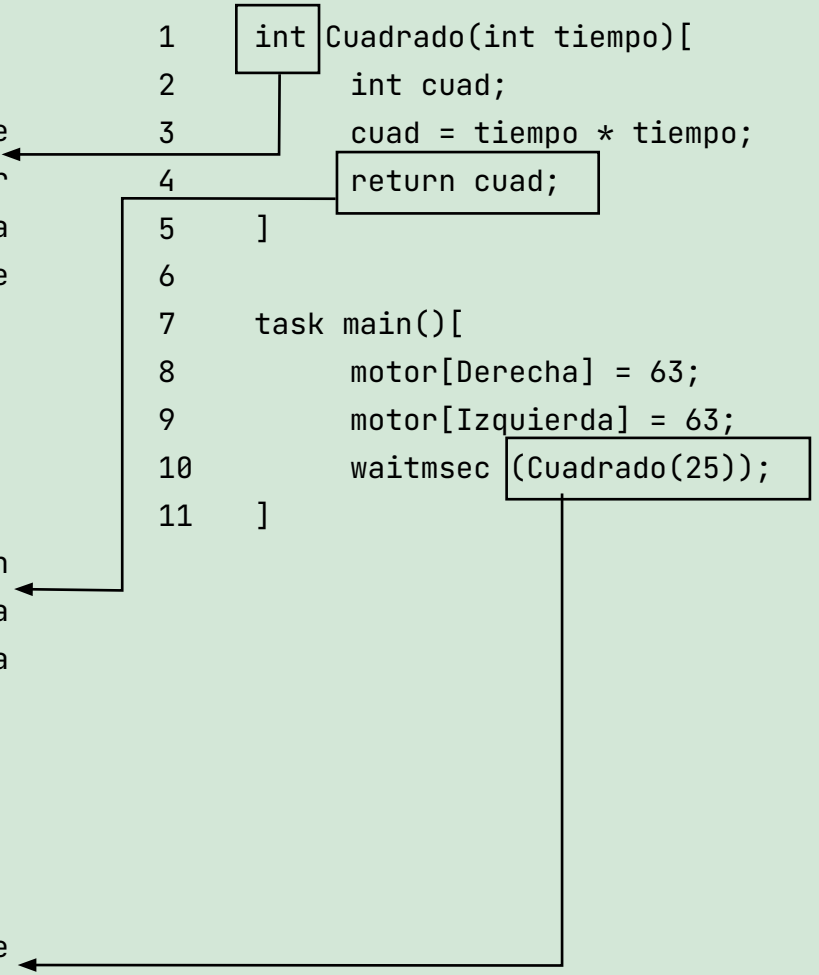
Como la función dará un valor de retorno al final, ésta debe ser declarada con un tipo que no sea “void”, indicando qué tipo de valor dará.

2. Valor de retorno.

La función debe “devolver” un valor. En este caso, significa devolver el resultado de la fórmula, el cuadro del parámetro.

3. el nombramiento
de funciones es un valor.

El nombramiento de funciones se vuelve un valor en la parte del programa que la nombra. Aquí está actuando como un valor entero por el comando de wait1Msec.

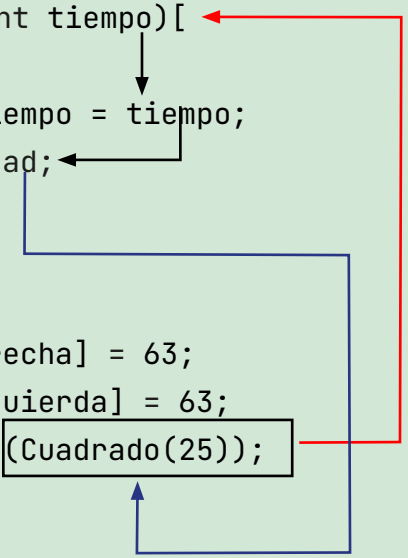


Substitución

El código de ejemplo de la función denominada Cuadrado, posee una serie de flechas que indican la trayectoria del valor de salida cuando es retornado y regresa hasta la función principal.

Para este ejemplo el valor de 25 es usado como un intervalo de tiempo dentro de la función, y la salida que genere la función depende de ese parámetro.

```
1    int Cuadrado(int tiempo)[
2          int cuad;
3          cuad = tiempo = tiempo;
4          return cuad;
5    ]
6
7    task main()[
8          motor[Derecha] = 63;
9          motor[Izquierda] = 63;
10         wait1msec(Cuadrado(25));
11   ]
```



```
1
2    wait1msec(Cuadrado(625));
3
```

Substitución

Lego EV3 Encoder (sensor de rotación)

Los motores que permiten que el robot LEGO® se desplace poseen un sensor de rotación de su eje (encoder), (Figura 2). En realidad, miden en grados el movimiento de las ruedas, por consiguiente, miden el desplazamiento del robot. El giro de las llantas es lo que impulsa el robot hacia adelante, la superficie exterior de las llantas genera una gran cantidad de fricción al rozar con el suelo. Si consideramos que la superficie exterior de la llanta es en realidad el perímetro de un círculo (Figura 3), es posible hacer predicciones correctas acerca de la distancia que el robot recorre con cada giro de 360° de la llanta. El sensor entrega mediciones en incrementos de 4° y registrará 90 lecturas por cada rotación completa de la llanta.

Que seas capaz de medir directamente los giros de las llantas te permite dictar el movimiento del robot en términos de distancia más que simplemente controlar el tiempo en el que el robot debe moverse. Este es un método efectivo ya que no es afectado por factores como la velocidad o la potencia de la batería. Medir los grados de giro resulta en una mejor consistencia del movimiento lineal del robot. Vea el programa 1 de este laboratorio para una mejor comprensión del funcionamiento del sensor de rotación.

Figura 2.

SENSORES DEL ROBOT VELADOR.

Figura 3.

RUEDA DEL ROBOT VELADOR.

```
task main()
[
    int lectura = 0;
    int grados = 0;
    // Inicializando el encoder del motor izquierdo
    nMotorencoder[Izquierda] = 0;
    // Evaluando los grados recorridos por el motor
    while (nMotorEncoder[Izquierda] < grados)[
      displayBigTextLine(3,"Lectura = %d", lectura);
      motor[Izquierda] = 33;
      motor[derecha] = 33;
      lectura = nMotorEncoder[Izquierda];
]
// apagando los motores cuando se alcanza la meta
motor[Izquierda] = 0;
motor[derecha] = 0;
]
```

SOLUCIÓN DEL DESAFÍO

Con la cinta aislante agrega las marcas en las esquinas del edificio, tal y como se ven en la figura 1. Ahora el robot no se moverá con tiempo; lo hará hasta que detecte el cambio de color del fondo blanco al color negro de la cinta de aislar. Incluso las vueltas pueden controlarse hasta que el sensor detecte el cambio de color.

Ahora se te proporcionan tres herramientas que te permitirán resolver el problema.

a) Solución del problema, como una receta de cocina (serie de pasos, observa la Tabla 1), explicada en idioma español.

b) Solución con pseudocódigo (Tabla 2), también expresado en idioma español.

c) Solución con algoritmo usando la notación de diagramas de flujo (Figura 4).

Asegúrate de estudiar completamente cada solución, ya que el siguiente paso es que generes el programa que resuelva el desafío. Relaciona estos conceptos con el Proceso de ingeniería que se te ha presentado en la sección "Conceptos básicos" del Laboratorio 3. A continuación, las tres herramientas ya mencionadas.

Tabla 1.

SOLUCIÓN PARECIDA

A UNA RECETA DE COCINA.

```
1. Inicio del programa
2. Leer el estado del interruptor
3. Si el interruptor está apagado vaya al paso 2
4. Número de vuelta es igual a 0
5. Si ya va en la cuarta vuelta ir al paso 21
6. -- Viaje del punto A al B.
7. Ir a función AvanzarLado()
8. -- Viaje del punto B al C
9. Ir a funcion AvanzarLado()
10. -- Viaje del punto C al D
11. Ir a funcion AvanzarLado()
12. -- Viaje del punto D al A
13. Ir a función AvanzarLado()
14. Número de vuelta = Numero de vuelta + 1.
15. Vaya al paso 5.
16. Fin del Programa.
22. Función AvanzarLado()
23. Ciclar mientras el sensor no “vea” color negro
24.   Avanzar a media potencia
25.   Si el robot se acerca a un obstáculo al menos
      a 10 cm
26.         Girar a la derecha
27.         Avanzar
28.         Girar a la Izquierda
29.         Avanzar
30.         Girar a la Izquierda
31.         Avanzar
32.         Girar a la Derecha
33.    Fin Si
34. Fin de Ciclo. Vaya al paso 23
35. Dar Vuelta a la Izquierda
36. regresar a programa principal
```

```
// Función AvanzarLado. Da una vuelta completa al edificio rectangular
// Espera una marca (cinta aislante) al llegar al extremo opuesto
// Si ahy obstáculo detectado por el sensor sónico, lo evade por la derecha
void AvanzarLado()[
     while(no vea color negro)
          // Avanzar en línea recta hasta encontrar marca de esquina
          Avanzar(50);
          // Dar vuelta a 10 cm del obstáculo
          IF (distancia obstáculo < = 10)[
          GirarDer(15);
          // Avanzar línea recta
          Avanzar(50);
          // Giro Izquierda en la esquina
          GirarIzq(15);
          // Avanzar línea recta
          Avanzar(50);
          // Giro Izquierda en la derecha
          GirarIzq(15);
          // Avanzar línea recta
          Avanzar(50);
          // Giro Derecha en la esquina
          GirarDer(15);
     ]
]
```

Tabla 2.

PSEUDOCÓDIGO

PARA SOLUCIONAR EL DESAFÍO.

(continúa en la página siguiente)

```
// Girar a la izquierda
GirarIzq(50);
]
// Desafío Robot Velador
// El robot da 3 vueltas alrededor del edificio rectangular
// por lo que el tiempo de recorrido en cada recta puede variar
// En cada esquina gira 90 grados
task main()[
      int NoVuelta = 0;
      int valorInt = 0
      // Evaluando el interruptor
      while (valorInt = 0 )[
            if SensoValue( interruptor ) = 1[
                  valorInt = 1;
            ]
      ]
      while(NoVuelta = 4)
      [
            // una vuelta completa
            AvanzarLado(50);
            // contador de vueltas
            NoVuelta = NoVuelta = 1
      ]
]
```

Tabla 2.

PSEUDOCÓDIGO

PARA SOLUCIONAR EL DESAFÍO.

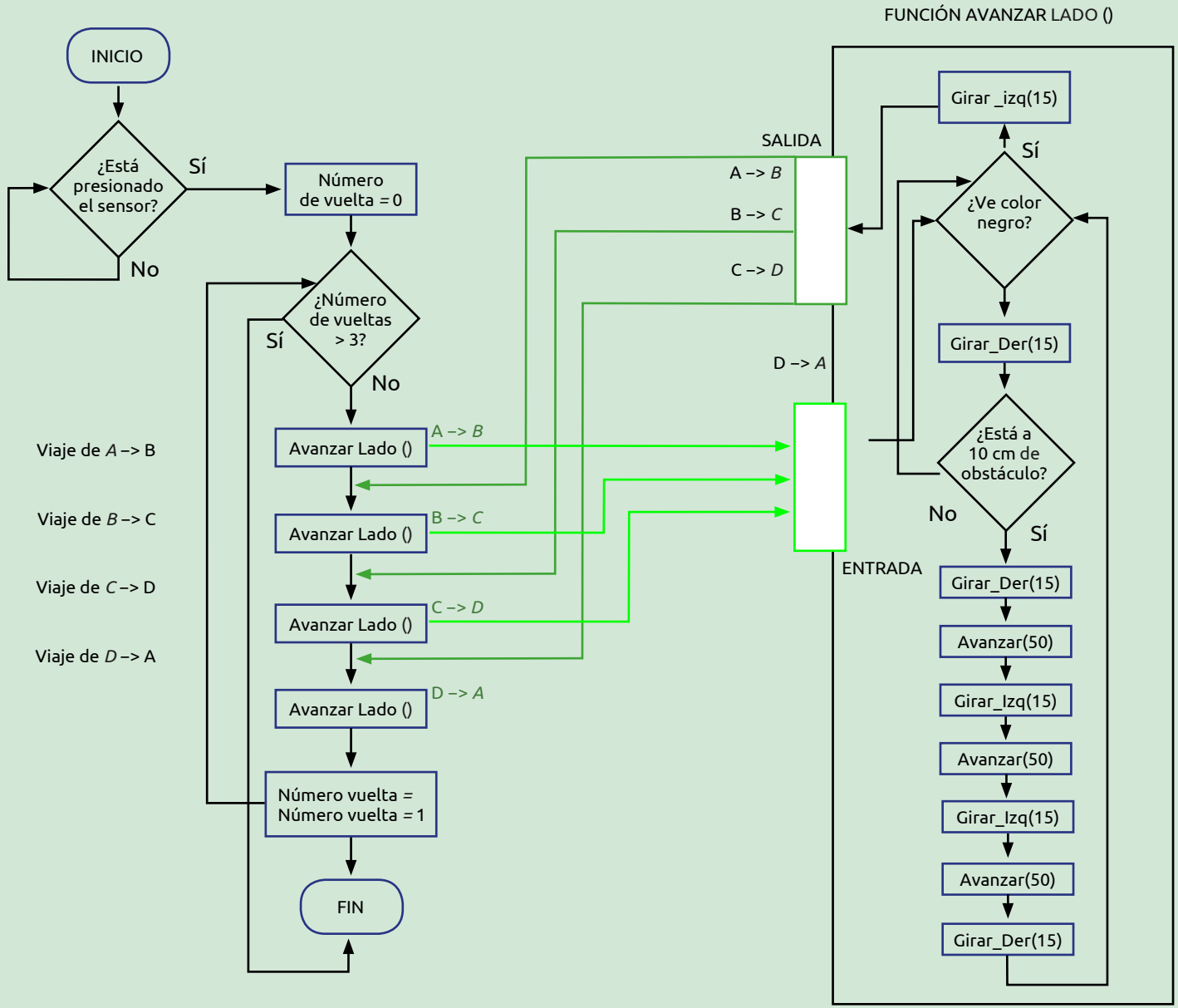
(finaliza)

Figura4.

DIAGRAMA DE FLUJO

PARA SOLUCIONAR

EL DESAFÍO..



Anota en el recuadro el programa resultante con el que diste solución a los pasos de este desafío.

TRABAJO ADICIONAL

Los desafíos que se encuentran en esta sección los debes resolver; te será necesario crear un algoritmo y el programa que da solución a cada desafío. La finalidad de este ejercicio es que vayas construyendo programas cada vez más complejos y que verifiques tus avances.

1) Para el problema propuesto en este laboratorio es posible crear funciones que controlen el movimiento del robot. Propón un diagrama de flujo y programa el comportamiento básico. Demuestra que es posible aplicar aún más el concepto de generalización del razonamiento computacional, para simplificar aún más tu proyecto.

2) Un proyecto adicional ha sido asignado a ACME Robotics. El edificio ahora tiene la forma de una cruz. ¿Qué cambios tendrías que hacer a tu programa para que el robot dé tres vueltas completas al edificio? ¿Qué nuevas funciones necesitas construir? Trata de que el proyecto trabaje con solo dos funciones, una para avanzar y otra para girar (en cualquier dirección). Construye su diagrama de flujo y el programa en ROBOTC® para hacer el recorrido. Usa la maqueta desarrollada en el laboratorio 3 para

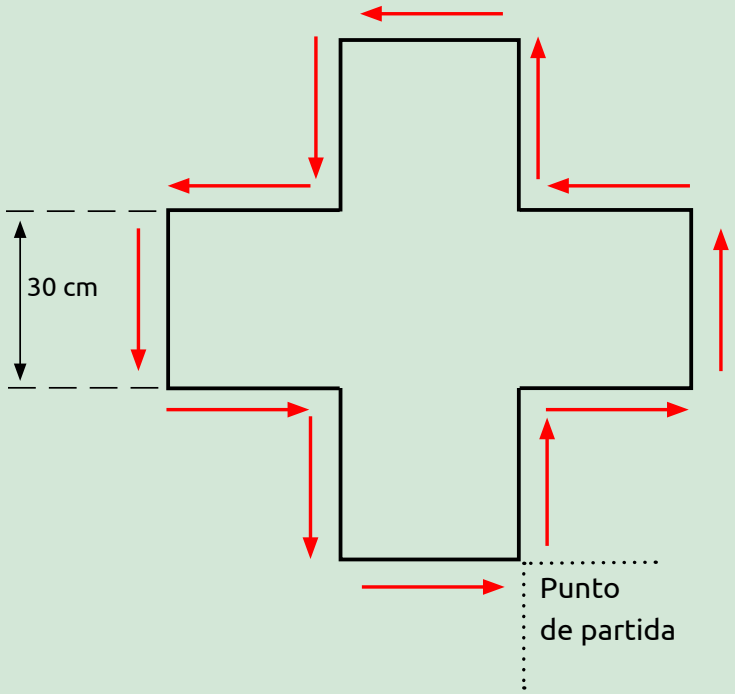
probar que el robot está funcionando adecuadamente. Observa la figura 5 para que tengas idea del recorrido del robot; aunque no se muestran las vueltas en la figura, considera que el robot debe girar al dar vuelta en las aristas. Las marcas de color verde indican el lugar en donde debes poner las marcas de cinta aislante.

3) Finalmente, construye un programa en ROBOTC® para tu maqueta del almacén de autos (rectángulo) en el que sólo le indiques la distancia por recorrer (en centímetros) y que el programa determine automáticamente los grados que debe contar para avanzar la distancia indicada. Usa funciones y el sensor de rotación del motor del robot. Deberás entregar el diagrama de flujo y el programa que solucionan el problema.

Figura 5.

RECORRIDO DEL ROBOT

PARA EL EDIFICIO CON FORMA DE CRUZ.



Referencias

Robomatter Inc. (2016). ROBOTC for LEGO Mindstorms 4.X - Users Manual. Disponible para consulta en el sitio web: https://www.robotc.net/WebHelpMindstorms/index.htm#Resources/topics/ROBOTC_Interface/Overview.htm

Carnegie Mellon University (2020). ROBOTC for TETRIX & LEGO® MINDSTORMS.https://www.cmu.edu/roboticsacademy/roboticscurriculum/Lego%20Curriculum/ROBOTC-Tetrix_Mindstorms.html

ROBOT VELADOR, LABORATORIO 7

BITÁCORA DE VIGILANCIA

OBJETIVOS DEL DESAFÍO QUE SE TE PLANTEA:

→ Desarrollar el pensamiento lógico necesario para usar estructura de datos que alamcenará una bitácora de movimientos del robot.

→ Diseñarás programas usando funciones, variables, estructuras, arreglos y punteros con ROBOTC® e implementarás dicho programa en un robot LEGO EV3.

→ Probarás que el robot ejecuta las instrucciones adecuadamente y almacena correctamente los datos del recorrido.

Luis Antonio Álvarez Oval, Christian Mauricio Castillo Estrada y Aron de la Cruz Vázquez, *Introducción a la robótica*, Sin fronteras, núm. 3, UNACH, México, 2022, 180 pp. ISBN: 978-607-561-130-3

www.unach.mx

Disponible en: www.editorial.unach.mx

INTRODUCCIÓN

En este séptimo laboratorio el estudiante se enfrenta al desafío de mejorar su técnica de programación incorporando estructuras estáticas y elementos avanzados que permitan el almacenamiento de datos en tiempo real, tales como: variables, arreglos, punteros y estructuras; sin perder de vista la relación que existe con los fundamentos teóricos de la metodología de la programación y el desarrollo de habilidades cognitivas en los estudiantes.

Para poder implementar los elementos mencionados el estudiante debe identificar las palabras reservadas y los comandos necesarios que le permitan declarar las diferentes estructuras estáticas para el almacenamiento de información. Así también, es necesario que realice las pruebas de funcionamiento evitando errores de acceso a segmentos de memoria no permitidos en el computador central del robot.

ROBOTC® y el robot nos permiten trasladar la programación del mundo abstracto dentro de la computadora al mundo real, en donde cada orden que se escribe puede percibirse como actividad del robot. El estudiante debe comprender la programación de computadoras como una actividad lógica sencilla de entender y aplicar; esa comprensión le permitirá desarrollar conceptos mucho más abstractos como la comunicación entre procesos remotos, programación orientada a objetos, programación web, etcétera.

El robot que estaremos usando a lo largo de este curso tiene una configuración de dos ruedas cada una con su propio motor, que serán los actuadores, y una rueda de arrastre omnidireccional; en el tema de sensores. Para percibir el ambiente que lo rodea el robot dispone de un sensor táctil, ultrasónico, y de otro para luz/color.

Para este desafío necesita los siguientes materiales:

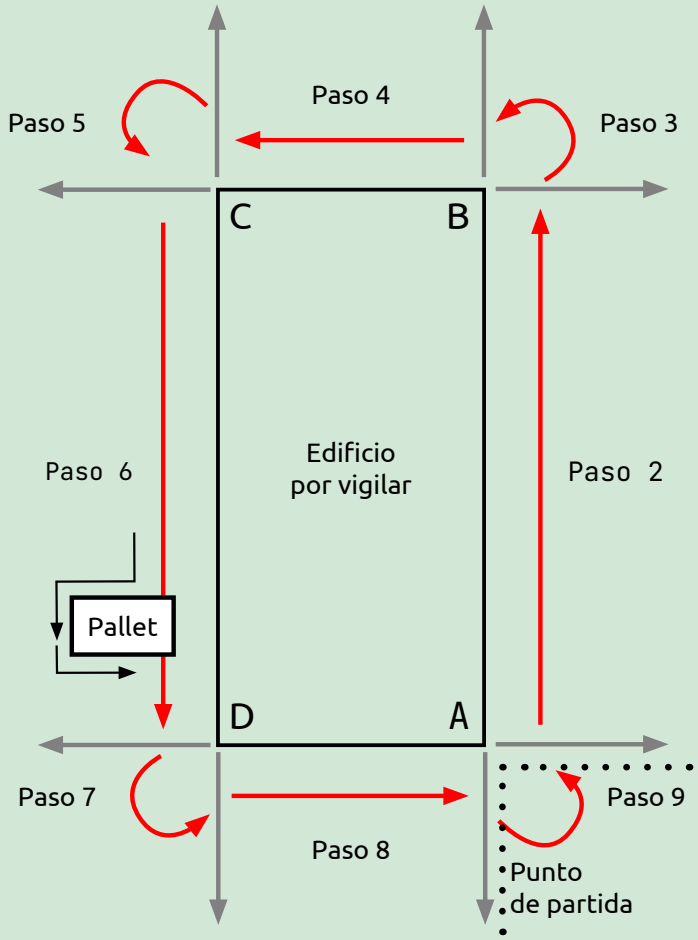
- Lenguaje de programación ROBOTC®.
- Robot Lego EV3 armado.
- Cinta aislante de color negro.

DESAFÍO POR RESOLVER

La compañía ACME Robotics ha ganado un proyecto para diseñar un robot programable que patrulle los alrededores de un edificio rectangular que almacena autos. ACME te ha contratado para desarrollar el programa para que ese robot prototipo pueda, de forma autónoma, "vigilar" los alrededores del edificio, desde las 20:00 horas de un día hasta las 6:00 horas del siguiente. Además de que el robot pueda evadir obstáculos en su ruta. Ha surgido un nuevo requerimiento: es necesario crear una bitácora que guarda el número de vuelta y el contador de grados acumulados por el sensor de rotación del motor; los datos se deben tomar al llegar a cada esquina del edificio. Estos datos deben ser mostrados en la pantalla del robot al finalizar la última vuelta. La figura 1 explica paso a paso el problema, las líneas color verde etiquetadas con la nota "almacenar datos" muestran en dónde se deben adquirir datos del robot y almacenar en variables, arreglos y apuntadores. Otro equipo de trabajo ya ha desarrollado el prototipo mecánico-electrónico del robot y está disponible para su uso.

Figura 1.

RECORRIDO DEL ROBOT VELADOR.



Es necesario que el robot recorra tres vueltas completas al edificio, eluda obstáculos y debe iniciar el recorrido hasta que el usuario presione un interruptor. Puesto que se trata de un prototipo no es necesario que ejecute una rutina de vigilancia constante. Sólo que estará aplicando el concepto almacenar datos usando primero variables, arreglos, estructuras y después apuntadores para completar el desafío. El proceso de ingeniería implica que la solución la vayas construyendo paso a paso, por lo que el problema lo resolverás en tres pasos, mismos que te explicaremos más adelante.

Primer problema. Uso de variables

El robot debe ser capaz de ejecutar tres vueltas y en cada una de tomar cuatro lecturas; así que debe tener disponibles 12 variables para almacenar el número de vuelta y otras 12 variables para almacenar los grados recogidos por el sensor de rotación.

Segundo problema. Uso de arreglos

El robot debe ser capaz de ejecutar tres vueltas y en cada una debe tomar cuatro lecturas, asi que el robot debe tener disponibles dos arreglos de tipo entero con 12 posiciones cada uno, uno para almacenar el número de vuelta y otro para almacenar los grados recorridos por el sensor de rotación.

Tercer problema. Uso de arreglos con estructuras

El robot debe ser capaz de ejecutar tres vueltas y en cada vuelta debe tomar cuatro lecturas, así que el robot debe tener disponibles un arreglo de tipo estructura con 12 posiciones cada uno, la estructura debe manejar una variable para almacenar el número de vuelta y otra para almacenar los grados recorridos por el sensor de rotación.

Cuarto problema. Uso de apuntadores

El robot debe ser capaz de ejecutar *n* vueltas y en cada una debe tomar cuatro lecturas, así que el robot debe tener disponibles un apuntador de tipo estructura con 12 posiciones cada uno, la estructura debe manejar una variable para almacenar el número de vuelta y otra para almacenar los grados recorridos por el sensor de rotación.

CONCEPTOS BÁSICOS

Aquí encontrarás material de apoyo para que entiendas el objetivo primario de este documento. En general la sección Conceptos básicos proviene de los documentos que el Laboratorio Nacional de Ingeniería Robótica de la Universidad Carnegie Mellon ha proporcionado en los cursos de robótica que imparte.

Apuntadores y estructuras de datos

En esta sección se te introducirá a los conceptos variables y apuntadores, usando diagramas, ejemplos y programas desarrollados en lenguaje C estándar. La versión 3.5 de ROBOTC® trajo una miríada de nuevas características, incluyendo la tan esperada implementación de punteros a variables. Teniendo esta funcionalidad abre una amplia gama de nuevas posibilidades, como la implementación de estructuras de datos complejas.

Variables

Antes de la versión 3.5, ROBOTC® solo soportaba variables normales; en otras palabras, una variable, ya sea un `int`, `float` o cualquier otra cosa, solo tenía un valor, como 712, 0,383 o `"howdy"`.

Por ejemplo:

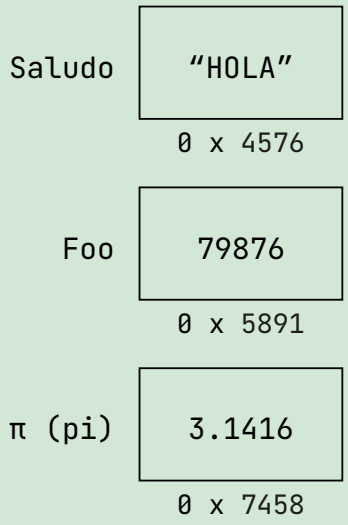
```
long foo = 79876;
float pi = 3.1416;
string saludo = "hola";
```

En efecto, cuando se declara una variable, el compilador reserva una cantidad de memoria del tamaño adecuado y le da un nombre simbólico, definido por el usuario, como "`saludo`" o "`foo`" o "`pi`". Un entero (`int`) tiene cuatro bytes de tamaño, un número de punto flotante (`float`) también tiene cuatro bytes, pero una cadena de caracteres (`string` - en el caso de ROBOTC®) suele tener 20 bytes de tamaño. Observa la figura 2 (verá los bloques de memoria, las etiquetas asignadas a ellos, sus contenidos y la dirección de memoria para ese bloque (los números hexadecimales debajo de los bloques).

Una variable tiene dos partes, un valor a la derecha (valor `-d`) y un valor a la izquierda (valor `-i`). El término valor `-d` se refiere al lado derecho de la declaración de asignación y el valor-i se refiere al lado izquierdo. Considera como referencia el programa 1, para la explicación que se ofrece abajo.

Figura 2.

LAS VARIABLES Y LA DIRECCIÓN DE MEMORIA EN DONDE SE ALOJAN.



Cuando se ejecuta el programa 2, la salida a "Debug Stream"* de ROBOTC® debería verse así:

i: 10, j: 51

i: 10, j: 10

i: 11, j: 10

Para abrir el ROBOTC® "Debug Stream", asegúrate de que la interfaz de ROBOTC® esté en el modo "Experto" o "Súper usuario" y abre el flujo de depuración desde el menú de ventanas del depurador normal. Lee tu antología

Programa 1.

EJEMPLO DE USO DE VARIABLES EN ROBOTC®.

```
// Ejemplo 1 Apuntadores
task main(){
   int i, j;
   i = 10;
   j = 51;
   // Este comando debe imprimir - i: 10, j: 51
   writeDebugStreamLine("i: %d, j: %d", i, j);
   // Asignar el valor r de i a j
   j = i;
   // Este comando debe imprimir - i: 10, j: 10
   writeDebugStreamLine("i: %d, j: %d", i, j);
   i++;
   // Este comando debe imprimir - i: 11, j: 10
   writeDebugStreamLine("i: %d, j: %d", i, j);
}
```

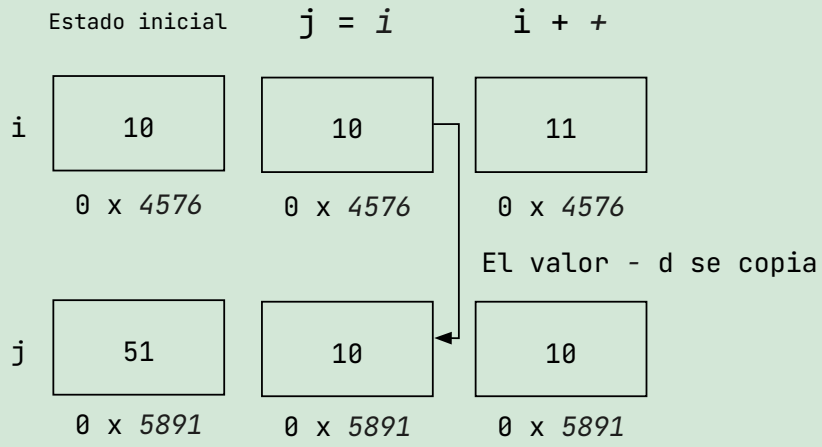
(sección 16, página 44) para que obtengas una descripción más detallada.

Al mirar las ubicaciones de la memoria y su contenido después de cada operación del programa 1, deberás ver algo cómo lo que se describe en la figura 3. En la primera asignación al inicio del programa, la variable i en el valor derecha (valor-d)

se le ha asignado un 10. Justo debajo de esta asignación, hay una segunda asignación donde valor -d de la variable j se le asigna un 51. Esto se considera como establecer un valor inicial. Más adelante en el programa, ¿qué pasará con la línea "j = i"? Simple, el valor -d de i se copia y se asigna al valor-d de j. Ahora ambos valores de j e i son 10. ¿Qué sucede cuando se incrementa? ¿Se cambia el valor de j? En resumen, no, los dos valores -d, tanto de i como de j están completamente separados.

Figura 3.

DIRECCIONES DE MEMORIA DE LA EJECUCIÓN DEL PROGRAMA 2.



En la primera asignación al comienzo del programa, variable. Su valor r tiene el valor 10. Justo debajo de esta asignación, hay una segunda asignación donde el valor r de la variable j recibe el valor 51.

Esto se considera establecer un valor inicial. Más adelante en el programa, ¿Qué pasará con la línea "j = i"?

Simple, el valor r de i se copia y se asigna al valor -d de j. Ahora ambos son los valores de j e i son 10. ¿Qué sucede cuando incrementamos? ¿Lo hace cambiar el valor r de j? En resumen, no, los dos valores -d son entidades completamente separadas. Cuando el compilador asigna el valor de la variable i a la variable j, "copiará" ese valor de una variable a la otra.

Como ya mencionamos, cuando se declara una variable el compilador le asigna un trozo de memoria de tamaño adecuado. Este pedazo tiene una dirección, un número, muy parecido a una casa. Si quieres que acceda a esta dirección, puede hacerlo con el operador de referencia (&). Sin embargo, esto no es muy útil en sí mismo. Los creadores de C no agregaron esta capacidad para obtener la dirección sin poder hacer algo práctico con eso también. Aquí es donde el puntero (*) del operador entra en la imagen.

Anteriormente, vimos que el valor-d de una variable contiene el valor real que le asignamos. Con un apuntador, el valor-d es de hecho la dirección de la variable a la que estamos apuntando. Pero, ¿y si queremos ver el valor de esta variable a la que estamos apuntando? Podemos hacer eso con el operador de referencia, también un " * ". Combinando todo este nuevo conocimiento encontrado, obtenemos el programa 2.

Programa 2.

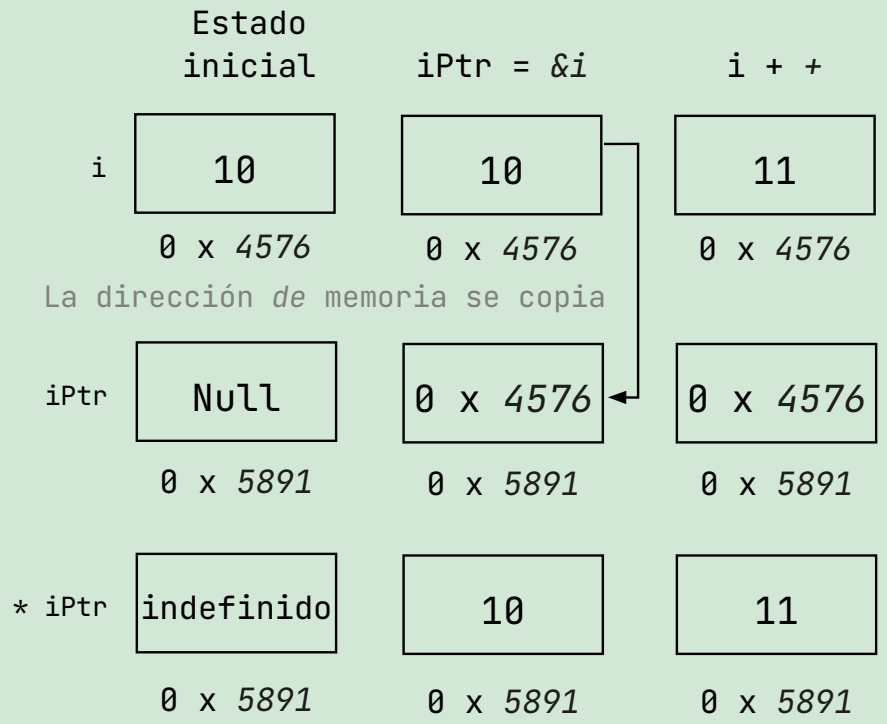
EJEMPLO RELACIONADO AL USO PUNTEROS EN ROBOTC®.

```
task main() {
   int numi = 10;
   int *intnumPtr;
   intnumPtr = &i; // intnumPtr ahora apunta a la dirección de i
   // Este comando debe imprimir:
   // numi: 10, intnumPtr: 674F4839, *intnumPtr: 10
   // (el valor de intnumPtr puede variar)
   writeDebugStreamLine("numi: %d, intnumPtr: %p, *intnumPtr:%d", numi, intnumPtr, *intnumPtr);
   i++;
   // Este comando debe imprimir:
   // numi: 11, intnumPtr: 656F6968, *intnumPtr: 11
   // (el valor de intnumPtr puede variar)
writeDebugStreamLine("i: %d, intnumPtr: %p, *intnumPtr:%d", numi, intnumPtr, *intnumPtr);
```

La salida en el ROBOTC Debug Stream debería verse así:

```
numi: 10, intnumPtr: 674F4839, *intnumPtr: 10
numi: 11, intnumPtr: 674F4839, *intnumPtr: 11
```

Si lo miras desde la perspectiva de la memoria, el programa 3 debería verse como en la figura 6.



Apuntador aritmético

Ahora que ya sabe cómo son los apuntadores, puedes empezar a divertirte un poco con ellos.

Considera el siguiente programa:

Programa 3.

USO DE APUNTADORES.

```
    // programa ejemplo
// Ejemplo 2 relacionado al uso de punteros
    task main(){
       ubyte arrNumeros[3] = {1, 2, 3};
       ubyte *punteroArrNum;
    // El puntero declarado apunta a la dirección de memoria de arrNumeros[0]
       punteroArrNum = &arrNumeros[0];
    // Esto debe imprimir
    // arrNumeros[0]: 1, punteroArrNum: F4DF6933, * punteroArrNum: 1
    // (el valor de punteroArrNum puede variar dependiendo del valor que se desea obtener)
       writeDebugStreamLine(“arrNumeros[0]: %d, punteroArrNum: %p, * punteroArrNum: %d”, arrNu-
meros[0], punteroArrNum, * punteroArrNum);

    // Incrementar el valor para pasar al siguiente elemento de arrNumeros[1]
       punteroArrNum ++;
    // Esto debe imprimir arrNumeros[1]: 2, punteroArrNum: F4DF7031, * punteroArrNum: 2
    // (el valor de punteroArrNum puede variar)
       writeDebugStreamLine(“arrNumeros [1]: %d, punteroArrNum: %p, * punteroArrNum: %d”, arrNu-
meros[1], punteroArrNum, *punteroArrNum);
```

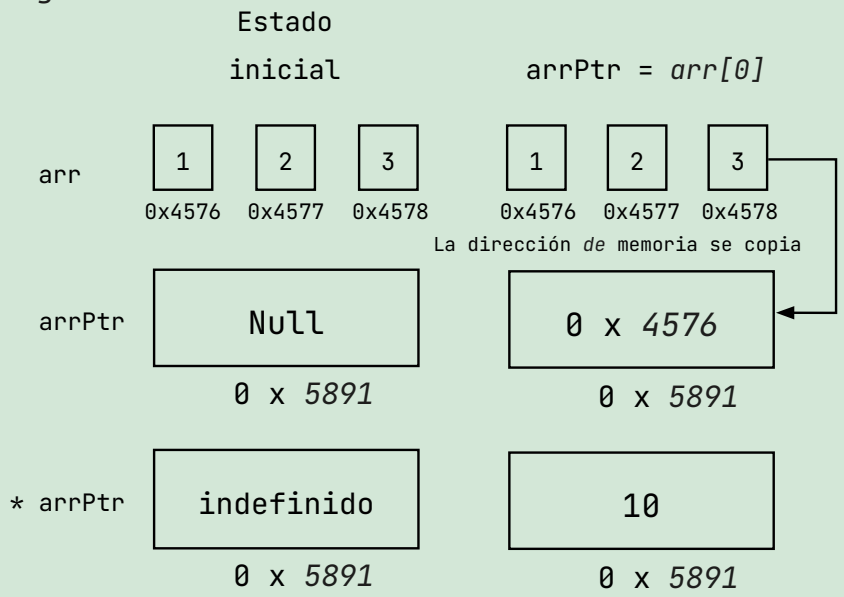
El programa 4 debería imprimir algo como esto en la consola del Debug Stream:

```
arrNumeros [0]: 1, punteroArrNum: F4DF6933, * punteroArrNum: 1
arrNumeros [1]: 2, punteroArrNum: F4DF7031, *punteroArrNum: 2
```

Considera que la dirección arrPtr puede ser diferente en cada ejecución. Es importante que, en la segunda línea, la dirección a la que apunta arrPtr sea más alto que en el primero. Cuando miras cómo se ve en la memoria, puedes obtener una buena idea de lo que está pasando, así que estudia la figura 5.

Figura 5.

DIRECCIONES DE MEMORIA DEL PROGRAMA 4.



Hechos interesantes de un apuntador

Cuando use `sizeof` () en una variable normal, como ubyte, `int` o `long`, se obtiene la cantidad de bytes que ocupa este tipo. En el caso de un ubyte eso es un byte, `int` ocupa dos y un `long` usa cuatro. Hacer un "tamaño de" de un puntero es, inútil; no obtendrás el tamaño del elemento al que apunta, siempre será 4 en el caso de ROBOTC®. Esto se debe a que la dirección almacenada en el puntero es un número de 4 bytes, uno largo. Considera esto cuando quieras usar `sizeof` ( ) para obtener el número de bytes que deseas borrar con la función `memset`( ), por ejemplo.

Estructuras

Las estructuras son variables que se componen de una colección de otras variables. También son conocidas como registros. Nuevamente, esta no es una característica única de ROBOTC®, sino que es útil en general, y no es ampliamente conocida entre los programadores principiantes. Te mostramos su uso con el siguiente ejemplo:

```
typedef struct{
  int velocidadBrazo;
  int valoresSensorReversa[10];
  int posicionMaxima;
  int posicionMinima;
} brazoRobot;
```

En el código de ejemplo anterior, se realiza la declaración de una estructura integrada por 4 atributos o variables, para ello, se utiliza la palabra reservada struct; tomando en consideración esa estructura, es posible dar solución al problema de controlar múltiples brazos en el robot al mismo tiempo, para ello, se puede usar una estructura que almacene los valores de las propiedades de cada uno de los brazos. Al implementar una estructura, la sintaxis para acceder a cada componente es la siguiente:

```
brazoRobot brazo_Central;
brazoCentral.posicionMaxima = 2000;
brazoCentral.posicionMinima = 750;
brazoCentral.velocidadBrazo = 700;
```

Arreglos

Los bucles permiten al programador ejecutar multitud de instrucciones y manipular grandes cantidades de información de forma controlada. Supongamos que quisiéramos escribir un programa que recopila mediciones de luz, una vez cada 15 minutos durante todo un día. Supongamos, además, que al final del período de medición queremos crear un gráfico de barras de las mediciones de luz para poder visualizar la variación de las mismas a lo largo del día. Para crear el gráfico de barras, necesitaríamos poder recuperar todas y cada una de las mediciones de luz, de principio a fin. Un total de 96 valores. Por lo general, la forma en que almacenamos y recuperamos valores es en variables. Sería tedioso tener que declarar 96 variables en un programa, por lo que la mayoría de los lenguajes de programación, incluido ROBOTC®, le permiten al programador declarar matrices de variables. Una matriz es una colección indexada de variables del mismo tipo de datos. Indexado significa que se puede acceder a cada variable (también llamada elemento) de la matriz usando un índice entero.

La sintaxis para declarar una matriz de variables (arreglo) es

```
[Tipo de datos] [Nombre del arreglo] [Tamaño]
```

Esto es como una declaración de variable ordinaria, pero el nombre de la variable va seguido de un entero entre corchetes que indica el número de elementos asociados con el nombre de la matriz. En el caso de nuestras mediciones de luz, que serán de tipo entero, podríamos usar la declaración

```
int lectura_luz[96]
```

Esta declaración directa le da al programador `96` variables de la forma `lectura_luz[0], lectura_luz[1], lectura_luz[2], ..., lectura_luz[95] trabajar con`. Observe que el primer elemento de la matriz tiene índice `0`, que el elemento subsiguiente índice cada incremento en uno, y que el último elemento tiene índice `95`. Los índices de matriz siempre comienzan en `0` y terminan con un índice menor que el tamaño declarado de la matriz.

Se produce un desborde del arreglo si intenta usar un índice fuera de rango para acceder a un elemento del mismo.

Este error común puede ser difícil de encontrar porque el compilador no puede detectarlo. Dichos errores pueden conducir a un comportamiento impredecible, de la misma manera que las variables no inicializadas. La corrección de los excesos vinculados al arreglo implica que inspecciones cuidadosamente todas las operaciones y la determinación clara del valor de cualquier variable de índice.

El programa 4 muestra el conjunto de instrucciones que recopila mediciones de luz cada 15 minutos durante 24 horas. Inicializa una variable de contador de tipo entero, `count` a 0, y declara una matriz de 96 elementos para las lecturas de luz. El ciclo `While` infinito, toma una lectura, incrementa el contador y hace una pausa de 15 minutos antes de repetir. Cuando el arreglo está lleno, la declaración de interrupción hace que el ciclo termine. Considera que podemos usar el recuento de variables como el índice de matriz. Comienza en 0 e incrementa en 1 cada vez a través del bucle, de modo que cada lectura de luz se asigna a un elemento de matriz diferente. Observa también que la declaración de interrupción se produce después del incremento del contador, pero antes de las llamadas `wait10Msec ( )`

Programa 4.

LECTURA DE LUZ CADA 15 MINUTOS.

```
#pragma config (Sensor, SenLuz1, luzSensor ,
sensorLuzInactivo)
task main() {
   int lectura_luz[

// Ciclo infinito
while(true) [
      // Almacenando la lectura del sensor
      lectura_luz[cuenta] = SensorValue[lightsensor];
      // Incremento al contador de lecturas
      cuenta ++;
      // Si el arreglo está lleno... terminar el ciclo
      if(cuenta == 95) [ break ]
      wait10Msec (30000);
      wait10Msec (30000);
      wait10Msec (30000);
      ]
// Fin del programa
]
```

SOLUCIÓN DEL DESAFÍO

Con la cinta aislante agrega las marcas en las esquinas del edificio, tal y como se ven en la figura 1 (pág. 101). Pero ahora el robot no se moverá con tiempo, va a moverse hasta que detecte el cambio de color del fondo blanco al color negro de la cinta aislante. Incluso las vueltas pueden ser controladas hasta que el sensor detecte el cambio de color.

Ahora se te proporcionan tres herramientas que te permiten resolver el problema.

A) Solución del problema como una receta de cocina (Tabla 1), explicada en idioma español.

B) Solución con pseudocódigo (Tabla 2), expresado también en idioma español concreto.

C) Solución con algoritmo usando la notación de diagramas de flujo (Figura 6).

Asegúrate de estudiar completamente cada solución ya que el siguiente paso es que generes el programa que resuelva el desafío. Relaciona estos conceptos con el proceso de ingeniería que se te ha presentado en la sección Conceptos básicos del Laboratorio 3.

Tabla 1.

SOLUCIÓN PARECIDA A UNA RECETA DE COCINA

(continúa en la página siguiente).

```
1. Inicio del programa
2. Leer el estado del interruptor
3. Si el interruptor está apagado vaya al paso 2
4. Número de vuelta es igual a 0
5. Si ya va en la cuarta vuelta ir al paso 21
6. -- Viaje del punto A al B.
7. Ir a función AvanzarLado()
8. -- Viaje del punto B al C
9. Ir a función AvanzarLado()
10. -- Viaje del punto C al D
11. Ir a función AvanzarLado()
12. -- Viaje del punto D al A
13. Ir a función AvanzarLado()
14. Numero de vuelta = Numero de vuelta + 1.
15. Vaya al paso 5.
16. Desplegar Datos
17. Fin del Programa.
```

Tabla 1.

SOLUCIÓN PARECIDA A UNA RECETA DE COCINA

(finaliza).

```
22. Funcion AvanzarLado ()
23. Ciclar mientras el sensor no “vea” color negro
24.    Avanzar a media potencia
25.    Si el robot se acerca a un obstáculo al menos a 10 cm
26.         Girar a la derecha
27.         Avanzar
28.         Girar a la Izquierda
29.         Avanzar
30.         Girar a la Izquierda
31.         Avanzar
32.         Girar a la Derecha
33.      Fin Si
34. Fin de Ciclo. Vaya al paso 23
35. Almacenar Datos
36. Dar Vuelta a la Izquierda
37. regresar a programa principal
```

Tabla 2.

PSEUDOCÓDIGO PARA EL DESAFÍO.

(continúa en la página siguiente)

```
// Función Avanzando. Da una vuelta completa al edificio rectangular
// espera una marca (cinta adhesiva) al llegar al extremo opuesto
// Si hay obstáculo detectado por el sensor sónico, lo evade por la derecha
void AvanzarLado()[
      while(no vea color negro)[
            // Avanzar en línea recta hasta encontrar marca de esquina
            Avanzar(50);
            // Dar vuelta a 10 cm del obstáculo
            IF (distancia obstáculo < = 10){
                  // Giro Derecha en la esquina
                  GirarDer(15);
                  // Avanzar línea recta
                  Avanzar(50);
                  // Giro Izquierda en la esquina
                  GirarIzq(15);
                  // Avanzarlínea recta
                  Avanzar (50);
                  // Giro Izquierda en la esquina
                  GirarIzq(15);
                  // Avanzar línea recta
                  Avanzar(50);
                  // Giro Derecha en la esquina
                  GirarDer(15);
```

Tabla 2.

PSEUDOCÓDIGO PARA EL DESAFÍO.

(finaliza)

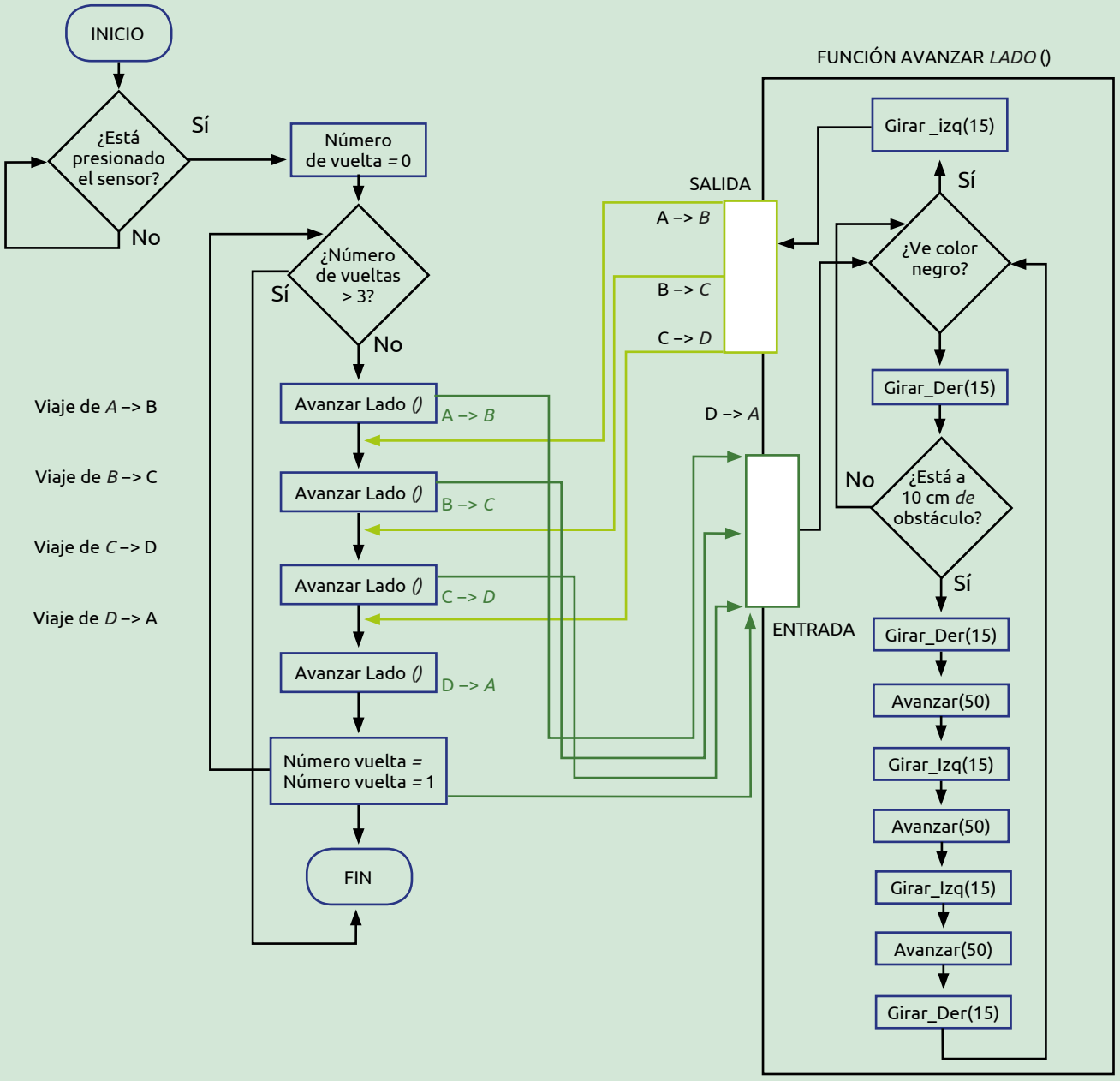
```
      ]
]
AlmacenarDatos();
// Giro a la Izquierda
GirarIzq(50);
]
// Desafio Robot Velador
// El robot da 3 vueltas alrededor del edificio rectangular
// por lo que el tiempo de recorrido en cada recta puede variar
// En cada esquina gira 90 grados
task main()[
      int NoVuelta = 0;
      int valorInt = 0
      // Evaluando el interruptor
      while(valorInt = 0 )[
            if Sensorvalue( interruptor )= 1[
                  valorInt = 1;
            ]
      ]
      while(NoVuelta) < 4)
      [
            // Una vuelta completa
            AvanzarLado(50);
            // Contador de vueltas
            NoVuelta = NoVuelta = 1
      ]
      DesplegarDatos();
]
```

Figura 6.

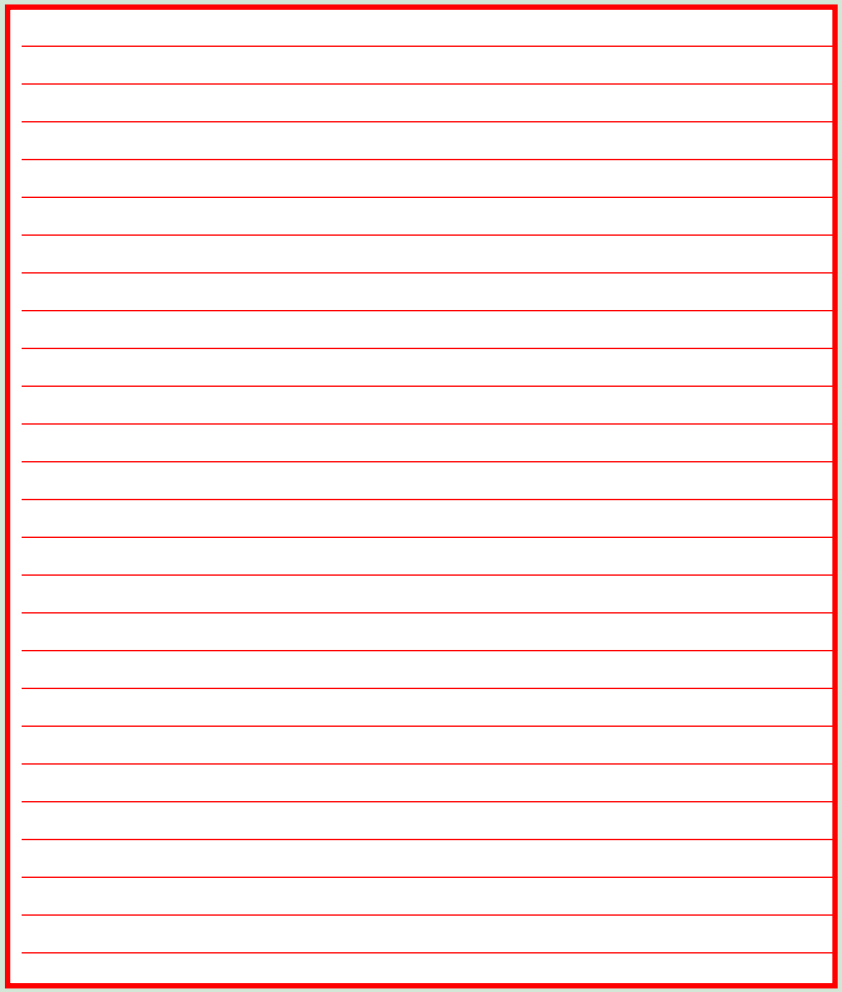
DIAGRAMA DE FLUJO

PARA SOLUCIONAR

EL DESAFÍO..



Anote en el recuadro el programa resultante con que diste solución a los pasos de este desafío.



TRABAJO ADICIONAL

Los desafíos que se encuentran en esta sección deben ser resueltos por el estudiante, es necesario que crees un algoritmo y el programa que da solución a cada desafío. La finalidad de este ejercicio es que vayas construyendo programas cada vez más complejos y verifiques tus avances.

Construye un robot que use el sensor ultrasónico y el sensor de rotación, aplica tu conocimiento de funciones y apuntadores. Es necesario que entreges un diagrama de flujo y el programa en ROBOTC®. El robot debe avanzar indefinidamente hasta que se encuentre un obstáculo, cuando se encuentre a 15 cm del mismo debe frenar, retroceder 5 segundos y girar a la izquierda; en este momento debe hacer un registro del tiempo de ejecución del programa sobre el robot (`nPgmTime`) en un apuntador con estructuras y de los grados que tiene acumulados el sensor de rotación, para después continuar con su avance recto. El robot sólo debe funcionar durante tres minutos. Al finalizar su ejecución debe mostrar los datos recolectados en la pantalla del robot.

Referencias

Robomatter Inc. (2016). ROBOTC for LEGO Mindstorms 4.X - Users Manual. Disponible para consulta en el sitio web: https://www.robotc.net/WebHelpMindstorms/index.htm#Resources/topics/ROBOTC_Interface/Overview.htm

Carnegie Mellon University (2020). ROBOTC for TETRIX & LEGO® MINDSTORMS.https://www.cmu.edu/roboticsacademy/roboticscurriculum/Lego%20Curriculum/ROBOTC-Tetrix_Mindstorms.html

Cairó, O (2005). Metodología de la programación. Algoritmos, diagramas de flujo y programas. Alfaomega. 3ª. edición.

ROBOT VELADOR, LABORATORIO 8

MONITOREO DE VIGILANCIA MEDIANTE UNA CÁMARA MÓVIL

OBJETIVOS DEL DESAFÍO QUE SE TE PLANTEA:

→ Analizar el entorno, tomar decisiones y establecer comunicación remota en acciones que coadyuven al desarrollo del pensamiento cognitivo y a la creatividad.

→ Mejorar la estructura física del robot, usar estructuras selectivas de decisión e incorporar otras tecnologías complementarias para añadir mayores funcionalidades.

→ Diseñar un algoritmo para controlar a un robot Lego EV3 y probar que ejecute las instrucciones en el orden adecuado y de forma precisa para simular la vigilancia mediante cámara móvil.

→ Enviar imágenes en tiempo real a un equipo de cómputo central (servidor), mediante el uso de una red inalámbrica local.

Luis Antonio Álvarez Oval, Christian Mauricio Castillo Estrada y Aron de la Cruz Vázquez, *Introducción a la robótica*, Sin fronteras, núm. 3, UNACH, México, 2022, 180 pp. ISBN: 978-607-561-130-3

www.unach.mx

Disponible en: www.editorial.unach.mx

INTRODUCCIÓN

Este último laboratorio de prácticas, presenta un desafío de alto grado de dificultad para el estudiante; en virtud de incorporar un componente electrónico para simular la capacidad de visión por parte del robot, para ello, es necesario el uso de una cámara móvil o web cam que permita tomar imágenes del entorno en tiempo real, y posteriormente seas procesadas a nivel código de programación.

Para poder codificar correctamente el algoritmo a través del Lenguaje ROBOTC, el estudiante debe seleccionar los comandos relacionados con el procesamiento de imágenes. Además deberá hacer uso de estructuras de control, selectivas, repetitivas y estructuras para el almacenamiento de datos, es decir, aplicar todos los fundamentos de programación aprendidos durante el desarrollo de cada laboratorio, con la finalidad de diseñar el algoritmo idóneo para este último desafío.

iVCam es una aplicación móvil comercial que se instala en dispositivos móviles con sistema operativo IOS o Android, que permite realizar transmisión de video en tiempo real haciendo uso de la cámara del dispositivo en formato HD hacia un equipo de cómputo (PC). En unos segundos, la cámara del dispositivo se comportará exactamente como una cámara web, enviando señal de video y audio a una computadora, sin requerir ningún tipo de configuración avanzada, tan sólo una conexión a una red inalámbrica WI-FI.

De acuerdo al sitio web oficial de la empresa de tecnología CISCO, el término de ACCESS POINT (AP) se define como un dispositivo de red que permite que un conjunto de dispositivos con capacidad inalámbrica se conecten a una red cableada (CISCO, 2019). Así también, el autor Enrlich realiza la definición de este término AP, como punto de acceso inalámbrico (WAP) por sus siglas en inglés Wireless Access Point; en ese sentido, resulta

ser un dispositivo que permite interconectar varios dispositivos o equipos de comunicación inalámbrica para implementar una red local. En otro orden de ideas, este concepto representa la única ruta de acceso que facilita o permite la comunicación de un grupo de dispositivos, en cierto rango de distancia dependiendo de la capacidad y características del mismo (Enrlich 2011).

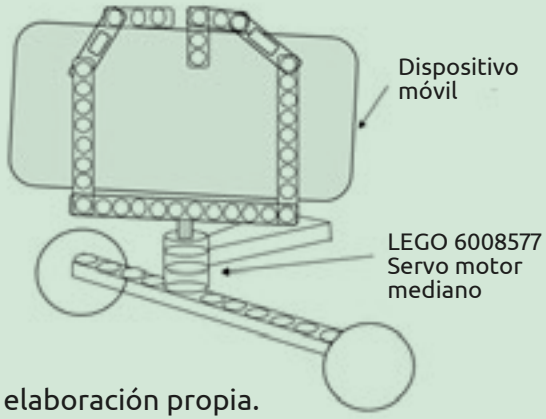
La estructura de robot que usaremos en este laboratorio, debe poseer una configuración de cuatro ruedas, dos de ellas con su propio motor, que serán los actuadores, y dos ruedas al frente; en el tema de sensores, para percibir el ambiente que lo rodea, el robot debe disponer de sensor táctil, ultrasónico, y de luz/color, además, se deberá construir una base giratoria para dar soporte a un dispositivo móvil (teléfono) usando dos ligas y algunas piezas de Lego® en forma de un contenedor; montado sobre un motor independiente que permitirá realizar un monitoreo de 360°. Tal como puedes observar en la figura 1.

Para este desafío necesita los siguientes materiales:

- Lenguaje de programación ROBOTC®.
- Robot Lego EV3 con soporte para dispositivo móvil.
- 1 dispositivo de comunicación inalámbrico Access Point (AP).
- Editor de diagramas DIA. Liga de descarga: http://dia-installer.de/
- 1 dispositivo móvil con cámara integrada y tarjeta de red inalámbrica WI-FI.
- 1 equipo de cómputo portátil o de escritorio con tarjeta de red inalámbrica WI-FI.
- Aplicación iVCam para transmisión de video (instalación en dispositivo móvil y en PC). Liga de descarga: https://www.e2esoft.com/ivcam/

Figura 1.

ESTRUCTURA DE SOPORTE PARA DISPOSITIVO MÓVIL GIRATORIO USANDO UN SERVO MOTOR.



Fuente: elaboración propia.

DESAFÍO POR RESOLVER

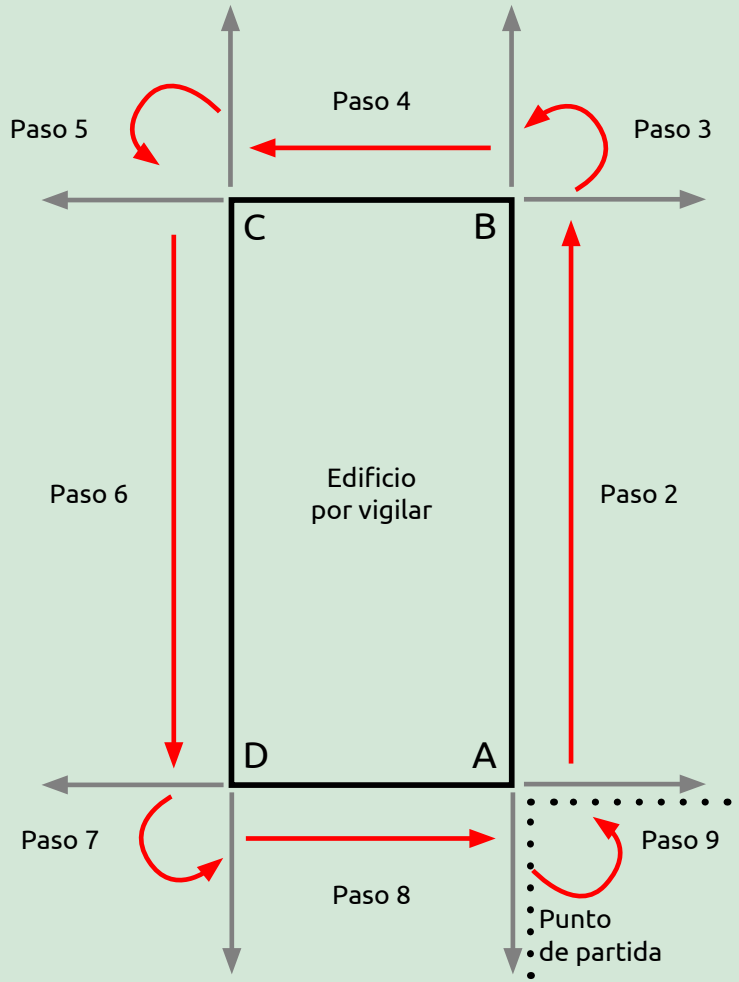
La compañía ACME Robotics desea mejorar al proyecto de un robot programable que patrulla los alrededores de un edificio rectangular y que almacena datos; sus directivos desean comunicar el robot con las oficinas de vigilancia usando la red inalámbrica de ese edificio. ACME te ha contratado para que adaptes a la estructura del robot un prototipo de base y desarrolles el programa que permita de, forma autónoma, "vigilar" los alrededores del edificio, transmitiendo una señal de video en tiempo real y en cada esquina del edificio se deberá efectuar una pausa en su trayectoria para realizar un giro de 360° de la cámara del dispositivo móvil acoplado, que estará conectado vía WI-FI a la red inalámbrica LAN de la empresa. La figura 2, te explica paso a paso el problema. Los directivos de la empresa adquirieron un software de comunicación denominado iVCam, que han instalado en el dispositivo móvil y en un equipo de cómputo en las oficinas de vigilancia y que actuará como un servidor de video.

El programa que debes construir en ROBOTC® debe contener las instrucciones para dar CINCO vueltas completas al edificio; como se trata de un prototipo no es necesario que ejecute cons-

Figura 2.

RECORRIDO DEL ROBOT

DE VIGILANCIA CON CÁMARA MÓVIL



tantemente. El proceso de ingeniería implica que construyas la solución paso a paso, por tal motivo, en este laboratorio se diseña la rutina de vigilancia mediante transmisión de señal de video en tiempo real.

CONCEPTOS BÁSICOS

Aquí encontrarás material de apoyo para entender el objetivo primario de este documento. En general, los apuntes provienen de los documentos que el Laboratorio Nacional de Ingeniería Robótica de la Universidad Carnegie Mellon ha proporcionado en sus cursos de robótica.

Comando wait1sec()

El comando `wait1sec ( )` jugará un papel importante en este laboratorio, le indicarás al robot el tiempo –en milisegundos– que debe esperar antes de que ejecute el siguiente comando. El número dentro de los paréntesis representa el número de milisegundos que tu robot debe esperar para poder ejecutar otra acción; en ese sentido, 3000 milisegundos equivalen a 3 segundos, así que el robot enciende su motor por tres segundos antes de ejecutar el siguiente comando. Deberás calcular el tiempo necesario para que el soporte del dispositivo móvil realice un giro de 360°, así decidirá el valor de potencia del motor más efectivo para el giro de la cámara y que la señal de video no sea perturbada por un movimiento brusco.

Asegúrate de estudiar completamente cada solución, ya que en el siguiente paso generarás el programa que resuelva el desafío. Relaciona estos conceptos con el proceso de ingeniería que se te ha presentado en la sección “Conceptos básicos” del Laboratorio 3.

SOLUCIÓN DEL DESAFÍO

Ahora se te proporcionan tres herramientas que permiten resolver el problema.

1) Solución del problema con una serie de pasos (Tabla 1) explicados en idioma español.

2) Solución con pseudocódigo (Tabla 2), expresado también en idioma español concreto.

3) Solución con algoritmo usando la notación de diagramas de flujo (Figura 3).

Tabla 1.

SOLUCIÓN PARECIDA A UNA RECETA DE COCINA.

(continúa en la pagina siguiente)

1. Inicio del programa
2. Número de vuelta es igual a 0
3. Medir el extremo más largo del rectángulo. Anota tu respuesta: _____
4. Si ya va en la sexta vuelta ir al paso 18.
5. Haz que el robot viaje del punto A al B a media potencia. Usando el cronómetro del celular determina cuánto tiempo le toma al robot. Apaga el motor al llegar al extremo contrario del marco. Considera que al llegar al extremo contrario del punto de partida el robot deberá realizar una pausa y activar el motor de soporte del dispositivo móvil que sostiene la cámara para efectuar un giro de 360°. Anota el tiempo en milisegundos y la potencia de motor necesarios para el giro: _____
6. El robot debe continuar su recorrido, y para esto debe dar vuelta a la izquierda, por lo que debe tener suficiente espacio para no chocar con el edificio.
7. Ahora, que gire el robot a la izquierda (aplica un ¼ de potencia al motor del robot). ¿Cuánto tiempo debe aplicar fuerza a las ruedas para dar un giro de 90°? Anota tu respuesta: _____.

Figura 3.

SOLUCIÓN PARECIDA A UNA RECETA DE COCINA.

(finaliza)

8. Haz que el robot viaje del punto B al C a ½ potencia. Usando los conceptos de matemáticas que ya conoces (regla de tres) determina cuánto tiempo le toma al robot alcanzar la siguiente esquina del edificio. Haz los ajustes necesarios a tu programa. Anota tu primera respuesta: _____
9. Ahora, que gire el robot a la izquierda (aplique un ¼ de potencia al motor del robot).
10. Haz que el robot viaje del punto C al D a ½ potencia.
11. Activa el motor del soporte del dispositivo con cámara y que gire 360°.
12. Ahora, que gire el robot a la izquierda (aplica ¼ potencia al motor del robot)
13. Haz que el robot viaje del punto D al A a ½ potencia
14. Activa el motor del soporte del dispositivo con cámara y que gire 360°.
15. Ahora, que gire el robot a la izquierda (aplica ¼ de potencia al motor del robot).
16. Número de vuelta = número de vuelta + 1.
17. Ve al paso 4.
18. Final del programa.

Fuente: Elaboración propia.

Tabla 2.

PSEUDOCÓDIGO PARA EL DESAFÍO

DEL ROBOT DE VIGILANCIA CON VIDEO.

(continúa en la pagina siguiente)

```
// Desafío Robot de Vigilancia con Video
// El robot da 5 vueltas alrededor del edificio rectangular
// por lo que el tiempo de recorrido en cada recta puede variar
// En cada esquina debe realizar una pausa y realizar giro del soporte del dispositivo
// Posteriormente realizar gire 90 grados para continuar recorrido

task main()
[
     int no Vuelta = 0;
     // Se daran 3 vueltas

     While (NoVuelta < 6)
     [
          // Movimiento del punto a al punto B
```

Tabla 2.

PSEUDOCÓDIGO PARA EL DESAFÍO

DEL ROBOT DE VIGILANCIA CON VIDEO.

(continúa en la pagina siguiente)

```
Avanzar el robot x segundos
// Detener y esperar x segundos
activar motor de soporte de cámara de video y girar 360 grados
// Giro en la esquina activando motores de ruedas
girar a la Izquierda
// Movimiento del punto B al punto C
avanzar el robot x segundos
// Detener y esperar x segundos
activar motor de soporte de cámara de video y girar 360 grados
// Giro en la esquina
girar a la Izquierda
// Movimiento del punto C al punto D
avanzar el robot x segundos
// Detener y esperar x segundos
```

Tabla 2.

PSEUDOCÓDIGO PARA EL DESAFÍO

DEL ROBOT DE VIGILANCIA CON VIDEO.

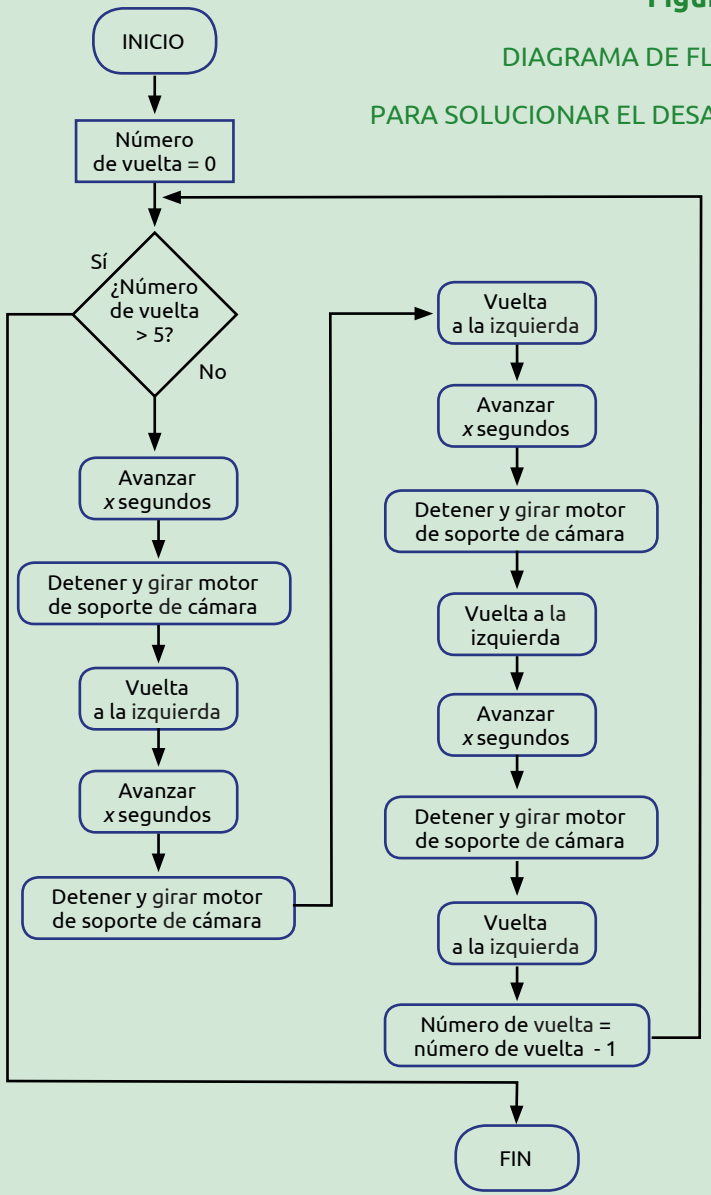
(finaliza)

```
                activar mortor de soporte de cámara de video y girar 360 grados
                // giro en la esquina
                girar a la Izquierda
                // Movimiento del punto D al punto A
                avanzar el robot x segundos
                // Detener y esperar x segundos
                activar motor de soporte de cámara de video y girar 360 grados
                // Giro en la esquina
                girar a la Izquierda
                // Contador de vueltas
                NoVueltas = NoVuelta + 1
            ]
        ]
```

Figura 3.

DIAGRAMA DE FLUJO

PARA SOLUCIONAR EL DESAFIO.



Anote en el recuadro el programa resultante con que diste solución a los pasos de este desafío.

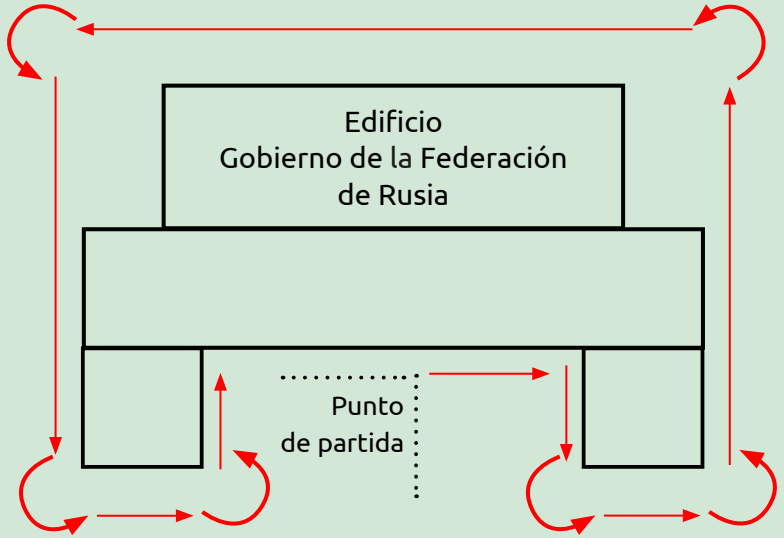
TRABAJO ADICIONAL

Los desafíos que se encuentran en esta sección deben ser resueltos por el estudiante, es necesario crear un algoritmo y el programa que da solución a cada desafío.

* El Gobierno de Rusia posee un edificio que concentra oficinas de inteligencia y su poder militar. Si ACME Robotics tuviera que vigilar dicho complejo con el mismo robot y enviar la señal de video del entorno en tiempo, ¿qué cambios tendría que hacer a su programa para dar cinco vueltas completas al edificio? Construye tu algoritmo y el programa en ROBOTC® para hacer el recorrido. Usa la maqueta desarrollada en el laboratorio 3 para probar que el robot está funcionando adecuadamente. Observa la figura 4, para que tengas idea del recorrido del robot y consideres que en las aristas principales del edificio, el robot se detendrá y efectuará un giro de 360° del soporte que contiene el dispositivo móvil (cámara).

Figura 4.

RECORRIDO DEL ROBOT DE VIGILANCIA PARA EL EDIFICIO DEL GOBIERNO DE RUSIA.



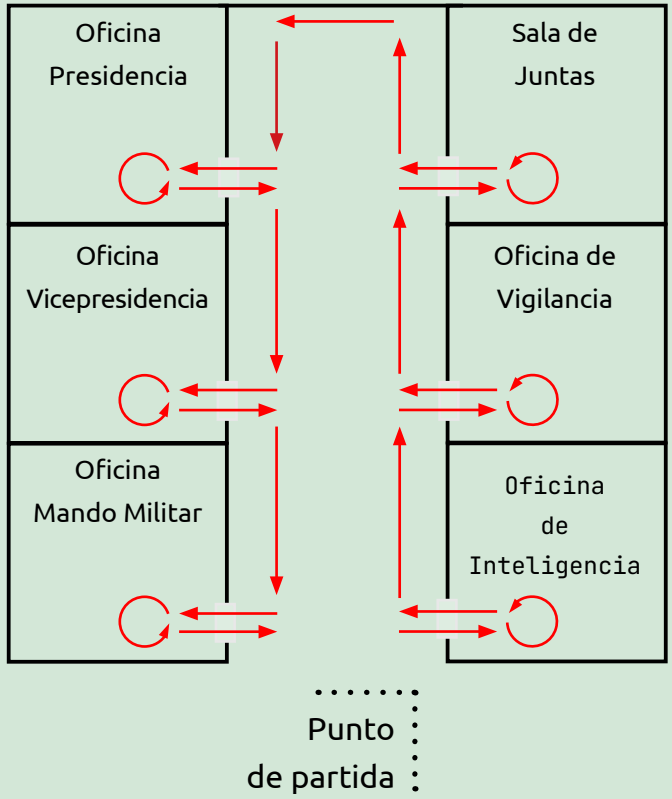
Fuente: elaboración propia.

Un proyecto adicional ha sido asignado a ACME Robotics. El Gobierno de Rusia desea implementar un segundo robot para vigilancia autónoma dentro del edificio, que recorrerá durante el horario nocturno. ¿Qué cambios tendrías que hacer a tu programa para que el robot dé tres vueltas completas al pasillo de las oficinas principales? Construye tu algoritmo y el programa en ROBOTC® para hacer el recorrido. Usa la maqueta desarrollada en el laboratorio 3. Observa la figura 5, para que tengas idea del recorrido del robot; debes considerar una pausa en la puerta de acceso de cada oficina, y efectuar un movimiento giratorio de la cámara para trasmitir la señal de video de esa esa zona, posteriormente debe continuar el recorrido a la siguiente puerta más próxima.

Figura 5.

RECORRIDO DEL ROBOT

DE VIGILANCIA PARA PASILLO DE OFICINAS



Fuente: elaboración propia.

Referencias

Robomatter Inc. (2016). ROBOTC for LEGO Mindstorms 4.X - Users Manual. Disponible para consulta en el sitio web: https://www.robotc.net/WebHelpMindstorms/index.htm#Resources/topics/ROBOTC_Interface/Overview.htm

Acerca de iVCam. (s.f). Consultado en el mes de agosto de 2020, en el sitio web https://www.e2esoft.com/ivcam/

Enrlich, S.D. (2011). Access Points: An Institutional Theory of Policy and Policy Complexity. New York: Oxford.

CISCO (2019). ¿Qué es un access point? Disponible para consulta en el sitio web: https://www.cisco.com/c/es_mx/solutions/small-business/resource-center/networking/what-is-access-point.html

CONSIDERACIONES FINALES

Con el objetivo de tener un mayor impacto en el lector, se le recomienda utilizare un kit completo del Robot LEGO Mindstorm EV3, adquirir una licencia del lenguaje de programación ROBOTC®, usar un dispositivo móvil con cámara y WIFI, y que aprenda el armado del robot, identificando las piezas y sensores que lo integran para facilitar su toma de decisiones durante el proceso de codificación.

Otro aspecto que consideramos relevante es crear un cuadernillo de ejercicios o bitácora de control para tomar nota de ciertos parámetros o datos de control que permitan soluciomar los problemas que se plantean en los diferentes temas; y con esto, crear una guía de referencia para futuros algoritmos que el lector pueda desarrollar para perfeccionar su aprendizaje.

Finalmente, se recomienda al lector implementar un repositorio de códigos, utilizando herramientas *open source* como `github`, para que pueda llevar un control de sus avances y respaldo de los mismos. Así mismo, es fundamental la validación e instalar el firmware correcto en el robot antes de iniciar con las pruebas del código.

CONCLUSIONES

En una sociedad globalizada donde la innovación tecnología avanza de manera significativa día a día, resulta inevitable que el ámbito de la educación primordial no se vea afectado por la misma. Consideramos que la robótica será un área de conocimiento que se implementará desde la educación básica en sus planes de estudios en los próximos años.

Estamos convencidos de que esta obra literaria despertará en el lector, y principalmente en los estudiantes, el interés por mejorar sus habilidades en programación con un enfoque en los principios de la robótica mejorando su razonamiento y creatividad; aplicando los fundamentos y metodologías de la programación, como: estructuras secuenciales, decisión y repetitivas; manipulación de datos en arreglos multidimensionales, diseño de funciones y archivos de texto.

Como profesores investigadores y autores de esta obra de texto educativa, estamos conscientes de que es neceario desarrollar en los jóvenes nuevas competencias relacionadas con la lógica de programación y mejorar sus capacidades cognitivas, aprovechando esa inmensa capacidad que poseen de explorar nuevas cosas. Todo esto permitirá formar nuevas generaciones de ingenieros que se apasionen por la ciencia y tecnología, capaces de resolver los retos del futuro.

Introducción a la robótica,
de Luis Antonio Álvarez Oval, Christian Mauricio
Castillo Estrada y Aron De la Cruz Vázquez, se terminó
de editar en junio de 2022 y se usaron tipos Ubuntu y
JetBrains mono.