Desarrollo de algoritmo de control y navegación para robot hexápodo.

Ingeniería electrónica, Facultad de Ingeniería, Universidad Nacional de Mar del Plata.

Integrantes:

Bendlin, Federico. Matrícula 11592.

Benintende, Mariano. Matrícula 10610.

Tutores:

Dra. Juana G. Fernández.

Ing. Walter A. Gemin.

1. RESUMEN

Los procesos productivos actuales se encuentran marcados por el empleo de sistemas mecánicos y electrónicos, basados en computadoras, para la operación y el control de la producción. En este contexto, la enseñanza de la robótica a nivel universitario se centra en el desarrollo de proyectos que faciliten el aprendizaje de los conocimientos en esta área.

El presente Proyecto Final de carrera surge como propuesta del Laboratorio de Instrumentación Virtual y Robótica Aplicada de la Facultad de Ingeniería de la Universidad Nacional de Mar del Plata. Se trata de la implementación de un robot insecto de seis patas (hexápodo) para su uso en prácticas de la asignatura Robótica Aplicada, con especial énfasis en el desarrollo de los algoritmos de comando, navegación y control del mismo.

Para la implementación de los algoritmos se partió de un robot hexápodo construido en plástico, dotado con una placa Arduino Nano. Arduino es una tecnología que se encuentra en auge debido a su versatilidad, bajos costos y alta reproducibilidad.

Los movimientos del hexápodo están a cargo de tres servos que trabajan conjuntamente para realizarlos. Los servos son motores de corriente continua con la diferencia de que se les puede controlar el ángulo de giro según sea necesario.

Posee un sensor ultrasónico mediante el cual realiza mediciones de distancia. Este tipo de sensor funciona enviando una señal al aire y esperando recibir un rebote de la misma, llamado “eco”.

El hexápodo también tiene un par de microswitches que permiten detectar colisiones con objetos.

Para implementar los algoritmos se utilizó el entorno de desarrollo propio de Arduino. Este software, además de estar especialmente diseñado para trabajar con placas Arduino, posee numerosas librerías para los diversos accesorios que se pueden utilizar a través de las cuales se simplifica la configuración y programación de la placa, así como la utilización de los dispositivos conectados a ella.

Para el control del hexápodo y la presentación de la información de manera clara, se desarrolló una interfaz gráfica en computadora, mediante el lenguaje de programación Visual Basic. Este lenguaje permite realizar aplicaciones con rapidez y de manera simple.

La comunicación entre el robot y la computadora se realiza de manera inalámbrica, mediante un módulo Bluetooth. Este protocolo es muy eficiente para el envío y recepción de pocos datos a una distancia cercana entre dos o más dispositivos.

Se implementaron los algoritmos necesarios para las tareas propuestas. El robot es capaz de mapear el recinto en el cual se mueve, midiendo las dimensiones del mismo. Al proveerle coordenadas de destino el hexápodo puede desplazarse de manera autónoma hacia éstas, y ante la aparición de obstáculos en su camino es capaz de detectarlos y esquivarlos.

Toda la información de las actividades del robot es presentada de manera gráfica en la pantalla de la computadora, en una interfaz clara y amigable con el usuario.

A pesar de ciertas limitaciones, tanto mecánicas como electrónicas, se pudo lograr un desarrollo adecuado para el objetivo del proyecto y se podría usar como base para diseños futuros.

2. INTRODUCCION

El objetivo de la enseñanza de la robótica, es lograr una adaptación de los alumnos a los procesos productivos actuales, en donde la automatización (tecnología que está relacionada con el empleo de sistemas mecánicos, electrónicos y basados en computadoras, en la operación y control de la producción) juega un rol muy importante.

La principal vertiente en los proyectos de robótica educativa a nivel universitario se centra en el desarrollo de proyectos de investigación que permitan la implementación de prácticas, que incentiven y motiven al estudiante en el aprendizaje de conocimientos asociados a los sistemas de control y robótica en general. A niveles más avanzados se consideran como aspectos principales, la prueba y validación de arquitecturas de control de robots, examinar algoritmos de control de navegación autónoma, semiautónoma y con diferentes configuraciones de sensores, actuadores y demás dispositivos electrónicos.

En este contexto de la enseñanza de la robótica, se encuadra este Proyecto Final, dado que surge como un requerimiento del Laboratorio de Instrumentación Virtual y Robótica Aplicada, para el desarrollo de algoritmos para la enseñanza de técnicas de reconocimiento y navegación de robots hexápodos.

- Objetivos principales
    - Actuar de manera autónoma.
    - Ir de una posición inicial a una posición final.
    - Esquivar objetos en el camino.

- Objetivos específicos
    - Medir distancia mínima a las paredes.
    - Representar gráficamente: recinto, posición actual y final del robot, trayecto y obstáculos.
    - Representar numéricamente la posición inicial y final.

3. ANTEPROYECTO

El robot con el que se trabajó se encuentra en la categoría de robot Zoomórfico. Este tipo de robots están en permanente evolución, la capacidad y semejanza con la especie animal permiten su adaptación a distintas geografías, por lo cual su estudio y experimentación es de importancia.

Los usos de este tipo de robots son muy diversos: operaciones de rescate, tecnología de inteligencia (su forma animal, permite obtener información sin ser descubiertos), prevención de catástrofes mediante monitoreo remoto (por ejemplo incendios forestales), estudio y monitoreo de actividad en volcanes y en general acceso a lugares hostiles y de riesgo para el ser humano.

Pueden ser de dos tipos:

- Con patas: Dependiendo el número de ellas pueden ser cuadrúpedos (robot que se utilizan como mascotas), hexápodos (imitan las características del movimiento de los insectos), etc.

- Sin patas: Dentro de este grupo están los robots conocidos como gusano o serpiente que asimilan el movimiento de estos.

El robot utilizado en este proyecto es un hexápodo (Fig. 1). Posee una estructura liviana para mantener bajo su consumo y alta autonomía, 3 servos para su desplazamiento, un sensor de ultrasonido y dos microswitch para la detección de objetos, un porta pilas con switch on/off para la alimentación en funcionamiento autónomo, comunicación remota mediante un módulo Bluetooth y una placa Arduino Nano conectada a un shield de expansión, la que ejecuta el programa de navegación y comportamiento.

Existen varias razones por las cuales utilizar Arduino. Son placas de bajo costo, tamaño reducido, alta velocidad y capacidad de cálculo. Existen una gran variedad de accesorios (sensores, shields, etc.) pensados para funcionar con ellas, que las hacen muy versátiles. Existe una extensa variedad de librerías para la utilización de la placa y los distintos accesorios. Son de fácil disponibilidad en el país en varios modelos con distintas características como tamaño, cantidad de pines, capacidad de memoria, con la ventaja de compatibilidad de software entre ellas.

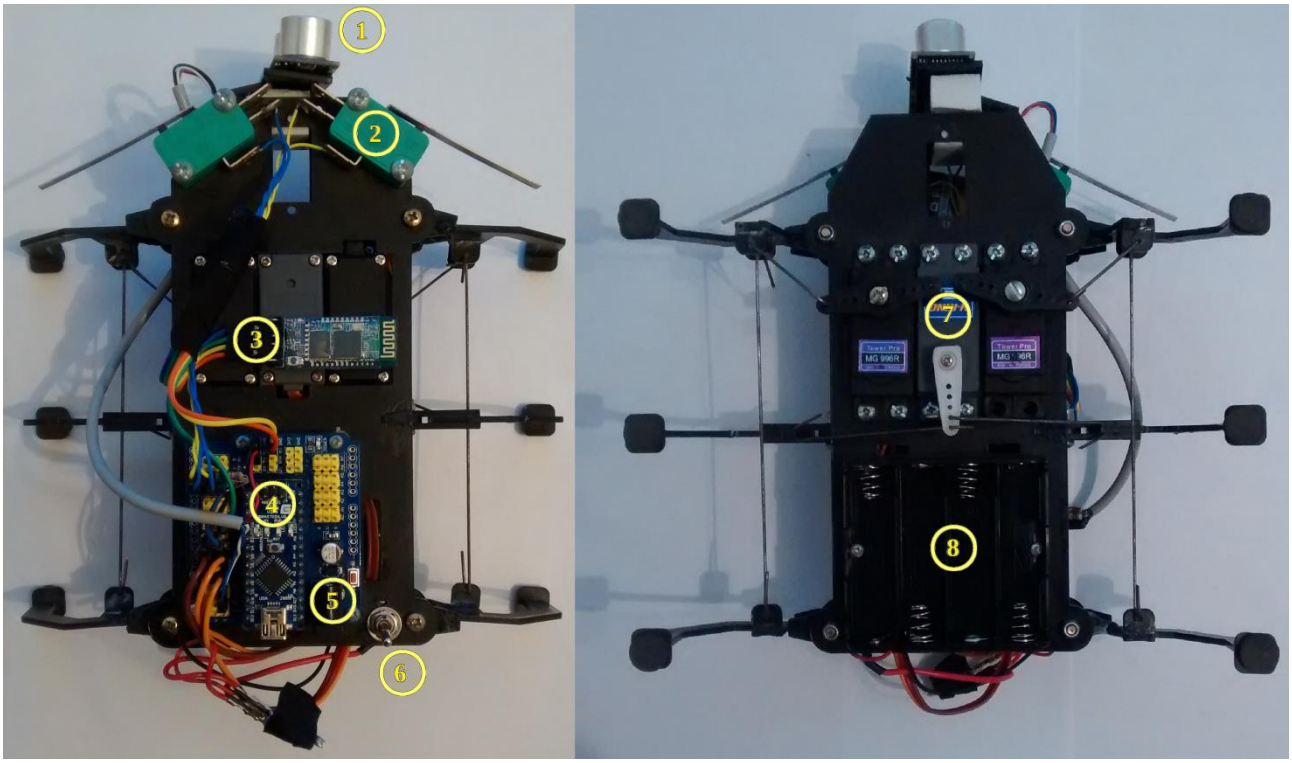


Fig. 1. Componentes del hexápodo

A continuación se describen los componentes del hexápodo numerados en la Fig. 1:

1- Sensor ultrasónico: emiten pulsos de ultrasonido que se reflejan en los objetos. El sonido reflejado o “eco” es posteriormente recibido por el sensor, el cual genera una señal de salida para ser usada con un actuador, controlador, o una computadora. La señal de salida puede ser tanto analógica como digital. Los sensores ultrasónicos son capaces de detectar la mayoría de los objetos (metales o no metales, claros u opacos, líquidos, sólidos o granulados). Sin embargo, los objetos que absorben el sonido (ropa, harina, espuma, etc.) generan ecos débiles y como resultado mediciones erróneas.
El sensor utilizado en el hexápodo es el HC – SR04.

2 - Microswitch: es un actuador eléctrico (interruptor), que es activado por una fuerza física pequeña. El switcheo ocurre de forma confiable en posiciones específicas y repetibles del actuador. Son muy comunes debido a su bajo costo y durabilidad. Tienen un uso muy difundido, entre sus aplicaciones están los electrodomésticos, maquinarias, controles industriales, vehículos, etc. Generalmente son utilizados para sensar posiciones de elementos de un sistema, señalizar estados, cerrar circuitos, encender pequeños motores, solenoides, lámparas u otros dispositivos. Algunas versiones más sensibles pueden sensar monedas en máquinas expendedoras o ser usados como parte de sensores de presión, flujo o temperatura.

En el proyecto se usaron los microswitch Dong Hai modelo KW3-0Z

3- Módulo Bluetooth: Bluetooth es un estándar de comunicación que se encuentra en una gran variedad de productos como auriculares, controles de video juegos, computadoras, teléfonos inteligentes, etc. Consiste principalmente en un protocolo para la transmisión inalámbrica, opera a 2.4GHz y posee un alcance de aproximadamente 10 m.

Se optó por el módulo Bluetooth HC-05 por su disponibilidad, tamaño, bajo consumo y costo respecto a otras opciones (ZigBee, WiFi) y permitir la comunicación a una distancia adecuada para la finalidad de este proyecto.

4- Placa Arduino Nano: es una placa basada en el microprocesador ATmega328 de 16 MHz, con una memoria Flash de 32 KB, SRAM de 2 KB y EEPROM de 1 KB. Posee un conector mini USB, 14 pines de entrada / salida digitales (6 pueden ser PWM) y 8 entradas analógicas. Además de las razones expuestas se comprobó experimentalmente que esta placa cumplía con los requisitos de capacidad de cálculo y memoria necesarios en este proyecto.

5- Nano shield: son placas circuitales pre construidas que se conectan sobre la placa Arduino y proveen distintas funcionalidades como control de motores, conexión a internet, comunicación inalámbrica, pantalla LCD, etc.

6- Switch ON/OFF: utilizado para proveer o interrumpir la alimentación.

7- Servos: es un dispositivo pequeño basado en un motor de continua y un circuito de realimentación que permite generar un movimiento angular programado por ancho de pulso PWM. Mientras la señal de control PWM exista en la línea de entrada, el servo mantendrá la posición angular del eje. Cuando ésta señal cambia, la posición angular del eje cambia en forma proporcional.

En la práctica, los servos son usados en distintos tipos de robots, aviones, autos a radio control, mecanismos de accionamiento remoto. El servo se compone de: un control circuital, el motor propiamente dicho, varios engranajes y la carcasa. Posee 3 cables, uno es para la alimentación (5V generalmente), otro para masa y el último para el control.

El hexápodo cuenta con dos modelos diferentes de servos: dos del modelo Tower Pro MG996R, encargados de mover un par de patas delanteras y traseras del hexápodo con un servo en cada lado.
El otro modelo es el H-KING HK15138, de menor potencia encargado de mover en sentido vertical las patas centrales del hexápodo.

8- Porta pilas: Sirve para alojar las pilas y alimentar el hexápodo en su desplazamiento autónomo.

3.1. Funcionamiento

Los servos son los encargados de realizar el movimiento del hexápodo. Los dos más potentes se encuentran conectados a las patas delanteras y traseras del hexápodo; la acción de estos servos hace que las patas realicen la tracción necesaria para impulsar el hexápodo.

El servo de menor potencia actúa sobre las patas centrales, provocando la inclinación hacia un lado u otro del robot. Esto significa que cuando la pata central izquierda hace fuerza contra el piso, las patas delantera y trasera de la derecha están en condiciones de realizar la tracción. Cuando la pata central derecha es la que realiza la fuerza, las que pueden traccionar son las patas delantera y trasera del lado izquierdo.

La acción sincronizada de los 3 servos permite que el hexápodo pueda realizar un conjunto de movimientos simples: avanzar, retroceder, rotar en sentido horario o antihorario.

Los sensores de ultrasonido y los microswitch, son los encargados de detectar objetos o bordes que afectarían su desplazamiento.

El módulo Bluetooth permite una comunicación desde y hacia un dispositivo para su programación, configuración, control y obtención de datos remotos.

4. PROYECTO

Inicialmente se desarrollaron varios algoritmos comunes a todas las tareas, éstos son:

. Avanzar
. Retroceder
. Rotar en sentido horario
. Rotar en sentido antihorario
. Detectar (mediante el sensor ultrasónico)
. Medir pared (ubicarse en forma perpendicular a una pared, en sentido horario o antihorario)

Para cumplir los objetivos principales del proyecto, se identificaron tres tareas esenciales.

. 5 - Mapeo del recinto.
. 6 - Navegación en el recinto.
. 7 - Detección y rodeo de objetos.

Los cuatro procedimientos de movimiento funcionan de manera similar. Al ejecutarse comprueban las posiciones de las patas izquierda y derecha. Luego determinan y ejecutan los posicionamientos necesarios en los servos, para ejecutar el paso correspondiente.

La Fig. 2, muestra la secuencia de movimientos que se generan para el avance:

4.1. Avanzar

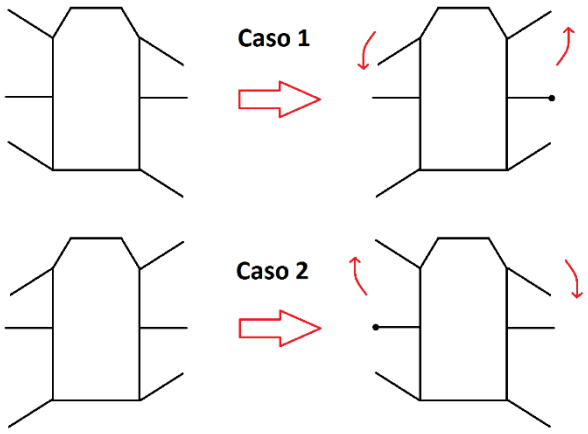


Fig. 2. Movimientos de la función avanzar.

- Si cumple el caso 1 (patas izquierdas hacia adelante, patas derechas hacia atrás) apoya la pata central derecha, mueve hacia atrás las patas izquierdas (las cuales traccionan) y hacia adelante las patas derechas.
- Si cumple el caso 2 (patas izquierdas hacia atrás, patas derechas hacia adelante) apoya la pata central izquierda, mueve hacia adelante las patas izquierdas y hacia atrás las patas derechas (estas traccionan ahora).
- Si no se cumple ninguno de los casos el programa lleva las patas a la posición del caso 1.

La Fig. 3, muestra el diagrama de flujo del algoritmo de avance descripto.

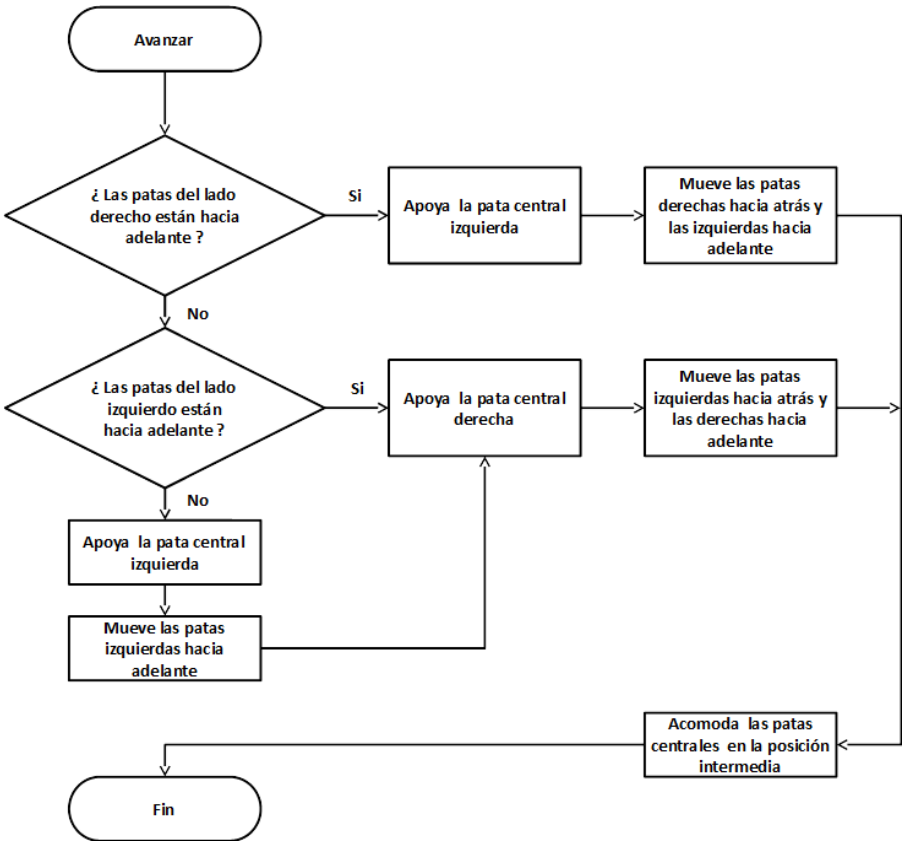


Fig. 3. Diagrama de flujo de la función avanzar.

4.2. Retroceder

La Fig. 4, muestra la secuencia de movimientos que se generan en el retroceso:

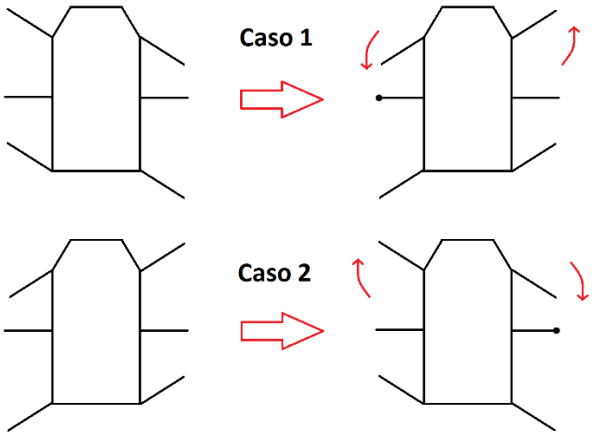


Fig. 4. Movimientos de la función retroceder.

- Si cumple el caso 1 se apoya la pata central izquierda y realiza el movimiento para este caso.
- Si cumple el caso 2 apoya la pata central derecha y realiza el movimiento correspondiente.
- Si no cumple ningún caso posiciona las patas para ejecutar el caso 2.

La Fig. 5, muestra el diagrama de flujo del algoritmo de retroceso descripto.

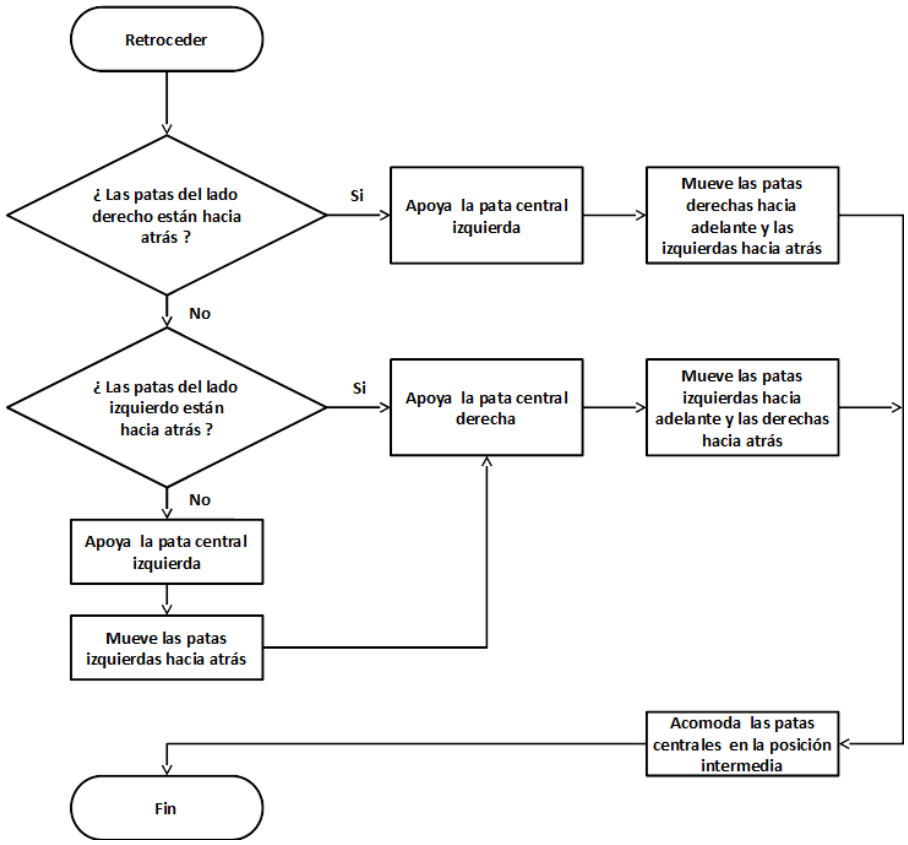


Fig. 5. Diagrama de flujo de la función retroceder.

4.3. Rotar en sentido horario

Se observa en la Fig. 6 y se describe a continuación.

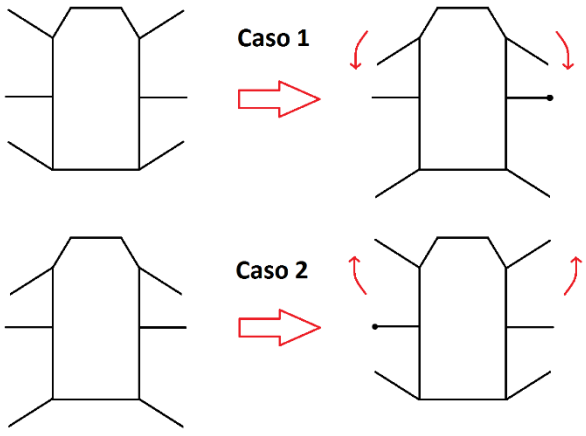


Fig. 6. Movimientos de la función rotar horario.

- Si cumple el caso 1 apoya la pata central derecha y mueve las patas izquierdas y derechas hacia atrás.
- Si se cumple el caso 2 apoya la pata central izquierda y mueve las patas izquierdas y derechas hacia adelante.
- Si no se cumple ninguno acomoda las patas al caso 1.

La Fig. 7, es el diagrama de flujo correspondiente.

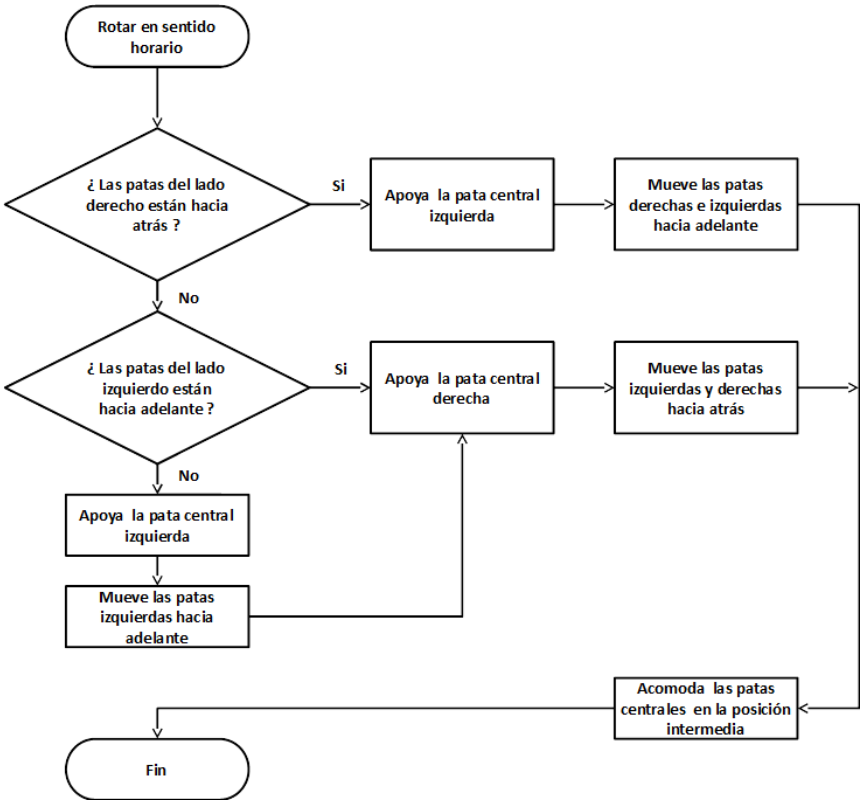


Fig. 7. Diagrama de flujo de la función rotar horario.

4.4. Rotar en sentido antihorario

Se observa en la Fig. 8 y se describe a continuación.

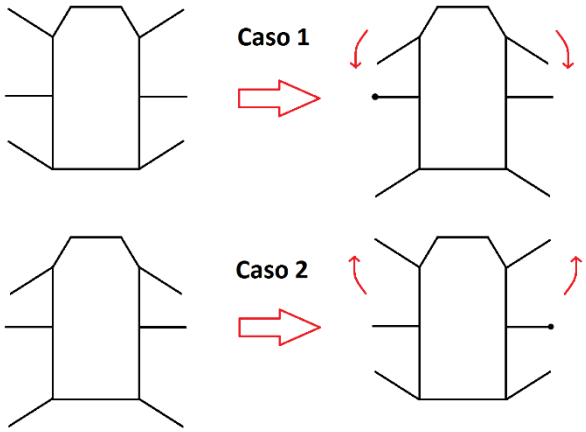


Fig. 8. Movimientos de la función rotar antihorario.

- Si cumple el caso 1 apoya la pata central izquierda y realiza el movimiento para este caso.
- Si cumple el caso 2 apoya la pata central derecha y realiza el movimiento correspondiente.
- Si no cumple ningún caso se reacomoda para cumplir el caso 1.

La Fig. 9, muestra el diagrama de flujo del movimiento anterior.

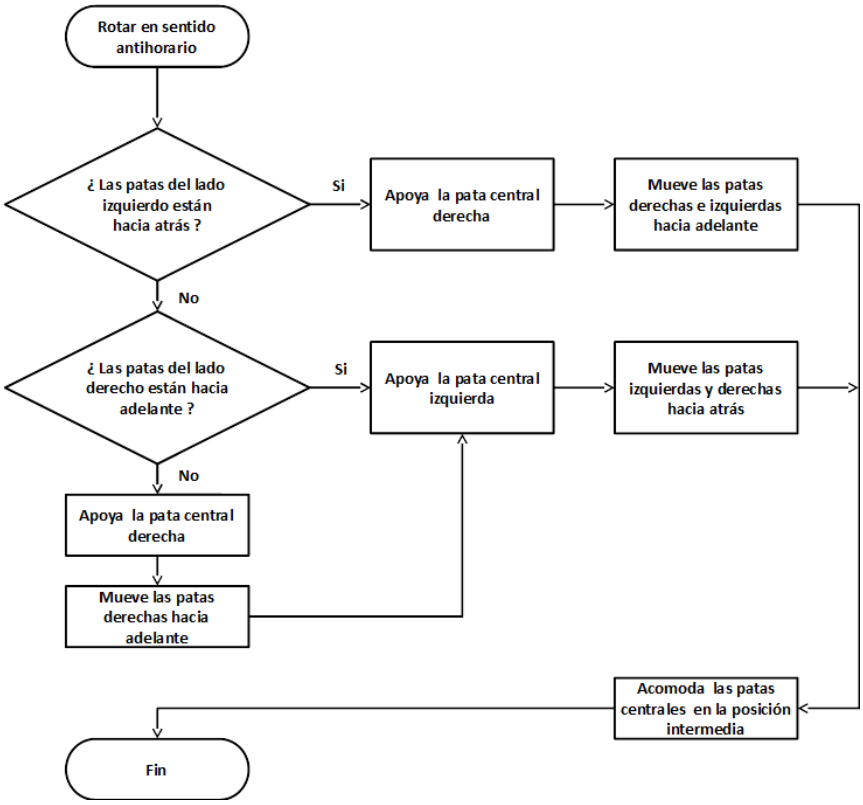


Fig. 9. Diagrama de flujo de la función rotar antihorario.

4.5. Detectar

Durante las pruebas realizadas con el hexápodo, se observan distintos comportamientos del sensor ultrasónico, que permiten registrar dos situaciones que suceden en forma esporádica, ante determinadas distancias y tipo de obstáculos. Los comportamientos registrados son:

- De manera ocasional, el sensor devuelve un valor correspondiente a una distancia menor a la pared medida, provocando que el robot suponga la aparición de un objeto.
- Otra situación que sucede en forma esporádica es que, al intentar medir la pared frente al robot estando muy cerca de la pared lateral, el eco recibido corresponde a ésta última, dando lugar a una medición errónea.

Para solucionar los problemas descriptos, se desarrolló el algoritmo del diagrama de flujo de la Fig.10, que se detalla a continuación.

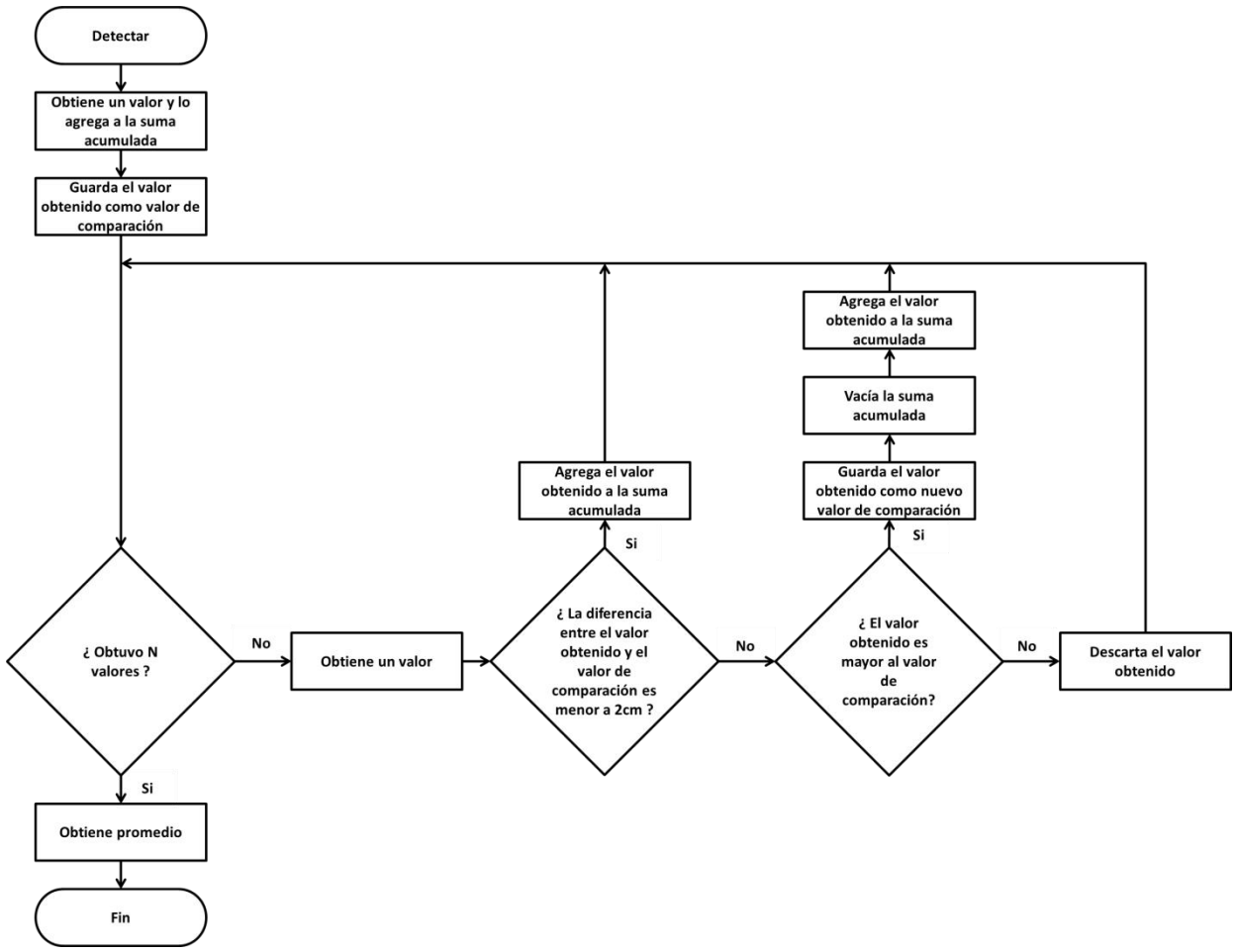


Fig. 10. Diagrama de flujo de la función detectar.

El algoritmo se basa en obtener el promedio de N mediciones.

Como se mencionó anteriormente, el sensor presenta el inconveniente de que, de manera ocasional, devuelve algunos valores que son menores a la distancia que está midiendo. Para solucionar este problema, el algoritmo descarta las mediciones que sean mucho más chicas a un valor de comparación.

El hexápodo obtiene un primer valor el cual utiliza como valor de comparación.

A partir de este momento con cada nuevo valor adquirido pueden ocurrir tres situaciones:

- Si la nueva medida difiere del valor de comparación en menos de 2 cm, por exceso o por defecto, la considera válida y la agrega a la suma acumulada.
- Si el valor nuevo es menor al de comparación por una diferencia de más de 2 cm, este valor es descartado.
- Si el valor es mayor al de comparación en por lo menos 2 cm, se considera que las mediciones realizadas hasta ese momento, incluyendo el valor de comparación, no son válidas. El último valor medido es tomado como nuevo valor de comparación, se vacía la suma acumulada hasta el momento y se inicia una nueva.

Una vez obtenidos los N valores válidos realiza el promedio y devuelve el valor final.

4.6. Medir pared

El objetivo de este algoritmo es lograr que el robot quede perpendicular a una pared. Esto es importante ya que referencia tanto su posición como el ángulo dentro del recinto, por lo tanto es necesario que sea lo más exacto posible.

El principal problema es que cuando el robot se encuentra midiendo en direcciones muy cercanas entre sí, separadas por un ángulo pequeño, los valores obtenidos son muy cercanos como para poder distinguirlos correctamente aun realizando promediación. Por lo tanto se diseñó un procedimiento que consta de dos partes.

4.6.1. Primera parte

Correspondiendo al diagrama de flujo de la Fig. 11

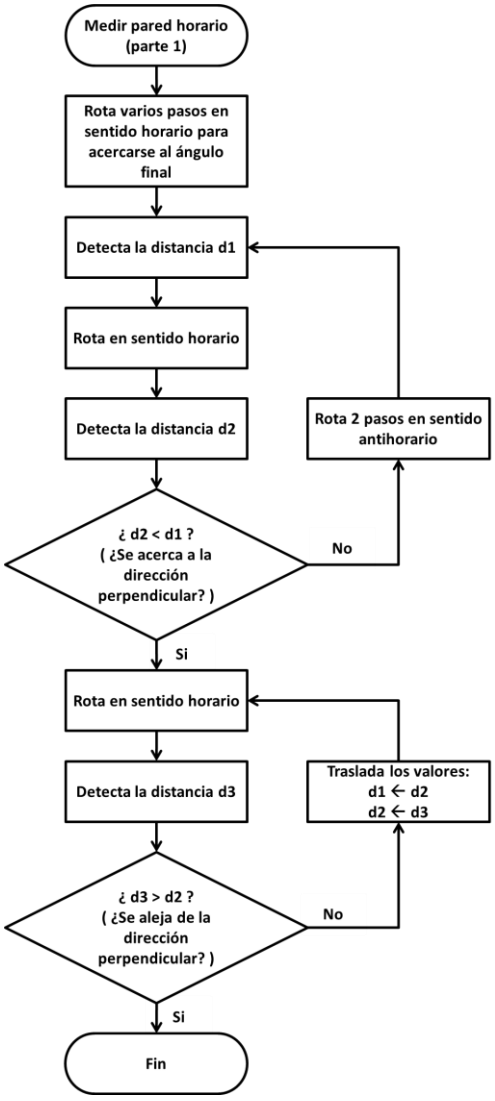


Fig. 11. Diagrama de flujo de la primera parte de medir pared (sentido horario).

Inicialmente el hexápodo rota en sentido horario (realizando pasos de alrededor de 8°) sin detectar hasta acercarse a la dirección perpendicular a la pared. Luego realiza la primera detección obteniendo la distancia d1, rota un paso en sentido horario y vuelve a detectar obteniendo d2.

A partir de este punto existen dos alternativas:

- Si la distancia d2 es menor que d1 significa que el hexápodo se está acercando a la dirección perpendicular, con lo cual hará un giro más en sentido horario para obtener d3 y seguirá ejecutando el resto del método.
- Si la distancia d2 es mayor que d1 significa que el hexápodo se está alejando de la dirección perpendicular, por lo tanto realiza dos giros en sentido antihorario para obtener nuevos valores de d1 y d2. Este proceso será iterativo hasta que d1 y d2 cumplan la condición del primer caso.

A continuación compara d3 con d2 pudiendo ocurrir nuevamente dos situaciones:

- Si d3 es mayor a d2 significa que la dirección en la que detectó d2 es la más cercana a la perpendicular, y constituye la primera aproximación a esa dirección.
- Si la distancia d3 obtenida es menor a d2, el hexápodo todavía no se pasó de la perpendicular por lo que traslada el valor de d2 a d1, el de d3 a d2, rota en sentido horario y obtiene un nuevo d3. Este proceso se realizará de manera iterativa hasta que se cumpla el primer caso.

Luego de finalizada la primera parte, el hexápodo tiene tres detecciones válidas: d1, d2 y d3. Siendo la dirección de d2 la primera aproximación a la perpendicular de la pared (Fig. 12).

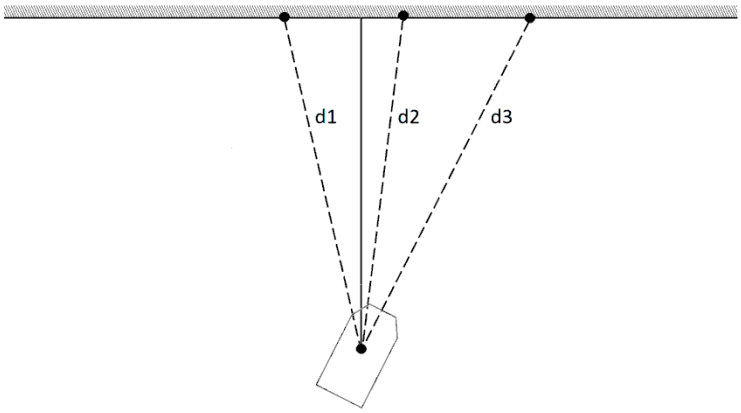


Fig. 12. Esquema de las tres detecciones válidas para la primera parte de medir pared.

4.6.2. Segunda parte

Correspondiendo al diagrama de flujo de la Fig. 13.

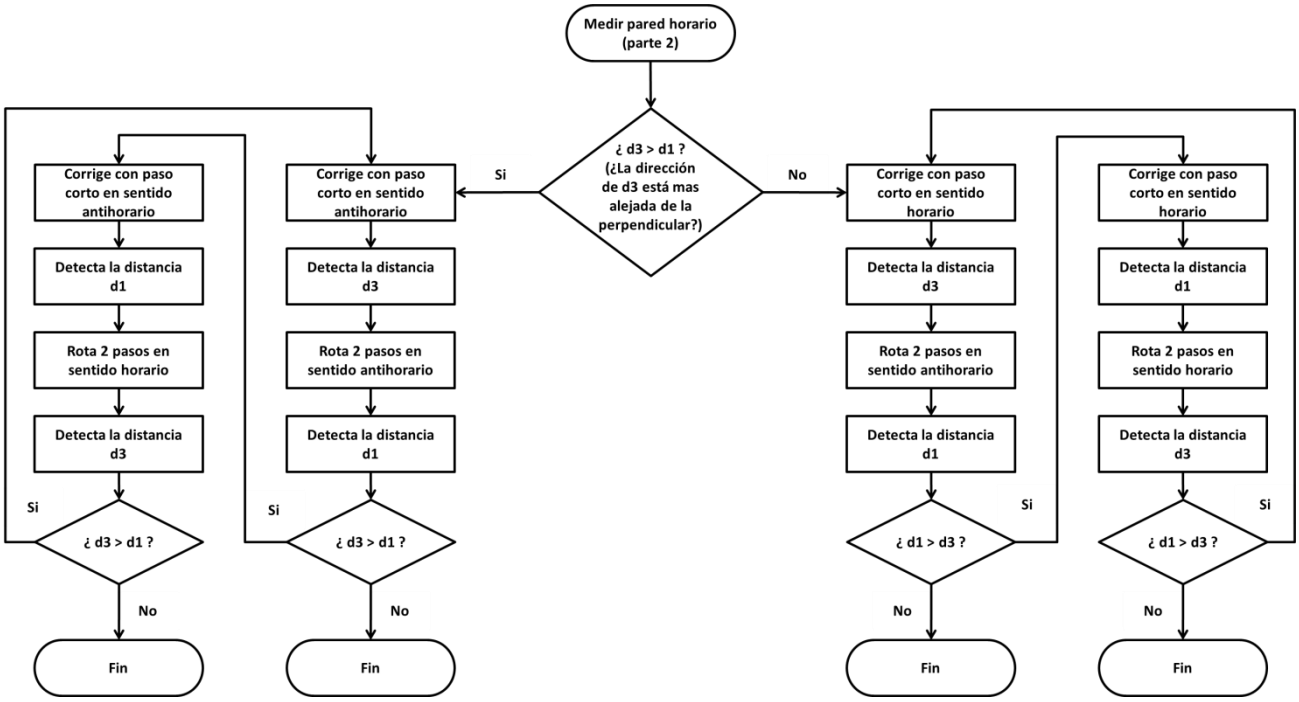


Fig. 13. Diagrama de flujo de la segunda parte de la función medir pared (sentido horario).

Luego de haber encontrado la primera aproximación a la dirección perpendicular, el robot realiza un proceso iterativo en el que se corrige haciendo pasos cortos (de alrededor de 4°) para acercarse a la verdadera dirección perpendicular.

El hexápodo compara d1 con d3 y gira en el sentido de la distancia que sea menor (horario si d3 es menor, antihorario si d1 es menor). Luego obtiene un nuevo valor de d3 (d3'), realiza dos giros en sentido antihorario (con pasos grandes de 8°) y actualiza d1 (d1'). Ver Fig. 14.

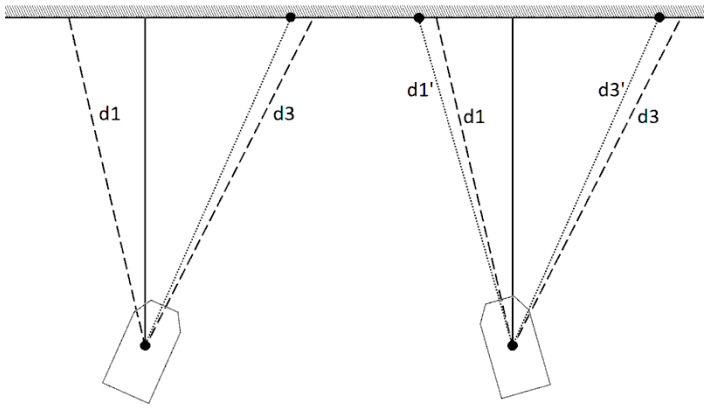


Fig. 14. Corrección de la distancia perpendicular a la pared.

Esta proceso de corrección se ejecutará hasta que d1 y d3 sean prácticamente iguales, como en la Fig. 15.

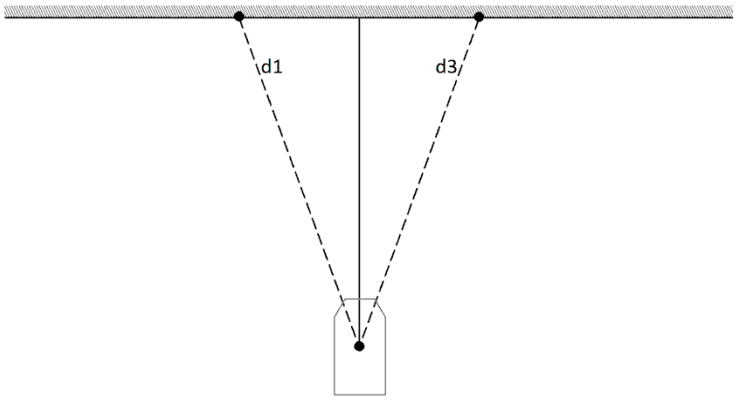


Fig. 15. Corrección finalizada.

Análogamente, las dos partes de la función medir pared tienen su método en sentido antihorario (Fig.16 y Fig. 17).

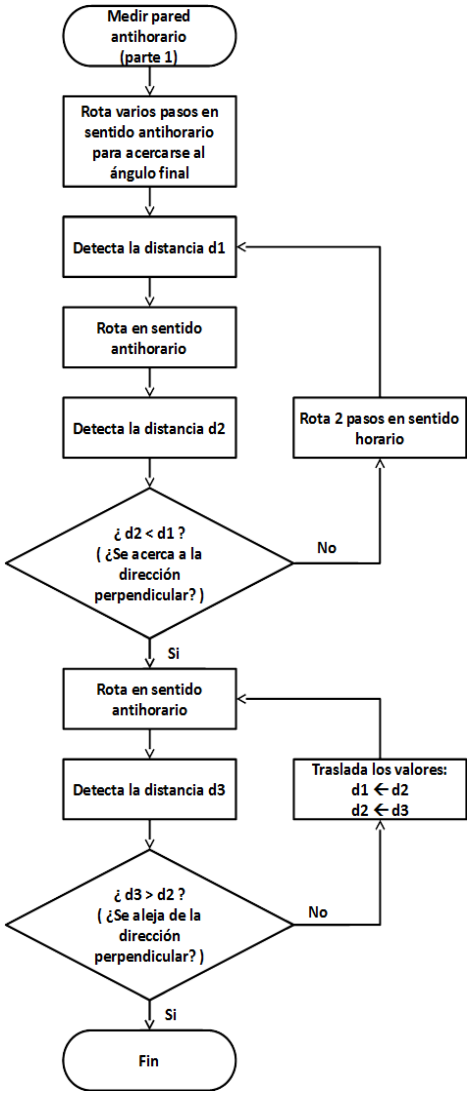


Fig. 16. Diagrama de flujo de la primera parte de medir pared (sentido antihorario).

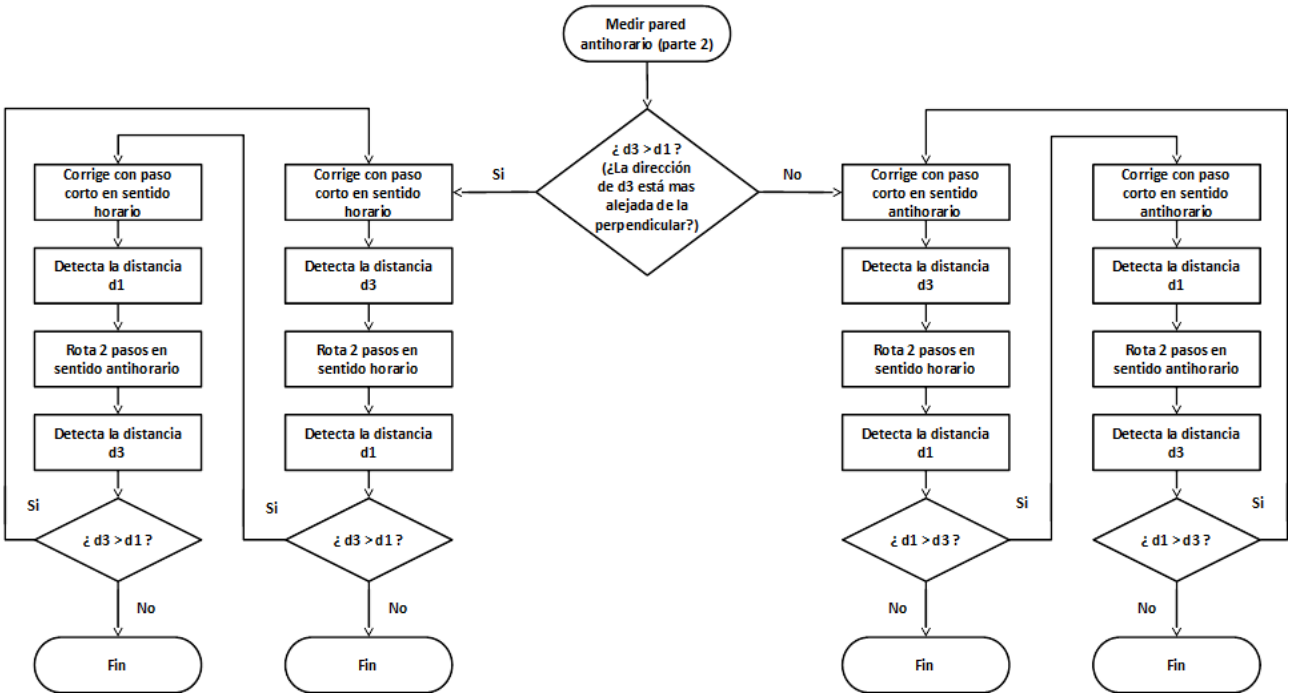


Fig. 17. Diagrama de flujo de la segunda parte de la función medir pared (sentido antihorario).

5. Mapeo del recinto

Para el mapeo la idea fue que el hexápodo, mediante el sensor ultrasónico, encontrara la distancia a cada una de las 4 paredes del recinto para calcular las dimensiones del mismo.

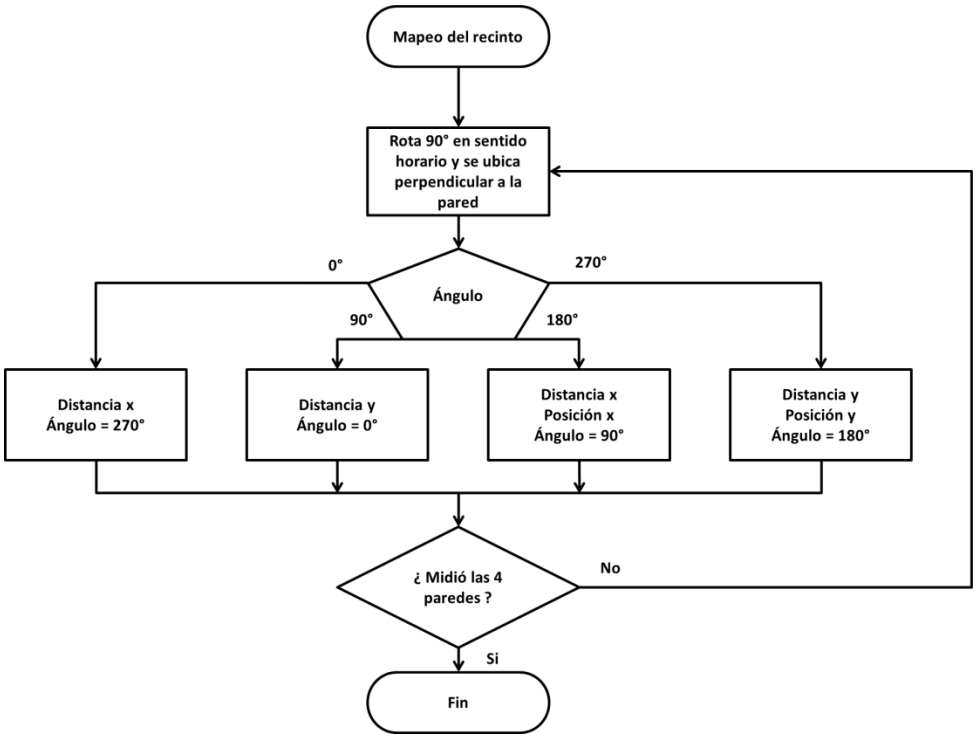


Fig. 17. Diagrama de flujo de la función de mapeo del recinto.

El método permite que el robot quede perpendicular a cada una de las cuatro paredes y mida esas distancias.

Según el ángulo que posee en cada instancia calculará distintos parámetros:

- 0° y 180°: La suma de las mediciones en estos dos ángulos resultará en la longitud total del recinto en la dirección X. A su vez, la medición en 180° dará la posición inicial en X del robot.
- 90° y 270°: La suma de las mediciones en estos dos ángulos resultará en la longitud total del recinto en la dirección Y. A su vez, la medición en 270° dará la posición inicial en Y del robot.

Al terminar este procedimiento el hexápodo se encuentra listo para recibir coordenadas e interpretarlas.

6. Navegación en el recinto

Para que el hexápodo logre ir de un punto a otro lo primero que se buscó fue que llegara a destino en dos tramos, en un primer tramo que alcanzara una de las coordenadas, y en el segundo tramo que lograra llegar a la otra, siempre haciendo caminos en "L".

Esta idea procura resolver las dificultades que se presentan para controlar el ángulo del desplazamiento del robot y actualizar la posición instantánea (al carecer de una pared de referencia) si se moviera en forma diagonal.

También se pretendía que el hexápodo minimizara la cantidad de veces que debía quedar enfrentado a una pared, principalmente porque el método para lograrlo implica un tiempo importante (aproximadamente 30 segundos), por lo tanto no era recomendable que tuviera que aplicarse más veces de las necesarias.

Primero se pensó cuales movimientos tendría que hacer el robot en cada cuadrante para llegar a destino. Se detallan las 4 posibilidades:

1er cuadrante	Avanza hasta la primer coordenada, rota 90° en sentido horario y avanza hasta la segunda coordenada.
2do cuadrante	Avanza hasta la primer coordenada, rota 90° en sentido antihorario y avanza hasta la segunda coordenada.
3er cuadrante	Rota 90° en sentido antihorario, avanza hasta la primer coordenada, vuelve a rotar 90° en sentido antihorario y avanza hasta la segunda coordenada.
4to cuadrante	Rota 90° en sentido horario, avanza hasta la primer coordenada, vuelve a rotar 90° en sentido horario y avanza hasta la segunda coordenada.

Fig. 18. Tabla de movimiento del hexápodo según el cuadrante.

Con los movimientos ya establecidos para los cuatro cuadrantes se pensó lo siguiente: según el ángulo y la posición actual del robot, se analiza dónde está la posición final y se calcula a que cuadrante respecto del hexápodo corresponde.

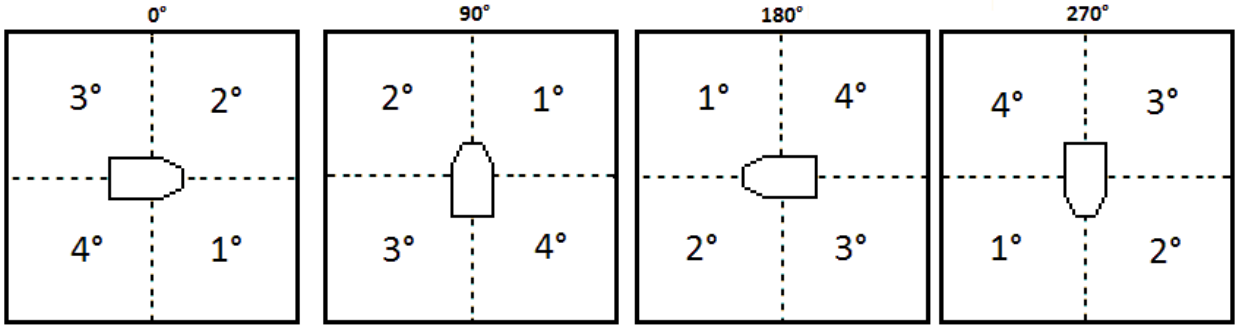


Fig. 19. Cuadrantes del hexápodo según el ángulo actual.

Para que recorra los tramos en “L” se diseñó el siguiente procedimiento:

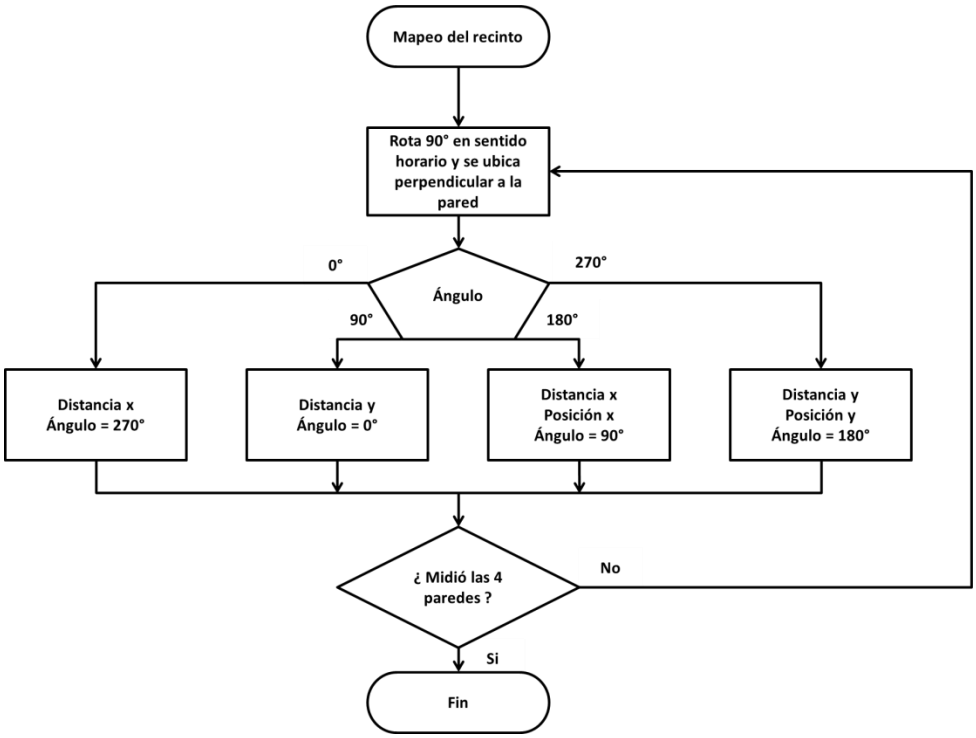


Fig. 20. Diagrama de flujo de la navegación en el recinto.

El algoritmo analiza las coordenadas actuales del robot y las de destino, y junto con el ángulo actual calcula el cuadrante en el que se encuentra la posición final. A partir de esto realiza los movimientos necesarios para cumplir con la primera coordenada actualizando la posición en cada momento.

Una vez terminado el primer tramo de la trayectoria evalúa si llego a la posición final y en caso contrario vuelve a ejecutar el método para cumplir con el segundo tramo.

7. Detección y rodeo de objetos

Se hace uso del sensor ultrasónico para detectar un objeto en el camino. Mientras el robot se encuentra ejecutando el procedimiento de navegación compara los valores sucesivos de la distancia a la pared a medida que avanza. Si ésta diferencia resulta en un valor grande, significa que algo se cruzó en el camino.

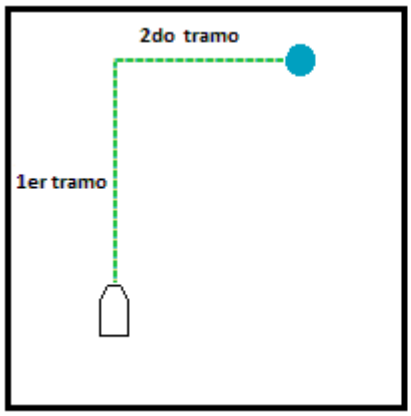
Ante la posible aparición de objetos en su paso, se dotó al robot del código necesario para que pueda sortearlos. De esta manera, ejecuta una serie de maniobras a fin de evitarlo y continuar su camino hacia las coordenadas de destino proporcionadas.

Para que el procedimiento sea eficiente en términos de desplazamiento del robot y del tiempo necesario para realizar las maniobras, se consideran dos situaciones.

- Evitar estando en el primer tramo de la “L”.
- Evitar estando en el segundo tramo o en el primer tramo de la “L” con poca distancia lateral entre hexápodo y destino.

En el primer caso el robot comienza recorriendo el primer tramo. Al detectar un objeto delante suyo rota 90° de manera de colocarse en posición de recorrer antes el segundo tramo y una vez terminado éste, vuelve a rotar

y completa un nuevo primer tramo, siempre teniendo en cuenta la premisa de que realice la menor cantidad de giros posibles.



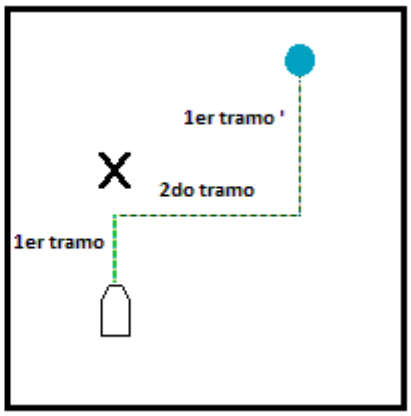


Fig. 21 (a). Recorrido sin obstáculo. **Fig. 21 (b). Recorrido con obstáculo en primer tramo.**

Para el segundo caso si al momento de detectar el objeto la distancia lateral entre el hexápodo y el destino es menor a un valor prefijado (ya sea que esté transitando el primero o el segundo segmento del recorrido), realiza una maniobra para esquivar el objeto buscando su borde, ya sea por izquierda o por derecha.

Adicionalmente se tuvo en cuenta el espacio entre el objeto y la pared que, de ser reducido, podría producir un mal funcionamiento en la detección con el sensor ultrasónico. Por lo tanto, si al detectar el objeto el espacio entre éste y la pared a través del cual debería pasar el hexápodo es menor a un valor predefinido, entonces el robot busca esquivar al objeto por el lado contrario.

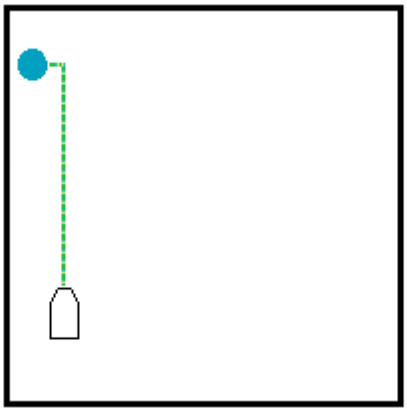


Fig. 22 (a). Recorrido sin obstáculo.

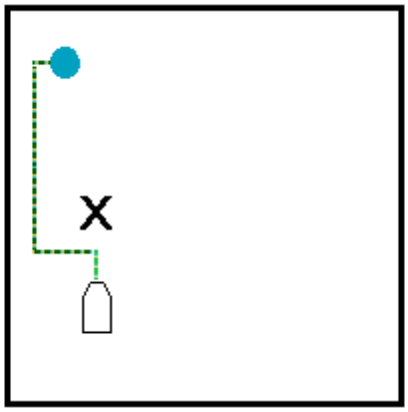


Fig. 22 (b). Recorrido rodeando al obstáculo por la izquierda.

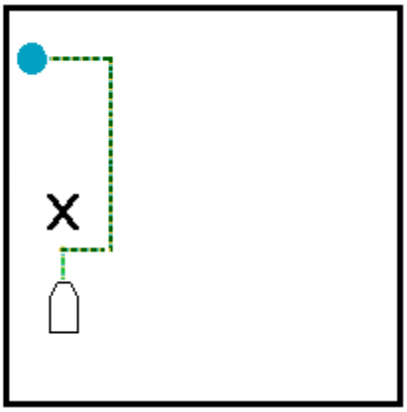


Fig. 22 (c). Recorrido rodeando al obstáculo por la derecha debido al espacio insuficiente por la izquierda.

El método es el siguiente, Fig. 23:

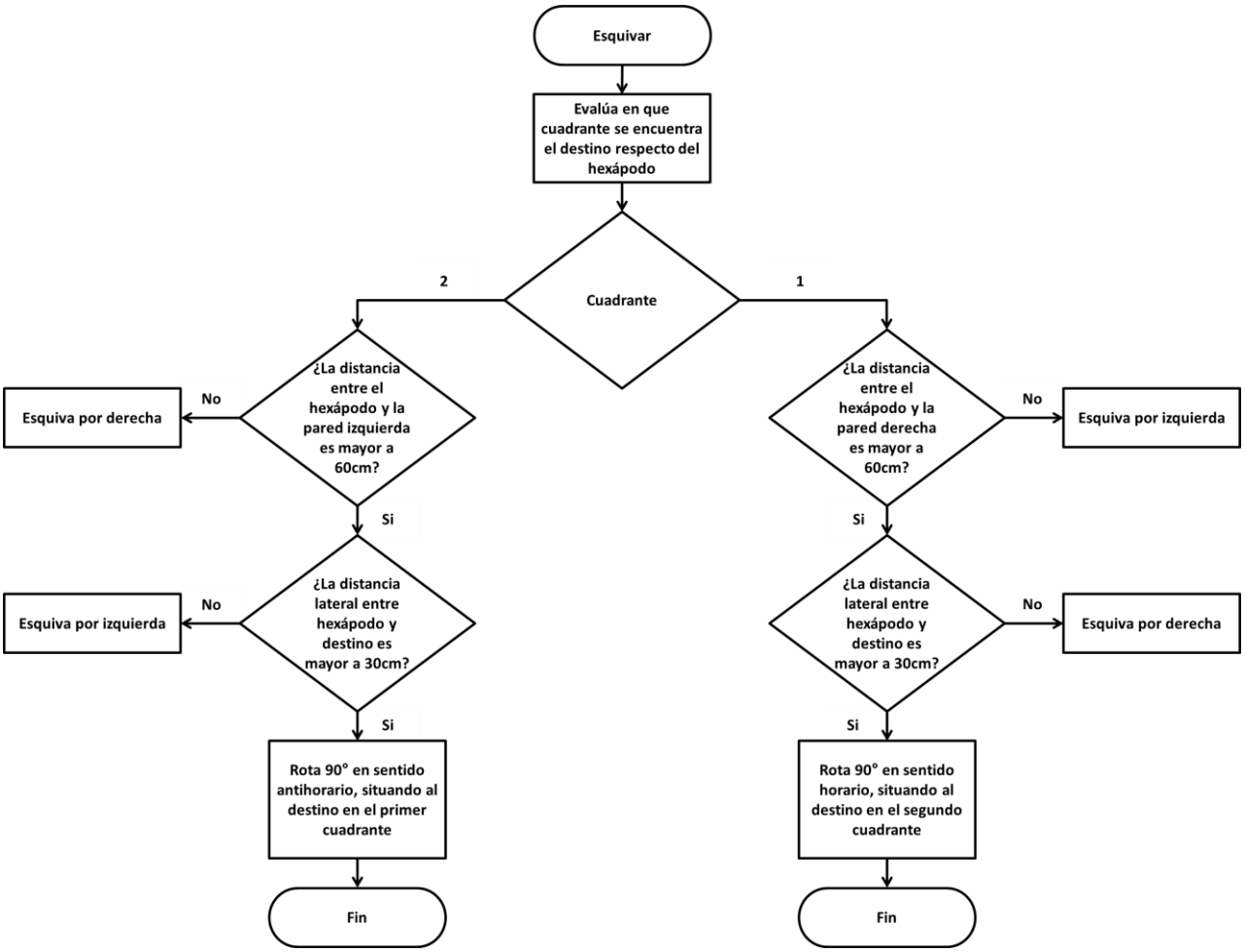


Fig. 23. Diagrama de flujo de esquivar obstáculo en el primer tramo.

Al detectar un objeto primero determina si el destino se encuentra en el primer o segundo cuadrante. Luego analiza tanto la distancia entre pared y objeto como la distancia lateral entre el robot y el destino. De acuerdo al resultado que obtenga pueden ocurrir dos cosas:

- Rotar 90° en un sentido u otro y avanzar hasta completar antes el segundo tramo.
- Recurrir a otro procedimiento el cual busca el borde del objeto (Esquivar por derecha o esquivar por izquierda).

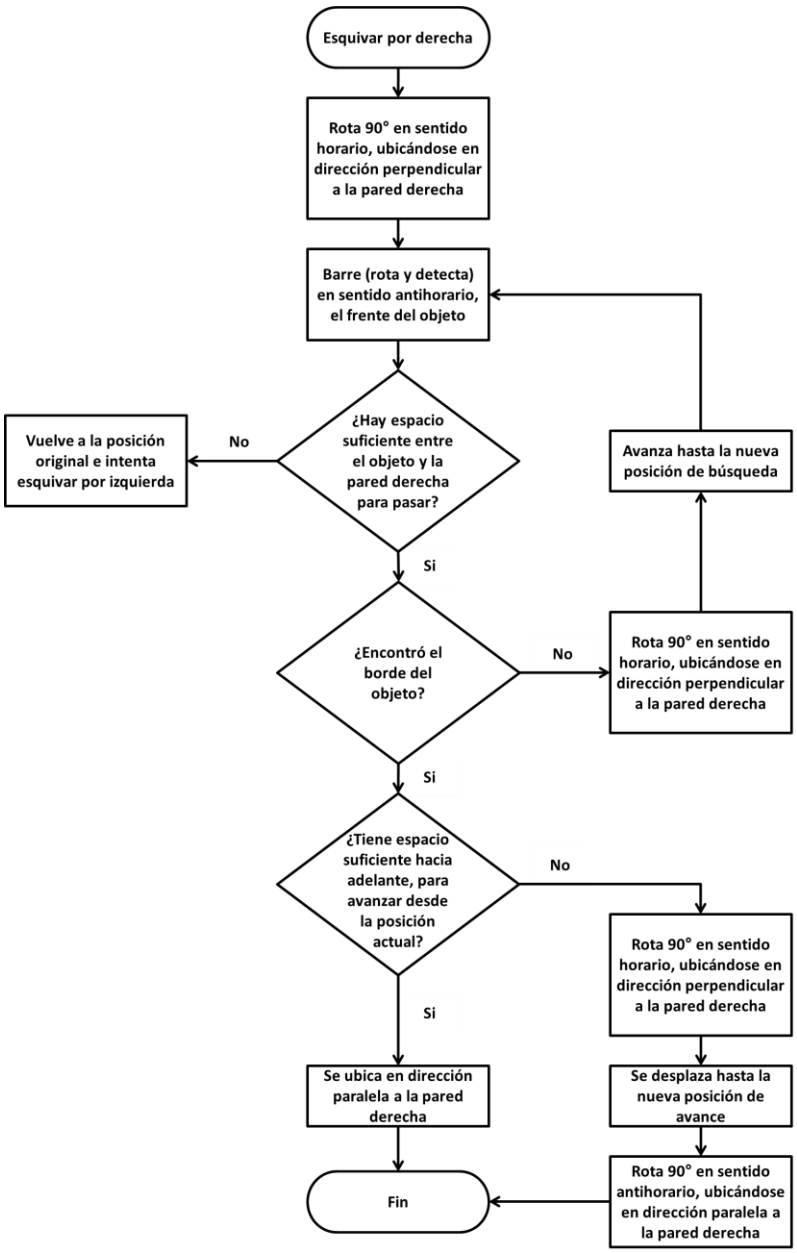


Fig. 24. Diagrama de flujo de esquivar rodeando el obstáculo (Por derecha).

El hexápodo se sitúa perpendicularmente a la pared derecha (punto A) y barre en sentido antihorario el frente del objeto buscando su borde y calculando si el espacio entre el objeto y la pared es suficiente:

- Si el espacio entre objeto y pared es insuficiente el robot vuelve a la posición original e intenta esquivar el objeto por el lado contrario (izquierda).
- Si el espacio es suficiente hay dos alternativas:
  - Si no encuentra el borde del objeto el robot se vuelve a poner perpendicular con la pared derecha, avanza hasta la nueva posición (punto B) y vuelve a barrer el frente del objeto en busca del borde (Fig. 25).
  - Si encuentra el borde evalúa el espacio disponible hacia adelante para avanzar:
    - Si es suficiente se pone perpendicular a la pared frente suyo y avanza (Fig. 26).
    - Si no es suficiente se pone perpendicular a la pared derecha, avanza hasta superar el borde del objeto (punto C), gira 90° en sentido antihorario y avanza (Fig. 27).

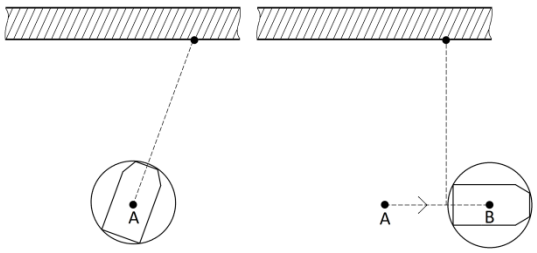


Fig. 25. Hexápodo buscando el borde del objeto.

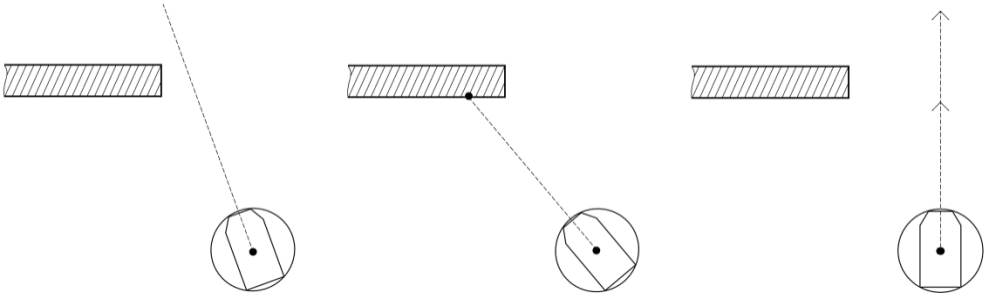


Fig. 26. Hexápodo con espacio suficiente para avanzar.

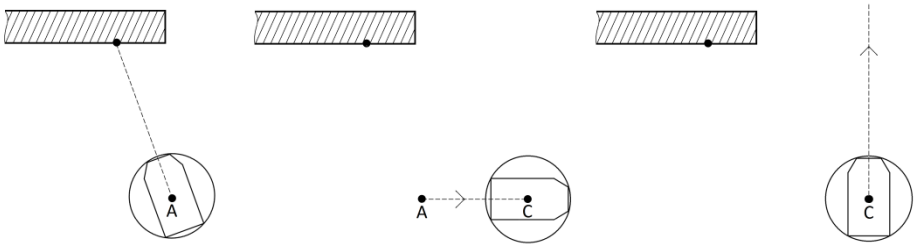


Fig. 27. Hexápodo se desplaza hasta encontrar suficiente espacio para avanzar.

Análogamente, existe una función para rodear el obstáculo por la izquierda.

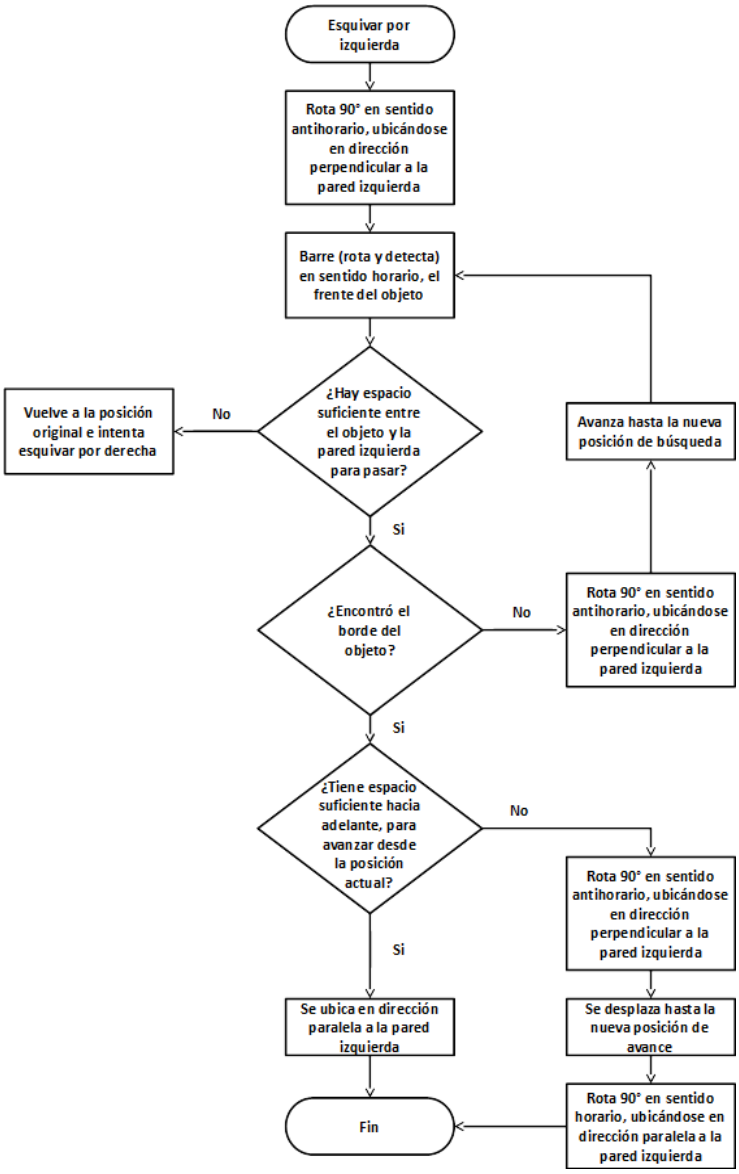


Fig. 28. Diagrama de flujo de esquivar rodeando el obstáculo (Por izquierda).

8. INTERFAZ DE COMANDO Y PRESENTACION

El control y visualización de la posición del robot se hace mediante una interfaz gráfica, desarrollada en Visual Basic 6.0, que presenta en tiempo real toda la información procesada por el robot. El entorno Visual es un lenguaje de programación orientado a objetos, que permite la interacción entre objetos mediante eventos y mantiene la información generada desde el robot actualizada en pantalla.

La mayor diferencia entre este tipo de programación y la estructurada reside en que la primera no tiene un orden predeterminado en el cual se va a ejecutar el algoritmo, sino que responde a las distintas interacciones con y entre los objetos. En la programación estructurada la secuencia de ejecución es siempre la misma.

El objetivo del algoritmo de presentación es que permita el control del robot de una manera muy simple, y que indique gráficamente lo que va sucediendo de la forma más sencilla y clara posible.

8.1. Funcionamiento

La primera parte del programa se encarga de enviar el comando de inicio para que el robot mapee el recinto.

Una vez que el robot comienza con las mediciones las envía al programa el cual las muestra en pantalla.

Cuando el mapeo termina el programa toma todos los datos recibidos, muestra en pantalla los más relevantes, genera una grilla con la misma forma del recinto e indica la posición inicial del hexápodo.

A partir de este punto se agregan las coordenadas de destino, el programa las envía al robot y éste comienza el proceso de navegación. Mientras el robot avanza hacia el destino envía los datos de las mediciones obtenidas al programa y éste va actualizando la posición en la grilla e indicando un objeto en caso de que apareciera uno.

Al llegar a destino el programa genera un cartel que indica que se llegó correctamente.

9. MANUAL DE PROCEDIMIENTO

Fig. 29 (a). Pantalla de presentación.

Fig. 29 (b). Selección de puerto y Angulo inicial.

Al abrirse el programa se ve la pantalla indicada (Fig. 29 (a)). Para comenzar se debe elegir primero el puerto COM correspondiente en el cual esta emparejado el modulo Bluetooth. Luego el programa habilita la opción del ángulo inicial del hexápodo. Con estos dos parámetros seteados se habilita el botón “Comenzar” (Fig. 29 (b)) el cual inicia el proceso de mapeo del recinto.

Fig. 30 (a). Medición de las cuatro paredes.

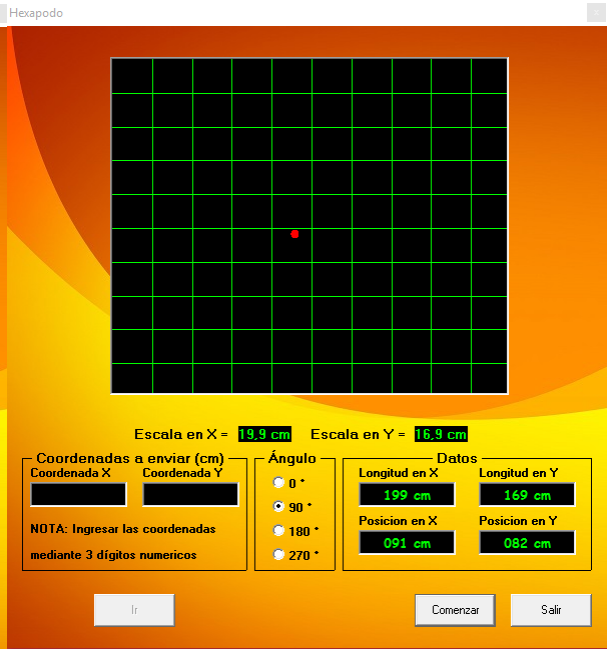


Fig. 30 (b). Datos iniciales y posición del hexápodo.

El hexápodo mide su distancia mínima a cada una de las cuatro paredes y las va indicando en los distintos cuadros de texto, una vez medidas las cuatro paredes se habilita el botón “Cerrar” para volver a la pantalla principal (Fig. 30 (a)).

Al volver a la pantalla principal se genera una grilla del recinto. Cabe aclarar que la forma que tenga la grilla depende de la propia forma del recinto, es decir, si el recinto es visiblemente más largo que ancho, la grilla adoptará esa forma. Dentro de esta grilla, mediante un punto rojo parpadeante, se indica la posición del hexápodo dentro del recinto (Fig. 30 (b)).

Además se observan diversos valores numéricos como el largo y ancho del recinto, posición X e Y del hexápodo, y escala de la grilla.

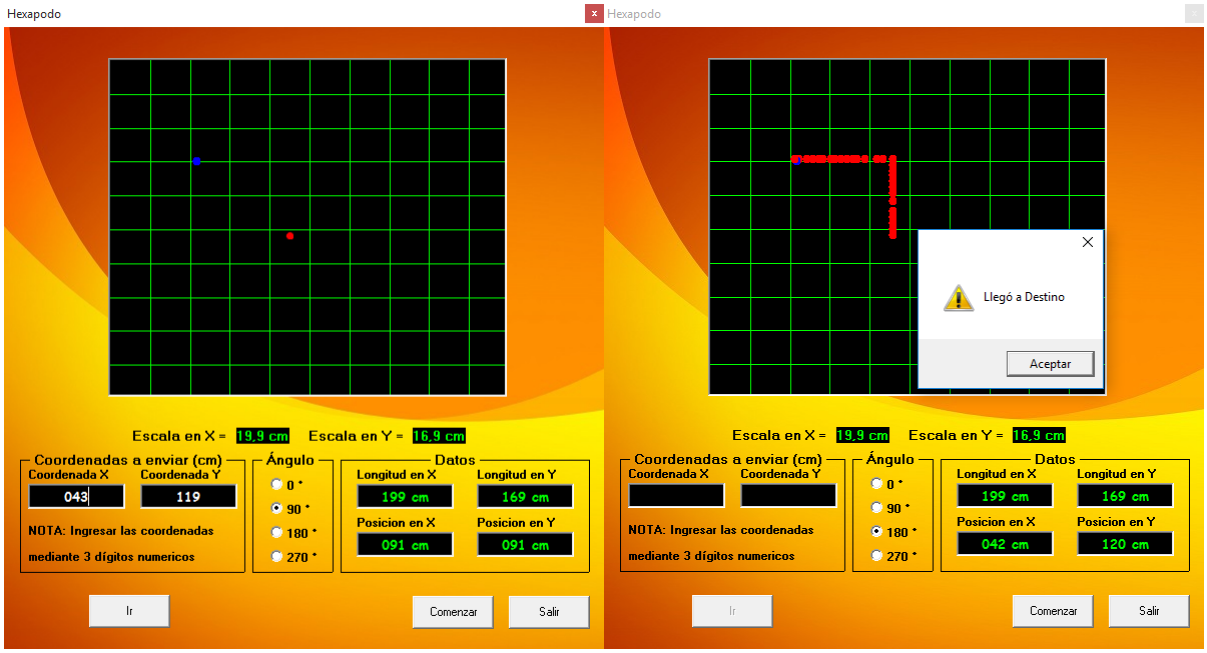


Fig. 31 (a). Ingreso de las coordenadas.

Fig. 31 (b). Llegada del hexápodo a destino.

El programa calcula las coordenadas mínimas y máximas para X e Y de manera tal que el hexápodo no tenga problemas de medición cuando se encuentra muy cerca de alguna de las paredes. En el caso que se ingrese un valor fuera de los límites calculados, el programa cambia este valor por el valor límite.

Las coordenadas se ingresan de a 3 dígitos. Primero la de X y luego la de Y. Una vez que las dos coordenadas son correctas se habilita el botón “Ir” (Fig. 31 (a)).

Una vez clickeado el botón “Ir” el hexápodo comienza su recorrido hacia el destino mientras que en la grilla se observa la trayectoria realizada además de indicarse la posición de destino mediante un punto azul. Cuando el robot llega a destino un cartel indica el fin del recorrido (Fig. 31 (b)).

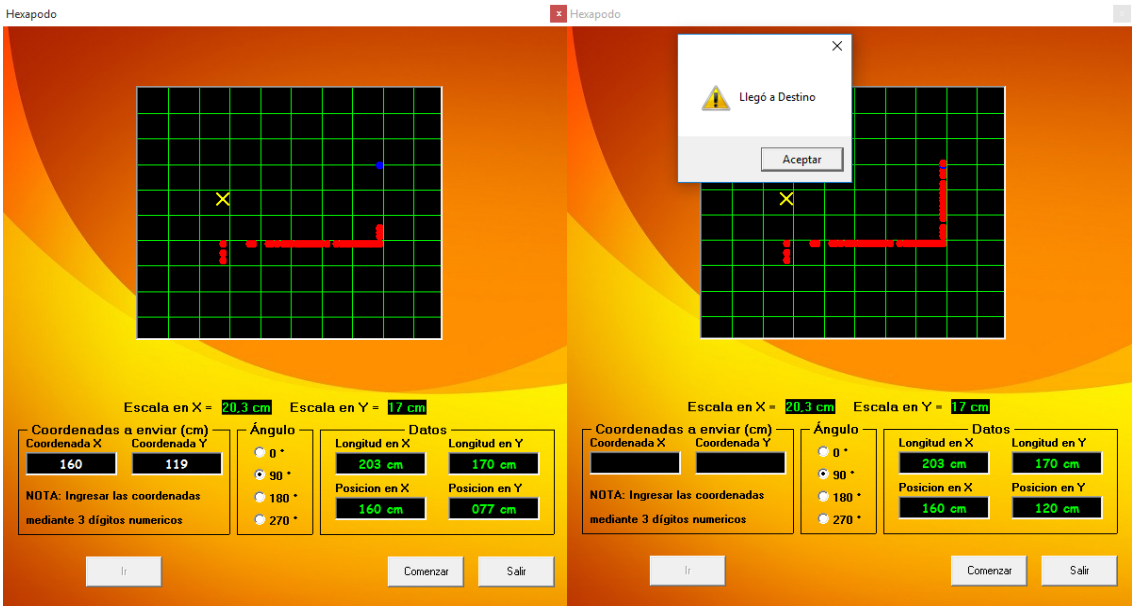


Fig. 32 (a). Trayecto al esquivar el obstáculo. **Fig. 32 (b). Llegada del hexápodo a destino.**

En caso de que aparezca un objeto se muestra en la grilla mediante una cruz amarilla, y se va indicando el nuevo camino para llegar a destino que realiza el hexápodo (Fig. 32 (a) y (b)).

10. CONCLUSION

Se cumplieron satisfactoriamente los requerimientos del proyecto, los algoritmos desarrollados permiten al hexápodo ubicarse espacialmente, la autonomía lograda es adecuada para la experimentación con este tipo de robot y la comunicación remota es eficiente y confiable.

La interfaz gráfica, permite monitorear en tiempo real el comportamiento del robot y comandarlo en forma remota inalámbrica. La presentación es intuitiva y la información mostrada posee los detalles más relevantes, tamaño del recinto, posición del hexápodo y presencia de objetos, que hacen simple la interacción con el robot.

En base a la experiencia obtenida en el desarrollo de este Proyecto, se pueden proponer otras tecnologías para mejorar su desempeño, tales como: baterías recargables de polímero de Litio (PoLi, Li-Pol) de mayor capacidad y menor peso para mejorar la autonomía, comunicación ZigBee o WiFi para aumentar el alcance y sensores de ultrasonido de mayor resolución para mejorar la medición de distancia. Respecto de la interfaz gráfica, podría incorporarse cursores para disponer de un modo de navegación manual, indicadores del nivel de batería, tiempo o distancia de autonomía, registro de hora de eventos tales como aparición de objetos y duración del viaje.

11. BIBLIOGRAFIA

Wikipedia, the free enciclopedia. “Miniature snap-action switch”.
https://en.wikipedia.org/wiki/Miniature_snap-action_switch.

Karym Melo. “Robot zoomórfico”.
http://analisisdelarobotica.blogspot.com.ar/

Seattle Robotics Society. “What’s a Servo?”.
http://www.seattlerobotics.org/guide/servos.html

Rockwell Automation. “Ultrasonic Sensing”.
http://www.ab.com/en/epub/catalogs/12772/6543185/12041221/12041229/print.html

Wikipedia, the free enciclopedia.“Visual Basic”.
https://es.wikipedia.org/wiki/Visual_Basic

Jimb0. “Bluetooth Basics”.
https://learn.sparkfun.com/tutorials/bluetooth-basics

12. ANEXO

12.1 Algoritmo Arduino

```
// Librerías
// Librería para el manejo de servos
#include <Servo.h>
// Librería para realizar comunicación serie mediante software, a través de cualquier par de pines digitales
#include <SoftwareSerial.h>
// Servos (creación de los objetos de tipo servo)
Servo servo1;   Servo servo2;   Servo servo3;
const unsigned int s1min=675, s1max=2275, s2min=600, s2max=2400, s3min=750, s3max=2350;
// Puerto (creación del puerto, para realizar comunicación serie (por bluetooth)
SoftwareSerial BT(7,8);            // (Rx,Tx)
// Variables globales
// Recepción y decodificación de comunicación serie
String inputString="";
// Tamaño del recinto, posición, destino
int distanciax=180, distanciay=180, posx=135, posy=20, irx=135, iry=20;int angulo=0;
// Función: detectar ()
float dm;
const int trigPin=12, echoPin=13;
// Funciones: avanzary_directo(), avanzary_indirecto(), avanzarx_directo() y avanzarx_indirecto
float dm_anterior=0;
// Función: recinto()
unsigned int angpared;
// Función coordenadas()
unsigned int cuadrante;
boolean destino=false;
// Funciones: movimiento(), avanzar(), rotarh(), etc
unsigned int s1ade=100, s1med=90, s1atr=44, s2izq=84, s2med=100, s2der=115,
s3ade=80, s3med=90, s3atr=136,_s1ade, _s1atr, _s3ade, _s3atr, delayPaso=300, delayS2=100;
// Funciones: medir_paredh(), medir_pareda() y perpendicular()
float dm1, dm2, dm3;
// Funciones: buscarBorde(), esquivarDer(), esquivarIzq() y esquivarObjeto()
int dAP, dOP, dVP, dAO, ventana=30, anguloPaso=18, int esquivar=0;
// Función: interrupcionBumpers()
int bumperD = 2;            // D2(5)
int bumperI = 3;            // D3(6)
int led = 13;
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

void setup() {
// Puerto
BT.begin(9600);
// Servos
servo1.attach(9,s1min,s1max); servo2.attach(10,s2min,s2max); servo3.attach(11,s3min,s3max);
servo1.write(s1med); servo2.write(s2med); servo3.write(s3med); delay(1000);
servo1.detach(); servo2.detach(); servo3.detach();
```

```
// Bumpers
pinMode(bumperI, INPUT);
pinMode(bumperD, INPUT);
// Ultrasonido
pinMode(trigPin, OUTPUT);
pinMode(echoPin, INPUT);
digitalWrite(trigPin,LOW);
}
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
void detectar(int N=1)
{
int duration=0, duration_correcto=0,i=0;
// Período entre pulsos; mínimo 60ms según datasheet
delay(60);
digitalWrite(trigPin,HIGH);
// Ancho de pulso; mínimo 10us según datasheet
delayMicroseconds(10);
digitalWrite(trigPin,LOW);
duration=pulseIn(echoPin,HIGH);                .
++i;
   duration_correcto=duration;
dm=duration_correcto;
while (i<N)
{
delay(60);
digitalWrite(trigPin,HIGH);
delayMicroseconds(10);
digitalWrite(trigPin,LOW);
duration=pulseIn(echoPin,HIGH);
if (abs(duration_correcto-duration)<(56*2)) {dm=dm+duration; ++i;}
else
      {
if (duration>duration_correcto) {duration_correcto=duration; dm=duration; i=1; }
      }
    dm=dm/N/56.0;
dm=dm+11.5;
BT.print("PR"); BT.println(dm);
}
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
void movimiento(int i)
{
switch(i)
   {
case 1: servo1.write(_s1atr); servo3.write(_s3ade); break;
case 2: servo1.write(_s1ade); servo3.write(_s3atr); break;
case 3: servo1.write(_s1atr); servo3.write(_s3atr); break;
case 4: servo1.write(_s1ade); servo3.write(_s3ade); break;
   }
delay(delayPaso);
```

```
}
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
void avanzar(int pasos=1, int a_s1ade=s1ade, int a_s1atr=s1atr, int a_s3ade=s3ade, int a_s3atr=s3atr)
{
servo1.attach(9,s1min,s1max); servo2.attach(10,s2min,s2max); servo3.attach(11,s3min,s3max);
   _s1ade=a_s1ade; _s1atr=a_s1atr;
   _s3ade=a_s3ade; _s3atr=a_s3atr;

for (int i=0;i<pasos;++i)
{
for (int i=0;i<2;++i) // Da los pasos de a pares
      {
// La pata 1 está adelante
if (servo1.read()==_s1ade)
// Da el paso con pata 1
         {servo2.write(s2der); delay(delayS2); movimiento(1);}
// La pata 3 está adelante
else if (servo3.read()==_s3ade)
// Da el paso con pata 3
{servo2.write(s2izq); delay(delayS2); movimiento(2);}
// Ninguna de las patas está en posición de dar el paso
else
// Pone pata 1 adelante
         {servo2.write(s2izq); delay(delayS2); _s3atr=servo3.read(); movimiento(2);
// Da el paso con pata 1
servo2.write(s2der); delay(delayS2); _s3atr=a_s3atr; movimiento(1); }
        // Acerca el servo 2 a la posición intermedia
If(servo1.read()==a_s1atr) {servo2.write(s2med+5);}
else if (servo3.read()==a_s3atr) {servo2.write(s2med-5);}
delay(delayS2);
      }
      if((digitalRead(bumperI)==LOW)||(digitalRead(bumperD)==LOW))    {interrupcionBumpers();}
}
servo1.detach(); servo2.detach(); servo3.detach();
if         (angulo==0)     {posx=posx+3;}
else if (angulo==90)    {posy=posy+3;}
else if (angulo==180) {posx=posx-3;}
else if (angulo==270) {posy=posy-3;}
}
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
void retroceder(int pasos=1, int a_s1ade=s1ade, int a_s1atr=s1atr, int a_s3ade=s3ade, int a_s3atr=s3atr)
{
servo1.attach(9,s1min,s1max); servo2.attach(10,s2min,s2max); servo3.attach(11,s3min,s3max);
   _s1ade=a_s1ade; _s1atr=a_s1atr;
   _s3ade=a_s3ade; _s3atr=a_s3atr;
for (int i=0;i<pasos;++i)
{
for (int i=0;i<2;++i) // Da los pasos de a pares
      {
```

```
// La pata 3 está atrás
if (servo3.read()==_s3atr)
// Da el paso con pata 3
        {servo2.write(s2izq); delay(delayS2); movimiento(1);}
// La pata 1 está atrás
else if (servo1.read()==_s1atr)
// Da el paso con pata 1
        {servo2.write(s2der); delay(delayS2); movimiento(2);}
// Ninguna de las patas está en posición de dar el paso
else
// Pone pata 1 atrás
        {servo2.write(s2izq); delay(delayS2); _s3ade=servo3.read(); movimiento(1);
// Da el paso con pata 1
servo2.write(s2der); delay(delayS2); _s3ade=a_s3ade; movimiento(2); }
// Acerca el servo 2 a la posición intermedia
if(servo1.read()==a_s1ade) {servo2.write(s2med+5);}
else if (servo3.read()==a_s3ade) {servo2.write(s2med-5);}
delay(delayS2);
      }
   }
servo1.detach(); servo2.detach(); servo3.detach();
if(angulo==0){posx=posx-3; BT.print("PX"); BT.println(posx);}
else if (angulo==90) {posy=posy-3; BT.print("PY"); BT.println(posy);}
else if (angulo==180) {posx=posx+3; BT.print("PX"); BT.println(posx);}
else if (angulo==270) {posy=posy+3; BT.print("PY"); BT.println(posy);}
}
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
void rotarh(int pasos=1, int a_s1ade=s1ade, int a_s1atr=s1atr, int a_s3ade=s3ade, int a_s3atr=s3atr)
{
servo1.attach(9,s1min,s1max); servo2.attach(10,s2min,s2max); servo3.attach(11,s3min,s3max);
   _s1ade=a_s1ade; _s1atr=a_s1atr;
   _s3ade=a_s3ade; _s3atr=a_s3atr;
for (int i=0;i<pasos;++i)
{
for (int i=0;i<2;++i) // Da los pasos de a pares
      {
// La pata 1 está adelante
if (servo1.read()==_s1ade)
// Da el paso con pata 1
        {servo2.write(s2der); delay(delayS2); movimiento(3);}
// La pata 3 está atrás
else if (servo3.read()==_s3atr)
// Da el paso con pata 3
        {servo2.write(s2izq); delay(delayS2); movimiento(4);}
// Ninguna de las patas está en posición de dar el paso
else

// Pone pata 1 adelante
        {servo2.write(s2izq); delay(delayS2); _s3ade=servo3.read(); movimiento(4);
```

```
// Da el paso con pata 1
servo2.write(s2der); delay(delayS2); _s3ade=a_s3ade; movimiento(3); }
// Acerca el servo 2 a la Posición intermedia
if (servo1.read()==a_s1atr) {servo2.write(s2med+5);}
else if (servo3.read()==a_s3ade) {servo2.write(s2med-5);}
delay(delayS2);
      }
angulo=angulo-anguloPaso;
if (angulo<0) {angulo=angulo+360;}
if((digitalRead(bumperI)==LOW)||(digitalRead(bumperD)==LOW))   {interrupcionBumpers();}
   }
servo1.detach(); servo2.detach(); servo3.detach();
}
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
void rotara(int pasos=1, int a_s1ade=s1ade, int a_s1atr=s1atr, int a_s3ade=s3ade, int a_s3atr=s3atr)
{
servo1.attach(9,s1min,s1max); servo2.attach(10,s2min,s2max); servo3.attach(11,s3min,s3max);
   _s1ade=a_s1ade; _s1atr=a_s1atr;
   _s3ade=a_s3ade; _s3atr=a_s3atr;
for (int i=0;i<pasos;++i)
{
for (int i=0;i<2;++i) // Da los pasos de a pares
      {
// La pata 3 está adelante
if (servo3.read()==_s3ade)
// Da el paso con pata 3
         {servo2.write(s2izq); delay(delayS2); movimiento(3);}
// La pata 1 está atrás
else if (servo1.read()==_s1atr)
// Da el paso con pata 1
         {servo2.write(s2der); delay(delayS2); movimiento(4);}
// Ninguna de las patas está en posición de dar el paso
else
// Pone pata 3 adelante
         {servo2.write(s2der); delay(delayS2); _s1ade=servo1.read(); movimiento(4);
// Da el paso con pata 3
servo2.write(s2izq); delay(delayS2); _s1ade=a_s1ade; movimiento(3);    }
// Acerca el servo 2 a la posición intermedia
if    (servo1.read()==a_s1ade) {servo2.write(s2med+5);}
else if (servo3.read()==a_s3atr) {servo2.write(s2med-5);}
delay(delayS2);
      }
angulo=angulo+anguloPaso;
if (angulo>=360) {angulo=angulo-360;}
if((digitalRead(bumperI)==LOW)||(digitalRead(bumperD)==LOW))   {interrupcionBumpers();}
   }

servo1.detach(); servo2.detach(); servo3.detach();
}
```

```
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
void avanpos(int pasos=1)
{
for (int i=0;i<pasos;++i)
  {
avanzar();
detectar();
if       (angulo==0)    {posx=distanciax-dm; BT.print("PX");BT.println(posx);}
else if (angulo==90)   {posy=distanciay-dm; BT.print("PY");BT.println(posy);}
else if (angulo==180) {posx=dm; BT.print("PX");BT.println(posx);}
else if (angulo==270) {posy=dm; BT.print("PY");BT.println(posy);}
  }
}
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
void interrupcionBumpers()
{
int opcion=0;
digitalWrite(led, HIGH);
esquivar=1;
BT.print(angulo); BT.println(" ]");
if (angulo==0)
    {BT.print("OY"); BT.println(posy + 3*(-(digitalRead(bumperD)==LOW) +
(digitalRead(bumperI)==LOW)) );}
else if (angulo==90)
  {BT.print("OX"); BT.println(posx + 3*((digitalRead(bumperD)==LOW) –
                                        (digitalRead(bumperI)==LOW)) );}
else if (angulo==180)
 {BT.print("OY"); BT.println(posy + 3*((digitalRead(bumperD)==LOW) –
                                        (digitalRead(bumperI)==LOW)) );}
else if (angulo==270)
 {BT.print("OX"); BT.println (posx + 3*((digitalRead(bumperD)==LOW) +
                                        (digitalRead(bumperI)==LOW)) );}
if (digitalRead(bumperD)==LOW)
  {opcion = 1*(((angulo==0)&&((distanciay-posy)>=30)&&((iry-posy)<=30)) ||
               ((angulo==90)&&(posx>=30)&&((posx-irx)<=30)) ||
               ((angulo==180)&&(posy>=30)&&((posy-iry)<=30)) ||
               ((angulo==270)&&((distanciax-posx)>=30)&&((irx-posx)<=30)) );}
else if (digitalRead(bumperI)==LOW)
  {opcion = 2*(((angulo==0)&&(posy>=30)&&((posy-iry)<=30)) ||
        ((angulo==90)&&((distanciax-posx)>=30)&&
        ((irx-posx)<=30))||((angulo==180)&&
        ((distanciay-posy)>=30)&&((iry-posy)<=30)) ||
        ((angulo==270)&&(posx>=30)&&((posx-irx)<=30)) );}
retroceder(4);
if       (opcion==1) {esquivar=0; medir_pareda(); avanpos(4); medir_paredh();}
else if (opcion==2) {esquivar=0; medir_paredh(); avanpos(4); medir_pareda();}
digitalWrite(led, LOW);
}
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
```

```
void perpendicular(boolean horario)
{
BT.println(dm3);
if (dm3<dm1)
   {
      dm2=dm1-dm3;
while ((dm3<dm1)&&(dm2<10))
{
dm2=dm1-dm3;
// Corrige con paso corto// Detecta dm1 primero y dm3 después
rotarh(1,s1ade,80,s3ade,100);
if (horario==true)
 {detectar(10); dm1=dm; rotarh(2); detectar(10); dm3=dm; horario=false;}
// Detecta dm3 primero y dm1 después
else {detectar(10); dm3=dm; rotara(2); detectar(10); dm1=dm; horario=true;}
}
// Vuelve un paso corto (ya se pasó del mínimo)
if (dm2<(dm3-dm1))   {rotara(1,s1ade,80,s3ade,100);}                                    //
}
else if (dm3>=dm1)
   {
      dm2=dm3-dm1;
while ((dm3>=dm1)&&(dm2<10))
{
        dm2=dm3-dm1;
// Corrige con paso corto
rotara(1,s1ade,80,s3ade,100);
if (horario==true)   {detectar(10); dm1=dm; rotarh(2); detectar(10); dm3=dm; horario=false;}
 // Detecta dm1 primero y dm3 después
else{detectar(10); dm3=dm; rotara(2); detectar(10); dm1=dm; horario=true;}
// Detecta dm3 primero y dm1 después
}
// Vuelve un paso corto (ya se pasó del mínimo)
if (dm2<(dm1-dm3))   {rotarh(1,s1ade,80,s3ade,100);}         }
// Va a la dirección perpendicular
if (horario==true)   {rotarh(1);}
else                 {rotara(1);}
}
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
void medir_paredh()
{
unsigned int angulofinal;
if (angulo==0)   {angulo=360;}
angulofinal = (((angulo-0)>0)&&((angulo-0)<=90))*0 +
                    (((angulo-90)>0)&&((angulo-90)<=90))*90 +
                    (((angulo-180)>0)&&((angulo-180)<=90))*180 +
                    (((angulo-270)>0)&&((angulo-270)<=90))*270;
while (angulo>angulofinal) {rotarh();}
// Se acerca al angulo final sin detectar
```

```
detectar(10); dm1=dm;
// Obtiene las primeras 2 detecciones, antes del mínimo.
rotarh(1,s1ade,70,s3ade,110); detectar(10); dm2=dm;
// Si las primeras 2 detecciones no son decrecientes (se está alejando del mínimo),
   while (dm2>=dm1)   {rotara(); detectar(10); dm1=dm; rotarh(1,s1ade,70,s3ade,110); detectar(10);
// rota en el sentido contrario, y vuelve a detectar.
dm2=dm;}
rotarh(1,s1ade,70,s3ade,110); detectar(10); dm3=dm;
// Sigue detectando (y rotando) hasta pasarse del mínimo (la dirección perpendicular), y guarda las últimas 3
detecciones: dm1, dm2 y dm3.
while ((dm3<dm2)&&(dm3>(dm2-5)))    {dm1=dm2; dm2=dm3; rotarh(1,s1ade,70,s3ade,110);
detectar(10); dm3=dm;}
// La detección central dm2 es el mínimo (la dirección más cercana a la perpendicular).
rotarh(1,s1ade,70,s3ade,110); detectar(10);                                                   //
Rota medio paso grande más, de manera de quedar a un paso grande del mínimo.
if (dm<dm3) {rotara(1,s1ade,70,s3ade,110); detectar(10);}
// Si luego de rotar la detección no fue buena, vuelve a ponerse a medio paso de la dirección central.
   dm3=dm; rotara(2); detectar(10); dm1=dm;
// Detecta el primer par de valores dm3 y dm1 que van a ser comparados entre sí.
// Hasta acá encontró la primera aproximación a la perpendicular.
// A partir de esta dirección, a un paso de distancia para cada lado (a una rotación, en cada sentido)
// obtuvo dm1 y dm3, que son las distancias que se van a comparar para acercar la dirección central a la
perpendicular, de manera iterativa
perpendicular(true);
angulo=angulofinal;
detectar();
}
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
void medir_pareda()
{
unsigned int angulofinal;
angulofinal = (((angulo-360)<0)&&((angulo-360)>=-90))*360 +
              (((angulo-90)<0)&&((angulo-90)>=-90))*90 +
              (((angulo-180)<0)&&((angulo-180)>=-90))*180 +
              (((angulo-270)<0)&&((angulo-270)>=-90))*270;
while (angulo<angulofinal)
 {rotara();}
// Se acerca al angulo final sin detectar
detectar(10); dm1=dm;
// Obtiene las primeras 2 detecciones, antes del mínimo.
rotara(1,s1ade,70,s3ade,110); detectar(10); dm2=dm;
// Si las primeras 2 detecciones no son decrecientes (se está alejando del mínimo),
   while (dm2>=dm1)   {rotarh(); detectar(10); dm1=dm; rotara(1,s1ade,70,s3ade,110); detectar(10);
// rota en el sentido contrario, y vuelve a detectar.
dm2=dm;}
rotara(1,s1ade,70,s3ade,110); detectar(10); dm3=dm;
// Sigue detectando (y rotando) hasta pasarse del mínimo (la dirección perpendicular), y guarda las últimas 3
detecciones: dm1, dm2 y dm3.
while ((dm3<dm2)&&(dm3>(dm2-5)))
```

```
// La detección central dm2 es el mínimo (la dirección más cercana a la perpendicular).
  {dm1=dm2; dm2=dm3; rotara(1,s1ade,70,s3ade,110); detectar(10); dm3=dm;}
rotara(1,s1ade,70,s3ade,110); detectar(10);                                                    //
Rota medio paso grande más, de manera de quedar a un paso grande del mínimo.
if (dm<dm3) {rotarh(1,s1ade,70,s3ade,110); detectar(10);}
// Si luego de rotar la detección no fue buena, vuelve a ponerse a medio paso de la dirección central.
   dm3=dm; rotarh(2); detectar(10); dm1=dm;
// Detecta el primer par de valores dm3 y dm1 que van a ser comparados entre sí.
// Hasta acá encontró la primera aproximación a la perpendicular.
// A partir de esta dirección, a un paso de distancia para cada lado (a una rotación, en cada sentido)
// obtuvo dm1 y dm3, que son las distancias que se van a comparar para acercar la dirección central a la
perpendicular, de manera iterativa
dm2=dm3; dm3=dm1; dm1=dm2;
perpendicular(false);
angulo=angulofinal; if (angulo==360) {angulo=0;}
detectar();
}
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
void buscarBorde(int sentido)
// Barre iterativamente en sentido horario ó antihorario, hasta encontrar el borde del objeto
{
// dWZ: distancia entre "W" y "Z"
//         A: Araña; P: Pared; O: Objeto; V: espacio Vacío
// angulo_relativo: angulo relativo de la araña, respecto a la dirección perpendicular al objeto (es 0° en la
// dirección perpendicular al objeto)
intangulo_relativo, dv, du;
// Calcula la distancia entre Araña y Pared
dAP = (angulo==0)*(distanciax-posx) + (angulo==90)*(distanciay-posy) + (angulo==180)*posx +
(angulo==270)*posy;
// Si la araña está pegada a la pared, comienza a barrer desde la dirección perpendicular al objeto
if (dAP<=30)         {
      angulo_relativo=-anguloPaso;
if         (sentido==1) {rotara(5);}
else if(sentido==2) {rotarh(5);}
   }
else
{
angulo_relativo=-3*anguloPaso;
if         (sentido==1) {rotara(3);}
else if (sentido==2) {rotarh(3);}
   }
dOP=0; dVP=0;
// Barre hasta detectar un punto perteneciente al objeto
```

```
while (dOP==0)
  {
if        (sentido==1)
// Rota
{rotara();}
else if (sentido==2)
{rotarh();}
// Detecta
detectar();
// Actualiza ángulo relativo
     angulo_relativo=angulo_relativo+anguloPaso;
// Calcula la distancia perpendicular entre araña y punto sensado
dv=dm*cos(angulo_relativo*3.141/180);
// Calcula la distancia paralela entre araña y punto sensado (da negativa cuando el punto está a la izquierda)
du=dm*sin(-angulo_relativo*3.141/180);
// El punto sensado pertenece al Objeto (cuando encuentra un punto quepertenece al objeto corta la
    // iteración)
if (dv<=dAO+3)
     {
dAO=dv;
// Distancia entre Objeto (punto) y Pared
dOP=dAP-du;
}
// El punto sensado corresponde a espacio Vacío
else
     {
dVP=dAP-dAO*tan(-angulo_relativo*3.141/180);
if (angulo_relativo==2*anguloPaso) {dOP=1;}
     }
  }
}
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
// Esquiva el objeto por la derecha (barre, busca el borde del objeto y se desplaza haciala derecha)
void esquivarDer()
{
int dAPopuesta;
boolean iterar=true;
// Obtiene la distancia a la pared opuesta, en caso de que tenga que intentar esquivar por el otro lado
if(angulo==0)    {dAPopuesta=posx;}
else if (angulo==90)   {dAPopuesta=posy;}
else if (angulo==180) {dAPopuesta=distanciax-posx;}
else if (angulo==270) {dAPopuesta=distanciay-posy;}
while (iterar==true)
  {
buscarBorde(1);
// No hay espacio para pasar entre el objeto y la pared. Vuelve a la posición inicial e intenta esquivar por el
     // otro lado
if ((dOP<=ventana)&&(dVP<=ventana))
     {
```

```
// Se pone perpendicular a la pared
rotara(); medir_pareda();
// Avanza hasta la posición de la que partió inicialmente
while (dm>dAPopuesta) {avanpos();}
// Se pone perpendicular a la pared (aunque ya esté, corrige)
rotara();medir_paredh();
// esquivar=1: inicia el algoritmo para esquivar por izquierda
iterar=false;esquivar=1;
  }
// FIN: Se coloca en la dirección de avance, porque ye tiene espacio para pasar.
else if ((dVP-dAP)>=(ventana/2))
{
// Se coloca en la dirección de avance
      medir_paredh();
// Esquivar=0: FIN
iterar=false; esquivar=0;
    }
// FIN: Hay espacio para pasar entre el último punto vacío detectado y la pared.
else if (dVP>ventana)                {
// Se pone perpendicular a la pared
rotarh(); medir_paredh();
// Avanza hasta el nuevo punto
while ((dm>(dVP-ventana/2))&&(dm>=20)) {avanpos();}
// Se coloca en la dirección de avance
      medir_pareda();
// Esquivar=0: FIN
iterar=false; esquivar=0;
    }
// Sigue barriendo
else if (dOP>ventana)
    {
// Se pone perpendicular a la pared
rotarh(); medir_paredh();
// Avanza hasta el nuevo punto de barrido
while ((dm>(dOP-ventana/2))&&(dm>=20)) {avanpos();}
// Se pone perpendicular a la pared (aunque ya esté, corrige)
rotarh(); medir_pareda();
// Vuelve a barrer
iterar=true;
    }
}
}
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
// Esquiva el objeto por la izquierda (barre, busca el borde del objeto y se desplaza hacia la izquierda)
void esquivarIzq()
{
int dAPopuesta;
boolean iterar=true;
// Obtiene la distancia a la pared opuesta, en caso de que tenga que intentar esquivar por el otro lado
```

```
if(angulo==0)   {dAPopuesta=posx;}
else if (angulo==90)   {dAPopuesta=posy;}
else if (angulo==180) {dAPopuesta=distanciax-posx;}
else if (angulo==270) {dAPopuesta=distanciay-posy;}
while (iterar==true)
  {
buscarBorde(2);
// No hay espacio para pasar entre el objeto y la pared. Vuelve a la posición inicial e intenta esquivar por el otro lado
if ((dOP<=ventana)&&(dVP<=ventana))
     {
// Se pone perpendicular a la pared
rotarh(); medir_paredh();
// Avanza hasta la posición de la que partió inicialmente
while (dm>dAPopuesta) {avanpos();}
// Se pone perpendicular a la pared (aunque ya esté, corrige)
rotarh(); medir_pareda();
// esquivar=2: inicia el algoritmo para esquivar por derecha
iterar=false; esquivar=2;
     }
// FIN: Se coloca en la dirección de avance, porque ye tiene espacio para pasar.
else if ((dVP-dAP)>=(ventana/2))
{
// Se coloca en la dirección de avance
        medir_pareda();
// esquivar=0: FIN
iterar=false; esquivar=0;
     }
// FIN: Hay espacio para pasar entre el último puntovacío detectado y la pared.
else if (dVP>ventana)
     {
// Se pone perpendicular a la pared
rotara(); medir_pareda();
// Avanza hasta el nuevo punto
while ((dm>(dVP-ventana/2))&&(dm>=20)) {avanpos();}
// Se coloca en la dirección de avance
        medir_paredh();
// Esquivar=0: FIN
iterar=false; esquivar=0;
     }
// Sigue barriendo
else if (dOP>ventana)
     {
// Se pone perpendicular a la pared
rotara(); medir_pareda();
// Avanza hasta el nuevo punto de barrido
while ((dm>(dOP-ventana/2))&&(dm>=20)) {avanpos();}
// Se pone perpendicular a la pared (aunque ya esté, corrige)
rotara(); medir_paredh();
// Vuelve a barrer
```

```
iterar=true;
      }
}
}
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
void esquivarObjeto()
{
esquivar=0;
detectar(); dAO=dm;
switch(angulo)
  {
case 0:
dm = (iry>=posy)*(distanciay-posy) + (posy>iry)*(posy);
     if      ((iry>=posy)&&(dm>60))   { medir_pareda(); if ((iry-posy)<30) {esquivarIzq();} }
else if ((iry>=posy)&&(dm<=60)){ medir_paredh(); esquivarDer(); }
else if ((posy>iry)&&(dm>60))     { medir_paredh(); if ((posy-iry)<30) {esquivarDer();} }
else if ((posy>iry)&&(dm<=60))   { medir_pareda(); esquivarIzq(); }
break;
case 90:
dm = (irx>=posx)*(distanciax-posx) + (posx>irx)*(posx);
     if ((irx>=posx)&&(dm>60))   { medir_paredh(); if ((irx-posx)<30) {esquivarDer();} }
else if ((irx>=posx)&&(dm<=60)) { medir_pareda(); esquivarIzq(); }
else if ((posx>irx)&&(dm>60))     { medir_pareda(); if ((posx-irx)<30) {esquivarIzq();} }
else if ((posx>irx)&&(dm<=60))   { medir_paredh(); esquivarDer(); }
break;
case 180:
dm = (iry>=posy)*(distanciay-posy) + (posy>iry)*(posy);
     if ((iry>=posy)&&(dm>60))   { medir_paredh(); if ((iry-posy)<30) {esquivarDer();} }
else if ((iry>=posy)&&(dm<=60)) { medir_pareda(); esquivarIzq(); }
else if ((posy>iry)&&(dm>60))     { medir_pareda(); if ((posy-iry)<30) {esquivarIzq();} }
else if ((posy>iry)&&(dm<=60))   { medir_paredh(); esquivarDer(); }
break;
case 270:
dm = (irx>=posx)*(distanciax-posx) + (posx>irx)*(posx);
     if ((irx>=posx)&&(dm>60))   { medir_pareda(); if ((irx-posx)<30) {esquivarIzq();} }
else if ((irx>=posx)&&(dm<=60)) { medir_paredh(); esquivarDer(); }
else if ((posx>irx)&&(dm>60))     { medir_paredh(); if ((posx-irx)<30) {esquivarDer();} }
else if ((posx>irx)&&(dm<=60))   { medir_pareda(); esquivarIzq(); }
break;
  }
if       (esquivar==1)    {esquivarIzq();}
else if (esquivar==2)    {esquivarDer();}
}
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
void avanzary_indirecto(){
detectar(5);
dm_anterior = dm;
posy = distanciay - dm;
// Esta funcion permite que el hexapodo avance siempre y cuando ocurran dos situaciones, que todavia no haya
```

```
//llegado a su destino, y que la distancia sea menor a 10 cm en sucesivas mediciones, caso contrario supone
// que hay un objeto adelante y comienza el procedimiento para esquivarlo. El "indirecto" hace referencia a que
// el calculo para saber la distancia que le falta recorrer no es directamente el valor que mide a la pared.
while (((distanciay-dm)<iry)&&(esquivar==0)) {
avanzar();
detectar();
if ((dm_anterior-dm)<10) {
posy = distanciay - dm;
dm_anterior = dm;
  }
else
  {
detectar(5);
if ((dm_anterior-dm)<10) {posy = distanciay - dm; dm_anterior = dm;}
else {posy=posy+3; esquivar=1; BT.print("OY"); BT.println(posy+dm); delay(100);}
}
BT.print("PY");BT.println(posy);
}
}
/////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
void avanzary_directo(){
detectar(5);
dm_anterior = dm;
posy = dm;
// Esta funcion permite que el hexapodo avance siempre y cuando ocurran dos situaciones, que todavia no haya
//llegado a su destino, y que la distancia sea menor a 10 cm en sucesivas mediciones, caso contrario supone
// que hay un objeto adelante y comienza el procedimiento para esquivarlo. El "indirecto" hace referencia a que
// el calculo para saber la distancia que le falta recorrer es directamente el valor que mide a la pared.
while ((dm>iry)&&(esquivar==0)) {
avanzar();
detectar();
if ((dm_anterior-dm)<10) {
posy = dm;
    dm_anterior = dm;
  }
else
  {
detectar(5);
if ((dm_anterior-dm)<10) {posy = dm; dm_anterior = dm;}
else {posy=posy-3; esquivar=1; BT.print("OY"); BT.println(posy-dm); delay(100);}
}
BT.print("PY");BT.println(posy);
}
}
```

```
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
void avanzarx_indirecto(){
detectar(5);
dm_anterior = dm;
posx = distanciax - dm;
// Esta funcion permite que el hexapodo avance siempre y cuando ocurran dos situaciones, que todavia no haya
//llegado a su destino, y que la distancia sea menor a 10 cm en sucesivas mediciones, caso contrario supone
// que hay un objeto adelante y comienza el procedimiento para esquivarlo. El "indirecto" hace referencia a que
// el calculo para saber la distancia que le falta recorrer no es directamente el valor que mide a la pared.
while (((distanciax-dm)<irx)&&(esquivar==0)) {
avanzar();
detectar();
if ((dm_anterior-dm)<10) {
posx = distanciax - dm;
      dm_anterior = dm;
   }
else
   {
detectar(5);
if ((dm_anterior-dm)<10) {posx = distanciax - dm; dm_anterior = dm;}
else {posx=posx+3; esquivar=1; BT.print("OX"); BT.println(posx+dm); delay(100);}
}
BT.print("PX");BT.println(posx);
}
}
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
void avanzarx_directo(){
detectar(5);
dm_anterior = dm;
posx = dm;
// Esta funcion permite que el hexapodo avance siempre y cuando ocurran dos situaciones, que todavia no haya
//llegado a su destino, y que la distancia sea menor a 10 cm en sucesivas mediciones, caso contrario supone que
// hay un objeto adelante y comienza el procedimiento para esquivarlo. El "indirecto" hace referencia a que el
// calculo para saber la distancia que le falta recorrer es directamente el valor que mide a la pared.
while ((dm>irx)&&(esquivar==0)) {
avanzar();
detectar();
if ((dm_anterior-dm)<10) {
posx = dm;
      dm_anterior = dm;
   }
else
   {
detectar(5);
if ((dm_anterior-dm)<10) {posx = dm; dm_anterior = dm;}
else {posx=posx-3; esquivar=1; BT.print("OX"); BT.println(posx-dm); delay(100);}
   }
BT.print("PX");BT.println(posx);
}
```

```
}
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
void recinto() {
distanciax=0; distanciay=0;
for (int k = 1 ; k <= 4 ; k++) {
medir_paredh();
switch(angpared) {
// En este case se va calculando los valores del recinto, en distanciax se guarda el largo total en el eje "x", en
// distanciay.el largo total en el eje "y", y en posx y posy se guardan las posiciones iniciales en ambos ejes del
// hexápodo.
case 0:
distanciax = distanciax + dm + 0.5;
angpared = 270;
break;
case 270:
distanciay = distanciay + dm + 0.5;
posy = dm + 0.5;
angpared = 180;
break;
case 180:
distanciax =   distanciax + dm + 0.5;
posx = dm + 0.5;
angpared = 90;
break;
case 90:
distanciay = distanciay +   dm + 0.5;
angpared = 0;
break;
        }
BT.print("DI"); BT.println(dm);
     }
delay(100);
BT.print("DX"); BT.println(distanciax);
delay(100);
BT.print("DY"); BT.println(distanciay);
delay(100);
BT.print("PX"); BT.println(posx);
delay(100);
BT.print("PY"); BT.println(posy);
delay(100);
BT.print("AN");BT.println(angulo);
irx=posx; iry=posy;

}
```

```
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
void coordenadas(){
////////////////////////// PARA 0° ////////////////////////////////////
switch (angulo) {
case 0:
cuadrante = (irx > posx && iry > posy)*2 + (irx <= posx && iry > posy)*3 +
          (irx <= posx && iry <= posy)*4 + (irx > posx && iry <= posy)*1;
switch (cuadrante) {
case 1: avanzarx_indirecto(); break;
case 2: avanzarx_indirecto(); break;
case 3: medir_pareda(); avanzary_indirecto(); break;
case 4: medir_paredh(); avanzary_directo(); break;
   }
break;
////////////////////////// PARA 90° ////////////////////////////////////
case 90:
cuadrante = (irx > posx && iry > posy)*1 + (irx <= posx && iry > posy)*2 +
                  (irx <= posx && iry <= posy)*3 + (irx > posx && iry <= posy)*4;
switch (cuadrante) {
case 1: avanzary_indirecto(); break;
case 2: avanzary_indirecto(); break;
case 3: medir_pareda(); avanzarx_directo(); break;
case 4: medir_paredh(); avanzarx_indirecto(); break;
    }
break;
////////////////////////// PARA 180° ////////////////////////////////////
case 180:
cuadrante = (irx > posx && iry > posy)*4 + (irx <= posx && iry > posy)*1 +
           (irx<= posx && iry <= posy)*2 + (irx > posx && iry <= posy)*3;
switch (cuadrante) {
case 1: avanzarx_directo(); break;
case 2: avanzarx_directo(); break;
case 3: medir_pareda(); avanzary_directo(); break;
case 4: medir_paredh(); avanzary_indirecto(); break;
   }
break;
////////////////////////// PARA 270° ////////////////////////////////////
case   270:
cuadrante = (irx > posx && iry > posy)*3 + (irx <= posx && iry > posy)*4 +
          (irx<= posx && iry <= posy)*1 + (irx > posx && iry <= posy)*2;
switch (cuadrante) {
case 1: avanzary_directo(); break;
case 2: avanzary_directo(); break;
case 3: medir_pareda(); avanzarx_indirecto(); break;
case 4: medir_paredh(); avanzarx_directo(); break;
   }
break;
  }
if   (( abs(posx - irx) < 4 ) && ( abs(posy - iry) < 4 )) {
```

```
esquivar=0;
BT.print("PX");BT.println( posx );
delay(100);
BT.print("PY");BT.println( posy );
delay(100);
BT.print("AN");BT.println(angulo);
delay(100);
BT.println("OK");
destino=false;
   }
}
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
void RXcomandos()
{
while(BT.available())
{
// Declaramos la variable inChar como tipo caracter, y le asignamos el valorque llega por el puerto serie,
     // aclarandole que lo que llega es un caracter
char inChar = (char)BT.read();
// Concatenamos lo que llega en inChar en la variable inputString
inputString += inChar ;
if (inChar == '\n') {
if (inputString.substring(0,8) == "comenzar") {
angpared = inputString.substring(8,12).toInt();
if (angpared == 0) { angpared = 270; }
else
{ angpared = angpared - 90;}
angulo = inputString.substring(8,12).toInt();
inputString = "";
recinto();
        }
else if (inputString.substring(3,4) == ",") {
// Los valores que ingrese por el puerto los convierto a numéricoy los uso de coordenadas
irx = inputString.substring(0,3).toInt();
iry = inputString.substring(4,7).toInt();
delay(100);
esquivar=0; destino=true;
        }
inputString = "";
}
    }
}
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
void loop() {
if (esquivar>0)            {esquivarObjeto();}
if (destino==true)         {coordenadas();}
if (BT.available()!=0)     {RXcomandos();}
}
```

12.2 Algoritmo Visual

```
Dim cadena, angulo As String
Dim cont, color, ox, oy As Integer
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
Private Sub cmdcomenzar_Click()
'Hace invisible la lista de puertos COM
Combo1.Visible = False
'Hace invisible la etiqueta de "COM"
Label9.Visible = False
'Abre el puerto
Comm.PortOpen = True
For i = 0 To 7
'Hace invisible las etiquetas del nombre del proyecto, alumnos, tutores.
Label3(i).Visible = False
Next
'envio del comando para que el hexapodo inicie
Comm.Output = "comenzar" + angulo + vbLf
'Inhabilita la opcion de modificar los objetos
OB0.Locked = True
OB90.Locked = True
OB180.Locked = True
OB270.Locked = True
'Abre el formulario 2
Form2.Show
Form2.txtpared.Text = cont
End Sub
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
Private Sub CmdIr_Click()
'envio de las coordenadas
Comm.Output = CoordX.Text + "," + CoordY.Text + vbLf
'Se deshabilita el texto de la coordenada X para que no se pueda ingresar valor
CoordX.Locked = True
'Se deshabilita el texto de la coordenada Y para que no se pueda ingresar valor
CoordY.Locked = True
'Se deshabilita el boton Ir
CmdIr.Enabled = False
Picture1.Cls
For i = 1 To 9
```

```
    'Se genera la grilla
    Picture1.Line (0, (Picture1.Height / 10) * i)-(Picture1.Width, (Picture1.Height / 10) * i), vbGreen
    Picture1.Line ((Picture1.Width / 10) * i, 0)-((Picture1.Width / 10 * i), Picture1.Height), vbGreen
Next
'Se deshabilita el timer del punto titilante
Timer1.Enabled = False
'Se habilita el timer de trayecto
Timer2.Enabled = True
End Sub
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
Private Sub Combo1_Click()
'Establece el puerto de comunicaciones
Comm.CommPort = Combo1.ListIndex + 1
' Habilita la opcion de elegir el angulo inicial
OB0.Enabled = True
OB90.Enabled = True
OB180.Enabled = True
OB270.Enabled = True
End Sub
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
Private Sub Comm_OnComm()
If Comm.CommEvent = comEvReceive Then
caracter = Comm.Input
cadena = cadena + caracter
longitud = Len(cadena)
ultimo = Mid(cadena, longitud, 1)
    If ultimo = vbLf Then
    'Toma de la palabra en la variable codigo desde el primer caracter, dos caracteres (1er y 2do en este caso)
codigo = Mid(cadena, 1, 2)
        If codigo = "PR" Then
            'Muestra el valor en un texto
Form2.txtprom.Text = Mid(cadena, 3, 6) + " cm"
        ElseIf codigo = "DI" Then
            Select Case cont
Case 1
                'Muestra el valor en un texto
Form2.txtp1.Text = Mid(cadena, 3, 6) + " cm"
Form2.txtpared.Text = 2
```

```
            Case 2
                'Muestra el valor en un texto
Form2.txtp2.Text = Mid(cadena, 3, 6) + " cm"
Form2.txtpared.Text = 3
            Case 3
                'Muestra el valor en un texto
Form2.txtp3.Text = Mid(cadena, 3, 6) + " cm"
Form2.txtpared.Text = 4
            Case 4
                'Muestra el valor en un texto
Form2.txtp4.Text = Mid(cadena, 3, 6) + " cm"
Form2.txtprom.Text = ""
            End Select
cont = cont + 1
'Guarda el valor de la distancia X, rellena con 0 en caso de ser necesario para cumplir los 3 dígitos
ElseIf codigo = "DX" Then
If Mid(cadena, 3, 3) < 10 Then
                txtdx.Text = "00" + Mid(cadena, 3, 3) + " cm"
            ElseIf Mid(cadena, 3, 3) < 100 Then
                txtdx.Text = "0" + Mid(cadena, 3, 3) + " cm"
            Else
                txtdx.Text = Mid(cadena, 3, 3) + " cm"
End If
        'Guarda el valor de la distancia Y, rellena con 0 en caso de ser necesario para cumplir los 3 digitos
ElseIf codigo = "DY" Then
If Mid(cadena, 3, 3) < 10 Then
                txtdy.Text = "00" + Mid(cadena, 3, 3) + " cm"
            ElseIf Mid(cadena, 3, 3) < 100 Then
                txtdy.Text = "0" + Mid(cadena, 3, 3) + " cm"
            Else
                txtdy.Text = Mid(cadena, 3, 3) + " cm"
End If
        'Guarda el valor de la posicion X, rellena con 0 en caso de ser necesario para cumplir los 3 digitos
ElseIf codigo = "PX" Then
If Mid(cadena, 3, 3) < 10 Then
                txtpx.Text = "00" + Mid(cadena, 3, 3) + " cm"
            ElseIf Mid(cadena, 3, 3) < 100 Then
                txtpx.Text = "0" + Mid(cadena, 3, 3) + " cm"
```

```
        Else
            txtpx.Text = Mid(cadena, 3, 3) + " cm"
End If
        'Guarda el valor de la posicion Y, rellena con 0 en caso de ser necesario para cumplir los 3 digitos
ElseIf codigo = "PY" Then
If Mid(cadena, 3, 3) < 10 Then
            txtpy.Text = "00" + Mid(cadena, 3, 3) + " cm"
        ElseIf Mid(cadena, 3, 3) < 100 Then
            txtpy.Text = "0" + Mid(cadena, 3, 3) + " cm"
        Else
            txtpy.Text = Mid(cadena, 3, 3) + " cm"
End If
        'Genera una cruz en la posicion donde aparece el objeto
ElseIf codigo = "OX" Then
        Picture1.DrawWidth = 2
            Picture1.Line ((CInt(Mid(cadena, 3, 3)) * escala) - 100, (CInt(Mid(txtdy.Text, 1, 3)) -
                    CInt(Mid(txtpy.Text, 1, 3))) * escala + 100)-((CInt(Mid(cadena, 3, 3))
                    *escala) + 100, (CInt(Mid(txtdy.Text, 1, 3)) - CInt(Mid(txtpy.Text, 1, 3)))
                    * escala - 100), vbYellow
        Picture1.Line ((CInt(Mid(cadena, 3, 3)) * escala) - 100, (CInt(Mid(txtdy.Text, 1, 3)) -
                CInt(Mid(txtpy.Text, 1, 3))) * escala - 100)-((CInt(Mid(cadena, 3, 3))
                    *escala) + 100, (CInt(Mid(txtdy.Text, 1, 3)) - CInt(Mid(txtpy.Text, 1, 3)))
                *escala + 100), vbYellow
        'Genera una cruz en la posicion donde aparece el objeto
ElseIf codigo = "OY" Then
        Picture1.DrawWidth = 2
            Picture1.Line ((CInt(Mid(txtpx.Text, 1, 3)) * escala) - 100, (CInt(Mid(txtdy.Text, 1, 3)) –
                    CInt(Mid(cadena, 3, 3))) * escala + 100)-((CInt(Mid(txtpx.Text, 1, 3))
                    *escala) + 100, (CInt(Mid(txtdy.Text, 1, 3)) - CInt(Mid(cadena, 3, 3))) *escala
                    - 100), vbYellow
Picture1.Line ((CInt(Mid(txtpx.Text, 1, 3)) * escala) - 100, (CInt(Mid(txtdy.Text, 1, 3)) -
                CInt(Mid(cadena, 3, 3))) * escala - 100)-((CInt(Mid(txtpx.Text, 1, 3))
                    *escala) + 100, (CInt(Mid(txtdy.Text, 1, 3)) - CInt(Mid(cadena, 3, 3)))
                    *escala + 100), vbYellow
'Actualiza el valor del Angulo que se muestra en pantalla
ElseIf codigo = "AN" Then
        If Mid(cadena, 3, 3) = 0 Then
            OB0.Value = 1
```

```
            ElseIf Mid(cadena, 3, 3) = 90 Then
                OB90.Value = 1
            ElseIf Mid(cadena, 3, 3) = 180 Then
                OB180.Value = 1
            ElseIf Mid(cadena, 3, 3) = 270 Then
                OB270.Value = 1
            End If
        ElseIf codigo = "OK" Then
    'Habilita el timer que genera el punto titilante
Timer1.Enabled = True
            'Deshabilita el timer marca el trayecto
            Timer2.Enabled = False
            'Limpia la coordenada X
            CoordX.Text = ""
            'Limpia la coordenada Y
            CoordY.Text = ""
            'Habilita el texto de la coordenada X para ingresar un valor
            CoordX.Locked = False
            'Genera un cuadro indicando que llego exitosamente a destino
            MsgBox "Llegó a Destino", vbExclamation, Info
        End If
            'Borra la variable
cadena = ""
End If
End If
End Sub
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
Private Sub Command1_Click()
Unload Form1
End Sub
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
Private Sub CoordX_Change()
'Ingresa al if cuando existen 3 digitos ingresados
If Len(CoordX.Text) = CoordX.MaxLength Then
'Limita el maximo valor en X a la distancia total en X - un 25% de la distancia en Y
If (CInt(CoordX.Text) > (CInt(Mid(txtdx.Text, 1, 3)) - 0.25 * CInt(Mid(txtdy.Text, 1, 3)))) Then
If ((CInt(Mid(txtdx.Text, 1, 3)) - 0.25 * CInt(Mid(txtdy.Text, 1, 3))) < 100) Then
'Agrega un 0 en caso que la coordenada X sea menor a 100
```

```
        CoordX.Text = "0" + CStr((CInt(Mid(txtdx.Text, 1, 3)) - 0.25 * CInt(Mid(txtdy.Text, 1, 3))))
Else
            CoordX.Text = (CInt(Mid(txtdx.Text, 1, 3)) - 0.25 * CInt(Mid(txtdy.Text, 1, 3)))
        End If
    ElseIf (CInt(CoordX.Text) < 0.25 * CInt(Mid(txtdy.Text, 1, 3))) Then
        CoordX.Text = "0" + CStr(0.25 * CInt(Mid(txtdy.Text, 1, 3)))
Else
        'Habilita el texto de coordenada Y para ingresar un valor
CoordY.Locked = False
    End If
Else
'Deshabilita el texto en Y en caso de que se borre el contenido de coordenada X
CoordY.Locked = True
End If
End Sub
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
Private Sub CoordX_KeyPress(KeyAscii As Integer)
If (KeyAscii < 48) Or (KeyAscii > 57) Then
    '8 es el borrar 'Esto para que se escriba solo numeros
If (KeyAscii <> 8) Then
KeyAscii = 0
    End If
End If
End Sub
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
Private Sub CoordY_KeyPress(KeyAscii As Integer)
If (KeyAscii < 48) Or (KeyAscii > 57) Then
'8 es el borrar 'Esto para que se escriba solo numeros
If (KeyAscii <> 8) Then
KeyAscii = 0
    End If
End If
End Sub
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
Private Sub CoordY_Change()
'Ingresa al if cuando existen 3 digitos ingresados
If Len(CoordY.Text) = CoordY.MaxLength Then
        'Limita el maximo valor en Y a la distancia total en Y - un 25% de la distancia en X
```

```
If (CInt(CoordY.Text) > (CInt(Mid(txtdy.Text, 1, 3)) - 0.25 * CInt(Mid(txtdx.Text, 1, 3)))) Then
        If ((CInt(Mid(txtdy.Text, 1, 3)) - 0.25 * CInt(Mid(txtdx.Text, 1, 3))) < 100) Then
'Agrega un 0 en caso que la coordenada X sea menor a 100
CoordY.Text = "0" + CStr((CInt(Mid(txtdy.Text, 1, 3)) - 0.25 * CInt(Mid(txtdx.Text, 1, 3))))
Else
CoordY.Text = (CInt(Mid(txtdy.Text, 1, 3)) - 0.25 * CInt(Mid(txtdx.Text, 1, 3)))
        End If
 ElseIf (CInt(CoordY.Text) < 0.25 * CInt(Mid(txtdx.Text, 1, 3))) Then
 CoordY.Text = "0" + CStr(0.25 * CInt(Mid(txtdx.Text, 1, 3)))
Else
'Habilita el Boton ir para el envio de coordenadas
CmdIr.Enabled = True
    End If
Else
'Deshabilita el boton ir si se borra la coordenada en Y
CmdIr.Enabled = False
End If
End Sub
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
Private Sub Form_Load()
    For i = 1 To 255
'Genera la lista puertos COM posibles
Combo1.AddItem ("COM" + Str(i))
Next
EnableCloseButton Me.hWnd, False
'Configuracion del puerto
Comm.Settings = "9600,N,8,1"
'Establece el tamaño del buffer de trasmision
Comm.OutBufferSize = 512
'Establece el tamaño del buffer de recepcion
Comm.InBufferSize = 1024
'Borra el contenido del buffer de trasmision
Comm.OutBufferCount = 0
'Borra el contenido del buffer de recepcion
Comm.InBufferCount = 0
'Se genera el evento OnComm por cada caracter recibido
Comm.RThreshold = 1
'Evita que se genere el evento OnComm cuando se trasmiten datos
```

```
Comm.SThreshold = 0
'Habilita que se genere el evento OnComm cuando llegue el caracter ascii 26
Comm.EOFEnable = True
cont = 1
color = vbBlack
End Sub
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
Private Sub OB0_click()
'Habilita el boton para iniciar
cmdcomenzar.Enabled = True
angulo = "0"
End Sub
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
Private Sub OB90_click()
'Habilita el boton para iniciar
cmdcomenzar.Enabled = True
angulo = "90"
End Sub
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
Private Sub OB180_click()
'Habilita el boton para iniciar
cmdcomenzar.Enabled = True
angulo = "180"
End Sub
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
Private Sub OB270_click()
'Habilita el boton para iniciar
cmdcomenzar.Enabled = True
angulo = "270"
End Sub
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
Private Sub Timer1_Timer()
'Este timer lo que hace es generar el punto titilante rojo, si color esta en rojo entonces grafica el punto,
'cuando el color cambia a negro borra toda la pantalla y genera la grilla nuevamente, se hizo así porque si
'solamente se borra el punto rojoy éste está sobre la grilla, al borrar el punto se borra parte de la línea de la
'grilla también
If color = vbBlack Then
    Picture1.Cls
```

```
For i = 1 To 9
        Picture1.Line (0, (Picture1.Height / 10) * i)-(Picture1.Width, (Picture1.Height / 10) * i), vbGreen
        Picture1.Line ((Picture1.Width / 10) * i, 0)-((Picture1.Width / 10 * i), Picture1.Height), vbGreen
    Next
color = vbRed
ElseIf color = vbRed Then
    Picture1.DrawWidth = 8
    Picture1.PSet (CInt(Mid(txtpx.Text, 1, 3)) * escala, (CInt(Mid(txtdy.Text, 1, 3)) - CInt(Mid(txtpy.Text, 1, 3))) * escala), color
    Picture1.DrawWidth = 1
color = vbBlack
End If
End Sub
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
Private Sub Timer2_Timer()
'Este timer va dibujando el trayecto que recorre el robot a medida que avanza a la nueva posicion
Picture1.DrawWidth = 8
Picture1.PSet ((CInt(CoordX.Text)) * escala, (CInt(Mid(txtdy.Text, 1, 3))
        - CInt(CoordY.Text)) * escala), vbBlue
Picture1.PSet (CInt(Mid(txtpx.Text, 1, 3)) * escala, (CInt(Mid(txtdy.Text, 1, 3))
        - CInt(Mid(txtpy.Text, 1, 3))) * escala), vbRed
Picture1.DrawWidth = 1
End Sub
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
Private Sub Command1_Click()
'Se guarda la longitud maxima, sea la de X o la de Y
If CInt(Mid(Form1.txtdx.Text, 1, 3)) > CInt(Mid(Form1.txtdy.Text, 1, 3)) Then
longitud = CInt(Mid(Form1.txtdx.Text, 1, 3))
Else
longitud = CInt(Mid(Form1.txtdy.Text, 1, 3))
End If
'Segun el valor de longitu se toma un valor de escala para la grilla
If (longitud > 250) And (longitud < 300) Then
escala = 20
ElseIf (longitud > 200) And (longitud < 250) Then
escala = 24
ElseIf (longitud > 150) And (longitud < 200) Then
escala = 30
```

```
ElseIf (longitud > 100) And (longitud < 150) Then
escala = 40
ElseIf (longitud > 50) And (longitud < 100) Then
escala = 60
ElseIf (longitud > 0) And (longitud < 50) Then
escala = 120
End If
'Se ajusta el ancho de la grilla
Form1.Picture1.Width = CInt(Mid(Form1.txtdx.Text, 1, 3)) * escala
'Se ajusta la altura de la grilla
Form1.Picture1.Height = CInt(Mid(Form1.txtdy.Text, 1, 3)) * escala
'Se ajusta la posicion desde la izquierda de la grilla
Form1.Picture1.Left = (Form1.Width - Form1.Picture1.Width) / 2
'Se ajusta la posicion desde arriba de la grilla
Form1.Picture1.Top = (6000 - Form1.Picture1.Height) / 2
'Se hacen visible varias etiquetas en el formulario 1
Form1.Label5.Visible = True
Form1.Label6.Visible = True
Form1.Label7.Visible = True
Form1.Label8.Visible = True
'Se habilita el cuadr de coordenada X para ingresar un valor
Form1.CoordX.Locked = False
'Se suma la palabra "cm" a los valores obtenidos
Form1.Label6.Caption = CStr(CInt(Mid(Form1.txtdx.Text, 1, 3)) / 10) + " cm"
Form1.Label7.Caption = CStr(CInt(Mid(Form1.txtdy.Text, 1, 3)) / 10) + " cm"
Form1.Timer1.Enabled = True
Form1.cmdcomenzar.Enabled = False
Unload Form2
End Sub
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
Private Sub Form_Load()
'Llama a un módulo que deshabilita la cruz de la ventana para que la cierre
EnableCloseButton Me.hWnd, False
End Sub
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
Private Sub txtp4_Change()
Command1.Enabled = True
End Sub
```