ROBOT BUSCADOR DE MANCHAS POR VISIÓN

Alejandro Ibarra
Antonio Blasco
Cristian Vázquez
Julio Álvarez

GRADO EN INGENIERÍA DE COMPUTADORES
FACULTAD DE INFORMÁTICA
UNIVERSIDAD COMPLUTENSE DE MADRID

Trabajo Fin de Grado en Grado en Ingeniería de Computadores

Curso Académico 2016/2017

Directores:

José Ignacio Gómez Pérez
Christian Tenllado van der Reijden

Autorización de difusión

Alejandro Ibarra
Antonio Blasco
Cristian Vázquez
Julio Álvarez

Madrid, a 1 de septiembre de 2017

Los abajo firmantes, matriculados en el Grado de Ingeniería Informática de la Facultad de Informática, autorizan a la Universidad Complutense de Madrid (UCM) a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a su autor el presente Trabajo Fin de Grado: "Robot buscador de manchas por visión", realizado durante el curso académico 16-17 bajo la dirección de José Ignacio Gómez Pérez y la co-dirección de Christian Tenllado Van der Reijden en el Departamento de Arquitectura de Computadores y Automática, y a la Biblioteca de la UCM a depositarlo en el Archivo Institucional E-Prints Complutense con el objeto de incrementar la difusión, uso e impacto del trabajo en Internet y garantizar su preservación y acceso a largo plazo.

"El ingeniero siempre se siente preocupado cuando sus planos empiezan a trocarse en piezas, en una máquina "viva". ¿Qué resultará, qué aspecto tendrá? En los planos de impecable diseño todo puede estar en su sitio, más en cuanto se hacen las piezas, ateniéndose al proyecto, en unos sitios no encajan, en otros funcionan mal."

Aleksandr Kótov (Ingeniero y jugador de ajedrez soviético)

Agradecimientos

A mi novia y a mi familia por toda la paciencia que me han tenido estos años.

Alejandro

A mi familia y amigos, que siempre me han apoyado.

Antonio

A todos los compañeros que han hecho esto posible y a mi familia por apoyarme tanto.

Cristian

A toda la gente que me apoyado durante estos años y principalmente a mi familia.

Julio

A José Ignacio Gómez Pérez y a Christian Tenllado van der Reijden por toda la ayuda prestada.

A todos ellos, muchas gracias.

Índice general

Índice de figuras

Resumen

La idea de este proyecto se toma del concurso organizado por Mathworks, Mission On Mars Robot Challenge. En él no se trata de crear un robot, el cual es provisto por la organización, sino de optimizar el funcionamiento que simule el de los robots que viajan a Marte para explorar el terreno superando obstáculos.

A más pequeña escala, se desarrolla el robot que, mediante un hardware más básico y un diseño propio, va a ser capaz de detectar una serie de manchas de un color determinado sobre el terreno gracias a la cámara que posee. Una vez vistas todas las que son alcanzables desde su posición, visitará la más cercana.

Por otro lado, se expondrán una serie de ideas para la optimización del robot, que debido a una serie de razones no han llegado a ser implementadas. Esta empieza por añadir un nuevo hardware al robot y así este sería capaz de generar un mapa del terreno, lo cual haría que pudiera visitar más objetivos y de una forma más eficiente.

Palabras clave

Raspberry, Raspbian, Robot, Buscador, Cámara, Impresora, 3D, OpenCV, 900-00008, WiringPi.

Abstract

The idea for this project is taken of the contest organized by Mathworks, Mission On Mars Robot Challenge. That contest is not based on create a new robot, which is provided by the organization, but to optimize how the robot works and simulate the behaviour of the robots sent to Mars to explore its surface.

On a small scale, a robot with a basic hardware and its own design is developed to be able to find color spots in the field using the camera. Once all the spots, reachable from the initial position are seen, it will visit the nearest.

On the other side, a set of ideas will be exposed to optimize the robot, but they haven't been finally implemented due several reasons. This set begins adding new hardware to the robot to be able to generate a map of the field, to make possible visit more spots and in an efficient way.

Keywords

Raspberry, Raspbian, Robot, Searcher, Camera, Printer, 3D, OpenCV, 900-00008, WiringPi.

Capítulo 1

Introducción

1.1. Idea del MarsRover

La idea para el desarrollo de este trabajo de fin de grado, surge del concurso organizado por MathWorks, "Mission on Mars Robot Challenge" [1]. Los robots Rover tienen una misión que llevan a cabo cumpliendo 2 objetivos: Explorar el planeta Marte mientras identifican ciertos elementos y además debe hacerlo evitando ciertos obstáculos del terreno.

La misión de los participantes es mejorar un robot dado, como el de la Figura 1.1, para que sea capaz de visitar varios puntos recorriendo la menor distancia posible y en un tiempo mínimo. Esto se hará optimizando tanto el propio robot de MATLAB como los algoritmos de Simulink. Tanto el modelo del robot creado con una impresora 3D, como la Raspberry Pi [2], el Arduino y el software de MATLAB y Simulink son provistos a los participantes por la organización MathWorks.

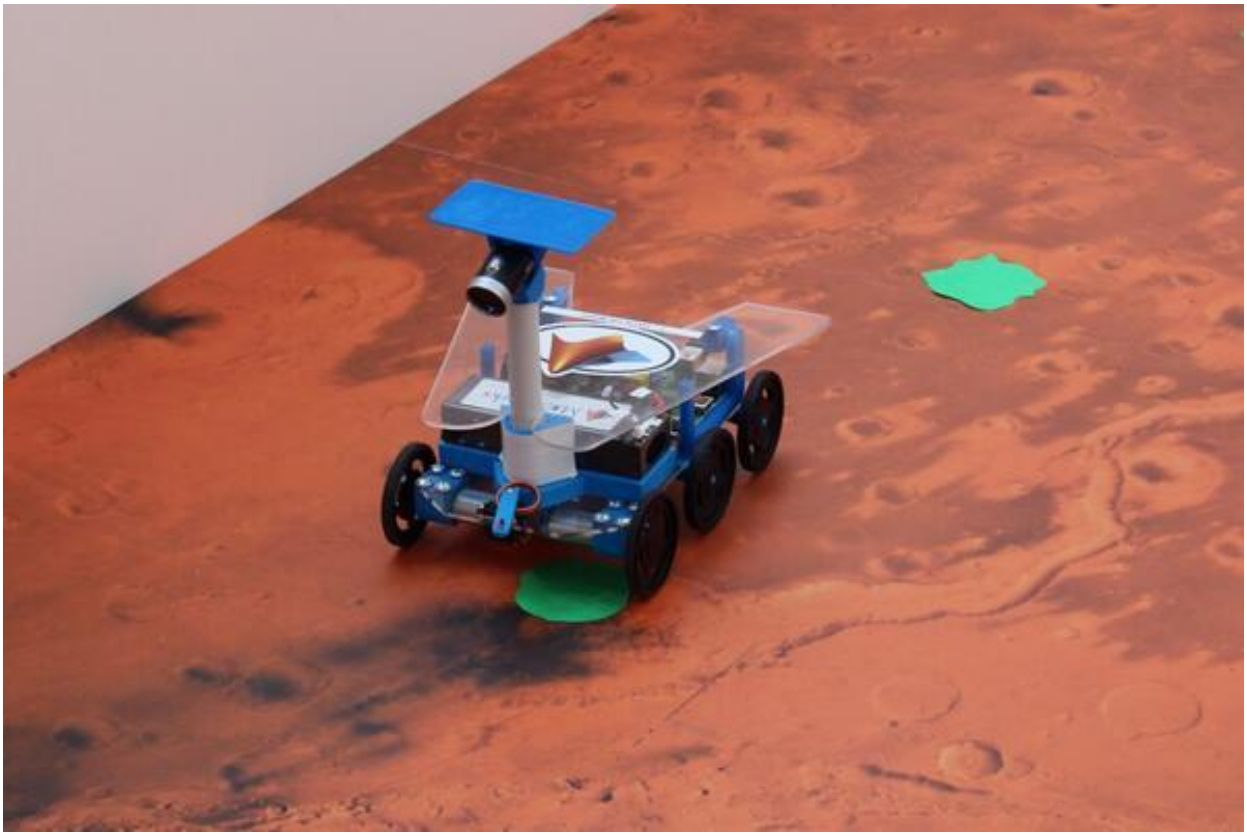


Figura 1.1: Rover que proporciona Mathworks a los participantes del concurso

1.2. Objetivo del proyecto

El objetivo del proyecto es realizar un robot mediante herramientas libres y abiertas, tanto de software como de hardware, este debe ser lo más asequible posible, y además se ha de emplear un lenguaje de programación común y extendido.

Este robot, ha de cumplir con las especificaciones del concurso MathWorks y realizar la tarea establecida que dicho concurso exige, todo ello, sin depender de la herramienta y lenguaje no libres que se emplean, ni de su hardware.

Por otro lado, con este sistema se pretende aprender acerca de qué elementos hardware

usar, qué tipo de lenguaje software emplear, qué esquemas y organizaciones son las mejores para estructurar un robot además de cómo se desarrollan las pruebas cuando existen factores físicos que salvar.

El escenario en el que se va a ejecutar el funcionamiento, es una superficie en la que el robot se encuentra rodeado de una serie de manchas de un determinado color. Con esta situación, gracias a una cámara, el autómata será capaz de percibir las manchas que lo rodean y una vez las haya descubierto todas deberá ir una por una a visitarlas, es decir, ir a su posición y situarse sobre ellas. Este objetivo ideal se deberá realizar de una forma en la que el robot, recorriendo la menor distancia posible y en un tiempo mínimo, complete la búsqueda de todos y cada uno de los objetivos sin repetir ninguno.

1.3. Especificaciones del robot

Para la implementación del robot hay que tener en cuenta el funcionamiento que va a desarrollar, así se podrán elegir debidamente los componentes y esquemas a emplear.

El diseño, para cumplir su cometido, ha de poseer necesariamente una cámara para la percepción de sus objetivos. Esta ha de ofrecer la suficiente calidad debido a que es la única fuente de entrada de información para que el robot obre.

Como la función que el sistema va a ejecutar es la de generar un determinado movimiento para que se vaya situando en unas localizaciones concretas, se necesita un actuador que transforme la información lógica del robot en un movimiento físico, dicho componente serán unos motores.

En cuanto al cuerpo del robot, se ha de tener un dispositivo que sea capaz de gestionar

los datos de entrada del mismo, que son los ofrecidos por la cámara, tratarlos, interpretarlos y actuar en consecuencia mandando a los motores las órdenes correctas según la información del exterior. Este componente central, para ello, tiene que poder correr un programa en un lenguaje con la suficiente potencia, adaptabilidad y capacidad de modularización del software como para encargarse de esta vital tarea. Por un lado deberá ser lo suficientemente potente como para saber recoger los datos que la cámara ofrece y por otro adaptarse y generar el tipo de información que los motores van a interpretar para desarrollar su función.

Con todo esto, el robot ya está listo para funcionar. El comportamiento esperado es aquel en el cual en primer lugar registre todos los objetivos alcanzables desde la posición inicial, tras esto decida la mejor forma de recorrerlos para que seguidamente se mueva a visitarlos.

1.4. Metodología y plan de trabajo

Aunque en un principio la organización del proyecto se realizó de una forma descentralizada, más adelante, se fue haciendo evidente la necesidad de estructurar y organizar el proyecto repartiendo las tareas y objetivos.

Durante la fase inicial no se realizó un reparto de tareas estricto, se trató de que todos los integrantes del grupo estuvieran al tanto de las tareas hasta que el proyecto tomara un poco de forma. Tras esto, se empezó a distribuir de una manera más clara los cometidos a realizar.

Empezando por la cámara y debido a su gran importancia, un integrante del grupo se encargó casi exclusivamente de todo lo relacionado con este tema, desde la investigación sobre su correcto y optimo funcionamiento, pasando por todo tipo de filtrados, hasta el desarrollo del módulo software que gestionase la labor de este componente.

Los otros tres integrantes, partiendo de un módulo básico de control de la cámara, se centraron más en las pruebas para la obtención de un funcionamiento estable del robot. Estas pruebas eran de vital importancia puesto que era imperativo que la máquina de estados que buscaba los objetivos, tuviera un comportamiento correcto, yendo lo más directo posible hacia la mancha detectada.

En la fase final del proyecto, se volvió a hacer un reparto de tareas en el que otro integrante, de los que formaban el grupo de tres en las pruebas del robot, se dedicó a organizar la memoria, distribuir el contenido a exponer y a establecer las pautas para realizarlo. Ese mismo integrante, además de gestionar la memoria, se encargó de la investigación del paso siguiente a realizar en la evolución del robot, solo la idea en cuanto a un mejor funcionamiento. Con todo esto, cada miembro del grupo, a la hora de redactar la documentación, tenía claro qué parte debía documentar puesto que era la labor que había desarrollado principalmente durante el proceso de creación del proyecto.

1.5. Estructura del documento

La memoria, dividida en capítulos, está organizada y estructurada de la siguiente forma:

- En el Capítulo 2 se abarca a teoría del procesamiento de la imagen para la detección de manchas gracias a la cámara. Se exponen los cálculos y acciones a realizar para poder detectar objetos y así llegar a visitarlos.
- El Capítulo 3 se detallan las razones por las cuales se han elegido los componentes hardware que forman el robot, que son la cámara, los motores y el microprocesador que gestiona todo el funcionamiento. Por otro lado se explica el diseño del chasis debido a las especificaciones de los componentes a usar y el funcionamiento que va a desarrollar el robot.

- El Capítulo 4 expone la arquitectura del software del robot, cómo se han realizado lo módulos de control de la cámara y de los motores, gracias a OpenCV [3] y WiringPi [4] respectivamente, y el cuerpo central encargado del funcionamiento del robot.

- En el Capítulo 5 se relatan los resultados obtenidos finales, es decir, el funcionamiento de la versión estable entregada.

- Es en el Capítulo 6 donde se va a desarrollar por completo la idea del siguiente paso a realizar en la evolución del robot, desde la implementación del nuevo hardware, hasta la explicación del software mejorado y funcionamiento paso a paso.

- En el Capítulo 7 se discuten las conclusiones alcanzadas después de desarrollar el robot, si el resultado obtenido es el mismo que el esperado inicialmente y los problemas encontrados así como las soluciones aplicadas. Además se comentan todos los conocimientos obtenidos y empleados tras la realización del mismo.

Capítulo 2

Detección de manchas por visión

El objetivo de esta parte del trabajo es el de detectar varios objetos (manchas), de un color determinado, sobre una superficie plana para luego identificar la posición (x,y) de cada uno de estos objetos. Para llevar a cabo todas esta tareas era necesario realizar una serie de pasos para conseguir la ubicación de estos objetos.

En la Figura 2.1 se describen los pasos llevados a cabo durante la detección de las manchas. Éstos consisten en:

1. Captura de un fotograma por medio de la cámara. Esta nos devuelve una imagen a color compuesta por tres canales (R, G, B).

2. Transformación de la imagen capturada (color) a una imagen en escala de grises con exceso de color verde de un solo canal. El exceso de color verde está basado en la forma en que percibimos las personas los colores. Se consigue acentuar los objetos de color verde dotando de mayor peso al canal G frente al resto de canales.

3. Normalización del histograma, lo cual consiste básicamente en conseguir que los valores mínimos y máximos de la escala de grises sean 0 y 255 respectivamente.

4. Binarización de la imagen. Con ello se consigue una imagen que pasa de estar a escala

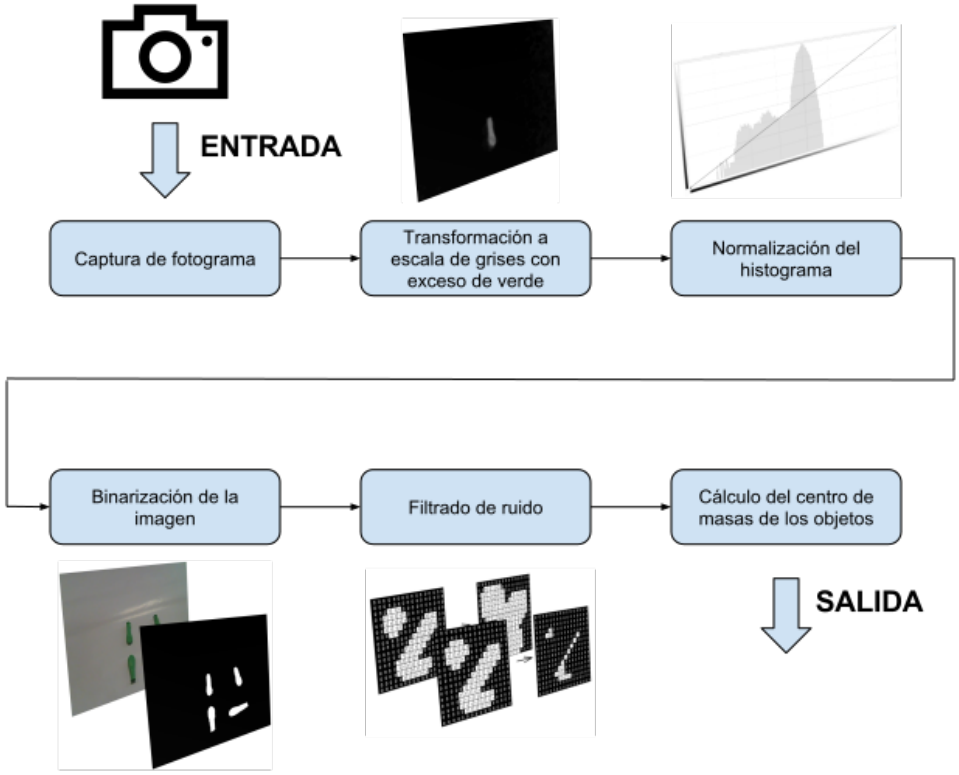


Figura 2.1: Detección de las manchas en la imagen

de grises a otra binaria (0 o 1). En esta imagen binaria, los objetos de color verde estarán representados con un valor 1 y el resto de la imagen con 0. Esto permite diferenciar de forma clara los objetos cuyas coordenadas se quiere conseguir.

5. Filtrado de ruido. Esto nos permite eliminar posibles manchas provocadas por el ruido en la imagen que se genera por problemas con la calidad de la lente de la cámara, diferencias en las intensidades de luz, etc.

6. Se obtiene la posición de los objetos dentro de la imagen mediante el cálculo del centro de masas de cada uno de ellos.

A continuación, se detalla en qué consiste cada uno de los pasos.

2.1. Transformación de la imagen a escala de grises con exceso de verde

Este apartado está basado en el estudio realizado por Martín Montalvo Martínez durante su tesis doctoral. En ella explica cómo realizar la detección de objetos mediante el filtrado por longitudes de onda.

Dado que la cámara proporciona una imagen de salida compuesta por los tres canales espectrales visibles R, G y B correspondientes a los colores Rojo, Verde y Azul respectivamente, se puede jugar con el peso que se le da a cada uno de estos canales para conseguir realzar los objetos que nos interesa detectar.

En concreto se hace uso de la técnica de "Exceso de Verde". En la tesis se explica que esta técnica se basa en que, debido a que existe una mayor relevancia de la componente verde frente a las otras dos (roja y azul), y a que existe una desigualdad al tratarse de un canal frente a dos, se le dota de mayor importancia al canal verde haciendo que se multiplique por un factor de dos. El "Exceso de Verde" se obtiene como el valor del canal verde menos el valor del canal azul y menos el valor del canal rojo, resultando en la siguiente ecuación:

$$ExG = 2G - B - R$$

Se quiere convertir la imagen original a escala de grises. Esto significa hacer una transformación lineal de las coordenadas RGB a un espacio de una sola dimensión. Para ello hay que realizar una operación aritmética en la que se multiplica una matriz de N x M dimensiones, correspondiendo estos valores a la resolución de la imagen (imagen original), por una matriz "auxiliar" de 1x3 dimensiones. Los valores de esta matriz auxiliar corresponden con

los pesos que se le asignan a cada uno de los canales RGB previamente calculados según el índice de coloración que se quiere conseguir. Dicha transformación se puede apreciar en la Figura 2.2.

(a) Imagen original (b) Imagen a escala de grises

Figura 2.2: Transformación de una imagen RGB a escala de grises

2.2. Normalización del histograma de la imagen

Esta operación permite acotar el rango de los valores de los píxeles de la imagen. Esto es de gran ayuda a la hora de agrupar los valores representativos de la imagen alrededor de un rango. Básicamente consiste en asignar los límites máximos y mínimos de dicho rango. Para unos valores máximos y mínimos de 255 y 0 respectivamente no se almacenaría la información de ninguno de los píxeles que que se encuentren por encima o por debajo de estos límites(Figura 2.3).

2.3. Binarización de la imagen

El algoritmo básico consiste en fijar un valor de umbral para nuestra imagen e ir seleccionando aquellos píxeles cuyos valores sean mayores que dicho umbral y descartando los que no.

(a) Imagen original (b) Imagen normalizada

Figura 2.3: Normalización del histograma de la imagen

```
if (P(x,y) > THRESHOLD) then
P(x,y) = 1
else
P(x,y) = 0
```

Una vez que se haya realizado este proceso para todas las regiones de la imagen, tendremos una imagen binaria. En la Figura 2.4 de abajo vemos un ejemplo.

(a) Imagen original (b) Imagen binaria

Figura 2.4: Binarización de la imagen

Durante el proceso de mejora en el filtrado de la imagen se han investigado varios métodos para conseguir realizar la binarización de ésta. Los métodos que se han probado son los descritos a continuación:

2.3.1. Simple Thresholding

La idea de este tipo de umbralización es muy fácil de entender. Consiste en que, si el valor del píxel es mayor que el valor de umbral, se le asigna un valor de 1 (comúnmente blanco), en caso contrario, se le asigna el valor 0 (comunmente negro). Los resultados conseguidos mediante este método han sido bastante buenos, ya que, permitían identificar cada uno de los objetos de forma clara. Por ello ha sido uno de los más utilizados durante las pruebas, demostrando ser el más robusto.

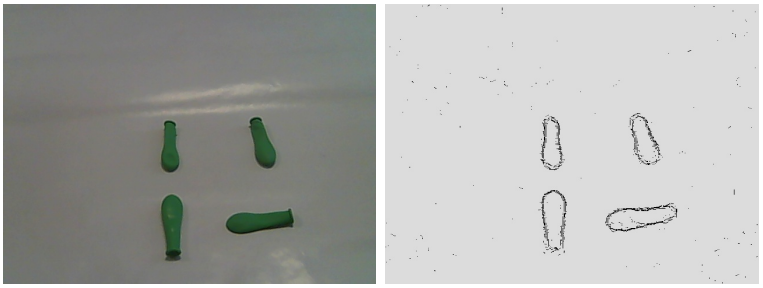
2.3.2. Adaptive Thresholding

En el caso anterior se describe el uso de un tipo de umbralización que utiliza valores globales. Pero esto no suele ser lo mejor cuando se trabaja con imágenes cuyas zonas tienen condiciones de luz distintas. Para este caso es necesario hacer uso de un método adaptativo. El algoritmo funciona calculando el valor de umbral para pequeñas regiones de la imagen, por lo que, tendremos diferentes valores para regiones diferentes de la misma imagen y esto no dará un mejor resultado para imágenes con luminosidad variable.

En la teoría este parece el mejor tipo de umbralización que se le puede aplicar al robot, sin embargo, en las pruebas realizadas se veía que no siempre se conseguía la luminosidad deseada y que esto afectaba a la detección de los objetos, tal y como podemos ver en la Figura 2.5.

Este tipo de umbralización permite utilizar dos métodos para calcular el valor de umbral:

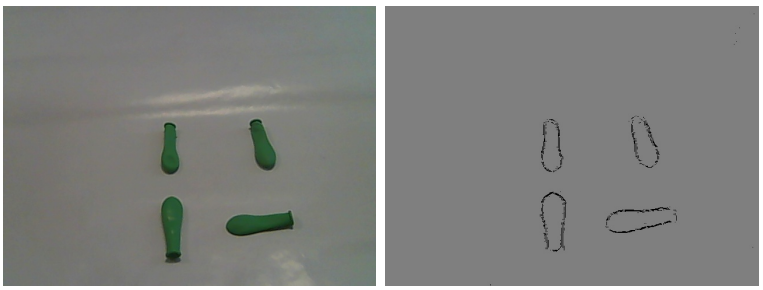
- El método en el cual el valor de umbral es la media del área de las regiones contiguas.
- Método en el cual el valor de umbral es la suma ponderada de los valores contiguos donde los pesos son una ventana gaussiana.



(a) Imagen original (b) Imagen binaria

Figura 2.5: Resultado del Adaptive Thresholding

Con el segundo método se consiguen resultados similares con pequeñas diferencias tal y como se aprecia en la Figura 2.6.



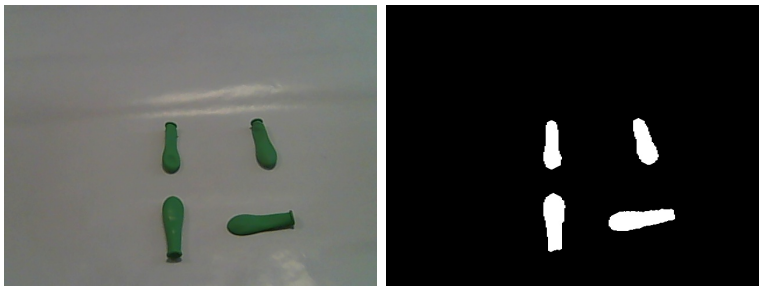
(a) Imagen original (b) Imagen binaria

Figura 2.6: Resultado del Adaptive Thresholding (Gaussiana)

2.3.3. Otsu's Binarization

Para el caso de la umbralización global (Simple Thresholding) se explicó que se usaban valores de umbral arbitrarios. A priori no se podía saber cuál era el valor adecuado, por lo que, la única forma de encontrar dicho valor era mediante la técnica de prueba y error. Una forma de evitar este tipo de procedimientos y poder calcular el valor de umbral de forma automática, es mediante el uso del método de umbralización por el algoritmo de Otsu. Éste trabaja con imágenes bimodales, es decir, una imagen cuyo histograma está representado por dos picos. En simples palabras, lo que hace este algoritmo es calcular de forma automática

el valor de umbral mediante la aproximación del valor central entre estos dos picos. Este método produce resultados muy buenos, tal y como se puede ver en la Figura 2.7, cuando se trabaja con imágenes bimodales, aunque durante las pruebas realizadas no siempre se conseguía esto.



(a) Imagen original (b) Imagen binaria

Figura 2.7: Resultado del Otsu's Binarization

2.4. Eliminación de ruido y detección de falsos objetivos mediante operaciones morfológicas

Este paso es necesario a la hora de conseguir detectar los objetivos debido a que evita que se detecten falsos objetivos. Uno de los principales problemas encontrados a la hora de realizar las pruebas consistía en un ruido de fondo provocado por el exceso de iluminación en la imagen capturada o por la "confusión" del sensor de la cámara al detectar partes de la imagen como objetos con el mismo rango de color. Es por esto que se aplican operaciones morfológicas para reducir dichos efectos. Las operaciones morfológicas son una teoría y una técnica para el análisis y procesamiento de estructuras geométricas. Es comúnmente aplicado a las imágenes digitales aunque también se aplica a muchas otras estructuras espaciales. Los operadores morfológicos básicos son erosión, dilatación, apertura y cierre.

2.4.1. Erosión

El efecto básico de la erosión consiste en "desgastar" los límites de las regiones de los píxeles que se encuentren en primer plano (representados en color blanco típicamente). Para entender el efecto de la erosión sobre una imagen binaria, podemos pensar en el elemento erosionador como una especie de "rejilla" que recorre todos los píxeles de la imagen. La forma de determinar si al final del proceso de erosión, los píxeles de la imagen quedarán a 0 ó a 1, es teniendo en cuenta que este se considerará 1 sólo si todos los píxeles que se encuentren debajo del elemento erosionador (rejilla) también están a 1. Por lo tanto, lo que sucede, es que todos los píxeles cerca de lo límites (entiéndase los bordes de las regiones blancas) serán descartados dependiendo del tamaño del elemento erosionador. Todo esto provoca que el área de la región blanca disminuya, de tal forma que solo los objetos más representativos permanecerán en la imagen (Figura 2.8).

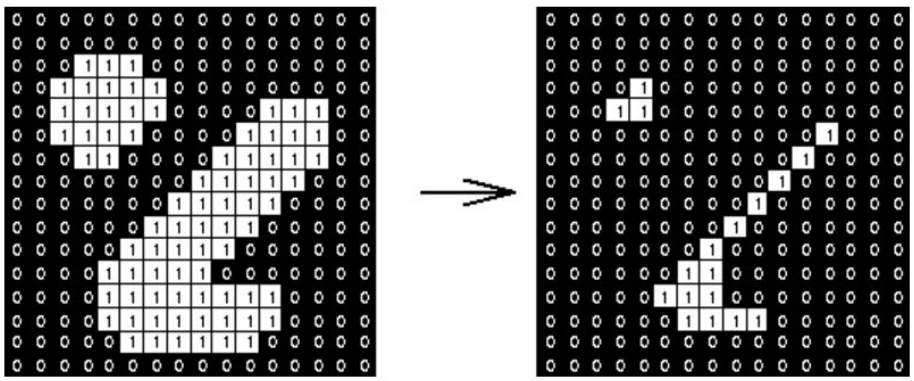


Figura 2.8: Erosión de la imagen

2.4.2. Dilatación

Esta operación es justo lo contrario a la erosión. En este caso se puede considerar la misma idea de la rejilla que recorre todos los píxeles de la imagen solo que ahora se trata de un elemento dilatador. Esto significa que para que en la imagen resultante, un pixel

resulte con valor 1, al menos uno de los píxeles debajo del elemento dilatador tiene que tener valor 1. Esta operación morfológica nos permite unificar partes de un objeto dividido como resultado de la binarización, dado que como resultado de aplicarla, se consigue que el área de las regiones blancas aumente (Figura 2.9).

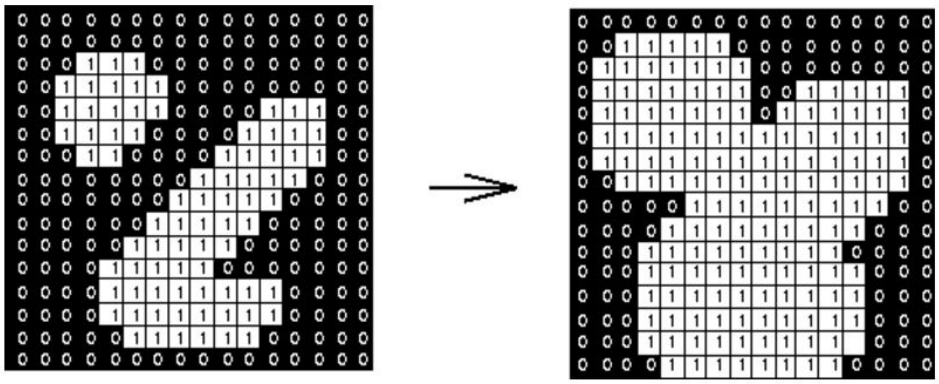


Figura 2.9: Dilatación de la imagen

2.4.3. Apertura

La operación de apertura se obtiene como la combinación de la erosión seguido por una dilatación. Esto permite la eliminación de ruido en la imagen mediante la erosión y con la dilatación se consigue que las regiones blancas vuelvan a tener aproximadamente el mismo área (Figura 2.10).

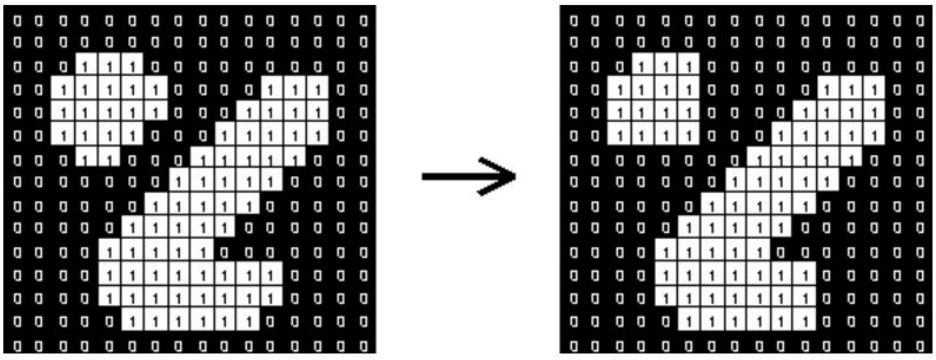


Figura 2.10: Apertura de la imagen

2.4.4. Cierre

Al igual que pasaba con las operaciones de erosión y dilatación, la operación de cierre es el opuesto a la de apertura, ya que, se consigue mediante la combinación de una dilatación seguido de una erosión. Esto permite corregir los errores de la binarización haciendo que se eliminen los huecos en las regiones blancas (Figura 2.11).

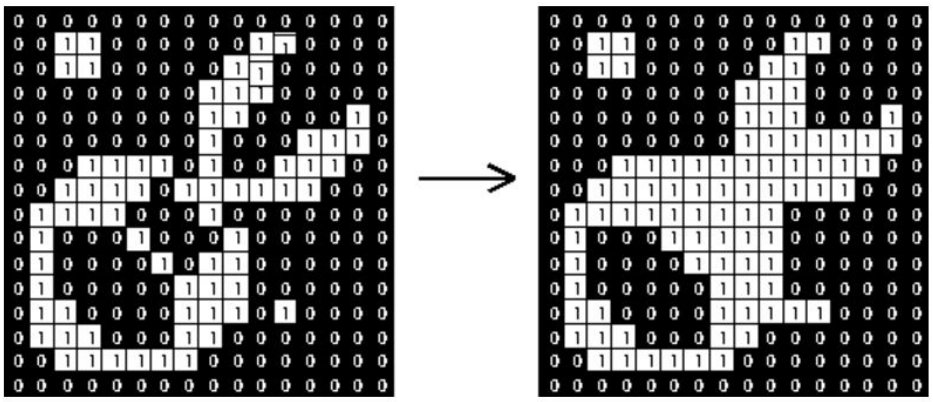


Figura 2.11: Apertura de la imagen

2.5. Obtención de la posición de los objetos

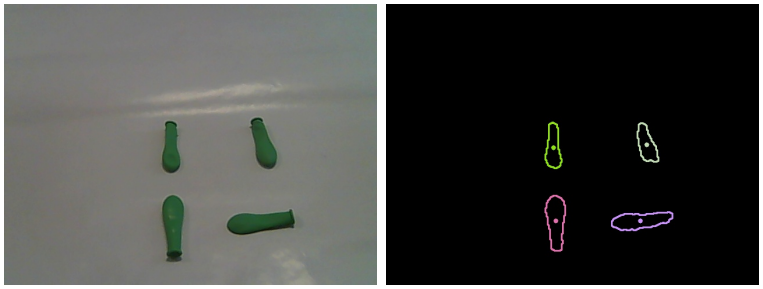
Una vez realizada la identificación de los objetivos dentro de la imagen frente a otros elementos, el siguiente paso consistía en el cálculo de las coordenadas de dichos objetivos. Para llevar a cabo este cálculo de la posición se ha hecho uso del concepto de los "Momentos". Se entiende este concepto como una medida cuantitativa que es utilizada en mecánica y estadística para describir la distribución espacial de un conjunto de puntos. Dicho de otra forma, los momentos son un conjunto de escalares que proporcionan una medida agregada de un conjunto de vectores. La elección de uno de estos momentos depende de la distribución de los datos y del conjunto de vectores que interesan distinguir.

Si los puntos de una imagen representan el concepto de masa, entonces m00 (momento 0) es el total de la masa (en nuestro caso, el área de la región blanca), m01 (momento 1)

dividido por m00 (momento 0) representa la coordenada X, y m10 (momento 2) dividido m00 representa la coordenada Y.

$$\overline{x} = \frac{m_{10}}{m_{00}} \qquad \overline{y} = \frac{m_{01}}{m_{00}}$$

Una vez obtenidas las coordenadas de aquellos objetos que se quiere detectar, podemos determinar en qué punto de la imagen se encuentran Figura 2.12.



(a) Imagen original (b) Imagen binaria

Figura 2.12: Resultado de la obtención de la posición de los objetos

Capítulo 3

Hardware del robot

En este capítulo se trata el tema de la elección de los componentes hardware, como son el microprocesador, la cámara y los motores. Además, se exponen los argumentos acerca de las razones del diseño del chasis del robot con una impresora 3D.

3.1. Microprocesador

A la hora de elegir un microprocesador, se ha optado por una Raspberry Pi 3 Modelo B [5] cuyo aspecto es el de la Figura 3.9.

dado que desde un primer momento se quería realizar un proyecto tanto de software como de hardware libre, se pensó en utilizar la Raspberry Pi. Ésta ofrece facilidades a la hora de trabajar con periféricos, dado el amplio número de librerías que existen. Además, cuenta con el respaldo de una comunidad muy amplia, lo cuál siempre es una ventaja.

Otra ventaja que llevó a elegir esta placa, es el gran potencial del microprocesador, suficiente para poder controlar y procesar tanto las tareas de procesamiento de imagen como la gestión de los motores.

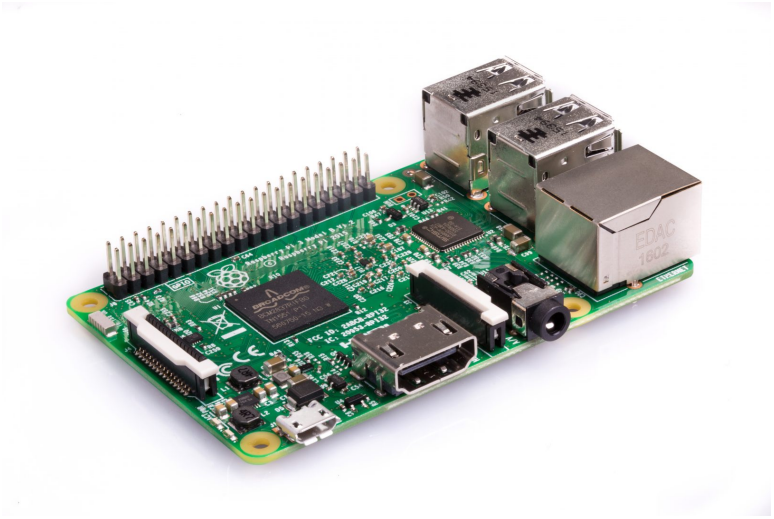


Figura 3.1: Raspberry Pi 3 Modelo B

Para la cámara, que se hablará más adelante, tenemos el CSI Camera Port, en el cual conectaremos la cámara; para los motores, se usarán los pines GPIO (40 pines), de los cuales 2 de ellos tienen la opción de generar una señal PWM.

Otras de las razones por las que se eligió este microprocesador es que está dotado de 4 puertos USB, entrada de Ethernet, Wi-Fi, entrada HDMI y otros componentes los cuales son más relevantes.

Por último se miró la documentación que hay sobre este microprocesador, la cual es mucha ya que al ser libre está muy estudiada y probada por mucha gente, lo cual da un gran empujón a la hora de elegirlo, si a todo esto se le suma el precio que es muy económico para todo lo que ofrece.

3.2. Cámara

A la hora de elegir la cámara, como nunca antes se había trabajado con este tipo de componentes, se escoge el módulo de la cámara que ofrece la Raspberry Pi [7], debido a que es sencillo de utilizar gracias a la compatibilidad que ofrece con el microprocesador. Además, las imágenes ofrecidas por esta cámara son de una calidad suficiente para el proyecto a desarrollar. Y si a estas razones se le añade que es un dispositivo económico, la elección de este componente está más que justificada. El aspecto que ofrece la cámara es el de la Figura 3.2.

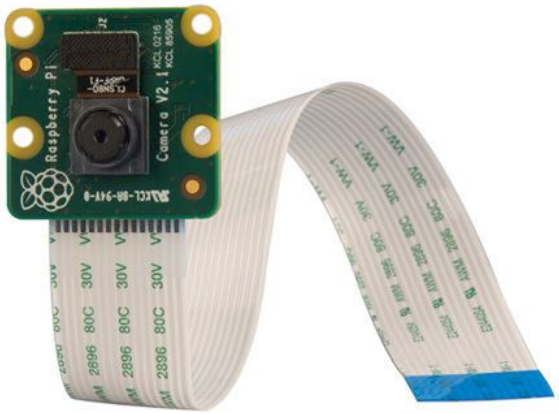


Figura 3.2: Vista de la cámara

3.2.1. Características

Utiliza el sensor de imagen IMX219PQ [6] de Sony que ofrece imágenes de vídeo de alta velocidad y alta sensibilidad.

Camera Module de Raspberry Pi contaminación de imagen reducida como ruido de patrón fijo y borrones. Dispone también de funciones de control automático como el control de exposición, el balance de blancos y la detección de luminancia.

Dispone de un enfoque fijo de 8 megapíxeles (incluye herramienta de ajuste de enfoque) siendo compatible con las resoluciones 1080p (30 fps), 720p (60 fps) y VGA (90 fps).

Un cable plano de 15 cm fijado a las ranuras del módulo directamente en el puerto de interfaz serie de su cámara Pi (CSI) como se muestra en la Figura 3.3.

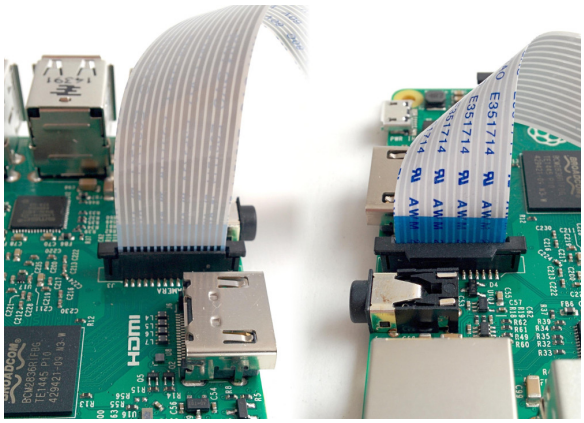


Figura 3.3: Vista de la conexión de la cámara

Una vez conectado, puede acceder a la placa de cámara a través de la capa de abstracción multimedia (MMAL) o el vídeo para API de Linux (V4L).

Si estamos trabajando con el sistema operativo Raspbian podemos hacer uso de la aplicación "Raspberry Pi Configuration Tool" para habilitar la cámara en el caso de que contemos con una pantalla (Figura 3.4).

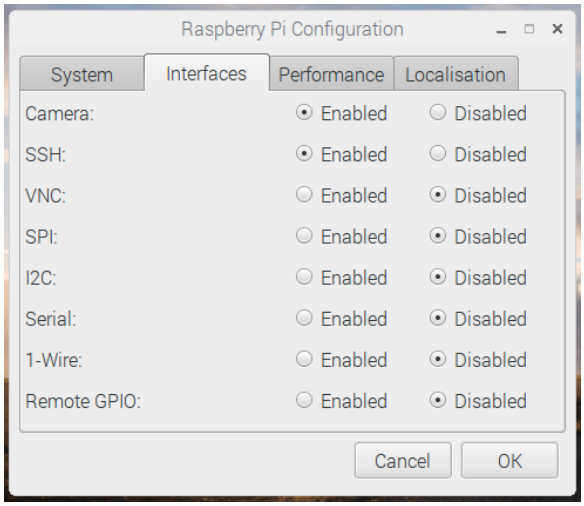


Figura 3.4: Pantalla de la Raspberry Pi Configuration Tool

Otra opción es hacerlo a través de un terminal mediante el uso de la aplicación para consola "raspi-config". Los pasos a seguir son:

- Abrir un terminal y ejecutar la linea sudo raspi-config ¡, lo cual hará que aparezca una pantalla como la de la Figura 3.5.

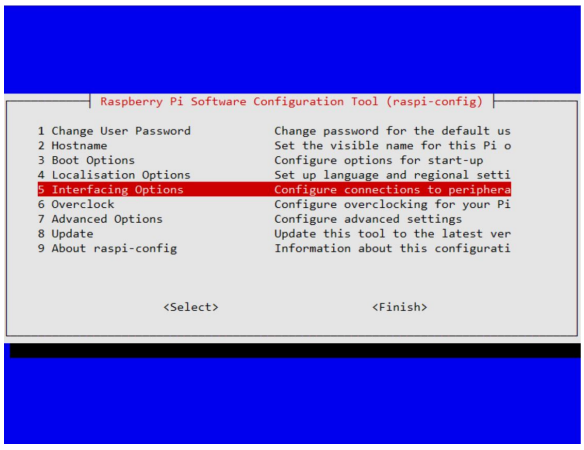


Figura 3.5: Pantalla de la configuración de la cámara con raspi-config

- Seleccionar la opción 5 (Interfacing Options).

- Seleccionar la opción 1 donde pone Camera. A continuación, seleccionar "Si" cuando nos pregunte si queremos habilitar el módulo de la cámara.

- Lo siguiente es comprobar que el driver de Linux para la cámara se encuentra cargado. Para ello ejecutamos el comando lsmod y comprobamos que en la lista de los módulos cargados se encuentra *bcm2835_ v4l2*. De no ser así habrá que añadirlo mediante el el uso del comando modprobe: sudo modprobe bcm2835_v4l2. A continuación hay que reiniciar la Raspberry.

Una vez se hayan realizado estos pasos ya podemos empezar a utilizar la cámara. Una forma fácil de comprobar que funciona es con un pequeño script en python:

```
1 from picamera import PiCamera
2 from time import sleep
3
4 camera = PiCamera()
5
6 camera.start_preview()
7 sleep(10)
8 camera.stop_preview()
```

3.3. Motores

A la hora de elegir los motores, se podía elegir entre servomotores o motores paso a paso. Se decidió usar los servomotores simplemente por su fácil disposición. Además, son sencillos de utilizar conjuntamente con la Raspberry Pi, ya que en la GPIO hay varios puertos PWM disponibles con los que poder enviar órdenes a los motores. Por otro lado proporcionan un gran precisión de movimiento, para así poder controlar los movimientos del robot, por pequeños que sean.

El modelo elegido es el Parallax Continuous Rotation Servo (#900-00008 [8]).

Los servomotores, son motores electrónicos en los que se puede controlar la velocidad de giro además de la posición dentro de su rango de operación.

Están formados por un motor eléctrico, una caja reductora con engranajes y un circuito electrónico de control. Se controlan mediante la modulación por ancho de pulsos (PWM).

3.3.1. Conexión

Los motores tienen las siguientes conexiones:

- Vdd: Cable rojo. Con él se proporciona corriente al motor (4 a 6 V).
- Vss: Cable negro. Tierra, se conecta a una tierra común entre la Rasperry y el motor.
- Signal: Cable blanco (amarillo en la foto). Se conecta al pin PWM de la Raspberry para transmitir señales a los motores.

Para conectar los motores y la Raspbery se ha usado una placa de prototipado. Por un lado, la corriente de los motores se proporciona de manera externa a través de la power-bank, ya que la Raspberry no tiene suficiente potencia para alimentar dos motores a parte del procesamiento del programa. La tierra es importante que sea común entre la Raspberry Pi y los motores. En cuánto a las señales de los motores, estas se conectan a los puertos PWM de la Raspberr, que corresponden con los pines 13 y 18 de la numeración GPIO, el 13 para el motor derecho y el 18 para el motor izquierdo. No confundir la numeración de los pines GPIO (Figura 3.6) con la numeración de los pines físicos (Figura 3.7).

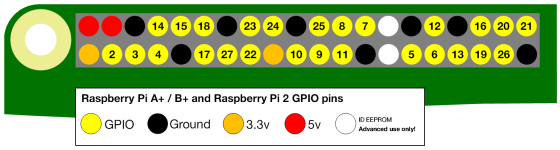


Figura 3.6: Numeración de los pines GPIO de la Raspberry Pi

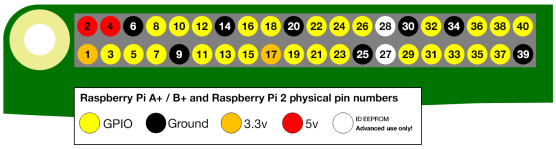


Figura 3.7: Numeración de los pines físicos de la Raspberry Pi

El esquema general de la conexión del robot se puede ver en la Figura 3.8.

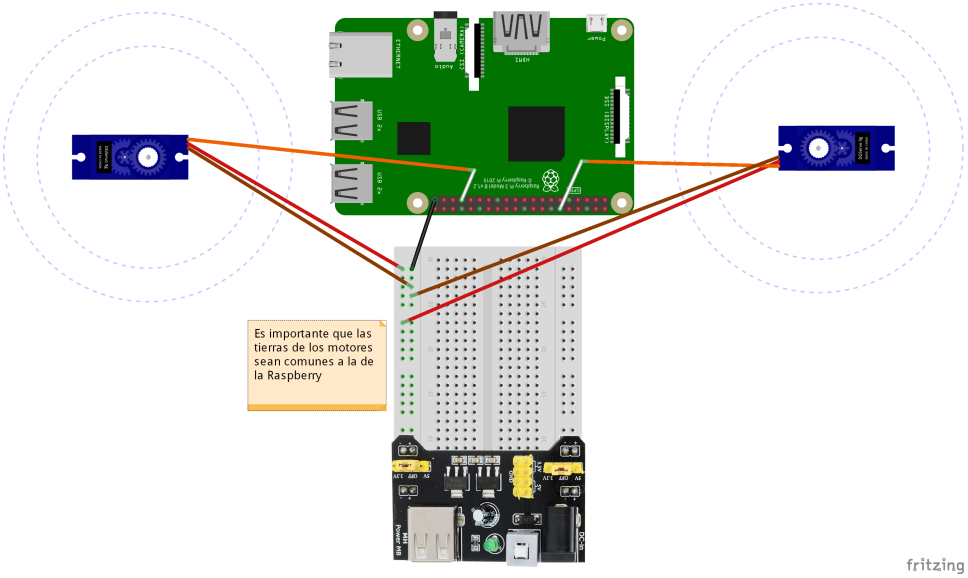


Figura 3.8: Esquema general de la conexión del robot

3.4. Diseño del chasis

La realización del chasis se ha realizado mediante una impresora 3D, con ello se consigue que el diseño se adecue a la perfección a las medidas, forma y óptimo funcionamiento del

robot.

La ventaja de hacerlo con una impresora 3D y por lo cual se eligió, es poderlo hacer a la medida de los elementos elegidos y poder diseñar el chasis a medida de esos componentes hardware.

Para realizarlo, se ha utilizado un programa de software libre de diseño asistido por ordenador (CAD), se trata de OpenSCAD[9]. Este programa está orientado más al diseño de piezas que a la animación debido a que no es programa de diseño interactivo, sino que es un compilador 3D que interpreta un lenguaje de definición de formas y lo traduce a un modelo tridimensional. De esta forma se le ofrece al diseñador un control total sobre el objeto pudiendo hacer cambios precisos de manera sencilla y establecer parámetros configurables.

3.4.1. Base

La primera parte del diseño es la de la base del robot, aquella sobre la que se asientan todos los elementos del mismo. Esta base tiene las dimensiones necesarias para soportar sobre ella, en primer lugar, el soporte de la cámara, el cual se ubicará en la parte frontal de la base. Este soporte irá atornillado sobre la base para una perfecta sujeción evitando así movimientos oscilantes de la cámara y una posible caída, de esta manera se asegura que la imagen de la cámara es del todo estable.

Por otro lado, tras el soporte para la cámara se colocará el soporte de la Raspberry el cual irá pegado con pegamento para plástico sobre la base.

El tamaño de la base viene principalmente definido por el de la batería que se emplea,

que para tener la mayor autonomía posible es de gran tamaño. Esto es porque la batería irá sobre la base, justo detrás del soporte de la cámara y debajo del de la Raspberry. Para sujetar la batería al robot, se han diseñado una serie pestañas a ambos lados pensadas para enganchar en ellas una goma elástica.

Por último, la base del robot ha de albergar también los motores, para ello se han creado por la parte de debajo unas solapas, dos a cada lado del robot, con dos taladros cada una para colocar los motores entre cada par de placas de plástico y atornillarlos a ellas.

En la Figura 3.9 se puede apreciar la vista principal de la base, desde un lateral se puede ver los soportes para los motores (Figura 3.11), la vista superior nos brinda la posibilidad de ver bien el tamaño de la base así como los taladros para el soporte de la cámara (Figura 3.10) y por último, viendo la placa desde atrás captamos perfectamente la forma de los soportes que alojarán los motores (Figura 3.12).

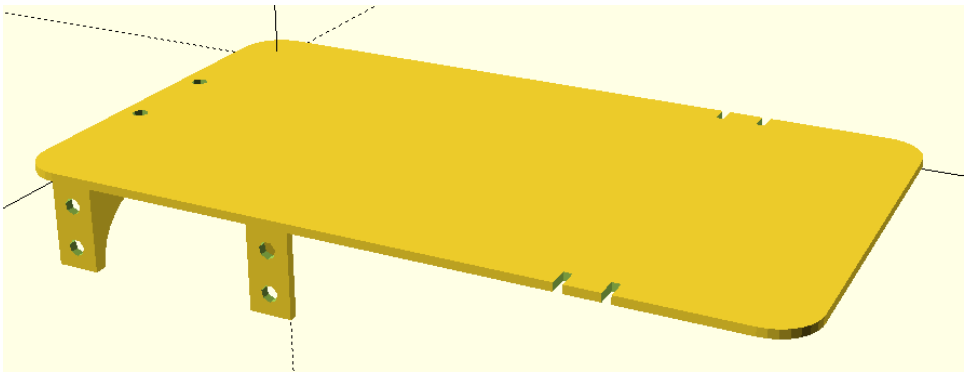


Figura 3.9: Vista con perspectiva de la base del robot

Figura 3.10: Vista superior de la base del robot

Figura 3.11: Vista lateral de la base del robot

.

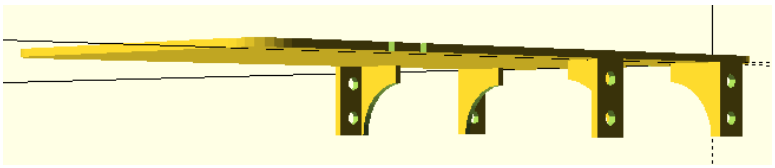


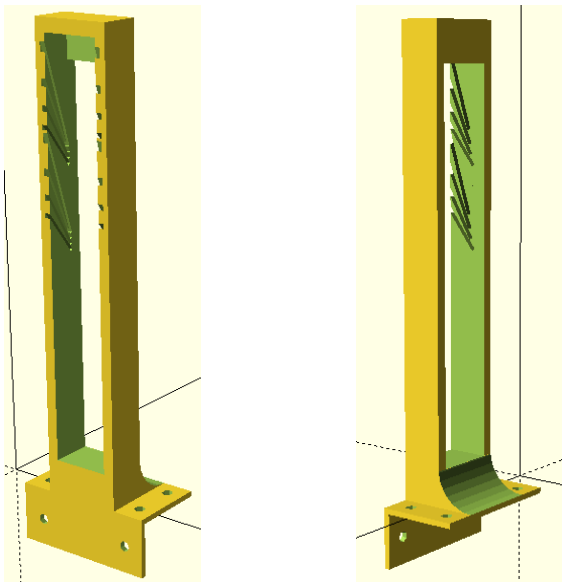
Figura 3.12: Vista desde atrás de la base del robot

3.4.2. Soporte de la cámara

El diseño del soporte de la cámara se basa en una estructura rectangular alta que ofrece la posibilidad de colocar la cámara a distintas alturas y ángulos como bien se puede ver en la Figura 3.13.

La base del soporte tiene unos taladros destinados a la sujeción del mismo con la base del robot. Unido a la base, se encuentra un soporte frontal que está destinado a la colocación de un sensor infrarrojo. Una posible mejora del robot era la de añadir a la parte frontal del robot este tipo de sensor para la detección de obstáculos en el camino, así que, para tener la posibilidad de implementar esta funcionalidad, se incorporó ese soporte al diseño.

Sobre la base se encuentra una estructura alta hueca la cual, a ciertas alturas, tiene una serie de muescas a modo de carriles (simétricas a ambos lados) destinadas a encajar la cámara deslizándola en ellas.



(a) Frontal (b) Posterior

Figura 3.13: Vista del soporte de la cámara

3.4.3. Soporte de la Raspberry

Este soporte irá ubicado detrás de la cámara y se pegará con pegamento para plásticos a la base del robot.

La estructura consiste en una especie de “tejado” de tal forma que sobre él irá la placa del microprocesador y bajo ella puede ir ubicada la batería del robot así como la placa de prototipado.

La finalidad de elevar la Raspberry, ya no solo por permitir ahorrar espacio, es la de poder colocar la cámara lo más alto posible debido a la corta longitud del cable que la conecta, así como la escasa movilidad que ofrece a consecuencia de la forma plana que tiene.

Con todo esto se tiene un soporte que permite tener la Raspberry por encima de la batería y la protoboard, y que se sujeta a este mediante 4 agujeros colocados conforme la placa los tiene dispuestos en su diseño.

En la Figura 3.14 y Figura 3.15 se puede ver la pieza colocada al revés puesto que, como se utiliza una impresora 3D, no se es posible imprimir demasiado material sin una base sobre la que apoyar. Mientras que en la Figura 3.16 se aprecia la posición en la que se colocará la estructura sobre la base del robot.

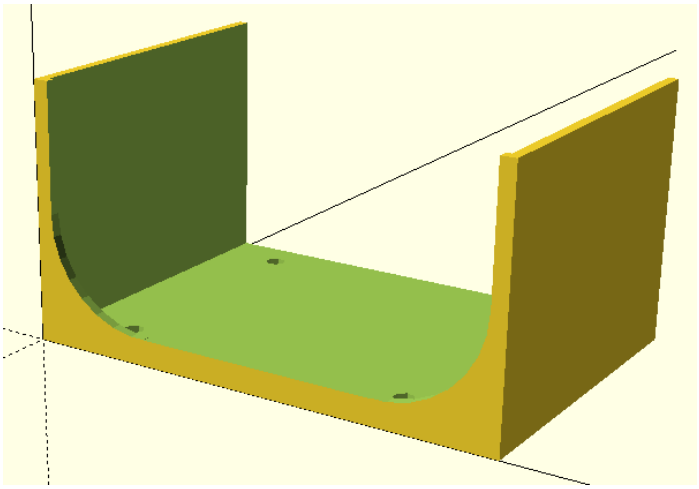


Figura 3.14: Vista inferior desde el frontal del soporte de la Raspberry Pi

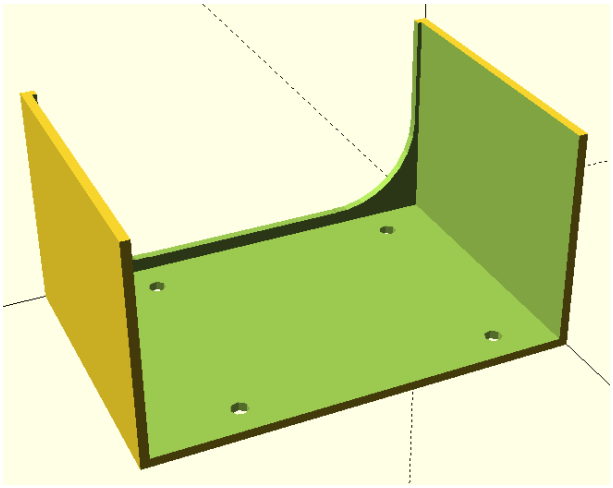


Figura 3.15: Vista inferior desde atrás del soporte de la Raspberry Pi

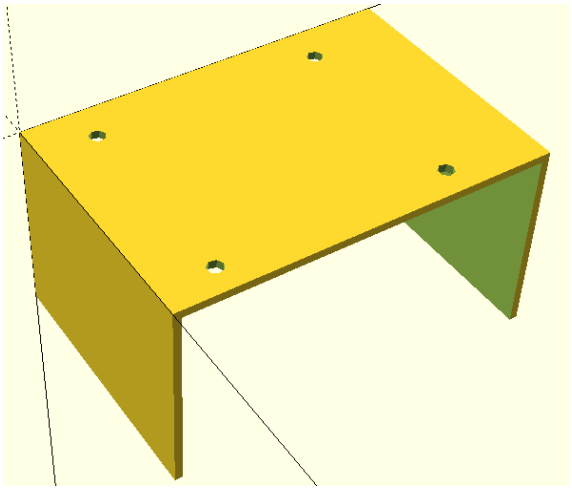


Figura 3.16: Vista superior del soporte de la Raspberry Pi

3.4.4. Soporte de la bola loca

El último elemento del chasis, que se aprecia en la Figura 3.17, consiste en un simple cilindro de un diámetro y altura acordes a la bola que se le colocará debajo, para que el robot quede equilibrado.

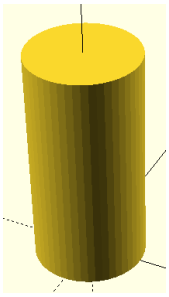


Figura 3.17: Vista del soporte de la bola loca

3.4.5. Simulación de la apariencia del robot en 3D

Con todo estos elementos es hora de realizar una simulación del prototipo con las piezas en su posición correcta.

En la Figura 3.18, la Figura 3.19 y la Figura 3.20 se puede apreciar cómo el soporte de la cámara se coloca en la parte frontal del robot para poder dirigirlo correctamente, la estructura que lleva anclada la Raspberry iría justo después para acercarla lo más posible al lugar donde se colocaría la cámara, permitiendo pasar su conector entre las estructuras verticales del soporte de esta.

Con este diseño además, se añade la simulación de cómo quedaría el robot con todos los elementos colocados. En la Figura 3.21, la Figura 3.22 y la Figura 3.23 se muestra en rojo la cámara y la Raspberry, colocadas en sus respectivos soportes, además de los motores, del mismo color, bajo la base del robot. Por otro lado, en gris, se ven las ruedas colocadas en su posición correcta respecto a los motores además de la batería que alimentará a todo el sistema. Todos los elementos modelados se han creados con las medidas reales y situados en su posición correcta con una precisión de milímetros que es lo que permite el programa.

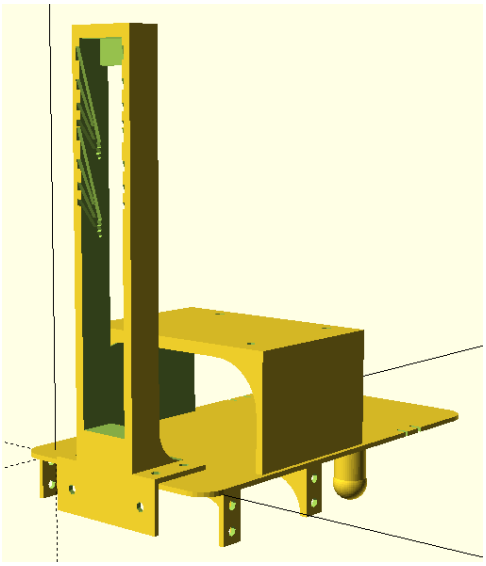


Figura 3.18: Vista frontal de la simulación del robot con todo el chasis en conjunto

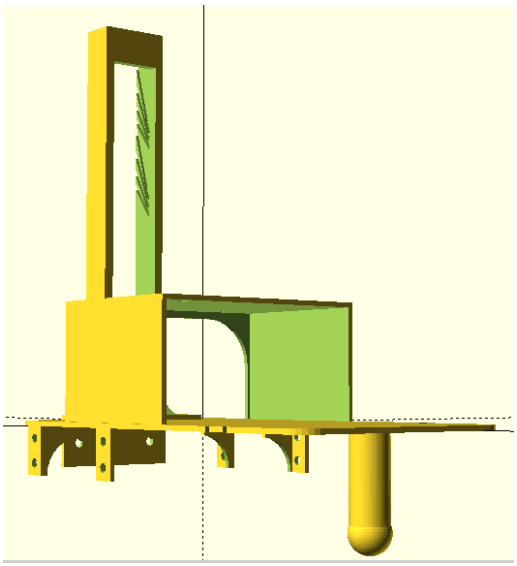


Figura 3.19: Vista desde atrás de la simulación del robot con todo el chasis en conjunto

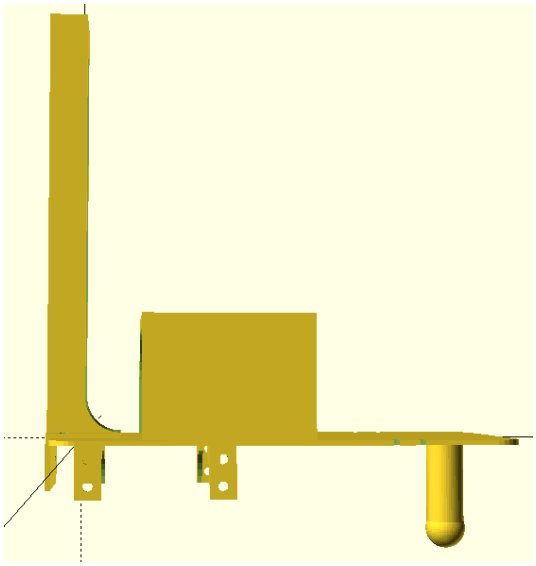


Figura 3.20: Vista desde el lateral de la simulación del robot con todo el chasis en conjunto

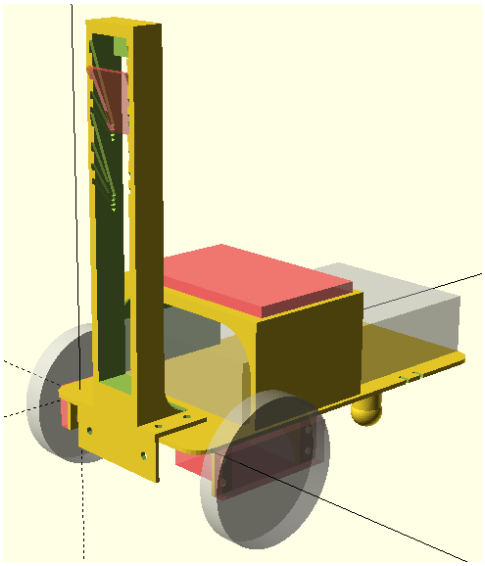


Figura 3.21: Vista desde atrás de la simulación del robot con el chasis completo y todos los elementos hardware simulados

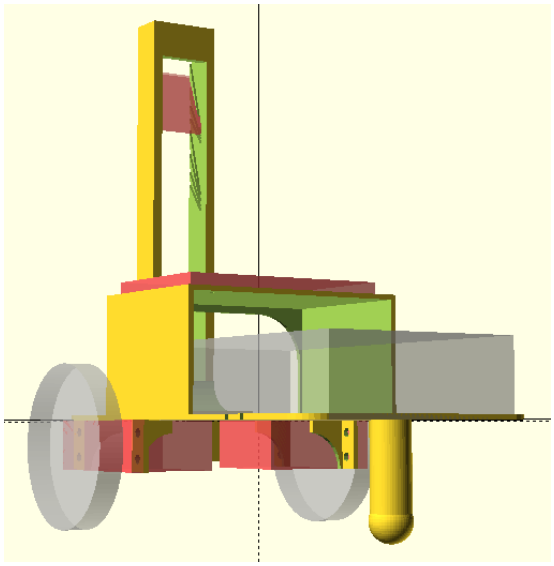


Figura 3.22: Vista desde atrás de la simulación del robot con el chasis completo y todos los elementos hardware simulados

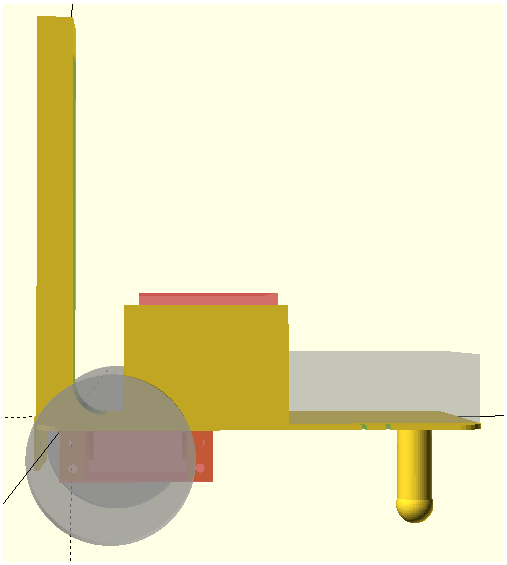


Figura 3.23: Vista desde atrás de la simulación del robot con el chasis completo y todos los elementos hardware simulados

Capítulo 4

Arquitectura del software

4.1. Estructura del software

La idea en cuanto a la estructura del software, es la de que se modularice lo máximo posible, de esta forma se organiza mucho mejor el código del proyecto, así se puede obsevar en la Figura 4.1.

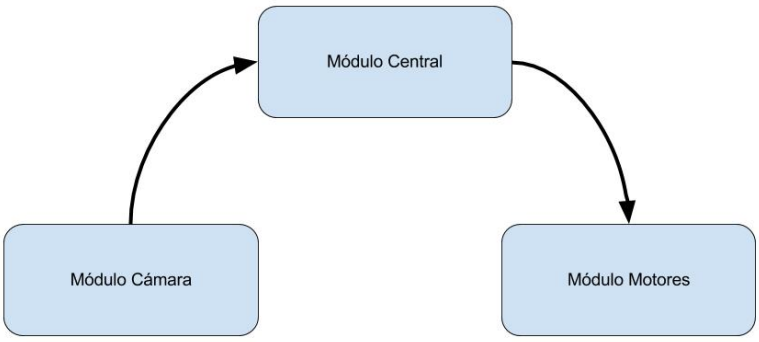


Figura 4.1: Esquema de los módulos software

Por un lado se ha de tener un módulo para el control de cada dispositivo, en este caso para la cámara y los motores. El módulo de la cámara será el encargado de obtener los datos de entrada del sistema, mientras que el módulo de los motores desarrollarán la salida ya que la finalidad del robot es la de que se mueva hacia los objetivos que se encuentren en el

mundo que percibe.

Por último y más importante, aparece el módulo central, gracias a él se controlará el funcionamiento de todo el robot. Posee una máquina de estados mediante la cual es capaz de registrar las imágenes procedentes del módulo de la cámara, procesar la información que estas le proporcionan y en consecuencia tomar las decisiones pertinentes para el movimiento que van a llevar a cabo los motores.

El cuerpo central del software del robot se basa en una máquina de estados de Moore, es decir, el comportamiento del robot depende únicamente del estado en el que se encuentre. En la Figura 4.2 se puede apreciar el diagrama de estados que deja claro la sucesión de acciones que llevará a cabo el robot en cada sesión de búsqueda de objetivos.

En cada una de estas sesiones el robot comenzará haciendo un giro completo sobre sí mismo con el objetivo de localizar los puntos visibles a su alrededor, y al mismo tiempo, calculando la distancia a cada uno de dichos puntos, se va almacenando el más cercano. En este momento se cambia de estado, y el comportamiento aquí comienza por orientarse hacia el punto que se va a visitar. Tras esto comienza la ejecución de la máquina de estados secundaria, la cual permite cambios entre todos sus subestados, en función de la posición actual del robot, con el fin de que el este se encuentre centrado respecto del objetivo. Finalmente, al encontrar el punto y situarse encima de el, se puede decir que habría terminado la ejecución de una iteración y comenzará el proceso de nuevo.

.

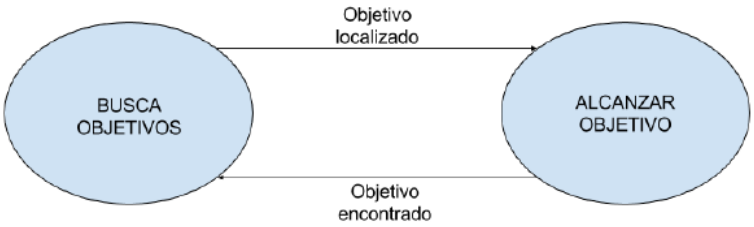


Figura 4.2: Máquina de estados

4.1.1. Estado de búsqueda de objetivos

En este estado el robot tiene que localizar todos los posibles puntos que se encuentren a su alrededor y clasificarlos para decidir cuál visitará. Los objetivos se ordenarán según la distancia a la que se encuentren del robot y el más cercano es aquel que visitará.

Para llegar a este punto el robot comenzará a girar sobre sí mismo, capturando imágenes con la cámara y almacenando dichos puntos. Para ello se realizan giros de 60 grados, hasta completar la vuelta completa tras 6 giros, de esta manera se divide el campo de visión que envuelve al robot en 6 sectores. Se sigue esta estrategia a fin de evitar también que en varias de las capturas se vea el mismo objetivo desde distinta posición, por tanto se podría afirmar que el ángulo de visión del robot son 60 grados aproximadamente.

Una vez ha dado una vuelta completa se elige el punto a visitar y se orienta el robot hacia dicho punto, es decir, se hacen tantos giros de 60^{o} como sectores haya recorrido el robot hasta que dicho punto quede en el campo de visión de este, y se pasa al siguiente estado.

4.1.2. Estado para alcanzar un objetivo

Cuando el robot se encuentra en este estado, se ejecuta otra máquina de estado de Moore encargada de buscar un objetivo en concreto. A través de la cámara se toman una serie de imágenes, para localizar al objetivo, y en función de la posición de éste, el robot se moverá.

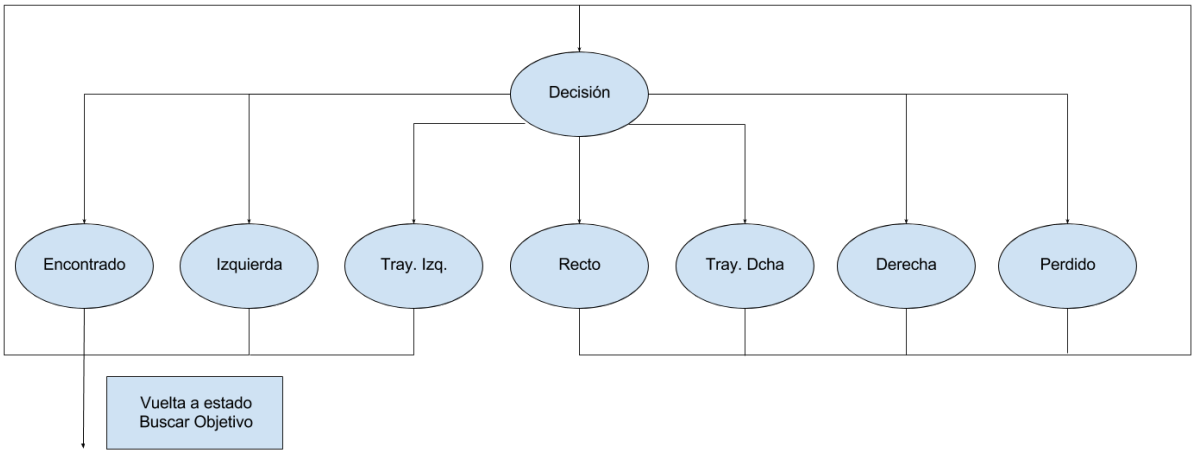


Figura 4.3: Máquina de estados secundaria

En esta máquina de estado, las entradas son las coordenadas en píxeles de los objetivos que el robot percibe en cada momento y según el valor de estas se irá a un estado o a otro independientemente del estado en el se encuentre cuando las entradas tomen su nuevo valor. Gracias a la Figura 4.4 se puede ver como el robot interpreta una captura de imagen. En función de en que sección se encuentre el objetivo, se elegirá uno de los estados de la máquina de estados secundaria.

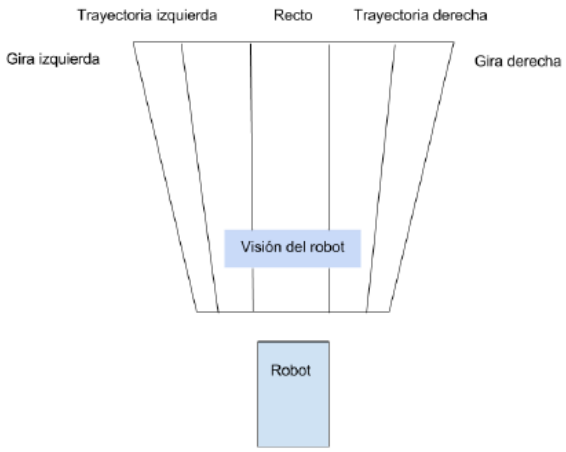


Figura 4.4: Esquema de la visión de robot y los subestados de busca objetivo

Recto

El objetivo percibido se encuentra en el eje central de la visión del robot, que tiene un margen comprendido entre la coordenada 250 y 390 de la componente X. Si el objetivo se encuentra en este punto el robot avanzará recto hacia él.

Trayectoria izquierda

Ahora el punto visto se encuentra un poco desplazado a la izquierda, entre las coordenadas 100 y 250 de las X, pero sin embargo muy cerca aún del eje central, por lo que la acción a desarrollar es la de corregir la trayectoria del robot haciendo que se oriente ligeramente a la izquierda para tratar de centrar el objetivo en la visión.

Trayectoria derecha

Es muy similar al estado anterior con la obvia diferencia de la posición del objetivo visto. En este caso el centroide del objeto está entre las coordenadas 390 y 540 así que solo habrá

que corregir ligeramente la trayectoria hacia la derecha.

Izquierda

Ahora la situación del objetivo ya no se encuentra lo suficientemente centrado como para avanzar hacia él y corregir la trayectoria, se encuentra en el resto de la pantalla por la izquierda, por tanto para solucionar esto hay que rotar en este estado, hacia la izquierda, hasta conseguir tenerlo centrado.

Derecha

Este estado es idéntico que el de "Izquierda", simplemente cambiando a derecha el sentido de giro debido a que la situación del punto percibido es ahora este otro lado.

Perdido

Cuando la cámara deja de percibir el punto (ha salido de su ángulo de visión) entonces nos encontramos con que hemos perdido el punto, por tanto la mejor opción es la de ejecutar el último estado distinto de éste con las últimas coordenadas distintas de -1 (valores que indican que el objeto ha dejado de percibirse) que se hayan obtenido, así tendremos muchas posibilidades de volver a encontrar el punto de nuevo.

Para gestionar este posible problema, se dispone de una serie de variables las cuales guardan el último estado que se ejecutó antes de perder de vista el objeto, así como de las coordenadas donde el robot vió al objetivo por última vez. Dichos datos sólo se actualizan cuando se ve el objeto o se le ha vuelto a encontrar tras haber dejado de verle.

Encontrado

El último estado es en el cual el objeto está centrado y lo suficientemente cerca del robot, o dicho de otra manera, el último estado ha sido "recto", "trayectoria izquierda" o "trayectoria derecha" y el centroide del objeto se encuentra por encima de las coordenadas 440 del eje Y (este valor marca cuando se pierde la visión del objeto por estar muy cerca de él y tenerlo prácticamente debajo de las ruedas del robot). Entonces simplemente se avanzará recto para situarse justo encima del objeto.

Una vez completado esto, el algoritmo se sale de la máquina del estados de "busca objetivo" y vuelve a la máquina de estados general, al estado busca objetivos de nuevo, para comenzar de nuevo el proceso y seguir buscando más objetivos.

Como conclusión a este apartado, se puede afirmar que este algoritmo funciona de manera bastante precisa, realizando las búsquedas de manera directa sin que el robot realice grandes correcciones en su trayectoria, aunque para ello se requiere que los movimientos sean cortos y lentos, lo que hace que el tiempo de ejecución para encontrar cada mancha sea un poco elevado. En general en entornos con varios puntos es capaz de detectar correctamente el más cercano, visitarlo y continuar buscando otras manchas, como así lo demuestran pruebas realizadas en entornos controlados, es decir, con los objetivos colocados en posiciones determinadas y con unas condiciones favorables de luz, para evitar reflejos, y superficie, para evitar que las ruedas pudieran deslizar.

4.2. Procesamiento de imagen: OpenCV

Para el control de la visión del robot se decidió utilizar una biblioteca open source escrita en C/C++ llamada OpenCV. Originalmente desarrollada por Intel, su primera versión alfa

apareció en enero de 1999, y ha sido utilizada para infinidad de aplicaciones, desde sistemas de seguridad con detección de movimiento, hasta aplicaciones de control de proceso donde se requieren reconocimiento de objetos. Esto es así gracias a su licencia BSD, que permite ser usada tanto para aplicaciones con propósitos comerciales y de investigación.

Esta librería permite llevar a cabo, de forma fácil, las tareas de procesamiento de imágenes y visión computarizada. Se trata de una librería con soporte para varios sistemas operativos, incluido el sistema Raspbian. Esta librería provee de una serie de estructuras básicas de datos para realizar operaciones con matrices y procesamiento de imágenes. También permite visualizar estos datos de forma muy sencilla y extraer información de imágenes y vídeo.

4.2.1. Descripción de la librería OpenCV

OpenCV se compone básicamente de los siguientes módulos:

- *cv*: Contiene las funciones principales de la biblioteca.
- *cvaux*: Contiene las funciones auxiliares (experimental).
- *cxcore*: Este es el módulo básico de OpenCV. Incluye las estructuras de datos básicas y las funciones básicas de procesamiento de imágenes. Este módulo también es usado por otros módulos como highgui.
- *highgui*: Este módulo provee interfaz de usuario, códecs de imagen y vídeo y capacidad para capturar imágenes y vídeo, además de otras capacidades como la de capturar eventos del ratón... etc.
- *imgproc*: Este módulo incluye algoritmos básicos de procesado de imágenes, incluyendo filtrado de imágenes, transformado de imágenes... etc.

- *video*: Este módulo de análisis de vídeo incluye algoritmos de seguimiento de objetos, entre otros.

El objetivo de esta parte del trabajo es el de detectar varios objetos (manchas), de un color determinado, sobre una superficie plana para luego identificar la posición (x,y) de cada uno de esos objetos. Para llevar a cabo todas estas tareas era necesario realizar una fase previa de filtrado de la imagen capturada y de este modo conseguir quedarnos solo con dichos objetos.

4.3. Módulo de la cámara

El módulo de la cámara es el componente software encargado del procesamiento de la imagen del robot. Este componente provee de una interfaz para proporcionar información acerca de los objetivos captados por la cámara. A continuación se describe dicha interfaz.

4.3.1. Descripción de las funciones

A continuación se describen las funciones del módulo mediante sus prototipos.

- `void * trackObject(void * obj);`

  Esta función se encarga de capturar fotogramas con la cámara del robot para hacer el seguimiento de un objeto mediante el calculo de sus coordenadas.

- `void * trackMultiObjects(void * objects);`

  Esta función se encarga de capturar fotogramas con la cámara del robot para detectar objetos en su campo de visión y calcular las coordenadas de dichos objetos. Estas coordenadas se retornan por referencia a través de del parámetro .ºbjects".

- `static void getMultiCoordinates(Mat input_img, list<t_Coordenada> &objects);`

  Permite calcular las coordenadas de cada objeto que aparezca en la imagen mediante la detección de contornos y el calculo del centro de masas de cada objeto.

- `static void toGrayscaleWithExcessOfColour(Mat input_img, Mat &output_img, double v`

  Transforma una imagen RGB a otra en escala de grises con exceso de color. Dependiendo de los valores del array "values"se aplicara un exceso de color a los canales R, G o B, lo que permitirá realzar los objetos cuyo color se encuentre entre estos espectros.

- `static void openingThreshold(Mat &image, int obj_radius);`

  Hace uso de las funciones .erode y dilate.en este orden para conseguir el efecto de apertura en la imagen. Permite eliminar el ruido provocado por la aparición de objetos pequeños.

- `static void closingThreshold(Mat &image, int obj_radius);`

  Hace uso de las funciones "dilate y erode.en este orden para conseguir el efecto de cierre en la imagen. Permite evitar que los objetos detectados aparezcan divididos.

- `static void thresholdOtsu(Mat input_img, Mat &output_img);`

  Permite generar una imagen binaria mediante el cálculo automático del valor de umbral a través del algoritmo de Otsu.

- `static void thresoldAdaptative(Mat input_img, Mat &output_img);`

Genera una imagen binaria usando el método adaptativo. De esta forma consigue no verse menos afectada ante condiciones de iluminación variable en diferentes zonas.

- `static void thresoldAdaptativeGaussian(Mat input_img, Mat &output_img);`

  El funcionamiento es igual al de la función "thresoldAdaptative"solo que en este caso se calcula el valor de umbral es la suma ponderada de los valores de cada región cuyos pesos corresponden con una ventana gaussiana.

- `void histogramCalculation(Mat image, Mat &histoImage);`

  Calcula el histograma de una imagen RGB y genera un gráfico con estos datos. Recibe como parámetros una imagen de tres canales y calcula el histograma de cada uno de ellos.

4.4. WiringPi

WiringPi es una librería escrita en C y liberada baja licencia GNU LGPLv3, que puede ser empleada en varios lenguajes de programación, en este caso será el lenguaje C++. Tiene una interfaz con las funciones basicas y faciles de usar, como para configurar valores Clock y Range, entre otros parametros y valores.

Su principal uso es el la programación de periféricos a través de los 26 pines de General Purpose Input Output (GPIO). En este proyecto, se emplearán para configurar los pines en modo PWM, los valores necesarios para el Clock y Ranger y para mandar la señal de PWM a los Servomotores. Pero se puede usar para configurar otros pines de la GPIO.

4.4.1. Funciones usadas

- *pinMode (int pin, int mode)*: Esta función permite configurar los pines usados de la GPIO y, en este caso particular, ponerlos en modo de PWM_OUTPUT.

- *pwmSetMode (int mode)*: Se usa para configurar el modo del PWM y a esta función se le pasará el PWM_MODE_MS, parámetro que indica que el PWM se transmitirá en milisegundos, ya que el ancho de pulso de los servos motores necesitan estar en estas unidades.

- *pwmSetRange (unsigned int range)*: Esta función sirve para asignar el rango (Range = 4300).

- *pwmSetClock (int divisor)*: Esta función sirve para asignar el valor del divisor del reloj (Clock = 96).

- *pwmWrite(pin int, valor int)*: Esta función escribe el valor PWM en el pin que se asigna en la primera variable.

4.4.2. Configuración motores

En cuanto al manejo de los motores, este se realiza gracias al empleo de la librería WiringPi, la cual provee la interfaz de control que permite asignarlos valores necesarios para sacar PWM para mover los motores, también hará que esta acción sea más sencilla y funcional.

4.4.3. PWM

PWM (Pulse-Width Modulation) es una técnica que se usa para modificar el ciclo de una señal por ejemplo cuadrada, esto sirve para sacar el ancho de la señal que se necesita transmitir, es una técnica usada para controlar la velocidad de los motores. Para ello se va

a usar dos controladores Hardware de PWM que tiene la RaspBerry Pi, con esto se evitará consumir recursos de la CPU y así ser mas eficiente. Con estos dos controladores se evita controlar los motores a través de software esto nos supondría una eficiencia menor, ya que se tendría que usar recursos de la CPU para controlarlo.

4.4.4. Configuración PWM

Una vez estén correctamente alimentados los motores, por la entrada Signal de estos se transmitirá un ancho de pulso con el cual se controla su funcionamiento, ya sea pararse, moverse hacia un lado o hacia el otro. Para conseguir esos anchos se necesita configurar unos valores en el *Clock* y *Range*.

El reloj Raspberry Pi PWM tiene una frecuencia de 19,2 MHz, la cual, dividida por el argumento de *Clock*, será la frecuencia la cual se transmite a través del pin, es una señal de onda cuadrada. El siguiente paso es configurar el comparador, es decir el *Range* especificado, este range es el que va contando ciclos de la frecuencia que se quedó y cada vez que llega a esa medida se resetea y se pone a 0, ahora el siguiente paso sería concretar qué zona del pulso estaría a alta y a baja, para eso se escribe en el pin correspondiente el valor necesario para comparar, cada vez que el pulso se resetea, este valor igual y mientras está contando, el valor está a 1 y cuando termina de contar el franco se queda a 0, con este método sacaremos un ancho de nivel alto correspondiente al necesario para realizar los movimientos.

Para configurar el PWM se ha de aplicar la siguiente fórmula

$$PWM(Hz) = (Frecuencia/clock)/range$$

Con los datos reales que se asignan al robot, son los siguientes:
Frecuencia del Bus: 19,2 MHz
Range: 4300

Clock: 96

$$192{,}000{,}000/4300 = 44{,}651$$

$$44{,}651/96 = 465Hz$$

Al realizar esas divisiones internamente dentro de la RaspBerry, se queda un valor en el bus de 465 Hz, con lo que, dependiendo del valor que se asigne, se sacará un ancho de pulso determinado en el pin para que el motor realice el movimiento deseado. En la Figura 4.5 se puede ver el ancho de pulso necesario para que el motor se encuentre parado, en la Figura 4.6 se ve para que el motor gire en un sentido y en la Figura 4.7 el correspondiente para que gire en el otro sentido.

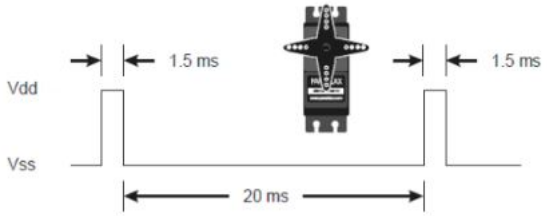


Figura 4.5: Diagrama del ancho de pulso necesario para que el motor se detenga

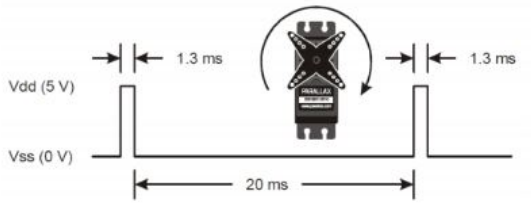


Figura 4.6: Diagrama del ancho de pulso necesario para que el motor gire en un sentido

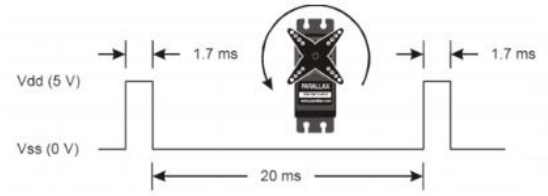


Figura 4.7: Diagrama del ancho de pulso necesario para que el motor gire en el otro sentido

4.5. Módulo de los motores

En este módulo se desarrollan las implementaciones necesarias para el control de los dos servomotores gracias a la librería descrita en la Sección 4.4.

Se tienen una serie de funciones para hacer distintos movimientos, para ello, en cada una se asignan los valores necesarios del PWM de forma que el robot pueda realizar todos los movimientos necesarios.

La mayoría de funciones corresponden con cada uno de los estados de la máquina de estados secundaria detallada anteriormente. De manera resumida se puede decir que cada uno de los estados de esta máquina son órdenes directas que se envían al módulo de los motores para realizar movimientos.

Además de esas funciones, este módulo contiene algunas otras, entre las que destacan:

- *initMotores()*: Esta función inicial es la encargada de establecer los parámetros de configuración para poder usar los servomotores. Aquí se configuran parámetros como el range y el clock de los motores. Es necesaria siempre al comienzo de cada ejecución.

- *girarFoto()*: Es la que asume la responsabilidad de realizar los giros para recorrer los sectores en la búsqueda inicial del punto más cercano. Realizará el giro de los 60º

aproximadamente y esperará un tiempo prudencial para la toma de las fotografías necesarias.

En la Tabla 4.1 se puede ver las distintas funciones que va a ejecutar el robot para las distintas situaciones con las que se va a topar. Para cada acción posible se muestran los distintos valores que se les asignan a los PWM de los motores para llevar a cabo dicha función.

Motor izquierdo (pin 18)	Acción	Motor derecho (pin 13)
319	parar	320
324	avanza	316
322	avanzaLento	318
312	retroceder	317
318	rotaIzq	317
322	rotaDer	322
321	trayectoriaIzq	317
322	trayectoriaDer	319
323	girarFoto	322

Cuadro 4.1: Tabla que muestra los valores de PWM de las funciones de los motores

Capítulo 5

Resultados experimentados

El robot, con todo el hardware empleado, con todos los módulos generados y unificadas todas las partes en un mismo sistema es capaz de realizar un comportamiento que similar al idea esperado en las especificaciones iniciales.

Para empezar, gracias a la cámara colocada en el soporte en la ranura correcta, el robot realizará giros de $60^{\underline{o}}$ (ángulo que se corresponde con el de la visión de la cámara) sobre sí mismo hasta un total de 6. En cada giro, realizará un fotografía del terreno y visualizará los objetivos, calculará la distancia hacia cada uno de ellos y se quedará con el más cercano. Por cada giro realizará estas acciones de tal forma que al completar el giro completo el punto que registra es que más cercano al robot.

Una vez decidido el punto que se va a visitar, se realizarán tantos giros de $60^{\underline{o}}$ como sectores (de este mismo ángulo) se hayan recorrido para encontrar dicho objetivo. De esta forma el robot se queda orientado hacia el sector el cual contiene el punto elegido.

Finalmente, se ejecutará el estado complejo en el cual se ha de ir a buscar el objetivo avistado. Este estado es la máquina de estados interna contenida dentro de este en la cual, el robot irá avanzando tratando de colocar el objetivo en el centro de su campo de visión y

en el momento en el que esté suficientemente cerca, se posicionará sobre el mismo.

Alcanzado el punto el robot volverá a empezar el algoritmo de funcionamiento a la búsqueda de más puntos que recorrer.

En la Figura 5.1 y la Figura 5.2 se aprecia el aspecto final del robot construido.

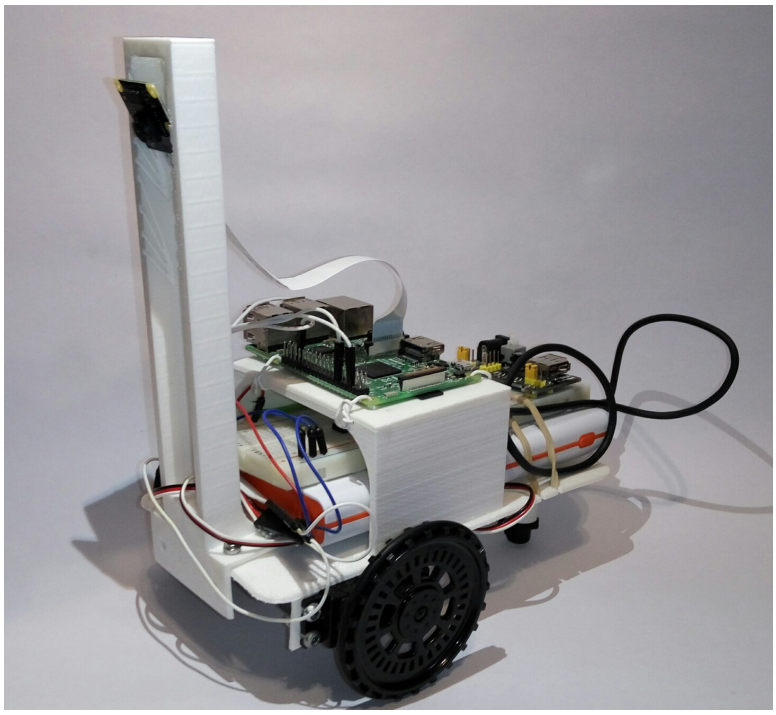


Figura 5.1: Vista frontal del robot finalizado

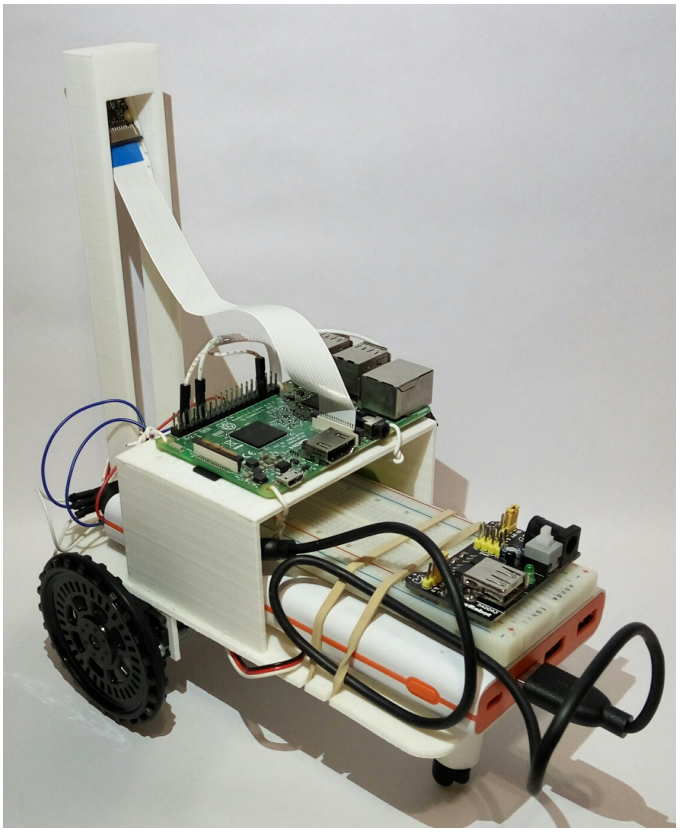


Figura 5.2: Vista posterior del robot finalizado

Capítulo 6

Propuestas de mejora

En este capítulo se expone el siguiente paso en la evolución del robot hacia un diseño con una mejor y mayor funcionalidad. Esta idea pese a que no ha podido llegarse a implementar debido a las limitaciones de los componentes hardware de los que se dispone, ha requerido un gran trabajo de investigación para su desarrollo y un considerable estudio para determinar la viabilidad y valoración de este nuevo planteamiento.

Por un lado se incorpora un nuevo hardware, son los llamados encoder, los cuales ofrecen una nueva información acerca del movimiento del robot sobre el terreno, la cual es vital para el desarrollo de esta mejora, y por otro, se expone cómo se amplían las competencias del robot desde el punto de vista del software del mismo.

6.1. Nuevo diseño

Mientras que en la versión estable desarrollada nuestro robot va de objetivo en objetivo de una manera un tanto ineficiente, en este nuevo plan, la idea es que se mejore tremendamente la eficacia en el recorrido de todos y cada uno de los objetivos, sin olvidar ninguno, en el menor tiempo posible y sin repetir.

Para conseguir todos estos avances la clave es mapear el terreno, de esta forma se podrán situar en el "mundo" que el robot es capaz de percibir todos y cada uno de los puntos, así como la situación actual del propio robot dentro de él.

Introducir este gran cambio conlleva rediseñar la máquina de estados antigua para que quede con una estructura como la que se muestra en la Figura 6.1.

Al principio el robot rotará sobre sí mismo en busca de todos los objetivos a visitar, tras esto los situará en el mapa y decidirá el orden en el que los recorrerá. Una vez obtenidos los datos irá escogiendo punto a punto, en orden, girará sobre sí mismo para orientarse hacia el punto a buscar y entonces irá hacia él. Hará esta última parte hasta que haya buscado todos los puntos, una vez lo haya realizado volverá a empezar el algoritmo.

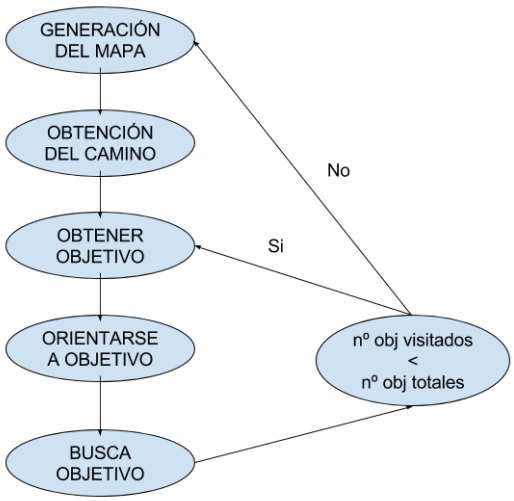


Figura 6.1: Máquina de estados del nuevo diseño

6.2. Función de los encoder

El nuevo elemento hardware va a ser el encargado de dar la posibilidad al robot de hacer un mapeo del terreno avistado y de situarse en él. Estos componentes no realizan directamente esta labor, sino que lo que hacen es proporcionar información real del movimiento del robot, por lo tanto, gracias a ellos se disminuye enormemente el margen de error que tendríamos de hacerlo únicamente con lo percibido por la cámara.

Los encoder, son en realidad dos sensores infrarrojos colocados cuidadosamente al lado de las ruedas que, correctamente conectados, ofrecen la posibilidad de distinguir, en una muy corta distancia, el blanco y el negro. Añadido a esto, a cada rueda se le colocaría codificador óptico transitivo con la alternancia de segmentos opacos y claros (o transparentes) como se muestra en la Figura 6.2, para que mientras la rueda gira, el sensor colocado perpendicularmente vaya percibiendo los cambios de división.

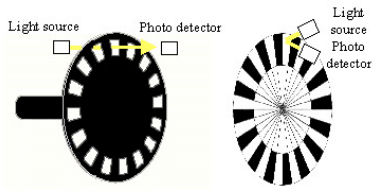


Figura 6.2: Codificador óptico transitivo de los encoder

La Figura 6.3 se aprecia cómo el detector de fotos ve las líneas claras y oscuras (o agujeros) indicando cuál es el voltaje resultante. Las tiras horizontales claras y oscuras representan las barras (o agujeros) en la tira del codificador rotatorio. Siempre que un área blanca está delante del detector, la tensión de salida cambia a alta (5 vdc). Siempre que un área oscura esté delante, la tensión es baja (0 vdc).

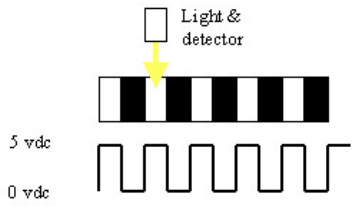


Figura 6.3: Voltajes ofrecidos por el encoder según la zona detectada

Con este dispositivo electromecánico que convierte la posición angular o el movimiento de un eje a una señal analógica o digital, se consigue que el robot sea capaz de percibir cuánto ha girado cada motor. Esto es debido a que se conoce el ángulo de cada división y el radio de la rueda se puede obtener el arco de la circunferencia de esta que corresponde a cada división, lo que sería cada paso perceptible y medible del robot. Para ello con una sencilla fórmula se obtendría dicha longitud:

$$L = \frac{2 * \Pi * \text{radio de la rueda} * \alpha}{360}, \text{ donde } \alpha \text{ es el ángulo de cada división}$$

Una vez obtenido este dato, gracias a estos elementos podemos sacar cuánto avanza cada rueda y con ello cuánto avanza el robot y hacia qué dirección queda orientado, información que nos será imprescindible para poder realizar las búsquedas de los objetivos.

6.3. Obtencíon de distancias con la cámara

Para controlar la superficie que el robot ve cuando realiza capturas con la cámara, medir las distancias y transformar la visión con perspectiva del robot en superficie plana, se utilizó una cuadrícula como plantilla que se puede apreciar en Figura 6.4.

Esta plantilla se dibujó a mano con recuadros de 5 centímetros de lado. Posteriormente se tomaron puntos como medidas para obtener las equivalencias entre los píxeles de la ima-

gen que ve el robot, y la distancia real que hay a dichos puntos.

De esta manera a partir de los píxeles que devuelve la cámara, tenemos la distancia en centímetros a dicho punto y distancias entre puntos, que se sirven para situarlo en el mapa global.

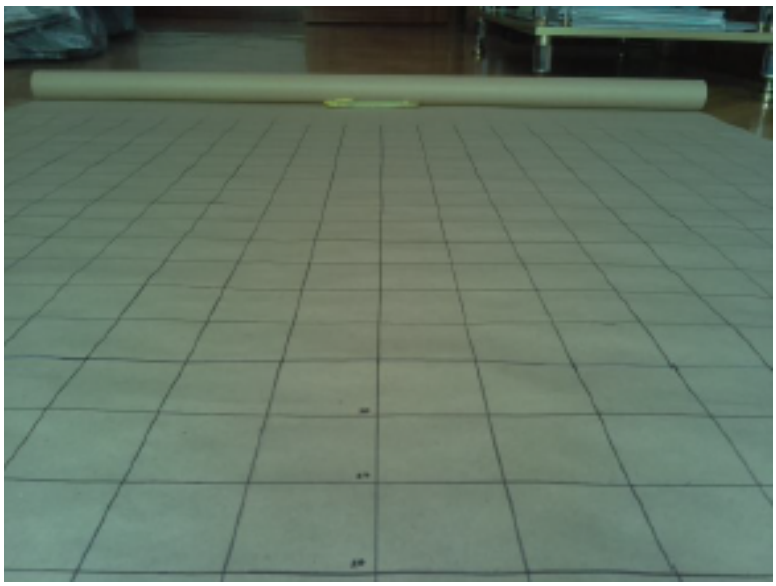


Figura 6.4: Foto de la cuadrícula

6.3.1. Estructura de datos

La gestión de los datos del apartado anterior se realiza de manera que toda una misma línea horizontal, a la misma distancia en centímetros, se identifica con el mismo valor en píxeles.

Para la gestión de toda la información del apartado anterior, se utiliza una estructura hash map (clave-valor), donde el valor en píxeles común es la clave, y todos los puntos tomados a lo largo de esa línea horizontal, cada 5 centímetros, son el conjunto de valores.

Además de ser la clave de la estructura, los valores de los ejes horizontales están también almacenados en un vector, debido a que los valores que obtendremos de la cámara difícilmente coincidirán con los valores tomados y es necesario encontrar el más cercano. Dicha estructura se esquematiza en Figura 6.5

Clave	Valor				
Valor en píxeles del eje horizontal	Conjunto de valores a 5 centímetros sobre el eje horizontal "clave"				
Por ejemplo, eje horizontal a 30cm del robot -> píxel 339	10 cm a la izquierda-> píxel 128	5 cm a la izquierda-> píxel 219	0cm en línea con el robot -> pixel 312	5 cm a la derecha -> píxel 405	10 cm a la derecha -> píxel 497

Figura 6.5: Esquema de la estructura del hashmap

6.4. Generación del mapa global

Para empezar, el robot realizará una vuelta sobre sí mismo en busca de todos los objetivos alcanzables desde su punto de partida, es decir, el mapa va a representar todo el terreno que el robot podrá percibir óptimamente con la cámara desde la posición inicial en todas las direcciones.

La cámara que se usa posee un alcance de unos 50 cm en el cual los colores los distingue a la perfección, por lo tanto, será esta distancia la que establecerá el radio de alcance del mapa, por lo que este tendrá unas dimensiones de 100x100 cm y la posición, en coordenadas cartesianas, de (50,50) como situación inicial, es decir, el mapa global estará situado en el

primer cuadrante (X e Y positivas) de un sistema de coordenadas cartesianas.

Por otro lado, se conoce el ángulo de visión de la cámara, que es de unos 60^{o}, por lo que se tiene que dar 6-7 giros para completar una vuelta completa sobre el eje central del robot sin dejar ninguna zona, dentro del radio de alcance de la cámara, sin observar.

Con todos estos datos se realizarán una serie de giros desde los cuales se tomará una fotografía y se devolverá una lista con todos los objetivos percibidos. Tras esto, habrá que traducir todos los puntos vistos, que están en píxeles, a centímetros, por lo que se sabrá a qué distancia real se encuentran del robot. Con los puntos situados correctamente respecto al punto de referencia, que es la visión del robot, ahora hay que situarlos en el mapa global y para ello hay que saber en qué posición se encuentra el robot en ese momento actual, es decir, su situación en el mapa y la dirección hacia la que está orientado.

Ahora, con estos cálculos realizados, hay que comprobar que el punto que acabamos de transformar no sea ya uno que estuviera previamente en el mapa y que se haya visto anteriormente (con un margen de fallo). En el caso de ser un nuevo objetivo, se procede a introducirlo en el mapa global que no es más que una cola con todos los puntos, cola que será tratada posteriormente, y a incrementar un contador con el número total de objetivos que vamos a buscar. Pero si ya se le había visto en un anterior escaneo del terreno, simplemente se le descarta.

Una vez se completen todos los giros, el robot se quedará en la misma posición que al inicio, situación que es la que establece el mapa, puesto que todo el mapa global es relativo a la posición inicial del robot.

6.5. Cálculo del camino

Una vez obtenidos todos los objetivos que componen el estado del terreno, se pasa a la obtención de un grafo valorado totalmente conectado, es decir, todos los vértices están conectados con todos, lo cual significa que desde cada punto es lógico que podamos llegar al resto moviéndonos por el terreno. Este cálculo se realiza mediante la fórmula del módulo de los vectores de dirección para dos puntos, uno A (x1, y1) y otro B (x2, y2):

$$\overline{AB} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Tras haber formado el grafo, en este estado lo que se hará será obtener la secuencia de puntos en el mapa que deberá ir siguiendo el robot para visitarlos todos recorriendo la menor distancia posible.

Para ello se emplearía un algoritmo recursivo, el cual, para un nodo, recorrería todos sus adyacentes, por cada uno de ellos devolvería la distancia mínima de recorrer el resto del grafo no visitado empezando por él (el actual/padre le sumaría el coste de ir por ese camino) y una pila con el orden de los nodos a visitar.

En el caso de que el nodo sea hoja (este caso se da cuando sus adyacentes ya se han visitado en el recorrido hasta el actual) se devolvería un coste de 0 al cual el padre le sumaría el coste de alcanzarlo y se introduciría en la pila vacía.

De no ser hoja, se recorre también sus adyacentes y de todos ellos el algoritmo se queda con aquel camino cuyo coste de recorrerlo sea mínimo, entonces lo sumamos al valor de la arista que une ese camino con el nodo actual, o lo que es lo mismo, sumar a la distancia total de elegir recorrer primero ese punto la distancia de alcanzarlo. Además, el nodo actual se añade a la cima de la pila proporcionada por el punto que hemos elegido visitar, el cual

ya estaba añadido.

Con la obtención de estos 2 datos se termina la llamada recursiva y así, se habrá conseguido proporcionar una pila de nodos de tal forma que con ir sacando el nodo que está en la cima de la se irán visitando los puntos en el mapa en el orden en el cual la distancia total de visitarlos es mínima.

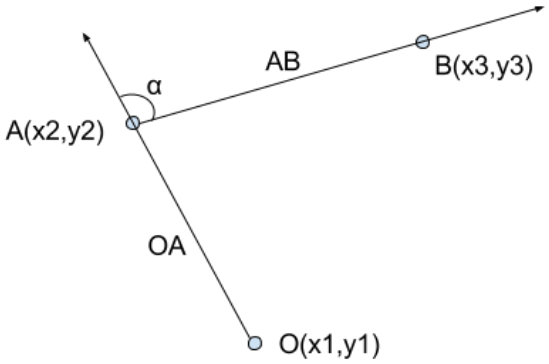
6.6. Orientación del robot hacia el objetivo

La finalidad de este estado es la de que el robot, tras saber cuál es el siguiente objetivo, gire para quedarse orientado de frente a él, lo que facilitará mucho su búsqueda y lo hará de una forma más rápida, por lo tanto se ha de calcular la dirección hacia la que tiene que hacerlo (derecha o izquierda) y el ángulo que debe de recorrer.

El razonamiento para realizar este cálculo se basa en la obtención de los vectores de dirección de 3 puntos, estos son:

- El punto origen O(x1,y1) que es el que se ha visitado antes de que se encuentre el robot situado en el punto actual.
- El punto actual A(x2,y2) en el que se encuentra el robot.
- El punto destino B(x3,y3) que es el siguiente a visitar desde el punto actual donde se encuentra.

Dicho de otro modo, el robot, en el instante en el que alcance un objetivo, tendrá un punto O que era de donde venía (el anterior visitado), el punto A que es el que acaba de encontrar y el punto B que es el siguiente al que se va a dirigir.
La esquematización de la idea se puede apreciar en Figura 6.6.



H

Figura 6.6: Esquema de los vectores de dirección

Gracias a estos 3 puntos se obtienen 2 vectores de dirección, el OA y el AB de la siguiente forma:

$$\overrightarrow{OA} = (x_2 - x_1, y_2 - y_1) = (a_1, a_2)$$

$$\overrightarrow{AB} = (x_3 - x_2, y_3 - y_2) = (b_1, b_2)$$

De esta manera se tiene un vector cuya dirección apunta hacia A viniendo de O, así se sabe la dirección hacia la que teóricamente ha quedado orientado el robot, y con el otro vector está la dirección hacia la cual se debe quedar orientados para tener de frente el siguiente punto a visitar.

Una vez obtenidos estos datos se aplica la siguiente fórmula matemática para obtener el ángulo:

$$\alpha = arccos(\frac{a_1 * b_1 + a_2 * b_2}{\sqrt{a_1^2 + a_2^2} * \sqrt{b_1^2 + b_2^2}})$$

Ahora se ha de averiguar hacia dónde ha de girar el robot y para ello hemos de obtener la dirección absoluta hacia la que apunta el robot, es decir, respecto hacia el mapa global cuál es la orientación. A la vista de la organización del mapa y del conjunto de las posibles

direcciones hacia las cuales se puede quedar orientado el robot, el mapa global quedaría tal como se vería en la Figura 6.7:

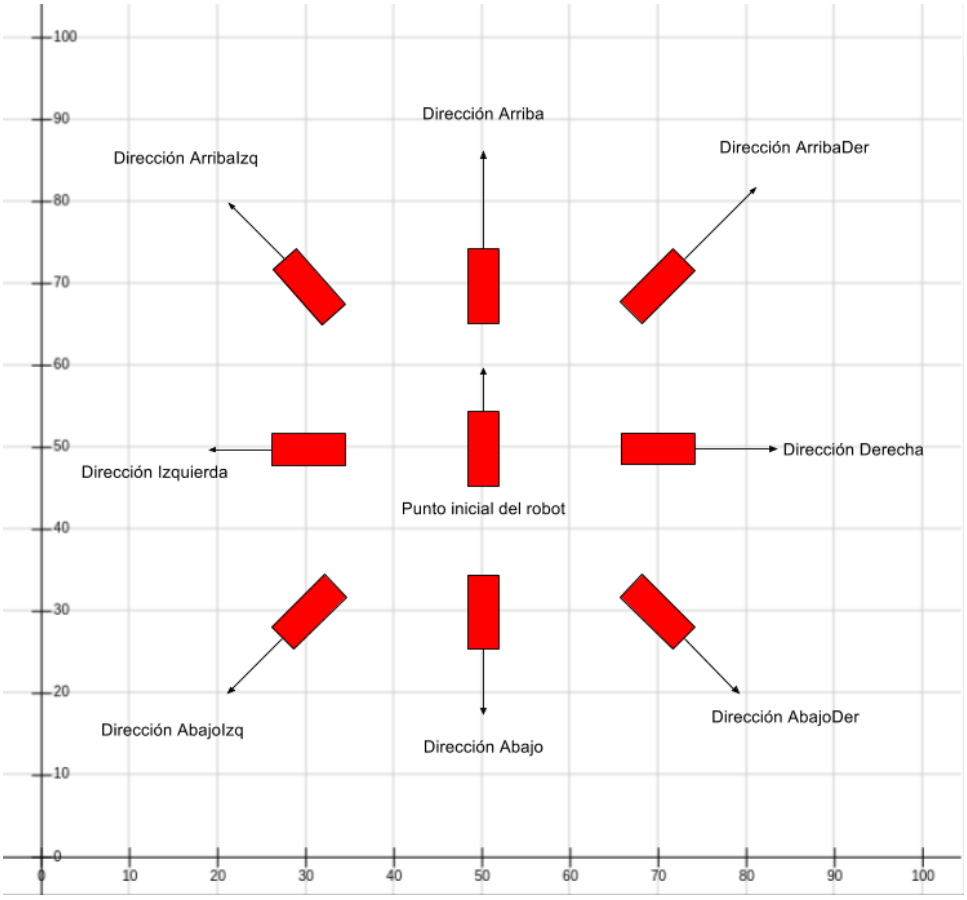


Figura 6.7: Esquema del mapa global y las direcciones hacia las que se orienta el robot

Se entiende por dirección absoluta aquella que referencia a una orientación dentro del mapa global del robot, aquellas que se muestran en la Figura 6.7. Para obtener la del punto actual en el que se encuentra el robot, basta con observar los signos de las componentes del vector de dirección OA, que es en el que se encuentra el robot en un punto determinado:

- Si la componente X es 0 significa que la dirección es *Arriba* en el caso de que las Y sean positivas, o *Abajo* en el caso de que las Y sean negativas, por otro lado.

- Si son las Y las que valen 0, la dirección es *Derecha* en el caso de X positivas o *Izquierda* en caso contrario.

- Si no se dan ninguno de esos casos, significa que estamos en una de las cuatro diagonales, las cuales se calculan igualmente mirando los signos del vector:

  - Con X e Y negativas tenemos *Abajo Izquierda*.
  - Con las X negativas e Y positivas será *Arriba Izquierda*.
  - Para las X e Y positivas, *Arriba Derecha* será la dirección.
  - Y por último, con las X positivas e Y negativas obtendremos la dirección *Abajo Derecha*.

Una vez obtenidas las direcciones, para aquellas que son paralelas a algún eje de coordenadas la obtención de la dirección de giro es muy simple:

- Cuando la dirección es *Arriba*, si la componente x de B es mayor que la componente x de A entonces el giro es hacia la derecha, sino hacia la izquierda.

- Si la dirección es *Abajo* entonces si la componente x de B es mayor que la componente x de A entonces el giro es hacia la izquierda, sino hacia la derecha.

- Para la dirección *Derecha*, si la componente y de B es mayor que la componente y de A entonces el giro es hacia la izquierda, sino hacia la derecha.

- Y por último, cuando la dirección es *Derecha*, si la componente y de B es mayor que la componente y de A entonces el giro es hacia la derecha, sino hacia la izquierda.

Pero para las diagonales el método difiere enormemente, se ha de obtener por un lado la ecuación de la recta que pasa por los puntos O y A, y por otro lado la de la recta paralela

(de igual pendiente) que pasa por el punto B. Para conseguir estas ecuaciones hemos de calcular m, que es común para ambas, y b para cada recta:

La ecuación de la recta explícita: $y = m * x + b$

siendo m la pendiente y b el punto de interceptión en la ordenada

La pendiente se obtiene de: $m = \frac{y_2 - y_1}{x_2 - x_1}$

La b de la recta OA: $b = y_1 - m * x_1$

La b de la recta paralela que pasa por B: $b = y_3 - m * x_3$

Después, simplemente se comparan los puntos de intercepción en la ordenada según la dirección hacia la que esté orientado el robot, se la siguiente forma:

- Cuando la dirección es *Arriba Izquierda* o *Abajo Izquierda*, entonces si la b de la recta paralela es mayor que la que pasa por el punto actual la dirección de giro es hacia la derecha, sino hacia la izquierda.
- Por el contrario, cuando la dirección es *Arriba Derecha* o *Abajo Derecha*, si la b de la recta paralela es mayor que la que pasa por el punto actual la dirección de giro es hacia la izquierda, sino hacia la derecha.

Obtenidos la dirección y el ángulo giro, se devolverá el ángulo positivo si la dirección a girar es la derecha o negativo si es hacia la izquierda. Así, se les puede indicar a los motores cuánto y hacia donde han de girar y los encoders se encargarán de verificar que se ha rotado esa distancia.

Hay que distinguir el caso en que la situación sea la inicial y el punto a visitar sea el primero del resto de situaciones. Esta diferencia se debe a que en la primera búsqueda del

objetivo el robot se encuentra en el punto de partida, es decir, no tiene punto origen para hacer el cálculo del ángulo y sentido de giro. Para solucionar esto, como se sabe la posición en la que se ha quedado el robot, que es la inicial, se simula un punto que se encuentre en la recta paralela al eje de las Y y que pasa por el punto 50 de las X y que esté situado tras el robot, el (50,40) por ejemplo, de esta forma ya tenemos punto origen para calcular el vector de dirección inicial y así obtener el ángulo y sentido de giro.

Debido a que el robot cuando llegue a un punto no va a tener la dirección teórica (no se va a mover nunca recto al 100 % en la búsqueda de un objetivo), por lo que se ha de saber en qué dirección se ha quedado y qué margen de error tiene respecto a la ideal, dato con el cual se calculará el ángulo real de giro que ha de realizar el robot para orientarse hacia el siguiente punto.

6.7. Búsqueda del objetivo

Este estado está compuesto de una máquina de estados integrada dentro de él, la cual tiene una función muy similar a la de la versión funcional desarrollada pero con una serie de diferencias debido al nuevo sistema de mapeo del terreno.

La misión de este punto es la de que una vez el robot se haya quedado orientado hacia el objetivo que ha de buscar ahora simplemente vaya hasta él. A diferencia de como se hacía antes, ahora además hay que llevar una serie de hilos paralelos a la ejecución de esta máquina de estados, en ellos, cada encoder irá recogiendo información acerca del movimiento de cada rueda, lo que va avanzando cada rueda, y con estos datos se calculará cuánto se desplaza el robot en el mapa y cual es la dirección hacia la que está orientado en todo momento.

Esta información es de vital importancia debido a que en cualquier momento, el robot puede percibir más de un objeto y ha de ser capaz de distinguir cual es el objetivo que está persiguiendo. Gracias al conocimiento de su propia posición y orientación dentro del mapa global, al detectar los puntos desde la cámara será capaz de mapearlos de nuevo, por lo que el punto que esté más cerca (porque existe un margen de error) del punto que tiene guardado como objetivo actual será el actual, así descartará el resto y solo tomará en consideración las coordenadas del punto que considera que se corresponde con el objetivo.

En el caso de que el robot, por un fallo de la cámara, no detecte el punto o detecte otros pero no el actual, volverá a mapear los puntos en el "mundo" y se verá que ninguno se corresponde con el que estaba persiguiendo (no están lo bastante cerca, es decir, el margen de error es muy grande) entonces seguirá en la misma dirección en la que se estaba moviendo la última vez que detectó el punto. Si pasado un cierto tiempo sigue sin detectarlo, el robot debería de detener la normal ejecución puesto que debería de considerar aquello un fallo grave, por lo que tiene dos opciones: esperar a detectar el punto de nuevo o (y esta es la más óptima) dar por perdido ese objetivo y pasar al siguiente.

Capítulo 7

Conclusiones

En este capítulo se exponen todos aquellos problemas que se han ido presentado durante la realización del proyecto así como las soluciones o alternativas adoptadas para sacar adelante el robot. Por otra parte, se relatan los conocimientos que se han adquirido durante el transcurso del proyecto y cuales aprendidos durante la carrera han sido de utilidad para llevarlo a cabo. Por último, se comentará cual ha sido el resultado obtenido desde la idea inicial hasta el proyecto actual con todos aquellos imprevistos e indagaciones hechas durante el desarrollo del mismo.

7.1. Trabajo realizado

La idea de cual sería el resultado final del proyecto ha ido variando, siendo los causantes el tiempo disponible, el hardware empleado, los problemas encontrados y la capacidad para superarlos.

La finalidad del robot en ningún momento fue la de desarrollar un proyecto igual que el modelo de MathWorks, ni tampoco la de que fuera un sistema muy preciso (no se contaba con el hardware necesario), sino la de crear un prototipo y aprender acerca del desarrollo de un proyecto con parte física y parte software, que estuviese debidamente modularizado

y que realizara una tarea sencilla pero efectiva.

7.2. Problemas y soluciones

Pese a que el planteamiento propuesto en el Capítulo 6 haría que el funcionamiento del robot alcanzara una categoría de una calidad superior, se han presentado una serie de problemas que no ha sido posible salvar, principalmente debido a los componentes hardware y en concreto con la cámara empleada. Estos obstáculos no solo han influido en la implementación de la versión mejorada, el desarrollo de cualquier tipo de avance en lo que a la cámara y todo lo relacionado con ella se refiere, se han visto afectados y muy limitados.

La cámara proporcionada no posee una gran calidad, lo que hace que ante distintas condiciones lumínicas los datos percibidos varíen enormemente aún haciendo todos los filtrados disponibles para que esto no suceda. Además, se ve afectado el movimiento del robot, esto se debe a que por un lado, la cámara posee un gran retardo entre las distintas fotos que puede ofrecer, con lo cual la velocidad del robot ha de ser muy reducida para que las distancias recorridas entre foto y foto sean mínimas. Reciprocamente, las imágenes tomadas pueden aparecer corruptas por la cantidad de ruido introducido por la misma a consecuencia del movimiento del robot, lo que hace que no siempre sean fiables lo datos que se proporcionan.

Con esto se tiene que el alcance de visión de la cámara es muy corto, las condiciones en que pueda funcionar muy concretas, el margen de error de la cámara muy grande y sus percepciones no siempre fiables.

Además de la cámara, el sistema de los motores con los encoder no es muy preciso, no es la mejor forma de realizar las mediciones para el seguimiento del robot, pero sí la más sencilla.

Entonces, por todos los problemas encontrados se ha visto que el margen de error que poseería el robot era tremendamente grande y añadido al poco tiempo del que se disponía para la implementación del diseño mejorado, esta idea no ha podido llevarse a cabo en el robot real, únicamente se ha podido exponer de una forma teórica.

7.3. Conocimientos adquiridos y usados

Para la realización de este proyecto se han tenido que adquirir una serie de competencias tales como el montaje y control de motores tipo servo, de procesamiento de imagen, de uso de librerías para el manejo de dispositivos y, por razones de la propia naturaleza del proyecto, de testeo de sistemas físicos en el mundo, es decir, depuración del funcionamiento del robot en un entorno real con problemas reales (iluminación, velocidad, etc).

En cuanto a los conocimientos aplicados que previamente fueron adquiridos en las distintas materias de la carrera, destacan:

- La programación en lenguaje C++ obtenida de asignaturas como Fundamentos de la Programación (FP) y Tecnologías de la Programación (TP).
- Las competencias matemáticas aplicadas en la parte de investigación para la orientación del robot, es decir, todas las fórmulas empleadas se obtuvieron de la asignatura de Métodos Matemáticos de la Ingeniería (MMI).
- El manejo de estructuras de datos y la algoritmia necesaria para la resolución de las distintas situaciones del robot fueron posibles gracias a asignaturas como Estructuras de Datos y Algoritmos (EDA) y Diseño de Algoritmos (DA).
- Tanto en la versión funcional como en la mejorada, se aplican de conocimientos de los sistemas operativos con base Linux, que es el sistema que maneja la Raspberry Pi,

aprendidos en Sistemas Operativos (SO), Ampliación de Sistemas Operativos (ASO), Redes y Ampliación de Redes (AR).

7.4. Resultado obtenido

En un principio, el resultado final que se esperaba era el de expuesto en el Capítulo 6, sólo la funcionalidad, no la forma de implementarla. Tras esto, se trató de finalizar el proyecto en junio con la versión estable y funcional que se presenta actualmente. Se decidió no entregar en junio sino en septiembre tratando de llegar con el proyecto a la idea original, es decir, que buscara todos los puntos de su mundo, sin repetir y de una manera óptima.

Finalmente se decidió llegar a un punto intermedio en el cual se desarrollara una versión del robot que funcionara buscando puntos sin ningún tipo de optimización y además, que se realizara una exhaustiva investigación acerca de la mejora del robot pero sin llegar a ponerla en práctica, lo que haría que la meta establecida fuera asequible al tiempo disponible y que el resultado obtenido fuera satisfactorio.

Bibliografía

[1] Concurso Mission on Mars Robot Challenge.
https://es.mathworks.com/academia/student-challenge/mission-on-mars.html.

[2] Raspberry Pi - Página principal del proyecto.
https://www.raspberrypi.org/.

[3] OpenCV - Documentación.
http://opencv.org/.

[4] WiringPi - Documentación.
http://wiringpi.com/.

[5] Raspberry Pi 3 Modelo B - Documentación.
https://www.raspberrypi.org/documentation/.

[6] IMX219PQ - Datasheet.
http://www.sony-semicon.co.jp/products_en/IS/sensor1/img/products/ProductBrief_IMX219PQ_20160425.pdf.

[7] Módulo de la cámara de la Raspberry Pi.
https://www.raspberrypi.org/documentation/hardware/camera/README.md.

[8] #900-00008 - Datasheet.
https://www.parallax.com/sites/default/files/downloads/900-00008-Continuous-Rotation-Servo-Documentation-v2.2.pdf.

[9] OpenSCAD - Página del programa.
http://www.openscad.org/.

Apéndice A

Introduction

A.1. MarsRover idea

The original idea for this final degree work, was taken of the "Mission on Mars Robot Challenge" [1] competition organized by Mathworks. Rover robots have a mission to accomplish, satisfying 2 objectives: to explore Mars planet identifying certain elements and do that avoiding obstacles in the field.

Participants have to improve the given robot, similar as the robot in figure 1.1, to be able to visit several spots travelling the shortest distance as possible and in a minimun time. That will be done optimizing the MATLAB robot and the Simulink algorithms. The model of the robot, created using a 3D printer, the raspberry pi [2], the Arduino and MATLAB and Simulink software are provided to the participants by the Mathworks organization.

A.2. Objective of the project

The project objective is to make a robot using open source tools, in a common and extended language, using accessible hardware.

This robot has to comply with MathWorks competition specifications and make the defined task, without dependencies with the Matlab tool, and its hardware.

On the other hand, it's pretended to learn which hardware components use, which kind of software language use, and what estructures are the best to organize the robot, in addition to how to develop the tests with physical factors to solve.

The scenario where it's is going to be executed, is a field where the robot is surrounded with spots of a determined color. In this situation, with a camera, it will be able to perceive the spots and when all of them are discovered it will go to visit the spots, i.e. go to their position and stay above them. The objective should be realized traveling less distance as possible, without repetitions, to accomlpish the task in a minimun time.

A.3. Robot specifications

To implemet the robot the functioning must be taken into account, to choose correctly the components and diagrams to use.

Design, must have a camera to perceive the objectives. I has to offer enough quality due it is the only input source for the robot.

As the function that the system is going to realize is to generate a determined movement, is necessary an actuator to convert the logical information to a physical movement. That component will be the motors.

About robot's body, it's necessary to have a device able to manage the input data, offered by the camera, treat them, and act accordingly, sending to the motors the correct

orders according to the external information. This central component has to be able to run a program in a language with the sufficient power, adaptability and modularization capacity of the software to handle this vital task. On the one hand, it must be able to collect the data that the camera offers and on the other hand, it must also be able to adapt and generate the type of information that the motors are able to interpret to develop their function.

With all this, the robot is ready to go. The expected behaviour is to register all the accessible objectives from the initial position at first, and then look for the best way to run through them and visit them.

A.4. Methodology and work plan

Although at first the organization was realized in a decentralized way, then was needed to structure and organize the project, distributing tasks and objectives.

During the initial phase a strict distribution of the tasks was not done. It was pretended to all the members of the group were updated in all the tasks until the project have an initial shape. After that, a clearer distribution was done.

Starting with the camera and due to its great importance, a member of the group was almost exclusively in charge of everything related to this subject, from the investigation on its correct and optimal functioning, going through all kinds of filters, to the development of the software module that would manage the work of this component.

The other three team mates, beginning from a basic module for the control of the camera, were focused in testing to obtain a stable functioning of the robot. These tests were of vital importance since it was imperative that the states machine that was looking for the

objectives had optimal behavior, going as directly as possible towards the detected spot.

In the final phase of the project, a new division of tasks was made again in which another member of the group of three doing the robot tests, was dedicated to organize the memory, distribute the content to expose and establishing the guidelines for doing so. This same member, in addition to managing the memory, was responsible for researching the next step in the robot's evolution, only the idea of better performance. With all this, each member, when writing the documentation, was clear about which part to document since it was the work he had done mainly during the process of creating the project.

A.5. Structure of the document

The memory, divided in chapters, is organized and structured as follows:

- In Chapter 2 robot hardware aspects are included, like chassis design, and used camera and motors.
- Chapter 3 explains image processing done after take a capture with the camera, the use of the library OpenCV and the development of the software of the camera.
- In Chapter 4 it is exposed the control, using WiringPi library, and the development of the code in charge to manage the motors.
- In Chapter 5 it is treated the main code of the robot and the states machine.
- Is in Chapter 6 where the ideas to improve the robot will be expanded, from the new hardware implementation to new software explanation and step by step functioning.

- In Chapter 7 it is discussed the reached conclusions after the development of the robot, if the result is the same as expected and found issues as well as applied solutions. In addition, acquired knowledge are commented.

Apéndice B

Conclusions

In this chapter, all the problems appeared during the project workflow are exposed as well as the choosen solutions or alternatives. On the other side, acquired knwoledgements during the project are reported and also the ones learned during the degree which have been useful here. Finally, results will be commented from the initial idea to the actual project, with all the incidentals and inquires done during the project development.

B.1. Work done

The idea of the final result of the project has changed during the development, due to the available time, used hardware, encountered issues and the capacity to overcome that problems.

Purpose of the robot never was to develop a similar project to Mathworks model, nor was it a very precise system (necessary hardware was not available), rather to create a prototype and learn about the development of a project with a physical part and a software part, properly modularized, to perform a basic and efective task.

B.2. Problems and solutions

Although the proposed approach in chapter 6 would do that the robot performance reach a superior quality category, problems not possible to resolve were presented, mainly due to hardware components and specifically with the used camera. These obstacles not only have influenced in the implementation of the improved version, the development of any kind of enhancement as far as the camera is concerned, was affected and very limited.

The given camera doesn't have a great quality, which makes that in different light conditions, the received data change enormously, even doing all the available filtered. In addition, robot movement is affected, due to the retards of the pictures offered by the camera, whereby robot speed will be reduced in order to minimize the covered distances between pictures. Also, taken images could be corrupted by the introduced noise consequence of the robot movement, that makes the data not always reliable.

The situation is that the range of vision is very short, conditions to work fine are concrete, error scope of the camera is very high and its perceptions are not reliable.

Appart from the camera, the system with motors and encoders is not very precise, is not the best way to realize measurements for robot tracking, but is the easiest way.

Then, due all the found issues it's seen that the range of error of the robot is very big, and with the short availabe time to implement the improved design, it was not possible to carry out this idea in the final robot, it only has been exposed theoretically.

B.3. Knowledge used and acquired

For this project many knowledgements had to be acquired such as the assembly and control of servomotors, image processing, using of libraries for the management of devices and, for reasons of the very nature of the project, testing of physical systems, i.e. debugging of the robot operation in a real environment with real problems (lighting, speed, etc).

About applied knowledgements previously acquired in the different subjects of the degree, stand out:

- C++ programming obtained from subjects as "Fundamentos de la Programación (FP)" and "Tecnologías de la Programación (TP)".
- Mathematical skills applied to the orientation of the robot, i.e. all the used formulas was obtained from "Métodos Matemáticos de la Ingeniería (MMI)".
- Data structures and necessary algorithms to resolve the different situations of the robot were taken from subjects as "Estructuras de Datos y Algoritmos (EDA)" y "Diseño de Algoritmos (DA)".
- Both in the improved version and functional version, knowledgements from Linux operative system are applied, because is the system used in Raspberry Pi, and were learned from "Sistemas Operativos (SO)", "Ampliación de Sistemas Operativos (ASO)", "Redes" y "Ampliación de Redes (AR)".

B.4. Obtained result

The idea of the final result of the project has changed during the developments, due to the available time, encountered issues and the capacity to overcome that problems.

At first, the expected result was the exposed in chapter Capítulo 6, only functionality, not the implementation. After that, it was tried to finish the project un June with a funtional and stable version which is being presented now. It was decided to present the project in September instead of June trying to complete the original idea, i.e. a robot which finds all the spots in the field, without repeating and in an optimal way.

Finally it was decided to implement a middle point, developing a version to find the spots without any optimization but including an exhaustive investigation about how to improve the robot without putting it into practice, to make the final goal accessible to the available time with a satisfactory final result.

Apéndice C

Aportación individual de cada integrante del proyecto

C.1. Alejandro Ibarra

Las aportaciones hechas por Alejandro se han enfocado sobre todo en el apartado del procesamiento de imagen. Dicho apartado resultaba fundamental para el funcionamiento del robot, ya que, de ello dependería la detección de los objetos.

Esto implicaba un periodo de investigación, puesto que, en ningún momento de la carrera se nos había formado en materia de visión artificial. También era necesario una etapa de análisis de las necesidades, para determinar las acciones que llevaría a cabo el robot en este apartado.

En cuanto a la fase de diseño del algoritmo principal, también ha participado en el desarrollo de éste, haciendo un análisis del supuesto comportamiento que debía tener el robot y aportando ideas para trasladar este análisis a una solución viable. Esto se traduciría a la colaboración con otros miembros del grupo para la creación de la máquina de estados.

Por último, ha colaborado en el desarrollo de las las propuestas de mejoras que se le

podrían añadir al robot. Esto incluía un cambio en el planteamiento del algoritmo actual y la adición de nuevos sensores que permitirían dotar de mayor información. En este sentido, ha trabajado en la investigación y documentación de los encoders.

C.2. Antonio Blasco

Antonio se encargó, al comienzo del proyecto, de los primeros movimientos de los motores. Esta tarea requería investigar los valores a escribir en los registros, así como la calibración de éstos para ejecutar movimientos con la mayor precisión posible. Una vez realizada la primera toma de contacto, también participó en el diseño del módulo de los motores.

En cuánto al apartado de investigación, la cuadrícula para el mapeo, así como la toma de valores de posiciones del robot y el esquema del módulo de navegación fueron realizados por Antonio.

También participó en el diseño de la máquina de estados de la versión estable y en la implementación del estado inicial, donde el robot reconoce su entorno en busca de puntos.

Más adelante, realizó tareas de adaptación y pruebas de la versión estable del robot, para asegurar que dicha versión funcionaba cumpliendo con lo requerido, teniendo que hacer adaptaciones cuando lo que ocurría en las pruebas físicas no era lo esperado y planeado en las implementaciones. Por último, escribió los apartados de la memoria correspondientes a las anteriores aportaciones.

C.3. Cristian Vázquez

Por su parte, Cristian en un primer momento fue el encargado de realizar el diseño del chasis del robot en una impresora 3D, teniendo en cuenta las dimensiones de todos los componentes así como la mejor distribución de los mismos para que la forma del robot fuera la más favorecedora de cara al comportamiento que iba a llevar a cabo.

Más adelante, desarrolló la primera versión de la máquina de estados que buscaba un punto avistado. Esta versión, simplemente era capaz de que si el punto estaba dentro del campo de visión del robot, este consiguiera avanzar orientándose hacia el objetivo con una precisión muy aceptable, y culminaba con la acción de situarse sobre el objetivo para considerarse el punto como visitado. El conjunto de valores del módulo de los motores, así como los estados establecidos, se han mantenido, en una gran parte, en la versión estable presentada.

Añadido a esto, planteó el diseño mejorado del robot y la investigación para un funcionamiento más eficiente del mismo, aunque luego no pudiera llevarse a cabo. Llevó a cabo la invención de la idea que correspondería con el paso siguiente en la evolución de robot, desde el uso que se le daría al nuevo hardware a la nueva máquina de estados que controla el funcionamiento, pasando por los cálculos matemáticos y algorítmicos (siempre teóricos) de la obtención del camino y la orientación hacia los objetivos.

Por último, ha sido el encargado de la gestión y organización del proyecto, sobre todo en las fases finales, y de la memoria en el sistema de composición de textos LaTeX, estableciendo la organización, revisión y estructuración de la documentación.

C.4. Julio Álvarez

Julio al comienzo del proyecto se encargó de la instalación del sistema operativo en la RaspBerry, en este caso se instaló Raspbian.

A continuación empezó con la instalación de las librerías necesarias, en este caso con la instalación de estas dos librerías openCV y WiringPi. Una vez que se instaló openCV, empezó con las pruebas de las cámaras y con el inicio del filtrado de imagen, este parte era lo básico para poder hacer pruebas y ver que funcionaba e ir en paralelo con los motores.

Una vez que se instalaron la librería WiringPi, empezó con las primeras pruebas e investigaciones de el funcionamiento de los motores, lo primero que realizó fue una investigación para saber qué valores asignar a Clock y Range, una vez que se consiguió que se movieran, empezó a desarrollar el modulo de motor y su funciones básicas, después de tener los dos módulos con unas funciones básicas, empezó con un algoritmo para poder ir a un punto el cual luego pasaron a una máquina de estados para tener más precisión.

Para terminar el robot, desarrolló el algoritmo del módulo central, el cual es el encargado de dar una vuelta sobre sí mismo, e ir guardando el punto más cercano, al cual ir primero, después repetir este proceso para así recorrer varios puntos. Realizó las pruebas necesarias para confirmar que funciona.

Por último, se encargó de parte del desarrollo de la memoria.