

Trabajo fin de Grado

Grado en Ingeniería Electrónica, Robótica y Mecatrónica

Navegación autónoma basada en reconocimiento de señales de tráfico

Autor: Francisco Márquez Chaves

Tutor: Federico Cuesta Rojo

Dep. Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2016



Trabajo Fin de Grado
Grado en Ingeniería Electrónica, Robótica y Mecatrónica

Navegación autónoma basada en reconocimiento de señales de tráfico

Autor:

Francisco Márquez Chaves

Tutor:

Federico Cuesta Rojo

Profesor titular

Dep. de Ingeniería de Sistemas y Automática

Escuela Técnica Superior de Ingeniería

Universidad de Sevilla

Sevilla, 2016

Trabajo Fin de Grado: Navegación autónoma basada en reconocimiento de señales de tráfico

Autor: Francisco Márquez Chaves

Tutor: Federico Cuesta Tojo

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

El Secretario del Tribunal:

Fecha:

Resumen

El presente proyecto expone el diseño y desarrollo de un robot móvil cuyo objetivo es poder seguir ciertas señales viales que obtendrá a través de una cámara de vídeo. La meta del trabajo es hacer un uso extendido de técnicas de visión por computador, electrónica y control. El proyecto puede servir como guía para nuevos proyectos de automatización de vehículos móviles.

Abstract

This project presents the design and development of a mobile robot whose goal is to follow certain road signs to get through a video camera. The goal of the work is to make widespread use of computer vision techniques, electronics and control. The project can serve as a guide for new automation projects moving vehicles.

Índice

Resumen	7
Abstract	9
Índice	10
1 Introducción	1
2 DESCRIPCIÓN GENERAL	3
2.1. <i>Esquema general.</i>	3
2.1.1 Automatización robot móvil	3
2.1.2 Procesamiento de imágenes	5
2.1.3 Envío y recepción de información	6
2.2. <i>Conclusiones</i>	7
3 AUTOMATIZACIÓN ROBOT MÓVIL	9
3.1 <i>Esquema general</i>	12
	12
3.2 <i>Componentes y montaje</i>	13
3.3 <i>Implementación de módulos</i>	16
3.4 <i>Obtención del controlador</i>	20
3.5 <i>Conclusiones</i>	27
4 PROCESAMIENTO DE IMÁGENES	29
4.1 <i>Esquema general</i>	30
4.2 <i>Raspberry Pi</i>	30
4.3 <i>Tratamiento de imágenes tomadas por la cámara</i>	31
4.3.1 Conversión de color	31
4.3.2 Desenfoque	32
4.3.3 Detección de contornos (CANNY CON BLUR Y SIN BLUR)	33
4.4 <i>Tratamiento de imágenes de muestra</i>	35
4.4.1 Conversión de color	35
4.4.2 Desenfoque	36
4.4.3 Detección de contornos	37
4.5 <i>Comparación de imágenes</i>	38
4.5.1 Error Cuadrático Medio	38
4.5.2 Índice de similitud estructural (ssim)	38
4.6 <i>Conclusiones</i>	40
5 ENVÍO Y RECEPCIÓN DE INFORMACIÓN	41
5.1 <i>Esquema general</i>	41
5.2 <i>Conexión inalámbrica</i>	41
5.2.1 Introducción	42
5.2.2 Módulo NRF24I01	42
5.2.3 SPI	43
5.3 <i>Implementación en Raspberry Pi</i>	44
5.3.1 Montaje	44
5.3.2 Instalación de la librería	45
5.3.3 Código	45

5.4	<i>Implementación en Arduino</i>	47
5.4.1	Montaje	47
5.4.2	Instalación de la librería	48
5.4.3	Código	48
5.5	<i>Experimento</i>	49
5.5.1	Comprobación de conexiones y librería.	49
5.6	<i>Conclusiones</i>	50
6	NAVEGACIÓN AUTÓNOMA	51
6.1	<i>Explicación del experimento.</i>	53
6.1.1	Reconocimiento de obstáculo frontal	53
6.1.2	Procesamiento de la imagen y envío al nrf24l01	55
6.1.3	Recepción de señal y movimiento.	56
6.2	<i>Conclusión del experimento.</i>	57
	Índice de Figuras	58
	Índice de Tablas	61
	Bibliografía	62

1 INTRODUCCIÓN

La automatización de vehículos es un tema que goza de mucha repercusión en la actualidad. Consiste en dotar a un vehículo convencional de una inteligencia artificial que permita tomar decisiones en tiempo real. Actualmente se está utilizando un sensor LIDAR que ofrece un barrido en 3D del entorno prácticamente al instante con lo cual se puede deducir y prevenir los eventos que ocurrirán próximamente. También se usan gps diferenciales para conseguir errores de ubicación muy pequeños e infinidad de sensores más que logran que el vehículo pueda ser conducido por si mismo sin necesidad de conductor.

El principal problema que tiene esta tecnología es la legislación ya que quitando ciertos estados de EEUU aún no se ha decidido, por ejemplo, que acciones legales tomar si un vehículo tiene un accidente con otro vehículo.

Las empresas más influyentes en el mundo de la tecnología están invirtiendo una gran cantidad de recursos en este campo. Cabe destacar el papel que juega la empresa Google, actualmente ALPHABET Inc, y Tesla que han presentado varios modelos de coches autónomos.

En este proyecto no se va a llegar a tal punto de dificultad y se aborda el tema desde una perspectiva diferente ya que la intención es poder dotar de ciertas funcionalidades de autonomía a una plataforma móvil con un bajo presupuesto. Simplemente se hace uso de reconocimiento y toma de decisiones ante varias señales de tráfico, se evitan obstáculos frontales y se intenta conseguir una conducción fluida.

Se dividirá el proyecto en tres partes, en la primera de ella se tratará la automatización de la base robótica, en la segunda se verá el procesamiento de las imágenes y por último se verá la manera de coordinar el envío y recepción de información de una parte a la otra. Estas partes a su vez están divididas en otras para poder ahondar más en cada aspecto. Finalmente, se expondrá un ejemplo en el que se engloban todas las funciones en un mismo experimento.

Para la base robótica se ha usado un kit de electrónica básica y un Arduino Mega.

Para la parte de procesamiento de imágenes una Raspberry Pi y una cámara web USB.

Para el tema de comunicación se ha usado una pareja de módulos nrf24l01.

Por último, se incluyen resultados experimentales para dar una visión global de todas las partes.

Todo esto será detallado por separado más adelante.

2 DESCRIPCIÓN GENERAL

Temo el día en que la tecnología sobrepase nuestra humanidad. El mundo solo tendrá una generación de idiotas.

- Albert Einstein -

Este proyecto tiene la finalidad de adquirir y desarrollar conocimientos prácticos básicos de navegación automática de robot móviles. Está enfocado a la conducción autónoma y seguimiento de un camino con reconocimiento de señales.

La plataforma usada consiste en un kit de robótica que contiene varios componentes tanto electrónicos como mecánicos integrados y coordinados mediante un módulo Arduino Mega. Gracias a este módulo se consigue gobernar diversas funciones entre las que cabe destacar: Movimiento, reconocimiento de señales externas gracias a sensores, recepción de información auxiliar.

El tratamiento y reconocimiento de las señales se lleva a cabo mediante un sistema embebido muy conocido por la comunidad “maker” llamado Raspberry Pi junto a una cámara USB.

La comunicación entre ambos sistemas se lleva a cabo mediante comunicación de radio frecuencia, usando para ello el módulo nrf24l01

2.1. Esquema general.

2.1.1 Automatización robot móvil

La automatización de un robot móvil consiste en darle una inteligencia a un sistema para realizar ciertas funciones que pueden facilitar la vida.

Como se comentó en la introducción la automatización de robots móviles está a la orden del día.

Entre los mecanismos automatizados con más repercusión actual caben destacar los sistemas voladores multirotóricos y los coches autónomos.



Figura 1: Ejemplo de vehículo autónomo.

En los vehículos terrestres, se tienen varios problemas que solucionar: ubicación del vehículo en un entorno, evitar obstáculos y seguir una trayectoria fijada.

En los vehículos aéreos hay que añadir una tercera dimensión a los problemas anteriores, con lo que su utilización requiere aún más atención.

Aunque pueda parecer sencilla, la ubicación de un vehículo en un entorno no es nada fácil ya que para según que casos, se necesita ubicarse de forma global usando por ejemplo un GPS diferencial, y de forma local, para lo que se necesitaría hacer un mapeado del entorno en pseudo tiempo real, para esto se hace uso de técnicas SLAM (Simultaneous Localization And Mapping).

Este mapa sirve de utilidad para prevenir choques ya que se puede al hacer mapas en tiempo real, ver como varía la posición de un objeto en el tiempo, con lo que se puede aproximar su trayectoria y su velocidad.

Todo esto requiere de unos procesadores potentes para poder recabar, procesar y usar tal cantidad de información en tiempo real.

En este proyecto la automatización consiste en recibir información de lo que se ve por la cámara a través de la Raspberry Pi y, según sea una señal u otra, tomar una decisión de hacer un STOP, girar a la derecha o girar a la izquierda. Todo ello debe ser posible garantizando la evasión de un choque frontal con un objeto intermedio.

Como se detallará más adelante en el capítulo dedicado a la automatización del robot, para controlar la distancia frontal se usarán unos sensores ultrasonidos. Para controlar la distancia recorrida se usa un encoder digital.

Al usar componentes de muy bajo coste, aparecen diversos problemas que se detallarán también en el siguiente capítulo. Como resumen se pueden destacar los problemas de odometría, ya que la resolución de los encoder no es muy grande, y sobretodo problemas de alimentación de los motores, se han usado pilas recargables y al poco tiempo de uso, el funcionamiento no era el adecuado.

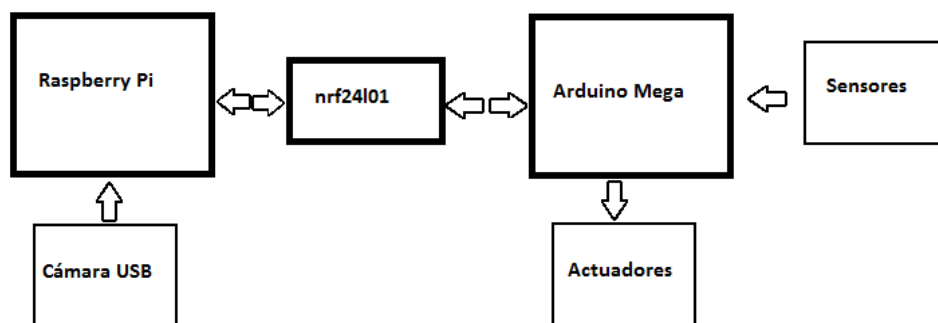


Figura 2: Esquema general del proyecto.

En la figura 2, se puede observar un esquema general del proyecto. En él se pueden distinguir distintos bloques interconectados entre sí, que globalmente cumplirían con las funciones demandadas por el proyecto.

El funcionamiento general sería el siguiente:

- Con una cámara USB se toma una imagen.
- Esa imagen es recibida por la Raspberry Pi, es procesada, y se detecta a que señal corresponde.
- Se envía a través del módulo nrf24l01 el mensaje hacia el Arduino.
- El Arduino recibe el mensaje, ordena el movimiento apropiado y controla que no existan choques y que todo funcione correctamente gracias a los sensores.

2.1.2 Procesamiento de imágenes

En el procesamiento de imágenes se parte de las imágenes tomadas por la cámara web para luego procesarla y recabar la información necesaria.

Es un tema bastante utilizado actualmente porque como sabemos, la vista es uno de los sentidos más importantes para interactuar con el entorno.

Su utilización es muy amplia ya que se puede usarlo en ámbitos muy distintos como podrían ser: Reconocimiento de alimentos en mal estados en una cinta transportadora o el reconocimiento de rostros en cámaras de seguridad por parte de la policía en algún acto criminal.

En este proyecto para este apartado se ha usado la librería abierta de procesamiento de imágenes OpenCV, la cual se puede instalar en casi cualquier sistema operativo.

Los pasos seguidos en el procesamiento de imágenes para este proyecto es el siguiente:

1. Se toman las imágenes y se almacena internamente.
2. Se realizan diversas técnicas de tratamiento de imágenes como por ejemplo el desenfoque.
3. Se buscan contornos en la imagen que tengan cuatro lados ya que las señales estarán dentro de rectángulos tal y como se explicará a continuación.
4. Mediante técnicas de comparación de imágenes se identifica que señal es la que se capturado previamente.
5. Una vez reconocida la señal se envía por el puerto SPI al módulo nrf24l01

Estos pasos serán detallados en sus respectivos capítulos.

2.1.3 Envío y recepción de información

La integración de distintos módulos requiere la interconexión de alguna manera para poder intercambiar información y datos recogidos por cada uno de ellos.

En este proyecto, se necesita pasar la información procesada de la cámara en la Raspberry Pi al Arduino. Esto se ha solucionado mediante comunicación de radio frecuencias bajo la frecuencia de 2.4Ghz con un módulo de muy bajo coste llamado nrf24l01.

Se han usado dos módulos, uno con antena externa y otro sin ella, aunque se podrían haber usado los dos sin antena externa, solo fue para mejorar el comportamiento.

Se ha hecho uso de una librería libre que está publicada en Github llamada RF24, creada y compartida por tmrh20 (ver bibliografía). Gracias a esta librería el hecho de comunicarse ha sido bastante más fácil ya que viene con ejemplos muy sencillos.

Realmente la información compartida es pequeña ya que solo se necesita saber que señal ha sido la reconocida por la Raspberry, cosa que se hace con un número y después mandar el OK desde Arduino a la Raspberry. Igualmente, más adelante se comentará con detalle el funcionamiento del programa y las conclusiones.

2.2. Conclusiones

La intención por la cual se eligió este proyecto fue para poder realizar un resumen del grado ya que de un modo u otro se tocan prácticamente todos los contenidos importantes de la carrera.

El grado es Electrónica, Robótica y Mecatrónica.

Por la parte de electrónica, se da uno cuenta de que, en un proyecto de cualquier robot, es prácticamente imposible no usarla ya que es un término muy amplio y engloba muchos aspectos. Para empezar, las pilas y batería que se usan aportan la energía necesaria para poder accionar los demás componentes. También se podría hablar sobre el microcontrolador que, a fin de cuentas, es un cúmulo de puertas lógicas. Todos los sensores y los motores, podríamos englobarlo también en la electrónica, por lo que vemos que es una parte imprescindible.

Después vendría la parte de robótica. Obviamente, al ser un proyecto de un robot móvil, la robótica es importante. Este aspecto podría asociarse a todos los componentes que realizan una función de control. Podríamos incluir tanto el Arduino, la Raspberry Pi o el conjunto entero del robot.

Por la parte de mecatrónica podría incluirse tanto el conjunto del robot, como el montaje. Aunque son aspectos que podrían parecer que no son útiles o que no son tan necesarios, son muy importantes ya que por muy bien que se tenga programado, o se tenga una idea magnífica de como poder hacer el proyecto, si no se puede probar no podemos ver si funciona o no.

Dejando eso a un lado, el proyecto me ha aportado muchos conocimientos en ciertos apartados que necesitan una dedicación práctica para poder conseguirse, como podría ser:

- Programación: Se han usado dos lenguajes de programación distintos, C y Python. También he aprendido que no es necesario un manejo completo del lenguaje para poder realizar algunos proyectos complejos.
- Procesamiento de imágenes: Usando la librería OpenCV, he adquirido conocimientos básicos de procesamiento de imágenes. Comparación de imágenes, reconocimiento de colores, seguimiento de objetos, etc.
- Comunicación: Aunque no es uno de los temas en los que más incapié se ha hecho durante el grado, la comunicación de distintos dispositivos es una cosa imprescindible. En este proyecto se ha usado comunicación inalámbrica de radiofrecuencia a través del módulo nrf24l01. Se ha aprendido a comunicar un Arduino con una Raspberry Pi con este módulo para poder intercambiar datos, o sincronizar los tiempos en los dos componentes.

3 AUTOMATIZACIÓN ROBOT MÓVIL

*Nunca permitas que el sentido de la moral te impida
hacer lo que está bien.*

- Isaac Asimov -

La automatización de sistemas consiste en dotar a máquinas a disponer de una autonomía que no tendría inicialmente. Esto le permite tomar decisiones en ciertos aspectos que pueden resultar repetitivos para el operario. Algunos ejemplos pueden ser: El descarte de aceitunas de un tamaño inferior a la media, el montaje de las piezas de vehículos mediante robot manipuladores.

El enfoque que se le ha dado a la automatización de robots móviles en este proyecto es un enfoque meramente educativo, intentando aprender lo máximo posible de aspectos prácticos que no se enseñan en la facultad y que puede que más adelante sean de utilidad. Los metas que se han ido consiguiendo han sido las siguientes:

- Se montó la plataforma uniendo los componentes necesarios para que el robot se moviera. Esto incluye, ruedas, motores, Arduino.
- Una vez se consigue que se moviese el siguiente paso fue añadir sensores ultrasonidos para evitar choques frontales. Se usó un controlador todo-nada, solo se buscó ver el funcionamiento del sensor de ultrasonidos.
- Luego se planteó el problema de que se moviese una cierta distancia para lo cual se necesitaba saber cuanto se había movido en cada momento, por lo cual se decidió añadir un encoder digital. Teniendo este sensor, se hizo necesario el uso de un controlador un poco más avanzado para lo cual se usó un controlador tipo PI.
- Una vez solucionado el problema de moverse una distancia, fue cuando se decidió unir el movimiento del robot con las señales reconocidas por la cámara.

Existen diferentes tipos de robots, a grandes rasgos podría hacerse la siguiente clasificación de acuerdo con su función:

- **Robots manipuladores:** Este tipo de robots también son conocidos como robots industriales.

La organización internacional de estándares (ISO) da una definición de robots manipuladores que es la siguiente:

Manipulador multifuncional reprogramable con varios grados de libertad, capaz de manipular materias, piezas, herramientas o dispositivos especiales según trayectorias variables programadas para realizar tareas diversas.

Un manipulador robótico consta de una secuencia de elementos estructurales rígidos, denominados eslabones, conectados entre sí mediante articulaciones.

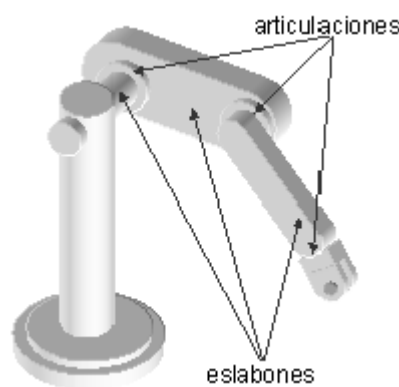


Figura 3: Elementos estructurales de un manipulador robótico

Estos robots están fijos por la base y sus eslabones tienen una libertad limitada ya que solo pueden moverse en una zona de trabajo.

- **Robots móviles:** Estos robots han sido diseñados para desplazarse. Cuentan con ruedas, patas, hélices u otros elementos que les permiten desplazarse.

Se pueden distinguir varios grupos de robots móviles:

- 1- **Robots móviles terrestres:** Este tipo de robots se desplazan en dos dimensiones, se podría decir que se mueven en el plano x, y. Es un concepto muy amplio ya que podría incluir: Vehículos autónomos, robot limpiador, Vehículos de guiado automático (AGV), y muchos más. Su papel principal es el de moverse de un lado a otro evitando obstáculos y puede servir para mover mercancías. Los robots ayudantes, como podrían ser los robots guías móviles en un museo, también entrarían en este apartado.



Figura 4: Robot limpiador de la marca Roomba

- 2- **Robots móviles aéreos:** Son un tipo de robots móvil capaces de moverse por el aire. Pueden encontrarse también como UAV, por sus siglas en inglés de vehículos aéreos no tripulados. Tiene un diseño similar a los vehículos aéreos convencionales, aviones, helicópteros, etc. Pueden cumplir muchas funciones entre las que cabe destacar: Inspección, observación y vigilancia.

Se usan mucho en el ámbito militar ya que, en caso de recibir un bombardeo, el país que lo posee no recibiría bajas humanas.

Puede usarse para observar y vigilar un monte y avisar rápidamente de un incendio. Esta función es muy útil en montes grandes ya que con los sistemas actuales de procesamiento de imágenes se puede detectar el fuego lo más rápidamente posible y no sería necesario tener a una persona dando vueltas en un helicóptero las 24 horas del día.

3.1 Esquema general

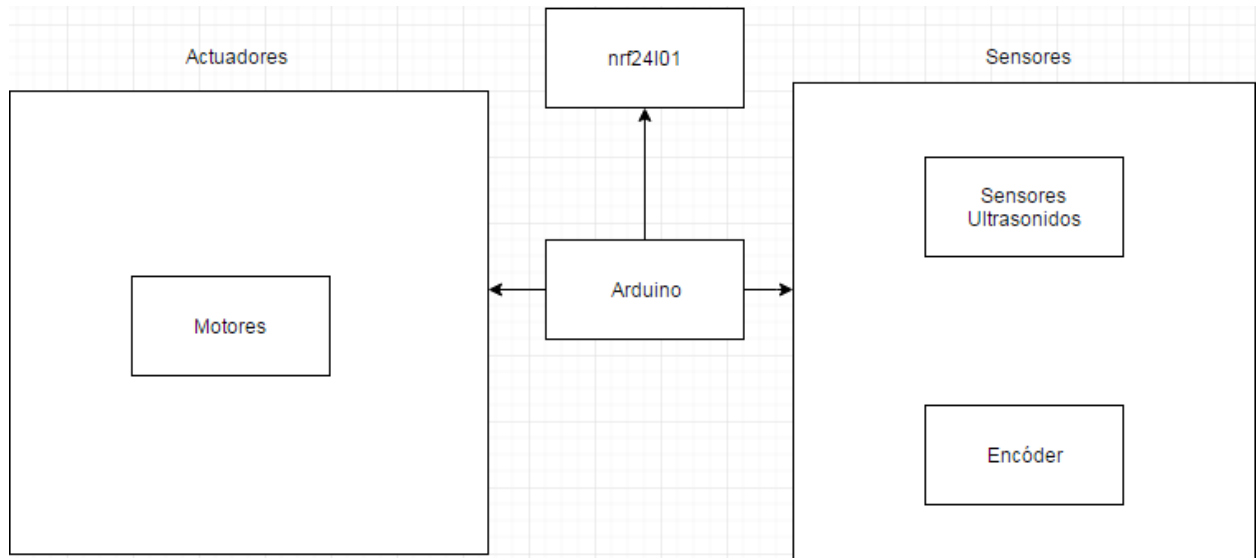


Figura 5: Esquema general Automatización Robot

Como se puede observar en el esquema anterior, se podría dividir este apartado en 2 grandes grupos:

- 1- Actuadores: En este grupo, se tienen los componentes sobre los que el Arduino realiza una acción. Estos componentes son los siguientes:
 - Motores: En este subgrupo se incluye las acciones que hay que realizar sobre el motor y sobre el puente H, tal y como se explicará en el siguiente apartado.
- 2- Sensores: Se incluye a los componentes que dan información del entorno al Arduino, que tendrá que procesarla, almacenarla y hacer lo que crea conveniente en cada caso.
 - Sensores Ultrasonidos: Usados para evitar choques frontales.
 - Encoder: Sirven para medir lo que se ha movido el robot.

El módulo nrf24l01 no lo se ha incluido en ningún apartado ya que realmente hace de sensor, informa de la señal reconocida, y de actuador, informa a la Raspberry Pi que ha recibido el dato.

3.2 Componentes y montaje

En este apartado se explicarán que componentes se han usado.

Los componentes son los siguientes:

- **Arduino Mega 2560.** Se trata de una placa microcontrolador basado en el chip ATmega2560. Tiene 54 entradas/salidas digitales, de las cuales 15 pueden ser usadas como PWM, 16 entradas analógicas, 4 UARTs, un cristal oscilador de 16 MHz. En general es una placa muy completa para el precio que tiene.

Se eligió este modelo en vez de otros ya que necesitaba muchos pines y necesitaba algo más de potencia sin querer pasarnos de presupuesto.

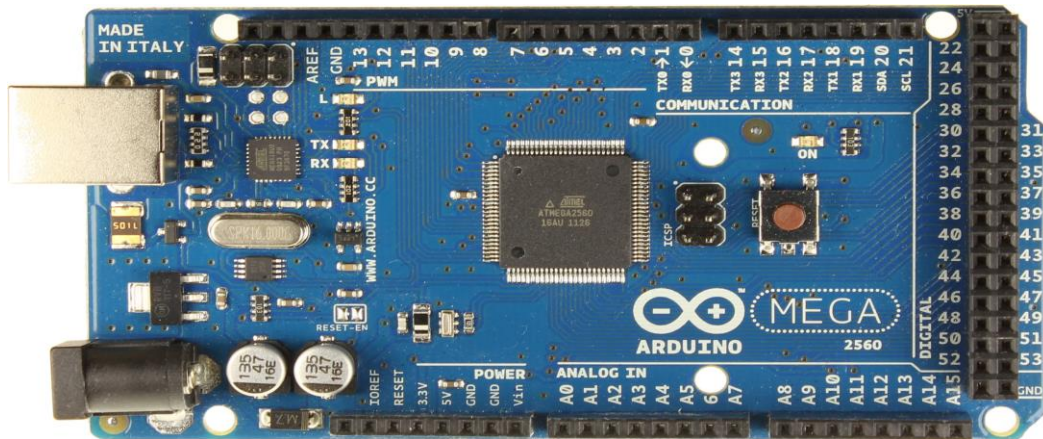


Figura 6: Arduino Mega

Es el sistema central de este apartado, es quien maneja los demás componentes.

- **Arduino Sensor Shield:** Placa independiente que se acopla sobre el Arduino, ayudándonos a conectar los pines, ofreciendo muchos pines de tierra y de voltaje.

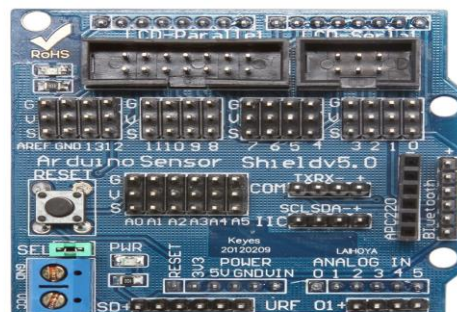


Figura 7: Arduino Sensor Shield

- Sensores Ultrasonidos: Se han usado sensores HC-SR04.



Figura 8: Sensor ultrasonido HC-SR04

- Puente H: Se utilizará para este proyecto el modelo de puente H L298N. Ofrece poder controlar fácilmente dos motores DC tanto en velocidad como en sentido.

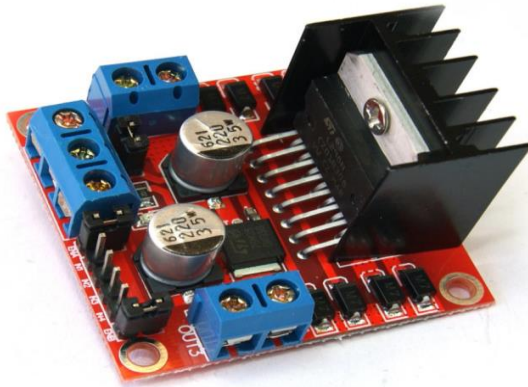


Figura 9: Puente H L298N

- Kit de robótica básico. Está formado por los siguientes componentes:
 - Base de metacrilato. Sirve como apoyo para sostener todos los componentes en una base. Tiene agujeros para poder asegurar los componentes.
 - Ruedas: Vienen tanto ruedas normales, ruedas de gomas que son las ruedas motoras, como una rueda loca que sirve para que la plataforma no vuelque hacia la parte de atrás.
 - Motores: El kit ofrece dos motores de DC que incluye las reductoras en un mismo moduli.
 - Encoder: Trae dos encóderes diferenciales.
 - Tornillos y tuercas. Sirve para sujetar bien los componentes a la base de metacrilato.



Figura 10: Kit Robótica

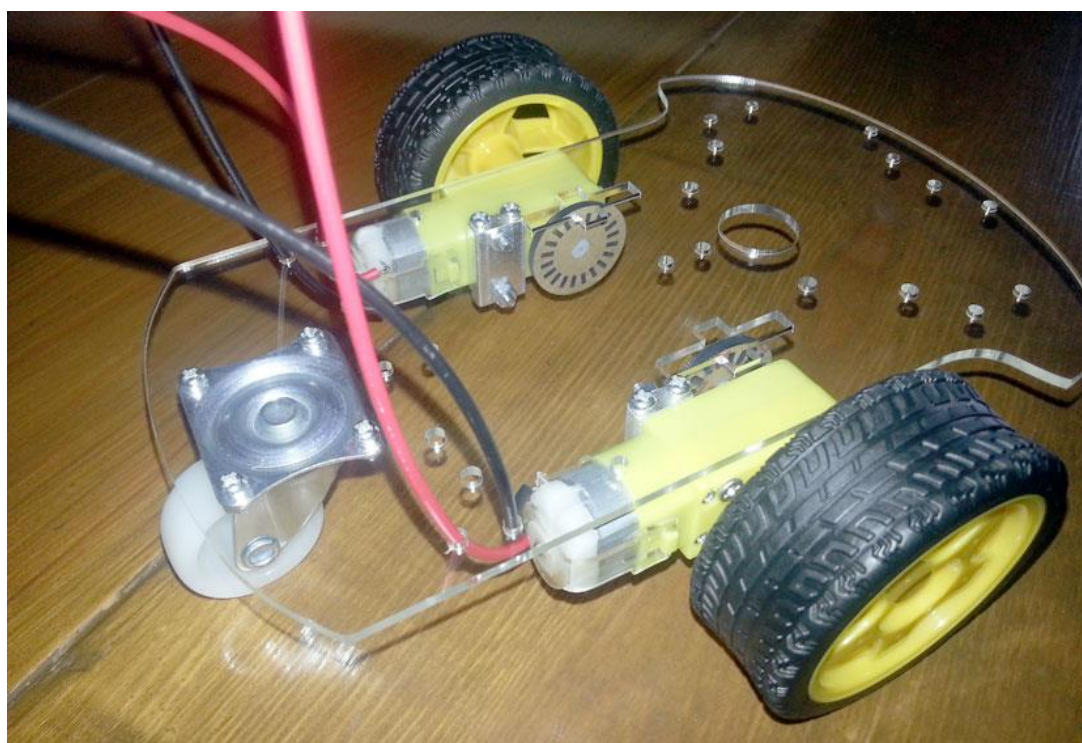


Figura 11: Kit Robótica Montado

3.3 Implementación de módulos

En este apartado se expondrá en detalle cada módulo como funciona como se implementa y su conexión al Arduino.

Como se comentó previamente, el Arduino es el eje central de la automatización en este proyecto. Se encarga de recibir información, tomar decisiones y actuar sobre los diferentes actuadores.

A continuación, se explicará por separado cada módulo:

- **Motor con reductora y rueda.** Este componente consta de un conjunto formado por un motor unido a una reductora al que se puede anclar una rueda.

El motor es un motor de DC, por lo tanto, se controla en intensidad teniendo una tensión constante. Dependiendo del signo de la tensión, positiva o negativa, el sentido de giro del motor varía, con lo que se consigue que la rueda gire en un sentido u otro.

- **Puente en H:** Como se comentó se ha utilizado un puente en H modelo L298N el cual viene con el disipador de calor y con las bornas preparadas para simplemente apretar los cables.

En la figura siguiente, se muestra la distribución de pines del módulo. Como se puede apreciar se tienen dos pares de salidas. Dos salidas para un motor y dos salidas para el otro.

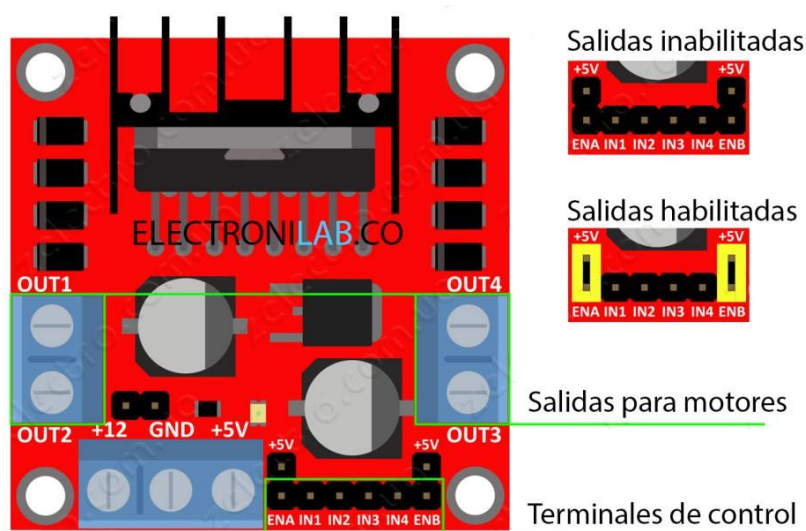


Figura 12: Conexiones Puente H L298N

Se tiene la alimentación propia del módulo que debe ser a 12V para que los motores no tengan problemas. En la figura, donde se ve salidas habilitadas, hay dos rectángulos en amarillo que representan dos jumpers, que se necesitan para habilitar las salidas de los motores.

Como dijimos, sobre el motor se puede actuar cambiando su velocidad y su sentido de giro, para ello se usan los terminales de control. Se tiene para cada motor 3 terminales de control, el terminal de activación, y dos terminales para determinar el giro.

En la tabla siguiente se va a comentar cuales son los valores que deben tomar los terminales para que se consiga una función u otra.

Terminal Activación	Terminal Giro 1	Terminal Giro 2	Función
Alto	Bajo	Alto	Giro derecha
Alto	Alto	Bajo	Giro Izquierda
Bajo	No importa	No importa	Detención

Tabla 1: Combinaciones Puente H L298N

Las demás combinaciones hacen que se detenga el motor.

- **Sensores ultrasonidos:** La tecnología ultrasonido se ha usado históricamente como medida de distancia en los sónares submarinos.

Para Arduino existen unos módulos baratos y sencillos de utilizar que dan buenos resultados. Estos son los módulos HC-SR04 como se comentó anteriormente. A continuación, se mostrará una tabla con las características del módulo.

Características	
Alimentación	+5v DC
Frecuencia de trabajo	40 KHz
Consumo (suspendido)	< 2mA
Consumo (trabajando)	15mA
Ángulo efectivo	< 15°
Distancia	2cm a 400cm *
Resolución	0.3 cm

*A partir de 250cm la resolución no es buena

Tabla 2: Características HC-SR04

Como se puede apreciar en la figura 13, consta de 4 pines:

- 1- **Pin VCC:** El cual se conectará a alguna salida de 5 V.
- 2- **Pin GND:** El cual se conectará a tierra.
- 3- **Pin Trigger:** Pin de disparo. Se conecta a algún pin digital.
- 4- **Pin Echo:** Pin de recepción. Se conecta a algún pin digital.

El funcionamiento de esta tecnología es simple: Se hace un disparo de un ultrasonido mediante el pin Trigger, se espera a que se reciba mediante el pin Echo y se calcula el tiempo que ha tardado en llegar.

Como sabemos, la velocidad del sonido en el aire es aproximadamente 340m/s, por lo que si se sabe cuanto tiempo ha tardado mientras se lanza hasta que vuelve, se puede calcular la distancia a la que está un objeto con la siguiente fórmula.

Teniendo la medida del tiempo en milisegundos, y se quiere calcular la distancia en centímetros:

$$Distancia = \frac{340 * 100}{1000 * 2} * tiempo(ms) = 17 * tiempo(ms)$$

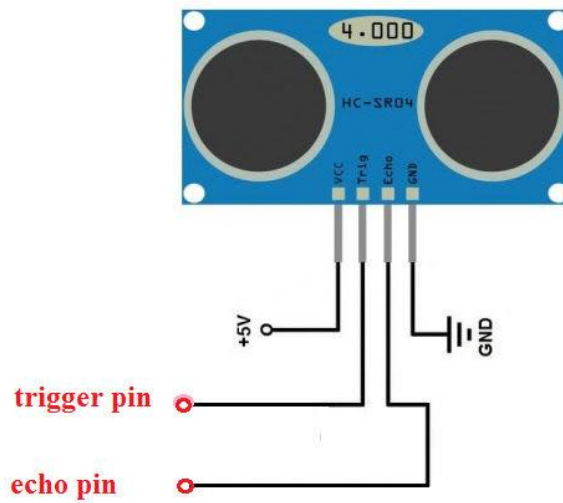


Figura 13: Conexiones HC-SR04

Para el ultrasonido, se usó una librería llamada NewPing que facilita las cosas si se quiere añadir más de uno.

La forma de utilizar esta librería es la siguiente:

- Primero se incluye la librería.
- Se define los pines de Trigger y Echo, por ejemplo: `#define TRIGGER_PIN1 4`. Esto quiere decir que el Trigger del ultrasonido número 1 está conectado en el pin 4.
- Se declara el ultrasonido. Ejemplo: Siendo `MAX_DISTANCE` la máxima distancia que se le quiere dar de resolución.
`NewPing sonar1(TRIGGER_PIN1, ECHO_PIN1, MAX_DISTANCE);`
- Cuando se quiera medir se hace lo siguiente: `uS1 = sonar1.ping();` Siendo `uS1` un entero en el cual se quiere almacenar la distancia medida por el ultrasonido uno.

- Encoder: Se ha utilizado un encoder de una banda, ya que, por problemas de sobrecarga, el Arduino no podía procesar más resolución.

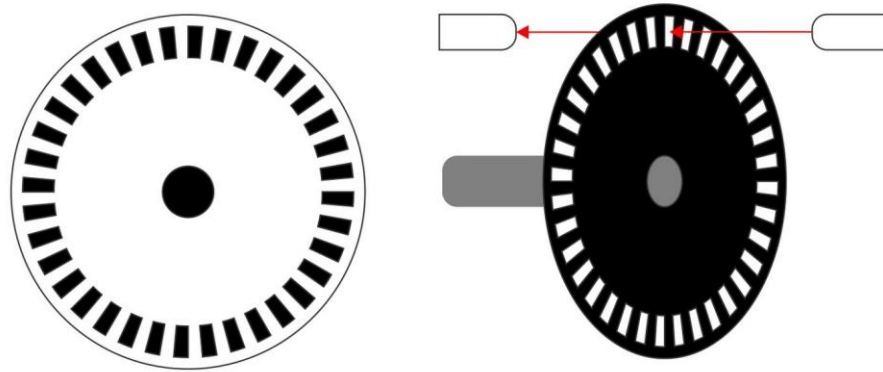


Figura 14: Encóder digital.

Aún teniendo solo una banda, se ha decidido a limitar la cantidad de huecos que tenemos, ya que como se ha comentado provocaba problemas de sobrecargas en la lectura.

Las lecturas del encoder se han realizado mediante interrupciones. Se han usado los pines 2 y 3 ya que son los pines que pueden leer interrupciones externas.

Los pasos a seguir para activar y asociar una interrupción a un pin son los siguientes

- Se define el pin como entrada.
- Se activa la resistencia interna de Pull-UP
- Se asocia los cambios que ocurran en uno de los dos pines de la interrupción a una función. Por tanto, cada vez que sucede una interrupción se llama a la función oportuna.

3.4 Obtención del controlador

Como se comentó anteriormente, es necesario el uso de un controlador para poder controlar, valga la redundancia, los movimientos del robot.

El método seguido para obtener el controlador es el siguiente:

1- Medir la velocidad a distintas intensidades de motor.

Nos hace falta calcular la velocidad. Para ello se puede hacer uso de los resultados que ofrecen los encoders, y el tiempo de ejecución.

Para este apartado es necesario hacer un programa en Arduino que consista en, para cada valor de intensidad, mostrar el tiempo del programa, las vueltas que da la rueda, y la velocidad que lleva la rueda. Antes de cambiar de una intensidad a otra, es necesario esperar un tiempo para que el sistema se estabilice. Los datos se muestran de tal forma que sea después fácil trabajar con ellos en MatLab, por lo que se muestran en cada fila: los valores del tiempo, vueltas y velocidad en ese momento. Un ejemplo de los datos podría ser el siguiente:

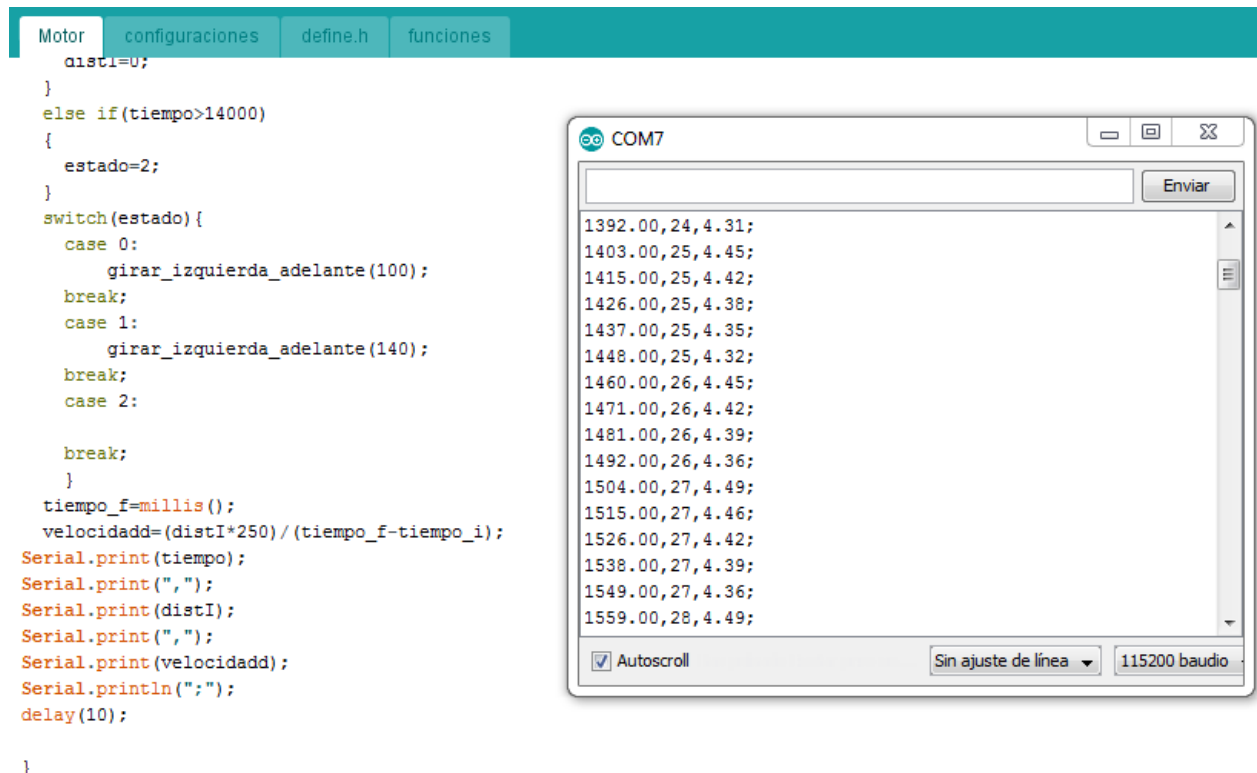


Figura 15: Tiempo, vueltas de encoder y velocidad.

Como se puede apreciar en la imagen se muestran varias filas que corresponden a distintas medidas. En cada fila aparecen tres datos separados por comas. Tiempo de ejecución, en milisegundos, vueltas y velocidad. La velocidad se saca a partir de las vueltas que da la rueda y haciendo una conversión de vueltas a centímetros. Cada vuelta completa de la rueda corresponde a 2.5cm por lo que se multiplica por 250 para obtener la velocidad en centímetros/segundos.

2- Sacar el modelo del robot a partir de esa velocidad.

Una vez se tienen estos datos, se crean en MatLab un script que separe los valores.

Lo siguiente que se hace es convertir el tiempo a segundos en el script y representar velocidad frente a tiempo.

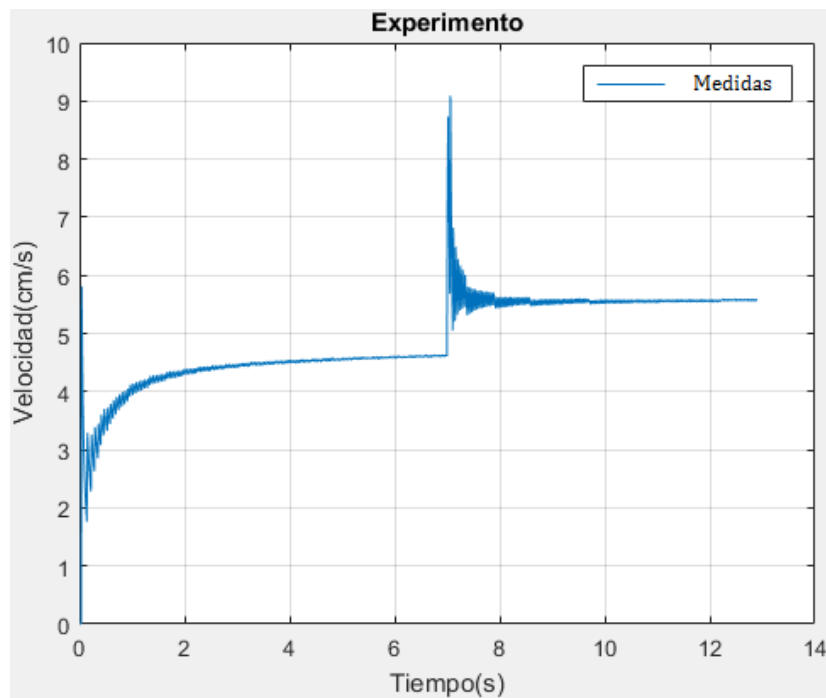


Figura 26: Experimento Medidas.

Como se puede apreciar, el cambio de velocidad ocurre en el segundo 7.01, que es el valor que se ha puesto en el programa, y se puede ver como existen valores que no son los que se tiene en régimen permanente. Esto es debido a un error en la medida del encóder.

Una vez se obtiene la gráfica, hay sacar un modelo matemático que se asemeje lo más posible a las medidas. Para ello se ha aproximado a un sistema de primer orden. Para conseguirlo se necesitan dos valores, τ y K ya que la entrada es un escalón de intensidad.

$$u(s) \rightarrow \boxed{\frac{K}{\tau s + 1}} \rightarrow y(s)$$

- τ : Constante de tiempo. Es el tiempo que tarda el sistema desde que se le proporciona el escalón hasta que llega al 63% del valor en régimen permanente. Para calcularlo se necesita los valores en régimen permanente de las dos intensidades:

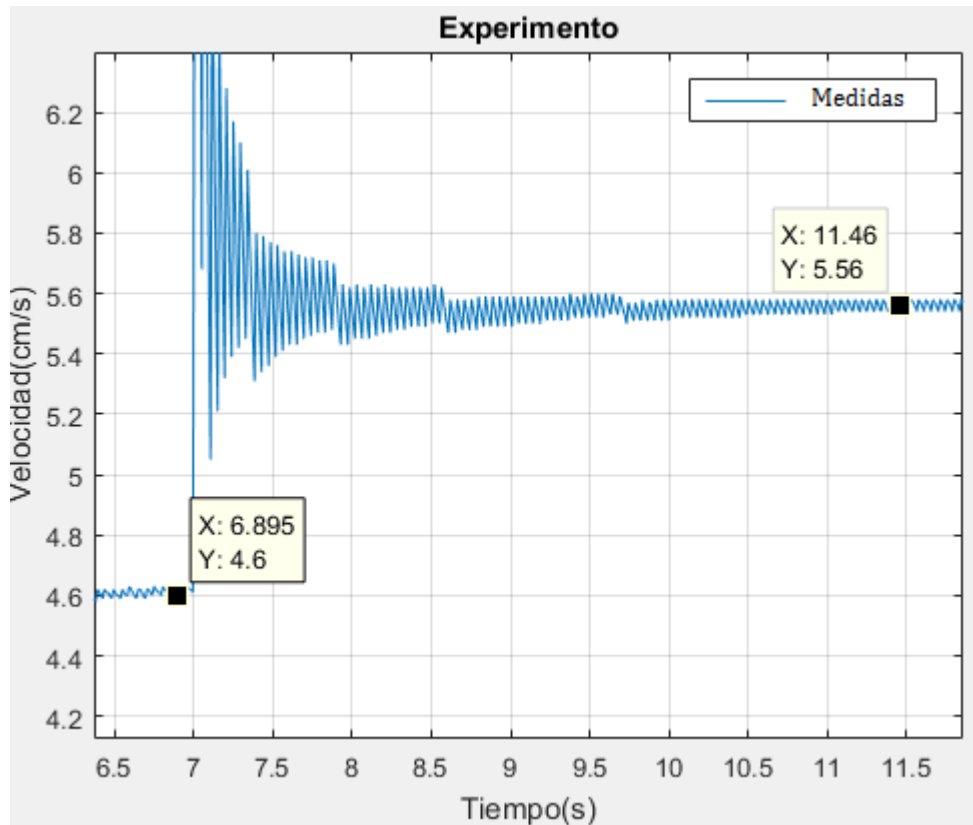


Figura 37: Valores en Régimen Permanente de las medidas.

Se puede observar que los valores son: 4.6 cm/s y 5.56 cm/s. Por tanto, el valor de régimen permanente que se necesita es:

$$Y_{rp} = 5.56 - 4.6 = 0.96 \text{ cm/s}$$

Se calcula el 63% de ese valor y se suma al régimen permanente de la primera intensidad:

$$Valor = (0.63 * 0.96) + 4.6 = 5.205 \text{ cm/s}$$

Lo siguiente es ver en que tiempo alcanza ese valor, para ello se une con una curva imaginaria el punto en el cual recibe el escalón con el valor en régimen permanente intentando asemejarlo a un sistema de primer orden.

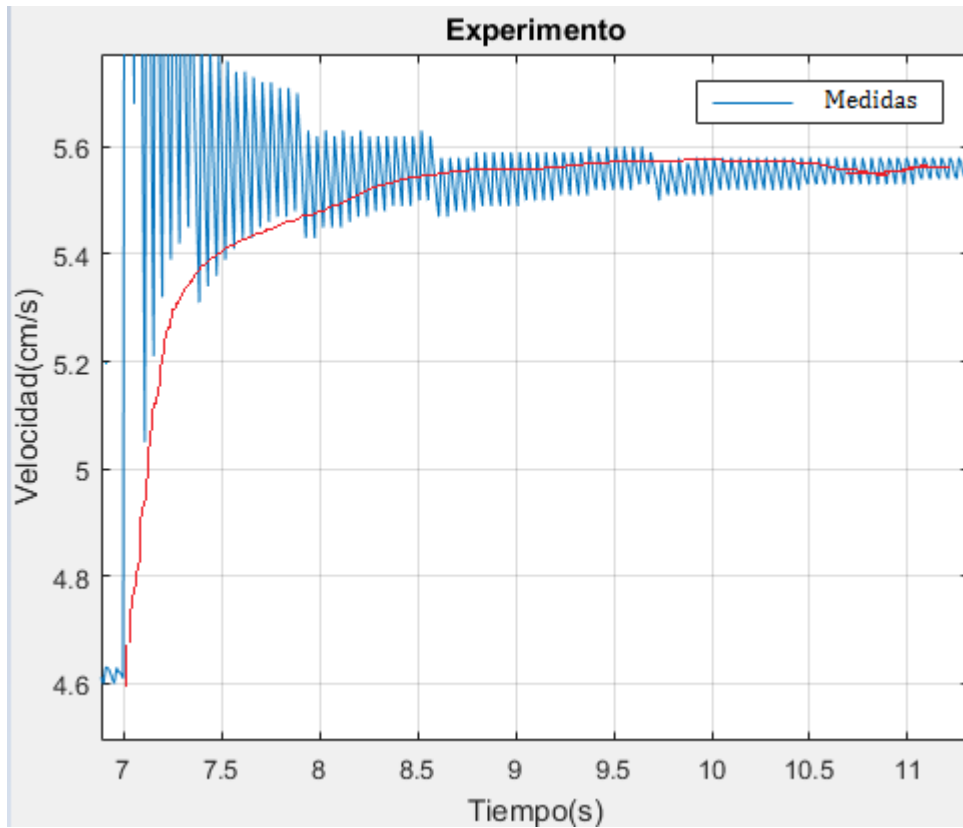


Figura 4: Curva auxiliar para calcular valores en las medidas.

El valor del tiempo en el que se alcanza el punto 5.2 es aproximadamente 7.24 segundos. Por lo que la constante τ se calcula con la siguiente ecuación:

$$\tau = 7.25 - 7.01 = 0.24 \text{ s}$$

- K: Ganancia del sistema. Para calcular este valor, se hace uso de la siguiente fórmula:

$$Y_{rp} = K * U$$

Siendo U el escalón que se ha introducido al sistema. Para calcularlo, se necesita saber cual es el voltaje que reciben las pilas, y el voltaje aplicado para cada prueba.

El voltaje que suministran las pilas es 7.2 V. Al ser una salida digital el que le proporciona la velocidad al motor, un valor de 255 corresponde a 7.2 Voltios y un valor de 0 corresponde a un valor de 0 Voltios. Por tanto, en el programa se ha pasado de un valor de 100 a un valor de 140, lo que corresponde a un escalón de 40. Este valor convertido a Voltios sería el siguiente:

$$\text{Escalón} = \frac{40}{255} * 7.2 \text{ V} = 1.1294 \text{ V}$$

Y_{rp} es el valor que se ha calculado antes: $Y_{rp} = 0.96 \text{ cm/s}$

Teniendo estos dos valores, se puede calcular el valor de K.

$$K = \frac{0.96}{1.1294} = 0.85 \text{ cm} * \text{V/s}$$

Por lo tanto, el modelo tiene la siguiente forma:

$$G(s) = \frac{K}{\tau * s + 1} = \frac{0.85}{0.24s + 1}$$

Una vez obtenido el modelo matemático, hay que compararlo con las medidas:

Para ello se realiza un esquema en Simulink que tiene como entrada el valor inicial más el escalón y tiene como salida el valor de la velocidad en cada momento:

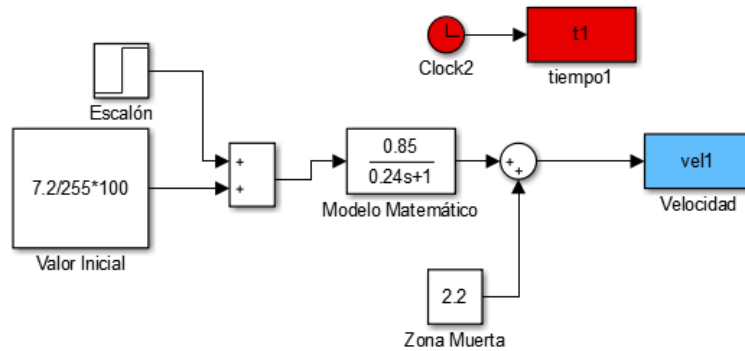


Figura 59: Esquema Simulink Modelo Matemático.

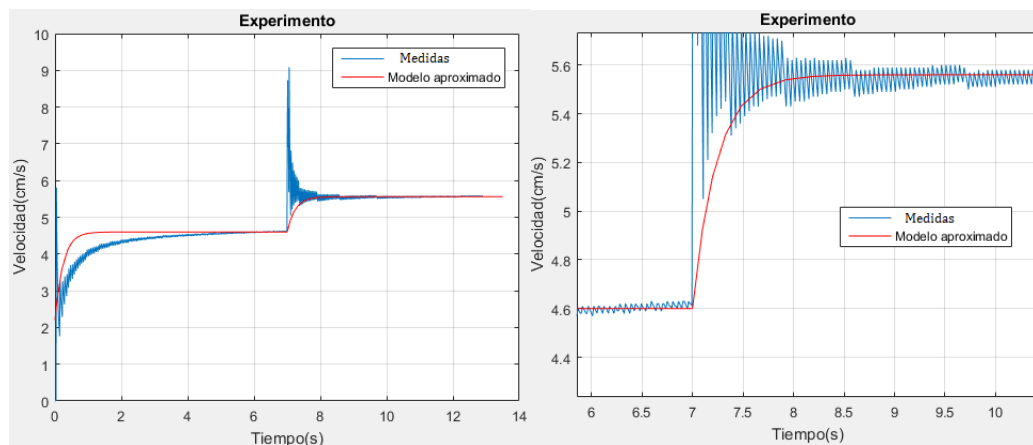


Figura 20: Comparación Medidas con Modelo Matemático (aproximado)

3- Sacar el controlador a partir del modelo.

A continuación, se va a calcular el controlador a partir del modelo matemático.

Para ello se ha usado el método de la cancelación de dinámica. Este método es muy sencillo de aplicar y da buenos resultados si el sistema es estable y rechazar perturbaciones no es el objetivo principal.

Como el sistema es un sistema de primer orden, el controlador tendrá la siguiente forma:

$$C(s) = \frac{K_c}{s} \frac{\tau s + 1}{K}$$

Lo que proporcionaría el siguiente sistema en bucle cerrado:

$$G_{BC}(s) = \frac{K_c}{s + K_c}$$

Los valores τ y K son los proporcionados por el modelo:

$$\tau = 0.24 \text{ s}$$

$$K = 0.85 \text{ cm} * V/s$$

Para calcular el valor de K_c , se usa la siguiente fórmula:

$$t_s = 3/K_c \text{ o } t_s = 4/K_c$$

Siendo t_s el tiempo de subida que se podría definir como el tiempo que tarda la salida del sistema en llegar desde el 5% al 90% del valor de referencia final.

Este tiempo es aproximadamente 0.71 segundos. Por lo cual K_c tomaría un valor de:

$$K_c = \frac{3}{0.71} = 4.225 \text{ s}^{-1}$$

Por lo tanto, el controlador queda de la siguiente forma:

$$C(s) = \frac{4.225 * (0.24s + 1)}{0.85 * s}$$

Teniendo estos datos, se procede a realizar un modelo en simulink para comparar la referencia a seguir con la salida del sistema con el controlador para ver si se consigue un resultado correcto.

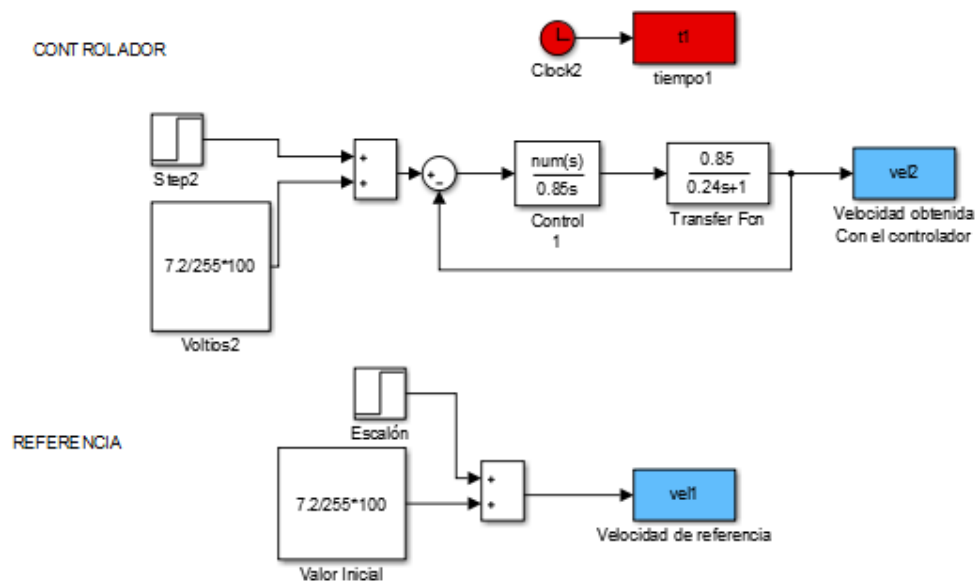


Figura 216: Esquema Simulink. Modelo con Controlador (arriba). Referencia (abajo).

El resultado de dicha comparación es el siguiente:

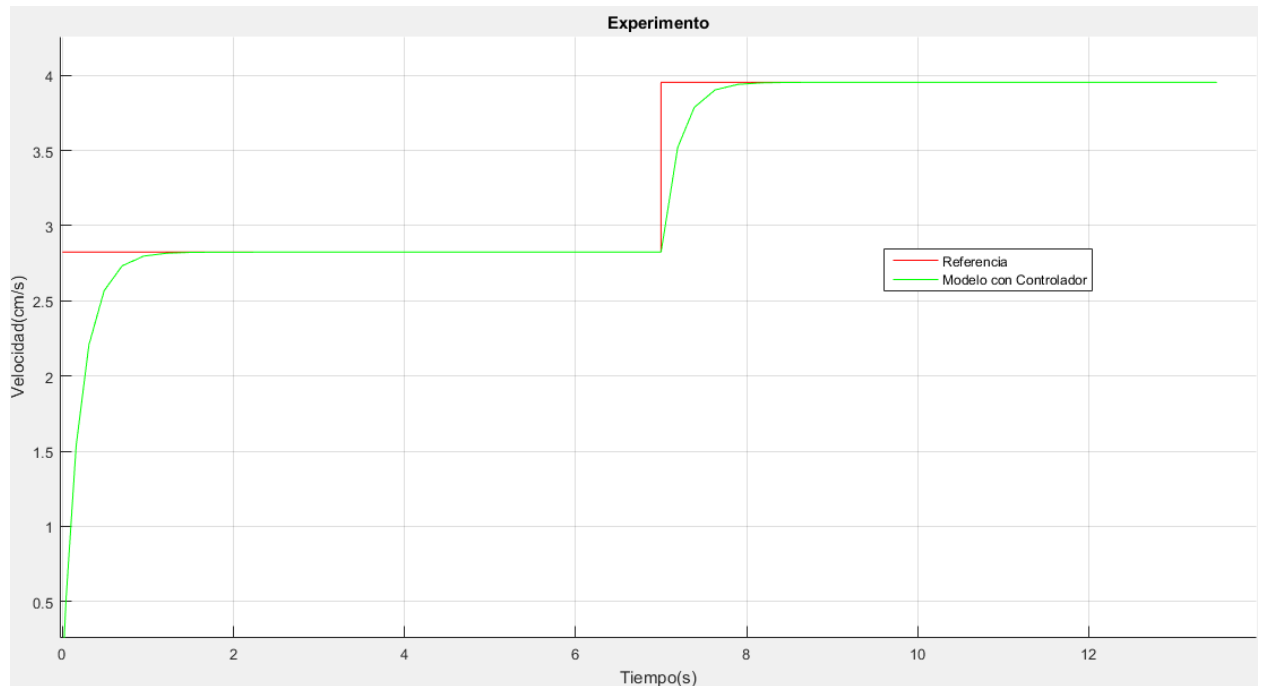


Figura 72: Gráfico para comparar la salida del sistema con controlador (verde) y de la referencia (rojo).

Como se puede apreciar el error en régimen permanente es nulo, y el seguimiento de la referencia se consigue en un tiempo que se puede permitir.

Teniendo el controlador, se puede obtener los valores para implementarlo en el microcontrolador. Para ello se sacan los valores suponiendo que el controlador es un PI:

$$\text{PI} \left\{ \begin{array}{l} T_I = \tau \\ K_P = \frac{K_c}{K} \tau \end{array} \right.$$

Los valores del controlador PI son los siguientes:

$$T_I = 0.24$$

$$K_P = 1.19$$

Con estos valores la implementación en un controlador es inmediata.

3.5 Conclusiones

A continuación, se van a explicar ciertos aspectos relacionados con este apartado.

En primer lugar, el robot usado en el proyecto presenta una topología llamada diferencial. Esta configuración no tiene ruedas directrices, el direccionamiento se realiza modificando la velocidad relativa de las ruedas motrices, ruedas que proporcionan la fuerza de tracción al robot.

Las ventajas de este tipo de locomoción son:

- Es un sistema barato.
- Tiene un diseño simple.
- Es fácil de implementar.

Como todo, también tiene inconvenientes:

- Es difícil de controlar.
- Requiere control de precisión para trayectorias rectas.

Otro de los aspectos a resaltar en este apartado es el problema de la odometría.

Al realizar el Arduino el control de las vueltas que ha dado el encóder entre otras cosas, se ha disminuido la resolución del encoder para que no se produjesen cuelgues en el microcontrolador. El problema radica en que al recibir las señales de movimiento del encoder por interrupciones, es necesario que exista un tiempo entre que llega la interrupción, se procesa y se libera la función de interrupción para seguir recibiendo señales.

4 PROCESAMIENTO DE IMÁGENES

La irrupción de las nuevas tecnologías nos obliga a educar a los niños de forma distinta.

- Howard Gardner -

En este capítulo se van describir las partes en las que se divide el procesamiento y tratamiento de imágenes en este proyecto. Primero se pondrá un esquema general en el cual se mostrará en un diagrama como se comporta este módulo, cual es su entrada, su salida y que se hace internamente. A continuación, se desarrollará cada subapartado del tratamiento de imágenes por separado y por último se expondrán unas conclusiones, para debatir los problemas encontrados, caminos en los que se podría mejorar y una pequeña impresión personal.

Este apartado se lleva a cabo en una Raspberry Pi, que es un sistema embebido en el cual se pueden usar distintos sistemas operativos cada uno con su funcionalidad. Para este proyecto, se ha utilizado una distribución de Linux llamada Debian en su versión Jessie. Este sistema operativo, se asemeja a un entorno de escritorio Linux convencional, por lo que podría crear

El modelo usado de Raspberry Pi es el modelo Raspberry Pi 2 modelo B.

Se va a hacer uso de la librería libre OpenCV en su versión 3.1.0 y para la programación del código se va a utilizar el lenguaje *Python*.

OpenCV lleva siendo más de 10 años una librería muy usada para la visión artificial por ser muy exportable, (funciona en los principales sistemas operativos) usa distintos lenguajes de programación, como puede ser C, C++ o Python.

Para este proyecto se va a usar la librería 3.1.0, la cual se puede descargar desde su repositorio oficial

Para el código, se utilizará el lenguaje Python, en su versión 2.6.7 ya que es muy sencillo de programar y no necesita ser compilado cada vez que se quiere hacer una prueba. Python se está convirtiendo en un lenguaje muy utilizado en robótica por ser fácil de programar es muy flexible y tiene una comunidad bastante amplia con buenas librerías casi para todo.

4.1 Esquema general

Como se puede ver en la imagen siguiente, se tiene como entrada las imágenes que se toman a partir de la cámara por un lado, y por otro las imágenes de muestra, que son las señales que se quieren detectar. A cada una se le realiza un tratamiento y después se comparan ambas imágenes procesadas. Una vez realizado esto, se envía el resultado de la señal correspondiente al módulo nrf24l01 tal y como se explicará en el siguiente apartado.

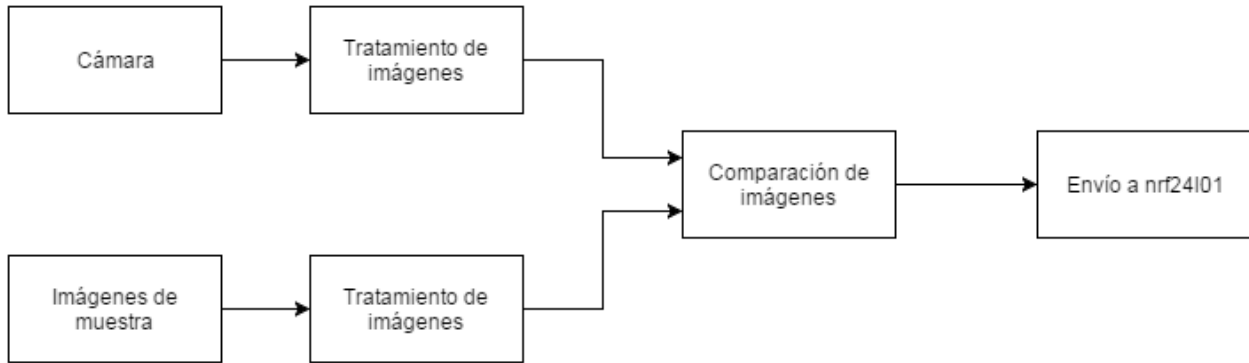


Figura 8: Esquema general del tratamiento de imágenes.

4.2 Raspberry Pi

Es un ordenador de placa reducida, lo que significa que tiene tanto el microprocesador, la memoria RAM, las entradas y salidas y las demás características en una sola tarjeta, que además suele ser de tamaño reducido.

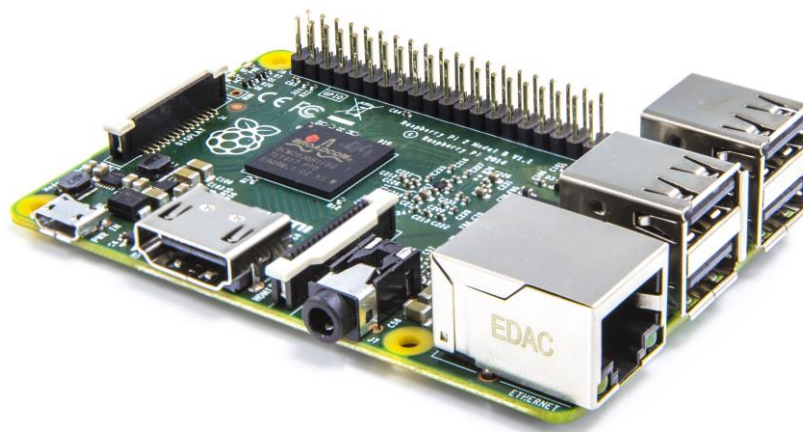


Figura 94: Raspberry Pi 2

El sistema operativo se incluye mediante una tarjeta SD.

La primera versión de este ordenador se lanzó en Febrero de 2012. Tenía un chip BCM2835, un procesador gráfico y 256MB de RAM. Tenía dos puertos USB.

En Julio de 2014 lanzaron el modelo B+ que contaba con más puertos USB, un total de cuatro, más memoria RAM 512, más GPIO ya que pasaba de 26 pines a 40. El sistema operativo se podía meter en una tarjeta microSD y algunas novedades más.

En Febrero de 2015 se lanzó el modelo 2 cuya principales novedades fueron el procesador: *900MHz quad-core ARM Cortex-A7 CPU* y un aumento de memoria RAM de 512MB a 1 GB.

Con estas especificaciones y manteniendo el precio inicial, cercano a los 40 €, se ha convertido en una herramienta casi imprescindible para la comunidad *maker*.

4.3 Tratamiento de imágenes tomadas por la cámara

En este apartado, se van a explicar cuales son los pasos necesarios que hay que seguir para transformar la imagen tomada por la cámara en una imagen preparada para ser comparada.

En resumen, lo que se hará será lo siguiente:

1. **Almacenar una imagen tomada por la cámara.**
2. **Conversión a blanco y negro.**
3. **Hacerle un desenfoque.**
4. **Detectar contornos.**
5. **Encontrar polígono de 4 lados.**
6. **Recortar el rectángulo.**

Con esto se tendría la imagen lista para ser comparada.

A continuación, se explicará en detalle cada paso organizándolo en subapartados.

4.3.1 Conversión de color

Primero de todo, al estar tomando muestras de la cámara de forma continua, se necesita capturar un momento y almacenarlo para poder trabajar con él.

Una vez se consigue la imagen, hay que hacer una conversión de color. En este caso se pasa de la escala RGB que es en la que se captura la imagen, a una escala de grises.

Se escoge esta escala, porque lo que interesa es encontrar contornos y como se van a utilizar señales dentro de un fondo rectangular blanco, al colocarlo sobre una pared negra, el contraste es mejor y es fácil encontrar el contorno en este caso.

Un ejemplo de conversión de color de escala RGB a escala de grises de una imagen tomada por una cámara sería la siguiente:



Figura 25: Imagen tomada en escala RGB



Figura 26: Imagen convertida a escala de grises.

Para convertir de RGB a escala de grises en opencv se hace uso de la función `cv2.cvtColor()`, por ejemplo, en este caso:

```
gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
```

Esta función es general, y sirve para hacer cualquiera cambio de escala de color. Toma como valores, la imagen de partida en la escala que se quiera, en este caso es una imagen en escala RGB, y por último un macro para elegir la conversión que queramos, en este caso de RGB a escala de grises.

Matemáticamente la conversión de RGB a escala de grises es sencilla, se coge por separado cada canal de color, en cada pixel se suman las intensidades de cada canal y se divide entre tres, en definitiva, una media.

4.3.2 Desenfoque

El enfoque sirve para realzar un elemento de una imagen sobre el resto de ella.

La técnica de desenfoque Gaussiano, consiste en suavizar la intensidad de un pixel haciendo una media de las intensidades de los píxeles adyacentes a una determinada distancia.

Una vez se tiene la imagen convertida a escala de grises, antes de comenzar con la detección y búsqueda de contornos, se necesita hacer un desenfoque para mejorar los resultados.

Para lo que se utiliza el desenfoque principalmente es para disminuir el ruido de la imagen y sea mucho más fácil encontrar los contornos.

A continuación, se muestra una figura en la que aparece la imagen original, y la imagen de salida al realizar un desenfoque Gaussiano.



Figura 27: Imagen original



Figura 28: Imagen con desenfoque Gaussiano

En OpenCV, la manera más cómoda de realizar este desenfoque es con la función `cv2.GaussianBlur()`. A la cual se tiene que pasarle como parámetros la cantidad de píxeles adyacentes tanto en x como en y, conocido también como vecindad.

Incluyendo el píxel con el que se está trabajando, se tiene que elegir siempre vecindad impar, por ejemplo, como se ve en la imagen inferior, se está trabajando con una vecindad (3,3).

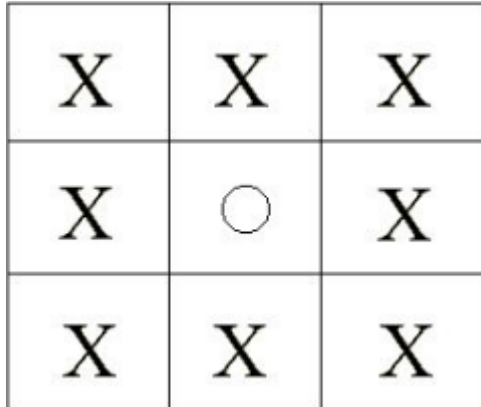


Figura 10: Vecindad 3x3

El otro parámetro que hay que pasarle es la desviación estándar gaussiana en la dirección x e y. Este parámetro no se ha utilizado ya que con un valor nulo se obtenía el resultado esperado.

4.3.3 Detección de contornos (CANNY CON BLUR Y SIN BLUR)

Lo que se necesita a continuación es encontrar en la imagen los contornos, esto es puntos en los que la diferencia de intensidad de un píxel y otro supere un cierto umbral.

OpenCV tiene una función que hace esto, pero primero se necesita quitar de la imagen lo que no sea contornos, esto se hace con la función:

`cv2.Canny()` a la cual hay que pasarle 3 parámetros: la imagen de partida, y los límites inferior y superior entre los cuales debe estar la diferencia de intensidad de los píxeles.

El problema radica ahora en buscar unos límites apropiados, para ello se hace uso de una función para que, en cada imagen, se calculen. Esta función recibe como parámetros: la imagen a la cual se quiere hacerle el tratamiento de convertirla en una imagen en la que solo se vean los contornos y un parámetro que permite ajustar la anchura del umbral. Un valor grande de este valor, deja un umbral amplio. Por otro lado, un valor pequeño, limita la anchura del umbral cosa que viene muy bien ya que, en este proyecto, las diferencias serán muy grandes. (Blanco contra negro).

Esta función lo que realiza es lo siguiente:

- 1- Calcula la mediana de los valores de intensidad de la imagen.
- 2- Calcula los límites inferior y superior en función de la mediana y el parámetro anteriormente introducido.
- 3- Usa la función `cv2.Canny()` con estos límites y saca como salida la imagen con los contornos.

En las siguientes imágenes se va a mostrar una comparación entre dos figuras: En una se muestran los contornos de la imagen inicial sin realizar el desenfoque y en la otra se muestran los contornos sobre una imagen que ha sido desenfocada previamente.

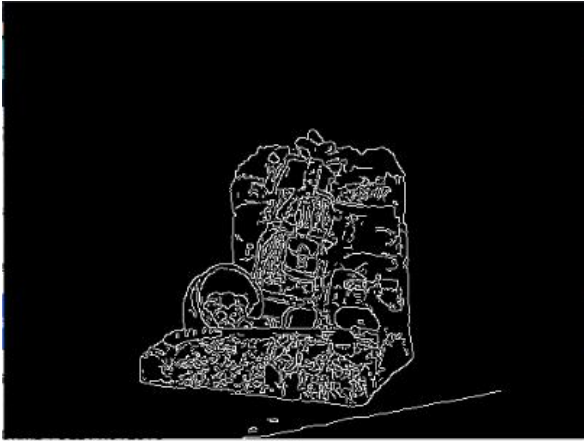


Figura 30: Contornos sin aplicar desenfoque



Figura 31: Contornos al aplicar desenfoque

Como se puede apreciar, al aplicar el desenfoque solo se observan solamente los contornos más llamativos con lo que el cálculo va a ser más eficiente y más preciso en este caso.

A continuación, con la función `cv2.findContours`, OpenCV devuelve los contornos en una variable.

Después los ordeno de mayor a menor, y solo cojo los 10 más grande.

Ahora hay que ver si alguno de los contornos se asemeja a un rectángulo de cuatro lados, para ello se tienen que seguir los siguientes pasos:

- 1- Calcular el perímetro del contorno. Para ello se usa la función `cv2.arcLength()` dándole como parámetros, el contorno al cual se quiere calcularle el perímetro y como segundo parámetro, `True`.
- 2- Aproximar el contorno a un polígono. Se usa la función `cv2.approxPolyDP()` a la cual hay que pasarle 3 parámetros. El primero de ellos es el contorno al cual se quiere hacerle la aproximación, el segundo es la máxima desviación del perímetro que puede realizar, por ejemplo, si se quiere convertir una curva en una recta, la distancia no es la misma. Para ello se ha calculado el perímetro.
- 3- Una vez se tenga la aproximación ya se puede ver si el contorno tiene 4 lados calculando la longitud de la aproximación con la función `len()`.
- 4- Se calcula el área del polígono con `cv2.contourArea()` y se descarta las que sean pequeñas.
- 5- Se hace un rectángulo alrededor del polígono y se recorta ese rectángulo dándole las dimensiones de las imágenes de muestra.

Una vez realizado esto, ya se tiene la imagen tomada por la cámara lista para ser comparada con la imagen de prueba.

4.4 Tratamiento de imágenes de muestra

En este apartado se van a explicar los pasos que hay que seguir para transformar las imágenes de muestra en imágenes para ser comparadas.

En resumen, hay que hacer:

1. **Leer y almacenar la imagen.**
2. **Convertir de RGB a escala de grises.**
3. **Desenfocar para suavizar bordes.**
4. **Tener imagen con bordes.**

4.4.1 Conversión de color

Como se explicó en el apartado anterior es necesario realizar esta conversión, aunque en el caso concreto de las imágenes de muestras que se usan no sería totalmente necesario ya que las imágenes están en blanco y negro desde el inicio.

Se ha dejado esta forma de tratar las señales por si se quieren ampliar las señales detectadas y alguna de ellas fuera en escala RGB.

Las señales usadas son las siguientes:

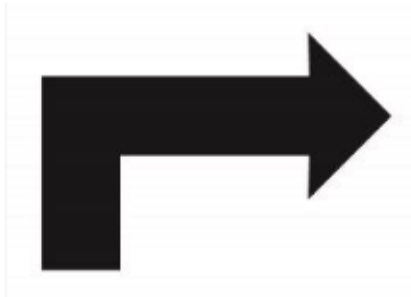


Figura 32: Señal giro derecha.

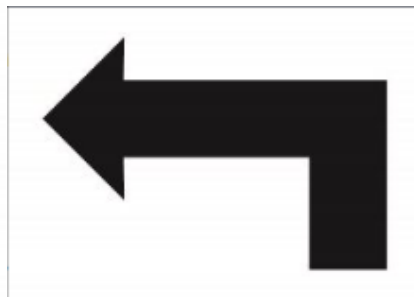


Figura 33: Señal giro izquierda.



Figura 34: Señal stop.

4.4.2 Desenfoque

Previamente se ha explicado la razón por la cual hay que usar la técnica de desenfoque.

A continuación, se muestran las señales desenfocadas:

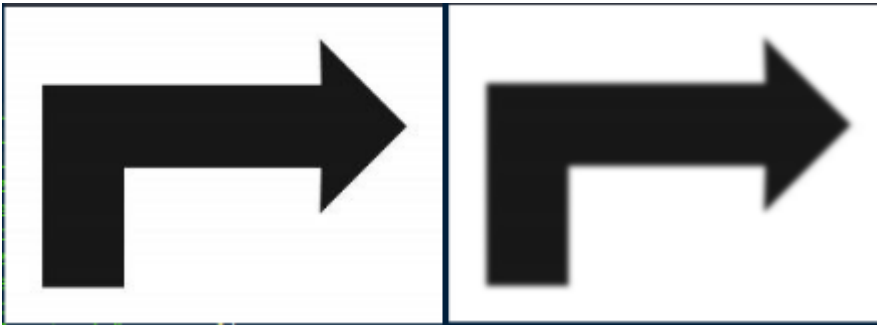


Figura 35: A la izquierda, señal derecha original. A la derecha señal derecha desenfocada.



Figura 36: A la izquierda, señal izquierda original. A la derecha señal izquierda desenfocada.

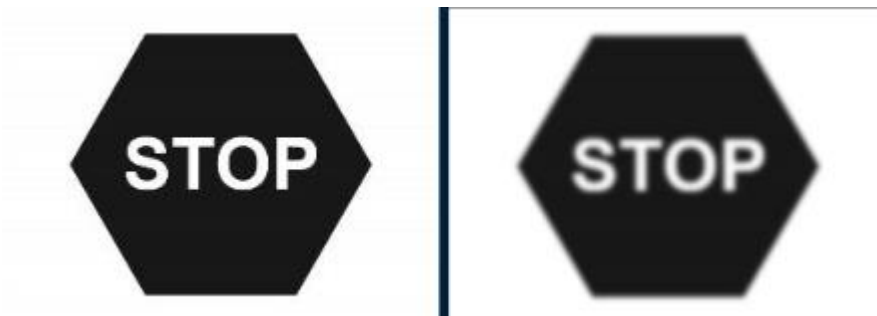


Figura 37: A la izquierda, señal stop original. A la derecha señal stop desenfocada.

4.4.3 Detección de contornos

Para la detección de contornos se usa la misma función que se explicó anteriormente, la función que convierte una imagen en una imagen con solo bordes, calculando los valores límites inferiores y superiores del umbral para cada señal.

A continuación, se pondrá una comparación entre las señales originales y las señales mostrando contornos:

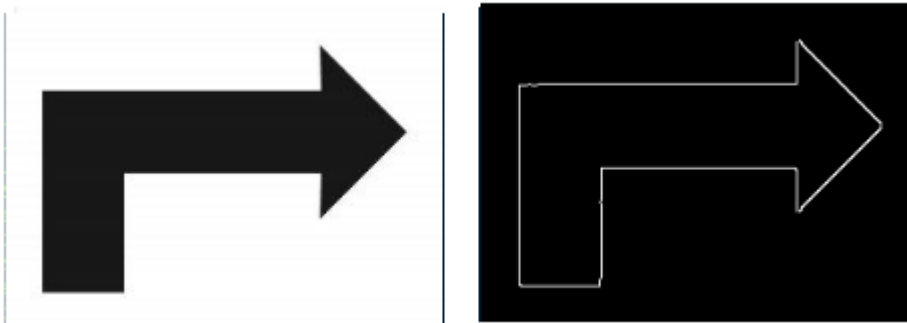


Figura 38: A la izquierda, señal derecha original. A la derecha señal derecha mostrando contornos.

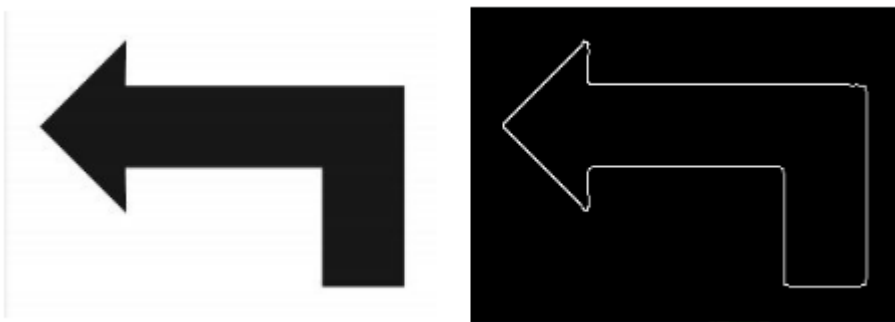


Figura 39: A la izquierda, señal izquierda original. A la derecha señal izquierda mostrando contornos.



Figura 40: A la izquierda, señal stop original. A la derecha señal stop mostrando contornos.

Una vez se tienen las imágenes con contornos, llega la hora de la comparación de las imágenes.

4.5 Comparación de imágenes

Existen distintas técnicas de comparación de imágenes. Las usadas en este proyecto son las siguientes, ya que son las que, buscando información, iban a cumplir mejor con la idea prevista inicialmente.

- 1- **Error Cuadrático Medio (MSE)**
- 2- **Índice de similitud estructural (SSIM)**

4.5.1 Error Cuadrático Medio

El error cuadrático medio de un estimador es el promedio de los errores al cuadrado, la diferencia entre el estimador y lo que se estima. Es muy usado en estadística.

En este caso se usará para ver las diferencias entre la imagen tomada por la cámara y la imagen de muestra.

$$MSE = \frac{1}{m \cdot n} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} [I(i, j) - K(i, j)]^2$$

Figura 41: Fórmula MSE.

Hay una función que se llama *mse* que sirve para calcular la diferencia entre dos imágenes, recibiendo como parámetros estas dos imágenes. Deben ser de igual tamaño.

Se aplica la fórmula y se saca el resultado.

Calculo el error cuadrático medio de cada señal con la imagen tomada por la cámara y el que tenga menor valor, es el correcto.

4.5.2 Índice de similitud estructural (ssim)

Este índice mide la similitud entre dos imágenes.

Fue usada inicialmente para calcular la calidad de la imagen emitida por televisión.

$$SSIM(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)}$$

Figura 42: Fórmula SSIM

Siendo

1. μ_j la media de los valores en el eje x o eje y.
2. σ_j La varianza de x o de y
3. σ_{ij} La covarianza de x e y.
4. c_i Un valor para estabilizar.

Este método forma parte de una librería llamada *skimage* creada por *scikit-image*. Con lo cual solo hay que llamar a la función *ssim()* y pasarle como parámetros las dos imágenes que se quieren comparar.

Para este proyecto, se ha preferido usar el *ssim* porque como se podrá ver es el que mejores resultados aporta.

Para contrastar lo que se ha dicho anteriormente, se expone un ejemplo en el cual se ha realizado un experimento comparando una imagen tomada por el robot previamente y aportada al programa con las señales de muestra.

Primero se va a mostrar la imagen tomada y la señal procesada y recortada (lista para ser comparada).



Figura 43: Imagen tomada por el robot.



Figura 44: Imagen procesada y recortada.

Una vez se tiene la imagen lista para ser comparada, se compara con las imágenes de la figura 32,33 y 34.

A continuación, se muestran los resultados de aplicar la técnica *mse* y *ssim* al comparar la imagen procesada con las imágenes de muestra.

```
M1:
3871.95462201

M2:
3753.92453401

M3:
4583.40679345

S1:
0.616284588556

S2:
0.645181809892

S3:
0.538743503856
Stop
```

Figura 45: Resultado MSE

```
M1:
3871.95462201

M2:
3753.92453401

M3:
4583.40679345

S1:
0.616284588556

S2:
0.645181809892

S3:
0.538743503856
Giro Derecha
```

Figura 46: Resultado SSIM

Siendo M1, M2 y M3, los resultados de aplicar la técnica **MSE** a las señales de giro izquierda, giro derecha y señal de STOP respectivamente.

Siendo S1, S2 y S3, los resultados de aplicar la técnica **SSIM** a las señales de giro izquierda, giro derecha y señal de STOP respectivamente.

Como se puede observar, la técnica que muestra el resultado correcto es la técnica SSIM ya que la imagen que se quiere detectar es la señal de giro a la derecha y es el que aparece al usar la comparación SSIM.

4.6 Conclusiones

Para cerrar este apartado me gustaría aportar mi opinión personal sobre el procesamiento de imágenes llevado a cabo en este proyecto.

Para empezar, hablando del software, se ha elegido la librería de uso libre OpenCV ya que tiene una amplia documentación en internet y nos permite conseguir los resultados deseados de una manera sencilla. Esta librería unida al lenguaje de programación Python hacen que el procesamiento de imágenes se vuelva más fácil. Desde una simple conversión de color a la detección y reconocimiento de una señal de tráfico.

El lenguaje Python he necesitado de aprenderlo para poder usarlo en este proyecto, pero no ha supuesto mucho problema ya que es un lenguaje muy sencillo, en comparación con el lenguaje C, no necesita ser compilado si no que puede ser ejecutado directamente.

En cuanto al hardware, se ha usado la placa Raspberry Pi ya que ha aportado muchos aspectos que me han sido útiles aprender y nos ofrecía todo lo deseado a un precio apropiado. Una de las cosas que más me ha gustado aprender gracias a la Raspberry Pi ha sido el uso del sistema operativo Raspbian. Hasta este momento, no había necesitado hacer uso de las grandes ventajas que el entorno Linux nos aporta a la hora de programar y la cantidad de opciones avanzadas de las que dispone. Nos permite un control casi total del hardware con lo que se aumenta la velocidad de procesado, algo que nos es muy útil en este proyecto.

5 ENVÍO Y RECEPCIÓN DE INFORMACIÓN

La nueva tecnología de la información, internet y el email, prácticamente han eliminado los costes físicos de la comunicación.

- Peter Drucker -

5.1 Esquema general

En la siguiente figura, se muestra la conexión que existe entre el módulo Arduino y el módulo Raspberry Pi.

La comunicación entre ambos módulos se lleva a cabo mediante radiofrecuencia a través de los módulos nrf24l01.

Tanto Arduino como la Raspberry Pi, se comunican con este módulo mediante el protocolo de comunicación SPI.

En este capítulo se va a explicar las características del módulo nrf24l01, las ventajas que aporta respecto a otras formas de comunicación. También se expondrá la manera en que se ha implementado tanto para usarlo en Arduino, como en Raspberry Pi.



Figura 47: Esquema General Envío y Recepción de información.

5.2 Conexión inalámbrica

En este apartado se va a explicar por qué se ha elegido una conexión inalámbrica, una breve introducción de lo que se puede hacer con esa comunicación. Se comentarán los aspectos que han sido claves para elegir esta tecnología y este módulo en lugar de otros que también podrían haber cumplido las funciones que se van a implementar.

5.2.1 Introducción

La tecnología inalámbrica consiste en comunicar dos o más nodos a través de ondas electromagnéticas.

En este caso se tienen dos nodos, uno sería el módulo Arduino y el otro es la Raspberry Pi. Aunque para ser más concretos el nodo en sí es el módulo nrf24l01 que se conecta por una parte en la Raspberry Pi y por otro lado en al Arduino.

Existen distintos tipos de conexiones inalámbricas, entre las que cabe destacar la tecnología WiFi, Bluetooth y la radiofrecuencia.

1. **WiFi:** Se rige bajo el estándar IEEE 802.11, ofrece una distancia máxima de comunicación alta. Actualmente trabaja en dos frecuencias: 2.4Ghz y 5GHz. Ultimamente se está optando por la frecuencia de 5GHz para evitar la alta congestión que existe con las frecuencias de 2.4GHz. Tiene un consumo elevado.
2. **Bluetooth:** Basado en el estándar IEEE 802.15.1, funciona a 2.4GHz. La distancia máxima de comunicación es pequeña en comparación con las otras tecnologías. Tiene un consumo reducido por lo que es muy utilizado para comunicar dispositivos, por ejemplo, los ratones y teclados inalámbricos usan Bluetooth, ya que al usar pilas o baterías conviene que tengan la máxima autonomía posible.
3. **Radio Frecuencia:** Puede usarse para transmitir a distancias muy grandes porque puede funcionar con frecuencias bajas. Tiene un consumo bajo, aunque aumenta proporcionalmente con la distancia de transmisión. Tiene una desventaja muy clara con los otros métodos y es que necesita una antena que, si es obstruida, pueden existir problemas de reconocimiento. Es por eso que se usa menos.

5.2.2 Módulo NRF24L01

Es posiblemente el módulo más económico que existe para comunicar microcontroladores.

Está basado en un chip Nordic de ultra bajo consumo. Es un transceptor, emisor y receptor integrados en el mismo aparato.

Tiene 8 pines:

- 1- Pin de Tierra o **GND**
- 2- Pin de Tensión. **3.3 Voltios**.
- 3- **CE**. Pin usado para la radio.
- 4- **CSN**. Pin usado para la radio.
- 5- **SCK**. Pin de la transmisión SPI.
- 6- **Mosi**. Pin de la transmisión SPI.
- 7- **Miso**. Pin de la transmisión SPI.
- 8- **IRQ**. Sirve para informar al master SPI si se ha completado la transmisión o recepción.

En la figura siguiente, se muestra una imagen del módulo nrf24l01 con su pinout.

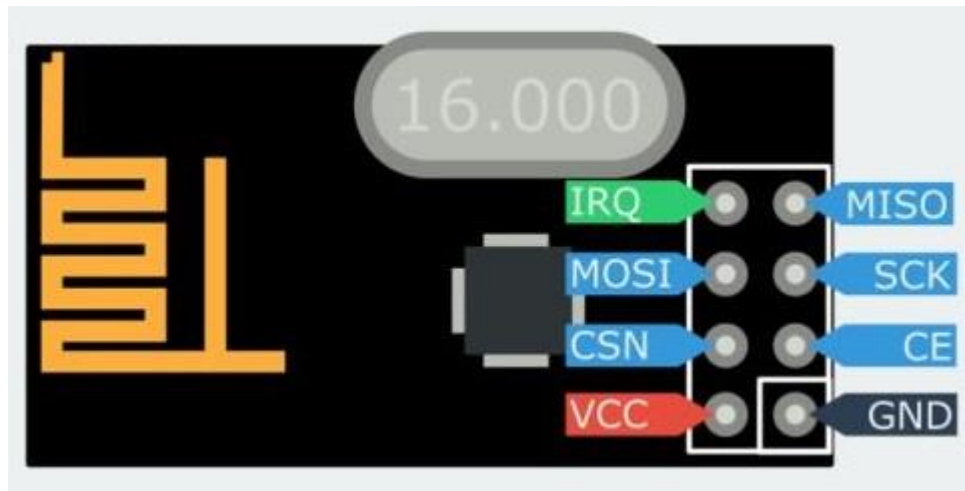


Figura 48: PINOUT nrf24l01.

Para poder identificar los pines, se puede mirar en un rectángulo que rodea al pin de tierra. Se va a conectar los nrf24l01 por puerto SPI tanto a la Raspberry Pi como a Arduino.

5.2.3 SPI

SPI es un estándar de comunicaciones síncrono entre dispositivos electrónicos.

Permite alcanzar velocidades altas de comunicación. Se usa normalmente para comunicar un microcontrolador con distintos periféricos.

Es un método síncrono, no se necesita pactar entre los periféricos la velocidad de comunicación ya que se tiene un hilo, que es el reloj, que se encarga de sincronizar todos los periféricos con el microcontrolador.

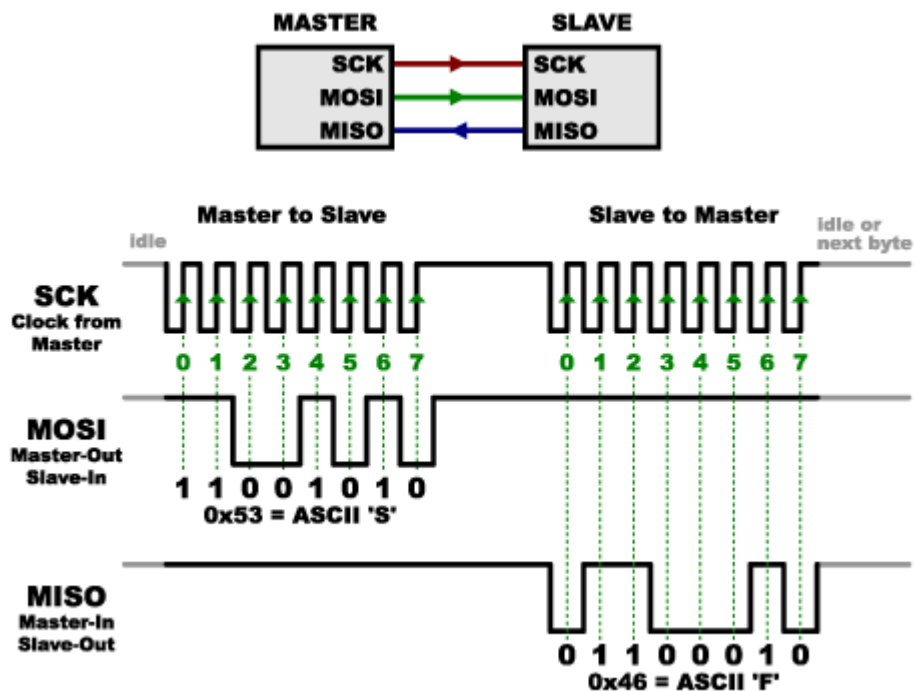


Figura 49: Esquema SPI.

Está formado por 3 hilos principalmente. El cuarto hilo se llama Slave Select, y se usa si hay más de un periférico con el que se quiere comunicar. No afecta en este caso.

Los tres hilos principales son:

- 1- **Reloj.** Puede encontrarse como SCK o CLK. El que genera el reloj es el Master o maestro. Es quien dirige la comunicación.
- 2- **MOSI.** Master Output, Slave Input. Es por la línea por la cual envía datos el maestro al esclavo.
- 3- **MISO.** Master Input, Slave Output. Línea por la cual el esclavo envía al maestro.

Tiene ventajas respecto a otros estándares de comunicación:

1. Conformar una comunicación full-duplex, envía y recibe, por lo que la velocidad aumenta.
2. Más rápido y menor consumo que I2C.

Obviamente tiene desventajas:

- Necesita más pines que I2C.
- Máster único.
- El esclavo no da señal de confirmación directamente.

5.3 Implementación en Raspberry Pi

5.3.1 Montaje

En este proyecto, se ha usado una Raspberry Pi 2. En la siguiente figura, se muestra el pinout de la Raspberry Pi 2.

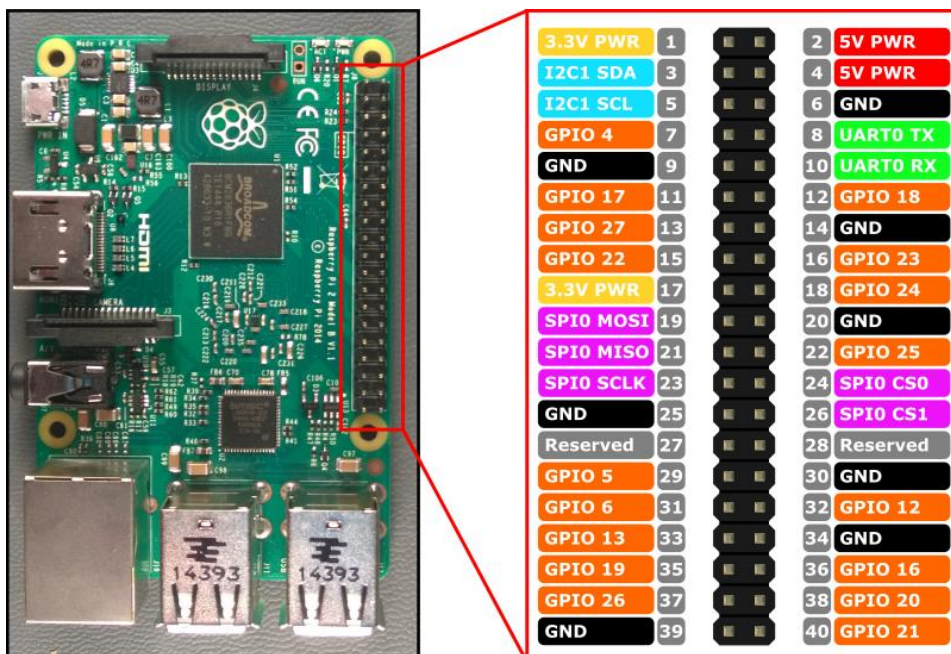


Figura 50: PINOUT Raspberry Pi 2.

Como se comentó anteriormente, se va a usar SPI para conectar la Raspberry Pi y el nrf24l01, por lo tanto, fijándonos en la imagen anterior, la conexión entre pines es la siguiente:

- El pin GND del nrf24l01 a cualquier tierra de la Raspberry Pi. Para el proyecto se ha usado el pin 6 de la Raspberry Pi.
- El pin de tensión 3.3V del nrf24 al pin 1 de la Raspberry Pi.
- El pin CE del nrf24l01 al pin 15 de la Raspberry Pi que se corresponde con GPIO 22.
- El pin CSN del nrf24l01 al pin 24 de la Raspberry Pi que se corresponde con SPI0 CS0.
- El pin SCK del nrf24l01 al pin 23 de la Raspberry Pi que se corresponde con SPI0 SCLK.
- El pin MISO del nrf24l01 al pin 21 de la Raspberry Pi que se corresponde con SPI0 MISO.
- El pin MOSI del nrf24l01 al pin 19 de la Raspberry Pi que se corresponde con SPI0 MOSI.
- El pin IRQ del nrf24l01 no se conecta a la Raspberry Pi, no hace falta.

5.3.2 Instalación de la librería

La librería usada se llama RF24 y ha sido creada por tmrh20 y compartida mediante su repositorio oficial de Github.

Para poder usarla hay que seguir ciertos pasos:

1. Primero de todo se necesita activar el puerto SPI en la raspberry, esto se hace usando el comando *sudo raspi-config*. Una vez dentro de ese menú pulsar en *avanzado* y activar el SPI.
2. A continuación, hay que actualizar el software y librerías de la raspberry con la combinación: *sudo apt-get update* y *sudo apt-get upgrade*.
3. Se descarga la librería del repositorio de tmrh20.
4. Se descomprime si es necesario.
5. Se entra en la carpeta y se usa el siguiente comando: *./configure --driver=SPIDEV*
6. Una vez configurado, se procede a instalar la librería usando: *sudo make install -B*.
7. Al instalar la librería ya se puede importar la librería y empezar a usarla.

5.3.3 Código

En este subapartado, se va a exponer el pseudocódigo necesario para cumplir con las siguientes ideas:

- Crear una radio en la Raspberry Pi
- Conectar la radio de la Raspberry Pi a la radio en Arduino, estableciendo un canal bidireccional.
- Enviar los datos que necesitamos.
- Comprobar que todo está correcto.

Para ello se sigue el siguiente procedimiento:

- *Importación de librerías. (Se necesita importar la librería RF24 y la librería GPIO para usar el puerto SPI de la Raspberry Pi).*
- *Se crea la radio en los pines indicados anteriormente con CE y CSN. (22,0)*
- *Se define la pasarela a usar. Debe ser la misma en Raspberry Pi como en Arduino, porque si no, no existe comunicación entre ambos módulos.*

- *Se dan valores estándar a parámetros que se necesitan, como pueden ser: Tamaño máximo del paquete, tamaño mínimo, etc.*
- *Se enciende la radio.*
- *Se fijan el número de intentos por si la comunicación falla.*
- *Se abre la pasarela para leer y para escribir. Esto permite tener un canal bidireccional, se puede tanto recibir como enviar información. Es de mucha utilidad para poder comprobar que no existen errores en el envío.*
- **Comienza un bucle.** *Aquí volverá el flujo del programa cuando se haya enviado y esté todo correcto o haya habido un error en la transmisión.*
- *Una vez se tienen los datos listos para ser enviados, se para el modo de escucha (leer), se carga el mensaje que se quiere enviar y lo se envía por la pasarela. Se activa el modo de escucha y se activa también un temporizador.*
- *Ahora pueden pasar dos cosas:*
 - 1- *Se termina el tiempo del temporizador, por lo tanto, puede que no haya llegado el mensaje al Arduino, o que no se haya enviado (o recibido por la raspberry) la confirmación del mensaje. Por lo tanto, se muestra un error y se vuelve al principio del bucle.*
 - 2- *Llega la confirmación por parte del Arduino y se muestra un mensaje para verificarlo. El programa vuelve al bucle.*
- *El programa nunca debería llegar a este punto ya que debería quedarse en el bucle hasta que se cierre de forma inesperada.*

Este sería la forma de actuar de este módulo en la Raspberry Pi.

A continuación, se explicará la forma de implementarlo en Arduino y seguidamente se pasará a exponer algún experimento.

5.4 Implementación en Arduino

5.4.1 Montaje

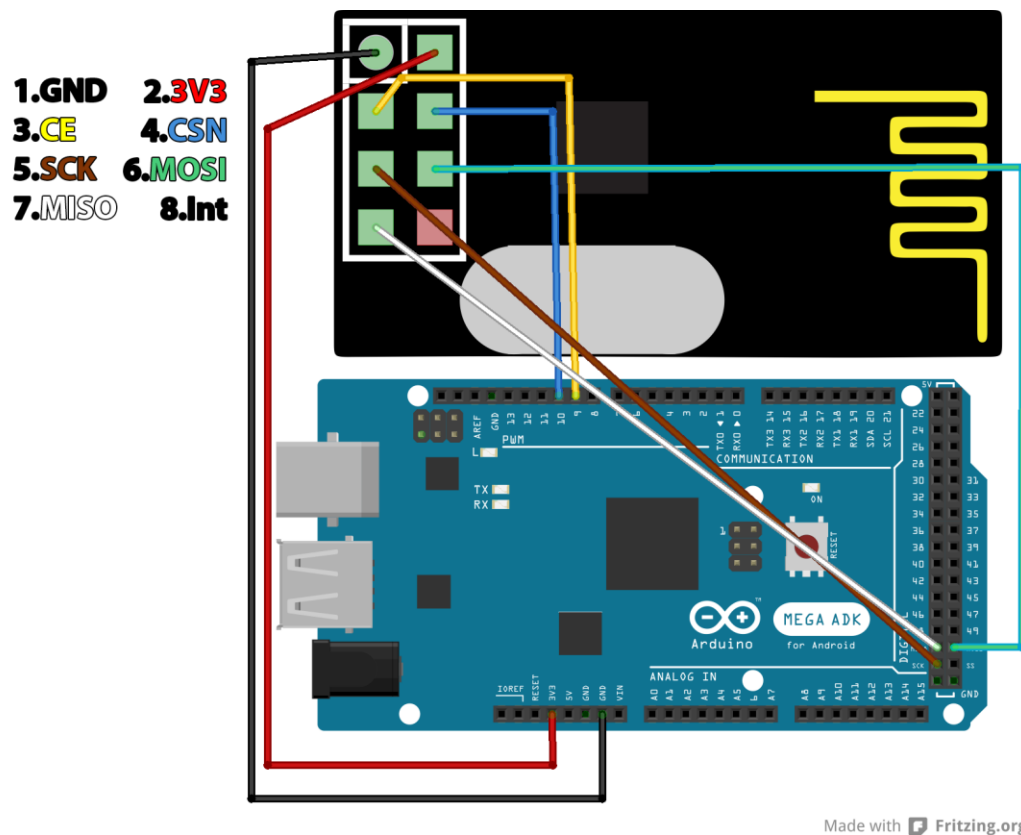


Figura 51: Conexión Arduino Mega con nrf24l01.

En la figura anterior, se muestra cuales son las conexiones entre el Arduino Mega

En resumen, la conexión es la siguiente:

- El pin GND del nrf24l01 se conecta a un pin de tierra del Arduino.
- El pin 3.3V del nrf24l01 al pin de 3.3V de Arduino.
- El pin CE del nrf24l01 se conecta en el pin 22 del Arduino.
- El pin CSN del nrf24l01 se conecta en el pin 23 de Arduino.
- El pin SCK del nrf24l01 se conecta al pin 52 de Arduino.
- El pin MISO del nrf24l01 se conecta al pin 50 de Arduino.
- El pin MOSI del nrf24l01 se conecta al pin 51 de Arduino.
- El pin IRQ del nrf24l01 se deja sin conectar.

5.4.2 Instalación de la librería

La librería usada es una librería de uso libre, publicada por su autor en su repositorio oficial, tmrh20, llamada RF24. Es la misma que se utiliza para usar el módulo nrf24l01 en la Raspberry Pi. La única salvedad es que en Raspberry Pi se programa en Python y en este caso se programa en lenguaje C.

Para instalar la librería se sigue el procedimiento habitual usado para instalar librerías en Arduino, que es el siguiente:

1. Se descarga la librería.
2. Se descomprime si es necesario.
3. Se copia la carpeta de la librería y se pega en la carpeta Arduino\libraries
4. Si se tiene el programa Arduino abierto, hay reiniciarlo para que se cargue bien la librería.
5. Una vez hecho esto, se necesita importarla con un `#include` “nombre de la librería”, en este caso `#include “RF24.h”` y ya se terminaría la instalación de la librería.

5.4.3 Código

En este subapartado, se va a exponer el pseudocódigo necesario para cumplir con las siguientes ideas:

- Crear una radio en el Arduino.
- Conectar la radio de la Raspberry Pi a la radio en Arduino, estableciendo un canal bidireccional.
- Recibir los datos que necesitamos.
- Enviar código de confirmación.

Para lo que se sigue el siguiente procedimiento:

- *Importación de librerías. (Se necesita importar la librería RF24 y la librería SPI, que viene instalada por defecto).*
- *Se crea la radio en los pines indicados anteriormente con CE y CSN. (22,23)*
- *Se define la pasarela a usar. Debe ser la misma en Raspberry Pi como en Arduino, porque si no, no existe comunicación entre ambos módulos.*
- *Se dan valores estándar a parámetros que se necesitan, como pueden ser: Tamaño máximo del paquete, tamaño mínimo, etc.*
- *Se enciende la radio.*
- *Se fijan el número de intentos por si la comunicación falla repetidamente.*
- *Se abre la pasarela para leer y para escribir. Esto permite tener un canal bidireccional, se puede tanto recibir como enviar información. Es de mucha utilidad para poder comprobar que no existen errores en el envío.*
- **Comienza un bucle.** *Aquí volverá el flujo del programa cuando se haya recibido bien, o haya habido algún problema.*
- *Se queda a la espera de un mensaje.*

- Cuando llega algún mensaje a la pasarela, se lee, se almacena.
- Se detiene el modo de escucha.
- Se envía la confirmación de llegada de ese mensaje.
- Dependiendo del mensaje recibido se va a realizar una acción u otra.
- Una vez enviada la confirmación de llegada del mensaje, se vuelve a encender el modo de escucha.
- Vuelve al bucle.
- Si todo va bien, siempre debería de funcionar el bucle.

5.5 Experimento

5.5.1 Comprobación de conexiones y librería.

Es muy fácil tener problemas de conexiones ya que son muchos pines los que hay que conectar, los cables pueden tener deformaciones internas y puede existir cierta interferencia.

Para comprobar que todo esté correcto y que tenía la librería funcionando, se hizo un programa muy sencillo, incluyendo las funciones que más tarde serían de utilidad.

El programa consiste en elegir que rol que se quiere en cada módulo, pudiendo elegir entre emisor o receptor. El emisor envía un dato, y si ese dato es el mismo que está esperando el receptor, éste imprimirá por pantalla el dato recibido y la palabra: Correcto. Si existiese algún problema, responde con Timeout o diciendo que no ha recibido lo esperado.

Para probarlo da igual que rol elijas, en la prueba que se ha realizado para esta memoria, se ha usado el Arduino como Receptor, y la Raspberry Pi como Emisor:

```
got response size=3 value="345"
Now sending length 3
got response size=3 value="345"
Now sending length 3
got response size=3 value="345"
Now sending length 3
got response size=3 value="345"
Now sending length 3
got response size=3 value="345"
Now sending length 3
got response size=3 value="345"
Now sending length 3
got response size=3 value="345"
Now sending length 3
```

Figura 52: Envío de información desde la Raspberry Pi.

```
Recibo dato de longitud=3 valor=345
Correcto
Respuesta enviada.
Recibo dato de longitud=3 valor=345
Correcto
Respuesta enviada.
Recibo dato de longitud=3 valor=345
Correcto
Respuesta enviada.
Recibo dato de longitud=3 valor=345
Correcto
Respuesta enviada.
Recibo dato de longitud=3 valor=345
Correcto
Respuesta enviada.
```

☒ Autoscroll

Figura 53: Respuesta del Arduino.

En la figura 52 se muestra lo que envía la Raspberry Pi. En la figura 53 se muestra lo recibido por Arduino.

5.6 Conclusiones

Para concluir este capítulo me gustaría añadir mi valoración personal respecto a este módulo y por qué lo elegí en detrimento de otros métodos.

En un principio probé usando una comunicación más sencilla que consistía en unir la Raspberry Pi con el Arduino mediante un cable USB. Durante un tiempo fue la opción que iba a mantener ya que no se me ocurría otra forma de comunicarlos. Este método resultaba teóricamente sencillo ya que solo necesita abrir un canal, como un puerto serie y empezar a enviar datos.

Finalmente descarté este método porque no me resultaba fácil la integración de todos los elementos en la plataforma robótica y por otro lado el puerto USB del Arduino tenía pensado usarlo para alimentación. Cosa que también descarté porque no me fue necesario.

Por lo tanto, me puse a indagar y encontré que existía un módulo llamado Xbee que resuelve el problema de la conexión inalámbrica de forma muy sencilla, pero tienen un inconveniente restrictivo en este caso. El precio excedía el presupuesto que tenía pensado para esta tarea.

Pensé en volver al método alámbrico cuando encontré que había un módulo de precio muy bajo que podría cumplir las especificaciones. Es aquí donde entra en juego el módulo nrf24l01. Este aparato como se explicó anteriormente es muy sencillo de utilizar, tiene bastantes referencias en la web y cumplía de sobras con los requerimientos que yo pedía.

No he tenido ningún problema con este módulo, creo que puede tener mucho futuro por el precio y las opciones que ofrece.

6 NAVEGACIÓN AUTÓNOMA

El verdadero progreso es el que pone la tecnología al alcance de todos.

- Henry Ford -

Este proyecto está pensado para ofrecer una serie de módulos que se pueden combinar para conseguir unos resultados más ambiciosos. Estas aplicaciones son las que se han estado explicando detalladamente en los apartados anteriores:

- Por un lado, se tiene una base robótica con todos los sensores y actuadores explicados detalladamente en el apartado 3.
- Por otro lado, se dispone de un módulo que realiza reconocimiento de imágenes teniendo unas imágenes de muestra.
- Por último, se tiene una forma de comunicar ambos dispositivos.

En este proyecto se han unido estos tres módulos para obtener una base robótica que reconozca 3 señales de tráfico y realice unos movimientos impuestos para cada una de las señales.

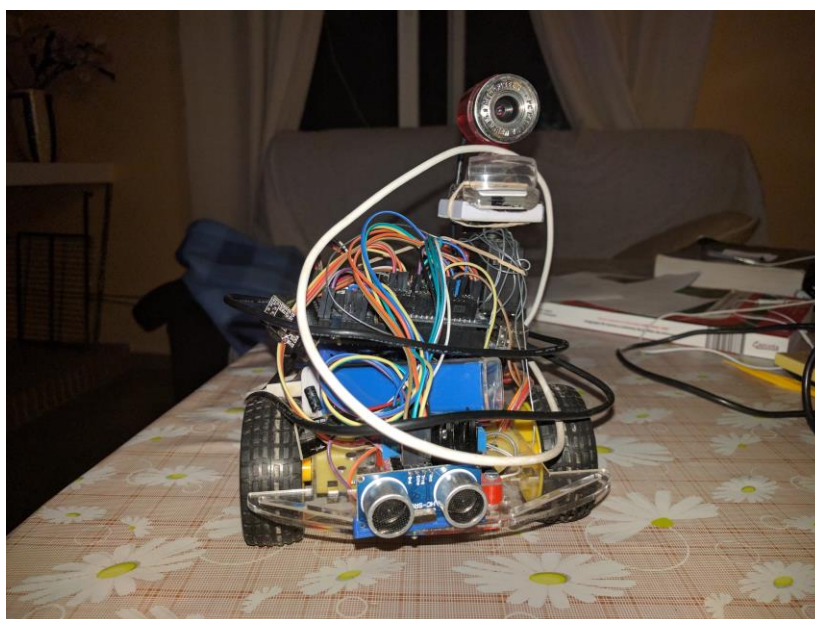


Figura 54: Imagen frontal del sistema completo.

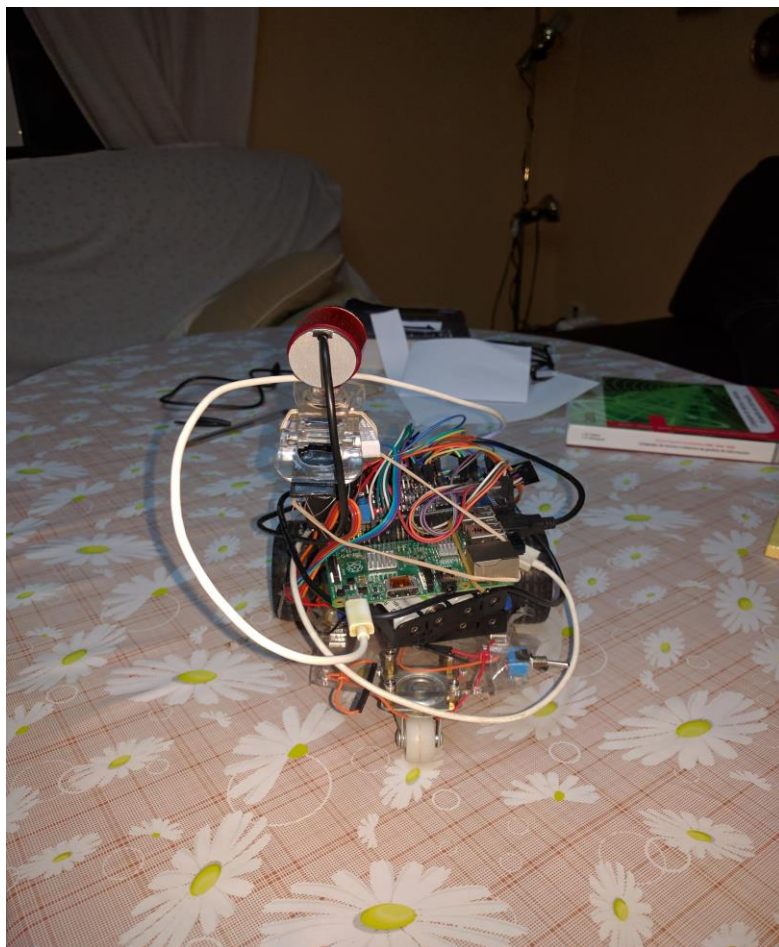


Figura 55: Imagen trasera del sistema completo.

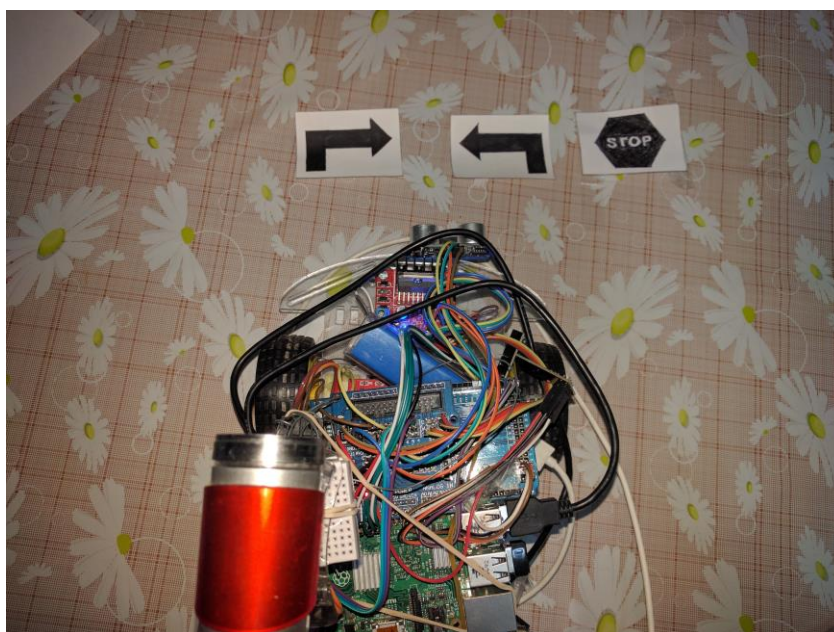


Figura 56: Comparación longitud señales con robot.

6.1 Explicación del experimento.

El experimento consiste en lo siguiente:

El robot empezará en un punto de partida, avanzará hasta encontrar un obstáculo de frente, ese obstáculo contendrá una señal que debe ser reconocida. Una vez reconocida la señal se procesa, compara y envía desde la Raspberry Pi al Arduino y este ordena que se realice un movimiento de cambio de dirección: o girar a la derecha o girar a la izquierda. En cuanto encuentre una señal de STOP el experimento habrá concluido.

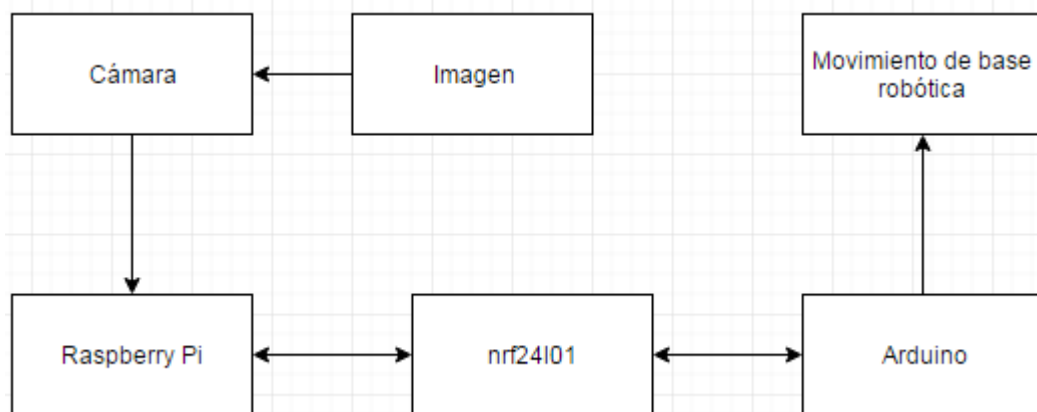


Figura 57: Diagrama de bloques del proyecto.

Por lo tanto, se podría resumir en los siguientes pasos:

- El vehículo avanza hasta toparse con un obstáculo situado en un rango fijado.
- La cámara toma una imagen del entorno.
- La Raspberry Pi recibe esa imagen de la cámara, la procesa y cuando distingue a que señal corresponde la envía a través del nrf24l01.
- El Arduino es quien recibe la información.
- Dependiendo de que señal es la recibida, se realiza un movimiento u otro.

A continuación, se detallarán los pasos anteriores.

6.1.1 Reconocimiento de obstáculo frontal

En este primer paso, el robot se encuentra en una posición inicial y su cometido es avanzar en línea recta como si estuviese en una carretera.

Una vez se encuentra a una distancia que elegiremos, se detendrá.

Para este paso solo es necesario el módulo de Arduino, con sus sensores y actuadores.

Inicialmente, el robot estará parado en una posición inicial, a continuación, se comienza a medir con los sensores de ultrasonido hasta que se encuentre en un rango de distancia en los cuales el reconocimiento de la imagen es mejor: entre 30 y 45 centímetros. Si se encuentra a una distancia menor, el robot se moverá hacia atrás hasta llegar a este rango, si la distancia es mayor, se moverá

hacia adelante para entrar en ese rango. Una vez dentro del rango apropiado, se activan las variables pertinentes para poder recibir la información de que señal se ha reconocido.

El pseudocódigo de este paso sería el siguiente (en Arduino):

- *Se inician todas las configuraciones pertinentes*
- ***Comienza un bucle*** *que no acaba hasta que el robot no está en el rango apropiado*
- *Se lee el sensor ultrasonido.*
- *Si la lectura del sensor está entre 30 y 45 cm, se activa la variable que permite salir del bucle.*
- *Si la lectura del sensor es menor a 30 cm, el robot se moverá un poco hacia atrás.*
- *Si la lectura es mayor de 45 cm, el robot se moverá un poco hacia adelante.*
- *Vuelta al bucle.*
- *Una vez el robot se encuentre en el rango, se sale del bucle.*
- *Se activan las variables necesarias para recibir señales.*



Figura 58: Robot parado a la distancia elegida.

6.1.2 Procesamiento de la imagen y envío al nrf24l01

Realmente, el procesamiento de imágenes se empieza desde que se inicia el programa en la Raspberry Pi, aunque los resultados son desechados hasta que el robot no se encuentra en su sitio.

Una vez que el robot se encuentra en su sitio, las señales ya tienen un destino.

El pseudocódigo para este paso sería el siguiente (Raspberry Pi):

- *Se hace una captura con la cámara.*
- *Se almacena, se le realiza el procesamiento de imágenes (conversión a escala de grises, desenfoque, detección de contorno y recorte).*
- *Se compara con las 3 señales de muestra. Mediante la técnica SSIM.*
- *Se envía un mensaje con el valor de la señal leída al módulo nrf24l01.*

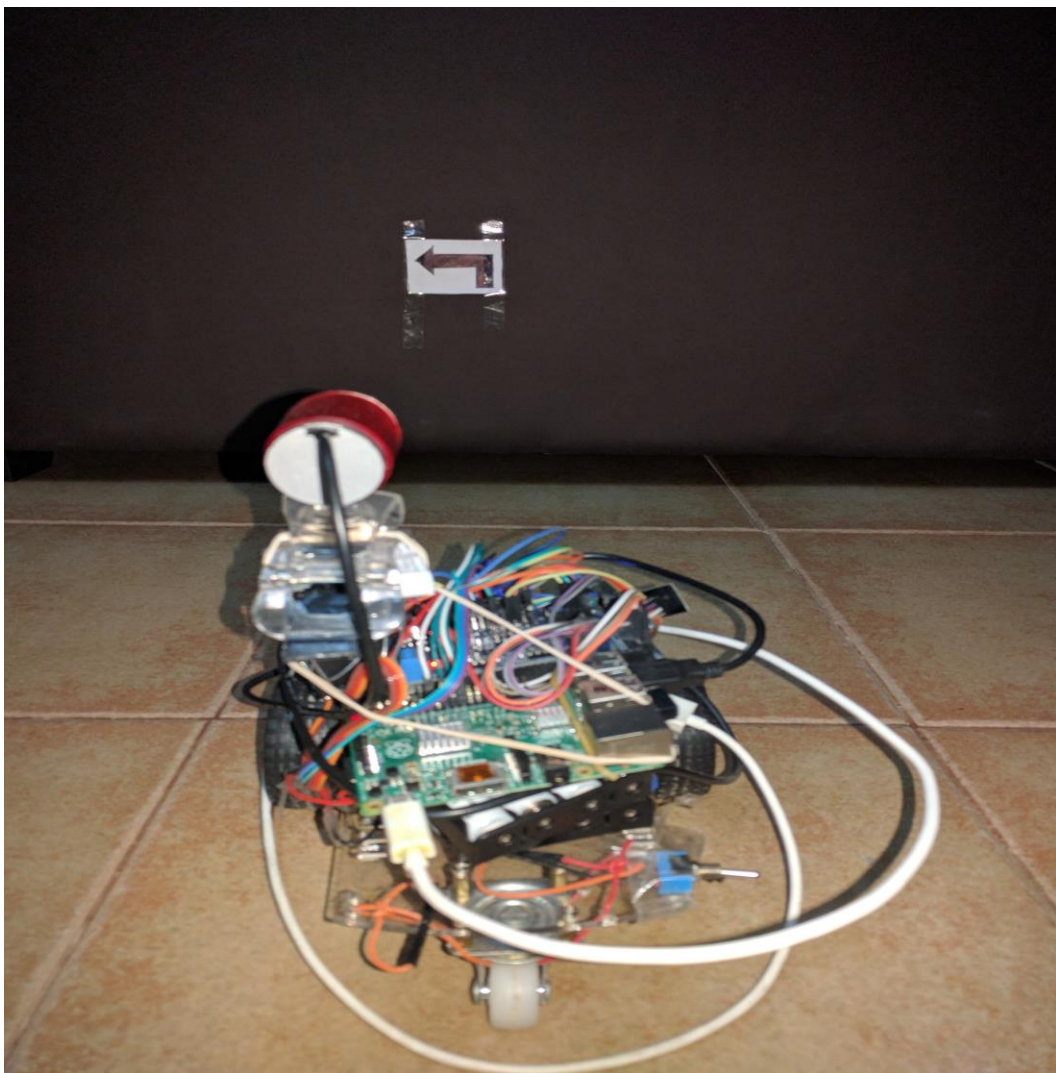


Figura 59: Robot parado, capturando imagen y procesándola.

6.1.3 Recepción de señal y movimiento.

Una vez que el robot está en su sitio, las señales reconocidas por la Raspberry Pi mediante el nrf24l01 son recibidas por el Arduino. Este descifra que señal es y realiza un movimiento o termina el programa.

El pseudocódigo realizado en este paso es el siguiente (Arduino):

- *Una vez activada la señal de que el robot está situado en el lugar apropiado, se pone el módulo nrf24l01 en modo escucha.*
- ***Comienza un bucle*** del cual no se sale hasta que no se recibe un mensaje mediante el nrf24l01.
- *Una vez recibido el mensaje, se sale del bucle y se deja de esperar datos por el módulo de comunicación, se apaga el modo escucha.*
- *Se lee el mensaje y se descifra que señal es la que se ha recibido.*
- *Si la señal es giro a la izquierda, se realiza un giro de 90° a la izquierda.*
- *Si la señal es giro a la derecha, se realiza un giro de 90° a la derecha.*
- *Si la señal recibida es la señal de STOP, el programa acaba.*
- *Si no es la señal de STOP, la variable que indicaba que el robot estaba en el sitio adecuado se desactiva.*
- *Se volvería al primer paso.*

Una vez concluido este paso, se volvería a empezar desde el paso de colocar el robot en el sitio apropiado.



Figura 60: Robot girado tras recibir la orden de girar a la izquierda.

6.2 Conclusión del experimento.

Esta aplicación englobaría todos los aspectos que se explican en este proyecto.

Partiendo de este experimento y teniendo los módulos expuestos anteriormente, se podrían realizar distintas aplicaciones de navegación autónoma. Algunos experimentos que se me ocurren directamente serían los siguientes:

- Añadir un servomotor para dotar a la cámara de una visión panorámica.
- Ampliar el número de señales.
- Cambiar las funciones de movimiento al recibir una señal.

Los problemas principales que se dan en este experimento son los siguientes:

- El problema de la odometría en los giros es crítico. Se necesita mucha precisión en las vueltas que da el encóder para poder tener una odometría algo más precisa en los giros. Esto conlleva cuelgues por sobreprocesamiento del Arduino. Aunque se pudiese resolver el problema de la precisión del encóder, seguiríamos teniendo problemas de odometría ya que las ruedas pueden deslizarse y el problema se complicaría exponencialmente.
- Otro problema igual de importante y que tiene relación con el problema de la odometría es el problema de la alimentación. Para este proyecto se han usado 8 pilas de 1.2V recargables. El problema de usar esta alimentación es que se nota la diferencia de unas pilas recién cargadas a unas pilas usadas. Lo que influye directamente en el movimiento del robot, ya que, si las pilas están cargadas totalmente, el movimiento del robot es fluido. Sin embargo, si las pilas llevan algún tiempo de uso, hay veces que se necesita una pequeña ayuda para que las ruedas comiencen a moverse.

ÍNDICE DE FIGURAS

Figura 1: Ejemplo de vehículo autónomo.	15
Figura 2: Esquema general del proyecto.	16
Figura 3: Elementos estructurales de un manipulador robótico.	22
Figura 4: Robot limpiador de la marca Roomba.	23
Figura 5: Esquema general Automatización Robot.	24
Figura 6: Arduino Mega.	25
Figura 7: Arduino Sensor Shield.	25
Figura 8: Sensor ultrasonido HC-SR04.	26
Figura 9: Puente H L298N.	26
Figura 10: Kit Robótica.	27
Figura 11: Kit Robótica Montado.	27
Figura 12: Conexiones Puente H L298N.	28
Figura 13: Conexiones HC-SR04.	30
Figura 14: Encóder digital.	31
Figura 15: Tiempo, vueltas de encoder y velocidad.	32
Figura 16: Experimento Medidas.	33
Figura 17: Valores en Régimen Permanente de las medidas.	34
Figura 18: Curva auxiliar para calcular valores en las medidas.	35
Figura 19: Esquema Simulink Modelo Matemático.	36
Figura 20: Comparación Medidas con Modelo Matemático (aproximado).	36
Figura 21: Esquema Simulink. Modelo con Controlador (arriba). Referencia (abajo).	37
Figura 22: Gráfico para comparar la salida del sistema con controlador y de la referencia.	38
Figura 23: Esquema general del tratamiento de imágenes.	42
Figura 24: Raspberry Pi 2.	42

Figura 25: Imagen tomada en escala RGB.	44
Figura 26: Imagen convertida a escala de grises.	44
Figura 27: Imagen original.	44
Figura 28: Imagen con desenfoque Gaussiano.	44
Figura 29: Vecindad 3x3.	45
Figura 30: Contornos sin aplicar desenfoque.	46
Figura 31: Contornos al aplicar desenfoque.	46
Figura 32: Señal giro derecha.	47
Figura 33: Señal giro izquierda.	47
Figura 34: Señal stop.	47
Figura 35: A la izquierda, señal derecha original. A la derecha señal derecha desenfocada.	48
Figura 36: A la izquierda, señal izquierda original. A la derecha señal izquierda desenfocada.	48
Figura 37: A la izquierda, señal stop original. A la derecha señal stop desenfocada.	48
Figura 38: A la izquierda, señal derecha original. A la derecha señal derecha mostrando contornos.	49
Figura 39: A la izquierda, señal izquierda original. A la derecha señal izquierda mostrando contornos.	49
Figura 40: A la izquierda, señal stop original. A la derecha señal stop mostrando contornos.	49
Figura 41: Fórmula MSE.	50
Figura 42: Fórmula SSIM.	50
Figura 43: Imagen tomada por el robot.	51
Figura 44: Imagen procesada y recortada.	51
Figura 45: Resultado MSE.	51
Figura 46: Resultado SSIM.	51
Figura 47: Esquema General Envío y Recepción de información.	53
Figura 48: PINOUT nrf24l01.	55
Figura 49: Esquema SPI.	55
Figura 50: PINOUT Raspberry Pi 2.	56
Figura 51: Conexión Arduino Mega con nrf24l01.	59
Figura 52: Envío de información desde la Raspberry Pi.	61
Figura 53: Respuesta del Arduino.	61

Figura 54: Imagen frontal del sistema completo.....	63
Figura 55: Imagen trasera del sistema completo.....	64
Figura 56: Comparación longitud señales con robot.	64
Figura 57: Diagrama de bloques del proyecto.	65
Figura 58: Robot parado a la distancia elegida.	66
Figura 59: Robot parado, capturando imagen y procesándola.	67
Figura 60: Robot girado tras recibir la orden de girar a la izquierda.	68

ÍNDICE DE TABLAS

Tabla 1: Combinaciones Puente H L298N.	29
Tabla 2: Características HC-SR04.	29

BIBLIOGRAFÍA

- [1] Self balancing Raspberry Pi OpenCV image processing Robot. <http://roboticssamy.blogspot.com.es/>
- [2] Página con muchos tutoriales de OpenCV con lenguaje Python. <http://www.pyimagesearch.com>
- [3] Librería para usar el nrf24l01 en la Raspberry Pi. <http://tmrh20.github.io/RF24/RPi.html>
- [4] Librería para usar el nrf24l01 en Arduino. <http://tmrh20.github.io/RF24/Arduino.html>
- [5] Recurso con múltiples tutoriales para Arduino. <http://www.prometec.net/>
- [6] Librería NewPing en Arduino. <http://playground.arduino.cc/Code/NewPing>