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

# Formato de Publicación de la Escuela Técnica Superior de Ingeniería de Sevilla

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

# Formato de Publicación de la Escuela Técnica Superior de Ingeniería de Sevilla

Autor:
Francisco Márquez Chaves

Tutor:
Federico Cuesta Rojo
Profesor titular

Dep. de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2016

Proyecto Fin de Carrera: Formato de Publicación de la Escuela Técnica Superior de Ingeniería de Sevilla

Autor: Pablo Aguilera Bonet

Tutor: Javier Payán Somet

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

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2013

El Secretario del Tribunal

*A mi familia*

*A mis maestros*

# Agradecimientos

Los estilos adoptados por nuestra Escuela y utilizada en este texto es una versión y adaptación a Word® del la versión $L^AT_EX$ que el Prof. Payán realizó para un libro que desde hace tiempo viene escribiendo para su asignatura. Por ello, la Escuela le está agradecida. Por otro lado, la adaptación se hizo sobre un formato que el prof. Aguilera arregló, basándose en su tesis doctoral. Su aportación ha sido muy relevante para que este formato vea la luz. Esta adaptación la llevamos a cabo el alumno Silvio Fernández, becario del Centro de Cálculo, y yo mismo, sobre un trabajo preliminar del alumno Julián José Pérez Arias.

A esta hoja de estilos se le incluyó unos nuevos diseños de portada. El diseño gráfico de las portadas para proyectos fin de grado, carrera y máster, está basado en el que el prof. Fernando García García, de la Facultad de Bellas Artes de nuestra Universidad, hiciera para los libros, o tesis, de la sección de publicación de nuestra Escuela. Nuestra Escuela le agradece que pusiera su arte y su trabajo a nuestra disposición.

*Juan José Murillo Fuentes*

*Subdirección de Comunicaciones y Recursos Comunes*

*Sevilla, 2013*

# Resumen

El presente proyecto expone el diseño y desarrollo de un robot móvil cuoyo 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, eléctronica y control. El proyecto puede servir como guía para nuevos proyectos de automatización de vehículos móviles.

# Abstract

In our school there are a considerable number of documents, many teachers and researchers. Our students also contribute to this production through its work in order of degree, master's theses. The aim of this material is easier to edit these documents at the same time promote our corporate image, providing visibility and recognition of our Center.

... -translation by google-

# Índice

# ÍNDICE DE TABLAS

# ÍNDICE DE FIGURAS

# Notación

| | |
|---|---|
| $A^*$ | Conjugado |
| c.t.p. | En casi todos los puntos |
| c.q.d. | Como queríamos demostrar |
| ■ | Como queríamos demostrar |
| e.o.c. | En cualquier otro caso |
| e | número e |
| IRe | Parte real |
| IIm | Parte imaginaria |
| sen | Función seno |
| tg | Función tangente |
| arctg | Función arco tangente |
| sen | Función seno |
| $\sin^x y$ | Función seno de $x$ elevado a $y$ |
| $\cos^x y$ | Función coseno de $x$ elevado a $y$ |
| Sa | Función sampling |
| sgn | Función signo |
| rect | Función rectángulo |
| Sinc | Función sinc |
| $\partial$y $\partial$x | Derivada parcial de $y$ respecto |
| $x_\circ$ | Notación de grado, $x$ grados. |
| Pr($A$) | Probabilidad del suceso $A$ |
| SNR | Signal-to-noise ratio |

| | |
|---|---|
| MSE | Minimum square error |
| : | Tal que |
| < | Menor o igual |
| > | Mayor o igual |
| \ | Backslash |
| ⇔ | Si y sólo si |

# 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 inviertiendo 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.

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.

Todo esto será detallado por separado más adelante.

# 2 DESCRIPCIÓN GENERAL

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.

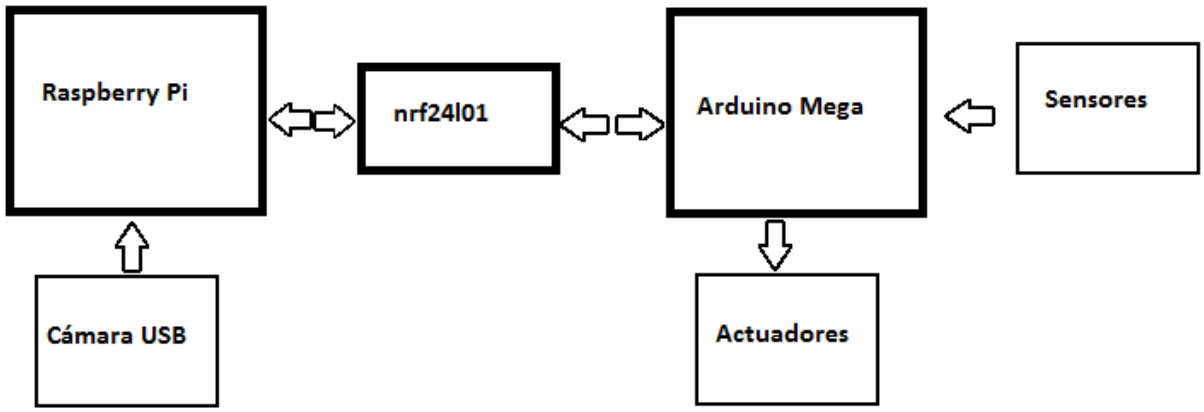
La plataforma usada consiste en un kit de robótica que contiene varios componentes tanto electrónicos como mecánicos integrados y coordinados mediante un módulo Arduino Mega. Gracias a este módulo conseguimos gobernar diversas funciones entre las que cabe destacar: Movimiento, reconocimiento de señales 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 Diagrama de bloques



### 2.1.2 Automatización robot móvil

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

Como comenté en la introducción la automatización de robots móviles esta a la orden del día.

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

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

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

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

Este mapa nos sirve de utilidad para prevenir choques ya que podemos al hacer mapas en tiempo real, ver como varía la posición de un objeto en el tiempo, con lo que podemos 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 usaremos unos sensores ultrasonidos. Para controlar la distancia recorrida he usado un encoder digital. También he colocado unos diodos Leds para ver que señal ha reconocido y controlar el tiempo del STOP.

Al usar componentes de muy bajo coste, me he encontrado ante diversos problemas que detallaré también en el siguiente capítulo. Como resumen podemos destacar los problemas de odometría, ya que la resolución de los encoder no es muy grande, y sobretodo problemas de alimentación de los motores, he usado pilas recargables y al poco tiempo de uso, el funcionamiento no era el adecuado.

### 2.1.3 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 podemos 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:

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

Estos pasos serán detallados en su capítulo.

### 2.1.4 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, necesitamos pasar la información procesada de la cámara en la Raspberry Pi al Arduino. Esto lo he 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 podría haber usado los dos sin antena externa, solo fue para mejorar el comportamiento.

He hecho uso de una librería libre que he encontrado en Github llamada RF24, creada y compartida por tmrh20. En referencias pondré un enlace a dicha página. 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 necesito saber que señal ha sido la reconocida por la Raspberry, cosa que hago con un número y después mandar el OK desde Arduino a la Raspberry. Igualmente, más adelante comentaré con detalle el funcionamiento del programa y las conclusiones.

## 2.2. Conclusiones

FALTAA

## 2.3. Bibliografía (POR SI TENGO QUE AMPLIAR)

Debe contener las referencias bibliográficas de los documentos consultados para demostrar las bases del trabajo realizado y avalar los datos incorporados y citados en el texto.

Se elaborará de forma normalizada, para lo que se aconseja utilizar la norma UNE vigente (actualmente la "UNE ISO 690:2013. Información y documentación. Directrices para la redacción de referencias bibliográficas y de citas de recursos de información").

Para la elaboración de esta parte del Trabajo se recomienda consultar la Web de la Biblioteca de Ingeniería que contiene recursos, guías y ayudas para la elaboración de las referencias bibliográficas.

# 3 AUTOMATIZACIÓN ROBOT MÓVIL

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

*- Isaac Asimov -*

Los capítulos nuevos pueden comenzarse de una forma cómoda copiando y pegando esta página (desde el principio hasta el final) en una página en blanco. Hay cuatro "profundidades" de Títulos que representan los capítulos, las secciones, las subsecciones y los apartados. Para asegurarse de que el capítulo comienza en página impar puede introducir un salto de sección impar.

## 3.1 Esquema general

## 3.2 Componentes y montaje

## 3.3 Implementación de módulos

## 3.4 Obtención del controlador

## 3.5 Experimentos

## 3.6 Conclusiones

## 3.7 Esquema general

# 4 PROCESAMIENTO DE IMÁGENES

*The fundamental problem of communication is that of reproducing at one point either exactly or approximately a message selected at another point.*

Claude Shannon, 1948

En este capítulo voy a describir las partes en las que se divide el procesamiento y tratamiento de imágenes en mi proyecto. Primero mostraré un esquema general en el cual mostraré 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 podemos usar distintos sistemas operativos cada uno con su funcionalidad. Para este proyecto, he 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

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

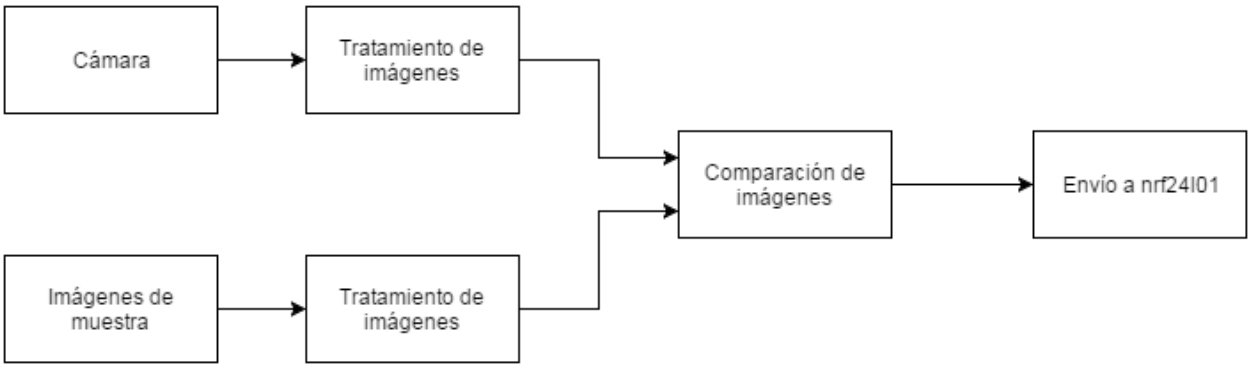
OpenCV lleva siendo más de 10 años una librería muy usada para la visión artificial por ser muy exportable, (funciona en los principales sistemas operativos) usa distintos lenguajes de programación, como puede ser C, C++ o Python.

En mi caso como he comentado anteriormente voy a usar la librería 3.1.0, la cual se puede descargar desde su repositorio oficial

Para el código, utilizaré 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 podemos ver en la imagen siguiente, tenemos como entrada las imágenes que tomamos a partir de la cámara por un lado, y por otro las imágenes de muestra, que son las seáles que queremos detectar. A cada una se le realiza un tratamiento y después se comparan ambas imágenes procesadas. Una vez realizado esto, enviamos el resultado de la señal correspondiente al módulo nrf24l01 tal y como explicaré en el siguiente apartado.




Esquema general del tratamiento de imágenes.

## 4.2 Tratamiento de imágenes tomadas por la cámara

En este apartado, voy 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 haremos será lo siguiente:

- Almacenar una imagen tomada por la cámara.
- Conversión a blanco y negro.
- Hacerle un desenfoque.
- Detectar contornos.
- Encontrar polígono de 4 lados.
- Recortar el rectángulo.

Con esto tendríamos la imagen lista para ser comparada.

A continuación, explicaré en detalle cada paso organizándolo en subapartados.

### 4.2.1 Conversión de color ((((((((((((((((((FALTA IMAGEN)))))))))))) PROBAR con blanco y negro en vez de gris

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

Una vez conseguimos la imagen, tenemos que hacer una conversión de color. En este caso pasaremos de la escala RGB que es en la que capturamos la imagen, a una escala de grises.

Escogemos esta escala, porque lo que nos interesa es encontrar contornos y como vamos a utilizar señales dentro de un fondo rectangular blanco, si la coloco 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 tomada por mi sería la siguiente:

Para convertir de RGB a escala de grises en opencv se hace uso de la función cv2.cvtColor(), por ejemplo, en mi 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 queramos, en mi 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.

### 4.2.2 Desenfoque (IMAGEN CON DESENFOQUE Y SIN ELLA, con grises y con color

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.

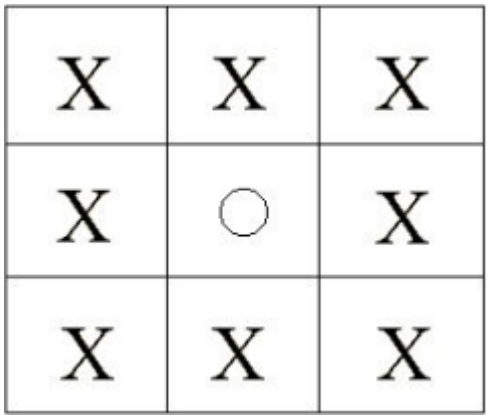
Una vez tenemos la imagen convertida a escala de grises, antes de comenzar con la detección y búsqueda de contornos, necesitamos hacer un desenfoque para mejorar nuestros resultados.

Para lo que se utiliza el desenfoque principalmente es para disminuir el ruido de la imagen y nos 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.

En OpenCV, la manera más cómoda de realizar este desenfoque es con la función cv2.GaussianBlur(). A la cual tenemos 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 estamos trabajando, tenemos que elegir siempre vecindad impar, por ejemplo, como vemos en la imagen inferior, estamos trabajando con una vecindad (3,3).



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

### 4.2.3 Detección de contornos (CANNY CON BLUR Y SIN BLUR

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

OpenCV tiene una función que hace esto, pero primero necesitamos 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 he usado una función para que, en cada imagen, se calculen. Esta función recibe como parámetros: la imagen a la cual queremos hacerle el tratamiento de convertirla en una imagen en la que solo se vean los contornos y un parámetro que nos permite ajustar la anchura del umbral. Un valor grande de este valor, nos deja un umbral amplio. Por otro lado, un valor pequeño, nos limita la anchura del umbral cosa que nos viene muy bien ya que, en nuestro 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 nos saca como salida la imagen con los contornos.

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

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

Ahora tengo que ver si alguno de los contornos se asemeja a un rectángulo de cuatro lados, para ello tengo 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 queremos calcularle el perímetro y como segundo parámetro, True.

2- Aproximar el contorno a un polígono. Usamos la función cv2.approxPolyDP() a la cual hay que pasarle 3 parámetros. El primero de ellos es el contorno al cual queremos 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 hemos calculado el perímetro.

3- Una vez tengamos la aproximación ya podemos ver si el contorno tiene 4 lados calculando la longitud de la aproximación con la función len().

4- Calculamos el área del polígono con cv2.contourArea() y descartamos las que sean pequeñas.

5- Hacemos un rectángulo alrededor del polígono y recortamos ese rectángulo dándole las dimensiones de las imágenes de muestra.

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

### 4.2.4 Pseudocódigo

FALTA

## 4.3 Tratamiento de imágenes de muestra

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

En resumen, hay que hacer:

- Leer y almacenar la imagen.
- Convertir de RGB a escala de grises.
- Desenfocar para suavizar bordes.
- Tener imagen con bordes.

### 4.3.1 Conversión de color PONER SEÑALES CONVERTIDAS

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

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

### 4.3.2 Desenfoque ( PONER SEÑALES DESENFOCADAS)

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

### 4.3.3 Detección de contornos SEÑALES CON CANNY

Para la detección de contornos he usado la misma función que expliqué 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.

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

## 4.4 Comparación de imágenes

Existen distintas técnicas de comparación de imágenes.

1- Error Cuadrático Medio

2- SSM

### 4.4.1 Error Cuadrático Medio (mse)(PONER EJEMPLO)

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 mi caso lo usaré para ver las diferencias entre la imagen tomada por la cámara y la imagen de muestra.

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

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.

### 4.4.2 Indice de similitud estructural (ssim) (ejemplo)

Este índice mide la similitud entre dos imágenes.

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

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

Siendo

- $\mu_j$ la media de los valores en el eje x o eje y.
- $\sigma_j$ La varianza de x o de y
- $\sigma_{ij}$ La covarianza de x e y.
- $c_i$ Un valor para estabilizar.

Este método lo utilizo a partir de una librería llamada skimage creada por scikit-image. Con lo cual solo tengo que llamar a la función *ssim*() y pasarle como parámetros las dos imágenes que queremos comparar.

# EXPERIMENTOS DE LOS DOS

Para este proyecto, he preferido usar el ssim porque como podemos ver es el que mejores resultados nos aporta.

## 4.5 Conclusiones

# 5 ENVÍO Y RECEPCIÓN DE INFORMACIÓN

*The fundamental problem of communication is that of reproducing at one point either exactly or approximately a message selected at another point.*

Claude Shannon, 1948

## 5.1 Esquema general

## 5.2 Conexión inalámbria

### 5.2.1 Introducción

### 5.2.2 Módulo NRF24l01

## 5.3 Implementación en Raspberry Pi

### 5.3.1 Montaje

### 5.3.2 Instalación de la librería

### 5.3.3 Código

## 5.4 Implementación en Arduino

### 5.4.1 Montaje

### 5.4.2 Instalación de la librería

### 5.4.3 Código

## 5.5 Conclusiones

# REFERENCIAS

[1] Autor, «Este es el ejemplo de una cita,» *Tesis Doctoral,* vol. 2, nº 13, 2012.

[2] O. Autor, «Otra cita distinta,» *revista,* p. 12, 2001.

# ÍNDICE DE CONCEPTOS

# GLOSARIO